VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SEGUIMIENTO VIRTUAL EN TIEMPO REAL DE MANIOBRAS DE ESTABILIZACIÓN DE UN SIMULADOR DE VUELO SATELITAL

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECATRÓNICO

P R E S E N T A :

MANUEL ALEJANDRO CASTILLA GALLARDO

DIRECTOR:

DR. ESAÚ VICENTE VIVAS

México D.F. 2012

Contenido

5. **Módulo de Procesamiento**
   5.1. ¿Qué es MatLab?
   5.2. ¿Qué es SIMULINK?
   5.3.Descripción del modelo implementado en SIMULINK

6. **Módulo de Visualización**
   6.1. ¿Qué es Solid Edge?
   6.2. ¿Qué es un editor de VRML?
      6.2.1. Proceso de edición del modelo
      6.2.2. Bloque VR Sink de SIMULINK
   6.3.Interfaz Gráfica de Usuario

7. **Pruebas experimentales**
   7.1. Seguimiento virtual en un eje
      7.1.1. Experimento
      7.1.2. Funcionamiento del sistema implantado en SIMULINK
   7.2. Seguimiento virtual en tres ejes
      7.2.1. Experimento
      7.2.2. Funcionamiento del sistema implementado en SIMULINK
   7.3.Control de posición en un eje
      7.3.1. Implementación del modelo matemático en SIMULINK

8. **Conclusiones**
   8.1. Resultados globales
   8.2. Conclusiones
   8.3.Recomendaciones

Introducción

Desde un punto de vista conceptual la realidad virtual es un software interfaz de alto nivel para computadoras, que integra dos principales elementos: simulaciones en tiempo real e interacciones con múltiples canales sensoriales. Desde una perspectiva funcional, la realidad virtual es una simulación en la cual, modelos bidimensionales y tridimensionales generados por computadora son usados para recrear un mundo de apariencia realista. Integrada en cualquier aplicación, puede ser vista como un importante elemento que, integrado como una herramienta de visualización y análisis, permite vincular la realidad física con una realidad artificial. Sin embargo, la realidad virtual va aún más allá, el mundo sintético que se construye en su entorno no es estático, ya que puede responder a estímulos de diferente origen (internos o externos) que pueden ser proporcionadospor un usuario o por otros medios. Esta característica define otra característica importante de la realidad virtual: interactividad en tiempo real. En este contexto, tiempo real se refiere a que la computadora es capaz de detectar los estímulos de entrada internos o externos y modificar las características del mundo virtual de forma inmediata.

Basados en las características que presenta el desarrollo con elementos de realidad virtual, se presenta en esta tesis el desarrollo de una herramienta que integra un modelo en realidad virtual para fines de monitoreo y análisis cinemático de un simulador de vuelo satelital, utilizado para la verificación de maniobras de orientación.

Particularmente en el desarrollo de estrategias para la orientación de satélites, el contar con una herramienta que permita, por un lado acelerar el proceso de integración de los diferentes elementos que componen al subsistema de control de orientación y por otro lado, verificar anticipadamente sobre un modelo virtual las maniobras de orientación del satélite, se convierte en una herramienta de alto valor agregado, no sólo para la comprobación de algoritmos de orientación del simulador satelital, sino también para el monitoreo y visualización virtuales en tiempo real de su posición, a partir de los datos obtenidos de los sensores de navegación.

Como objetivo principal, se plantea el diseño e integración de una herramienta de apoyo para la generación de tecnología aplicada en sistemas aeroespaciales, que sea particularmente útil en sus fases intermedias de desarrollo experimental con la finalidad de acelerarlos y obtener resultados de una manera rápida y confiable. Esta herramienta será controlada por medio de una interfaz gráfica de alto nivel de

abstracción, la cual facilitará la interacción entre el usuario y el sistema y permitirá una mejor comprensión del mismo, y en la cual se describirá detalladamente de forma esquemática, cada uno de los componentes que integran al sistema de seguimiento virtual. El esquema estará integrado por bloques propios del ambiente de desarrollo, pero también por bloques personalizados que contendrán programas propios para la ejecución de tareas específicas.

Sin duda, la experiencia obtenida luego de haber trabajado en el desarrollo de sistemas, en primera aproximación utilizando ambientes y modelos virtuales, se convierte es un vehículo fiable con el cual es posible determinar sus características (fortalezas y debilidades), y así mejorarlas o, como en el caso de esta tesis, crearlas. La meta que persigue el desarrollo asociado a este trabajo, busca el desarrollo de un sistema que permita analizar el desempeño de un cuerpo real dentro de un ambiente con características simuladas, con el objetivo que prever su comportamiento una vez que es sometido a las mismas características, pero esta vez reales.

La idea de efectuar un seguimiento virtual radica en la necesidad surgida al interior del grupo de desarrollo de sistemas aeroespaciales (GDSA) del Instituto de Ingeniería, UNAM (II-UNAM), de contar con una herramienta de desarrollo alternativa, que permita la aceleración en etapas de diseño y la validación de elementos de subsistemas satelitales. Esta herramienta será de gran importancia para la validación virtual de estrategias de control de orientación triaxial para satélites pequeños y que podrá ser validada también de forma práctica utilizando la plataforma educativa satelital SATEDU. SATEDU es actualmente, uno de los principales proyectos en el que están convergiendo todos los esfuerzos del grupo, en lo que se refiere a la evolución y actualización de los subsistemas que lo integran. SATEDU es un satélite educativo desarrollado totalmente en México por el GDSA del II-UNAM, terminado en su primera versión en 2004. Es una plataforma tecnológica para uso en aulas y laboratorios que integra todos los subsistemas que contiene un satélite real, con el principal objetivo de ser utilizado como una herramienta didáctica en tierra para la formación de recursos humanos en el área de tecnología satelital.

No obstante todo el trabajo hasta ahora ha realizado el GDSA en el campo satelital, actualmente se trabaja en la implementación práctica de algoritmos de control de orientación en tres ejes, los cuales ya han sido validados exitosamente por medio de simulaciones numéricas, pero se busca su transferencia a hardware reconfigurable, utilizando sistemas embebidos basados en dispositivos FPGA y utilizando ruedas inerciales de diseño propio como actuadores activos. Ya se tienen los primeros resultados exitosos de la transferencia a hardware, para el control de posición en un eje de rotación de un simulador de vuelo satelital basado en un colchón de aire y ahora se trabaja en la implementación en tres ejes. Por ello, resulta de gran importancia validar previamente dichos algoritmos en plataformas paralelas que arrojen

resultados de manera rápida para así proporcionar información acerca de la dificultad de su implementación.

El presente trabajo está estructurado de manera secuencial, describiendo en cada uno de sus capítulos desde el planteamiento de la idea para efectuar el seguimiento virtual, el desarrollo conceptual hasta llegar al desarrollo de un prototipo útil para aplicaciones de seguimiento validado en un simulador de vuelo satelital.

El escrito se encuentra dividido en 8 capítulos. El capítulo uno aborda las generalidades de la realidad virtual, sus orígenes y aplicaciones en la vida e ingeniería cotidianas.

En el capítulo dos se abordan los antecedentes teóricos que conceptualizan al cuerpo rígido en la mecánica clásica, cómo es que se representa la orientación de un cuerpo rígido en el espacio y cuál o cuáles son las mejores formas para hacerlo.

En el capítulo tres se describe al sistema de visualización en general, es decir, como es que se conceptualiza el sistema implementado en esta tesis, descomponiéndolo en sus partes esenciales (módulos) para así tratar de conceptualizarlo no como un todo, sino como un conjunto de sistemas que conforman a uno mayor.

En el capítulo cuatro se aborda el primer módulo que integra al sistema, el *Módulo de Adquisición de Datos*, qué partes lo conforman y una descripción de su funcionamiento. En particular, este módulo es el de mayor relevancia dado que su implementación se basa en un FPGA, un sistema electrónico altamente eficiente y ya validado para aplicaciones satelitales reales dentro del laboratorio. Se describen también otros elementos como sistemas embebidos en FPGA, núcleos de procesamiento empotrables como Microblaze y de IP cores que son, básicamente, el corazón de este módulo.

El capítulo cinco aborda todo lo referente al segundo módulo que integra al sistema, el *Módulo de Procesamiento de Datos*. Se describen los módulos desarrollados tanto en el FPGA como en la PC, utilizando el ambiente de desarrollo Simulink.

El capítulo seis describe al último módulo que conforma al sistema completo, el *Módulo de Visualización.* En el se describen las técnicas empleadas que para la utilización de ambientes de realidad virtual, que herramientas adicionales son requeridas para su uso, y además se detalla el por que de la utilización de la plataforma de SIMULINK para el desarrollo de este trabajo.

En el capítulo siete se da el desarrollo de las pruebas experimentales pertinentes que justifican al proyecto, que problemas se presentaron y como fueron resueltos. En general se describen las pruebas

efectuadas dentro del laboratorio que llevaron a la prueba a resultar exitosa.

Dentro del capítulo ocho se reportan las conclusiones obtenidas acerca del proyecto, además de las ventajas del uso de esta herramienta y el trabajo futuro a partir de esta herramienta. Como parte final del presente trabajo, se enlistan las fuentes de las cuales se tomó referencia, así como también de las cuales se obtuvo información relevante en cuanto al estado del arte del software empleado.

Capítulo 1

GENERALIDADES SOBRE MODELADO, SIMULACIÓN Y REALIDAD VIRTUAL

Las herramientas desarrolladas utilizando modelos de realidad virtual, son poderosos medios por medio de los cuales es posible realizar un análisis detallado del comportamiento de un sistema. Debido al avance de la electrónica, particularmente en lo que respecta a plataformas de procesamiento e interfaces digitales, es posible vincular modelos embebidos en ambientes de realidad virtual con estímulos externos reales, lo cual permite aún una mejor apreciación del comportamiento real de un modelo virtual en tiempo real, aún en un ambiente de cosimulación.

En este capítulo se presentan los conceptos generales que permitirán entender los elementos y la lógica completa del esquema de cosimulación planteado, los modelos de realidad virtual utilizados para la implementación del sistema que se describe en esta tesis y su vinculación con señales externas reales para la pre-visualización de las maniobras de orientación de la plataforma de simulación de vuelo satelital del Instituto de Ingeniería, UNAM.

1.1. Modelado y simulación

El modelado y la simulación, o M&S como es comúnmente referido, está convirtiéndose en uno de los programas académicos de elección para estudiantes en todas las disciplinas. M&S es una disciplina con su propio cuerpo de conocimiento, teoría y metodología de investigación. En el corazón de la disciplina está la noción fundamental de que los modelos son aproximaciones para el mundo real. El M&S consiste en, primeramente, crear un modelo aproximado de un evento. El modelo es entonces seguido por una simulación, la cual permite la observación repetida del modelo. Luego de efectuar simulaciones, un tercer paso es efectuar el análisis. Por análisis se entiende la habilidad de obtener conclusiones, verificar, validar una investigación y hacer recomendaciones basadas en varias iteraciones o simulaciones del modelo. Estos preceptos básicos son acoplados con visualización, que es la habilidad de representar información como un modo de interactuar con el modelo. Esta tesis precisamente se encuentra basada en está metodología de diseño, todo ello con el fin de visualizar el seguimiento y la orientación de un simulador de vuelo satelital. Primeramente se desarrolla un modelo del objeto, posteriormente se efectúa

la simulación correspondiente de su posición para luego visualizarlo en un ambiente de realidad virtual y finalmente, establecer los análisis pertinentes sobre su comportamiento [1].

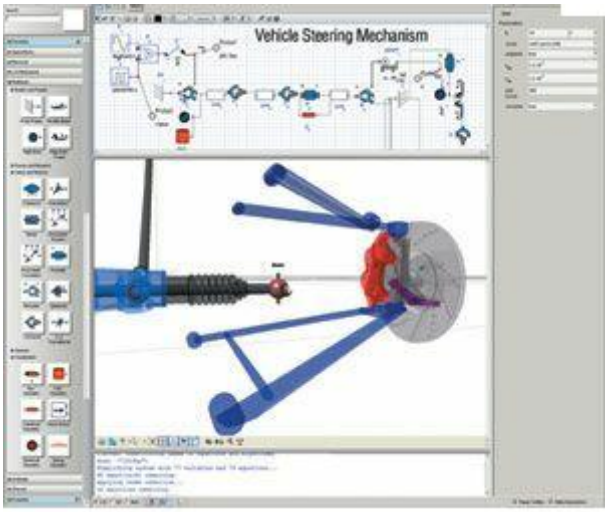


Fig.1.1. Modelado y simulación computarizada de un sistema o proceso real

1.1.2. Modelos: Aproximaciones de eventos del mundo real

Un modelo es la representación de un evento y/o algún objeto que es real. Puede ser algo usado en lugar del objeto real para entender mejor un cierto aspecto acerca de él. Para producir un modelo, este debe abstraerse de la realidad. El modelo puede representar al sistema en algún punto de abstracción o en niveles múltiples de la abstracción con la meta de representar al sistema en una manera matemáticamente confiable. Una vez que el modelo es creado, se puede efectuar una hipótesis bien planteada y creíble, aunque no debe olvidarse que el modelo puede estar sujeto a cambios. La única manera de poder contener estos cambios es a través de múltiples iteraciones o simulaciones del modelo, con la finalidad de encontrar sus singularidades y tratar de corregirlas [1].

1.1.3. Simulaciones

Una simulación es una metodología aplicada que puede describir el comportamiento de un sistema usando un modelo matemático o un modelo simbólico. Simplemente, una simulación es la imitación del proceso o sistema del mundo real en un periodo de tiempo.

El proceso de simulación puede ser descrito como sigue:

a) Desarrollo de un modelo que describa el comportamiento de un proceso o sistema real
b) Desarrollar simulaciones computarizadas basados en el modelo o sistema

c) Ejecutar ese modelo en una computadora digital
d) Analizar los resultados de salida

¿Cuándo debe ser usada una simulación? Una simulación es usada cuando el sistema real no puede ser analizado directamente. Las causas de esto pueden ser que:

1) El sistema pudiera no ser accesible
2) El sistema pudiera ser peligroso de analizar
3) El sistema pudiera simplemente no existir

Así que para contrapesar estas objeciones, una computadora imitará la ejecución de algunos o todos los procesos reales asociados al sistema. El modelado, por otra parte, depende de la ciencia computacional para lograr la visualización y simulación de fenómenos complejos y de gran escala. Estos modelos pueden ser usados para replicar sistemas complejos que exhiben comportamientos caóticos a través de simulaciones que provean observaciones más detalladas del sistema. La simulación también permite la investigación por medio del uso de herramientas de visualización en realidad virtual, donde el analista está inmerso dentro de un mundo simulado a través del uso de dispositivos tales como sensores, monitores y otros diversos componentes de realimentación.

Ahora bien, tanto el modelado como la simulación pueden ser complementados por herramientas que permitan desplegar datos generados durante la ejecución de los procesos asociados a ellos, con el fin de facilitar a los analistas el entendimiento de los resultados obtenidos. Existen diversas formas para representar la información que se obtiene de los procesos de simulación, una de ellas es la que utiliza interfaces gráficas en donde se encuentran espacios de despliegue de datos, botones de selección y otros recursos que fácilmente permiten establecer parámetros y ubicar datos requeridos por el usuario. Como complemento al uso de una interfaz gráfica, en este trabajo de tesis se emplearon herramientas basadas en realidad virtual cuya meta es la de precisamente validar a través de elementos virtuales el seguimiento y orientación de un simulador de vuelo satelital. Está herramienta cuenta con su propio cuerpo de desarrollo, y de la cual existe un importante número de referencias bibliográficas [1].

1.2. Visualización: Orígenes del VRML

El término Realidad Virtual fue concebido a finales de la década de los 80s por Jaron Lanier. Su compañía desarrolló la primera pantalla personal (para uso sobre la cabeza) comercialmente disponible, un dispositivo como el mostrado en la figura 1.2, que permite al usuario sumergirse en un ambiente de realidad virtual, por medio de unos lentes que integraban un proyector y unos audífonos que permitían

integrar audio a las escenas proyectadas, dando mayor realismo a lo observado por medio de este dispositivo. El desarrollo de este dispositivo marcó la pauta para seguir con el diseño de nuevas herramientas virtuales que actualmente resuelven una gran cantidad de problemas de ingeniería con base en simulaciones [2].

Fig.1.2. HMD

El Lenguaje de Modelado de Realidad Virtual, mejor conocido como VRML, fue reconocido en la primavera de 1994 como un lenguaje de descripción de formas durante una reunión convocada por Tim Berners-Lee y DaveRagget para tratar de vincular los desarrollos de la realidad virtual a internet. Un año después, los desarrolladores de VRML lo extendieron para soportar sonido, recreación de ambientes exteriores, imágenes de fondo, animación e interacción con el usuario. De esta forma, en la primera Conferencia Mundial de la *World Wide Web* (WWW) en Ginebra, Suiza, se aprobó el desarrollo de un nuevo lenguaje (fig.1.3) que permitiese crear mundos en tres dimensiones, a los que se pudiera acceder por medio de la WWW [3].

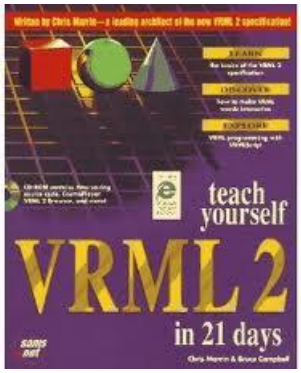


Fig.1.3. Una de muchas referencias bibliográficas sobre VRML.

Incluso, alrededor de esos años se planteó realizar un nuevo medio de entretenimiento basado en VRML montando un performance de la obra de William Shakespeare, llamada “A midsummer Night’s Dream” y

reproducirla en la red. Absolutamente todos los ambientes de la obra serían modelados en VRML. Posteriormente se grabarían las voces de un grupo de actores y los movimientos serían provistos por títeres reales para luego ser digitalizados, comprimidos y enviados a Internet en tiempo real. Sin embargo, todo quedó allí, sin llevarse a cabo tan ambicioso plan. No obstante, luego de un tiempo, el proyecto *VRML Project* fue el primero en hacer historia al estar en la red por más de dos minutos, probando que el envío de voz y movimiento a la red era posible. La figura 1.4 ilustra una de muchas animaciones actuales empleando esta técnica [4].

Fig.1.4. Modelos VRML para el montaje en la red de "A midsummer Night's Dream"

Dado el éxito de tan increíble herramienta de programación se produjeron un mundo de aplicaciones y usos, renovando los contenidos que se pueden encontrar en Internet, permitiendo una mayor atracción de los usuarios a distintas páginas electrónicas que allí se encuentran publicadas. Además, este lenguaje de programación se ha extendido más allá de la *web*, abarcando terreno en medios del entretenimiento como por ejemplo, base de películas completamente llevadas a cabo en mundos virtuales, personajes virtuales, ambientes virtuales, pero con mecánicas muy reales, aproximándose mucho al comportamiento mecánico del mundo real, tal y como se aprecia en la figura 1.5, en la cual se aprecia el modelado virtual de un conjunto de viviendas y ambiente muy parecidos a los reales.

Fig.1.5. Modelo en VRML de un conjunto de viviendas.

1.2.1 Aplicaciones del VRML

Las aplicaciones que con VRML se pueden llevar a cabo son vastas e incontables, aparte de ser muy útiles hoy en día. Por ejemplo, hoy es posible obtener información geográfica (fig.1.6) digitalizada en 3D, con la cual los científicos son capaces de estudiar la topología de ciertas regiones de interés. También es posible el modelado tridimensional de la dinámica del sistema solar, con la ventaja de que el resultado de los algoritmos pudieran ser visualizados utilizando modelos computarizados, dando una perspectiva más amplia y cercana a la realidad a los astrónomos [5].

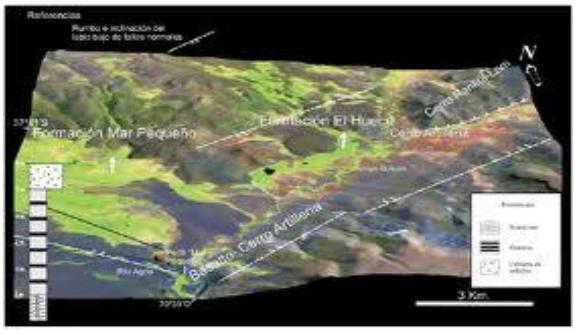


Fig.1.6. Ejemplo de una topografía digitalizada.

Máquinas como automóviles, aviones, robots industriales, inclusive procesos de manufactura no serían posibles hoy si los sistemas CAD no existieran. Estos sistemas son capaces de proporcionar al diseñador imágenes altamente fieles de diseños mecánicos en 2D y 3D, además de contar con sistemas de unidades, dotando al diseño de una mayor fiabilidad y precisión. Los procesos de manufactura, por otro lado, son también previamente diseñados en ambientes de software de tres dimensiones, con el objetivo de prever posibles errores, visualizar espacios y tamaños, así como también el de visualizar virtualmente cómo es que se llevará a cabo el proceso. Con esas herramientas es posible elevar la efectividad y la eficiencia del proceso de diseño, permitiendo la detección temprana de anormalidades, las cuales una vez que han sido detectadas en diseños reales provocarían perdidas tanto económicas como de horas-hombre. Por todo lo anteriormente señalado, los ambientes de realidad virtual han revolucionado el proceso de diseño, y con esto queda claro que cuando se habla de procesos de diseño, en el ámbito que fuere, al final es posible pre-visualizar algún objeto que se pretende llevar a la realidad. La figura 1.7 es muestra de ello, puesto que estos sistemas dotan al diseño de un realismo impresionante y fiel con respecto al objeto o proceso real que se analiza.

Fig.1.7. Sistemas CAD

No sólo en el área industrial o de investigación científica se utiliza el lenguaje VRML. Actualmente existen diversas aplicaciones médicas que se basan en él. Por ejemplo, técnicas basadas en VRML permiten el entrenamiento de médicos cirujanos, estableciendo imágenes digitalizadas correspondientes a partes características del cuerpo humano, figura 1.8. Incluso, el sistema proporciona realimentación al médico al emitir señales de alarma cuando se efectúa un movimiento erróneo que en la realidad pudiera ser perjudicial para una paciente. En general, VRML es una técnica de programación muy potente y prometedora, la cual cada vez está teniendo un mayor uso entre la comunidad científica y tecnológica, incluso, del entretenimiento [6].

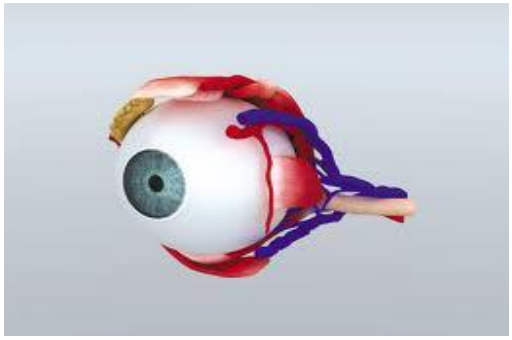


Fig.1.8. Ejemplo de una aplicación médica de VRML.

El modelado y la simulación pueden ser aplicados también en cualquier campo donde la experimentación sea conducida usando modelos dinámicos. En esta vertiente de análisis, se incluyen a todos los tipos de estudios de ingeniería y ciencia así como también ciencias sociales, negocios, medicina y educación. La industria militar fue la primera en beneficiarse del uso de estas herramientas, pasando de simulaciones en el campo de batalla a simulaciones de misiles de defensa.

