Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Simulación con Hardware en Lazo para un Vehículo Autónomo Submarino

Autor: Abraham Medina Pérez

Tutor: Ing. Rubén Eduardo Carlés Barrero

Santa Clara

2016

"Año 58 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Simulación con Hardware en Lazo para un vehículo autónomo submarino

Autor: Abraham Medina Pérez

abmedina@uclv.cu

Tutor: Ing. Rubén Eduardo Carlés Barrero

Profesor Asistente

Departamento de Automática y Sistemas Computacionales

Facultad de Ingeniería Eléctrica

ruben_eduardo@uclv.edu.cu

Santa Clara

2016

"Año 58 de la Revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Compramos cosas que no necesitamos con dinero que tenemos para impresionar personas que no nos importan.

DEDICATORIA

A mis padres por todo el apoyo que me han brindado durante tantos años

AGRADECIMIENTOS

A mis amigos, que estuvieron conmigo en las buenas o en las malas, a mi familia, a mis profesores.

RESUMEN

Hoy en día los vehículos autónomos subacuáticos tienen una gran vigencia en el mundo actual. Sus usos llegan a ser elevados debido a su gran cantidad de aplicaciones, esta es la razón por la cual sus estudios se han hecho muy populares en la actualidad. En nuestro caso la importancia de estos vehículos es mayor ya que son de gran valor monetario tanto su adquisición como su construcción. Con este trabajo se presenta la conformación de un *software* capaz de realizar la simulación con *hardware* en lazo para el vehículo existente en nuestra universidad, utilizándolo como una alternativa más para asegurar el buen funcionamiento del mismo y evitando costosos errores de software en las pruebas reales

Para la confección del *software* se utilizó el "Matlab" como principal herramienta, una "PC-104" para la navegación mientras que para el supervisor se utilizó una computadora independiente. La descripción de la construcción del mismo, así como todo *hardware* utilizado se enuncian a lo largo del trabajo.

TABLA DE CONTENIDOS

PENSAMIENTO i

DEDICATORIA ii

AGRADECIMIENTOS iii

RESUMEN iv

TABLA DE CONTENIDOS v

ÍNDICE DE TABLAS viii

ÍNDICE DE FIGURAS ix

INTRODUCCIÓN 1

Organización del informe. 4

CAPÍTULO 1. ANTECEDENTES TEÓRICOS DEL DESARROLLO DE LOS AUVs Y DEL CONTROL CON HARDWARE EN LAZO 5

1.1 Estudios de los AUVs. 5

1.2 Metodología. 7

1.3 Sistemas de tiempo real. 8

1.4 Modelo Matemático. 9

1.4.1 Modelo Dinámico Linealizado. 10

1.4.2 Lazos de Control. 11

1.5 Hardware en lazo. 12

1.5.1 Funcionamiento del hardware en lazo. 12

1.5.2 Ejemplos de simulaciones con hardware en lazo. 13

1.6 Conclusiones parciales del capítulo. 15

CAPÍTULO 2. ARQUITECTURA DEL HARDWARE Y EL SOFTWARE 16

2.1 Entorno de ejecución. 16

2.2 Arquitectura de la aplicación. 16
2.3 Descripción del hardware. 17
2.3.1 Sensor inercial. 17
2.3.2 Pc-104. 22
2.4 Descripción del software. 23
2.4.1 Software PC-104. 23
2.4.2 Unidad Ds-PIC. 25
2.4.3 Supervisor. 25
2.5 Modelo Matemático. 26
2.5.1 Lazos de control. 27
2.5.2 Extracción de Datos. 28
2.5.3 Sistema de posicionamiento global. 28
2.6 Tiempos de Comunicación. 29
2.7 Conclusiones parciales del capítulo. 30
CAPÍTULO 3. RESULTADOS Y ANÁLISIS 32
3.1 Confección del software. 32
3.1.1 S-Function. 32
3.1.2 Comunicación. 33
3.1.3 Bloques de comunicación serie de Matlab. 34
3.1.4 Subsistemas. 36
3.2 Simulaciones. 37
3.2.1 Pruebas finales. 38
3.3 Análisis económico. 45
3.4 Conclusiones parciales. 46

CONCLUSIONES Y RECOMENDACIONES 47

Conclusiones. 47

Recomendaciones. 47

REFERENCIAS BIBLIOGRÁFICAS 48

ANEXOS 50

Anexo I Código de Fuente *S-Function* 50

Anexo II Bloques Simulink con subsistemas. 54

ÍNDICE DE TABLAS

Tabla 2.1 Especificación de datos.....20

Tabla 2.2 Tiempos de adquisición y computación. 30

Tabla 3.1 Especificación de comunicaciones. 34

Tabla 3.2 Primeros resultados.....38

Tabla 3.3 Resultados segunda simulación. 45

ÍNDICE DE FIGURAS

Figura 1.1 Metodología a seguir 7

Figura 1.2 Sistemas coordenados y definición de desplazamientos 10

Figura 1.3 Función de transferencia entre el Angulo del timón de profundidad δE y el cabeceo θ 11

Figura 1.4 Función de transferencia entre el Angulo de cabeceo θ y la profundidad z figura 11

Figura 1.5 Función de transferencia entre el ángulo del timón de rumbo δT y el rumbo ψ fue determinada experimentalmente 11

Figura 1.6 Estructura del lazo de control de profundidad 11

Figura 1.7 Estructura del lazo de control de rumbo 12

Figura 1.8 Simulación con hardware en lazo para un pequeño helicóptero 14

Figura 1.9 Simulación con hardware en lazo para una pila de combustible 15

Figura 2.1 Diagrama de funcionamiento 17

Figura 2.2 Sensor inercial 18

Figura 2.3 Campos de cadena 19

Figura 2.4 Estructura del mensaje 20

Figura 2.5 Unidad Pc-104 22

Figura 2.6 Conexión IMU-Pc-104 23

Figura 2.7 Tareas ejecutadas por la Pc-104 24

Figura 2.8 Tareas ejecutadas por Ds-PIC 25

Figura 2.9 Software supervisor.....26

Figura 2.10 Software Matlab modelo matemático.....27

Figura 2.11Ajuste por la tierra y centrado por la tierra 29

Figura 2.12 Cálculo de tiempo de transmisión 30

Figura 3.1 Matlab *S*-Function 32

Figura 3.2 Tareas a realizar por el bloque *S*-Function.....33

Figura 3.3 Configuración del bloque *To Instrument*.....34

Figura 3.4 Configuración del bloque *Query Instrument*.....35

Figura 3.5 Primer subsistema.....36

Figura 3.6 Segundo subsistema 37

Figura 3.7 Bloque de simulación tiempo real 37

Figura 3.8 Tramas enviadas por la IMU y por Matlab 38

Figura 3.9 Supervisor primera simulación.....39

Figura 3.10 Diagrama de simulación.....39

Figura 3.11 Supervisor segunda simulación primer descenso 40

Figura 3.12 Supervisor segunda simulación segundo descenso 40

Figura 3.13 Supervisor segunda simulación retorno a la profundidad cero 41

Figura 3.14 Balanceo en el tiempo segunda simulación hasta los 100 segundos.....41

Figura 3.15 Balanceo en el tiempo segunda simulación hasta los 1200 hasta los 1500 segundos.....42

Figura 3.16 Cabeceo en el tiempo segunda simulación.....43

Figura 3.17 Guiñada en el tiempo segunda simulación 44

Figura 3.18 Profundidad en el tiempo segunda simulación.....44

INTRODUCCIÓN

Como todos sabemos en la industria marítima y su vertiginoso desarrollo, la creación vehículos subacuáticos con el objetivo de aprovechar todos los recursos de las profundidades marinas es vital. Entre estos vehículos en los últimos años una clase ha destacado los AUV (vehículos autónomos sumergibles). Los Vehículos Autónomos Subacuáticos AUV (*Autonomous Underwater Vehicles*) pertenecen a la familia de vehículos no tripulados y se definen como submarinos que portan consigo una fuente de energía y unidades de cómputo, ejecutando *softwares* y soluciones de control que le permiten cumplir misiones sin intervención humana (Fjellstad 1994) (Antonelli, Fossen et al. 2008). Es una estructura robótica con una fuente de energía propia y un sistema de cómputo capaz de procesar datos provenientes de sensores, comunicarse y ejecutar algoritmos de navegación que le permiten cierta autonomía en el cumplimiento de misiones. En la carrera por obtener mayores beneficios de los recursos oceánicos se desarrollan los AUVs como herramientas de alto valor en el reconocimiento de lugares inaccesibles, los avances en estas tecnologías propician la exploración de zonas desconocidas. Sus prestaciones brindan una gran aplicabilidad en entornos marinos y su investigación.

Los océanos representan más del 70% de la superficie terrestre y contienen extensas zonas de elevado interés para la humanidad, ya sea con fines científicos, económicos o de preservación medioambiental. Las principales empresas que impulsan el desarrollo de los AUVs están relacionadas con el auge en la explotación de recursos naturales marinos, como el petróleo, el gas natural y las comunicaciones intercontinentales. El estudio de la diversidad marina y de los ecosistema, el monitoreo, mantenimiento y protección de las comunicaciones entre otros hacen gala de la importancia de las investigaciones realizadas

(Lemus 2011). Por el mercado que generan y las ventajas que ofrecen también se pueden encontrar estos vehículos en escenarios como el desarrollo de investigaciones universitarias, el monitoreo de zonas costeras y el reconocimiento del fondo marino. En el campo militar por otra parte los sitúan en un nivel estratégico, empleándolos en la detección de minas y misiones de reconocimiento

En relación a este tema la experiencia en nuestro país se reduce a los trabajos del Grupo de Automatización Robótica y Percepción (GARP) de la Universidad Central de Las Villas (UCLV) en colaboración con el Centro de Investigación y Desarrollo Naval (CIDNAV) con el objetivo de desarrollar un sistema de supervisión y control para un prototipo que lleva por nombre HRC-AUV. Este proyecto está basado en *hardware* y sensores de bajo costo para que sea viable su producción y explotación. El costo en el mercado internacional de estos productos asciende a cientos de miles de dólares más los gastos en piezas de repuesto y asesoramiento técnico, precios inaccesibles para cualquier empresa nacional. Por esta causa es tan importante el progreso de la actividad colaborativa entre estas dos instituciones, representando una oportunidad para nuestro país de disponer de estos vehículos, además de la adquisición de sólidos conocimiento en el área y el crecimiento del nivel docente en nuestra universidad (Lemus 2011).

No es de extrañar que al ser la confección de un AUV tan costosa, sea necesario antes de ser puesto en funcionamiento definitivamente, la simulación de su sistema de cómputo, así como toda su lógica interna. La Simulación con *hardware* en Lazo (HIL) es un método utilizado para garantizar el éxito de la confección del software, tanto como el funcionamiento del mismo. Este trabajo precisa la confección de una Simulación con *hardware* en Lazo con el objetivo de ser implementada en el vehículo AUV existente en el nuestro laboratorio, así como su desarrollo.

Justificación.

En el desarrollo de los AUV pueden presentarse dificultades en la realización de pruebas sobre la planta en la que son implementados, debido a factores como la disponibilidad cuando se trata de procesos de trabajo ininterrumpido o cuando la planta tiene un costo elevado y existen riesgos de daño si el controlador falla o no trabaja adecuadamente. Es posible orientar una metodología hacia el desarrollo de la simulación de los AUVs

basándose en la implementación del *hardware* en interacción con modelos físicos simulados en tiempo real que permitan dar una idea aproximada al comportamiento de un sistema. Este proceso requiere el modelado o la identificación de la planta.

Entorno del trabajo.

"MATLAB-Simulink" es un entorno de "Mathworks" que permite para dar soporte a los Sistemas de Control tanto en el ámbito investigativo como el educativo; esto es debido a que posee múltiples herramientas que facilitan el procesamiento numérico y el trabajo con diagrama en bloques. En la carrera de Ingeniería en Automática de la UCLV se usa "MATLAB" desde el año 1985. El mismo se utilizará con el objetivo de realizar una Simulación con *hardware* en Lazo (HIL) para el vehículo HRC-AUV. Una vez validado el sistema de control en simulación, en el entorno "MATLAB-Simulink" también ofrece la posibilidad de general el código necesario para la ejecución del controlador en el *hardware* embebido que controlará el sistema real, siendo este código compatible con requerimientos de funcionamiento en tiempo real. Por otra parte se realiza el mismo procedimiento para obtener el código correspondiente al modelo de la planta. De esta forma se pueden llevar a cabo simulaciones con el *hardware* embebido real tratando de controlar al modelo del sistema. Una razón muy importante para hacer esto es que en muchas ocasiones la planta puede tener partes donde la seguridad es crítica o el *hardware* sea muy costoso. En estos casos es conveniente probar completamente el *software* y la electrónica del sistema de control antes de implementarlo en la planta real.

Situación del problema.

Sin duda alguna la posibilidad de tener un AUV es muy ventajosa para cualquier institución científica, su construcción, obtención o desarrollo puede ser costoso. Nuestra institución no cuenta con una vía, además de las pruebas reales para evitar posibles fallas o realizar alguna modificación asegurar su funcionamiento.

Objetivo general.

- Desarrollo de un *software* capaz de lograr una simulación con *hardware* en lazo para el vehículo HRC-AUV.

Objetivos específicos.

- Elegir el entorno más adecuado para la simulación de *hardware* que se ajuste a las características del AUV.
- Obtención de todos los datos necesarios para el correcto funcionamiento del *software*.
- Caracterizar los dispositivos empleados para la simulación.
- Utilizar un modelo que sea capaz de reproducir las características reales de la planta.
- Crear los métodos para la correcta comunicación entre los *hardwares* utilizados para la simulación.
- Validar el sistema obtenido mediante comparaciones entre los datos de la simulación del HIL y el software del supervisor.

Organización del informe.

El trabajo de diploma se estructura en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

CAPÍTULO I: En el primer capítulo se presentan antecedentes teóricos acerca de las investigaciones realizadas acerca de los AUVs, así como la trayectoria del control con *hardware* en lazo en el mundo actual. Se realiza una revisión de la literatura especializada para determinar las principales tendencias en la simulación con *hardware* en lazo. Se caracteriza la versión en cuestión del programa para identificar posibles mejoras y funcionalidades que tributen al cumplimiento de los objetivos planteados.

CAPÍTULO II: Se seleccionan y describen las herramientas y métodos necesarios para llevar a cabo el desarrollo del *software*, teniendo en cuenta las particularidades y trabajos anteriores.

CAPÍTULO III: Se describen los resultados del trabajo mediante pruebas experimentales sobre el producto y el análisis de los datos recopilados.

