Tecnológico de Costa Rica

Escuela de Ingeniería Mecatrónica


# Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares

Informe de Proyecto de Graduación para optar por el título de

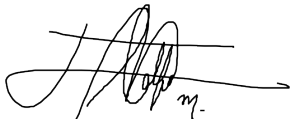
Ingeniera en Mecatrónica con el grado académico de Licenciatura

Francella María Campos Alfaro

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.



Francella María Campos Alfaro

Cartago, 01 de marzo de 2024

Céd: 504390901

Instituto Tecnológico de Costa Rica
Programa de Licenciatura en Ingeniería Mecatrónica
Proyecto Final de Graduación
Acta de Aprobación

La profesora asesora del presente trabajo final de graduación, indica que el documento presentado por la estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniera en Mecatrónica, con el grado académico de Licenciatura

Estudiante: Francella María Campos Alfaro

Proyecto: *Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares*

Ing. Ana María Murillo Morgan
Profesora Asesora

Cartago, 11 de marzo de 2024

**Instituto Tecnológico de Costa Rica**
**Programa de Licenciatura en Ingeniería Mecatrónica**
**Proyecto Final de Graduación**
**Acta de Evaluación del Jurado Evaluador**

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniera en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: Francella María Campos Alfaro

Proyecto: *Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares*

Miembros del jurado evaluador

Dr. -Ing. Juan Luis Crespo Mariño
Jurado

MSc. -Ing. Iván Araya Meneses
Jurado

Los miembros de este jurado dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 15 de marzo de 2024

# Resumen

Este proyecto aborda el desafío de incrementar la autonomía en robots modulares específicamente el EMERGE, enfrentando el problema de su adaptabilidad en dominios complejos y cambiantes. Dado que las capacidades de actuación de estos robots varían con su morfología, el objetivo principal fue desarrollar un sistema que permitiese al robot identificar y aprender posibilidades de actuación adecuadas a su forma sin necesidad de reprogramar constantemente sus códigos de control. Incorporando componentes de la arquitectura cognitiva e-MDB, como el proceso deliberativo de toma de decisiones y la integración de las motivaciones intrínsecas para genearar modelos de utilidad, se dotó al EMERGE de una autonomía de aprendizaje abierto a lo largo de la vida, permitiéndole operar en situaciones desconocidas y reutilizar conocimientos previos. Esta metodología se aplicó exitosamente en diversos dominios, tanto simulados como reales, permitiendo al robot interactuar eficazmente con su entorno y cumplir sus objetivos en diversas configuraciones morfológicas. Este enfoque promete avanzar significativamente en la robótica cognitiva, contribuyendo al trabajo del GII y sus respectivos proyectos de investigación.

**Palabras clave:** arquitecturas cognitivas, autonomía de aprendizaje abierto, robótica cognitiva, robots modulares.

# Abstract

This project addresses the challenge of increasing autonomy in modular robots, specifically the EMERGE robot, facing the problem of their adaptability in complex and changing domains. Since the performance capabilities of these robots vary with their morphology, the main objective was to develop a system that would allow the robot to identify and learn performance possibilities appropriate to its shape without the need to constantly reprogram its control codes. By incorporating components of the e-MDB cognitive architecture, such as the deliberative skills and the integration of intrinsic motivations to generate utility models, the EMERGE was endowed with lifelong open-ended learning autonomy, allowing it to operate in unfamiliar situations and reuse previous knowledge. This methodology was successfully applied in a variety of domains, both simulated and real, allowing the robot to interact effectively with its environment and accomplish its goals in various morphological configurations. This approach promises to significantly advance cognitive robotics, contributing to the work of the GII and its respective research projects.