1.3 Ventajas y desventajas del uso del modelado y la simulación

En 1998 el Instituto de Ingenieros Industriales listó las ventajas y las desventajas de usar modelado y simulación. De la lista se percibe porqué muchos elegirían aplicar M&S a investigación y entrenamiento. He aquí algunas de las ventajas de usar M&S:

a) Habilidad de elegir correctamente probando cada aspecto de un cambio propuesto sin cometer errores adicionales.
b) Comprimir y expandir el tiempo para permitir al usuario acelerar o desacelerar el comportamiento o fenómeno para facilitar la investigación profunda.
c) Entender el "porqué" reconstruyendo el escenario y examinando el escenario cerradamente controlando el sistema.
d) Explorar posibilidades, operar procedimientos, u otros métodos sin alterar al sistema actual.
e) Diagnosticar problemas entendiendo la interacción entre variables que describan sistemas complejos.
f) Identificar las restricciones revisando retrasos en los procesos, información, materiales, examinando sí o no la restricción es la causa o el efecto.
g) Desarrollar el entendimiento mediante la observación de cómo opera un sistema más que sólo realizar predicciones acerca de como operará.
h) Visualizar el plan con el uso de animaciones para observar el desempeño del sistema u organización operando en realidad.
i) Construir consensos para una opinión objetiva acerca de por qué M&S puede evitar inferencias.
j) Prepararse para el cambio respondiendo el "*¿y qué si…?*" en el diseño o modificación del sistema.
k) Invertir sabiamente, ya que un estudio simulado cuesta mucho menos que el costo de un cambio o modificación a un sistema.
l) Especificar requerimientos para el diseño de un sistema que puede ser modificado para alcanzar la meta deseada

Resulta obvio pensar que hay muchos usos y muchas ventajas al usar M&S. El IIE también ha mencionado algunas de las desventajas de usar M&S, la lista es mas corta e incluye puntos tales como: la necesidad de un entrenamiento especial requerido para construir modelos; la dificultad en la interpretación de resultados, cuando la observación pudiera ser el resultado de las interrelaciones de sistemas o probabilidades; costo monetario y en tiempo debido al hecho de que el modelado, la simulación y en análisis pudieran consumir tiempo además de ser caros y el uso inapropiado del modelado y simulación cuando una solución analítica es buena de origen. No obstante, dados los

rápidos avances tecnológicos (principalmente computacionales), también se han mejorado las técnicas de programación, lo que conlleva a un mejor y más rápido aprendizaje de los usuarios, con lo que los tiempos de diseño de modelos se reducen significativamente, además de la enorme cantidad de información disponible en diversas fuentes que proveen información acerca de distintos modelos reales, que bien pudieran ser simplemente utilizados en simulaciones para fines específicos de algún desarrollo que lo requiera.

Como pudo observarse durante este capítulo, una herramienta de simulación que incluye modelos virtuales se convierte en un desarrollo de alto valor agregado, principalmente por la gran oportunidad que ofrece en algunas de las etapas del proceso de desarrollo de un sistema, de validar anticipadamente sobre modelos computacionales, que consideran características físicas reales del sistema, el comportamiento que observaría una vez puesto en marcha. Algo realmente invaluable, no sólo por el impacto visual que ofrece al ver recreado al sistema, funcionando en tiempo real junto al sistema real, sino porque ofrece al diseñador electrónico una oportunidad para ajustar anticipadamente características de desempeño de su sistema físico y que pudieran llegar a representar problemas futuros.

En el Instituto de Ingeniería, esta herramienta se vuelve de un alto interés, particularmente porque permitirá anticipar la validación de modelos de estimación, filtrado y control triaxial aplicados en un simulador de vuelo satelital, en los que actualmente se trabaja en su implementación en un dispositivo de arquitectura reconfigurable FPGA. En el siguiente capítulo se abordan las generalidades teóricas en las que se basa el proceso de simulación y modelado virtual [1].

Capítulo 2

DETERMINACIÓN DE LA ORIENTACIÓN DE UN CUERPO RÍGIDO

En este capítulo se abordan algunos de los conceptos teóricos generales que permitirán entender el concepto de la herramienta computacional que se desarrolló en esta tesis. Se comienza abordando el problema desde un enfoque analítico, considerando el análisis de cuerpo rígido del sistema.

2.1. Concepto de cuerpo rígido

Un cuerpo rígido es aquel cuya forma no varía al ser sometido a la acción de fuerzas externas, es decir, permanece intacto al entrar en contacto con ellas. Con ello, se supone que la distancia entre las diferentes partículas que lo conforman resulta invariante a lo largo del tiempo.

El cuerpo rígido es un modelo ideal y tiene una función muy importante en la mecánica, pues a través de él es como se realizan estudios de cinemática y dinámica. Sin embargo, un cuerpo rígido en realidad sí se deforma, aunque sea de forma mínima y dicha forma se refleja en la magnitud de la fuerza externa que haya actuado sobre él. La figura 2.1 ilustra este concepto en donde se tiene como referencia el marco de referencia inercial [7].

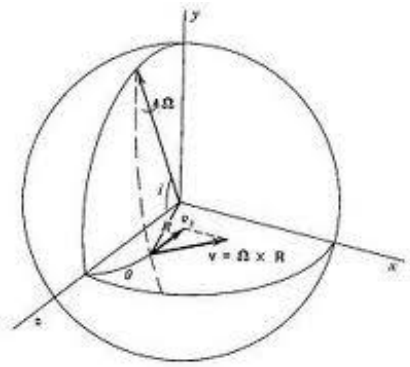


Fig.2.1. Representación conceptual de un cuerpo rígido

2.2. Orientación de un cuerpo rígido

La orientación de un cuerpo rígido puede ser representada matemáticamente por estructuras que trabajan con ángulos de rotación respecto a un eje, así como también vectores de posición relativos al marco de referencia inercial.

Existen estructuras matemáticas para la representación de la orientación, pero existen tres principales: (1) Ángulos de Euler, (2) Matriz de rotación y (3) Cuaterniones. Estos tres tipos de representación serán también discutidos como parte del *background* teórico de esta tesis, dada su relevancia y utilidad que presentan. Algunas estructuras adicionales menos usadas son las conocidas como Cayley-Klein y la representación eje-ángulo, así como también el vector de rotación [8].

2.2.1. Ángulos de Euler

Los ángulos de Euler representan la orientación de un cuerpo respecto a un sistema de referencia *xyz* o con respecto a otro cuerpo rígido. Para describir la orientación en el espacio de un cuerpo en términos de ángulos de Euler es necesario definir tres rotaciones:

- θ (roll) describe la rotación alrededor del eje x.
- φ (pitch) describe la rotación alrededor del eje y.
- ψ (yaw) describe la rotación alrededor del eje z.

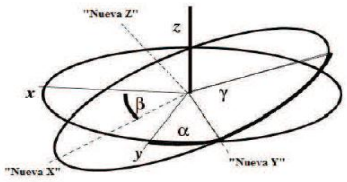


Fig.2.2. Ángulos de Euler

Los ángulos de Euler (fig.2.3) especifican la orientación de un sistema de referencia fijo al cuerpo relativo a un sistema de referencia fijo al espacio, y por tanto actúan como tres coordenadas generalizadas.

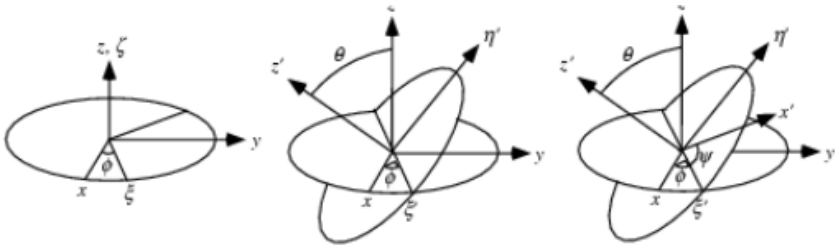


Fig.2.3. Secuencia de los Ángulos de Euler

Es una ventaja tomar como ejes del sistema de referencia del cuerpo a los ejes principales, o sea, los ejes para los cuales el tensor de inercia es diagonal. Adicionalmente, el eje de simetría es uno de los ejes principales y será escogido como el eje z del sistema de coordenadas fijo al cuerpo [8].

2.2.2. Matriz de rotación

Una matriz de rotación es una matriz cuya multiplicación con un vector rota dicho vector mientras preserva su longitud. El grupo ortonormal especial de todas las matrices de 3 x 3 es denotada por *SO(3).*

Así, si R ∈ *SO(3*), entonces:

$$\text{Det } R = \pm 1 \text{ y } R^{-1} = R^T$$

Las matrices de rotación cuyo determinante es igual a *det R = 1* son llamadas **propias**, mientras que aquellas cuyo determinante *det R = -1* son llamadas **impropias**.

Una matriz de rotación puede ser interpretada de una de tres formas:

a) Como la transformación de un vector representado en un sistema coordenado a otro sistema coordenado.
b) Como la rotación de un vector en el mismo sistema coordenado.
c) Como la descripción de una orientación mutua entre 2 sistemas coordenados.

La matriz de rotación [8] puede describirse al usar los ángulos de Euler:

$$R(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sen(\theta) \\ 0 & -sen(\theta) & \cos(\theta) \end{bmatrix}$$

$$R(\phi) = \begin{bmatrix} \cos(\phi) & sen(\phi) & sen(\phi) \\ 0 & 1 & 0 \\ -sen(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

$$R(\psi) = \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -\text{sen}(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Así, la representación de la rotación de un cuerpo rígido en tres dimensiones es:

$$Rot = R(\theta) * R(\phi) * R(\psi)$$

2.2.3. Cuaterniones

Los cuaterniones [8] son una extensión de los números reales, similar a los números complejos. Mientras que los números complejos son una extensión de los números reales por la adición de la unidad imaginaria i, tal que $\boldsymbol{i^2 = -1}$, los cuaterniones son una extensión generalizada y de manera análoga añadiendo las unidades: i, j y k a los números reales, son tales que: $\boldsymbol{i^2 = j^2 = k^2 = -1}$.

Matemáticamente un cuaternión se define como un número de la forma :

$$\boldsymbol{q = a + a_1 i + a_2 j + a_3 k}$$

Tal que a, a_1, a_2, $a_3 \in$ R, donde:

- "a" se denomina parte real o escalar y
- "a_1i + a_2 j + a_3 k" se denomina parte imaginaria o vectorial.

Un cuaternión, dadas las características vectoriales, puede ser unitario, derivando de ello una aplicación interesante de estas estructuras. Dicha aplicación se trata de la representación de rotaciones en tres dimensiones de forma sencilla. Si q es un cuaternión unitario, éste puede pensarse como una esfera de radio 1 en el espacio 4-D. Ahora bien, podemos representar una rotación en el espacio 4- D, en donde 1, 2, y 3 son las componentes de cualquier eje arbitrario y "a" el ángulo de rotación.

Un cuerpo rígido puede ser representado por la rotación alrededor de un eje, como ejemplifica la figura 2.4. Sin embargo, este eje es un vector unitario λ con componentes en un sistema coordenado de referencia *xyz*, por lo que son necesarios cuatro parámetros para describirlo. Estos parámetros son las tres componentes del vector unitario λ y un ángulo de rotación θ [9].

Un cuaternión se define por tener una parte escalar η y una vectorial ε, de manera que:

$$q = [\eta\ \varepsilon 1\ \varepsilon 2\ \varepsilon 3]$$

Un cuaternión unitario se define como:

$$\eta^2 + \varepsilon 1^2 + \varepsilon 2^2 + \varepsilon 3^2 = q^T q = 1$$

Donde:

$$\eta = \cos\frac{\beta}{2}$$

y:

$$\varepsilon = [\varepsilon 1\ \varepsilon 2\ \varepsilon 3] = \lambda \text{sen}\frac{\beta}{2}$$

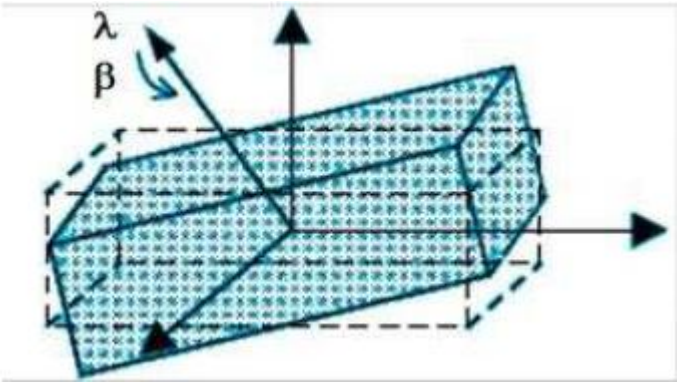


Fig.2.4. Conceptualización del cuaternión.

2.2.4. Comparativa de las tres formas de representación de un cuerpo rígido

Una vez analizadas las formas más comunes de representar a un cuerpo rígido, es posible emplear cualquiera de ellas. No obstante, cada una de ellas presenta ciertas características que pueden servir como base para tomar una determinación en cuanto a la mejor elección. Por un lado, los ángulos de Euler presentan desventajas, tales como las discontinuidades que pudieran presentarse durante el procesamiento. El uso de Cuaterniones y de la matriz de rotación, por otro lado, no presentan dichas discontinuidades. Dado este hecho, sería preferible no usar los ángulos de Euler y sí alguno de las dos restantes [8].

Ahora bien, un cuaternión está representado sólo por cuatro componentes, mientras que la matriz de rotación esta compuesta por nueve. La ventaja entonces se reduce meramente al número de componentes de cada método, lo que conlleva a un procesamiento de datos relativamente más rápido o más lento, según la elección. Habitualmente una de las mejores elecciones es el uso de cuaterniones, aunque en este trabajo se optó por emplear la matriz de rotación como medio de representación de la orientación del cuerpo rígido. Más adelante se explicaran los motivos de dicha elección. La figura 2.5 ilustra como a través de cuaterniones se puede expresar la matriz de rotación, aunque no es lo más usual.

$$R = \begin{pmatrix} i_x i_u & i_x j_v & i_x k_w \\ j_y i_u & j_y j_v & j_y k_w \\ k_z i_u & k_z j_v & k_z k_w \end{pmatrix}$$

Fig.2.5. Cuaternión expresado en matriz de rotación.

A partir de la matriz de rotación es posible obtener un cuaternión y viceversa. Estos algoritmos matemáticos se ofrecen como alternativas de implementación dependiendo de las necesidades del usuario y los requerimientos de la aplicación.

2.2.5. Métodos determinísticos para la estimación de la orientación de un cuerpo rígido

Para la estimación de la actitud es posible considerar principalmente dos tipos de algoritmos: determinísticos y recursivos. Particularmente en el caso de los algoritmos determínisticos sólo toman en cuenta los datos actuales proporcionados por los sensores de navegación.

En un satélite, los diferentes sensores integrados en su plataforma colectan información del medio físico que le rodea. Sin embargo, ellos sólo miden una magnitud física que depende de su actitud. En algunos satélites experimentales, los magnetómetros y el sensor de sol instalados recaban datos de las componentes del campo magnético en el marco de referencia del satélite y la dirección del sol respectivamente. Estos pares de vectores son utilizados por algoritmos de estimación determinísticos para obtener una expresión de la actitud del satélite en términos de una matriz de cosenos directores o de un cuaternión normalizado. Uno de los algoritmos más sencillos para la estimación de la actitud cuando se cuenta con dos pares de vectores de medición, es el algoritmo TRIAD [10].

2.2.5.1. Método TRIAD

El método TRIAD [10] (*Tri-Axis Attitude Determination*) consiste en encontrar una matriz de transformación que relaciona dos sistemas coordenados usando dos vectores de medición y dos vectores de referencia, (todos ellos ortogonales) expresados en el sistema coordenado en el que se efectuó la medición y en el de referencia respectivamente. A pesar de la sencillez del método TRIAD, se ha observado que falla en los casos en los cuales hay un solo sensor de medición disponible y cuando ambos sensores se encuentran coalineados.

El algoritmo TRIAD, parte de encontrar una serie de vectores ortogonales entre sí, "*r1, r2 y r3"* en el sistema coordenado donde se efectuó la medición, tales que:

$$\boldsymbol{r1} = \frac{\boldsymbol{uM}}{\|\boldsymbol{uM}\|}$$

$$\boldsymbol{r2} = \frac{\boldsymbol{r1} \times \boldsymbol{vM}}{\|\boldsymbol{r1} \times \boldsymbol{vM}\|}$$

$$\boldsymbol{r3} = \boldsymbol{r1} \times \boldsymbol{r2}$$

Así como una serie de vectores ortogonales entre sí, "*s1, s2 y s3"* en el sistema coordenado de referencia, tales que:

$$\boldsymbol{s1} = \frac{\boldsymbol{uR}}{\|\boldsymbol{uR}\|}$$

$$\boldsymbol{s2} = \frac{\boldsymbol{r1} \times \boldsymbol{vR}}{\|\boldsymbol{r1} \times \boldsymbol{vR}\|}$$

$$\boldsymbol{s3} = \boldsymbol{s1} \times \boldsymbol{s2}$$

Donde *uR* y *vR* son los vectores de referencia del sistema. Los vectores de referencia son vectores proporcionados por el usuario, es decir, no existe regla o modelo matemático para obtenerlos, simplemente puede ser el vector obtenido en alguna lectura del sensor en alguna posición arbitraria. En el capítulo 5 se abordará el tema con mayor profundidad.

La matriz de transformación que relaciona a ambos sistemas coordenados es la matriz de rotación y se calcula de la siguiente manera:

$$\boldsymbol{A_m^r} = \boldsymbol{r1} * \boldsymbol{s1^T} + \boldsymbol{r2} * \boldsymbol{s2^T} + \boldsymbol{r3} * \boldsymbol{s3^T}$$

Opcionalmente es posible convertir la matriz de rotación obtenida a un cuaternión, el cual es frecuentemente usado para la aplicación de filtros especializados y algoritmos de control dado su reducido número de componentes.

2.2.5.2. Método Gauss-Newton [11]

Este método se basa en un criterio de mínimos cuadrados del error. A diferencia del método TRIAD, se pueden integrar más de dos vectores de medición con sus respectivos vectores de referencia.

$$\boldsymbol{S^k} = \boldsymbol{\sigma^T \sigma} = (\boldsymbol{y_c^i} - \boldsymbol{M y_m^b})^T (\boldsymbol{y_c^i} - \boldsymbol{M y_m^b})^T$$

donde y_c^i contiene los n-vectores de referencia en un sistema coordenado determinado, y_m^b contiene los n vectores de mediciones expresado en el sistema coordenado donde se efectuó la medición. M es definido como:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{R_v^e(q_k)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_v^e(q_k)} \end{bmatrix}$$

donde $R_v^e(q_k)$ es la matriz de rotación que relaciona ambos sistemas coordenados, expresado en términos del cuaternión q_k.

Para encontrar el cuaternión q_k que relaciona los vectores de medición y de referencia, se realiza un proceso iterativo de minimización de la función de error. Este proceso se describe en las siguientes ecuaciones:

$$\boldsymbol{q_{k+1} = q_k - [J^T(q_k)J(q_k)]^{-1}J^T(q_k)\sigma(q_k)}$$

Donde $J(q_k)$ es el Jacobiano de la función M.

El método TRIAD, en comparación con el método Gauss-Newton, presenta una menor complejidad, puesto que está diseñado para trabajar solo con cuatro vectores, determinando la orientación de un cuerpo en el espacio de forma relativamente sencilla. El método Gauss-Newton, por otro lado, puede emplear dos o más vectores de medición y de referencia, convirtiéndolo en un método con mayor precisión, pero con un procesamiento mucho más pesado y complicado.

El capítulo 3 abordará la conceptualización del sistema descrito en esta tesis de forma general, a fin de proveer al lector de un panorama más amplio y sencillo, para posteriormente poder interpretar los temas subsecuentes de una mejor manera.

Capítulo 3

MÓDULOS QUE INTEGRAN A LA HERRAMIENTA DE SEGUIMIENTO VIRTUAL

Un diseño modular, es un diseño basado en la modularización de funciones o tareas que permitan optimizar el tiempo de construcción e integración de un sistema, debido a que le confieren características tales que permiten, por ejemplo, impulsar múltiples funcionalidades y su reutilización al generar un uso diferente para el que fueron creados. Sin embargo, una de las características más importantes es la flexibilidad en el diseño; la modularidad ofrece beneficios como la adición de una nueva solución con tan sólo conectar un nuevo módulo en la estructura del sistema.

En este capítulo se muestra conceptualmente el diseño modularizado del sistema de seguimiento virtual, integrado por bloques funcionales que permiten desde la adquisición de datos de sensores de navegación hasta la visualización de maniobras utilizando un modelo de realidad virtual.

3.1 Introducción

El sistema de seguimiento en realidad virtual, desarrollado en esta tesis, está integrado por una serie de bloques funcionales, tales como: adquisición de datos, procesamiento y visualización en realidad virtual. Estos módulos permiten la apreciación gráfica del seguimiento de maniobras de la plataforma de simulación de vuelo satelital, con que se cuenta en el II-UNAM.

A su vez, el sistema completo está particionado sobre dos plataformas de desarrollo, por un lado una tarjeta comercial de desarrollo Spartan 3E-Starter Kit, cuyo núcleo principal de procesamiento es un dispositivo FPGA, por otro lado, una computadora personal (PC) en donde se ejecuta el software de análisis y simulación matemática MATLAB.

El objetivo de implementar al sistema en dos plataformas tecnológicas (FPGA y PC), obedece a la necesidad, en primer término para resolver la adquisición de datos de la unidad de mediciones inerciales (IMU) que contiene a los sensores de navegación, utilizando un protocolo serial de alta velocidad I^2C, mismo del que carece una PC convencional y por otro lado, la solución del procesamiento de datos con la determinación de la actitud de la plataforma de simulación utilizando un algoritmo estadístico

denominado TRIAD, utilizando como plataforma de desarrollo a una PC convencional donde se ejecuta Matlab. Los algoritmos que describen al procesamiento de datos (determinación de la actitud) se describen en lenguaje gráfico SIMULINK, donde además se integra el modelo en realidad virtual que simula a la plataforma de simulación satelital.

El uso de un FPGA para la implementación de la lógica del microcontrolador, permite aprovechar su potencial como una plataforma altamente flexible, la cual permitirá además para futuras actualizaciones de este sistema, la integración de nuevos núcleos periféricos de protocolos de comunicación, así como la integración de bloques de co-procesamiento para la realización de diversas tareas de pre-procesamiento, filtrado y estimación de datos de los sensores de navegación.

El alcance de esta tesis considera sólo la integración de un sistema base en el FPGA que incluye núcleos de adquisición de datos (Microcontrolador, I2C, FIT, interrupciones, comunicaciones seriales), que se muestra en la figura 3.1. Sin embargo, dentro del grupo de trabajo en el II-UNAM, se trabaja en líneas paralelas de desarrollo que harán uso de esta herramienta de visualización para la verificación *a priori* de maniobras de control de orientación en tres ejes de la plataforma de simulación, donde ya se han desarrollado, integrado y validado exitosamente otros módulos de cómputo como un estimador basado en un filtro de Kalman extendido (EKF) y módulos de control.

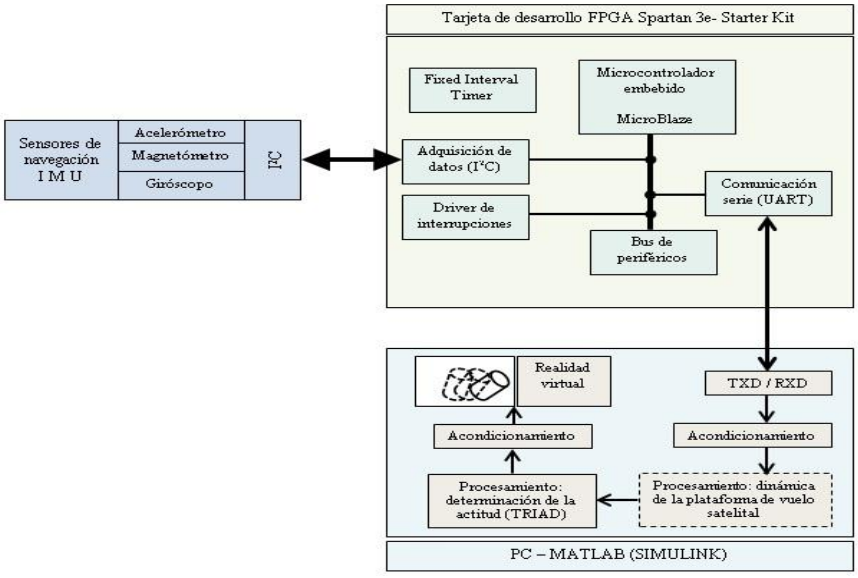


Fig.3.1. Esquema general del sistema de seguimiento en realidad virtual.

