Escuela Técnica Superior de Ingeniería
Universidad de Huelva

Doble Grado de Ingeniería Electrónica Industrial e Ingeniería Mecánica

Trabajo Fin de Grado

Diseño de un controlador para la realización de maniobras automáticas en un ROV sumergible

Autora: Olga Marín Cañas
Tutores: Fernando Gómez Bravo
Rafael López De Ahumada Gutiérrez

Septiembre 2022

Agradecimientos

Terminar esta doble titulación ha sido una tarea larga y ardua, y nunca hubiera sido posible sin la ayuda de las muchas personas a las que guardo un afecto especial. A todas ellas quiero expresar mi agradecimiento.

En primer lugar, a mis padres y a mi hermana, por apoyarme durante estos años y aguantar mis momentos de máximo estrés. A mis amigos, por llenar este camino de momentos felices. También a mis compañeros del laboratorio, especialmente a Alejandro Garrocho Cruz y Sara Rua Tirado, por su continua ayuda y motivación.

En segundo lugar, a mi tutores Dr. Fernando Gómez Bravo, director del Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Huelva, y Dr. Rafael López De Ahumada Gutiérrez, profesor del grado de Ingeniería Electrónica Industrial, por introducirme en el campo de la investigación y guiarme durante la realización de este trabajo.

Por último, a todo el profesorado del Doble Grado de Ingeniería Electrónica Industrial e Ingeniería Mecánica, por contribuir a mi aprendizaje.

"Cuéntamelo y lo olvidaré. Enséñamelo y quizás lo recordaré. Hazme partícipe y lo aprenderé."

Benjamin Franklin

Resumen

De acuerdo con el plan propuesto por la Unión Europea “Blue Growth Plan”, la explotación de los recursos pesqueros basada en una acuicultura inteligente, sostenible y con una producción inclusiva constituirá un recurso de alimentos de gran importancia para la población europea.

Por tanto, es fundamental utilizar diferentes soluciones tecnológicas que apoyen la explotación sostenible de las granjas acuícolas y garanticen su viabilidad. Entre las tecnologías emergentes más prometedoras cabe destacar el uso de Vehículos Submarinos Operados Remotamente (ROVs), gracias a su capacidad para obtener información utilizando técnicas no destructivas y mínimamente intrusivas.

La utilidad de estos vehículos en tareas de mantenimiento y explotación de instalaciones acuícolas es múltiple. Entre sus diferentes aplicaciones cabe destacar la localización de elementos de interés, como peces u otro tipo de objetos de naturaleza variada, cuya presencia en el estanque de la granja pueda condicionar su explotación, y cuya localización y visita resulten de interés para la adecuada gestión de la misma.

Es por ello que este Trabajo de Fin de Grado se centra en el diseño de un sistema de control que permitirá convertir el vehículo en un ROV Híbrido (HROV); es decir, dotar al vehículo submarino de cierta capacidad de autonomía, manteniendo la posibilidad de la interacción continua con el operador. Esto permitirá que el usuario pueda definir ciertas tareas que el ROV realizará por sí solo. Por ejemplo, se podrán definir objetivos de interés hacia los que la nave navegará de forma autónoma, o establecer la ejecución otro tipo de maniobras, como la realización de transeptos. Todo basándose en la información del entorno proporcionada por sensores acústicos (sonares) y el resto de sensores que incorpora el vehículo.

Palabras clave

Vehículos submarinos operados remotamente, ROV, navegación autónoma, sonar, ROV Híbridos, Control de Navegación.

Abstract

According to the Blue Growth Plan proposed by the European Union, the exploitation of fishery resources based on smart, sustainable, and inclusive aquaculture production will constitute a relevant resource of food for the European population.

Therefore, it is essential to use different technological solutions that support the sustainable exploitation of aquaculture farms and ensure their viability. Among the most promising emerging technologies to be applied, it is worth highlighting the use of Remotely Operated Vehicles (ROVs), due to their ability to obtain information using non-destructive and minimally intrusive techniques.

The usefulness of these vehicles in maintenance and operation tasks of aquaculture facilities is multiple. Among its different applications, it is worth highlighting the location of elements of interest, such as fish or other types of objects of a varied nature, whose presence in the farm pond can influence its exploitation, and whose positioning and visit are of interest for its proper management.

For this reason, this Final Degree Project focuses on the design of a control system that will allow the vehicle to become a Hybrid ROV (HROV); that is, a system with a certain degree of autonomy, maintaining the possibility of continuous interaction with the operator. Through this capability, the user could define specific tasks that the ROV will perform on its own. For example, objectives of interest can be defined so that the vehicle navigates autonomously towards them, or other types of maneuvers, such as the execution of transepts. All based on the environment information provided by the acoustic sensors (sonars) and the set of sensors integrated in the vehicle.

Key words

Underwater Remotely Operated Vehicles, ROV, autonomous navigation, sonar, Hybrid ROV, Navigation Control.

Lista de abreviaturas

AUV	Vehículo Submarino Autónomo (Autonomous Underwater Vehicle)
DMP	Procesador de movimiento digital (Digital Motion Processor)
EKF	Filtro de Kalman Extendido (Extended Kalman Filter)
ESC	Controlador electrónico de velocidad (Electronic Speed Control)
GCS	Unidad de control de tierra (Ground Control Station)
GNC	Guía, navegación y control (Guidance, Navigation and Control)
GNSS	Sistema de navegación global por satélite (Global Navigation Satellite System)
GPS	Sistema de posicionamiento global (Global Positioning System)
UGPS	Sistema de posicionamiento global submarino (Underwater Global Positioning System)
HAL	Capa de abstracción de hardware (Hardware Abstraction Layer)
HROV	Vehículo Operado Remotamente Híbrido (Hybrid Remotely Operated Vehicle)
IMU	Unidad de Medida Inercial (Inertial Measurement Unit)
LQR	Regulador Lineal Cuadrático (Linear Quadratic Regulator)
MAVlink	Micro Air Vehicle Link
MEMS	Sistemas microelectromecánicos (Micro-electro-mechanical Systems)
MIMO	Múltiples entradas y múltiples salidas (Multiple Input Multiple Output)
PWM	Modulación de ancho de pulso (Pulse Width Modulation)
RMS	Valor cuadrático medio (Root Mean Square)
ROV	Vehículo Operado Remotamente (Remotely Operated Vehicle)
RTOS	Sistema operativo de tiempo real (Real Time Operating System)
UGPS	Sistema de posicionamiento global submarino (Underwater Global Positioning System)
UKF	Filtro de Kalman Unscented (Unscented Kalman Filter)

Lista de figuras

Índice

Capítulo 1

Introducción

Este Trabajo Fin de Grado (TFG) se centra en el diseño de un sistema de control a implementar sobre el Sibiu PRO. Este es un Vehículo Operado Remotamente (ROV) sumergible, con el que se pueden desarrollar trabajos de inspección y mantenimiento de instalaciones acuáticas de manera eficiente y segura, ya que cuenta con la capacidad de recoger y transmitir información sobre el estado general del entorno que lo rodea.

1.1. Antecedentes

El presente trabajo constituye el Trabajo Fin de Grado de Ingeniería Electrónica Industrial, siendo el primero de dos TFG realizados para la obtención del título de Doble Grado de Ingeniería Electrónica Industrial e Ingeniería Mecánica por la Universidad de Huelva. Ha sido realizado por Olga Marín Cañas, estudiante del mencionado doble grado. Se lleva a cabo bajo la supervisión del Dr. Fernando Gómez Bravo, director del Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Huelva, y del Dr. Rafael López De Ahumada Gutiérrez, profesor del grado de Ingeniería Electrónica Industrial.

Este TFG se enmarca en el proyecto internacional KTT SeaDrones (POCTEP 0622_KTTSEADRONES_5_E) cofinanciado por el Fondo Europeo de Desarrollo Regional FEDER a través del Programa Interreg V-A España-Portugal (POCTEP) 2014-2020. Este proyecto promueve el conocimiento y la transferencia de tecnología sobre vehículos aéreos y acuáticos para el desarrollo transfronterizo de ciencias marinas y pesqueras. La innovación tecnológica y la transferencia son factores esenciales para lograr el desarrollo del sector pesquero-acuícola regional a la vez que se protege el medio ambiente y se fomenta un uso racional y sostenible de los recursos naturales.

En concreto, este TFG se ha llevado a cabo en los últimos meses de duración del proyecto, por lo que su contenido se ha desarrollado a partir del trabajo previo realizado por otros investigadores.

En este trabajo se han desarrollado las siguientes competencias vinculadas con el Grado en Ingeniería Electrónica Industrial de la Universidad de Huelva:

Competencias básicas	
CB1	Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área general de estudio que parte de la base de educación secundaria general, y se suele encontrar un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
CB2	Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
CB3	Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes (normalmente dentro de su área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
CB4	Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
CB5	Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Competencias generales	
G01	Capacidad para la resolución de problemas.
G02	Capacidad para toma de decisiones.
G03	Capacidad de organización y planificación.
G04	Capacidad de aplicar los conocimientos en la práctica.
G06	Actitud de motivación por la calidad y mejora continua.
G07	Capacidad de análisis y síntesis.
G08	Capacidad de adaptación a nuevas situaciones.
G09	Creatividad y espíritu inventivo en la resolución de problemas científico-técnicos.
G10	Capacidad para comunicarse con personas no expertas en la materia.
G12	Capacidad para el aprendizaje autónomo y profundo.
G14	Capacidad de gestión de la información en la solución de situaciones problemáticas.
G16	Sensibilidad por temas medioambientales.
G17	Capacidad para el razonamiento crítico.

Competencias transversales	
TC2	Desarrollo de una actitud crítica en relación con la capacidad de análisis y síntesis.
TC3	Desarrollo de una actitud de indagación que permita la revisión y avance permanente del conocimiento.
TC4	Capacidad de utilizar las Competencias Informáticas e Informacionales (CI2) en la práctica profesional.

Competencias específicas	
B01	Capacidad para la resolución de los problemas matemáticos que puedan plantearse en la ingeniería. Aptitud para aplicar los conocimientos sobre: álgebra lineal; geometría; geometría diferencial; cálculo diferencial e integral; ecuaciones diferenciales y en derivadas parciales; métodos numéricos; algorítmica numérica; estadística y optimización.
B02	Comprensión y dominio de los conceptos básicos sobre las leyes generales de la mecánica, termodinámica, campos y ondas y electromagnetismo y su aplicación para la resolución de problemas propios de la ingeniería.
B03	Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
B05	Capacidad de visión espacial y conocimiento de las técnicas de representación gráfica, tanto por métodos tradicionales de geometría métrica y geometría descriptiva, como mediante las aplicaciones de diseño asistido por ordenador.
C05	Conocimientos de los fundamentos de la electrónica.
C06	Conocimientos sobre los fundamentos de automatismos y métodos de control.
E03	Conocimiento de los fundamentos y aplicaciones de la electrónica digital y microprocesadores.
E05	Conocimiento aplicado de instrumentación electrónica.
E07	Conocimiento y capacidad para el modelado y simulación de sistemas.
E08	Conocimientos de regulación automática y técnicas de control y su aplicación a la automatización industrial.
E09	Conocimientos de principios y aplicaciones de los sistemas robotizados.
E10	Conocimiento aplicado de informática industrial y comunicaciones.
E11	Capacidad para diseñar sistemas de control y automatización industrial.

1.2. Motivación

En los últimos años, se han desarrollado diferentes soluciones tecnológicas que apoyan la explotación sostenible de instalaciones acuáticas, tales como piscifactorías o granjas acuícolas. Entre las tecnologías emergentes más prometedoras destaca el uso de vehículos submarinos operados remotamente, gracias a su capacidad para obtener información utilizando técnicas no destructivas y mínimamente intrusivas.

En este sentido, el proyecto se alinea con el plan propuesto por la Unión Europea ”Blue Growth Plan”, donde la explotación de los recursos pesqueros, basada en una acuicultura inteligente, sostenible y con una producción inclusiva, jugará un papel fundamental en el futuro y constituirá un recurso relevante de alimentos para la población europea.

Por tanto, los resultados de este proyecto contribuirán al desarrollo e implantación de soluciones tecnológicas que apoyen la explotación eficiente y segura de las granjas acuícolas, y garanticen su viabilidad.

1.3. Objetivos

Habitualmente, el ROV sirve de auxilio al operador para realizar labores de supervisión de la instalación, así como de identificación de obstáculos de interés, entre otras. Para llevar a cabo estas tareas, el Sibiu PRO cuenta con un equipamiento adicional: cámara, ecosonda, sonar Ping360, sistema de localización Water Linked y sensores inerciales.

Por esta razón, el objetivo principal de este TFG es diseñar un sistema de control que auxilie al operador en su trabajo a realizar en una instalación acuática, convirtiendo el vehículo en un ROV Híbrido (HROV). Concretamente, el propósito del sistema de control es que el operador no tenga que encargarse de conducir el ROV manualmente, ya que el control del desplazamiento será automático, y pueda dedicar su atención a analizar la información recibida en tiempo real por los sensores.

El desarrollo de HROVs es especialmente interesante para su aplicación en determinadas instalaciones acuáticas, en las que debido a la profundidad a la que se encuentra el ROV, o a la escasa visibilidad dentro del agua debida a la turbidez de la misma, el operario no es capaz de dirigir al ROV adecuadamente, por lo que se hace necesario el control automático.

En concreto, el funcionamiento del control automático diseñado en este TFG es el siguiente. El operario debe seleccionar, por medio de un interfaz, uno de los objetos detectados por el sensor de ultrasonidos y, a continuación, el ROV debe desplazarse hacia el objeto en cuestión, hasta situarse a una determinada distancia de este.

El desarrollo de los objetivos del proyecto, dada su mayor complejidad, se divide en diferentes paquetes de trabajo, que se describen a continuación.

⌦ Primer paquete de trabajo: *desarrollo de los controladores básicos*

Tarea 1

Introducir por teclado u otro interfaz un ángulo de referencia, y comprobar que el ROV se reorienta adecuadamente utilizando un controlador de orientación básico.

Tarea 2

Navegación hacia delante hasta alcanzar una distancia con el objeto frontal. Existen ciertos aspectos a considerar:

- Limitar el rango de muestreo del sonar.
- Limitar la distancia a la que se acerca al objeto, para que no se confunda con ruido que genera el propio ROV.
- Disminuir progresivamente su velocidad a medida que se acerca al punto final.
- Probar primero con la pared de la piscina, y luego con un obstáculo situado en la esquina.
- Desarrollar un controlador que mantenga el ángulo de heading mientras se avanza.

⌦ Segundo paquete de trabajo: *integración de los controladores y desarrollo del controlador final*

Tarea 3

Incluir la información de los sensores inerciales.

Tarea 4

Desarrollo de un controlador continuo.

⌦ Tercer paquete de trabajo: *desarrollo de la aplicación final*

Tarea 5

Selección manual del objetivo a partir de la información inicial que nos proporciona el sonar Ping360.

1.4. Estado del arte y definición del proyecto

El control automático de barcos y otros vehículos acuáticos comenzó a desarrollarse ampliamente a partir de la invención del giroscopio en 1908 [1][13]. Este fue el instrumento básico de los primeros sistemas de control de la trayectoria de vehículos submarinos, ya que permitió una retroalimentación fiable del ángulo de guiñada de la nave.

El desarrollo de sistemas de posicionamiento local en la década de 1970 supuso otro gran avance para el control automático de vehículos acuáticos. Posteriormente, en 1994, se comenzaron a emplear los sistemas de navegación por satélite.

En la actualidad, los sistemas de control de movimiento para barcos y otros vehículos acuáticos constan tradicionalmente de tres bloques independientes: Guía, Navegación y Control (GNC) [21]. Estos bloques interactúan entre ellos mediante la transmisión de datos y señales, tal como se muestra en la Figura 1.

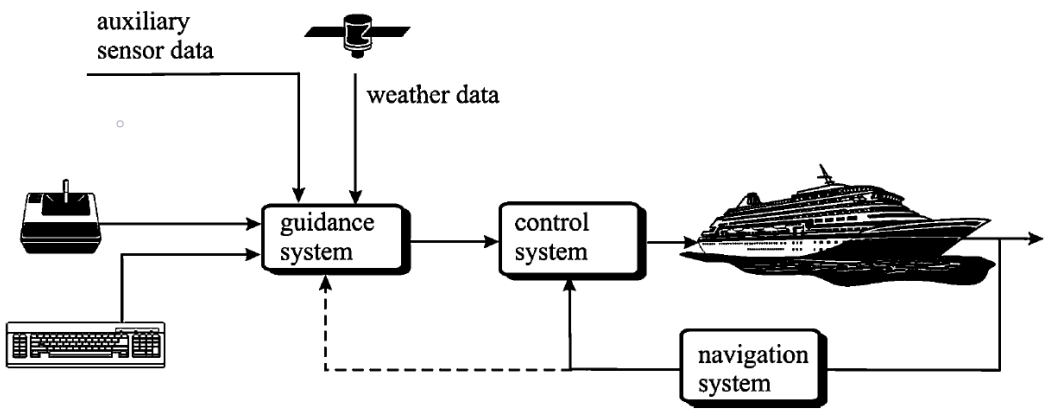


Fig. 1. Diagrama de GNC [1]

El módulo de Guiado se encarga de calcular continuamente la posición de referencia, la velocidad y aceleración de la embarcación. Este bloque emplea la información procedente de entradas de joystick o teclado, así como otros datos del medio externo: datos meteorológicos (velocidad y dirección del viento, altura de las olas, velocidad y dirección de la corriente, etc), datos topológicos de la Tierra (datos de radar y sonar) y, ocasionalmente, datos proporcionados por el módulo de Navegación. La información recogida es procesada y trasmitida posteriormente al bloque de Control.

El bloque de Navegación es responsable de determinar la posición, orientación (actitud), rumbo y distancia recorrida por la nave. Para ello, generalmente emplea

un sistema de navegación global por satélite (GNSS) junto con sensores de movimiento, tales como acelerómetros y giroscopios.

El bloque de Control se encarga de determinar las fuerzas y momentos necesarios que debe generar el vehículo para satisfacer un determinado objetivo de control. Este objetivo puede ser la regulación de un valor de referencia, seguimiento de trayectorias, control de maniobras, etc. En general, dicho objetivo es proporcionado por el módulo de Guiado.

Además de contar con la arquitectura tradicional de Guiado, Navegación y Control, los sistemas de control modernos se basan en una variedad de técnicas de diseño más complejas, tales como control PID, control óptimo lineal cuadrático (LQR), control estocástico, H_∞, redes neuronales, etc.

Estas técnicas tratan de reducir los diversos factores que dificultan el control autónomo de un vehículo submarino, entre los que destacan los siguientes:

- Los vehículos acuáticos son sistemas no lineales, con múltiples entradas y múltiples salidas (MIMO).
- La resolución y frecuencias de actualización de los sistemas de posicionamiento submarino (UGPS), así como de los sensores sonar no son muy elevadas.
- Perturbaciones externas impredecibles causadas por corrientes submarinas, el cable de conexión, etc.

Todos estos factores deben ser tenidos en cuenta al diseñar un controlador para un vehículo submarino.

Esta memoria se compone de 6 capítulos:

- En el primer capítulo se realiza una introducción al control automático de vehículos marinos, y se establecen las diferentes etapas a llevar a cabo durante el diseño del sistema de control propuesto.
- El segundo capítulo describe el Sibiu PRO, vehículo en el que se va a implementar el sistema de control, tanto sus características hardware como su arquitectura software.
- El tercer capítulo se centra en el ángulo de heading, un parámetro fundamental durante la navegación del vehículo. Contiene dos métodos para su estimación, el primero es mediante la IMU del ROV, y el segundo es mediante sensores sonar.
- El cuarto capítulo incluye una descripción de las diferentes etapas en las que se divide el algoritmo de control diseñado.

- El quinto capítulo contiene las descripciones y resultados de las pruebas experimentales llevadas a cabo para comprobar la exactitud de las medidas de la IMU, así como las pruebas realizadas durante el proceso de diseño del algoritmo de control.
- El sexto capítulo termina con las conclusiones obtenidas de todo el proceso experimental llevado a cabo a lo largo del presente documento, así como sugerencias de trabajo futuro.

Capítulo 2

Descripción del Sibiu PRO

Los vehículos submarinos se clasifican generalmente en dos categorías: vehículos tripulados y no tripulados. Dentro de los vehículos no tripulados, podemos distinguir entre Vehículos Submarinos Autónomos (AUVs), Vehículos Operados Remotamente (ROVs) y Vehículos Operados Remotamente Híbridos (HROVs).

Un AUV puede viajar bajo el agua largas distancias de forma independiente, sin tener cables conectados ni recibir indicaciones o comandos de los operadores. En cambio, un ROV es controlado por un operador a través de un cable umbilical y, generalmente, opera a bajas velocidades.

La ventaja que presentan los HROVs es que, además de poder realizar las mismas funciones automatizadas que un AUV, permiten la intervención humana o incluso el control humano total en su navegación. De esta forma, en un HROV podemos complementar el control automático y manual, según lo requieran las circunstancias.

El vehículo sobre el que se ha implementado el sistema de control diseñado es el Sibiu Pro (ver Figura 2), comercializado por *Nido Robotics*. Este ROV fue adquirido por el grupo de Electrónica y Robótica Inteligentes TIC-266 de la Universidad de Huelva en el marco del proyecto internacional KTT SeaDrones.

Fig. 2. Sibiu PRO [2]

A continuación, se detallan sus características principales y se describe su arquitectura hardware y software.

2.1. Características principales

El Sibiu PRO incorpora ocho propulsores, lo que le concede suavidad y estabilidad en la navegación, así como una cámara de 1080p, específicamente optimizada para el entorno submarino, y cuatro luces regulables de 1500 lúmenes. Utiliza baterías recargables de litio ($LiFePO_4$) como fuente de energía para su funcionamiento bajo el agua.

Sus especificaciones técnicas se muestran a continuación [2]:

Peso	16 kg
Tamaño	52 x 39 x 29 cm
Profundidad máxima	300 m
Velocidad máxima	3 nudos
Cámara	1080p
Luces	4x1500 lúmenes

Tabla 1. Especificaciones técnicas del Sibiu PRO

2.2. Sensores

El Sibiu PRO incluye un conjunto de sensores integrados en su plataforma, cuya información es vital para una correcta navegación.

2.2.1. Unidad de medida inercial

La unidad de medida inercial (IMU) está formada por dos giroscopios, dos acelerómetros y dos magnetómetros, todos triaxiales. Estos sensores son sistemas microelectromecánicos (MEMS), lo que resulta en una IMU de pequeño tamaño y económica. Se encuentran repartidos en tres módulos [3]:

- MicroL3GD20H (de STMicroelectronics)
- MPU6000 (de TDK InvenSense)
- LSM303D (de STMicroelectronics)

El módulo MicroL3GD20H integra un giroscopio. El módulo MPU6000 integra el segundo giroscopio y un acelerómetro, así como un procesador de movimiento digital (DMP). Su tasa de muestreo es de 100-1000 Hz. El módulo LSM303D integra otro acelerómetro y un magnetómetro. Su tasa de muestreo es de 800 Hz. Estos módulos trasmiten información a la PixHawk mediante los protocolos de comunicación serie I^2C o SPI.

La combinación de acelerómetro, magnetómetro y giroscopio es bastante común. Mediante la fusión de estos tres sensores, podemos obtener la orientación rotacional absoluta (balanceo, inclinación y guiñada) del ROV, con respecto al sistema de referencia de la estructura del vehículo.

En cuanto a la exactitud de estos sensores, cabe destacar el acelerómetro es muy sensible a vibraciones, mientras que el magnetómetro puede ser influenciado por la proximidad a los metales y perturbaciones eléctricas en general.

2.2.2. Sensor de presión

El sensor de presión que utiliza el ROV es el MS5837-30BA, que tiene una resolución de 0.2 mbar [19]. Este sensor puede dar lecturas hasta 300 m, aproximadamente. Su frecuencia de actualización es de 50 Hz.

2.2.3. Sonar Ping360

Los sensores SONAR (SOund Navigation And Ranging) son un tipo de sonar activo que funciona transmitiendo pulsos de sonido al agua y registrando los ecos que se devuelven a medida que cada pulso de sonido se refleja en los objetos frente a él. Son especialmente útiles en circunstancias donde la visibilidad es baja para obtener imágenes acústicas de objetivos, así como para la navegación, en ausencia de un sistema de posicionamiento alternativo.

Fig. 3. Sonar Ping360

Los sonares de barrido, como lo es el Ping360 mostrado en la Figura 3, se definen por tener un haz acústico en forma de “abanico” con un haz vertical ancho y un haz horizontal estrecho [4], que le permite obtener secciones transversales acústicas del entorno, tal como se muestra en la Figura 4.

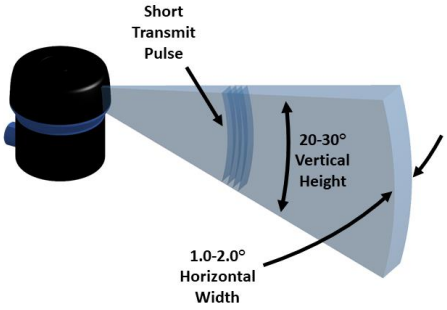


Fig. 4. Haz acústico del sonar [4]

Este transductor está montado sobre un motor que lo gira en incrementos de 0.9 grados y, mientras lo hace, genera una imagen circular de los alrededores de la sonda con un alcance máximo de 50 metros (ver Figuras 5 y 6).

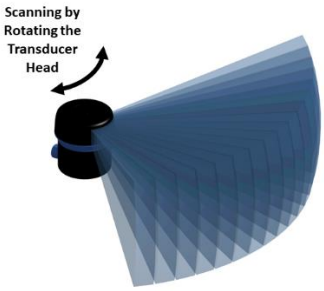


Fig. 5. Muestreo del sonar

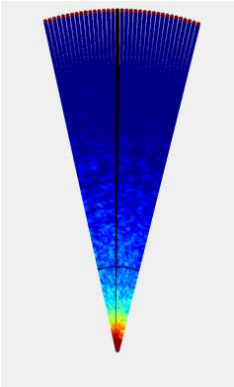


Fig. 6. Imagen obtenida con el sonar

Al combinar la velocidad conocida del sonido en el agua con el tiempo en el que se recibieron los ecos, el sonar puede calcular la distancia que ha viajado el sonido. La ecuación para determinar la distancia acústicamente es la siguiente:

$$distancia = \frac{1}{2} \cdot velocidad\ del\ sonido\ en\ el\ agua \cdot tiempo \quad (1)$$

Por lo general, la velocidad del sonido en agua salada es de aproximadamente 1500 m/s, pero esto puede variar según la temperatura del agua, la salinidad y la profundidad de funcionamiento del sonar.

Los objetivos materiales con densidades muy diferentes a las del agua (como gas, roca, hormigón o metal) serán muy reflectantes y tendrán fuertes ecos. Los ecos de materiales como barro, limo, arena y plantas serán más débiles, ya que tienen una densidad similar a la del agua o absorben energía acústica.

El sonar Ping360 obtiene 1200 valores de la intensidad del eco en cada ángulo. Cada uno de ellos hace referencia a una distancia concreta y consiste en un valor entre 0 y 255 (sin rebote e intensidad máxima, respectivamente). Estos se muestran en el interfaz de visualización a través de una paleta de colores, teniendo un color diferente dependiendo de la intensidad del rebote.

En determinadas circunstancias donde las condiciones de visibilidad sean mínimas, la información del sonar acústico será imprescindible, puesto que la función de la cámara será mínima.

2.2.4. Ecosonda PingSonar

Este sensor nos permite conocer la distancia al fondo de la instalación acuática donde se encuentre el ROV. Su funcionamiento se basa en un transductor piezoeléctrico, que envía un pulso acústico al agua y luego escucha el retorno del eco. Con esa información puede determinar la distancia recorrida por el eco más fuerte, que suele ser el fondo del océano o un objeto grande.

La ecosonda PingSonar utiliza una frecuencia de 115 kHz [19], diferente de las que se utilizan en la mayoría de las ecosondas de barcos para evitar interferencias. Tiene un rango de medición de 30 metros y un ancho de haz de medición de 30 grados.

2.2.5. Sonar de barrido lateral

El sonar de barrido lateral (modelo BR-ROV), mostrado en la Figura 7, consta de dos partes: la caja de la sonda y los transductores [19]. Ambas están montadas en el exterior del ROV. Su funcionamiento es similar a los otros dos sensores sonar explicados anteriormente.

Fig. 7. Sonar de barrido lateral

2.3. Arquitectura de control

En el Sibiu PRO, el sistema electrónico que controla su funcionamiento está formado por una Raspberry Pi y una PixHawk. Además, debemos disponer de un ordenador donde ejecutar el programa que nos permite comunicarnos con el ROV, ya sea *QGroundControl* o un script en *Matlab*. Es decir, el PC actúa como estación de tierra virtual (GCS). Esta arquitectura se muestra en la Figura 8.

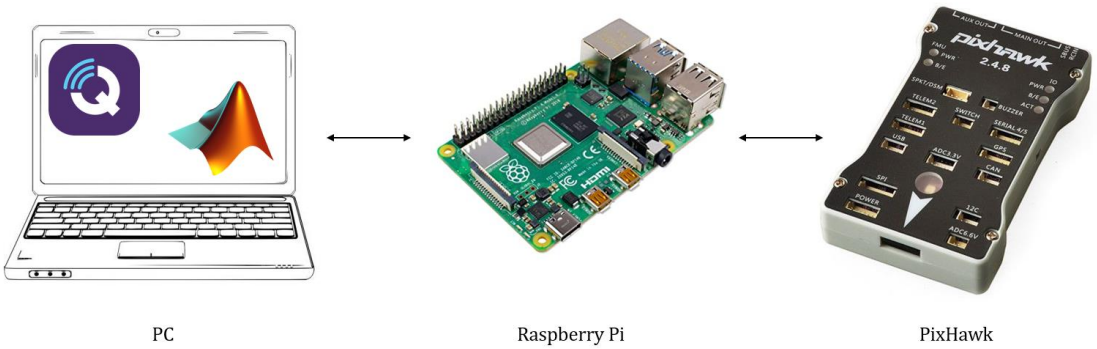


Fig. 8. Arquitectura básica de control del Sibiu PRO

La Raspberry Pi es un ordenador reducido en una pequeña placa, mientras que la PixHawk es un autopiloto o controlador de vuelo, muy común en los drones aéreos. Ambos elementos se describen a continuación.

2.3.1. Raspberry Pi

La Raspberry Pi es un ordenador de placa única de uso general que ejecuta Linux como sistema operativo embebido. En el ROV, la Raspberry Pi se utiliza como computadora complementaria (companion computer) del piloto automático PixHawk.