CAPÍTULO 1. ANTECEDENTES TEÓRICOS DEL DESARROLLO DE LOS AUVs Y DEL CONTROL CON HARDWARE EN LAZO

En este capítulo se realiza un análisis de puntos importantes del desarrollo de vehículos submarinos autónomos y del control con *hardware* en lazo, a partir de la investigación de trabajos realizados con anterioridad y de otras bibliografías enmarcaremos el planteamiento de la hipótesis de este trabajo.

1.1 Estudios de los AUVs.

Los AUVs ha demostrado tener un gran impacto debido a su gran variedad de aplicación, los mismos se han ido esparciendo por todo el planeta. Estos se han desarrollado en gran medida debido a que las principales fuentes de recursos naturales en tierra firme han comenzado a escasear, mientras los ríos y mares permanecen pobremente explotados, como era de esperarse muchas universidades del mundo demostraron su interés en los mismos y así surgieron un sin número de investigaciones y proyectos acerca de los mismos, se han publicado gran cantidad de artículos de alto grado científico entre los que se encuentran: "Universidad Nacional del Sur" (UNS) en Argentina (Jordan, Lemus et al. 2008); "Universidad de Oporto", Portugal (Patricia and Mario 2008), "Universidad BEIHANG", China (Liang, Wei et al. 2008), "Universidad de Zagreb" en Croacia (Nikola, Vukic et al. 2008) y "Universidad de Newcastle" en Australia. Por otra parte, se suman los estudios realizados por instituciones como el "Instituto de Problemas Tecnológicos Marinos de la Academia Rusa de Ciencias del Lejano Oriente" (IMTP FEB RAS) con trabajos muy destacados y de la "Agencia de ciencia y tecnología de tierra y mar" en Japón con todas estas entidades han logrado mantener un desarrollo positivo de la tecnología de los

vehículos sumergibles, creando de esta forma las condiciones científico-técnicas para aprovechar los recursos subacuáticos (Lemus 2011). Un papel destacadísimo dentro de este tipo de investigación lo tiene la labor de la "Universidad Noruega de Ciencia y Tecnología" (NTNU) con aportes muy importantes reflejados en libros y artículos de gran calidad (Fossen, Johansen et al. 2008). También se plantea que en el "UUV Master Plan" de La Marina norteamericana (2004), se puede notar una especial atención en el desarrollo de las capacidades de los UUV, así como en la investigación de sus usos potenciales.

En el estudio de la literatura especializada se encontraron publicaciones de varias universidades y centros de investigación en áreas de la navegación, comunicación, control, *software* y componentes, principalmente para reducir los costos tanto como sea posible. Entre estos grupos de investigación se encuentran: "*The Korea Institute of Science and Technology*" (KIOST por sus siglas en inglés), Corea del Sur; "Universidad de Oporto", Portugal; "Universidad de Girona", España; "Universidad Noruega de Ciencia y Tecnología" (NTNU); "Universidad Nacional de Singapur", República de Singapur; "*Wood Hole Oceanographic Institution*", Estados Unidos y "*Naval Postgradúate School*", Estados Unidos. Además, las competencias entre AUVs incrementan la popularidad de los mismos e incentiva el progreso y perfeccionamiento de estos, como ejemplo se incluyen la "*Student Autonomous Underwater Vehicle Challenge-Europe*" (SAUC-E) y la "AUVSI *Robosub Competition*" en Australia (Lemus 2011).

En un principio debido al gran peso de los sistemas de cómputo los AUV debían tener un tamaño y un peso limitado, además eran grandes consumidores de energía. Hacia los años 80 gracias al desarrollo de la tecnología, estos se convierten en unidades capaces de llevar implementados sistemas de navegación y control además de ser capaces de manejar datos de sensores de forma autónoma. Para los 90 los prototipos de sistema operacionales evolucionan a prototipos capaces de cumplir objetivos bien definidos, convirtiéndose para el 2000 en parte del mercado (Lemus 2011).

Cuba por su parte no es muy desarrollada en este ámbito, aunque tenemos pocas publicaciones, se ha demostrado un gran interés en los mismos, además existen instituciones como el Centro de Investigación y desarrollo Naval (CIDNAV), que tienen la

idea de tener nuestro propio desarrollo nacional en estos temas debido a que estos equipos son muy útiles para nosotros por nuestra posición geográfica.

De la importancia que tiene para nosotros este tipo de vehículos el grupo de Automática Robótica y Percepción (GARP) de la Universidad "Marta Abreu" de las Villas comienza a realizar estudios sobre los mismos. De esta forma surge la cooperación para crear el autopiloto de un submarino perteneciente al CIDNAV. El grupo GARP ostenta varias publicaciones y estudios de gran relevancia entorno al trabajo en el diseño y perfeccionamiento del sistema de supervisión y control del HRC-AUV. Entre estas se encuentran trabajos sobre modelado y control del vehículo, diseño e implementación de mejoras del sistema de control, arquitectura de *software* y *hardware*.

1.2 Metodología.

La metodología planteada para este trabajo está basada en la simulación HIL la cual comenzaría con el uso del modelo matemático de la planta hasta la validación y pruebas finales con el estudio de los resultados obtenidos. En la figura 1.1 se muestran las fases a seguir:

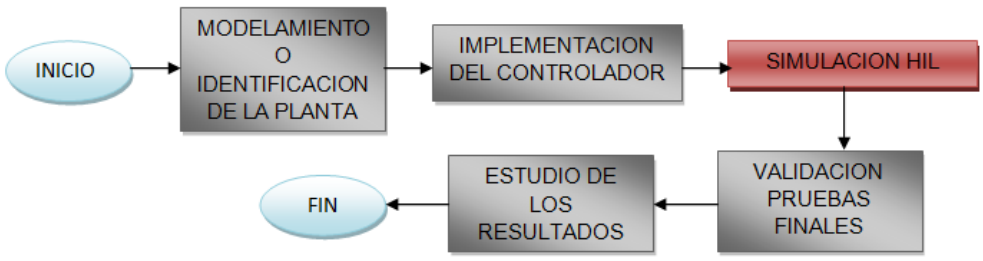


Figura 1.1 Metodología a seguir

La primera fase es la construcción del modelo de la planta, que puede ser de forma analítica o experimental, para el caso del modelado analítico se plantea el funcionamiento del sistema dinámico mediante la relación de ecuaciones diferenciales que describen el comportamiento de una variable de salida ante una entrada determinada todo esto a partir de datos experimentales. Este trabajo fue realizado con anterioridad en la Universidad

Central “Marta Abreu” Central de Las Villas (UCLV) por el Grupo de Automatización Robótica y Percepción (GARP) (Valeriano 2013). En un segundo momento se utilizará el controlador el cual consiste en la PC-104 perteneciente al departamento de automática, se creara el *software* en “Matlab” para poder realizar la simulación con *hardware* en lazo, y se procederá a la fase experimental donde se obtendrán resultados para su posterior estudio.

1.3 Sistemas de tiempo real.

Un Sistema de Tiempo Real es aquel cuyo desempeño no depende solo de los resultados de sus cálculos y procedimientos computacionales, sino que además depende del instante de tiempo en el que estos se producen (Kopets 2002).

Para su estudio, se descompone en tres subsistemas:

- **Subsistema controlado:** Está conformado por el objeto de control (planta, sensores y actuadores).
- **Subsistema computacional:** Está conformado por el sistema computacional de tiempo real.
- **Subsistema operador:** Está conformado por el operador humano.

Atendiendo a características específicas los Sistemas de Tiempo Real se pueden clasificar en: Tiempo Real Fuerte (HRT por sus siglas en inglés) y Tiempo Real Suave (SRT por sus siglas en inglés) (Kopets 2002). El diseño de sistemas de Tiempo Real Fuerte es diferente de los sistemas de Tiempo Real Suave por los requisitos y tolerancias inherentes a ambos.

Tiempos de respuesta: Las demandas de tiempo de respuesta en los sistemas HRT, son generalmente del orden de los milisegundos o menos, debe ser respetada en todo momento y requiere autonomía de la intervención de operadores tanto en operación normal como en situaciones críticas. Los sistemas con requerimientos de SRT, por el contrario, presentan tiempos de respuesta en el orden de los segundos, y si se incumple un plazo de entrega, los resultados no son graves.

Desempeño ante picos de carga: En los sistemas HRT, el escenario de los picos de carga debe estar bien definido, se debe garantizar que la arquitectura computacional del sistema permita el cumplimiento de los plazos de entrega a pesar de los picos. En los sistemas SRT,

el desempeño medio del sistema es lo realmente importante, y la degradación de las condiciones de operación a razón de un pico de carga que ocurre de forma aislada, es tolerada por razones económicas.

Sincronismo: El sistema de HRT, debe mantenerse sincronizado con su medio ambiente (objeto de control y operador), ante todos los posibles escenarios. Los SRT, en cambio, pueden ejercer algún tipo de control sobre el medio (por ejemplo, modificar la velocidad de respuesta) en caso que no se pueda procesar la carga de datos requerida.

Determinismo Temporal: Es fundamental que el comportamiento temporal de los sistemas de tiempo real sea determinista. Esto no significa que sea eficiente, sino que el sistema debe responder correctamente en todas las situaciones, previendo el comportamiento en el peor caso posible.

Fiabilidad y seguridad: Un fallo en un sistema de control puede hacer que el sistema controlado se comporte de forma peligrosa o antieconómica. Es importante asegurar que si el sistema de control falla lo haga de forma que quede en un estado seguro, es decir, teniendo en cuenta los posibles fallos o excepciones del diseño.

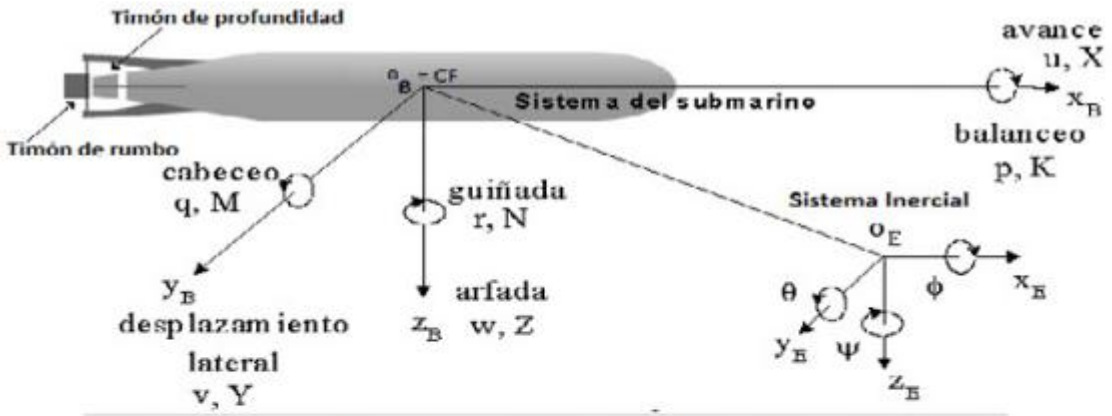
Concurrencia: Los componentes del sistema controlado funcionan simultáneamente. El sistema de control debe atenderlo y generar las acciones de control simultáneamente. En este caso existen dos opciones: sistemas monoprocesador con procesos multi-hilos o computadoras multiprocesador.

1.4 Modelo Matemático.

El modelo matemático del HRC-AUV fue realizado con anterioridad por el GARP. Para la modelación del AUV se escogió un método basado en análisis geométrico, este consiste principalmente en encontrar parámetros que han sido bien definidos a través de leyes físicas, las cuales describen el movimiento de un cuerpo rígido en un ambiente líquido y la realización de una serie de pruebas experimentales de menor complejidad, mediante las cuales se obtienen modelos aproximados que describen la dinámica del cuerpo en determinadas condiciones operacionales. Este último método es ampliamente descrito en las publicaciones del profesor Thor I. Fossen y se ha aplicado con éxito al modelado de varios vehículos sumergibles (Fossen 1994) (Fossen and Ross 2006) (Fossen, Johansen et

al. 2008) por estas razones ha sido seleccionado como método básico de modelado matemático para el HRC-AUV (Valeriano 2013)

Posee seis grados de libertad (6GDL): tres coordenadas para movimientos de desplazamiento y tres coordenadas para movimientos de rotación las cuales presentan una relación no lineal y relación dinámica acoplada donde el movimiento se describe con respecto a un sistema inercial. En la figura 1.2 se representan los sistemas de coordenadas y la definición de movimientos de traslación y rotación del vehículo.



Traslación	Fuerza	Velocidad lineal	Posición
Avance	X	u	x
Lateral	Y	v	y
Arfada	Z	w	z
Rotación	**Momento**	**Velocidad angular**	**Ángulo**
Balanceo	K	p	ϕ
Cabeceo	M	q	θ
Guiñada	N	r	ψ

Figura 1.2 Sistemas coordenados y definición de desplazamientos

1.4.1 Modelo Dinámico Linealizado.

Como es usual en los AUVs tipo crucero se dividen las ecuaciones de movimiento en seis grados de libertad en dos subsistemas desacoplados o pobremente acoplados con tres grados de libertad cada uno.

