UNIVERSIDAD DE MÁLAGA

ESCUELA DE
INGENIERÍAS
INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

Escuela de Ingenierías Industriales

Departamento de Ingeniería de Sistemas y Automática

Trabajo Fin de Máster

Aprendizaje por refuerzo del balanceo de un robot de dos ruedas con microcontrolador de bajas prestaciones

Máster en Ingeniería Mecatrónica

Autor: Eduardo Lucena Alonso

Tutor: Juan Antonio Fernández Madrigal

Cotutora: Ana Cruz Martín

26 de junio de 2025

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a mis tutores, Juan Antonio y Ana, por su orientación, apoyo y paciencia durante todo el desarrollo de este trabajo.

También quiero dar las gracias a todos los miembros del grupo de investigación MAPIR por haber contribuido a crear un entorno tan agradable, facilitándome el día a día y haciendo mucho más llevadera la experiencia.

Agradezco profundamente a mis padres, a mi hermano y a mi pareja, por su compañía incondicional, su ánimo constante y su confianza en mí durante todo este proceso.

Por último, agradezco a la empresa Bricogeek no solo por proporcionar el kit necesario para este proyecto, sino también por su amabilidad al reemplazar el robot cuando se produjo un fallo en los motores. Su atención y profesionalidad fueron fundamentales para poder continuar con el desarrollo sin interrupciones.

Autor: Eduardo Lucena Alonso

Tutor: Juan Antonio Fernández Madrigal

Cotutora: Ana Cruz Martín

Departamento: Ingeniería de Sistemas y Automática

Titulación: Máster en Ingeniería Mecatrónica

Palabras clave: aprendizaje por refuerzo, Q-Learning, robot balanceador, entrenamiento en tiempo real, Arduino, robot de dos ruedas, control de equilibrio

RESUMEN

En este Trabajo Fin de Máster se ha desarrollado un sistema de control por aprendizaje por refuerzo para lograr el equilibrio autónomo de un robot de dos ruedas. El sistema se implementa sobre el robot Balboa 32U4 de Pololu, que cuenta con un microcontrolador de bajas prestaciones, sensores inerciales y motores de corriente continua como actuadores.

El proyecto aplica el algoritmo Q-Learning para que el robot aprenda a mantenerse en equilibrio sin modelo matemático ni controladores clásicos como el PID. Se han evaluado distintas definiciones del espacio de estados, a partir de los sensores inerciales integrados, junto con un conjunto discreto de acciones que ajustan la velocidad de los motores.

Una de las características clave del trabajo es que el entrenamiento se realiza íntegramente en el sistema real, sin simuladores. El aprendizaje tiene lugar en tiempo real sobre el microcontrolador, por lo que se han analizado en profundidad las limitaciones del hardware y las condiciones físicas del entorno.

Para el desarrollo e implementación se ha utilizado el entorno de programación Arduino, compatible con el microcontrolador ATmega32U4 del robot. Esta compatibilidad, junto con el uso de librerías oficiales del fabricante, ha facilitado el acceso al hardware y ha condicionado la metodología experimental del proyecto.

Este trabajo de fin de estudios ha sido financiado por el proyecto de investigación "*Tyrell: Time-Ready Reinforcement Learning for Robotic Skills and Tasks*", código PID2023-147392NB-I00, por MICI-U/AEI/10.13039/501100011033 y los fondos FEDER de la Unión Europea.

Author: Eduardo Lucena Alonso

Supervisor: Juan Antonio Fernández Madrigal

Co-supervisor: Ana Cruz Martín

Department: Systems and Automatic Control Engineering

Degree: Master in Mechatronic Engineering

Keywords: reinforcement learning, Q-Learning, balancing robot, real-time training, Arduino, two-wheeled robot, balance control

ABSTRACT

This Master Thesis presents the development of a reinforcement learning-based control system designed to achieve autonomous balancing of a two-wheeled robot. The system is implemented on the Pololu Balboa 32U4 robot, which integrates a low-performance microcontroller, inertial sensors, and DC motors as actuators.

The project applies the Q-Learning algorithm, enabling the robot to learn how to maintain balance without relying on a mathematical model or classical control strategies such as PID. Various definitions of the state space have been evaluated, using data from the onboard inertial sensors, along with a discrete set of actions corresponding to different motor speeds.

A key feature of this work is that training is carried out entirely on the physical system, without the use of simulators. The learning process runs in real time on the microcontroller, thus we have copied with the hardware limitations and the challenges of the physical environment.

The development and implementation have been deployed on the Arduino programming environment, which is compatible with the robot's ATmega32U4 microcontroller. This compatibility, along with the use of official libraries provided by the manufacturer, has facilitated access to the hardware and shaped the experimental methodology followed throughout the project.

This final Master Thesis has been funded by the research project "Tyrell: Time-Ready Reinforcement Learning for Robotic Skills and Tasks", code PID2023-147392NB-I00, by MICI-U/AEI/10.13039/501100011033 and the European Union FEDER funds.

ÍNDICE GENERAL

GLOSARIO

t: Paso de tiempo.

A_t: Acción seleccionada en el estado actual.

a_t: Acción concreta tomada en el instante t.

S_t: Estado actual.

S_{t+1}: Estado resultante tras ejecutar la acción.

s_t: Valor concreto del estado en el instante t.

r_t: Recompensa obtenida tras la transición de estado.

a': Acción candidata en el estado siguiente s'. Notación simplificada de a_{t+1}.

s': Estado siguiente tras ejecutar $a_t$en s_t. Notación simplificada de s_{t+1}.

π: Política de aprendizaje por refuerzo.

$\pi(s)$: Política determinista.

$\pi(a|s)$: Política estocástica.

γ: Factor de descuento. Controla la importancia relativa de recompensas futuras.

G_t: Retorno total esperado desde el instante t.

$v_\pi(s)$: Valor esperado del retorno para la función de valor de un estado.

$q_\pi(s, a)$: Valor esperado del retorno para la función de valor de acción.

$\mathbb{P}$: Operador de probabilidad.

$\mathbb{E}_\pi[\cdot]$: Valor esperado bajo la política π.

$argmax$: Operador que devuelve el valor de la variable que maximiza una función.

ϵ: Parámetro de exploración en la política ϵ-*greedy*. Controla la probabilidad de explorar.

$Q(s, a)$: Valor estimado del par estado-acción (s, a).

$Q^*(s, a)$: Valor óptimo del par estado-acción (s, a).

α: Tasa de aprendizaje (Learning rate).

θ_{acc}: Ángulo estimado a partir de los valores del acelerómetro, calculado mediante la función arcotangente.

a_x: Aceleración medida en el eje X, expresada en unidades de gravedad (g).

a_z: Aceleración medida en el eje Z, expresada en unidades de gravedad (g).

ω: Velocidad angular corregida en el eje Y, expresada en grados por segundo (º/s).

ω_{raw}: Velocidad angular bruta medida por el giroscopio en el eje Y, antes de compensar en el *offset*, en grados por segundo (º/s).

ω_{offset}: Valor de compensación correspondiente al desplazamiento cero del giroscopio, necesario para corregir ω_{raw}, expresada en grados por segundo (º/s).

θ_{gyro}: Ángulo de inclinación del robot obtenido exclusivamente de la integración del giroscopio.

α_{fexp}: Coeficiente del filtro exponencial.

α_{fcom}: Coeficiente del filtro complementario.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

1.1. Objeto

El objetivo principal de este Trabajo Fin de Máster es comprobar la viabilidad de que un robot diferencial, concretamente el robot Balboa 32U4 [1] (véase Figura 1.1), equipado con un microcontrolador de bajas prestaciones, sea capaz de mantenerse en equilibrio mediante un sistema de control basado en aprendizaje por refuerzo [2]. A diferencia de los métodos más clásicos, que requieren un modelo matemático del sistema o del uso de controladores tradicionales, como el PID [3], este proyecto plantea una solución en la que el comportamiento del robot se optimiza a partir de la experiencia directa de este con el entorno.

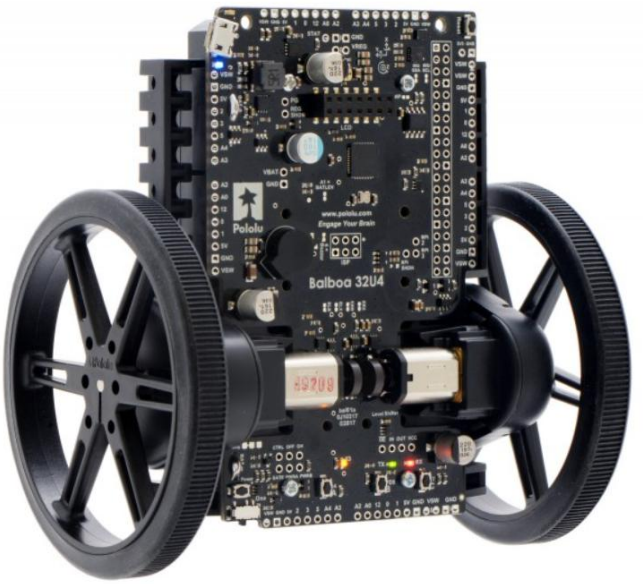

Figura 1.1. Robot Balboa 32U4 [1].

Más allá del propio equilibrio del robot, que puede conseguirse de manera más eficiente y robusta mediante controladores clásicos, el interés del proyecto radica en analizar si este tipo de control es factible en un entorno con recursos computacionales muy limitados. En concreto, se ha empleado el microcontrolador ATmega32U4 [4], que dispone de una memoria SRAM de tan sólo 2560 bytes. Esta limitación impone importantes restricciones no sólo en la representación interna del conocimiento necesaria para el aprendizaje por refuerzo (tabla Q), sino también en la ejecución en tiempo real del propio algoritmo.

El sistema deberá procesar los valores proporcionados por los sensores inerciales del robot para obtener el ángulo de inclinación del mismo y, a partir de dicho ángulo, seleccionar las acciones que modifiquen la velocidad de los motores de forma que compensen las variaciones en su inclinación. El aprendizaje debe permitir que el robot descubra, a lo largo de múltiples interacciones con el entorno, las secuencias de acciones más adecuadas para mantener su estabilidad y así cumplir lo mejor posible su objetivo.

Este planteamiento permite explorar el potencial del aprendizaje por refuerzo como alternativa de control en sistemas con capacidades muy reducidas, y se encuadra en su aplicabilidad a problemas reales de robótica.

1.2. Antecedentes

El aprendizaje por refuerzo ha sido ampliamente estudiado como una técnica dentro del aprendizaje automático orientada a la toma de decisiones. Su aplicación en robótica ha demostrado ser prometedora en tareas como la navegación [5], la manipulación [6] o el control de sistemas dinámicos [7], especialmente en situaciones en las que obtener un modelo matemático preciso del entorno resulta difícil o inviable.

En muchos trabajos previos, el desarrollo de algoritmos de aprendizaje por refuerzo en robots se ha basado en el uso de ordenadores de a bordo con gran capacidad de cálculo, o bien en un entrenamiento previo en entornos simulados [8], donde el agente puede interactuar con una réplica virtual del entorno antes de ser transferido al sistema físico. Hacer un uso directo de estos algoritmos directamente sobre el agente, sin asistencia externa y en tiempo real, sigue representando un reto relevante.

Uno de los principales desafíos radica en ejecutar el algoritmo en microcontroladores con capacidades muy limitadas [9], tanto en memoria como en velocidad de procesamiento. La necesidad de representar los espacios de estados y acciones, así como de almacenar las estimaciones de valor que guían el comportamiento del agente, imponen fuertes restricciones en este tipo de sistemas.

Este trabajo se enmarca dentro de esta línea de investigación, abordando de forma práctica la cuestión de si es posible ejecutar un algoritmo de aprendizaje por refuerzo directamente en un microcontrolador de bajas prestaciones y bajo condiciones reales de funcionamiento sin recurrir a simulaciones previas o procesamiento auxiliar. No es un objetivo del proyecto el conseguir igualar ni superar el rendimiento para esta tarea de otros controladores no basados en aprendizaje automático.

1.3. Metodología del proyecto

Para desarrollar el algoritmo de aprendizaje por refuerzo y cumplir los objetivos expuestos anteriormente se ha seguido la siguiente metodología:

Fase 1. Preparación del robot y estudio preliminar:

- Montaje del robot y prueba de funcionamiento básico de cada uno de los componentes que se verán involucrados en el funcionamiento del mismo.
- Estudio del algoritmo de aprendizaje por refuerzo y su implementación en lenguaje de programación C dentro del IDE de Arduino [10].
- Exploración del sistema de comunicación por puerto serie con el robot para la captura y recepción de los datos de interés que se obtendrán en el aprendizaje.

Fase 2. Desarrollo del algoritmo de aprendizaje:

- Adaptación de un algoritmo de aprendizaje por refuerzo a las limitaciones computacionales y su escritura en lenguaje C.
- Realización de pruebas unitarias y preliminares en el propio robot para validar el funcionamiento del algoritmo.

- Obtención y análisis de datos.

Fase 3. Análisis de resultados y extracción de conclusiones.

- Realización de distintas pruebas de aprendizaje completas para corroborar el rendimiento del aprendizaje de la tarea en cuestión.
- Análisis de las pruebas realizadas y planteamiento de las limitaciones encontradas.
- Proposición de posibles soluciones a las distintas limitaciones encontradas.

Fase 4. Preparación de la memoria.

- Documentación del trabajo realizado en el proyecto.

1.4. Herramientas del proyecto

Para el desarrollo del presente proyecto se han utilizado tanto recursos hardware como software, los cuales se detallan a continuación.

1.4.1. Herramientas hardware

La plataforma robótica empleada ha sido el kit Balboa 32U4 [11], que consta de una placa de control con microcontrolador ATmega32U4, sensores inerciales (IMU), interfaz de usuario y sistema de alimentación. Este kit no incluye algunos elementos mecánicos y electrónicos fundamentales, por lo que fue necesario complementarlo con ruedas de 80 mm de diámetro [12], motores de corriente continua con reductora de 75:1 [13] y baterías recargables para la alimentación del robot. Todo este material, y el propio kit, se adquirieron a la empresa Bricogeek [14].

Para la programación del sistema, el almacenamiento de datos y como soporte general del proyecto, se ha hecho uso de un portátil con procesador Intel Core i7, 16GB de RAM y unidad SSD, orientado al uso multimedia y técnico, que ha permitido compilar y cargar el código, así como registrar y visualizar los datos obtenidos durante el proceso de aprendizaje.

1.4.2. Herramientas software

El entorno principal de desarrollo ha sido el IDE de Arduino [15], compatible con el microcontrolador ATmega32U4 y que permite utilizar librerías específicas facilitadas por el fabricante, tanto para el uso del robot Balboa32U4 [16] como para los sensores inerciales [17]. Además, se ha empleado MATLAB [18] para almacenar los datos registrados y representarlos gráficamente. Por último, se ha utilizado Processing [19] como interfaz para el envío y recepción de datos a través del puerto serie, facilitando así la comunicación entre el robot y el ordenador.

Estas herramientas han permitido desarrollar e integrar todos los componentes necesarios del sistema de aprendizaje.

1.5. Estructura de la memoria

La memoria se compone de siete capítulos principales, además del resumen inicial, el apartado final de referencias y los anexos, todo esto con el objetivo de presentar de forma ordenada y progresiva el desarrollo del proyecto.

- **Capítulo 2:** Introduce el marco teórico necesario para comprender el algoritmo utilizado. Se explican los conceptos básicos del aprendizaje por refuerzo y el funcionamiento del algoritmo Q-Learning.
- **Capítulo 3:** Describe la plataforma física utilizada en el proyecto. Se detallan sus componentes más relevantes: microcontrolador, sensores, actuadores, *encoders* y sistema de alimentación; con especial atención a los sensores inerciales y las distintas técnicas que se han usado para obtener el ángulo.
- **Capítulo 4:** Presenta y desarrolla el uso de las herramientas software utilizadas: Arduino como entorno de desarrollo principal, MATLAB para análisis y almacenamiento de datos y Processing como interfaz de comunicación serie.
- **Capítulo 5:** En este capítulo se detalla todo lo relativo a la discretización del sistema para poder aplicar el algoritmo Q-Learning, es decir, los espacios de estados y acciones, la función de recompensa, la política de exploración y la selección de los parámetros que determinan el aprendizaje.

- **Capítulo 6:** En este capítulo se comparan los resultados obtenidos tras el entrenamiento del sistema con los distintos espacios de estados y para distintos tiempos de muestreo. También se analizan las limitaciones encontradas y se discuten los resultados.
- **Capítulo 7:** Expone las conclusiones generales del trabajo y se proponen posibles líneas de desarrollo futuro que podrían mejorar o ampliar los resultados obtenidos.
- **Anexos:** Muestran información complementaria al proyecto que no ha sido incluida en los capítulos principales.

2. Fundamentos de aprendizaje por refuerzo

En este capítulo se presentan los fundamentos teóricos necesarios para comprender el planteamiento adoptado en el presente trabajo. De esta manera, se introduce el concepto de aprendizaje por refuerzo, su funcionamiento general y los elementos clave que lo forman. Finalmente, se describe en detalle el algoritmo de Q-Learning, que ha sido seleccionado como núcleo del sistema de control implementado.

2.1. Introducción al aprendizaje por refuerzo

El **aprendizaje por refuerzo** (*Reinforcement Learning*, RL) es un paradigma del aprendizaje automático que permite a un agente aprender a tomar decisiones a través de su propia interacción con el entorno que le rodea. A diferencia de otros tipos de aprendizaje automático, como puede ser el aprendizaje supervisado, donde el sistema se entrena con muestras ya etiquetadas, en el aprendizaje por refuerzo el conocimiento se adquiere mediante la experiencia, **sin supervisión explícita**. Tal como se recoge de forma general en [20], esta experiencia se estructura en forma de **prueba y error**, es decir, el agente ejecuta acciones, observa sus consecuencias, y recibe señales de recompensa que guían su comportamiento futuro (véase Figura 2.1).

Este método resulta especialmente adecuado para problemas en los que no se dispone de un modelo explícito del entorno, y en los que el comportamiento óptimo debe descubrirse a través de la interacción con él. En el contexto de la robótica, donde las dinámicas pueden ser complejas o no lineales, el aprendizaje por refuerzo representa una alternativa eficaz a los

métodos clásicos de control, debido a que no es necesaria la obtención de un modelo matemático, lo cual aumentaría el nivel de complejidad significativamente.

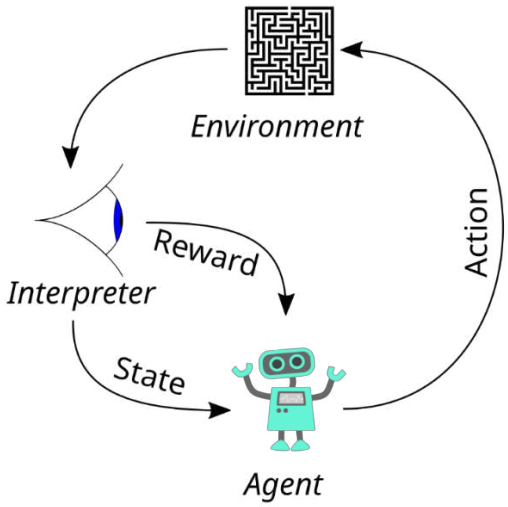

Figura 2.1. Esquema básico de la interacción entre agente y entorno en un sistema de aprendizaje por refuerzo [20].

Tal como describen Sutton y Barto [2], el aprendizaje por refuerzo se fundamenta en la **hipótesis de la recompensa**, la cual sostiene que todo objetivo puede expresarse como la maximización de una señal de recompensa acumulada. Lo más habitual es que el objetivo del agente sea aprender una política consistente en una función que determine qué acción tomar en cada estado con el fin de maximizar el **rendimiento esperado** (*expected return*), es decir, la suma futura de recompensas ponderadas a lo largo del tiempo.

2.2. Elementos clave del aprendizaje

En este apartado se presentan los elementos fundamentales que definen el marco teórico del aprendizaje por refuerzo. Dichos elementos permiten formalizar matemáticamente la interacción entre un agente y su respectivo entorno, lo que posibilita el desarrollo de algoritmos que optimizan su comportamiento a través de la experiencia. La comprensión clara de estos conceptos es fundamental antes de poder abordar el funcionamiento específico del algoritmo Q-Learning, que se describirá en el Apartado 2.3.

2.2.1. Entorno y agente

El aprendizaje por refuerzo se basa en la interacción continua entre un agente y su entorno. El agente es el ente que toma decisiones, mientras que el entorno representa todo aquello con lo que el agente interactúa y que responde a sus acciones, pudiendo verse afectado por las mismas. En cada paso de tiempo t, el agente recibe una observación del estado del entorno S_t, selecciona una acción A_t, y, como resultado, el entorno transiciona a un nuevo estado S_{t+1}, proporcionando una recompensa R_t que refleja el valor de la acción tomada.

Este proceso se repite de forma iterativa, constituyendo así el bucle de interacción del aprendizaje por refuerzo, tal como se representó en la Figura 2.1. De esta manera, se concluye que el objetivo del agente es aprender una estrategia (política) que le permita tomar decisiones que **maximicen las recompensas acumuladas a lo largo del tiempo**.

Cabe señalar que el comportamiento del entorno puede clasificarse en función de su grado de determinismo. Se dice que es **determinista** si, dada una acción A_t en un estado S_t siempre lleva exactamente al mismo estado siguiente S_{t+1} y da una misma recompensa R_t. En cambio, se dice **estocástico** si, dada la misma acción en el mismo estado, puede llevar a distintos estados siguientes con cierta probabilidad, y como consecuencia, también ofrece distintas recompensas.

Asimismo, el entorno se puede clasificar en **completamente observable** (si el agente tiene acceso total al estado real del entorno) o **parcialmente observable**, en cuyo caso el agente solo percibe una parte de la información relevante del estado, lo que añade complejidad al proceso de aprendizaje.

2.2.2. Estados, acciones y recompensas

En el marco del aprendizaje por refuerzo, la interacción entre el agente y el entorno se formaliza mediante tres elementos fundamentales: el **estado**, la **acción** y la **recompensa**. Estos componentes permiten definir el proceso de toma de decisiones y evaluar las consecuencias de las acciones ejecutadas por el agente.

El **estado S_t** representa la información que el agente percibe del entorno en el instante t. Este estado puede reflejar una descripción completa del entorno o una observación parcial,

dependiendo de si el entorno es completamente observable o no. En un entorno físico, como el de un robot móvil, el estado podría incluir, por ejemplo, el ángulo de inclinación, la posición o la velocidad angular del sistema. El conjunto de los posibles estados que pueden existir recibe el nombre de **espacio de estados**.

La **acción** $\boldsymbol{A_t}$ es la decisión que toma el agente en función del estado percibido. El conjunto de acciones posibles en un estado determinado conforma el **espacio de acciones**. En tareas de control, las acciones suelen corresponder a señales de control, como encender un motor, modificar una velocidad, o aplicar una determinada fuerza.

Tras ejecutar una acción, el entorno devuelve una recompensa $\boldsymbol{R_t}$, que es un valor numérico que refleja el resultado inmediato de dicha acción. Esta señal tiene como objetivo indicar al agente la utilidad o conveniencia de la acción ejecutada, en función del objetivo general del sistema. La recompensa no evalúa necesariamente el rendimiento final del agente, sino que sirve como guía local para ajustar su comportamiento.

De forma general, el objetivo del agente es aprender a seleccionar acciones que, acumuladas en el tiempo, maximicen la suma de recompensas recibidas. Esta secuencia de estados, acciones y recompensas constituye la base sobre la cual se construyen las políticas de decisión en los algoritmos de aprendizaje por refuerzo. Esta secuencia puede representarse formalmente mediante la tupla $(\boldsymbol{S_t, A_t, R_t, S_{t+1}})$, la cual describe un paso completo de interacción entre el agente y el entorno.

2.2.3. Política

En el aprendizaje por refuerzo, el comportamiento del agente se define a través de una política, representada comúnmente por la letra griega π. La política determina la forma en la que el agente selecciona sus acciones en función del estado en el que se encuentre, es decir, es una función que mapea estados a acciones.

Matemáticamente, una política puede ser:

- **Determinista**, si asigna a cada estado una única acción, tal como se muestra en la Ecuación (2.1).

$$\pi(s) = a \tag{2.1}$$

Lo que significa que, si el agente se encuentra en el estado s, siempre ejecutará la acción a.

- **Estocástica**, si define una distribución de probabilidad sobre las acciones en cada estado, tal como queda reflejado en la Ecuación (2.2).

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s] \tag{2.2}$$

Es decir, especifica la probabilidad de elegir la acción a dado el estado s.

Por otra parte, los algoritmos de aprendizaje por refuerzo se pueden clasificar, en función de cómo construyen o utilizan la política, en dos grandes grupos [2][21]:

- **Algoritmos basados en valor** (*value-based*):

  En este algoritmo, el agente no aprende directamente la política, sino que construye una función que estima el valor de cada acción o estado (por ejemplo, la función $Q(s, a)$, que representa el valor esperado de realizar la acción a en el estado s en el tiempo total). A partir de esta información, el agente actúa eligiendo la acción que maximiza ese valor, tal como puede verse en la Ecuación (2.3).

$$\pi(s) = \arg\max_a Q(s, a) \tag{2.3}$$

  Este es el caso de algoritmos como **Q-Learning** [22] y **SARSA** [23].

- **Algoritmos basados en política** (*policy-based*):

  En este caso, el agente aprende directamente una función que representa la política, sin necesidad de calcular una función de valor. Este tipo de algoritmo se emplea sobre todo en casos en los que el espacio de acciones es continuo o cuando se desea aprender una política estocástica. Ejemplos típicos son **Policy Gradient** [24] o **REINFORCE**[25].

Por último, otra clasificación importante distingue entre *on-policy* y *off-policy*, según la relación entre la política utilizada mientras se aprende y la política objetivo del aprendizaje. La primera, también llamada *behaviour policy* o política de comportamiento, permite que el agente pruebe acciones distintas a las que considera óptimas en cada momento, con el objetivo de descubrir nuevas estrategias o estados que podrían resultar más beneficiosos a largo plazo.

De este modo, se puede diferenciar entre dos métodos:

- **On-Policy**: El agente aprende usando la política que en cada momento cree que es la óptima. Es decir, la política que se sigue para actuar es la misma que se está optimizando. Un ejemplo clásico es el algoritmo **SARSA**.
- **Off-Policy**: El agente aprende usando una política de selección de acciones distinta a la que busca. Es decir, puede explorar con una política (ϵ-*greedy* [26]) mientras que aprende una política óptima diferente. Esta estrategia permite mayor flexibilidad y reutilización de experiencias. El algoritmo **Q-Learning** es el ejemplo más representativo de esta categoría.

En este trabajo se utiliza el algoritmo Q-Learning, lo que implica un planteamiento **basado en valor** y ***off-policy***. Comprender el papel de la política, sus formas de representación y su influencia en el comportamiento del agente resulta esencial para el diseño y análisis de cualquier sistema de control basado en aprendizaje por refuerzo.

2.2.4. Rendimiento esperado y funciones de valor

Como ya se ha comentado, el objetivo fundamental del aprendizaje por refuerzo es maximizar la **recompensa acumulada** que un agente puede obtener a lo largo del tiempo como consecuencia de sus decisiones. Para formalizar este objetivo, se define el concepto de **retorno** (G_t), que representa la suma de recompensas futuras a partir del instante t.

En su forma más común, el retorno se expresa como una suma ponderada mediante un **factor de descuento** γ, teniendo dicho factor que cumplir la Ecuación (2.4). El factor de descuento permite controlar la importancia de las recompensas inmediatas frente a las futuras, tal como queda reflejado en la Ecuación (2.5).

$$0 \leq \gamma \leq 1 \tag{2.4}$$

$$G_t = R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k * R_{t+k+1} \tag{2.5}$$

El factor γ permite que el agente valore más las recompensas inmediatas y penalice aquellas que están muy alejadas en el tiempo. Un valor de $\gamma = 0$ implica que solo se considera la recompensa inmediata, mientras que valores cercanos a 1 dan casi igual peso a las recompensas futuras.

A partir de este retorno, se definen dos funciones clave:

- Función de valor de un estado:

  La función de valor de un estado s bajo una política π, denotada como $v_\pi(s)$, se define como el valor esperado del retorno cuando el agente comienza en el estado s y sigue la política π desde ese preciso instante (véanse Ecuación (2.6) y Figura 2.2).

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s] \tag{2.6}$$

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \cdots \mid S_t = s\right]$$

Función de valor | Rendimiento esperado con descuento | Comenzando en el estado s

Figura 2.2. Desglose de la función de valor de un estado $v_\pi(s)$.

  Esta función cuantifica lo bueno que es estar en un estado determinado, suponiendo que el agente sigue la política π.

- Función de valor de acción:

  De forma análoga, la función de valor de acción $q_\pi(s, a)$ representa el valor esperado de realizar la acción a en el estado s, y luego seguir la política π a partir de dicho instante (véanse Ecuación (2.7) y Figura 2.3).

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{2.7}$$

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma * R_{t+2} + \gamma^2 * R_{t+3} + \cdots \mid S_t = s, A_t = a]$$

Función de valor

Rendimiento esperado con descuento

Comenzando en el estado s y realizando la acción a

Figura 2.3. Desglose de la función de valor de un estado $q_\pi(s,a)$.

Estas funciones de valor son esenciales para evaluar la calidad de una política y guiar su mejora en los algoritmos de aprendizaje por refuerzo.

2.2.5. Propiedad de Markov

La **propiedad de Markov** es un concepto fundamental en el aprendizaje por refuerzo, ya que permite modelar el sistema como un **proceso de decisión de Markov** (*Markov Decision Process, MDP*), marco matemático sobre el que se basan la mayoría de los algoritmos clásicos como Q-Learning o SARSA [2]. Por esta razón, resulta necesario introducir esta propiedad antes de profundizar en dichos algoritmos.

Esta propiedad establece que la dinámica del entorno solo depende del estado y acción actuales, sin necesidad de considerar el historial completo de interacción entre el agente y el entorno. Formalmente, esto se expresa según lo indicado en la Ecuación (2.8):

$$\mathbb{P}[S_{t+1} = s', R_t = r \mid S_t = s, A_t = a] = \mathbb{P}[S_{t+1} = s', R_t = r \mid H_t] \tag{2.8}$$

Donde H_t representa el historial completo hasta el instante t y S_t, A_t y R_t denotan el estado, la acción y la recompensa asociada, respectivamente. La igualdad indica que **el futuro es condicionalmente independiente del pasado, dado el presente**.

Este supuesto permite que los algoritmos de aprendizaje por refuerzo enfoquen su aprendizaje únicamente en la relación entre el estado actual, la acción ejecutada y las consecuencias inmediatas, lo que **simplifica en gran medida el análisis e implementación de políticas**. Sin embargo, para que se cumpla esta propiedad, es necesario que el agente tenga acceso al estado del entorno; en caso contrario, el entorno se considera parcialmente observable

y pueden requerirse técnicas adicionales, como estimadores de estado o memoria. Hay que tener en cuenta que en un sistema físico robótico los sensores tienen ruido, lo que significa que **no hay observabilidad total**; habrá siempre alguna desviación de la hipótesis de markovianidad.

A pesar de eso, asumir que el entorno cumple la propiedad de Markov es una **condición teórica clave** que justifica la validez de los modelos y algoritmos utilizados a lo largo de este trabajo.

2.2.6. Tipos de tareas: episódicas y continuas

En aprendizaje por refuerzo, los casos de estudio pueden dividirse en dos grandes categorías: **tareas episódicas** y **tareas continuas**. Esta distinción afecta directamente a la forma en la que se calcula el retorno y al modo en que el agente regula su comportamiento.

- **Tareas episódicas:** Una tarea se considera episódica cuando tiene un comienzo y un final definidos. Para ello, el agente realiza una serie de acciones que, eventualmente, lo llevará a un estado terminal. Cuando dicho estado se alcance, el episodio termina, y el entorno deberá reiniciarse para iniciar así un nuevo episodio. Algunos ejemplos comunes son juegos o entornos de simulación donde el agente intenta conseguir un objetivo o evitar un fallo.

  Este tipo de estructura facilita el aprendizaje, ya que el agente puede aprovechar la segmentación en episodios para poder reiniciar, explorar nuevas estrategias y así ser capaz de corregir errores a lo largo de múltiples experiencias.

- **Tareas continuas:** Estas tareas, por el contrario, no tienen un estado terminal. Esto permite que la interacción entre agente y entorno se extienda indefinidamente, lo que obliga al agente a tener que aprender a comportarse correctamente sin un punto de reinicio. Un ejemplo común podría ser el control de un sistema robótico que debe navegar indefinidamente.

El tipo de tarea que se plantea influye directamente en el diseño de la función de retorno, las estrategias de exploración y el proceso de aprendizaje. En este caso de estudio, el problema de balanceo continuo del robot se considera una **tarea episódica**, ya que cada vez que el robot

supere un ángulo establecido, se considerará que ha caído y por lo tanto se iniciará otro episodio, partiendo siempre de una misma posición inicial de equilibrio.

2.2.7. Equilibrio exploración-explotación

Para finalizar esta breve introducción al aprendizaje por refuerzo, se va a tratar uno de los desafíos más importantes de este campo de estudio, el cual consiste en lograr un equilibrio adecuado entre exploración y explotación. Este dilema recibe el nombre de **dilema exploración-explotación** y surge del hecho de que un agente debe, por un lado, aprovechar el conocimiento que ha acumulado para obtener recompensas lo más elevadas posibles (explotación) pero también debe probar acciones distintas que podrían conducirlo a mejores resultados futuros (exploración).

Una de las estrategias de selección de acción más usadas para resolver este dilema es el uso de la política de comportamiento ϵ-*greedy*. En esta técnica, el agente elige con probabilidad ϵ una acción aleatoria (exploración) y con probabilidad $(1-\epsilon)$ la acción con mayor valor esperado (explotación) (véase Ecuación (2.9)).

$$\pi(s) = \begin{cases} acción\ aleatoria & con\ probabilidad\ \epsilon \\ \arg\max_a Q(s,a) & con\ probabilidad\ 1-\epsilon \end{cases} \tag{2.9}$$

Este planteamiento permite controlar de manera directa el nivel de exploración mediante el parámetro ϵ, que puede mantenerse constante o decrecer a lo largo del tiempo, para así poder facilitar una explotación progresiva con la política que el agente aprende.

El equilibrio entre estas dos estrategias no tiene una solución única, ya que es totalmente dependiente del entorno, del tiempo disponible para realizar el entrenamiento y de los objetivos particulares del agente de cada caso de estudio. En este trabajo se empleará una política ϵ-*greedy* con valor decreciente de ϵ, lo que permite una exploración inicial más amplia que se va reduciendo a medida que el agente gana experiencia. Hay que hacer notar que si ϵ no se mantiene constante, esta estrategia tenderá asintóticamente a la política óptima con el tiempo, siendo por tanto válida tanto para métodos *off-policy* como *on-policy*, mientras que si se deja constante sólo sería válido para los primeros.