Fig. 9. Raspberry Pi

Su función principal es transmitir los mensajes entre la PixHawk y el ordenador, utilizando el protocolo de comunicación MAVLink [19]. Para ello, utiliza una

dirección IP fija (192.168.2.2), de la misma forma que el PC debe estar configurado en la dirección 192.168.2.1, tal como muestra la Figura 10:

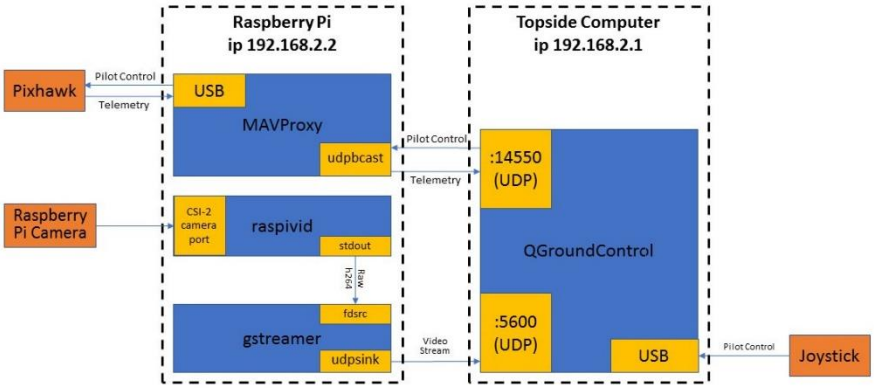


Fig. 10. Diagrama de software entre la Raspberry Pi y el PC [19]

La comunicación entre la Raspberry Pi y el PC se realiza mediante Ethernet, mientras que la comunicación entre esta y la PixHawk se lleva a cabo mediante USB. En los demás puertos USB de la Raspberry Pi se conectan los siguientes periféricos (ver Figura 11):

- Cámara
- Sonar de barrido lateral
- Sonar Ping360

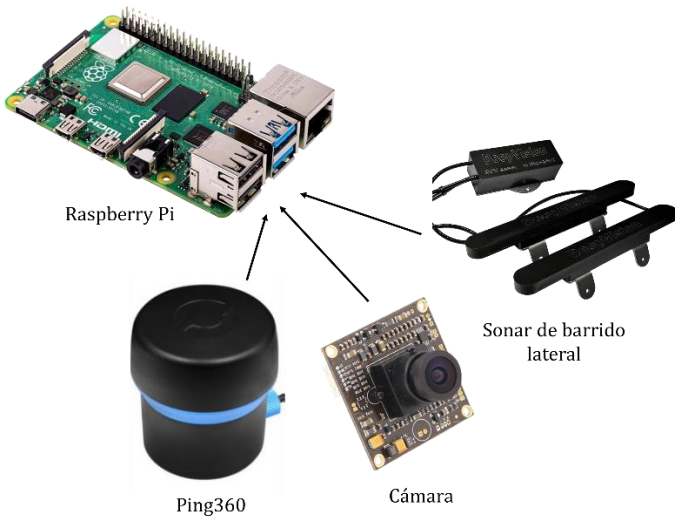


Fig. 11. Conexionado mediante USB de diferentes sensores a la Raspberry Pi

2.3.2. PixHawk

PixHawk es un controlador de vuelo de bajo coste y alto rendimiento. Un controlador de vuelo o autopiloto es la forma más básica de un sistema GNC (Guía, Navegación y Control). Integra el sensor de presión y la IMU, siendo una de sus principales características la capacidad de soportar sensores duales.

Se encarga de recibir los datos procedentes de los sensores, ejecutar los algoritmos de fusión de datos, controlar los propulsores y comunicarse con la Raspberry Pi (companion computer), que está ejecutando nuestro algoritmo de control.

A continuación, se describen algunos aspectos referentes al hardware y software.

Hardware

La ventaja principal de este controlador de vuelo es que es de hardware libre [5]. Su encapsulado incluye conectores dedicados para diferentes periféricos (sensores, motores, etc.), tal como se muestra en la Figura 12.

Fig. 12. Encapsulado de la PixHawk

Sin embargo, su modularidad no es completa, ya que no es posible cambiar la IMU o el barómetro que lleva incorporados, aunque sí pueden ser calibrados y configurados.

En el Sibiu PRO, la PixHawk se encarga de controlar los siguientes actuadores:

- Los ocho motores DC acoplados a los propulsores, a través de controladores electrónicos de velocidad (ESC). Un ESC es un circuito electrónico que permite variar la velocidad y dirección de un motor eléctrico. La PixHawk

envía señales de modulación de ancho de pulso (PWM) a los ESC para regular la velocidad de estos propulsores.

- Las cuatro luces.
- El servomotor que gira la cámara con un movimiento de guiñada.

El microcontrolador de 32 bits que integra es el STM32f427 (de STMicroelectronics). El núcleo de estos microcontroladores es el procesador Arm® Cortex®-M4. Presentan un elevado rendimiento, capacidad de respuesta en tiempo real, procesamiento de señales digitales y funcionamiento a baja potencia y baja tensión. Sus especificaciones técnicas se muestran a continuación:

256 kB SRAM
2 MB Flash
CPU 180 MHz
Temporizadores de 16/32 bits

Software

La PixHawk se encarga de ejecutar ArduSub, que es un software de código abierto especialmente diseñado para AUVs y ROVs derivado de ArduPilot. ArduPilot es, junto con PX4, uno de los softwares más utilizados actualmente para el control de vehículos no tripulados, ya que cumple con los requisitos de respuesta en tiempo real que requieren este tipo de sistemas, y además presenta potentes herramientas de registro y análisis de datos [6], [7]. Este software es soportado por numerosas plataformas y, al ser de código abierto, se encuentra en constante actualización.

ArduPilot basa su funcionamiento en un sistema operativo de tiempo real (RTOS) denominado ChibiOS. Este sistema operativo compacto y rápido está especialmente diseñado para sistemas embebidos. Se encarga de acceder a las funcionalidades del microcontrolador y gestionar su funcionamiento.

Además, para garantizar su portabilidad y adaptación a diferentes arquitecturas hardware, ArduPilot se basa en diferentes capas de abstracción [7], [8], tal como se muestra en la Figura 13. Cada una de estas capas se describe a continuación.

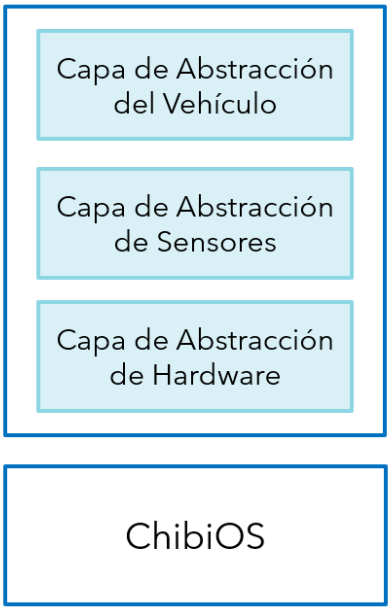


Fig. 13. Arquitectura de ArduPilot

- **Capa de abstracción de hardware (HAL)**

La abstracción de hardware implica transparencia con respecto a los detalles técnicos de la placa en la que se ejecuta ArduPilot. Por lo tanto, cada tipo de placa tiene su propia HAL.

Esta capa de abstracción se encarga de que el resto de la arquitectura sea completamente independiente del hardware. De esta forma, el resto de capas utilizan exclusivamente las abstracciones definidas por la HAL para interactuar con los periféricos y las memorias.

Por ejemplo, para recuperar los datos medidos por un sensor, el driver de este sensor manda la información a la capa HAL mediante el protocolo correspondiente (I^2C, SPI, UART, etc.). Posteriormente, la capa HAL transmite la información obtenida a las capas superiores.

La variedad de sensores que se pueden conectar a la PixHawk también se gestiona en la capa HAL. Podemos distinguir dos tipos de sensores: los que están integrados en la PixHawk (IMU y barómetro) y los que se pueden añadir.

En cuanto a estos últimos, ArduPilot escanea los diferentes puertos de comunicación en el arranque, y establece la lista de sensores disponibles al reconocerlos mediante sus identificadores.

- **Capa de abstracción de sensores**

El segundo nivel de abstracción afecta a los sensores. Permite al usuario centrarse en la información obtenida a partir de los mismos, en lugar de en la forma en que esta ha sido adquirida.

En esta capa se integran diferentes bloques de procesamiento de información, como el Filtro de Kalman Extendido (EKF), que se explica posteriormente.

- **Capa de abstracción del vehículo**

El tercer nivel de abstracción está relacionado con la estructura física del vehículo. Permite definir las características constructivas de la plataforma (vehículo aéreo, vehículo submarino, etc.) y sus especificaciones técnicas (número de motores, su posición en relación con el centro de gravedad, su sentido de giro, etc.).

Además, incluye varios procedimientos de seguridad y protección adaptados a cada configuración (aérea, submarina o terrestre).

2.3.3. Otros componentes de la arquitectura

Caja de superficie

Para conectar nuestro ordenador con el ROV, se utiliza una caja de superficie, tal como muestra la Figura 14.

Fig. 14. Arquitectura de control

La comunicación entre el ROV y la caja de superficie se realiza a través de un cable umbilical, permitiendo obtener los datos de la cámara, sensores y sonares situados en el vehículo.

Placas Fathom-X

Tanto en el interior de la caja de superficie como del ROV, se encuentran las placas Fathom-X (ver Figura 15), que proporcionan una robusta conexión ethernet, de alta velocidad y larga distancia a través de un solo par de cables. Permiten la transmisión de vídeos HD y datos de gran ancho de banda a más de 300 m de longitud de conexión.

Fig. 15. Placa Fathom-X

De esta forma, la arquitectura de control completa se muestra en la Figura 16:

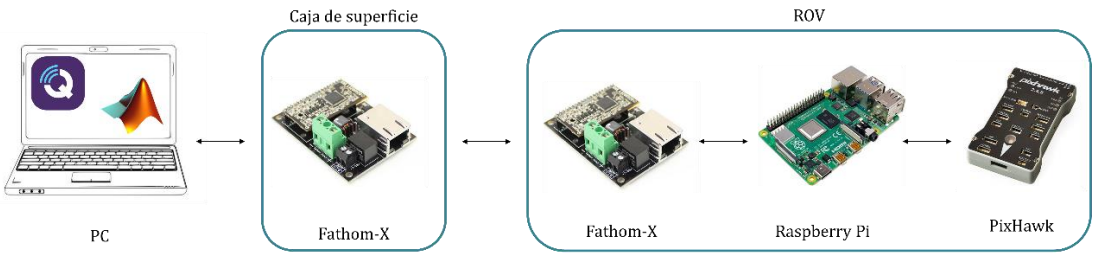


Fig. 16. Arquitectura de control completa

2.3.4. Visión general

En la Figura 17, podemos ver los diferentes componentes de la arquitectura, tanto sensores como elementos de control, así como las conexiones entre los mismos:

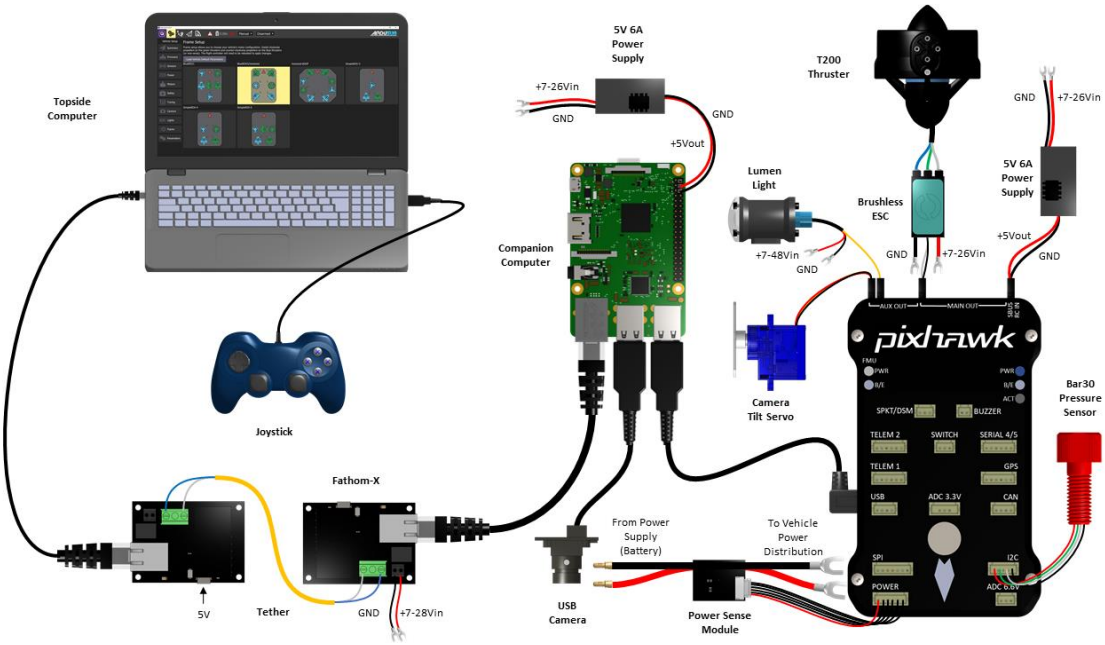


Fig. 17. Componentes de la arquitectura del Sibiu PRO [20]

Como vemos, en el PC (topside computer) se recibe y muestra la transmisión de vídeo y la información de telemetría. También permite que el operador controle el vehículo desde un joystick.

Los comandos introducidos en el PC se transmiten a través de las placas Fathom-X y el cable umbilical a la Raspberry Pi. Esta, además de gestionar la imagen de la cámara, manda la información correspondiente a la PixHawk.

Finalmente, la PixHawk se encarga de procesar los datos de la IMU y el sensor de presión que integra, así como de enviar órdenes a los diferentes actuadores (motores, luces y servomotor de la cámara).

2.4. Comunicación entre los componentes

A continuación se describen los dos protocolos de comunicación empleados entre los diferentes componentes del Sibiu PRO.

2.4.1. MavLink

La comunicación entre la PixHawk y el ordenador se lleva a cabo a través del protocolo Micro Air Vehicle Link (MAVlink) [9]. Este protocolo serie es el más usado para trasmitir datos y comandos entre vehículos y estaciones de tierra [10]. Su principal ventaja es que es muy fiable, ya que proporciona métodos para detectar pérdidas, corrupción y autenticación de paquetes.

Mediante este protocolo podemos recibir datos telemétricos, señal GPS, estado del ROV y otros muchos parámetros, además de poder enviarle comandos con los cuales el ROV podrá realizar misiones específicas decididas por el usuario en cualquier momento, gracias a la visión de los datos en tiempo real.

Existe un gran número de mensajes predefinidos en este protocolo, algunos de ellos son los siguientes [11]:

- HEARTBEAT: este mensaje debe ser mandado periódicamente. Indica que el dispositivo conectado se encuentra activo.
- GLOBAL_POSITION_INT: indica la posición global (ángulos de inclinación y datos de velocidad, entre otros, tal como muestra la Figura 18).

Field Name	Type	Units	Description
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).
lat	int32_t	degE7	Latitude, expressed
lon	int32_t	degE7	Longitude, expressed
alt	int32_t	mm	Altitude (MSL). Note that virtually all GPS modules provide both WGS84 and MSL.
relative_alt	int32_t	mm	Altitude above ground
vx	int16_t	cm/s	Ground X Speed (Latitude, positive north)
vy	int16_t	cm/s	Ground Y Speed (Longitude, positive east)
vz	int16_t	cm/s	Ground Z Speed (Altitude, positive down)
hdg	uint16_t	cdeg	Vehicle heading (yaw angle), 0.0..359.99 degrees. If unknown, set to: UINT16_MAX

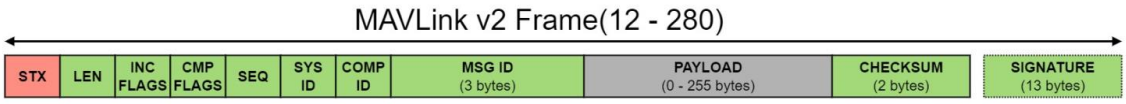
Fig. 18. Datos del mensaje GLOBAL_POSITION_INT

- PARAM_SET: nos permite establecer el valor de cualquier parámetro modificable.

- SYS_STATUS: informa del estado general del sistema, incluido el voltaje de la batería.
- ATTITUDE_QUATERNION: indica la inclinación del vehículo mediante cuaterniones, en lugar de ángulos.

Los mensajes se definen dentro de archivos XML. Cada archivo XML define el conjunto de mensajes admitido por un sistema MAVLink en particular, también denominado "dialecto". El conjunto de mensajes de referencia que implementan la mayoría de las estaciones de control de tierra y los pilotos automáticos se define en common.xml. Además, la mayoría de los dialectos se basan en esta definición.

En la Figura 19, se muestra el formato del paquete del protocolo MAVLink v2:



Byte Index	C version	Content	Value	Explanation
0	`uint8_t magic`	Packet start marker	0xFD	Protocol-specific start-of-text (STX) marker used to indicate the beginning of a new packet. Any system that does not understand protocol version will skip the packet.
1	`uint8_t len`	Payload length	0 - 255	Indicates length of the following `payload` section. This may be affected by payload truncation.
2	`uint8_t incompat_flags`	Incompatibility Flags		Flags that must be understood for MAVLink compatibility (implementation discards packet if it does not understand flag).
3	`uint8_t compat_flags`	Compatibility Flags		Flags that can be ignored if not understood (implementation can still handle packet even if it does not understand flag).
4	`uint8_t seq`	Packet sequence number	0 - 255	Used to detect packet loss. Components increment value for each message sent.
5	`uint8_t sysid`	System ID (sender)	1 - 255	ID of *system* (vehicle) sending the message. Used to differentiate systems on network. Note that the broadcast address 0 may not be used in this field as it is an invalid *source* address.
6	`uint8_t compid`	Component ID (sender)	1 - 255	ID of *component* sending the message. Used to differentiate *components* in a *system* (e.g. autopilot and a camera). Use appropriate values in MAV_COMPONENT. Note that the broadcast address `MAV_COMP_ID_ALL` may not be used in this field as it is an invalid *source* address.
7 to 9	`uint32_t msgid:24`	Message ID (low, middle, high bytes)	0 - 16777215	ID of *message type* in payload. Used to decode data back into message object.
For *n*-byte payload: `n=0` : NA, `n=1` : 10, `n>=2` : 10 to (9+n)	`uint8_t payload[max 255]`	Payload		Message data. Depends on message type (i.e. Message ID) and contents.
(n+10) to (n+11)	`uint16_t checksum`	Checksum (low byte, high byte)		CRC-16/MCRF4XX for message (excluding `magic` byte). Includes CRC_EXTRA byte.
(n+12) to (n+25)	`uint8_t signature[13]`	Signature		(Optional) Signature to ensure the link is tamper-proof.

Fig. 19. Formato del mensaje en MAVLink

Los mensajes incluidos en el dialecto common.xml se pueden clasificar en 3 categorías [19]:

- Comandos: en ellos se definen los valores de 7 parámetros, que hacen referencia a diferentes magnitudes dependiendo del ID del comando.

- Mensajes: determinan el estado general del sistema. Si el sistema sigue el estándar MAVLink, el estado del sistema se define principalmente por tres modos: MANUAL (sistema bajo control RC), HOLD DEPTH (modo para mantener la profundidad) y STABILIZE (permite movimientos más suaves). El resto de los modos automáticos no están disponibles para drones submarinos.
- Enumeraciones: son valores que definen la configuración del ROV.

Para utilizar este protocolo a través de Matlab nos hemos ayudado del paquete *UAV Toolbox*, que nos facilita la creación de los mensajes en este formato, la apertura de puertos y la creación de una GCS (Ground Control Station) virtual, como lo es QGroundControl. Este software toma los datos de telemetría enviados por el ROV a través del cable umbilical y los muestra al usuario mediante un interfaz.

Para crear la estación de tierra virtual (GCS), en primer lugar, hay que detallar el dialecto que vamos a usar (common.xml) y crear el objeto de la estación de tierra con las siguientes características: SystemID = 255, ComponentID = 190 y componente GCS. A continuación, conectamos ese objeto con el puerto UDP del PC adecuado (14550) y creamos un objeto cliente que se suscriba al nodo de la estación de tierra.

A continuación, procedemos a capturar o enviar los mensajes de interés. Para ello, MAVLink utiliza TShark, una herramienta que permite realizar análisis del tráfico de una red activa. A partir de los mensajes capturados, se genera un archivo de formato *.pcap*. Posteriormente, este se transforma al formato *.mat*, generando así un archivo con el que Matlab puede trabajar. Además, es necesario crear un objeto para recibir cada tipo de mensaje, procesarlo y obtener los datos. Esto se realiza mediante la función mavlinksub.

Por último, creamos los temporizadores que permiten enviar o recibir mensajes a una determinada frecuencia. Esto es de especial importancia para el mensaje HEARTBEAT, ya que como vimos anteriormente, ha de enviarse continuamente con una frecuencia específica. Una vez que se pone en marcha el timer de este mensaje, se completa la conexión con el ROV, y comenzamos a recibir el resto de los mensajes en la dirección IP fija.

Estos mensajes de interés (ATTITUDE, GLOBAL_POSITION_INT, etc.) también tienen asociado su propio timer. De esta forma, cada cierto tiempo especificado por el usuario en la creación del timer, se llama a una función que permite recibir la información de los mensajes, almacenándolos en un array struct. Este es un tipo de dato que agrupa diferentes variables mediante contenedores de datos denominados campos.

2.4.2. PingProtocol

La trasmisión de datos entre los sensores sonar (Ping360 y ecosonda) y la Raspberry Pi se realiza mediante el protocolo de comunicación PingProtocol. Este protocolo está diseñado para comunicación síncrona en una red de tipo maestro-esclavo [12], ya que el esclavo (sonar) solo envía información cuando el maestro (Raspberry Pi) lo solicita.

Para hacer una solicitud de la información del sonar ping360 y la ecosonda, enviamos el mensaje a la dirección IP de la Raspberry Pi del ROV y al puerto adecuado. El puerto pertinente a la ecosonda es el 9090, mientras que el sonar Ping360 usa el puerto 9092.

PingProtocol utiliza un formato de mensaje fijo, formado por un encabezado, una carga de datos y la verificación por suma (ver Figura 20).

Byte	Type	Name	Description
0	u8	start1	Start frame identifier, ASCII 'B'
1	u8	start2	Start frame identifier, ASCII 'R'
2-3	u16	payload_length	Number of bytes in payload.
4-5	u16	message_id	The message id.
6	u8	src_device_id	The device ID of the device sending the message.
7	u8	dst_device_id	The device ID of the intended recipient of the message.
8-n	u8[]	payload	The message payload.
(n+1)-(n+2)	u16	checksum	The message checksum. The checksum is calculated as the sum of all the non-checksum bytes in the message.

Fig. 20. Formato del mensaje en PingProtocol

Los mensajes de este protocolo están divididos en 4 categorías:

- General: de propósito general.
- Read/get: permiten responder a una solicitud de mensaje general del maestro. Con estos mensajes leeremos datos del dispositivo.
- Write/set: permiten configurar algunos parámetros del dispositivo. Con estos mensajes escribiremos datos para el dispositivo.
- Control: para realizar alguna acción más compleja.

Para obtener la información del sonar Ping360, en primer lugar, es necesario determinar el ángulo de la cabeza del rotor, así como otros parámetros (ver Figura 21). Para ello, usaremos un mensaje de tipo write/set. Algunos de los parámetros son modificables, y otros son fijos:

Parámetros modificables
Ganancia (baja, normal o alta)
Ángulo del rotor deseado
Duración de la transmisión acústica
Periodo de cada muestra
Distancia de muestreo
Sector de muestreo

Parámetros fijos
Modo de operación
Frecuencia de transmisión
Número de muestras (1200 datos para cada ángulo)
Transmisión automática de los datos tras cada toma

Type	Name	Description	Units
u8	mode	Operating mode (1 for Ping360)	
u8	gain_setting	Analog gain setting (0 = low, 1 = normal, 2 = high)	
u16	angle	Head angle	gradians
u16	transmit_duration	Acoustic transmission duration (1~1000 us)	microseconds
u16	sample_period	Time interval between individual signal intensity samples in 25 ns increments (80 to 40000 == 2 to 1000 us)	eicosapenta-nanoseconds
u16	transmit_frequency	Acoustic operating frequency (500~1000 kHz). It is only practical to use say 650 to 850 kHz due to the narrow bandwidth of the acoustic receiver.	kilohertz
u16	number_of_samples	Number of samples per reflected signal (supported values: 200~1200)	samples
u8	transmit	0 = do not transmit; 1 = transmit after the transducer has reached the specified angle	
u8	reserved	reserved	

Fig. 21. Parámetros del sonar Ping360

Mediante la modificación de ciertas variables, podemos optimizar la forma de obtener los datos para cada situación en la que se encuentre el ROV. De esta forma, es posible restringir la toma de datos a un sector angular concreto si fuese de interés para la tarea de exploración.

Capítulo 3

Estimación de la orientación del vehículo

Por lo general, un vehículo submarino como lo es el Sibiu PRO consta de 6 grados de libertad [13]. Estos se traducen en un conjunto de tres movimientos independientes a lo largo de tres direcciones X, Y, Z definidos como:

- Avance o *surge* (movimiento longitudinal)
- Ronza o *sway* (movimiento lateral)
- Ascenso o *heave* (movimiento vertical)

Así como tres rotaciones alrededor de los ejes X, Y, Z definidos como:

- Balanceo o *roll* (rotación sobre el eje X)
- Inclinación o *pitch* (rotación sobre el eje Y)
- Guiñada o *yaw* (rotación sobre el eje Z). Para referirse a este ángulo frecuentemente se utiliza el término *heading*.

Estos tres ángulos se conocen como ángulos de Euler. También se podría describir la orientación del vehículo mediante cuaterniones, aunque los ángulos de Euler son más intuitivos y fáciles de visualizar.

Estos 6 grados de libertad se ilustran en la Figura 22.

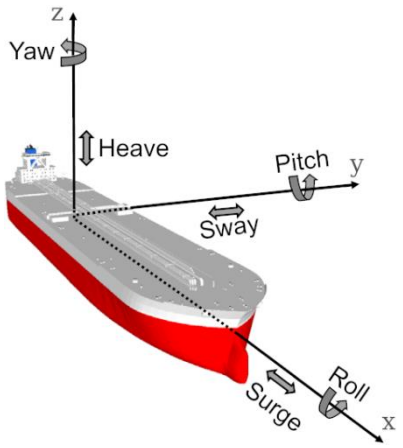


Fig. 22. Movimientos y rotaciones independientes de un vehículo submarino

En situaciones en las que el ROV se desplaza en el plano XY, el ángulo *yaw* o *heading* determina la orientación espacial del vehículo. Puesto que los movimientos automáticos desarrollados en este TFG tienen lugar en dicho plano, la estimación del ángulo de heading adquiere gran importancia para una correcta navegación.

En este trabajo, se han estudiado dos métodos para realizar dicha estimación: mediante los sensores inerciales de la IMU del ROV y mediante la información proporcionada por el sonar Ping360. A continuación, se describen ambos métodos.

3.1. Estimación mediante sensores inerciales

En este apartado, se describe el método empleado para estimar el ángulo de heading del vehículo a través de los sensores inerciales que incorpora el mismo.

3.1.1. Fusión sensorial

En los sistemas electrónicos actuales, es frecuente utilizar diferentes tipos de sensores o múltiples sensores del mismo tipo para tener una medida más completa o exacta de la magnitud de interés. Este concepto se denomina fusión sensorial.

Cuando se utilizan múltiples sensores, puede haber conflictos en las medidas. La jerarquía de los sensores y la fiabilidad de sus lecturas juegan, por tanto, un papel importante.

El algoritmo de fusión debe contar con información sobre los sensores, como el ruido del sensor y la frecuencia de muestreo, así como la opción de establecer pesos para los diferentes sensores. De esta forma, el algoritmo puede decidir qué lecturas son más confiables y hacer una estimación mejorada del valor real.

Los sistemas de control de vehículos submarinos convencionales generalmente implementan un estimador de estado encargado de procesar los datos recogidos por los diferentes sensores.

Por lo tanto, la exactitud en la estimación de la actitud del ROV dependerá tanto de la calidad de los sensores involucrados, como del algoritmo utilizado para fusionar los datos de estos sensores. El algoritmo que utiliza la PixHawk es un Filtro de Kalman Extendido.

3.1.2. Filtro de Kalman y Filtro de Kalman Extendido

Dado que la mayoría de los sensores son sensibles a múltiples parámetros físicos, es teóricamente posible mejorar la exactitud de la medida mediante la fusión de los datos de sensores complementarios. Kalman presentó en 1960 un método para realizar esto, conocido como el filtro de Kalman. Este filtro ha sido ampliamente utilizado sistemas de navegación, ya que puede reconstruir estados no medidos, así como eliminar el ruido de las estimaciones [1].

El filtro de Kalman es de gran utilidad para sistemas lineales de tiempo discreto con ruido gaussiano, permitiendo estimar el estado interno de un sistema en base a las medidas y al modelo de estado. Es un filtro recursivo o realimentado, ya que utiliza la salida de su estimación anterior para procesar la siguiente [14].

Se basa en la teoría de la probabilidad condicionada (teorema de Bayes), por lo que comparte muchas características con los filtros Bayesianos, siendo su objetivo obtener la estimación con el mínimo error cuadrático medio (RMS), así como la covarianza asociada, es decir, la incertidumbre. Se divide en tres etapas: predicción, comparación y corrección.

- Primero, el filtro hace una suposición sobre el próximo valor de salida del sensor basada en el modelo de estado.
- Luego toma el valor medido y lo compara con la estimación.
- Finalmente, actualiza el modelo para hacer una estimación más exacta en la próxima medición.

Durante las últimas décadas, los filtros de Kalman y sus extensiones para sistemas no lineales, el filtro de Kalman Extendido (EKF) y el filtro de Kalman Unscented (UKF), se han utilizado ampliamente en diversas aplicaciones, como satélites, naves espaciales o aviones, para asistir al control automático de estos sistemas. Sin embargo, debido a su complejidad computacional, no se habían utilizado ampliamente en sistemas embebidos, hasta que aumentó la potencia de procesamiento de los microcontroladores.

El EKF basa su funcionamiento en dos etapas. En primer lugar, linealiza el sistema originalmente no lineal en torno al estado estimado actual. A continuación, aplica el filtro de Kalman sobre el estado linealizado. Este algoritmo requiere más capacidad computacional que el KF, aunque los resultados obtenidos mejoran significativamente.

3.1.3. Estimación del ángulo de heading

Los sensores que intervienen en la medida del ángulo de heading del ROV son el acelerómetro, giroscopio y magnetómetro, integrados en la IMU, así como el sistema UGPS.

En aplicaciones de bajo coste en las que es de gran importancia reducir el peso del vehículo, los sensores MEMS para giroscopio, acelerómetro y magnetómetro son los más habituales. En nuestro caso, el sistema UGPS (Water Linked) no estaba disponible, por lo que el algoritmo de control implementado en la PixHawk solo cuenta con el acelerómetro, giroscopio y magnetómetro.

El problema principal de estos sensores es que se alteran fácilmente, además de su baja exactitud. Los acelerómetros se ven perturbados por las aceleraciones inerciales y las vibraciones, y los magnetómetros solo funcionan correctamente en entornos libres de perturbaciones magnéticas.

Al implementar el filtro de Kalman Extendido, se reducen los errores inherentes a cada sensor, por lo que aumenta la exactitud de la medida final. Además, este algoritmo proporciona medidas en tiempo real, sin sobrecargar el controlador de vuelo.

El funcionamiento básico del algoritmo se ilustra en la Figura 23. Puesto que varios sensores pueden proporcionar medidas relativas a la posición angular, el filtro de Kalman Extendido (EKF) se encarga de evaluar los datos transmitidos por cada sensor y proporcionar un único resultado. En concreto, el EFK fusiona los datos procedentes del acelerómetro y magnetómetro, por un lado, y del giroscopio, por otro lado.