En (Valeriano 2013) los investigadores de GARP plantean el modelo dinámico acoplado del HRC-AUV, así como las simplificaciones y condiciones necesarias para arribar a las

funciones de transferencia necesarias para plantear los lazos de control de rumbo, cabeceo y profundidad del vehículo. Las cuales se muestran en las figuras 1.3, 1.4 y 1.5.

$$\frac{\theta}{\delta_E} = \frac{b_5}{(I_{yy} - M_q)s^2 - M_q s + WBG_z}$$

Figura 1.3 Función de transferencia entre el Angulo del timón de profundidad δE y el cabeceo θ

$$\frac{z(s)}{\theta(s)} = \frac{-u_0}{s}$$

Figura 1.4 Función de transferencia entre el Angulo de cabeceo θ y la profundidad z figura

$$\frac{\psi(s)}{\delta_T(s)} = \frac{0.14}{4s^2 + s}$$

Figura 1.5 Función de transferencia entre el ángulo del timón de rumbo δT y el rumbo ψ fue determinada experimentalmente

1.4.2 Lazos de Control.

De los experimentos descritos (Valeriano 2013) fue posible obtener los términos necesarios para sustituir en las ecuaciones anteriormente expuestas dando como resultados los lazos de control de profundidad figura 1.6 y rumbo figura 1.7

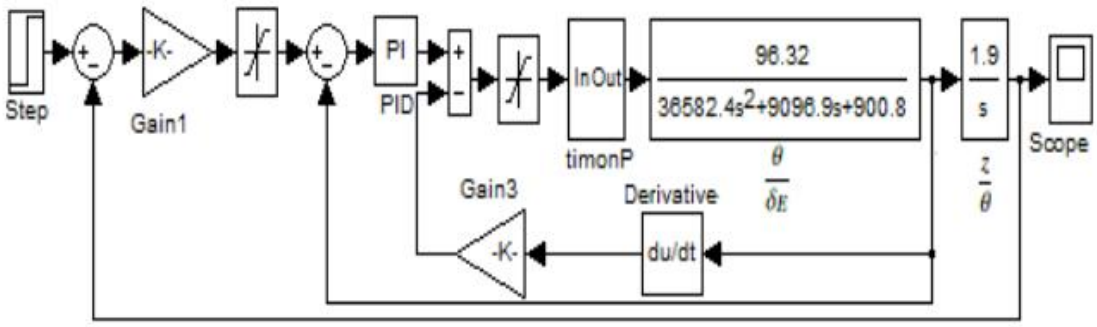


Figura 1.6 Estructura del lazo de control de profundidad

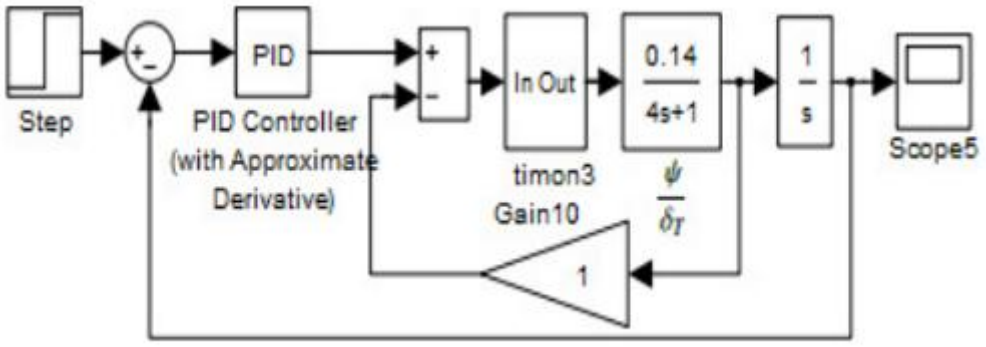


Figura 1.7 Estructura del lazo de control de rumbo

1.5 Hardware en lazo.

En sistemas de control son frecuentes las simulaciones del controlador y la planta para verificar la respuesta del sistema y comparar parámetros deseados, en la simulación HIL la planta es simulada y el controlador es real con el objetivo de disminuir los riesgos de avería en el proceso de prueba y el controlador sobre la planta real, además permite el desarrollo y la evaluación de nuevos algoritmos de control de una manera rápida, incluyendo trabajo con sensores reales o simulados, siendo una ventaja la implementación del controlador sobre una plataforma reconfigurable obteniendo un sistema de prueba económico y con una disminución en los tiempos de desarrollo. Esta metodología tiene grandes ventajas en situaciones en las que no es posible o no es factible parar un proceso de producción para realizar pruebas de control sobre la planta o simplemente cuando se tiene una disponibilidad limitada de la misma por cuestiones de distancia, tiempo u otros motivos. Esta metodología permite detectar errores en proyectos o problemas de diseño o una mala implementación, muy adecuada para desarrollar y probar *hardware* nuevos y sistemas reales electrónicos, mecánicos y mecatrónicos antes de la fabricación real de las máquinas.

1.5.1 Funcionamiento del hardware en lazo.

Funciona haciendo creer que está interactuando con la planta existente cuando en realidad se trata de un modelo simulado en tiempo real. Para poder validar esta metodología se emplean dos prototipos de planta. La idea es obtener características de funcionamiento

semejantes a las que se obtienen con el sistema físico real con el mismo controlador, lo cual es la parte final de la metodología al permitir la validación del controlador verificando la respuesta deseada del sistema. La obtención de resultados de alta fidelidad depende de las condiciones bajo las cuales se caracteriza el sistema físico que pretende simular y la identificación del mismo (Hans 2011).

Para realizar la simulación con *hardware* en lazo se necesita seguir los pasos a continuación.

- El desarrollo de un modelo matemático. Desarrollar un modelo matemático que tenga en cuenta el ambiente real en donde se piensa usar el dispositivo.
- Simulación con *hardware* en lazo (*hardware + software*). Ejecución de pruebas con el dispositivo en un proceso simulado (modelo matemático).
- Implementación del *hardware* en un proceso real. Obteniendo resultados positivos se procederá con las pruebas del *hardware* en el ambiente real donde se planea ser usado.

Lo usual es que la simulación con *hardware* en lazo se encuentre integrada en un ciclo de pruebas, el mismo provee un alto grado de complejidad en la planta bajo control, a medida que aumenta el nivel de dificultad del *hardware* a controlar, aumentara el nivel de complejidad del sistema diseñado para controlarlo (Hans 2011).

1.5.2 Ejemplos de simulaciones con hardware en lazo.

A lo largo de todo el mundo científico hay varias investigaciones utilizando simulaciones con *hardware* en lazo podemos citar varios ejemplos como el caso de Zahari Taha, Abdelhakim Deboucha, Azeddein Kinsheel y Raja Ariffin Bin Raja Ghazilla con el Desarrollo de un *hardware* en Lazo con Tiempo Real para un pequeño Helicóptero a Escala basado en MPC (Zahari, Abdelhakim et al. 2012).

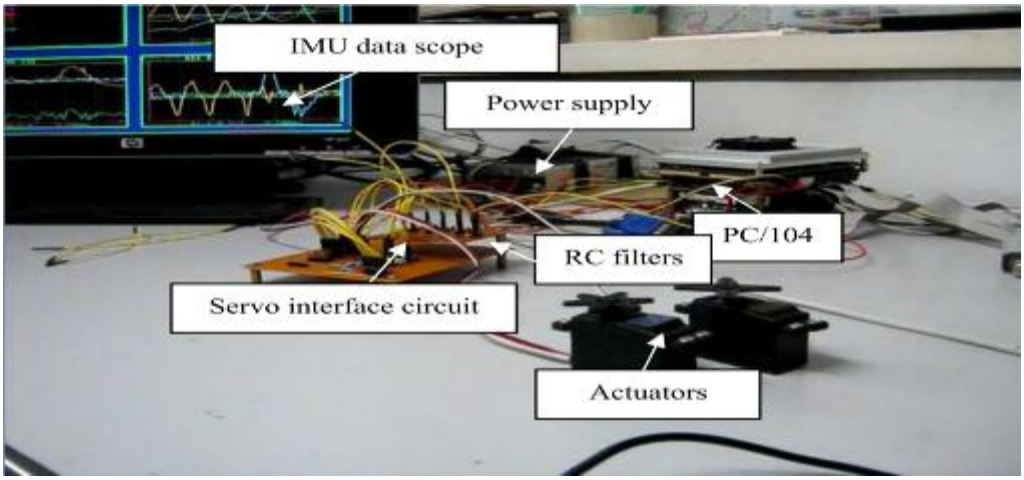


Figura 1.8 Simulación con hardware en lazo para un pequeño helicóptero

Por otro lado encontramos trabajos como el de Marcel Jacomet, Josef Goette, Markus Hager, William Chigutsa y Nicolas Leuba desarrollando la simulación con *hardware* en lazo utilizando "Matlab - Simulink" para una plataforma rápida de diseño con un HS/SW, (Jacomet, Goette et al. 2012). Otras investigaciones como la de F. Casellas, J. Esteban, F. Guinjoan, R. Piqué, H. Martínez y G. Velasco para "Simulación Mediante Hardware In the Loop de un Convertidor Buck" (Casellas, Esteban et al. 2014) y la tesis de maestría "Modelado y simulación Hardware in the loop de un sistema pila de combustible batería" de Lucía Gauchía Babé (Gauchía Babe 2008) forman parte de los tantos trabajos realizados. En el caso de este último trabajo se realizó el modelado de una pila de combustible de baja temperatura y el de una batería de plomo ácido, el modelado fue eléctrico basado en medidas experimentales. Además del modelado y simulación de la pila de combustible y la batería, se construyó una plataforma de ensayos de sistemas de energía siguiendo la metodología para efectuar la simulación con *hardware* en lazo como muestra la figura 1.9.

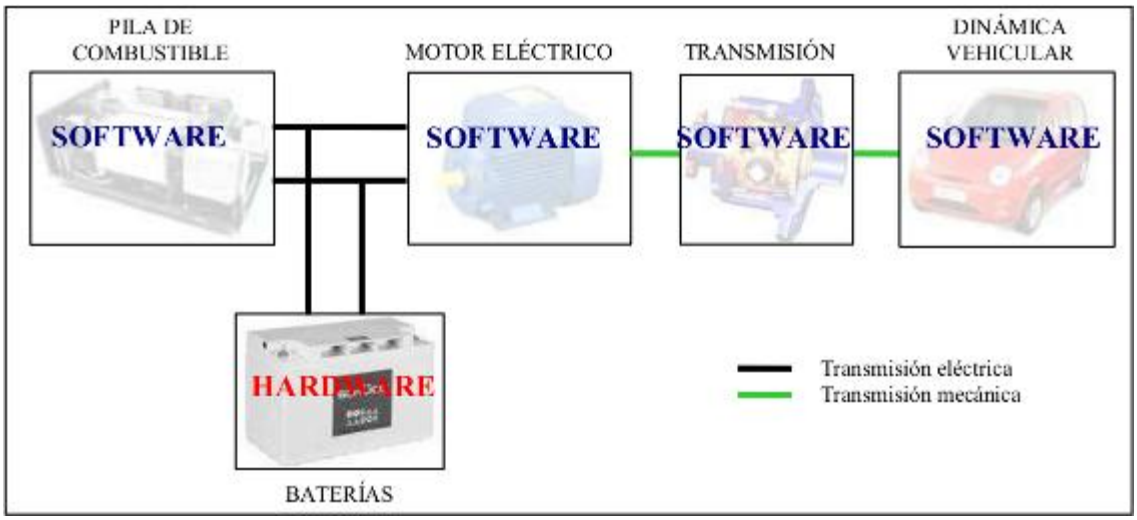


Figura 1.9 Simulación con hardware en lazo para una pila de combustible

Todos demuestran la gran cantidad de aplicaciones que pueden tener este tipo de metodología y el gran auge que ha alcanzado en el mundo científico.

1.6 Conclusiones parciales del capítulo.

Con este primer capítulo hemos hecho un análisis de parte de la bibliografía existente en el mundo de los vehículos autónomos sumergibles, así como de todas las tecnologías aplicadas en los mismos.

Hemos estudiado los diferentes tipos de simulación y en nuestro caso la simulación con h*ardware* en lazo gracias al modelado del matemático de un vehículo autónomo, realizado en trabajos anteriores del grupo GARP (Valeriano 2013) se utilizará para el vehículo HRC-AUV existente en nuestro centro.

CAPÍTULO 2. ARQUITECTURA DEL HARDWARE Y EL SOFTWARE

En este capítulo se realizará una descripción del *hardware* utilizado, así como también del *software* creado para implementar la simulación con *hardware* en lazo para el HRC-AUV, se explicarán los problemas existentes y las herramientas utilizadas en el proceso a lo largo de toda la construcción del *software* así como las soluciones sugeridas a los problemas planteados.

2.1 Entorno de ejecución.

El medio donde se desarrolla el *hardware* así como el *software* de la aplicación diseñada es el entorno de ejecución, el mismo debe mostrar las facilidades que se le brindan al desarrollador de la aplicación. En nuestro caso analizaremos la confección del *software* y una porción del *hardware* que consideramos vital para la fundamentación del proyecto, ya que la mayor parte del mismo se encuentra descrito en otras documentaciones (Xsens 2010), (Nokland 2011).

2.2 Arquitectura de la aplicación.

Como se muestra en la figura 2.1 para realizar nuestra simulación con *hardware* en lazo, seleccionamos la siguiente variante. El s*oftware* se encontrará ejecutándose mientras envía los datos constantemente a la PC-104, la cual está controlada por el supervisor que ejecutará nuevas órdenes que serán transmitidas para retroalimentar el s*oftware* y cerrar el ciclo, a lo largo del capítulo explicaremos más detalladamente todo el procedimiento.

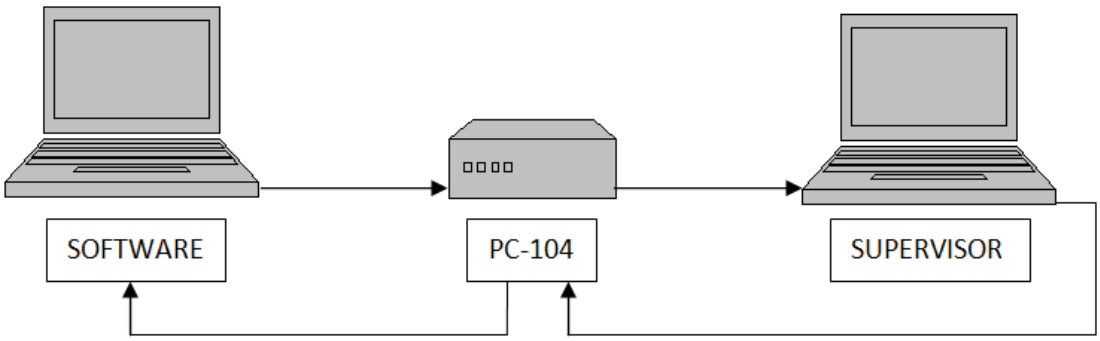


Figura 2.1 Diagrama de funcionamiento

2.3 Descripción del hardware.

A continuación, se describirán los elementos utilizados para la construcción ciclo que presentará la simulación con hardware en lazo que consisten en el sensor inercial y la Pc-104.

2.3.1 Sensor inercial.

La unidad de medidas inerciales más conocida en inglés como *inertial measurement unit* (IMU) es un GPS (Global Positioning System) integrado en un procesador con un sistema de navegación, altitud y cabeceo de referencia. También contiene un magnetómetro 3D, un sensor de presión estática. La señal de bajo nivel del procesador corre un filtro de Kalman de la Xsens estimando posiciones y velocidades 3D inerciales, además de proveer datos de orientación, aceleración calibrada, rango de giro, campo magnético terrestre. Su campo de acción se extiende desde la robótica pasando por la industria aeroespacial, la marina y en nuestro caso en los vehículos autónomos.

Figura 2.2 Sensor inercial

La IMU presenta varias formas de comunicación con la plataforma de "Windows" entre las cuales se encuentran:

- Interface a través de COM (Puerto de Comunicación Serie) objeto API (Apication Programing Interface).
- Interface a través de DLL (Dynamic Link Library) API.
- Comunicación directa con el MTi-G de bajo nivel.

