Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Automática y Sistemas Computacionales

TRABAJO DE DIPLOMA

Implementación del controlador *I-LOS* en lenguaje *C* para el seguimiento de caminos rectos en el *HRC-AUV*.

Tesis presentada en opción al grado de
Ingeniero en Automática

Autor: José Daniel Fernández Peña

Tutor: Ing. Anailys Hernández Julián
Ing. Hector Daniel Alvarez Pérez

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

Implementación del controlador *I-LOS* en lenguaje *C* para el seguimiento de caminos rectos en el *HRC-AUV*.

Tesis presentada en opción al grado de
Ingeniero en Automática

Autor: José Daniel Fernández Peña
email: jfernandez@uclv.cu

Tutor: Ing. Anailys Hernández Julián
Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV
email: anailyshj@uclv.cu

Ing. Hector Daniel Alvarez Pérez
Dpto. de Automática, Facultad de Ing. Eléctrica, UCLV
email: hectorap@uclv.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 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.

José Daniel Fernández Peña — Fecha
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.

José Daniel Fernández Peña — Fecha
Autor

Iván Santana Ching, Dr.C — Fecha
Jefe del Departamento

Responsable ICT o J' de Carrera, (Dr.C., M.Sc. o Ing.) — Fecha
Responsable de Información Científico-Técnica

PENSAMIENTO

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa".

Mahatma Gandhi.

DEDICATORIA

A mis padres José Luis e Isbel,
por ser los responsables de mi formación y educación.

A mi tía Herminia,
por todo el amor y la confianza que me ha dado.

A mi hermano Luisi,
por permitirme ser su ejemplo.

A mi familia,
por brindarme su apoyo incondicional y su cariño.

A mis amigos,
por compartir conmigo en los buenos y malos momentos.

AGRADECIMIENTOS

A mis tutores Anailys y Hector, por su tiempo, paciencia, disposición y ayuda incondicional en la elaboración de este trabajo.
A Yunier por creer en mí y motivarme a dar mi mejor esfuerzo en todo momento.
A Erick por enseñarme a "aprender a aprender".
A Boris por su apoyo y todos sus consejos.
A ellos, un eterno agradecimiento.

A todos los profesores que de una forma u otra durante la carrera han intervenido en mi formación intelectual y profesional.

A todos mis amigos y compañeros de estudio, por compartir juntos estos 5 años.
A Alejandro, Dizahab, Dianelis, Leidis, Charlie, Rogelio, Julio, Héctor y Nelson.

A todas aquellas personas que me brindaron su ayuda en la realización de este trabajo de diploma.

Santa Clara, Cuba, 2016

RESUMEN

El tema de los vehículos autónomos subacuáticos es de gran interés para la comunidad científica nacional e internacional. El sistema de guiado de un AUV es importante en el cumplimiento de misiones sin intervención humana. En esta investigación se implementa un controlador de tipo PI en lenguaje C, permitiendo su inclusión en el software de navegación del $HRC\text{-}AUV$ para corroborar su desempeño en pruebas experimentales en el mar. Se hace necesario realizar algunas modificaciones en el sistema de software del vehículo para garantizar una adecuada integración del algoritmo de guiado. La validez del código en lenguaje C se constata mediante co-simulación con el modelo obtenido analíticamente, mostrando una convergencia adecuada en los resultados obtenidos.

TABLA DE CONTENIDO

Página

ÍNDICE DE FIGURAS

Figura — Página

INTRODUCCIÓN

Los *AUV* (*Autonomous Underwater Vehicles*) son vehículos que poseen el control de sí mismos y su propia fuente de alimentación a bordo. Estos establecen comunicación con la superficie sólo mediante señales inalámbricas (Fossen, 2011). El ámbito de aplicación de un *AUV* es muy amplio. Debido a su capacidad de operar en ambientes hostiles puede ser utilizado en misiones de inspección, mantenimiento, reparación, monitoreo, transporte, observación y búsquedas subacuáticas. El desarrollo de estos vehículos constituye actualmente una de las metas más importantes de las industrias petrolera, militar y de las comunicaciones, así como de áreas de la ciencia tales como la oceanografía y la biodiversidad marina.

Las investigaciones que están asociadas a este tipo de vehículos se encuentran dirigidas al desarrollo de las capacidades de autonomía, navegación, sensores y sistemas de comunicación (Fossen, 2011; Breivik, 2010; Valeriano Medina, 2013; Lemus, 2011). Con ese propósito, importantes universidades y centros de investigación de todo el mundo llevan a cabo proyectos de desarrollo, tales como Theseus (Ferguson, 1999), ABE XBenthic (Butler, 1993; Blidberg, 2001) y Odyssey (Eskesen, 2009). En el caso de Cuba, el proyecto *HRC-AUV* que llevan a cabo el Grupo de Automática, Robótica y Percepción (GARP) y el Centro de Investigación y Desarrollo Naval (CIDNAV) se presenta como un primer intento por desarrollar la tecnología de los *AUV* en el ámbito nacional.

Este proyecto se desarrolla con fines científicos para su posterior aplicación en la exploración del ambiente marino y el reconocimiento de las costas cubanas. En él se integran diferentes líneas de desarrollo, entre las cuales destacan los sistemas de navegación, control y guiado.

El grupo de investigación GARP ha logrado una evolución en el seguimiento de caminos para el *HRC-AUV* usando *LOS* como estrategia de guiado. En un primer momento se propone determinar el ángulo de rumbo deseado a partir de las coordenadas de los puntos de la ruta. Esta propuesta no garantiza la convergencia del vehículo al camino de forma adecuada ante los efectos causados por las corrientes marinas. Como continuación de esta investigación se plantea mantener *LOS* como estrategia de guiado, utilizando la ley de dirección basada en la distancia *lookahead* y la velocidad del vehículo (Lemus, 2011). Esta estrategia reduce el efecto causado por las perturbaciones marinas durante

el desplazamiento del vehículo, pero tiene como limitante la necesidad de contar con mediciones fiables de la velocidad del vehículo, las cuales no pueden ser obtenidas mediante el sistema de sensores de bajo costo del *HRC-AUV*. Ante este inconveniente se propone como solución la estrategia *I-LOS* que asegura la convergencia del vehículo al camino y reduce el efecto causado por las corrientes marinas, tanto para caminos rectos como curvos (Valeriano Medina, 2015; Fernández, 2015).

El tema de guiado constituye un tópico que por su importancia ha sido ampliamente abordado por GARP en sus investigaciones. Específicamente, el tema de la reducción de los errores de seguimiento que provocan las corrientes marinas ha estado en el centro de atención. En el trabajo de Anailys Hernández (Hernández Julián, 2014), se propone utilizar un controlador *I-LOS* para lograr el seguimiento de caminos rectos. Los resultados en simulación han demostrado que este controlador logra un buen desempeño. Sin embargo, esta estrategia no ha podido ser probada en maniobras en el mar, debido a que no se cuenta con una implementación de la misma que pueda ejecutarse en el software de navegación instalado en el vehículo. Es por ello que en el caso de esta investigación, se plantea:

Problema científico: No se dispone de una implementación de la estrategia de guiado *I-LOS* que pueda ser ejecutada en el software de navegación instalado a bordo del vehículo *HRC-AUV*.

Hipótesis: *La implementación en lenguaje C de la estrategia de guiado I-LOS permite su inclusión en el software de navegación del vehículo HRC-AUV, para corroborar su desempeño durante la realización de maniobras de seguimiento de caminos rectos.*

Es por ello que se proponen los siguientes objetivos:

Objetivo general: Implementar en lenguaje *C* la estrategia de guiado *I-LOS* respetando la programación realizada en el software de navegación del *HRC-AUV*.

Objetivos específicos:

1. Estudiar los aspectos teóricos relacionados con la temática que aparecen reportados en la literatura.
2. Estudiar la arquitectura y la estructura del software de navegación del *HRC-AUV*.
3. Programar el algoritmo de guiado *I-LOS* en lenguaje *C*.
4. Constatar la validez del algoritmo mediante simulación.

Para cumplir con los objetivos del trabajo se plantean las siguientes tareas investigativas:

- Revisión de los aspectos teóricos relacionados con la temática que aparecen reportados en la literatura.
- Estudio de la arquitectura de clases y el estilo de programación del software de navegación del *HRC-AUV*.
- Estudio del controlador *I-LOS* diseñado para el *HRC-AUV*.
- Programación del controlador *I-LOS* en lenguaje *C*.
- Simulación del código implementado.
- Comparación de los resultados obtenidos con la implementación realizada en *Simulink* de *MATLAB*.
- Realización de cambios en los software de supervisión y navegación para integrar el nuevo código.
- Elaboración del informe científico de la investigación.

Los resultados de la investigación brindan un aporte, desde el punto de vista teórico y práctico, de gran trascendencia para los vehículos subacuáticos que está desarrollando el CIDNAV. El análisis del desempeño del controlador *I-LOS* implementado para el seguimiento de caminos rectos, podría ser evaluado en el *HRC-AUV* durante pruebas experimentales con el fin de obtener la mejor solución de control posible para la etapa de desarrollo en la que se encuentra el proyecto.

La investigación incluye tres capítulos, además de las conclusiones, recomendaciones, referencias bibliográficas y anexos correspondientes. Los temas que se abordan en cada capítulo se encuentran estructurados de la forma siguiente:

Capítulo I: A partir de un estudio de la bibliografía especializada, se efectúa un análisis de los principales aspectos relacionados con el diseño del controlador *I-LOS* y de los software utilizados en el mundo que se han desarrollado para los *AUV*. Además, se presenta una descripción general del vehículo autónomo subacuático *HRC-AUV*.

Capítulo II: En este capítulo se describe la estructura del sistema de software en el *HRC-AUV*, especificando las principales características de la estación a bordo y la estación remota. Se presentan además, las ecuaciones utilizadas en la implementación del controlador *I-LOS* en lenguaje *C*.

Capítulo III: Se demuestra mediante simulación el funcionamiento y desempeño del algoritmo del controlador *I-LOS* ya programado en lenguaje *C*. Se proponen modificaciones a realizar en el software de supervisión con vistas a incluir la opción del controlador *I-LOS*. Además, se presenta la valoración económica.

CAPÍTULO 1
ESTUDIO DE LOS SISTEMAS DE SOFTWARE QUE SE EMPLEAN EN VEHÍCULOS AUTÓNOMOS

1.1. Introducción

En este capítulo se efectúa un análisis de los principales aspectos relacionados con el guiado de vehículos marinos y el controlador *I-LOS*, a partir de un estudio de la bibliografía especializada. Adicionalmente se presentan ejemplos de *AUV*, así como los software que se utilizan en este tipo de vehículos. Además, se presenta una descripción general del vehículo autónomo subacuático *HRC-AUV* y de la arquitectura de hardware y software que lo compone.

1.2. Ejemplos y aplicaciones de vehículos autónomos submarinos

La amplia gama de aplicaciones que poseen los vehículos autónomos submarinos deja en evidencia la importancia que representa su uso para varias industrias y áreas de investigación, tal como se ha expuesto anteriormente. Varias entidades han logrado mantener un desarrollo creciente de este tipo de tecnología, enfocados en aspectos tales como las capacidades de autonomía, navegación, sensores y sistemas de comunicación. Con ese propósito se llevan a cabo proyectos de desarrollo, algunos de los cuales se presentan a continuación.

1.2.1. *AUV Theseus*

El desarrollo del *AUV Theseus* comenzó en 1992 por la *ISE* (*International Submarine Engineering Ltd.*) como parte del proyecto conjunto entre Estados Unidos y Canadá conocido como *Spinnaker*. En la Figura 1–1 se muestra el vehículo, el cual tiene una longitud de 10.7 m, un diámetro de 1.27 m, alcanza una velocidad crucero de 2.0576 m/s y puede operar hasta una profundidad de 1000 m. Este vehículo autónomo fue desarrollado originalmente para la instalación de grandes longitudes de cable de fibra óptica bajo la capa de hielo del Ártico (Butler, 1993). El vehículo completó misiones exitosas en el Ártico en 1995 y 1996.

El funcionamiento global del vehículo se maneja por una computadora MC68030 que mantiene ejecutando en tiempo real un *kernel* desarrollado por *ISE* para el control de *AUV*. Este *kernel* está implementado en C++, en un modelo orientado a objetos basado en una arquitectura por capas. Una de las características fundamentales de este *kernel* es la habilidad de reconfigurar el sistema de control sin modificar el código fuente. Para aumentar la tolerancia a fallos, el sistema de control maneja las respuestas a estos eventos usando una tabla de fallos predefinida. Dicha tabla permite al usuario dividir la misión en una serie de fases, donde cada fase consiste en una o más maniobras entre los puntos de la ruta (Ferguson, 1999).

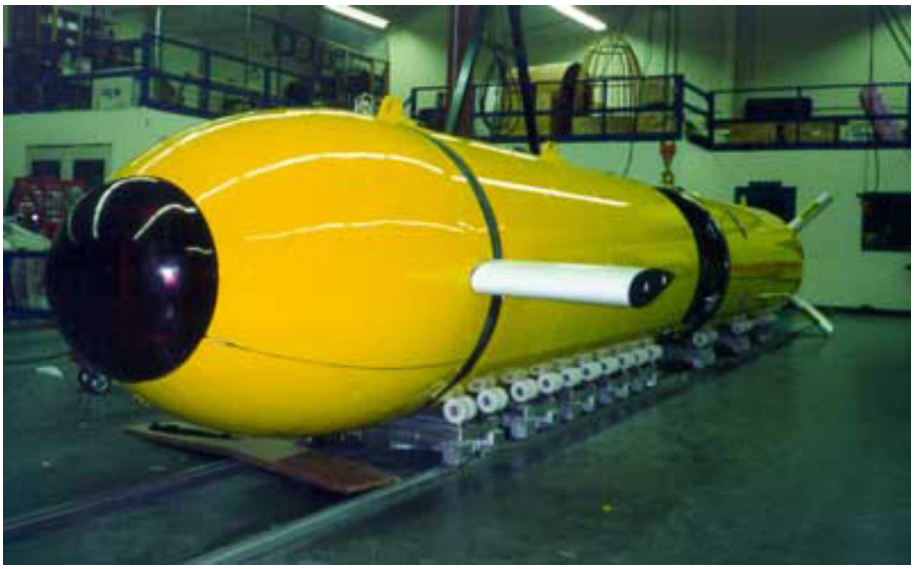

Figura 1–1: *Theseus AUV*

1.2.2. *AUV* Pirajuba

El proyecto Pirajuba está enfocado al desarrollo de un AUV de bajo costo para investigaciones sobre la dinámica y navegación de esta clase de vehículos. El mismo es un vehículo autónomo de tipo crucero diseñado para tener autonomía por cuatro horas a una velocidad de 2 m/s. Posee una longitud de 1.742 m y un diámetro de 0.234 m. Este AUV que se aprecia en la Figura 1–2, fue concebido para apoyar el estudio sobre estimaciones de los derivados hidrodinámicos. Además, es utilizado como plataforma para investigaciones académicas en el campo de la tecnología subacuática.