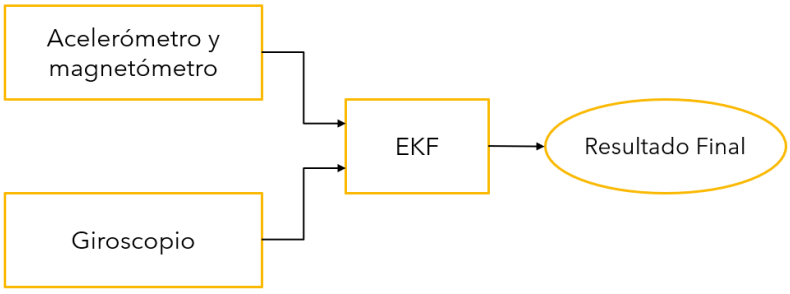


Fig. 23. Funcionamiento del EKF

El acelerómetro permite obtener la aceleración lineal con respecto a los ejes X, Y, Z. Para ello, se determina la desviación con respecto al eje Z, que se relaciona con la aceleración debida a la gravedad. Integrando dos veces, se obtiene la posición lineal.

Por otro lado, el magnetómetro de la IMU mide la intensidad del campo magnético de la Tierra, y así determina la posición relativa del norte magnético de la Tierra.

Finalmente, el giroscopio mide la velocidad angular, por lo que para determinar la posición angular, es necesario integrar los datos obtenidos de este sensor. La desventaja de este método es que, tras sucesivas integraciones, se van acumulando pequeños errores (resultantes del proceso de integración), lo que genera una desviación en la medida con respecto al valor real conforme pasa el tiempo.

Los acelerómetros detectan cambios en la dirección con respecto a la gravedad, y esos datos pueden usarse para orientar un giroscopio. De esta forma, el filtro de Kalman Extendido inicializa la orientación con los datos del acelerómetro y magnetómetro. Posteriormente, usa los datos de estos dos sensores para corregir las medidas del giroscopio [15].

Para hacer eso, tenemos para elegir cuánto confiamos en las medidas del acelerómetro y el magnetómetro en comparación con las medidas del giroscopio. En un filtro de Kalman, esto se calcula automáticamente después de evaluar una serie de parámetros, como los que definen el ruido de cada sensor, etc. Además, el filtro de Kalman puede integrar los datos procedentes de un sistema de posicionamiento por satélite, si estuviera disponible.

3.1.4. Configuración del EKF

Versiones del EKF

En ArduSub, existen dos versiones del filtro de Kalman Extendido que podemos implementar sobre el ROV: EKF2 y EKF3. Esta última es una versión más reciente del algoritmo, que utiliza esencialmente las mismas interfaces, buffers de datos y lógica de control que EKF2, y además incluye ciertas mejoras con respecto a la versión anterior. Algunas de sus diferencias se muestran a continuación [16],[17]:

- EKF3 estima el sesgo del acelerómetro para los 3 ejes, mientras que EKF2 solo lo hace en el eje vertical (eje Z).
- EKF3 admite transiciones de GPS a no GPS.
- EKF2 estima los factores de escala del giroscopio mientras que EKF3 no lo hace. Sin embargo, los factores de escala de los giroscopios actuales son casi siempre 1.0, por lo que esto no es muy relevante.
- En la versión EKF3 los tres errores de orientación se reemplazan con una representación mediante cuaterniones, lo que genera 4 estados de error.

En definitiva, la versión EKF3 proporciona en teoría una medida de la inclinación del ROV más exacta para la mayoría de los casos de uso.

Parámetros en ArduSub

Podemos configurar el filtro de Kalman Extendido implementado en la PixHawk a través de ArduSub, mediante la modificación de más de 50 parámetros [18], los más relevantes se describen a continuación. Para modificar un parámetro, podemos hacerlo desde QGroundControl o mediante MAVLink, con el comando PARAM_SET.

En primer lugar, para seleccionar qué versión utilizar debemos cambiar el parámetro *AHRS_EKF_TYPE*.

AHRS_EKF_TYPE	
Valor	Significado
0	Disabled
2	Enable EKF2
3	Enable EKF3

- Para habilitar EKF2, debemos cambiar también el parámetro *EK2_ENABLE* a '1':

EK2_ENABLE	
Valor	Significado
0	Disabled
1	Enabled

  Habilitar EKF2 solo hace que se ejecute el algoritmo, no que se use para el control de vuelo. Para usarlo en el control, hay que establecer *AHRS_EKF_TYPE* = '2'. Es necesario reiniciar QGroundControl después de cambiar el valor de EK2_ENABLE para que surta efecto.

- Para habilitar EKF3, debemos cambiar también el parámetro *EK3_ENABLE* a '1':

EK3_ENABLE	
Valor	Significado
0	Disabled
1	Enabled

  Habilitar EKF3 solo hace que se ejecute el algoritmo, no que se use para el control de vuelo. Para usarlo en el control, hay que establecer *AHRS_EKF_TYPE* = '3'. Es necesario reiniciar QGroundControl después de cambiar el valor de EK3_ENABLE para que surta efecto.

Una vez habilitada la versión del algoritmo más conveniente, podemos cambiar multitud de parámetros según las condiciones de vuelo [18]. Algunos de ellos son los siguientes:

- *AHRS_YAW_P*: controla el peso que tiene el magnetómetro o el GPS en la medida del ángulo de heading. Un valor más alto significa que la medida se actualizará de acuerdo con el magnetómetro o el GPS más rápidamente. Varía de 0.1 a 0.4.

- *GCS_PID_MASK*: bit de máscara que habilita el control PID para las siguientes medidas:

    - 0: deshabilitado
    - 1: ángulo de balanceo (roll)
    - 2: ángulo de inclinación (pitch)
    - 4: ángulo de guiñada (yaw)

- *AHRS_GPS_USE*: habilita el uso del GPS durante la navegación.

- *AHRS_GPS_GAIN*: controla cuánto influencia el GPS en la medida del ángulo de heading. Varía entre 0.0 y 0.1.

- *EK2_ALT_M_NSE/ EK3_ALT_ M_NSE*: permite introducir el valor cuadrático medio del ruido en la medición de la altitud.

- *EK2_MAG_M_NSE/ EK3_MAG_ M_NSE*: permite introducir el valor cuadrático medio del ruido de la medición del magnetómetro.

- *SURFACE_DEPTH*: valor de la profundidad que el sensor de presión leerá cuando el vehículo se encuentre en la superficie.

- *RC_SPEED*: frecuencia en Hz a la que los controladores electrónicos de velocidad (ESC) se actualizan. Varía de 50 a 490 Hz.

- *AHRS_RP_P*: controla la frecuencia a la que el acelerómetro corrige el ángulo de heading. Varía entre 0.1 a 0.4.

- *AHRS_ORIENTATION*: indica la orientación de la PixHawk en relación con la orientación estándar. Gira las lecturas de la IMU para permitir que la placa se oriente en el vehículo en cualquier ángulo de 90° o 45°. En el Sibiu PRO, la orientación de la PixHawk hace que este parámetro sea *Roll90*.

- *ARMING_CHECK*: bit de mascara que habilita una serie de comprobaciones que tendrán lugar antes de que se permita armar el vehículo. Se pueden configurar

los tipos de comprobaciones a realizar, por ejemplo, comprobaciones del nivel de batería o de la disponibilidad de las medidas de sensores.

- *SERIAL0_PROTOCOL*: permite seleccionar el protocolo de comunicación implementado en el puerto USB. Por ejemplo, MAVLink1, MAVLink2, etc. Existen varios parámetros similares a este para cada puerto: *SERIAL1_PROTOCOL*, *SERIAL2_PROTOCOL*, etc.

- *SERIAL0_BAUD*: modifica la velocidad de comunicación del puerto USB, entre 1200 y 1500000 baudios. Existen varios parámetros similares a este para cada puerto: *SERIAL1_BAUD*, *SERIAL2_BAUD*, etc.

- *FS_GCS_ENABLE*: permite seleccionar qué procedimiento seguir si se pierde el heartbeat de la estación de tierra.

- *FS_TEMP_ENABLE*: controla qué procedimiento seguir si la temperatura interna supera el valor definido en este parámetro.

- *FS_EKF_ACTION*: controla qué procedimiento seguir si se produce un fallo en el algoritmo del EKF.

- *JS_GAIN_MAX*: permite modificar la ganancia máxima del joystick, entre 0.2 y 1.0.

- *JS_LIGHTS_STEPS*: modifica el número de pasos comprendidos entre los valores máximo y mínimo de la intensidad de las luces, entre 1 y 10.

3.2. Estimación mediante el sonar Ping360

Como se explica en el capítulo 5, las pruebas realizadas sobre la estimación del ángulo de heading mediante los sensores inerciales muestran que la medida proporcionada no alcanza el nivel de exactitud deseado. Esto pudo ser debido a las perturbaciones magnéticas del entorno en el que se realizaron las pruebas, ya que la piscina se sitúa muy próxima a una pared de hormigón armado. En cualquier caso, es preciso buscar otro método que nos permita obtener el ángulo de heading con mayor fiabilidad.

En este apartado, se describe la solución empleada para estimar el ángulo del vehículo con respecto a un objeto situado dentro de la instalación acuática y con respecto a la pared de la misma, usando la información proporcionada por el sonar Ping360.

3.2.1. Ángulo con respecto a un objeto

Muestreando el entorno completo con el sonar Ping360 a 3 metros de distancia, obtenemos una imagen como la mostrada en la Figura 24:

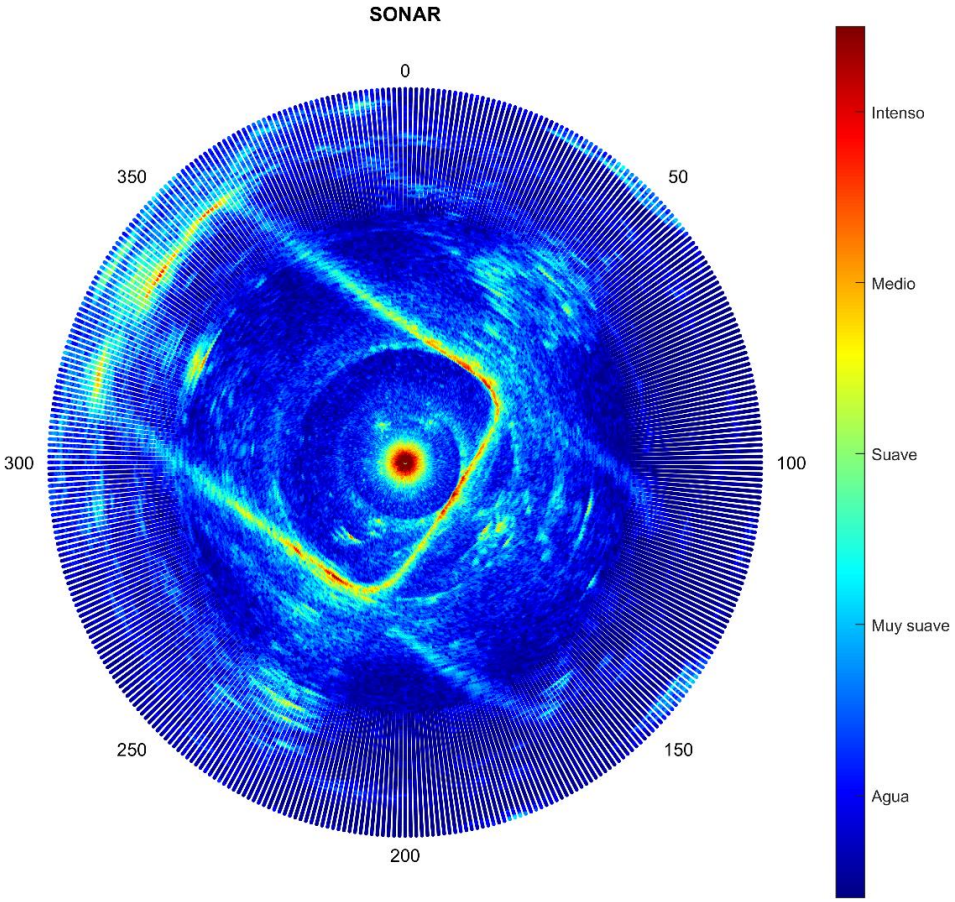


Fig. 24. Imagen obtenida mediante el sonar (I)

En esta imagen acústica, podemos distinguir el propio ROV, las paredes de la piscina y un objeto situado dentro de la misma, tal como se muestra en la Figura 25.

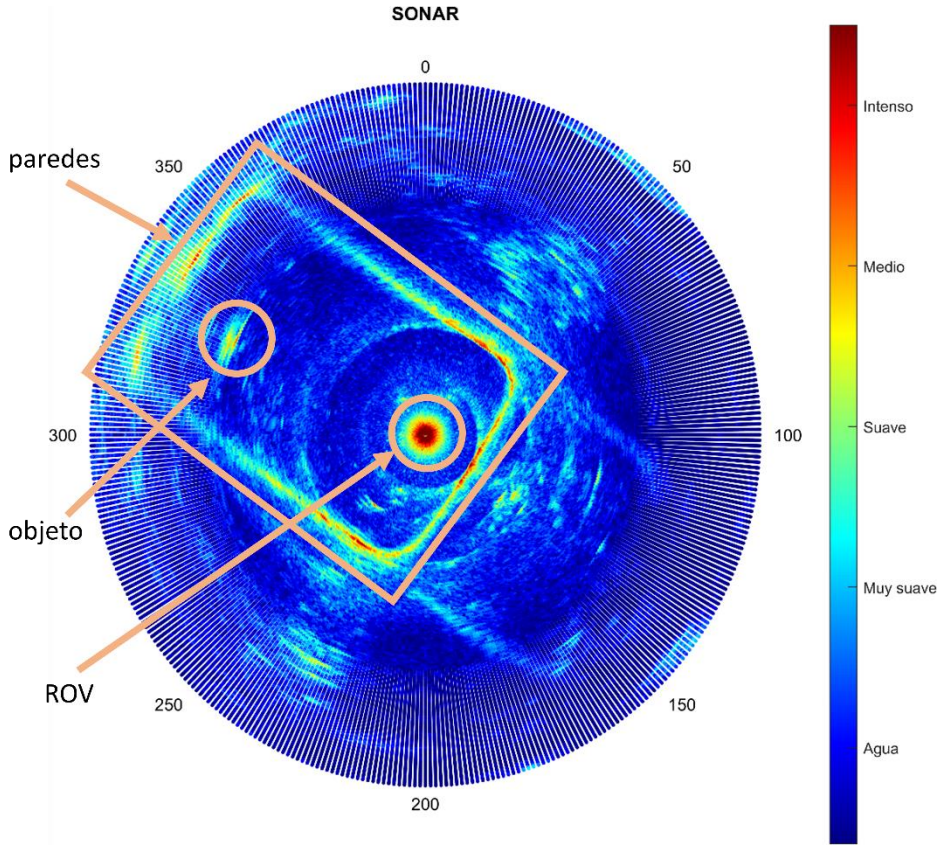


Fig. 25. Imagen obtenida mediante el sonar (II)

Para situar el objeto con respecto al vehículo, es necesario definir en primer lugar el sistema de referencia local del ROV. Este sistema está formado por dos ejes perpendiculares entre sí: eje *x* y eje *y*. El eje *x* coincide con la dirección de avance del vehículo, mientras que el eje *y* coincide con la dirección de desplazamiento lateral, tal como muestran las Figuras 26 y 27.

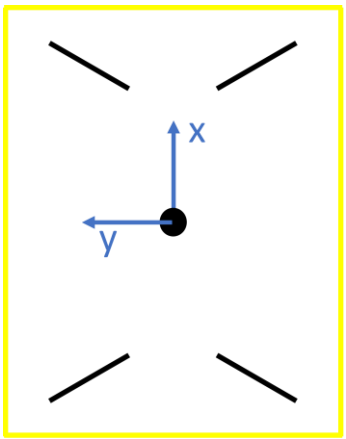


Fig. 26. Sistema de referencia local del ROV (I)

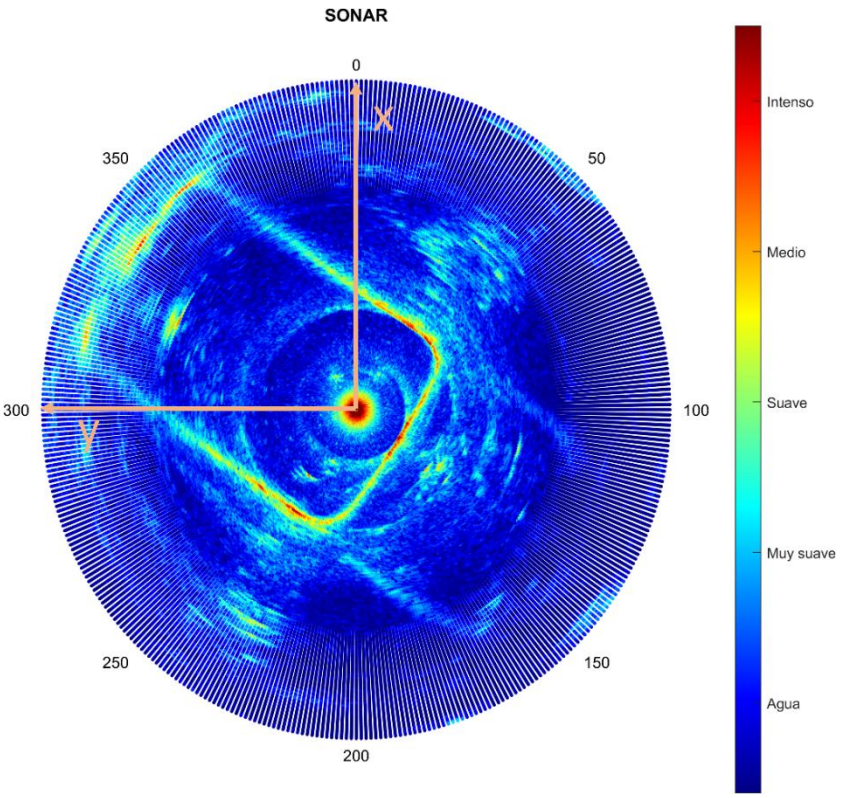


Fig. 27. Sistema de referencia local del ROV (II)

Para situar el centroide de este objeto, es necesario llevar a cabo un procesamiento de los datos obtenidos mediante el Ping360. En concreto, se agrupan los pixeles cuya intensidad del eco devuelto es similar entre sí, y mayor a un valor de referencia determinado utilizando técnicas de 'clustering'. Este proceso de identificación se describe con más detalle en el capítulo 4.

De esta forma, obtenemos las coordenadas del centroide del objeto en el sistema de referencia local del ROV (x_1, y_1), tal como ilustra la Figura 28.

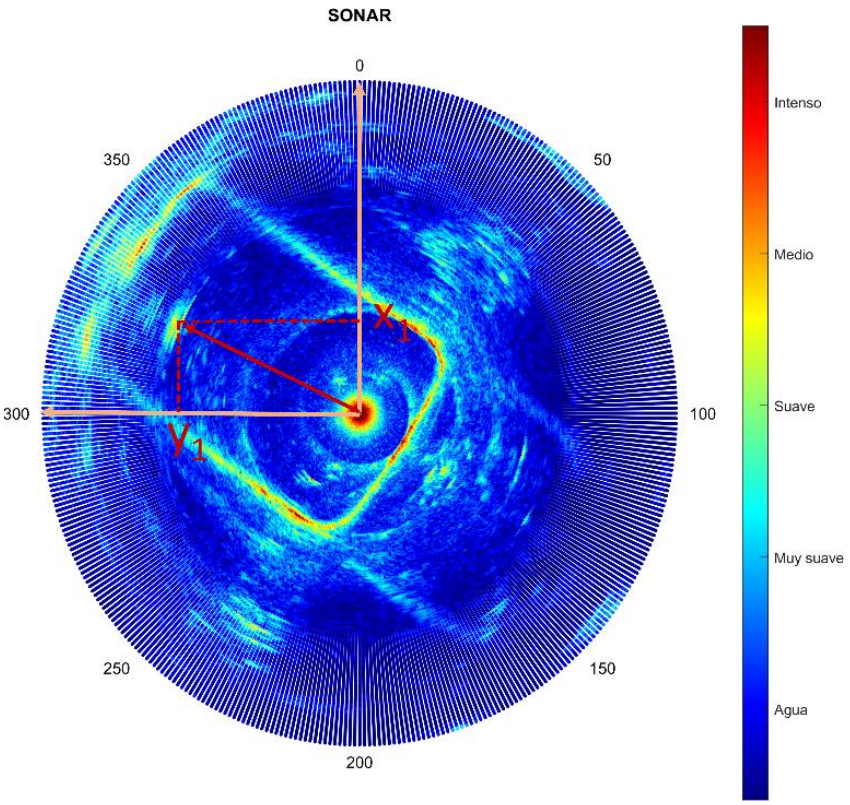


Fig. 28. Coordenadas del objeto

Una vez que hemos calculado las coordenadas del centroide del objeto con respecto al ROV, podemos determinar el ángulo entre ellos, α, tal como muestra la Figura 29:

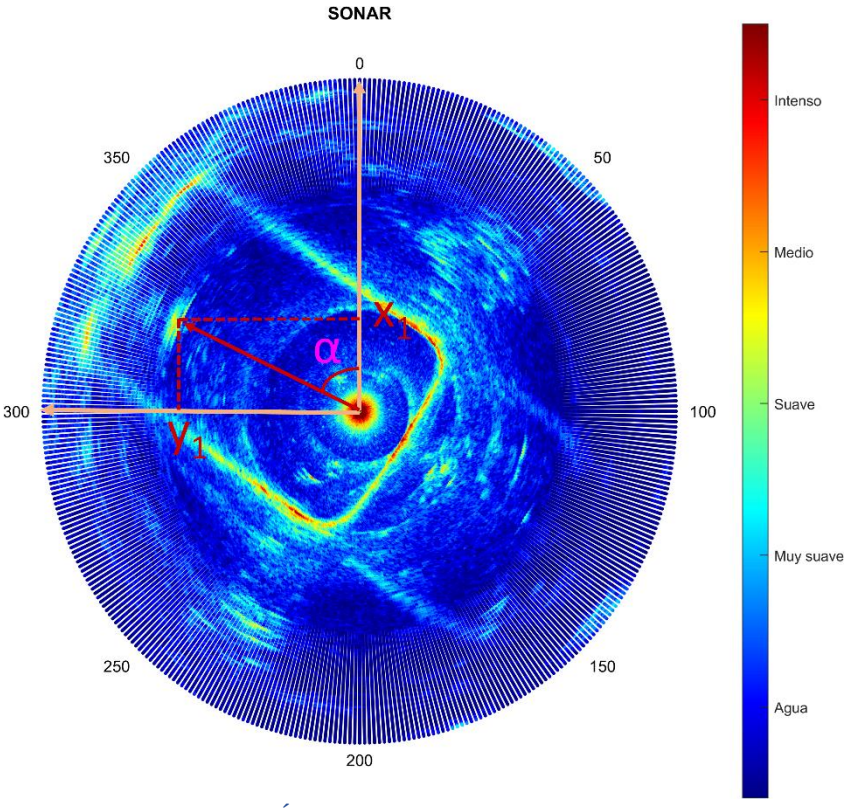


Fig. 29. Ángulo entre el objeto y el ROV (α)

Analíticamente:

$$\alpha = \arctan\frac{y_1}{x_1} \tag{2}$$

3.2.2. Ángulo con respecto a la pared

En este caso, acotando el sector de muestreo a 20°-30°, obtendremos una imagen mediante el sonar Ping360 similar a la mostrada en la Figura 30.

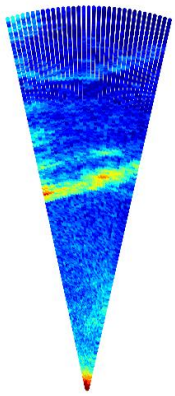


Fig. 30. Imagen obtenida con el sonar (I)

A partir de esta imagen acústica, calculamos la distancia y el ángulo con respecto a la pared. Para ello, identificamos tres puntos en la pared: el punto central (en color magenta) y los puntos en los extremos de la imagen (en color blanco), tal como ilustra la Figura 31.

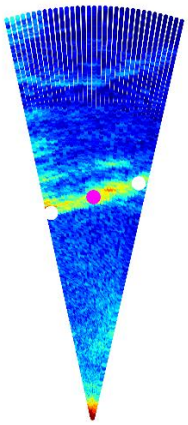


Fig. 31. Identificación de la pared (I)

Como se explicó en el capítulo 2, cada haz del sonar recoge 1200 valores de intensidad. Entre estos valores, seleccionamos aquel cuya intensidad sea mayor, y repetimos este proceso para un número determinado de haces. Por ejemplo, para situar el punto de la izquierda, se toman los cinco primeros haces del sector muestreado (ver Figura 32).

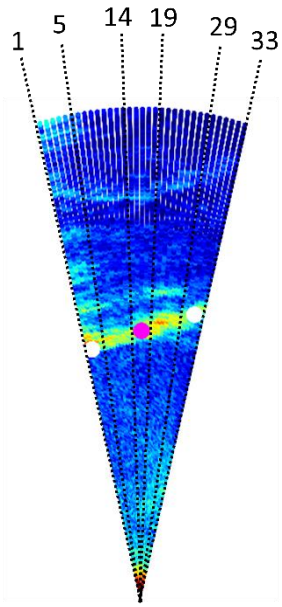


Fig. 32. Haces tomados para identificar cada punto

Las coordenadas del punto en cuestión se calculan a partir de las coordenadas del punto de mayor intensidad de cada haz considerado, según las siguientes expresiones:

$$x = \frac{\sum_{i=lím_{inferior}}^{lím_{superior}} x_i}{lím_{superior} - lím_{inferior}} \quad (3)$$

$$y = \frac{\sum_{i=lím_{inferior}}^{lím_{superior}} y_i}{lím_{superior} - lím_{inferior}} \quad (4)$$

A partir de los tres puntos anteriores, obtenemos la geometría de un triángulo (ver Figura 33):

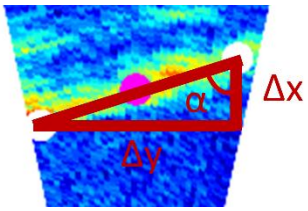


Fig. 33. Triángulo obtenido

Por semejanza de triángulos el ángulo α es el igual al ángulo que forman la pared y el ROV, tal como ilustra la Figura 34.

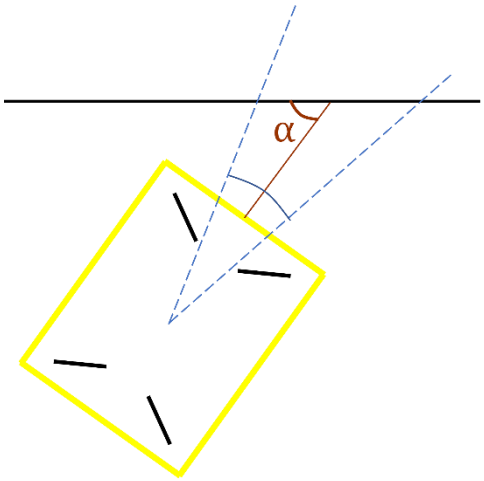


Fig. 34. Ángulo entre el ROV y la pared

El objetivo es que el ROV se sitúe a 90° con respecto a la pared (α = 90°), tal como muestra la Figura 35:

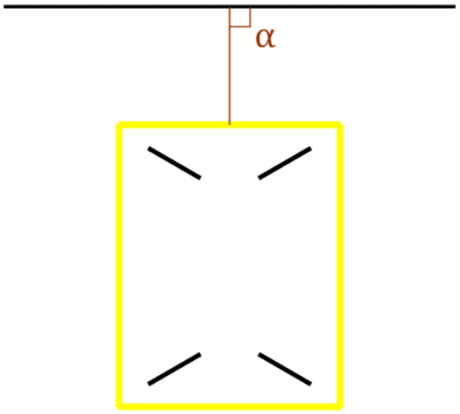


Fig. 35. Ángulo de 90° entre el ROV y la pared

A partir del triángulo obtenido en la Figura 33, podemos determinar la expresión analítica de α:

$$\alpha = \arctan\frac{\Delta y}{\Delta x} \tag{5}$$

donde:

- Δx es la diferencia entre los dos puntos medida en el eje x.
- Δy es la diferencia entre los dos puntos medida en el eje y.

Como ejemplo, a continuación, se ilustra el cálculo para la imagen de la Figura 30. Los valores para las coordenadas de los puntos marcados en la Figura 36 son:

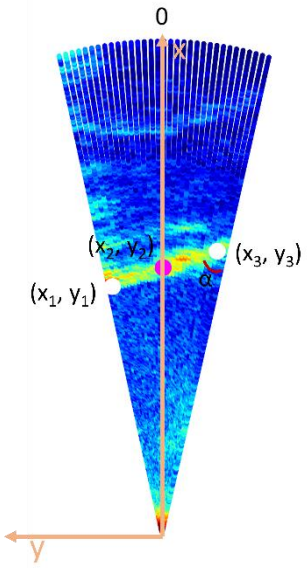


Fig. 36. Coordenadas de cada punto

$(x_1, y_1) = (1502.0, 327.4)$ [mm]
$(x_2, y_2) = (1594.3, 19.23)$ [mm]
$(x_3, y_3) = (1714.7, -379.8)$ [mm]

Por lo tanto:

$$\Delta x = x_3 - x_1 \tag{6}$$

$$\Delta x = 1714.7 - 1502.0 = 212.7 \tag{7}$$

$$\Delta y = y_1 + |y_3| \tag{8}$$

$$\Delta y = 327.4 + |-379.8| = 707.2 \tag{9}$$

$$\alpha = \arctan\frac{\Delta y}{\Delta x} = \arctan\frac{707.2}{212.7} = 73.26° \tag{10}$$

$$\alpha = 73.26°$$

En general, en las instalaciones acuáticas donde operan los ROVs están presentes otros elementos que el sonar detecta, tales como peces u objetos de distinta naturaleza. En la Figura 37 se muestra la imagen tomada por el Ping360 en una instalación piscícola de forma circular:

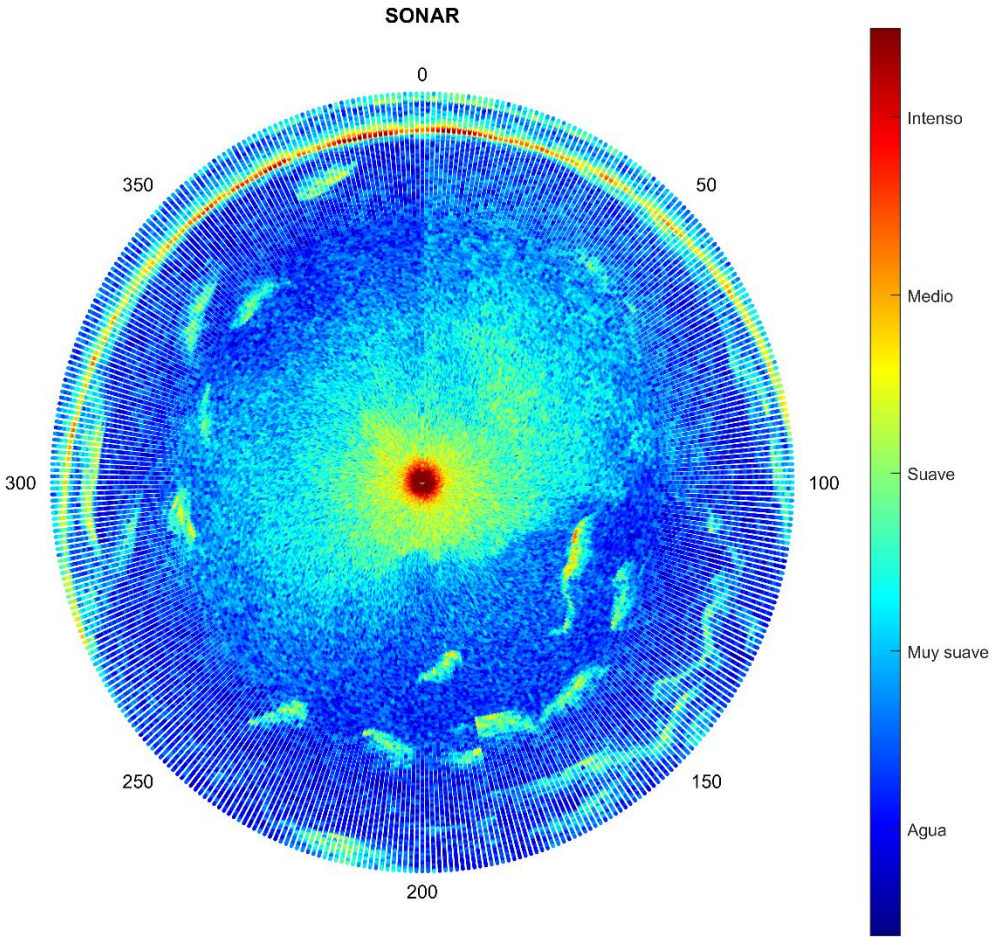


Fig. 37. Imagen obtenida con el sonar (II)

Como vemos, además de la pared de la instalación, podemos identificar a simple vista numerosos peces. En este caso, es necesario filtrar de manera más estricta los datos obtenidos mediante el sonar, atendiendo a la intensidad del rebote. De esta forma, conseguimos eliminar los elementos indeseados, y así detectar la pared correctamente, como se muestra en la Figura 38.