3.1 Módulo de adquisición de datos

La adquisición de datos consiste en tomar muestras de una variable del mundo real a través de sensores que reaccionan ante tal variable. Un buen muestreo depende en buena medida de la calidad de los sensores que se pretendan emplear. Se considerará buen sensor a aquel que posea características como la de alta sensibilidad, alta resolución, rapidez de respuesta, precisión y exactitud, principalmente. Solo así se estará adquiriendo información fiel a la real, con el fin de poder hacer algo útil con ella. Aunado a lo anterior, un buen muestreo de señales no sólo depende de la calidad de los sensores, sino también de la calidad de la adecuación de las mismas. La adecuación no es más que la forma en que se maneja la señal una vez adquirida por el sensor. Dicha forma se refiere a cómo se debe tratar la señal, si se debe atenuar o aumentar, para que una vez acondicionada, pueda ser empleada en cualquier procesamiento. El acondicionamiento de una señal se puede hacer por dos formas:

a) Vía Software
b) Vía Hardware

La opción *Software* adecúa la señal con ayuda de lenguajes de programación, por lo regular basados en ambientes de desarrollo que permiten dicha interacción. Es decir, el usuario debe empelar programación para poder acondicionar la señal.

La opción *Hardware* adecúa la señal con ayuda de dispositivos electrónicos basados en circuitos integrados o bien, con ayuda de ambientes de desarrollo que permitan desarrollar *hardware embebido* dentro de dispositivos lógicos tales como FPGA (fig.3.2) o CPLD. En este caso, el hardware embebido se refiere a técnicas de programación que permiten crear hardware virtual que describan las características del hardware físico, con la ventaja de que el hardware virtual obedece a una arquitectura provista por el usuario y no por una ya previamente establecida.

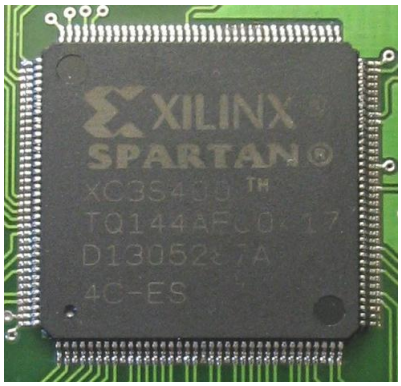


Fig.3.2. FPGA

Particularmente, dentro del desarrollo de este proyecto se optó por emplear hardware embebido dentro de un FPGA. Un FPGA es un dispositivo lógico que contiene una gran cantidad de bloques programables que pueden ser interconectados para así formar sistemas embebidos. Cabe señalar que como parte deldesarrollo de tecnología para pequeños satélites al interior del II-UNAM, ya han sido validados sistemas que integran procesos como la adquisición de datos, el envío y recepción de datos en tiempo real, así como también procesos de actuación con ruedas inerciales dentro de esquemas de control de orientación en un eje.. En el capítulo 4 se abordará más a fondo el tema de sistema embebido y sus particularidades.

3.2. Módulo de procesamiento

El módulo de procesamiento es en donde la información adquirida por los sensores y adecuada por el dispositivo ingresa para efectuar cálculos que determinen la orientación del objeto en tres ejes. Lo interesante de este módulo sucede dentro del bloque que proporciona la orientación en tres ejes. De inicio, la información adquirida pasa directamente a SIMULINK, una plataforma de simulaciones que utiliza lenguaje gráfico incluida dentro de MATLAB (fig.3.3), que permite la utilización de información real y simulada. Los datos ingresan en código ASCII, por lo que primero deben ser convertidos a enteros decimales para poder trabajar con ellos. El FPGA envía los datos en lo que se conoce como trama, la cual posee las coordenadas registradas por los sensores, en este caso, un acelerómetro y un magnetómetro. Cada sensor proporciona tres datos, correspondientes a cada eje que compone al sistema. Una vez formada la trama, esta es ingresada a SIMULINK y es separada en grupos de tres elementos, los cuales conformaran a cada vector de posición y así poder ser integrados en la función correspondiente. Ambos vectores ingresan a un modulo personalizado de SIMULINK, llamado *MatLab Function*, en donde el proceso de determinación de orientación es programado por el usuario. La técnica mediante la cual se obtiene la posición del objeto a través de la fusión de dos vectores se denomina TRIAD (Tri-Axis-Attitude-Determination), que ya fue explicado en el capítulo 1. Por medio de funciones vectoriales es posible obtener una matriz de rotación que proporcionará la orientación del objeto en el espacio. Dicha matriz, por ende, es el producto del procesamiento efectuado por el TRIAD y la cual estará entonces disponible para ser ingresada al siguiente módulo. En el capítulo 5 se abordará el tema con mucha más profundidad.

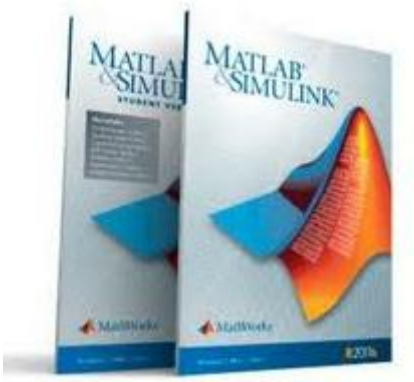


Fig.3.3. Matlab& Simulink

3.3. Módulo de visualización

El módulo de visualización, al igual que el de procesamiento, también se encuentra incluido en SIMULINK, y forma parte de las herramientas propias de dicha plataforma, llamada *Simulink 3D animation*, que entre otras cosas, permite efectuar simulaciones en Realidad Virtual y con información real. A su vez, cuenta con una herramienta llamada *VR Builder*, donde el usuario puede crear y editar ambientes virtuales. En SIMULINK, el bloque *VR Sink* es el bloque que permite visualizar una figura u objeto en realidad virtual. Por si fuera poco, MatLab permite la importación de archivos en formato en VRML (fig.3.4.), lo que significa que los objetos pueden ser diseñados en un sistema CAD, aportando una precisión y un realismo más fieles.

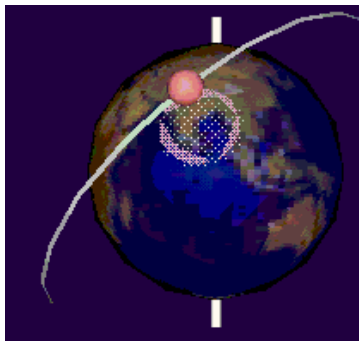


Fig.3.4. Lenguaje de Modelado de Realidad Virtual

Una vez adaptada la figura al bloque de visualización, lo único que resta por hacer es conectar la información que proporciona la matriz de rotación al mundo virtual. No obstante, es una tarea un poco complicada, pues debe existir compatibilidad con el mundo virtual. En el capítulo 6 se abordará de manera más profunda esta temática.

La finalidad de este capítulo fue la de introducir las generalidades del sistema, que partes lo conforman y como está integrado. Los capítulos posteriores abordarán cada uno de los módulos a detalle, definiendo conceptos y características referentes a cada uno de ellos.

Capítulo 4

MÓDULO DE ADQUISICIÓN DE DATOS

El objetivo básico de la adquisición de datos es la integración de los diferentes recursos que lo integran: tales como sensores, transductores, acondicionadores y unidades de procesamiento. Un sistema de adquisición de datos no es más que un conjunto de equipos y/o componentes electrónicos que se encargan de registrar una o varias variables de un proceso cualquiera.

En este capítulo se describirán a fondo los elementos esenciales que conforman al módulo de adquisición de datos que se implementó en una plataforma FPGA, aunque pudo haberse logrado con el uso de un microcontrolador. No obstante, la razón de la utilización de un FPGA radica en el creciente uso de estos dispositivos dentro del desarrollo satelital, además de que ya se han probado y validado algoritmos de comunicaciones basados en estos dispositivos en el grupo de sistemas aeroespaciales del Instituto de Ingeniería. Ahora bien, el módulo está integrado, a su vez, por una serie de núcleos embebidos dentro de un sistema, los cuales realizan diversas funciones, desde la negociación y transferencia de datos con equipos externos utilizando protocolos de comunicaciones, el procesamiento y adecuación de los datos adquiridos así como la transmisión de una trama ordenada para la realización de actividades posteriores del proceso.

4.1. Dispositivos FPGA

Un FPGA es un dispositivo lógico que contiene un arreglo bidimensional de celdas lógicas genéricas o elementos lógico, además de tener interruptores programables, [13] y [14]. El uso de estos dispositivos permite implementar funciones y circuitos digitales sumamente complejos en un solo circuito integrado (IC), [15].

Las celdas lógicas o LC (por sus siglas en inglés) pueden configurarse para desempeñar una función, en tanto que los interruptores pueden programarse para realizar interconexiones entre las celdas lógicas, [13]. A partir de ambas configuraciones, es posible crear en el dispositivo una funcionalidad específica.

Tanto la arquitectura interna de un FPGA como la forma de implementar las celdas lógicas varía entre fabricantes, sin embargo, el concepto de su estructura es el mismo, [16]. Las celdas lógicas usualmente están formadas por un circuito combinacional programable y por un "flip-flop" tipo D (DFF).

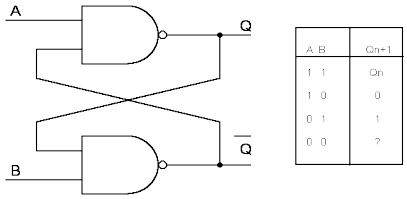


Fig.4.1. representación esquemática de un Flip-flop D y su tabla de verdad

Algunos FPGA también están compuestos por Macro Celdas o Macro Bloques, los cuales están diseñados y fabricados a nivel de transistores, de tal forma que sus funcionalidades complementan las características de las celdas lógicas.

4.2. Programación de un FPGA

La programación de un FPGA consiste básicamente en configurar las celdas lógicas y especificar los puntos de conexión entre los diferentes tipos de bloques lógicos. En los FPGA reprogramables, la información de configuración e interconexión o ruteo se almacena en RAM estática dentro del mismo dispositivo, [15] y [18]. La tarea de programar se resume entonces en crear una cadena de bits (bitstream) que contenga la información de configuración para descargarla al FPGA.

El proceso de programación (o flujo de programación, como lo ilustra la figura 2) comienza típicamente con la descripción del sistema mediante un lenguaje descriptor de hardware (HDL) como Verilog o VHDL (Very High Speed Integrated Circuit Hardware Description Lenguage). Estos lenguajes permiten el modelado de sistemas digitales en diferentes niveles, grados de abstracción y jerarquía, como pueden ser los de estructura y comportamiento, [19]. En el nivel de comportamiento, el más abstracto, se indican las funciones que realiza el sistema, aunque no se indica cómo es que se implementan dichas funciones. El nivel de estructura tiene que ver con la forma en que el sistema está compuesto y la interconexión de los subsistemas, [20]. Cada uno de estos niveles puede dividirse a su vez en distintos grados de abstracción, de ahí la clasificación de jerárquico.

El paso siguiente es la síntesis, en el cual la lógica de alto nivel especificada en el modelo de estructura y el modelo de comportamiento, son convertidos a lógica de bajo nivel, como compuertas lógicas, por ejemplo. Después el proceso de mapeo separa las compuertas en grupos que mejor se adapten a los recursos lógicos del FPGA. Finalmente, se crea un archivo binario que contiene la información para configurar los bloques lógicos y para realizar el ruteo apropiadamente, [18].

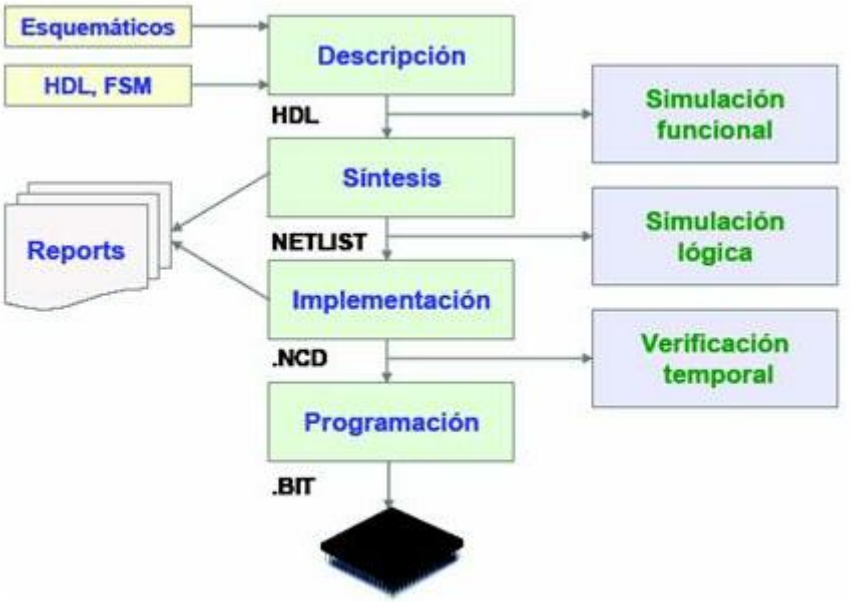


Fig.4.2. Flujo de diseño en FPGA

4.3. Familias de FPGA

En la actualidad existe un importante número de fabricantes de FPGA, así como una amplia variedad de modelos con distintas características y capacidades, los cuales están orientados a diferentes propósitos y aplicaciones.

Un FPGA se puede distinguir de otros dispositivos de su mismo tipo por características como la capacidad (número de celdas lógicas o compuertas lógicas), número de pines de entrada y salida, estructura interna, dispositivos embebidos, memoria, entre otras.

Dichas características pueden servir como criterios de selección al momento decidir el dispositivo FPGA en particular que se utilizará en un proyecto. Además, los recursos tanto del dispositivo como de la plataforma FPGA en general deben ser suficientes para cubrir con los requerimientos del proyecto establecidos previamente como podrían ser: velocidad, capacidad de cómputo, tiempo de procesamiento, consumo de energía, tamaño del empaquetado y costos.

En la tabla 1 se presenta una comparativa de las características básicas de algunas de las familias de dispositivos de dos fabricantes de FPGA.

Fabricante	Xilinx		
Familia	Spartan	Kintex	Virtex
Descripción	Se utilizan principalmente en aplicaciones automotrices. Establecen un balance óptimo entre costo, consumo de energía	Son dispositivos de alto desempeño con un costo mejorado, principalmente dirigido a comunicaciones inalámbricas.	Orientado a sistemas de alto desempeño y gran ancho de banda. Algunos modelos presentan calificación militar y aeroespacial.
Recursos	Cuentan hasta con 150 mil celdas lógicas, 4.8 Mb de memoria, 540 entradas/salidas y hasta 104 bloques multiplicadores de 18x18 bits (69) (70).	Cuentan hasta con 480 mil celdas lógicas, 34 Mb de memoria, 500 entradas/salidas y bloques multiplicadores de 18x18 bits (71).	Cuentan hasta con 2 millones de celdas lógicas, 85 Mb de memoria, 1200 entradas/salidas y bloques multiplicadores de 18x18 bits (71).
Fabricante	Altera		
Familia	Cyclone	Arria	Stratix
Descripción	Son FPGA de bajo costo, bajo consumo de energía e ideales para producciones de altos volúmenes.	Proveen un óptimo balance entre desempeño, consumo de energía y precio para aplicaciones basadas en transceptores.	Son los FPGA de mayor densidad o escala de integración y ancho de banda. Actualmente existen cinco generaciones de esta familia.
Recursos	Cuentan con hasta 300 mil celdas lógicas, bloques de memoria interna de hasta 12Mb, hasta 688 GPIO[1] y hasta 812 multiplicadores de 18x18 bits. Su frecuencia de reloj es de 625MHz (72).	Cuentan con amplia variedad en capacidad de memoria, lógica y funciones de DSP. Cuentan con hasta 495 mil celdas lógicas, memoria de hasta 23.8Mb, 688 GPIO y más de 2000 multiplicadores. Su frecuencia de reloj es de 625MHz (73).	Incluyen transceptores con una tasa de transferencia de hasta 28 Gb. Cuenta con hasta 397 mil módulos lógicos adaptables (equivalente a 597K celdas lógicas), hasta 704 bloques DSP para operaciones aritméticas de precisión variables que incluyen 798 multiplicadores y hasta 2640 bloques de memoria de 20k equivalentes a 52Mb (74).

Tabla 1. Familias de FPGA y sus principales características.

4.4. Plataformas FPGA

Una plataforma FPGA es una serie de recursos y funcionalidades que rodean a un dispositivo FPGA y que permiten el desarrollo de un sistema embebido por completo, [12]. Los fabricantes de FPGA, además de proveer los dispositivos físicos, también brindan a sus clientes plataformas muy completas para crear sistemas embebidos. Dos de los más grandes y populares fabricantes de estos dispositivos son XILINX y ALTERA. A través de sus entornos integrados de desarrollo (IDE), Quartus II en el caso de Altera e ISE para el caso de XILINX, es posible manejar una plataforma FPGA con todos sus recursos y un gran número de herramientas que facilitan el desarrollo de un proyecto.

Un ejemplo de estos recursos son los llamados núcleos o "*cores*". Un "*core*" es un dispositivo prediseñado y reutilizable con una funcionalidad específica que se incorpora dentro de algunos FPGAs, [24]. Su complejidad puede ir desde decodificadores y multiplexores sencillos, hasta ALUs (Unidad Lógica Aritmética), DSPs (Procesador Digital de Señales), microprocesadores, periféricos estándar como puertos de comunicaciones serie y unidades para la solución de algoritmos específicos como FFT (Transformada Rápida de Fourier).

Un núcleo de este tipo está descrito más por su función lógica que por su implementación física. Cuando un bloque es diseñado por una compañía o un desarrollador independiente al proyecto en donde se utiliza, se le llama *IP core* (Intelectual Property), [23]. Los usuarios pueden comprar estos bloques a las compañías que los desarrollan y les permite acortar sus costos y tiempos de desarrollo, [16] y [23]. Estos

dispositivos embebidos son optimizados y permiten a los FPGA que los contienen, convertirse en un una solución muy adecuada al problema, a lo que se conoce como sistema en un chip o SOC (System On a Chip), lo cual permite disminuir el área en las tarjetas electrónicas y su consumo de energía, [16].

Una plataforma FPGA es una serie de recursos y funcionalidades que rodean a un dispositivo FPGA y que permiten el desarrollo de un sistema embebido por completo, [12]. Los fabricantes de FPGA, además de proveer los dispositivos físicos, también brindan a sus clientes plataformas muy completas para crear sistemas embebidos. Un núcleo de este tipo está descrito más por su función lógica que por su implementación física. Cuando un bloque es diseñado por una compañía o un desarrollador independiente al proyecto en donde se utiliza, se le llama *IP core* (Intelectual Property), [23]. Los usuarios pueden comprar estos bloques a las compañías que los desarrollan y les permite acortar sus costos y tiempos de desarrollo, [16] y [23]. Estos dispositivos embebidos son optimizados y permiten a los FPGA que los contienen, convertirse en un una solución muy adecuada al problema, a lo que se conoce como sistema en un chip o SOC (System On a Chip), lo cual permite disminuir el área en las tarjetas electrónicas y su consumo de energía, [16].

4.4.1. Procesadores embebidos: MicroBlaze y Power PC

En la sección anterior se indicó que es posible tener un microprocesador embebido dentro de un FPGA mediante el concepto de *"IP core"*. Un ejemplo de este tipo de dispositivos son los procesadores PowerPC y MicroBlaze, ambos desarrollados por la compañía Xilinx como parte de su plataforma para la creación de sistemas embebidos.

PowerPC es un procesador RISC (Computadora Con Conjunto Reducido de Instrucciones) de 32 bits integrado como un núcleo "*hard core"* a la familia de FPGA Virtex de XILINX. MicroBlaze es también un procesador RISC de 32 bits, pero en este caso se trata de un núcleo "soft core" que forma parte de la plataforma de desarrollo EDK (Embedded Development Kit) de XILINX, [25].

Este tipo de procesadores son capaces interactuar con otros núcleos o "*IP cores"*, generalmente mediante interfaces y buses que provee la misma plataforma de desarrollo. De esta forma, el procesador gana funcionalidades, pues es posible por ejemplo, conectarle un puerto de comunicaciones, una memoria adicional y algún otro "*core"* de propósito específico. Inclusive, en el caso del MicroBlaze, es posible conectarle otro procesador idéntico.

MicroBlaze implementa una arquitectura Harvard, lo que significa que utiliza buses separados para acceso a los datos e instrucciones almacenados en los bloques de memoria RAM. Para conectarse a cada bus LMB (Local Memory Bus), el MicroBlaze requiere de un bloque controlador, el cual también forma parte de la plataforma.

El uso de uno u otro procesador (*hard core o soft core*) dependerá de los requerimientos de la aplicación. Para construir programas ejecutados por estos procesadores es posible utilizar el set de instrucciones del procesador.

El módulo de adquisición de datos está basado en el procesador MicroBlaze debido a que es un sistema versátil de interconexiones que soporta una extensa variedad de aplicaciones embebidas, permitiendo que el usuario logré realizar conexiones con periféricos dedicados.

4.4.2. Enlaces FSL y PLB

Con el fin de establecer conexiones con otros núcleos y periféricos, los procesadores embebidos de Xilinx utilizan típicamente dos clases de enlaces: *FSL* (Fast Simple Link) y PLB (Processor Local Bus).

Los enlaces *FSL* son canales dedicados unidireccionales punto a punto para la transmisión de datos. Tienen un tamaño de 32 bits y se pueden alcanzar tasas de transmisión de hasta 300 Mbps, aunque este tamaño puede ser configurado. Se puede utilizar un bit adicional para indicar si la palabra transmitida (o recibida) es de control o de datos. Este enlace es ideal para comunicaciones entre procesadores MicroBlaze o *streaming2*. Un MicroBlaze tiene puertos suficientes para ocho conexiones FSL de entrada y ocho de salida, [26].

Las comunicaciones a través de un enlace FSL están basadas en un esquema FIFO (First Input First Output). Los dispositivos conectados a través del bus FSL deben ser configurados ya sea como maestros o como esclavos. Un dispositivo maestro sólo puede escribir datos a la FIFO del bus, mientras que un dispositivo esclavo sólo podrá leer.

En la figura 4.3 se muestra el principio de funcionamiento del bus FSL y las señales con las que cuenta.

El “core” que se encuentra al centro de la figura tiene un enlace FSL maestro de lado izquierdo y un enlace FSL esclavo del lado derecho, implementando de esta forma una comunicación bidireccional para el núcleo.

PLB es un enlace primario compartido de alto desempeño que provee la infraestructura necesaria para conectar un número determinado de dispositivos maestros y esclavos. El bus incluye una unidad de control, un temporizador *watchdog*, direcciones de memoria separadas para la escritura y la lectura y un registro de control opcional DCR (Device Control Register), [27].

El bus empleado en este proyecto es el bus PLB, debido a que presenta altas velocidades de transmisión y es ideal para establecer comunicaciones entre distintos procesadores.

4.4.3. Entorno de desarrollo

Los fabricantes suelen proporcionar al cliente su propia plataforma de desarrollo de sistemas embebidos en software, optimizada para los dispositivos que fabrican. En el caso de Xilinx, se cuenta con los

entornos de desarrollo EDK e ISE, los cuales a su vez, están compuestos de distintas herramientas que se mencionan a continuación.

a) EDK

El Kit de Desarrollo Embebido EDK (*Embedded Development Kit*) es un conjunto de herramientas de diseño que permite crear de principio a fin, sistemas basados en un procesador embebido y que además permite implementarlos en un dispositivo FPGA de Xilinx, [28]. Estas herramientas son:

- La interfaz gráfica de usuario XPS (*Xilinx Platform Studio*).
- La suite de herramientas del sistema embebido.
- Núcleos de procesadores embebidos y otros "*IP cores"* como periférico.
- El Kit de Desarrollo de Software SDK (*Software Development Kit*).

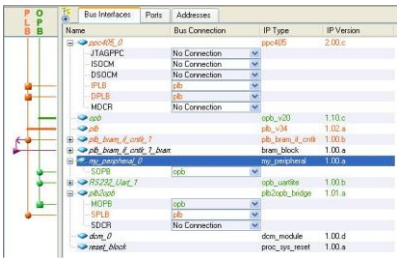


Fig.4.3. Interfaz de EDK

b) ISE

Junto con el EDK, es necesario instalar también el Entorno de Software Integrado ISE (Integrated Software Environment). EDK depende de ISE para realizar la síntesis del diseño en hardware del

microprocesador embebido, mapearlo al FPGA destino, así como generar y descargar el archivo binario al dispositivo, [29]. Algunas de las herramientas con las que cuenta ISE son las siguientes:

- El navegador de proyectos.
- La herramienta de síntesis XST (Xilinx Synthesis Tool).
- Las herramientas de ruteo y posicionamiento de lógica.
- Herramientas para la programación del dispositivo (iMPACT).
- Librerías de componentes lógicos de propósito general.