El software utilizado en Pirajuba tiene una arquitectura compuesta por capas. El mismo incluye un conjunto de librerías en lenguaje C, siguiendo los conceptos de programación estructurada. El sistema operativo que se ejecuta en el vehículo es el $\mu C - OSII$, el cual es una sistema de poca complejidad.

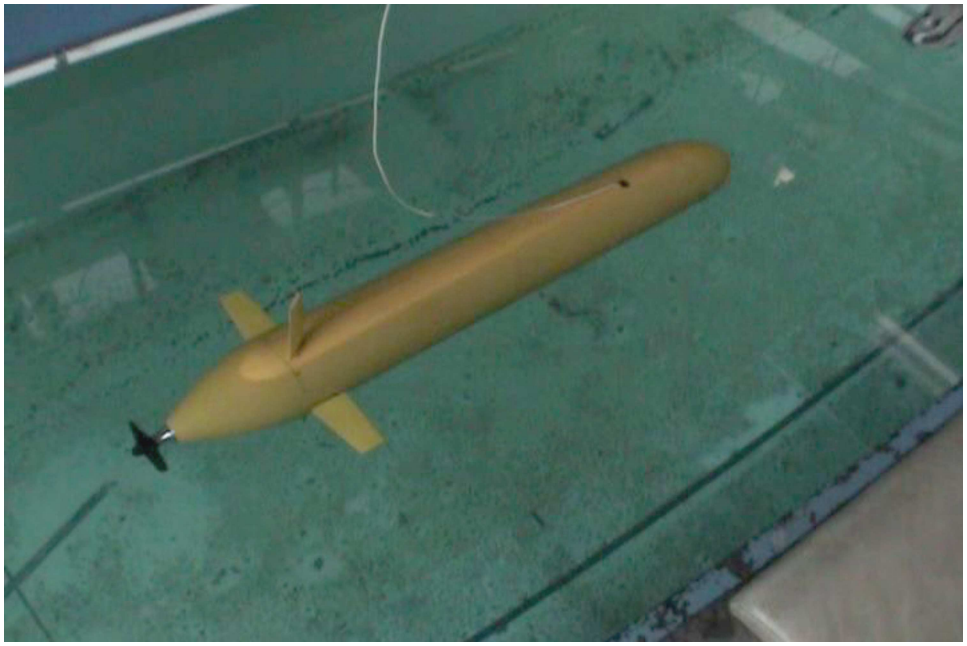

Figura 1–2: Pirajuba *AUV*

1.2.3. *AUV Odyssey* IV

El *AUV Odyssey* IV mostrado en la Figura 1–3, fue desarrollado para satisfacer las necesidades de la industria y la investigación. Este *AUV* está diseñado para rechazar las corrientes en el ambiente del océano abierto y puede ser desplegado desde un pequeño barco de pesca. Además, es capaz de operar hasta profundidades de 6000 m.

El vehículo tiene cuatro propulsores ubicados en la proa: dos laterales con empuje vectorial y dos con empuje lateral. Estos dispositivos permiten controlar cuatro grados de libertad (GDL), lo cual aumenta la maniobrabilidad y la precisión durante las misiones. La existencia de un área de carga útil y adaptable permite el montaje de sensores, actuadores u otro hardware adecuado para misiones de carácter específico.

Las variadas aplicaciones del *Odyssey* IV incluyen el estudio y la inspección de los corales de agua fría, sitios arqueológicos e infraestructura submarina. También puede ser utilizado como una plataforma de investigación en servomecanismos basada en la visión informática y el control acústico de supervisión.

En el hardware de cómputo del vehículo se ejecuta el software *MOOS (Mission Oriented Operanting Suite)*, el cual es una recopilación de aplicaciones y librerías de código abierto para operaciones con vehículos autónomos. Las operaciones necesarias para poner en funcionamiento el vehículo tales como navegación, control, interfaz de instrumentación, autentificación e interfaz de usuario se encuentran en procesos separados. Todos los procesos intermedios de comunicación están hechos a través *MOOSDB*, contenido en el servidor central. Además de *MOOSDB*, el *Odyssey* IV utiliza un filtro de navegación

proporcionado por *MOOS*, llamado *pNav*, el cual es capaz de obtener datos de los sensores de navegación y calcular su posición a través del uso de un Filtro de Kalman Extendido (*Extended Kalman Filter, EKF*) (Eskesen, 2009).

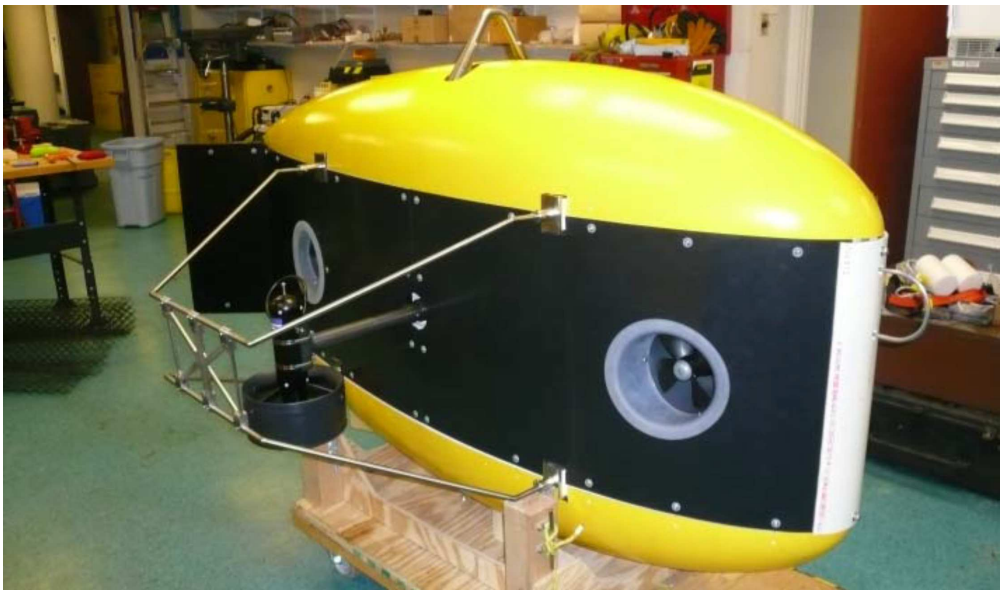

Figura 1–3: *Odyssey IV AUV*

1.3. Estudio de las estrategias de guiado utilizadas en vehículos marinos

La autonomía de un *AUV* en el cumplimiento de misiones se logra a partir de un sistema de control de movimiento, el cual está compuesto por: un sistema de guiado, un sistema de control y un sistema de navegación, como se aprecia en la Figura 1–4.

Las funciones que cumplen cada uno de estos sistemas son (Fossen, 2011; Lekkas, 2014):

Sistema de navegación: se encarga de sensar y estimar los valores de variables tales como la posición, la velocidad, la aceleración del vehículo y el curso, así como de la orientación.

Sistema de guiado: utiliza los valores proporcionados por el sistema de navegación para calcular los parámetros de posición, velocidad y aceleración deseados que se envían a los controladores.

Sistema de control: tomando en cuenta los parámetros proporcionados por el sistema de guiado y las perturbaciones a las cuáles se encuentra sometido el vehículo, establece el rumbo del vehículo logrando que tome la trayectoria adecuada, según los puntos de ruta establecidos y cumpla su misión de manera estable.

Aunque el control de movimiento está determinado por los anteriores sistemas en su conjunto, cada uno constituye una línea de desarrollo independiente. Esta investigación tributa al desarrollo del sistema de guiado.

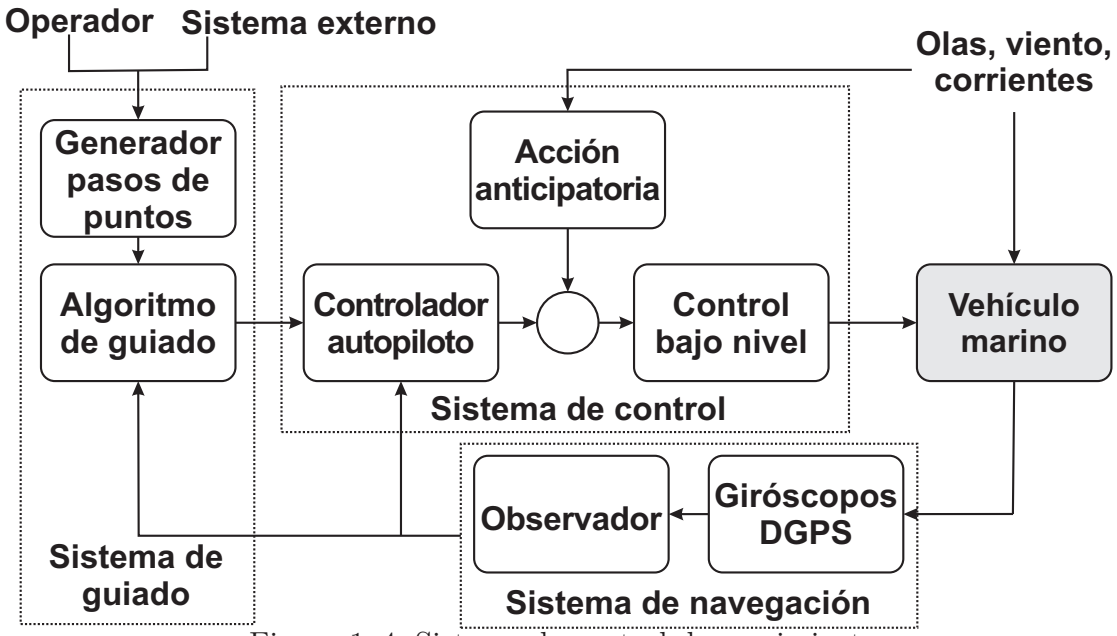

Figura 1–4: Sistema de control de movimiento

El concepto de guiado se define como el proceso para guiar la trayectoria de un objeto hacia un punto dado o camino determinado, el punto u objetivo a alcanzar puede estar en movimiento (Breivik, 2008). Según los puntos de ruta que se seleccionen, se establece la trayectoria que seguirá el vehículo determinando su movimiento en el espacio (Lekkas, 2014).

Para el control de movimiento pueden presentarse diferentes escenarios: (Breivik, 2010)

Seguimiento de un objetivo (*Target Tracking, TT*): en este escenario el objetivo de control involucra el seguimiento de un objetivo, el cual puede encontrarse en reposo (estabilización de punto) o en movimiento, de forma tal que sólo su movimiento instantáneo es conocido. En cualquiera de estos casos las restricciones espacio-temporales son imposibles de separar.

Seguimiento de camino (*Path Following, PF*): en este escenario se precisa converger hacia una ruta predefinida y seguirla. Aunque en este caso no interesan las restricciones temporales, siguen presentes las limitaciones espaciales.

Seguimiento de un objetivo a lo largo de una trayectoria (*Path Tracking, PT*): este escenario implica el seguimiento de un objetivo que debe moverse dentro de una trayectoria predefinida. Para este escenario es posible separar en dos restricciones las limitaciones espacio-temporales asociadas a la posición del objetivo.

Maniobrabilidad del camino (*Path Maneuvering, PM*): este escenario se caracteriza por el empleo de los conocimientos sobre las limitaciones de maniobrabilidad del vehículo para optimizar el seguimiento de una trayectoria. En estas circunstancias se consideran las restricciones espaciales con mayor prioridad que las temporales.

Varios de los conceptos acerca del guiado que actualmente se usan proceden de la industria armamentista, específicamente la dedicada al estudio de misiles (Breivik, 2007). En cuanto al seguimiento de objetivos aparecen dos conceptos fundamentales: interceptor y objetivo. Un interceptor o perseguidor es un objeto que debe interceptar a otro que se define como objetivo.

El interceptor típicamente transcurre por tres fases durante su operación: fase de lanzamiento, fase de medio recorrido, y fase final. La mayor exactitud se requiere en la fase final, donde el sistema de guiado del interceptor debe compensar los errores acumulados de las fases anteriores para lograr el menor error posible con respecto al objetivo (Breivik, 2007; Hernández Julián, 2014). Para ello se aplican en la fase final estrategias de guiado para el seguimiento de camino, como se muestra en la Figura 1–5:

Línea de visión (*Line of Sight, LOS*): se caracteriza por ser un esquema de guiado de tres puntos, ya que involucra un punto de referencia además del interceptor y el objetivo. La denotación *LOS* se refiere al hecho de que el interceptor debe dirigirse hacia el punto de intercepción entre un vector *LOS* y el camino que se forma entre la referencia y el objetivo.

Persecución (*Pure Pursuit, PP*): es un esquema de guiado de dos puntos en el cual se consideran solamente el interceptor y el objetivo. En este caso el interceptor modifica su velocidad a medida que se acerca al objetivo.

Dirección constante (*Constant Bearing, CB*): es también un esquema de guiado de dos puntos , el interceptor y el objetivo. En este caso la velocidad del interceptor debe quedar alineada en dirección a la velocidad relativa interceptor-objetivo.

Para la implementación de las estrategias de guiado es importante contar con leyes o algoritmos. Estas leyes pueden ser de velocidad y/o de dirección, las cuales pueden combinarse de varias maneras para alcanzar diferentes objetivos de control de movimiento (Breivik, 2010; Hernández Julián, 2014).

El grupo de investigación GARP enfoca especial interés en *LOS* para el seguimiento de caminos rectos. Se presentan dos leyes de guiado: la ley de dirección basada en el encierro circular del camino y la ley de dirección basada en la distancia *lookahead*. Estas leyes tienen como fin orientar al vehículo hacia un punto de intersección con la ruta deseada, la cual se encuentra definida por la secuencia en la que están ordenados los puntos de la ruta. En este caso la ley de dirección basada en la distancia *lookahead* tiene menor complejidad matemática, resultando más sencilla su implementación (Breivik, 2010); por lo cual resulta la ley de guiado escogida. La distancia *lookahead* está definida como la distancia desde la proyección de la posición del vehículo sobre el camino hasta el punto de intersección del vector *LOS* con el camino (Lekkas, 2014).