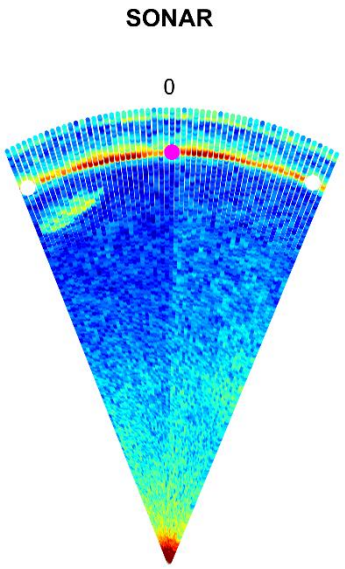


Fig. 38. Identificación de la pared (II)

Además, cabe destacar que, en este caso, la instalación acuática tiene forma circular. Puesto que el vehículo está situado en el centro de la piscina, el algoritmo interpreta que el vehículo está situado perpendicularmente con respecto a la pared, ya que los puntos identificados en los extremos son prácticamente simétricos con respecto el eje x del sistema de referencia local del ROV.

Capítulo 4

Algoritmo de control de navegación

Existen numerosos métodos de control aplicados a vehículos operados remotamente basados en modelos matemáticos, obtenidos a partir de las propiedades hidrodinámicas de dichos vehículos. Sin embargo, generalmente es difícil obtener un modelo matemático preciso que tenga en cuenta las influencias del medio sobre la dinámica del ROV. De hecho, los métodos de control basados en modelos con frecuencia son reemplazados por algoritmos de control sin modelo, como el que se desarrolla en este TFG.

Diseñar un algoritmo de control de navegación para un ROV implica una serie de desafíos, debido principalmente a las perturbaciones impredecibles del entorno y la frecuencia de actualización de los sensores que aportan la información necesaria para el control, especialmente el sonar Ping360.

En este trabajo se presentan dos objetivos de navegación, descritos posteriormente en este capítulo:

- Navegación realizando transectos perpendiculares a las paredes frontales de la piscina.
- Reorientación y acercamiento a un objeto seleccionado.

Para llevar a cabo una correcta navegación nos basaremos fundamentalmente en la estimación de la orientación mediante el sonar Ping360, puesto que, en base a nuestra experiencia y tal como se recoge en el capítulo 5, las medidas proporcionadas por la IMU no son estables en muchos entornos. Únicamente utilizaremos la IMU para realizar maniobras de reorientación en las que la medida del ángulo de heading del vehículo no necesita ser muy precisa.

4.1. Navegación realizando transectos

Entre las diferentes aplicaciones que un ROV puede desarrollar en una instalación acuática, cabe destacar la navegación autónoma dentro de la instalación mientras toma datos del entorno. De esta forma, el operador puede centrar su atención en otras tareas, tales como analizar la información recogida por el vehículo en tiempo real, etc.

En este caso, el vehículo recorre la piscina de un extremo a otro, manteniendo la perpendicularidad con la pared a la que se acerca. Una vez que se sitúa a cierta distancia de la pared, gira 180° y comienza a navegar hacia la pared opuesta, tal como ilustra la Figura 39.

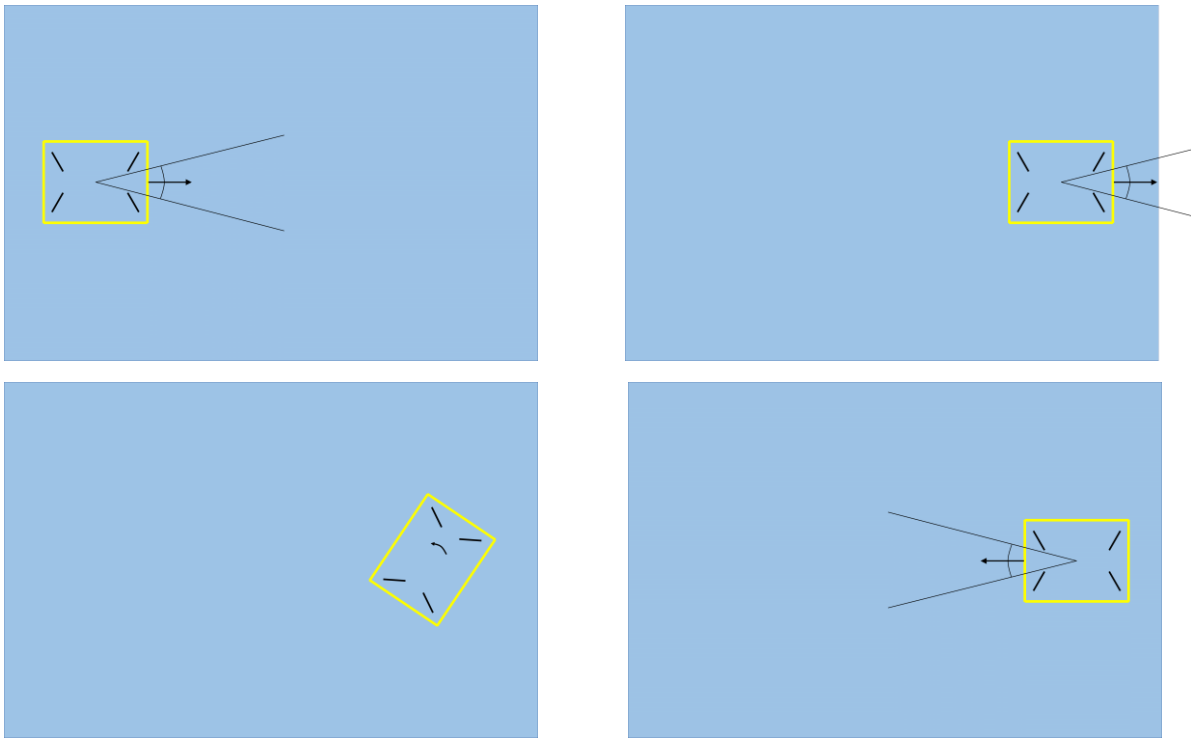


Fig. 39. Navegación realizando transectos

Para conseguir esto, es necesario acotar el ángulo de muestreo del sonar Ping360 a 20°-30°, aumentando así la frecuencia de actualización de los datos más relevantes para la navegación. Estos datos, que nos permiten determinar el rumbo hacia la pared, son la distancia y el ángulo relativo entre esta y el ROV.

Una vez determinado el rumbo, el controlador de vuelo se encarga de modular las velocidades de avance y de giro para acercar la nave a la pared. Por lo tanto, para regular estas dos velocidades, se han implementado dos controladores independientes que corrigen constantemente el rumbo:

- El controlador de la velocidad de avance, que depende de la distancia a la pared.
- El controlador de la velocidad de giro, que depende del ángulo entre la pared y el ROV.

Estos controladores son de tipo proporcional (P). El controlador proporcional es uno de los más utilizados en la industria. Su principal ventaja es la simplicidad de implementación, ya que, aunque el modelo del sistema sea completamente desconocido, mediante un ajuste manual es posible encontrar la ganancia k_p que obtenga un rendimiento adecuado del controlador. En concreto, la ganancia proporcional disminuye el tiempo de subida, aunque aumenta el sobreimpulso y degrada la estabilidad del sistema.

En la navegación del vehículo intervienen diversos factores externos difícilmente medibles que debemos tratar de corregir, tales como la tensión del cable umbilical, la deriva del propio vehículo, etc. Es por esto que resulta conveniente dividir el proceso de navegación en diferentes etapas, estudiar cada una de ellas por separado y, posteriormente, resolver el problema completo.

A continuación, se describen cada una de estas etapas. Los resultados de las pruebas realizadas para comprobar el correcto funcionamiento de las mismas se encuentran en el capítulo 5.

4.1.1. Controlador de navegación a la pared sin controlar el ángulo de giro basado en el Ping360

El objetivo de esta etapa es comprobar que el algoritmo calcula correctamente la distancia a la pared de la piscina. Para ello, controlamos únicamente la velocidad de avance del ROV, que es proporcional a la distancia. Cuando el ROV se encuentra lo suficientemente cerca de la pared, se detiene.

El funcionamiento de este algoritmo se muestra en el siguiente diagrama de flujo (ver Figura 40).

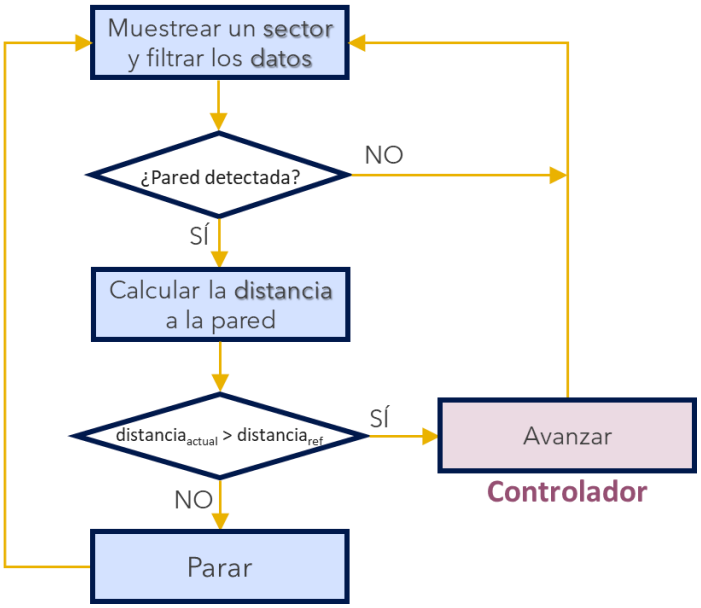


Fig. 40. Diagrama de flujo (I)

Una vez determinada la distancia a la pared, el controlador proporcional se encarga de calcular la velocidad de avance, que manda al controlador de vuelo PixHawk.

Tal como se recoge en el capítulo 5, durante esta prueba, aunque el vehículo debería avanzar en línea recta, observamos un movimiento de heading no deseado. Es decir, el ángulo de heading no permanece constante, ya que el vehículo se desvía hacia un lado.

Este comportamiento era de esperar, ya que se trata de una navegación en bucle abierto respecto a la orientación del vehículo y, como resultado de cualquier perturbación por mínima que sea, es lógico que el vehículo se desvíe. Esta deriva indeseada tuvo que ser corregida posteriormente en el algoritmo de control de orientación.

4.1.2. Controlador de perpendicularidad con respecto a la pared basado en el Ping360

En esta etapa, tratamos de comprobar que el algoritmo calcula correctamente el ángulo con respecto a la pared de la piscina. Para ello, controlamos únicamente la velocidad de giro, que es proporcional al ángulo. Para que cambie el ángulo, empujamos al ROV suavemente. Cuando se encuentra a 90° de la pared, se detiene.

El funcionamiento de este algoritmo se muestra en el siguiente diagrama de flujo (ver Figura 41).

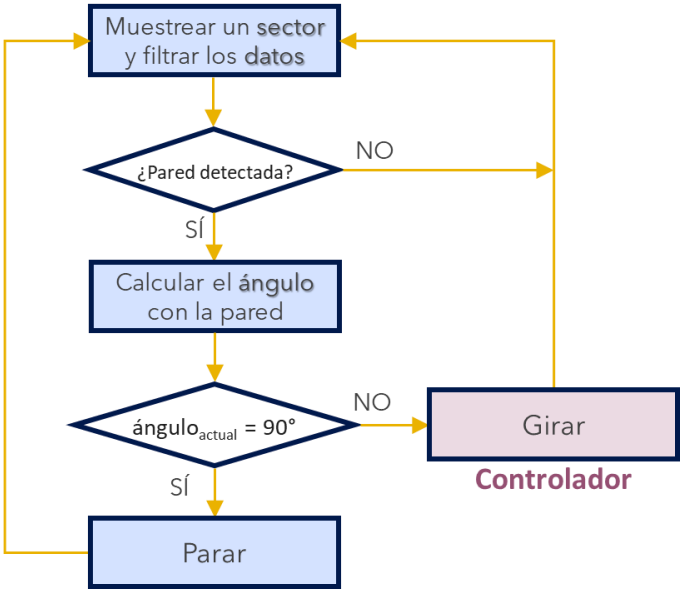


Fig. 41. Diagrama de flujo (II)

Una vez determinado el ángulo entre la pared y el ROV, el controlador proporcional se encarga de calcular la velocidad de giro, que manda al controlador de vuelo PixHawk.

4.1.3. Controlador de navegación perpendicular a la pared y parada a cierta distancia

El objetivo de esta prueba es comprobar que podemos fusionar los dos algoritmos explicados anteriormente. En este caso, el sistema de control debe calcular simultáneamente la distancia a la pared y el ángulo con respecto a la misma, y establecer las velocidades de avance y giro correspondientes.

El funcionamiento de este algoritmo se muestra en el siguiente diagrama de flujo (ver Figura 42).

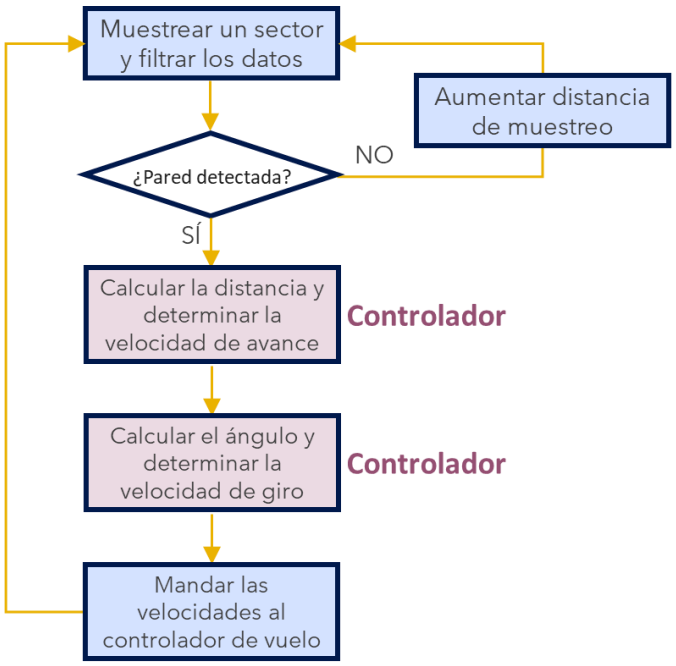


Fig. 42. Diagrama de flujo (III)

Una vez determinados la distancia a la pared y el ángulo entre esta y el ROV, los controladores proporcionales se encargan de calcular las velocidades de avance y giro, que mandan al controlador de vuelo PixHawk.

4.1.4. Control de navegación de un extremo a otro de la piscina

Esta etapa es similar a la anterior, con la diferencia de que una vez que el ROV se ha estabilizado frente a la pared, gira 180° y, a continuación, avanza hacia el otro extremo de la piscina. El funcionamiento de este algoritmo se muestra en el siguiente diagrama de flujo (ver Figura 43).

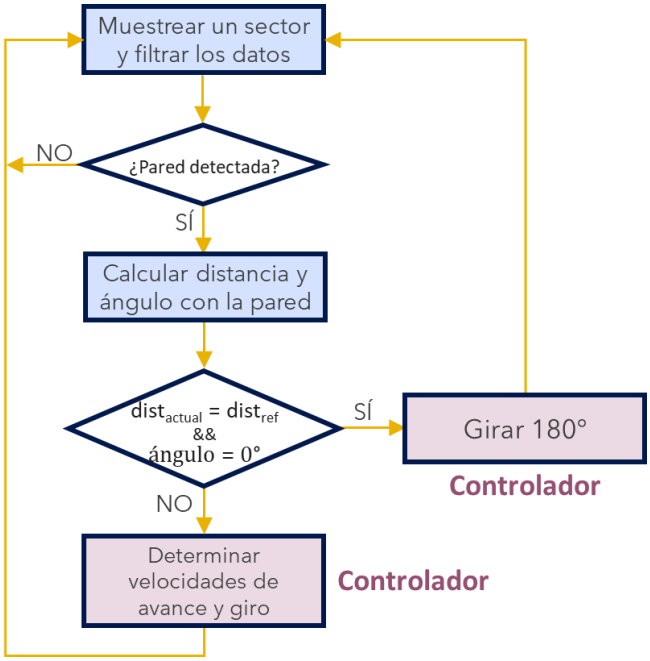


Fig. 43. Diagrama de flujo (IV)

Para realizar el giro de 180°, el controlador se basa en los datos proporcionados por la IMU, tal como se muestra a continuación (ver Figura 44). Puesto que para realizar este giro no es necesario que la medida del ángulo de heading sea muy precisa, podemos utilizar la IMU. Una vez que el vehículo ha girado 180° aproximadamente, vuelve a orientarse adecuadamente con respecto a la pared según la información que aporta el sonar Ping360.

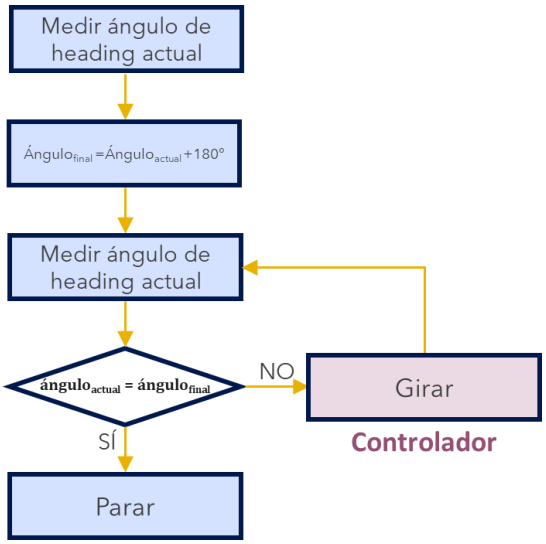


Fig. 44. Diagrama de flujo (V)

Si repetimos este proceso indefinidamente, el vehículo navega realizando transectos entre los extremos de la piscina.

4.2. Navegación hacia un objeto

Otra de las aplicaciones que un ROV puede llevar a cabo consiste en la localización de elementos de interés, tales como peces, cardúmenes u otros objetos de naturaleza variada. Una vez se han localizado dichos elementos, el siguiente paso es dotar al ROV con la capacidad de navegación autónoma hacia el objetivo de interés seleccionado.

En este tipo de navegación, en primer lugar, realizamos un barrido con el sonar de 360°, y así identificamos los objetivos disponibles. Una vez seleccionado el objetivo, el ROV se orienta hasta quedar alineado con el mismo. Finalmente, avanza hasta situarse a cierta distancia del objeto en cuestión. Cuando alcanza la posición deseada, muestrea de nuevo el entorno para identificar nuevos objetivos si los hubiera, tal como ilustra la Figura 45.

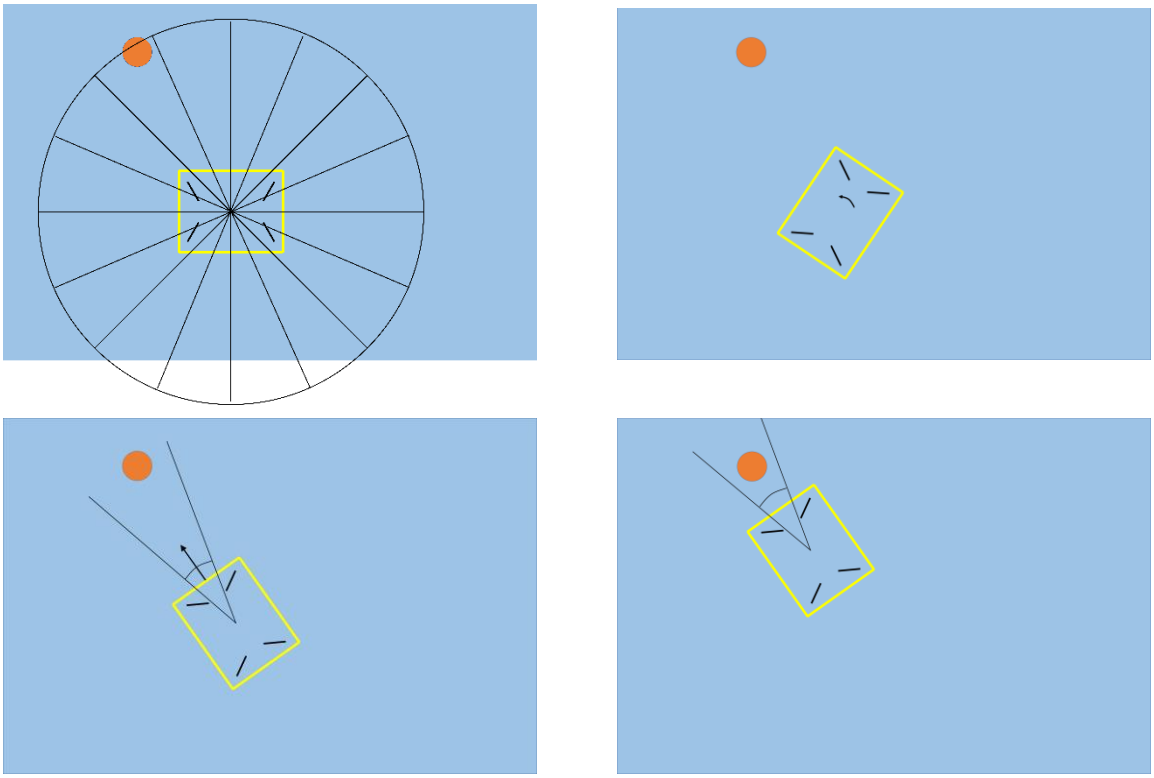


Fig. 45. Navegación hacia objeto

El sonar Ping360 aporta los datos que nos permiten determinar el rumbo hacia el objeto de interés, que son la distancia y el ángulo relativo entre este y el ROV. Ambos están relacionados con la posición del objetivo en el sistema de referencia local del ROV.

En este caso, el algoritmo de control implementado se divide tres etapas diferentes, tal como ilustra la Figura 46.

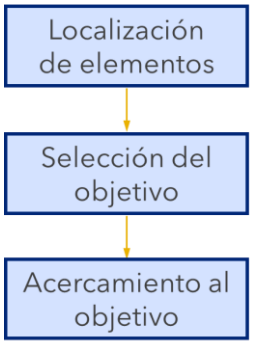


Fig. 46. Diagrama de flujo del algoritmo principal

Las etapas del algoritmo de control se describen a continuación.

4.2.1. Localización de elementos

Este módulo está orientado a dar información al usuario sobre los elementos que se encuentran alrededor del ROV, proporcionando su localización dentro de la instalación explotada y marcando el rumbo necesario para visitar las proximidades del elemento.

Cuando se trata de explorar instalaciones en entornos naturales o al aire libre, es normal que el agua presente un alto grado de turbidez, lo que perjudica la visión submarina e impide, por tanto, que las cámaras, que tradicionalmente portan los ROVs, generen información útil.

Por este motivo, la detección de elementos de interés en estas circunstancias requiere el uso de sensores de proximidad, tales como sonares, que proporcionen al operador información sobre el entorno que rodea al ROV.

Es por esto por lo que la propuesta presentada en este trabajo se basa en la información proporcionada por el sonar Ping360, que aporta información del entorno que rodea al vehículo, posicionando distintos tipos de elementos que se encuentren a su alrededor.

La utilización del sonar Ping360 para la localización de objetivos requiere tener en cuenta varias cuestiones. En primer lugar, es necesario adecuar la velocidad de barrido con el fin de procesar adecuadamente los datos, caracterizando la naturaleza de los objetos reflectantes e identificando los posibles elementos de interés. Por último, es necesario vincular las medidas de distancia y ángulo con la

posición de los elementos en el sistema de referencia donde se haya definido la tarea.

Los datos adquiridos por el sonar Ping360, al incluir datos redundantes y ruido, necesitan ser filtrados para identificar los posibles objetivos a seguir. Por este motivo, en primer lugar, se realiza un proceso de filtrado de los datos, que se compone de una serie de etapas:

- Eliminación de datos inferiores a un límite, despreciando así todos los datos que hacen referencia a rebotes con una intensidad leve, parecida a la que devuelve el agua.

- Eliminación de ruido de proximidad (datos cercanos a la posición del ROV).

A continuación, se lleva a cabo un procesamiento de los datos filtrados, dividido en las siguientes etapas:

- Identificación de elementos a partir de la agrupación de elementos (clustering) tomando como característica la intensidad del eco devuelto.

- Cálculo de los datos que caracterizan la posición de los objetos identificados (distancia y ángulo con respecto al ROV).

Una vez efectuado el procesamiento, y conocida la distancia a cada elemento identificado y el sector donde se encuentra, habremos posicionado dichos elementos en el sistema de referencia local del ROV.

En la piscina de la ETSI, muestreando a una distancia de 3 m, obtenemos una imagen mediante el sonar Ping360 como la mostrada en la Figura 47:

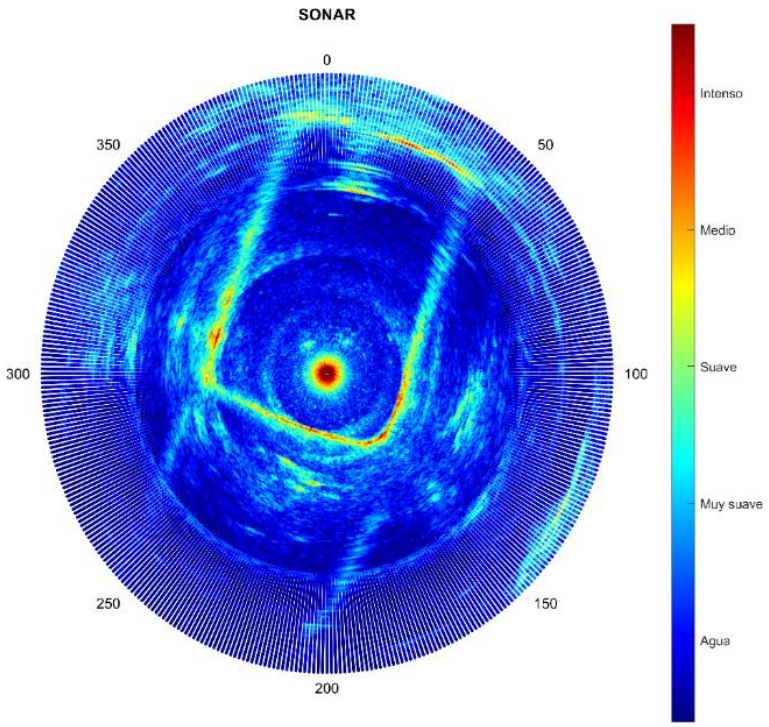


Fig. 47. Imagen obtenida con el sonar

A continuación, se lleva a cabo el procesamiento de datos. Tras aplicar el filtro descrito anteriormente, obtenemos la Figura 48:

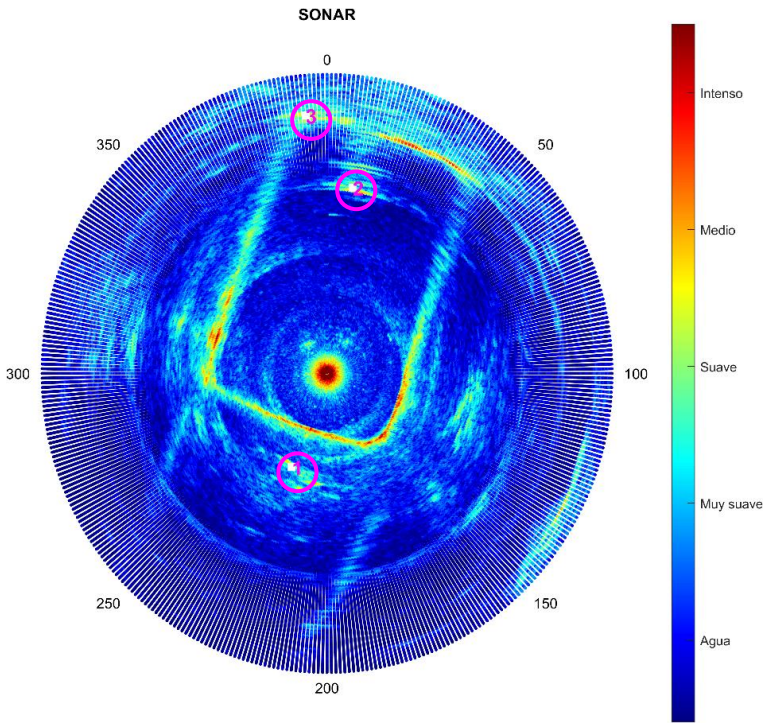


Fig. 48. Identificación de objetos

Como vemos en esa figura, el algoritmo identifica tres objetos, que aparecen numerados en dicha imagen. En este caso, el objeto nº 1 representa ruido exterior a

la piscina, por lo que los únicos objetivos válidos son el nº 2 (objeto introducido en la piscina) y el nº 3 (esquina de la piscina).

Modificando los parámetros del filtro, podemos conseguir que este sea menos estricto y, por lo tanto, que el algoritmo identifique una mayor cantidad de objetos, tal como ilustra la Figura 49.

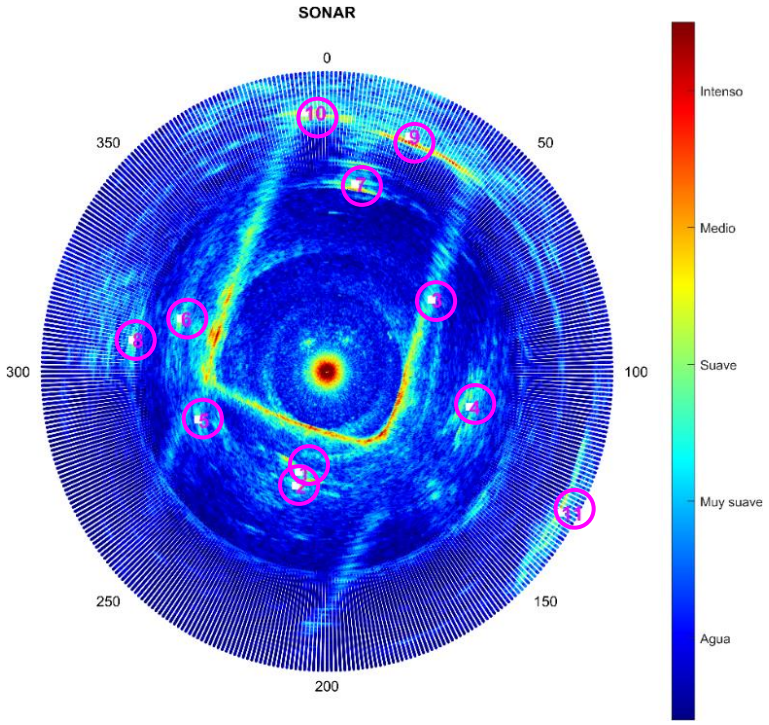


Fig. 49. Identificación de objetos con un filtro menos estricto

Sin embargo, puede ocurrir que varios de los objetos detectados no sean objetivos de navegación válidos. Para nuestra aplicación esto no es relevante, pues es el usuario el que puede discriminar por la posición de los puntos detectados si estos son reales o falsos, y en cada caso decidir a cuál de ellos debe acercarse el ROV.

