POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Álvaro Provencio Carralero

05 TRABAJO FIN DE GRADO

INDUSTRIALES

TRABAJO FIN DE GRADO

Control de movimiento de un robot submarino mediante Reinforcement Learning

TRABAJO FIN DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE **GRADUADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

SEPTIEMBRE **2023**

Álvaro Provencio Carralero

DIRECTOR DEL TRABAJO FIN DE GRADO:

Ramón Suárez Fernández

Claudio Rossi

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

AGRADECIMIENTOS

Por todo el apoyo que he recibido durante este proyecto quiero agradecer a mis padres, por haber estado pendientes y dispuestos a ayudar en todo momento, y a mis amigos, en especial a Laura, Carlos y Daniel, por haberse interesado en todos los avances que iba teniendo y por haberme prestado una mano en la realización de las pruebas.

RESUMEN

En los últimos años la inteligencia artificial (IA) ha empezado a ser un tema cada vez más recurrente en la sociedad hasta el punto de formar ya parte del día a día. No hay una definición exacta de IA, ya que algunas literaturas se enfocan más en procesos mentales y de razonamiento, mientras que otras se enfocan más en la conducta de las máquinas. Stuart J. Russell y Peter Norvig definen en [1] 4 sistemas principales, los que piensan como humanos, los que actúan como humanos, los que piensan racionalmente y los que actúan racionalmente, y son en los 2 primeros en los que este trabajo centrará su estudio, en concreto, en la Robótica y el Aprendizaje Automático o Machine Learning.

La robótica ya es un campo muy investigado e implementado en la sociedad, los robots forman parte prácticamente de cualquier proceso industrial, se emplean en la domótica del hogar, como el robot Rumba, o ha servido de inspiración de un gran número de películas de ciencia ficción. Sin embargo, sigue siendo una técnica con una gran línea de investigación en busca de una mayor eficiencia y capacidades.

Por otro lado, si bien el concepto de Machine Learning (ML) ha empezado a conocerse en la última década, en parte por el creciente interés en el manejo de Big Data, con el desarrollo de las redes neuronales y con el reciente lanzamiento de plataformas como ChatGPT, de la empresa OpenAI, se está volviendo una revolución en la que cada vez hay más ramas de investigación, hasta el punto en el que hablar ahora de IA es prácticamente sinónimo de Machine Learning.

En este Trabajo Fin de Grado se busca adaptar una técnica de ML llamada Reinforcement Learning (RL) a la teoría de control. Se comenzó por la documentación sobre los entrenamientos de distintos algoritmos de aprendizaje y la investigación en Deep Learning para el uso de redes neuronales artificiales.

Desde el Centro de Automática y Robótica (CAR) de la Universidad Politécnica de Madrid se propuso realizar las pruebas en un prototipo subacuático basado en los proyectos UNEXMIN y UNEXUP, en los que varios investigadores de la universidad colaboran actualmente. Este prototipo fue diseñado y construido en trabajos anteriores.

De esta forma, se adaptó un modelo en el programa MATLAB/Simulink para simular el comportamiento del robot y poder estudiar las características de varios algoritmos de aprendizaje. Además, se entrenaron agentes para usarlos en las pruebas con el robot y comprobar la eficacia del método de control.

Por último, se estudió ArduSub, el software de la maqueta, y su sistema de comunicación basado en ROS para adaptar un nuevo modelo de Simulink que usara los agentes entrenados para el control desacoplado de 2 de los movimientos del robot. Las pruebas se realizaron en un tanque en el CAR Robotics Arena usando además del controlador por RL, un controlador PID ajustado en un trabajo previo para comparar los resultados.

Las pruebas en el robot permitieron verificar la diferencia entre el entorno ideal del simulador y el entorno real, debido al hardware o perturbaciones provocadas por las pequeñas dimensiones del tanque. Se pudo comprobar un buen resultado de control en los agentes usados pese a la diferencia con el entorno en el que aprendieron, demostrando así una cierta capacidad de adaptación del agente.

Códigos UNESCO

1203.04 Inteligencia Artificial

1203.26 Simulación

3311.02 Ingeniería de control

3318.04 Servicios mineros

3319.13 Vehículos Submarinos

Palabras Clave

Vehículos Submarinos Autónomos (AUVs), Reinforcement Learning (RL), UNEXUP, Robótica submarina, Control.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SÍMBOLOS Y ABREVIATURAS

Modelado

{NED}	Sistema de referencia inercial
{B}	Sistema de referencia fijo
η	Vector de posición y orientación
v	Vector de velocidades lineales y angulares
$J(\eta)$	Matriz Jacobiano de η
M	Matriz de masas e inercias
$C(v)$	Matriz de Coriolis
$D(v)$	Matriz hidrodinámica
$g(\eta)$	Vector hidrostático
τ	Vector de fuerzas y momentos
P	Matriz de configuración de los propulsores
χ	Vector de propulsores

Abreviaturas y Acrónimos

ANN	Artificial Neural Network
AUV	Autonomous Underwater Vehicle
CAD	Computer Aided Design
CAR	Centro de Automática y Robótica
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
GDL	Grados De Libertad
GPU	Graphics Processing Unit
IA	Inteligencia Artificial
IAUVS	Intervention Autonomous Underwater Vehicle

MATLAB	MATrix LABoratory
MDP	Markov Decision Process
ML	Machine Learning
MSE	Mean Square Error
NED	Nord-East-Down
OS	Overshoot [%]
PPO	Proximal Policy Optimization
PID	Proportional, Integral, Derivative
RL	Reinforcement Learning
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
TD3	Twin-Delayed DDPG
Tr	Rise Time [s]
TRPO	Trust Region Policy Optimization
Ts	Settling Time [s]
UAV	Unmanned Aereal Vehicle
UUV	Unmanned Underwater Vehicle

1. INTRODUCCIÓN

1.1 Robots submarinos no tripulados (UUV)

El ser humano por naturaleza siempre ha buscado explorar y comprender su entorno, investigando volcanes, cuevas, zonas de gran altura o en las que han ocurrido desastres y por su puesto el fondo marino. Todos estos territorios suponen condiciones peligrosas o prácticamente inaccesibles, debido a altas presiones, temperaturas, gases, etc. Es en este punto donde entran en juego los robots exploradores. En este apartado se hará un breve repaso de la historia de los robots submarinos no tripulados.

Las primeras investigaciones se realizaron por el Grupo de Estudios e Investigaciones Marinas de Francia en 1945 con el robot operado remotamente "POODLE" [2], que constaba de un cilindro con un propulsor de una hélice en un extremo. Le siguieron avances generalmente con fines militares como, por ejemplo, las versiones del robot CURV (Cable-Controlled Underwater Recovery Vehicle), desarrollado en los años 60 por los norteamericanos, aunque también se diseñaron para la exploración de petróleo.

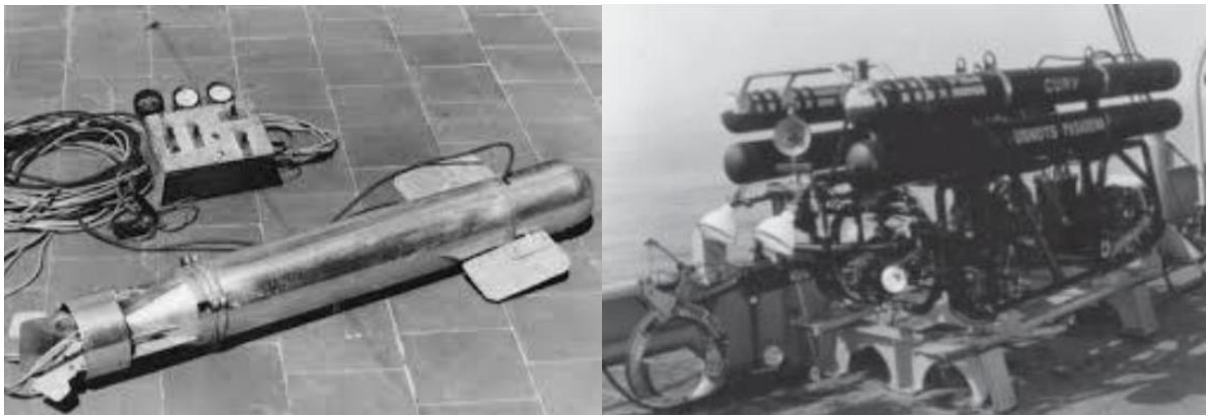

Figura 1.1: Robots POODLE a la izquierda y CURV a la derecha (Fuente: [2])

Sin embargo, el estudio de estos Vehículos Remotamente Operados (ROVs en inglés) evolucionó a comienzos de la década de los 80, principalmente por empresas dedicadas a la explotación petrolífera [3], algunos ejemplos se pueden ver en la figura 1.3. El desempeño de estos vehículos permite una mayor autonomía de operación, sobre todo los que son alimentados por un cordón umbilical, ya que las baterías se recargan constantemente desde la superficie.

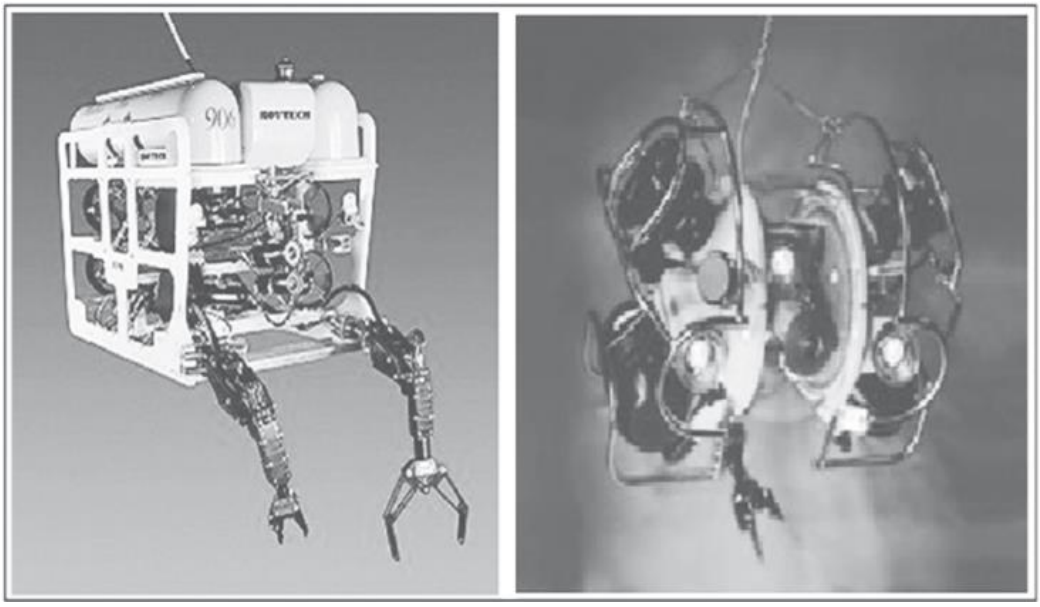

Figura 1.2: ROVs Panter Plus de la empresa SUB-FIND a la izquierda y Hyball de la empresa HYBALL OFFSHORE a la derecha (Fuente: [3])

Los ROVs sirvieron de precedente para el desarrollo de los vehículos sumergibles autónomos o AUV (Autonomous Underwater Vehicle), siendo el primero el robot SPURV, creado por la Universidad de Washington en 1957. Aunque su investigación empezó a aumentar a partir de la década de los 90 con ejemplos como el robot *Hugin*, desarrollado por una asociación internacional compuesta por *Kongsberg Maritime*, Departamento de Investigaciones de Defensa de Noruega, *Statoil*, *Nui As*, *C&C Technologies*, *Geoconsult AS*, *BP* y *Norsk Hydro*. Los objetivos de este sumergible eran tanto de investigación y control ambiental, como militares para el conteo de minas.

Figura 1.3: AUV Hugin (Fuente: [2])

Además de los 2 tipos presentados anteriormente, se deben mencionar los submarinos autónomos para intervenciones o IAUVs, considerados como un nivel intermedio de autonomía. Sin embargo, esta característica no es la única forma de clasificarlos, también se pueden agrupar en función del tipo de misión o de su propulsión [4].

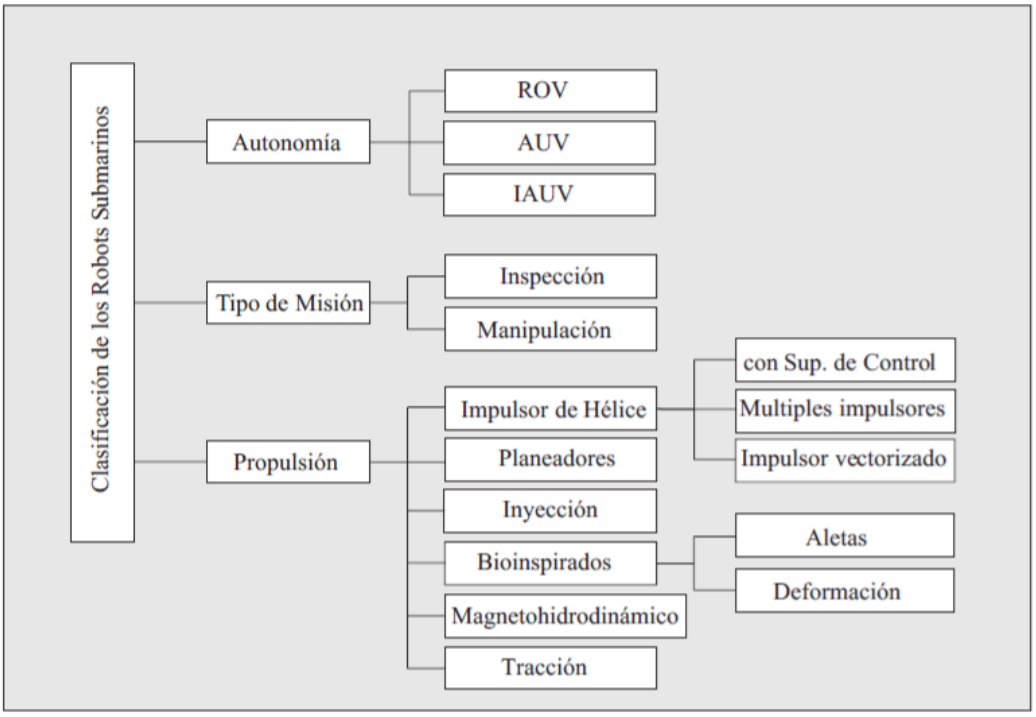

Figura 1.4: Clasificación de los robots submarinos (Fuente: [4])

Tanto los ROVs como los AUVs contienen subsistemas de sensores y comunicaciones para la recogida de información, la diferencia la marca que el segundo no necesita de un pilotaje humano, sino que tiene implementado un sistema de navegación. Sentados estos precedentes, durante los siguientes años gran variedad de empresas y universidades han ido investigando robots más eficientes, de mejor control, mayor autonomía y para aplicaciones más específicas.

Algunos ejemplos se pueden ver con el sumergible RU27 “Caballero Escarlata” [2], creado en 2009 por la universidad de Rutges, en New Jersey, que fue capaz de atravesar el Atlántico. Otro ejemplo se puede ver en avances con propulsión inspirada en los movimientos de los peces, como la tesis doctoral de Fausto Ramiro Cabrera [5], en esta misma escuela. Uno de los últimos proyectos ha sido UNEXMIN, que sirviéndose de los conocimientos ya bien asentados de los AUVs está diseñando robots exploradores de minas inundadas, y ha servido de inspiración para este trabajo.

1.2 Proyecto UNEXUP

En la actualidad hay en torno a 30.000 minas abandonadas en Europa [6], gran parte de ellas están ahora inundadas. La búsqueda de materia prima, sobre todo de nuevos materiales que en su momento no eran de tanto interés, y de nuevas tecnologías de minado, ha empujado a buscar la reapertura de estas. Sin embargo, teniendo en cuenta que la información de la que se dispone no está actualizada, junto con la complicada estructura de las galerías que supone dificultades para mantener la comunicación y numerosos riesgos para la exploración humana directa, surge la necesidad de usar vehículos autónomos, de tamaño menor y con más resistencia a este entorno.

A raíz de esta problemática nace el proyecto UNEXMIN (UNderwater EXplorer for flooded MINes). Su principal objetivo fue diseñar una flota de 3 AUVs que obtuvieran la topología de la mina sin ningún contacto o métodos que pudieran dañar la estructura. De esta forma fue diseñado el robot UX-1.