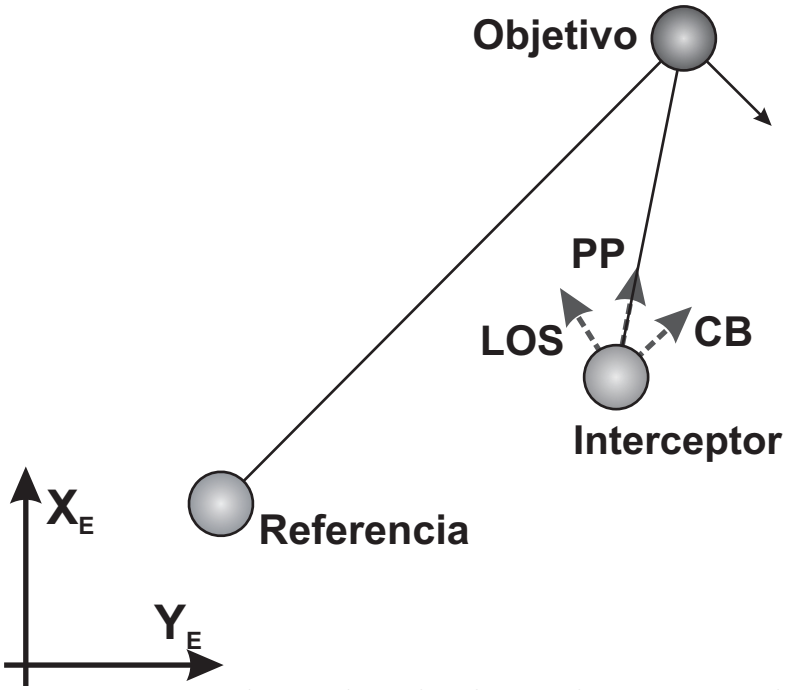

Figura 1–5: Estrategias de guiado utilizadas en el seguimiento de camino

A partir de dicha ley de guiado se pueden utilizar dos métodos para disminuir la influencia de las perturbaciones durante el desplazamiento del vehículo. El primer método consiste en la compensación del ángulo de deslizamiento, a partir de la medición de la velocidad relativa del vehículo. Esta variante es capaz de reducir el efecto causado por las corrientes marinas, pero tiene como limitante que para ello se necesita contar con mediciones fiables de la velocidad del vehículo. Otra alternativa consiste en utilizar un controlador con acción integral, tomando el ángulo de deslizamiento lateral como una perturbación pequeña de poca variación. Este método asegura la convergencia del vehículo al camino y reduce el efecto causado por las perturbaciones marinas, tanto para caminos curvos como rectos (Valeriano Medina, 2015; Hernández Julián, 2014; Fernández, 2015).

1.4. Ejemplos de software profesionales utilizados en el autopiloto de un *AUV*

Los recientes avances en sensores, comunicaciones, computadoras e inteligencia artificial han hecho posible el diseño de AUV más sofisticados. El autopiloto de un AUV es el encargado de mantener una profundidad y un rumbo predeterminados durante el cumplimiento de misiones (Martínez, 2013). El complejo sistema de un vehículo robótico requiere una arquitectura de software de control avanzada a bordo, que sea capaz de interactuar con una amplia gama de sensores y actuadores para llevar a cabo distintas misiones; teniendo en cuenta posibles problemas como controlabilidad, estabilidad, tiempo de respuesta, comunicación, manejo de datos y modularidad. A continuación se

presenta la arquitectura de software de los vehículos autónomos en desarrollo *SAUVIM* y *STARFISH*.

1.4.1. Arquitectura de software del *SAUVIM*

En el Laboratorio de Sistemas Autónomos de la Universidad de Hawaii se desarrolla un vehículo submarino semiautónomo para misiones de intervención (*Semi-Autonomous Underwater Vehicle for Interventions Missions, SAUVIM*) (Kim, 2003). Este vehículo presenta un largo de 6.1 m y una masa de 6100 kg, alcanza una velocidad crucero de hasta 1.5432 m/s y una profundidad de 6000 m.

La arquitectura de software del *SAUVIM*, *SDBCA* (*Sensor Data Bus based Control Architecture*) es una estructura modular y flexible que consta de tres capas: capa de aplicación, capa de tiempo real y capa de dispositivo, como se aprecia en la Figura 1–6. Cada módulo sigue sus propias especificaciones para mantener la arquitectura software/control.

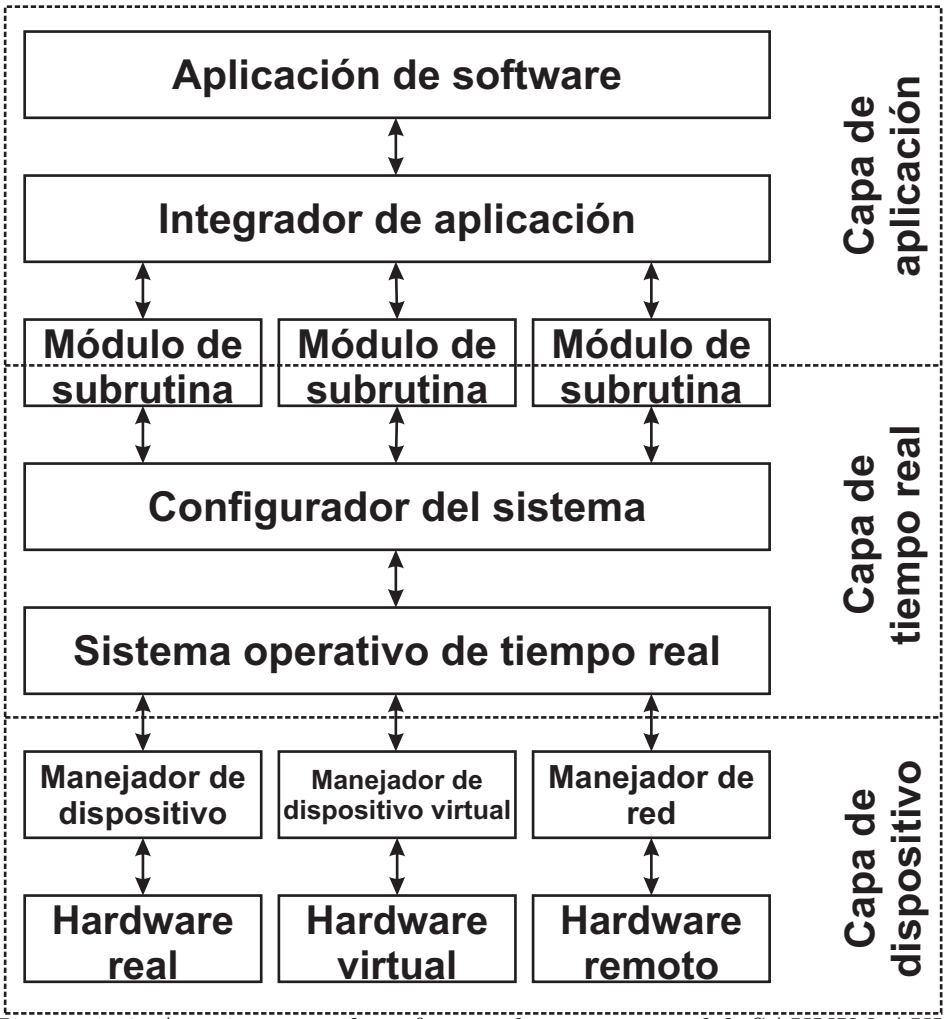

Figura 1–6: Arquitectura de software de tres capas del *SAUVIM AUV*

La capa de aplicación está compuesta por un software de aplicación, un integrador de aplicación y varios módulos de subrutina. El software de aplicación incluye un hardware independiente con módulos de alto nivel como interfaz de usuario, un intérprete para lenguaje de descripción de rutinas y un algoritmo de control supervisorio *SAUVIM*. El integrador de aplicación está a cargo de la conexión entre el software de aplicación y los módulos de subrutina usando un método de manejo de tareas, que incluye la creación y borrado de tareas, así como la comunicación y sincronización entre las mismas.

La capa de tiempo real está formada por un configurador de sistema, un sistema operativo de tiempo real y algunas partes de los módulos de subrutina. La función principal del configurador de sistema es el mapeo entre el software de aplicación del hardware independiente y los módulos de hardware relacionados.

La capa de aplicación es la única parte dependiente del hardware. La mayoría de los módulos de software en esta capa están directamente conectados al hardware para enviar datos de comando para los actuadores y obtener datos de los sensores.

En la arquitectura descrita anteriormente puede verificarse la controlabilidad y estabilidad del sistema. Sin embargo, presenta poca flexibilidad, largo tiempo de respuesta, y dificultad en la integración/fusión del sensor, debido a la comunicación indirecta entre el control de alto nivel y los periféricos de bajo nivel. Para sobreponerse a estas dificultades, un *sensor data bus* (*SBD*) es usado para suplir un canal de comunicación directo entre las capas de bajo, medio y alto nivel (Kim, 2003).

1.4.2. Arquitectura de software del *STARFISH*

Como una iniciativa del Laboratorio de Investigación Acústica de la Universidad Nacional de Singapur se ha desarrollado una arquitectura abierta para el prototipo colaborativo de *AUV* conocido como *STARFISH* (Sangekar, 2008). Los componentes de software en este *AUV* están basados en la Arquitectura de Software Distribuida para Vehículos Autónomos (*Distributed Software Architecture for Autonomous Vehicles, DSAAV*).

Esta implementación está basada en una arquitectura distribuida, pues divide la carga y el tráfico entre todos los procesadores del *AUV* y evita altas cargas y dependencia en un mismo procesador y las bases de datos. Sin embargo, un sistema centralizado es fácil de administrar y monitorear; por esta razón, *DSAAV* provee una base de datos de configuración del sistema y un servicio de registro.

En esta arquitectura cada módulo provee una interfaz de software al cual otros módulos del *AUV* pueden acceder. Dicha interfaz permite la configuración de módulos, registro de información crítica, monitoreo y funcionalidades de actualización de software.

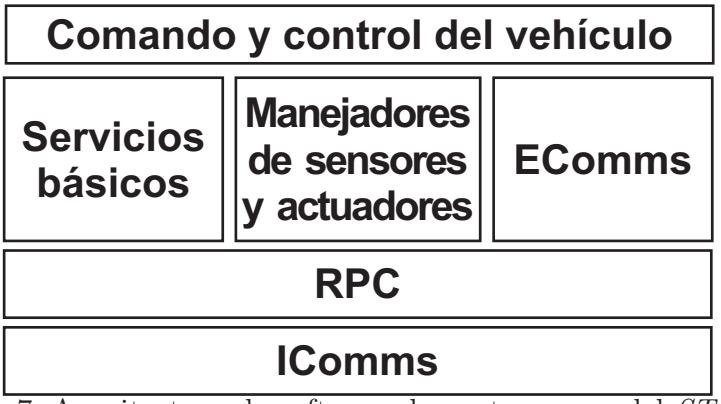

Figura 1–7: Arquitectura de software de cuatro capas del *STARFISH*

La implementación de *DSAAV* es una arquitectura de cuatro capas, como se muestra en la Figura 1–7, escrita en *C++* y compilada para Linux (Chitre, 2008). La capa más baja es la de comunicación (*IComms*), la cual provee la implementación de un servicio de mensajería no fidedigno sobre la comunicación principal disponible. La siguiente capa es *RPC* (*Remote Procedure Call*), en la que se implementa una llamada de procedimiento remoto usando el servicio de mensajería de la capa *IComms*. La tercera capa incluye servicios básicos para la configuración del vehículo tales como registro de eventos y monitoreo, además incluye manejadores de hardware para todos los sensores y actuadores, así como una interfaz externa de comunicación (*EComms*) para la comunicación con otros vehículos y/o con el centro de control. La capa más alta alberga componentes de comando y control que utilizan los servicios provistos por las capas más bajas para completar la misión del vehículo.

1.5. Descripción general del *HRC-AUV*

Los *AUV* constituyen una tecnología en progreso que brinda toda una nueva gama de posibilidades. El desarrollo nacional de este tipo de vehículos contribuiría al acceso de incontables recursos existentes alrededor del archipiélago nacional.

El diseño mecánico y la construcción naval del *HRC-AUV* ha sido desarrollado por CIDNAV. Como resultado del mismo, se cuenta con un vehículo de forma cilíndrica de aproximadamente 0.8 m de diámetro, 9.5 m de largo y una masa de 4096 kg. Este sumergible puede operar hasta 10 m de profundidad con una velocidad crucero de 1.9 m/s gracias a un sistema de actuadores compuesto por una hélice y dos timones, los cuales son manejados eléctricamente (Rodríguez, 2011). Su estructura es similar al *HUGIN* 4500 (Hegrenaes, 2007) y al *STARFISH* (Sangekar, 2008).

El GARP, por su parte, se ha dado a la tarea de diseñar la arquitectura de hardware y el sistema sensorial del *HRC-AUV*. La arquitectura de hardware del vehículo mostrada en la Figura 1–8, se compone por unidades de control digital, elementos sensoriales, sistemas de mando y sistemas de supervisión y comunicaciones. El vehículo posee una estación de cómputo a bordo que intercambia datos, mediante una comunicación inalámbrica, con una estación en tierra firme.

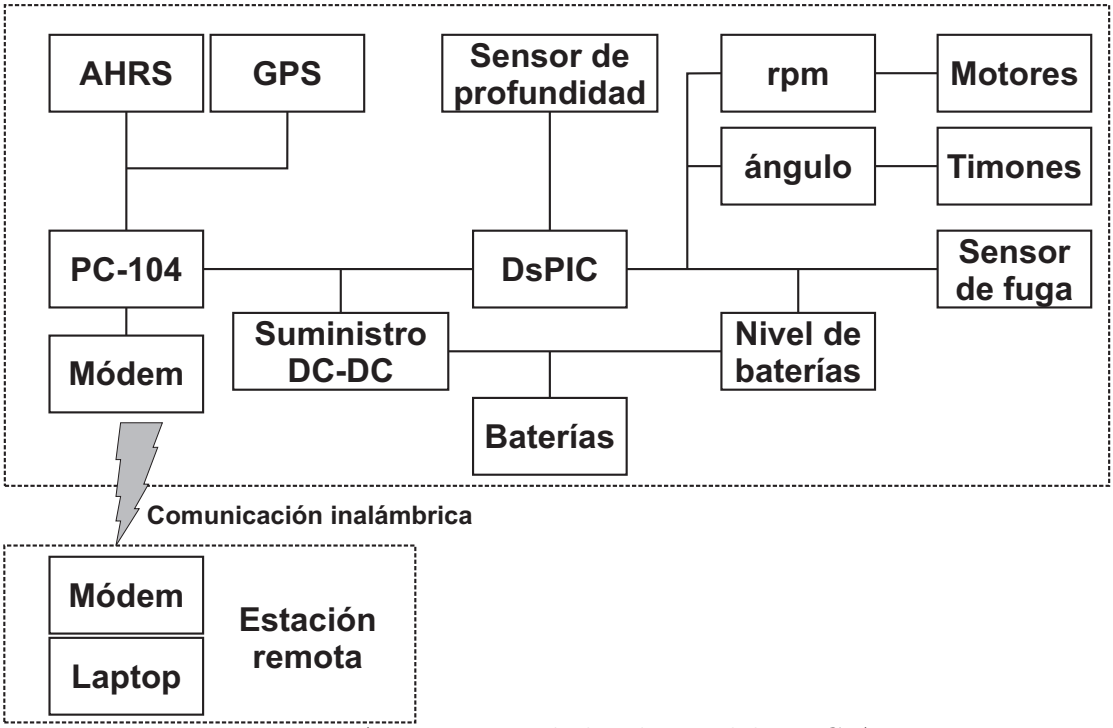

Figura 1–8: Arquitectura de hardware del *HRC-AUV*

La estación en tierra firme consiste en una laptop que ejecuta un sistema remoto de supervisión y configuración del autopiloto. Esta estación se encuentra conectada a un radio enlace, permitiendo el intercambio de datos durante las misiones mientras el vehículo está en la superficie, y un dispositivo de red inalámbrica o punto de acceso para establecer cierta redundancia en la comunicación.