Además, debemos tener en cuenta que al aumentar la distancia de muestreo, el rebote de intensidad será menor (ver Figura 50, en la que se representa un muestreo a 5 m de distancia). En este caso, será necesario modificar el algoritmo de filtrado para que considere los posibles objetos de menor intensidad.

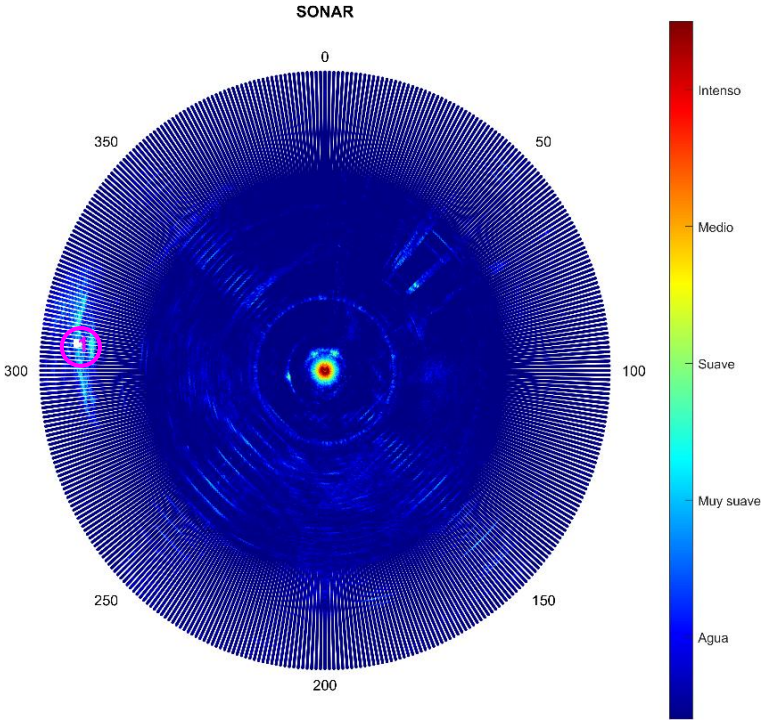


Fig. 50. Muestreo a 5 m

4.2.2. Selección del objetivo de navegación

Como se ha comentado anteriormente, una vez determinadas las posiciones de los distintos elementos de interés, es posible que el operador del ROV esté interesado en visitar las proximidades de uno o varios de ellos.

A través de un interfaz, el operario escoge el elemento identificado que sea de su interés. En ese momento, el elemento seleccionado se convierte en el objetivo de navegación.

Dicho interfaz es la figura donde se representa la imagen acústica obtenida mediante el sonar Ping360, en la que aparece un menú desplegable donde se muestran los números identificativos de todos los objetos detectados, tal como ilustra la Figura 51:

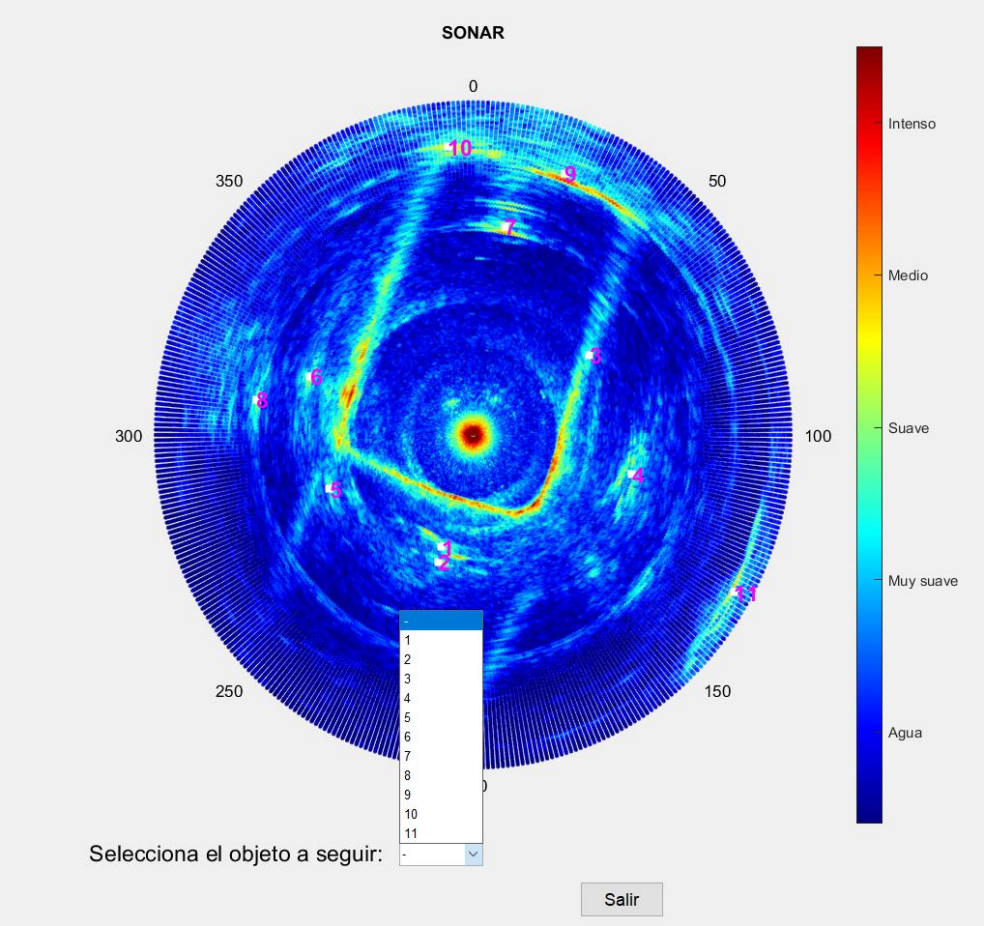


Fig. 51. Interfaz para la selección del objetivo

Este interfaz permite seleccionar, mediante el uso del ratón, de forma cómoda y sencilla el objetivo de navegación, sin necesidad de utilizar un teclado.

4.2.3. Acercamiento hacia el objetivo

Los controladores independientes de velocidad de avance y velocidad de giro que corrigen constantemente el rumbo del vehículo son los mismos que los explicados en el apartado anterior.

Además, cabe destacar que mientras el ROV se acerca al objeto, es necesario acotar el ángulo de muestreo del sonar Ping360 a 20°-30°, aumentando así la frecuencia de actualización de los datos más relevantes para la navegación. Estos datos, que nos permiten determinar el rumbo hacia el objeto, son la distancia y el ángulo relativo entre este y el ROV.

4.2.4. Controlador de navegación hacia un objeto

Para navegar hacia un objeto, en primer lugar, realizamos un barrido con el sonar de 360°, y así identificamos los objetivos disponibles. Una vez seleccionado el objetivo, el ROV se orienta hasta quedar alineado con el mismo. Finalmente, avanza hasta situarse a cierta distancia del objeto en cuestión. Cuando alcanza la posición deseada, muestrea de nuevo el entorno para identificar nuevos objetivos si los hubiera.

El diagrama de flujo de este algoritmo se muestra a continuación (ver Figura 52).

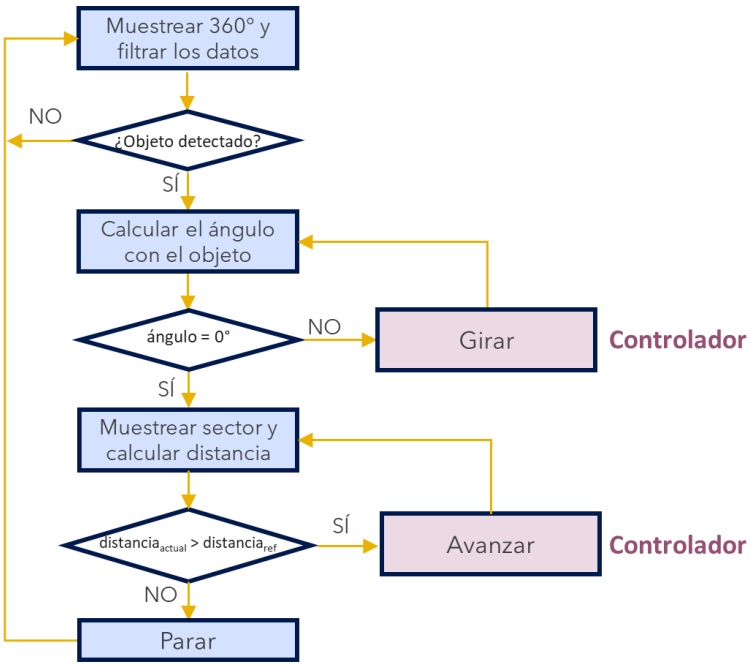


Fig. 52. Diagrama de flujo (VI)

Para realizar el giro del ángulo en cuestión, el controlador se basa en los datos proporcionados por la IMU. Puesto que para realizar este giro no es necesario que la medida del ángulo de heading sea muy precisa, podemos utilizar la IMU. Una vez que el vehículo ha completado el giro, se acerca al objetivo según la información que aporta el sonar Ping360.

Capítulo 5

Desarrollo experimental

En este capítulo se presentan los experimentos llevados a cabo durante la realización del presente trabajo, tanto los relacionados con las medidas de la IMU, como las pruebas de navegación, así como los resultados obtenidos en cada uno de ellos.

La mayor parte de las pruebas descritas a continuación tuvo lugar en la piscina situada en el sótano del edificio de la Escuela Técnica Superior de Ingeniería, en el campus El Carmen, aunque algunas se realizaron en la instalaciones de la explotación acuícola Salinas de Astur (situada en el municipio de Punta Umbría) y en la piscina municipal de Isla Cristina.

Fig. 53. Piscina de la ETSI

La piscina situada en el Carmen mide 3 m de longitud, 2 m de ancho y 1 m de profundidad, aproximadamente, tal como se muestra en la Figura 53.

5.1. Medidas de la IMU

Con el objetivo de comprobar la exactitud de la medida de la orientación proporcionada por la IMU del Sibiu PRO, llevamos a cabo una serie de pruebas experimentales. En la Figura 54 se muestra la fecha aproximada en la que se realizaron los diferentes experimentos.

Pruebas	Marzo				Abril				Mayo			
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
1												
2												
3												
4												
5												
6												
7												

Fig. 54. Calendario de pruebas (I)

Para comprobar la exactitud de la medida, hemos llevado a cabo varias pruebas en las que el vehículo se encuentra en reposo, así como otras en las que está en movimiento.

5.1.1. Coherencia de los datos proporcionados por la IMU en reposo

Experimentación nº 1: datos de la IMU (I)

En esta prueba, situamos el ROV en un punto aleatorio de la piscina, y guardamos el valor del ángulo de heading proporcionado por la IMU en cuatro orientaciones diferentes, separadas entre sí 90°, aproximadamente, tal como se muestra en la Figura 55:

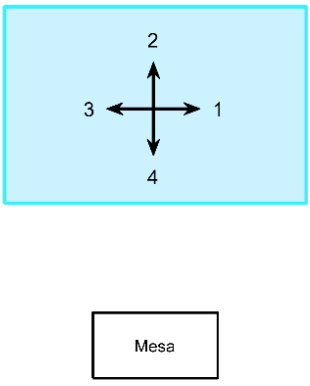


Fig. 55. Orientaciones del ROV

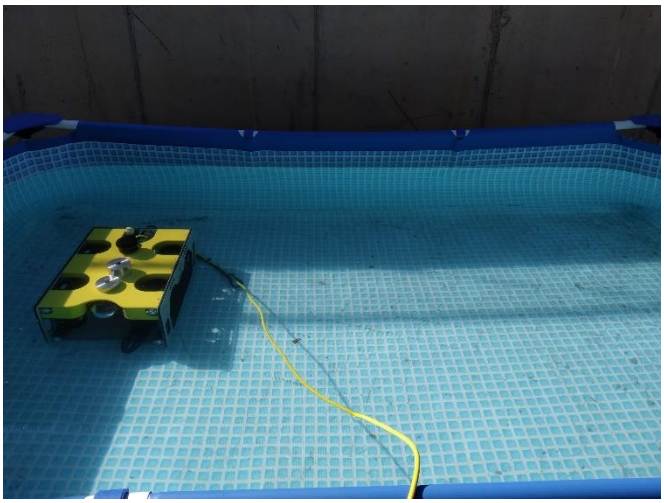


Fig. 56. ROV situado en la orientación 4

Los datos obtenidos se muestran en la Figura 57:

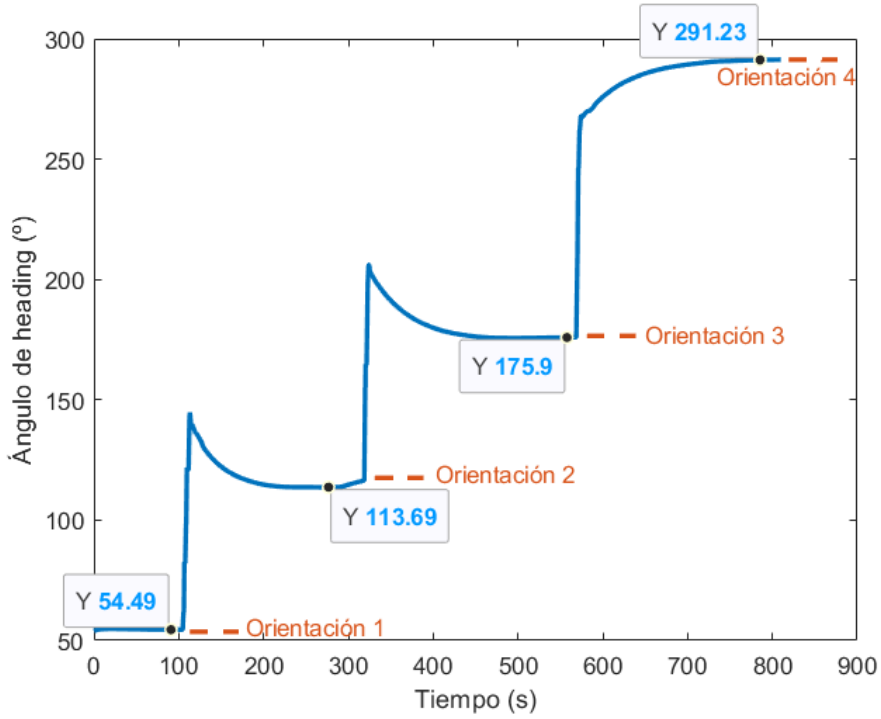


Fig. 57. Ángulo de heading (en la piscina)

Como vemos, el tiempo que tarda en estabilizarse la medida del ángulo es bastante elevado. Además, la diferencia del ángulo medido entre diferentes orientaciones difiere bastante de 90° (ver Tabla 2).

Diferencia	Error
$\mathbf{113.69° - 54.49° = 59.20°}$	$\frac{\lvert 90 - 59.20 \rvert}{90} = \mathbf{34.2\%}$
$\mathbf{175.90° - 113.69° = 62.21°}$	$\frac{\lvert 90 - 62.21 \rvert}{90} = \mathbf{30.9\%}$
$\mathbf{291.23° - 175.90° = 115.33°}$	$\frac{\lvert 90 - 115.33 \rvert}{90} = \mathbf{28.1\%}$
$\mathbf{\lvert 54.49° - 291.23° \rvert = 123.26°}$	$\frac{\lvert 90 - 123.26 \rvert}{90} = \mathbf{37.0\%}$

Tabla 2. Resultados obtenidos (I)

En la primera columna, restamos al valor del ángulo de heading en una orientación el valor del ángulo en la orientación anterior. Puesto que giramos el vehículo en incrementos de 90° (ver Figura 55), en teoría la diferencia debería ser 90° en todos los casos. Sin embargo, los resultados de la primera columna muestran resultados muy dispares. Para tener una idea más clara del error cometido en la medida, calculamos el error relativo de cada medida en la segunda columna de la Tabla 2, según la siguiente expresión:

$$Error\ (\%) = \frac{\lvert 90° - Valor\ medido \rvert}{90°} \cdot 100 \tag{11}$$

Como vemos en la tabla, los errores cometidos son bastante elevados.

Experimentación nº 2: datos de la IMU (II)

A continuación, realizamos una prueba similar a la anterior en el laboratorio de la ETSI. Los datos obtenidos se muestran en la Figura 58:

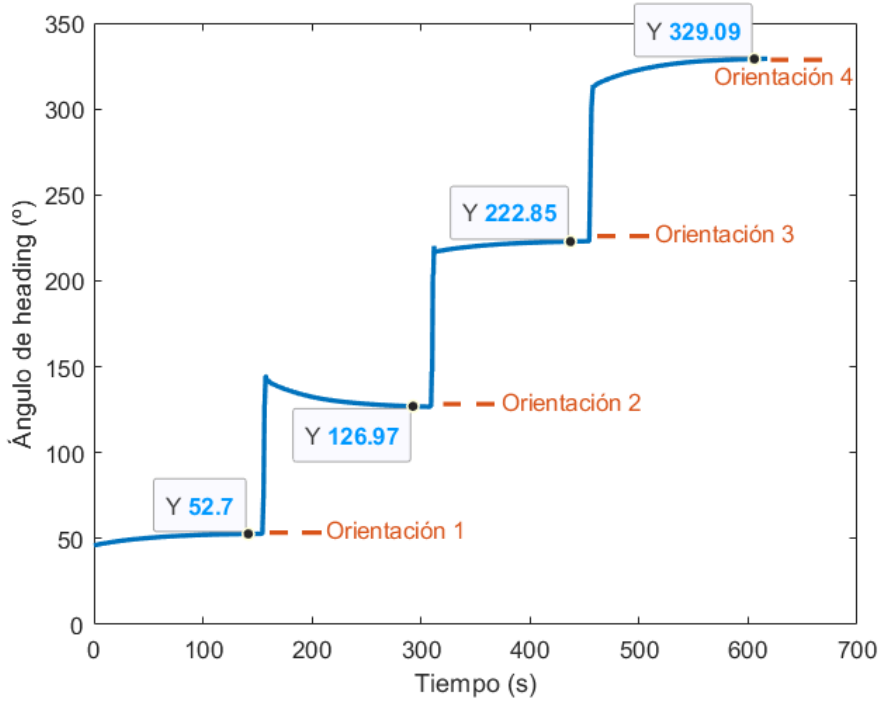


Fig. 58. Ángulo de heading (en el laboratorio)

Como vemos, el tiempo que tarda en estabilizarse la medida del ángulo en este caso es considerablemente menor. Además, la diferencia del ángulo medido entre diferentes orientaciones se aproxima más a 90° (ver Tabla 3).

Diferencia	Error
$\mathbf{126.97° - 52.70° = 74.27°}$	$\frac{\lvert 90 - 74.27 \rvert}{90} = \mathbf{17.5\%}$
$\mathbf{222.85° - 126.97° = 95.88°}$	$\frac{\lvert 90 - 95.88 \rvert}{90} = \mathbf{6.53\%}$
$\mathbf{329.09° - 222.85° = 106.24°}$	$\frac{\lvert 90 - 106.24 \rvert}{90} = \mathbf{18.0\%}$
$\mathbf{52.07° - 329.09° = 83.61°}$	$\frac{\lvert 90 - 83.61 \rvert}{90} = \mathbf{7.10\%}$

Tabla 3. Resultados obtenidos (II)

Comparando las Tablas 2 y 3, podemos observar que el error en las medidas de la IMU en la piscina es considerablemente superior al error en las medidas tomadas en el laboratorio. Esto puede deberse a perturbaciones magnéticas causadas por la armadura de la pared de hormigón situada junto a la piscina.

Experimentación nº 3: datos de la IMU (III)

A continuación, realizamos una prueba cuyo objetivo es verificar que el valor obtenido del ángulo de heading se mantiene constante al variar la ubicación del ROV en la piscina.

Para ello, se guardó el valor del ángulo de heading proporcionado por la IMU en seis ubicaciones distintas dentro de la piscina, tal como se muestra en la Figura 59:

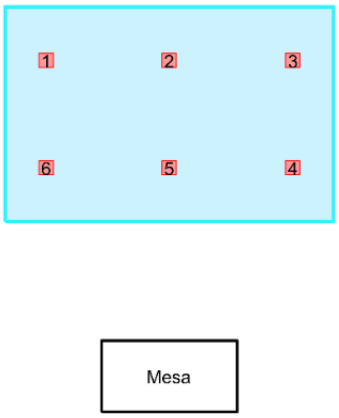


Fig. 59. Posición de las seis ubicaciones dentro de la piscina

Además, en cada ubicación situamos el ROV en 4 orientaciones diferentes, tal como ilustra la Figura 60.

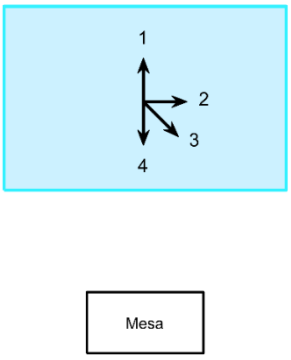


Fig. 60. Orientaciones del ROV en cada ubicación

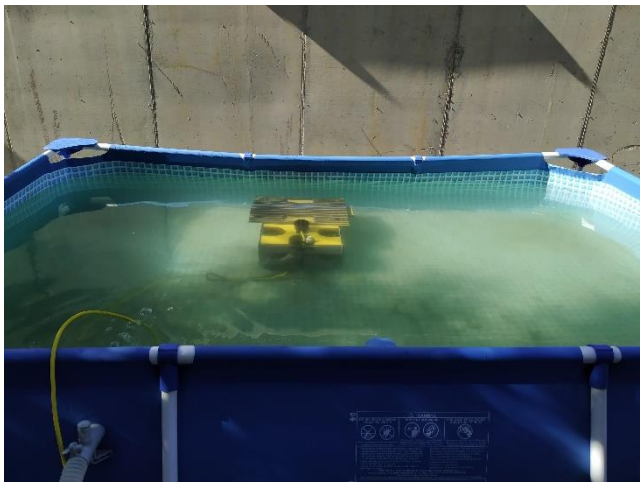


Fig. 61. ROV situado en la posición nº 2, orientación nº 1

Situamos el ROV en la orientación adecuada y, una vez que ha transcurrido 1 min aproximadamente, guardamos la medida del ángulo y cambiamos la orientación. Los datos obtenidos se muestran a continuación (ver Figuras 62, 63, 64, 65, 66 y 67).

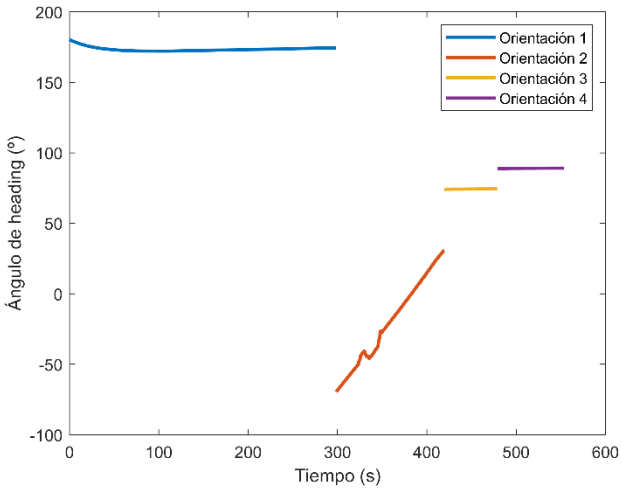


Fig. 62. Ubicación 1

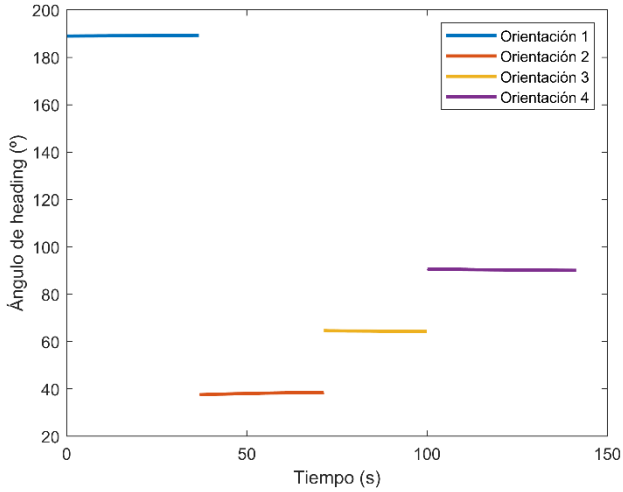


Fig. 63. Ubicación 2

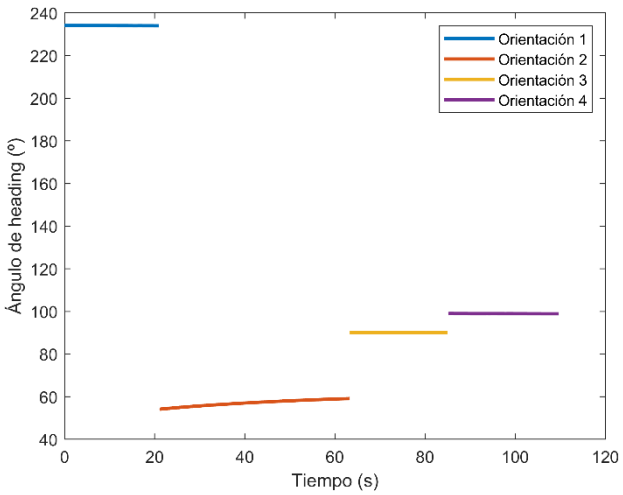


Fig. 64. Ubicación 3

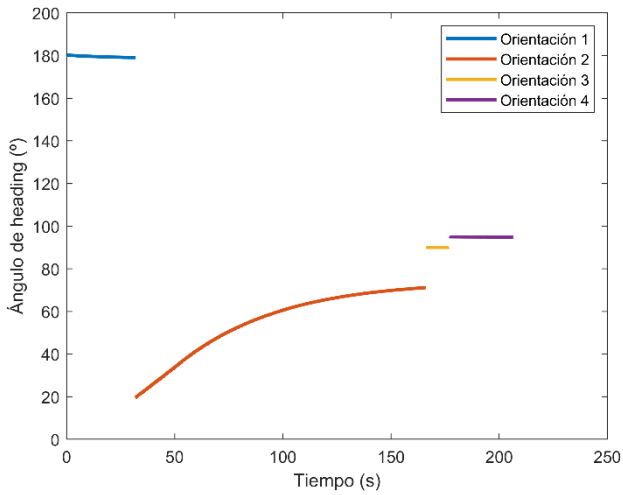


Fig. 65. Ubicación 4

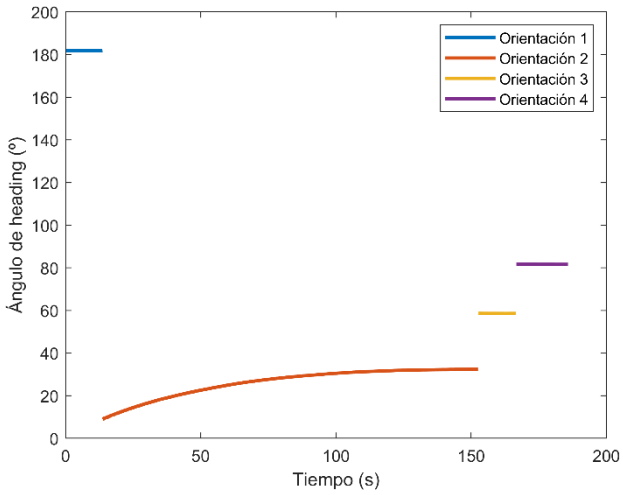


Fig. 66. Ubicación 5

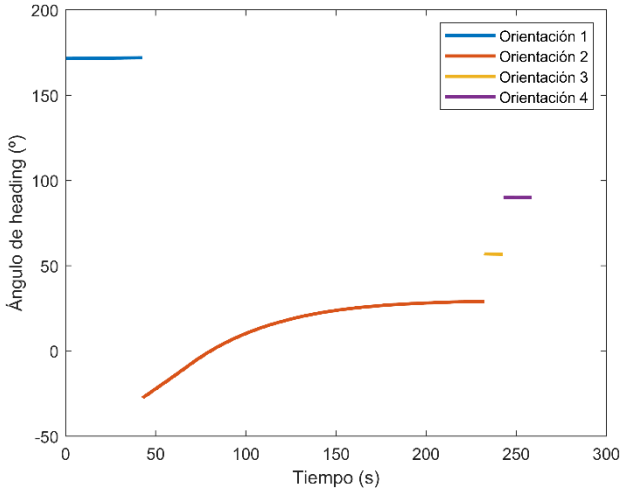


Fig. 67. Ubicación 6

La siguiente tabla muestra el valor medio del ángulo de heading en cada orientación, considerando los valores medios obtenidos para cada orientación, en las seis ubicaciones:

Orientación	Valor medio
Orientación 1	188.31°
Orientación 2	31.92°
Orientación 3	72.39°
Orientación 4	90.81°

Tabla 4. Valores medios en cada orientación

A continuación, se muestran los errores para cada orientación (ver Figuras 68, 69, 70 y 71).