En nuestro caso utilizaremos la primera opción, debido a que al confeccionar el *software* en "MATLAB" podemos utilizar el objeto COM que es una operación registrada en "Windows", entonces si la misma está bien instalada nos permite acceder a sus funciones en todas las aplicaciones de Windows. Por otro lado esta opción permite acceder directamente a las capacidades de la MTi-G desde *software* como "MATLAB", "LabVIEW", "Excel", "Visual Basic", etc. Aunque por estas razones es la vía que utilizaremos es importante resaltar que mediante la misma se puede obtener con facilidad un tiempo real aproximado (*soft real time*), pero no un tiempo real fuerte, para la obtención de este último habría que utilizar la comunicación directa con el MTi-G de bajo nivel (la tercera opción) (Xsens 2010).

Limitaciones.

El filtro de Kalman de 6 grados de libertad de la Xsens en la MTi-G es un filtro casual o sea funciona en tiempo real, solo la información actual y la de un momento pasado es

utilizada para obtener las posiciones y la orientación, esto significa básicamente que después de su encendido se necesita un poco de tiempo, alrededor de 30 segundos para que el mismo se estabilice. El iniciado mínimo es de 1 minuto y continua mejorando su rendimiento durante los próximos 15 minutos, el cual es el tiempo que en realidad se necesita para lograr establecer una buena posición. Adicionalmente el MTi-G usa el GPS y el barómetro para sus cálculos así que en situaciones de cambio de clima o de nivel del mar la medición puede verse comprometida(Xsens 2010).

Salida.

Cuando la MTi-G inicia el proceso de comunicación se forma un mensaje (cadena digital), la cual contiene una estructura estándar. El contenido del mensaje va desde el cero hasta los 254 bytes, mientras que el tamaño total de la cadena es desde cinco hasta 259 bytes, la imagen siguiente muestra los campos que normalmente contiene una cadena:

Figura 2.3 Campos de cadena

Preámbulo (PRE).

Todos los mensajes comienzan con un preámbulo que siempre contiene el valor de 250 (=0xFA).

Bus Identificador o Dirección (BID).

Todos los mensajes utilizados por la MT utilizan el valor de dirección de 255 (=0xFF).

Identificador de Mensaje (MID).

Este campo especifica el tipo de mensaje que se está enviando.

Tamaño (LEN).

Especifica la cantidad de bytes que se encuentran en el campo de Datos. El valor de 255 (=0xFF) está reservado lo que implica que el máximo de bytes es 254. Si la cantidad de bytes es cero significa que no hay campo de Datos.

Datos (DATA).

Este campo contiene todo lo referente a los datos y tiene una longitud variable la cual especificada en el campo LEN. La interpretación de los datos del mensaje es específica y depende del valor del campo de MID, si el valor del MID cambia así también cambiara la interpretación.

En el caso de este campo es el necesitado para enviarlo a la PC-104 la cual se encarga de utilizar todos los datos recibidos para realizar la navegación y efectuar nuevos eventos, que son enviados al *software* en una PC destinada para el mismo. La estructura de este mensaje se muestra a continuación (Xsens 2010):

Para esta cadena que contiene el sensor de presión y el GPS serían los siguientes parámetros:

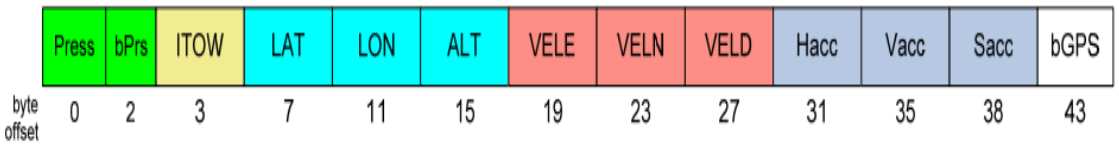


Figura 2.4 Estructura del mensaje

La siguiente tabla muestra algunos datos como el nombre la posición sus unidades de medida y el significado de los bytes en la trama digital anterior.

Tabla 2.1 Especificación de datos.

NOMBRE	POSICION	UNIDADES DE MEDIDA	COMENTARIOS
Press	0	Pa	Presión en pascales
bPrs	2	-	Estado del sensor de presión

NOMBRE	POSICION	UNIDADES DE MEDIDA	COMENTARIOS
ITOW	3	ms	GPS milisegundos
LAT	7	deg	Latitud
LON	11	deg	Longitud
ALT	15	mm	Altitud por sobre la elipsoide
VEL_N	19	cm/s	Velocidad en x (*North Velocity*)
VEL_E	23	cm/s	Velocidad en y (*East Velocity*)
VEL_D	27	cm/s	Velocidad en z (*Down Velocity*)
Hacc	31	mm	Precisión estimada horizontal
Vacc	35	mm	Precisión estimada vertical
Sacc	39	cm/s	Precisión estimada velocidad
bGPS	43	-	Estado del GPS
TS	44	-	Contador de muestreo

(Xsens 2010)

Chequeo (CS).

Este campo se usa para la comunicación y la detección de errores, si todos los bytes excluyendo el preámbulo son sumados y el byte de más pequeño valor resulta igual a cero, entonces el mensaje es válido y puede ser procesado, el valor del byte de chequeo debe ser incluido en la suma (Xsens 2010).

2.3.2 Pc-104.

Esta unidad desarrolla el algoritmo de navegación. Es la encargada de estimar la posición y actitud del vehículo a partir de la relación INS/DNVM/GPS, maneja los históricos y administra las comunicaciones con la estación remota. La misma se conecta como se muestra en la figura 2.6 con los demás elementos del HRC-AUV. La unidad PC-104 figura 2.5 ejecuta de la solución de navegación y de la adquisición de los datos desde la IMU y el GPS. La IMU proporciona la actitud y razones de giro del vehículo, variables esenciales para los lazos de control. Este sensor opera a una frecuencia configurable entre 25 y 100Hz, los datos que provee son usados para implementar la solución de navegación inercial (Lemus 2011).

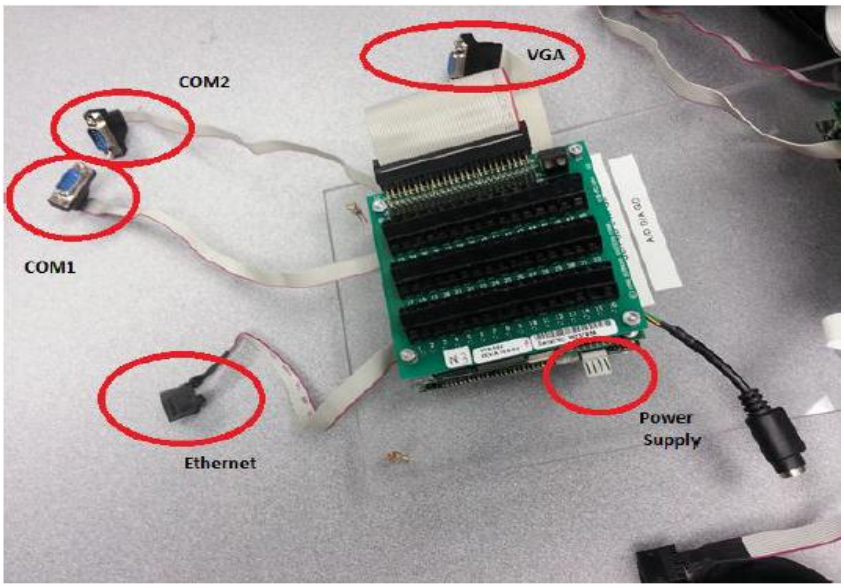


Figura 2.5 Unidad Pc-104

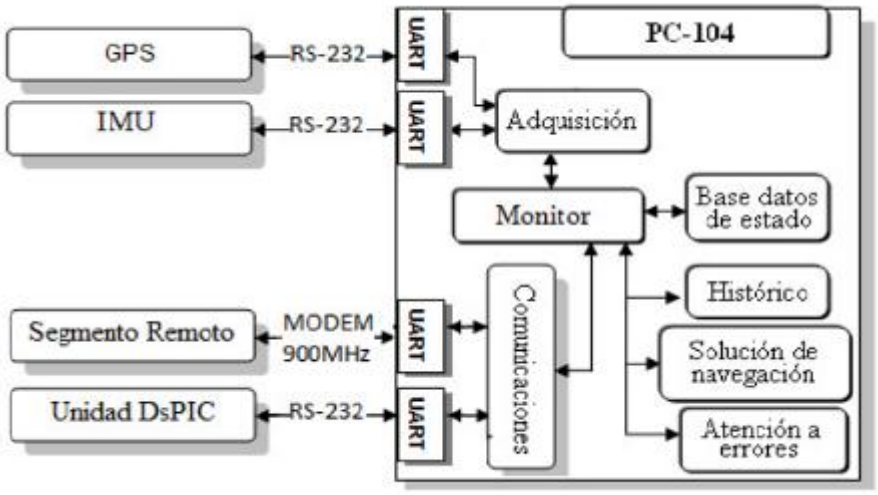


Figura 2.6 Conexión IMU-Pc-104

La unidad PC-104 es responsable de la manipulación de la base de datos del sistema y la generación de históricos. De la misma forma provee información: del estado, posición, actitud y velocidad del vehículo a la estación del operador, a través de un enlace inalámbrico vía MODEM; recibiendo de la estación del operador los ajustes de los controladores y los comandos de tele-operación.

2.4 Descripción del software.

A continuación, se realizará una descripción de todo *software* utilizado a lo largo del trabajo para llegar a la aplicación final la cual fue realizada en la versión "Matlab 2010".

2.4.1 Software PC-104.

El *software* que se ejecuta en la PC-104 está descrito en varios documentos, pero debido a su importancia realizaremos una breve explicación apoyándonos en el Diseño de un AUV (Martínez, Rodrígues et al. 2013)

Las diferentes tareas que ejecuta la pc-104 poseen diferentes niveles de prioridad para así lograr una atención especializada a los sensores que requieran mayor tiempo de muestreo, entre estas tareas se encuentran:

- Adquisición de la información brindada por la IMU y el GPS.
- La navegación inercial asistida. Determinación de posición actual (lineal y angular), rumbo y velocidad del vehículo.
- Actualizar la base de datos del sistema.
- Adquirir los valores de ganancia de los controladores de la estación remota.
- Enviar la información de la actitud del vehículo, valores deseados y ajustes de los controladores de la unidad de Ds-PIC (encargada del control).
- Ejecutar el *software* de comunicación.
- Reportar el estado y la posición del medio a la estación remota que se encarga de la supervisión.

Todo este procedimiento se muestra en la figura 2.7 (Martínez, Rodrígues et al. 2013):

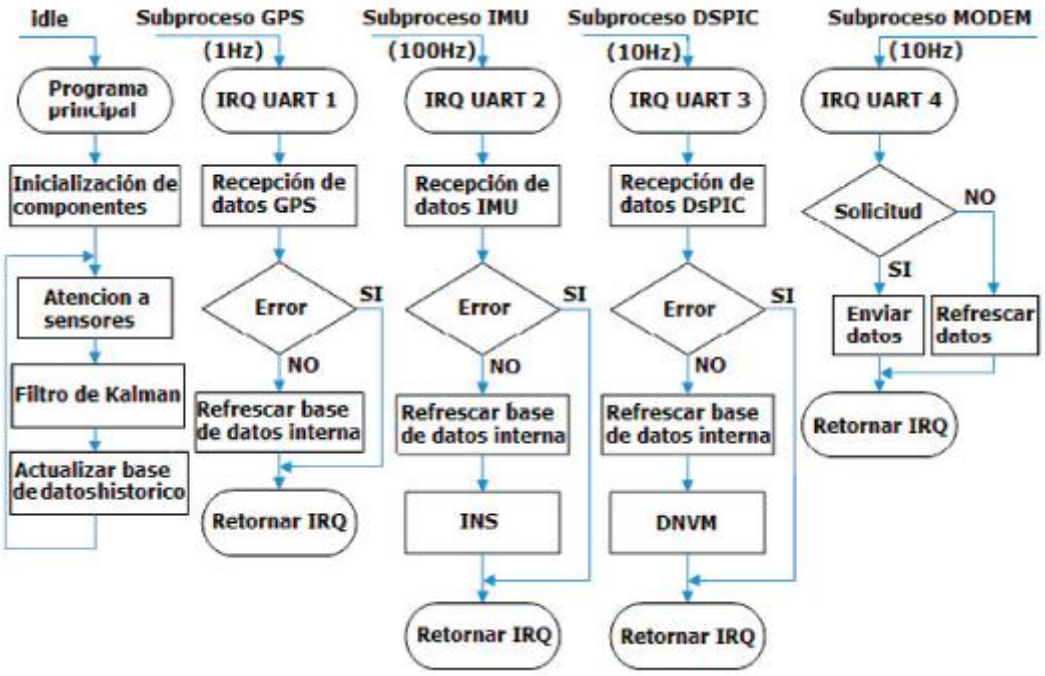


Figura 2.7 Tareas ejecutadas por la Pc-104

2.4.2 Unidad Ds-PIC.

Esta unidad posee un programa informático que establece la lógica de más bajo nivel y que controla los circuitos electrónicos, permitiéndole reaccionar ante cualquier tipo de error. Mediante la misma se maneja el algoritmo de control del sistema y se provee los mandos a los actuadores que regulan los timones del vehículo. Básicamente el *software* se dedica a decidir cuáles son los elementos que funcionaran en el lazo de control y cuáles no. Además los elementos que estén habilitados tendrán una parametrización realizada por el mismo (Martínez, Rodrígues et al. 2013).