La estación a bordo del vehículo cuenta con un radio enlace y un punto de acceso para comunicación con la laptop en la superficie. Posee también una computadora industrial tipo PC-104 encargada de la adquisición de los valores digitales de inercia y posición provenientes de los sensores, del manejo de la comunicación y además de la ejecución del software de navegación y control. La estación consta también de un *DsPIC* 30F4013 que se encuentra conectado a la PC industrial utilizando comunicación serie. Este microcontrolador actúa como colector y procesador de la información sensorial primaria (Rodríguez, 2011).

La programación está implementada en lenguaje *C*, además le fueron agregadas librerías optimizadas, de las cuales se hace uso en el transcurso del programa. El compilador empleado es el C30 que es soportado por el software de alto nivel *MPLAB* (Guerra, 2010).

La programación del software de navegación del vehículo está hecha en lenguaje *C*, que al ser un lenguaje de bajo nivel permite el control directo del hardware, y constituye uno de los lenguajes más adecuados en la implementación de aplicaciones en tiempo real. Este software incluye funciones de lectura de los sensores, acondicionamiento de los datos, navegación inercial y guiado del vehículo (Lemus, 2011).

En la estación en tierra se ejecuta un supervisorio, llamado *SharkSoft*, que tiene el propósito de supervisar y configurar remotamente el autopiloto del *HRC-AUV*. Este sistema muestra, mediante una interfaz de usuario apropiada, toda la información sensorial relevante. Esta información está relacionada con la posición espacial del vehículo, los ángulos de cabeceo y balanceo que experimenta, así como su rumbo, profundidad, velocidad y su estado (presencia de agua dentro del casco y motor propulsor encendido o apagado). El servicio de configuración del autopiloto comprende el ajuste de los controladores de rumbo y profundidad. El sistema incorpora gráficas en la interfaz de usuario que sirven para evaluar el desempeño de los controladores. Además, cuenta con una pantalla que permite la gestión de misiones a través de mapas georreferenciados (Rodríguez, 2011). La aplicación fue implementada en *C#*. Este lenguaje constituye una versión avanzada de *C* y *C++*. *C#* es una solución ideal para aplicaciones de alto nivel pues mezcla la potencia de *C*, las capacidades de orientación a objetos de *C++* y la interfaz gráfica de *Visual Basic* (Arora, 2002).

El grupo de investigación GARP ha logrado una evolución en el seguimiento de caminos para el *HRC-AUV* usando *LOS* como estrategia de guiado. Como primera variante se prueba determinar el ángulo de rumbo deseado a partir de las coordenadas de los puntos de la ruta, dicha propuesta fue implementada en el software de navegación del vehículo y probada en maniobras reales evidenciando que no garantizaba la convergencia del vehículo al camino de forma adecuada ante los efectos causados por las corrientes marinas. En un segundo momento se plantea mantener *LOS* como estrategia de guiado, pero utilizando la ley de dirección basada en la distancia *lookahead* (Lemus, 2011). La misma fue igualmente implementada y probada demostrando que era posible reducir el efecto causado por las perturbaciones marinas durante el desplazamiento del vehículo, sólo si se contaba con mediciones fiables de la velocidad del vehículo, las cuales no pueden ser obtenidas mediante el sistema de sensores de bajo costo del *HRC-AUV*. Por lo que el grupo propone como nueva solución manteniendo la misma ley de dirección utilizando

la estrategia conocida como *I-LOS*, la cual asegura la convergencia del vehículo al camino y reduce el efecto causado por las corrientes marinas, sin hacer uso de la medición de velocidad del vehículo (Valeriano Medina, 2015; Hernández Julián, 2014; Fernández, 2015). Aunque esta estrategia cuenta con buenos resultados en simulación, no se dispone de una implementación que pueda ser ejecutada en las unidades de cómputo a bordo del vehículo con vistas a su futura evaluación durante pruebas experimentales en el mar.

1.6. Consideraciones finales

Luego de realizar un análisis crítico de la literatura consultada, se arriban a las siguientes consideraciones:

El tema de guiado constituye un tópico que, por su importancia, ha sido ampliamente abordado por GARP en sus investigaciones. Específicamente, el tema de la reducción de los errores de seguimiento que provocan las corrientes marinas ha estado en el centro de atención. Los resultados en simulación del controlador *I-LOS* han demostrado ser muy buenos, sin embargo esta estrategia no ha podido ser probada en maniobras en el mar debido a que no se cuenta con una implementación de la misma que pueda ejecutarse en el software de navegación instalado en el *HRC-AUV*.

El sistema de software es vital e importante en el desarrollo de vehículos autónomos, ya que permite el control y manejo del hardware asociado a este, comunicación e intercambio de información entre módulos y además, proporciona una interfaz con el usuario. En el desarrollo de este tipo de tecnologías, se hace común el uso de una arquitectura por capas con el objetivo de potenciar la flexibilidad y modularidad del sistema. Esta es la causa por la cual en el diseño del sistema de software del *HRC-AUV* se emplea esta arquitectura, haciendo posible agregar la implementación de un controlador *I-LOS* sin comprometer las demás funcionalidades del mismo. Además, este controlador cuenta con un desarrollo matemático que hace sencilla su programación en lenguaje *C*.

CAPÍTULO 2
DESCRIPCIÓN DEL SISTEMA DE SOFTWARE DEL *HRC-AUV*

2.1. Introducción

En el presente capítulo se describe el sistema de software implementado en el *HRC-AUV*. En tal sentido se hace énfasis en la concepción del software de navegación incluido en la estación a bordo del vehículo, específicamente en la computadora industrial PC-104, así como del software de supervición y configuración *SharkSoft* que se ejecuta en la estación en tierra. Igualmente se presenta la estrategia *I-LOS* utilizada en el seguimiento de caminos rectos, mencionando las ecuaciones que permiten la programación en lenguaje *C* del algoritmo de guiado.

2.2. Estructura del sistema de software en el *HRC-AUV*

El desarrollo de un vehículo autónomo implica numerosos obstáculos y retos de alto nivel para cualquier equipo de investigación. El sistema de software implementado en el *HRC-AUV* está estructurado por capas, las cuales comprenden a los software de bajo y alto nivel.

El sistema de software está dividido en dos partes, la primera parte se ejecuta en el vehículo y la segunda en la estación remota. Las principales funciones de la estación a bordo son el control del vehículo, la ejecución del software de navegación[1] y el manejo de errores y alarmas. Por otra parte, la estación remota es destinada a: planificación de la misión, ajuste de reguladores y supervisión (Martínez, 2013).

El diseño del sistema de software del *HRC-AUV* está basado en el Lenguaje Unificado de Modelado (*UML, Unified Modeling Language*), permitiendo la especificación y adecuada documentación de la funcionalidad del sistema. Con este tipo de lenguaje es posible

[1]El software de navegación incluye, entre otras funcionalidades, el sistema de navegación, el sistema de guiado y el sistema de control.

generar diseños que capturen ideas de una forma convencional y de fácil entendimiento, permitiendo visualizar, especificar, construir y documentar un sistema (Jacobson, 2000).

A continuación en la Figura 2–1, se muestran los casos de uso relativos al sistema de software del *HRC-AUV* que cuenta con tres actores definidos:

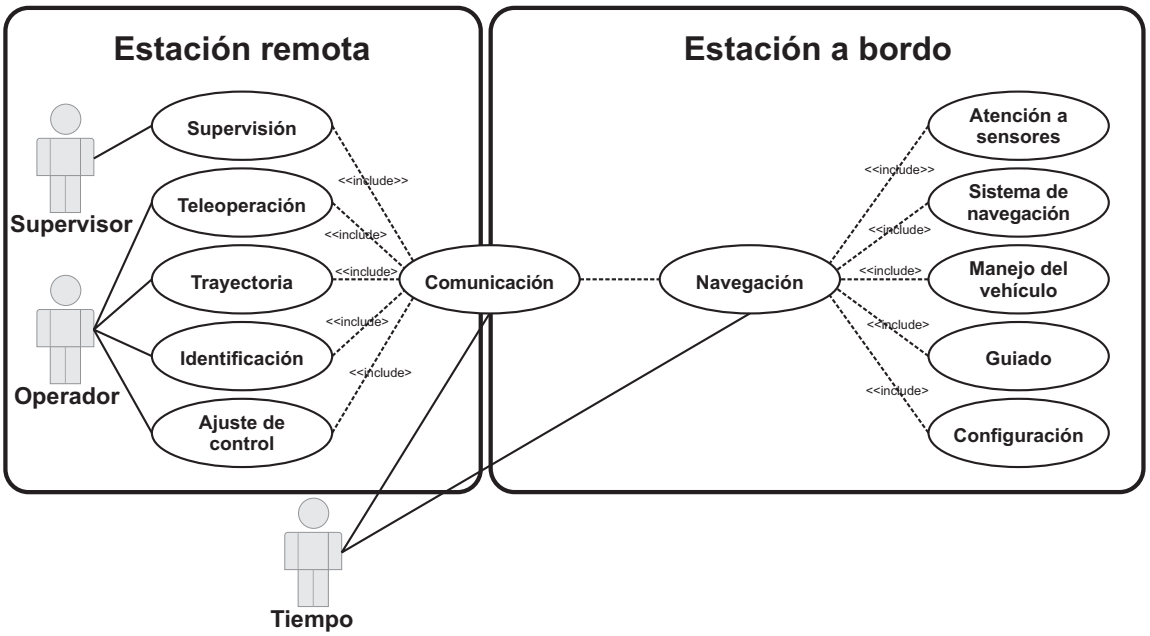

Figura 2–1: Diagrama general de casos de uso

Supervisor: se encuentra limitado sólo al uso del servicio de Supervisión, a partir del cual puede visualizar las variables que definen el estado del sistema mediante una interfaz gráfica.

Operador: tiene acceso a los servicios de Teleoperación, Ajuste de Control, Identificación y Planificación de Trayectoria. De esta manera tiene permitido manipular el guiado del vehículo a través de la teledirección sin la intervención de los lazos de control, realizar experimentos a lazo abierto para la obtención de juegos de datos con vista a la identificación experimental, cambiar y probar el ajuste de los controladores que intervienen en el manejo automático del móvil y trazar trayectorias a seguir por el vehículo.

Tiempo: es un efecto externo fuera del control del sistema que activa ciertos casos de usos; los eventos de activación por tiempo están enlazados al reloj de tiempo real del sistema, cuando el tiempo fijado es alcanzado el caso de uso es activado automáticamente.

2.3. Estación a bordo

La estación a bordo está compuesta por las unidades *DsPIC* y PC-104, entre las cuales se dividen el trabajo de adquisición de datos desde los sensores y las tareas de navegación y control (Martínez, 2013).

Los casos de uso mostrados en la Figura 2–2, describen los requerimientos generales del sistema y definen las funcionalidades necesarias para el cumplimiento de su propósito. A continuación se describen las particularidades de los casos de uso de la estación a bordo expuestos en el diagrama general.

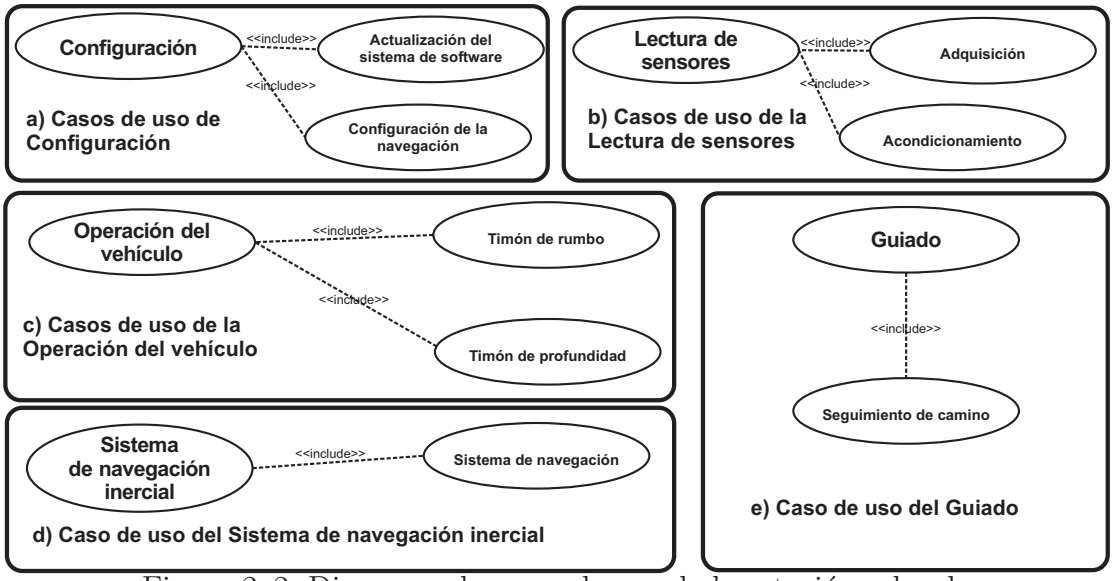

Figura 2–2: Diagrama de casos de uso de la estación a bordo

Atención a sensores: consiste en la adquisición de los datos provenientes de los sensores, así como de su acondicionamiento y conversión a unidades de ingeniería.

Sistema de navegación: incluye el algoritmo de estimación de la posición del vehículo.

Configuración: permite modificar la configuración de los controladores, ajustando tanto las ganancias como la estrategia de control a partir de la activación/desactivación de lazos específicos en el esquema general de control.

Manejo del vehículo: permite cambiar de estado manual a automático y viceversa, así como definir respectivamente el mando de los actuadores (en lazo abierto) o de los controladores (en lazo cerrado).

Guiado del vehículo: el guiado se lleva a cabo a partir de una trayectoria determinada manipulando el mando del lazo de control de dirección y venciendo cada uno de los puntos definidos en la trayectoria.