En nuestro caso, ISE resultó ser una herramienta muy útil para crear y sintetizar núcleos personalizados (*Customized Cores*) y componentes propios, a través de su editor de VHDL y las herramientas XST y Xilinx Core Generator.

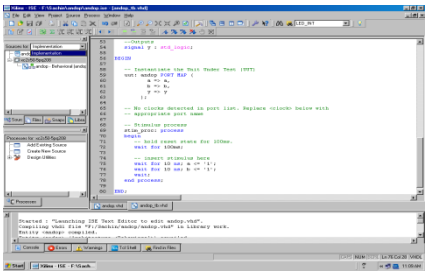


Fig.4.4. Interfaz de ISE

c) XPS

XPS brinda una interfaz gráfica de usuario (GUI) para crear las especificaciones de hardware y software de sistemas basados en los procesadores embebidos MicroBlaze y PowerPC. Provee de un editor y un administrador de proyectos para crear y editar código fuente. También posee un editor gráfico para personalizar las conexiones del procesador, periféricos y buses, [28]. Mediante XPS es posible realizar las siguientes tareas:

- Añadir núcleos, editar sus parámetros y realizar sus conexiones.
- Crear y modificar los archivos MHS (Microprocessor Hardware Specification) y MSS (Microprocessor Software Specification).
- Generar y visualizar un diagrama de bloques del sistema, así como el reporte del diseño.

El archivo MHS define la "plataforma en hardware", la cual consiste básicamente en los componentes que comprenden el sistema embebido (procesador y periféricos), sus características y la forma en que están interconectados (buses y puertos).

El archivo MSS captura la "plataforma en software", que se refiere a la colección de controladores (software) para los distintos elementos y componentes en hardware (*IP cores*, buses, procesadores embebidos), y opcionalmente, al sistema operativo sobre el cual se construirá una aplicación.

Mediante esta herramienta es posible también integrar los núcleos diseñados previamente en ISE al proyecto principal, mediante la interconexión de éstos con el MicroBlaze a través de enlaces tipo FSL. En ocasiones, es necesario modificar los archivos MHS y MSS con el fin de especificar ciertos requerimientos de la aplicación.

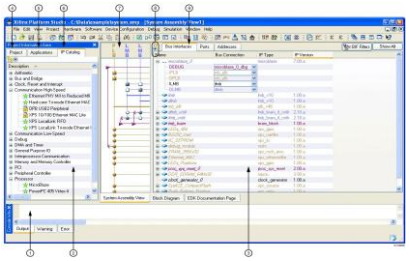


Fig.4.5. Interfaz de XPS

d) SDK

Es una interfaz gráfica de usuario complementaria a XPS, que provee un entorno de desarrollo para aplicaciones de software incluidas en el proyecto, como son los programas que el procesador embebido deberá ejecutar. Algunas de sus características son las siguientes, [30]:

- Un editor de código C/C++, junto con un entorno para corrección de errores, depuración de código y compilación.

- Un administrador de proyectos.

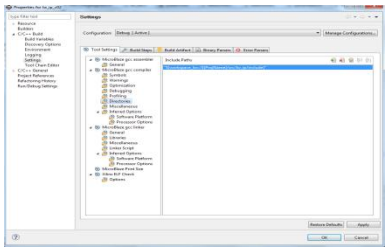


Fig.4.6. Interfaz de SDK

4.5. Diseño y desarrollo de sistemas embebidos sobre plataformas FPGA

Una gran ventaja de los FPGA sobre otros dispositivos, ya mencionada en varias ocasiones, es su capacidad para integrar al diseño núcleos personalizados (*Costumized Cores o IP Cores*) lo que resulta en una dramática aceleración de los procesos y tiempo de ejecución en software, debido a que los algoritmos se ejecutan paralelamente en bloques de hardware y no secuencialmente en código de software, [26].

Por otro lado, los procesadores *"soft core"* (SCP) también presentan ventajas sobre otros procesadores como es la de sólo utilizar las características mínimas requeridas del procesador para una aplicación en específico.

Gracias a esta dualidad de recursos, es decir, a la combinación de la flexibilidad del software con el alto desempeño del hardware, un diseñador que trabaja con FPGA debe pensar diferente de aquellos diseñadores que utilizan otros dispositivos. Los desarrolladores de software típicamente escriben programas secuenciales que explotan la habilidad de los microprocesadores para ejecutar con rapidez una serie de instrucciones. En contraste, un diseño de alta calidad en un FPGA requiere pensar en el "paralelismo espacial", esto es, utilizar simultáneamente los recursos distribuidos a lo largo del chip para realizar una gran cantidad de operaciones de cómputo, [18].

La importancia de este modelo de desarrollo radica en que permite considerar las aplicaciones y algoritmos en nuevas maneras, como la de utilizar técnicas de cómputo en paralelo (fig.4.7) que tal vez

sean poco comunes, con el fin de obtener el máximo desempeño posible.

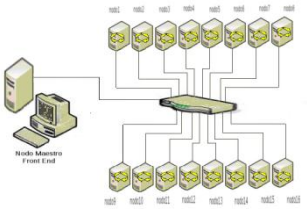


Fig.4.7. Cómputo en paralelo

Una gran mayoría de aplicaciones que requieren de tareas de procesamiento en tiempo real pueden ser reducidas a un conjunto de operaciones matriciales básicas o primitivas, tales como la multiplicación de una matriz por un vector o por otra matriz, la suma de matrices y la inversión de matrices.

Muchos algoritmos han sido desarrollados para efectuar este tipo de operaciones básicas. Éstos varían en sus características y requerimientos de cómputo, como memoria, cantidad y tipo de operadores, así como la cantidad y tipo de paralelismo. Generalmente los algoritmos utilizados se basan en subrutinas del álgebra lineal básica y requieren altas velocidades de cómputo para cumplir con los tiempos que requieren las aplicaciones.

4.6. IMU y sensores inerciales

Las mediciones del INS son llevadas a cabo mediante sensores inerciales: acelerómetros para la medición de la aceleración lineal y giróscopos, ya sea para medir la velocidad angular o bien, el ángulo de rotación mediante giróscopos de desplazamiento. Al ser estas mediciones cantidades vectoriales, los sensores de inercia tienen especificado un eje de entrada que determina la componente del vector que es capaz de medir [31].

La cantidad de diseños de sensores de inercia es muy grande y pueden estar basados en una variedad de principios físicos o tecnologías diferentes.

En aplicaciones que demandan bajos costo y tamaño reducido los sensores basados en sistemas electromecánicos micro maquinados (MEMS) es la opción más viable para implementar un INS, dadas sus características y el desarrollo que estos dispositivos han tenido en los últimos años.

Sin embargo, la reducción en los elementos de sensado trae consigo problemas para alcanzar un buen desempeño de los propios sensores por lo que es necesario aplicar técnicas complementarias con el fin de obtener mediciones más confiables [32].

4.6.1. Errores en los sensores de inercia

La integración de la aceleración medida causa errores en la estimación de la velocidad, que crecen linealmente con el tiempo y en consecuencia, así lo harán los errores de la posición. Estos errores generalmente son desconocidos, excepto por sus propiedades estadísticas [31].

Entre los más comunes tipos de error presentes en un sensor se encuentran:

(a) Sesgo por descentramiento (bias), es una salida del sensor diferente de cero cuando la entrada al sensor es cero, es decir, cuando la salida debería de ser cero.

(b) Error de factor de escala, es generalmente el resultado del uso del sensor (aging) o de las tolerancias de manufactura.

(c) No linealidades, las cuales están presentes en todos los sensores hasta cierto grado.

(d) Asimetría de signo del factor de escala.

(e) Zona muerta, cuando el sensor no presenta una salida con respecto a una entrada.

(f) Error de cuantización, inherente a los sistemas digitales.

4.6.2. Acelerómetros

La mayoría de los acelerómetros detectan la aceleración ya sea mediante el desplazamiento de una masa de prueba o mediante el cambio de frecuencia de un elemento vibratorio, ambos causados por un cambio en la tensión mecánica sobre dicho elemento resultado de la aceleración [31] [32].

La forma en la que se mide el desplazamiento de la masa de prueba involucra generalmente la evaluación de los cambios en la capacitancia que se generan al acercar o alejar estructuras de material semiconductor de los que está compuesto el acelerómetro. La implementación del segundo tipo de acelerómetros implica la utilización de algún tipo de material piezoeléctrico.

En comparación con los acelerómetros piezoeléctricos, los acelerómetros capacitivos presentan un amplio rango de salida, son estables y consumen poca energía, además, muestran una mejor sensibilidad y son menos dependientes a los cambios de temperatura. Su principal desventaja es su susceptibilidad a las fuentes de interferencia electromagnética [34] [35]. Por su parte, los acelerómetros piezoeléctricos presentan una buena linealidad e inmunidad ante la interferencia electromagnética [35].

Fig.4.8. Acelerómetro de tres ejes

4.6.3. Magnetómetros

Pueden ser utilizados para obtener una referencia inicial de baja precisión [31] de la orientación de un

cuerpo a través de la medición del campo magnético de la Tierra. Combinando la magnitud y orientación de dicho campo, medido con respecto al cuerpo del satélite, con un modelo del campo magnético Terrestre y con la ubicación del satélite, es posible determinar la orientación del satélite [36].

Fig.4.9. Magnetómetro de tres ejes

4.6.4. Unidad de Medición

El proveedor de componentes electrónicos Sparkfun desarrolló una tarjeta o unidad de mediciones inercial (fig.10) que integra tres sensores sobre un bus de comunicaciones I2C: Acelerómetro, Magnetómetro y Giróscopo. Esta solución beneficia al proyecto en muchos sentidos: el costo de la tarjeta es menor que el que se tendría si se adquirieran los sensores por separado, se resuelve la problemática del soldado de componentes de montaje superficial de tamaños muy reducidos, se reducen los tiempos de diseño y fabricación así como los costos y tamaño de la tarjeta que albergará a al subsistema denavegación. Así, la medición de los vectores de posición se vuelve una tarea relativamente sencilla y práctica, además de ahorrar espacio y contar con tres sensores de estimación de la orientación de cuerpos rígidos: acelerómetro, magnetómetro y giróscopo.

Fig.4.10. Inertial Measure Unit "IMU"

4.7. Implementación

Dentro de esta sección se describirá de una manera más a fondo el proceso de adquisición de datos en FPGA una vez conocidos los subsistemas que lo integran.

4.8. Adquisición de datos

Como se mencionó en la sección 4.7.4, los sensores acelerómetro, giróscopo y magnetómetro están integrados en una sola tarjeta y conectados a un bus I2C. La lectura de las mediciones puede llevarse a cabo añadiendo al desarrollo un periférico (*IP core*) que maneje este protocolo de comunicación serial en

adición a un componente en software (driver) que gestione el control de las comunicaciones, así como el propio programa encargado de la adquisición de datos que será ejecutado dentro del procesador embebido MicroBlaze.

Para entender mejor este desarrollo, se explicarán brevemente las bases del estándar I2C. Posteriormente se hablará del funcionamiento y utilización de la interfaz del periférico, y finalmente de la estructura del programa de adquisición.

4.9. Transferencia da datos mediante el estándar I2C

IIC (Inter Integrated Circuit) ó I2C es un estándar de comunicaciones serial bi-direccional entre circuitos integrados que utiliza únicamente dos señales: SDA para la transferencia de datos y SCL para la señal de reloj. Cada dispositivo conectado al bus tiene una dirección única de 7 o 10 bits, que opera como receptor y transmisor al mismo tiempo y también como maestro o esclavo [37] [38].

La transferencia de datos se lleva a cabo como sigue: el dispositivo configurado como maestro (en este caso el MicroBlaze) genera la señal de reloj e inicia la transferencia de datos enviando la dirección del dispositivo con quien desea comunicarse más un bit de lectura o escritura (R/W), todo esto a partir de una condición de inicio o START definida por una combinación específica de los niveles de las señales SDA y SCL.

4.10. IP Core y driver para la transferencia de datos

La adquisición de datos es una función más de la computadora de navegación. Al tener ya un procesador embebido MicroBlaze como pieza central de ésta, sólo es necesario agregar dicha funcionalidad mediante elementos de hardware y software extra.

El elemento de hardware agregado es el periférico XPS IIC Interface v.2.00.a que se encuentra dentro del catálogo de *IP cores* del EDK. El periférico se conecta al MicroBlaze mediante el bus PLB y se especifican sus características como la frecuencia de la señal SCL y la utilización de una dirección I2C de 7 bits. Además, en al archivo UCF (User Constraints File), se configura la interfaz eléctrica, que involucra las resistencias de pull-up en los pines de salida (activadas) y los niveles de voltaje (3.3 [V]).

Al agregar el bloque de hardware, el software controlador se añade automáticamente al proyecto, aunque es posible modificar la versión manualmente, alterando el archivo MSS.

El controlador está formado por diversos archivos y librerías escritas en código C que contienen funciones básicas que permiten al usuario enviar datos a través del bus I2C con cierto grado de transparencia. Éstas permiten primordialmente enviar los bits de dirección del dispositivo, los bits de lectura y escritura (R/W) y los datos a transferir. Queda por parte del usuario programar la forma en que estás funciones serán utilizadas, es decir, programar la secuencia o estructura de transferencia de datos especificada en la sección anterior.

4.11. Validación de la lectura de las mediciones de los sensores

Con el fin de validar el funcionamiento de las funciones programadas y de la correcta utilización de los periféricos, se implementó un diseño que permite obtener los datos de los sensores y enviarlos a una PC de forma serial bajo el estándar RS-232, tal y como se ilustra en la figura 4.11.

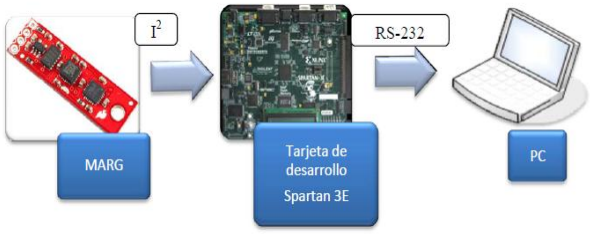


Fig.4.11. Proceso de adquisición

El sistema implementado en el FPGA consta de un procesador embebido MicroBlaze que controla las comunicaciones de la interface I2C con los sensores y las comunicaciones del bloque UART (Transmisor-Receptor Asíncrono Universal) con la PC. Ambos periféricos se conectan con el procesador a través del bus PLB.

Con el fin de establecer un periodo de muestreo de sensores, se agregó también al desarrollo un temporizador y un controlador de interrupciones. El temporizador de periodo fijo FIT (Fixed Interval Timer) genera una señal interrupción cada vez que se alcanza el periodo previamente configurado. Dado que el MicroBlaze sólo es capaz de manejar las interrupciones generadas por una sola fuente, y pensando en que futuros desarrollos podrían requerir de más fuentes de interrupciones, se agregó al presente desarrollo un *core* que permite gestionar diversas fuentes de interrupciones. El esquema del diseño implementado se presenta en la figura 4.12.

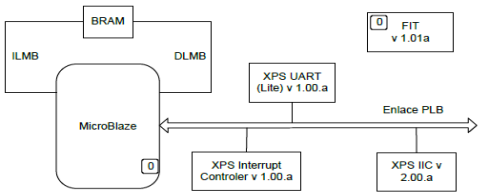


Fig.4.12. Arquitectura de hardware para la adquisición de datos de sensores

Después, se toman las mediciones de cada uno de los sensores para ser finalmente enviadas a la PC. El diagrama presentado a continuación resume el flujo del programa de adquisición de datos.

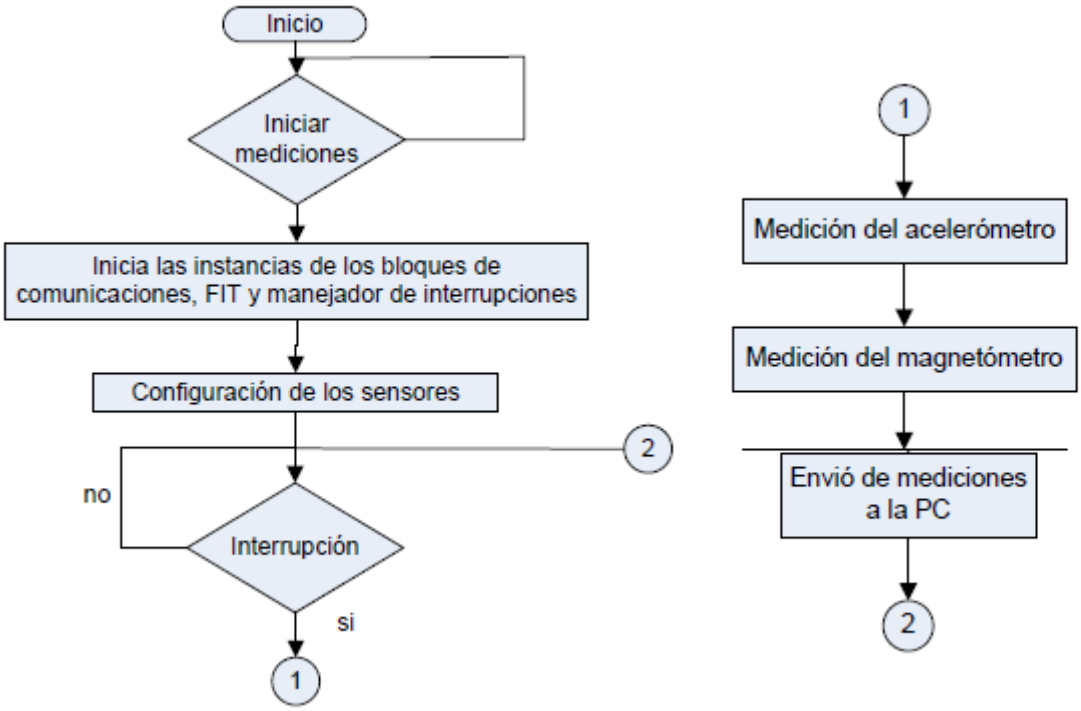


Fig.4.13. Diagrama de flujo del módulo de adquisición de datos

4.12. Transmisión de datos a la PC

A lo largo de este capítulo se ha abordado el esquema general de adquisición de datos y su preprocesamiento dentro de una arquitectura definida basada en FPGA, describiendo en qué consiste, como está constituida, además de que unidades sensoriales fueron empleadas para dicha actividad. Ahora bien, siguiendo la línea que recorre la información una vez preprocesada, el paso siguiente sería inevitablemente su ingreso a la plataforma de SIMULINK, la cual se encuentra alojada en la PC. No obstante, dada la necesidad de realizar un seguimiento virtual versátil y de vanguardia, emulando a los sistemas modernos basados en comunicaciones inalámbricas, fue introducido un módulo de radiofrecuencia, que entre otras cosas, ayudará a validar algoritmos de comunicación remotos, permitiendo analizar y evaluar la respuesta en sistemas satelitales previamente en tierra.

4.12.1. Módulo de radiofrecuencia

Se trata de un radio-módem [39] con un rango de transmisión entre 900Mhz a 2.4Ghz, permitiendo al usuario manejar una amplia gama del espectro de frecuencias de radio disponibles. Posee un alcance de transmisión de 10 km en línea de vista y 300 m en ambiente urbano, con protocolo RS232 y un margen de voltaje de alimentación entre 7 y 18V. Adicionalmente puede modificarse la velocidad de transmisión (baudaje) dependiendo de las aplicaciones para las que serán utilizados. El módulo cuenta con dos radio-módems, donde uno de ellos se instalará sobre el simulador de vuelo satelital y el otro a un costado de la

PC. Ambos se configuran con un baudaje de 9600 bits por segundo.

Fig.4.14. Módulo RF

El siguiente capítulo abordará lo referente al módulo de procesamiento, considerando sus entradas y salidas, y como es que la información preprocesada ingresa a SIMULINK para ser empleada en los procesos de determinación de la orientación de un cuerpo rígido.

Capítulo 5

MÓDULO DE PROCESAMIENTO

El procesamiento realizado sobre una arquitectura computacional es una serie de pasos lógicos que permite efectuar distintas operaciones y cálculos complejos, es un proceso caracterizado por su alta velocidad y precisión. A su vez, se puede interpretar al procesamiento como una caja negra, la cual posee entradas, salidas, así como operaciones y procedimientos que se llevan a cabo al interior de la caja, en un nivel de abstracción tal, que para el usuario es un proceso transparente.

Actualmente, ésta es una cualidad de alta importancia para varios de los procesos y aplicaciones en ingeniería moderna, dado el crecimiento de aplicaciones que demandan la operación con datos en tiempo real. Se ha vuelto también indispensable la utilización de dispositivos lógicos con grandes recursos, los cuales permitan efectuar tareas de procesamiento complicadas en el menor tiempo posible. Adicionalmente existen plataformas en software ejecutándose sobre PCs que permiten la utilización de datos en tiempo real, un ejemplo de ello es MatLab, plataforma en la cual está basado el presente desarrollo de tesis. En este capítulo se abordará con cierto detalle el segmento de procesamiento del sistema de seguimiento virtual para la MSA.

5.1. ¿Qué es MatLab?

MatLab (Matrix Laboratory) es ambiente de programación para el desarrollo de algoritmos, análisis de datos, visualización y computación numérica. MatLab es una herramienta poderosa capaz de solucionar problemas técnicos de cómputo incluso más rápido que los lenguajes de programación tradicionales, como C, C++ y Fortran. Además, MatLab puede ser utilizado en un amplio rango de aplicaciones, incluyendo procesamiento de imágenes y de señales, comunicaciones, diseño de controladores, pruebas y mediciones, modelado financiero y biología computacional.

A pesar de contar con su propio formato de archivos (formato .m), MatLab también posee la característica de poder ejecutar archivos desarrollados en otros lenguajes como C, C++ y Fortran, ejecutarlos dentro de su ambiente de desarrollo, convirtiéndolo en una herramienta sumamente versátil y flexible. No obstante lo anterior, MatLab permite gestionar operaciones en tiempo real, convirtiéndola en una plataforma ideal

para desarrollar aplicaciones basadas en una técnica llamada *Hardware in the loop*, también conocida como *HIL*, en donde datos reales que ingresan al sistema, son procesados y retransmitidos a la fuente de origen.

5.2. Simulink

Simulink es un entorno de programación gráfico que funciona sobre el entorno de programación MatLab. Es un ambiente para simulación multidominio y diseño, basado en modelado para sistemas dinámicos y embebidos. Provee un ambiente interactivo gráfico y un set de librerías de bloques predefinidos (toolboxes) el cual permite diseñar, simular, implementar y probar una variedad de sistemas variantes en el tiempo incluyendo comunicaciones, control, procesamiento de señales y procesamiento de imágenes y video. Simulink es la herramienta idónea para modelar, simular y validar complejos algoritmos y sistemas que resultan difíciles de analizar en el mundo real. Adicionalmente, esta poderosa herramienta cuenta con visualización en 3D, un medio más por el cual es posible visualizar el comportamiento de sistemas físicos utilizando un modelo virtual que puede ser animado.

Simulink cuenta con una serie de librerías capaces de desempeñar casi cualquier tipo de tarea o proceso. Es capaz de desarrollar simulaciones de índole mecánica, electrónica, hidráulica, procesamiento de imágenes, sistemas aeronáuticos, entre otros, posicionándolo como un simulador de amplio espectro de aplicación. La desventaja de usar Simulink es que se tiene que trabajar con bloques predefinidos, lo que a la larga limitará las características del modelo o sistema, debido a las restricciones que imponen, en términos de la configuración de sus parámetros de operación. Sin embargo, es posible crear bloques de simulación propios, aunque puede llegar a resultar en una tarea altamente compleja y no existe mucha documentación al respecto.

5.3. Descripción del modelo implementado en Simulink

Una vez presentadas las generalidades relativas al entorno en software en el cual fue desarrollado el segmento de procesamiento, se describe la implementación del modelo cinemático y dinámico que describe a la MSA.