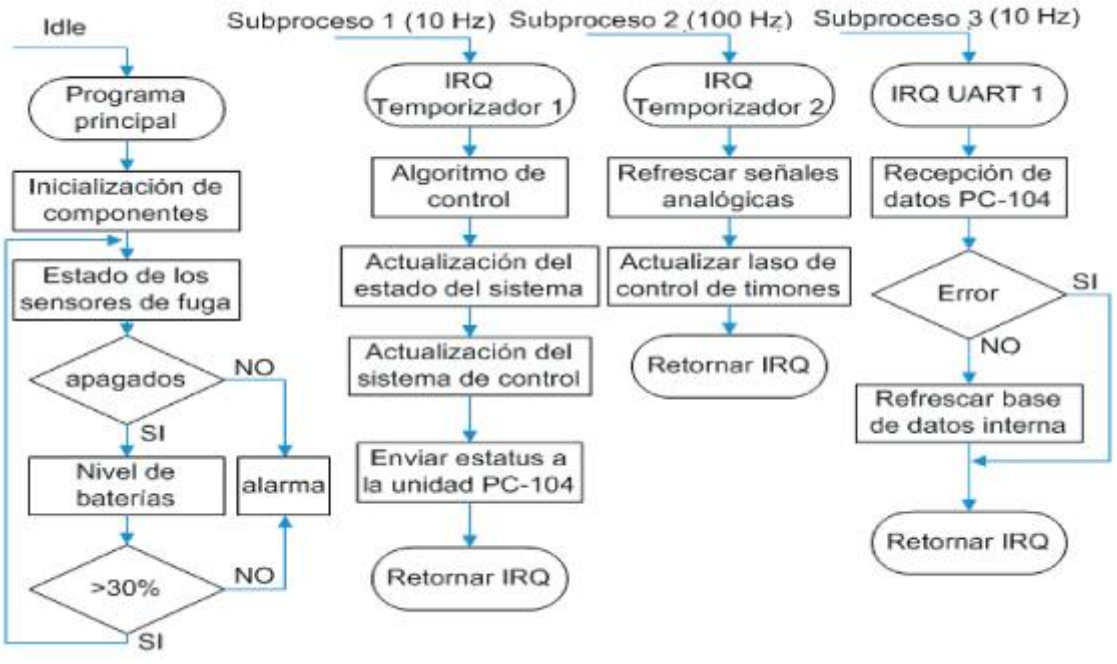


Figura 2.8 Tareas ejecutadas por Ds-PIC

2.4.3 Supervisor.

Este servicio mostrará la información sensorial relevante la cual consiste en la posición espacial dada en latitud/longitud, los ángulos de cabeceo y de balanceo en conjunto con el rumbo, la profundidad, velocidad y su estado (motor encendido o apagado). Este servicio se muestra en una interfaz de usuario independiente que permite monitorear el estado del vehículo mientras se realizan otras tareas de configuración (Martínez, Rodrígues et al. 2013) La figura 2.9 muestra cómo queda la interfaz implementada para su uso

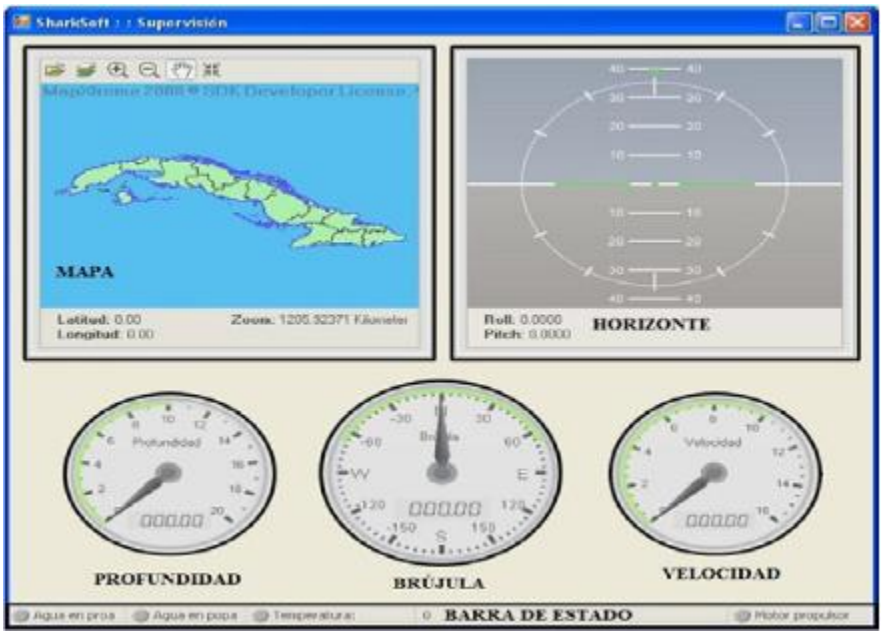


Figura 2.9 Software supervisor

En la interfaz mencionada se pueden encontrar los siguientes elementos:

- **Mapa:** Mapa geográfico sobre el cual se refleja la trayectoria y posición actual del vehículo.
- **Horizonte:** Indica los ángulos de cabeceo y balanceo.
- **Brújula:** Indica el rumbo respecto al norte magnético.
- **Profundidad:** Muestra la profundidad en metros a la que se navega.
- **Velocidad:** Muestra la velocidad a la que navega el vehículo.

2.5 Modelo Matemático.

Del modelo matemático del vehículo desarrollado por el GARP podemos obtener la información que necesitamos para conformar la cadena que se enviará a la IMU, parámetros como las velocidades lineales y angulares en los tres ejes, las aceleraciones lineares y angulares en los tres ejes, magnetómetros en los tres ejes, posición en los tres ejes, así como también podemos obtener la latitud, la longitud, y la altura, y otras como el balanceo *(roll)*, el cabeceo *(pitch)* y la guiñada (*yaw).*

De modo que para la conformación de la misma creamos un bloque en "Matlab" el cual tiene como funcionalidad extraer todos estos datos del modelo matemático del AUV (Valeriano 2013), para conformar la cadena y enviarla por el puerto serie de la PC hacia la PC-104.

2.5.1 Lazos de control.

Como fue mencionado en el capítulo anterior contábamos con los lazos de control de rumbo y de profundidad para el diseño del *software* de simulación con *hardware* en lazo. Estos mismos ya se encuentran implementados en "Matlab" y en conjunto con el modelo matemático para el vehículo submarino autónomo, conformarán parte del *software*.

Básicamente el bloque subsistema creado para el rumbo posee como entradas un rumbo fijo (yaw_d) y el vector guiñada, rango de la guiñada (yaw, yaw_rate), mientras que en la salida tenemos la variación del timón (delta_T). Por su parte el bloque subsistema de control de profundidad tiene al cabeceo (pitch), la profundidad y la constante (zd) como entradas y de salida el estabilizador.

Las salidas de estos dos bloques son las entradas para el subsistema desarrollado a partir del modelo matemático para un vehículo submarino autónomo y la forma en que se conectan se muestra en la figura 2.10 (Valeriano 2013):

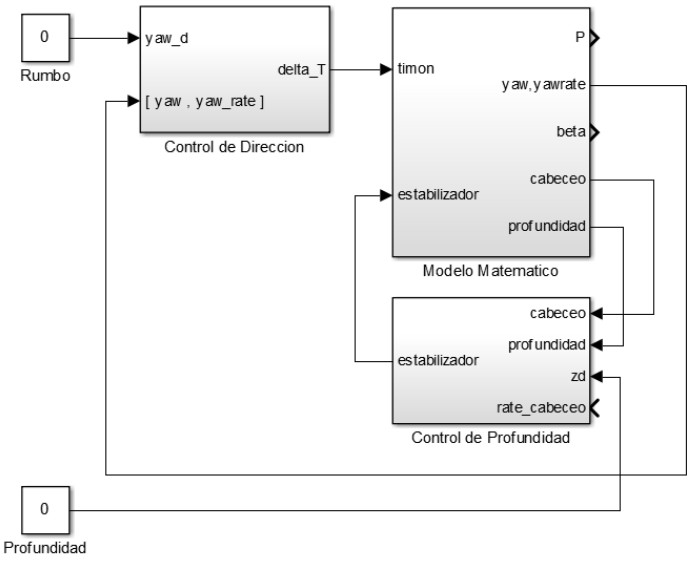


Figura 2.10 Software Matlab modelo matemático

2.5.2 Extracción de Datos.

Del “Matlab” (function_1) que corresponde al modelo matemático de 6 grados de libertad (6DOF-NL) salen 6 aceleraciones tres lineales y tres angulares, las cuales pasan por un integrador, de aquí la integral de la aceleración es la velocidad obtenemos las 6 velocidades, tres angulares y tres lineales y en un final vuelven a ser integradas para terminar obteniendo las posiciones en los tres ejes, así como el balanceo (roll), el cabeceo (pitch), y la guiñada (yaw).

Con esto resolvemos una parte del problema, pero en la cadena encontramos en los bytes con las posiciones 7, 11 y 15 (latitud, longitud, altura) datos que no podemos extraer del modelo matemático del vehículo AUV. Para este caso en específico se realizó la construcción de un subsistema, que a partir de las posiciones extraídas obtiene los bytes faltantes.

2.5.3 Sistema de posicionamiento global.

En el mundo existen varios sistemas de coordenadas los cuales por sus características en específico poseen diferentes prestaciones, por ejemplo, en mediciones y determinación de orbitas GPS satelitales, se utiliza el de Centrado Terrestre Inercial (ECI) siglas en inglés, el cual tiene como punto de referencia el centro de masa de la Tierra y se ajusta de acuerdo a la posición de las estrellas.

En nuestro caso se encuentra el Centrado por la Tierra y Ajustado por la Tierra (ECEF) siglas en inglés, el mismo es el utilizado con el propósito de computar información recibida de un GPS, en este caso la posición, por lo cual lo más idóneo sería que el sistema de coordenadas se moviera conjuntamente con la Tierra.

En el caso de del ECEF el plano XY coincide con el plano del Ecuador de la Tierra, el eje de las abscisas está orientado al punto longitud 0° o más conocido como el meridiano Greenwich, el eje de las ordenadas estará apuntando 90° Este.

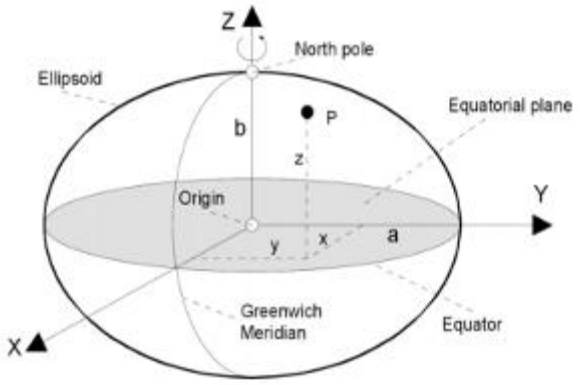


Figura 2.11Ajuste por la tierra y centrado por la tierra

Como estas coordenadas son cartesianas el siguiente paso sería su transformación en la típica longitud, latitud y altura. En nuestro caso utilizamos para la latitud y la longitud los grados como unidad de medida y para la altura los metros. Para realizar esta conversión se necesita un modelo terrestre para los cálculos, así que utilizaremos el sistema geodético mundial de 1984 WGS84 siglas en inglés debido a que la MTi-G lo utiliza así y para conformar la cadena debemos cumplir estos parámetros, el mismo ofrece una descripción matemática de la capa de la tierra en conjunto con sus irregularidades gravitacionales (Xsens 2010).

2.6 Tiempos de Comunicación.

Para la mayoría de las aplicaciones es importante el manejo de los tiempos de comunicación para conseguir un funcionamiento óptimo, en nuestro caso no es diferente debido a la gran cantidad de comunicaciones que posee nuestro trabajo. En general existen dos factores que rigen la variación de tiempo cuando ocurre un evento.

- El tiempo de adquisición y cálculo
- El tiempo de transmisión

La primera depende del escenario donde esté operando más el modo en que se transmite la salida. La tabla siguiente fue extraída de datos recolectados de la Xsens (del filtro de

kalman de 6 grados de libertad XKF-6) donde muestra algunos tiempos de adquisición y computación de diferentes escenarios (Xsens 2010).

Tabla 2.2 Tiempos de adquisición y computación.

XKF-6 modo de salida	Peor caso adquisición y computación (calibrados)	Peor caso adquisición y computación (orientados)
General	0.38ms	4.99ms
Aeroespacial	0.38ms	5.38ms
Automotor	0.38ms	4.97ms
Marina	0.38ms	5.44ms

Por otra parte el tiempo de transmisión puede ser calculado con facilidad con solo tener la cantidad de bytes que conforman el mensaje y la tasa de baudios que no es más que el número de unidades de señal por segundo (*baudrate*), la ecuación se muestra en la figura 2.12.

$$\frac{total\ de\ bytes\ en\ el\ mensaje * 10bits/byte}{baudrate\ de\ la\ comunicación\ (bits/s)} = tiempo\ de\ transmición$$

Figura 2.12 Cálculo de tiempo de transmisión

En nuestro caso es necesario destacar que el preámbulo más el identificador del BUS, más el identificador del mensaje, más el indicador del tamaño del mensaje y sumado por ultimo al chequeo conforman un total de 5 bytes.

2.7 Conclusiones parciales del capítulo.

A modo de conclusión en este capítulo tenemos la selección del “Matlab” como herramienta para desarrollar el *software* que permita la simulación con *hardware* en lazo debido a la existencia en esta plataforma del modelo matemático desarrollado por el GARP

para el vehículo HRC-AUV sumado a los algoritmos de control que proveen los mandos a los actuadores que regulan los timones del vehículo. Además de sus prestaciones para realizar simulaciones en tiempo real (RT) y tiempo real aproximado (SRT).

A partir de la extracción de datos del modelo matemático desarrollado por el GARP para el vehículo HRC-AUV se pueden conformar las cadenas de transmisión a la Pc-104 con la estructura indicada por la IMU para realizar la simulación.

CAPÍTULO 3. RESULTADOS Y ANÁLISIS

En capítulos anteriores se presentaron todos los mecanismos teóricos que dieron paso a esta investigación, así como también los materiales y métodos utilizados para llevar a cabo la misma, en este capítulo se presenta la confección del programa en conjunto con las pruebas realizadas sobre el mismo además de los resultados alcanzados y su análisis.

3.1 Confección del software.

En un comienzo tenemos el modelo matemático desarrollado por el GARP para el vehículo HRC-AUV en “Matlab”, con condiciones iniciales especificadas. A partir de este punto se construye un *software* apoyándonos en la herramienta de “Matlab-Simulink”. Para el mejor entendimiento explicaremos brevemente cada bloque funcional creado.

3.1.1 S-Function.

La extracción de los datos del modelo matemático desarrollado por el GARP para el vehículo HRC-AUV en “Matlab” se realiza mediante un bloque “S-Function”, los mismos serán utilizados para la confección de la cadena que será transmitida con posterioridad a través del puerto serie.

Figura 3.1 Matlab *S*-Function

El mismo obtiene valores del modelo antes mencionado como las aceleraciones lineales y angulares en los tres ejes, las velocidades lineales y angulares en los tres ejes, latitud,

longitud, y altura sumados al balanceo (*roll*), el cabeceo (*pitch*) y la guiñada (*yaw*). Todos estos datos son utilizados para formar el vector que será enviado por el puerto serie en el formato descrito con anterioridad en el capítulo 2 de cadena. Después de la transformación de los datos para que puedan ser ubicados en el vector en forma de bytes son llevados al "Estándar IEEE 754" en el que se transmiten desde la IMU. Siguiendo la organización antes planteada de los campos para conformar la cadena de mensaje se fijan los bytes de preámbulo, bus identificador e identificador de mensaje con los siguientes datos 250 (0xFA), 255 (0xFF) y 50 (0x32) respectivamente, se prosigue con el tamaño del mensaje y los demás datos obtenidos. Para terminar, realiza el cálculo del campo de chequeo el cual le permitirá decidir si la cadena a transmitir es válida o no.