Orientación 1

Posición	Valor medio	Error
1	173.54°	7.84%
2	189.16°	0.45%
3	234.06°	24.3%
4	179.52°	4.67%
5	181.87°	3.42%
6	171.73°	8.80%

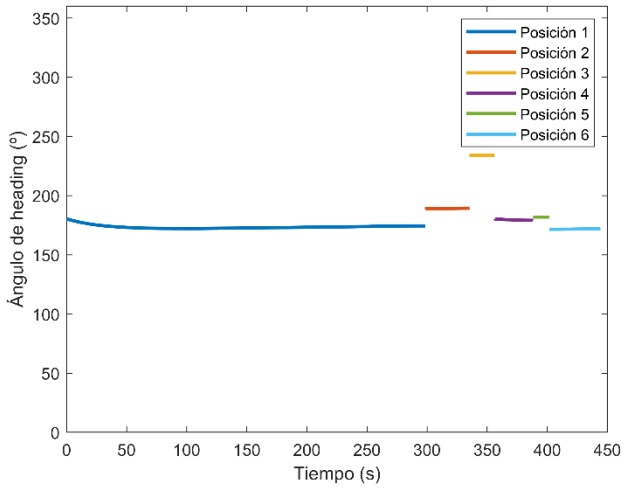


Fig. 68. Orientación 1

Orientación 2

Posición	Valor medio	Error
1	−19.70°	161.7%
2	38.17°	19.6%
3	57.10°	78.9%
4	55.17°	72.9%
5	26.31°	17.6%
6	14.75°	53.8%

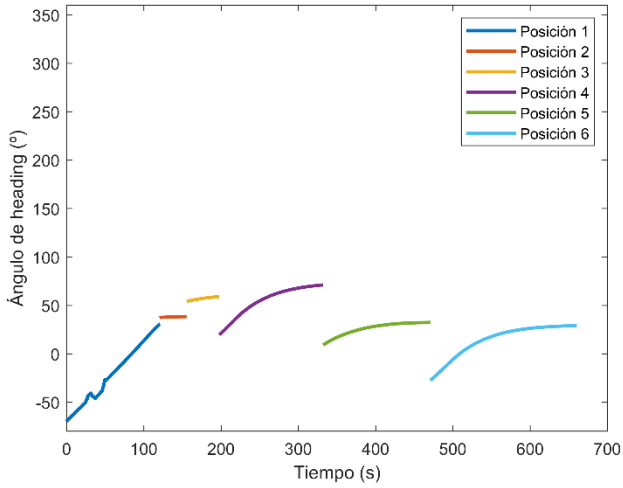


Fig. 69. Orientación 2

Orientación 3

Posición	Valor medio	Error
1	74.33°	2.68%
2	64.42°	11.0%
3	90.05°	24.4%
4	90.06°	24.4%
5	58.57°	19.1%
6	56.90°	21.4%

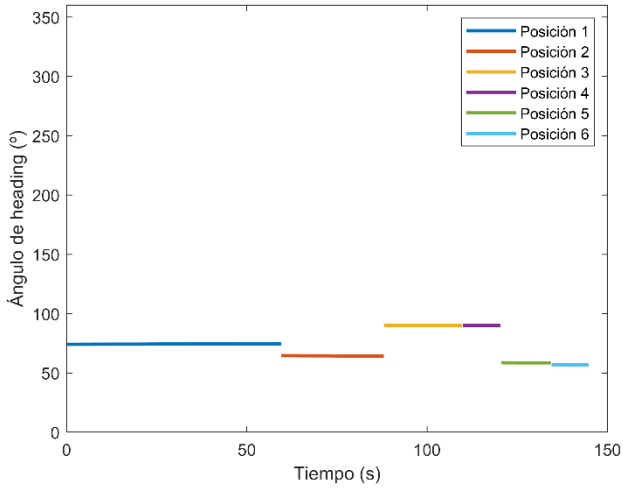


Fig. 70. Orientación 3

Orientación 4

Posición	Valor medio	Error
1	89.06°	1.94%
2	90.30°	0.56%
3	98.96°	8.96%
4	94.90°	4.50%
5	81.65°	10.09%
6	90.01°	0.88%

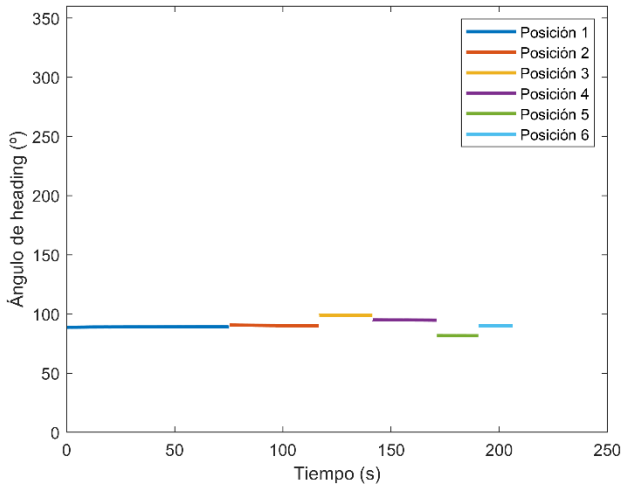


Fig. 71. Orientación 4

Como vemos, los errores varían notablemente en cada orientación, por lo que es difícil establecer alguna tendencia. De nuevo, esto puede deberse a perturbaciones magnéticas causadas por la armadura de la pared de hormigón situada junto a la piscina, ya que el magnetómetro de la IMU es muy sensible a este tipo de perturbaciones.

Otro factor que interviene en la medida del ángulo de heading es el filtro de Kalman Extendido implementado en la PixHawk, cuya influencia se estudia en la siguientes pruebas.

5.1.2. Filtro de Kalman Extendido

Experimentación nº 4: versiones del filtro de Kalman Extendido

El objetivo de esta prueba es comprobar la influencia del filtro de Kalman Extendido. Para ello, situamos el ROV en un punto fijo de la piscina, y lo rotamos 90º una vez que se ha estabilizado la medida del ángulo de heading, tal como ilustra la figura 72. Realizamos el mismo procedimiento modificando los parámetros mencionados en el capítulo 3 que configuran el EKF.

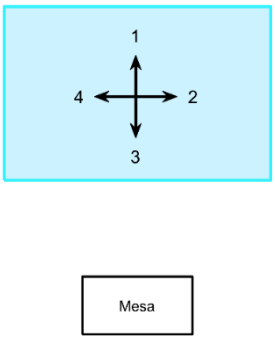


Fig. 72. Orientaciones del ROV

De esta forma, tomamos las medidas activando EKF2, EKF3 y sin activar el filtro. Los resultados obtenidos se muestran en la Figura 73.

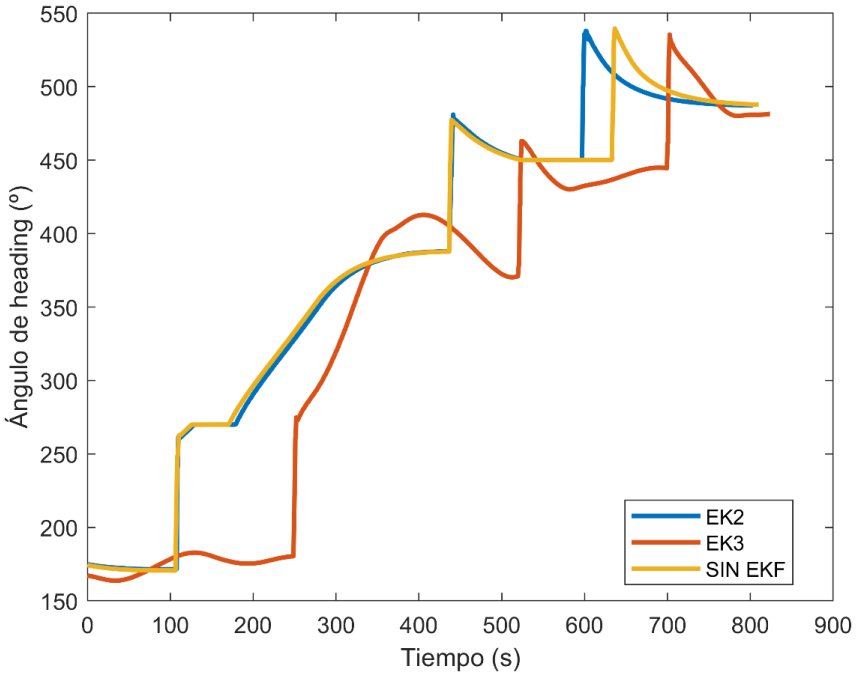


Fig. 73. Ángulo de heading modificando el filtro de Kalman

Como vemos, existen ciertas diferencias entre las versiones del filtro EK2 y EK3, siendo esta última algo más inestable. Como se explicó anteriormente, el filtro de Kalman utiliza los datos procedentes del acelerómetro, magnetómetro y giroscopio, así como del sistema de posicionamiento submarino (UGPS) para proporcionar una estimación de la medida.

En nuestro caso, la información procedente del sistema UGPS no estaba disponible, al no estar el Water Linked operativo. Esta podría ser una de las razones por las que las estimaciones que proporciona el filtro de Kalman no tengan el nivel de exactitud esperado.

Dado que la versión EK2 es la que viene activada por defecto, la seguiremos manteniendo.

Experimentación nº 5: modificación de parámetros

El objetivo de esta prueba es comprobar la influencia del parámetro AHRS_YAW_P. Como se comentó en el capítulo 3, este parámetro varía entre 0.1 y 0.4, y controla el peso que tiene el magnetómetro en la medida del ángulo de heading. Un valor más alto significa que la medida se actualizará de acuerdo con el magnetómetro más rápidamente.

Para ello, situamos el ROV en un punto fijo de la piscina y lo rotamos 90º una vez que se ha estabilizado la medida del ángulo de heading. Realizamos el mismo procedimiento para los siguientes valores del parámetro:

- AHRS_YAW_P = 0.1
- AHRS_YAW_P = 0.4

Los resultados obtenidos se muestran en la Figura 74.

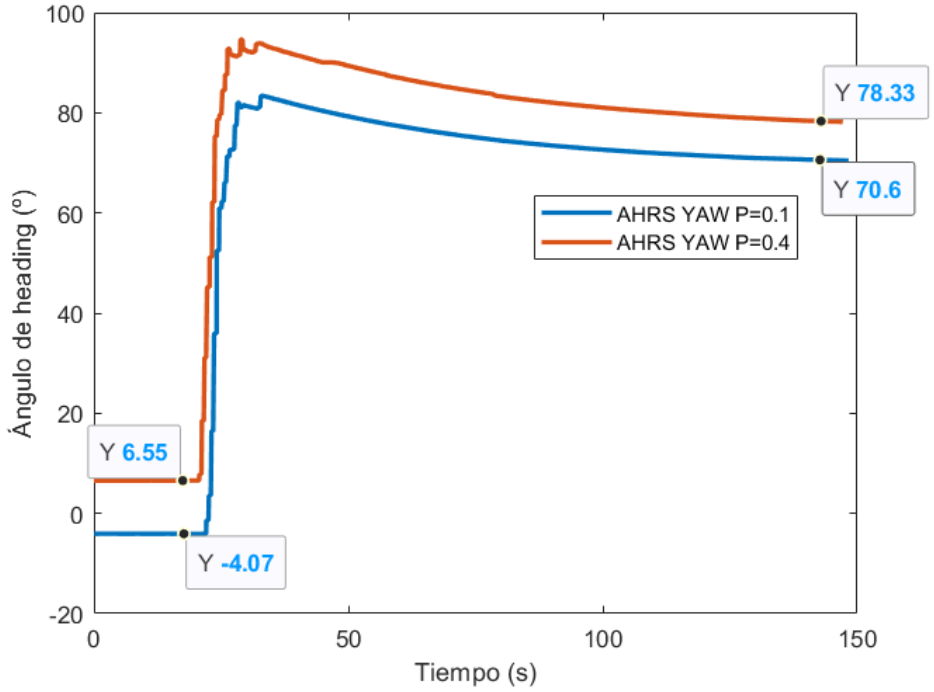


Fig. 74. Ángulo de heading para diferentes valores del parámetro AHRS_YAW_P

Como vemos, las medidas del ángulo varían entre sí 10° aproximadamente. Sin embargo, el tiempo de estabilización es similar en ambos casos, por lo que la modificación de este parámetro no resulta de gran utilidad.

5.1.3. Coherencia de los datos proporcionados por la IMU en movimiento

Experimentación nº 6: datos de la IMU (IV)

La prueba descrita a continuación tuvo lugar en las instalaciones de Salinas del Astur (Punta Umbría), unos esteros dedicados a la producción acuícola de doradas, lubinas y lenguados, entre otros.

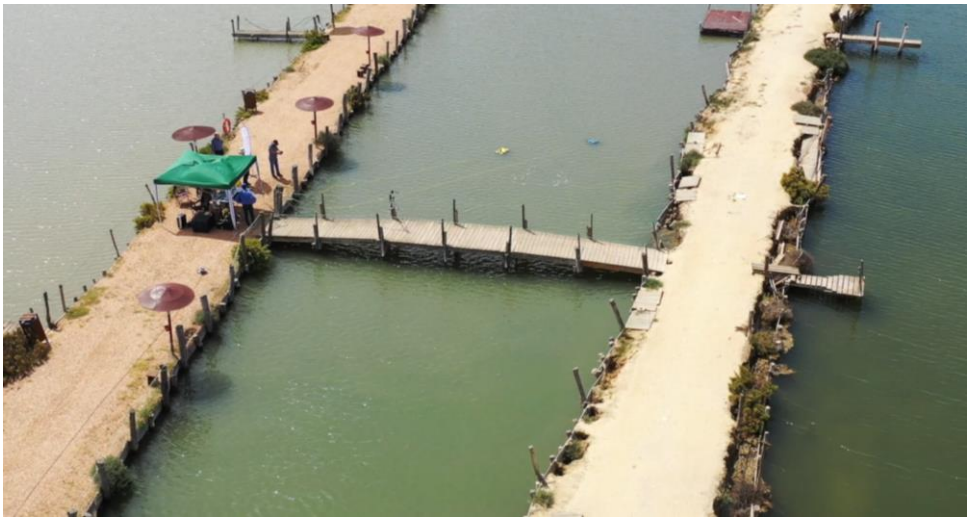


Fig. 75. Salinas del Astur

A diferencia de las pruebas anteriores, en este caso implementamos un control de orientación en el ROV para que gire hasta situarse según un ángulo de referencia dado. Este controlador de velocidad fue diseñado por Alejandro Garrocho Cruz, en el marco del proyecto internacional KTT SeaDrones.

El controlador compara continuamente el valor del ángulo de heading proporcionado por la IMU con el valor de referencia, y calcula una determinada velocidad proporcional a la diferencia entre ambos.

Realizamos esta prueba en repetidas ocasiones. En cada una, modificamos el ángulo de referencia transcurrido un tiempo. Los resultados obtenidos se muestran a continuación (ver Figura 76).

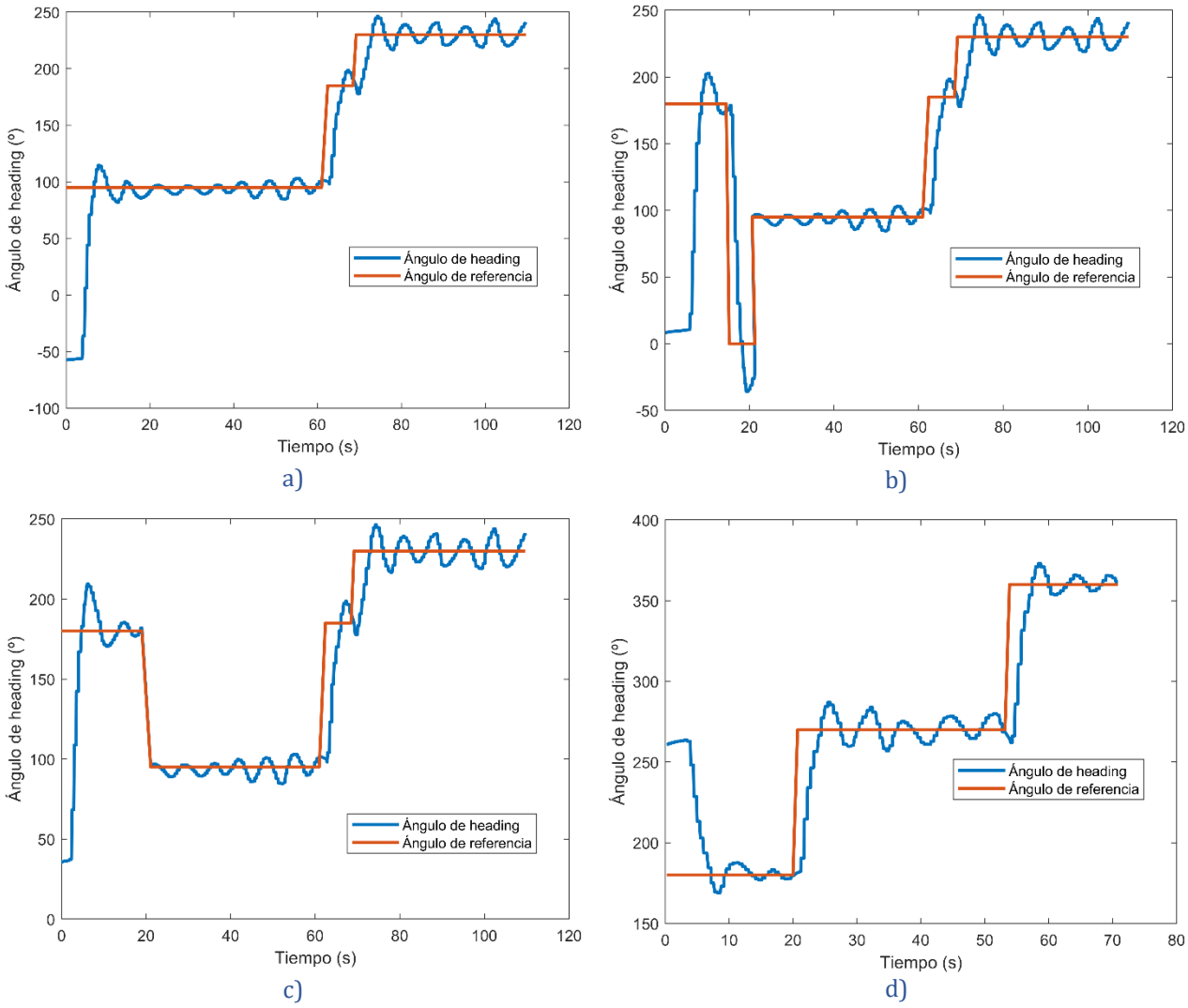


Fig. 76. a) Control del ángulo de heading (I); b) Control del ángulo de heading (II); c) Control del ángulo de heading (III); d) Control del ángulo de heading (IV)

En las gráficas anteriores se observan ciertas oscilaciones en torno al valor de referencia, causadas por el leve empuje de las olas del estero. Comprobamos visualmente que para cada valor de referencia que le fue entregado al controlador, la orientación del ROV se mantenía estable en el entorno de las oscilaciones, sin presentar deriva a lo largo del tiempo. En consecuencia, verificamos que, en este entorno, el controlador basado en las lecturas de la IMU es perfectamente capaz de seguir al valor de referencia indicado en cada momento.

Experimentación nº 7: datos de la IMU (V)

Posteriormente, realizamos la misma prueba en la piscina de la ETSI. Los resultados obtenidos se muestran a continuación (ver Figura 77).

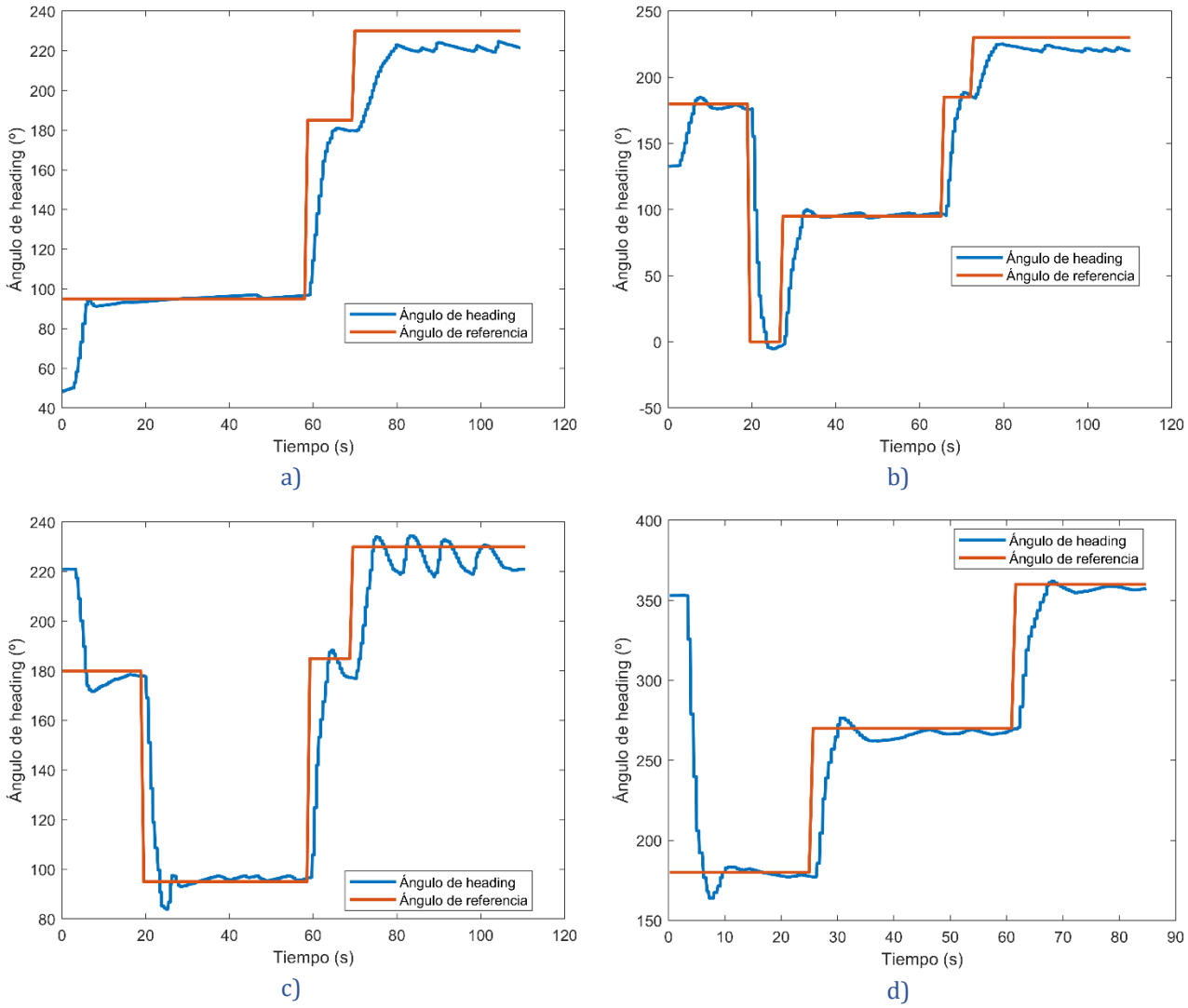


Fig. 77. a) Control del ángulo de heading (V); b) Control del ángulo de heading (VI); c) Control del ángulo de heading (VII); d) Control del ángulo de heading (VIII)

De nuevo, el controlador lleva a cabo correctamente el seguimiento del ángulo de referencia. En este caso, no existen perturbaciones producidas por la marea, por lo que las oscilaciones disminuyen notablemente.

Además, podemos observar cierta tendencia en el ángulo de heading a mantener un error constante con respecto al ángulo de referencia. Este hecho es congruente comprobación visual de la deriva que sufría el ROV para los distintos valores de referencia que se le proporcionaban al controlador. Como consecuencia de ello, concluimos que la no estabilidad de las medidas proporcionadas por la IMU, y las

derivas mostradas en este escenario, al inicio del capítulo, era la causa del error estacionario y la deriva.

5.1.4. Conclusiones

Tras la realización de estos experimentos, podemos concluir que en las medidas del ángulo de heading proporcionadas por la IMU del ROV, cuando este se encuentra en entornos delimitados por paredes de hormigón, la robustez esperada en las medidas es baja, y tarda un tiempo considerablemente elevado en estabilizarse. Esto puede ser debido a los siguientes factores:

- La ausencia del sistema de posicionamiento submarino (UGPS), que permitiría al filtro de Kalman estimar la actitud con más fiabilidad.
- La existencia de perturbaciones magnéticas producidas por las barras de acero presentes en el hormigón armado de la pared situada junto a la piscina. Como se explicó anteriormente, la IMU, especialmente el magnetómetro, es muy sensible a este tipo de interferencias.

Por otro lado, las pruebas realizadas en las Salinas del Astur con el ROV en movimiento implementando un algoritmo de control de la orientación demuestran que las medidas de la actitud sí son válidas en este caso, debido a la ausencia de perturbaciones magnéticas.

Sin embargo, durante estas mismas pruebas realizadas en la piscina de la ETSI, observamos que el vehículo presenta cierta deriva, por lo que no se puede considerar que el resultado del control de orientación basado en la información proporcionada por la IMU sea aceptable.

No obstante, decidimos usar la IMU para las maniobras de reorientación porque, aunque impreciso, el controlador es capaz de llevar el vehículo hasta un valor lo suficientemente próximo al valor deseado, como para que, cuando cambiamos al controlador que controla la orientación mediante la medida del sonar, éste sea capaz de estabilizar la orientación del vehículo cerca del valor de referencia deseado.

5.2. Pruebas de navegación

Durante el proceso de diseño del algoritmo completo, se crearon programas independientes que realizan cada una de las tareas por separado. De esta forma, una vez comprobado el correcto funcionamiento de estos programas, se añaden al algoritmo principal.

En la Figura 78 se muestra la fecha aproximada en la que se realizaron los diferentes experimentos:

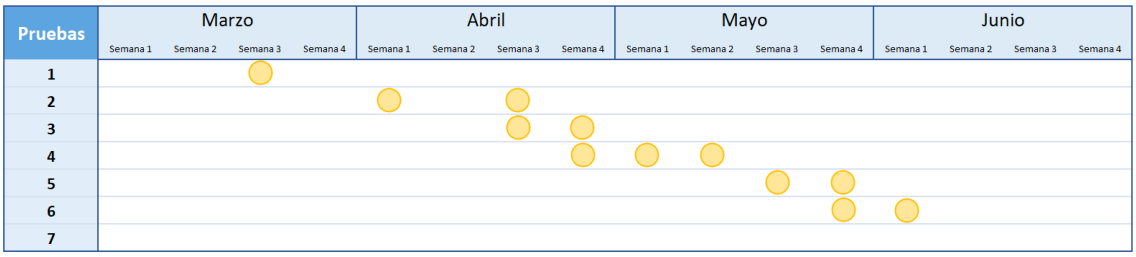


Fig. 78. Calendario de pruebas (II)

En este capítulo, se describen las pruebas llevadas a cabo para la comprobación de los diferentes programas, así como los resultados observados en cada una.

5.2.1. Experimentación nº 1: Sector de muestreo del sonar Ping360

Al acotar el ángulo de muestreo del sonar a 30°, advertimos que la imagen obtenida mediante el sonar Ping360 no representaba los objetos que se encontraban justo delante del ROV, sino los que estaban detrás. Es decir, el sector de muestreo era opuesto a la velocidad de avance del ROV, tal como muestra la figura 79:

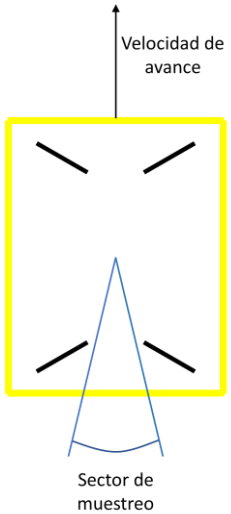


Fig. 79. Sector de muestreo por defecto

Por lo tanto, en primer lugar fue necesario modificar el sector de muestreo, para que coincidiera con el sentido de la velocidad de avance (ver Figura 80).

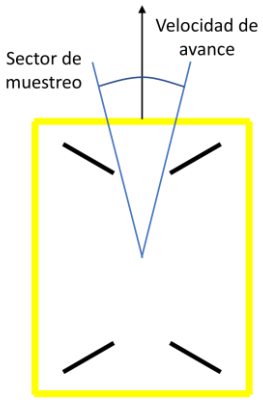


Fig. 80. Sector de muestreo modificado

Una vez corregido el ángulo de muestreo, diseñamos el algoritmo encargado de identificar la pared, explicado anteriormente en el apartado 3.2.2. A continuación, empezamos las pruebas de los sistemas de control de navegación del ROV.

5.2.2. Experimentación nº 2: Navegación hacia la pared de la piscina sin controlar el ángulo de giro

El objetivo de esta prueba es comprobar que el algoritmo explicado en el apartado 4.1.1. calcula correctamente la distancia a la pared de la piscina. Para ello, controlamos únicamente la velocidad de avance del ROV, que es proporcional a la distancia. Cuando el ROV se encuentra lo suficientemente cerca de la pared, se detiene. Los resultados obtenidos se muestran a continuación (ver Figura 81).

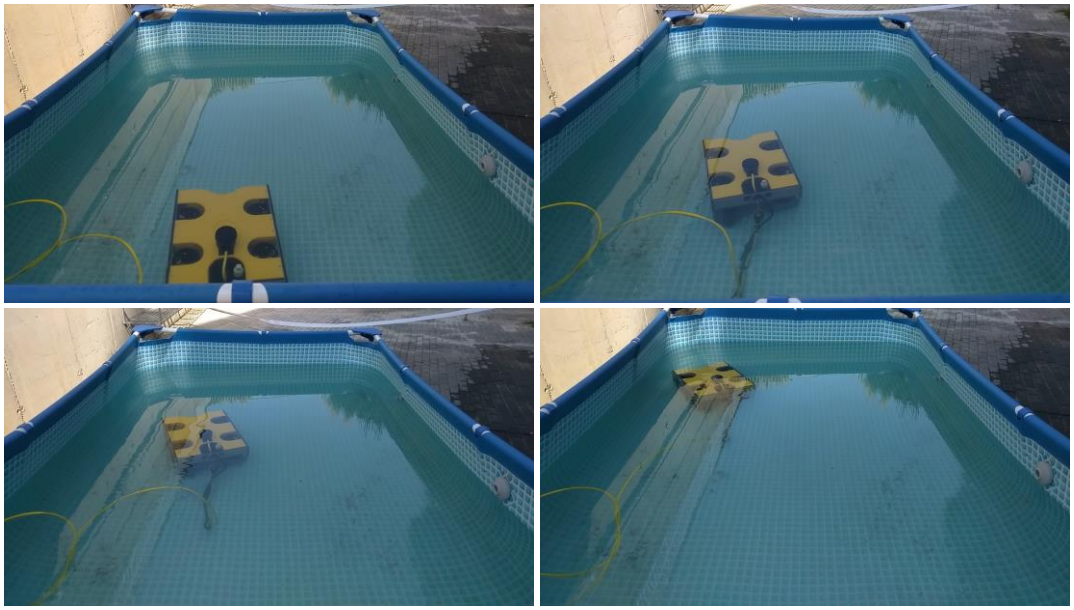
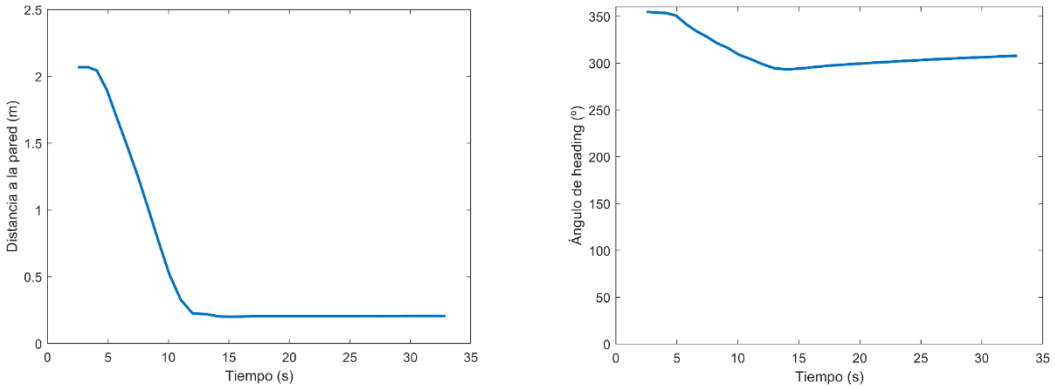


Fig. 81. Secuencia de movimiento (Experimentación nº 2)

La Figura 82 muestra la distancia a la pared durante la prueba realizada (a), así como el ángulo de heading (b), frente al tiempo:



a) b)

Fig. 82. a) Distancia a la pared (I); b) Ángulo de heading (I)

Como vemos, la distancia va disminuyendo mientras el ROV avanza, hasta permanecer constante cuando el ROV se detiene en frente de la pared, a unos 20 cm aproximadamente.

Durante esta prueba, aunque el vehículo debería avanzar en línea recta, observamos un movimiento de cabeceo no deseado. Es decir, el ángulo de heading no permanece constante, ya que el vehículo se desvía hacia un lado.