2.3.1. Unidad *DsPIC*

La unidad *DsPIC* tiene como funciones principales la adquisición de datos procedentes de los sensores de origen analógico tales como los que se usan para medir la profundidad, el ángulo de desviación de los timones, los parámetros de propulsión, el nivel de voltaje en las baterías y la presencia de agua en el vehículo, así como también la ejecución de los algoritmos de control de profundidad y rumbo del *AUV*. El diseño del software implementado en esta unidad se basa en habilitar o deshabilitar diferentes elementos de los lazos de control, así como la parametrización de los elementos habilitados.

En el diagrama de flujo que se presenta en la Figura 2–3 es posible apreciar que la arquitectura de software en la unidad *DsPIC* está conformada básicamente por cuatro procesos fundamentales y sus correspondidentes subprocesos, los cuales definen el funcionamiento de este sistema. La mayor parte de estos procesos están constituidos por subrutinas de atención a interrupciones, por lo que se generan de forma concurrente y no dependen del flujo secuencial en el programa. Cada uno de estos procesos está organizado según su nivel de prioridad y el tiempo en el que se ejecuta, por lo que en algunos casos pueden ser periódicos o aperiódicos.

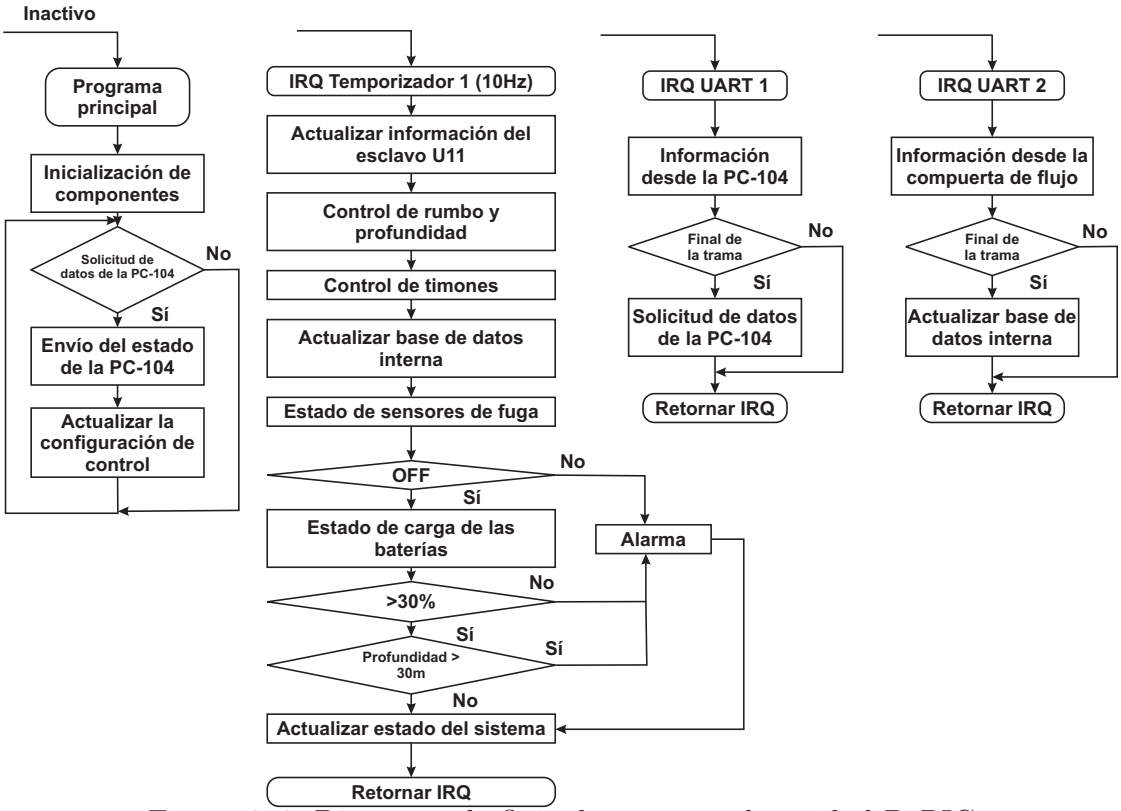

Figura 2–3: Diagrama de flujo de tareas en la unidad DsPIC

Esta unidad además provee interfaces al operador remoto para la manipulación directa del vehículo y el ajuste de los parámetros de los controladores. Igualmente se encarga

de la toma de datos desde los sensores y los procesa mediante una serie de filtros analógicos y digitales para actualizar la base de datos históricos generada por el software de navegación de la PC-104. Luego de la medición se inicializan los mecanismos para la detección de errores en el sistema, seguidos por los cálculos de las acciones de control correspondientes (Martínez, 2013; Guerra, 2010).

2.3.2. Unidad PC-104

Esta unidad está basada en un computador de placa reducida (*SBC*) *PCM-3362* de *Advantech*, como se aprecia en la Figura 2–4. Un *SBC* o *Single Board Computer* centra su diseño en un solo microprocesador, *RAM*, E/S y demás características de un ordenador funcional en una sola tarjeta que suele ser de tamaño reducido.

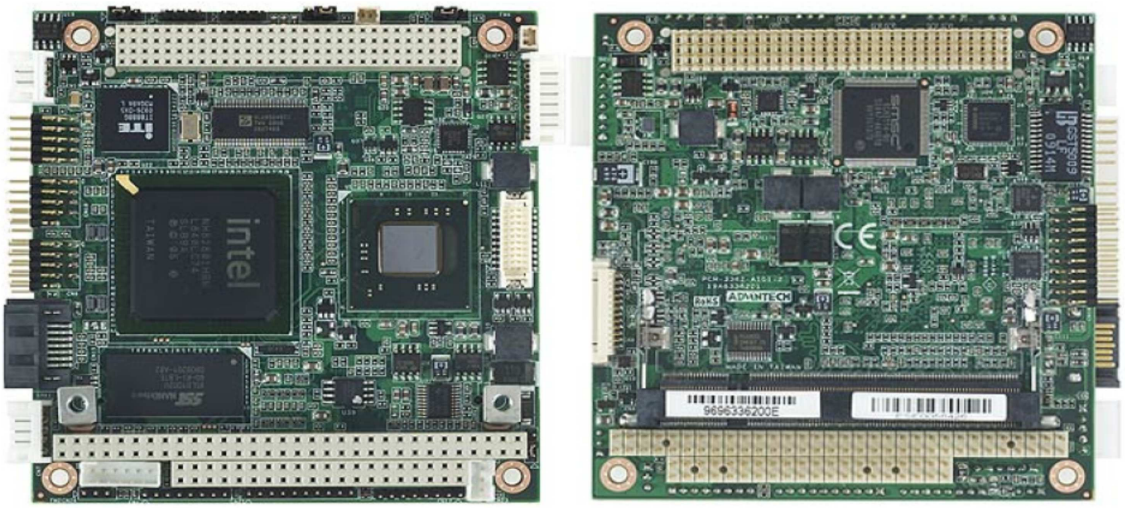

Figura 2–4: Unidad PC-104

El modelo *PCM-3362* fue seleccionado gracias al amplio rango de interfaces que incorpora su diseño y a la excelente relación rendimiento/precio. Para el caso del *HRC-AUV*, el *PCM-3362* fue ensamblado con 2GB de *RAM* y 4GB de *compact flash* conectado a través de un módulo de expansión *PCM-3835* de *Advantech* (Martínez, 2015). Como sistema operativo fue seleccionado *GNU-Linux*, específicamente la distribución *Debian* de 32 bits, versión 6.0.7 *Squeeze*.

En la Figura 2–5 se presenta un diagrama de flujo que muestra la operación general de la PC-104. Esta cuenta con cuatro procesos en ejecución, y al igual que la unidad *DsPIC*, cada uno de estos procesos está organizado según su nivel de prioridad y el tiempo en el que se ejecuta definiendo de esta forma el funcionamiento del sistema.

El software de nivel táctico que se ejecuta en la PC-104 funciona como un nodo de comunicación central para el resto de los sistemas en el vehículo y lleva a cabo las siguientes tareas:

- Adquirir la información brindada por el *AHRS* y el *GPS*.
- Adquirir información desde el software de nivel operativo que se ejecuta en la unidad *DsPIC* relacionada con los sensores y el estado del vehículo.
- Intercambiar información con el software de nivel estratégico desplegado en la estación de control de misión, usada para dar instrucciones (valores deseados y ajuste de reguladores) y supervisar las operaciones del vehículo a una frecuencia de 10 Hz.
- Configurar del sistema.
- Actualizar la base de datos interna del sistema.
- Ejecutar la solución del guiado.

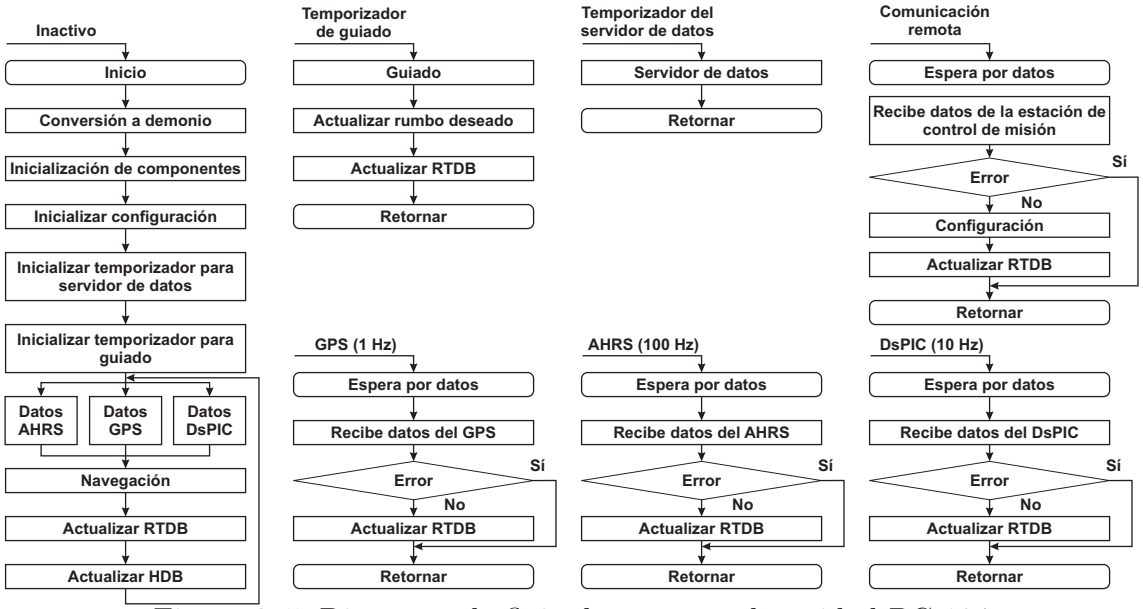

Figura 2–5: Diagrama de flujo de tareas en la unidad PC-104

2.4. Estación remota

El software de la estación remota, denominado *SharkSoft*, tiene como propósito principal supervisar y configurar remotamente el autopiloto del *HRC-AUV*. Este sistema mediante una interfaz de usuario muestra toda la información sensorial relevante, la cual está relacionada con las variables de rumbo, profundidad, ángulos de cabeceo y balanceo, entre otras. En la Figura 2–6 se presenta la interfaz de trayectoria del software.

El servicio de configuración comprende el ajuste de los reguladores de rumbo y profundidad, el sistema además incorpora gráficas que permiten la evaluación del desempeño de los controladores y cuenta con la gestión de misiones a través de mapas georreferenciados. Como herramienta adicional, el sistema posee un servicio de simulación que permite visualizar los resultados de un experimento luego de su realización, así como también incorpora un mecanismo de exportación de datos que facilita el post-procesamiento de la información luego de las misiones realizadas (Rodríguez, 2011).

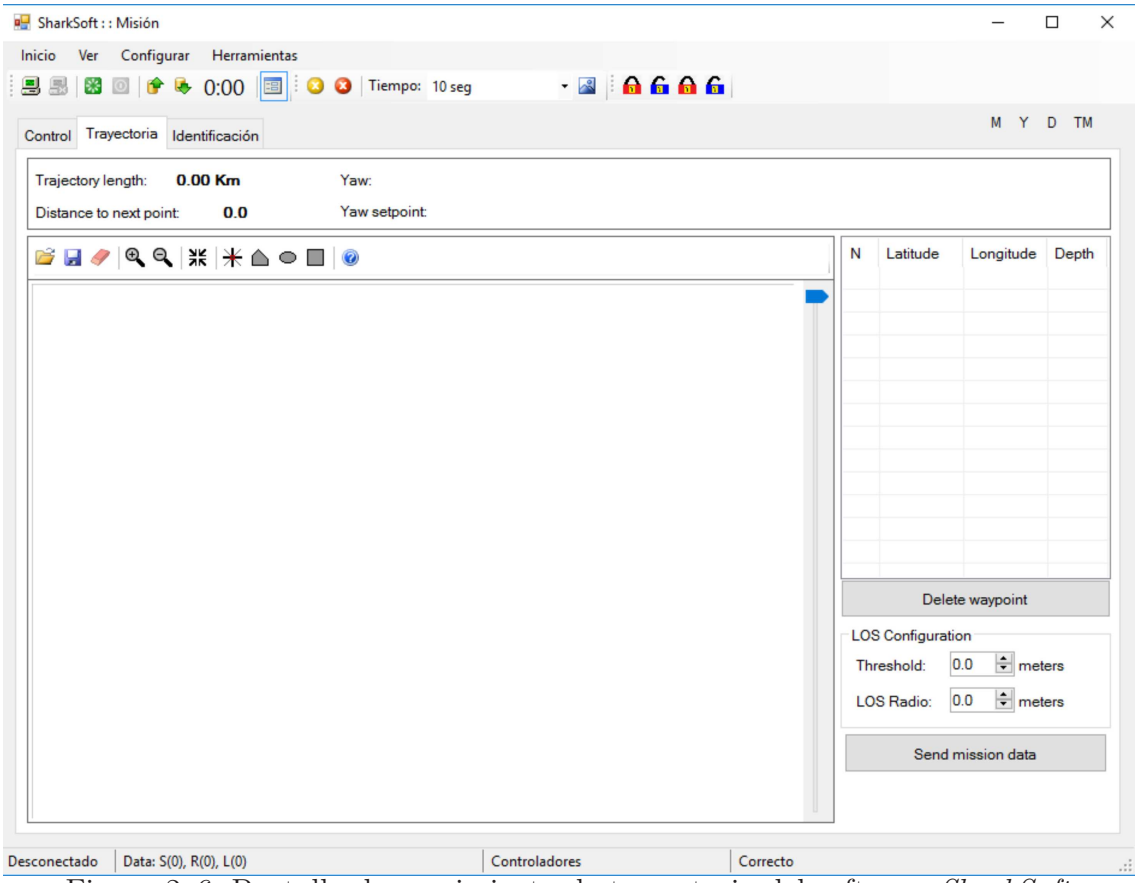

Figura 2–6: Pantalla de seguimiento de trayectoria del software *SharkSoft*

A continuación en la Figura 2–7 se describen las particularidades de los casos de uso de la estación remota expuestos en el diagrama general.