El modelo en general consta de subsistemas que tratan los datos desde su recepción hasta su visualización en simulink. El procesamiento comienza desde la entrada de datos a través del bloque serial, pasando después por un bloque de conversión de datos, convirtiéndolos a enteros antes de ser decodificados a formatos compatibles con la plataforma. Los datos son entonces pasados al bloque TRIAD en donde se determinará la orientación del cuerpo rígido a partir de la lectura de los sensores, para que finalmente

sean visualizados en ambiente de realidad virtual. La figura 5.1 ilustra la generalidad del concepto por medio de un diagrama de bloques. A continuación se detallara cada uno de estos subsistemas, así como también una breve explicación de su funcionamiento.

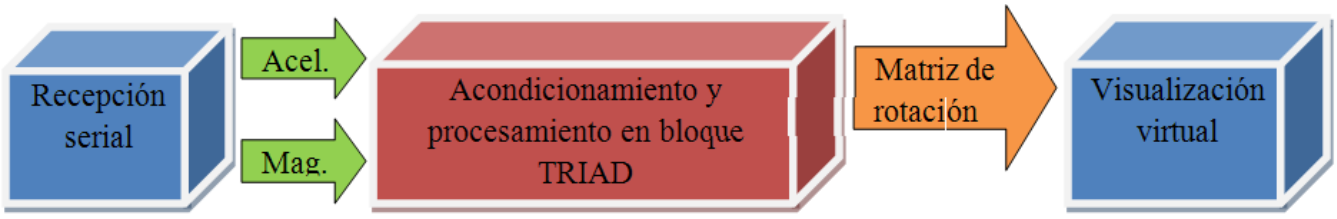
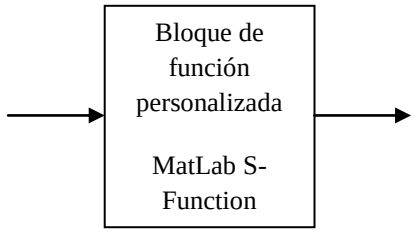


Fig.5.1. Esquema general de la implementación en Simulink

La implementación del modelo cinemático en Simulink resultó ser relativamente sencilla, sin embargo, es necesario conocer muy a fondo cada una de las librerías, así como también cada uno de los bloques que lo conforman, dado que funcionan como cajas negras en las cuales entra información sale reprocesada, sin conocer, en primera instancia, que procesos internos se ejecutan [42]. Como ya se ha comentado, Simulink es un ambiente de programación gráfico que permite efectuar simulaciones ya sea con datos externos o internos. El medio de comunicación entre Simulink y el módulo de adquisición previamente descrito en el capitulo anterior fue el puerto serie de la computadora en modo de recepción de datos únicamente. En el ambiente gráfico de MatLab existen bloques dedicados a efectuar la recepción de datos del puerto serie, permitiendo, entre otras cosas, crear simulaciones con datos reales en tiempo real. Sin embargo, estos bloques predefinidos poseen características que limitan dicha actividad, como por ejemplo, la de crear un buffer de datos variable. De especial interés para este proyecto es poseer un buffer de entrada que sea capaz de ajustarse a la cantidad de datos ingresada por tiempo de muestreo, para así tener acceso al mismo dato de origen. Para satisfacer esta necesidad, tuvo que ser creado un bloque de recepción serial que permitiera realizar el ajuste necesario para los datos de entrada. Para esto, se requiere tomar de la librería "*User-defined Functions*" de SIMULINK el bloque personalizado *MATLAB S-Function* [41]. Este bloque trabaja al ejecutar un script de MatLab en el cual el usuario definirá las funciones que desea realizar. Cabe señalar que estos bloques, a pesar de ser más flexibles en cuanto a las distintas funciones que les pueden ser programadas, también poseen limitantes que deben ser primero atendidas antes de intentar programarlas. La recepción serial es una función permitida en estos bloques. Así, dentro del archivo .m se declaran y definen características del puerto serie, como baudaje, bits de paridad, bits de paro, tiempo de muestreo, etc. A diferencia de los bloques predefinidos, dentro de un bloque personalizado es posible declarar un buffer de entrada que almacene los datos esperados, colocando cada dato muestreado en una posición tal que sea siempre la misma durante el tiempo de

simulación, sin tener imprevistos como el corrimiento de datos. La figura 5.2 muestra como fue programado el bloque personalizado en SIMULINK. Las sentencias del lado derecho deben ser forzosamente incluidas para que el bloque funcione adecuadamente. La creación de bloques personalizados es una tarea muy difícil y es necesario entender cada una de las instrucciones que conforman su programación.



- `mdlInitializeSizes`

Inicializa los tamaños de los arreglos.

- `mdlInitializeSampleTimes`

Inicializa el arreglo de tiempos de muestreo.

- `mdlInitializeConditions`

Inicializa los estados. Aquí se establecen la o las funciones que el usuario desea que se ejecuten en Simulink.

- `mdlOutputs`

Procesa las salidas. Dependiendo de la función o funciones programadas, será el número de salidas del bloque.

- `mdlUpdate`

Actualiza los estados discretos.

- `mdlDerivatives`

Procesa las derivadas de los estados continuos.

- `mdlTerminate`

Limpia registros cuando la simulación termina.

Fig.5.2. Pseudocódigo correspondiente a la programación de un bloque personalizado.

Una vez adquiridos los datos provenientes del FPGA, es necesario efectuar una conversión del tipo de dato, de tal forma que sea compatible con Simulink, el cual puede trabajar con datos enteros, enteros largos, dobles, signados y no signados. Para fines prácticos se optó por convertir los datos a enteros de 8

bits (fig.5.3) con ayuda del bloque predefinido llamado *Data type Conversion*, (ubicado en la librería *Commonly used blocks*) donde es posible efectuar conversiones a cualquier tipo de dato soportado en Simulink.

Fig.5.3. Bloque de Conversión

Antes de tener la información ya útil para procesos de Simulink, se realiza la decodificación de los datos de formato ASCII a números enteros signados. Esto se logra gracias a un bloque llamado *ASCII Decode* (ubicado en la librería *XPC Target/RS232*) (fig.5.4), en el cual es posible definir qué tipo de cadena se espera recibir y cómo es que se requiere decodificar, adicionalmente puede definirse el tipo de dato al que se desean transformar los datos de entrada del bloque. A la salida del mismo, se tendrán disponibles los seis datos correspondientes a los vectores de posición registrados por los sensores y con los cuales ya no es necesario realizar alguna otra operación, simplemente estarán ya disponibles para ingresar a cualquier proceso subsecuente en Simulink.

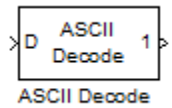


Fig.5.4. Bloque ASCII Decode

Las señales provistas por los sensores contienen un cierto grado de ruido que eventualmente afectará la posición del cuerpo rígido aunque este se encuentre en estado estático. Para resolver este problema, fue necesario introducir filtros digitales de tipo pasobajas de orden dos, con el objetivo de reducir las oscilaciones que presentan las mediciones. Un filtro pasobajas atenúa más altas frecuencias dejando pasar las señales de baja frecuencia. El diagrama de Bode para este tipo de filtros se asemeja a uno de primer orden, excepto que este cae más rápidamente. Un filtro Butterworth reducirá la señal de amplitud a un cuarto de su nivel original cada vez que la frecuencia se atenúa [40].

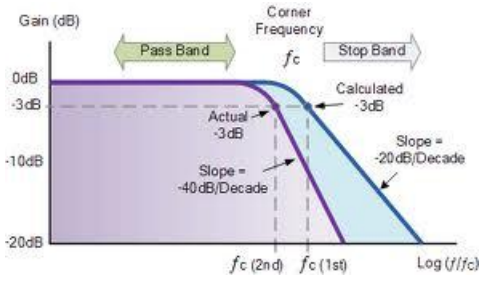


Fig.5.5. Diagrama de Bode de un filtro pasobajas de segundo orden

Luego de integrar los filtros propuestos al sistema, el ruido de los sensores fue exitosamente atenuado, con lo que se logró estabilizar el seguimiento del objeto en buena medida. Como lo ilustra la figura 5.6, cada una de las señales correspondientes a cada una de las coordenadas medidas por el acelerómetro fueron filtradas, provocando un mejoramiento en la calidad de la señal previo a ingresar al bloque TRIAD.

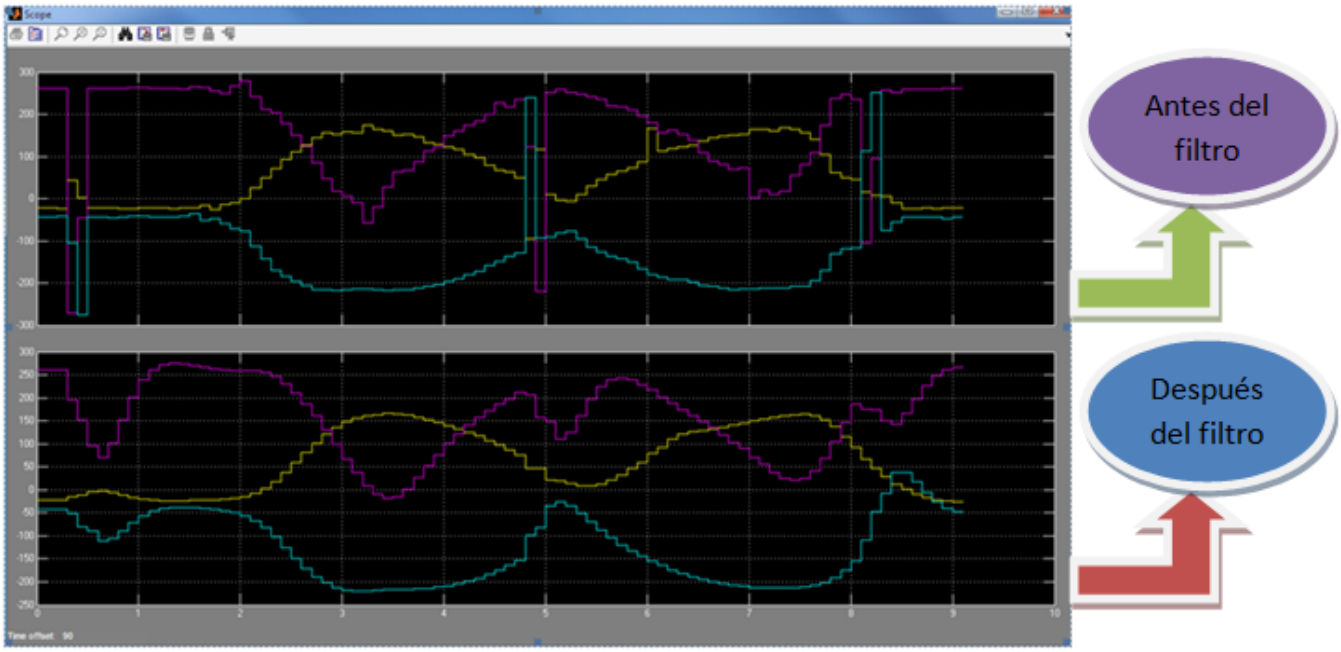


Fig.5.6. Gráfica que muestra las señales provenientes del acelerómetro antes y después del filtro.

El proceso inmediato es ingresar la información filtrada al bloque del TRIAD, el cual ha sido implementado en un bloque personalizado llamado *MatLab Function*, en donde el algoritmo requiere dos vectores de tres elementos cada uno, así que para lograr este objetivo, es necesario emplear otro bloque llamado *Reshape* (ubicado en la librería *Math Operations*) (fig.5.7), el cual, como su nombre lo indica, reestructura los elementos en un vector o matriz de tamaño *nxn*, agrupando los datos en arreglos según el usuario. Los vectores necesarios para este proceso deben ser de *1 x 3*, así que dentro de las propiedades de

este bloque se especifica el tamaño del arreglo.

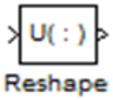


Fig.5.7. Bloque Reshape

Con el reordenamiento de los vectores, ahora ya es posible emplearlos como datos de entrada para ingresar al bloque TRIAD. Dentro del bloque *MatLab Function* (ubicado en la librería *User-defined Functions*) (fig.5.8) se programan las ecuaciones descritas en el capítulo 2 correspondientes al algoritmo TRIAD, las cuales obedecen a operaciones vectoriales. La programación del bloque se realiza mediante la asignación de un archivo .m utilizando un archivo en .m embebido en un bloque de función propia, lo que significa que se emplea semántica exclusiva de MatLab. Adicionalmente ingresan los dos vectores de referencia, uno para el magnetómetro y otro para el acelerómetro, los cuales estarán determinados por el usuario para poder establecer puntos de referencia con respecto a los vectores medidos y así lograr la determinación de la orientación en el espacio del objeto.

Fig.5.8. TRIAD implementado en un MatLab Function

El bloque TRIAD genera una matriz de rotación, cuyo concepto fue tratado en el capítulo 2 y que proporciona la orientación de un cuerpo rígido en el espacio. La matriz de rotación luego debe ser ajustada al visualizador de Simulink por medio del bloque *Rotation Matrix to VRML rotation* (fig.5.9), el cual pertenece la librería de SIMULINK llamada *3D animation.* Este bloque convierte la matriz de rotación a un vector de cuatro elementos que es compatible con el visualizador de realidad virtual.

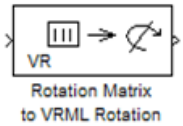


Fig.5.9. Bloque de ajuste

Finalmente, los datos obtenidos del bloque anterior ingresan a otro bloque, llamado *VR Sink* (fig.5.10), el cual funge como el visualizador de modelos en realidad virtual. Este bloque también pertenece a la librería de *3D animation*.

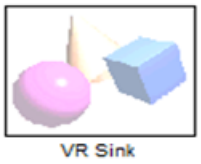


Fig.5.10. VR Sink

Dentro de este bloque se puede encontrar un menú en donde se pueden modificar parámetros como tiempo de simulación, características cinemáticas del modelo, etc., tal y como lo ilustra la figura 5.11. Dentro de este menú es posible, entre otras cosas, buscar el modelo en formato VRML para editarlo. Otras características como tiempo de muestreo, desplegado del bloque en forma automática, y el tipo de maniobra que se va a ejecutar (rotación, traslación, etc.).

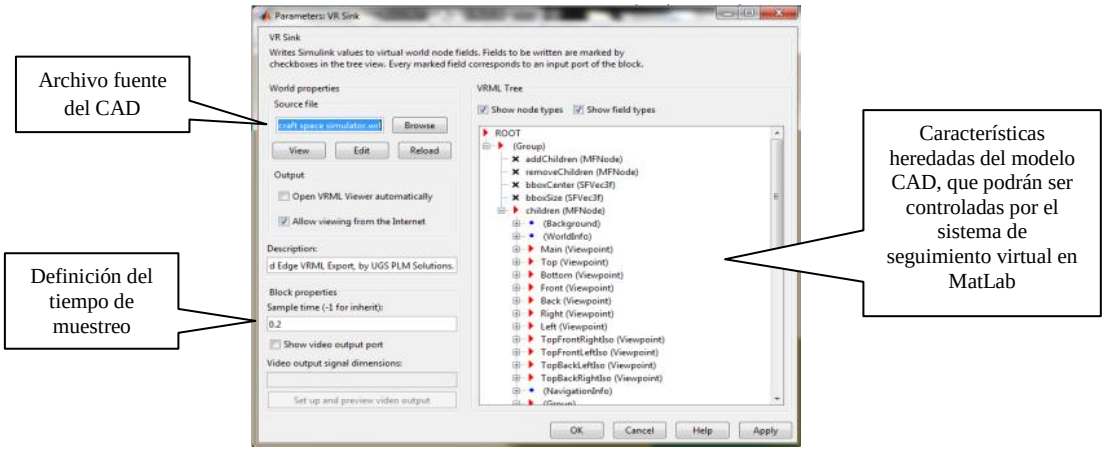


Fig.5.11. Menú VRML

El capítulo siguiente abordará la parte final que complementa a este módulo, es decir, el módulo de visualización, en donde se describirá más a detalle la funcionalidad de los bloques de animación en 3D aquí descritos.

Capítulo 6

MÓDULO DE VISUALIZACIÓN

La tecnología de la realidad virtual proporciona a científicos e ingenieros los medios de entrada y realimentación que elevan sus esfuerzos creativos. El acoplamiento intuitivo con grandes cantidades de datos se vuelve un poco más sencillo para un usuario dentro de un entorno virtual, permitiendo la interacción con visualizaciones sensacionales que acomodan sus dudas subverbales.

Sin duda, el gran crecimiento en el desempeño de supercomputadoras y en particular de computadoras con grandes alcances en capacidades de graficación, han permitido la incorporación de nuevas tecnologías de visualización y modelado como lo es la realidad virtual, la cual ha proporcionado un mejor entendimiento de un fenómeno o hecho real mediante su simulación tridimiensional e interacción con equipo especializado. En este capítulo se abordarán los detalles asociados con la visualización de los modelos virtuales para la comprobación gráfica de los movimientos realizados en la plataforma del simulador de vuelo satelital.

6.1. Marco teórico de la visualización

En los primeros días de las simulaciones por computadora, el despliegue de datos de la simulación era presentado por medio de tablas o matrices, mostrando cómo dichos datos eran afectados por varios cambios en los parámetros de simulación. Con el surgimiento y posterior evolución de herramientas gráficas de hardware y software cada vez más potentes, se comenzaba a notar que la representación visual, en forma de gráficas (fig.6.1) o imágenes móviles generadas a partir de ciertos datos, se convertía poco a poco en un modo efectivo de interacción con una simulación. Los intentos por permitir la portabilidad y la interoperabilidad de presentaciones virtuales,así como la aceleración de su generación, guió al desarrollo de estándares de despliegue computacionales así como archivos con formato de gráficas [1].

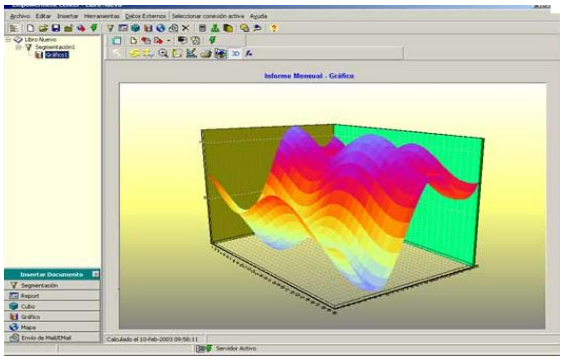


Fig.6.1. Representación visual en forma de gráficos

La creciente evolución de la capacidad de los sistemas computacionales, en cuanto al manejo masivo de información gracias a una mayor capacidad de procesamiento, ha permitido desarrollar simulaciones avanzadas de sistemas altamente complejos, resaltando la característica de una no-linealidad (fig.6.2) que la mayoría de ellos presentan. Así mismo, las plataformas en las cuales se desarrollan dichas simulaciones, poseen un mayor número de herramientas que mejoran, entre otras cosas, la calidad y la rapidez de la visualización ejecutándose prácticamente en tiempo real. No obstante, a pesar de la notable mejoría que estos sistemas informáticos modernos han adquirido, la demanda de recursos de la plataforma de cómputo se ve afectada en función de las operaciones que el sistema requiera. Con lo anterior queda claro que si la intención del diseñador es crear ambientes altamente complejos, será necesaria la utilización de una computadora muy potente capaz de satisfacerlos requerimientos creativos. Adicionalmente, a medida que se incrementa la complejidad de los sistemas de procesamiento, aumenta también la complejidad del modelado de sistemas, en el sentido de que el usuario deberá tener experiencia en el uso de tales sistemas, puesto que se requerirá de una amplio *background*teórico y técnico para efectuar simulaciones más avanzadas [1].

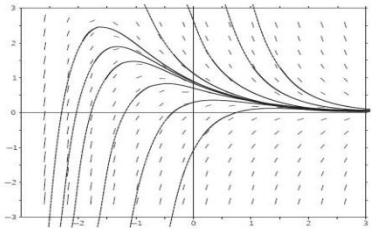


Fig.6.2. Representación gráfica de un sistema no lineal

La representación visual permite un descubrimiento interactivo de nuevo conocimiento. Las animaciones pueden ser usadas para experimentar una simulación en tiempo real.

La demanda, por más interactividad con simulaciones virtuales,guía al uso y adaptación de varias

tecnologías, incluyendo herramientas que usan la *realidad virtual*o elLenguaje de Modelado de Realidad Virtual (VRML), entre otras.

6.1.2. Categorías de Representaciones Virtuales

Un número de diferentes clasificaciones y taxonomías han sido propuestas para Representaciones Virtuales (o VisRep, por sus siglas en inglés), entre las más importantes se pueden mencionar:

- VisRep científica
- VisRep de Datos/Información
- Análisis virtual
- VisRep de dispositivos móviles

VisRep científicas

Las VisRep científicas [1] (fig.6.3) son desplegados interactivos visuales de datos espaciales/geométricos asociados con procesos científicos. Desde sus inicios, los científicos y los ingenieros han usado VisRep para explicar fenómenos científicos y procesos asociados con sus simulaciones. A mediados de los 80s, supercomputadoras gráficas y sistemas de visualización avanzados ya tenían incorporados modelado, animación y facilidades para renderizarmodelos virtuales.

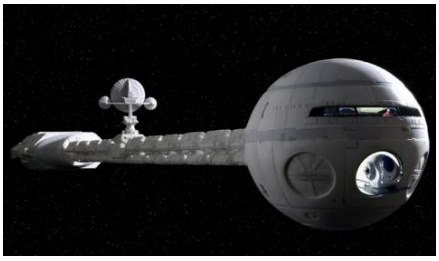


Fig.6.3. VisRep científica

VisRepde Datos/Información

Este tipo de representación virtual se enfoca en información o datos no geográficos, e involucra la selección, la transformación y representación de información no geográfica en una forma que facilita la interacción humana para su exploración y entendimiento. Distintas técnicas han sido desarrolladas para permitir al usuario modificar la representación visual en tiempo real, proveyendo de esta forma percepción no paralela de patrones y de relaciones estructurales en la información.

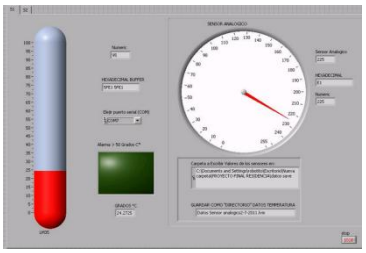


Fig.6.4. VisRep de datos

Análisis virtual

Esta es una variante de VisRep de datos/información y VisRep científicas y se enfoca en ir más allá de la interacción con la representación visual de la simulación para el análisis de información. El uso de representaciones visuales e interacciones para acelerarla hacia información compleja es lo que distingue al análisis virtual de otros tipos de herramientas analíticas. Dichas herramientas incorporan tecnologías de otros campos del conocimiento, como estadística, ciencia cognitiva, probabilidad, entre otras. Así mismo estas herramientas son usadas para maximizar la capacidad humana para percibir, entender y razonar sobre información dinámica compleja y diversas situaciones.

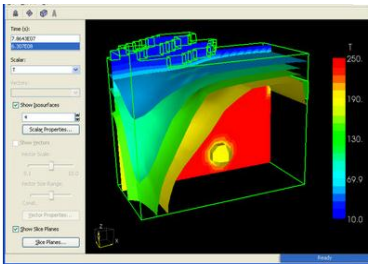


Fig.6.5. VisRep analítica

VisRep de dispositivos móviles

Los dispositivos móviles inalámbricos están siendoomnipresentes debido a su pequeño tamaño y su extenso margen de funcionalidad en diferentes aplicaciones. Con sus habilidades crecientes, pueden proveer representaciones visuales útiles en varias aplicaciones, incluyendo entrenamiento, mantenimiento y reparación, cuidado de la salud, respuesta en emergencias y administración.

Fig.6.6. VisRep en smartphones

La representación visual empelada en esta tesis recae principalmente en las tres primeras categorías listadas arriba, dado su carácter de análisis cinemático, el despliegue de información no geográfica y su manipulación, así como el uso de disciplinas como la estadística y la probabilidad para su análisis. Como ya se abordó en el capítulo anterior, MatLab incorpora estos tipos de categorías en su plataforma de simulación, convirtiéndolo en una herramienta útil y de gran ayuda para científicos e ingenieros. Adicionalmente, MatLab permite la simulación de mundos virtuales que son creados en Lenguaje de Modelado de Realidad Virtual, dotando de un mayor realismo a cualquier reproducción de algún fenómeno físico.

6.2. Realidad virtual y VRML

La realidad virtual y los mundos virtuales son ejemplos de tecnologías que resultan altamente eficacesen la representación visual de simulaciones, incluso aunque las motivaciones para su desarrollo y sus aplicaciones vayan más allá de eso.