Este comportamiento era de esperar, ya que se trata de una navegación en bucle abierto respecto a la orientación del vehículo y, como resultado de cualquier perturbación por mínima que sea, es lógico que el vehículo se desvíe. Esta deriva indeseada pudo ser corregida posteriormente en el algoritmo de control de orientación.

5.2.3. Experimentación nº 3: Mantener un ángulo de 90° con respecto a la pared de la piscina utilizando el sonar Ping360

El objetivo de esta prueba es comprobar que el algoritmo explicado en el apartado 4.1.2. controla correctamente el ángulo con respecto a la pared de la piscina, de manera que mantiene el vehículo perpendicular a la misma. Para ello, actuamos únicamente sobre la velocidad de giro, que será proporcional al error de orientación. Para provocar una perturbación en el ángulo, empujamos al ROV suavemente desde la posición inicial con la orientación correcta. El controlador rectifica la orientación del vehículo hasta estabilizarla en una configuración en la que está orientado a 90° respecto a la pared frontal (ver Figuras 83 y 84).

Fig. 83. Posición inicial (Experimentación nº 3)

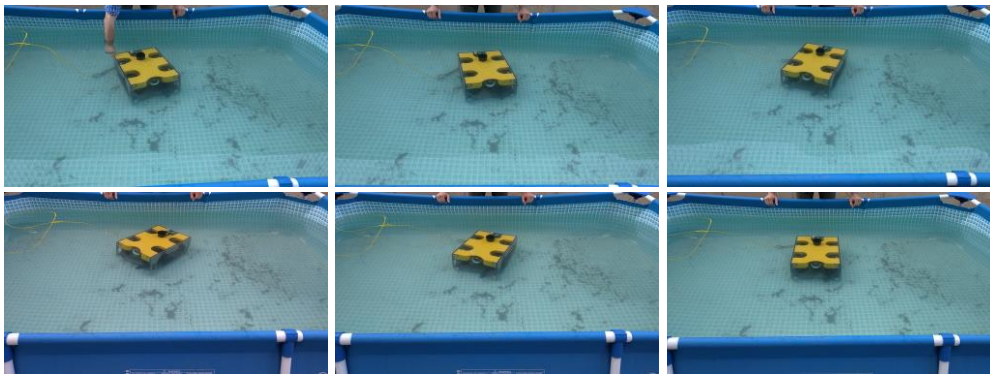
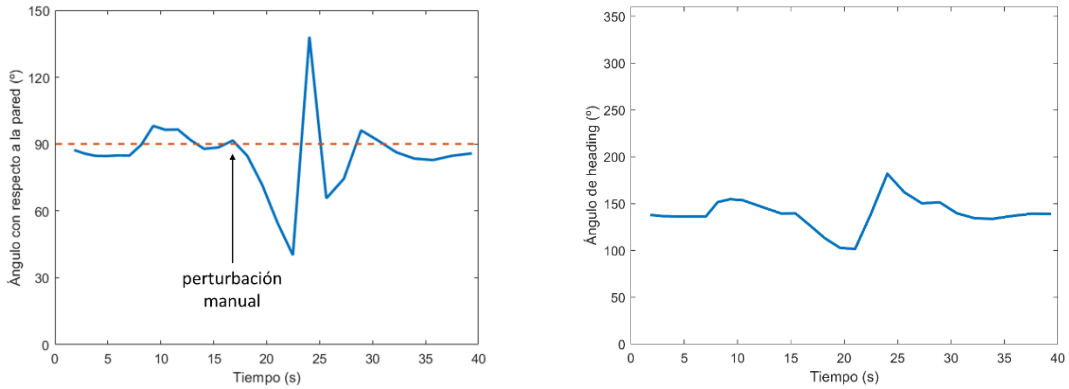


Fig. 84. Secuencia de movimiento (Experimentación nº 3)

La Figura 85 muestra el ángulo con respecto a la pared frontal durante la prueba realizada (a), así como el ángulo de heading (b), frente al tiempo:



a) b)

Fig. 85. a) Ángulo con respecto a la pared (II); b) Ángulo de heading (II)

Como vemos, el ángulo medido tiende a estabilizarse en torno a 90°. Además, debido al tiempo que tarda el sonar en muestrear el sector (unos 2 segundos aproximadamente), la frecuencia de actualización no es muy elevada.

Por otro lado, el ángulo de heading sufre cambios similares al ángulo con respecto a la pared, al ser el ROV empujado manualmente. Finalmente, cuando dejamos el vehículo en reposo, se estabiliza en un valor determinado.

5.2.4. Experimentación nº 4: Navegación hacia la pared manteniendo un ángulo de 90° con la misma

El objetivo de esta prueba es comprobar que podemos fusionar los dos algoritmos explicados anteriormente. En este caso, el sistema de control debe calcular simultáneamente la distancia a la pared y el ángulo con respecto a la misma, y establecer las velocidades de avance y giro correspondientes. Tras realizar la prueba, obtenemos los siguientes resultados (ver Figura 86).

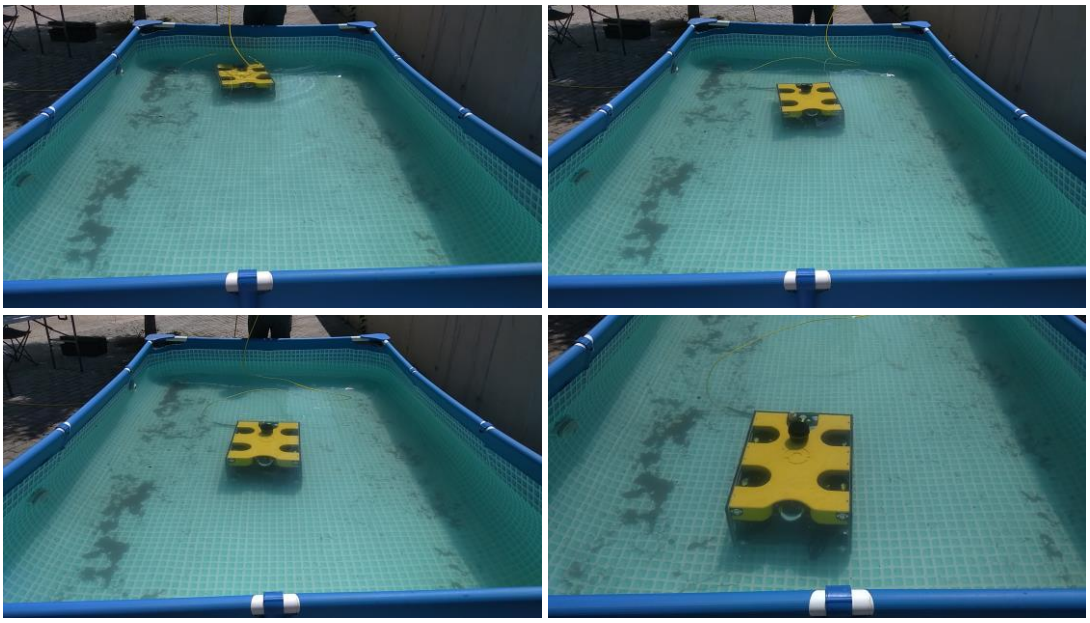
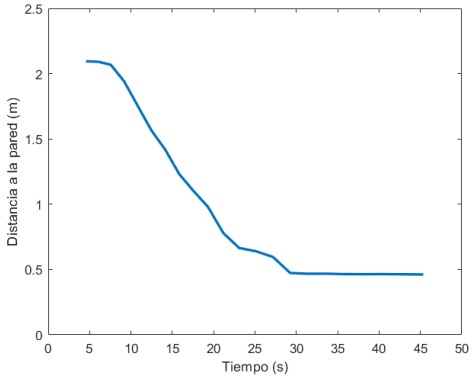
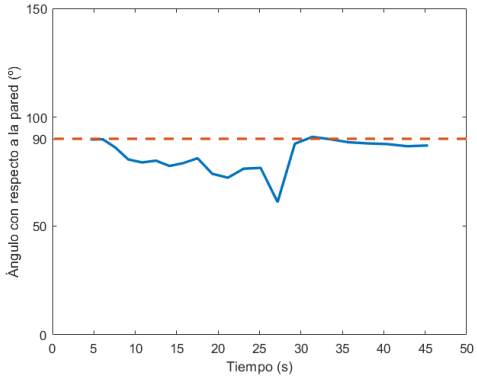


Fig. 86. Secuencia de movimiento (Experimentación nº 4)

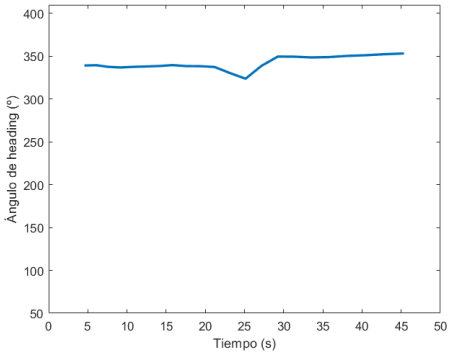
La Figura 87 muestra la distancia a la pared frontal durante la prueba realizada (a), así como el ángulo con respecto a la pared (b), y el ángulo de heading (c), frente al tiempo:



a)



b)



c)

Fig. 87. a) Distancia a la pared (III) ; b) Ángulo con respecto a la pared (III); c) Ángulo de heading (III)

Como vemos, el ROV avanza hasta situarse a una determinada distancia con la pared. Finalmente se detiene. En cuanto al ángulo con respecto a la pared, podemos ver que inicialmente se desorienta notablemente, aunque finalmente el vehículo se estabiliza correctamente en torno a 90°. Por otro lado, el ángulo de heading se estabiliza en un determinado valor, por lo que es coherente con los resultados de distancia y ángulo con respecto a la pared mostrados en la misma figura.

5.2.5. Experimentación nº 5: Navegación de un extremo a otro de la piscina

Esta prueba es similar a la anterior, con la diferencia de que una vez que el ROV se ha estabilizado frente a la pared, gira 180° y, a continuación, avanza hacia el otro extremo de la piscina. Los resultados obtenidos se muestran a continuación (ver Figura 88).

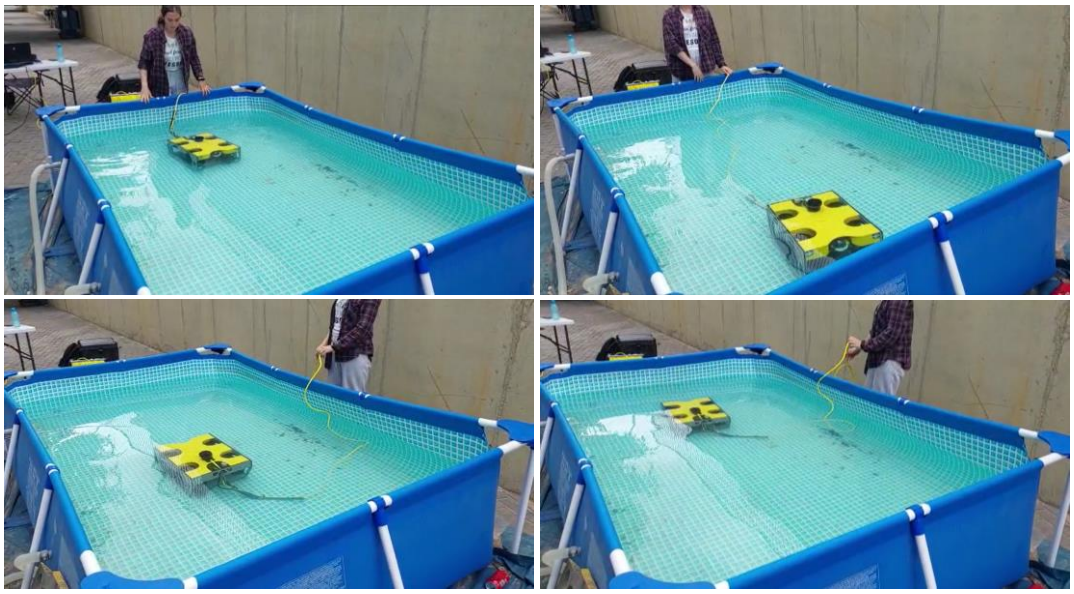
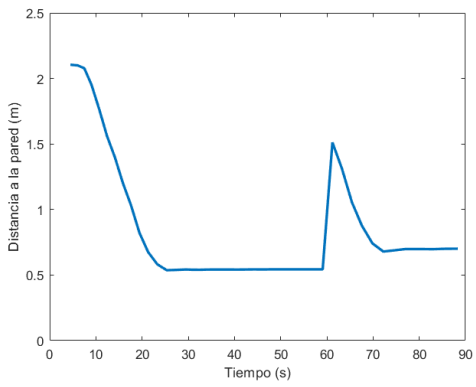
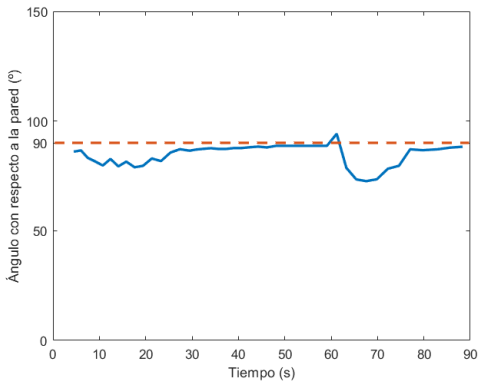


Fig. 88. Secuencia de movimiento(Experimentación nº 5)

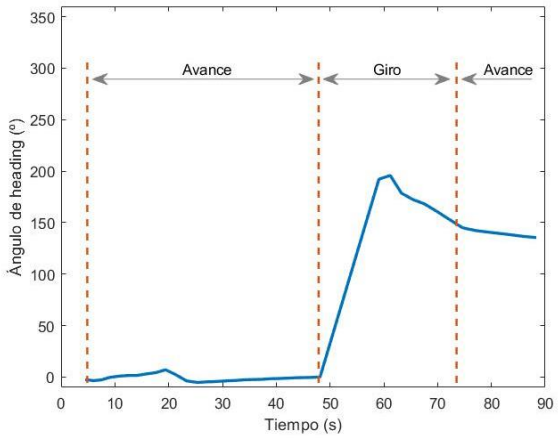
La Figura 89 muestra la distancia a la pared frontal durante la prueba realizada (a), así como el ángulo con respecto a la pared (b), y el ángulo de heading (c), frente al tiempo:



a)



b)



c)

Fig. 89. a) Distancia a la pared (IV); b) Ángulo con respecto a la pared (IV); c) Ángulo de heading (IV)

Como vemos, la distancia a la pared disminuye progresivamente mientras el ROV se va acercando. Una vez se ha estabilizado, gira 180º (mientras gira no se muestrea el entorno, por lo que no se mide la distancia). De esta forma, la distancia a la pared de enfrente aumenta al instante siguiente de terminar de girar. De nuevo, el ROV se aproxima a la pared hasta estabilizarse.

Además, el ROV se estabiliza correctamente en torno a 90°. De nuevo, mientras gira 180° no se muestrea el entorno, por lo que no se mide el ángulo. Es por esto que se ve un cambio más brusco al terminar de girar y muestrear de nuevo, aunque finalmente vuelve a estabilizarse.

Por otro lado, mientras el ROV se acerca a una pared de la piscina, el ángulo de heading permanece prácticamente constante. Al girar, el ángulo aumenta bruscamente, hasta que se estabiliza de nuevo.

En la prueba anterior, el ROV recorre la piscina dos veces. En realidad, mientras se ejecute el algoritmo, podría recorrer la piscina de un extremo a otro infinitas veces. A continuación, se muestran los resultados cuando el ROV recorre la piscina en tres ocasiones (ver Figura 90).

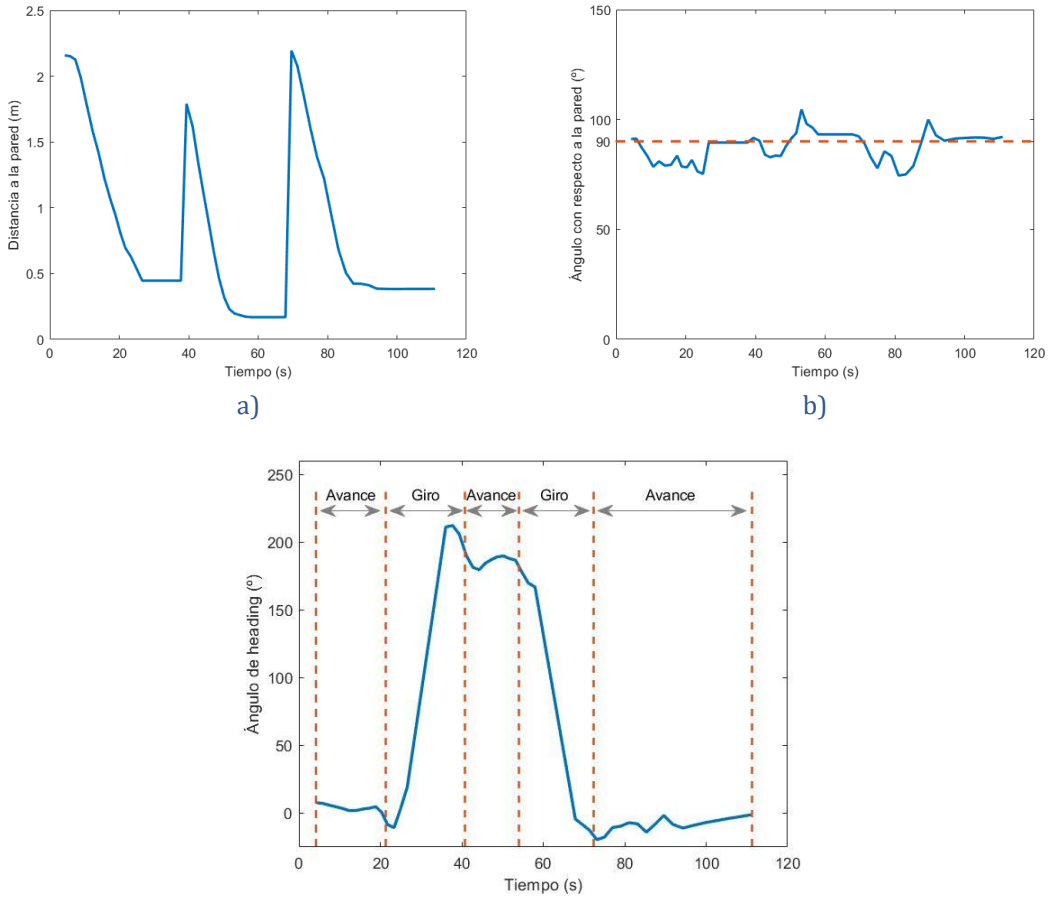
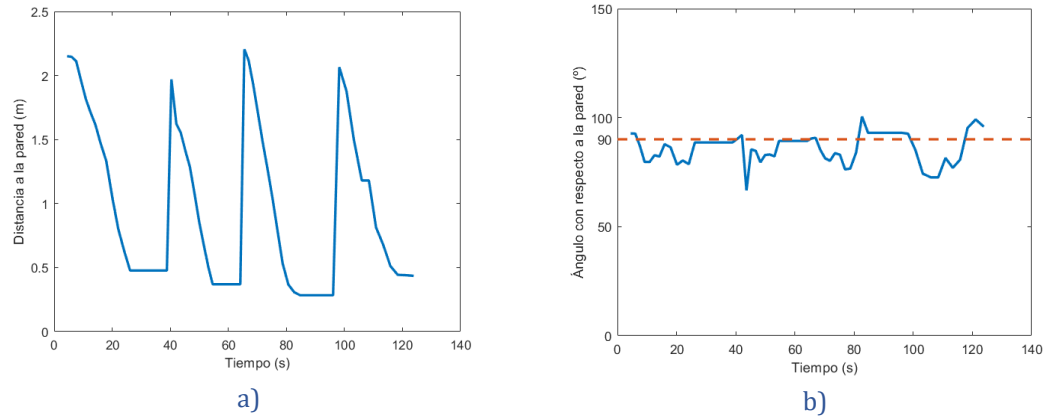
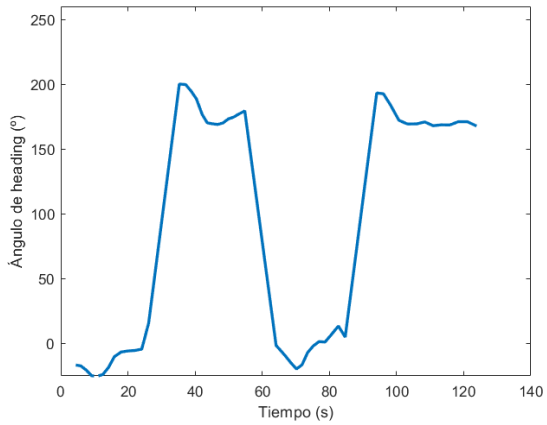


Fig. 90. a) Distancia a la pared (V); b) Ángulo con respecto a la pared (V); c) Ángulo de heading (V)

Finalmente, si recorre la piscina cuatro veces, obtenemos los siguientes resultados (ver Figura 91).





c)

Fig. 91. a) Distancia a la pared (VI); b) Ángulo con respecto a la pared (VI); c) Ángulo de heading (VI)

5.2.6. Experimentación nº 6: Navegación hacia un objeto

Para navegar hacia un objeto, en primer lugar, realizamos un barrido con el sonar de 360°, y así identificamos los objetivos disponibles. Una vez seleccionado el objetivo, el ROV se orienta hasta quedar alineado con el mismo. Finalmente, avanza hasta situarse a cierta distancia del objeto en cuestión. Cuando alcanza la posición deseada, se mantiene en ese punto durante un tiempo y, finalmente, muestrea de nuevo el entorno para identificar nuevos objetivos si los hubiera.

La imagen obtenida en una de las pruebas realizadas se muestra en la Figura 92.

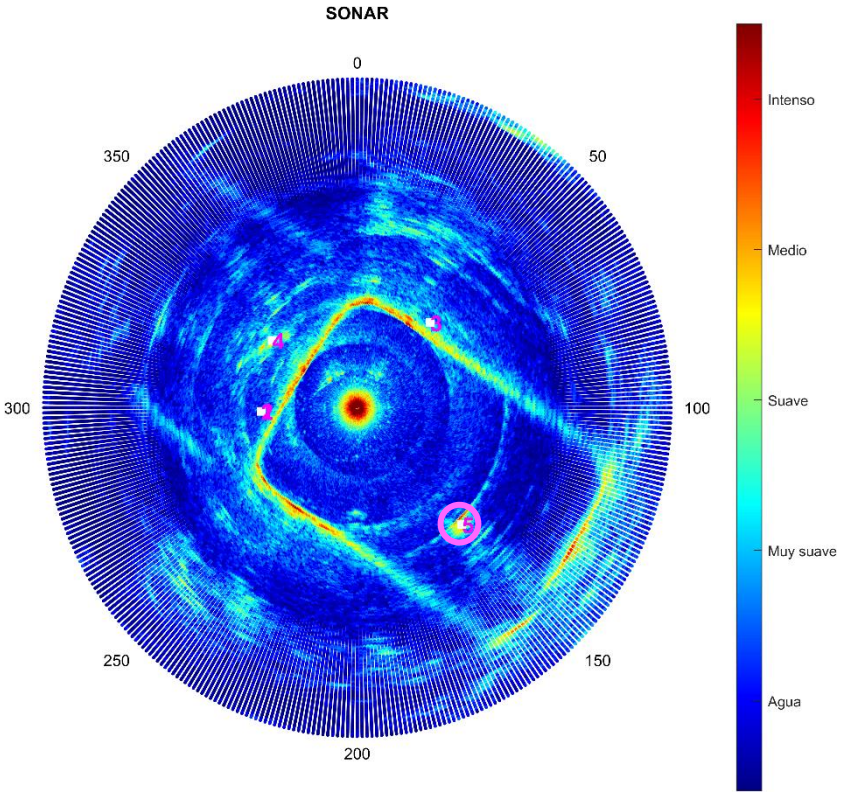


Fig. 92. Muestreo de 360°

Como vemos, en este caso, solo existe un objeto dentro la piscina (nº 5). Tras seleccionar este objeto, el ROV en primer lugar se orienta hacia el mismo, y posteriormente avanza hacia él. Los resultados de la prueba se muestran a continuación (ver Figura 93).

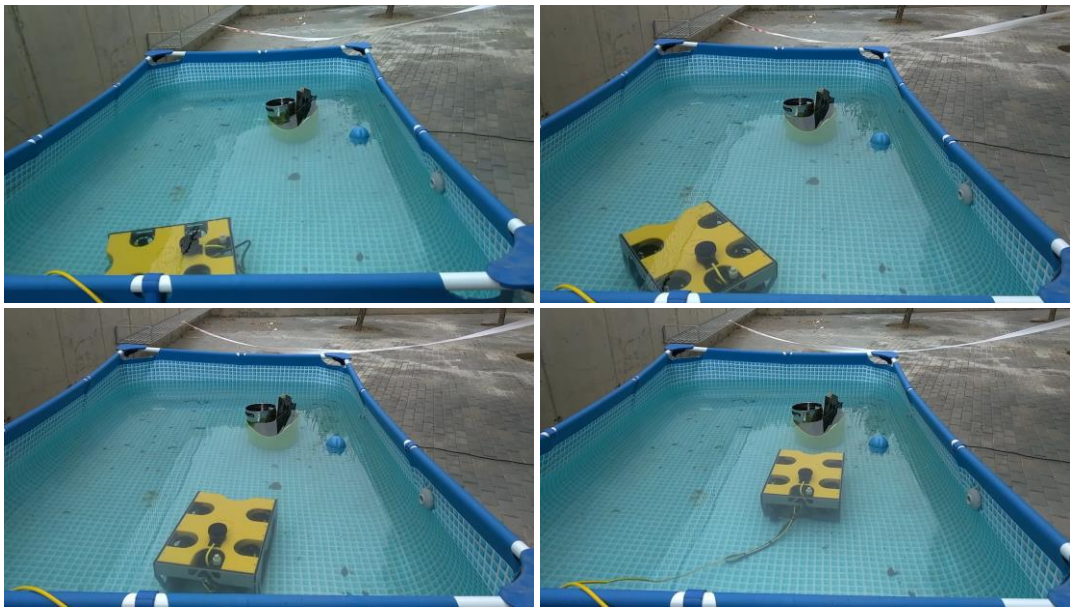
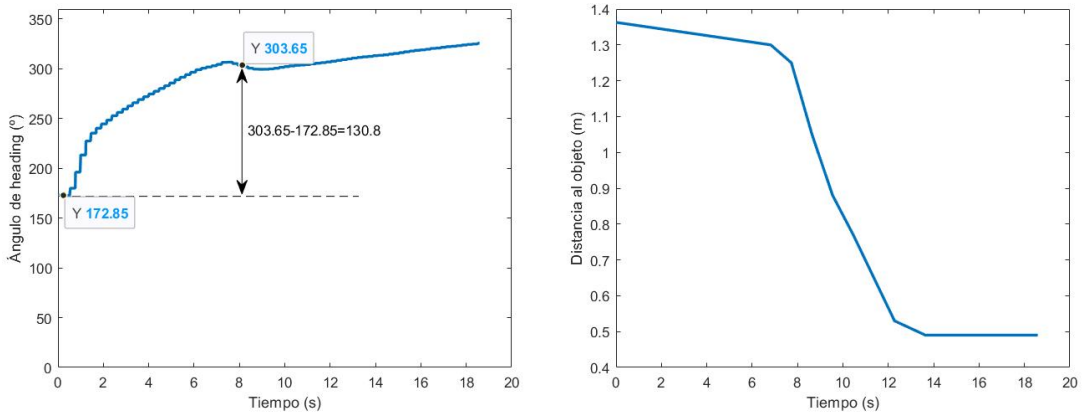
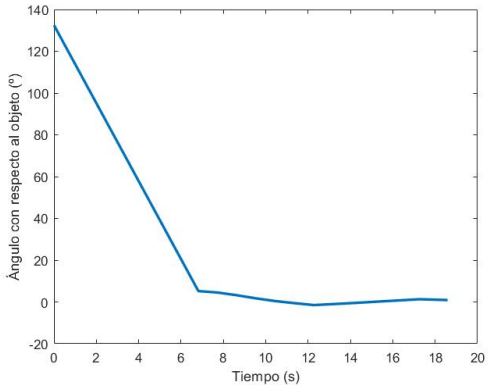


Fig. 93. Secuencia de movimiento (Experimentación nº 6)

La Figura 94 muestra el ángulo de heading durante la prueba realizada (a), así como la distancia al objeto (b), y el ángulo con respecto al mismo (c), frente al tiempo:



a) b)



c)

Fig. 94. a) Ángulo de heading (VII); b) Distancia al objeto; c) Ángulo de heading (IV)

Como vemos en a), el ángulo girado es aproximadamente 131°, tal como se puede apreciar en la imagen obtenida mediante el sonar (ver Figura 95).

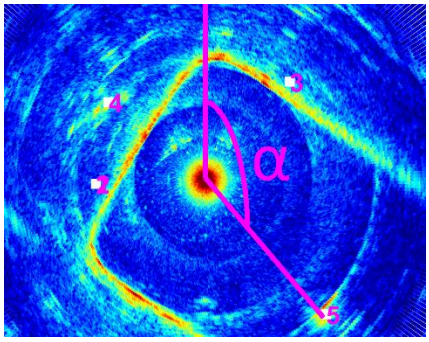


Fig. 95. Ángulo girado

Cuando se ha orientado adecuadamente, el ángulo de heading permanece prácticamente constante. A continuación, el ROV muestrea un sector menor mientras se acerca al objeto.

En la cuanto a la distancia con respecto al objeto (b), como vemos, mientras el vehículo gira no se actualiza el valor de la distancia. Al completar el giro, se actualiza el valor. Mientras avanza, la distancia disminuye hasta que se alcanza la posición deseada.

De forma similar, el ángulo con respecto al objeto (c), tampoco se actualiza mientras el vehículo gira. Al completar el giro, este ángulo es prácticamente cero, y continúa en este valor mientras el ROV avanza.

Una vez alcanzada la posición deseada, el ROV continúa muestreando un sector, tal como ilustra la Figura 96:

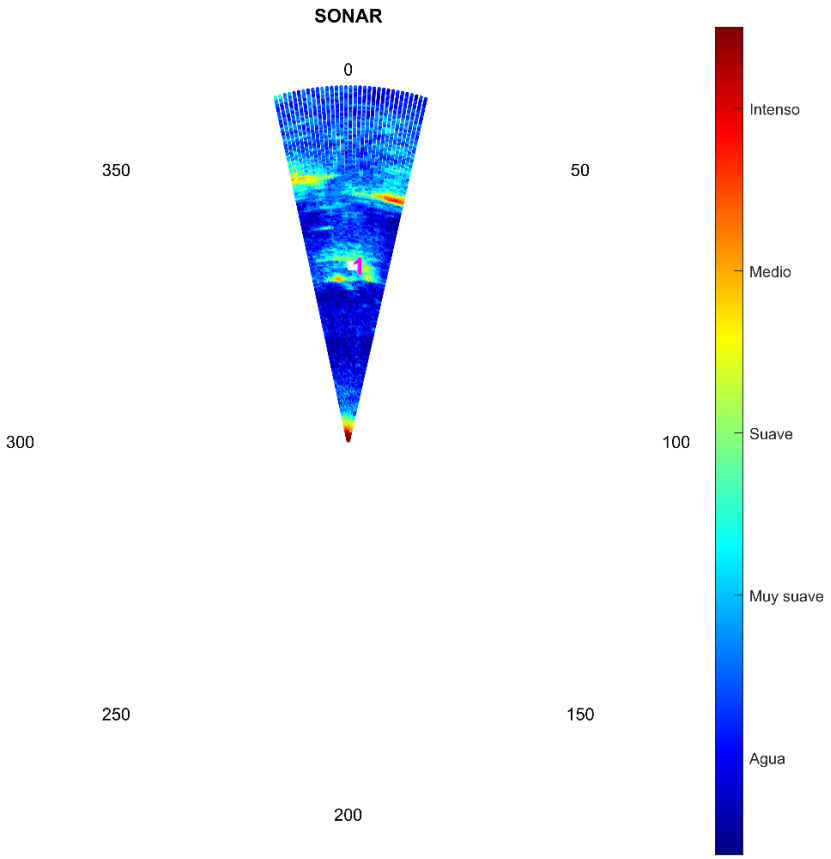


Fig. 96. Muestreo de 30°

De esta forma, si alejamos el objeto del vehículo o modificamos ligeramente el ángulo de este (para que siga apareciendo en el sector muestreado), el ROV volvería a orientarse o acercarse al objeto identificado de manera automática.