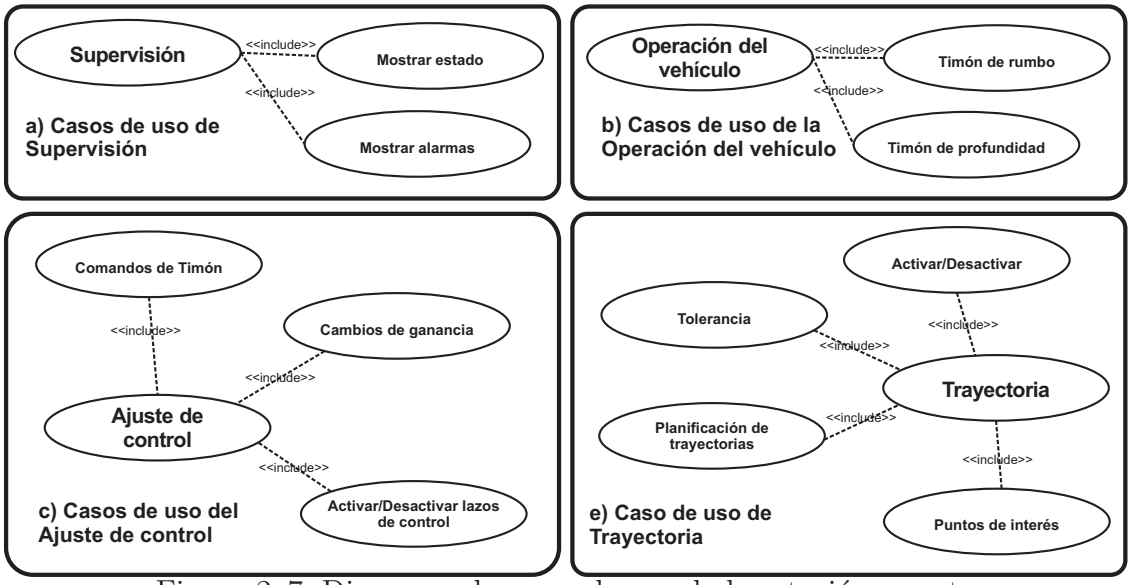

Figura 2–7: Diagrama de casos de uso de la estación remota

Supervisión: provee al supervisor con información visual sobre las variables que describen el estado del sistema. También presenta importantes indicadores de alarma tales como la presencia de agua dentro del casco, estado del motor principal (encendido/apagado) y la temperatura interna del vehículo.

Teleoperación: permite al operador manipular mediante una palanca de mando o un teclado al vehículo sin la intervención del control. El control remoto se realiza mediante el software de monitoreo *SharkSoft*, y es útil cuando el vehículo realiza maniobras que requieren la experiencia de un operador humano.

Identificación: permite realizar experimentos en lazo abierto para los subsistemas de rumbo y profundidad, con la ayuda de los grupos de datos obtenidos para la identificación de los parámetros del modelo dinámico del vehículo.

Ajuste del control: permite modificar la estructura o las ganancias de los controladores en el vehículo, y definir los valores deseados de rumbo y profundidad para verificar el rendimiento del sistema de control.

Trayectoria: permite al operador definir una trayectoria deseada a partir de un conjunto de puntos de interés que el vehículo debe alcanzar.

Figura 2–8: Trama de comunicación

El intercambio de información que se establece entre la *PC-104* y la estación de control de misión está definida en una comunicación remota. Los mensajes intercambiados se clasifican en solicitud de paquetes de datos y paquetes de configuración. La estructura de la trama de comunicación queda definida en la Figura 2–8.

Preámbulo: principio de reconocimiento para asegurar una apropiada recepción, ya que en ambientes de navegación la señal trasmitida es afectada por el ruido de fondo y el desvanecimiento multidireccional.

ID del mensaje: identificador que define el tipo de mensaje a enviar.

Longitud de datos: establece la cantidad de bytes que serán enviados en el paquete.

Datos: es el paquete de parámetros requeridos según el tipo de mensaje solicitado.

Suma de comprobación: detecta cambios accidentales en la secuencia de datos para proteger la integridad de los mismos, verificando que no haya discrepancias entre los valores obtenidos al hacer una comprobación inicial (antes de enviar) y otra final (después de recibir).

2.5. Estrategia de guiado *I-LOS*

Como parte del sistema de guiado del vehículo, se hace necesario implementar en el software de navegación un algoritmo que permita la realización de maniobras de seguimiento de caminos rectos. En el proyecto *HRC-AUV* se ha optado por utilizar un controlador *I-LOS*. Dicha estrategia tiene el objetivo de lograr que la embarcación converja a una trayectoria deseada reduciendo los efectos causados por perturbaciones marinas.

Este algoritmo de guiado está basado en la distancia *lookahead*. Dicha ley, se utiliza con el propósito de conducir al vehículo hacia el camino en dirección al vector *LOS* y al mismo tiempo, lograr la convergencia a cero del error de seguimiento perpendicular al camino *e(t)* (Fossen, 2011). El vector *LOS* se orienta desde la posición del vehículo hasta el punto p_{int}, que está situado en una línea tangencial al camino, a una distancia *lookahead* (Δ) de la proyección de la posición de la embarcación sobre el camino (Kapelios, 1992).

Para el caso de esta investigación es de interés el seguimiento de caminos rectos. Un camino en línea recta bajo las condiciones antes mencionadas está definido esencialmente por dos puntos, a través de los cuáles el vehículo debe pasar. Estos puntos pueden ser definidos como $\boldsymbol{p_k} = [y_k, x_k]^T \in \Re^2$ y $\boldsymbol{p_{k+1}} = [y_{k+1}, x_{k+1}]^T \in \Re^2$ respectivamente, como se muestra en la Figura 2–9. Considerando $\boldsymbol{p_k}$ como origen de la referencia al camino, cuyo eje x ha sido rotado un ángulo positivo α_k tal que:

$$\alpha_k = \arctan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \tag{2.1}$$

también es posible calcular el error de seguimiento perpendicular al camino *e(t)* como:

$$e(t) = -(x_t - x_k)\sin\alpha_k + (y_t - y_k)\cos\alpha_k \tag{2.2}$$

La distancia *lookahead* puede considerarse constante o variable según el camino a seguir. Lekkas, en su investigación (Lekkas, 2014), propone considerar este valor costante cuando se realiza el seguimiento en línea recta. En este caso, la ley de dirección propone determinar la rumbo que debe seguir el vehículo a partir de la suma de dos ángulos:

$$\chi(e) = \chi_r(e) + \chi_p \tag{2.3}$$

donde:

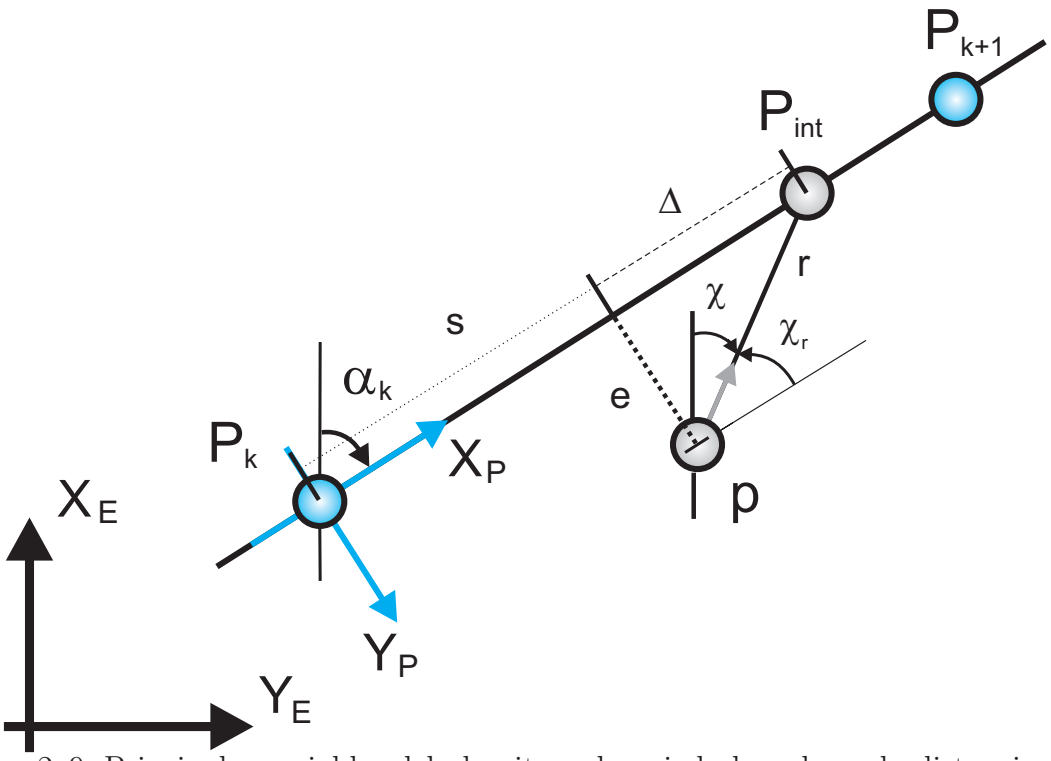

Figura 2–9: Principales variables del algoritmo de guiado basado en la distancia de la proyección del vehículo sobre el camino

$$\chi_p = \alpha_k \tag{2.4}$$

siendo χ_p el ángulo de rotación del camino con respecto al eje x del sistema de referencia en tierra y $\chi_r(e)$ un ángulo de correción, el cual asegura que el vehículo corrija su dirección hacia el punto de intersección con el camino ($\boldsymbol{p_{int}}$) al punto del camino hacia el cual la embarcación se dirige.

$$\chi_r(e) = \arctan\left(\frac{-e(t)}{\Delta}\right) \tag{2.5}$$

El objetivo de control del seguimiento de camino consiste en determinar un valor del ángulo de rumbo deseado, que le permita al vehículo ir venciendo los puntos de la trayectoria. En este tipo de estrategia de control, uno de los métodos que se utilizan para disminuir la influencia de las perturbaciones marinas consiste en introducir una acción integral dentro de la ley de dirección[2], teniendo en cuenta que con la inclusión de dicha

[2]Estrategia que utiliza la velocidad del vehículo para contrarrestar el efecto de las corrientes marinas, abordada en el epígrafe 1.3.

acción puede producirse un efecto *wind up*[3]. Por lo cual es conveniente implementar un regulador PI que permita mantener el valor de salida del regulador en los límites concebidos para el ángulo $\chi_r(e)$ (Borhaug, 2008; Lekkas, 2014). La estructura matemática que permite la implementación del regulador PI anti *wind up* se define como:

$$\chi_r(e) = -\arctan(k_p e(t) + k_i y_{int}) \tag{2.6}$$

donde $k_p = \frac{1}{\Delta}$, $k_i = k k_p$, siendo k $(k > 0)$ un parámetro de diseño y $\dot{y}_{int}$ queda definida como:

$$\dot{y}_{int} = \frac{e\Delta}{\Delta^2 + (e + k y_{int})^2} \tag{2.7}$$

2.5.1. Criterio de punto vencido

Si el camino a seguir está formado por n segmentos de línea recta conectados por $n+1$ puntos del camino, es necesario emplear una condición de punto vencido permitiendo que el vehículo se dirija hacia el siguiente punto del camino. Para dicha condición se precisa contar con el error de seguimiento a lo largo del camino *s(t)*:

$$s(t) = -(x_t - x_k)\cos\alpha_k + (y_t - y_k)\sin\alpha_k \tag{2.8}$$

y la distancia total entre los puntos $\boldsymbol{p_k}$ y $\boldsymbol{p_{k+1}}$, que se define como s_{k+1}:

$$s_{k+1} = (x_{k+1} - x_k)\cos\alpha_k + (y_{k+1} - y_k)\sin\alpha_k \tag{2.9}$$

de ahí que la condición de punto vencido queda establecida de la forma:

$$s_{k+1} - s(t) \geq R_{k+1} \tag{2.10}$$

[3]Este efecto es característico de los controladores que presentan acción integral. Suele producirse cuando en un sistema de control con un amplio rango de operación, la variable de control alcanza los límites prefijados del actuador; cuando esto sucede, la salida del lazo de control permanece en su límite independientemente de la salida del proceso.

2.6. Consideraciones finales

El sistema de software implementado en el *HRC-AUV* posee una arquitectura modular que le permite ser flexible a cambios en su programación, posibilitando en este caso agregar la estrategia de guiado *I-LOS* en el software de navegación del vehículo. Dicha tarea incluye modificaciones tanto en el software de nivel táctico como en el software de supervisión y monitoreo. Además el algoritmo de guiado no posee una complejidad matemática relevante, por lo que su implementación es relativamente sencilla.

CAPÍTULO 3
IMPLEMENTACIÓN DE LA ESTRATEGIA *I-LOS* EN LENGUAJE *C*

3.1. Introducción

El objetivo principal de esta investigación, es implementar la estrategia de guiado *I-LOS* en lenguaje *C*, que asegure un adecuado comportamiento del vehículo durante las pruebas experimentales ante perturbaciones marinas. Para cumplir con este objetivo se hizo necesario programar el algoritmo de guiado en lenguaje *C* y validarlo mediante la herramienta *Simulink* del software *MATLAB*. Además, con el propósito de incluir dicho algoritmo en la actual versión del software de navegación fue vital realizar modificaciones al sistema de software del vehículo.

3.2. Ejecución de código *C* utilizando el software *MATLAB*

El ámbito de aplicación del software de *MATLAB* es muy amplio, pues incluye prestaciones básicas como la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos. El uso de *MATLAB* (MATLAB, 2015) permite analizar datos, desarrollar algoritmos, y crear modelos y aplicaciones, puesto que posee un lenguaje de alto nivel y un ambiente interactivo para el cálculo numérico, la visualización y la programación.

Una de las herramientas más usadas de este software matemático es *Simulink*. Esta herramienta provee un editor gráfico, una librería de bloques personalizable y diferentes métodos numéricos seleccionables por el usuario para la solución de sistemas tanto estáticos como dinámicos, así como también ecuaciones íntegro-diferenciales. Una de las características claves de *Simulink* es la Herramienta de Código Heredado (*Legacy Code Tool*) para la importación de código *C* o *C++* a los modelos. Esta característica puede ser utilizada a través de varios bloques predefinidos por la librería *Simulink*, como por ejemplo el bloque *S-Function*.