La realidad virtual (VR) es una simulación tridimensional generada o asistida comúnmente por una computadora, que describe algún aspecto del mundo real o ficticio, en el cual el usuario tiene la sensación de pertenecer a ese ambiente sintético ó interactuar con él. La VR permite interactuar con mundos tridimensionales de una manera más natural, por ejemplo, un usuario puede realizar ciertas acciones dentro de un modelo de realidad virtual como desplazarse, moverse, caminar a través de él o levantar algún objeto, de esta forma asemeja de forma experimental situaciones que asemejan al mundo real.

Las partes básicas de un sistema VR son:

1) El modelo de simulación
2) La representación del ambiente virtual
3) La(s) entrada(s) y salida(s)
4) Usuario [1]

Como se muestra en la figura 6.7, un diagrama a bloques en el que se describe el modelo genérico de un sistema VR.

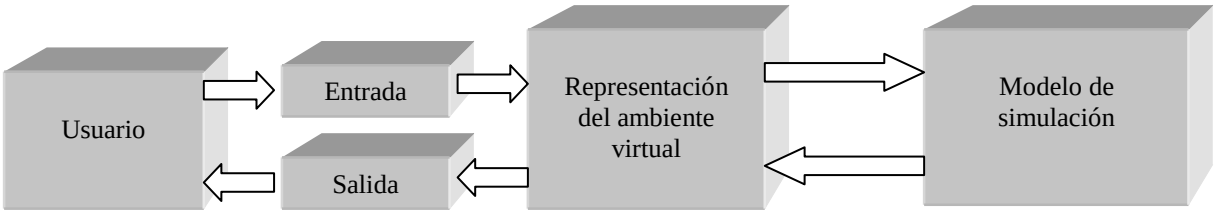


Fig.6.7. Modelo genérico de un sistema VR.

6.2.1. Modelo de simulación

El modelo es la representación matemática del sistema que se está presentando. Un modelo virtual necesita responder de forma dinámica ante la entrada del usuario. Se pueden crear modelos matemáticos de realidad virtual de alta complejidad, sin embargo lo más importante de estos modelos es la forma en la que se asocian con un sistema visual y auditivo.

6.2.2. Entrada

En este bloque se incluyen los datos o bien, los dispositivos de los cuales se obtienen los datos de entrada que interactúan con el ambiente y el modelo virtual.

6.2.3. Salida

En el esquema de la figura 6.7, el bloque de salida se refiere a la tecnología mediante la cual el usuario percibe los estímulos y presencia los movimientos gráficos y los sonidos que puede tener asociada la simulación virtual.

6.2.4. Usuario

Es quien recibe los estímulos de parte del sistema y a su vez, se encarga de realimentarlo y definir su comportamiento.

La tecnología asociada a un sistema VR va más allá de los medios físicos que se utilizan para su visualización, interacción o realimentación en este tipo de sistemas, también se debe incluir el software que se utiliza para el modelado de este tipo de sistemas. Ambos elementos se complementan el uno con el otro, el hardware se encarga no sólo de mandar señales sino también de recibirlas, mientras que el software es el encargado de procesarlas y transformarlas en un nuevo comportamiento del mundo virtual.

La realidad virtual es una técnica descrita en el lenguaje especializado (VRML) en el cual es posible describir mundos virtuales, así como también la descripción de algoritmos, resultado del modelado matemático de un sistema o proceso físico. Así, editores como *VR Builder*, *VRMLPad*, *OpenGL*, *BabelX3d*, entre otros, permiten desarrollar mundos virtuales, sus formas, sus movimientos, etc. En la figura 6.8, se muestra la visualización de un modelo VR creado en el ambiente *VRealmBuilder*.

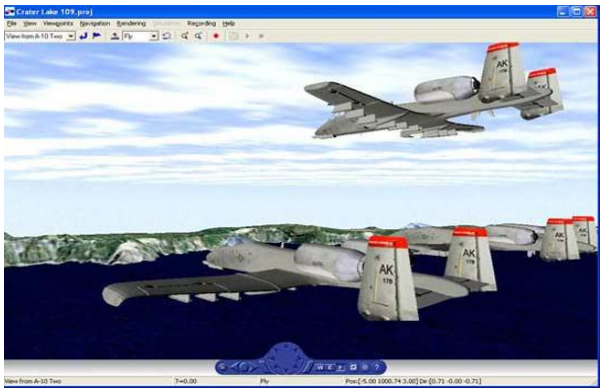


Fig.6.8. Visualización de un sistema VR .

MatLab cuenta con su propio editor, *VRealmBuilder*, el cual, entre otras funciones, es un editor gráfico que posee estructuras y figuras predeterminadas, las cuales pueden ser usadas simplemente al ir seleccionandoalgunas de ellas. Algunos otros editores, como *VRML Pad* se caracterizan por editar al mundo virtual con base en código, siendo más flexible y versátil que un editor gráfico. Para el desarrollo de esta tesis se empleó el ambiente de *VRML Pad* para el desarrollo del modelo de la plataforma de simulación satelital.

6.3. VRML Pad

Es un ambiente de desarrollo creado por Parallel Graphics [43] que permite la creación de mundos virtuales, el cual es un poderoso editor que ofrece los siguientes beneficios:

- Detección dinámica de errores
- Autocompletado inteligente
- Soporte visual para las operaciones de árbol
- Previsualización de la escena
- Previsualización de nodos individuales

Estas características convierten a *VRML Pad* en una herramienta versátil, intuitiva y útil,dada la flexibilidad ofrecida por cada una de ellas.

Este lenguaje de programación se basa en la creación de árboles, esto es, estructuras que ponderan parámetros del mundo virtual, siendo los de mayor ponderación aquellos asignados al principio de la codificación. Esta ponderación se da gracias al uso de estructuras algebraicas llamadas TRANSFORMS, permitiendo la creación de distintos marcos de referencia móviles, respetando siempre al marco de referencia inercial.

Este editor también permite la modificación de fondos de animación con base en colores definidos por el usuario o bien por un conjunto de imágenes que conforman un cubo virtual, como sigue:

- Frente
- Superior
- Inferior
- Izquierda
- Derecha

Fig.6.9. Cubo virtual que muestra la integración de un escenario para un modelo de simulación.

Este set de imágenes puede dotar de un realismo mayor a la simulación, si son parte del ambiente del sistema real sobre el cual se hará la simulación, ya que permiten la integración de un escenario donde estará embebido el modelo de simulación.

6.3.1. Creación de figuras

La creación de figuras se desarrolla dentro de estructuras que agrupan una serie de comandos que definen su geometría, el color, el aspecto y luminosidad. Las figuras base que se pueden definir en este editor son:

- Esferas
- Conos
- Cubos
- Cilindros
- Prismas

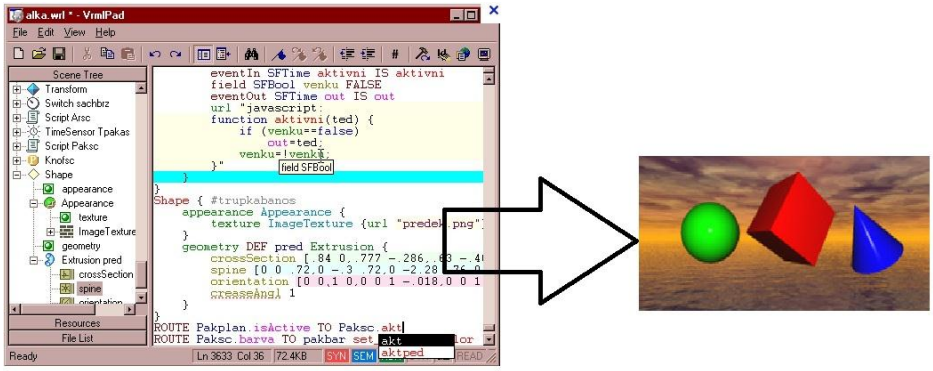


Fig.6.10. Creación de figuras en el editor de VRML

No obstante, el uso de estas figuras limita el realismo debido a que sus formas son relativamente sencillas y un tanto atípicas en sistemas complejos reales, más bien forman parte de sistemas ideales. Si el usuario desea dotar a su mundo virtual de mayor realismo, el editor VRML también puede trabajar con archivos de imagen importados. Estos archivos pueden ser importados de sistemas CAD, en donde abunda la calidad del detalle y complejidad de sistemas virtuales.

La mayoría de los sistemas CAD, tal como sucede con *Solid Edge*, poseen la opción de guardar el diseño actual en formato VRML. Así mismo, como ya se había mencionado en el capítulo1, los sistemas CAD son también sistemas basados en realidad virtual, pero con propósitos distintos. Una vez que se ha guardado el diseño en formato VRML, ahora es posible editarlo en *VRML Pad*.

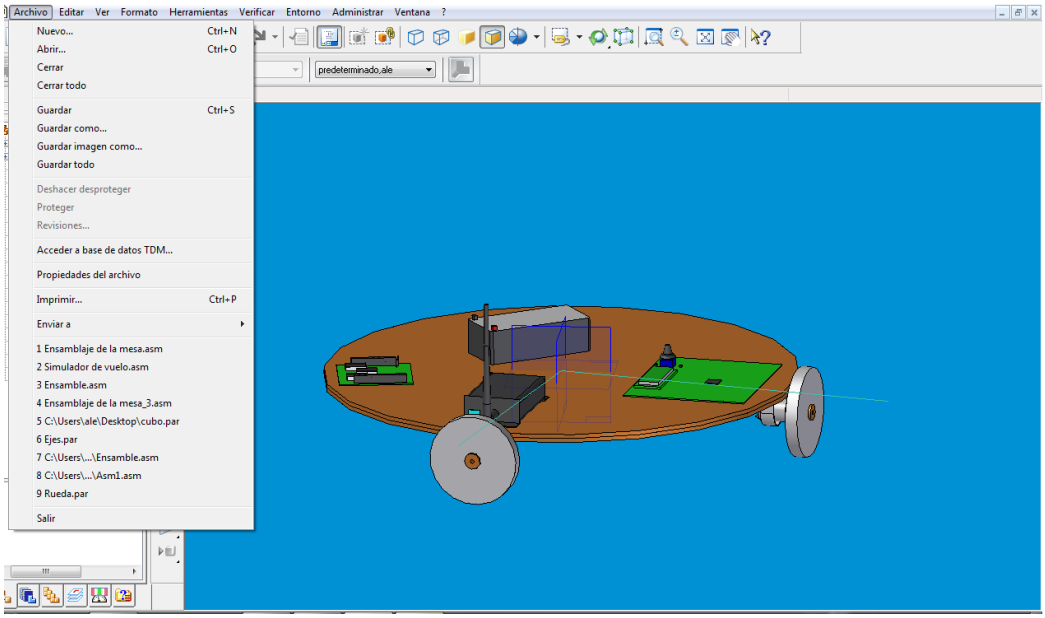


Fig.6.11 Interfaz de desarrollo del Software CAD Solid Edge

El archivo, una vez abierto en el editor de VRML, despliega una gran cantidad de subpartes llamadas *shapes* (fig.6.12), las cuales son resultado de cada una de las operaciones que se realizaron en el software

CAD para desarrollarel diseño. Entre mayor sea el número de operaciones que se realicen en el software CAD, mayor será también el número de subpartes que tendrán que ser agrupadas en las estructuras TRANSFORMS. Una recomendación sería que se efectué un diseño lo más concreto y rápido posible en el software CAD, para que la reagrupación en el editor VRML sea amena.

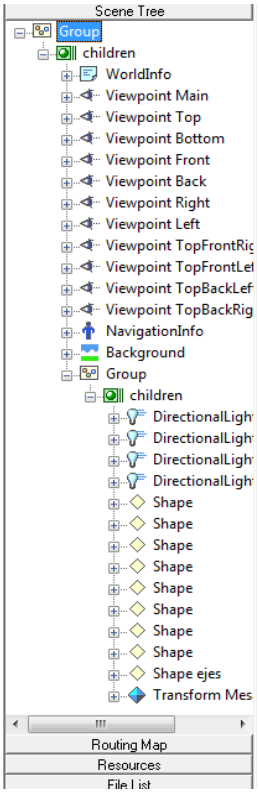


Fig.6.12. Árbol que agrupa las subpartes que conforman al modelo virtual (shapes).

Una vez reagrupados todos los *shapes* en el editor, está listo para ser exportado nuevamente a SIMULINK, dentro del bloque *VR Sink*, ya descrito en el capitulo anterior. Sobre el agrupamiento principal de *shapes* se aplicará el resultado de la simulación, reflejándose en la rotación en tres ejes del simulador de vuelo satelital diseñado en *Solid Edge* y cuyo modelo se muestra en la figura 6.13.

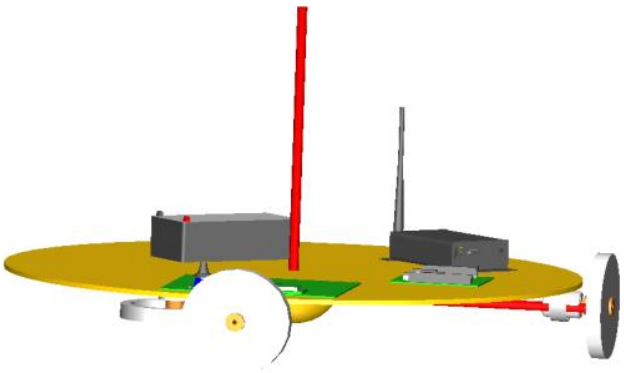


Fig.6.13. Simulador de vuelo satelital diseñado en CAD y exportado como archivo VRML

La estructura TRANSFORM agrupa a los elementos que conforman un diseño y permite efectuar los tipos de movimientos que se llevaran a cabo dentro del mundo virtual. Una vez declarado el tipo de operación, el bloque automáticamente abrirá una entrada para que la información obtenida en la simulación ingrese y corresponda a esa operación, reflejándose en la actuación del sistema.

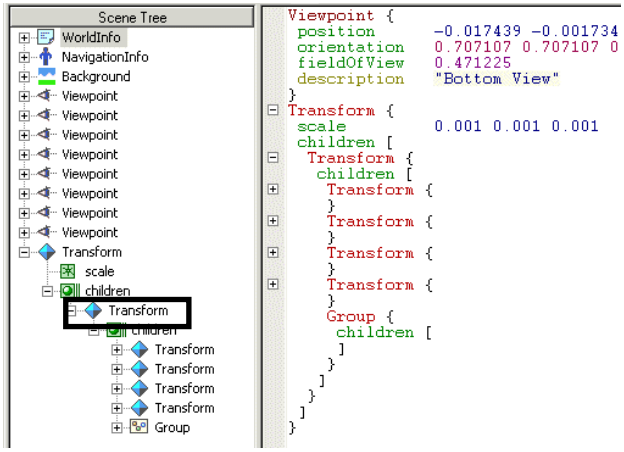


Fig.6.14. Imagen que ilustra la estructura TRANSFORM dentro de un editor de VRML

El bloque VR Sink cuenta con su propia interfaz gráfica de usuario en la cual es posible manipular al objeto antes y después de ejecutar la simulación. Parámetros como tipo de vista, zoom, tiempo de ejecución, etc., pueden ser modificables por medio de esta herramienta. En la figura 6.15 se esquematiza el módulo de visualización *Grosso Modo* , en donde juegan un papel fundamental los bloques TRIAD, el de la adecuación del sistema de orientación generado por el TRIAD al ambiente de realidad virtual y por supuesto, el bloque de visualización animada en 3d.

Fig.15. Representación esquemática del funcionamiento del módulo de visualización en Simulink

El módulo de visualización tendrá que satisfacer las expectativas con respecto al seguimiento realizado al simulador de vuelo real que se encuentra en el laboratorio. Por ello radica sobre él un especial interés, además de que, una vez que se evalúe su desempeño, podrá convertirse en una herramienta gráfica de alta calidad para el desarrollo de sistemas aeroespaciales.

6.4. Interfaz Gráfica de Usuario [44]

Adicionalmente se diseñó una Interfaz Gráfica de Usuario, o *GUI*, con la finalidad de facilitar la interacción entre el usuario y el modelo implantado en Simulink. La *GUI* está conformada por botones e indicadores de texto, los cuales son programados desde un script en MatLab, obedeciendo a las instrucciones del usuario. La figura 6.16 muestra la GUI usada en este proyecto y en la cual se aprecian tres botones y seis editores de texto. Los botones se conforman por el botón de arranque/paro de la simulación, el segundo es el de Play/pause y el tercero para cerrar y abandonar la aplicación. Los editores de texto modifican los vectores de referencia que alimentan al bloque TRIAD ya mencionados en el capítulo 5, con la ventaja de que pueden ser modificados en cualquier momento durante el tiempo de simulación y sin tener que ingresar a cada bloque dentro del modelo implementado.

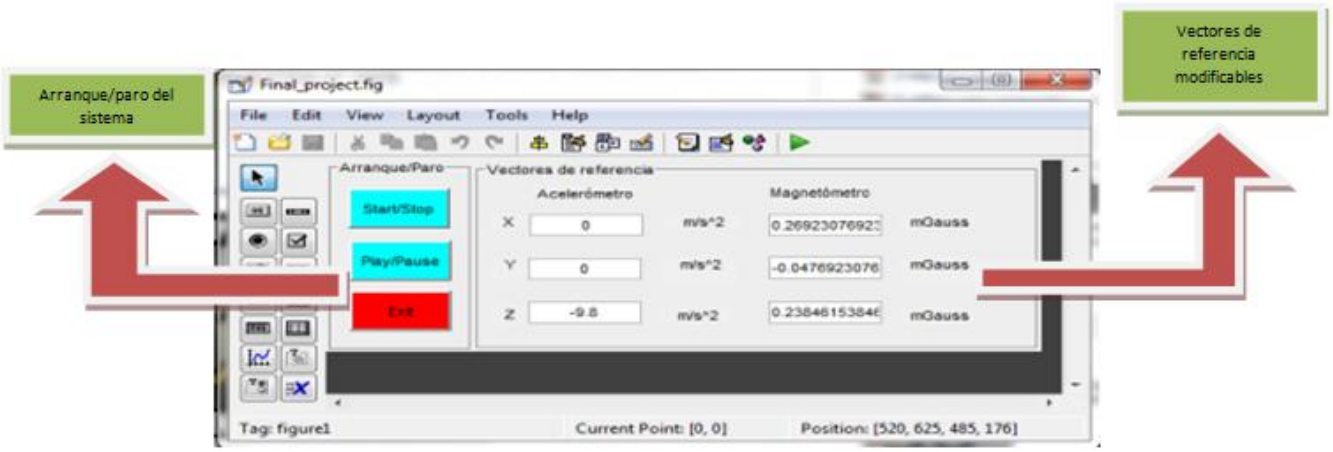


Fig.6.16. Interfaz Gráfica de Usuario del sistema

El capítulo 7 abordará los resultados experimentales de este proyecto, reportando entre otras cosas, los beneficios de haber empleado una GUI para su control. Así mismo se describirán las primeras pruebas realizadas, como se llegó al seguimiento y cuales son las vertientes a las que este podría estar sometido en un futuro.

Capítulo 7

PRUEBAS EXPERIMENTALES

Una de las fases más importantes dentro del flujo de desarrollo de cualquier sistema o innovación tecnológica es el que se refiere al desarrollo de pruebas experimentales. Es así porque es en esta fase se validan y depuran muchas de las suposiciones que *a priori* se pudieron haber realizado en algunos de los subprocesos que integran al sistema o al desarrollo en general.

En este capítulo se abordarán con cierto detalle las pruebas efectuadas al sistema de seguimiento en tiempo real, las cuales fueron realizadas en el laboratorio de sistemas aeroespaciales del Instituto de Ingeniería de la U.N.A.M. Las pruebas que se documentan en esta sección, inician con pruebas de seguimiento del modelo de la MSA en un eje, para posteriormente dar paso a las pruebas de seguimiento en tres ejes. Adicionalmente se mencionan parte de las experiencias obtenidas en pruebas de control de orientación en un eje, con lo cual se sientan las bases para la realización de futuros desarrollos y pruebas incluyendo un bloque de control de orientación en tres ejes.

7.1. Seguimiento virtual en un eje

El desarrollo de algoritmos de seguimiento virtual para su validación práctica sobre un simulador de vuelo satelital (MSA),parte del planteamiento de su realización desde un punto de vista que incluye dos aspectos importantes, por un lado, el análisis de factibilidad técnica y por otro lado su integración, considerando en todo momento las herramientas (hardware y software)que serán de gran utilidad para su desarrollo. Para llegar al objetivo que plantea este trabajo de tesis, se planteó al inicio, como la ruta crítica de desarrollo, efectuar el seguimiento en un solo eje (Z), no sólo por la relativa facilidad técnica de su implementación, sino porque dentro del grupo de trabajo en el II-UNAM, se encontró una línea de trabajo coincidente, en la cual ya se contaba con resultados exitosos de algoritmos de control en un eje implementados en un dispositivo reconfigurable FPGA, así como diversas simulaciones en MatLab, lo cual aceleraría para el trabajo de esta tesis, la obtención de los primeros resultados de seguimiento en tiempo real utilizando un modelo virtual.

7.1.2. Experimento

Previo a la realización de las pruebas que se describen en esta sección, ya se contaba con algunos recursos tales como sensores de navegación; sistemas embebidos desarrollados en una plataforma FPGA

(integrados por módulos de adquisición de datos, procesamiento y actuación), así como módems RF. Estos módulos y equipos ya se encontraban montados sobre el simulador de vuelo satelital y debidamente acondicionados para comenzar a realizar pruebas e implantación de algoritmos de determinación y control de orientación en uno y tres ejes. Se contaba también con MatLab y su plataforma de simulación SIMULINK. Faltaba únicamente la implementación de un modelo matemático para efectuar el seguimiento de un modelo u objeto virtual.

Como ya se ha mencionado, una de las primeras tareas dentro de la ruta crítica de desarrollo de esta tesis fue la posibilidad de llevar a cabo la implementación de un esquema en hardware y software, el cual permitiría efectuar el seguimiento en un eje del modelo virtual de un objeto ante la realización de diversas maniobras, que para efectos de este trabajo fue el simulador de vuelo satelital. La experiencia obtenida, luego de su desarrollo y validación experimental, permitió avanzar con mayor seguridad y confianza hacia la implementación en tres ejes.

El esquema de seguimiento virtual en un eje está basado en un sistema embebido que proporciona, en un tiempo de muestreo determinado, la posición angular de un objeto en un eje. Esta posición angular es calculada a partir de dos coordenadas (X e Y) medidas por un magnetómetro triaxial. La arquitectura del sistema embebido en el FPGA incluye un núcleo periférico de procesamiento (CORDIC), el cual se conecta al bus de periféricos del procesador MicroBlaze, y que provee el ángulo de posición a partir del cálculo de la función ángulo cuya tangente (atan) del cociente de la componente "y" y "x" del magnetómetro triaxial. Posteriormente, este ángulo es enviado vía RF a la PC, específicamente a la plataforma de simulación SIMULINK, donde es acondicionado para ser compatibilizado con los algoritmos de determinación de la posición angular. Finalmente, estos datos son aplicados al movimiento (seguimiento) de un objeto virtual, en este caso el diseño en CAD del simulador de vuelo satelital, el cual refleja exactamente las maniobras en tiempo real que sufre el simulador de vuelo satelital real. La figura 7.1 muestra, en diagrama de bloques, la dirección del flujo de la información desde que es medida por los sensores hasta donde es visualizada virtualmente en SIMULINK.