**Keywords:** cognitive architectures, cognitive robotics, lifelong open-ended learning autonomy, modular robots

*a mi querida madre, por ser mi mayor fuente de motivación cuando estuve en Ferrol.*

# Agradecimientos

El desarrollo de este proyecto hubiese sido imposible sin el apoyo incondicional de mi familia. Mi mayor agradecimiento es a mi papá, por nunca dudar de mí y darme su apoyo incondicional con todo el amor del mundo en cada una de mis aventuras; gracias papá, por hacer hasta lo imposible para dejarme seguir mis sueños. Gracias a mi hermana por ser mi amiga del alma, la persona que siempre me hace ver las cosas con claridad, me recuerda quien soy y por qué estoy luchando, y es la que siempre sabe cómo alegrarme los días tristes. Gracias a mi mamá, a quien le dedico este trabajo, porque a pesar recibir un segundo diagnóstico de cáncer en septiembre mientras yo estaba en Ferrol, y armada con una fe inquebrantable, fue la que me motivó a seguir adelante con mi proyecto hasta terminarlo y no desistir a mitad de camino. La resiliencia y perseverancia que demuestra mi mamá es mi mayor fuente de motivación y definitivamente el pilar para lograr todo lo que he conseguido en la vida. Gracias mamita por darme todo el amor del mundo.

También tengo un agradecimiento profundo hacia los compañeros del GII, por darnos una acogida calurosa y hacernos sentir como en casa, a pesar de estar en otro país. Principalmente, agradezco a Martín y Alex por ser mis guías durante el desarrollo del proyecto. Me siento muy agradecida por todas su dedicación y paciencia para ayudarme a entender conceptos y códigos. Muchas gracias por confiar en mi trabajo.

Gracias a las amigas que hice en Ferrol, a Iria, Milica y Harriet, por estar ahí cuando más las necesitaba, por compartirme de sus culturas y tradiciones, y siempre motivarme a salir a buscar el sol. Estoy muy agradecida por todos los amigos que hice de ERASMUS por alegrarme hasta en los días más grises de Galicia (todos).

Agradezco también a mis amigos del TEC, a las personas que conocí desde el primer día de carrera y me han acompañado hasta el final, siempre apoyandome e inspirandome a ser mejor: Rashell, Melanie, César, Joshua y Jose Pablo. Estoy agradecida con la vida por haberlos puesto en mi camino.

A mi novio Gonzalo por darme su completa confianza y estar ahí para mí incondicionalmente. Por acompañarme a lo largo de la carrera y ser mi mejor compañero de equipo.

Francella María Campos Alfaro

Cartago, 01 de marzo de 2024

# Índice general

# Índice de figuras

# Índice de tablas

# Lista de abreviaciones

## Abreviaciones

| | |
|---|---|
| EMERGE | Easy Modular Embodied Robot Generator |
| FIFO | First in - first out |
| GII | Grupo Integrado de Ingeniería de la Universidade da Coruña |
| LOLA | Lifelong open-ended learning autonomy |
| RNA(s) | Red(es) Neuronal(es) Artificial(es) |
| ROS | Robot Operation System |
| SCM | Sistema de Captura de Movimiento |
| UDC | Universidade da Coruña |

# Capítulo 1

# Introducción

En la presente introducción, se explora el contexto del proyecto, que se desarrolló en el Grupo Integrado de Ingeniería de la Universidade da Coruña (UDC) en Ferrol, España, con un enfoque en robótica cognitiva. Se abordará el problema fundamental el cual corresponde a que un robot modular identifique sus posibilidades de actuación, cada vez que sus percepciones varían de acuerdo a su morfología y el dominio de operación del robot. Luego, se describirán los objetivos del proyecto y una breve descripción de la estructura del documento los cuales guiarán el resto del mismo.