Figura 1.5: Robot UX-1 (Fuente: [6])

El primer prototipo, el UX-1 a, se desarrolló y ensambló en abril de 2018 y se probó en las cuevas Kaatiala e Idrija, en Finlandia y Eslovenia respectivamente. El robot está equipado para su navegación con varios sonares, cámaras digitales, Sistemas de Navegación Inercial (INS) y sensores de Velocidad Doppler (DVL), y para la recogida de datos, sensores de pH, conductividad eléctrica, temperatura y presión, entre otros. Por otra parte, contiene 8 propulsores, 4 a cada lado en una disposición en cruz, capaces de controlar los 6 grados de libertad del robot. Por último, tiene una autonomía de 5 horas y puede llegar hasta profundidades de 500 m.

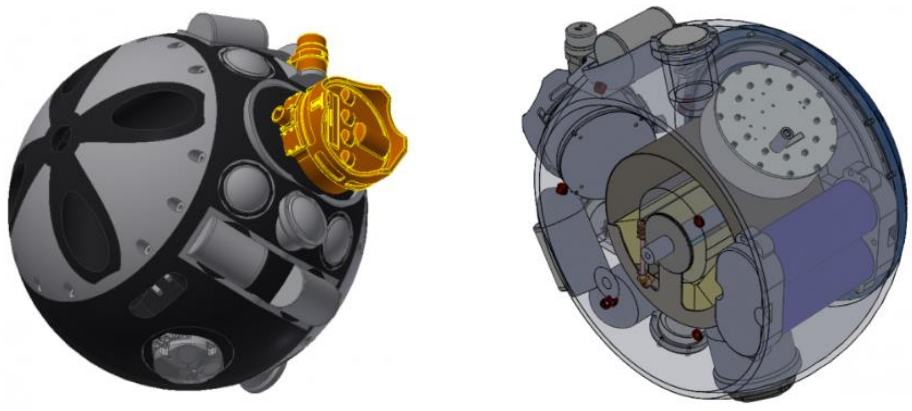

Figura 1.6: Estructura del UX-1 (Fuente: [6])

A partir de 2020 surge el proyecto UNEXUP [7] con el objetivo de lanzar al mercado la tecnología diseñada y probada en el proyecto anterior, mejorando sus capacidades, sistemas de software y hardware, y con un diseño modular. De esta forma se desarrolla el robot UX-1Neo como una versión mejorada del UX-1.

Figura 1.7: Robot UX-1Neo (Fuente: [7])

Ambos tienen las mismas dimensiones y funcionalidades, sin embargo, el UX-1Neo es más ligero y tiene un mejor hardware, software e instrumentación geocientífica. Entre sus características se aprecia una distinta distribución de los propulsores, habiendo 5 en una cara y 3 en la otra, una autonomía de 8 horas de operación y profundidad máxima de 500 m.

Por último, en 2022 se empezó a construir un nuevo robot, el UX-2, con sensores e instrumentación capaces de soportar 1 km de profundidad.

1.3 Trabajos previos

Este trabajo se propuso a raíz de los estudios realizados por los investigadores Ramón Suárez, Claudio Rossi y Sergio Domínguez en la Universidad Politécnica de Madrid. En sus trabajos [8], [9] plantean la dinámica y realizan estudios del control de los 6 movimientos en una versión simplificada del robot UX-1, colaborando con el proyecto UNEXMIN.

Asentadas las bases, les siguieron 2 trabajos adicionales por alumnos de la UPM también. El primero fue el Trabajo Fin de Grado de Miguel del Monte Martín [10], donde diseña y empieza la manufacturación de la estructura de un prototipo que replica el robot UX-1Neo y que busca compactar toda la electrónica en una cámara estanca. Además, define el modelo matemático de la estructura del robot y diseña un modelo en Simulink con una propuesta de control.

El segundo proyecto es el Trabajo Fin de Máster de Daniel Aldaz Burgoa [11], quien recogió la electrónica necesaria y se encargó del correcto funcionamiento de las conexiones y de los sistemas de comunicación utilizados, dando una descripción detallada del software y del hardware. A parte, realizó pruebas con el prototipo ya funcional ajustando un controlador PID y proponiendo un segundo controlador más complejo basado en la lógica difusa Fuzzy.

Por tanto, este proyecto pretende realizar el estudio de un segundo sistema de control complejo basado en Machine Learning, mediante la técnica de Reinforcement Learning (RL), y presentar los resultados basados en la misma maqueta para que sirva como comparativa. De esta forma, el robot aprende en función de un premio los comportamientos que ha de tener.

Por último, se están publicando numerosos artículos que implementan la técnica de Reinforcement Learning para la navegación o para evitar colisiones. Si bien se han encontrado artículos sobre el control de movimiento de UAVs, por ser los más solicitados, para conocimiento del autor, por el momento no hay investigaciones sobre el control de movimiento en UUVs que empleen dicha técnica de control.

1.4 Objetivos

En este Trabajo Fin de Grado se pretende mostrar un método de control alternativo a los convencionales y comprobar su eficacia para nuevos entornos. Los principales objetivos son:

- Comprender la robótica submarina, los avances que ha supuesto y sus líneas de investigación actuales. Además, tener contacto con un robot para conocer el hardware que usa y cómo mantener su estanqueidad.
- Documentar la técnica de Reinforcement Learning para el control de sistemas, estudiar algunos de los algoritmos de entrenamiento y aprender a entrenarlos en un entorno de Simulink.
- Estudiar ROS y su uso en el software Ardusub para la comunicación y operación de la maqueta, y diseñar un modelo de Simulink capaz de comunicar el robot con los agentes entrenados.
- Realizar una serie de pruebas en el robot que permitan comprobar la eficacia de los agentes entrenados en simulación.
- Documentar la investigación en robótica submarina y en Reinforcement Learning, y presentar los resultados de las pruebas para explicar el uso de estas tecnologías y sus aplicaciones.

1.5 Metodología

Para cumplir los objetivos del proyecto, en este apartado se explicará cómo se ha estructurado el documento y en qué consistirán los distintos capítulos.

Tras este primero de introducción, en el segundo se describirán los sistemas de referencia usados en la navegación submarina, la cinemática y la dinámica del sistema para tener un entendimiento del comportamiento del robot, que posteriormente permita implementar un entorno en el software MATLAB/Simulink para entrenar los distintos agentes.

En el tercero se presentará la maqueta proporcionada por el departamento del CAR para realizar las pruebas de control y se explicarán las principales características del hardware y software necesarias para el buen comportamiento de la maqueta.

En el capítulo 4 se dará una introducción del método de Reinforcement Learning y sus principales conceptos. Se presentarán 2 algoritmos de aprendizaje, el TD3 y el PPO, donde se analizarán sus características y formas de actuar para concluir la aptitud de control que tenga cada uno. Además, se introducirá otro controlador, el PID, para tener una comparativa durante las pruebas.

En el quinto capítulo se mostrarán los resultados obtenidos en las pruebas con el robot realizadas en 2 de sus movimientos y se hará una comparativa de ambos controladores.

Por último, se dará una conclusión del propio Trabajo Fin de Grado y se opinará sobre la disponibilidad de plantear la teoría de Reinforcement Learning para aplicaciones de control más complejas.

Tras el capítulo 6, se adjuntan la planificación y presupuesto del proyecto, la bibliografía empleada para su documentación y un anexo con el script utilizado para crear y entrenar un nuevo agente.

2. MODELADO

El comportamiento del vehículo se puede describir con un modelo matemático que permita entender el sistema en el que se va a trabajar. Es necesario definir los marcos de referencia, la cinemática y dinámica del modelo para obtener la relación entre posición y velocidad del vehículo, y controlar las fuerzas a las que está sometido.

En este caso obtener un modelo preciso es muy complicado, ya que algunos términos dependen de la estructura del robot, de la velocidad que tiene en cada momento o del propio fluido en el que se encuentra. Por tanto, con el objetivo de definir un entorno en el programa Simulink en el que se entrenen los distintos agentes, se usaron los resultados de [10], donde se define un modelo simplificado con un análisis más extenso de las ecuaciones no lineales a partir de un modelo de la maqueta diseñado en el software CAD SolidEdge.

2.1 Sistemas de referencia

Para definir los 6 grados de libertad (GDL) en la navegación submarina se necesita considerar 2 marcos de referencia: un marco de referencia inercial (NED) y un marco fijo al cuerpo (B).

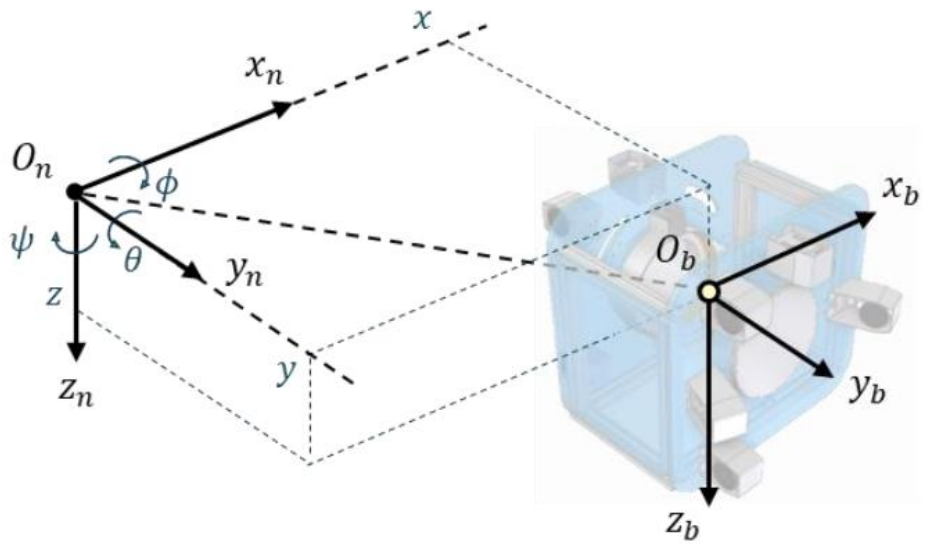

Figura 2.1: Marcos de referencia Inercial y Fijo (Fuente: [11])

El marco NED se ubica sobre la superficie de la tierra y debe su nombre a las siglas en inglés de las palabras norte-este-abajo (Nord-East-Down), de tal forma que el eje x_n apunta al norte, el eje y_n al este y el eje z_n hacia abajo, por tanto, es perpendicular a la superficie de la tierra.

El marco B suele situar su origen de referencia en el centro de gravedad del cuerpo (cuando este se encuentra en el plano principal de simetría), los ejes se eligen de forma que coincidan con los ejes principales del marco inercial, de tal forma que x_B es el eje longitudinal, y_B el eje transversal y z_B el eje dirigido hacia abajo.

A partir de estos 2 marcos se expresa el vector de posición y orientación $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T$ respecto al marco {NED} y el vector de velocidades lineales y angulares $v = [u\ v\ w\ p\ q\ r]^T$ respecto al marco fijo {B}. De esta forma se pueden definir las 3 traslaciones y las 3 rotaciones, descritas en el sistema {NED} según los ángulos de Euler roll (ϕ), pitch (θ) y yaw (ψ).

2.2 Cinemática

La ecuación cinemática relaciona los vectores de velocidades de los 2 sistemas de referencia de forma que para transformar un vector expresado en el sistema de referencia local {B} al {NED} se usan rotaciones sucesivas según los ángulos de Euler entorno a los ejes *z*, *y*, y por último *x*.

$$\dot{\eta} = J(\eta) \cdot v \qquad (1)$$

Debido a que la transformación de {B} a {NED} para las velocidades lineales es distinta al de las velocidades angulares, se define la matriz Jacobiano, J (η), la cual se puede descomponer en las 2 submatrices R(η) y W(η) para las velocidades lineales y angulares respectivamente:

$$J(\eta) = \begin{bmatrix} R_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & W_{3\times3} \end{bmatrix} \qquad (2)$$

Siendo la matriz de rotación resultante:

$$R(\eta) = \begin{bmatrix} \cos\theta \cdot \cos\psi & \sin\phi \cdot \sin\theta \cdot \cos\psi - \cos\phi \cdot \sin\psi & \cos\phi \cdot \sin\theta \cdot \cos\psi + \sin\phi \cdot \sin\psi \\ \cos\theta \cdot \sin\psi & \sin\phi \cdot \sin\theta \cdot \sin\psi + \cos\phi \cdot \cos\psi & \cos\phi \cdot \sin\theta \cdot \sin\psi - \sin\phi \cdot \cos\psi \\ -\sin\theta & \sin\phi \cdot \cos\theta & \cos\phi \cdot \cos\theta \end{bmatrix} \qquad (3)$$

Y la matriz de transformación angular:

$$W(\eta) = \begin{bmatrix} 1 & \sin\phi \cdot \tan\theta & \cos\phi \cdot \tan\theta \\ 0 & \sin\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \qquad (4)$$

Por último, se debe mencionar la aparición de una singularidad en la transformación para la posición de $\theta = \pm 90°$, aunque no suele afectar al modelado de este robot porque se diseñó para que no hiciera falta operar en esa orientación.

2.3 Dinámica

La ecuación dinámica expresa la 2ª ley de Newton, que permite relacionar los movimientos del robot con las fuerzas que se ejercen y, por tanto, en función de la acción de los propulsores tener una noción de la posición, que es la que queremos controlar. La ecuación dinámica para 6 GDL para un vehículo submarino es:

$$\mathrm{M} \cdot \dot{\mathrm{v}} + C(\mathrm{v}) \cdot \mathrm{v} + D(\mathrm{v}) + g(\eta) = \tau \quad (5)$$

Donde M es la matriz de masas e inercias, C(v) la matriz de Coriolis (y términos centrípetos), D(v) la matriz de resistencia o amortiguamiento hidrodinámico, g(η) el vector hidrostático y τ el vector de fuerzas y momentos de propulsión.

A continuación, se explicará con más detalle cada término de la expresión. Además, para restar complejidad al sistema, se asume que el origen del sistema de referencia {B} y el centro de gravedad del robot coinciden.

2.3.1 Matriz de masas e inercias

La matriz de masas e inercias se compone de la matriz de masas para un sólido rígido $\mathrm{M_{RB}}$, propia de la maqueta, y la matriz de masas añadidas $\mathrm{M_A}$, que se relaciona con el volumen de agua que rodea al prototipo y que será arrastrado en los movimientos del robot.

$$\mathrm{M} = \mathrm{M}_{RB} + \mathrm{M}_{A} \quad (6)$$

Donde $\mathrm{M_{RB}}$ y $\mathrm{M_A}$ son:

$$\mathrm{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{xy} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{xz} & -I_{yz} & I_z \end{bmatrix} \quad (7)$$

$$\mathrm{M}_A = -diag\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\} \quad (8)$$

2.3.2 Matriz de Coriolis

La matriz de Coriolis, como su nombre lo indica, describe el efecto de la fuerza de Coriolis y las fuerzas centrípetas. Asumiendo que el centro de gravedad coincide con el sistema de referencia {B}, la matriz se puede describir:

$$\mathrm{C}(v) = \begin{bmatrix} 0_{3\times3} & -S(M_{11}\cdot v_1 + M_{12}\cdot v_2) \\ -S(M_{11}\cdot v_1 + M_{12}\cdot v_2) & -S(M_{21}\cdot v_1 + M_{22}\cdot v_2) \end{bmatrix} \quad (9)$$

Donde se definen:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Y la transformación S:

$$S([a_1, a_2, a_3]^T) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \quad (10)$$

2.3.3 Matriz hidrodinámica

La matriz de resistencia o amortiguamiento hidrodinámico se debe al arrastre del fluido o por fricción superficial con este, es decir, está compuesta por las fuerzas que aparecen por la interacción del fluido con la geometría del robot al moverse dentro de él. La que se presenta es una versión simplificada y aproximada de la matriz, ya que para estos rangos de velocidades se puede considerar lineal y constante.

$$D(v) = \begin{bmatrix} X_{|u|u}|\dot{u}| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|\dot{v}| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{|w|w}|\dot{w}| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{|p|p}|\dot{p}| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{|q|q}|\dot{q}| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{|r|r}|r| \end{bmatrix} \quad (11)$$

2.3.4 Vector hidrostático

Los efectos que se han descrito anteriormente dependen del movimiento del objeto, sin embargo, este último término se corresponde a las fuerzas de flotación que son el peso del vehículo y el empuje. Se definen:

- El peso del vehículo aplicado sobre el centro de gravedad: $W = m \cdot g$
- La posición del centro de gravedad respecto al sistema {B}: $\{x_g, y_g, z_g\}$
- La fuerza de empuje aplicada sobre el centro de flotación, en función del volumen de agua desalojado: $B = \rho_{agua} \cdot g \cdot V$
- La posición del centro de flotación respecto al sistema {B}: $\{x_b, y_b, z_b\}$

Una vez se le aplica la matriz de rotación para referenciarlo en el sistema {NED}, se define el vector de fuerzas y momentos hidrostáticos:

$$g(\eta) = \begin{bmatrix} (W-B)\cdot \sin(\theta) \\ -(W-B)\cdot \cos(\theta)\cdot \sin(\phi) \\ -(W-B)\cdot \cos(\theta)\cdot \cos(\phi) \\ -\left(y_g W - y_b B\right)\cdot \cos(\theta)\cdot \cos(\phi) + \left(z_g W - z_b B\right)\cdot \cos(\theta)\cdot \sin(\phi) \\ \left(x_g W - x_b B\right)\cdot \cos(\theta)\cdot \cos(\phi) + \left(z_g W - z_b B\right)\cdot \sin(\theta) \\ \left(x_g W - x_b B\right)\cdot \cos(\theta)\cdot \sin(\phi) + \left(y_g W - y_b B\right)\cdot \sin(\theta) \end{bmatrix} \quad (12)$$

Para un mayor control de la maqueta el objetivo es conseguir un nivel de flotabilidad neutro, es decir que las fuerzas de peso y empuje sean las mismas, de esta forma los propulsores no necesitarán mantener una fuerza constante para mantener la altura. En este caso, el modelo que diseñó Miguel y el que ha servido para entrenar los agentes presentaba una masa de 20.054 Kg y un volumen de 0.02017 m^3, unos datos muy cercanos a los que se comprobaron posteriormente en la maqueta. Por último, conviene que la fuerza de empuje sea ligeramente mayor que el peso para que en caso de problema, el prototipo tienda a emerger.