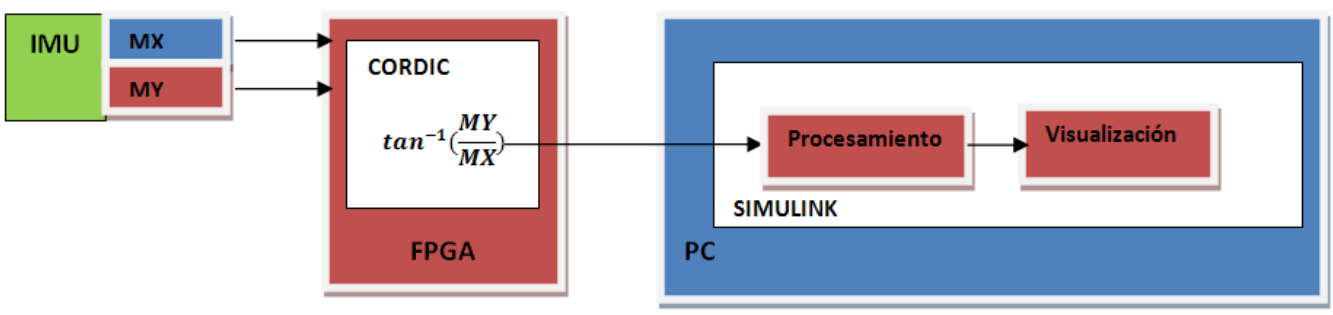


Fig.7.1. Implementación esquemática del seguimiento en un eje, descrito en diagrama de bloques

La figura 7.2 muestra como está instrumentado el simulador de vuelo satelital, resaltando los subsistemas esenciales para el desarrollo de la prueba de seguimiento en un eje.

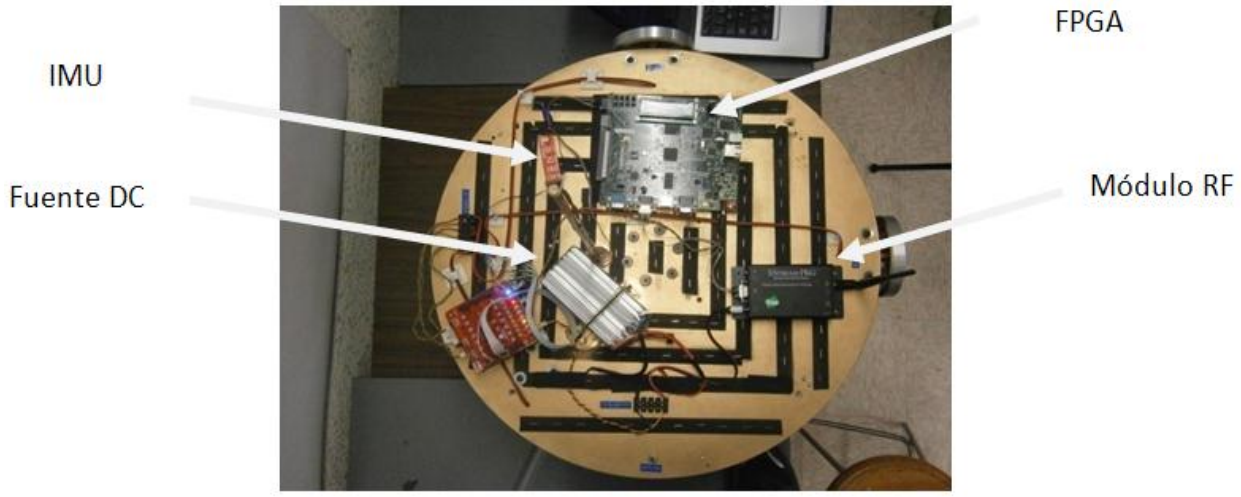


Fig.7.2. Simulador de vuelo satelital con vista alrededor del eje Z

El ángulo es proporcionado en punto flotante y en radianes. Este ángulo se envía a SIMULINK gracias a los bloques mostrados en la figura 3, *Serial Configuration* y *Serial Receive,* donde se definen opciones como la selección del número de puerto, el Baudaje, los bits de datos, paridad, tiempo de muestreo, tamaño de buffer, etc. La diferencia de estos bloques con el bloque personalizado usado en tres ejes radica en la limitante del tamaño de buffer de entrada, cuyo tamaño es fijo. Por tal motivo, y en virtud de los requerimientos de la aplicación, en el sentido de contar con un buffer de dimensión variable, debido a los cambios en la extensión de los datos recibidos del sensor de navegación, se decidió el diseño de bloques personalizados para las funciones de comunicación serie. El esquema completo que se utilizó para el seguimiento en un eje se muestra en la figura 7.3, en él se puede apreciar que está integrado por bloques personalizados y bloques propios de Simulink.

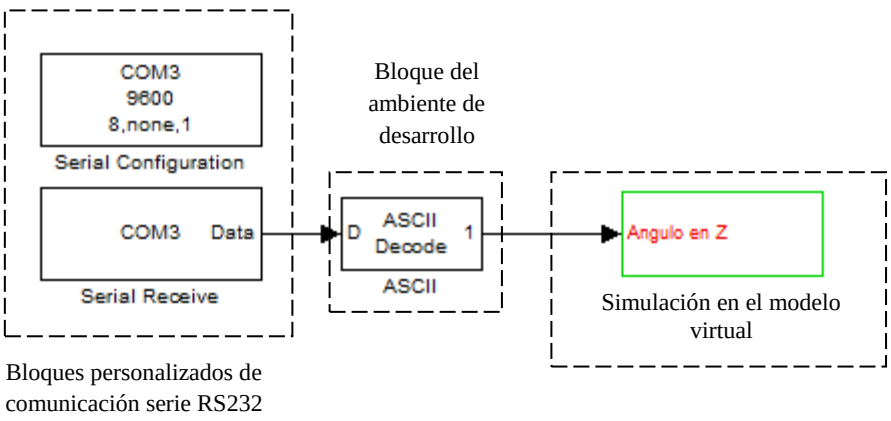


Fig.7.3. Implementación del modelo de seguimiento en un eje (Z) en SIMULINK.

7.1.3. Funcionamiento del sistema implantado en SIMULINK

El ángulo ingresa a SIMULINK en formato ASCII, así que tiene que ser decodificado para compatibilizarlo y poder ejecutar el algoritmo que genera la matriz de rotación. Una vez que el ángulo es acondicionado para efectuar simulaciones, ingresa a un bloque personalizado llamado *MatLabFunction*, caracterizado por permitir la programación del comportamiento del bloque a partir de un código *script*. La función implementada en este bloque es la matriz de rotación en torno al eje Z con base en los ángulos de Euler (Fig.7.4). En este caso no afectó usar el método de ángulos de Euler (dadas las eventuales singularidades) puesto que sólo es necesario establecer una posición definida sobre el plano perpendicular al eje Z. El ángulo alimenta a la matriz y ésta arroja la posición en un solo un eje de cualquier cuerpo rígido.

$$R(\psi) = \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -\text{sen}(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fig.7.4. Matriz de rotación.

Donde ψ es el ángulo de Euler correspondiente al eje Z.

La matriz de rotación correspondiente al eje Z luego es transformada a una rotación compatible con la visualización de realidad virtual gracias al bloque *RotationMatrix to VRML rotation.*

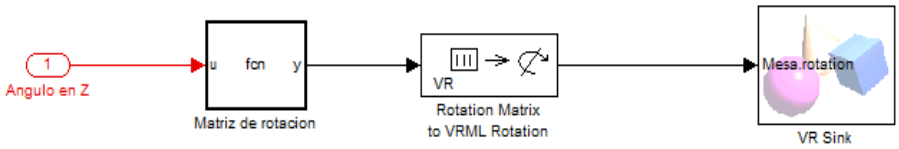


Fig.7.5. Esquema que muestra el ingreso del ángulo al bloque MatLab Function, su compatibilización con la rotación en VRML y su ingreso al bloque de visualización.

Las pruebas fueron exitosas, muy precisas y lograron validar el seguimiento virtual del objeto en un eje. El esquema experimental se muestra en la figura 7.6.

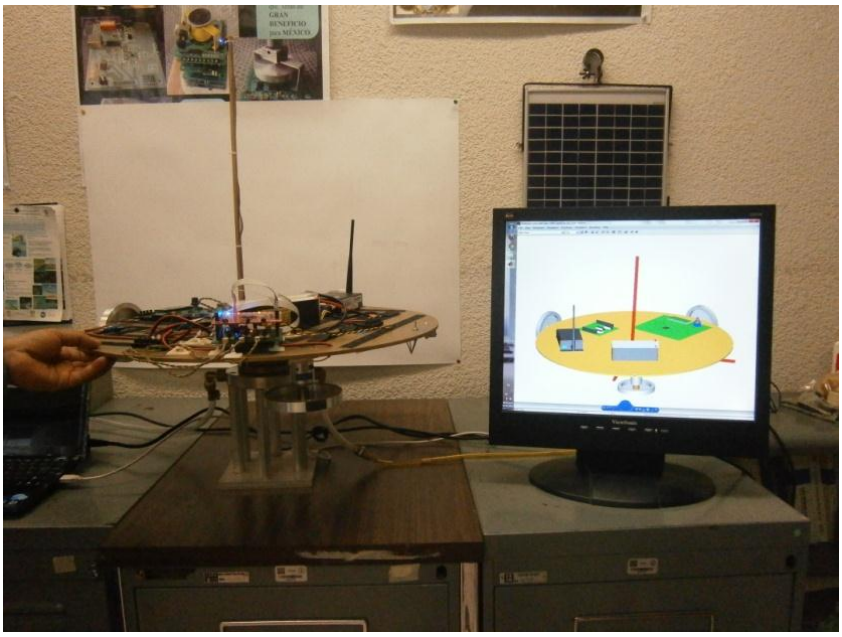


Fig. 7.6. Pruebas de seguimiento en un eje.

7.2. Seguimiento virtual en tres ejes

El desarrollo de algoritmos para el seguimiento virtual en tres ejes surge a partir del éxito obtenido en las pruebas de seguimiento virtual en un eje. No obstante, hubo retos que afrontar, como el hecho de que el sistema ya no sólo contaría con la adquisición de un solo dato, sino de seis. Otra novedad incluía también un algoritmo de determinación de la orientación. En la bibliografía se presentan algunos tipos de algoritmos, entre los más utilizados se encuentra el denominado TRIAD y el Gauss-Newton. Sin embargo, como se mencionó en el capítulo 2, por efectos de practicidad en cuanto a la complejidad del procesamiento asociado, para esta tesis se decidió utilizar el TRIAD. TRIAD es un algoritmo de determinación de la orientación, que persigue la fusión de sensores de navegación para la obtención de un cuaternión, que es una de las representaciones más utilizadas de la orientación de un cuerpo rígido en el espacio. El algoritmo TRIAD describe un sistema que necesita de dos vectores de medición y dos vectores de referencia para que a partir de ellos genere una matriz de rotación, o bien un cuaternión. Los vectores de medición corresponden a los medidos por el magnetómetro y el acelerómetro.

Los vectores de referencia son los provistos por el usuario. Posteriormente estos datos, producto de las operaciones ejecutadas dentro del bloque TRIAD, conformarán una matriz de rotación, otra forma común de describir los movimientos de un cuerpo en el espacio, y que permitirán conocer la posición en tres ejes del simulador de vuelo satelital. Finalmente, con el bloque *Rotation Matrix to VRML rotation* se compatibilizará la orientación obtenida a la orientación del modelo en realidad virtual.

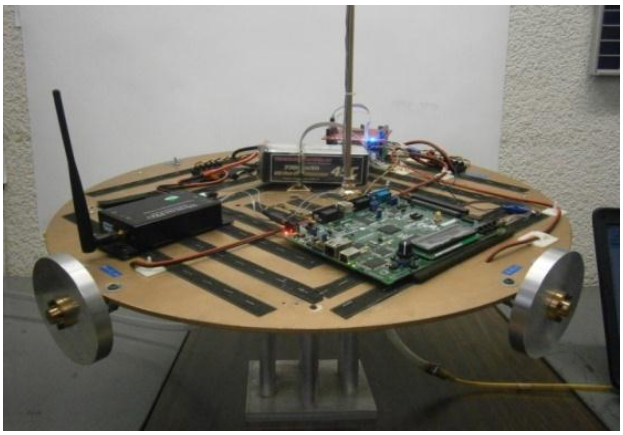


Fig.7.7. Vista isométrica del simulador de vuelo satelital

La conceptualización anterior parece sencilla y fácil de implementar, sin embargo hubo problemas que frenaron el avance en su implementación, puesto que se debieron realizar una serie de adecuaciones en el ambiente de SIMULINK para poder trabajar con una cantidad de datos mayor a la utilizada en el seguimiento en un eje.

7.2.1. Experimento

La adquisición de datos en SIMULINK se logró con ayuda de la creación de un bloque personalizado, utilizando el bloque *MatLab S-Function* en el cual, a diferencia del bloque *MatLab Function*, se pueden programar funciones mucho más complejas, como las requeridas para la recepción de datos en SIMULINK del puerto serie. Los bloques predefinidos en SIMULINK presentan limitantes, debido a que fijan un tamaño de buffer de entrada, lo que afecta la recepción de datos del puerto serie al no contener la trama de datos correspondiente a cada tiempo de muestreo, ocasionando problemas al momento de determinar la orientación del cuerpo rígido. Con la ayuda de un bloque personalizado, inmediatamente se logró solucionar el problema, permitiendo la entrada de datos correspondiente a cada tiempo de muestreo, permitiendo ingresar la coordenada correcta al TRIAD (fig.7.8) sin problemas de desfase.

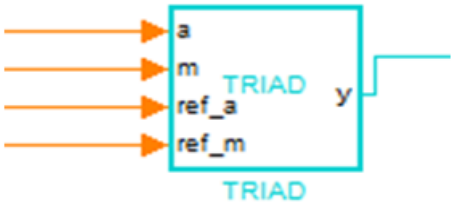


Fig.7.8. Bloque TRIAD con dos vectores de medición y dos vectores de referencia

7.2.2. Funcionamiento del sistema implementado en SIMULINK

El módulo de adquisición de datos implementado en el FPGA envía una trama de seis datos correspondientes a las coordenadas medidas por cada sensor y enviadas vía RF hacia la PC, donde se ejecuta el ambiente de simulación SIMULINK y en donde ingresan por medio del bloque personalizado de comunicaciones seriales RS232. Luego de ser decodificados, los datos son separados y reagrupados en dos grupos, correspondientes a cada sensor (magnetómetro y acelerómetro). Posteriormente ingresan al bloque TRIAD junto con los vectores de referencia, en donde luego de un procesamiento rápido es posible obtener la posición del cuerpo rígido en el espacio. Dicho procesamiento se lleva a cabo en tiempo real, lo que permite comparar el movimiento del cuerpo real con el virtual, y así efectuar comparaciones y análisis. La figura 7.9 esquematiza en un diagrama de bloques el flujo de la información desde que es adquirida hasta que es desplegada virtualmente.

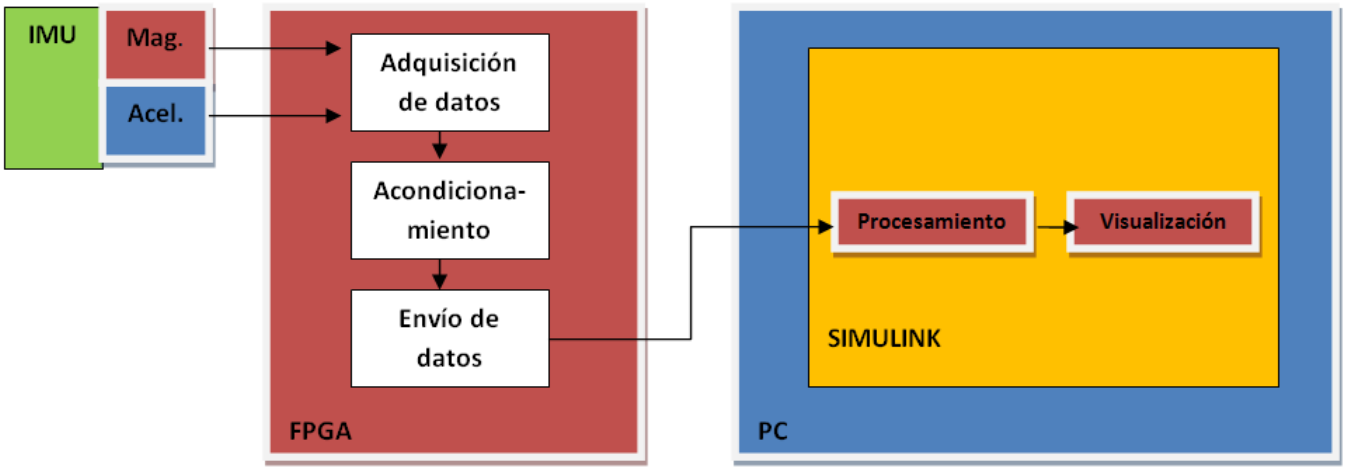


Fig.7.9. Sistema en diagrama de bloques implementado en SIMULINK para el seguimiento en tres ejes

El modelo final implantado en Simulink quedó de la siguiente manera:

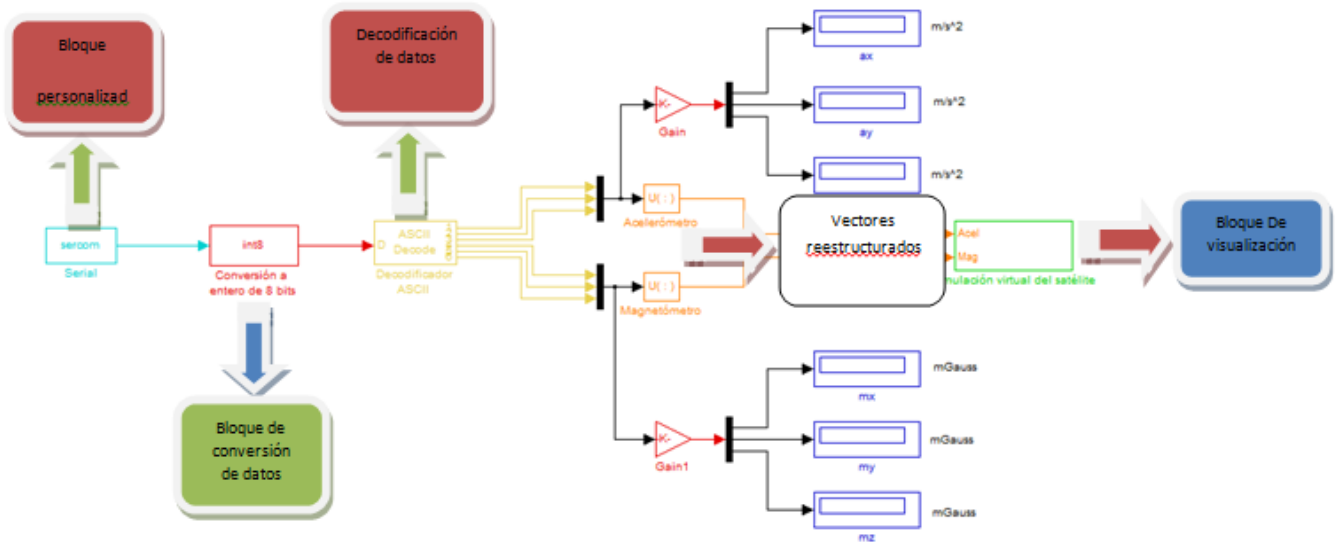


Fig. 7.10. Modelo de seguimiento implantado en Simulink

Luego de algunas pruebas, el resultado obtenido fue exitoso, como se muestra en la figura 7.11.

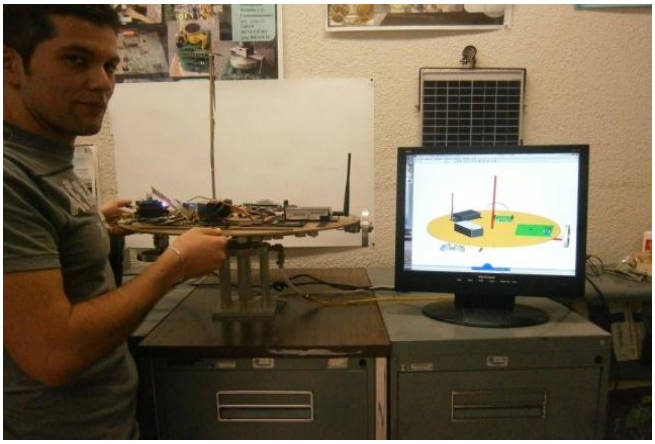


Fig.7.11. Maniobras de seguimiento virtual en tres ejes efectuadas en el laboratorio de sistemas aeroespaciales, del II-UNAM.

El seguimiento virtual de satélites es de gran importancia, dado el hecho que una vez puestos en órbita, es prácticamente imposible atender a cualquier falla o desajuste en ellos. Es primordial efectuar monitoreos constantes tanto externos como internos utilizando complejos sistemas de comunicaciones, para tratar de advertir y evaluar cuando sea posible reparar cualquier problema, o bien, simplemente para estar monitoreando la posición y la trayectoria del objeto, tal y como lo hacen técnicas como la telemetría, en la que, por cierto, se deben destinar periodos de tiempo relativamente largos para el análisis y la toma de decisiones, generando incertidumbre dentro de ese lapso de tiempo. No obstante, a pesar de lograr un exitoso monitoreo, es imprescindible el contar con algoritmos de control de estabilización y de apuntamiento para mantener un régimen de orden, como la mayor parte de los vehículos espaciales, los

cuales tienen instrumentos o antenas que deben apuntar en una dirección concreta, por lo que se vuelve prioritario mantener la orientación de aquél respecto a un punto específico, la cual se puede ver afectada por una serie de pares perturbadores presentes en el espacio, tales como gradiente gravitacional, presión por radiación solar, efectos de flexibilidad, etc. en el sistema. Dentro del GDSA-II-UNAM cuenta ya con algoritmos y pruebas de control efectuadas en un eje sobre el simulador de vuelo satelital y esta tarea se encamina a crear algoritmos de control en tres ejes. Sin embargo, aunque se trate de solo el control en un eje, es una tarea difícil, complicada y tardada, dadas las complicaciones al momento de programar un FPGA; así que se planteó el desarrollo de una plataforma virtual que contenga al modelo que se desea controlar, con el objetivo de implementar algoritmos de control mucho más complejos y validarlos antes de que sean introducidos a los dispositivos lógicos. A continuación se presenta el desarrollo matemático y la implementación del modelo real a Simulink.

7.3. Control de orientación en un eje

La necesidad de integrar nuevos y amplios módulos de control a SATEDU ha impulsado el desarrollo de plataformas alternativas de validación de algoritmos que permitan estabilizar un satélite, como el uso de plataformas virtuales, o bien, el uso de un simulador de vuelo satelital basado en una mesa suspendida en aire, el cual consiste en un disco de 50 cm de diámetro por 1 cm de ancho, una semiesfera que se encuentra en la parte inferior, además de contar con ruedas de reacción capaces de llevar a la mesa a la estabilización y apuntarla hacia una dirección deseada. Entonces, por medio de un flujo de aire inyectado por de un compresor, la mesa literalmente flota en el aire, lo que simula el efecto antigravedad en la medida en que el simulador no posee ningún punto de apoyo. El simulador debe estar perfectamente balanceado, es decir, las masas sobre el deben generar un centro de masa coincidente con el centro geométrico, a fin de no generar desbalanceos indeseados. La hipótesis radica en aplicar una fuerza externa al simulador, provocando un descontrol que deberá ser corregido por las ruedas inerciales controladas por los algoritmos de control instalados en el FPGA.

Ahora bien, con la conceptualización puesta en marcha, se planteó desarrollar la implementación virtual del simulador de vuelo satelital por medio del modelado matemático, a partir de considerar tomando como punto de partida el análisis de la aportación de el momentum angular que cada uno de los elementos que componen a la MSA tendría, considerando el desempeño del sistema como un todo integrado, es decir, plataforma con balero y ruedas inerciales de la mesa suspendida en aire. A continuación se presenta el desarrollo del modelo matemático del simulador de vuelo satelital.

No obstante, antes de comenzar a integrar los distintos elementos que componen al simulador satelital, habría primero que conceptualizar vectorialmente dicho sistema, como lo muestra la siguiente figura:

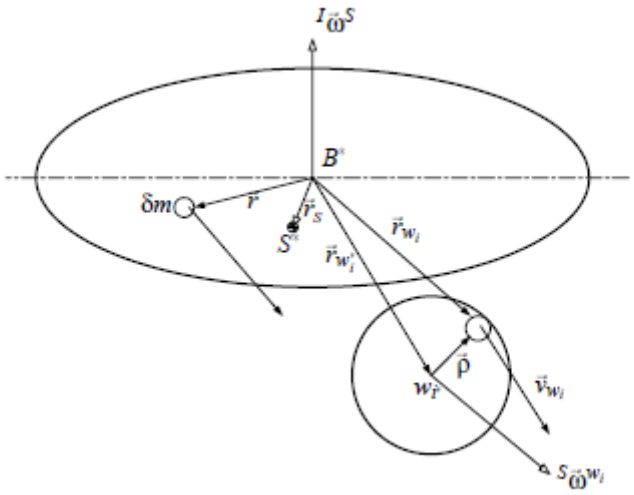


Fig.7.12. Conceptualización del simulador de vuelo satelital

La figura 7.11 que se muestra es interesante, puesto que se modela con la consideración de una rueda inercial, además se toma en cuenta una diferencial de masa dm, la cual debe ser tomada en cuenta para el desarrollo matemático posterior. Nótese que 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.