2.3. Q-Learning

Tras la introducción al aprendizaje por refuerzo, este apartado se centra en el algoritmo Q-Learning, que constituye el núcleo del sistema de control desarrollado en este proyecto. Se trata de un método basado en valor que permite al agente encontrar una política, sin necesidad de un modelo del entorno. Esta característica lo hace idóneo para su uso en entornos físicos reales y sistemas con recursos limitados, como es el caso del robot empleado en este trabajo.

Q-Learning, como ya se ha comentado en el Capítulo 2, es un algoritmo *off-policy*, lo que le permite aprender la política óptima incluso mientras actúa con una política diferente, típicamente una ϵ-*greedy* (previamente introducida en la Sección 2.2.7). Esta estrategia ofrece una separación clara entre exploración y explotación, aumentado su robustez en entornos no deterministas o parcialmente observables, como es el caso del robot.

A lo largo de los siguientes subapartados se describirá el funcionamiento general de Q-Learning, su base matemática en la ecuación de Bellman [2], y la manera en la que se representa y actualiza el conocimiento del agente mediante la tabla Q. Finalmente, se abordarán algunas consideraciones prácticas que justifican su elección.

2.3.1. Objetivo y funcionamiento del algoritmo Q-Learning

El algoritmo Q-Learning constituye uno de los planteamientos más representativos dentro del aprendizaje por refuerzo basado en valor. Su objetivo principal es estimar la función de valor de acción óptima, definida como $Q^*(s, a)$, la cual asigna a cada par estado-acción el valor esperado del retorno acumulado si se realiza la acción a en el estado s, y se sigue una política **óptima** a partir de ese instante. A diferencia de otros algoritmos que requieren un modelo del entorno, Q-Learning aprende únicamente a partir de su interacción con el mismo.

Una de las claves de Q-Learning es el uso de una estructura denominada tabla Q, donde se almacenan los valores de la función $Q(s, a)$ de todos los pares de estados y acciones existentes. Esta tabla se actualiza de forma iterativa a medida que el agente interactúa con el entorno, lo que permite construir progresivamente una representación de los datos aprendidos por el agente.

Para ello, Q-Learning se apoya en la ecuación de Bellman [27], la cual permite expresar el valor óptimo esperado de un par estado-acción $Q^*(s,a)$ de forma recursiva, como se muestra en la Ecuación (2.10).

$$Q^*(s,a) = \mathbb{E}\left[R_t + \gamma \max_{a'} Q^*(s',a') \mid S_t = s, A_t = a\right] \quad (2.10)$$

Esta fórmula establece que el valor óptimo de una acción para un estado dado equivale a la recompensa inmediata R_t más el valor descontado del mejor valor de acción posible en el siguiente estado s', considerando que a partir de ese punto se actúa de manera óptima. En otras palabras, **el valor de una acción no depende únicamente de su efecto inmediato, sino también de las consecuencias futuras que desencadena** [2].

Dado que un entorno real normalmente es estocástico, esta estimación se expresa como una esperanza matemática sobre todas las posibles transiciones, lo que permite generalizar el comportamiento del agente en situaciones con incertidumbre.

Durante el aprendizaje, esta ecuación no se resuelve explícitamente, sino que se utiliza como base para una regla de actualización incremental que se encargará de ir actualizando el valor de $Q(s,a)$ a medida que el agente acumula experiencia. Esta regla se detallará en la siguiente sección.

La actualización de los valores de la tabla Q se realiza tras cada interacción del agente con el entorno siguiendo la Ecuación (2.11).

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[R_t + \gamma * \max_{a'} Q^*(s',a') - Q(s_t,a_t)\right] \quad (2.11)$$

Donde:

- $\alpha \in (0,1]$ es la **tasa de aprendizaje,** que regula cuánto influye la nueva información.
- $\gamma \in [0,1]$ es el **factor de descuento**, ya descrito, que controla la importancia de las recompensas futuras.
- R_t es la recompensa obtenida tras ejecutar la acción a_t.
- $\max_{a'} Q^*(s',a')$ es el valor estimado de la mejor acción en el siguiente estado.

Esta regla de actualización hace que Q-Learning pertenezca a la familia de algoritmos conocidos como **aprendizaje por diferencias temporales (*Temporal-Difference Learning, TD*)**. En concreto, Q-Learning usa una variante denominada **TD(0)**, en la que el valor de $Q(s, a)$ se actualiza tras cada paso del agente, usando la recompensa inmediata y una estimación del valor del futuro. A diferencia de los métodos Monte-Carlo, que requieren completar un episodio para calcular el retorno total, TD(0) permite aprender de forma incremental en entornos continuos o en tiempo real.

Respecto a la implementación interna de la tabla Q, esta es una matriz bidimensional donde:

- Las **filas** corresponden a los estados posibles del sistema.
- Las **columnas** representan las acciones disponibles en cada estado.

Cada celda almacenará el valor $Q(s, a)$ correspondiente. En entornos discretos y acotados, esta implementación resulta eficiente y sencilla, ya que permite acceder directamente a cualquier valor mediante índices.

A continuación se muestra un ejemplo ilustrativo del contenido inicial de una tabla Q para un problema básico (véase Figura 2.4), en el cual un ratón (agente) deberá explorar el entorno mediante una serie de acciones, pudiendo así encontrar recompensas positivas (queso) o recompensas negativas que finalizarán el episodio (veneno).

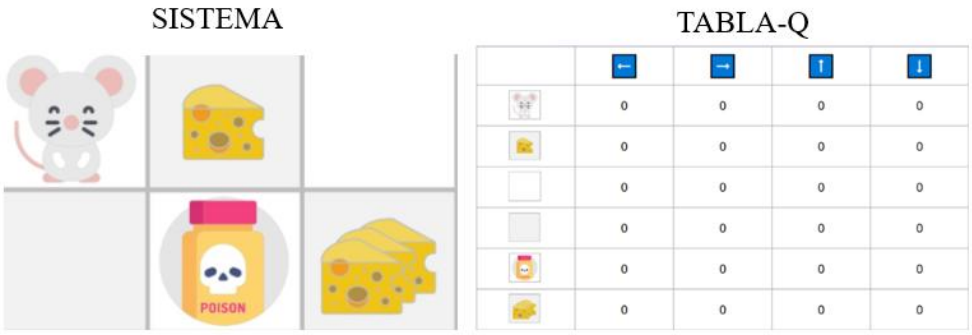

Figura 2.4. Ejemplo de una tabla Q con 6 estados y 4 acciones. Adaptado [27].

Tal como se puede observar, hay tantas columnas como acciones posibles tiene el agente, y por otro lado existen tantas filas como estados posibles. Inicialmente, la tabla Q contiene valores nulos; según el agente comience a interactuar con el entorno, los valores de dicha tabla

se irán modificando e influyendo directamente en el comportamiento del agente, obteniendo así recompensas acumuladas mayores según avance el aprendizaje.

2.3.2. Consideraciones prácticas de la aplicación de Q-Learning

La elección de Q-Learning como núcleo del sistema de aprendizaje desarrollado en este trabajo no ha sido arbitraria. Se trata de un algoritmo que presenta ventajas relevantes para su implementación en entornos físicos con recursos computacionales limitados, como es el caso del robot Balboa 32U4.

En primer lugar, Q-Learning es un algoritmo **model-free**, es decir, no requiere una descripción explícita de la dinámica del entorno ni una función de transición entre estados. Esta propiedad resulta clave en robótica, donde modelar con precisión el comportamiento del sistema físico puede ser inviable o demasiado costoso.

En segundo lugar, su mecanismo de aprendizaje por diferencias temporales (TD) le permite **aprender de forma incremental paso a paso**, lo que se ajusta a las restricciones de procesamiento en tiempo real del microcontrolador ATmega32U4. Gracias a ello, el agente no necesita esperar al final de un episodio para actualizar su conocimiento.

Además, el uso de una **tabla Q discreta** permite almacenar los valores de la función $Q(s, a)$ en estructuras de memoria sencillas y compactas. Este tipo de representación es compatible con las limitaciones de memoria del sistema (2.5 kB de SRAM) y evita la necesidad de recurrir a redes neuronales o funciones de aproximación complejas.

Finalmente, la naturaleza ***off-policy*** del algoritmo aporta una ventaja adicional: el agente puede explorar el entorno usando una política de exploración (como ϵ-*greedy*), mientras aprende a partir de estimaciones basadas en la política óptima. Esta separación entre exploración y aprendizaje aporta estabilidad y permite que el agente converja hacia una política eficaz incluso en entornos estocásticos.

Por todo ello, Q-Learning se presenta como una solución viable y eficaz para entrenar un agente de equilibrio en un entorno real, sin simulación previa, y utilizando exclusivamente los recursos del sistema empotrado. Los mayores inconvenientes que puede tener su uso son que

se requiere una discretización adecuada de estados y acciones y que la observabilidad debería ser cercana a la total, cosa imposible en un sistema físico dado el ruido sensorial, por lo que el aprendizaje puede verse perjudicado cuanto más ruido de este tipo haya (hay que tener en cuenta que las soluciones a este último problema requieren computaciones muy intensivas y almacenamientos muy grandes en memoria, algo imposible en el sistema tratado).

3. Especificaciones del Robot Balboa 32U4

A lo largo de este capítulo se detallan los principales componentes hardware del robot empleado en este TFM. Para aquellos lectores que deseen ampliar la información técnica o consultar detalles específicos sobre el hardware, se recomienda acceder a la **guía oficial del Balboa 32U4** proporcionada por el fabricante [1].

3.1. Introducción

El **Balboa32U4** es un robot móvil de dos ruedas diseñado específicamente para realizar tareas de equilibrio. Su placa de control integra el microcontrolador de arquitectura AVR [28] **ATmega32U4**. Dicho microcontrolador es compatible con el entorno de desarrollo Arduino (IDE), desde el cual puede programarse fácilmente. Esta placa incorpora controladores de motor tipo *H-bridge*, un conjunto de sensores inerciales, *encoders* de cuadratura, botones de usuario, indicadores LED y un sistema de alimentación estable y eficiente.

A diferencia de otras plataformas que requieren de hardware adicional para implementar control de alto nivel, el Balboa32U4 permite el desarrollo e implementación de algoritmos de control completamente en su propio microcontrolador, lo que lo convierte en un entorno ideal para validar algoritmos de aprendizaje. Además, cabe recalcar que, a pesar de que el sistema tiene compatibilidad con dispositivos como Raspberry Pi [29] (véase Figura 3.1), lo cual permitiría aumentar las prestaciones considerablemente, en este proyecto el robot se ha usado sin ningún tipo de apoyo computacional adicional con el fin de comprobar las limitaciones del aprendizaje por refuerzo en un sistema de bajas prestaciones.

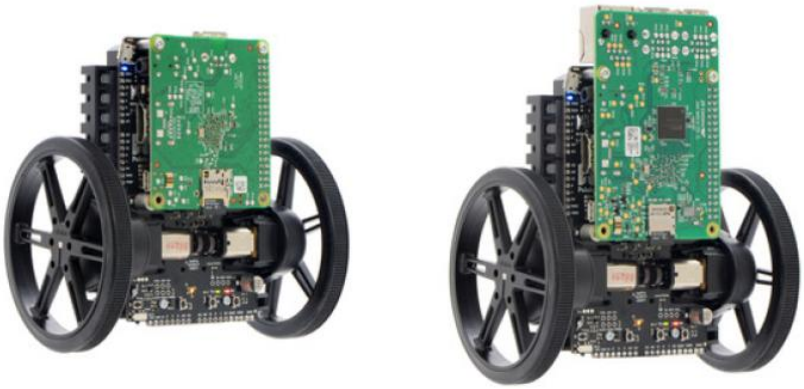

Figura 3.1. Compatibilidad del robot Balboa 32U4 con dos modelos de Raspberry Pi [1].

Esta plataforma tiene un diseño compacto, compatibilidad con el IDE de Arduino y posibilidad de acceder directamente a datos de sensores de alta frecuencia, como el acelerómetro y el giroscopio, siendo estos elementos clave para el balanceo del robot. Sus características de memoria y recursos limitados imponen un reto a la implementación del algoritmo Q-Learning, forzando decisiones como la discretización del espacio de estados y acciones para reducir el tamaño de la tabla Q perjudicando lo menos posible al aprendizaje.

En los siguientes apartados se describen con mayor detalle los distintos elementos que conforman el sistema. En caso de que se requiera consultar la asignación específica de pines del sistema Balboa 32U4, esta puede encontrarse en el ANEXO I (Figura A1.1). Dicho anexo recopila el esquema completo proporcionado por el fabricante y se incluye con el objetivo de facilitar la consulta rápida de los pines, sin necesidad de recurrir al datasheet original.

3.2. Microcontrolador

Como ya se ha mencionado anteriormente, el núcleo del sistema Balboa 32U4 es el microcontrolador **ATmega32U4** (véase Figura 3.2), fabricado por **Microchip Technology** [30] y basado en la **arquitectura AVR de 8 bits**. Este funciona a una frecuencia de reloj de **16 MHz** e integra periféricos suficientes para realizar control en tiempo real, incluyendo interfaz USB nativa y conversores analógico-digitales.

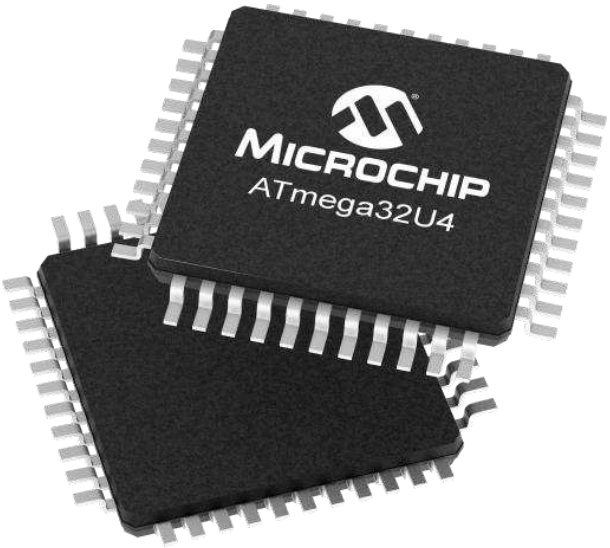

Figura 3.2. Microcontrolador ATmega32U4 [30].

Esta elección ha permitido desarrollar, compilar y cargar el código directamente desde el **IDE de Arduino**, de forma sencilla y eficiente.

Las características principales de memoria de este microcontrolador se resumen en la Tabla 3.1. Estas han condicionado directamente el diseño del sistema de aprendizaje, imponiendo la necesidad de utilizar una representación tabular para la función $Q(s, a)$, discretizar el espacio de estados y minimizar el uso de estructuras dinámicas, tal como ya se había comentado anteriormente.

TIPO DE MEMORIA	CAPACIDAD TOTAL	USO ESPECÍFICO EN ESTE PROYECTO
FLASH	32 KB (de los cuales, 4 KB son ocupados por el bootloader)	Almacenamiento del código del programa. Quedan 28 KB disponibles para el usuario.
SRAM	2.5 KB (2560 bytes)	Almacenamiento de variables, buffers y estructuras dinámicas como la tabla Q.
EEPROM	1 KB	Almacenamiento persistente no volátil. No utilizado en el proyecto

Tabla 3.1. Capacidad de las distintas memorias del microcontrolador.

Por otro lado, el ATmega32U4 dispone de **cuatro temporizadores hardware** que permiten control de ondas PWM y generación de interrupciones (véase Tabla 3.2). Aunque estos temporizadores pueden configurarse directamente a bajo nivel, su uso en este proyecto está parcialmente condicionado por el entorno de Arduino y las librerías empleadas, que asignan automáticamente ciertos temporizadores a funciones específicas (como la generación de señales PWM o el control del tiempo con funciones predeterminadas de Arduino).

TEMPORIZADOR	USO PRINCIPAL	RESERVADO POR	OBSERVACIONES
TIMER0	Funciones de temporización.	Entorno Arduino.	No recomendable modificar; afecta al sistema base.
TIMER1	Generación de señales PWM para control de motores.	Librería Balboa 32U4.	Usado para controlar la velocidad mediante funciones.
TIMER3	Libre para uso personalizado.	Ninguno.	Puede usarse si se requiere control más preciso.
TIMER4	Control del zumbador integrado.	Librería Balboa 32U4.	Limitado a salidas de sonido y tonos.

Tabla 3.2. Temporizadores del ATmega32U4.

3.3. Interfaz de usuario

El sistema Balboa 32U4 incorpora una sencilla pero útil interfaz de usuario compuesta por **tres botones programables**, **tres LEDs de usuario** y un **zumbador integrado**, todos accesibles desde el microcontrolador y fácilmente gestionables mediante el entorno Arduino.

Los botones A, B y C están conectados a los pines PB3, PD5 y PB0 respectivamente (véase ANEXO I), y su funcionamiento consiste en llevar el pin a nivel bajo cuando son presionados. En este proyecto se han utilizado para **controlar manualmente el flujo de aprendizaje** durante las pruebas, permitiendo iniciar o finalizar entrenamientos, así como activar funciones como el **envío o recepción de datos por puerto serie**. Esta interacción física directa ha resultado especialmente útil para depurar el sistema sin necesidad de una interfaz gráfica o comunicación constante con el ordenador.

Los LEDs de usuario, conectados a los mismos pines, han sido empleados como **indicadores visuales del estado del sistema**. Por ejemplo, se han utilizado para señalar si el

agente se encuentra en fase de entrenamiento, en modo espera, o si ha ocurrido algún error en la comunicación.

Finalmente, el **zumbador**, controlado mediante el temporizador 4, ha permitido ofrecer señales acústicas que notifican al usuario eventos clave, como el inicio o finalización de un episodio, la recepción de un mensaje o un reinicio del sistema. Esto ha sido especialmente útil en pruebas prolongadas donde la atención visual no estaba permanentemente en el robot.

La Tabla 3.3 resume los elementos de la interfaz de usuario utilizados en este proyecto, destacando la funcionalidad asignada a cada componente durante las distintas fases del aprendizaje.

COMPONENTE	PIN DE LA CPU	USO EN EL PROYECTO
Botón A	PB3	Inicializar tabla Q
Botón B	PD5	Recibir tabla Q para continuar con el aprendizaje
Botón C	PB0	Enviar tabla Q para almacenarla tras aprendizaje
LED amarillo	PC7	Señal visual en eventos clave del aprendizaje.
LED verde	PD5	
LED rojo	PB0	
Zumbador	Controlado por Timer4 (salida OC4D)	Señal sonora en eventos clave del aprendizaje.

Tabla 3.3. Pines de interfaz de usuario del Balboa 32U4.

3.4. Actuadores del sistema y *encoders* de cuadratura

El Balboa 32U4 usa como actuadores dos **motores de corriente continua** situados a ambos lados del chasis. Estos motores reciben la señal de control mediante modulación por ancho de pulsos (PWM), lo que permite ajustar su velocidad de forma continua dentro de un rango determinado.

Para medir la respuesta del sistema ante distintas órdenes de control, cada motor cuenta con un ***encoder* de cuadratura magnético** acoplado al eje del motor. Estos *encoders* permiten obtener información sobre la posición angular del eje del motor.

A lo largo de este apartado se describen las características principales de los actuadores, el funcionamiento de los *encoders* y los resultados experimentales obtenidos al analizar la relación entre la entrada de control y la velocidad resultante.

3.4.1. Motores de corriente continua

El sistema Balboa 32U4 emplea dos **motores de corriente continua (DC)** [13], fabricados por **Pololu** (véase Figura 3.3), cada uno de los cuales acciona una de las ruedas laterales del robot. Estos motores están acoplados a una **reductora metálica interna**, cuya **relación real es de 75.84:1**, lo que permite transformar la alta velocidad angular del motor, en un par motor elevado y una velocidad lineal adecuada para tareas de equilibrio.

Figura 3.3. Micro Metal Gearmotor 75:1 [13].

Los motores están conectados a un sistema de **engranajes externos** cuya finalidad es trasmitir el movimiento desde el eje del motor hacia el eje de las ruedas. En este proyecto se ha seleccionado los engranajes con relación **41:25**, que proporcionan un factor de reducción de **1.64:1**, el menor entre las opciones disponibles (véase Tabla 3.4). Esta elección se ha realizado

con el objetivo de **maximizar la velocidad de respuesta del sistema** ante perturbaciones, aspecto clave para mantener el equilibrio del robot.

ENGRANAJE EXTERNO	RELACIÓN DE REDUCCIÓN
49:17	2.88:1
47:19	2.47:1
45:21	2.14:1
43:23	1.87:1
41:25 **(seleccionado)**	1.64:1

Tabla 3.4. Relaciones de reducción de los engranajes disponibles para Balboa 32U4.

Cada motor está conectado a través de una etapa de potencia tipo *H-bridge* integrada en la placa, lo que permite invertir el sentido de giro y regular la velocidad mediante señales PWM generadas por el microcontrolador ATmega32U4.

El control se realiza mediante cuatro pines digitales (véase ANEXO I):

- La dirección de giro se define con los pines **PB1** (motor derecho) y **PB2** (motor izquierdo).
- La velocidad se regula mediante señales PWM aplicadas a los pines **PB5** (motor derecho) y **PB6** (motor izquierdo), generadas por el temporizador 1, como ya se ha comentado en el Apartado 3.2.

De esta forma, el sistema puede modular de forma precisa la velocidad de las ruedas y ajustar su dirección, lo que resulta fundamental para ejecutar las acciones seleccionadas por el algoritmo de aprendizaje por refuerzo de forma segura y eficiente.

3.4.2. *Encoders* de cuadratura

Cada motor del Balboa 32U4 cuenta con un ***encoder* de cuadratura magnético** acoplado al eje del motor (véase Figura 3.4). Estos dispositivos están formados por un disco magnético y **dos sensores de efecto Hall** que generan señales digitales denominadas **canal A** y **canal B**.

Ambas señales presentan forma de onda cuadrada y están desfasadas 90º eléctricamente, lo que permite conocer no sólo la posición angular del eje, sino también la dirección de giro del mismo.

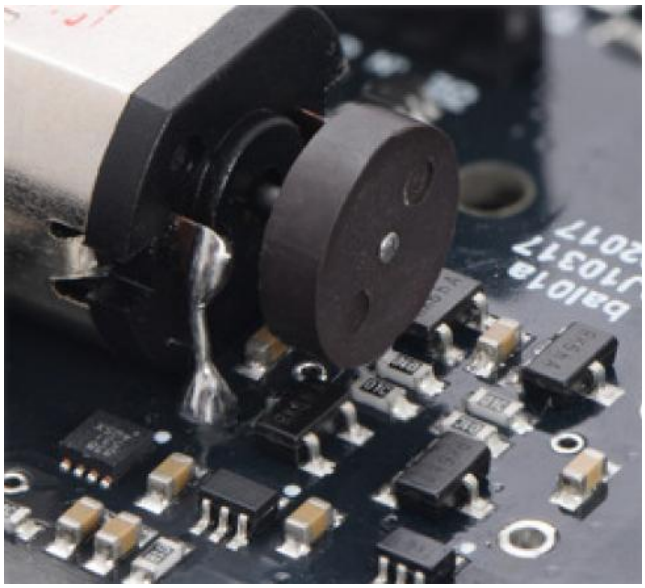

Figura 3.4. *Encoder* de cuadratura [1].

Cuando el motor gira en la dirección *forward* (es decir, la dirección que haría avanzar al robot), el canal B adelanta al canal A, mientras que si el motor gira en la dirección *backward* (es decir, la dirección que haría retroceder al robot) será el canal A el que adelante al canal B.

Debido a las limitaciones en el número de pines con capacidad de interrupción, el sistema combina las señales A y B mediante una puerta lógica XOR, y dirige la señal resultante a un pin con interrupción (véase Figura 3.5). La señal del canal B se mantiene conectada a un pin de lectura normal. Esto permite detectar cambios de estado con alta frecuencia sin saturar al microcontrolador. De esta manera, teniendo dos señales disponibles, se puede incluso reconstruir el canal A mediante una segunda operación XOR (véase Ecuación (3.1)).

$$A = (A \wedge B) \wedge B \tag{3.1}$$

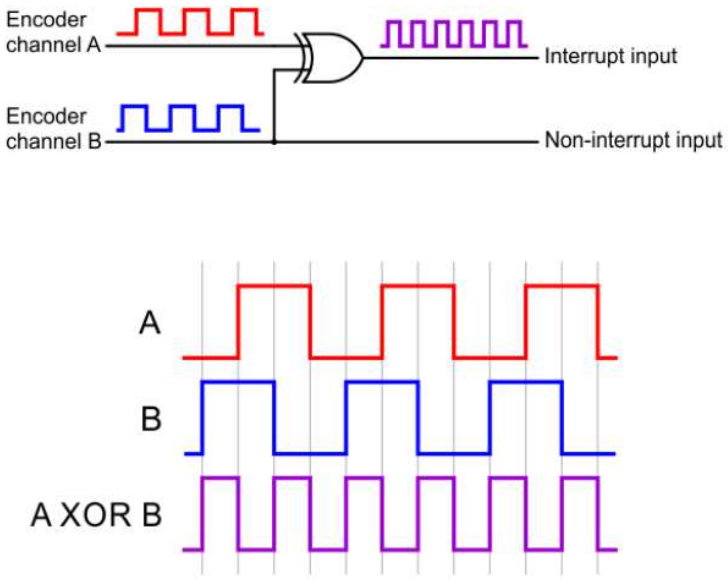

Figura 3.5. Representación gráfica de las señales canal A, canal B y optimización XOR del *encoder* de cuadratura [1].

Respecto a la asignación de pines para la lectura de *encoders*, esta queda reflejada en la Tabla 3.5.

ENCODER	**SEÑAL XOR (CANAL A COMBINADO)**	**CANAL B**	**TIPO DE ENTRADA**
Derecho	PE6	PF0	INT6 (interrupción externa)
Izquierdo	PB4	PE2	PCINT4 (interrupción por cambio)

Tabla 3.5. Asignación de pines para la lectura de *encoders*.

También cabe destacar que, a diferencia de sensores con muestreo periódico, los ***encoders* de cuadratura generan pulsos de forma asincrónica**, cuya frecuencia depende directamente de la velocidad de giro del eje. En este sistema se alcanzan **frecuencias de hasta varios miles de pulsos por segundo** en funcionamiento normal, gestionados eficientemente mediante interrupciones.

Otro detalle importante a tener en cuenta es la **resolución**, y es que cada *encoder* genera 12 pulsos por vuelta completa del eje del motor. Teniendo en cuenta la relación real de la

reductora interna (75.84:1) y la reducción de los engranajes externos (41:25), la resolución total queda como:

$$cuenta\ por\ vuelta = 12 \times 75.84 \times \frac{41}{25} \approx 1965 \qquad (3.2)$$

Es decir, gracias al *encoder* se pueden medir desplazamientos de hasta 0.183º en las ruedas. Esta resolución permite medir con precisión la posición y velocidad angular de las ruedas mediante recuento de pulsos.

Lamentablemente, debido a las **limitaciones de memoria del microcontrolador ATmega32U4, no se ha utilizado la información de los *encoders*** durante el proceso de aprendizaje. Incorporarlo como parte del estado habría generado un espacio de estados excesivamente grande e inasumible para una implementación basada en la tabla Q. Sí se han utilizado para realizar una caracterización experimental de la velocidad de los motores en función de la señal de entrada, como se expone en la Sección 3.4.3. Para este propósito, se ha contado el número de pulsos generados por los *encoders* en intervalos de tiempo que han sido medidos, permitiendo así estimar la velocidad real del sistema con alta precisión.

3.4.3. Relación experimental entrada-velocidad

Con el objetivo de **caracterizar la respuesta dinámica** de los motores del Balboa 32U4, se ha llevado a cabo un experimento específico para analizar la relación existente entre la señal de entrada aplicada a los motores y la velocidad angular obtenida en las ruedas. En este contexto, la entrada se ha definido como un valor numérico entre 0 y 300, el cual representa la magnitud del ciclo de trabajo de la señal PWM enviada al motor. La elección de este rango de valores, así como su justificación funcional y técnica, se detalla en el Capítulo 4.

Para el experimento se ha incrementado progresivamente la entrada en pasos de 10 unidades, manteniéndola constante durante un breve intervalo de tiempo para permitir una medición estable. Durante cada uno de estos intervalos, se ha contabilizado el número de pulsos generados por el *encoder* correspondiente, permitiendo así estimar la velocidad media asociada a cada entrada. La estimación de velocidad se ha realizado aplicando una fórmula basada en la cuenta por vuelta del *encoder* (CPR) registrada durante un tiempo fijo, asumiendo un comportamiento constante en ese intervalo (véase Figura 3.6).

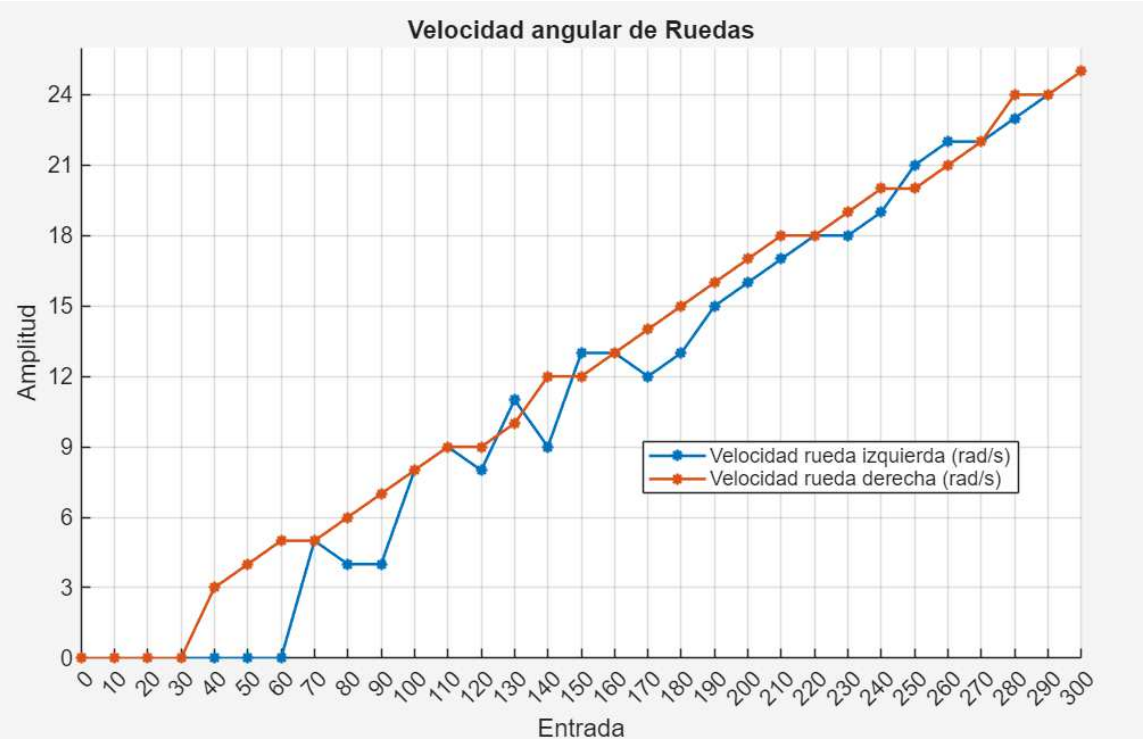

Figura 3.6. Datos obtenidos experimentalmente de la velocidad angular de las ruedas en función de la entrada.

El análisis de los resultados muestra que existe una relación aproximadamente lineal entre la magnitud de la entrada y la velocidad angular resultante, especialmente en el rango medio de operación. Sin embargo, en los extremos más bajos del rango se observa un comportamiento no lineal asociado al vencimiento de la inercia estática y al umbral de activación de los motores, fenómeno habitual en sistemas de este tipo.

Aunque esta información no ha sido utilizada directamente en el proceso de aprendizaje, por tratarse de un método basado en refuerzo sin modelado explícito, ha resultado útil para validar el funcionamiento de los motores, verificar la precisión de las lecturas de los *encoders* y entender la dinámica del sistema. En definitiva, ha servido como base de confianza para el correcto funcionamiento del hardware durante la implementación del algoritmo.

3.5. Sensores inerciales

Los sensores inerciales permiten obtener información sobre el movimiento y la orientación del robot. En el caso del Balboa 32U4, estos sensores son fundamentales para conocer ese estado dinámico, ya que proporcionan datos sobre inclinación y velocidad angular. A continuación, se describen sus características principales y su aplicación práctica en el proyecto.

Cabe mencionar que la placa también incluye un **magnetómetro LIS3MDL** [31], aunque en este proyecto se ha descartado su uso al no ser útil para los objetivos de control y aprendizaje planteados.

3.5.1. Características y respuesta del acelerómetro y giroscopio

El Balboa 32U4 incorpora el sensor inercial **LSM6DS33** [32] (véase Figura 3.7), desarrollado por STMicroelectronics, que integra en un solo encapsulado un **acelerómetro triaxial** y un **giroscopio triaxial**. Este sensor se comunica con el microcontrolador mediante el bus I²C interno de la placa y permite configurar distintos parámetros, como el rango de medida y la frecuencia de muestreo mediante registros específicos accesibles por software.

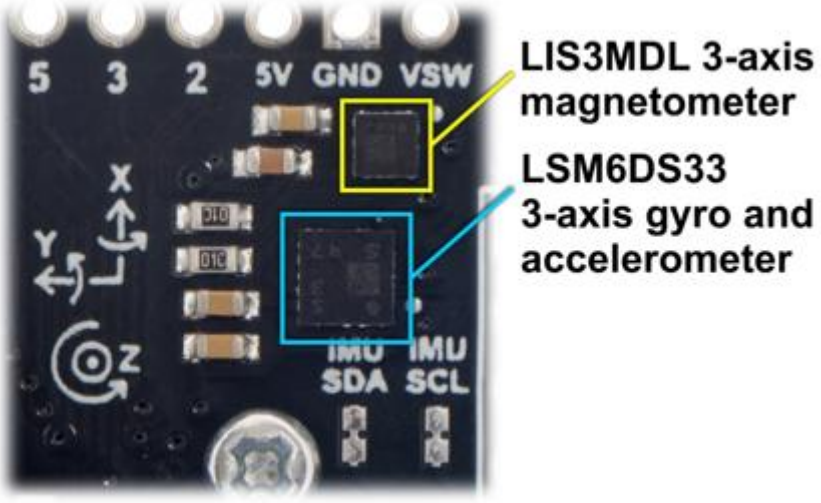

Figura 3.7. Sensores inerciales ubicados en la placa de Balboa 32U4 y su respectiva orientación de ejes [1].