2.4 Asignación de propulsores

Como se ha indicado anteriormente, este robot puede moverse en los 6 Grados de Libertad, para ello dispone de 8 propulsores dispuestos en las 2 caras laterales de la maqueta, 5 en el lado derecho y 3 en el izquierdo

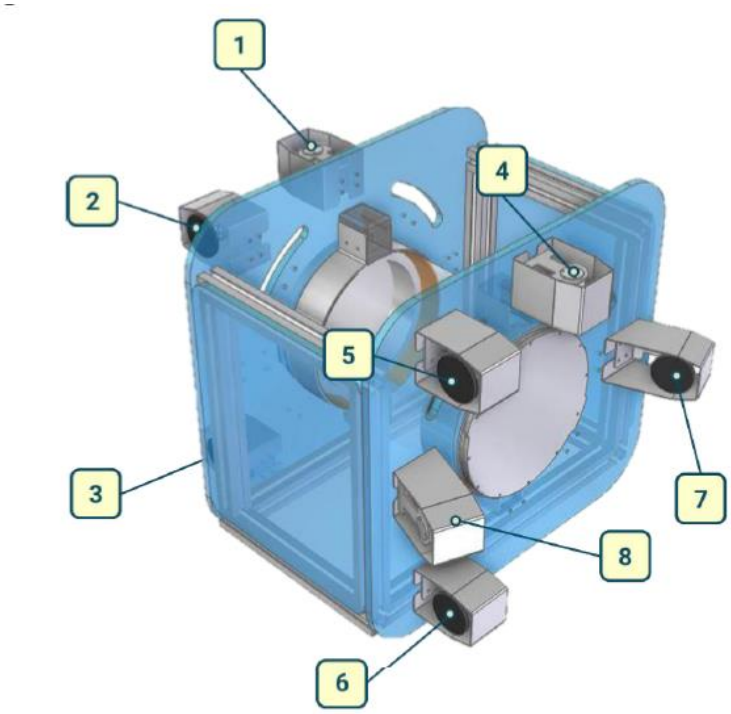

Figura 2.2: Disposición de los propulsores (Fuente: [11])

En la siguiente tabla se detalla cada tipo de movimiento, con su nombre en inglés y los propulsores que se encargan de él.

Tabla 1: Asignación de los propulsores a los 6 GDL.

Movimiento	**Eje**	**Propulsores necesarios**
Forward (x)	Traslación en el eje x	P_2, P_3, P_5, P_6
Lateral (y)	Traslación en el eje y	P_7, P_8
Throttle (z)	Traslación en el eje z	P_1, P_4
Roll (ϕ)	Rotación en torno al eje x	P_1, P_4
Pitch (θ)	Rotación en torno al eje y	P_2, P_3, P_4, P_5
Yaw (ψ)	Rotación en torno al eje z	P_7, P_8

De esta forma, se define la matriz de configuración de los propulsores P como una matriz 6x8 que servirá para relacionar las fuerzas y momentos aplicadas en cada movimiento a partir del vector de propulsores $\chi = [P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8]$.

$$\tau = \mathrm{P} \cdot \chi \tag{13}$$

La matriz P se define en función de la posición de los propulsores en el robot. Para este proyecto, la matriz resultante es:

$$\mathrm{P}=\begin{bmatrix} 0 & 0.35 & 0.35 & 0 & 0.35 & 0.35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.7 & 0.7 \\ 0.3 & 0 & 0 & -0.3 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.35 & -0.35 & 0 & 0.35 & -0.35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.3 \end{bmatrix} \tag{14}$$

3. CARACTERÍSTICAS DEL ROBOT DE PRUEBAS

Para comprobar la viabilidad de un agente entrenado tan solo en un simulador ideal, se decidió realizar las pruebas en una maqueta con los modelos que se plantearán en el siguiente apartado.

El prototipo está inspirado en el robot UX-1Neo del proyecto UNEXUP. Se empezó a diseñar y manufacturar en el trabajo de Miguel del Monte en [10] y se terminó de ensamblar, conectar la electrónica y configurar el programa de pilotaje durante el proyecto de Daniel Aldaz en [11], de tal forma que, tras unos pequeños ajustes, la maqueta era totalmente funcional para las pruebas.

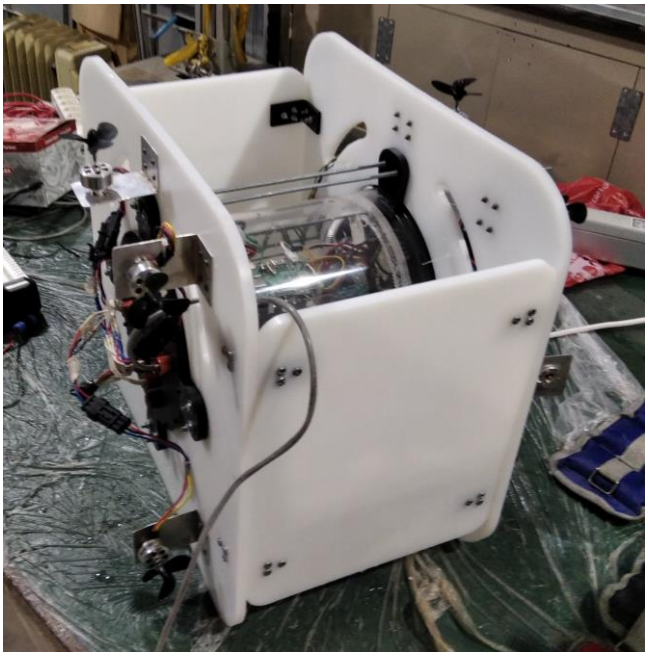

Figura 3.1*: Imagen de la maqueta (Fuente Propia)*

En este capítulo se pretende introducir los principales elementos que conforman la maqueta al igual que los programas que se encargan de su pilotaje.

3.1 Estructura y principales componentes del robot

3.1.1 Cápsula

Se trata de una capsula estanca de metacrilato comprendida entre 2 placas de HDPE (polietileno de alta densidad) de 10 mm de espesor, por tanto, la maqueta tiene una estructura cúbica. La capsula estanca pertenece a la empresa Bluerobotics [12] y sus dimensiones son 322 mm de longitud y 216 mm de diámetro. La cápsula está preparada para la robótica submarina y contendrá las baterías Li-Po, los ordenadores de placa única, los drivers de los motores y el interruptor magnético para el encendido y apagado de la maqueta.

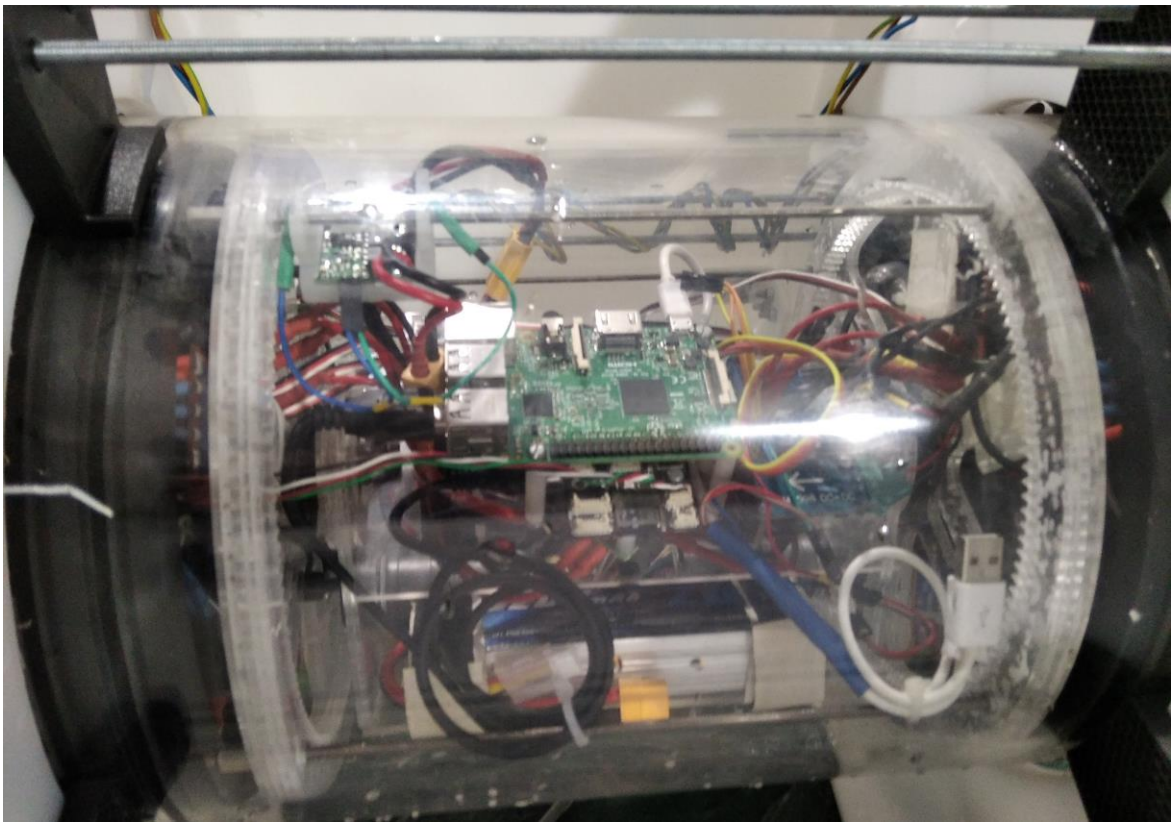

Figura 3.2: Cápsula estanca "Watertight Enclosure for ROV/AUV (8" Series)" (Fuente Propia)

3.1.2 Ordenadores de placa única

Para el software del robot se han empleado una 3DR Pixhawk 1 Flight Controller [13] y una Raspberry Pi 3 Model B [14], que se encargan de controlar el movimiento del prototipo, dar comunicación entre usuario y robot, y actuar sobre los drivers de los propulsores. En la figura 3.2 se pueden observar la placa Raspberry y la Pixhawk justo debajo.

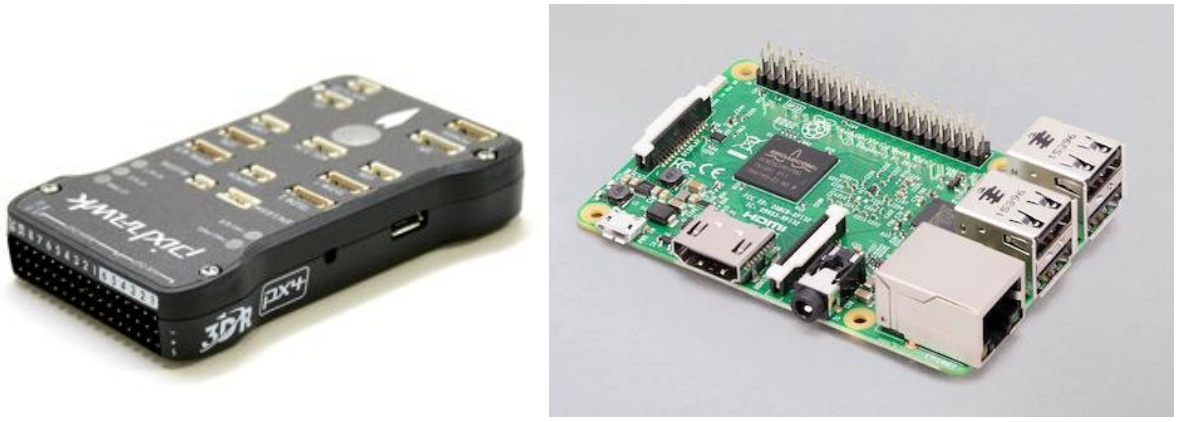

Figura 3.3: 3DR Pixhawk 1 Flight Controller (izquierda) y Raspberry Pi 3 Model B (derecha) (Fuente: [13], [14])

La Pixhawk contiene el programa de pilotaje, llamado ArduSub, actúa sobre los drivers de los propulsores y, por último, recibe las señales de presión del barómetro incorporado y las señales de la IMU de aceleraciones lineales, velocidades angulares y orientación de los giros. Por otra parte, la Raspberry se encarga de permitir la comunicación a través de ROS entre la Pixhawk y el usuario a través de un ordenador (en este caso se trata de un MSI-PX60 proporcionado por los tutores).

3.1.3 Sensor de presión

Para poder controlar en circuito cerrado los movimientos del eje z, la maqueta contiene en una de las tapas de la cámara estanca un sensor de presión Bar30 High-Resolution 300m también proporcionado por la empresa Bluerobotics, que se conectará a la IMU de la Pixhawk.

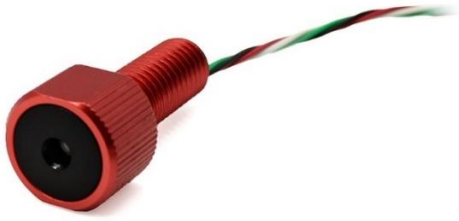

Figura 3.4: Sensor de presión Bar30 High-Resolution 300m (Fuente: [12])

A partir del barómetro se pueden conseguir las coordenadas absolutas del prototipo calibrando las posiciones iniciales y finales de las profundidades en las que se moverá la maqueta, ya que para los rangos de profundidades en los que se encuentra, se puede asumir un comportamiento lineal de la variación de presión respecto a la variación de altura.

3.1.4 Propulsores

En cuanto a los actuadores, el prototipo está capacitado de 8 propulsores, en la misma distribución que el proyecto UNEXUP, que pertenecen a la empresa Turnigy, siendo de las dos tipologías Aerodrive DST-700 y Aerodrive DST-1200. Por último, los propulsores contienen unas hélices de 3 palas, ya empleadas en modelos anteriores del proyecto. Además, se debe mencionar la necesidad de ajustar el sentido de giro de la pala en direcciones contrarias para propulsores opuestos, de tal forma que se anulen los momentos que se inducen al girar.

Figura 3.5: Propulsor Aerodrive DST-700 y hélice de 3 palas (Fuente: [15])

3.2 Sistemas software

La comunicación del robot se basa en 2 programas principales, el programa de pilotaje ArduSub, que como se ha mencionado antes, se encuentra en la Pixhawk, y MATLAB/Simulink, el cual permite aplicar la teoría de control de una forma más amigable e intuitiva desde el ordenador. En la figura 3.6 se representa la estructura de comunicación del robot diseñada por Daniel.