Una *S-Function* es una descripción de lenguaje de computadora de un bloque del entorno de *Simulink* escrito en *MATLAB*, *C*, *C++* o *Fortran*. Las *S-Functions* de *C*,

C++ y *Fortran* son compiladas como archivos *MEX* usando la utilidad *mex*. Como otros archivos *MEX*, las *S-Functions* son subrutinas relacionadas dinámicamente que el intérprete de *MATLAB* puede cargar y ejecutar automáticamente. Los archivos en *C* se relacionan con una *S-Function* mediante la pantalla de configuración *Simulation Target*, Figura 3–1, en la que además pueden ser relacionadas librerías.

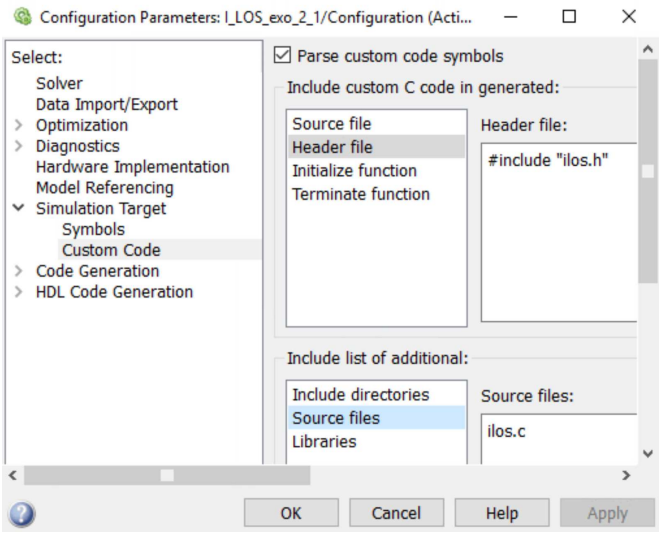

Figura 3–1: Ventana para enlazar los archivos en *C*

Con los archivos cabecera y fuente enlazados a la *S-Function* es posible hacer llamadas a sus funciones utilizando la función *coder.ceval* de *MATLAB*. Esta función es capaz de ejecutar una función externa de *C/C++* especificando el nombre de la función implementada y además los argumentos de entrada si los lleva.

En la Figura 3–3 es posible apreciar el diagrama de bloques que simula el comportamiento del vehículo controlando el rumbo mediante la estrategia de guiado *I-LOS*. En la misma se presenta una *S-Function* con archivos *C* incorporados (*ilos.h* e *ilos.c*) denominada **Guiado en *C***.

En el archivo fuente *ilos.c* se implementa la función *navigate()*, en la que en un primer momento se comprueba si se ha iniciado la trayectoria o si se ha cambiado de tramo para calcular los valores de α_k, Ecuación 2.1, y *td*, Ecuación 2.9, pues como son valores que se mantienen constantes a lo largo de un tramo recto no es necesario realizar su cálculo en todas las iteraciones del programa. Luego se procede al cálculo de los errores *st*, Ecuación 2.8, y *et*, Ecuación 2.2, así como de la variable *yintd* tomando inicialmente $yint = 0$, Ecuación 2.7. A continuación se realiza una integración de tipo *Euler-Forward* del valor de *yintd* obteniéndose un nuevo valor para *yint*, que se utiliza en el cálculo de *yaw*, Ecuación 2.6. Por último, se comprueba la condición de punto vencido para asegurar si es necesario cambiar o no el tramo de la trayectoria.

En el código implementado se hace necesario imponer una corrección a la integración de tipo *Euler-Forward* utilizada. Dicha corrección se establece mediante métodos empíricos y consta de dividir por tres el valor instantáneo a integrar, con el objetivo de lograr el menor error posible entre los parámetros y gráficas de la co-simulación. Sin esta corrección los resultados de la co-simulación divergen a medida que se incrementa el tiempo de simulación, lo cual está dado por el particular funcionamiento del integrador del *MATLAB* y de las características propias del algoritmo de guiado. Esta particularidad consiste en un bucle interno de integración, el cual se ejecuta durante cada iteración hasta que el solucionador alcanza la precisión deseada.

3.3. Resultados de la co-simulación

La validez de la programación del algoritmo de guiado, realizada en lenguaje *C*, puede ser corroborada a partir de comparar la similitud entre las señales de salida de esta implementación y la realizada anteriormente en *MATLAB*. Teniendo en cuenta lo anterior se realiza la co-simulación, haciendo uso de las variantes mostradas en las Figuras 3–2 y 3–3 que implementan el mismo algoritmo de guiado.

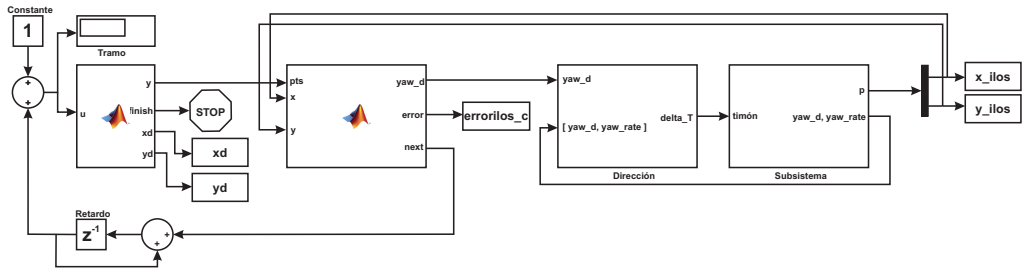

Figura 3–2: Implementación en *MATLAB* del algoritmo de guiado *I-LOS*

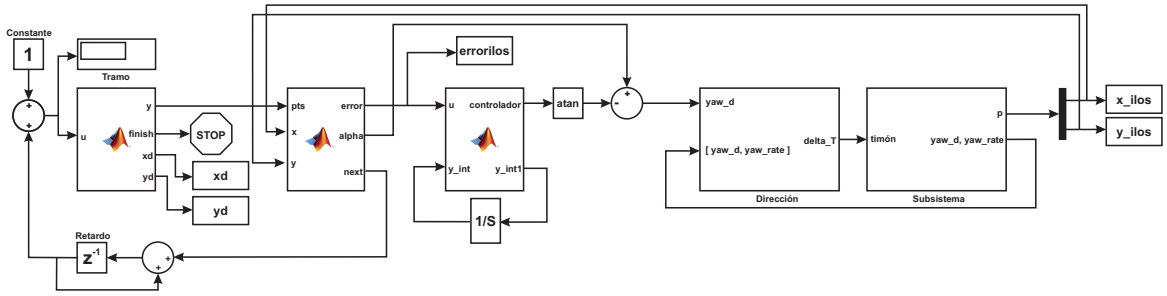

Figura 3–3: Implementación en *C* del algoritmo de guiado *I-LOS*, ejecutada en *MATLAB*

La Figura 3–2 muestra la propuesta realizada en el trabajo de Anailys (Hernández Julián, 2014). Por su parte, la Figura 3–3 consiste en la implementación en código *C* del bloque de generación de puntos y el controlador *I-LOS* de la propuesta antes mencionada. La comparación se realiza entre las señales de posición y el error perpendicular al camino obtenidas durante el seguimiento.

Tabla 3–1: Puntos de la trayectoria del experimento del año 2010

x(m)	0	215	921	120	-546
y(m)	0	646	520	-55	-294

Para realizar la co-simulación se utilizan datos obtenidos durante experimentos con el vehículo en el mar, realizados en el año 2010. La trayectoria estuvo descrita por los puntos mostrados en la Tabla 3–1.

En dicho año, el vehículo contaba con un motor de 24 V de alimentación, el cual giraba a una velocidad de 500 rpm.

Para la simulación se recrearon las condiciones medioambientales siguientes: velocidad de las corrientes marinas $V_c = 0.18$ m/s con un ángulo de dirección $\beta_c = 240°$, frecuencia fundamental de las olas en el orden de los 6 rad/s y coeficiente de intensidad de las olas $\sigma = 0{,}5$. Además, se considera un período de muestreo T = 0.01 s y el radio para el círculo de aceptación de 20 m. Las gráficas resultantes de la co-simulación se presentan en las Figuras 3–4, 3–5 y 3–6.

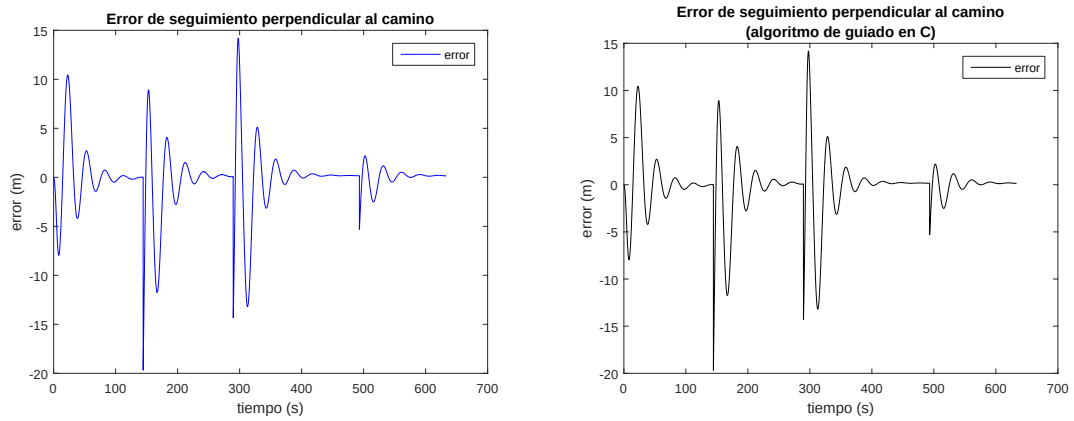

Figura 3–4: Error de seguimiento perpendicular al camino

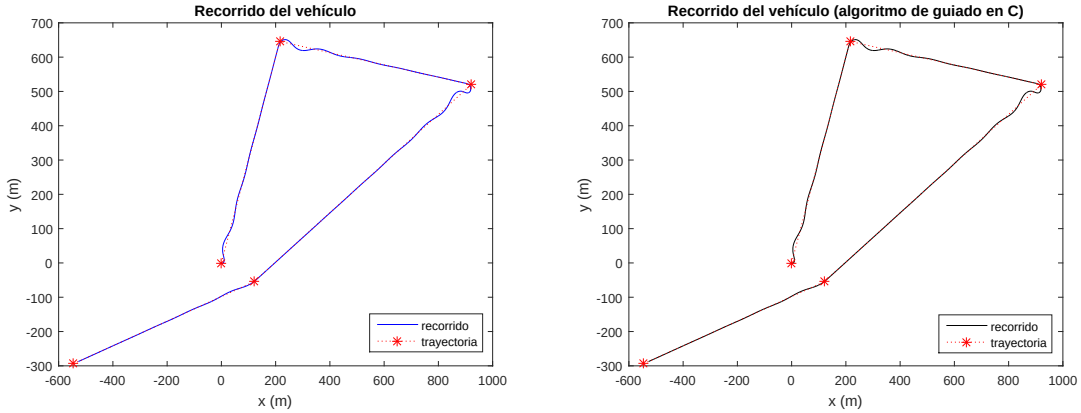

Figura 3–5: Recorrido del vehículo durante la maniobra simulada

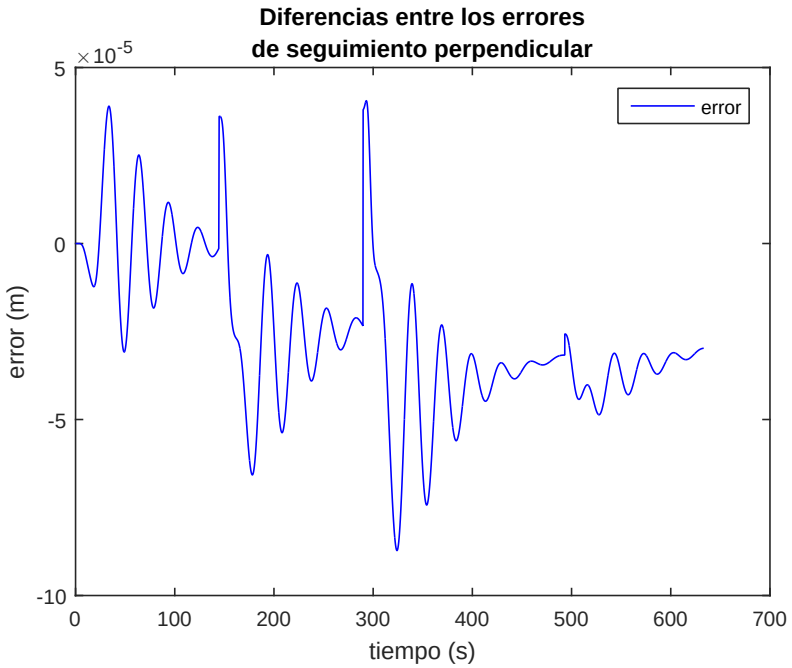

Figura 3–6: Diferencia entre los errores de seguimiento perpendicular

En las figuras 3–4 y 3–5 se presentan las gráficas del error de seguimiento perpendicular al camino y el recorrido del vehículo, que se obtienen durante la maniobra que se simula mediante *MATLAB Simulink*. Entre estas gráficas no se muestra una divergencia apreciable que asegure la validación adecuada de la implementación del algoritmo de guiado *I-LOS* en lenguaje *C*, por lo que en la Figura 3–6 se presenta la diferencia que existe entre las señales de error perpendicular al camino de ambas implementaciones. Con ello se corrobora que existe una gran similitud entre las mismas, de ahí que la implementación en *C* es válida y puede ser integrada al supervisorio del *HRC-AUV*.

3.4. Principales modificaciones realizadas al sistema de software del *HRC-AUV*

Como parte de la implementación del controlador *I-LOS* en lenguaje *C* para el seguimiento de caminos en el *HRC-AUV*, fueron necesarias modificaciones en los software de nivel estratégico y táctico. Todas las modificaciones realizadas tienen su base en la adición de código por lo que no fue requerida una reestructuración del sistema de software del *HRC-AUV*, el cual hace dicha tarea posible gracias a su modularidad y flexibilidad.

3.4.1. Software de nivel estratégico

Para la implementación del algoritmo de guiado *I-LOS* se hizo necesario incluir nuevas variables tales como una ganancia proporcional (Kp), un parámetro de diseño (K) y una variable que indique la selección entre estrategias de guiado. La modificación de la interfaz gráfica del software de supervisión, mostrada en la Figura 3–7, permite contar

con dichos parámetros y con la posibilidad de seleccionar la estrategia de guiado a utilizar en futuras maniobras.