## 1.1 Antecedentes del proyecto

El proyecto se llevó a cabo en el laboratorio del Grupo Integrado de Ingeniería (GII) de la Universidade da Coruña en España. El proyecto forma parte de una investigación en conjunto, más amplia, en el campo de la robótica cognitiva que involucra el estudio sobre robots modulares, específicamente el robot *EMERGE* (Easy Modular Embodied Robot Generator) desarrollado en el REAL Lab de la IT University of Copenhague, el cual fue generado en un proyecto conjunto con el GII

El Grupo Integrado de Ingeniería de la UDC (GII) lleva trabajando desde hace tiempo en el desarrollo de una arquitectura cognitiva denominada e-MDB (epistemic Multilevel Darwinist Brain) que busca aumentar el nivel de autonomía en los robots autónomos, independientemente del entorno en el en que se opere. Entre los trabajos más destacables de este equipo de trabajo, así como uno de los enfoques que se le dió a este proyecto, es la investigación en el campo de la autonomía de aprendizaje abierto a lo largo de la vida (lifelong open-ended learning autonomy).

Por otro lado, en este mismo laboratorio se completó otro proyecto final de graduación en Ingenería Mecatrónica, por parte del compañero Carlos Eduardo Jara Vasquez. El compañero Carlos se enfocó en diseñar una solución para que el robot pudiese identificar su propia morfología. El nombre de su proyecto es: *Desarrollo de un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE.*

## 1.2 Descripción del problema

El campo de la robótica autónoma enfrenta actualmente el desafío de diseñar robots con niveles de autonomía que les permitan ser útiles en dominios complejos y cambiantes [1]. El problema existe a la hora de utilizar robots modulares como en el caso del robot EMERGE, dado que sus posibilidades de actuación dependen de las percepciones del mismo robot, y dichas percepciones cambian en función de la morfología del mismo. Cada una de sus diferentes morfologías implican diferentes dominios de operación. Por lo que el principal problema se basa en diseñar actuaciones para cada uno de los posibles dominios del robot. Se tenía que buscar una solución para no estar cambiando constantemente los códigos de control de estos robots. La búsqueda de una solución efectiva para esta problemática se ha convertido en un objetivo crucial en el campo de la robótica cognitiva y el desarrollo del trabajo en el GII.

### 1.2.1 Justificación

El GII, en su trabajo para el desarrollo de la arquitectura cognitiva e-MDB persigue el objetivo de dotar a los robots autónomos de un mayor nivel de autonomía: *la autonomía de aprendizaje abierto a lo largo de la vida* (lifelong open-ended learning autonomy). Esta implica que deben ser capaces de realizar tareas útiles en dominios desconocidos al momento de diseñar el robot, Además, significa que los robots deben ser capaces de reutilizar e integrar el conocimiento en su proceso de aprendizaje.

Además, la aplicación de la autonomía de aprendizaje abierto a lo largo de la vida en el robot EMERGE es significativo para el trabajo en conjunto que desarrollan el GII y el REAL Lab de la IT University of Copenhague. Así como en el desarrollo de diferentes proyectos a nivel de la Unión Europa en la cual están involucrados miembros del GII, el proyecto Pillar Robots [2].

## 1.3 Síntesis del problema

El robot EMERGE no poseía la capacidad para identificar sus posibilidades de actuación en dominios cambiantes. Esta limitación impedía que el robot adquiriese habilidades de generalización similares a las humanas, lo cual es fundamental para lograr niveles de autonomía avanzados en robótica cognitiva. Resolver esta limitación es esencial para permitirle al robot aprender por sí mismo, reutilizar conocimientos y adaptarse a nuevas situaciones.

## 1.4 Objetivos del proyecto

El objetivo general que se desarrolló fue la integración de la arquitectura e-MDB para su utilización en el sistema robótico modular EMERGE, de manera que el robot descubra y aprenda, a través de ella, las posibilidades de actuación en función de su morfología para ser capaz de utilizarlas en condiciones y dominios adecuados.

Para alcanzar el objetivo general mencionado anteriormente, se formularon los siguientes objetivos específicos:

1. Obtener las características perceptuales y acciones generales relevantes en base a la sensorización del robot para la morfología dada.
2. Descubrir las posibilidades de actuación mediante una asociación de las percepciones del dominio con las posibilidades de actuación del robot.
3. Integrar la arquitectura e-MDB para su utilización con cada morfología robótica establecida.
4. Validar los resultados obtenidos en simulación mediante la implementación en el robot real.