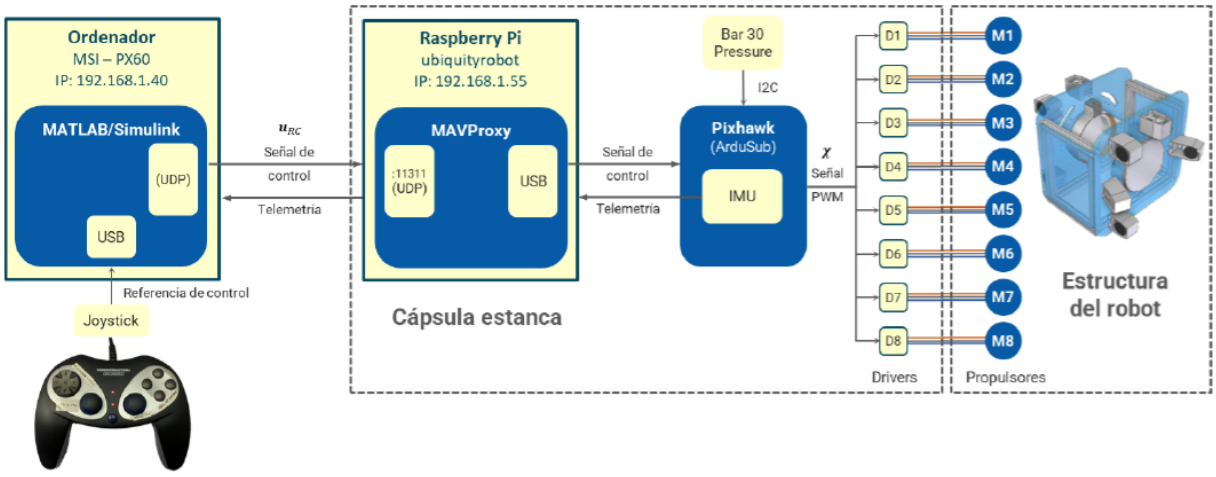

Figura 3.6: Diagrama de comunicación del sistema (Fuente: [11])

3.2.1 ArduSub

El software ArduSub forma parte de ArduPilot, el cual es un proyecto de código abierto dirigido al pilotaje de distintos vehículos no tripulados, como, por ejemplo, para UAVs (ArduCopter), aviones (ArduPlane), o en este caso el de submarinos.

La comunicación de estos sistemas se basa en ROS (Robot Operating System), una plataforma de desarrollo para sistemas robóticos. De esta forma ArduSub emplea el paquete *mavros*, que permite la utilización del protocolo MAVLink entre ordenador y el sistema de control.

La arquitectura ROS presenta 2 nodos para establecer dicha comunicación, el nodo maestro, que se encuentra en la Raspberry Pi, como se ha mencionado en el apartado anterior, y el ordenador. De esta forma, el nodo del ordenador se suscribirá al nodo maestro para recibir la información que se publique de los sensores y a su vez, el nodo maestro se suscribirá al del ordenador para recibir la acción de los propulsores enviada por el controlador.

El ordenador recibirá información de 2 tópicos de la IMU para realizar el control cerrado de los movimientos:

- Presión: Empleado para conocer la profundidad del robot y por tanto controlar el movimiento de Throttle. El tópico al que se suscribe es *mavros/imu/diff_pressure*, siendo el tipo de mensaje *sensor_msgs/FluidPressure*.
- Orientación: Proveniente directamente de la IMU para controlar el movimiento de Yaw. El tópico es *mavros/imu/data* y el tipo de mensaje *sensor_msgs/imu*.

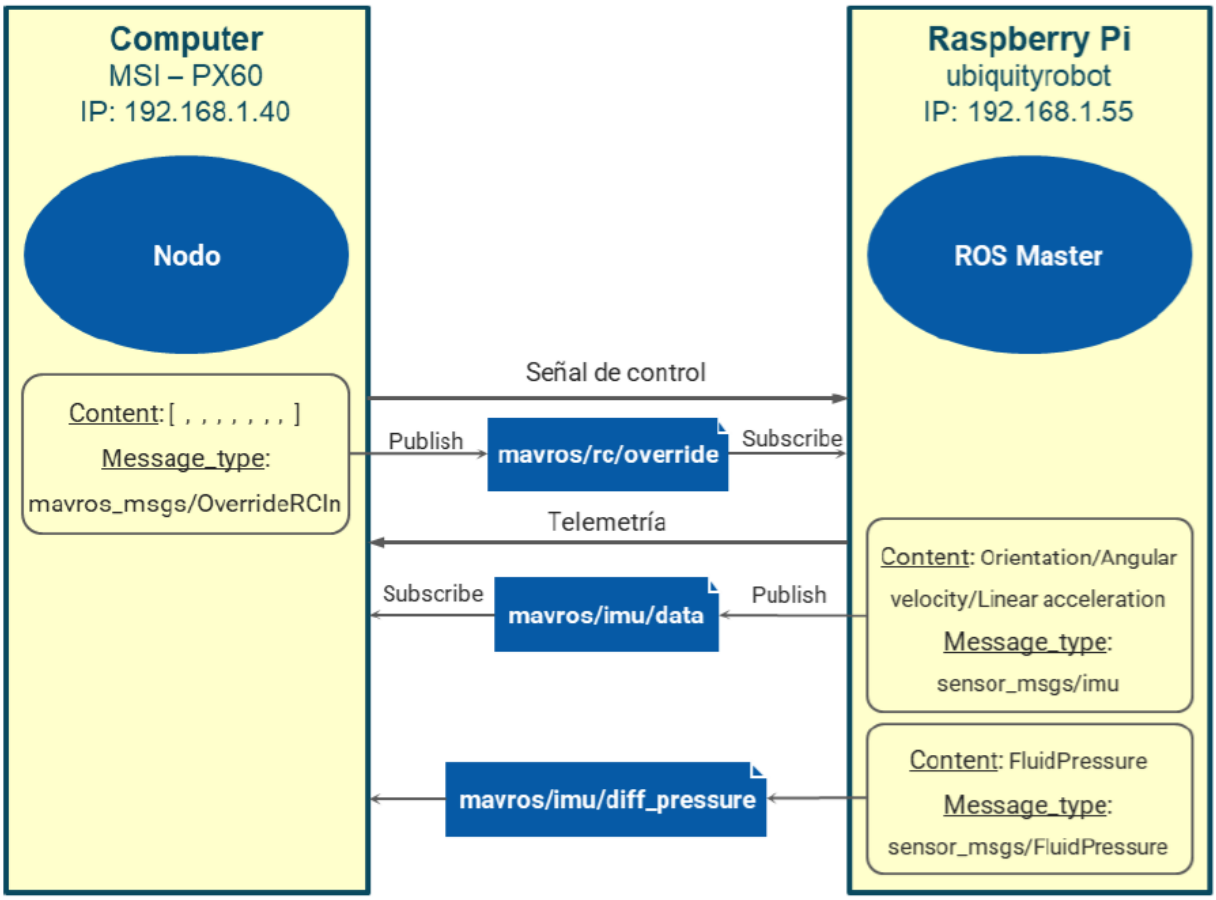

Figura 3.7: Diagrama de comunicación ROS (Fuente: [11])

Por otra parte, el nodo maestro solo recibirá la información de 1 tópico para ejercer acción sobre los propulsores. Esta señal de control se emite desde Simulink a través del tópico *mavros/rc/override* y su tipo de mensaje *mavros_msgs/OverrideRCIn*.

Este tópico contiene 8 canales RC In que representan los distintos movimientos que podría hacer el robot, aunque en este caso se definen 6, siendo el orden [Pitch, Roll, Throttle, Yaw, Forward, Lateral, − , −]. Cada canal enviará valores comprendidos entre [1000, 2000] para los comandos de control, ya que ArduSub define para el valor de 2000 su acción máxima en el sentido positivo del movimiento y en 1000 la acción máxima en el sentido contrario, siendo por tanto 1500 el valor para no actuar.

De esta forma, cada canal corresponde a la acción de control de un movimiento y mediante la matriz de configuración de propulsores P descrita en (13) ArduSub indicará los propulsores necesarios y su sentido de giro.

Cabe mencionar que, en algunas ocasiones, cuando se alternaba de un valor cercano a 2000 a otro cercano a 1000, los drivers de los propulsores dejaban de recibir órdenes y bloqueaban por tanto los propulsores. Este problema se solucionaba desarmando el servicio que permite la publicación de tópicos y volviendo a armarlo.

3.2.2 MATLAB / Simulink

El segundo software usado para el movimiento del robot ha sido MATLAB, un programa de cómputo numérico de la empresa MathWorks, referente en ingeniería. Tiene un lenguaje de programación propio (lenguaje M), aunque similar al de C/C++ y una gran variedad de paquetes especializados en la mayoría de las técnicas de ingeniería. A parte, se ha usado en especial una de sus herramientas llamada Simulink, la cual permite programar algoritmos de control según un diagrama de bloques y realizar simulaciones.

Además de usar Simulink, para el desarrollo de este proyecto se han utilizado 3 paquetes: *ROS Toolbox*, el cual permite la comunicación con el robot mediante ROS empleando los bloques de publicación y suscripción de tópicos y, por otra parte, los paquetes *Deep Learning Toolbox y Reinforcement Learning Toolbox*, los cuales permiten diseñar redes neuronales y crear un agente para su entrenamiento, respectivamente.

De esta forma, para poder comunicar con la maqueta se necesita incluir el bloque "Set Pace", que limita el tiempo de simulación de MATLAB al real indicado por el reloj de la Pixhawk y, además, referenciar en la configuración del modelo de Simulink el nodo maestro:

- Hostname: ubiquityrobot.
- Puerto: 11311.

Una vez el ordenador se suscribe el nodo maestro, Simulink puede recibir la información de los 2 tópicos de los sensores. Desde *mavros/imu/data* se pueden recibir la información de los 3 giros en cuaterniones, por lo que se añade un bloque de rotación a los ángulos de Euler. Por otra parte, desde *mavros/imu/diff_pressure* se podrá calibrar la altura en función de la señal de presión

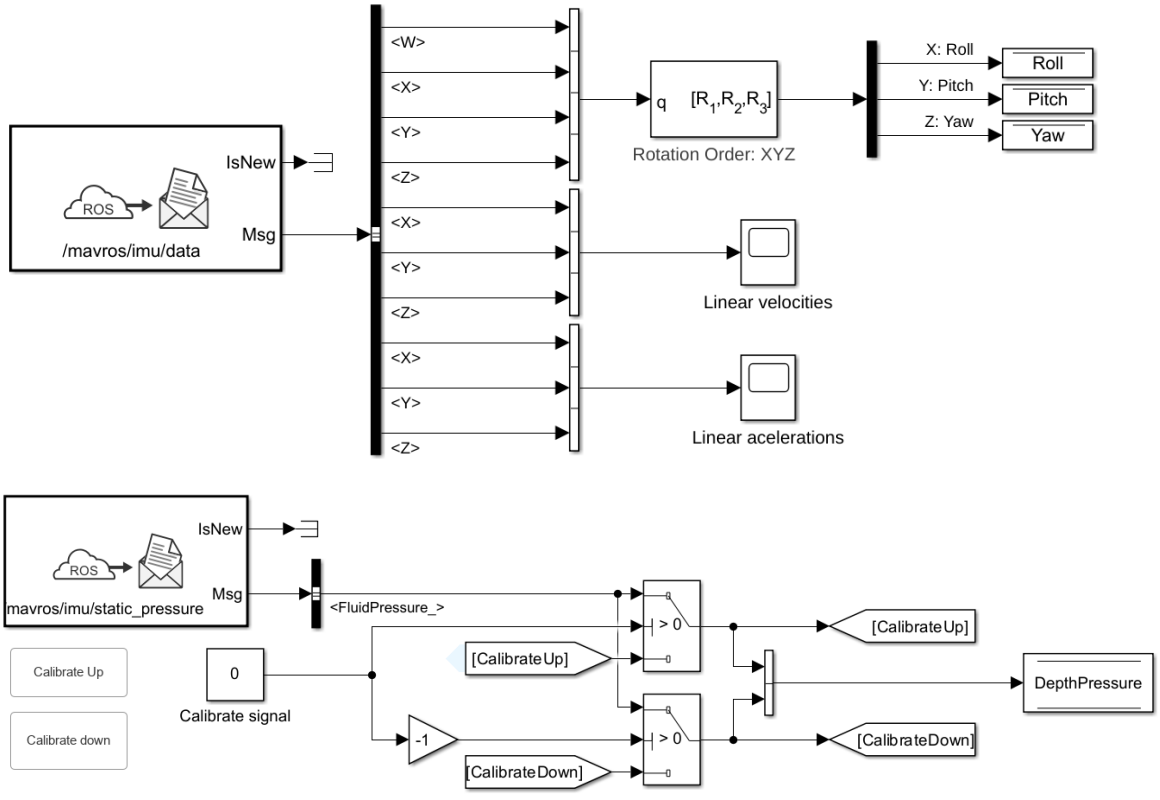

Figura 3.8: Suscripción a los tópicos del nodo maestro (Fuente propia)

Mientras tanto, se necesita usar el servicio */mavros/cmd/command* de ROS para activar la publicación de tópicos. El tipo de mensaje que utiliza es *mavros_msgs/CommandLongRequest* y el parámetro 1 es la señal booleana que se encarga de armarlo o desarmarlo. De esta forma se podrá publicar la acción de control en el tópico *mavros/rc/override*. Para evitar grandes variaciones que puedan perjudicar a los propulsores, la señal de control se limita en el intervalo [-1, 1] y posteriormente se extrapola al intervalo [1000, 2000] para enviarlo al nodo maestro.

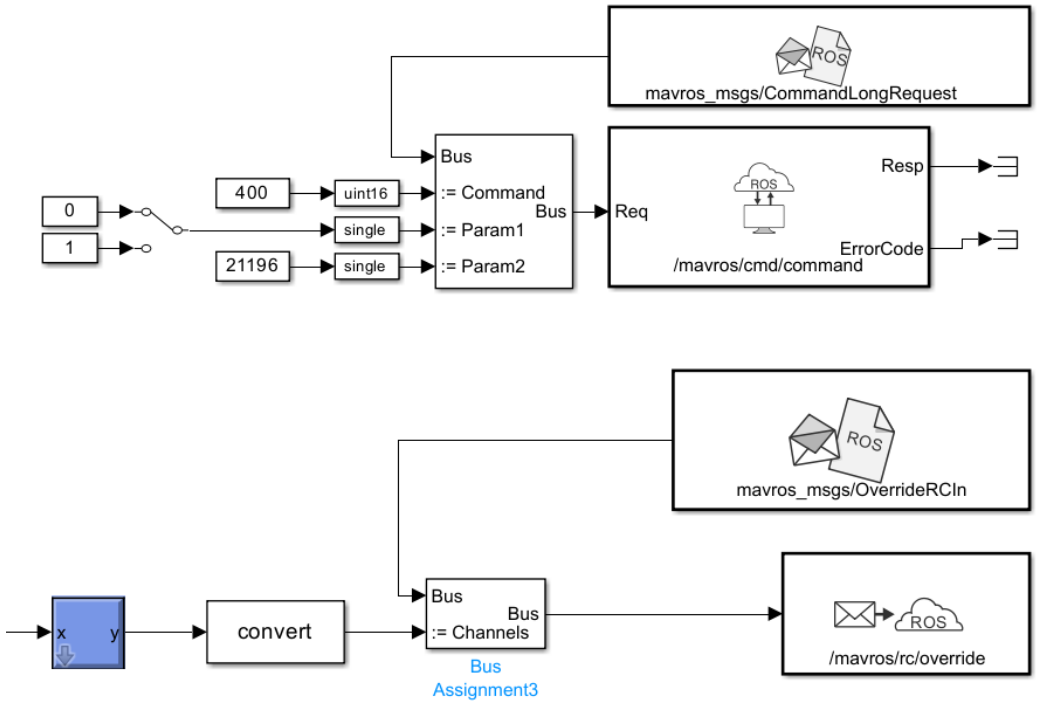

Figura 3.9: Publicación de la señal de control (Fuente propia)

Por último, se muestra el bloque del agente, que recibe como entradas las observaciones, la función de premio y una condición por si el agente debe dejar de actuar, y realiza la acción de control.

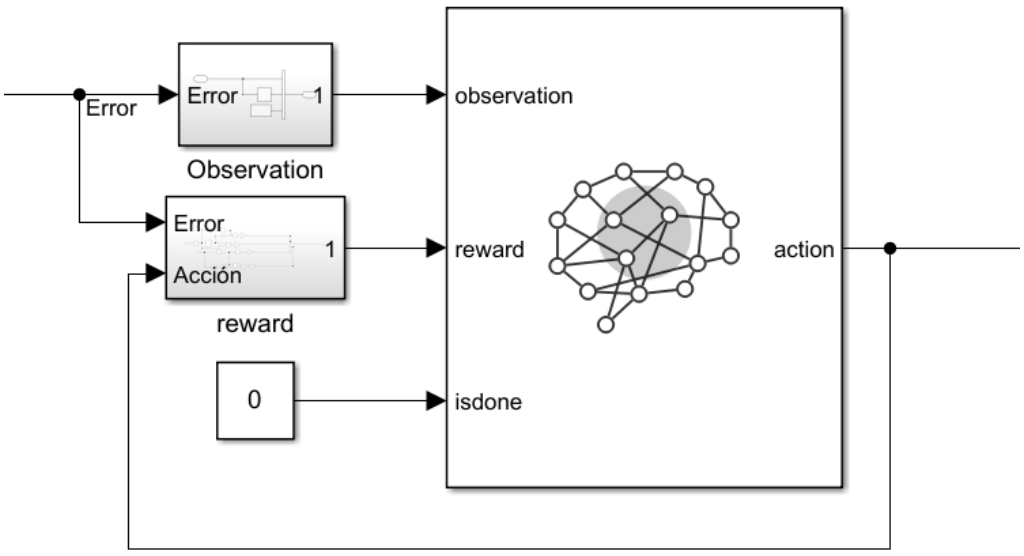

Figura 3.10: Bloque Agente (Fuente propia)

4. SISTEMAS DE CONTROL

Prácticamente cualquier entorno físico depende de ciertas variables que se necesitan mantener entre un rango de valores, ya sea por los objetivos del propio sistema o por seguridad, por lo que surge la necesidad de usar controladores. Algunos ejemplos se ven en entornos industriales para el control de presión o temperatura o, como es en este caso, en la robótica para el control de posición.