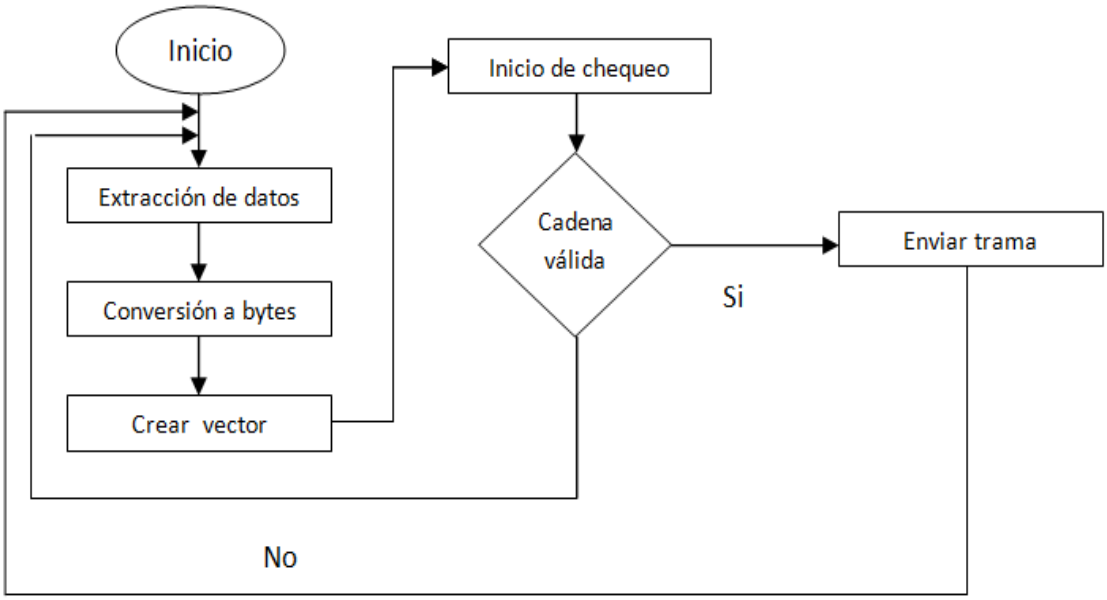


Figura 3.2 Tareas a realizar por el bloque *S*-Function

3.1.2 Comunicación.

Para la comunicación fueron utilizados varios cables series, con diferentes protocolos. En un primer lugar tenemos la comunicación entre el *software* creado y la Pc-104, en un segundo lugar tenemos la comunicación entre el Ds-PIC y la Pc-104 y por ultimo tenemos la comunicación entre el supervisor y la Pc-104. Los valores de estas comunicaciones se presentan en la siguiente tabla.

Tabla 3.1 Especificación de comunicaciones.

Nombre	COM	RS	Baudrate
(Comunicación MTI)	COM 1	232	57600
Ds-PIC	COM 3	485	115200
Supervisor	COM 2	232	9600

En el caso de la comunicación con el Ds-PIC fue conectado a un convertidor RS-232 para su uso.

3.1.3 Bloques de comunicación serie de Matlab.

Conformado un vector, se transmite por el puerto serie de la computadora donde se ejecuta el *software*, mediante el elemento "*To Instrument*" perteneciente a la librería del "*Simulink*", el cual tiene como función principal la transmisión por el puerto serie de forma constante, para lograrlo es necesario configurar el tiempo de muestreo a (-1), concibiendo así una comunicación continua.

Figura 3.3 Configuración del bloque *To Instrument*

Es importante resaltar que mediante este bloque se pueden hacer un conjunto de modificaciones las cuales nos permiten configurar parámetros como la velocidad de

transferencia, número de puerto por el que se transmite, así como el tipo de datos que se va a transmitir.

Por otro lado, es necesario efectuar las lecturas del puerto serie para recibir los datos que son enviados de la Pc-104 y así poder cerrar el lazo del programa enviando los nuevos valores a los mandos y los actuadores.

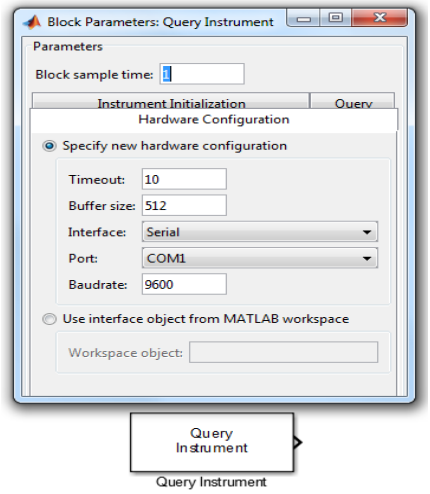


Figura 3.4 Configuración del bloque *Query Instrument*

En este caso en específico utilizamos el "Query Instrument" que posee el mismo modo de operación que el bloque "*To Instrument*" diferenciándose en que su funcionalidad es la lectura del puerto serie. Con este bloque se completaría la comunicación de la Pc-104 con el Ds-PIC el cual se encuentra implementado en conjunto con el *software*, para la actualización de los mandos y los actuadores. Es necesario identificar las cadenas recibidas desde el navegador y se usan como setpoint para los mandos del modelo matemático, debido a que son enviadas dos cadenas diferentes. La primera es más larga con un MID 01 se utiliza para la configuración, la segunda con MID 02 contiene la información solo se deben separar los bytes que pertenecen al rumbo y a la profundidad (los últimos 8 bytes) los cuales son los que cierran el lazo en el modelo matemático realizado por el GARP. Con la construcción de otro bloque S-function se logra extraer la cadena separar estos valores y enviarlos para actualizar los parámetros antes mencionados.

3.1.4 Subsistemas.

Para la obtención de los datos es necesario percatarse que del modelo hay una serie de datos que deben ser ajustados para poder ser enviados en la cadena hacia la “Pc-104” y sean mostrados en el formato correcto.

En un primer lugar solo podemos extraer la posición en los tres ejes, sin embargo necesitamos la latitud la longitud y la altura, de aquí la confección de un subsistema para el cálculo de las mismas utilizando el bloque funcional “ECEF *to* LLA” del “*Simulink*”. El mismo permite la conversión adaptándose a los parámetros con los que trabaja la IMU para lograr una conversión lo más acercada a la realidad, la figura 3.5 muestra la conformación del mismo.

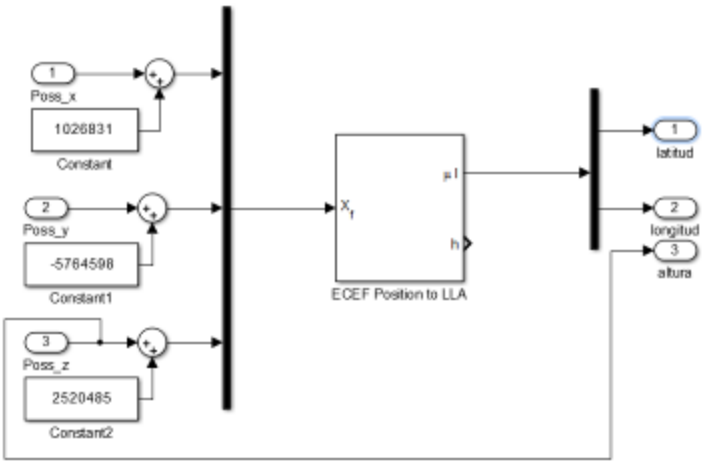


Figura 3.5 Primer subsistema

Como se podrá apreciar a los valores obtenidos se le suman una serie de ganancias, las mismas representan las coordenadas en las que se encuentra actualmente el vehículo hipotéticamente o sea el laboratorio en donde se realiza actualmente el experimento.

Por otro lado, el segundo subsistema creado corresponde a los campos dedicados al balanceo, el cabeceo, y la guiñada los cuales son enviados en radianes sin embargo es necesaria su conversión debido a que al realizar la simulación entre los datos revelados por el supervisor encontramos tanto el balanceo como el cabeceo en grados. La figura 3.6 muestra el subsistema creado.

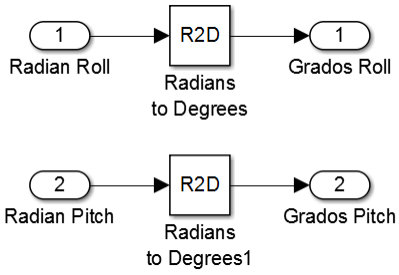


Figura 3.6 Segundo subsistema

Además, se puede apreciar en la misma el uso del bloque R2D de "*Simulink*", el cual se dedica a realizar esta conversión.

3.2 Simulaciones.

Para la sincronización de la simulación con el tiempo real tenemos el bloque "*Real-Time Sync*" el cual se muestra en la siguiente figura 3.7 en conjunto con su configuración.

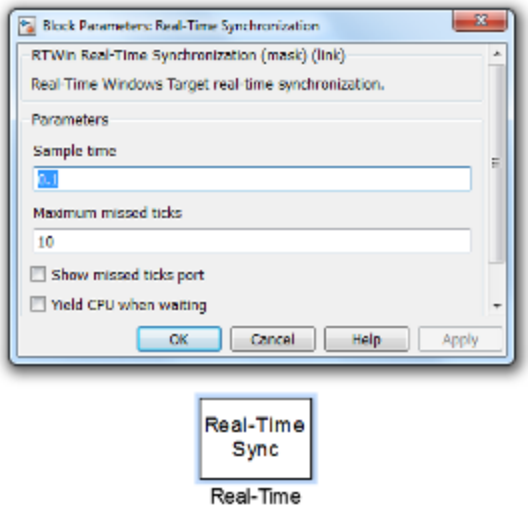


Figura 3.7 Bloque de simulación tiempo real

Las primeras pruebas realizadas se enfocaron en la comunicación del *software* conjuntamente con la "Pc-104", las siguientes tramas muestran cómo se comportaron las simulaciones, utilizando el "COM 1" de la computadora donde se ejecuta el *software* y el "Serial Port Monitor" como herramienta para monitorear el puerto serie, se obtuvieron los siguientes resultados figura 3.8, los mismos fueron comparados con las simulaciones antes realizadas con la IMU.

Al estudiar estos datos apreciamos, que las tramas enviadas por la IMU monitoreada con anterioridad, contienen una gran similitud con las obtenidas en simulaciones, lo cual era esperado debido a que se construyó siguiendo los estándares de la misma. A continuación, se presenta una de las cadenas enviadas por la IMU en conjunto con las tramas enviadas por el *software* creado.

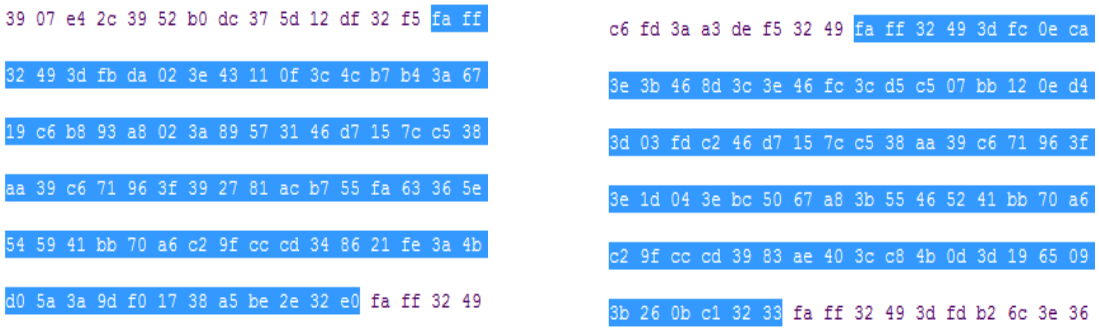


Figura 3.8 Tramas enviadas por la IMU y por Matlab

3.2.1 Pruebas finales.

Después de realizar todas las conexiones para comunicar tanto el *software* con la Pc-104 y con el supervisor, se decidió realizar pruebas para un rumbo y una profundidad constante y en un segundo lugar una prueba manteniendo un rumbo fijo y variando en la profundidad en diferentes instantes de tiempo para un rpm del motor de 500.

Para la primera simulación obtuvimos los siguientes resultados en un instante de tiempo.

Tabla 3.2 Primeros resultados

Latitud	Longitud	Profundidad	Balanceo	Cabeceo
23.4301	-79.9006	-0.34	0.0049	0.3455

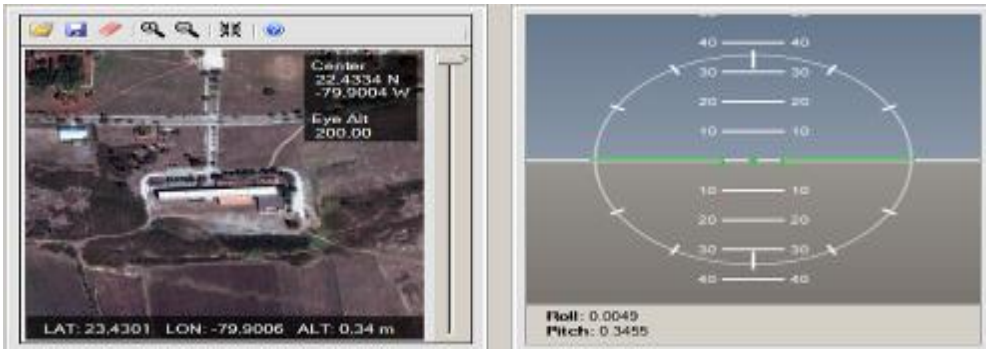


Figura 3.9 Supervisor primera simulación

Para la segunda simulación se establecen varios *set point*, los mismos comenzarán a partir de la depresión cero. Con la primera inmersión se pretende lograr una profundidad de 20 m, en un segundo momento se descenderá una vez más hasta alcanzar los 50 m y para finalizar se retornará a la posición inicial, además en el transcurso de la simulación se realizarán modificaciones al rumbo (0 a 180 grados).

En el diagrama siguiente se muestra el modo de operación de la simulación para su mejor entendimiento.

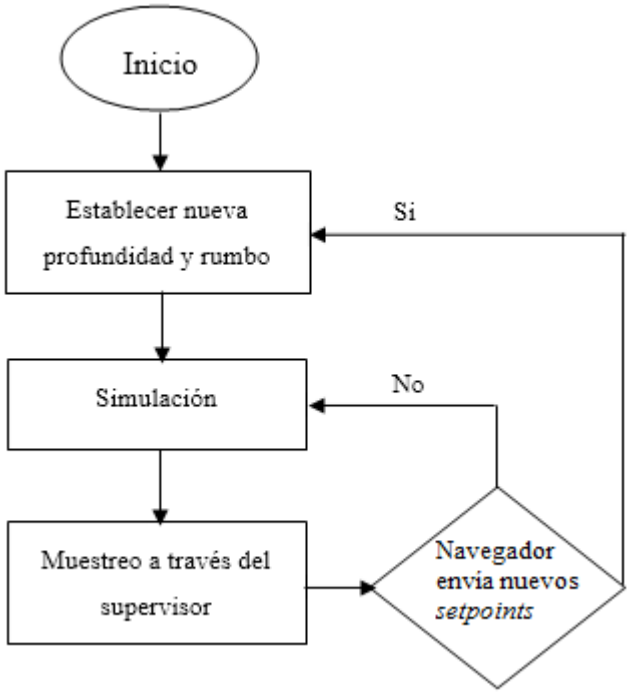


Figura 3.10 Diagrama de simulación

A continuación, se muestra el comportamiento del supervisor durante los tres cambios de profundidad.