## 1.5 Estructura del documento

Este documento está estructurado en seis capítulos, donde se expone el diseño y ejecución de un sistema para el aumento de la autonomía de del robot modular EMERGE, a través de componentes de la arquitectura cognitiva e-MDB en entornos de simulación y condiciones reales. El segundo capítulo establece el marco teórico, introduciendo los conceptos clave relacionados con el proyecto. El tercer capítulo detalla la metodología adaptada a este proyecto, incluyendo las etapas y procedimientos seguidos. En el cuarto capítulo se desarrolla la propuesta de diseño, se sigue la metodología descrita y se especifican los detalles de diseño e implementación de la solución. El quinto capítulo está dedicado al análisis de resultados, donde se evalúa el desempeño del sistema tanto en simulación como en el robot real. Finalmente, el sexto capítulo presenta las conclusiones del proyecto, junto con recomendaciones para investigaciones futuras y posibles mejoras.

# Capítulo 2

# Marco teórico

Es importante entender el concepto de la robótica cognitiva y cómo esta varía con respecto a la robótica tradicional que se da a conocer en la carrera de Ingeniería Mecatrónica.

En las siguientes sub-secciones se explicarán temas relacionados a la robótica cognitiva, entre los cuales se destacan: las arquitecturas cognitivas, la autonomía de aprendizaje abierto a lo largo de la vida (LOLA: *lifelong open-ended learning autonomy*), así como las motivaciones intrínsecas. Estos temas son escenciales en la robótica cognitiva puesto que son las bases para dotar de mayor autonomía a los robots en la actualidad.

## 2.1 Robots Autónomos

### 2.1.1 ¿Qué es un robot autónomo?

En la actualidad la industria cinematográfica nos ha mostrado múltiples ejemplos de lo que sería un robot, desde una máquina que puede ayudar a los seres humanos en sus tareas cotidianas, hasta una máquina que podría dominar el mundo si quisiera. Sin embargo, el concepto de *robot* se popularizó en 1921 durante la obra del guinista checo Karel Capek denominada *Rossum's Universal Robots* donde se identificada a un robot como una máquina lista con habilidades parecidas a las humanas, y los métodos utilizados para generarlos habrían sido por ejemplo movimientos basados en una frecuencia de reloj. Estas fueron las primeras aproximaciones a robots autónomos, sin embargo, en la actualidad, con el desarrollo de la ciencia y la tecnología un robot es mucho más que una máquina controlada por una frecuencia de reloj. Con el desarrollo de la capacidad de los sensores y actuadores, así como la capacidad computacional de hoy en día Matarić define a un robot como: *un sistema autónomo que existe en el mundo físico, puede detectar su entorno, y puede actuar en él para lograr algunas metas.*

Esta definición es importante en la robótica cognitiva, porque en un sistema autónomo el robot actúa con su propio conocimiento y toma sus propias decisiones, las cuales no están

controladas por un ser humano. Además es importante que el robot exista en el mundo físico, ya que, con el uso de los sensores este puede obtener información del entorno en el que se encuentra. Una vez que el robot obtiene información del entonrno físico o dominio en el que se encuentra, este debe tener la posibilidad de actuar sobre él, o realizar acciones. Para finalmente alcanzar un objetivo o realizar una tarea.

Por lo tanto, como la robótica es el estudio de los robots, esto significa que se encarga de estudiar la autonomía de los robots, sus sensores y sus acciones en el mundo físico[3].

### 2.1.2 Robótica cognitiva

Al pensar en robótica cognitiva se puede pensar también en robótica inteligente. Sin embargo, este concepto no es exactamente lo mismo que la robótica cognitiva. Puesto que, la robótica cognitiva a lo largo de los años se ha inspirado en la cognición humana, la psicología y la neurociencia para desarrollar robots autónomos, incluyendo, en algunos casos ejemplos de inteligencia y comportamientos de los animales.