La teoría de control es muy extensa, con gran variedad de algoritmos, unos más básicos y enfocados a sistemas lineales y otros más complejos capaces de controlar sistemas no lineales. Por tanto, en este capítulo se introducirán los dos controladores que se usarán para las pruebas en la maqueta, el primero más sencillo y muy utilizado, el PID, y el segundo más reciente y menos estudiado, el basado en Reinforcement Learning.

4.1 Control por PID

Para comprender la eficacia del efecto controlador del agente seleccionado se comparará con un controlador PID, uno de los controladores más robustos y estudiados en la teoría de control y que actualmente es el más utilizado a nivel industrial por su sencillez y facilidad de uso en entornos cerrados y con pocas perturbaciones. El control PID se centra en actuar sobre el error entre la referencia establecida y el valor de la señal.

$$e(t) = y_{ref} - y(t) \quad (15)$$

Al ser el objetivo reducir el error, como se detalla en [16], el controlador le aplica una acción proporcional al propio error, una acción a su integral y una acción a su derivada. Para conseguir el efecto de control, se ajustan los valores de las ganancias K_P, K_I y K_D (ganancias proporcional, integral y derivativa respectivamente), dando a la variedad de los controladores P, PI o PD, si alguna de las ganancias es nula.

- Término proporcional: Da una acción de control general al ser proporcional al error.
- Término integral: Reduce el error estacionario por compensación de frecuencias bajas
- Termino diferencial: Reduce las oscilaciones compensando las frecuencias altas.

Tabla 2: Efectos independientes de las acciones P, I y D (Fuente: [16])

Ciclo cerrado	Tr	OS	Ts	Error estacionario	Estabilidad
Aumentar K_p	Disminución	Aumento	Pequeño Aumento	Disminución	Empeora
Aumentar K_I	Pequeña Disminución	Aumento	Aumento	Gran Disminución	Empeora
Aumentar K_D	Pequeña Disminución	Disminución	Disminución	Poco Cambio	Mejora

De esta forma, el efecto de control es la suma de las 3 acciones, siendo su función de transferencia:

$$G(s) = K_p + \frac{K_i}{s} + K_d s \qquad (16)$$

El controlador PID es muy sensible a las características del sistema, como por ejemplo la masa y pierde su capacidad de control si estas tienen una cierta variación. Esto implica que las ganancias deben ajustarse en el propio sistema, por lo que para este proyecto se usarán los PID que ajustó Daniel en [11] mediante el segundo método de Ziegler-Nichols.

4.2 Control por Reinforcement Learning

RL supone uno de los 3 principales paradigmas del Machine Learning a parte del aprendizaje supervisado y del no supervisado, sin embargo, a diferencia de los otros 2, en este caso no se necesitan datos etiquetados para el entrenamiento, sino que aprende interactuando con el entorno. RL se puede explicar como un Proceso de Decisión de Markov (MDP) [17], donde el agente aprende experimentando según escoja una acción en el entorno, provocando un cambio en el estado de este y recibiendo por tanto un determinado premio según haya sido una buena o mala elección.

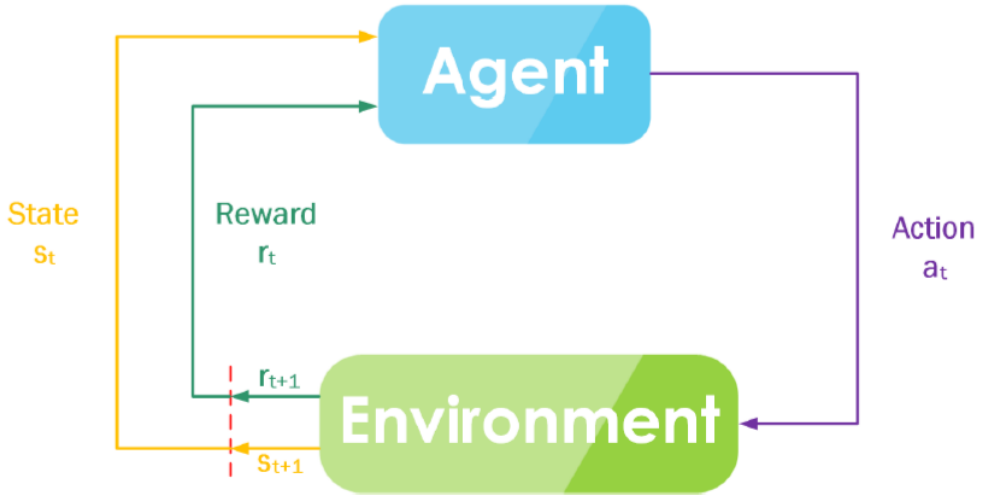

Figura 4.1: Proceso de Decisión Markov (Fuente: [17])

Se define la ecuación (17) como la probabilidad de transición del estado *s* tomando la acción *a* al nuevo estado *s'* con un premio *r* para todo $s' \in \mathcal{S}, s \in \mathcal{S}, r \in R, a \in \mathcal{A}(s)$.

$$P(s', r|s, a) = Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \qquad (17)$$

El premio *R* se genera según una función de premio, de tal forma que según van aumentando los entrenamientos, el agente conseguirá cada vez resultados mejores de la función premio tomando buenas acciones y evitando las malas.

Por tanto, se define la política $\pi(a|s)$ (o policy en inglés) como el comportamiento del agente según la probabilidad de tomar una acción *a* estando en un estado *s*, de tal forma que el agente estima su comportamiento según una función de valor. El valor producido por esta función se llama Q-value (que procede de Quality) y se expresa según la suma total de premios descontados.

El objetivo es maximizar el Q-value esperado según una política óptima mediante la experimentación con el entorno. Durante este proceso, según el ajuste de los distintos hiperparámetros del agente, este alterna entre 2 fases, exploración y explotación, siendo el primero una toma de acciones aleatoria y el segundo una toma de acciones en función de su experiencia.

Combinar RL con las Redes Neuronales artificiales (ANN) permitió resolver problemas más complejos y aumentar la investigación dentro de la inteligencia artificial, dando al Deep Reinforcement Learning. Las redes neuronales han demostrado su versatilidad de aplicaciones, dependiendo de las capas que se usen, de esta forma, su estructura puede ser muy variada según el problema y el algoritmo de aprendizaje que se utilice. Sin embargo, debido a que las ANN ajustan los pesos y sesgos de las neuronas tras cada episodio y sumado a los cálculos de la simulación, los entrenamientos requieren de una gran computación, recomendándose el uso de GPUs en vez de CPUs.

Estos algoritmos se clasifican según si el espacio de estados es limitado o ilimitado, y dentro de este, si el espacio de acciones es discreto o continuo. Además, distintos algoritmos contienen distintas estructuras para su política, sin embargo, una de las más comunes y en la que se basarán los agentes que se entrenen en este proyecto es la de Actor-Crítico [18].

En esta estructura se definen 2 redes neuronales, una que será el Crítico, que se encargará de definir la función de valor, y otra que será el Actor, que actualizará la distribución de política en la dirección sugerida por el crítico.

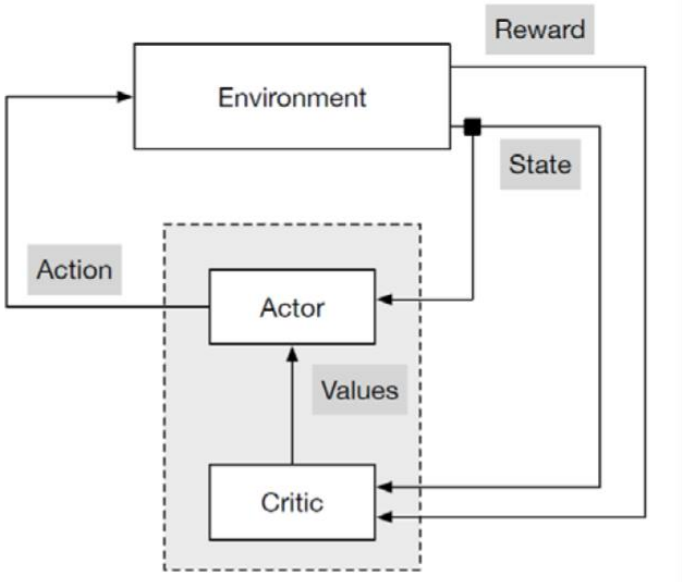

Figura 4.2: Estructura Actor-Crítico (Fuente: [18])

Como se ve en la figura 4.2 al crítico se le proporciona el estado alcanzado y el premio obtenido, y le envía el Q-value al actor, mientras tanto, el actor toma una acción en función del estado en el que se encuentra y el Q-value recibido. De esta forma, con el paso de las iteraciones ambas redes neuronales irán mejorando su comportamiento.

Las principales aplicaciones de RL se han basado en el dominio de juegos o la conducción de vehículos autónomos, y en este último caso, siendo muy empleados en el campo de los drones o UAVs para sistemas de navegación y poder evitar colisiones. Sin embargo, la versatilidad de RL le permite desenvolverse en una gran variedad de aplicaciones y en este trabajo se comprobará su capacidad de control.

4.2.1 Principales características

Una de las grandes ventajas que brinda esta técnica es que no se necesita conocer con exactitud el entorno en el que se va a implementar, ya que el propio agente irá descubriendo en su entrenamiento como afecta su comportamiento dentro de él y, por tanto, se puede aplicar en el control de sistemas no lineales sin necesidad de linealización previa, a diferencia de otros controladores óptimos, como el LQR.

Los modelos entrenados son muy sensibles a las distintas características que conforman el agente, siendo las principales las observaciones, la función de premio y el algoritmo de aprendizaje, siendo este el que dictará el comportamiento del agente y cómo acercarse al resultado deseado.

Para plantear el control de la maqueta se hizo inicialmente un estudio de estas 3 características principales, para encontrar un modelo que aprendiese el comportamiento deseado en el menor tiempo y fuera adecuado.

En cuanto a la función de premio, se busca que se cumplan 2 requisitos, el primero y principal que se reduzca el error (penalizando en función de que este aumente) y, en segundo lugar, requerir del menor esfuerzo de los motores (penalizando la propia acción del agente). Además, se añadió un término que premiase si el error era menor del 0.005 para que el agente aprendiera más rápido. La función de premio es, por tanto:

$$Premio = -10 \cdot Error^2 + 100 \cdot B - 0.1 \cdot Acción^2 \quad (18)$$

- Error: Diferencia entre la señal de referencia y la posición en la que se encuentra.
- B: Variable booleana que toma el valor de 1 si el error se encuentra en el intervalo [-0.005, 0.005].
- Acción: Acción ejercida por el agente. En el modelo se necesita añadir un retraso unitario porque si no el premio evaluaría la siguiente acción.

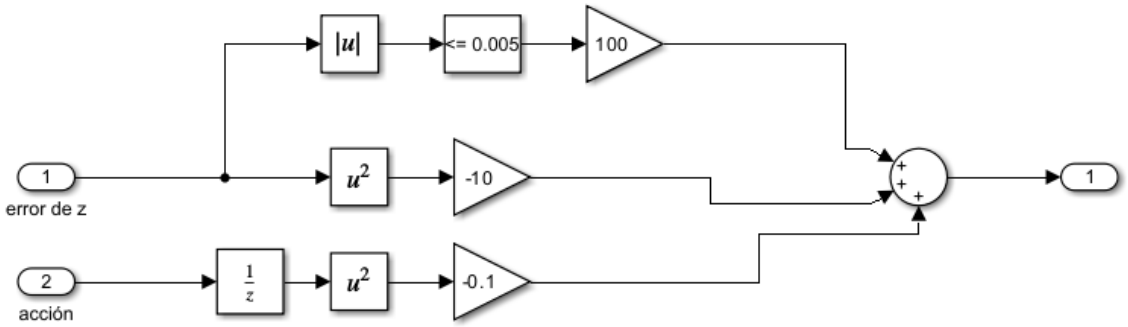

Figura 4.3: Diagrama de bloques de la función de premio en Simulink (Fuente propia)

En cuanto a las observaciones, se hicieron varios planteamientos, por ejemplo, la señal de posición junto con el error o con el MSE, aunque finalmente se encontró el mejor comportamiento si las observaciones eran el error y el efecto diferencial de este, como era de suponer según la teoría de control. Por otra parte, al ser las redes neuronales sistemas no lineales, aprendía a evitar el error en estado estacionario sin necesidad del efecto integral de este.

En cuanto a las redes neuronales, a pesar de las diferencias de las estructuras necesarias para cada algoritmo, consisten en una sucesión de las capas "Fully-Connected" (que conecta todas las neuronas entre ellas) con 64 o 32 neuronas y "ReLU" (la cual aplica la función de unidad lineal rectificada).

Por último, se buscó aprender las principales diferencias entre distintos algoritmos de entrenamiento que permitan actuar en un espacio continuo, no solo discreto, y se decidió experimentar entre 2, el TD3 y el PPO, los cuales se explicarán a continuación.

4.2.2 Algoritmo TD3

Para hablar del algoritmo TD3, conviene primero introducir el DDPG o "Deep Deterministic Policy Gradient" [19], el cual es un agente "off-policy" (Define una nueva política a partir de un grupo de resultados obtenidos por la política anterior) basado en la metodología actor-crítico que busca maximizar el premio acumulado a largo plazo. Durante el entrenamiento, las actualizaciones del actor y el crítico dan a una mejora del gradiente de la función política mediante "descenso estocástico de gradiente".

De esta forma, el algoritmo TD3 o "Twin-Delayed DDPG" [20] es una extensión del algoritmo DDPG que pretende solucionar problemas de sobreestimación de la función de valor introduciendo un segundo crítico que aprenderá otra, y usando el valor mínimo de las 2 funciones para estimar la actualización de la política. El espacio de acciones del actor es continuo y la política es determinista, es decir, para ciertos valores de observaciones, solo hay una acción.

A continuación, se presenta la gráfica del entrenamiento de uno de los agentes para el movimiento de Throttle, que se realizó en 3900 episodio y tardó aproximadamente 7 horas y 20 minutos. En ella se muestran los premios obtenidos durante cada prueba y se comprueba que el agente empieza a corregir el error entorno a la iteración 900, aunque convenía que siguiera

entrenando para mejorar su política y que el crítico optimizara su función de valor, como se ve en el crecimiento del premio.

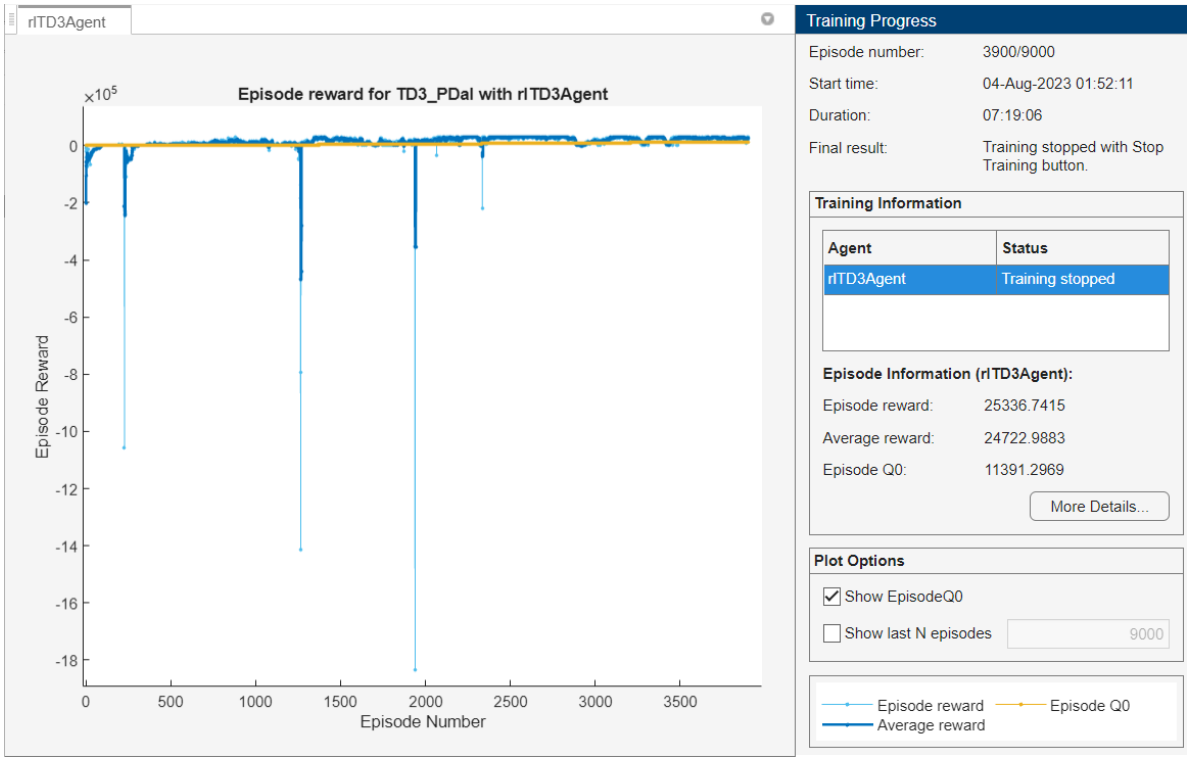

Figura 4.4: Entrenamiento del agente entrenado por TD3 en MATLAB (Fuente propia)

Se comprobó su comportamiento con una señal escalón de un metro, mostrando a continuación la señal, en la gráfica de arriba, y la acción ejercida por el agente, en la gráfica de abajo.