Para estimar de forma inicial el ángulo de inclinación del robot en el plano longitudinal (véase Figura 3.8) y así poder ver el funcionamiento básico del acelerómetro, se han utilizado los **ejes X y Z** del mismo. Aplicando la función arcotangente a la relación entre ambos, es posible calcular la inclinación respecto a la horizontal en condiciones de reposo o aceleraciones suaves (véase Ecuación (3.3).

$$\theta_{acc} = \arctan\left(\frac{a_x}{a_z}\right) \tag{3.3}$$

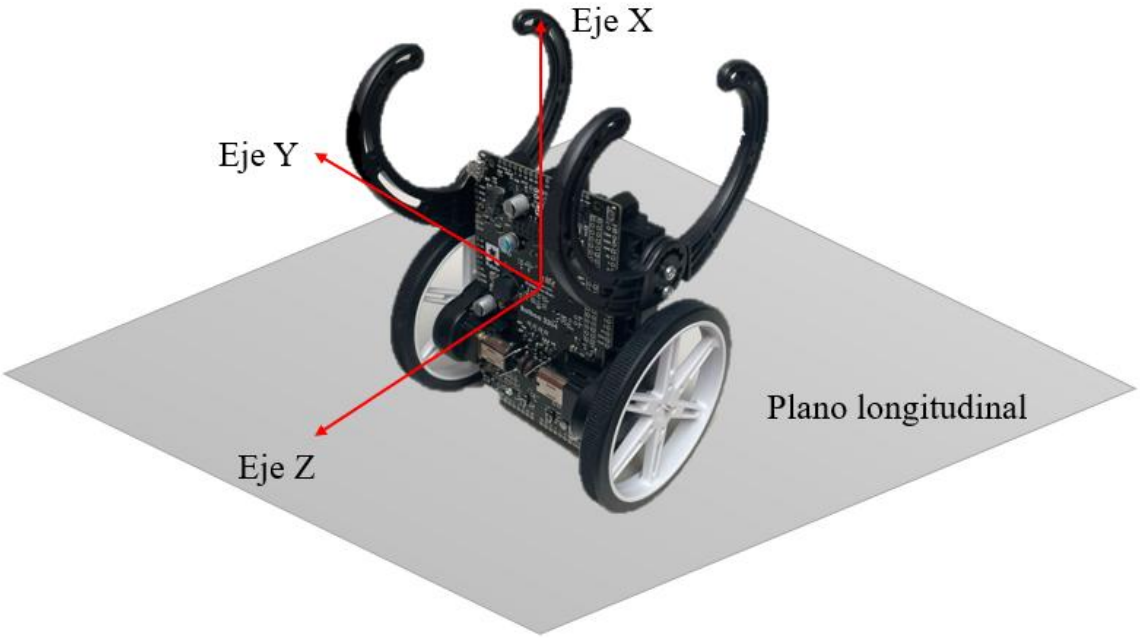

Figura 3.8. Representación de los ejes inerciales en el robot Balboa 32U4

En la ecuación, $\boldsymbol{a_x}$ y $\boldsymbol{a_z}$ representan las aceleraciones medidas en los respectivos ejes en unidades de gravedad (g). Este cálculo resulta válido bajo la hipótesis de que la aceleración medida proviene únicamente de la gravedad, como se analizará en los siguientes apartados. Conviene señalar que, debido a esta formulación, la **posición de equilibrio** (robot completamente vertical) corresponde a un **ángulo de 90º**, respecto a la horizontal. Por lo tanto, en dicha posición de equilibrio, el eje X de los sensores se orienta hacia arriba (véase Figura 3.8).

Por otro lado, para medir la velocidad angular se ha utilizado el eje Y del giroscopio, que registra la rotación del sistema en el plano de inclinación. Una particularidad del **giroscopio es que presenta un *offset* de salida**, es decir, un valor distinto de cero incluso en ausencia de movimiento. Este desplazamiento debe ser medido previamente y restado de cada lectura para

obtener valores precisos. Si se denota como ω_{raw} la salida directa del sensor y ω_{offset} el valor de compensación, la velocidad angular corregida viene dada por la Ecuación (3.4).

$$\omega = \omega_{raw} - \omega_{offset} \tag{3.4}$$

En este proyecto se han elegido configuraciones que permiten una buena resolución y un amplio rango dinámico, capaces de captar desde pequeñas oscilaciones hasta movimientos bruscos. La configuración final de sensores es la presentada en la Tabla 3.6.

SENSOR	EJES USADOS	RANGO SELECCIONADO	RESOLUCIÓN TÍPICA	FRECUENCIA DE MUESTREO
Acelerómetro	X y Z	±16 g	0.488 mg/LSB	1.66 kHz
Giroscopio	Y	±1000 dps (degrees per second)	35 mdps/LSB	1.66 kHz

Tabla 3.6. Parámetros de configuración del LSM6DS33 utilizados en el proyecto.

Tanto el acelerómetro como el giróscopo permiten seleccionar una **amplia gama de frecuencias de muestreo**, conforme a la documentación del fabricante [32]. En este proyecto se ha optado por utilizar 1.66 kHz, que representa la frecuencia máxima disponible para el giroscopio. Si bien el acelerómetro permite configuraciones de hasta 6.66 kHz, se ha decidido mantener ambos sensores funcionando a la misma frecuencia para **garantizar una lectura sincronizada** y coherente de los datos. Para una descripción más detallada de las posibles configuraciones, se recomienda consultar la **hoja de datos oficial del LSM6DS33** proporcionada por Pololu [32].

Para analizar la respuesta de ambos sensores se han registrado sus señales durante una serie de movimientos manuales, evitando impactos o aceleraciones bruscas. El objetivo es observar el comportamiento natural del sistema bajo condiciones controladas.

De esta manera, se han llevado a cabo distintas situaciones:

- **Caso de estudio 1:** Movimiento de un extremo (robot tumbado) a posición de equilibrio (90º) (véase Figura 3.9).

- **Caso de estudio 2:** Movimiento desde la posición de equilibrio a un extremo, para después volver a la posición de equilibrio y finalmente llegar al otro extremo (véase Figura 3.10).
- **Caso de estudio 3:** Movimiento desde un extremo a otro y viceversa (véase Figura 3.11).

Se debe tener en cuenta que las señales del giroscopio han sido procesadas mediante el uso de la Ecuación (3.4); de hecho, se puede observar que, como tan sólo hay rotación respecto al eje Y, **las velocidades medidas en los ejes X y Z son nulas en todos los casos**. De esta manera, aunque estas señales no vayan a ser usadas directamente para el aprendizaje por refuerzo, queda constatado el funcionamiento correcto de los sensores.

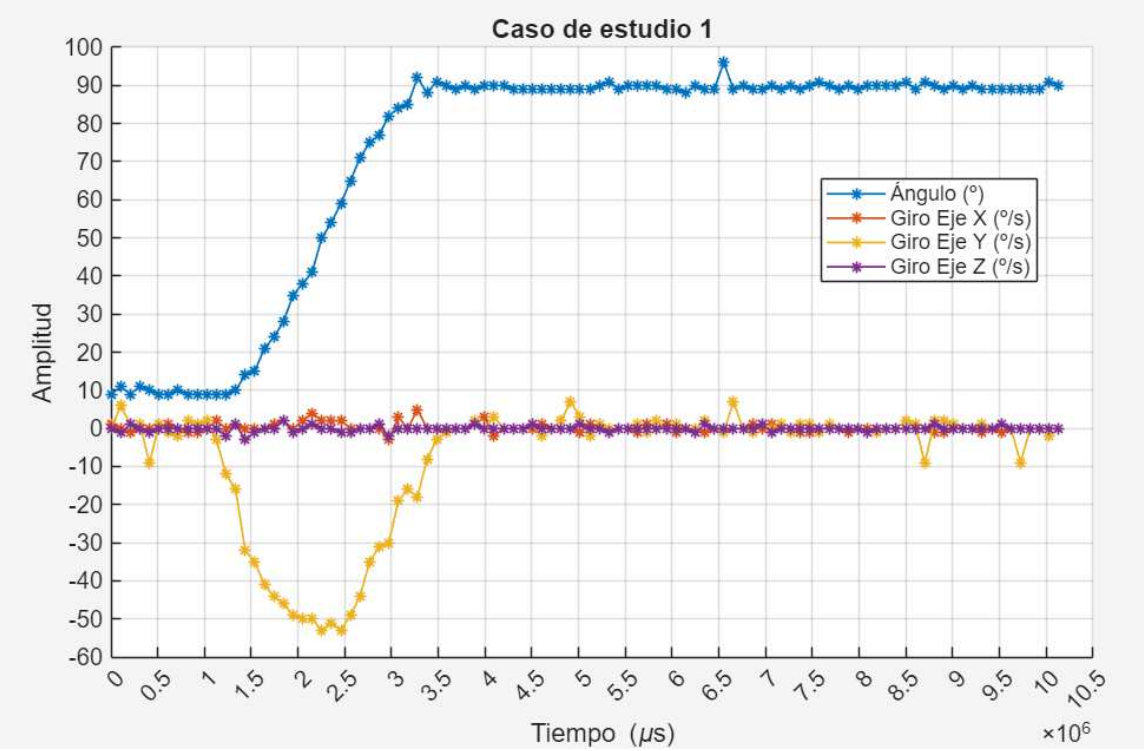

Figura 3.9. Ángulo obtenido del acelerómetro y velocidades de los ejes del giroscopio (Caso de estudio 1).

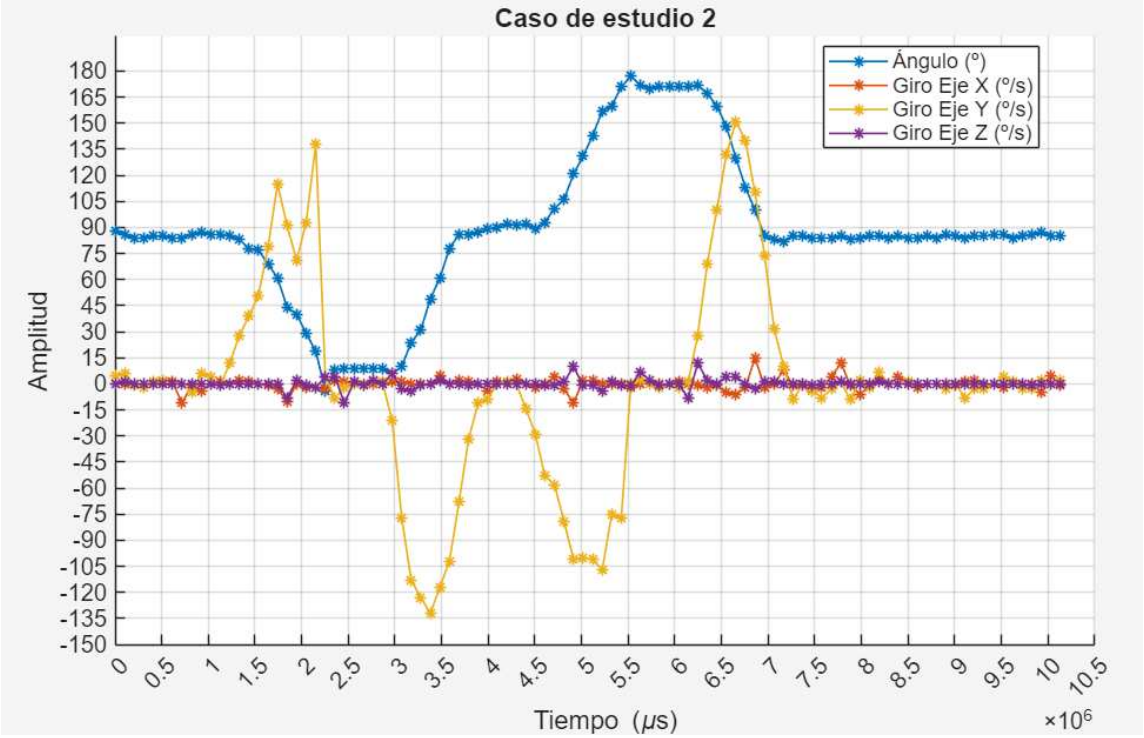

Figura 3.10. Ángulo obtenido del acelerómetro y velocidades de los ejes del giroscopio (Caso de estudio 2).

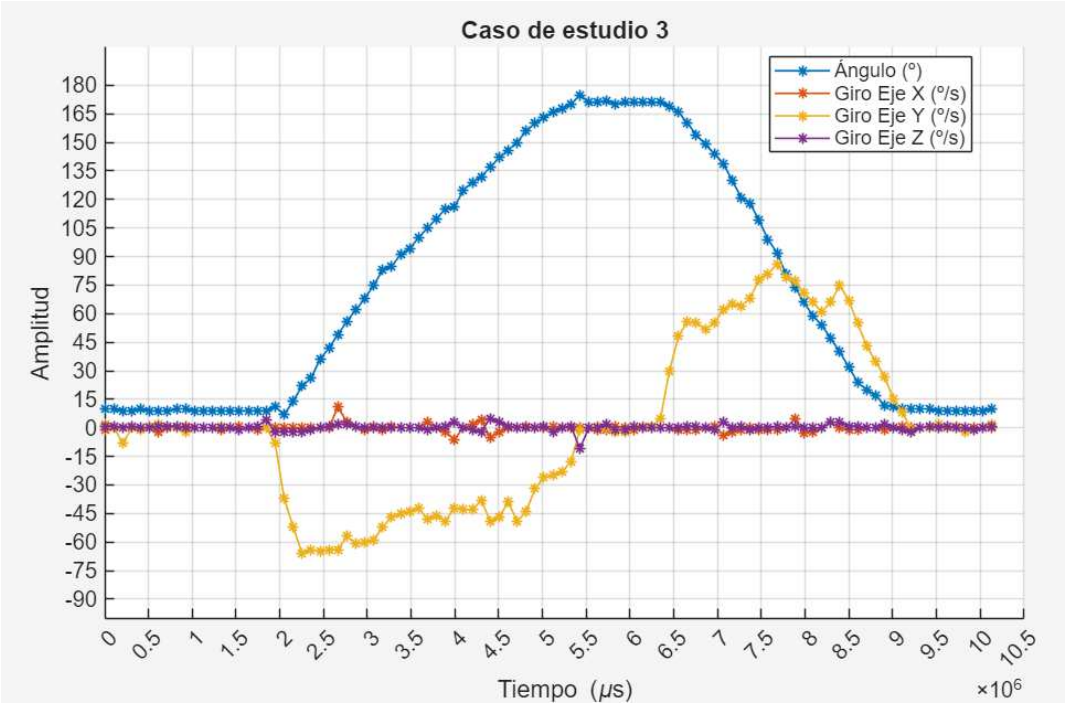

Figura 3.11. Ángulo obtenido del acelerómetro y velocidades de los ejes del giroscopio (Caso de estudio 3).

3.5.2. Obtención del ángulo con integración del giróscopo

Además de estimar el ángulo de inclinación mediante el acelerómetro, otra opción ampliamente utilizada es integrar la **velocidad angular** proporcionada por el giróscopo. Dado que este sensor entrega una medida continua de la rotación en el plano del movimiento, es posible obtener una estimación del ángulo acumulado si se conoce su valor inicial (se puede obtener del acelerómetro) y se dispone de la frecuencia de muestreo del sistema.

El principio fundamental de esta técnica se basa en la expresión de la Ecuación (3.5).

$$\theta(t) = \theta_0 + \int_0^t \omega(\tau) d\tau \tag{3.5}$$

Donde:

- $\theta(t)$ es el ángulo estimado en el instante t,
- θ_0 es el ángulo inicial conocido (se puede medir con el acelerómetro, debido a que se parte del reposo),
- $\omega(\tau)$ es la velocidad angular corregida en cada instante.

En un sistema digital de tiempo discreto como el que se utiliza en este proyecto, esta integral se aproxima mediante una suma de rectángulos (véase Ecuación (3.6)).

$$\theta_{gyro,k} = \theta_{gyro,k-1} + \omega_k \cdot \Delta t \tag{3.6}$$

Donde Δt será la frecuencia de muestreo que, en este caso, será el tiempo de cada uno de los pasos del aprendizaje que se realizará, y ω_k es la lectura de velocidad angular del giroscopio corregida por su *offset*, tal como se explicó en el apartado anterior.

Aunque este método proporciona una estimación continua del ángulo sin depender de la orientación respecto a la gravedad, presenta una limitación fundamental: **la acumulación de error** debida al ruido del sensor, inestabilidades térmicas y pequeñas imprecisiones en la calibración del *offset*. Esta acumulación se manifiesta como una deriva progresiva en la estimación del ángulo, incluso cuando el sistema permanece en reposo.

3.5.3. Limitaciones individuales de cada sensor

El uso individual de los sensores inerciales presenta una serie de limitaciones prácticas que deben tenerse en cuenta a la hora de estimar el ángulo de inclinación de forma fiable. A continuación se analizan los problemas más relevantes observados en este proyecto, tanto en el acelerómetro como en el giróscopo.

En primer lugar, el **acelerómetro**, aunque permite obtener una estimación directa del ángulo respecto a la gravedad mediante una función trigonométrica entre los ejes **X** y **Z**, presenta una señal afectada por **fluctuaciones de alta frecuencia** incluso en condiciones estáticas. Este comportamiento es característico del ruido inherente al sensor, que se ve amplificado cuando se realiza el cálculo del ángulo. Además, en escenarios reales como el proceso de aprendizaje por refuerzo, se espera que el robot realice movimientos bruscos y cambios rápidos de sentido. Estas acciones implican la presencia de **aceleraciones lineales puntuales de gran magnitud** que se superponen a la componente gravitatoria, lo que provoca estimaciones erróneas del ángulo: el sensor interpreta cualquier aceleración como un cambio de inclinación, generando valores que no se corresponden con la orientación real del sistema.

Este fenómeno queda reflejado en la Figura 3.12, donde se muestran los valores de aceleración en los ejes X y Z, junto con el ángulo estimado a partir de dichos datos. Dichos valores han sido tomados mientras se producían manualmente movimientos abruptos en la zona de equilibrio (≈ 90°). Se puede ver que, mientras que la aceleración en el eje Z apenas varía debido a que el robot se encuentra en torno a la posición de equilibrio, la aceleración en el eje X presenta mucho más ruido. Como consecuencia, también existen valores en el ángulo obtenido que no reflejan la realidad, por lo que este método no es válido para llevar a cabo el aprendizaje.

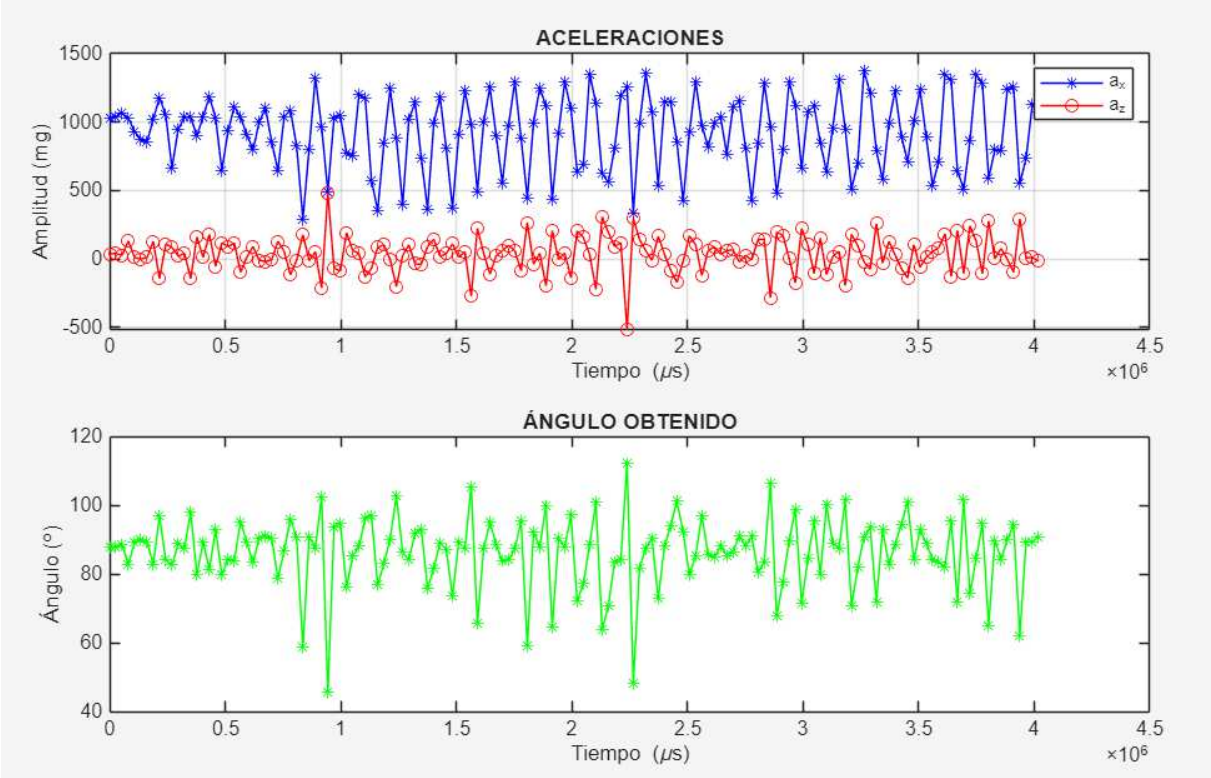

Figura 3.12. Aceleraciones y ángulo obtenidos al realizar movimientos rápidos en torno a la zona de equilibrio.

Por otro lado, el **giróscopo** ofrece una señal más estable en el corto plazo, basada en la velocidad angular. Sin embargo, cuando se emplea el método de integración para obtener el ángulo acumulado, se observa la aparición de una **deriva progresiva** en la estimación (véase Figura 3.13). Como ya se ha comentado, este fenómeno es consecuencia de pequeñas imprecisiones en la medición, ruido blanco, errores de cuantificación y, especialmente, de una calibración imperfecta del *offset* inicial.

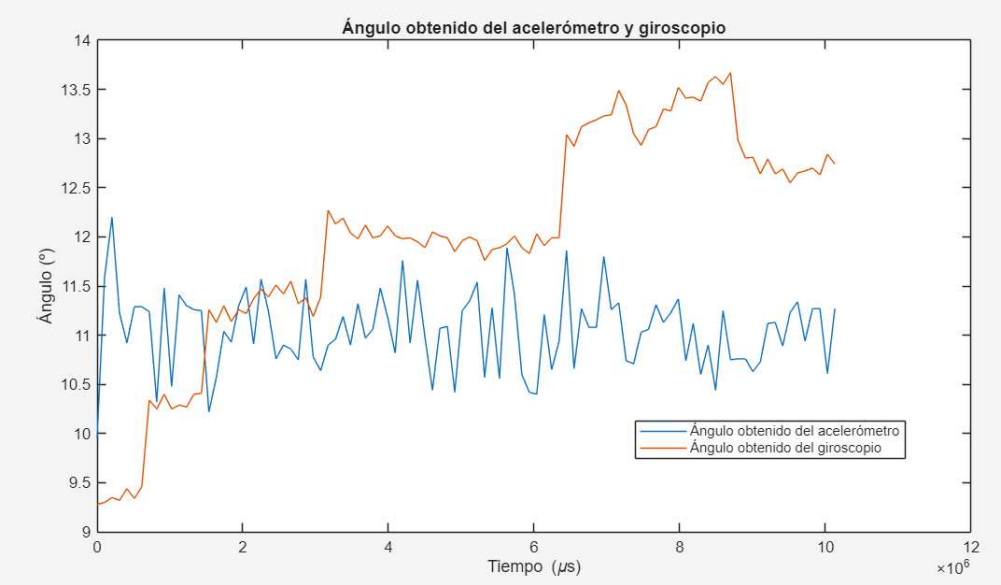

Figura 3.13. Ejemplo de deriva del error presente en la obtención del ángulo mediante integración del giroscopio.

Puede apreciarse que, incluso cuando el robot se encuentra inmóvil, el resultado acumulado de la integración muestra una desviación creciente. Este tipo de error sistemático representa una limitación crítica para aplicaciones donde se requiere precisión absoluta a largo plazo, y justifica la **necesidad de aplicar métodos de corrección o combinación sensorial**. Es por ello que se descartó el uso de la integración del giroscopio para llevar a cabo el aprendizaje.

Estas limitaciones, observadas de forma independiente, motivan la utilización de técnicas de filtrado que permitan aprovechar las ventajas de ambos sensores mientras se mitigan sus respectivas debilidades, como se aborda en los siguientes apartados.

3.5.4. Aplicación de filtros exponencial y complementario

En el análisis anterior se ha evidenciado que el uso individual del acelerómetro o del giróscopo presenta limitaciones importantes particularmente en la implementación de métodos de aprendizaje por refuerzo como Q-Learning que están diseñados para sistemas con observabilidad total. El acelerómetro proporciona una estimación directa del ángulo, pero está fuertemente afectado por ruido de alta frecuencia. El giróscopo, en cambio, ofrece una señal suave y precisa a corto plazo, pero sufre una deriva acumulada debido a pequeñas imprecisiones

y a la integración continua de la velocidad angular. Para compensar estas limitaciones y obtener una estimación del ángulo más fiable y útil para el sistema de aprendizaje, se han aplicado distintas **técnicas de filtrado**.

En primer lugar, se ha utilizado un **filtro exponencial** [33] aplicado sobre el ángulo calculado a partir del acelerómetro. Este filtro atenúa eficazmente el ruido sin requerir un gran coste computacional; su comportamiento depende del coeficiente de suavizado empleado. En los experimentos realizados se han probado **distintas constantes de filtrado**, lo que ha permitido comprobar el compromiso entre **estabilidad y rapidez**: una constante más baja proporciona una señal más limpia, pero con mayor retardo, mientras que una constante más alta permite obtener una señal más rápida, pero deja pasar más ruido.

Posteriormente se ha implementado un **filtro complementario** [34], que consiste en una combinación ponderada de las señales del acelerómetro y del giróscopo. En este caso se ha utilizado de forma constante un **coeficiente de 0.98** para la parte asociada al giróscopo y de 0.02 para la corrección derivada del acelerómetro. Esta elección no solo es habitual en sistemas de fusión inercial por su capacidad para equilibrar respuesta rápida y corrección de deriva, sino que además ha demostrado ofrecer muy buenos resultados experimentales en el sistema implementado. El principal beneficio del filtro complementario es que permite **aprovechar simultáneamente la capacidad del giróscopo para seguir los movimientos rápidos del sistema y la estabilidad a largo plazo que aporta el acelerómetro**, reduciendo tanto el ruido como la deriva. Esto lo convierte en una solución especialmente adecuada para sistemas de control como el presente, donde se requiere una estimación del ángulo precisa, estable y sin retardo perceptible.

Se han llevado a cabo tres pruebas comparativas en las que se han representado conjuntamente el ángulo estimado directamente a partir del acelerómetro, el ángulo resultante del filtro complementario (con un coeficiente fijo de 0.98) y el ángulo obtenido mediante un filtro exponencial utilizando distintos coeficientes de filtrado: 0.05, 0.10 y 0.15 (véanse Figura 3.14, Figura 3.15 y Figura 3.16).

Tal como se puede observar, el filtro complementario ofrece una solución equilibrada, ya que permite suprimir el ruido proveniente del acelerómetro, a la vez que corrige la deriva del giróscopo, proporcionando una estimación estable y con respuesta rápida ante los cambios de

inclinación. En contraste, el filtro exponencial no logra encontrar ese equilibrio: si se utiliza un coeficiente bajo, se obtiene una señal muy suavizada, pero con un retardo evidente; si el coeficiente es más alto, mejora la respuesta dinámica, pero deja pasar un mayor nivel de ruido.

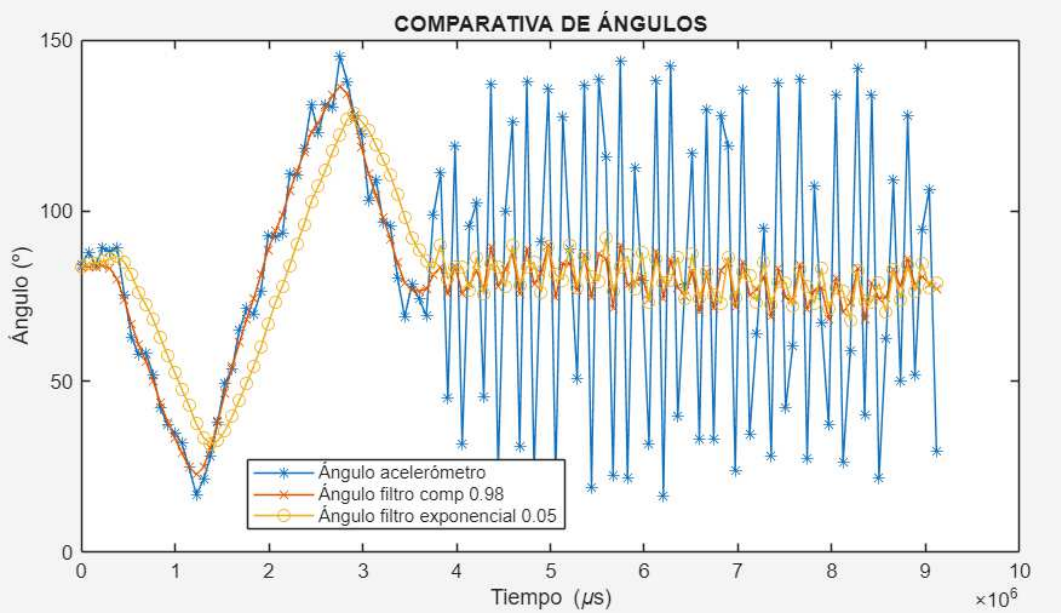

Figura 3.14. Comparativa de ángulos filtrados con constante de filtro complementario de 0.98 y coeficiente de filtro exponencial de 0.05.

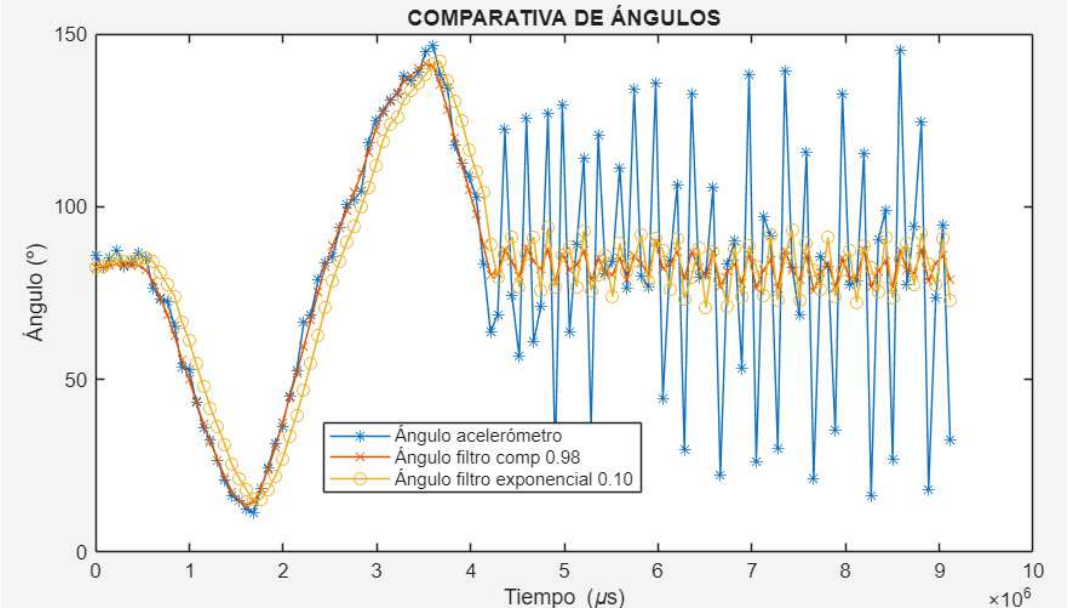

Figura 3.15. Comparativa de ángulos filtrados con constante de filtro complementario de 0.98 y coeficiente de filtro exponencial de 0.10.

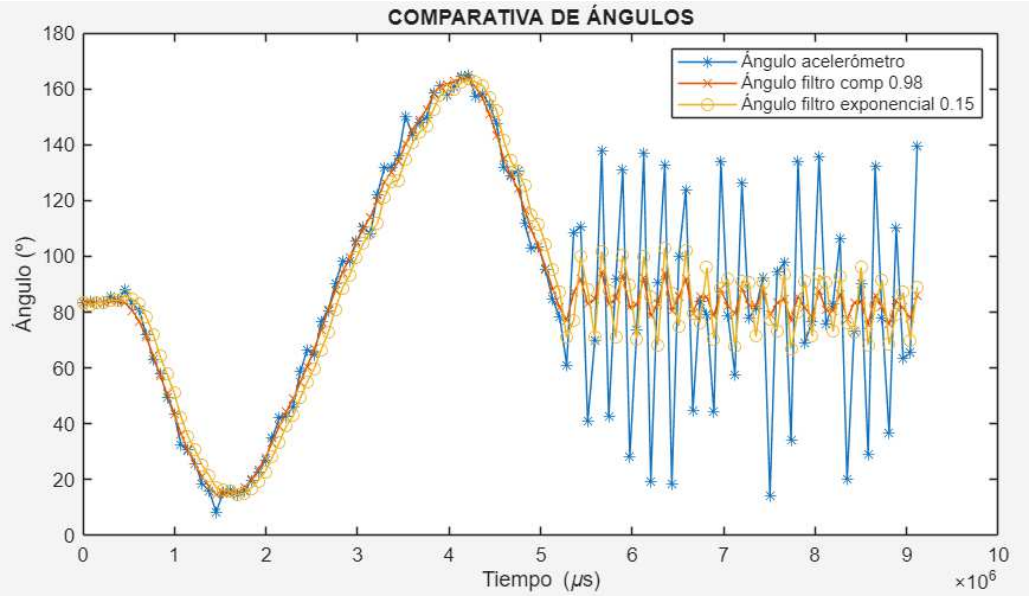

Figura 3.16. Comparativa de ángulos filtrados con constante de filtro complementario de 0.98 y coeficiente de filtro exponencial de 0.15.

El desarrollo matemático y la formulación discreta de ambos filtros se presentan en el ANEXO II.

3.6. Alimentación

El sistema Balboa 32U4 se alimenta mediante **seis baterías recargables tipo AA**, dispuestas en serie y alojadas en el compartimento integrado del chasis (véase Figura 3.17). En este proyecto se han utilizado **pilas HR6 de 1.2 V y 2800 mAh** del fabricante **Powerowl**, lo que proporciona una tensión teórica nominal total de **7.2 V** (aunque en la práctica puede llegar a ser mayor) y una capacidad adecuada para alimentar tanto el microcontrolador como los motores durante sesiones prolongadas de entrenamiento.

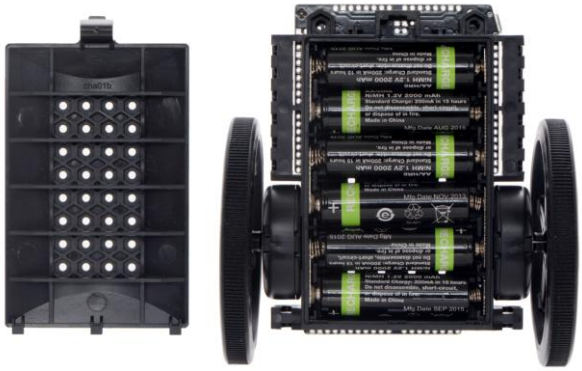

Figura 3.17. Sistema de alimentación del robot Balboa 32U4 basado en baterías recargables [1].