Capítulo 6

Conclusiones

En este capítulo se exponen las reflexiones y resultados de este trabajo de fin de grado, así como estrategias para futuras mejoras.

6.1. Resultados

La aplicación de nuevas tecnologías es un factor clave para lograr un correcto aprovechamiento y una explotación sostenible de los recursos pesqueros y acuícolas. En este sentido, la implantación de tecnologías 4.0, como la robótica, serán determinantes en la gestión y desarrollo de estos sectores.

A lo largo de este documento, se expone un procedimiento para hacer posible que un ROV comercial, destinado a la toma de datos e inspección de granja acuícolas, sea convertido en un ROV Hibrido capaz de localizar elementos de interés para la explotación de la granja, y dirigirse hacia ellos de manera automática.

Estas maniobras pueden resultar de gran utilidad en tareas de exploración y mantenimiento de instalaciones acuáticas, ya que permiten al operador delegar el control manual del vehículo y prestar toda su atención a la información recibida en tiempo real por los sensores que incorpora el ROV.

Para lograr una correcta navegación, es de vital importancia conocer el ángulo de heading del vehículo. Para ello, podemos basarnos en los datos proporcionados por dos sensores: la IMU y el sonar Ping 360. Como se ha visto en el capítulo anterior, las medidas proporcionadas por la IMU del ROV carecen de la exactitud necesaria en determinadas circunstancias, por lo que fue necesario diseñar un algoritmo que calcule el ángulo a partir de la información recibida por el sonar.

Combinando los datos de la IMU y del Ping360 en cada situación, conseguimos desarrollar un controlador que llevara a cabo las maniobras propuestas de manera eficaz.

6.2. Trabajo futuro

Si bien este TFG logró resolver todos los problemas planteados al principio, todavía hay espacio para más mejoras.

- Medidas del ángulo de heading

Para mejorar la exactitud de los datos del ángulo de heading proporcionados por el algoritmo de fusión de sensores, sería conveniente añadir una IMU adicional externa a la PixHawk, por ejemplo, la Xsens MTi-3-DK. La ventaja de esta IMU es que incorpora un filtro de Kalman Extendido en su algoritmo, lo que reduce la carga de procesamiento de la PixHawk.

- Ubicación del ROV dentro de la instalación acuática

Para que el ROV pueda navegar de forma autónoma, resulta de gran utilidad conocer la posición y actitud del vehículo en un sistema de referencia global. De hecho, conocer la verdadera posición del vehículo supone uno de los principales problemas, ya que el sistema GPS no funciona bajo el agua y, por lo tanto, debemos emplear otro tipo de sensores.

Algunos vehículos submarinos están equipados con sistemas de navegación inerciales (INS) y sistemas DVL (Doppler Velocity Logger). Estas soluciones suelen ser demasiado costosas y de elevado tamaño para un ROV pequeño. En su defecto, los ROVs normalmente incorporan sistemas UGPS (Underwater GPS) junto con una IMU basada en sensores MEMS. Estos son, en comparación con los más sofisticados, más propensos a perturbaciones y derivas, como hemos podido comprobar. Además, el sistema UGPS que acompaña al Sibiu PRO (Water Linked), no ha funcionado correctamente en las diversas pruebas llevadas a cabo.

Por lo tanto, sería de interés investigar las causas del fallo del sistema Water Linked y tratar de buscar soluciones que mejoren su funcionamiento, así como considerar la incorporación de un sistema DVL adecuado.

Bibliografía

[1] Fossen, T.I.: A Nonlinear Unified State-Space Model for Ship Maneuvering and Control in a Seaway (2005).

[2] https://www.nidorobotics.com/sibiu-pro (Accedido en 02/2022)

[3] Feng, L., Fangchao, Q.: Research on the Hardware Structure Characteristics and EKF Filtering Algorithm of the Autopilot PixHawk (2016).

[4] https://bluerobotics.com/learn/understanding-and-using-scanning-sonars/ (Accedido en 02/2022)

[5] https://pixhawk.org/ (Accedido en 03/2022)

[6] https://www.ardusub.com/ (Accedido en 03/2022)

[7] Luo, Z., Xianbo, X., Qin, Z.: Autopilot system of remotely operated vehicle based on Ardupilot. International Conference on Intelligent Robotics and Applications (2019).

[8] Beaudoin, L., Avanthey, L., Villard, C.: Porting Ardupilot to ESP32: towards a universal open-source architecture for agile and easily replicable multi-domains mapping robots (2020).

[9] https://mavlink.io/en/ (Accedido en 03/2022)

[10] https://ardupilot.org/dev/docs/mavlink-basics.html (Accedido en 03/2022)

[11] https://mavlink.io/en/messages/common.html#messages (Accedido en 04/2022)

[12] https://docs.bluerobotics.com/ping-protocol/ (Accedido en 05/2022)

[13] Fossen, T. I.: Handbook of marine craft hydrodynamics and motion control (2011).

[14] Ribeiro, M.I.: Kalman and Extended Kalman Filters: Concept, Derivation and Properties (2004).

[15] https://se.mathworks.com/videos/sensor-fusion-part-2-fusing-a-mag-accel-and-gyro-to-estimate-orientation-1569411056638.html (Accedido en 03/2022)

[16] https://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html (Accedido en 03/2022)

[17] https://ardupilot.org/dev/docs/extended-kalman-filter.html (Accedido en 03/2022)

[18] https://www.ardusub.com/developers/full-parameter-list.html (Accedido en 02/2022)

[19] Garrocho-Cruz, A., Gómez-Bravo, F.: Informe técnico sobre el desarrollo de vehículos operados remotamente (ROVs). Obtención, análisis y representación de datos de los sensores y creación de algoritmos de control (2020).

[20] http://www.ardusub.com/introduction/hardware-options/connection-diagrams.html (Accedido en 03/2022)

[21] Gómez-Bravo, F., Garrocho-Cruz, A., Gutiérrez-Estrada, J.C., Pulido-Calvo, I., Castro-Gutiérrez, J., Peregrín-Rubio, A., López-Domínguez, S.: Localizando elementos de interés mediante vehículos operados remotamente para la explotación sostenible de granjas acuícolas (2021).

Anexo I: Código

En este anexo se muestran los códigos utilizados en el algoritmo de control diseñado.

A. Comunicación con el sonar Ping360 y muestreo de un sector determinado

```
%% Comunicación con sonar 360

global u;
u = udpport("byte", "IPV4");

distance=Distancia*1000; % En mm
ganancia=0;              % 0=low, 1=normal, 2=high
duration=11;             % 5~500 us  (en la web pone 1~1000)
freq=750;                % Don't change recommended, it is only practical to
                           use say 650kHz to 850kHz due to the narrow bandwidth
                           of the acoustic receiver.
n_samples=1200;          % (FIJO) número de datos por cada angulo
transmit=1;              % (FIJO) 0=not transmit, 1=transmit when angle
                           reached

Paso_motor_rad=pi/200;   %0.9*pi/180
SoundSpeedWater=1.435;   %en mm/us
time_to_return=distance/SoundSpeedWater*2;
time_interval=time_to_return/n_samples;      %en us
sample_period=round(time_interval/0.025);
distancia_puntos=distance/n_samples;

if sample_period<80
    sample_period=80;
    time_interval=0.025*sample_period;       %en us
    time_to_return=time_interval*n_samples;
    distance=SoundSpeedWater*(time_to_return/2); %distance en mm
    distancia_puntos=distance/n_samples;
elseif sample_period>39999
    sample_period=39999;
    time_interval=0.025*sample_period;       %en us
    time_to_return=time_interval*n_samples;
    distance=SoundSpeedWater*(time_to_return/2); %distance en mm
    distancia_puntos=distance/n_samples;
end


[mensaje2601,  sum_parcial]=CrearMensajeSonar360_TR(duration,  sample_period,
freq, n_samples, ganancia, transmit);

figure(f1);

%% Muestrear un sector

i=1:n_samples;
```

```
Matriz_Datos=[];
tstart=tic;

Sector_1=Sector*400/360;

ang_min=round((180-Sector/2)/0.9);
ang_max=round((180+Sector/2)/0.9);

for HeadAngle=ang_min:ang_max

    mensaje2601(12)=fix(HeadAngle/256);
    mensaje2601(11)=HeadAngle-256*mensaje2601(12);

    checksum=sum_parcial+mensaje2601(11)+mensaje2601(12);
    mensaje2601(24)=fix(checksum/256);
    mensaje2601(23)=checksum-256*mensaje2601(24);

    write(u, mensaje2601, "uint8", "192.168.2.2", 9092);
    Time=toc(tstart);

    while u.NumBytesAvailable~=1224
        espera=toc(tstart)-Time;
        if espera > 3
            aux=1;
            if u.NumBytesAvailable>0
                a=read(u,u.NumBytesAvailable, 'uint8');
            end
            disp("Mensaje erróneo");
            write(u, mensaje2601, "uint8", "192.168.2.2", 9092);
            Time=toc(tstart);
        end
    end

    Mensaje=read(u,1224,"uint8");
    Matriz_Datos=[Matriz_Datos;[Time HeadAngle Mensaje]];

    [m,~]=size(Matriz_Datos);

    if HeadAngle==49 || HeadAngle==99 || HeadAngle==149 || HeadAngle==199 ||
       HeadAngle==249 || HeadAngle==299 || HeadAngle==349 ||HeadAngle==399

      [x,y,c]=Representacion_PV_TR_2(Paso_motor_rad,distancia_puntos,i,Matri
      z_Datos,HeadAngle,m);
       prueba(HeadAngle+1)=scatter(x,y,50,'black','.');

    else

      [x,y,c]=Representacion_PV_TR_2(Paso_motor_rad,distancia_puntos,i,Matri
      z_Datos,HeadAngle,m);
       prueba(HeadAngle+1)=scatter(x,y,50,c,'.');

    end
end
```

B. Filtrar los datos, identificar la pared y calcular la distancia y el ángulo con respecto al ROV

```
datos=Matriz_Datos(1:Sector_1, :);

sample_period=round(time_interval/0.025);
time_interval=0.025*sample_period; %en us (time_interval=25ns*sample_period)
time_to_return=time_interval*n_samples;
distance=SoundSpeedWater*(time_to_return/2); %distance en mm
distancia_puntos=distance/n_samples;

Matriz_datos_360_intensidades=[];
Posiciones_x=[];
Posiciones_y=[];

for pos=1:Sector_1
    angle=pos-Sector_1/2;
    x = distancia_puntos*i*sin(angle*Paso_motor_rad);
    Posiciones_x=[Posiciones_x ; x];
    y = distancia_puntos*i*cos(angle*Paso_motor_rad);
    Posiciones_y=[Posiciones_y ; y];
    Matriz_datos_360_intensidades=[Matriz_datos_360_intensidades; datos(pos,
    25:1224)];
end

% Identificación de la pared

datos_360_objetos=Matriz_datos_360_intensidades;
pixel_pared=0;
pared_detectada=0;

for i=1:Sector_1
    for j=lim_inferior:1200
        if datos_360_objetos(i, j) >= lim_pixel
            pixel_pared=pixel_pared+1;
        end
    end
end

% Distancia a la pared

if pixel_pared>250
    pared_detectada=1;

    %Centroide de la pared
    a=0;
    b=0;
    n=0;
    centroide_x=0;
    centroide_y=0;
    k_mean=0;
    for ang=1:Sector_1
        c=0;
        for k=lim_inferior:1200

            if datos_360_objetos(ang, k)>=lim_pixel && c==0
                a=Posiciones_x(ang, k);
```

```
            b=Posiciones_y(ang, k);
            n=n+1;
            c=1;
            centroide_x=centroide_x+a;
            centroide_y=centroide_y+b;
            k_mean=k_mean+k;
        end
    end
end

centroide_x_mean=centroide_x/n;
centroide_y_mean=centroide_y/n;
k_mean=round(k_mean/n);
distancia_pared = centroide_y_mean - 350; % en mm

%Representación del centroide
M=round(Sector_1/2);
Centroide_pared=scatter(Posiciones_x(M, k_mean),Posiciones_y(M,
k_mean),50,"magenta",'filled');
drawnow

if distancia_pared>=1000
    disp("Distancia a la pared: " + distancia_pared/1000 + " m");
else
    disp("Distancia a la pared: " + distancia_pared/10 + " cm");
end

% Ángulo con la pared

%Primer punto

a=0;
b=0;
n=0;
centroide_x=0;
centroide_y=0;
k_mean=0;
p1=0;
for ang=1:5
    c=0;
    for k=lim_inferior:1200
        if datos_360_objetos(ang, k)>=lim_pixel && c==0
            a=Posiciones_x(ang, k);
            b=Posiciones_y(ang, k);
            n=n+1;
            c=1;
            centroide_x=centroide_x+a;
            centroide_y=centroide_y+b;
            k_mean=k_mean+k;
            p1=1;
        end
    end
end

if p1==1
    centroide_x_mean1=centroide_x/n;
    centroide_y_mean1=centroide_y/n;
    k_mean1=round(k_mean/n);
```

```
    %Representación del punto 1
    M=2;
    Centroide_pared=scatter(Posiciones_x(M, k_mean1),Posiciones_y(M,
    k_mean1),50,"white",'filled');
    drawnow
end

%Segundo punto

a=0;
b=0;
n=0;
centroide_x=0;
centroide_y=0;
k_mean=0;
p2=0;
for ang=(round(Sector_1)-5):round(Sector_1)
    c=0;
    for k=lim_inferior:1200
        if datos_360_objetos(ang, k)>=lim_pixel && c==0
            a=Posiciones_x(ang,k);
            b=Posiciones_y(ang,k);
            n=n+1;
            c=1;
            centroide_x=centroide_x+a;
            centroide_y=centroide_y+b;
            k_mean=k_mean+k;
            p2=1;
        end
    end
end
if p2==1
    centroide_x_mean2=centroide_x/n;
    centroide_y_mean2=centroide_y/n;
    k_mean2=round(k_mean/n);
    %Representación del punto 2
    M=round(Sector_1)-2;
    Centroide_pared=scatter(Posiciones_x(M, k_mean2),Posiciones_y(M,
    k_mean2),50,"white",'filled');
    drawnow
end

if p1==1 && p2==1
    %Esquina
    if (centroide_y_mean-centroide_y_mean1>30 && centroide_y_mean-
        centroide_y_mean2>30) || centroide_y_mean-centroide_y_mean1>200
        || centroide_y_mean-centroide_y_mean2>200
        disp("Esquina");
        angulo_pared=45;
    else
        error=centroide_y_mean1-centroide_y_mean2;
        if error == 0
            error=0.000000000001;
        end

        dif_x=abs(centroide_x_mean1)+centroide_x_mean2;
        angulo_pared=atand(dif_x/error);
```

```
            disp("Ángulo con respecto a la pared: " +
            round(abs(angulo_pared)) + "º");
        end
    elseif p1==1 && p2==0
        error=centroide_y_mean-centroide_y_mean1;
        if error == 0
            error=0.000000000001;
        end

        dif_x=abs(centroide_x_mean1)+abs(centroide_x_mean);
        angulo_pared=atand(dif_x/error);
        disp("Ángulo con respecto a la pared: " + round(abs(angulo_pared)) +
        "º");
    elseif p1==0 && p2==1
        error=centroide_y_mean-centroide_y_mean2;
        if error == 0
                error=0.000000000001;
        end
        dif_x=abs(centroide_x_mean)+centroide_x_mean2;
        angulo_pared=atand(dif_x/error);
        disp("Ángulo con respecto a la pared: " + round(abs(angulo_pared)) +
        "º");
    else
      disp("Ángulo con respecto a la pared: error");
      angulo_pared=-1;
    end
else
    distancia_pared=-1000;
    angulo_pared=-1;
    disp("Pared no detectada");
end
```

C. Filtrar los datos e identificar objetos

```
datos=Matriz_Datos(1:Sector_1, :);

sample_period=round(time_interval/0.025);
time_interval=0.025*sample_period; %en us (time_interval=25ns*sample_period)
time_to_return=time_interval*n_samples;
distance=SoundSpeedWater*(time_to_return/2); %distance en mm
distancia_puntos=distance/n_samples;

Matriz_datos_360_intensidades=[];
Posiciones_x=[];
Posiciones_y=[];

for pos=1:Sector_1
    angle=pos-Sector_1/2;
    x = distancia_puntos*i*sin(angle*Paso_motor_rad);
    Posiciones_x=[Posiciones_x ; x];
    y = distancia_puntos*i*cos(angle*Paso_motor_rad);
    Posiciones_y=[Posiciones_y ; y];
    Matriz_datos_360_intensidades=[Matriz_datos_360_intensidades; datos(pos,
    25:1224)];
end


% Identificación de objetos

conn=4; % conectividad vertical-horizontal (4) o en asterisco (8)
umbral_conectividad=300; % mínimo nº pixeles para ser componente
lim_cant_datos=10000; % a partir de este límite se considera objeto
distancia_limite=700; % si el objeto está a una distancia mayor que este
                        valor, se hace filtro estricto
lim_inf_intensidad=80; % por debajo de este valor se considera agua (primera
                         mitad de la muestra)
lim_min_cant_obj=150;

N_objetos=0;
datos_360_objetos=Matriz_datos_360_intensidades;
datos_360_objetos(datos_360_objetos<lim_inf_intensidad)=0;

XY_Centroids_ID=[];
ID_objetos=[];

    % Clustering

datos_360_objetos_bin(1:angle,:)=bwareaopen(datos_360_objetos(1:angle,:),
umbral_conectividad,conn); %elimina todos los conjuntos de pixeles compuestos
                             por menos de x elementos

L_r=bwlabel(datos_360_objetos_bin, conn); %etiqueta todos los pixeles
                                            distintos de cero
CC = bwconncomp(L_r, conn);
s_r=regionprops(L_r, 'centroid'); %calcula el centroide de cada grupo de
                                  pixeles, obtenidos en columnas y filas
centroids_r = cat(1, s_r.Centroid);

if CC.NumObjects>0
    for j=1:CC.NumObjects
```

```
        [pixeles,~]=size(CC.PixelIdxList{1,j});
        x_cent=distancia_puntos*centroids_r(j,1)*sin(centroids_r(j,2)*
        Paso_motor_rad);
        y_cent = distancia_puntos*centroids_r(j,1)*cos(centroids_r(j,2)*
        Paso_motor_rad);
        XY_Centroids_ID=[XY_Centroids_ID ;[x_cent y_cent
        sqrt(x_cent^2+(y_cent-12.1875)^2)]];

        if XY_Centroids_ID(j,3)>=distancia_ruido
            if pixeles>lim_cant_datos %ruido central
                datos_360_objetos(CC.PixelIdxList{1,j})=0;
            elseif pixeles>lim_min_cant_obj
                ID_objetos=[ID_objetos;j];
            else % ruido minúsculo
                datos_360_objetos(CC.PixelIdxList{1,j})=0;
            end
        end
    end
end

[m,~]=size(ID_objetos);
N_objetos=m;

Objetos.String=num2str(N_objetos,'%u');

if N_objetos>0
    for k=1:N_objetos
        x_cent = distancia_puntos*centroids_r(ID_objetos(k,1),1)*
        sin(centroids_r(ID_objetos(k,1),2)*Paso_motor_rad);
        y_cent = distancia_puntos*centroids_r(ID_objetos(k,1),1)*
        cos(centroids_r(ID_objetos(k,1),2)*Paso_motor_rad);
        Centroide_objetos(k)=scatter(x_cent,y_cent,50,'w','s','filled');
        if k==1
            text(x_cent,y_cent,'1', 'Color', [1 0 1], 'FontSize', 14,
            'FontWeight','bold');
        elseif k==2
            text(x_cent,y_cent,'2', 'Color', [1 0 1], 'FontSize', 14,
            'FontWeight','bold');
        elseif k==3
            text(x_cent,y_cent,'3', 'Color', [1 0 1], 'FontSize', 14,
            'FontWeight','bold');
        end
        drawnow
    end
end
```

D. Controladores de velocidad

- **Velocidad de avance**

```
function [v]=control_v_avance(distancia_objeto)
    kp=0.3;
    v_max=350;
    if distancia_objeto == -1000
        v=v_max;
    elseif distancia_objeto > 400
        v=kp*distancia_objeto;
        if v > v_max
            v=v_max;
        end
    elseif distancia_objeto < 200
        v=-kp*distancia_objeto;
        if v < -v_max
            v=-v_max;
        end
    else
        v=0;
    end
end
```

- **Velocidad de giro**

```
function [v_hdg]=control_v_hdg(angulo_pared)
    kp=1.25;
    v_hdg_max=90;
    if abs(angulo_pared) > 75
        kp=45;
    end
    if angulo_pared == -1
        v_hdg=0;
    elseif angulo_pared>=0
        dif = 90 - angulo_pared;
        v_hdg=kp*dif;
        if v_hdg > v_hdg_max
            v_hdg=v_hdg_max;
        end
    else
        dif = 90 + angulo_pared;
        v_hdg=-kp*dif;
        if v_hdg < -v_hdg_max
            v_hdg=-v_hdg_max;
        end
    end
end
```

E. Desplazamiento entre paredes

```
% Muestreo e identificación de pared

[distancia_objeto, angulo_pared, fig1]=MuestreaFiltraDistanciaAngulo1(Sector,
Distancia, lim_inferior, fig1, lim_pixel);

% Cálculo de velocidades

vel_avance=control_v_avance(distancia_objeto);
if ARM == true && abs(angulo_pared) > 80
    vel = vel_avance;
elseif ARM == true && abs(angulo_pared) < 80
    vel = vel_avance/1.5;
else
    vel = 0;
end

v_hdg=control_v_hdg(angulo_pared);
if ARM == true && vel<300
    vel_hdg = v_hdg/1.15;
    deriva=-20;
elseif vel>=300
    vel_hdg = v_hdg/20;
    deriva=-20;
else
    vel_hdg = 0;
    deriva=0;
end

hdg=vel_hdg+deriva;

if ARM==true && distancia_objeto ~= -1000 && vel<180 && abs(angulo_pared)>=83
    girar = true;
elseif distancia_objeto < 350 && abs(angulo_pared) >= 85
    girar = true;
end

manual_control.Payload.x(:) = vel;
manual_control.Payload.y(:) = 0;
manual_control.Payload.z(:) = 500;
manual_control.Payload.r(:)= hdg;
manual_control.Payload.buttons(:)= 64;
sendudpmsg(gcsNode,manual_control,"192.168.2.2",MavLinkPort);

disp("Velocidad: " + vel);
disp("Velocidad hdg: " + hdg);
disp("-------------------");

% Girar 180º

if girar == true

    ARM=true;
    girar=false;
    [~,n]=size(Datos_GlobalPositionINT.hdg);
    hdg_actual=(Datos_GlobalPositionINT.hdg(1,n));
```

```
    hdg_final=hdg_actual+18000;


    if hdg_final > 36000
        hdg_final=hdg_final-36000;
    end

    k=0;

    while abs(hdg_final-hdg_actual) > 500 && salir==0 && k<5
        if abs(hdg_final-hdg_actual) < 500
            k=k+1;
        end
        [axes, buttons, povs] = read(joy);
        if buttons(1)==1
            salir=1;
        end

        dif=hdg_final-hdg_actual;

        vel_giro=control_v_giro(dif);

        manual_control.Payload.x(:) = 0;
        manual_control.Payload.y(:) = 0;
        manual_control.Payload.z(:) = 500;
        manual_control.Payload.r(:)= vel_giro;
        manual_control.Payload.buttons(:)= 64;
        sendudpmsg(gcsNode,manual_control,"192.168.2.2",MavLinkPort);

        if x>100 %Muestra datos por pantalla
            x=0;
            disp("Velocidad giro: " + vel_giro);
            disp("Diferencia: " + dif/100 + "º");
            disp("--------------------");
        end
        x=x+1;

        %leer ángulo de nuevo
        [~,n]=size(Datos_GlobalPositionINT.hdg);
        hdg_actual=(Datos_GlobalPositionINT.hdg(1,n));

    end
end
```

F. Desplazamiento hacia objeto

```
% Muestreo e identificación de objetos

[N_objetos, ID_objetos, centroids_r, distancia_puntos, fig1] =
MuestreaObjetos(Sector, Distancia, fig1);

if N_objetos == 0
    disp("Objetos no detectados");
else
    objetivo=0;
    opciones=zeros(1,N_objetos);
    for i=1:N_objetos
        opciones(i)=(ID_objetos(i, 1));
    end
    opciones_str=string(opciones);
    op_0='-';
    opciones_completo=[op_0 opciones_str];

    annotation('textbox', [0.1, 0.1, 0, 0], 'string', 'Selecciona el objeto a
    seguir:', 'FitBoxToText', 'on', 'LineStyle', 'none', 'FontSize',14);
    c = uicontrol(fig1,'Style','popupmenu');
    c.Position = [350 43 70 30];
    c.String = opciones_completo;
    drawnow;

    % Selección del objetivo

    while objetivo==0
        drawnow;
        if c.Value>1
            objetivo=c.Value-1;
        end
    end

    % Cálculo de la distancia y ángulo al objetivo

    x_cent = distancia_puntos*centroids_r(ID_objetos(objetivo,1),1)*
    sin(centroids_r(ID_objetos(objetivo,1),2)*Paso_motor_rad);
    y_cent = distancia_puntos*centroids_r(ID_objetos(objetivo,1),1)*
    cos(centroids_r(ID_objetos(objetivo,1),2)*Paso_motor_rad);

    if x_cent>0 && y_cent>0
        alpha=(90-atand(y_cent/x_cent))*100;
    elseif x_cent<0 && y_cent>0
        alpha=(-atand(y_cent/x_cent)-90)*100;
    elseif x_cent<0 && y_cent<0
        alpha=(-atand(y_cent/x_cent)-90)*100;
    elseif x_cent>0 && y_cent<0
        alpha=-1*(atand(y_cent/x_cent)-90)*100;
    end

    % Esperamos a que se arme el vehículo
    while ARM == false && salir==0
        [axes, buttons, povs] = read(joy);
        if buttons(8)==1
            ARM=true;
        end
```

```
    if buttons(1)==1
         salir=1;
    end
end

% Girar

[~,n]=size(Datos_GlobalPositionINT.hdg);
if n_anterior~=n
    hdg_actual=double(Datos_GlobalPositionINT.hdg(1,n));
    if abs(hdg_actual-hdg_actual_ant)>100 || hdg_actual_ant==0
        hdg_actual_ant=hdg_actual;
    end
    n_anterior=n;
end

hdg_final=hdg_actual+alpha;

if hdg_final > 36000
    hdg_final=hdg_final-36000;
elseif hdg_final<-36000
    hdg_final=hdg_final+36000;
end

dif=hdg_final-hdg_actual;
if dif<-18000
    dif=dif+36000;
end

k=0;
while abs(dif) > 1000 && salir==0 && k<5
    if abs(dif) < 1000
        k=k+1;
    end
    [axes, buttons, povs] = read(joy);
    if buttons(1)==1
        salir=1;
    end

    dif=hdg_final-hdg_actual;
    if dif<-18000
        dif=dif+36000;
    end

    kp=1/35;
    max=380;
    min=250;

    vel_giro=dif*kp;
    if abs(vel_giro)<min
        if dif>0
            vel_giro=min;
        else
            vel_giro=-min;
        end
    elseif abs(vel_giro)>max
        if dif>0
            vel_giro=max;
        else
```

```
            vel_giro=-max;
        end
    end

    manual_control.Payload.x(:) = 0;
    manual_control.Payload.y(:) = 0;
    manual_control.Payload.z(:) = 500;
    manual_control.Payload.r(:)= vel_giro;
    manual_control.Payload.buttons(:)= 64;
    sendudpmsg(gcsNode,manual_control,"192.168.2.2",MavLinkPort);

    if x>250 %Muestra datos por pantalla
        x=0;
        disp("Velocidad giro: " + vel_giro);
        disp("Diferencia: " + dif/100 + "º");
        disp("------------------");

    end
    x=x+1;

    [~,n]=size(Datos_GlobalPositionINT.hdg);
    if n_anterior~=n
        hdg_actual=double(Datos_GlobalPositionINT.hdg(1,n));


        if abs(hdg_actual-hdg_actual_ant)>100 || hdg_actual_ant==0
            hdg_actual_ant=hdg_actual;
        end
        n_anterior=n;
    end
end

% Avanzar

close all;
Sector=30;
Distancia=2;
[fig1]=crea_figuras1(Distancia);

while salir == 0 %distancia en mm

    [axes, buttons, povs] = read(joy);
    if buttons(1)==1
        salir=1;
    end

    [N_objetos, ID_objetos, centroids_r, distancia_puntos, fig1] =
    MuestreaObjetosSector(Sector, Distancia, fig1);

    if N_objetos > 0

        y_cent = distancia_puntos*centroids_r(ID_objetos(1,1),1)*
        cos(centroids_r(ID_objetos(1,1),2)*Paso_motor_rad);
        x_cent = distancia_puntos*centroids_r(ID_objetos(1,1),1)*
        sin(centroids_r(ID_objetos(1,1),2)*Paso_motor_rad);
        distancia_objeto=sqrt(x_cent^2+y_cent^2);
        if x_cent>0 && y_cent>0
            alpha=(90-atand(y_cent/x_cent));
        elseif x_cent<0 && y_cent>0
```

```
            alpha=(-atand(y_cent/x_cent)-90);
        end
        angulo_objeto=alpha;

        kp=0.7;
        v_max=300;

        if distancia_objeto > 200
            v=kp*distancia_objeto;
            if v > v_max
                v=v_max;
            end
        else
            v=0;
        end
        kp2=1/30;
        v2_max=50;
        if angulo_objeto > 5 || angulo_objeto < 5
            v_g=kp2*angulo_objeto;
            if v_g > v2_max
                v_g=v2_max;
            elseif v_g < -v2_max
                v_g=-v2_max;
            end
        else
            v_g=0;
        end
        deriva=0;
        manual_control.Payload.x(:) = v;
        manual_control.Payload.y(:) = 0;
        manual_control.Payload.z(:) = 500;
        manual_control.Payload.r(:)= v_g;
        manual_control.Payload.buttons(:)= 64; %4+64
        sendudpmsg(gcsNode,manual_control,"192.168.2.2",MavLinkPort);

        if x>0 %Muestra datos por pantalla
            x=0;
            disp("Velocidades: " + v + " / " + v_g);
            disp("Dist y ángulo: " + distancia_objeto + " / " +
            angulo_objeto + "º");
            disp("------------------");
        end
        x=x+1;
    end
end
disp("Posición alcanzada");
end
```