Partiendo del momento de una diferencial de masa dm, tenemos que su velocidad está dada por:

$$\vec{r} = \vec{\omega}_S^I \times \vec{r}$$

Donde r es el vector de posición desde B*.

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

$$\overrightarrow{r_{wi}} = \vec{\omega}_S^I \times \overrightarrow{r_{wi}} + \vec{\omega}_{wi}^S \times \vec{\rho}$$

Ahora bien, 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, el cual es el siguiente:

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

Nótese que se incluye el símbolo de sigma debido a que para este análisis se tomará en cuenta el uso de tres ruedas inerciales*. 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:

$$\vec{H} = \vec{H}m + \vec{H}r$$

Donde:

- $\vec{H}m$ es el momento angular de la mesa
- $\vec{H}r$ es el momento angular de la rueda

$\vec{H}m$:

$$\vec{H}m = \int_S \vec{r} \times (\vec{\omega}_S^I \times \vec{r})\delta m \dots (a)$$

$\vec{H}r$:

$$\sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times (\vec{\omega}_S^I \times \overrightarrow{r_{wi}} + \vec{\omega}_{wi}^S \times \overrightarrow{\rho})\delta m \dots (b)$$

Desarrollando b:

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

Aquí también podemos simplificar la expresión $\vec{H}r$ analizando cada uno de sus miembros por separado:

$$\vec{H}r = \vec{H}r1 + \vec{H}r2$$

Donde:

$$\vec{H}r1 = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times (\vec{\omega}_S^I \times \overrightarrow{r_{wi}})\delta m \dots (a')$$

$$\vec{H}r2 = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{wi}} \times (\vec{\omega}_{wi}^S \times \vec{\rho})\delta m \dots (b')$$

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:

$$\overrightarrow{r_{wi}} = \overrightarrow{r_{wi}*} + \vec{\rho} \dots (2)$$

Sustituyendo (2) en (b'):

$$\vec{H}r2 = \sum_{i=1}^{3} \int_{wi} (\overrightarrow{r_{w\imath}*} + \vec{\rho}) \times (\vec{\omega}_{wi}^{S} \times \vec{\rho})\delta m$$

Desarrollando:

$$\vec{H}r2 = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{w\imath}*} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho})\delta m + \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho})\delta m \ldots (c')$$

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

$$\vec{H}r2 = \sum_{i=1}^{3} \int_{wi} \overrightarrow{r_{w\imath}*} \times \left(\vec{\omega}_{wi}^{S} \times \int \vec{\rho}\right)\delta m + \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho})\delta m \ldots (3)$$

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

$$\int_{wi} \vec{\rho}\ \delta m = 0$$

Lo que significa que:

$$\vec{H}r2 = \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^{S} \times \vec{\rho})\delta m \ldots (4)$$

Tomando las ecuaciones anteriores (a') y (b'), se observa que ambas integrales son de línea, además de que poseen la propiedad de aditividad con respecto al camino de integración, en este caso delimitado por ω. Así, se tiene que:

$$\vec{H}mt = \vec{H}m + \vec{H}r1$$

Donde

- $\vec{H}mt$ es el momento angular total de la mesa
- $\vec{H}m$ es el momento angular de la mesa sin considerar la rueda inercial
- $\vec{H}r1$ es el momento angular de la rueda que influye en el momento angular de la mesa

Por lo que:

$$\vec{H}mt = \int_S \vec{r} \times (\vec{\omega}_S^I \times \vec{r})\delta m + \int_{wi} \overrightarrow{r_{wi}} \times (\vec{\omega}_S^I \times \overrightarrow{r_{wi}})\delta m$$

Con la propiedad de aditividad, se tiene que:

$$\vec{H}mt = \int_{S+w} \vec{r} \times (\vec{\omega}_S^I \times \vec{r})\delta m \ldots (c)$$

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

$$\vec{H} = \int_{S+w} \vec{r} \times (\vec{\omega}_S^I \times \vec{r})\delta m + \sum_{i=1}^{3} \int_{wi} \vec{\rho} \times (\vec{\omega}_{wi}^S \times \vec{\rho})\delta m \ldots (5)$$

Hasta este paso, se han logrado integrar en una sola ecuación todos los momentos angulares involucrados en la rotación del simulador satelital. Lo siguiente es descomponer las operaciones vectoriales para así obtener un modelo matemático más digerido con posibilidades de reducción. Para ello, es conveniente emplear las propiedades vectoriales, como lo es el siguiente teorema.

Se tiene el siguiente triple producto vectorial:

$$a = \vec{b} \times (\vec{c} \times \vec{b})$$

Al descomponerlo, se produce la siguiente expresión:

$$\vec{b} \times (\vec{c} \times \vec{b}) = \vec{c}(\vec{b} \cdot \vec{b}) - \vec{b}(\vec{b} \cdot \vec{c})$$

Aplicando este teorema a la ecuación (5), queda:

$$\vec{H} = \int_{S+w} \vec{\omega}_S^I (\vec{r} \cdot \vec{r})\delta m + \sum_{i=1}^{3} \vec{\omega}_{wi}^S (\vec{\rho} \cdot \vec{\rho})\delta m \ldots (6)$$

Integrando:

$$\vec{H} = \underline{I}^{S+w/B*}\vec{\omega}_S^I + \underline{I}'^{wi/wi*}\vec{\omega}_{wi}^S \ldots (7)$$

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

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

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:

$$\vec{T} = \overrightarrow{rs} \times mg\vec{k}$$

Donde:

- M es la masa del sistema
- G es la constante de gravitación
- $\overrightarrow{rs}$ 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 (7) de hecho 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.

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

En términos de momento angular:

$$\overrightarrow{Mnet} = \dot{H} + \dot{\omega} \times H$$

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 (7):

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

Al desarrollar (8), se tiene que:

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

Donde:

$$\vec{\omega}_{wi}^I = \vec{\omega}_S^I + \vec{\omega}_{wi}^S$$

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:

$$\vec{\omega}_{wi}^S \times \underline{I^i}\vec{\omega}_{wi}^S = 0$$

Con la deducción anterior en cuenta, y sustituyendo en el teorema de Euler la ecuación 9, se tiene que:

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

La última ecuación representa el comportamiento dinámico del simulador de vuelo satelital, teniendo como variables a las velocidades angulares. Las magnitudes físicas, como las inercias, se calculan a partir de las características mecánicas de los elementos.

7.3.1. Implementación del modelo matemático en SIMULINK

Como ya fue tratado en capítulos anteriores, SIMULINK es una plataforma de simulación avanzada que permite efectuar simulaciones numéricas complejas de sistemas o procesos reales con fines analíticos. En este caso, la integración de la ecuación de comportamiento dinámico (fig.7.13) en función de los momentos angulares de la mesa con dicha plataforma resultó ser complicada, debido al gran número de componentes relacionados

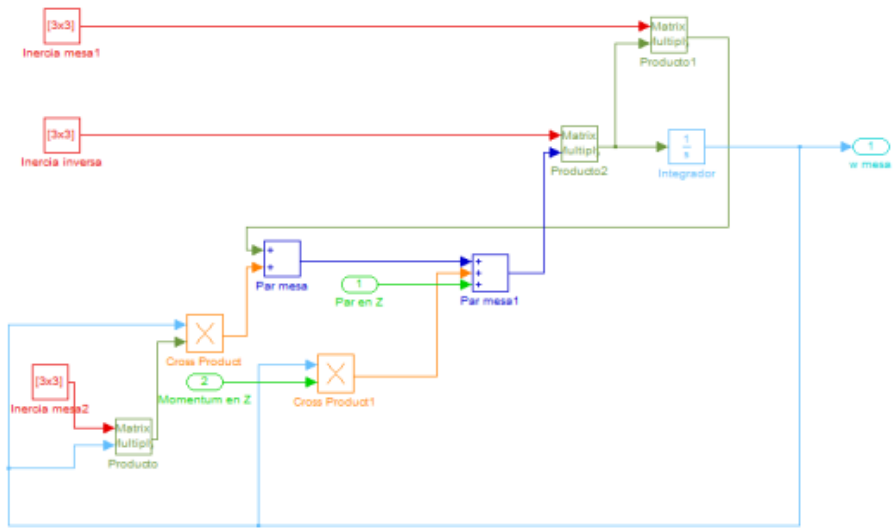


Fig.7.13. Dinámica de la mesa implantada en SIMULINK

El sistema reacciona al aplicarle una perturbación, permitiendo efectuar maniobras de estabilización regidas por el algoritmo implantado. En este caso, un simple controlador PD (fig.7.14) fue suficiente para lograr el objetivo.

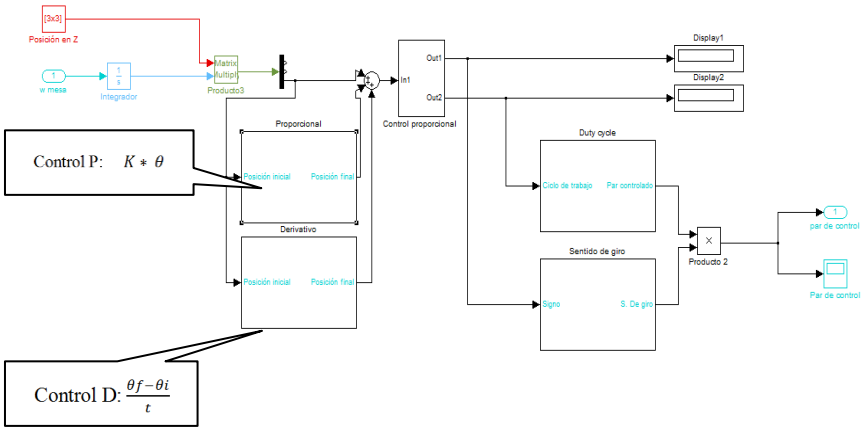


Fig.7.14. Módulo de control implementado en Simulink.

Donde θ es el ángulo obtenido luego de la perturbación, θi es la posicion inicial y θf es la posición final del instrumento.

Estos controladores se operan por separado para después sumarlos, todo ello dentro del bloque *MatLab Function*.

Finalmente, por medio de una interfaz gráfica de usuario de diseño propio, desarrollada también en MatLab, como se muestra en la fig.7.15, donde se explica el funcionamiento de cada una de las opciones que integran a la interfaz gráfica, es posible modificar algunos de los parámetros del controlador, como

por ejemplo cambiar las ganancias mientras que al mismo tiempo se puede observar el comportamiento de la mesa por medio de la simulación virtual, con lo que es factible efectuar análisis profundos y concretos acerca del controlador usado.

Fig.7.15. Interfaz gráfica de usuario

Por el momento y para efectos demostrativos del sistema, actualmente se emplean señales producidas por un tren de pulsos generados por medio de un bloque de instrumentación, propio de una de las librerías de Simulink. En trabajos futuros se espera poder incluir datos reales obtenidos por los sensores de navegación, esto con el fin de convertir al sistema en una herramienta aún más fiable, consistente y una firme alternativa para opción de análisis para el desarrollo de subsistemas satelitales.

La finalidad de mostrar el modelado matemático y el control de posición, radica en la necesidad de efectuar dicho control sobre los tres ejes que conforman al simulador de vuelo satelital, con el objetivo de validar algoritmos que permitan ser empleados posteriormente en satelitales como SATEDU. Una técnica económica, rápida y eficaz de validar tales algoritmos de una forma rápida es aquella que se encuentra basada en simulaciones avanzadas utilizando modelos en realidad virtual para la verificación de resultados, y cuyo preámbulo fue precisamente el seguimiento virtual en tiempo real. Se espera que al corto plazo el grupo de sistemas aeroespaciales, con el aprovechamiento de esta herramienta, logre llevar a cabo validaciones importantes que, a corto plazo aceleren los trabajos de actualización del subsistema de control de orientación satélite educativo SATEDU, y a largo plazo permitan la transferencia hacia una plataforma satelital real.

En el siguiente capítulo se abordarán las conclusiones y recomendaciones obtenidas durante y al final del presente proyecto, resaltando la integración de nuevos elementos y la posible depuración de algunos otros que lo conforman.

Capítulo 8

CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

Del trabajo presentado a lo largo de esta tesis se concluye lo siguiente:

Se desarrolló y validó exitosamente un sistema de seguimiento virtual para el simulador de vuelo satelital, basado en una mesa suspendida en aire del GDSA, II-UNAM. El sistema está integrado por dos segmentos principales: adquisición de datos y procesamiento y visualización virtual. Ambos segmentos integran tanto elementos en hardware como en software, conformando un sistema de seguimiento en tiempo real de altas prestaciones, el cual puede ser alimentado a partir de datos de sensores reales o muestreos previamente almacenados.

El segmento de adquisición de datos lo componen una unidad de mediciones inerciales (IMU), que integra sensores de navegación tales como: acelerómetro, magnetómetro y giróscopo, todos triaxiales. La adquisición de los datos, propiamente dicha se realiza a bordo de una tarjeta de evaluación y desarrollo Spartan 3E, la cual tiene como dispositivo principal de cómputo a un FPGA. En el FPGA (XC3S500E de Xilinx) integra un sistema embebido desarrollado en torno al núcleo de un microprocesador empotrable MicroBlaze de 32 bits, RISC de gama media, así como núcleos periféricos de comunicaciones seriales para interfaz tanto con la tarjeta de sensores (I2C) como con la PC (UART-RS232), que es donde se tiene montado el modelo de simulación y visualización virtual.

Se optó por el uso de un FPGA para el desarrollo de una parte del sistema debido a las características de flexibilidad que brinda un dispositivo de este tipo. Flexibilidad en el sentido de poder integrar de manera sencilla, y para trabajos posteriores, núcleos de cómputo que describan algoritmos de estimación y control, que puedan validarse operativamente ya sea en software o hardware. En software mediante su inclusión directa en el modelo de simulación en la PC, o bien por hardware, integrándose como un núcleo periférico dentro de un sistema embebido en una plataforma FPGA.

El modelo virtual de la plataforma de simulación de vuelo satelital que se integró en el sistema, tiene asociadas características reales del simulador de vuelo real con que se cuenta en el laboratorio del GDSA del II-UNAM, y que está sirviendo como plataforma de desarrollo para la evaluación de sistemas de control de orientación triaxial aplicado a pequeños satélites, en líneas de trabajo actual del grupo usando plataformas de cómputo reconfigurable FPGA y de alto rendimiento DSP. Fue desarrollado en software CAD y tiene la flexibilidad de poder ser actualizado en función de cambios de distribución en los componentes a bordo, o bien cambios más fuertes en su arquitectura general.

El ambiente de desarrollo en la PC, donde se tiene descrito el modelo de seguimiento y visualización virtual fue desarrollado en Simulink. El ambiente de desarrollo Simulink, permitió la integración de un modelo funcional, a partir de la inclusión de bloques predefinidos y algunos otros personalizados, desarrollados enteramente por el autor, para la realización de tareas que van desde la adquisición de datos, el pre-procesamiento de las tramas de datos con información de los sensores de navegación, hasta el acondicionamiento de dichos datos para alimentar al modelo de realidad virtual que describe el simulador de vuelo satelital.

Sin lugar a duda, con el desarrollo del sistema que se describió en esta tesis se advierte un gran avance en cuanto a la integración de herramientas de desarrollo y evaluación con que cuenta el GDSA II-UNAM. El desarrollo de subsistemas satelitales, particularmente en el caso del subsistema de determinación y control de orientación de satélites pequeños en el que desde hace algunos años trabaja el GDSA, conlleva una serie de desarrollos previos antes de su integración final. En el proceso de diseño de este subsistema en tierra, esta herramienta se vuelve de una enorme importancia, ya que permitirá dar información *a priori* del comportamiento de la plataforma de simulación y facilitará la toma de decisiones. Las expectativas que deja al descubierto el sistema de seguimiento virtual con el alcance obtenido en esta tesis, incluyen la integración posterior de más bloques funcionales, que contengan algoritmos de estimación y control, lo cual permitirá contar con una herramienta aún más potente que permita acelerar los procesos de diseño antes de ser implementados de forma definitiva en una plataforma de cómputo detereminada.

Con la conclusión de este trabajo, queda la satisfacción personal del autor de ver un trabajo terminado, que aunque de apariencia modesto por sus alcances iniciales, la infraestructura desarrollada en torno a él y la experiencia ganada en cuanto a los conceptos teóricos asociados, le confieren un alto valor agregado. En México, particularmente en el área de desarrollo de sistemas para satélites y aeronaves, el desarrollo es incipiente. Sin embargo, trabajos como éste son muestra del gran empuje y pasión de la juventud por

posicionar en un futuro no muy lejano a nuestro país más que como un simple consumidor de tecnología, como un semillero de soluciones tecnológicas en áreas de interés estratégico.

8.2. Recomendaciones

Como parte de las recomendaciones que podrían mejorar el potencial de esta herramienta virtual, está la opción, bien de contar con sensores de navegación de una mejor calidad, o bien de implementar un esquema de filtrado que permita el manejo más limpio de las señales provenientes de los sensores actuales, y que pueden provocar en el modelo un comportamiento no deseado, en función de la magnitud del ruido de cada uno de los sensores. Así mismo, aplicar posibles depuraciones al sistema con la finalidad de reducir los tiempos de procesamiento en la PC, los cuales generan "pequeños retrasos" al momento de estar monitoreando ambos objetos, es decir, tanto al simulador de vuelo satelital real como al simulador de vuelo satelital diseñado en CAD.

Como recomendación final, se sugiere la integración de subsistemas capaces aportar algoritmos de control de posición y apuntamiento al modelo virtual, con la finalidad de probar, comparar y validar formas de ejercer un control integral de gran utilidad para sistemas aeroespaciales, particularmente en pequeños satélites.

Bibliografía

1. "*Principles of modelling and simulation. A multidisciplinary Approach"*. **John A. Skolowsky and Catherine M. Banks**. Ed. Wiley, 2009.

2. Paper: *"Web-Based Virtual Reality in Design and Manufacturing Applications".*

3. Paper: *"Building virtual worlds with VRML".* **David R. Nadeau**, San Diego Super Computer Center.

4. Paper: "*Bottom, thou art translated": the making of VRML Dream".* **Stephen N. Matsuba and Bernie Roehl.**

5. Paper: *"Visualizing 3-D geographical data with VRML"*. **Jie Shan**. Depertament of Technology, Unoversity College Gavle.

6. Paper: "*The impact of Web3D technologies on medical education and training."*

7. *"Orbital mechanics for engineering students"*. **Howard D. Curtis**. Ed. Elsevier, 2005.

8. Paper: *"Representing Attitude: Euler Angles, Unit Quaternions and Rotation Vectors".* **James Diebel**, Standford University. 20 October, 2006.

9. Tesis de Mestría: "*S.T. El Moussaoui Brembo. "Sensor Modelling and Attitude Determination for Micro-Satellite."* Norwegian University of Science and Technology, 2005.

10. Tesis de maestría: *"Design and Implementation of attitude determination algoritm for Cubesat UWE-3".* **Sajid Ghuffar**, Lulea University of Technology, 2010.

11. **Jo˜ao Luis Marins, Xiaoping Yun, Eric R. Bachmann, Robert McGhee, Michael J. Zyda**. *"An extended-kalman filter for quaternion-based orientation estimation using marg sensors"*. En Proceedingsof the 2001 IEEE/RSJ, International Conference on Intelligent Robots and Systems, páginas 2003–2011. 2001.

12. **Sass, Ron y Schmidt, Andrew G**. *"Embedded Systems Design with Platform FPGAs Principles and Practices"*. EUA : Morgan Kaufmann Publishers, 2010.

13. **Chu, Pong P**. *"FPGA Prototyping by VHDL Examples".* s.l. : John Wiley & Sons, Inc., 2008.

14. Altera Corporation. FPGAs. [En línea] 2011. *http://www.altera.com/products/fpga.html*.

15. **Charles H. Roth, Jr**. "*Digital Systems Design Using VHDL"*. EUA : PWS Plushing Company, 1998.

16. **Zeidman, Bob**. *"Designing with FPGAs and CPLDs"*. EUA : CMP Books, 2002.

17. Xilinx, Inc. [En línea] 2011. *http://www.xilinx.com/company/gettingstarted/index.htm.*

18**. Hauck, Scott y DeHon, André**. "*Reconfigurable computing: the theory and practice of FPGA-based computation"*. EUA : Morgan Kaufmann Publishers, 2008.

19. **Ashenden, Peter J.** *"The designer's guide to VHDL: Systems on sylicon. EUA"*. Morgan Kaufmann, 2001.

20. **Daniel D. Gajski, Samar Abdi,Andreas Gerstlauer,Gunar Schirner**. "*Embedded System Design: Modeling, Synthesis and Verification"*. EUA : Springer, 2009.

21. Xilinx, Inc. Spartan-6 Family Overview. March 21, 2011. DS160 (v1.7).

22. Altera Devices. [En línea] 2011. *http://www.altera.com/products/devices/dev-index.jsp.*

23. **Cofer, R. C. y Harding, Benjamin F.** "*Rapid system prototyping with FPGAs"*.

24. **Meyer-Baese, Uwe**. "*Digital signal processing with field programmable gate arrays"*.

25. Xilinx, Inc. Embedded Processing. [En línea] 2011. *http://www.xilinx.com/products/technology/embedded-processing/index.htm*. **Rosinger, Hans-Peter.** Connecting Customized IP to the MicroBlaze Soft Proce

26. **Rosinger, Hans-Peter**. "*Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel. s.l*". Xilinx, Inc., May 12, 2004. XAPP529 (v1.3).

27. Xilinx, Inc. LogiCORE IP Processor Local Bus (PLB) v4.6 (v1.05a). September 21, 2010. DS531.

28. "*Embedded System Tools Reference Manual"*. June 23, 2006. UG111 (v6.0).

29. Xilinx Platform Studio (XPS) Overview. [En línea] 2005. *http://www.xilinx.com/itp/xilinx8/help/platform_studio/html/ps_b_gst_overview.htm.*

30. Software Development Kit (SDK). [En línea] 2011. *http://www.xilinx.com/tools/sdk.htm.*

31. **Mohinder, S. G., Lawrence, R.W. y Angus**, "*P.A. Global Positioning Systems, Inertial Navegation and Integration"*. EUA : John Wiley & Sons, 2001.

32. **Titterton, David y Weston, John**. "*Strapdown Inertal Navegation"*. GB : The Institute of Electrical Engineering, 2004.

33. **Banerjee, Paritosh, Somnath,Puri, Amitava,Bose**. "*Modern Inertial Sensors and Systems. s.l."* Prentice Hall, 2008.

34. **Beeby, Stephen**. "*MEMS mechanical sensors"* .

35. **Bao, M.-H**. "*Handbook of sensors and actuators"*. Paises Bajos: Elsevier Science B.V., 2000.

36. **Pisacane, Vincent L**. "*Fundamentals of space systems"*. EUA: Oxford University Press, 2005.

37. Philips Semiconductors. The I2C-BUS Specification Version 2.1. Enero 2000.

38. Xilinx, Inc. XPS IIC Bus Interface (v2.00a). July 22, 2008. DS606 of the 2001 IEEE/RSJ, 2*International Conference on Intelligent Robots and Systems".*

39. Data sheet radio modem*: http://www.digi.com/products/wireless-modems-peripherals/wireless-range-extenders-peripherals/xtend#overview*

40. Diseño de filtros activos: *http://www.allaboutcircuits.com/vol_2/chpt_8/2.html*

41. Documentación de Simulink: *"MatLab S-Function".*

42. Documentación de Simulink. "*Librerías de bloque"*

43. VRML Pad: "*http://www.parallelgraphics.com/products/vrmlpad/"*

44. Documentación de MatLab.

45. Paper*: "A spacecraft simulator for research and education"*. **Byung Moon Kim, Efstathios Valenis, Patrick Kriengsiri and Panagiotis Tsiotras**. Georgia Institute of Technology.