La elección de pilas recargables responde a criterios de sostenibilidad, reutilización y coste, además de facilitar la realización de múltiples pruebas sin necesidad de sustitución continua. El sistema de alimentación incluye un regulador conmutado, lo cual garantiza una tensión estable para los componentes sensibles del sistema.

Con el objetivo de estimar la **duración operativa real de la batería**, algo importante puesto que los experimentos de aprendizaje por refuerzo se realizarán en vivo y serán numerosos, se ha realizado un experimento en el que los motores se mantuvieron en funcionamiento continuo mientras se monitorizaba la **tensión de alimentación (en milivoltios).** En la Figura 3.18 se muestra la evolución de esta tensión a lo largo del tiempo.

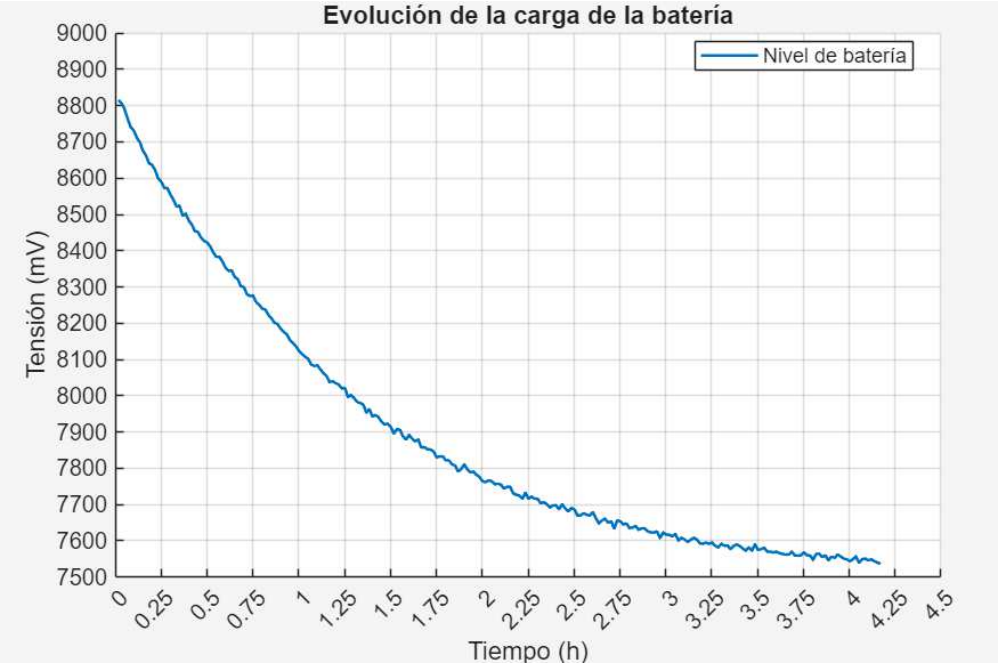

Figura 3.18. Duración operativa real de la batería.

Los resultados muestran una **caída progresiva de tensión** desde aproximadamente 8800 mV hasta valores en torno a 7500 mV tras 4 horas de actividad. Aunque el sistema sigue siendo funcional en estos niveles, se observa una **disminución progresiva del rendimiento**, especialmente en aspectos como la velocidad máxima alcanzable por los motores. Esta bajada de prestaciones no impide el funcionamiento general del sistema, pero debe tenerse en cuenta en pruebas prolongadas o en condiciones exigentes de equilibrio.

4. Software usado en el aprendizaje

Para desarrollar e implementar el sistema de aprendizaje por refuerzo descrito en este trabajo se ha utilizado un conjunto de herramientas software que han permitido programar el microcontrolador, gestionar la comunicación con el exterior y analizar los resultados obtenidos. A diferencia del Apartado 1.4, donde se describieron brevemente estos recursos como parte del entorno general del proyecto, en este capítulo se detalla su papel funcional dentro del proceso de aprendizaje.

Con el objetivo de facilitar la validación del sistema, se ha creado un repositorio público en GitHub que contiene el código fuente desarrollado durante el proyecto. Dicho repositorio incluye las implementaciones en Arduino, los programas de comunicación en *Processing* y las herramientas de análisis en MATLAB.

Para una descripción detallada de los *scripts* incluidos y su funcionalidad, se recomienda consultar el ANEXO III**.**

4.1. Arduino IDE

El presente apartado describe el papel del entorno Arduino IDE [15] dentro del sistema de aprendizaje por refuerzo, comenzando por la integración del microcontrolador ATmega32U4 en dicho entorno, seguido del uso de las librerías específicas del fabricante [16] y [17], las decisiones de implementación, y aspectos técnicos como la gestión precisa del tiempo.

4.1.1. Integración con el microcontrolador

La elección del entorno de programación Arduino como base para el desarrollo se basa en su total compatibilidad con el microcontrolador ATmega32U4 integrado en la placa Balboa 32U4, así como en su ligereza y facilidad de uso. Su entorno simplificado permite cargar y probar código directamente en el robot sin herramientas externas, lo que resulta especialmente útil en sistemas con recursos limitados. Además, la existencia de librerías oficiales específicas para esta plataforma ha permitido acceder al hardware de forma directa y eficiente, tal como se describe en el siguiente apartado.

4.1.2. Acceso al hardware mediante librerías oficiales

Para facilitar la interacción con los componentes integrados en la placa Balboa 32U4, tal como ya se ha comentado, se han utilizado dos librerías oficiales proporcionadas por Pololu: la librería principal *Balboa32U4* [16] y la librería *LSM6* [17] para el sensor inercial. Ambas están diseñadas específicamente para el entorno Arduino y permiten acceder al hardware de forma sencilla y eficiente, sin necesidad de programar directamente a bajo nivel.

Librería Balboa32U4

Esta librería proporciona clases específicas para controlar los motores, leer *encoders*, gestionar los botones de usuario, encender o apagar los LEDs y reproducir sonidos a través del zumbador. Entre las funciones más utilizadas destacan:

- **Motores:** `setSpeeds()`, `setLeftSpeed()`, `setRightSpeed()` para controlar la velocidad y dirección de giro mediante señales PWM. En este proyecto se ha utilizado principalmente `setSpeeds()`, ya que permite modificar simultáneamente la velocidad de ambas ruedas. Tal como se comentó en la Sección 3.4.3, estas funciones aceptan valores en el rango de -300 a 300, donde el signo determina la dirección de giro.
- ***Encoders*:** `getCountsLeft()`, `getCountsRight()` para leer la posición angular de las ruedas y funciones adicionales para reiniciar y comprobar errores. No se

profundizará en estas funciones, porque los *encoders* de cuadratura no forman parte del aprendizaje debido a limitaciones de memoria.

- **Botones:** `isPressed()` o `waitForButton()` son funciones que se han usado en el proyecto para controlar el flujo de trabajo.
- **LEDs:** `ledRed()`, `ledGreen()` y `ledYellow()` son las funciones empleadas para señalizar eventos o estados del sistema.
- **Zumbador:** `playFrequency()` ha sido utilizada para emitir avisos acústicos durante la ejecución del algoritmo.

Estas funciones se integran con los recursos del microcontrolador y aprovechan los temporizadores internos sin requerir configuración manual, lo cual ha simplificado la implementación del sistema.

Para más información y detalles completos de esta librería, se recomienda consultar la documentación oficial disponible en [16].

Librería LSM6

El sensor inercial del Balboa 32U4, modelo LSM6DS33, incorpora un acelerómetro y un giroscopio triaxiales. Para su lectura se ha utilizado la librería LSM6, también proporcionada por Pololu.

Las principales funciones empleadas han sido:

- `imu.init()`: inicializa el sensor con la configuración por defecto.
- `imu.writeReg()`: permite modificar los registros para configurar la resolución y frecuencia de muestreo deseados.
- `imu.read()`: actualiza los valores del sensor, que luego pueden consultarse mediante atributos como `imu.a.x`, `imu.a.z` o `imu.g.y`.

El uso de esta librería ha permitido capturar en tiempo real las señales necesarias para estimar el ángulo de inclinación del robot, utilizando tanto métodos individuales como técnicas de fusión sensorial descritas en el Apartado 3.5.

Para más información y detalles completos de esta librería, se recomienda consultar la documentación oficial disponible en [17].

Cabe destacar que estas librerías han sido también las utilizadas para capturar las señales reales mostradas en el Capítulo 3, como las respuestas del sistema a los comandos de motor, las lecturas del sensor inercial y la evolución de la batería. Su uso ha permitido obtener datos precisos y representativos del comportamiento físico del robot durante las distintas fases del proyecto.

4.1.3. Estructura del algoritmo y detalles de implementación

La implementación del algoritmo Q-Learning en el entorno Arduino se estructura en dos fases principales: la inicialización (`setup()`) y el ciclo principal (`loop()`), donde se lleva a cabo el proceso de aprendizaje (véase Figura 4.1).

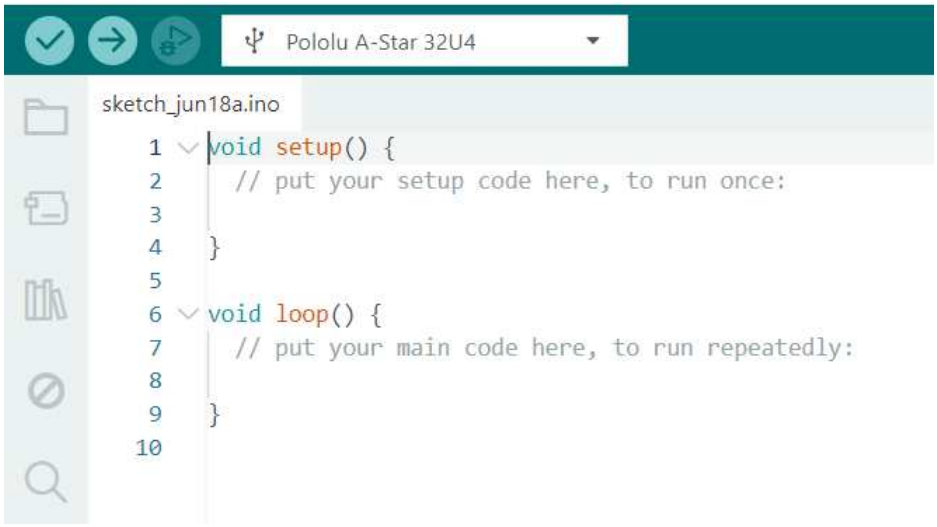

Figura 4.1. Entorno de desarrollo Arduino con estructura básica de programa.

En la función `setup()` se realiza la configuración inicial del sistema. En esta fase el sistema permitirá al programador decidir si desea iniciar un nuevo entrenamiento desde cero o continuar uno anterior, mediante los botones de la interfaz de usuario. En función de esta decisión, se procede a **inicializar la tabla Q con el valor -10000 o a cargarla desde almacenamiento externo**. Esta lógica permite preservar el conocimiento aprendido entre sesiones o reiniciar el proceso de forma limpia. Además, durante esta fase se configuran los

registros del sensor inercial (IMU) y se realiza una **medición del *offset* del giroscopio**, necesaria para asegurar la precisión en la lectura de la velocidad angular durante el entrenamiento.

Una vez completada la inicialización, se entra en el ciclo `loop()`, donde se llama a una función `learn()`. Esta función ha sido diseñada en este TFM para ejecutar **10 episodios de aprendizaje** consecutivos por cada llamada. Al finalizar estos 10 episodios, el sistema realiza la **exportación de datos clave**: la tabla Q y la recompensa y número de pasos por episodio. Esta frecuencia de exportación se ha elegido por una cuestión de **eficiencia y seguridad**: dado que los procesos de carga de datos, envío por puerto serie y almacenamiento son relativamente lentos en un entorno empotrado, se ha optado por dividir el entrenamiento en bloques de 10 episodios. Esto permite **recuperar los datos intermedios con regularidad y evitar posibles pérdidas** en caso de reinicio inesperado o bloqueo.

Si bien la tabla Q tiene un tamaño fijo y no se ve afectada por el número de episodios, los vectores que almacenan la recompensa acumulada y el número de pasos por episodio sí **crecen linealmente** con cada episodio adicional. Dado que el sistema dispone de una memoria limitada, esta estrategia también permite **controlar el uso de memoria** y mantener el funcionamiento estable.

La **tabla Q** se define como una matriz de tipo `float`, donde cada entrada representa el valor estimado de una acción en un estado determinado. Para los casos en los que el estado combina múltiples variables (por ejemplo, ángulo y velocidad de giro), se utiliza una **función de mapeo bidimensional** que convierte dicha combinación en un índice único dentro de la tabla. Esta función se encuentra documentada en el repositorio de código adjunto (véase ANEXO III).

Una de las decisiones más relevantes ha sido la ya mencionada **inicialización de la tabla Q con el valor -10000**. Esto se hace para evitar falsos máximos durante las primeras etapas del aprendizaje: un estado-acción no visitado podría parecer inicialmente tan válido como uno que haya obtenido un valor de 0 tras aprender, lo cual afectaría negativamente al proceso de exploración. En este entorno, las recompensas pueden ser negativas, por lo que es fundamental distinguir entre valores penalizados por experiencia y aquellos aún no explorados. Para evitar que este valor negativo interfiera en la actualización real de la tabla, **cuando se aplica la**

fórmula del algoritmo Q-Learning (véase Ecuación (2.11))**, el valor de la celda que contiene -10000 se sustituye por 0** para el cálculo. De esta forma, se logra mantener el efecto disuasorio durante la selección de acciones, pero sin perjudicar la actualización progresiva de los valores a partir de recompensas reales observadas.

Por otro lado, la elección del valor -10000 no ha sido arbitraria. Dado que el valor máximo y mínimo que puede alcanzar un elemento de la tabla Q puede obtenerse mediante la Ecuación (4.1), la cual se basa en los parámetros del algoritmo (recompensa máxima y factor de descuento), se ha verificado que, en las condiciones utilizadas en este proyecto, **ese valor nunca será alcanzado**. Aunque los valores concretos de los parámetros de aprendizaje se describen en detalle en el Capítulo 5, se deja constancia del motivo por el que se ha seleccionado dicho valor.

$$Q_{max} = \frac{R_{max}}{1-\gamma} \tag{4.1}$$

En nuestro sistema, la recompensa se asocia al estado alcanzado tras ejecutar una acción, y no al ángulo ni a la acción de forma aislada. Esta decisión simplifica la lógica de implementación y facilita la evaluación de la política aprendida en función de su capacidad para alcanzar estados deseados o evitar situaciones de fallo.

Por otro lado, las decisiones relacionadas con la política de comportamiento, la función de recompensa y la representación del espacio de estados se detallan en el Capítulo 5, centrado en el diseño lógico del sistema de aprendizaje. Lo único que se debe tener claro es que se usa una política ϵ-*greedy*, y que el valor del parámetro ϵ se puede modificar si se desea cuando se terminan las series de 10 episodios de entrenamiento.

Además de la versión del programa orientada al entrenamiento, se ha implementado una variante específica para la fase de **explotación de la política aprendida**. En esta versión se parte de una tabla Q previamente entrenada que se carga desde almacenamiento externo. Para garantizar una ejecución puramente determinista, se fija el valor del parámetro ϵ a cero, eliminando por completo la exploración aleatoria. Asimismo, se **desactiva la actualización de la tabla Q**, de modo que las decisiones tomadas no afectan al conocimiento previamente adquirido. Esta variante permite evaluar el comportamiento del sistema bajo una política fija, y

ha sido utilizada para analizar la estabilidad y eficacia del aprendizaje en situaciones reales de explotación.

4.1.4. Gestión del tiempo mediante polling

Una parte fundamental de la implementación del aprendizaje por refuerzo en sistemas embebidos es la correcta gestión del tiempo entre pasos de entrenamiento. La duración de este período es especialmente crítica en algoritmos con estados y acciones discretos, dado que un valor u otro cambian la cantidad de información que evoluciona en el sistema tras cada paso de aprendizaje. Para caracterizar este aspecto se diseñó un experimento en el que se midió el tiempo que tarda en ejecutarse cada paso de un episodio de aprendizaje cuando se ejecutaban secuencialmente sin esperas intermedias. Para ello, se utilizó la función micros() [35] de Arduino, que permite obtener marcas de tiempo en microsegundos, con la que se calculó la duración de cada iteración. Posteriormente, los resultados se exportaron y se representaron gráficamente para su análisis.

Inicialmente, se observó que un paso de aprendizaje completo (es decir, aplicar una acción, esperar el resultado y observar el nuevo estado, llevando a cabo la actualización de la tabla Q) tenía una duración aproximada de 2900 µs (véase Figura 4.2), sin introducir ningún tipo de retardo artificial. Sin embargo, la varianza existente es relativamente alto, y esto podía generar inconsistencias en la dinámica del aprendizaje, especialmente al comparar resultados o analizar la influencia del tiempo entre acción y percepción del nuevo estado.

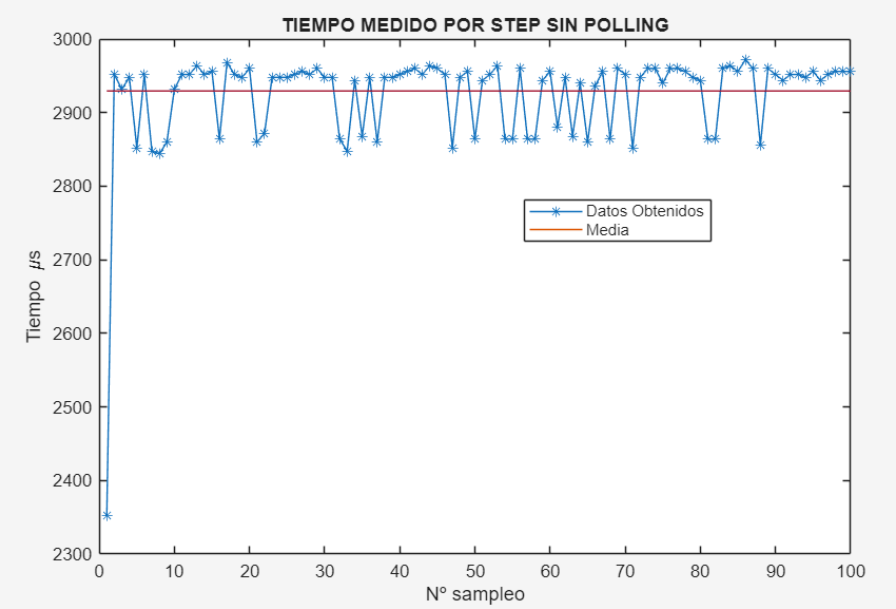

Figura 4.2. Tiempo medido por paso de aprendizaje sin polling.

Para solucionar este problema, se implementó un mecanismo de **espera activa** [36], que garantiza una duración mínima estable por paso. Concretamente, se aplica una **espera activa de 3 ms**, de forma que el sistema permanece en espera activa hasta que se alcanza este tiempo mínimo antes de pasar al siguiente paso del aprendizaje (véase Figura 4.3). Este control se realiza dentro de la propia función `learn()`.

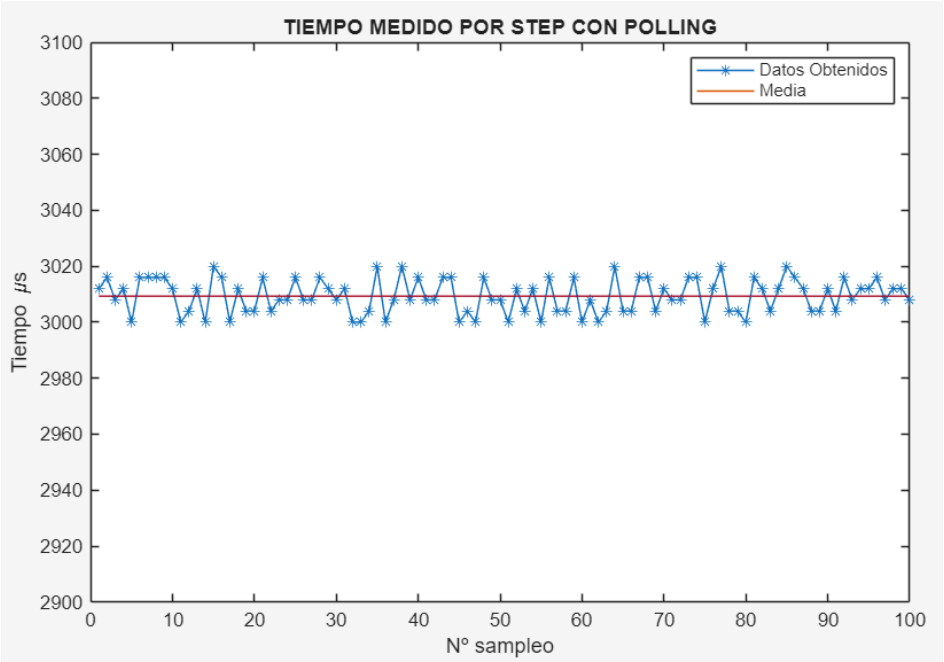

Figura 4.3. Tiempo medido por paso de aprendizaje utilizando técnica de polling.

Tal y como se puede observar, al aplicar la espera activa, la varianza existente se ha reducido drásticamente, consiguiendo así mucha más precisión y repetibilidad.

Además, para poder estudiar la influencia del retardo entre la ejecución de una acción y la lectura del nuevo estado (cuestión que se analiza en los Capítulos 5 y 6), se introdujo un **retardo adicional configurable**. De esta forma, el sistema espera durante $3\,ms\, +\, \Delta t$, donde Δt representa el retardo adicional entre acción y percepción. Esto ha permitido realizar comparaciones experimentales entre diferentes configuraciones temporales, y evaluar cómo afectan al proceso de aprendizaje. Para verificar el funcionamiento de la espera activa junto con el retardo adicional, se repitió el experimento anterior utilizando un $\Delta t = 10\,ms$. El efecto de esta modificación puede observarse en el incremento de la escala temporal en la Figura 4.4, donde se muestra la respuesta obtenida.

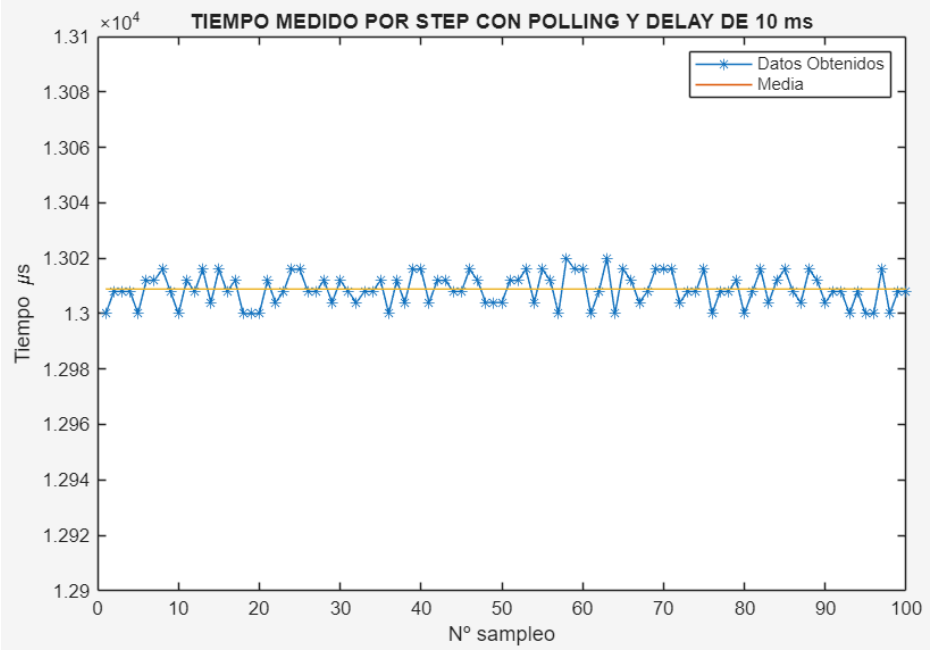

Figura 4.4. Tiempo medido por paso de aprendizaje utilizando técnica de espera activa y agregando un retardo de 10 ms.

Estas gráficas permiten comprobar que la solución implementada estabiliza el sistema y proporciona una base temporal consistente para el aprendizaje. La gestión del tiempo mediante espera activa ha resultado ser una herramienta fundamental para asegurar la repetibilidad de los experimentos y permitir un análisis riguroso del comportamiento del agente ante variaciones en la dinámica del entorno.

Además de utilizarse para estabilizar la duración de cada paso de aprendizaje, los datos temporales obtenidos mediante `micros()` han sido empleados también en otros procesos clave del sistema. Concretamente, se han utilizado para llevar a cabo la integración temporal de la velocidad angular proporcionada por el giroscopio, necesaria para obtener el ángulo de inclinación en el robot. Esta integración es un componente fundamental del filtro complementario, descrito en el Capítulo 3, que fusiona la información procedente del acelerómetro y el giroscopio para estimar el ángulo de forma precisa y robusta.

4.1.5. Limitaciones de memoria y uso de recursos

El entorno Arduino proporciona, al finalizar la compilación del programa, un resumen del uso de memoria tanto en almacenamiento de programa (*flash*) como en memoria de datos (SRAM). En la Figura 4.5 se muestra el mensaje obtenido para la versión final del código, donde puede observarse que el sistema utiliza **1958 bytes de los 2560 disponibles**, es decir, **un 76% de la memoria de datos total**, quedando solo **602 bytes libres** para variables locales.

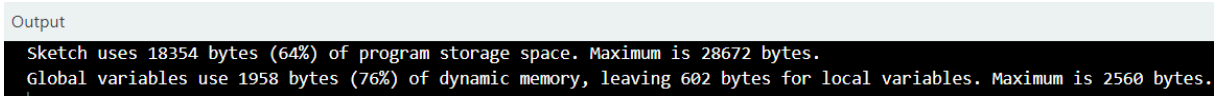

Figura 4.5. Estimación del uso de memoria de datos y de programa obtenida tras la compilación del programa de entrenamiento en Arduino.

Aunque este resultado indica que aún queda un margen de memoria libre, deben tenerse en cuenta dos consideraciones importantes:

- En primer lugar, el sistema **no se comporta de forma completamente estable al alcanzar niveles de uso superiores al 90%.** Durante las pruebas se ha comprobado que, aunque la compilación es exitosa, el comportamiento del microcontrolador puede volverse impredecible o inestable, presentando bloqueos aleatorios, reinicios espontáneos o fallos en la comunicación serie. Estos efectos indican que el sistema opera en condiciones cercanas al límite de sus recursos disponibles, lo que compromete su fiabilidad debido a desbordamiento de pila.

- En segundo lugar, **el número de filas de la tabla Q crece exponencialmente** con cada variable adicional incorporada al estado. Por ejemplo, si se hubieran añadido los *encoders* como parte del vector de estado, discretizando estos en tan solo 3 intervalos, el número de estados posibles se habría multiplicado, y también el tamaño de la tabla Q. Este aumento habría imposibilitado el funcionamiento del sistema.

Estas limitaciones han condicionado el diseño, obligando a realizar un uso muy cuidadoso de los recursos disponibles. Entre otras decisiones, se destacan:

- Dividir el entrenamiento en bloques de 10 episodios, evitando el crecimiento descontrolado de los vectores de recompensas y pasos.
- Restringir el espacio de estados a variables esenciales para el aprendizaje (como ángulo y velocidad de giro), dejando fuera señales adicionales que, aunque útiles, supondrían un coste en memoria no asumible.
- Exportar periódicamente los datos acumulados para liberar y evitar almacenamientos prolongados.

Cabe destacar que el resultado mostrado en la Figura 4.5 corresponde al código completo del sistema de **aprendizaje**, que es el que presenta un mayor uso de memoria debido a la inclusión de funciones como la actualización de la tabla Q, la gestión de recompensas y la exportación periódica de datos. Otros modos de ejecución, como el de explotación, tienen un impacto menor en los recursos disponibles, ya que, **para facilitar el uso eficiente de memoria en esta fase,** se diseñó un tratamiento específico sobre la tabla Q que permite convertirla en una política determinista reduciendo drásticamente su tamaño

Este proceso, realizado previamente en MATLAB, consiste en convertir el valor máximo de cada fila en un 1 y el resto en 0, lo que permite reinterpretar la tabla como una matriz de tipo `uint8_t`.

4.2. *Processing*

Processing [19] es un entorno de desarrollo **basado en Java** ampliamente utilizado en áreas como el diseño interactivo, la visualización de datos y el arte generativo. Su sencillez sintáctica y su integración directa con gráficos, sensores y comunicación por puerto serie lo han convertido en una herramienta muy popular tanto para proyectos educativos como para prototipos rápidos. Aunque no fue diseñado específicamente para aplicaciones de robótica o aprendizaje automático, **su flexibilidad lo hace adecuado para gestionar flujos de datos entre dispositivos empotrados y un ordenador personal**.

En este proyecto se valoraron diferentes alternativas para gestionar el intercambio de datos entre el robot y el entorno de análisis externo. Inicialmente se realizaron pruebas con Python, dada su potencia y versatilidad para el tratamiento de datos. Sin embargo, los primeros resultados satisfactorios se obtuvieron más rápidamente con *Processing*, cuya **arquitectura orientada a *sketchs*** y su biblioteca serie integrada permitieron una implementación ágil y eficaz del sistema de comunicación. Esta elección permitió centrar los esfuerzos de desarrollo en el algoritmo de aprendizaje, sin dedicar tiempo excesivo a resolver problemas de compatibilidad o configuración.

El papel de *Processing* en el sistema ha sido doble. Por un lado, actúa como **puente de exportación**: recibe a través del puerto serie los datos enviados por el robot tras cada bloque de 10 episodios de entrenamiento. Estos datos incluyen la tabla Q completa, la recompensa acumulada por episodio y el número de pasos por episodio. Para su almacenamiento, se utiliza un archivo de texto en el que cada elemento se escribe en una línea independiente, siguiendo el siguiente orden: primero los elementos de la tabla Q (recorrida secuencialmente), luego las recompensas y finalmente los pasos. Este formato lineal y separado por saltos de línea ha facilitado tanto la lectura posterior desde MATLAB como la posibilidad de recuperar los datos de forma precisa.

Por otro lado, *Processing* también cumple la función de **cargador externo de datos**. En sesiones posteriores de entrenamiento o explotación, se ha utilizado para leer un archivo de texto previamente guardado y reenviar los datos al microcontrolador, restaurando así la tabla Q y el estado del aprendizaje sin necesidad de comenzar desde cero. Esta funcionalidad ha

resultado esencial para validar el comportamiento del sistema entrenado, así como para explorar configuraciones alternativas sin repetir todo el proceso desde el inicio.

Para mantener sincronizados los datos entre los distintos *scripts* desarrollados en *Processing* (uno para recibir datos y otro para enviarlos), se ha utilizado el comando `mklink` [37] de Windows, que permite crear enlaces simbólicos entre archivos. De esta forma, aunque cada *sketch* se encuentre ubicado en una carpeta diferente y acceda a su propio archivo de texto, ambos están realmente vinculados al mismo archivo físico en disco. Esta solución ha simplificado el mantenimiento de los datos persistentes del sistema, asegurando que el contenido exportado por Arduino y el utilizado posteriormente para reiniciar el aprendizaje coincidan en todo momento.

4.3. MATLAB

MATLAB [18] es un entorno de programación de alto nivel ampliamente utilizado en ingeniería, matemáticas aplicadas y análisis de datos. Su potencia radica en la facilidad con la que permite manipular matrices, realizar cálculos numéricos complejos y representar gráficamente grandes volúmenes de información. Estas capacidades lo convierten en una herramienta ideal para el procesado y visualización de datos procedentes de sistemas embebidos, como el empleado en este proyecto.

En el marco del presente trabajo, MATLAB ha desempeñado un papel fundamental en dos frentes principales: la **representación de resultados** y la **gestión de datos exportados**. Todas las gráficas incluidas a lo largo de esta memoria se han generado con MATLAB a partir de los ficheros .txt generados por *Processing*. La lectura de dichos archivos se realiza de forma secuencial, por ejemplo, para el caso del aprendizaje, reconstruyendo primero la tabla Q, seguida del vector de recompensas acumuladas y finalmente el número de pasos por episodio.

Para otros casos, como las gráficas temporales de la espera activa o las señales de los sensores inerciales, se ha seguido el mismo formato secuencial, utilizando vectores que contienen los datos de interés. Esta estructura lineal ha permitido automatizar el análisis de los datos, generar gráficos comparativos y validar visualmente el comportamiento del sistema de aprendizaje.

Además de su papel en la representación gráfica y la gestión de datos, MATLAB también se ha utilizado para transformar la tabla Q antes de cargarla en el microcontrolador, con el objetivo de reducir el uso de memoria en la fase de explotación. Esta transformación se describe con más detalle en la Sección 4.1.5.

Por razones de claridad y organización, en el repositorio de código asociado a este proyecto (véase ANEXO III) solo se ha incluido un ***script* de ejemplo de representación gráfica:** uno correspondiente a la fase de aprendizaje y otro a la fase de explotación, evitando así ser redundante con los numerosos gráficos ya incluidos en el documento.

El resto de *scripts* utilizados durante el proyecto estarán presente en dicho repositorio, incluyendo aquellos destinados a:

- Simplificar la tabla Q para su uso en modo explotación y minimizar así el uso de memoria.
- Registrar los datos generados en cada episodio de aprendizaje.
- Escribir la tabla Q en ficheros de texto, lo que permite su posterior transferencia mediante *Processing*.

4.4. Flujo de trabajo durante el aprendizaje

Para facilitar la replicación del sistema y la validación de resultados, se ha documentado en un anexo el procedimiento necesario para ejecutar tanto experimentos de entrenamiento como de explotación, incluyendo la gestión de los datos generados.

Esta documentación incluye el uso del entorno Arduino IDE para la carga del código, la interacción mediante *Processing*, y la recepción y gestión de datos en MATLAB, cubriendo de forma completa el ciclo de trabajo.

La guía detallada para ejecutar experimentos, registrar los datos y procesarlos se encuentra en el ANEXO IV**.**

5. Implementación y diseño del aprendizaje.

En este capítulo se presenta el proceso de diseño del sistema de aprendizaje por refuerzo, detallando las decisiones tomadas para adaptar el algoritmo Q-Learning a las limitaciones del entorno embebido. A diferencia del capítulo anterior, centrado en la implementación, aquí se analiza cómo se definieron los parámetros del algoritmo, el espacio de acciones, el espacio de estados, la función de recompensa y la estructura de la tabla Q.

Se describe también la evolución progresiva del diseño, desde configuraciones iniciales hasta alternativas más efectivas, teniendo siempre en cuenta las restricciones del sistema y la capacidad de almacenamiento disponible. Los resultados cuantitativos obtenidos con cada configuración se presentan y analizan en el Capítulo 6.

5.1. Parámetros del algoritmo de aprendizaje

El algoritmo Q-Learning utilizado para el aprendizaje depende de tres parámetros principales que condicionan la velocidad de aprendizaje, la estabilidad y la capacidad de exploración del entorno: el *learning rate* (α), el factor de descuento (γ) y el parámetro de exploración (ϵ), todos ellos explicados en capítulos previos de esta memoria.