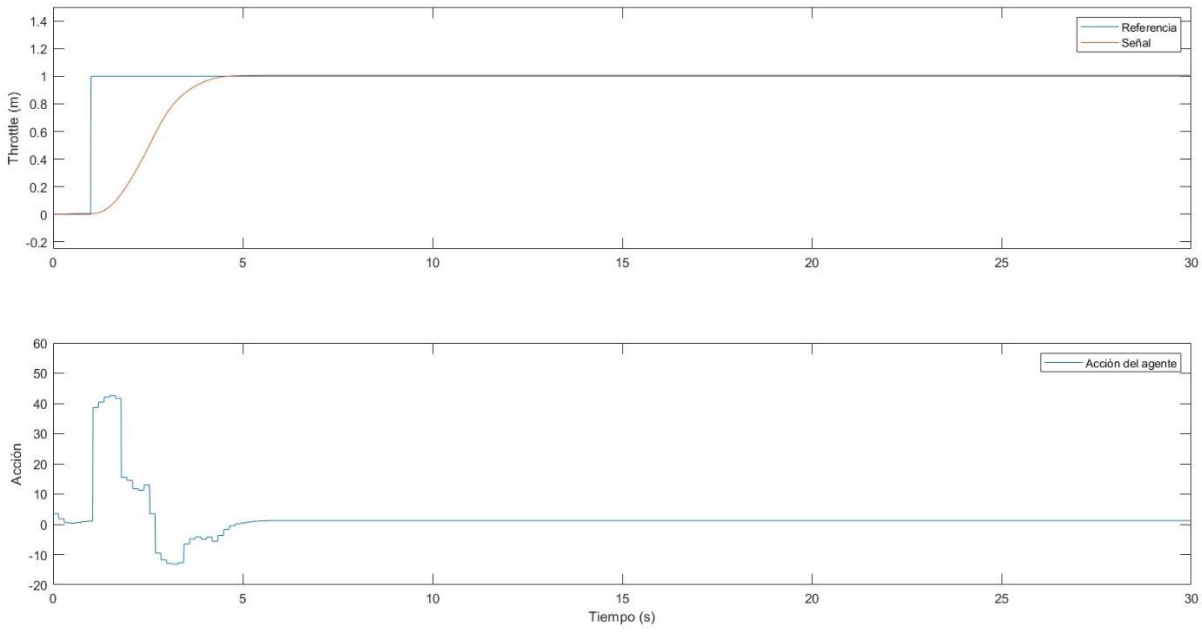

Figura 4.5: Respuesta frente a escalón de 1m del agente entrenado por TD3 (Fuente propia)

En ella podemos concluir que el agente ha aprendido a reducir el error de manera adecuada y, por tanto, se puede considerar para el control de la maqueta.

4.2.3 Algoritmo PPO

Análogamente al apartado anterior, conviene introducir primero el algoritmo TRPO o "Trust Region Policy Optimization" [21]. Este algoritmo optimiza el agente "on-policy" (actualiza la nueva política a partir de la política anterior y el Q-value del estado obtenido) alternando entre muestrear los datos a través de interacción con el entorno y actualizar los parámetros de la política. A diferencia de los agentes basados en métodos de gradiente, este actualiza la política dentro de una región de confianza cercana a la política actual.

El algoritmo PPO o "Proximal Policy Optimization" [22] es una versión simplificada del TRPO que requiere de una menor computación. El espacio de acciones en este caso es continuo, aunque PPO permite también actuar en un espacio discreto, y la política es estocástica, es decir, para unos ciertos valores de las observaciones, hay varias acciones con distintas probabilidades y el agente decide aleatoriamente una de ellas.

Al igual que en el apartado anterior, se presenta la gráfica de entrenamiento del agente para el movimiento de Throttle, el cual también hizo 3900 iteraciones, con una duración del entrenamiento de aproximadamente 1 hora y 45 minutos. Se puede apreciar como hubiera bastado con unos 500 episodios para que el agente aprendiera del entorno y maximizase la función de premio, aunque se dejó entrenar más tiempo para que siguiese explorando y para tener una mejor comparativa con el agente entrenado por TD3.

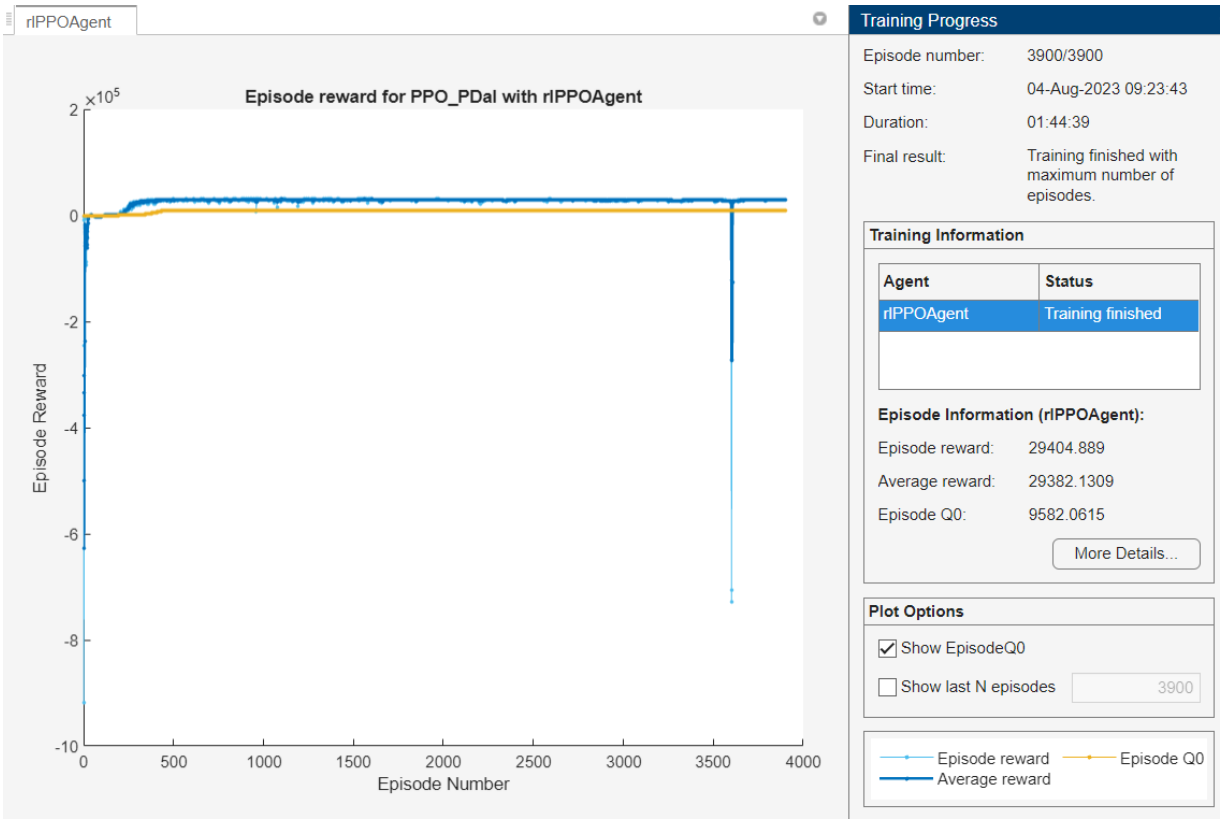

Figura 4.6: Entrenamiento del agente entrenado por PPO en MATLAB (Fuente propia)

Análogamente a la sección anterior, se adjuntan también las 2 gráficas de señal y acción, arriba y abajo respectivamente, donde se puede comprobar que también ha aprendido a reducir el error.

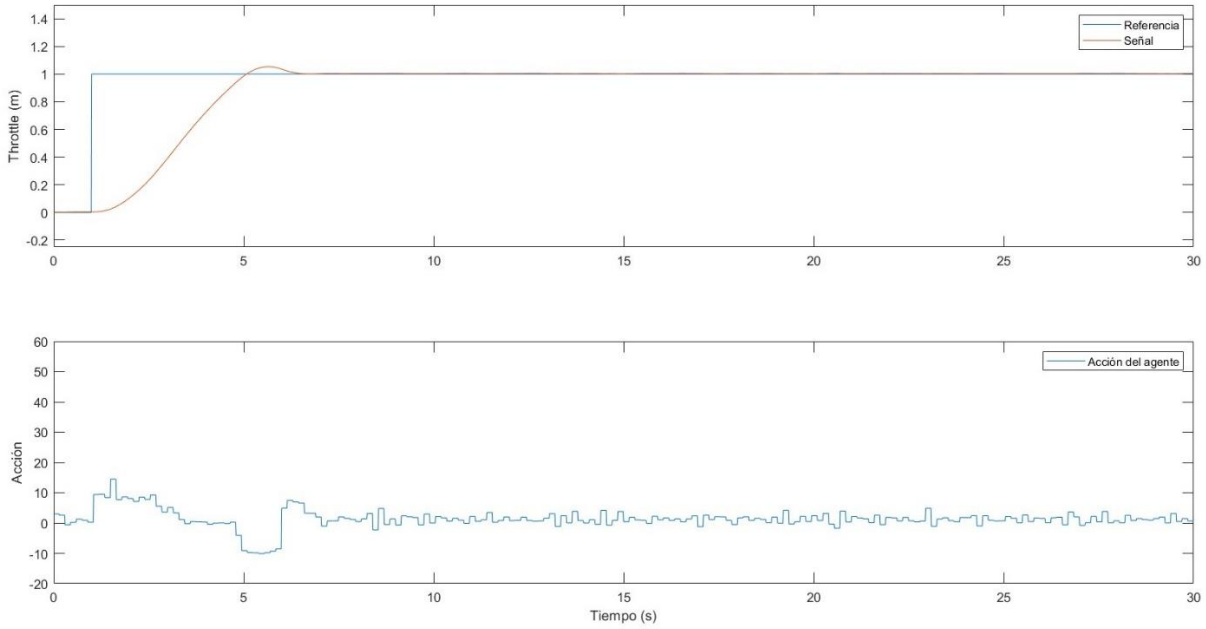

Figura 4.7: Respuesta frente a escalón de 1m del agente entrenado por PPO (Fuente propia)

4.2.4 Comparación de cada modelo

En primer lugar, cada algoritmo tiene una estructura distinta, por lo que las redes neuronales no serán las mismas. Para empezar el agente entrenado por TD3 requiere de 3 redes neuronales, 1 para el actor y 2 para los críticos, a diferencia de solo las 2 que necesita el agente entrenado por PPO. Por otra parte, cada algoritmo optimiza su política de manera distinta, siendo el primero el que requiere de más cálculos. Debido a estos dos motivos, se comprueba que el algoritmo PPO es más eficiente, por una parte, por la gran diferencia de tiempo que ha tardado el agente TD3 para los mismos episodios y en otra parte, porque en tan solo 500 ya había encontrado su máximo premio.

Este buen comportamiento del agente entrenado por PPO permitió hacer más pruebas con distintas observaciones, mejorar la función de premio y ajustar algunos hiperparámetros del entrenamiento, aunque estos también dependen del algoritmo que se use. Además, se comprobó que, para menos episodios, el agente entrenado por TD3 con la acción del error y su efecto diferencial como observaciones, presentaba cierto error estacionario (lo cual es un comportamiento inusual para las redes neuronales ya que su acción es no lineal), mientras que el modelo entrenado por PPO no llegaba a tenerlo.

Sin embargo, pese a su mayor eficacia, se comprueba comparando las figuras 4.5 y 4.7 que la acción del agente entrenado por PPO es oscilatoria y no se mantiene fija en un valor tras corregir el error. Este efecto no es conveniente para los propulsores en general, que incluso se observó más tarde en las pruebas reales, que podían quedarse bloqueados cuando se les mandaban acciones sucesivas en sentido contrario.

Tras varias pruebas variando la función de premio para que penalizara estas oscilaciones, no se llegó a una mejora, por lo que se concluye que este comportamiento se debe a que el espacio de acciones del actor sea estocástico y no pueda dar un valor fijo cuando sean las mismas observaciones.

A parte, se hicieron entrenamientos en los que la señal de referencia o la masa del sistema tomaban un rango de valores aleatorios, y se comprobó que el agente entrenado por TD3 se acababa adaptando, mientras que el agente entrenado por PPO, al no recibir un valor de premio fijo, acababa cambiando la política y perdiendo la capacidad de control. Es posible que, con un mayor tiempo de investigación y mejores recursos como el uso de una GPU, acabaran llevando a un mejor comportamiento del agente entrenado por PPO que solucionase los 2 problemas que se han mencionado, sin embargo, este objetivo se escapa del alcance de este TFG, el cual era presentar una cierta comparativa entre los 2 agentes.

De esta forma, se concluye que el algoritmo PPO es una buena opción para otros entornos que requieran de una mayor exploración, sin embargo, debido a que no cumple uno de los requerimientos del proyecto y comprobando que el agente entrenado por TD3 era capaz de controlar manteniendo la acción sin variaciones debido a su comportamiento determinista, se decidió finalmente llevar este último agente a las pruebas de los movimientos Throttle y Yaw en la maqueta.

5. PRUEBAS EN LA MAQUETA

En este último capítulo se expondrán los experimentos de control de los modelos entrenados por el algoritmo TD3 en la maqueta, como se acabó concluyendo en el apartado anterior, junto con los controladores PID. Las pruebas se hicieron en un tanque de agua de 2m x 1m x 1m en el CAR Robotics Arena del Centro de Automática y Robótica de la ETSI Industriales y se estudiaron los movimientos de Throttle y de Yaw, ya que se podía obtener una medida directa de ellos mediante un barómetro y la brújula de la IMU.

El resto de movimientos no se estudiaron ya que para las traslaciones de Forward y Lateral no se disponía de ningún sensor que pudiera medir exactamente la posición, y para los giros de Pitch y Roll porque el planteamiento del proyecto no contempla grandes variaciones, sino solo su estabilidad.

Conseguir buenos resultados en estas pruebas supuso un reto, ya que, debido a las dimensiones del tanque, al realizar varias señales de escalón el prototipo acababa yéndose cerca de las paredes, superficie o el fondo y, por tanto, recibiendo alguna perturbación debido al comportamiento del agua cerca de estas superficies.

Por otra parte, se apreció un efecto de "Deadzone" muy acentuado, de tal forma que para pequeñas acciones de control los propulsores no se movían, dando a la necesidad de añadir una acción de control constante para superar el umbral y provocando un comportamiento más oscilatorio del que supondría la propia acción de control. Este efecto provocaba que para el movimiento de Throttle, los propulsores no se movieran hasta que la acción no fuera superior al 50% de la que se proporcionaba a ArduSub, y de un 70% para el movimiento de Yaw. Esto suponía una segunda problemática, puesto que, al hacer cambios de sentido bruscos, los motores se bloqueaban y había que reiniciar el servicio que enviaba la señal de control a la IMU.

Por último, para que el agente pudiera hacer una buena acción de control se ha necesitado ajustar el tiempo de simulación de Simulink al *Sample time* con el que fue entrenado el agente, siendo este de 0.15 s, frente al predeterminado de Simulink de 0.01 s. Esto provocó que acciones integrales empezaran a dar valores cada vez más altos hasta generar un error de cálculo en la simulación y, por tanto, imposibilitando la combinación de un PID junto con el agente, por ejemplo. La solución sería reducir el "Sample time" del agente a 0.01s, lo cual requeriría de mayor computación y un mayor tiempo de entrenamiento, viéndose de nuevo la necesidad de las GPUs.