Figura 3.11 Supervisor segunda simulación primer descenso

Figura 3.12 Supervisor segunda simulación segundo descenso

Figura 3.13 Supervisor segunda simulación retorno a la profundidad cero

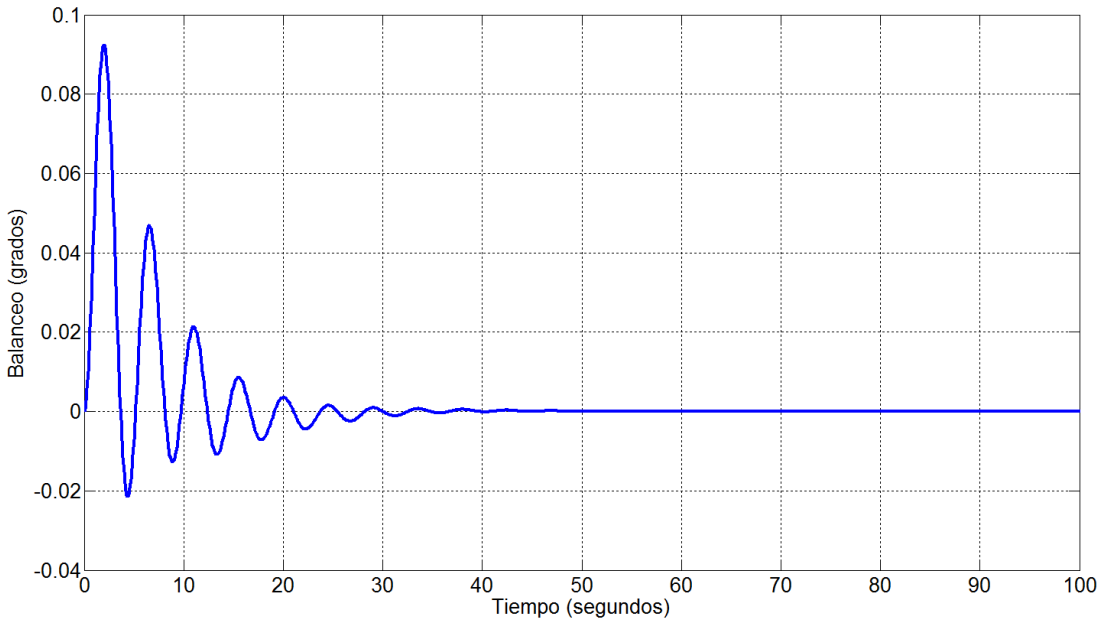


Figura 3.14 Balanceo en el tiempo segunda simulación hasta los 100 segundos

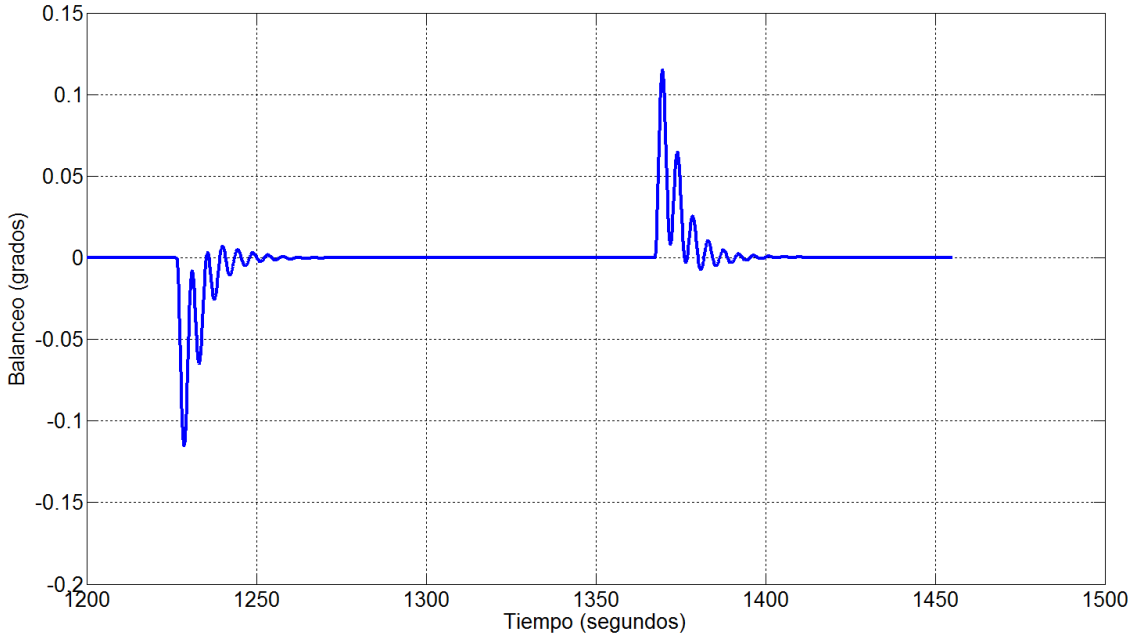


Figura 3.15 Balanceo en el tiempo segunda simulación hasta los 1200 hasta los 1500 segundos

Como se puede apreciar en las figuras anteriores, existen tres ocasiones en las que se observa el balance del vehículo durante la simulación, en un primer momento durante la arrancada, el balanceo alcanza valores superiores a los 0.08 grados pero no sobrepasa el 0.1 grado. En el segundo 50 mantiene una estabilidad y continuará en estas condiciones hasta que reciba dos variaciones de rumbo, las mismas pueden ser apreciadas a partir del segundo 1200 y el segundo 1350 en la figura 3.17, los dos instantes de tiempo anterior pueden estudiarse en la figura 3.15, el primero estabiliza en el segundo 1275 mientras que el segundo logra la estabilidad durante el segundo 1410 aproximadamente.

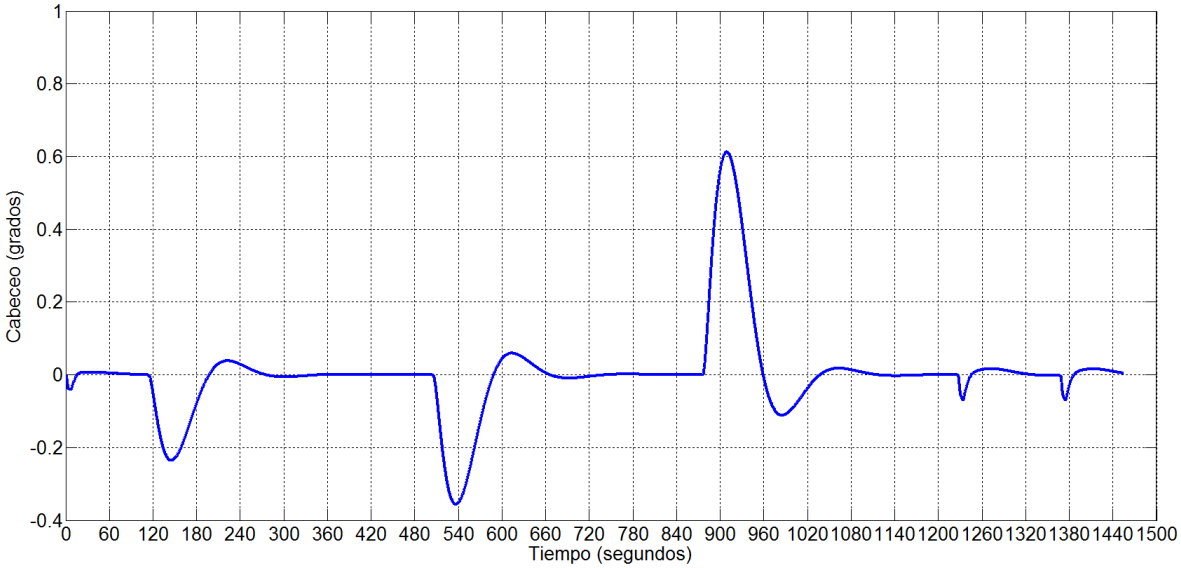


Figura 3.16 Cabeceo en el tiempo segunda simulación

Para el cabeceo en el gráfico mostrado en la figura 3.16 son notables las tres ocasiones en las que se cambia la profundidad. Durante la primera inmersión alcanza los -0.22 grados y se estabiliza sobre los 300 segundos aproximadamente, la segunda casi alcanza los -0.4 grados y estabiliza en los 780, mientras que para el cazo del retorno hacia la superficie alcanza los 0.6 grados y obtiene el estado estable a partir de los 1140 segundos. Las dos fluctuaciones en el cabeceo a partir del segundo 1200 y 1250 pertenecen a los dos cambios de rumbo explicados con anterioridad.

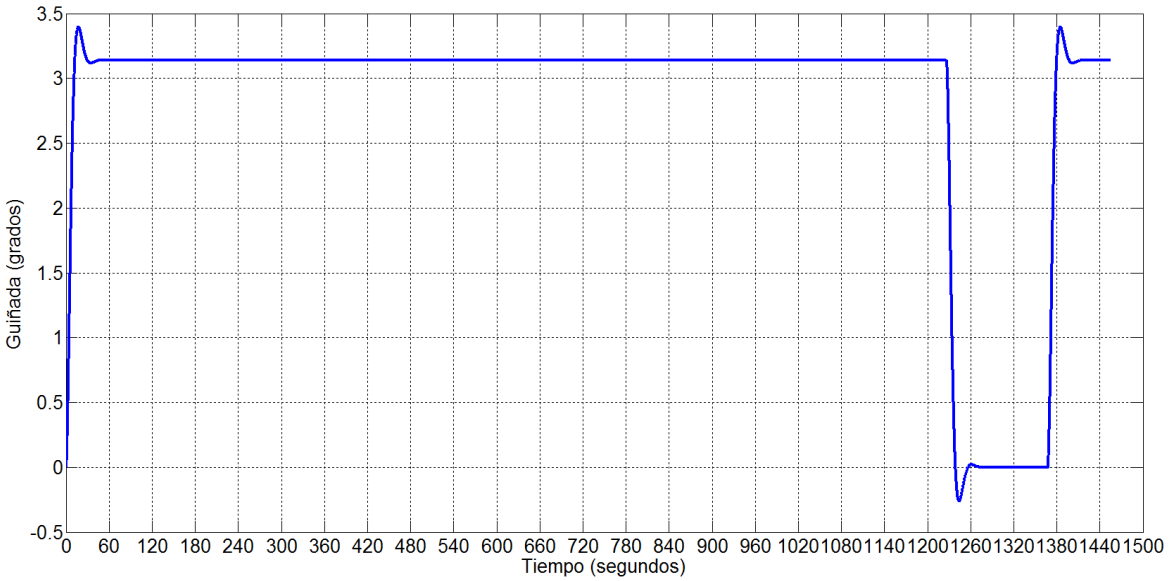


Figura 3.17 Guiñada en el tiempo segunda simulación

En la figura 3.17 se observan los tres cambios de rumbo realizados a lo largo de la simulación, los mismos se basan en las entradas especificadas al *software* las cuales fueron solo dos. En el inicio se le especifico el valor de 3.14 y a partir del segundo 1200 se regresó este valor a 0 hasta que se alcanzan los 1350 segundos volviendo a establecer los 3.14.

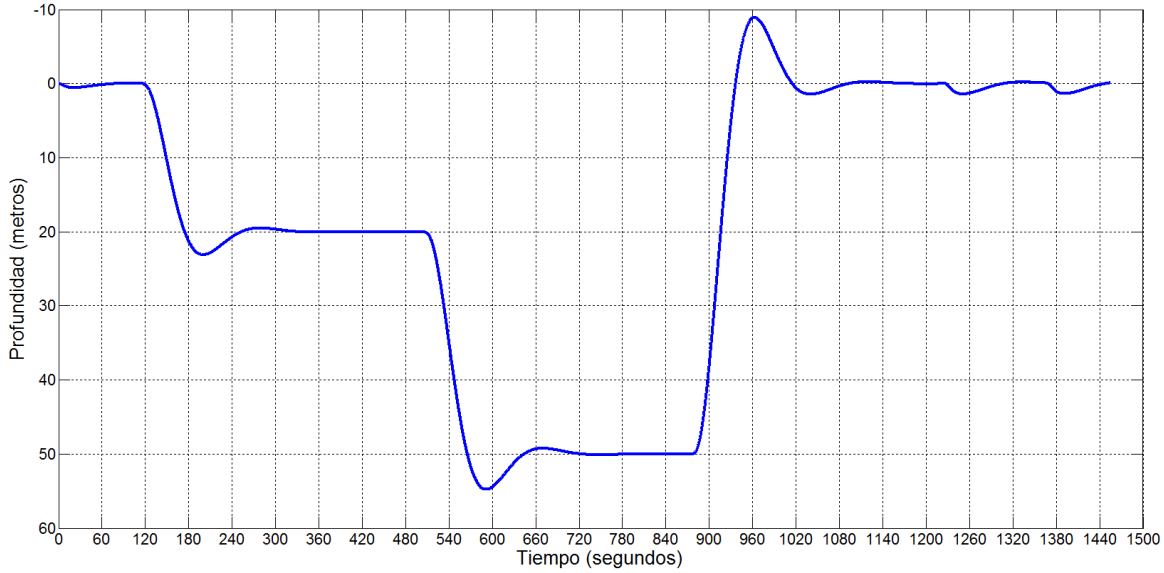


Figura 3.18 Profundidad en el tiempo segunda simulación

En la figura 3.18 podemos encontrar el comportamiento de la profundidad en función del tiempo las dos inmersiones en conjunto con sus tiempos de estabilización.

Tabla 3.3 Resultados segunda simulación.

Profundidad deseada	Estabilización tiempo(s)	Latitud	Longitud	Balanceo	Cabeceo
20	320	23.4301	-79.9004	0.0120	-12.7878
50	740	23.4312	-79.9118	0.0000	-16.2863
0	1080	23.4312	-79.9175	0.0000	17.8932

3.3 Análisis económico.

Para el análisis económico nos enfocaremos en la comparación del gasto monetario que representa una prueba real contra una simulación con hardware en lazo. La ejecución de una prueba real al vehículo HRC-AUV desarrollado por el GARP se debe contar con varios factores que representan un gasto económico elevado. La renta de un equipo para el transporte hacia una zona costera puede rondar los (90 USD), la inmersión del mismo, o sea una embarcación para adentrarlo en aguas profundas más el contrato de un equipo de buceo para situarlo en la superficie marina y garantizar su recuperación del vehículo submarino en caso de falla de la prueba real debe sumar alrededor de unos (750 USD).