El ***learning rate*** o tasa de aprendizaje se mantuvo constante en un valor de $\boldsymbol{\alpha = 0.1}$ a lo largo de todo el proceso. Este valor permite que el agente actualice su tabla Q de forma constante, evitando oscilaciones bruscas en las estimaciones de valor, especialmente relevantes en entornos ruidosos como el sistema físico implementado. Valores superiores fueron descartados en pruebas iniciales por provocar una alta variabilidad entre episodios, mientras

que valores más bajos ralentizaban la convergencia sin aportar mejoras significativas en estabilidad.

El **factor de descuento**, que pondera la importancia de las recompensas futuras respecto a las inmediatas, se fijó en $\gamma = 0.95$. Este valor se escogió para incentivar al agente a maximizar su rendimiento a largo plazo, permitiendo que decisiones que favorezcan la estabilidad durante varios pasos sean mejor valoradas que aquellas que generan recompensas inmediatas pero conducen a caídas rápidas. Esta elección es coherente con la naturaleza acumulativa del problema, donde mantenerse en equilibrio el mayor tiempo posible es el objetivo prioritario.

El parámetro ϵ controla el equilibrio entre exploración y explotación en la política ϵ-*greedy*. Para lograr un aprendizaje eficiente, se diseñó una **estrategia escalonada propia** que permite una exploración amplia en las fases iniciales, seguida de una progresiva transición hacia la explotación de las acciones aprendidas. Concretamente, se aplicaron los siguientes valores:

- $\epsilon = 0.75$ durante los **30 primeros episodios**, para fomentar la exploración y evitar caer en óptimos locales prematuros.
- $\epsilon = 0.5$ durante los **30 episodios siguientes**, reduciendo gradualmente la aleatoriedad.
- A partir del episodio **61 en adelante, ϵ se mantuvo fijo en 0.2**, permitiendo cierta variabilidad en las decisiones, pero primando ya la explotación de la política aprendida.

Este esquema de reducción escalonada de ϵ ofreció una buena relación entre descubrimiento de nuevas acciones útiles y consolidación del comportamiento del agente. **Existen otros mecanismos conocidos para disminuir ϵ de forma progresiva**, como la reducción lineal, exponencial o basada en una función inversa del número de episodios, pero se optó por una estrategia por tramos al permitir un mayor control experimental sobre cada fase del aprendizaje.

Además, implementar un esquema continuo de decremento hubiera requerido conocer con precisión el número de episodio actual en cada momento. Para ello habría sido necesario almacenar ese dato de forma persistente (por ejemplo, en la memoria EEPROM) o recibirlo externamente en cada arranque, lo que habría añadido complejidad al sistema y un mayor uso

de recursos. En cambio, el planteamiento actual se adapta mejor a las restricciones del entorno embebido utilizado.

5.2. Discretización del espacio de acciones

En el contexto del aprendizaje por refuerzo, el espacio de acciones representa las posibles decisiones que el agente puede tomar para influir en el entorno. En este proyecto, dichas acciones se corresponden con las órdenes de velocidad enviadas simultáneamente a ambos motores del robot Balboa 32U4, reguladas mediante modulación por ancho de pulsos (PWM).

Tal como se explicó en la Sección 4.1.2, la función `setSpeeds()` de la librería Balboa32U4 permite aplicar valores entre -300 y 300, donde el signo determina la dirección de giro. No obstante, dado que el sistema debe aprender en un entorno empotrado con memoria limitada, resulta fundamental reducir la dimensionalidad del espacio de acciones mediante una discretización adecuada.

Para realizar esta discretización de forma razonada, se partió del código de ejemplo proporcionado por el fabricante, que permite mantener el robot equilibrado mediante un sistema de control clásico. Este código, basado en un controlador similar a un PID, fue modificado para registrar los valores de velocidad aplicados en tiempo real. A partir de estas mediciones, se generó un histograma con las frecuencias de uso de cada valor de salida, reflejado en la Figura 5.1.

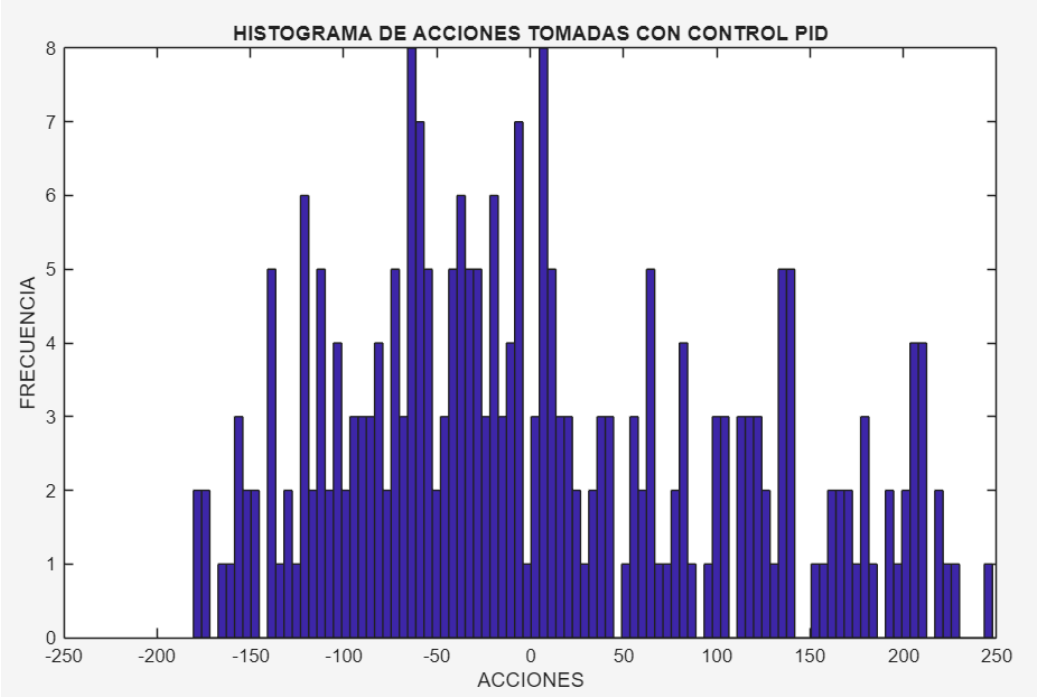

Figura 5.1. Histograma de velocidades usadas por un control PID aplicado al balanceo del robot.

Tal como se observa, el controlador emplea con mayor frecuencia valores cercanos a cero y, en menor medida, comandos más agresivos en ambos sentidos. A partir de esta distribución, se definió el siguiente conjunto discreto de acciones:

$$A = \{-200, -150, -100, -50, 0, 50, 100, 150, 200\} \quad (5.1)$$

Esta elección permite cubrir una gama representativa de acciones suaves y fuertes, evitando valores extremos como ±300, que pueden inducir oscilaciones violentas (su uso prolongado puede llegar a dañar los motores). Además, se mantiene un número reducido de elementos (nueve en total), lo que garantiza una tabla Q de tamaño asumible para el microcontrolador ATmega32U4.

5.3. Definición del espacio de estados

Otra de las decisiones clave en el diseño de un agente de aprendizaje por refuerzo es la definición del espacio de estados, es decir, la información que se proporciona al agente para que pueda tomar decisiones. Esta elección determina en gran medida la complejidad del problema, el tamaño de la tabla Q y, en última instancia, la calidad del aprendizaje obtenido.

A lo largo del desarrollo del sistema, se exploraron distintas configuraciones del espacio de estados con el objetivo de encontrar un equilibrio entre riqueza informativa y viabilidad computacional. En concreto, se analizaron tres enfoques progresivos: inicialmente se empleó una combinación de ángulo y velocidad de giro; posteriormente se probó una representación basada únicamente en el ángulo, pero utilizando un filtro exponencial más sencillo para obtener ese ángulo; finalmente se adoptó un diseño más depurado con una única variable (el ángulo) estimada mediante un filtro complementario.

En todos los casos se partió de una configuración común en la que se mantuvo un retardo fijo de 25 ms entre la ejecución de cada acción y la lectura posterior de sensores. Sin embargo, como se desarrolla posteriormente en este capítulo, en el último enfoque se estudiaron distintos retardos.

Cada uno de estos enfoques conllevó implicaciones distintas en cuanto a la estabilidad del aprendizaje y la eficiencia de la implementación. En este apartado se describen las características y motivaciones de cada configuración, mientras que el análisis de resultados y rendimiento asociado a cada una de ellas se recoge en el Capítulo 6.

5.3.1. Estados definidos por ángulo y velocidad angular

En una primera aproximación se decidió definir el estado del agente a partir de dos variables clave para la tarea de balanceo: el ángulo de inclinación del robot y su velocidad angular. La motivación de esta elección se basaba en considerar que no solo es relevante conocer la desviación respecto al equilibrio, sino también la rapidez con la que dicha inclinación cambia, ya que ambos factores influyen directamente en la estabilidad del sistema.

Ambas variables fueron obtenidas a partir de los sensores inerciales del Balboa 32U4. El ángulo se estimó mediante un **filtro complementario**, el cual combina la señal del acelerómetro (estable a largo plazo pero sensible al ruido) y la del giroscopio (precisa a corto plazo pero propensa a deriva de error), tal como se describe en detalle en la Sección 3.5.4. La velocidad angular, por su parte, se tomó directamente del eje Y del giroscopio, una vez compensado su *offset*.

Para poder representar estos estados en una tabla Q, fue necesario discretizar ambos valores en intervalos. La discretización inicial se realizó con los siguientes límites:

- Ángulo (en grados):

$$\theta_{límites} = \{0, 40, 60, 75, 95, 110, 130, 175\} \quad (5.2)$$

Cabe señalar que, aunque teóricamente la posición de equilibrio del robot debería corresponder a los 90º, en la práctica se observó que el ángulo de inclinación que mantiene el equilibrio real del sistema se situaba ligeramente desplazado (≈ 85°). Por este motivo, el intervalo central de la discretización se definió entre 75º y 95º, lo que permitió capturar con mayor precisión el comportamiento del robot en la zona de equilibrio.

- Velocidad angular (en º/s):

$$\omega_{limites} = \{-1000, -360, -120, 120, 360, 1000\} \quad (5.3)$$

Esto dio lugar a un total de $7 \times 5 = 35$ combinaciones distintas de estado que, al combinarse con las 9 acciones definidas en el Apartado 5.2, resultaron en una tabla Q de 315 elementos; que al ser estos de tipo `float`, supone un espacio en memoria de 1260 bytes, es decir, aproximadamente el 50% de la memoria SRAM ocupado exclusivamente por la tabla Q.

Sin embargo, tras ejecutar un número significativo de episodios de entrenamiento, se observó que **ciertas filas de la tabla Q no llegaban a ser actualizadas nunca** (aquellas que corresponden a las velocidades más altas en módulo), tal como se muestra en la Figura 5.2. Esto indicaba que varios rangos de velocidad angular no se alcanzaban y, por tanto, no aportaban información útil al proceso de aprendizaje.

12	2.1100	1.1600	3.9200	22.2600	2.3800	1.1400	9.7100
13	11.2400	15.9900	14.0900	8.2100	5.5500	11.1800	25.2200
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	-0.0200	0	29.1300	0	4.6400	0	0
17	69.4600	73.4700	73.2200	73.9600	69.2200	74.8800	75.9000
18	75.0700	76.4600	76.8400	72.7400	73.3400	76.9600	76.4600
19	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0
21	-0.0500	-0.4500	-0.6100	-0.6500	-0.2800	-0.9700	-0.0400
22	-0.3100	-0.1700	-0.2500	-0.1800	-0.3100	-0.1900	-0.2300
23	-0.0300	-0.0500	-0.0400	-0.0300	-0.0400	-0.0400	-0.0700
24	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0

Figura 5.2. Tabla Q con filas sin modificar durante el entrenamiento, debido a una discretización inadecuada de la velocidad angular.

Con el objetivo de solventar este problema, se terminó empleando la siguiente nueva discretización:

$$\omega_{limites} = \{-1000, -150, -50, 50, 150, 1000\} \qquad (5.4)$$

Este nuevo conjunto de límites agrupaba de forma más realista las velocidades típicas observadas durante el aprendizaje, concentrando la resolución en la zona próxima al equilibrio, donde ocurren la mayoría de transiciones, y consiguiendo de esta manera que todos los estados sean alcanzables (véase Figura 5.3).

12	21.6200	28.3200	13.6800	16.7300	16.9900	32.0300	20.2400	50.1200	17.0600
13	26.5800	17.5000	21.3000	22.2000	20.5400	41.5200	15.0900	16.4100	7.2500
14	4.2200	10.6400	11.2300	10.2300	10.6700	10.4400	11.0800	41.2400	4.5700
15	6.2100	6.1800	6.3200	20.3700	12.8800	28.5400	11.0400	24.7600	0.5000
16	41.8600	45.5000	45.8500	46.5200	45.2400	39.8600	45.5000	40.9700	49.5300
17	49.1200	49.8900	52.1200	46.3800	50.7100	49.2200	50.5600	45.6400	51.4900
18	51.9700	54.7200	51.3200	51.4600	51.2300	50.5500	48.6700	49.7300	51.3700
19	49.7400	50.8900	46.2900	53.3600	49.1600	48.2100	50.7100	48.8300	49.1800
20	45.4200	46.8700	48.2600	46.4100	40.1600	53.3500	40.9400	48.7200	38.2100
21	26.0300	16.4200	21.5600	7.4000	27.3900	20.2200	28.9500	18.8900	26.1800
22	17.7700	29.4900	17.7500	14.6600	28.4000	19.1700	24.3900	20.9800	43.1500
23	32.4600	23.7300	28.9900	25.6500	36.5400	18.1400	39.9600	24.7300	48.8300
24	36.3600	37.0600	39.6600	33.8400	39.6700	31.2300	49.8700	19.8700	42.8000
25	34.4800	51.7400	38.9200	43.4900	18.1500	31.8100	33.0800	20.0200	37.4700

Figura 5.3. Tabla Q con todas las filas modificadas gracias a una mejor discretización de la velocidad angular.

5.3.2. Estados definidos por ángulo con filtro exponencial

Como segundo enfoque, se decidió explorar una representación del estado basada exclusivamente en el ángulo de inclinación del robot, con el objetivo de **incrementar la resolución angular** y comprobar si una mayor granularidad en esta variable permitiría al agente aprender un comportamiento de equilibrio más estable. Para ello, se optó por eliminar la velocidad angular de la definición del estado, porque **el aumento en el número de intervalos del ángulo imposibilitaba mantener ambas variables** dentro de los límites de memoria del sistema.

Para la estimación del ángulo se empleó en este caso un **filtro exponencial**, una técnica más simple que el filtro complementario, cuya respuesta temporal y propiedades de suavizado fueron analizadas en la Sección 3.5.4. Se seleccionó un valor fijo del factor de suavizado de **0.10**, ya que durante las pruebas iniciales se compararon valores de 0.05, 0.10 y 0.15, y este ofrecía un equilibrio adecuado entre respuesta rápida y estabilidad sin introducir excesivo ruido.

No obstante, esta técnica introduce una latencia considerable, lo cual puede comprometer la precisión de estimación en entornos con dinámicas rápidas como el balanceo. Por tanto, es **razonable anticipar que este enfoque de filtrado no resultará el más adecuado** para el aprendizaje por refuerzo. Aun así, se decidió llevar a cabo el proceso de entrenamiento con esta configuración con el objetivo de evaluar empíricamente su impacto en el comportamiento del agente.

La discretización del ángulo se realizó dividiendo el intervalo [0º, 175º] en pasos de 5º, lo que genera un total de 35 intervalos. Este diseño pretendía ofrecer al agente una representación más detallada del ángulo, permitiendo teóricamente una política más fina y adaptada.

El análisis de la tabla Q obtenida reveló que, para un mismo estado, muchas acciones distintas presentan valores de Q muy similares, o bien que ciertas acciones llegaban a registrar valores altos en estados en los cuales no eran coherentes. Este comportamiento puede deberse a la latencia inherente del filtro exponencial, que introduce un retardo entre el cambio físico real y el valor estimado del ángulo.

En la Figura 5.4 se puede observar, por ejemplo, que en la **fila 18**, correspondiente al intervalo angular [85°, 90°], el valor máximo se encuentra en la **columna 4**, que representa una acción de **+100**. Aplicar un impulso de esa magnitud cuando el robot está prácticamente equilibrado no resulta coherente desde el punto de vista del control.

Del mismo modo, en la **fila 11** (intervalo [50°, 55°]), la acción con mayor valor es la de la **columna 5**, correspondiente a **−100**, lo cual implicaría empujar al robot aún más lejos del equilibrio. Estas inconsistencias refuerzan la hipótesis de que la representación del estado basada en el filtro exponencial compromete la calidad de las políticas aprendidas.

	1	2	3	4	5	6	7	8	9
10	-6.7400	-8.1900	-6.8800	-6.6000	-6.5800	-10000	-5.9100	-6.8200	-10000
11	-2.8700	-2.2400	-3.3600	-4.0600	-1.9300	-3.8000	-1.8600	-2.2200	-2.0200
12	-10000	-3.1300	-3.7500	-2	-1.8400	-2.5600	-2.0300	-1.9900	1.0300
13	-3.2500	-1.8800	-0.7800	-2.0800	0.3300	-0.3400	-2.2600	-1.5700	7.3900
14	-1.2600	-0.2200	-1.4200	0.6500	11.8100	-0.1600	0.7300	-2.3200	-1.4700
15	7.5400	3.4200	25.4900	3.0900	12.7500	6.5800	5.0700	-0.8500	10.3400
16	16.3500	20.7100	12.2100	13.3900	37.5100	10.3200	28.7000	20.7800	26.1100
17	42.1300	40.4600	38.5500	30.0400	41.1700	39.5900	40.6700	31.7500	49.1400
18	45.3000	44.1800	41.1400	46.9300	41.7700	44.2500	44.8100	45.9600	40.9700
19	11.8700	23.3900	19.9200	31.8400	24.1200	28.8700	14.6800	45.9400	15.7800
20	0.9900	1.8400	1.1800	9.1100	2.7800	5.3100	0.2900	33.9700	1.5700
21	2.8000	-10000	0.0300	2.3100	11.9700	3.3600	0.0300	0.0700	-10000
22	-10000	1.8300	-10000	0.8700	4.3900	-10000	1.9800	1.3900	-10000
23	-10000	-0.1200	-10000	-0.0800	-10000	-10000	-0.1000	-10000	-10000

Figura 5.4. Fragmento de la tabla Q obtenida con el filtro exponencial, en la que se aprecian decisiones incoherentes del agente.

Esta arquitectura dio lugar a una tabla Q de tamaño **35 × 9 = 315 elementos**, igual que en el caso anterior, ya que el aumento en la resolución angular compensó la eliminación de la segunda variable. Al estar implementada en tipo `float`, el espacio requerido fue también de **315 × 4 = 1260 bytes**, lo que representa aproximadamente el 50 % de la memoria SRAM disponible en el microcontrolador.

5.3.3. Estados definidos por ángulo con filtro complementario

Tras evaluar las limitaciones del filtro exponencial, se optó por recuperar el uso del **filtro complementario** para estimar el ángulo de inclinación del robot, manteniendo, eso sí, una

arquitectura simplificada basada en una única variable de estado. Esta decisión se tomó con el objetivo de conservar la baja complejidad computacional de los enfoques anteriores, pero mejorando la precisión y velocidad de respuesta del sistema de estimación, aspectos fundamentales en una tarea de control como el balanceo.

Para mantener la comparabilidad con el enfoque anterior, se conservó la misma discretización del ángulo, empleando intervalos de 5° en el rango [0°, 175°].

Esto generó un total de **35 intervalos** que, al combinarse con las 9 acciones disponibles, dio lugar a una tabla Q de tamaño **35 × 9 = 315 elementos**. Como en los casos anteriores, el uso del tipo `float` implicó un consumo de memoria de **1260 bytes**, aproximadamente el 50 % de la memoria SRAM del microcontrolador ATmega32U4.

Con este último enfoque se observó una evolución del aprendizaje más rápida, estable y consistente.

5.4. Función de recompensa y compensación temporal

Uno de los elementos esenciales en el diseño de un agente de aprendizaje por refuerzo es la definición de la función de recompensa. Esta determina el criterio que guía el aprendizaje del agente, condicionando qué situaciones considera deseables o penalizables, y, por tanto, qué política termina aprendiendo.

En este trabajo se ha adoptado un planteamiento basado en recompensas asociadas al estado actual, lo que simplifica el cálculo y se adapta mejor a las limitaciones del entorno embebido. En los tres enfoques de espacio de estados planteados anteriormente se utilizaron distintos sistemas de recompensa o ajustes sobre ellos, que se describen a continuación.

5.4.1. Recompensa en el enfoque con ángulo y velocidad angular

En el primer enfoque, en el que el estado se define por el ángulo de inclinación y la velocidad angular, se utilizó una recompensa escalonada en función de la posición angular.

La recompensa se asignó en función del intervalo angular y del intervalo de velocidad angular al que pertenece el estado actual. En este caso, se definieron **35 estados** discretos, cada uno con una recompensa fija asociada (véase Tabla 5.1).

Además, se debe tener en cuenta que los estados cuyas recompensas tienen fondo rojo suponen el **fin del episodio**, ya que una vez alcanzado dicho ángulo, el robot no es capaz de recuperar el equilibrio. Aquellos con el fondo verde son los estados más beneficiosos para el agente.

Esta recompensa busca reforzar claramente los estados cercanos al equilibrio, con recompensas máximas asignadas a los estados del intervalo [75º, 95º], penalizando severamente las situaciones en los extremos (valores cercanos a 0º o 175º), que se consideran situaciones de caída.

Índice del estado	INTERVALO(°)	INTERVALO(°/s)	RECOMPENSA
0	0°:40°	-1000°/s : -150°/s	-20
1	0°:40°	-150°/s : -50°/s	-20
2	0°:40°	-50°/s : 50°/s	-20
3	0°:40°	50°/s : 150°/s	-20
4	0°:40°	150°/s : 1000°/s	-20
5	40°:60°	-1000°/s : -150°/s	0
6	40°:60°	-150°/s : -50°/s	0
7	40°:60°	-50°/s : 50°/s	0
8	40°:60°	50°/s : 150°/s	0
9	40°:60°	150°/s : 1000°/s	0
10	60°:75°	-1000°/s : -150°/s	0
11	60°:75°	-150°/s : -50°/s	0
12	60°:75°	-50°/s : 50°/s	0
13	60°:75°	50°/s : 150°/s	0
14	60°:75°	150°/s : 1000°/s	0
15	75°:95°	-1000°/s : -150°/s	5
16	75°:95°	-150°/s : -50°/s	5
17	75°:95°	-50°/s : 50°/s	5
18	75°:95°	50°/s : 150°/s	5
19	75°:95°	150°/s : 1000°/s	5
20	95°:110°	-1000°/s : -150°/s	0
21	95°:110°	-150°/s : -50°/s	0
22	95°:110°	-50°/s : 50°/s	0
23	95°:110°	50°/s : 150°/s	0
24	95°:110°	150°/s : 1000°/s	0
25	110°:130°	-1000°/s : -150°/s	0
26	110°:130°	-150°/s : -50°/s	0
27	110°:130°	-50°/s : 50°/s	0
28	110°:130°	50°/s : 150°/s	0
29	110°:130°	150°/s : 1000°/s	0
30	130°:175°	-1000°/s : -150°/s	-20
31	130°:175°	-150°/s : -50°/s	-20
32	130°:175°	-50°/s : 50°/s	-20
33	130°:175°	50°/s : 150°/s	-20
34	130°:175°	150°/s : 1000°/s	-20

Tabla 5.1. Recompensas asociadas a los estados discretos definidos por ángulo y velocidad angular.

5.4.2. Recompensa en los enfoques definidos sólo por el ángulo

En los enfoques segundo y tercero, el estado del agente se define únicamente a partir del ángulo de inclinación, eliminando la velocidad angular. Esta simplificación del espacio de estados se reflejó también en el sistema de recompensas, que se mantuvo idéntico en ambos casos: sin depender de la velocidad angular asociada al movimiento.

La recompensa se asigna directamente en función del intervalo angular al que pertenece el estado actual, siguiendo una estructura escalonada. Los valores se asignaron a cada estado con el objetivo de premiar la permanencia en la zona de equilibrio central, penalizar las caídas y asignar valores neutros a los estados intermedios.

En este caso, se definieron **35 estados discretos** como resultado de dividir el intervalo [0°, 175°] en pasos de 5°. Cada uno de ellos tiene una recompensa fija asociada, mostrada en la Tabla 5.2. Esta misma tabla se utilizó tanto en el enfoque 2 (con filtro exponencial) como en el enfoque 3 (con filtro complementario), lo que permite una comparación directa entre ambos.

Como en el caso anterior, los estados con recompensa de **-20**, mostrados con fondo rojo, representan situaciones de caída y suponen el **fin del episodio**. A diferencia del planteamiento anterior, los estados intermedios (con fondo ámbar) también penalizan al agente, pero en menor medida que los estados terminales. En la zona de equilibrio se ha diferenciado entre dos estados más extremos, que reciben una recompensa positiva inferior (fondo verde claro), mientras que la zona ideal de equilibrio otorga la mayor recompensa positiva (verde intenso).

Índice del estado	INTERVALO(°)	RECOMPENSA
0	0°:5°	-20
1	5°:10°	-20
2	10°:15°	-20
3	15°:20°	-20
4	20°:25°	-20
5	25°:30°	-20
6	30°:35°	-20
7	35°:40°	-20
8	40°:45°	-1
9	45°:50°	-1
10	50°:55°	-1
11	55°:60°	-1
12	60°:65°	-1
13	65°:70°	-1
14	70°:75°	-1
15	75°:80°	2
16	80°:85°	5
17	85°:90°	5
18	90°:95°	2
19	95°:100°	-1
20	100°:105°	-1
21	105°:110°	-1
22	110°:115°	-1
23	115°:120°	-1
24	120°:125°	-1
25	125°:130°	-1
26	130°:135°	-20
27	135°:140°	-20
28	140°:145°	-20
29	145°:150°	-20
30	150°:155°	-20
31	155°:160°	-20
32	160°:165°	-20
33	165°:170°	-20
34	170°:175°	-20

Tabla 5.2. Recompensas asociadas a los estados definidos únicamente por el ángulo.

5.4.3. Compensación temporal aplicada en el segundo enfoque

En el tercer enfoque, en el que se emplea el filtro complementario para estimar el ángulo y se define el estado únicamente a partir de esta variable, se observa un comportamiento más estable durante el proceso de aprendizaje. Esto permite ir un paso más allá y evaluar experimentalmente el impacto de diferentes **valores de retardo (adicional)** entre la ejecución de la acción y la lectura de sensores.

Este cambio introduce una dificultad importante: los episodios con diferentes retardos tienen **duraciones distintas por paso**, lo que hace que las recompensas acumuladas no sean directamente comparables entre ellas. Para resolver esta situación, se optó por **multiplicar la recompensa base de cada paso** (véase Tabla 5.2) **por el tiempo de paso correspondiente**, es decir, el tiempo total transcurrido entre la acción anterior y la lectura del nuevo estado.

El tiempo de paso se pudo obtener de forma muy sencilla, ya que es el mismo valor temporal que se usa para llevar a cabo la integración del ángulo usada en el filtro complementario. La recompensa modificada, por tanto, se calcula como:

$$Recompensa\ compensada = Recompensa\ base\ \times (tiempo\ de\ paso\ medido)[ms] \qquad (5.5)$$

Gracias a esta compensación, es posible comparar de forma justa el rendimiento del agente entre episodios con distintos valores de retardo, ya que se tiene en cuenta la duración efectiva de cada paso. Los resultados de estas comparativas se presentan el Apartado 6.4.

5.4.4. Justificación del valor de inicialización de la tabla Q

En la Sección 4.1.3 se explicó que, al inicio del entrenamiento, todos los valores de la tabla Q se inicializan con un valor constante de $\mathbf{-10000}$. Esta elección tenía como objetivo establecer un valor inicial suficientemente bajo como para asegurar que no se escogiesen acciones inexploradas para un estado dado en el comienzo del aprendizaje como las mejores acciones posibles (evitar falsos máximos).

Dicha decisión se justificaba con la fórmula del valor Q esperado en el peor de los casos. En concreto, se planteó la Ecuación (4.1).

Una vez definidos los valores concretos en los apartados anteriores, se puede aplicar esta fórmula. La recompensa más negativa que pude recibir el agente en los 3 enfoques planteados, es $R_{min} = -20$ y el valor del factor de descuento es $\gamma = 0.95$; se puede obtener el valor del valor mínimo que puede darse en la tabla Q como:

$$Q_{min} = \frac{-20}{1 - 0.95} = -400 \tag{5.6}$$

Este valor representa el límite inferior que puede alcanzar una celda de la tabla Q, incluso en el caso más extremo. Por tanto, la elección de **−10000** como valor inicial garantiza una diferencia suficientemente grande para que cualquier valor aprendido lo supere con rapidez durante las primeras iteraciones del algoritmo. Esta técnica acelera la exploración y evita bloqueos iniciales en políticas subóptimas.

6. Evaluación de resultados

Una vez definidos los distintos enfoques de diseño, en este capítulo se analizan los resultados obtenidos durante el proceso de aprendizaje del agente en cada una de las configuraciones propuestas. El objetivo principal es evaluar el impacto que tienen las decisiones de representación del estado, filtrado de señales y formulación de la recompensa sobre el rendimiento final del sistema. El capítulo concluye con una comparación global entre los distintos enfoques.

6.1. Metodología de evaluación

Para valorar el rendimiento del sistema bajo las distintas configuraciones propuestas se ha optado por un análisis basado tanto en métricas cuantitativas como en observaciones cualitativas del comportamiento del agente. En particular, se estudian la **evolución de la recompensa acumulada por episodio** y el **tiempo en equilibrio por episodio** a lo largo del aprendizaje, dos indicadores clave para evaluar la estabilidad del comportamiento y el progreso hacia una política eficiente.

Además del análisis durante el entrenamiento, en aquellos casos en los que se observó una mejora significativa del rendimiento se llevó a cabo una **fase de explotación pura**, en la que se desactivó la exploración aleatoria ($\epsilon = 0$) y no se actualizó la tabla Q. Dado que no todas las configuraciones dieron lugar a una convergencia satisfactoria, existen casos en los que **no se realizó explotación**, al considerarse que el agente no había aprendido una política útil.

Por otro lado, es importante destacar que **el número de episodios de entrenamiento no ha sido fijo en todos los casos**. Esta decisión fue tomada de forma empírica, con el objetivo de optimizar el análisis de cada enfoque: configuraciones que mostraban convergencia rápida no requerían un entrenamiento prolongado, mientras que otras que evolucionaban de forma más lenta o inestable fueron entrenadas durante más tiempo para que el proceso de aprendizaje llegara a un resultado estacionario.

6.2. Resultados del primer enfoque

Este primer enfoque, en el que el estado del agente se define a partir del ángulo de inclinación y la velocidad angular del robot, fue el caso en el que se realizaron un mayor número de episodios de entrenamiento, debido a las oscilaciones permanentes en la curva de recompensa acumulada, siendo en total 500 episodios.

A continuación se muestran los resultados obtenidos durante el entrenamiento. Para facilitar la interpretación, todas las curvas presentadas han sido suavizadas aplicando un filtro de media móvil, que reduce las variaciones rápidas y resalta las tendencias generales del comportamiento del agente. Dado que el agente se encuentra con casi todas las situaciones posibles en cada episodio, se considera que el sistema es ergódico, y, por tanto, esta media sobre un solo experimento dará parecida información a realizar la media de las curvas completas de varios experimentos; esto reduce bastante la cantidad de tiempo empleado en la evaluación empírica.

La **Figura 6.1** muestra la recompensa acumulada en función del número de episodios. Puede observarse una mejora progresiva durante el comienzo del entrenamiento, con incrementos en la media de la recompensa acumulada, aunque posteriormente se aprecian oscilaciones notables que indican inestabilidad en la política aprendida.

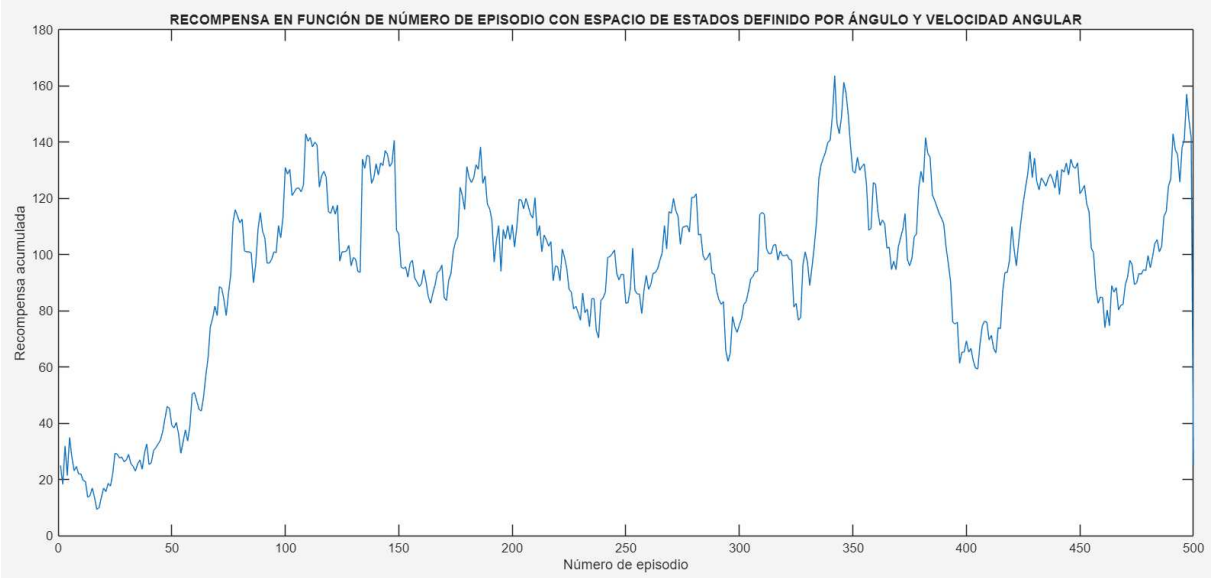

Figura 6.1. Recompensa acumulada por episodio con espacio de estados definido por ángulo y velocidad angular.

La **Figura 6.2** representa la recompensa acumulada en función del tiempo real de entrenamiento, lo que permite valorar el aprendizaje desde una perspectiva temporal más realista. La tendencia general es similar a la observada en la Figura 6.1.