5.1 Control del movimiento Throttle

Para comparar ambos controladores se hicieron una sucesión de varios escalones comprendidos entre las profundidades de 0.1 a 0.2 m, ya que, debido a la altura del tanque, solo se podía hacer un recorrido de 23.5 cm, sin poder contar con los primeros 10 cm porque situaban los propulsores cerca de la superficie, provocando un efecto de cavitación que desestabilizaba a la maqueta.

5.1.1 Controlador PID

Los parámetros del PID de [11] son K_p: 0.35, K_i: 0.05, K_d: 0.02. Además, se usó una acción constante de 0.45 para superar el umbral de acción de los motores y que pudiera sustentar la maqueta cuando la acción de control fuera prácticamente nula, ya que en la prueba tendía a hundirse. A continuación, se presentan las gráficas de la señal y la acción de control.

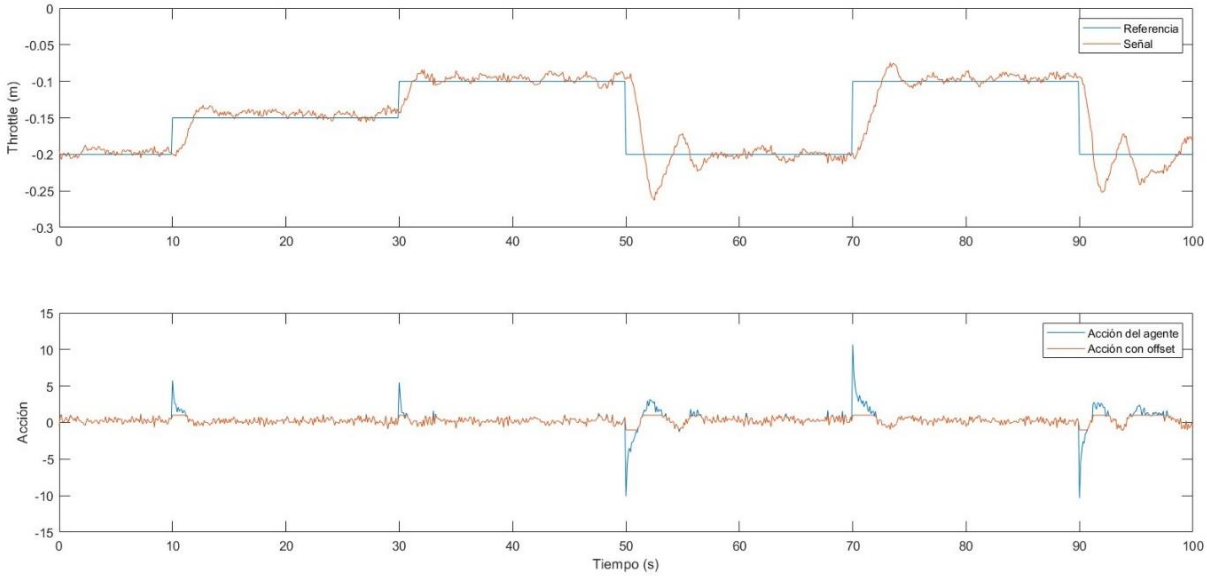

Figura 5.1: Señal del movimiento Throttle frente a una sucesión de escalones controlado por PID (Fuente propia)

En cuanto a la acción de control, se muestra en azul la señal dada por el PID y en naranja la señal saturada que recibía la IMU, limitada entre 1 y -1, donde se ve cómo en cada escalón, la acción superaba los límites de saturación, dando a una respuesta más rápida, que en ocasiones producía el bloqueo de los motores.

5.1.2 Controlador por RL

El agente que se propuso para realizar el control recibía como observación, a parte del error y su derivada, una referencia de la masa de la maqueta, porque el modelo fue entrenado en un rango entre 17 kg y 20 kg. A parte de esto, también se necesitó la acción constante para sobrepasar el umbral.

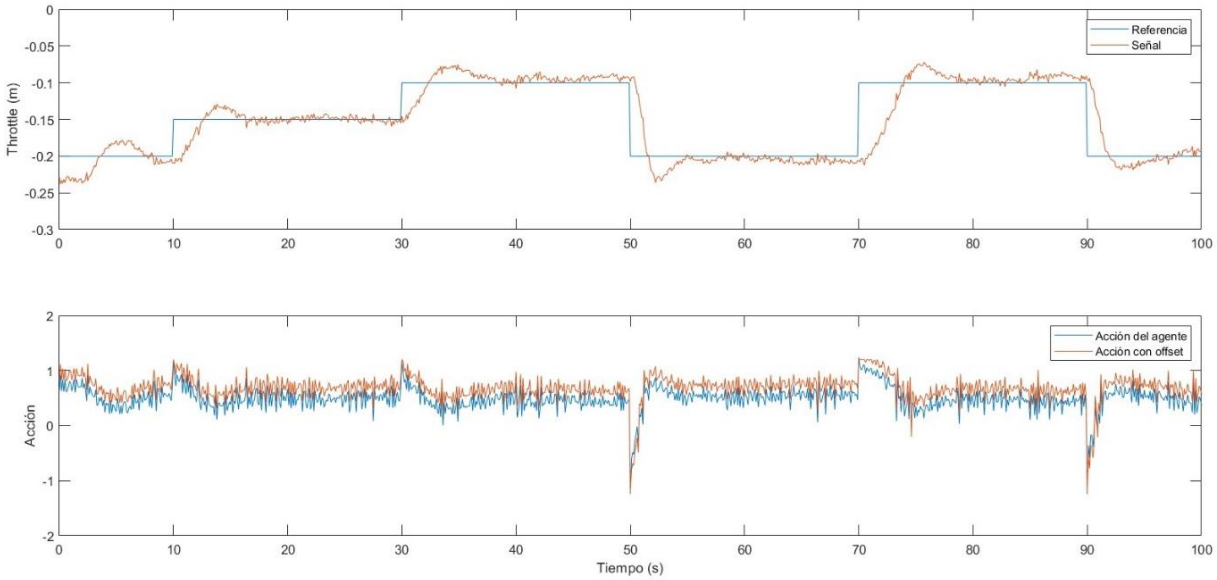

Figura 5.2: Señal del movimiento Throttle frente a una sucesión de escalones controlado por RL (Fuente propia)

En cuanto a la acción, se observa una menor que la del PID, sin necesitar prácticamente del efecto saturado.

5.2 Control del movimiento Yaw

Al igual que en el apartado anterior, se comprobaron los 2 controladores a partir de una sucesión de escalones, sin embargo, como en este movimiento no tienen efecto las componentes hidrostáticas cuando el marco de referencia {B} coincide con el {NED}, y el umbral se encontraba entorno al 70% del rango de acciones de ArduSub, en vez de añadir una acción constante, se añadió un incremento de 0,65 en el sentido de la acción del controlador.

5.2.1 Controlador PID

De esta forma, las ganancias del PID presentadas en [11] son K_p: 0.1, K_i: 0.01, K_d: 0.005 y la acción de control resultante:

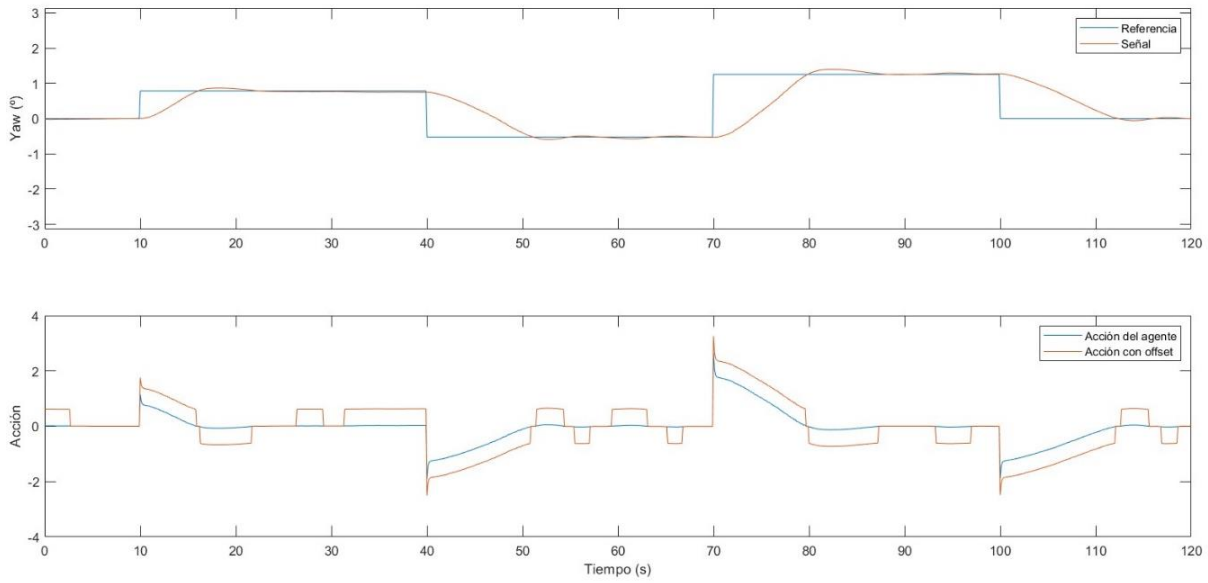

Figura 5.3: Señal del movimiento Yaw frente a una sucesión de escalones controlado por PID (Fuente propia)

5.2.2 Controlador por RL

Al igual que el agente anterior, este obtenía la misma función de premio y las mismas observaciones. El resultado del control fue.

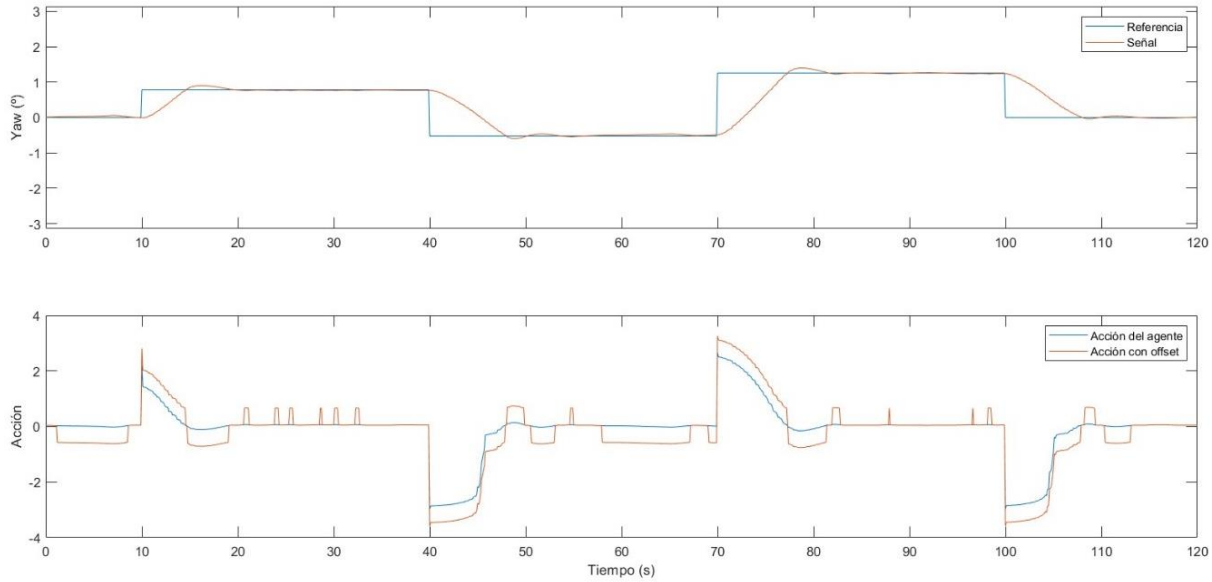

Figura 5.4: Señal del movimiento Yaw frente a una sucesión de escalones controlado por RL (Fuente propia)

5.3 Comparativa entre RL y PID

A continuación, se incluyen 2 tablas con los datos de sobreoscilación (OS), tiempo de subida (Tr) y tiempo de establecimiento (Ts) de los resultados de control mostrados en los apartados anteriores.

Tabla 3: Características de ambos controladores en el movimiento de Throttle

Escalón	1º (-0.15 m)		2º (-0.1 m)		3º (-0.2 m)		4º (-0.1 m)		5º (-0.2 m)	
Controlador	PID	RL	PID	RL	PID	RL	PID	RL	PID	RL
OS (%)	**34.82**	41.28	**31.83**	49.08	63.03	**36.07**	**25.62**	27.77	52.25	**18.83**
Tr (s)	**1.41**	1.43	**1.14**	1.77	**0.79**	0.82	**1.85**	2.4	**0.72**	1.02
Ts (s)	19.96	**6.52**	19.85	**18.92**	**19.23**	19.72	19.75	**19.61**	16.63	**9.81**

Tabla 4: Características de ambos controladores en el movimiento de Yaw

Escalón	1º (45º)		2º (-60º)		3º (72º)		4º (0º)	
Controlador	PID	RL	PID	RL	PID	RL	PID	RL
OS (%)	**10.38**	14.60	**5.20**	5.30	**7.91**	8.04	4.53	**2.93**
Tr (s)	4.13	**3.04**	8.17	**5.82**	6.71	**5.39**	8.53	**5.82**
Ts (s)	11.12	**8.74**	13.17	**9.20**	15.18	**10.32**	11.66	**7.12**

Para empezar, comparando el Tr se observa en el movimiento de Throttle en general una respuesta un poco más rápida en el PID que en el RL, aunque este consigue estabilizarse antes. Se puede apreciar, como el controlador por RL tiene una mayor sobreoscilación en los escalones ascendentes, pero una bastante menor en los descendentes, esto se debe a que se le había indicado en la observación de la masa que la maqueta se hundía ligeramente, provocando que el agente mandara una señal menor al bajar, a diferencia del PID que, al actuar directamente sobre el error, mandaba una señal a los propulsores más exagerada.

Por otra parte, en el movimiento de yaw, se comprueba generalmente una respuesta más rápida en el agente, tanto en tiempo de subida como en el de establecimiento, con unas sobreoscilaciones muy parecidas.

Por tanto, se comprueba la efectividad de los agentes entrenados como algoritmo de control, abriendo la posibilidad de entrenar el modelo en una simulación y que, a pesar de ser un entorno ideal, el agente siga manteniendo un buen comportamiento.

En este caso, el modelo no era muy fiel al entorno de las pruebas reales, puesto que la definición del centro de masas y el volumen no han tenido por qué ser exactamente los mismos al de la maqueta real, para las matrices de fuerzas se han tomado modelos ideales como el de sólido rígido o se han simplificado, sobre todo el de la fuerza hidrodinámica y, además, no se han contemplado perturbaciones como las debidas a las paredes. Sin embargo, el agente ha aprendido a reducir el error con resultados muy fiables.

Por otra parte, se ha podido comprobar que el modelo ha aprendido el comportamiento del entorno, ya que al indicar en la observación de la masa si esta era superior o inferior a la de referencia, que en el simulador era la de flotabilidad neutra, se aprecian diferencias en el control, puesto que cuando en la maqueta había más peso que empuje, el agente enviaba una acción más leve para referencias en las que tenía que hundirse que en las que subir, como se aprecia en la figura 5.4 y de forma análoga cuando esta tendía a flotar.

En comparación con el efecto de control del PID, se puede apreciar una estabilización más rápida por parte del agente y con acciones menos agresivas, ya que las del PID eran más amplias, teniendo incluso que ser saturadas. Este comportamiento de los agentes da a respuestas suaves lo cual es beneficioso para los propulsores y para el movimiento que se espera de este robot, ya que se buscan movimientos más estables que rápidos. Se concluye por tanto que el agente ha conseguido un mejor comportamiento de control y que se podría haber apreciado una respuesta menos oscilatoria si el efecto de “Deadzone” hubiera sido menor.

6. CONCLUSIÓN DEL TRABAJO

En este capítulo se darán una ultimas conclusiones sobre el método de control y se analizarán el cumplimiento de los objetivos, además, se comentarán algunos trabajos que pueden seguirle a este proyecto y las implicaciones éticas que suponen las técnicas empleadas.

6.1 Cumplimiento de Objetivos

En la documentación presentada se comprueba que los objetivos han sido cumplidos y se concluye la técnica de Reinforcement Learning como un controlador más complejo.

Cabe mencionar la gran curva de aprendizaje e investigación que ha supuesto entender la comunicación empleando ROS y la teoría de Reinforcement Learning porque no se partía de ningún conocimiento previo desde el grado. Sirvieron de gran ayuda la serie de cursillos de *Machine Learning with MATLAB, Deep Learning with MATLAB* y *Reinforcement Learning with MATLAB* [23], proporcionados por la propia empresa de MathWorks, que sirvieron para asentar las bases teóricas al mismo tiempo de aprender cómo usarlo en el programa. Además, los trabajos previos a este sirvieron de gran ayuda para comprender la dinámica del sistema y las principales características del robot.