En nuestro caso la suma de dinero se enfocaría en la adquisición del MATLAB para la confección del software la cual rondaría los (4000 USD) (Mathworks 2016), y el costo del hardware de comunicación entre los diferentes elementos utilizados para realizar la simulación con hardware en lazo, debido que se utilizó la conexión mediante los puertos series descritos anteriormente podemos decir que se resumiría a la adquisición de tres cables series lo cual sumaria unos 9 USD además del costo de una computadora para la ejecución del software creado, en nuestro caso serían 350 USD.

Como se puede observar realizando las sumas de los cálculos finales obtenemos que para la realización de una prueba real debamos contar con la suma de 840 USD mientras que para poner en marcha nuestro trabajo sumarian unos 4359 USD. Además, podemos afirmar que la inversión en nuestro trabajo se ejecutaría en una sola ocasión, por otro lado, el gasto monetario de las pruebas reales será tan grande como pruebas reales se quieran plantear en un periodo de tiempo dificultando aún más este procedimiento en otras palabras después de realizadas 6 pruebas reales la suma se elevará hasta los 5040 USD.

3.4 Conclusiones parciales.

En este capítulo a modo de conclusión tenemos en un comienzo, la confección del *software* en el "MATLAB/*Simulink*" permitirá un gran uso en trabajos posteriores debido a que es el *software* más utilizado en nuestra docencia.

La gran cantidad de información que permite monitorear el *software* creado para realizar la simulación con *hardware* en lazo concederá al usuario del mismo aumentar la capacidad de detección de errores en futuras aplicaciones.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones.

1 Mediante la simulación con hardware en lazo realizada para el vehículo HRC-AUV perteneciente al GARP se posibilita la ejecución de pruebas, capaces de prevenir posibles fallas futuras durante su uso.

2 Para el desarrollo de una simulación de con *hardware* en lazo es de gran importancia el modelado matemático fiable del sistema.

3 El *software* desarrollado en "MATLAB/Simulink" para la simulación con *hardware* en lazo cumple con los requisitos mínimos para la realización de pruebas con tiempo real blando cumpliendo las expectativas.

Recomendaciones.

A modo de recomendación tenemos la confección de un método de comunicación más versátil que el utilizado en nuestro trabajo para poder realizar pruebas a distancia y no depender de la conectividad por medio de protocolos utilizando el puerto serie de las máquinas.

La obtención de un modelo más fiable para mejorar los resultados alcanzados ya que este es imprescindible en la realización de una simulación con hardware en lazo.

REFERENCIAS BIBLIOGRÁFICAS

Antonelli, G., T. I. Fossen and D. R. Yoerger (2008). Underwater Robotics. Springer Handbook of Robotics, Springer-Verlag**:** 987–1008.

Casellas, F., J. Esteban, F. Guinjoan, R. Pique, H. Martínezy and G. Velasco (2014). "Simulación Mediante "Hardware In the Loop" de un Convertidor Buck." Universitat Politècnica de Catalunya - BarcelonaTECH (UPC).

Fjellstad, O. (1994). Control of unmanned underwater vehicles in six degrees of freedom a quaternion feedback approach. Doctoral thesis, NTNU.

Fossen, T. I. (1994). "Guidance and Control of Ocean Vehicles." John Wiley & Sons.

Fossen, T. I., T. A. Johansen and T. Peréz (2008). Underwater vehicles. A survey of control allocation methods for underwater vehicles. Vienna, Austria, InTech**:** 109–128.

Fossen, T. I. and A. Ross (2006). Nonlinear modelling, identification and control of UUVs. Advances in unmanned marine vehicles. Gran Bretaña, Peter Peregrinus LTD. **69:** 13–42.

Gauchía Babe, L. (2008). Modelado y Simulación HIL de un Sistema Pila de Combustible-Bateria. Maestría en Ingeniería Eléctrica, Electrónica y Automática, Carlos III de Madrid.

Hans, P. (2011). "Hardware in the Loop Simulation." Hogskolen i Telemark **56**: 13.

Jacomet, M., J. Goette, M. Hager, W. Chigutsa and N. Leuba (2012). "Hardware-in-the-Loop Simulation by linking Matlab/Simulink with a HW/SW Co-Design Rapid-Prototyping Platform." University of Applied Sciences Berne.

Jordan, J. Lemus, A. Mario and Bustamante (2008). "Underwater vehicles." I-Tech: 251–278.

Kopets, H. (2002). "Real-Time Systems, Design Principles for Distributed Embedded Aplications." Kluwer Academic Publishers.

Lemus, J. (2011). Trabajo de Diploma Software de navegación y guiado en tiempo real para Vehículo Autónomo Sumergible. Tesis de grado. Ingeniería, Universidad Central Marta Abreu de Las Villas.

Liang, J., H. Wei, T. Wang and L. Wen (2008). "Underwater vehicles." I-Tech: 173–194.

Martínez, A., Y. Rodrígues, L. Hernández, C. Guerra, J. Lemus and H. Sahli (2013). "Diseño de AUV.Arquitectura de hardware y software." Revista Iberoamericana de Automática e Informática industrial **10**(3): 333-343.

Mathworks. (2016). "Pricing and Licensing." from http://www.mathworks.com/pricing-licensing/index.html?prodcode=SL&requestedDomain=www.mathworks.com.

Nikola, M., Z. Vukic and M. Barisic (2008). "Underwater vehicles." I-Tech: 327–346.

Nokland, H. (2011). Nonlinear Observer Design for GNSS and IMU Integration. master tesis de maestría, Norwegian University of Science and Technology.

Patricia, R. and N. Mario (2008). "Underwater vehicles." I-Tech: 417–436.

Valeriano, Y. (2013). Modelado dinámico de un vehículo autónomo subacuático. Tesis de Maestría. Ingeniería, Universidad Central Marta Abreu de Las Villas.

Xsens (2010). MTi-G User Manual and Technical Documentation, Xsens Technologies B.V.

Zahari, T., D. Abdelhakim, K. Azeddein and A. Raja (2012). "Development of Real-Time Hardware in the Loop Based MPC for Small-Scale Helicopter." InTech: 15.

ANEXOS

Anexo I Código de Fuente *S-Function*

```
function y =
fcn(AccL_x,AccL_y,AccL_z,AccA_x,AccA_y,AccA_z,VelL_x,VelL_y,VelL_z,VelA_x
,VelA_y,VelA_z,Mag_x,Mag_y,Mag_z,latitud,longitud,altura,roll,pitch,yaw)

% #codegen

%  [x,y] =
fcn(AccL_x,AccL_y,AccL_z,AccA_x,AccA_y,AccA_z,VelL_x,VelL_y,VelL_z,VelA_x
,VelA_y,VelA_z,Poss_x,Poss_y,Poss_z,roll,pitch,yaw)
    AccX = num2hex(single(AccL_x));
    AccY = num2hex(single(AccL_y));
    AccZ = num2hex(single(AccL_z));

    GyrX = num2hex(single(VelA_x));
    GyrY = num2hex(single(VelA_y));
    GyrZ = num2hex(single(VelA_z));

    MagX = num2hex(single(Mag_x));
    MagY = num2hex(single(Mag_y));
    MagZ = num2hex(single(Mag_z));

    Roll = num2hex(single(roll));
    Pitch = num2hex(single(pitch));
    Yaw = num2hex(single(yaw));

    Lat = num2hex(single(latitud));
    Lon = num2hex(single(longitud));
    Alt = num2hex(single(altura));

    VelX = num2hex(single(VelL_x));
    VelY = num2hex(single(VelL_y));
    VelZ = num2hex(single(VelL_z));

    Status = '32';

    AccX_1=AccX(1:2);
    AccX_2=AccX(3:4);
    AccX_3=AccX(5:6);
```

```
AccX_4=AccX(7:8);

AccY_1=AccY(1:2);
AccY_2=AccY(3:4);
AccY_3=AccY(5:6);
AccY_4=AccY(7:8);

AccZ_1=AccZ(1:2);
AccZ_2=AccZ(3:4);
AccZ_3=AccZ(5:6);
AccZ_4=AccZ(7:8);

GyrX_1=GyrX(1:2);
GyrX_2=GyrX(3:4);
GyrX_3=GyrX(5:6);
GyrX_4=GyrX(7:8);

GyrY_1=GyrY(1:2);
GyrY_2=GyrY(3:4);
GyrY_3=GyrY(5:6);
GyrY_4=GyrY(7:8);

GyrZ_1=GyrZ(1:2);
GyrZ_2=GyrZ(3:4);
GyrZ_3=GyrZ(5:6);
GyrZ_4=GyrZ(7:8);

MagX_1=MagX(1:2);
MagX_2=MagX(3:4);
MagX_3=MagX(5:6);
MagX_4=MagX(7:8);

MagY_1=MagY(1:2);
MagY_2=MagY(3:4);
MagY_3=MagY(5:6);
MagY_4=MagY(7:8);

MagZ_1=MagZ(1:2);
MagZ_2=MagZ(3:4);
MagZ_3=MagZ(5:6);
MagZ_4=MagZ(7:8);

Roll_1=Roll(1:2);
Roll_2=Roll(3:4);
Roll_3=Roll(5:6);
Roll_4=Roll(7:8);

Pitch_1=Pitch(1:2);
Pitch_2=Pitch(3:4);
Pitch_3=Pitch(5:6);
Pitch_4=Pitch(7:8);

Yaw_1=Yaw(1:2);
Yaw_2=Yaw(3:4);
Yaw_3=Yaw(5:6);
```

```
    Yaw_4=Yaw(7:8);

    Lat_1=Lat(1:2);
    Lat_2=Lat(3:4);
    Lat_3=Lat(5:6);
    Lat_4=Lat(7:8);

    Lon_1=Lon(1:2);
    Lon_2=Lon(3:4);
    Lon_3=Lon(5:6);
    Lon_4=Lon(7:8);

    Alt_1=Alt(1:2);
    Alt_2=Alt(3:4);
    Alt_3=Alt(5:6);
    Alt_4=Alt(7:8);

    VelX_1=VelX(1:2);
    VelX_2=VelX(3:4);
    VelX_3=VelX(5:6);
    VelX_4=VelX(7:8);

    VelY_1=VelY(1:2);
    VelY_2=VelY(3:4);
    VelY_3=VelY(5:6);
    VelY_4=VelY(7:8);

    VelZ_1=VelZ(1:2);
    VelZ_2=VelZ(3:4);
    VelZ_3=VelZ(5:6);
    VelZ_4=VelZ(7:8);


    DataToSend
=['FA';'FF';'32';'49';AccX_1;AccX_2;AccX_3;AccX_4;AccY_1;AccY_2;AccY_3;Ac
cY_4;AccZ_1;AccZ_2;AccZ_3;AccZ_4;GyrX_1;GyrX_2;GyrX_3;GyrX_4;GyrY_1;GyrY_
2;GyrY_3;GyrY_4;GyrZ_1;GyrZ_2;GyrZ_3;GyrZ_4;MagX_1;MagX_2;MagX_3;MagX_4;M
agY_1;MagY_2;MagY_3;MagY_4;MagZ_1;MagZ_2;MagZ_3;MagZ_4;Roll_1;Roll_2;Roll
_3;Roll_4;Pitch_1;Pitch_2;Pitch_3;Pitch_4;Yaw_1;Yaw_2;Yaw_3;Yaw_4;Lat_1;L
at_2;Lat_3;Lat_4;Lon_1;Lon_2;Lon_3;Lon_4;Alt_1;Alt_2;Alt_3;Alt_4;VelX_1;V
elX_2;VelX_3;VelX_4;VelY_1;VelY_2;VelY_3;VelY_4;VelZ_1;VelZ_2;VelZ_3;VelZ
_4;Status];
    DataDec=hex2dec(DataToSend);
    Check= sum(DataDec)-250;
    CheckSum=256-rem(Check,256);
    if (CheckSum==256)
            CheckSum= '00';
        elseif (CheckSum==0)
            CheckSum= '00';
        elseif (CheckSum==1)
            CheckSum= '01';
        elseif (CheckSum==2)
            CheckSum= '02';
        elseif (CheckSum==3)
            CheckSum= '03';
```

```
        elseif (CheckSum==4)
            CheckSum= '04';
        elseif (CheckSum==5)
            CheckSum= '05';
        elseif (CheckSum==6)
            CheckSum= '06';
        elseif (CheckSum==7)
            CheckSum= '07';
        elseif (CheckSum==8)
            CheckSum= '08';
        elseif (CheckSum==9)
            CheckSum= '09';
        elseif (CheckSum==10)
            CheckSum= '0A';
        elseif (CheckSum==11)
            CheckSum= '0B';
        elseif (CheckSum==12)
            CheckSum= '0C';
        elseif (CheckSum==13)
            CheckSum= '0D';
        elseif (CheckSum==14)
            CheckSum= '0E';
        elseif (CheckSum==15)
            CheckSum= '0F';
        else
            CheckSum= dec2hex(CheckSum);
        end
    CheckSum;
    %CheckSum= '00';
    DataToSend
=['FA';'FF';'32';'49';AccX_1;AccX_2;AccX_3;AccX_4;AccY_1;AccY_2;AccY_3;Ac
cY_4;AccZ_1;AccZ_2;AccZ_3;AccZ_4;GyrX_1;GyrX_2;GyrX_3;GyrX_4;GyrY_1;GyrY_
2;GyrY_3;GyrY_4;GyrZ_1;GyrZ_2;GyrZ_3;GyrZ_4;MagX_1;MagX_2;MagX_3;MagX_4;M
agY_1;MagY_2;MagY_3;MagY_4;MagZ_1;MagZ_2;MagZ_3;MagZ_4;Roll_1;Roll_2;Roll
_3;Roll_4;Pitch_1;Pitch_2;Pitch_3;Pitch_4;Yaw_1;Yaw_2;Yaw_3;Yaw_4;Lat_1;L
at_2;Lat_3;Lat_4;Lon_1;Lon_2;Lon_3;Lon_4;Alt_1;Alt_2;Alt_3;Alt_4;VelX_1;V
elX_2;VelX_3;VelX_4;VelY_1;VelY_2;VelY_3;VelY_4;VelZ_1;VelZ_2;VelZ_3;VelZ
_4;Status;CheckSum];
    DataDec=hex2dec(DataToSend);
% DataDec;
y = DataDec;

.
```

Anexo II Bloques Simulink con subsistemas.