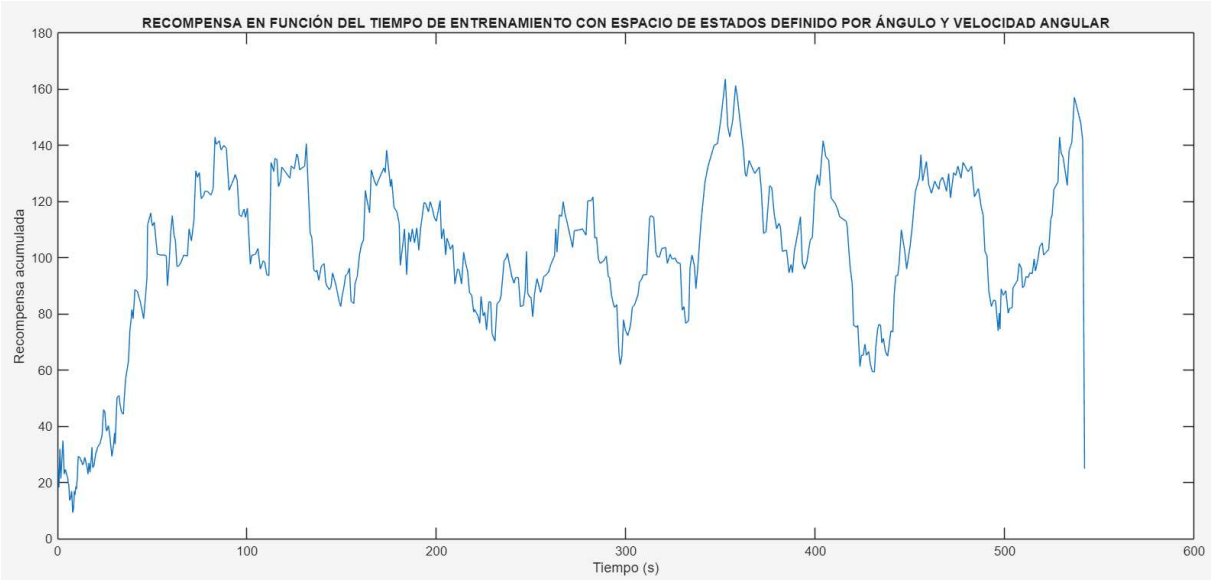

Figura 6.2. Recompensa acumulada en función del tiempo real de entrenamiento con espacio de estados definido por ángulo y velocidad angular.

La evolución de cuánto tiempo es capaz el robot de mantenerse en equilibrio durante el entrenamiento se muestra en la **Figura 6.3**, donde se aprecia un crecimiento progresivo, con fases de estabilización y pequeños retrocesos. Esta métrica es especialmente relevante, ya que representa directamente la efectividad de la política aprendida.

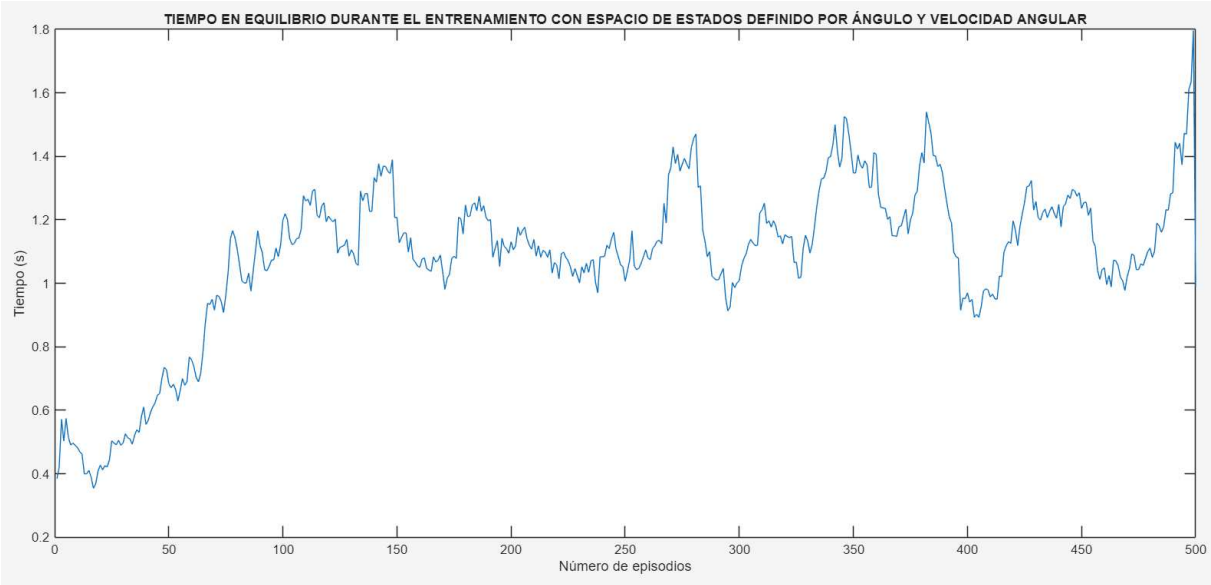

Figura 6.3. Tiempo en equilibrio del robot por episodio con espacio de estados definido por ángulo y velocidad angular.

Por último, la **Figura 6.4** muestra el mismo tiempo de equilibrio, pero representado en función del tiempo de entrenamiento. Al igual que en las gráficas anteriores, se aprecia un progreso inicial seguido de oscilaciones, lo que confirma que, aunque hubo aprendizaje, este no fue completamente estable.

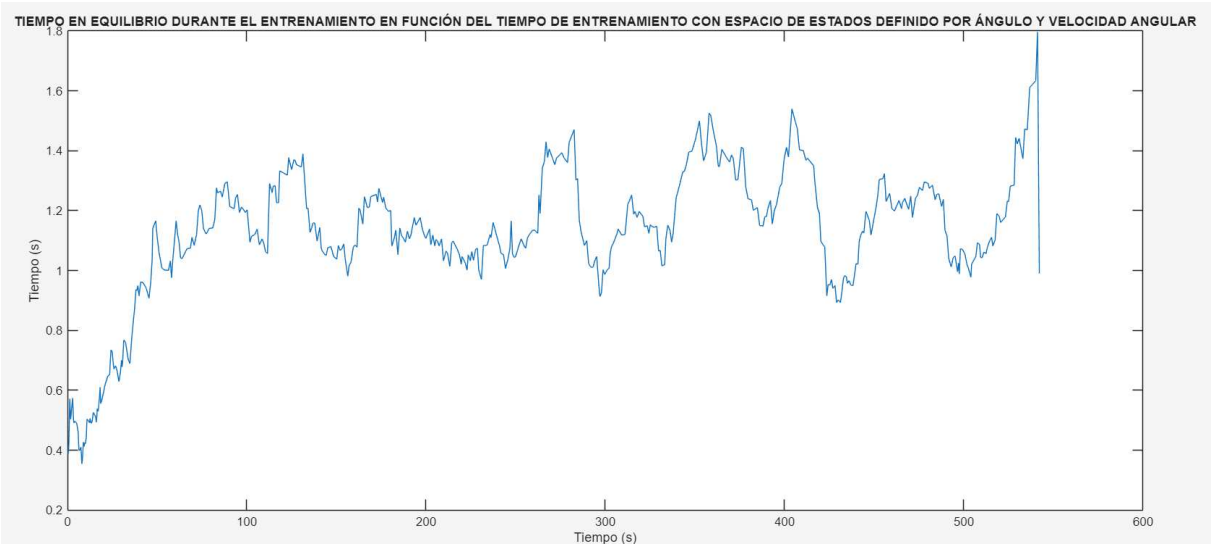

Figura 6.4. Tiempo en equilibrio del robot en función del tiempo de entrenamiento real con espacio de estados definido por ángulo y velocidad angular.

A pesar del entrenamiento prolongado y cierta mejora observable, la política aprendida no logró estabilizarse completamente. Las oscilaciones en la recompensa y el tiempo en equilibrio sugieren que, aunque el espacio de estados era el mismo que en los otros enfoques, la cobertura insuficiente del valor del ángulo, impide que el aprendizaje condujera a una política estable y efectiva.

Para completar el análisis de este enfoque, se llevó a cabo una fase de **explotación pura**, desactivando la exploración ($\epsilon = 0$) y evaluando el comportamiento del agente a lo largo de **50 episodios consecutivos**. A continuación, se presentan los resultados obtenidos para la recompensa acumulada y el tiempo de equilibrio en cada episodio.

La **Figura 6.5** muestra el diagrama de caja (*boxplot*) de la recompensa acumulada por episodio durante esta fase. Se observa una elevada dispersión, con valores que varían entre aproximadamente 50 y 600, y varios episodios con recompensas muy altas, que aparecen como valores atípicos. Esto indica que el agente fue capaz de aprender comportamientos relativamente efectivos en algunos casos, aunque no de forma consistente, obteniendo así una dispersión elevada.

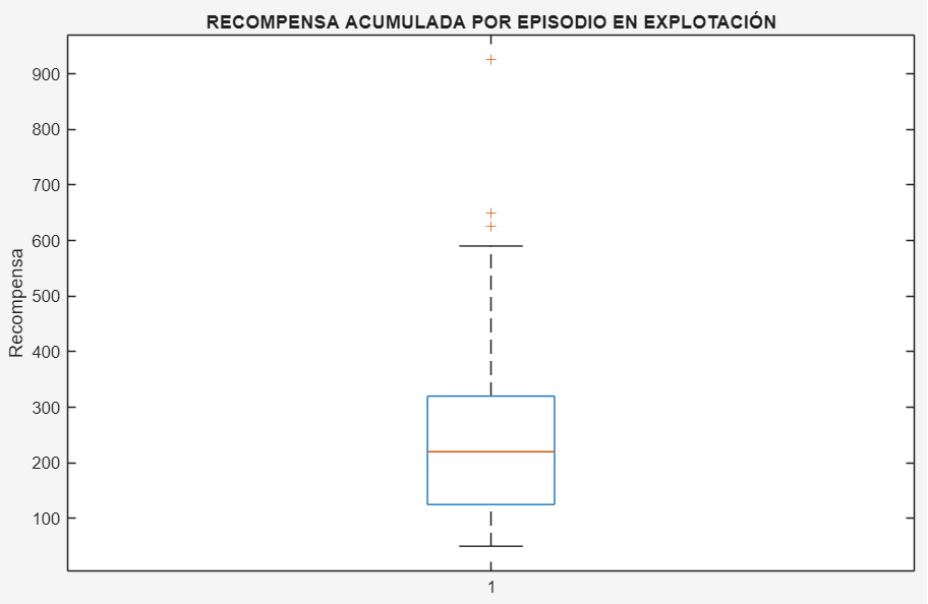

Figura 6.5. Recompensa acumulada por episodio en explotación con espacio de estados definido por ángulo y velocidad angular.

En la **Figura 6.6** se presenta el tiempo en equilibrio alcanzado por episodio durante la misma fase de explotación. Al igual que en la figura anterior, se aprecia una notable

variabilidad, con una mediana situada en torno a los 1700 ms y presencia de múltiples valores atípicos por encima de los 4000 ms. Este comportamiento refuerza la idea de que el aprendizaje fue parcial: el agente es capaz de mantener el equilibrio en ciertos episodios, pero **no de forma fiable o estable**.

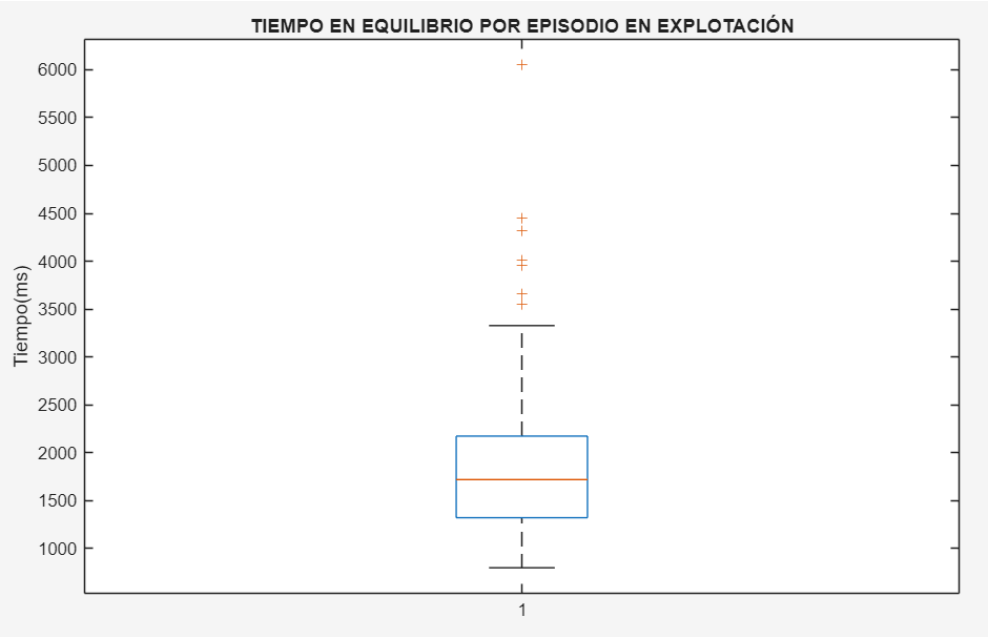

Figura 6.6. Tiempo en equilibrio del robot por episodio durante explotación con espacio de estados definido por ángulo y velocidad angular.

En conjunto, estos resultados muestran que, aunque el agente logró cierta mejora respecto a su comportamiento inicial, la combinación de ángulo y velocidad angular como variables del estado no permitió alcanzar una política robusta.

Para completar la documentación de este enfoque, en el ANEXO V, se incluye la tabla Q final obtenida tras el entrenamiento. Esta tabla recoge los valores aprendidos para cada combinación de estado y acción, y sirve como evidencia adicional del comportamiento alcanzado por el agente bajo esta configuración.

6.3. Resultados del segundo enfoque

Este segundo enfoque simplificó la representación del estado utilizando únicamente el ángulo de inclinación del robot, estimado mediante un **filtro exponencial**. Aunque esta reducción del espacio de estados perseguía aligerar el aprendizaje, en la práctica **la latencia**

propia del filtro y su pobre capacidad de reacción ante cambios bruscos comprometieron gravemente la calidad de las decisiones del agente.

El entrenamiento se realizó durante **80 episodios**, y no se llevó a cabo ninguna fase de explotación, ya que **no se observó una mejora significativa en el comportamiento del robot**.

Al igual, que en el caso anterior, todas las curvas presentadas han sido suavizadas aplicando un filtrado que reduce las variaciones rápidas y resalta las tendencias generales del comportamiento del agente.

La **Figura 6.7** muestra la recompensa acumulada por episodio. Aunque aparentemente se observa una tendencia creciente durante la primera mitad del entrenamiento, esta se estabiliza y comienza a decrecer a partir del episodio 50, con oscilaciones significativas.

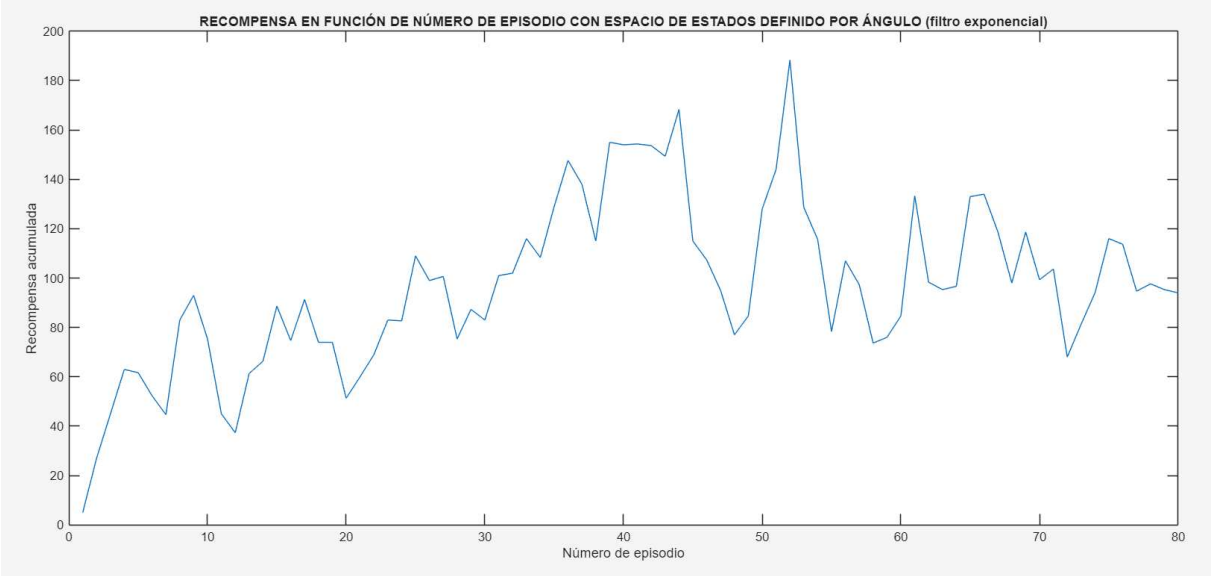

Figura 6.7. Recompensa acumulada por episodio con espacio de estados definido por ángulo (filtro exponencial).

La **Figura 6.8** refleja el mismo comportamiento en función del tiempo de entrenamiento.

A pesar de que las curvas muestran cierta mejora respecto al comportamiento inicial, los resultados reales observados en el robot no la respaldan. Este desfase entre los gráficos y la realidad puede explicarse por un efecto no deseado: **cuando el robot caía, el impacto**

generaba un pico de aceleración tan alto que la señal estimada por el filtro exponencial se veía afectada durante un tiempo prolongado, además de la latencia ya descrita anteriormente.

Como consecuencia, se generaban **recompensas equivocadas** durante estos intervalos, y el sistema, en ciertos casos, consideraba que se mantenía en equilibrio cuando en realidad ya había fallado.

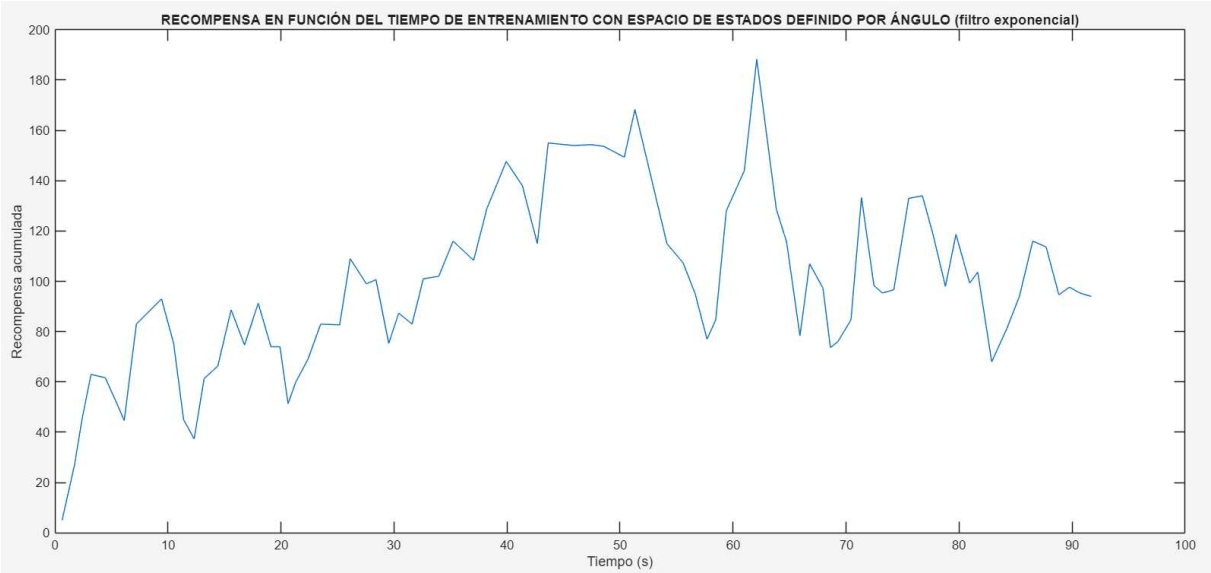

Figura 6.8. Recompensa acumulada en función del tiempo real de entrenamiento con espacio de estados definido por ángulo (filtro exponencial).

La **Figura 6.9** representa el tiempo en equilibrio por episodio. Se observa una cierta mejora inicial, pero sin estabilidad ni consolidación en los últimos episodios. Esto es consecuencia directa de que muchos de estos valores estén **inflados por el error en la estimación del estado real**, causado por el retardo del filtro.

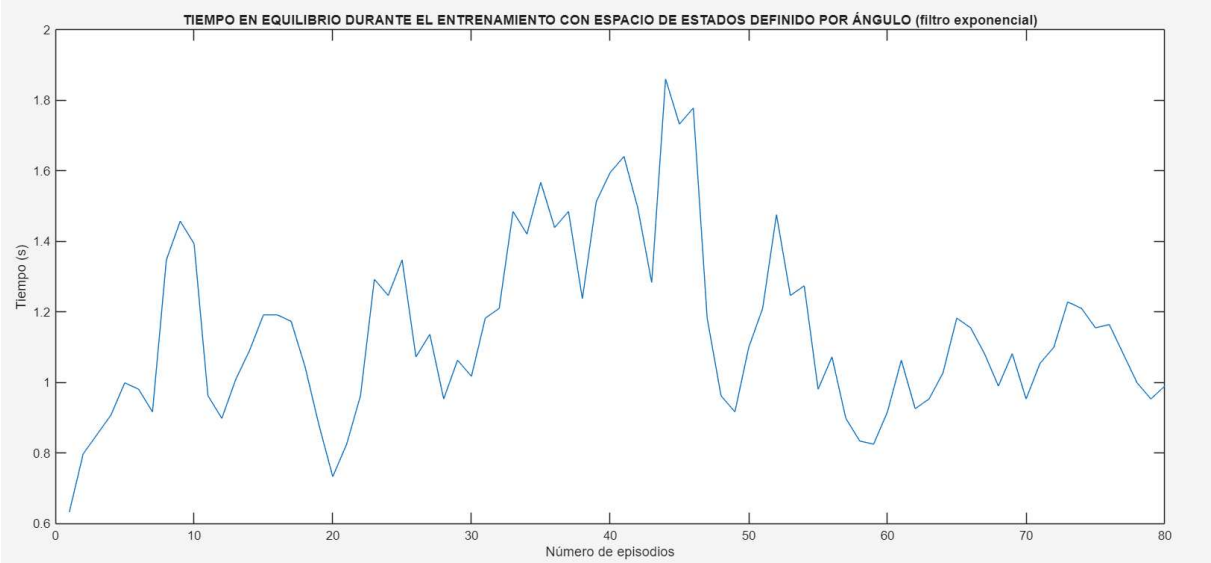

Figura 6.9. Tiempo en equilibrio del robot por episodio con espacio de estados definido por ángulo (filtro exponencial).

La **Figura 6.10** confirma esta misma tendencia cuando se observa el tiempo de equilibrio en función del tiempo de entrenamiento. Aunque la curva muestra valores en torno a los 1.2–1.6 segundos, la ejecución física del sistema demostró que el robot **no estaba realmente en equilibrio durante estos tiempos**.

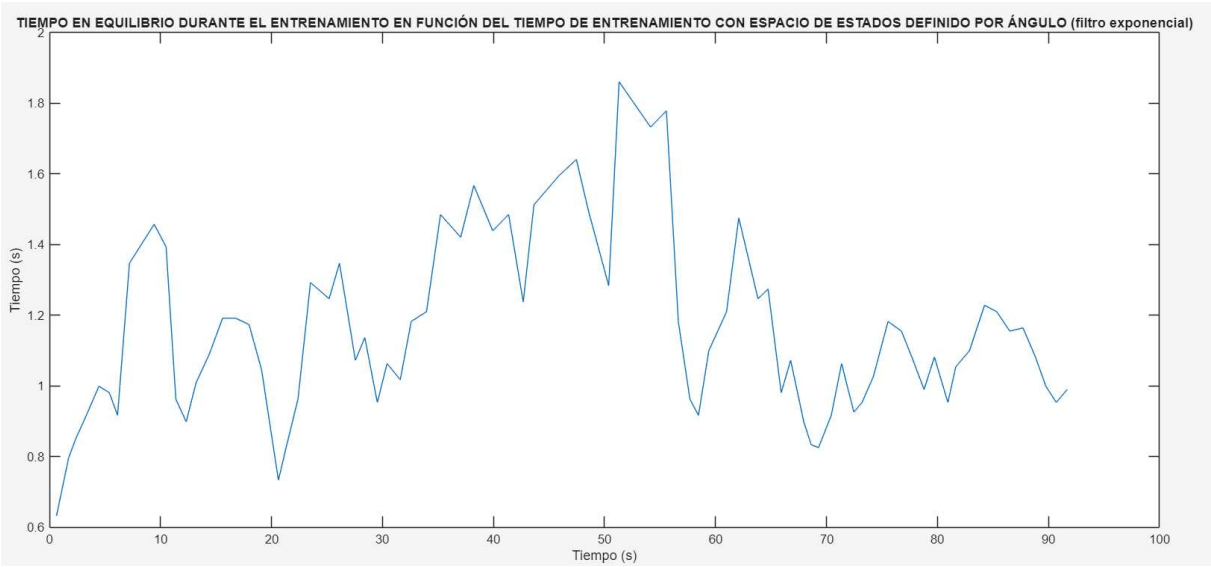

Figura 6.10. Tiempo en equilibrio del robot en función del tiempo de entrenamiento real con espacio de estados definido por ángulo (filtro exponencial).

Este conjunto de resultados evidencia que, a pesar de una aparente mejora respecto al inicio del aprendizaje en las métricas registradas, el comportamiento físico real del sistema fue

claramente deficiente. El agente no aprendió una política efectiva, y la latencia del filtro, unida al efecto de los picos de aceleración tras la caída, generó una visión distorsionada del entorno. Como consecuencia, el proceso de aprendizaje se vio comprometido, ya que el agente tomaba decisiones en base a valores sensoriales que no reflejaban el estado real del sistema.

Para completar la documentación de este enfoque, la tabla Q se incluye en el ANEXO V.

6.4. Resultados del tercer enfoque

Este último enfoque utilizó como única variable de estado el ángulo de inclinación del robot, pero, a diferencia del caso anterior, usando un filtro complementario, lo que permitió una respuesta más rápida y estable en comparación con el filtro exponencial usado anteriormente. Dado que durante el entrenamiento preliminar este enfoque demostró un comportamiento más robusto, se decidió estudiar con más detalle cómo afecta el **tiempo entre la ejecución de la acción y la lectura de los sensores**.

Para ello, se evaluaron cuatro valores de retardo: **10 ms, 25 ms, 40 ms y 55 ms**, y se aplicó en todos los casos la **compensación temporal** sobre la recompensa base (descrita en la Sección 5.4.3) de modo que la recompensa acumulada reflejara el tiempo real que el robot permanecía en equilibrio, permitiendo una comparación justa entre configuraciones.

Al igual que en los casos anteriores, todas las curvas presentadas han sido suavizadas aplicando un filtrado que reduce las variaciones rápidas y resalta las tendencias generales del comportamiento del agente.

La **Figura 6.11** muestra la evolución de la recompensa acumulada por episodio. Se observa que los retardos de **25 ms** y **40 ms** conducen a una curva estable y creciente, indicando una política funcional. En contraste, el retardo de **10 ms**, aunque presenta una ligera mejora inicial, no consigue mantener una tendencia clara de convergencia. El caso de **55 ms** resulta el más negativo: la recompensa acumulada permanece constantemente por debajo de cero, reflejando un comportamiento caótico.

Este mal rendimiento con 55 ms puede explicarse por la elevada latencia: al haber incrementado el tiempo entre acciones y lectura de sensor, el agente actúa de forma incorrecta

y se aleja sistemáticamente del equilibrio, acumulando recompensas negativas (−1 por cada paso, véase Tabla 5.2).

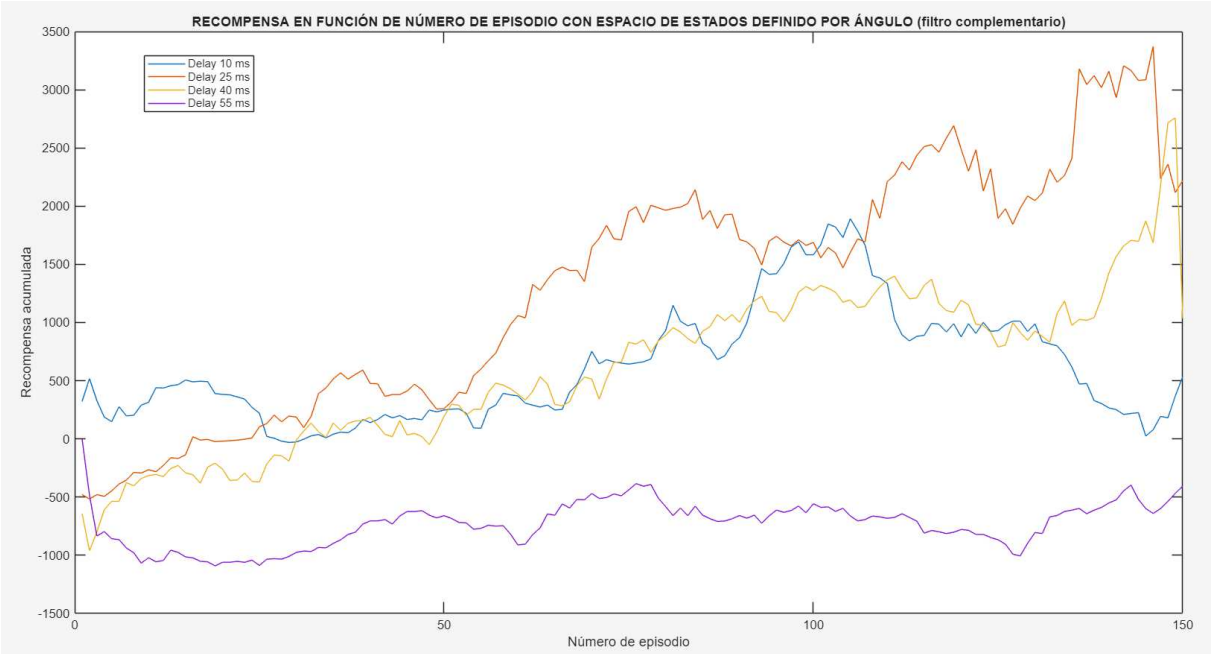

Figura 6.11. Recompensa acumulada por episodio para distintos retardos con espacio de estados definido por ángulo (filtro complementario).

La **Figura 6.12** presenta la misma métrica en función del tiempo real de entrenamiento. El patrón se mantiene: los retrasos intermedios (25–40 ms) permiten consolidar el aprendizaje, mientras que los extremos (10 ms y 55 ms) no ofrecen tan buenos resultados; además, en el caso de 10 ms, se ve cómo aunque al principio la recompensa acumulada crece, termina cayendo rápidamente.

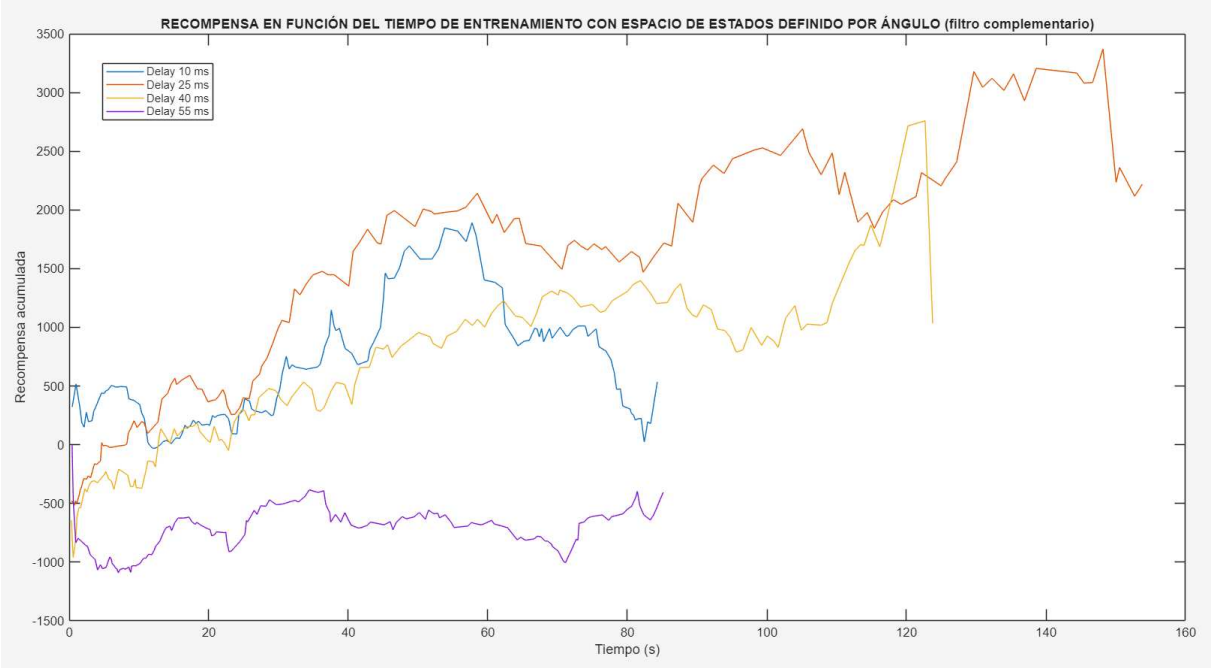

Figura 6.12. Recompensa acumulada en función del tiempo real de entrenamiento para distintos retardos con espacio de estados definido por ángulo (filtro complementario).

Los resultados sobre el tiempo en equilibrio durante el entrenamiento refuerzan lo anterior. En la **Figura 6.13**, los valores de retardo de 25 ms y 40 ms permiten alcanzar picos cercanos a los **1900 ms** (señal suavizada, por lo que los picos reales son mayores). Aunque estos picos también se alcanzan en los casos anteriores, en esta ocasión son menos esporádicos. El retardo de 10 ms muestra una tendencia creciente durante buena parte del entrenamiento aunque al final vuelve a los valores iniciales, mientras que el de 55 ms se mantiene por debajo de 700 ms en la mayoría de episodios.

Nótese el corto espacio de tiempo que el robot es capaz de mantener el balanceo incluso con esta definición de la tarea; se quiere recordar que los objetivos de este trabajo no incluían la obtención de un algoritmo de balanceo comparable en efectividad final a un control clásico, sino que se enfocaban en el análisis de la posibilidad de implementar un aprendizaje por refuerzo en una máquina con fuertes limitaciones computacionales.