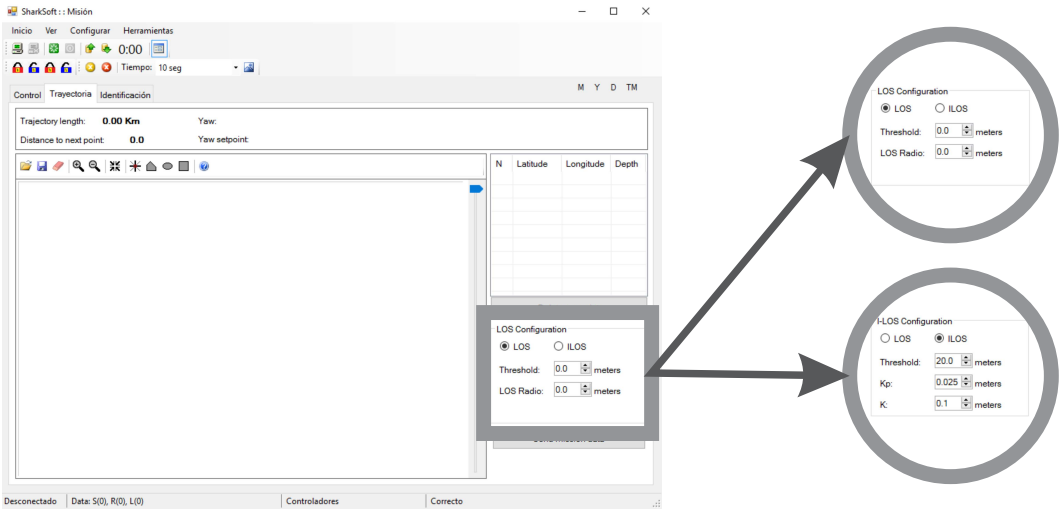

Figura 3–7: Modificación realizada en la pantalla de Trayectoria del software *SharkSoft*

También se hizo necesario modificar el mensaje que se envía al software de nivel táctico con los datos de la misión para incluir en la trama los valores de Kp, K y el tipo de controlador a utilizar (*I-LOS* o *LOS*[1]). Es válido aclarar que para la estrategia *LOS* se utilizan parámetros como el radio *LOS* (RLOS) y el radio de aceptación para la declaración de punto vencido (*Threshold*), pero en el caso del algoritmo *I-LOS* como el valor del radio *LOS* no es utilizado, se envía en su lugar el valor de la ganancia proporcional.

3.4.2. Software de nivel táctico

El software de nivel táctico al recibir la trama de comunicación, ejecuta la función *GARPCOM_GetMessage()* implementada en el archivo fuente *garpcom.c*. Dicha función comprueba la autenticidad e integridad del mensaje recibido. Cuando el mensaje es auténtico y sin pérdidas mediante, una estructura *switch*, se verifica el tipo de mensaje que fue enviado según su identificador, ejecutando una función acorde para procesar la información.

La función que se ejecuta según el tipo de mensaje, se encarga de leer los datos enviados y asignarlos a las variables correspondientes que serán usadas en otras implementaciones del software de navegación. Todo lo explicado anteriormente compone la función *REMOTECOM_RecieveThreadCallback()* que se encuentra implementada en el archivo fuente *remote_comunication.c*, la cual se ejecuta de forma periódica.

[1]Controlador implementado en el software de navegación por Lemus (Lemus, 2011)

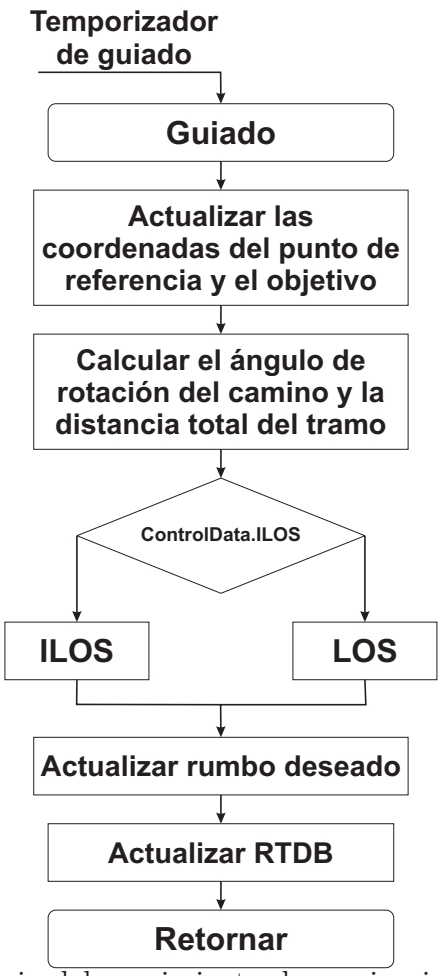

Figura 3–8: Diagrama de flujo del seguimiento de camino implementado en el software de navegación

En la función *Trajectory()* de *remote_comunication.c* se agregaron nuevas variables (K_ilos e ILOS) a las cuales les son asignadas valores que se adicionaron en el mensaje de los datos de la misión que se envía desde el supervisorio. Las variables K_ilos e ILOS fueron previamente definidas en la estructura ControlData implementada en el archivo cabecera *rtdb.h*.

Las principales modificaciones realizadas al código del software de navegación están dirigidas al sistema de guiado, específicamente en las estrategias para el seguimiento de camino. El objetivo fue implementar el algoritmo *I-LOS* en conjunto con la estrategia que se encuentra programada en el archivo fuente *path_following.c* del software de navegación, como se muestra en la Figura 3–8. Para esto se realiza una comprobación del tipo de estrategia a utilizar mediante un estructura *if/else*, cambiando entre una u otra según lo solicite el usuario.

3.5. Valoración económica

Las investigaciones que basan su temática en el desarrollo de *AUV* poseen un alto valor económico. La compra de un vehículo de este tipo consta de una inversión de 1.5 a 2 millones de *USD*, según sea el equipamiento instalado. Estas cifras no incluyen el costo de reparaciones y piezas de respuesto específicas, además de que se adquiere la aplicación pero no se dispone del conocimiento para realizar actualizaciones o reparaciones.

Todo lo anterior constituye una estrategia de la entidad proveedora para crear dependencia del cliente a sus servicios, pues los mismos representan un inversión de 15 a 20 mil *USD* diarios. Un país subdesarrollado como Cuba no cuenta con la infraestructura económica necesaria para comprometerse en este tipo de inversiones, por lo que le es muy difícil tener acceso a este tipo de tecnología. El proyecto *HRC-AUV*, desarrollado por la colaboración conjunta entre GARP y CIDNAV, se realiza con el objetivo de sobreponerse a este obstáculo constituyendo un ahorro significativo a la economía del país.

El vehículo *HRC-AUV* respresenta una solución totalmente nacional que está diseñado e implementado con un sistema de software de bajo costo y su ámbito de aplicación es adaptable a las necesidades propias del país. Por tanto con el objetivo de lograr una buena relación costo/prestaciones, es imprescindible que las deficiencias surgidas de utilizar un sistema de hardware de bajo costo sean contrarrestadas con soluciones ingenieriles basadas fundamentalmente en el modelado y control del vehículo. El desarrollo del proyecto *HRC-AUV* demuestra que en Cuba también es posible realizar aplicaciones de este tipo y además, representa un paso de avance hacia la independencia tecnológica y el desarrollo económico.

3.6. Consideraciones finales del capítulo

La implementación del controlador *I-LOS* en lenguaje *C* y su inclusión en el sistema de software del *HRC-AUV* brinda la posibilidad de evaluar el desempeño del mismo en pruebas experimentales en el mar. Dicho algoritmo, al no poseer una alta complejidad matemática, puede ser implementado sin dificultad y sin llevar a cabo cambios significativos que conduzcan a gastos adicionales.

La efectividad del controlador *I-LOS* programado en *C* ha quedado demostrada, al comparar su desempeño con la implementación del mismo realizada con la herramienta *Simulink* del software *MATLAB*. La comparación se realiza en base a la diferencia entre los errores de seguimiento perpendicular al camino, la cual al ser tan pequeña valida el adecuado desempeño de la implentación en *C*. Para todos estos análisis se utilizaron los datos recopilados durante experimentos en el mar en el año 2010.

CONCLUSIONES

Como resultado final de esta investigación se logra la implementación e inclusión del algoritmo de guiado *I-LOS* en el software de navegación del *HRC-AUV*, dándole cumplimiento al objetivo general. A partir de este resultado, se plantean las siguientes conclusiones:

- La estrategia de guiado *I-LOS* asegura la convergencia del vehículo a un camino recto y reduce el efecto causado por las corrientes marinas.
- La arquitectura modular y flexible del software de navegación, permite la inclusión del algoritmo *I-LOS* sin necesidad de realizar modificaciones importantes en el autopiloto del *HRC-AUV*.
- La validez de la programación en *C* del algoritmo *I-LOS* ha quedado demostrada al comparar su desempeño con la implementación del mismo realizada en el software *MATLAB*.
- La implementación del controlador *I-LOS* en lenguaje *C* permite la realización de maniobras de seguimiento de caminos rectos por parte del *HRC-AUV*.

RECOMENDACIONES

Para establecer la necesaria continuidad que debe tener este trabajo se recomienda lo siguiente:

- Evaluar el desempeño del controlador *I-LOS* durante el seguimiento de caminos rectos en pruebas experimentales con el *HRC-AUV* en el mar. De esta manera se corroborarían los resultados obtenidos mediante simulación que se presentan en esta investigación.
- Extender la implementación del controlador *I-LOS* para el seguimiento de caminos rectos en tres dimensiones. Además, incluir dicha implementación en el software de nivel táctico del *HRC-AUV* para corroborar su desempeño durante la realización de maniobras con el vehículo.

REFERENCIAS BIBLIOGRÁFICAS

Arora, G.; Aiaswamy, B.; Pandey N. (2002). *Microsoft C# Professional Projects*. Premier Press. Indianapolis, IN, Estados Unidos.

Blidberg, D. R. (2001). The development of autonomous underwater vehicles (auv); a brief summary. In: *International Conference on Robotics and Automation (ICRA)*. IEEE Xplore. Seul, Corea del Sur.

Borhaug, E.; Pavlov, A.; Pettersen K. Y. (2008). Integral los control for path following of underactuated marine surface vessels in the presence of constant ocean currents. In: *47th IEEE Conference on Decision and Control. IEEE Xplore*. Cancún, México. pp. 4984–4991.

Breivik, M. (2010). Topics in guided motion control of marine vehicles. Tesis doctoral. NTNU. Noruega.

Breivik, M.; Fossen, T. I. (2007). Applying missile guidance concepts to motion of marine craft. In: *Control Applications in Marine Systems*. Vol. 7. IFAC. Croacia. pp. 349–354.

Breivik, M.; Fossen, T. I. (2008). *Underwater vehicles*. Chap. Guidance Laws for Autonomous Underwater Vehicles, pp. 51–76. InTech. Vienna, Austria.

Butler, B.; den Hertog, V. (1993). Theseus: a cable-laying auv. In: *Proceedings. Engineering in Harmony with Ocean. OCEANS '93*. Vol. 1. IEEE. Victoria, British Columbia, Canadá. pp. I210 – I213.

Chitre, M. (2008). Dsaav - a distributed software architecture for autonomous vehicles. In: *OCEANS 2008*. IEEE. Quebec, Canadá. pp. 1–10.

Eskesen, J.; Owens, D.; Soroka M.; Morash J.; Hover F. S.; Chryssostomidis C. (2009). Design and performance of odyssey iv: a deep ocean hover-capable auv. Reporte técnico. Massachusetts Institute of Technology. Sea Grant College Program. Cambridge, Massachusetts, Estados Unidos.

Ferguson, J; Pope, A.; Butler B.; Verrall R. I. (1999). Theseus auv - two record breaking missions. *Sea Technology* **40**(2), 65–70.

Fernández, J. E. (2015). Controlador I-LOS considerando la distancia lookahead variable para el seguimiento de caminos curvos en un AUV. Tesis de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons. Nueva York, Estados Unidos.

Guerra, C. E. (2010). Diseño e implementación de hardware y software de bajo nivel para vehículo submarino autónomo. Tesis de diploma. Santa Clara, Cuba.

Hegrenaes, O.; Hallingstad, O.; Jalving B. (2007). Comparison of mathematical models for the *hugin* 4500 *auv* based on experimental data. In: *THE IEEE Internatinal Symposium on Underwater Technology*. IEEE Xplore. Japón. pp. 558 – 567.

Hernández Julián, A. (2014). Estrategia de control para el seguimiento de camino de un vehículo autónomo subacuático. Trabajo de diploma. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Jacobson, I.; Booch, G.; Rumbaugh J. (2000). *El proceso unificado de desarrollo de software*. Vol. 7. Addison Wesley Reading.

Kapelios, I. (1992). Stability of turning rate guidance and control laws for autonomous vehicles. Tesis de maestría. Naval Postgraduate School. California, Estados Unidos.

Kim, T. W.; Yuh, J. (2003). Development of a real-time control architecture for a semi-autonomous underwater vehicle for intervention missions. *Control Engineering Practice* **12**(12), 1521–1530.

Lekkas, A. M. (2014). Guidance and Path-Planning Systems for Autonomous. Tesis doctoral. NTNU. Noruega.

Lemus, J. L. (2011). Software de adquisición y control en tiempo real para Vehículo Sumergible Autónomo. Trabajo de diploma. UCLV. Santa Clara, Cuba.

Martínez, A. (2015). Model aided Inertial Navigation for AUV. PhD thesis. Department of Electronics and Informatics, Faculty of Engineering, Vrije Universiteit Brussel. Bruselas, Bélgica.

Martínez, A.; Rodríguez, Y.; Hernández L.; Guerra C.; Lemus J.; Sahli H. (2013). Arquitectura de hardware y software para auv, resultados experimentales. *Revista Iberoamericana de Automática e Informática industrial* **10**(3), 333–343.

MATLAB (2015). *versión 8.5.0.197613 (R2015a)*. The MathWorks Inc. Natick, Massachusetts.

Rodríguez, Y. (2011). Sistema remoto de supervisión y configuración de autopiloto de vehículo autónomo subacuático. Tesis de maestria. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Sangekar, M.; Chitre, M.; Koay T. B. (2008). Hardware architecture for a modular autonomous underwater vehicle $starfish$. In: *OCEANS*. IEEE Xplore. Quebec, Canadá. pp. 1–8.

Valeriano Medina, Y. (2013). Modelado dinámico de un vehículo autónomo subacuático. Tesis de maestría. UCLV. Dpto. de Automática y Sistemas Computacionales. Santa Clara, Cuba.

Valeriano Medina, Y.; Hernández Julián, A.; Hernández L. (2015). Controlador i-los para el seguimiento de caminos en línea recta de un vehículo autónomo subacuático. *Revista de Ingeniería Electrónica, Automática y Comunicaciones* **XXXVI**(2), 15–28.