Por otro lado, dada la versatilidad de los agentes de RL, a pesar de las numerosas pruebas realizadas durante los entrenamientos, surge la duda de si hubieran sido más eficaces usando otras observaciones, hiperparámetros o funciones de premio y si se hubieran entrenado durante más episodios. Un ejemplo se ve en que a los modelos que se entrenaron una segunda vez con masas aleatorias, aprendieron a como afectaba al entorno.

Por último, las pruebas en la maqueta demostraron unas buenas aptitudes de control y ofrecen un comportamiento parecido al del PID, aunque algo mejor. Además, este controlador tiene la condición de que necesita ser ajustado en el propio robot, mientras que el agente por RL puede dar buenos resultados a partir de un modelo fiable del sistema, sin embargo, requiere de más recursos y de un tiempo de entrenamiento, en algunos casos largos, que pueden hacer al control por RL ineficiente.

6.2 Próximas líneas de investigación

En cuanto al control mediante RL y en línea con lo mencionado en el apartado anterior, gracias a la versatilidad de los agentes, a los que se les pueden dar más de una observación y pueden realizar más de una acción, se podría entrenar un solo agente capaz de controlar los 6 grados de libertad del robot. Esto conllevaría una mayor computación y tiempo de entrenamiento, reflejándose de nuevo la necesidad de usar una GPU. A parte, también convendría entrenar el propio robot y comprobar si se mejoran las capacidades de control.

Por otra parte, los trabajos anteriores, incluyendo este, han analizado el control de movimientos en una dirección, sin embargo, se puede investigar la combinación de ellos, empleando controladores complejos, como el Fuzzy que propuso Daniel o esté basado en RL.

Por último, se podrían añadir a la maqueta los sensores de localización GPS para tener un control absoluto de los movimientos de forward y lateral, y por tanto hacer un estudio más amplio. Además, también se puede contemplar la creación de un nuevo prototipo que comunique mediante un cordón umbilical con el software, de forma que contenga en su interior los elementos fundamentales y sea de menor tamaño, ya que el tanque de pruebas no permite grandes movimientos.

6.3 Implicaciones éticas

La robótica submarina no solo permite al ser humano aumentar su capacidad de exploración, dando a proyectos como el de UNEXMIN, sino también se están viendo más robots en misiones de rescate, que han permitido tener un reconocimiento de la zona más rápido y seguro. Por ello, el desarrollo de estos robots supone una línea de investigación en la que merece hacer hincapié.

Por otra parte, el concepto de Deep Learning está suponiendo una revolución, viendo su utilidad en un gran abanico de aplicaciones, desde en automoción hasta algoritmos de búsqueda o desarrollo de juegos. De esta forma, están naciendo cada vez más herramientas basadas en ello capaces de simplificar en grandes cantidades el trabajo.

Sin embargo, para los grandes progresos siempre hay otra cara de la moneda, en cuanto al Machine Learning ya se están promoviendo leyes que lo limiten debido a que se puede asemejar tanto al comportamiento humano que a veces no se puede distinguir, dando lugar a fraudes.

Mientras que la robótica submarina opera sobre un entorno muy peligroso y se necesita tener un conocimiento completo de los requerimientos, ya que algunos resultados pueden ser fatales, como los que se han comprobado recientemente por desgracia en la nave Titán. Por otro lado, gran parte de las investigaciones del siglo pasado se basaban en propósitos bélicos, dando un trasfondo negativo a los avances que han llevado a la robótica actual.

Con todo esto, sin dejar de lado los aspectos perjudiciales, se puede apreciar como la combinación de nuevas tecnologías como estas pueden llevar a resultados prometedores, consiguiendo un gran progreso en la exploración del fondo marino.

7. PLANIFICACIÓN Y PRESUPUESTO

Por último, una vez terminado el proyecto y realizado su estudio, se añade una planificación del trabajo para tener una noción del orden en el que se han cumplido los objetivos y el tiempo que han llevado. Además, también se describe el presupuesto necesario para su realización.

7.1 Planificación

El proyecto empezó el día 1 de abril de 2023, donde se propuso inicialmente realizar el control de movimientos combinados basándose en los trabajos previos y se contempló el uso de Machine Learning.

Durante las primeras semanas se empezó la documentación en ML y se definió el método de Reinforcement Learning como método de control. Al mismo tiempo se comentó con los tutores cambiar el objetivo del control de los movimientos combinados al control de los 6 movimientos individualmente en un entorno de Gazebo o de 2 de ellos en la maqueta.

De esta forma, se comprueba el buen estado de la maqueta y se termina de definir el proyecto para usarla en las pruebas.

A continuación, se adjunta un diagrama de Gannt con las fases del proyecto para mostrar de una forma ordenada como fue su avance.

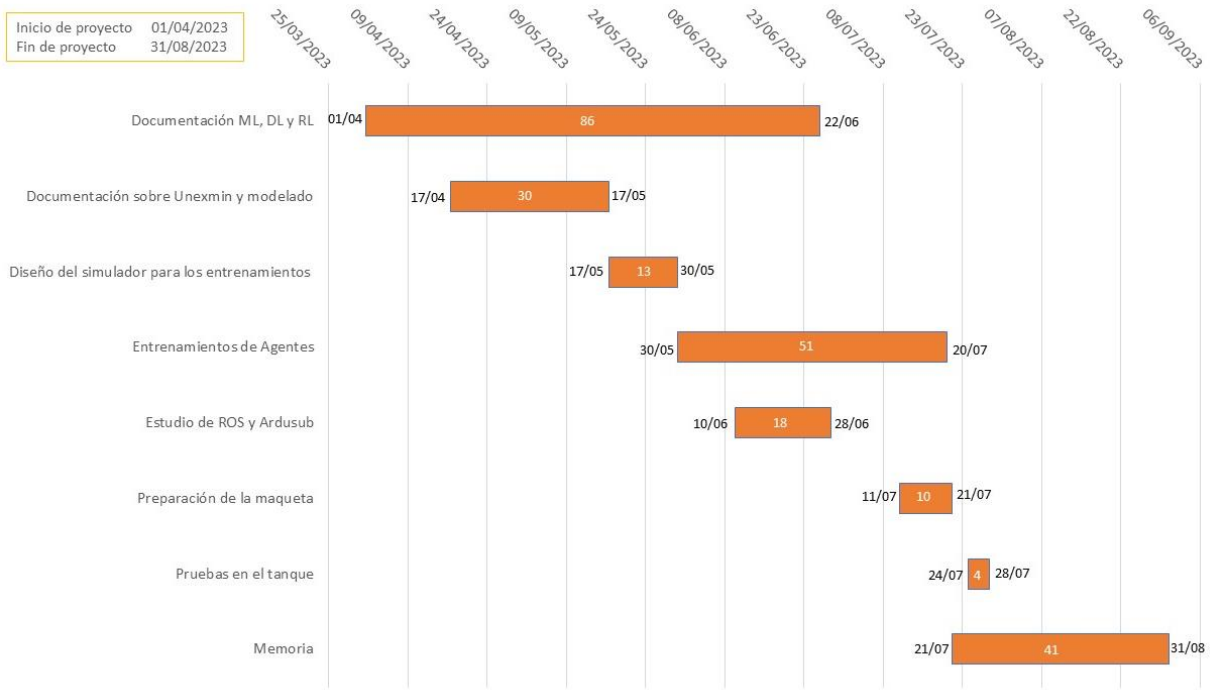

Figura 7.1: *Diagrama de Gantt (Fuente propia)*

7.2 Presupuesto

Para los costes del proyecto se va a tener en cuenta el valor de los programas utilizados junto con el recuento de horas de trabajo de las personas implicadas en el proyecto.

Tabla 5: Presupuesto.

Concepto	Precio Unitario	Cantidad	Precio Total
Licencia MATLAB R2022a	262 €	1 año	262 €
Reinforcement Learning Toolbox	105 €	1 año	105 €
Deep Learning Toolbox	105 €	1año	105 €
ROS Toolbox	105 €	1año	105 €
Ordenador portátil MSI MS-16H8	1200 €	-	1.200 €
Autor del proyecto	20 €/hora	320	6.400 €
Tutor del proyecto	40 €/hora	20	800 €
Cotutor del proyecto	40 €/hora	10	400 €
Total			**9.377**

De esta forma se estima que el proyecto ha costado un total de 9.377 €.

8. REFERENCIAS (IEEE)

[1] S. J. Russell, P. Norvig, y E. Davis, *Artificial intelligence: a modern approach*, 3rd ed. en Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010.

[2] M. Alcaraz-Carrazco, J. Gudiño-Lau, Ó. Issac-Zamora, S. M. Charre-Ibarra, J. A. Alcalá-Rodríguez, y D. Vélez-Díaz, «Robot submarino: estado del arte y diseño», *XIKUA Bol. Científico Esc. Super. Tlahuelilpan*, vol. 10, n.º 19, pp. 10-16, ene. 2022, doi: 10.29057/xikua.v10i19.7950.

[3] C. H. Valencia Niño y M. Suell Dutra, «Estado del arte de los vehículos autónomos sumergibles alimentados por energía solar», *ITECKNE Innov. E Investig. En Ing.*, vol. 7, n.º 1, pp. 46-53, 2010.

[4] H. A. Moreno, R. Saltarén, L. Puglisi, I. Carrera, P. Cárdenas, y C. Álvarez, «Robótica Submarina: Conceptos, Elementos, Modelado y Control», *Rev. Iberoam. Automática E Informática Ind. RIAI*, vol. 11, n.º 1, pp. 3-19, ene. 2014, doi: 10.1016/j.riai.2013.11.001.

[5] F. R. Cabrera Aguayo, «Diseño y desarrollo de robots submarinos bio-inspirados», PhD Thesis, Universidad Politécnica de Madrid, 2021. doi: 10.20868/UPM.thesis.69989.

[6] «Objectives», 16 de junio de 2016. https://www.unexmin.eu/the-project/objectives/ (accedido 15 de agosto de 2023).

[7] «ROBOTIC PLATFORM – UNEXUP». https://unexup.eu/robotic-platform/ (accedido 19 de agosto de 2023).

[8] R. Fernandez, E. R., Z. Milosevic, S. Dominguez, y C. Rossi, «Nonlinear Attitude Control of a Spherical Underwater Vehicle», *Sensors*, vol. 19, n.º 6, p. 1445, mar. 2019, doi: 10.3390/s19061445.

[9] Z. Milosevic, R. A. S. Fernandez, S. Dominguez, y C. Rossi, «Guidance for Autonomous Underwater Vehicles in Confined Semistructured Environments», *Sensors*, vol. 20, n.º 24, p. 7237, dic. 2020, doi: 10.3390/s20247237.

[10] M. del Monte «Diseño, modelado y simulación de algoritmos de control para un robot submarino», feb 2021.

[11] D. A. Burgoa, «Diseño y desarrollo software y hardware para robot submarino», jun. 2022.

[12] «Home», *Blue Robotics*. https://bluerobotics.com/ (accedido 21 de agosto de 2023).

[13] «3DR Pixhawk 1 Flight Controller (Discontinued) | PX4 User Guide». https://docs.px4.io/main/en/flight_controller/pixhawk.html (accedido 21 de agosto de 2023).

[14] R. P. Ltd, «Raspberry Pi», *Raspberry Pi*. https://www.raspberrypi.com/ (accedido 21 de agosto de 2023).

[15] «Turnigy Aerodrive DST-700 sin escobillas del motor Outrunner 700kv», *Hobbyking*. https://hobbyking.com/es_es/turnigy-aerodrive-dst-700-brushless-outrunner-motor-700kv.html (accedido 21 de agosto de 2023).

[16] Kiam Heong Ang, G. Chong, y Yun Li, «PID control system analysis, design, and technology», *IEEE Trans. Control Syst. Technol.*, vol. 13, n.º 4, pp. 559-576, jul. 2005, doi: 10.1109/TCST.2005.847331.

[17] F. AlMahamid y K. Grolinger, «Reinforcement Learning Algorithms: An Overview and Classification», en *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, sep. 2021, pp. 1-7. doi: 10.1109/CCECE53047.2021.9569056.

[18] E. Diederichs, «Reinforcement Learning - A Technical Introduction», *J. Auton. Intell.*, vol. 2, n.º 2, p. 25, jul. 2019, doi: 10.32629/jai.v2i2.45.

[19] T. P. Lillicrap *et al.*, «Continuous control with deep reinforcement learning». arXiv, 5 de julio de 2019. Accedido: 10 de agosto de 2023. [En línea]. Disponible en: http://arxiv.org/abs/1509.02971

[20] S. Dankwa y W. Zheng, «Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent», en *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, Vancouver BC Canada: ACM, ago. 2019, pp. 1-5. doi: 10.1145/3387168.3387199.

[21] J. Schulman, «Trust Region Policy Optimization».

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, y O. Klimov, «Proximal Policy Optimization Algorithms». arXiv, 28 de agosto de 2017. Accedido: 13 de agosto de 2023. [En línea]. Disponible en: http://arxiv.org/abs/1707.06347

[23] «Cursos online a su ritmo - MATLAB & Simulink». https://matlabacademy.mathworks.com/es (accedido 21 de agosto de 2023).

9. ANEXO

Se adjunta el script usado para la creación de los agentes y su entrenamiento, en este caso es para un agente entrenado por PPO.

```
%% Definición del entorno

Modelo = "PPO_PDal";
BlckAgent = "PPO_PDal/RL Agent";

numObs =2;
obsInfo = rlNumericSpec( [numObs 1]);
obsInfo.Name = "observations";
obsInfo.Description = "Error y efecto diferencial";


numAct = 1;
actInfo = rlNumericSpec([numAct 1]);
actInfo.Name = "acción de control";

env = rlSimulinkEnv(Modelo,BlckAgent,obsInfo,actInfo); %creación del entorno

%% Creación del actor y el crítico

L = 64; % numero de neuronas

% Red neuronal crítico
statePath = [
    featureInputLayer(numObs,Name ="netObsIn")
    fullyConnectedLayer(L,Name="CriticStateFC2")
    reluLayer
    fullyConnectedLayer(L)
    reluLayer
    fullyConnectedLayer(L/2)
    reluLayer
    fullyConnectedLayer(1,Name="CriticOutput")];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);

criticNetwork = dlnetwork(criticNetwork,Initialize=false); %creación de la red
neuronal
rng(0)

critic = rlValueFunction(initialize(criticNetwork),obsInfo); %creación del crítico


%Red neuronal de actor
inPath = [ featureInputLayer(prod(obsInfo.Dimension),'Name','myobs')
    fullyConnectedLayer(L)
    reluLayer
    fullyConnectedLayer(L)
    reluLayer
    fullyConnectedLayer(prod(actInfo.Dimension),'Name','infc') ];

meanPath = [ tanhLayer('Name','tanh'); % range: (-1,1)
```

```
            scalingLayer('Name','scale',...
            'Scale',10) ]; % range: (-10,10)

sdevPath = [softplusLayer('Name', 'splus')];

net = layerGraph(inPath);
net = addLayers(net,meanPath);
net = addLayers(net,sdevPath);

net = connectLayers(net,'infc','tanh/in');
net = connectLayers(net,'infc','splus/in');


actorNet = dlnetwork(net); %creación de la red neuronal

actor = rlContinuousGaussianActor(actorNet, ...
    obsInfo,actInfo, ActionMeanOutputNames="scale",...
    ActionStandardDeviationOutputNames="splus",...
    ObservationInputNames="myobs"); %creación del actor


%% Ajuste de hiperparámetros y condiciones de entrenamiento
Ts = 0.15;
Tf = 50;

agentOpts = rlPPOAgentOptions(...
    SampleTime=Ts,...
    ExperienceHorizon=256,...
    MiniBatchSize=256, ...
    DiscountFactor=0.99);


agentOpts.CriticOptimizerOptions.LearnRate = 1e-3;
agentOpts.CriticOptimizerOptions.GradientThreshold = 10;


agentOpts.ActorOptimizerOptions.LearnRate = 1e-3;
agentOpts.ActorOptimizerOptions.GradientThreshold = 10;
agentOpts.ActorOptimizerOptions.L2RegularizationFactor = 1e-3;


agent = rlPPOAgent(actor, critic, agentOpts); %creación del agente


maxepisodes = 10000;
maxsteps = ceil(Tf/Ts);
trainingOpts = rlTrainingOptions(...
    MaxEpisodes=maxepisodes,...
    MaxStepsPerEpisode=maxsteps,...
    Verbose=false,...
    Plots="training-progress");



trainingStats = train(agent,env,trainingOpts); %Entrenamiento
```