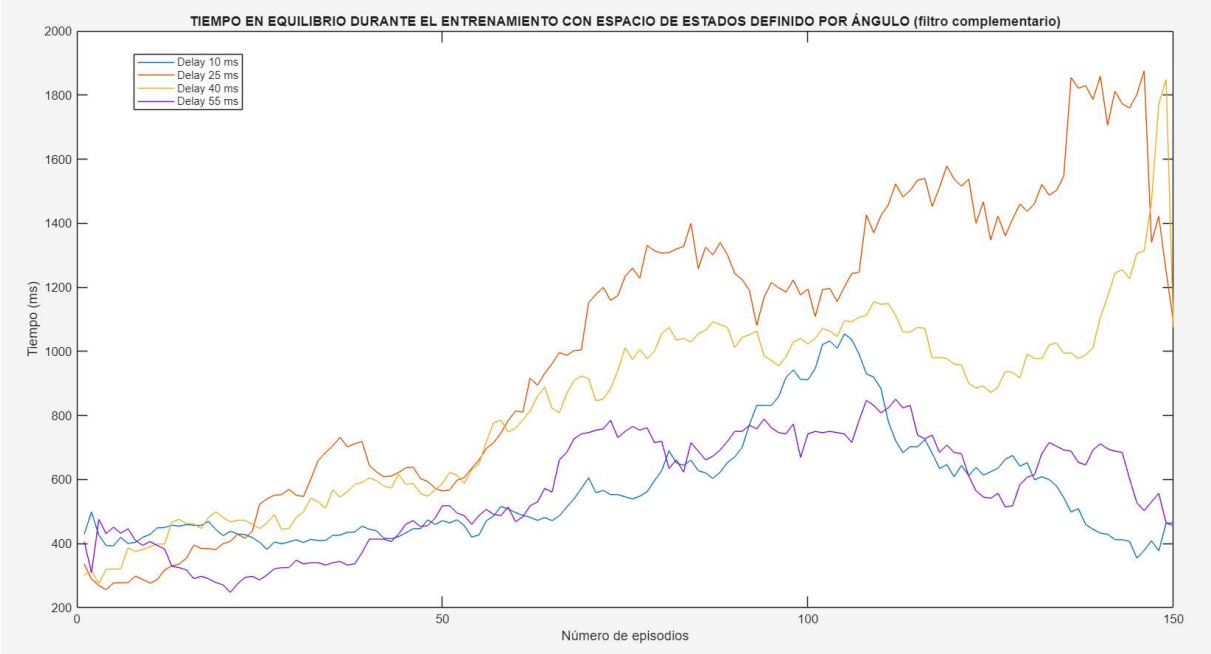

Figura 6.13. Tiempo en equilibrio del robot por episodio para distintos retardos con espacio de estados definido por ángulo (filtro complementario).

La **Figura 6.14** muestra el tiempo en equilibrio en función del tiempo de entrenamiento. El comportamiento de los valores intermedios de retardo vuelve a destacar sobre el resto, confirmando que se trata de configuraciones más adecuadas para este sistema.

En resumen, los valores intermedios de retardo (25–40 ms) logran un equilibrio óptimo entre la capacidad de reacción del sistema y la estabilidad de la señal estimada. Con retrasos demasiado bajos (10 ms), es muy probable que al agente no le dé tiempo de distinguir los distintos efectos de las posibles acciones disponibles, lo que genera un aprendizaje errático, **que necesitará de entrenamientos mucho más largos**. Por el contrario, con retrasos elevados (55 ms), la latencia acumulada hace que el agente actúe tarde, generando un comportamiento incorrecto.

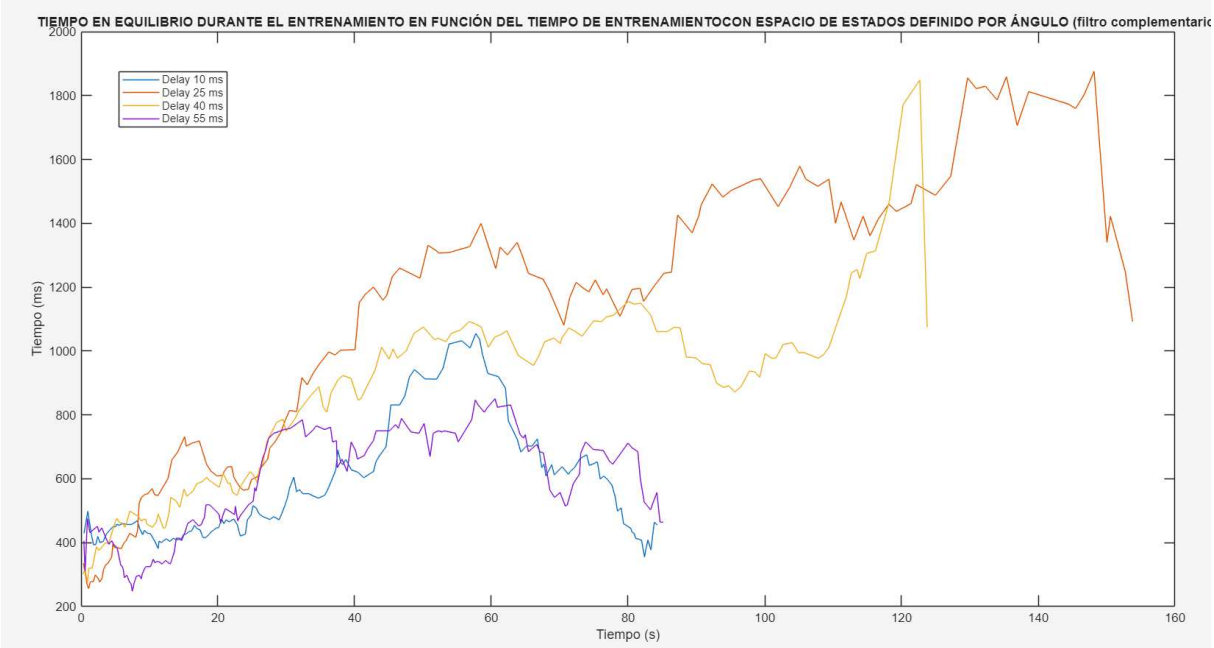

Figura 6.14. Tiempo en equilibrio del robot en función del tiempo de aprendizaje real para distintos retardos con espacio de estados definido por ángulo (filtro complementario).

Para validar los aprendizajes obtenidos durante el entrenamiento, se realizó una **fase de explotación pura** ($\epsilon = 0$) para los cuatro valores de retardo evaluados, registrando 20 episodios consecutivos en cada caso. Los resultados obtenidos en cuanto a recompensa acumulada y tiempo en equilibrio por episodio se muestran mediante diagramas de caja en las figuras siguientes.

La **Figura 6.15** presenta la recompensa acumulada por episodio en la fase de explotación. Como puede observarse, los casos de **25 ms y 40 ms** obtienen recompensas significativamente más altas, con medianas en torno a 2500 y 2800 respectivamente, siendo la dispersión del caso de 25 ms inferior a la del caso de 40 ms. El retardo de **10 ms** muestra un rendimiento mucho más pobre, y el de **55 ms** presenta incluso valores negativos en varios episodios, confirmando el mal comportamiento ya detectado durante el entrenamiento.

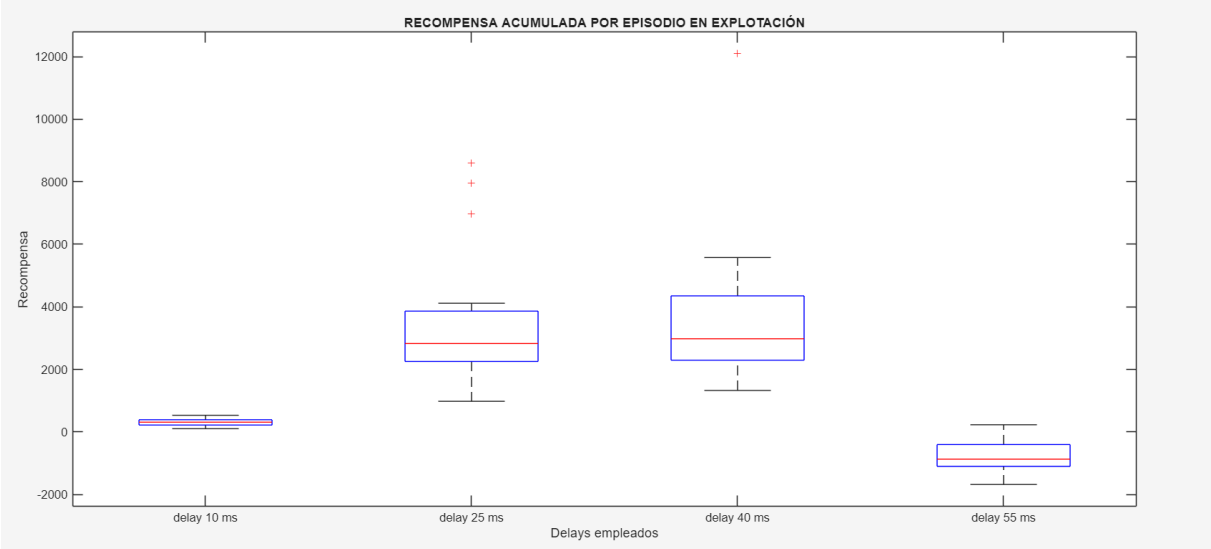

Figura 6.15. Recompensa acumulada por episodio para distintos retardos en explotación con espacio de estados definido por ángulo (filtro complementario).

La **Figura 6.16** muestra el tiempo en equilibrio por episodio durante la misma fase. Nuevamente, los valores de retardo intermedios obtienen los mejores resultados, con tiempos medios por encima de 1700 ms en el caso de 40 ms. Los retardos de 10 y 55 ms vuelven a ser los menos eficiente, con una mediana por debajo de 700 ms.

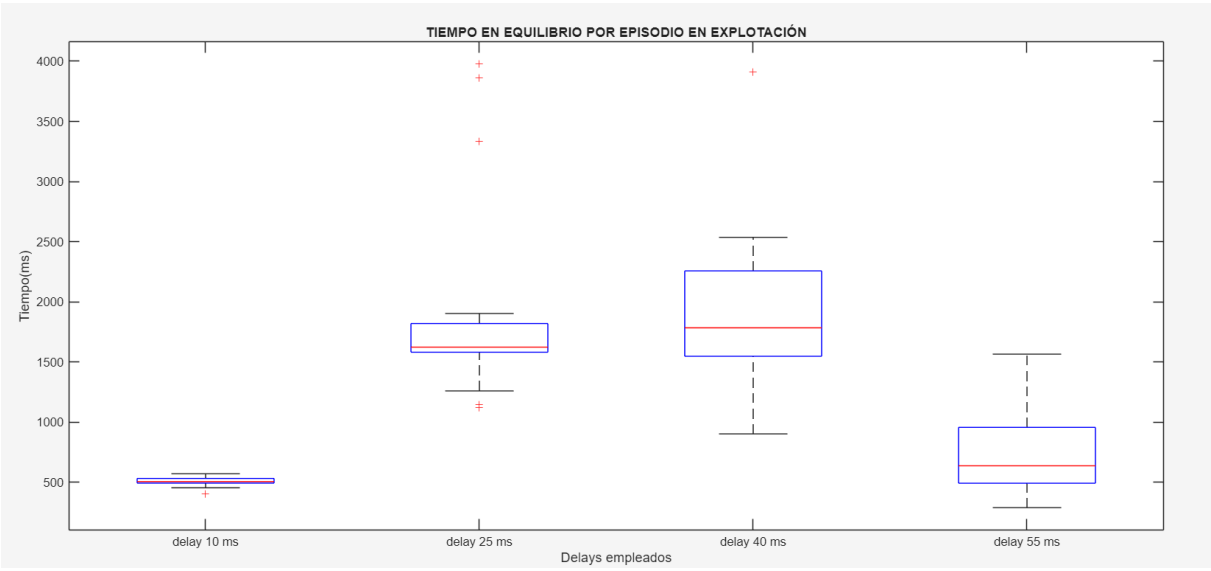

Figura 6.16. Tiempo en equilibrio del robot por episodio para distintos retardos en explotación con espacio de estados definido por ángulo (filtro complementario).

Estos resultados consolidan la conclusión obtenida durante el entrenamiento: **los valores intermedios de retardo (25 ms y 40 ms)** representan la mejor opción para lograr un control

eficaz y estable en este sistema. Retrasos demasiado bajos o excesivos dificultan la sincronización entre percepción y acción, perjudicando seriamente tanto el aprendizaje como el rendimiento final del agente.

Para cada uno de estos casos, se incluyen las tablas Q finales tras la fase de entrenamiento completa en el ANEXO V.

Por último, para ilustrar de forma visual el comportamiento del sistema, se adjunta un vídeo[1] que recoge una serie de demostraciones en las que el robot ejecuta la política aprendida durante la **fase de explotación con un retardo de 25 ms**, debido a que dicha configuración ha ofrecido buenos resultados visuales de balanceo. En estas grabaciones puede apreciarse cómo el agente, partiendo siempre de la misma condición inicial, consigue mantener el equilibrio. Aunque en algunos episodios se observa un desplazamiento lateral progresivo, el control sobre el ángulo de inclinación se mantiene estable, validando así el funcionamiento del sistema en condiciones reales, aunque aún lejos del rendimiento que ofrecen otros sistemas de control clásicos como el PID.

6.5. Comparación de resultados en fase de explotación

Una vez finalizado el proceso de entrenamiento, únicamente los enfoques 1 y 3 generaron políticas suficientemente estables como para ser evaluadas en una fase de explotación.

Con el objetivo de evitar posibles errores derivados de la necesidad de aplicar compensaciones temporales, la comparación se ha realizado empleando como única métrica el **tiempo medio en equilibrio** obtenido durante la ejecución de la política aprendida en condiciones reales. Esta medida directa y empírica resulta especialmente útil para valorar la efectividad práctica de cada enfoque, independientemente de cómo se haya definido la función de recompensa o del retardo exacto empleado durante el entrenamiento.

[1] Enlace al vídeo: https://shorturl.at/GH2cJ

CASO DE ESTUDIO	Retardo (ms)	Mediana de tiempo en equilibrio (ms)	Percentil 25 de tiempo en equilibrio (ms)	Percentil 75 de tiempo en equilibrio (ms)	Máximo (ms)
Espacio de estados definido por ángulo y velocidad angular	25	1718.75	1320	2172.5	6050
Espacio de estados definido por ángulo (filtro complementario)	10	507	494	533	572
	25	1624	1582	1820	3976
	40	1784.5	1548	2257.5	3913
	55	638	493	957	1566

Tabla 6.1. Comparación de la explotación para los distintos casos de estudio.

En el espacio de estados basado **únicamente en el ángulo** estimado mediante un filtro complementario (enfoque 3), se observó un comportamiento especialmente sólido. Tanto con 25 ms como con 40 ms de retardo, los resultados fueron consistentemente elevados, con **medianas superiores a 1600 ms** y percentiles altos que evidencian una notable capacidad de mantener el equilibrio. Además, ambos casos fueron entrenados con solo 150 episodios, lo que refuerza su eficiencia y viabilidad práctica.

En cuanto al espacio de estados definido por **ángulo y velocidad angular**, el caso con **25 ms** alcanzó el **mayor valor máximo individual (6050 ms)**, demostrando su potencial. No obstante, este resultado tiene un matiz importante: para alcanzar dicho rendimiento se necesitaron **500 episodios de entrenamiento**, frente a los 150 utilizados en el resto de casos. Además, la dispersión fue considerablemente mayor, con episodios que alternaban buenos tiempos con caídas más frecuentes, lo que sugiere **mayor dificultad de convergencia y menor previsibilidad, teniendo además un mayor coste temporal de entrenamiento**.

En conjunto, los resultados obtenidos permiten concluir que el **enfoque 3**, basado en el ángulo estimado mediante filtro complementario, **ofrece el mejor compromiso entre rendimiento, estabilidad y eficiencia de entrenamiento**. En particular, las configuraciones con retardos de 25 ms y 40 ms han demostrado ser las más eficaces, logrando un equilibrio prolongado con baja variabilidad entre episodios y un número reducido de episodios de entrenamiento en comparación al primer enfoque.

7. Conclusiones y líneas de trabajo futuras

El presente capítulo recoge, por un lado, las conclusiones generales derivadas del desarrollo y análisis del proyecto, y por otro, las posibles líneas de continuación o mejora que podrían abordarse en futuros trabajos.

7.1. Conclusiones

Este proyecto ha demostrado la viabilidad de implementar un algoritmo de aprendizaje por refuerzo directamente sobre un microcontrolador de muy bajas prestaciones, sin recurrir a simulaciones ni sistemas de asistencia externa. A lo largo del trabajo, se ha llevado a cabo el estudio detallado del hardware del robot Balboa 32U4, así como el diseño y ejecución de un sistema completo de entrenamiento en tiempo real. El algoritmo Q-Learning se ha adaptado e implementado íntegramente en lenguaje C, respetando las limitaciones de memoria, frecuencia de muestreo y procesamiento propias de la plataforma utilizada.

Además, se ha explorado el impacto de distintas configuraciones del espacio de estados, permitiendo no solo validar el funcionamiento del algoritmo, sino también **comparar tres enfoques diferentes** en cuanto a precisión, estabilidad y eficiencia. Esta comparación ha puesto de manifiesto que ciertos enfoques (especialmente aquellos basados únicamente en el ángulo con filtro complementario y retardos intermedios) pueden generar políticas suficientemente estables para mantener el equilibrio del robot sin asistencia, incluso con un número reducido de episodios.

Pese a estos avances, los resultados obtenidos se encuentran lejos del rendimiento que ofrecen enfoques clásicos como el control PID, o de otros algoritmos de aprendizaje por refuerzo aplicados en simulación. Esta diferencia se debe a varias limitaciones fundamentales del sistema:

- El tipo de algoritmo que pueden soportar estos sistemas es tabular, que son especialmente sensibles a la observabilidad parcial producida por el ruido en los sensores.
- La **limitada memoria del microcontrolador** ha obligado a trabajar con espacios de estados y de acciones discretizados, lo que reduce la sensibilidad del agente a variaciones sutiles del entorno.
- El **entrenamiento se ha realizado íntegramente en el sistema físico**, sin apoyo de simuladores. Esto ha limitado el número total de episodios de entrenamiento a decenas o cientos, cuando en condiciones simuladas suelen emplearse miles o decenas de miles para obtener resultados estables y reproducibles.
- Como consecuencia de la limitada memoria**, no ha sido posible incorporar otras variables relevantes**, como el desplazamiento lateral. Durante el entrenamiento se ha observado que el robot tiende a desplazarse lateralmente a medida que mejora su capacidad para mantenerse en equilibrio. Esta dinámica no se refleja en el espacio de estados ni en la función de recompensa, pero tiene un impacto directo sobre la estabilidad a largo plazo.

En la Figura 7.1 se muestra una secuencia típica donde el ángulo del robot (en grados) se mantiene estable durante un tiempo, mientras que la posición angular de las ruedas (medida en pulsos del *encoder*) disminuye progresivamente. Sin embargo, a partir de cierto punto, se produce un **aumento abrupto en los pulsos**, coincidiendo con una caída repentina del ángulo. Esto indica que se ha producido un desplazamiento lateral brusco que **termina provocando la caída del robot**. Este fenómeno evidencia que, aunque el agente consigue mantener el equilibrio momentáneamente, la acumulación de pequeños desplazamientos no controlados deriva en fallos súbitos, lo que subraya la necesidad de considerar esta variable en desarrollos futuros.

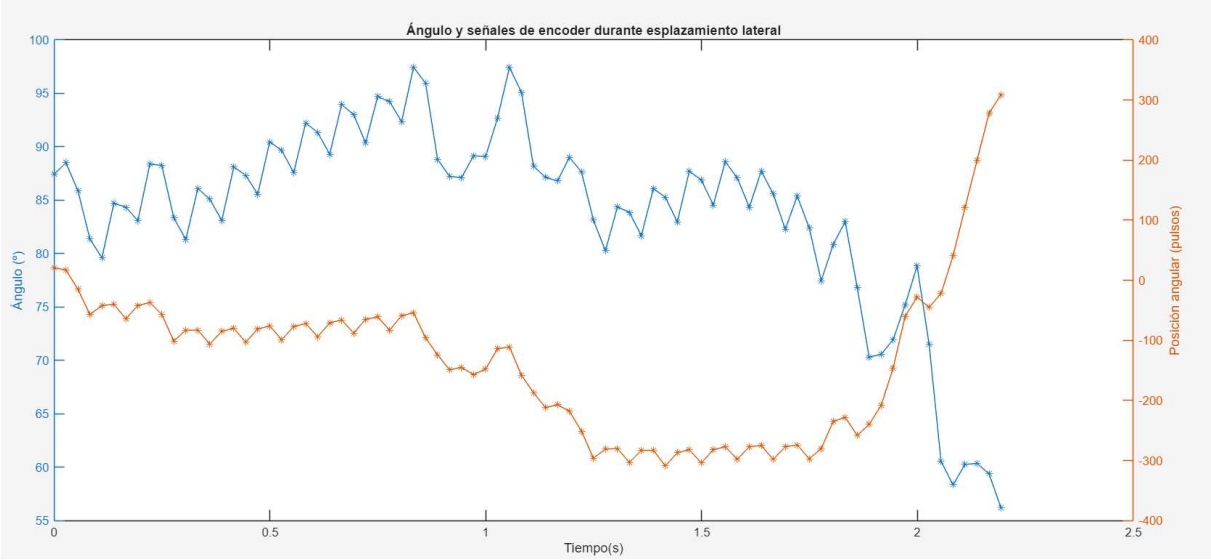

Figura 7.1. Ángulo y pulsos del encoder durante desplazamiento lateral.

Finalmente, **parte de los resultados de este trabajo han sido integrados como parte de un artículo científico que será enviado próximamente para su revisión en una revista científica internacional**, lo cual refuerza el valor académico y experimental de la investigación llevada a cabo.

7.2. Líneas de trabajo futuro

A partir de las conclusiones obtenidas, pueden plantearse varias líneas de trabajo con el objetivo de superar algunas de las limitaciones detectadas y explorar nuevas posibilidades:

- **Aumentar la memoria del sistema**, por ejemplo, mediante el uso de una Raspberry Pi auxiliar como se comentó en el Apartado 3.1, lo suficiente como para permitir:
    - La inclusión de nuevos sensores como los *encoders*, que podrían utilizarse para detectar desplazamientos laterales.
    - La ampliación del espacio de estados con nuevas variables.
    - La definición de un espacio de acciones más preciso, lo que podría mejorar la calidad de las políticas aprendidas.

- **Investigar técnicas de optimización de memoria en Arduino** que permitan reducir el espacio ocupado por la tabla Q. Algunas posibles líneas serían:

  - Tablas dispersas (*sparse*).
  - Codificar los valores con menor resolución o con formatos alternativos.

- **Explorar plataformas de hardware alternativas** que permitan mantener el espíritu del proyecto (aprendizaje en sistemas embebidos), pero con mejoras en robustez y memoria. Se han considerado tres opciones:

  - **Elegoo Tumbler** [38], con estructura similar pero basada en ATmega328P, aún más limitado que el Balboa32U4.
  - **Bala2-Fire** [39], basado en ESP32 (520 KB de RAM), que permite más capacidades pero pierde el carácter de sistema embebido de muy bajos recursos.
  - **Desarrollar un robot propio desde cero**, seleccionando estructura, motores, sensores y microcontrolador en función de los requisitos de aprendizaje y control. De esta manera, se podría mejorar la calidad y robustez del hardware, ya que el robot Balboa32U4 empleado en este proyecto sufrió dos averías críticas durante el desarrollo (una por unidad), lo que afectó negativamente a la continuidad del entrenamiento y la recogida de resultados. Una estructura más resistente permitiría ciclos de entrenamiento más intensos y fiables y además, el desarrollar el robot desde cero, a pesar de tener una dificultad añadida, permitiría otorgar al sistema las cualidades deseadas.

- **Rediseñar la tabla Q** para agrupar todos los estados terminales en un único estado de caída. Esta simplificación permitiría ahorrar memoria y mejorar la resolución en las zonas críticas cercanas al equilibrio.
- **Explorar el uso de simuladores para acelerar el entrenamiento**. Una vía prometedora consiste en implementar un modelo del sistema en **Simulink Multibody** de MATLAB, incluyendo tanto la dinámica del robot como los sensores relevantes. Esto permitiría aplicar técnicas de entrenamiento intensivo utilizando la *Reinforcement Learning Toolbox*, que ofrece herramientas nativas para definir entornos, agentes y funciones de recompensa. Una vez entrenado el

modelo en simulación, se podría exportar la política aprendida al entorno físico, evaluando su transferencia y adaptabilidad al hardware real. Esta estrategia permitiría comparar directamente los planteamientos físico y simulado, y abriría la puerta a nuevas optimizaciones sin comprometer la integridad del sistema embebido.

Referencias

[1] Pololu, Balboa 32U4 Balancing Robot User's Guide, 2019. [Online]. Available: https://www.pololu.com/docs/0J70

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[3] K. J. Aström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC: ISA, 2006.

[4] Microchip Technology Inc., *ATmega32U4: 8-bit AVR microcontroller with 32 KB ISP flash, USB controller and 10-bit ADC [Datasheet]*, 2021. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf

[5] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots" in Proc. *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7989163

[6] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection" *Int. J Robot. Res.*, vol. 37, no. 4-5, Apr. 2018. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0278364917710318

[7] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, Jan. 2000. [Online]. Available: https://doi.org/10.1162/089976600300015961

[8] T. Haarnoja, S. M. Jayaraman, J. Tan, V. Kumar, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2017. [Online]. Available: https://proceedings.mlr.press/v70/haarnoja17a.html

[9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, Sep. 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721

[10] "Aprendizaje por refuerzo en lenguaje C" *Medium,* Jun. 15, 2020. [Online]. Available: https://medium.com/swlh/q-learning-using-c-language-f072fab75e1b

[11] Pololu, "Balboa 32U4 Balancing Robot Kit (No Motors or Wheels)," *Pololu*, 2024. [Online]. Available: https://www.pololu.com/product/3575

[12] Pololu, "Pololu Wheel 80×10mm Pair – Black," *Pololu*, 2024. [Online]. Available: https://www.pololu.com/product/1430

[13] Pololu, "Micro Metal Gearmotor HP 6V with 75:1 gearbox," *Pololu*, 2024. [Online]. Available: https://www.pololu.com/category/60/micro-metal-gearmotors

[14] Bricogeek, "Tienda de electrónica y robótica", [Online]. Available: https://www.bricogeek.com. [Accessed: 22-Jun-2025].

[15] Arduino, "Arduino IDE 2.3.2," *Arduino Software*, 2024. [Online]. Available: https://www.arduino.cc/en/software

[16] Pololu, “Balboa 32U4 Arduino Library” *Pololu GitHub Pages*, 2024. [Online]. Available: https://pololu.github.io/balboa-32u4-arduino-library/
[17] Pololu, “LSM6 Arduino Library” *Pololu GitHub Pages,* 2024. [Online]. Available: https://github.com/pololu/lsm6-arduino
[18] MathWorks, “MATLAB R2024a,” *MathWorks*, 2024. [Online]. Available: https://www.mathworks.com/products/matlab.html
[19] Processing Foundation, “Processing 4,” *Processing.org*, 2024. [Online]. Available: https://processing.org/
[20] Wikipedia, “Aprendizaje por refuerzo” Wikipedia, la enciclopedia libre, May 2024. [Online]. Available: https://es.wikipedia.org/wiki/Aprendizaje_por_refuerzo
[21] M. Wawrzynski, *Grokking Deep Reinforcement Learning*. Manning Publications, 2021.
[22] Wikipedia, “Q-learning,” *Wikipedia, la enciclopedia libre*, May 2024. [Online]. Available: https://es.wikipedia.org/wiki/Q-learning
[23] Wikipedia, “State–action–reward–state–action,” *Wikipedia*, May 2024. [Online]. Available: https://es.wikipedia.org/wiki/Estado-Acci%C3%B3n-Recompensa-Estado-Acci%C3%B3n_(SARSA)
[24] Wikipedia, “Policy gradient method,” *Wikipedia*, May 2024. [Online]. Available: https://en.wikipedia.org/wiki/Policy_gradient_method
[25] R. J. Williams, “REINFORCE Algorithm,” in *Reinforcement Learning: An Introduction*, Sutton & Barto, 2018.
[26] Sutton & Barto, *Reinforcement Learning*, cap. 2.4 – "Exploration and exploitation".
[27] Hugging Face, “Deep RL Course” *Huggin Face* – Deep RL Course, 2024. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unit2/
[28] S. Sivananthan, *AVR Microcontroller and Embedded Systems: Using Assembly and C for ATMega32*, Pearson Education, 2011.
[29] Raspberry Pi Foundation, “Raspberry Pi 3 Model B,” *Raspberry Pi Documentation*, 2024. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/
[30] Microchip Technology Inc., “Company Overview,” *Microchip Technology*, 2024. [Online]. Available: https://www.microchip.com/
[31] STMicroelectronics, *LIS3MDL: Ultra-low-power three-axis magnetometer*, 2024. [Online]. Available: https://www.st.com/en/mems-and-sensors/lis3mdl.html
[32] Pololu, *Pololu LSM6DS33 3-Axis Accelerometer and Gyro Carrier,* 2024. Available: https://www.pololu.com/product/2736
[33] Tsui, “What is Exponential Moving Average (EMA)?”, *Towards Data Science*, Jul. 2019. [Online]. Available: https://towardsdatascience.com/what-is-exponential-moving-average-ema-5973c5c162f4
[34] P. Lutz, “Complementary Filter vs Kalman Filter,” *All About Circuits*, Dec. 2016. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/complementary-filter-vs-kalman-filter/
[35] Arduino, “micros() function,” *Arduino Reference*, 2024. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/time/micros/

[36] Wikipedia, “Polling (informática),” *Wikipedia, la enciclopedia libre*, 2024. [Online]. Disponible en: https://es.wikipedia.org/wiki/Polling

[37] Microsoft, “Mklink,” *Microsoft Learn*, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/mklink

[38] ELEGOO, *ELEGOO Tumbller Self-Balancing Robot Car Kit with Remote Control V2.0*, 2023. [Online]. Available: https://www.elegoo.com/blogs/arduino-projects/elegoo-tumbller-self-balancing-robot-car-tutorial?srsltid=AfmBOorwnvvGE2L2ZAx-XjPaeB8M4GYHf3QgJwpoojYc9B4ZjEXoPkZk

[39] FIRE, *Bala2-Fire: Open Source Self-Balancing Robot with ESP32 and OLED*, 2022. [Online]. Available: https://docs.m5stack.com/en/app/bala2fire

ANEXO I. Esquema de pines del Balboa 32U4

El esquema que se muestra a continuación presenta la distribución completa de pines del Balboa 32U4, incluyendo asignaciones digitales, entradas analógicas, salidas PWM, interrupciones externas, buses de comunicación (I2C, SPI, UART), y periféricos integrados como LEDs, botones y zumbador. Esta información resulta de gran utilidad para el desarrollo de software embebido, depuración de señales, y comprensión general de la arquitectura de la placa.

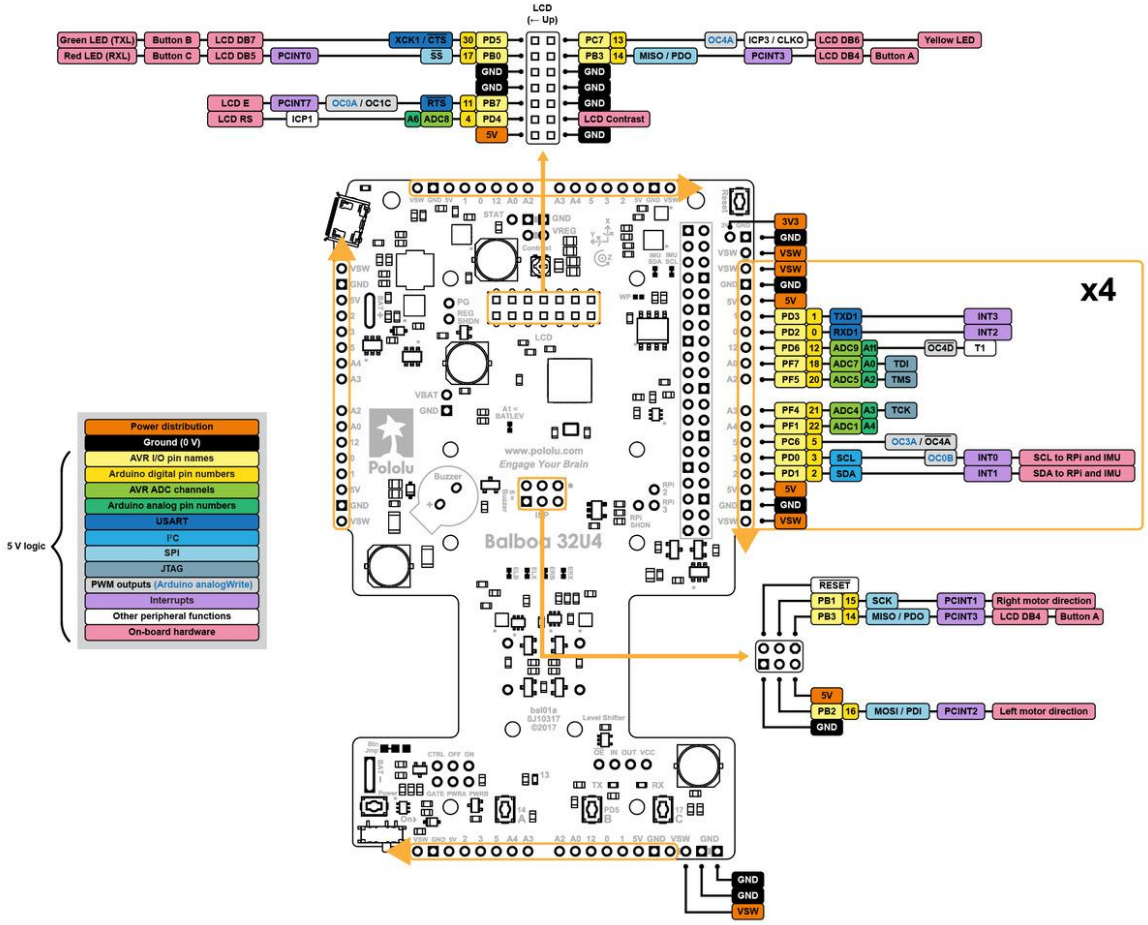

Figura A1.1. Distribución de pines del Balboa 32U4 [1].

ANEXO II. Bases teóricas de los filtros aplicados

En este anexo se desarrollan los fundamentos teóricos y formulación matemática necesaria para los filtros mencionados en la Sección 3.5.4, estos han sido usados en este proyecto para obtener el ángulo de inclinación del robot Balboa 32U4 minimizando problemas como el ruido y la deriva del error, que son propios de las señales que ofrecen los sensores sin tratar.

1. Filtro exponencial

El **filtro exponencial** (o filtro de media móvil ponderada exponencialmente, EWMA por sus siglas en inglés) es un filtro recursivo de primer orden ampliamente utilizado para atenuar ruido en señales discretas. Su sencillez lo hace especialmente útil en sistemas embebidos, como el utilizado en este proyecto.

1.1. Formulación en tiempo discreto

La versión discreta del filtro se define como una **ecuación de diferencias** (véase Ecuación (A2.1)).

$$y_k = \alpha_{fexp} * x_k + \left(1 - \alpha_{fexp}\right) * y_{k+1} \qquad \text{(A2.1)}$$

Donde:

- y_k: salida filtrada en el instante k,
- x_k: entrada bruta en el instante k,
- y_{k+1}: salida anterior,
- $\alpha_{fexp} \in (0,1)$: coeficiente de filtrado.

Esta expresión define un filtro IIR (*Infinite Impulse Response*) de primer orden, ya que la salida depende tanto del valor actual de la entrada como del valor anterior de la salida. El sistema tiene "memoria", y esta memoria es controlada directamente por el parámetro α_{fexp}.

1.2. Obtención a partir del dominio Z

La ecuación puede transformarse en el dominio Z, obteniendo su función de transferencia:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\alpha_{fexp}}{1 - (1 - \alpha_{fexp})z^{-1}} \quad \text{(A2.2)}$$

Esta función de transferencia representa un filtro paso bajo digital, con una respuesta de primer orden. Su polo se encuentra en $z = 1 - \alpha_{fexp}$, lo cual garantiza estabilidad para cualquier $\alpha_{fexp} \in (0,1)$, como ya se había mencionado.

1.3. Respuesta en frecuencia

La magnitud de la respuesta en frecuencia se obtiene evaluando la función de transferencia sobre el eje unitario $z = e^{j\omega}$:

$$\left|H(e^{j\omega})\right| = \frac{\alpha_{fexp}}{\sqrt{1 - 2 + (1 - \alpha_{fexp})\cos(\omega) + (1 - \alpha_{fexp})^2}} \quad \text{(A2.3)}$$

Este análisis permite ver cómo el filtro atenúa las componentes de alta frecuencia (grandes ω) mientras que conserva las bajas (pequeño ω). Es decir, suprime el ruido de alta frecuencia, como el proveniente de sensores como el acelerómetro.

1.4. Interpretación del coeficiente α_{fexp}

- Si $\alpha_{fexp} \to 1$: el filtro responde rápidamente a cambios, pero deja pasar más ruido.
- Si $\alpha_{fexp} \to 0$: el filtro suaviza muy bien, pero reacciona lentamente.

El retardo aproximado viene dado por la siguiente expresión:

$$T_{retardo} \approx \frac{1}{\alpha_{fexp} * f_s} \quad \text{(A2.4)}$$

Donde f_s es la frecuencia de muestreo.

En este proyecto, se han utilizado valores de $\alpha_{fexp} = 0.05, 0.10, 0.15$, evaluando su efecto sobre la señal de ángulo estimada a partir del acelerómetro.

1.5. Ventajas y desventajas del filtro exponencial

A continuación, se presentan las ventajas y desventajas de usar el filtro exponencial (véase Tabla A2.1).

VENTAJAS	DESVENTAJAS
Muy eficiente computacionalmente	Introduce retardo proporcional al suavizado
Fácil de ajustar mediante un único parámetro	No es adecuado para señales que requieren seguimiento dinámico rápido
Ideal para señales con tendencia suave y ruido blanco	No corrige errores acumulativos como la deriva de integración del giróscopo

Tabla A2.1. Ventajas y desventajas del filtro exponencial.

2. Filtro complementario

El **filtro complementario** es una técnica de fusión sensorial que combina señales de distinta naturaleza aprovechando sus **características complementarias** en el dominio de la frecuencia. Su objetivo es obtener una estimación más precisa y robusta de una magnitud física, (en este caso, el ángulo de inclinación), a partir de la información conjunta de dos sensores: un acelerómetro (ruidoso pero estable a largo plazo) y un giróscopo (suave a corto plazo, pero propenso a deriva de error).

A diferencia del filtro exponencial, que actúa sobre una única señal, el filtro complementario realiza una **combinación ponderada de dos señales** con propiedades frecuencias opuestas.

2.1. Fundamento teórico

En el dominio continuo, el principio del filtro complementario se basa en la combinación de un filtro paso bajo (aplicado a la señal del acelerómetro) y un filtro paso alto (aplicado a la señal integrada del giróscopo), con la propiedad de:

$$Paso\ bajo + Paso\ alto = 1 \tag{A2.5}$$

Esta propiedad asegura que se mantiene el contenido total de la señal original, repartido entre dos fuentes.

2.2. Formulación en tiempo discreto

El filtro complementario en su versión recursiva discreta se expresa como:

$$\theta_k = \alpha_{fcom} * (\theta_{k-1} + \omega_k * \Delta t) + \left(1 - \alpha_{fcom}\right) * \theta_{acc,k} \tag{A2.6}$$

Donde:

- θ_k: estimación del ángulo en el instante k,
- θ_{k-1}: estimación anterior,
- ω_k: velocidad angular medida por el giróscopo,
- Δt: intervalo de muestreo (en el caso del aprendizaje, será el tiempo de cada paso en un episodio),
- $\theta_{acc,k}$: ángulo estimado a partir del acelerómetro,
- $\alpha_{fcom} \in (0, 1)$: coeficiente de filtrado (típicamente cercano a 1).

Esta expresión combina la predicción del ángulo a partir del giróscopo integrado (componente de alta frecuencia) con una corrección basada en la medida del acelerómetro (componente de baja frecuencia).

2.3. Interpretación del coeficiente α_{fcom}

Un elemento esencial para comprender el funcionamiento del filtro complementario es el efecto que tiene el valor de α_{fcom}:

- $\alpha_{fcom} \approx 1$: el filtro confía principalmente en el giróscopo y corrige solo lentamente los errores acumulados.
- $\alpha_{fcom} \approx 0$: el filtro confía más en el acelerómetro, eliminando la deriva, pero incorporando ruido.

El parámetro α_{fcom} representa una constante de fusión entre dos fuentes de información, ajustando el equilibrio entre estabilidad y velocidad. En este proyecto se ha utilizado un valor constante de $\alpha_{fcom} = 0.98$, una elección habitual en aplicaciones de fusión sensorial con sensores inerciales, ya que proporciona una respuesta rápida sin dejar de corregir la deriva. Además, esta configuración ha demostrado ofrecer resultados especialmente satisfactorios en las pruebas realizadas con el sistema.

2.4. Relación con la respuesta en frecuencia

En términos de frecuencia, el filtro complementario puede verse como una implementación práctica de:

$$\theta(s) = H_{HP}(s) * \theta_{gyro}(s) + H_{LP}(s) * \theta_{acc}(s) \qquad \text{(A2.7)}$$

Donde:

- $H_{HP}(s) = \frac{\tau s}{\tau s+1}$→Filtro de paso alto,
- $H_{LP}(s) = \frac{1}{\tau s+1}$→Filtro de paso bajo,
- $H_{HP}(s) + H_{LP}(s) = 1$

Al discretizar esta relación usando métodos como Euler, se obtiene precisamente la forma recursiva implementada en la Ecuación (A2.6).

2.5. Ventajas y desventajas del filtro complementario

A continuación, se presentan las ventajas y desventajas de usar el filtro complementario (véase Tabla A2.2).

VENTAJAS	DESVENTAJAS
Reduce simultáneamente el ruido del acelerómetro y la deriva del error	El valor de α_{fcom} no se adapta automáticamente a las condiciones
Bajo coste computacional	No considera explícitamente incertidumbre, como sí haría un filtro de Kalman
Robusto frente a perturbaciones breves o errores acumulados sin inyectar retardo	No es capaz de corregir los efectos de aceleraciones lineales continuadas en el tiempo
Muy adecuado para sistemas con bajo poder de procesamiento, como microcontroladores de 8 bits	

Tabla A2.2. Ventajas y desventajas del filtro complementario.

ANEXO III. Repositorio de software

Con el objetivo de facilitar la comprensión, la reproducibilidad y la evaluación del sistema implementado, se ha habilitado un repositorio en GitHub que contiene todo el código fuente desarrollado para este Trabajo Fin de Máster. El repositorio está disponible en la siguiente dirección[2].

El contenido está organizado en tres bloques principales:

- **Arduino:** incluye los programas principales cargados en el microcontrolador Balboa 32U4. Se aportan tres versiones del algoritmo de aprendizaje (LEARN_FC, LEARN_Fexp y LEARN_FC_angleandgyro) y una versión específica para la fase de explotación (EXPLOTACION_FC). Cada versión refleja una configuración distinta en cuanto a filtros aplicados, variables que definen el estado y la forma de calcular la recompensa. Aunque estas variantes ya incorporan procesos como discretización, definición del espacio de estados o cálculos de recompensa, estos aspectos se detallan en profundidad en el Capítulo 5, centrado en el diseño lógico del sistema.
- ***Processing*:** contiene dos programas desarrollados para gestionar la comunicación entre Arduino y el PC por puerto serie. Uno de ellos permite recibir datos del microcontrolador y almacenarlos en un fichero de texto; el otro permite enviar datos desde el PC a Arduino para continuar el entrenamiento.
- **MATLAB:** recopila funciones utilizadas para cargar, procesar y representar los datos generados por el sistema. Incluye herramientas para guardar y recuperar matrices, simplificar la tabla Q para su uso en explotación y *scripts* específicos para representar gráficamente los resultados del entrenamiento y de la explotación. El repositorio proporciona ejemplos completos para ambos casos de representación gráfica.

La organización del código está descrita de forma más extensa en el archivo `README.md` incluido en el propio repositorio, donde también se indican las dependencias necesarias para cada entorno y se facilita la navegación entre carpetas.

[2] Repositorio disponible en: https://github.com/EduardoLucena01/balboa-qlearning-tfm

No se han incluido *scripts* auxiliares empleados para caracterizar el comportamiento físico del robot (Capítulo 3), ya que estos no forman parte directa del sistema de aprendizaje. El repositorio se ha centrado exclusivamente en el código necesario para entrenar, explotar y analizar la política aprendida mediante el algoritmo Q-Learning.

ANEXO IV. Ejecución de experimentos

Con el objetivo de representar de forma clara y resumida el funcionamiento general del sistema, se ha elaborado un flujograma centrado en la lógica de ejecución implementada en la placa Balboa 32U4, programada mediante el entorno Arduino. El esquema recoge los pasos fundamentales que se ejecutan durante la fase de aprendizaje (aunque es aplicable también a la fase de explotación, ya que comparten una estructura común, salvo por el valor de ϵ y la actualización de la tabla), e incorpora anotaciones que indican la participación del usuario y el papel de los entornos auxiliares *Processing* y MATLAB en los momentos clave.

El diagrama (véanse Figura A4.1, Figura A4.2 y Figura A4.3) refleja de forma operativa cómo se lleva a cabo un bloque de 10 episodios de aprendizaje, desde la inicialización del sistema hasta la exportación de datos.

No debe pasarse por alto que el flujograma es independiente de las decisiones específicas relacionadas con el diseño lógico del algoritmo de aprendizaje, como la elección de la función de recompensa, el método de discretización de los estados o el tipo de filtrado aplicado a las señales. Estos aspectos se detallan en el Capítulo 5, mientras que aquí se muestra exclusivamente la estructura general del funcionamiento del sistema durante el proceso de entrenamiento o explotación.

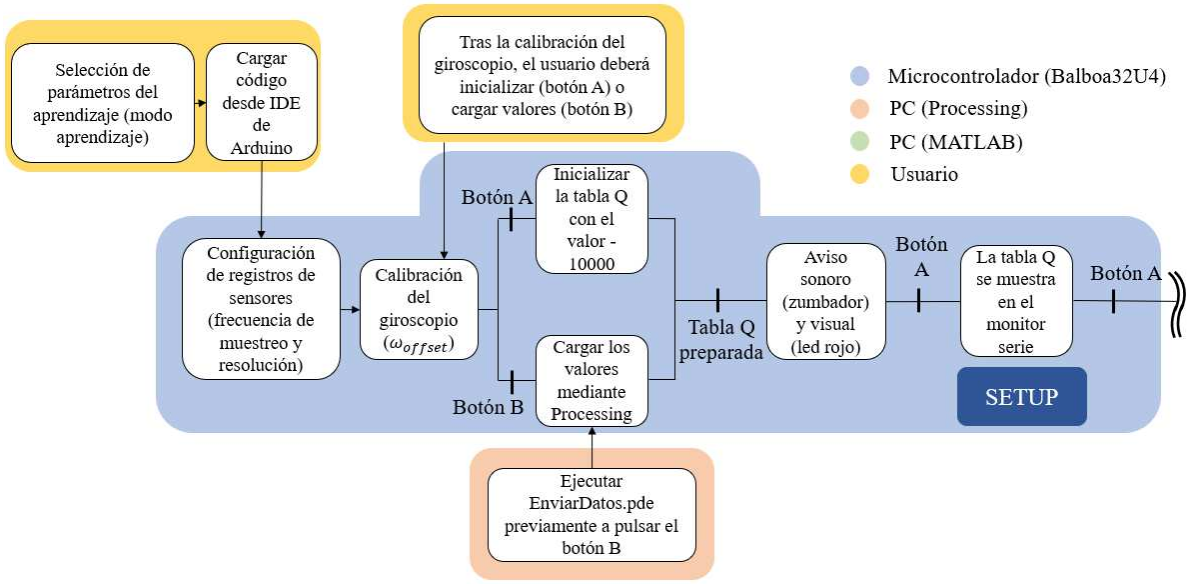

Figura A4.1. Flujo de trabajo durante el aprendizaje parte 1.

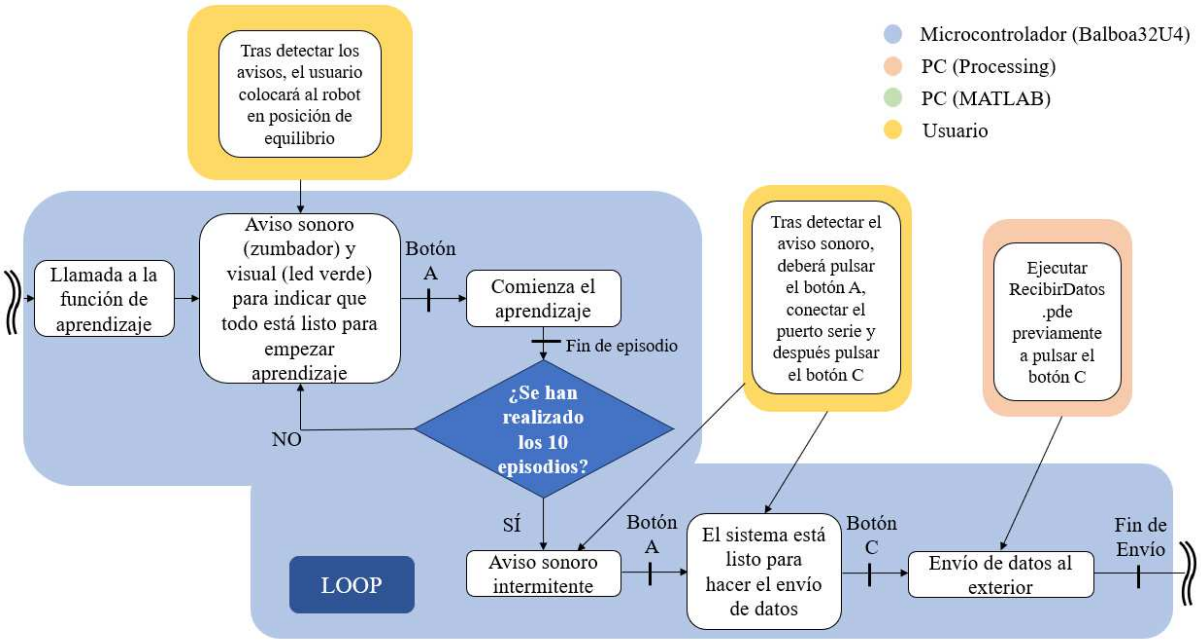

Figura A4.2. Flujo de trabajo durante el aprendizaje parte 2.

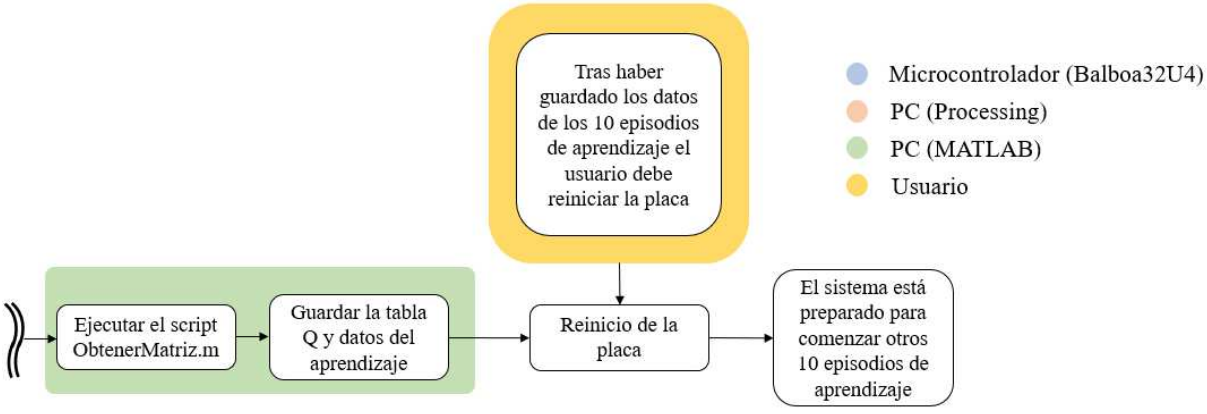

Figura A4.3. Flujo de trabajo durante el aprendizaje parte 3.

ANEXO V. Tablas Q

A continuación se presentan las tablas Q finales obtenidas tras el entrenamiento en cada uno de los enfoques evaluados en el capítulo 6.

Nota sobre las tablas Q:

En las tablas Q que se muestran a continuación, correspondientes a los distintos enfoques y configuraciones evaluadas en el capítulo 6, se puede observar que **las primeras y últimas filas contienen exclusivamente el valor −10000**. Este valor corresponde al inicializado al comienzo del entrenamiento, y no ha sido actualizado posteriormente.

Esto se debe a que dichas filas representan **estados terminales**, es decir, aquellos en los que el robot ha detectado que ha caído o se encuentra fuera de la zona operativa. En esos casos, **no se realiza ninguna transición posterior hacia otro estado**, ya que el episodio finaliza. Como consecuencia, el algoritmo no tiene oportunidad de actualizar esas entradas de la tabla Q, puesto que para ello sería necesario **tomar una acción válida y observar una transición hacia un nuevo estado**, lo cual no ocurre en los estados de caída.

Este comportamiento es **esperado y correcto** dentro del marco del algoritmo Q-learning, y confirma que el sistema respeta la lógica de finalización de episodios al alcanzar condiciones críticas.

Sin embargo, también puede darse el caso de que si no se ha realizado un número de episodios suficiente, debido a la mala evolución del aprendizaje, puedan encontrarse algún estado inexplorado.

			ACCIONES								
INT ÁNG	INT VEL	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:40°	-1000°/s : -150°/s	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
0°:40°	-150°/s : -50°/s	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
0°:40°	-50°/s : 50°/s	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
0°:40°	50°/s : 150°/s	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
0°:40°	50°/s : 1000°/s	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:60°	-1000°/s : -150°/s	5	2,93	-0,03	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:60°	-150°/s : -50°/s	6	-10000,00	1,12	-0,16	-10000,00	-0,40	13,02	-1,74	-10000,00	-10000,00
40°:60°	-50°/s : 50°/s	7	-0,99	-0,65	-2,57	-0,05	-2,42	-2,90	-2,00	9,91	-2,35
40°:60°	50°/s : 150°/s	8	-8,46	-7,46	-8,71	-8,16	-7,91	-7,47	-7,70	-1,43	-7,56
40°:60°	50°/s : 1000°/s	9	-16,01	-15,76	-15,81	-16,02	-16,52	-16,02	-16,33	-16,55	-16,89
60°:75°	-1000°/s : -150°/s	10	15,36	22,66	30,40	21,73	30,81	24,01	16,67	54,59	16,85
60°:75°	-150°/s : -50°/s	11	21,62	28,32	13,68	16,73	16,99	32,03	20,24	50,12	17,06
60°:75°	-50°/s : 50°/s	12	26,58	17,50	21,30	22,20	20,54	41,52	15,09	16,41	7,25
60°:75°	50°/s : 150°/s	13	4,22	10,64	11,23	10,23	10,67	10,44	11,08	41,24	4,57
60°:75°	50°/s : 1000°/s	14	6,21	6,18	6,32	20,37	12,88	28,54	11,04	24,76	0,50
75°:95°	-1000°/s : -150°/s	15	41,86	45,50	45,85	46,52	45,24	39,86	45,50	40,97	49,53
75°:95°	-150°/s : -50°/s	16	49,12	49,89	52,12	46,38	50,71	49,22	50,56	45,64	51,49
75°:95°	-50°/s : 50°/s	17	51,97	54,72	51,32	51,46	51,23	50,55	48,67	49,73	51,37
75°:95°	50°/s : 150°/s	18	49,74	50,89	46,29	53,36	49,16	48,21	50,71	48,83	49,18
75°:95°	50°/s : 1000°/s	19	45,42	46,87	48,26	46,41	40,16	53,35	40,94	48,72	38,21
95°:110°	-1000°/s : -150°/s	20	26,03	16,42	21,56	7,40	27,39	20,22	28,95	18,89	26,18
95°:110°	-150°/s : -50°/s	21	17,77	29,49	17,75	14,66	28,40	19,17	24,39	20,98	43,15
95°:110°	-50°/s : 50°/s	22	32,46	23,73	28,99	25,65	36,54	18,14	39,96	24,73	48,83
95°:110°	50°/s : 150°/s	23	36,36	37,06	39,66	33,84	39,67	31,23	49,87	19,87	42,80
95°:110°	50°/s : 1000°/s	24	34,48	51,74	38,92	43,49	18,15	31,81	33,08	20,02	37,47
110°:130°	-1000°/s : -150°/s	25	-14,27	-11,54	-12,31	-13,74	-10,07	-10,82	-11,36	-12,21	-8,71
110°:130°	-150°/s : -50°/s	26	-6,54	-5,59	-5,64	-8,70	-8,11	-7,59	-5,80	-6,58	-2,22
110°:130°	-50°/s : 50°/s	27	-1,14	-3,56	-2,26	-3,03	-1,55	-1,56	4,68	-2,01	16,09
110°:130°	50°/s : 150°/s	28	0,04	0,78	-10000,00	-0,54	22,56	-1,11	4,39	0,07	7,20
110°:130°	50°/s : 1000°/s	29	-10000,00	-10000,00	-10000,00	-10000,00	26,70	-10000,00	5,10	-10000,00	-10000,00
130°:175°	-1000°/s : -150°/s	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
130°:175°	-150°/s : -50°/s	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
130°:175°	-50°/s : 50°/s	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
130°:175°	50°/s : 150°/s	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
130°:175°	50°/s : 1000°/s	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.1. Tabla Q del enfoque 1.

		ACCIONES								
INT ÁNG	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:5°	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
5°:10°	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
10°:15°	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
15°:20°	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
20°:25°	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
25°:30°	5	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
30°:35°	6	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
35°:40°	7	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:45°	8	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
45°:50°	9	-6,74	-8,19	-6,88	-6,60	-6,58	-10000,00	-5,91	-6,82	-10000,00
50°:55°	10	-2,87	-2,24	-3,36	-4,06	-1,93	-3,80	-1,86	-2,22	-2,02
55°:60°	11	-10000,00	-3,13	-3,75	-2,00	-1,84	-2,56	-2,03	-1,99	1,03
60°:65°	12	-3,25	-1,88	-0,78	-2,08	0,33	-0,34	-2,26	-1,57	7,39
65°:70°	13	-1,26	-0,22	-1,42	0,65	11,81	-0,16	0,73	-2,32	-1,47
70°:75°	14	7,54	3,42	25,49	3,09	12,75	6,58	5,07	-0,85	10,34
75°:80°	15	16,35	20,71	12,21	13,39	37,51	10,32	28,70	20,78	26,11
80°:85°	16	42,13	40,46	38,55	30,04	41,17	39,59	40,67	31,75	49,14
85°:90°	17	45,30	44,18	41,14	46,93	41,77	44,25	44,81	45,96	40,97
90°:95°	18	11,87	23,39	19,92	31,84	24,12	28,87	14,68	45,94	15,78
95°:100°	19	0,99	1,84	1,18	9,11	2,78	5,31	0,29	33,97	1,57
100°:105°	20	2,80	-10000,00	0,03	2,31	11,97	3,36	0,03	0,07	-10000,00
105°:110°	21	-10000,00	1,83	-10000,00	0,87	4,39	-10000,00	1,98	1,39	-10000,00
110°:115°	22	-10000,00	-0,12	-10000,00	-0,08	-10000,00	-10000,00	-0,10	-10000,00	-10000,00
115°:120°	23	-0,11	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-0,10	-10000,00	-10000,00
120°:125°	24	-0,10	-10000,00	-10000,00	1,80	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
125°:130°	25	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
130°:135°	26	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
135°:140°	27	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
140°:145°	28	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
145°:150°	29	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
150°:155°	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
155°:160°	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
160°:165°	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
165°:170°	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
170°:175°	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.2. Tabla Q del enfoque 2.

		ACCIONES								
INT ÁNG	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:5°	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
5°:10°	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
10°:15°	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
15°:20°	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
20°:25°	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
25°:30°	5	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
30°:35°	6	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
35°:40°	7	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:45°	8	-125,46	-134,74	-124,63	-136,96	-138,14	-137,18	-131,41	-136,40	-135,78
45°:50°	9	-72,42	-97,02	-75,43	-74,38	-81,42	-71,22	-70,47	-82,99	-79,39
50°:55°	10	-33,88	-37,77	-55,46	-36,56	-34,52	-37,33	-37,11	-42,38	-64,84
55°:60°	11	-24,77	-25,55	-26,49	-24,52	-26,30	-25,30	-25,52	-24,59	-26,77
60°:65°	12	-26,62	-27,11	-28,02	-26,91	-26,80	-26,19	-29,31	-26,57	-27,30
65°:70°	13	35,43	3,95	-12,49	-2,82	-5,99	-10,86	-6,12	30,48	16,63
70°:75°	14	125,32	87,06	174,07	121,39	103,99	103,05	89,94	237,08	197,72
75°:80°	15	286,83	331,80	270,20	541,97	226,88	259,98	311,52	369,53	249,29
80°:85°	16	580,09	423,52	468,43	734,27	578,29	451,19	440,56	527,52	587,45
85°:90°	17	632,09	625,14	632,28	630,47	593,10	621,42	623,95	628,65	589,51
90°:95°	18	321,25	359,83	308,13	342,13	330,11	348,63	348,86	355,14	342,93
95°:100°	19	17,33	37,76	36,77	33,72	37,45	30,30	17,24	35,65	19,68
100°:105°	20	-29,53	-30,05	-32,64	-29,90	-28,26	-30,01	-29,48	-30,10	-30,52
105°:110°	21	-36,43	-35,92	-36,92	-36,05	-36,62	-37,00	-36,53	-36,79	-36,00
110°:115°	22	-36,62	-37,35	-37,98	-38,77	-39,21	-41,06	-47,97	-37,42	-41,69
115°:120°	23	-56,69	-56,68	-62,53	-61,87	-60,39	-69,56	-57,77	-57,89	-58,99
120°:125°	24	-105,70	-105,04	-111,67	-112,13	-115,07	-105,31	-109,84	-116,30	-116,66
125°:130°	25	-180,91	-189,13	-182,46	-182,31	-181,90	-186,58	-183,33	-186,13	-181,09
130°:135°	26	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
135°:140°	27	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
140°:145°	28	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
145°:150°	29	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
150°:155°	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
155°:160°	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
160°:165°	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
165°:170°	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
170°:175°	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.3. Tabla Q del enfoque con 3 retardo de 10 ms.

		ACCIONES								
INT ÁNG	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:5°	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
5°:10°	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
10°:15°	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
15°:20°	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
20°:25°	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
25°:30°	5	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
30°:35°	6	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
35°:40°	7	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:45°	8	-263,99	-293,53	-263,99	-10000,00	-151,85	-293,60	-10000,00	-10000,00	-292,25
45°:50°	9	-10000,00	-231,11	-194,55	-199,82	-229,52	-226,55	-199,81	-202,93	-231,14
50°:55°	10	-133,68	-125,94	-151,88	-10000,00	-114,73	-123,31	-151,15	-155,80	-133,76
55°:60°	11	-83,38	-83,60	-113,15	-81,17	-90,13	-83,14	-103,77	-80,27	-85,07
60°:65°	12	-22,99	-9,93	-35,65	-8,80	-17,25	-16,95	-22,59	-2,89	-19,99
65°:70°	13	-9,27	76,43	152,85	71,63	8,82	52,53	119,78	303,99	-2,71
70°:75°	14	339,89	294,93	260,07	678,09	250,71	236,46	170,18	261,85	345,78
75°:80°	15	465,70	511,42	597,81	847,36	579,95	813,91	753,69	629,70	608,48
80°:85°	16	953,91	940,85	918,96	999,43	898,99	1119,39	1027,61	993,80	953,19
85°:90°	17	1075,30	1005,10	1006,80	966,16	971,51	940,15	940,59	968,19	976,19
90°:95°	18	832,06	856,22	953,76	926,25	925,86	792,61	1069,83	696,80	866,66
95°:100°	19	609,62	671,98	600,85	702,97	656,58	653,11	557,93	556,91	988,31
100°:105°	20	291,74	176,87	328,13	236,39	219,66	352,83	401,77	274,57	707,45
105°:110°	21	35,19	91,34	100,44	80,77	-19,90	-15,35	30,78	22,11	337,87
110°:115°	22	-57,18	-56,41	-56,44	-58,33	-60,09	-53,57	-76,12	-60,76	-52,33
115°:120°	23	-106,69	-111,23	-109,30	-121,56	-102,40	-104,75	-141,67	-103,55	-112,98
120°:125°	24	-207,10	-200,72	-197,63	-199,09	-228,58	-218,58	-223,72	-210,79	-231,66
125°:130°	25	-263,91	-264,31	-263,95	-264,22	-241,56	-264,29	-264,16	-232,70	-264,11
130°:135°	26	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
135°:140°	27	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
140°:145°	28	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
145°:150°	29	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
150°:155°	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
155°:160°	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
160°:165°	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
165°:170°	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
170°:175°	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.4. Tabla Q del enfoque 3 con retardo de 25 ms.

		ACCIONES								
INT ÁNG	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:5°	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
5°:10°	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
10°:15°	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
15°:20°	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
20°:25°	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
25°:30°	5	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
30°:35°	6	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
35°:40°	7	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:45°	8	-352,12	-352,09	-356,79	-402,84	-402,69	-352,01	-402,60	-402,73	-163,28
45°:50°	9	-370,37	-403,06	-352,23	-386,76	-402,85	-364,69	-10000,00	-368,52	-354,47
50°:55°	10	-363,59	-335,70	-306,83	-328,48	-282,70	-311,31	-354,34	-299,65	-311,56
55°:60°	11	-259,25	-204,43	-229,91	-230,34	-306,47	-162,55	-334,78	-238,80	-318,73
60°:65°	12	-84,74	-80,54	-66,88	-71,05	-108,68	-32,21	-163,31	227,72	-214,91
65°:70°	13	-27,32	-40,24	12,14	121,48	-89,91	176,92	-53,33	767,38	-95,52
70°:75°	14	248,57	245,99	81,26	851,31	-26,42	538,59	29,33	543,77	12,73
75°:80°	15	708,95	628,70	509,45	1379,96	229,07	671,31	316,39	1013,91	-7,60
80°:85°	16	727,94	1011,92	805,56	1351,31	869,18	837,43	584,45	871,94	680,28
85°:90°	17	1066,90	1016,71	1106,89	943,86	1333,64	944,95	1079,86	690,89	875,10
90°:95°	18	449,55	422,88	904,64	589,40	653,40	528,17	1449,07	548,94	777,16
95°:100°	19	99,22	0,06	313,72	281,03	258,01	140,22	160,99	95,24	1439,28
100°:105°	20	42,72	-24,13	233,56	40,46	95,40	11,65	5,37	-23,75	828,05
105°:110°	21	-24,46	22,87	-24,58	-63,52	-23,52	-83,44	41,95	-109,75	540,54
110°:115°	22	-55,23	-94,65	136,39	-10000,00	-101,49	-103,07	-56,42	-10000,00	-106,95
115°:120°	23	-254,39	-255,67	-10000,00	-223,22	-10000,00	-10000,00	-10000,00	-237,16	-10000,00
120°:125°	24	-240,39	-243,24	-233,89	-10000,00	-179,70	-10000,00	-234,05	-10000,00	-198,12
125°:130°	25	-289,45	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-297,14	-10000,00	-237,29
130°:135°	26	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
135°:140°	27	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
140°:145°	28	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
145°:150°	29	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
150°:155°	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
155°:160°	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
160°:165°	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
165°:170°	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
170°:175°	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.5. Tabla Q del enfoque 3 con retardo de 40 ms.

		ACCIONES								
INT ÁNG	ESTADO	0	50	-50	100	-100	150	-150	200	-200
0°:5°	0	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
5°:10°	1	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
10°:15°	2	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
15°:20°	3	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
20°:25°	4	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
25°:30°	5	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
30°:35°	6	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
35°:40°	7	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
40°:45°	8	-468,52	-10000,00	-398,86	-406,40	-398,81	-321,52	-398,54	-398,50	-10000,00
45°:50°	9	-318,23	-283,05	-10000,00	-240,57	-314,37	-10000,00	-314,17	-308,05	-314,39
50°:55°	10	-239,15	-229,10	-10000,00	-249,83	-220,20	-203,63	-314,17	-288,22	-604,63
55°:60°	11	-155,00	-150,04	-149,13	-10000,00	-129,46	-138,43	-10000,00	-222,29	-220,26
60°:65°	12	-143,37	-90,04	-99,95	-116,07	-116,07	-162,74	-229,21	-10000,00	-223,31
65°:70°	13	-1,89	-16,58	-13,45	48,36	-18,00	165,28	31,37	4,40	-28,44
70°:75°	14	33,72	49,99	-139,33	95,70	-12,98	286,18	-109,69	-10000,00	-13,52
75°:80°	15	25,93	73,23	46,43	104,96	77,06	101,58	84,71	341,69	-55,43
80°:85°	16	118,12	105,00	68,03	93,09	157,77	137,47	94,95	312,95	45,32
85°:90°	17	45,96	205,00	275,75	187,84	228,84	9,69	416,37	40,68	137,50
90°:95°	18	115,00	86,60	137,62	24,64	30,86	108,06	136,33	54,40	239,63
95°:100°	19	-10000,00	2,05	22,65	-31,69	49,13	-23,37	82,77	22,38	311,32
100°:105°	20	22,16	-57,82	142,22	-110,24	-24,27	-127,91	67,07	-101,86	219,07
105°:110°	21	-33,62	-26,53	-34,44	-207,20	4,08	-116,18	42,06	-116,16	-6,63
110°:115°	22	-10000,00	-99,86	-121,04	-119,94	-10000,00	-231,75	-16,14	-134,84	-56,29
115°:120°	23	-10000,00	-178,40	-256,49	-253,97	-10000,00	-234,33	-286,06	-10000,00	-173,24
120°:125°	24	-10000,00	-416,66	-251,69	-423,87	-10000,00	-10000,00	-206,10	-10000,00	-10000,00
125°:130°	25	-315,42	-315,25	-408,86	-324,31	-321,50	-10000,00	-315,90	-315,27	-317,29
130°:135°	26	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
135°:140°	27	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-115,07	-10000,00	-10000,00	-10000,00
140°:145°	28	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
145°:150°	29	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
150°:155°	30	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
155°:160°	31	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
160°:165°	32	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
165°:170°	33	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00
170°:175°	34	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00	-10000,00

Tabla A5.6. Tabla Q del enfoque 3 con retardo de 55 ms.