Por ejemplo, una diferencia entre el comportamiento de una inteligencia artificial y las ciencias cognitivas es ChatGPT. Esta herramienta, que corresponde a una inteligencia artificial, se destaca por ser un modelo de lenguaje que se encarga de procesar y generar texto en función de las consultas y solicitudes del usuario. Sin embargo, esta no es una cualidad bio-inspirada, ni podría ser parte de las ciencias cognitivas, dado que ni los animales, ni los humanos poseen la capacidad de procesar tanto texto en tan poco tiempo.

En la literatura se menciona que la cognición es un proceso que incluye 6 atributos: autonomía, percepción, acción, anticipación, aprendizaje y adaptación. Porque en un sistema cognitivo se debe aprender mediante la observación o las percepciones, que es de donde se recibe la información de lo que sucede en el entorno. A partir de esta información se aprende y se adapta la forma en que suceden las cosas y de esta manera el sistema podría anticipar eventos futuros. Toda la información que obtiene el sistema viene de un proceso de obtener percepciones y aplicar acciones.

Por lo tanto, la robótica cognitiva es un campo interdisciplinario que busca combinar principios de la inteligencia artificial con conocimientos de ciencias cognitivas y biológicas para desarrollar robots que pueden interactuar y adaptarse de manera más inteligente y autónoma en entornos complejos [4].

### 2.1.3 Niveles de autonomía de los robots

Una vez definido un robot autónomo y la disciplina que los estudia, se identifica como el aumentar la autonomía en los robots tiene como objetivo acercarlos más a escenarios de la "vida real". Es darle herramientas al robot para que este pueda enfrentarse a situaciones menos estructuradas y con mayor variabilidad, sin la necesidad de la supervisión de un humano.

Por lo tanto el nivel de autonomía que posee un robot se ve influenciado por su capacidad de actuar ante dominios cambiantes. Esto depende principalmente de que si el dominio y la tarea son conocidos con anterioridad por el diseñador [1].

**Tabla 2.1:** Niveles de autonomía de los robots. Adaptada de [1]

| Nivel de autonomía | Tarea | Dominio |
|---|---|---|
| 1 | Conocida | Conocido con variaciones pequeñas |
| 2 | Conocida | Conocido con variaciones grandes |
| 3 | Desconocida | Desconocido |
| 4 (reutilización del conocimiento) | Desconocida | Desconocido |

En la tabla 2.1 se muestra cómo se clasifican los diferentes niveles de autonomía de los robots.

#### Nivel 1: Tareas y dominios conocidos con variaciones pequeñas

El diseñador en este caso debe conocer de antemano las tareas que realizará el robot y el dominio en el que trabajará. Para este caso la autonomía está relacionada a pequeños cambios que pueden suceder en el dominio, pero se parte de que el robot se adaptará gracias a las habilidades que ya posee. Un ejemplo de este tipo de robots se puede hallar en la robótica industrial donde se les asignan tareas como soldadura, manipulación o ensamblaje de piezas, las cuales son tareas que deben suceder en dominios muy precisos, por ende varían muy poco y en muchos casos se utiliza el control de tipo lazo-cerrado para las manejar las estrategias del robots [5].

#### Nivel 2: Tareas y dominios conocidos con variaciones grandes

En este nivel de autonomía los robots se enfrentan a dominios que varían mucho más que los anteriores, por lo tanto, los robots deben adaptar sus habilidades para enfrentarse al nuevo dominio. En estos casos, es usual utilizar herramientas de machine learning para que los robots aprendan las habilidades. Usualmente se utilizan técnicas de aprendizaje supervisado o aprendizaje por refuerzo [6]. De esta manera, el diseñador conocerá la tarea y el dominio, pero este no provee el algoritmo con el que el robot actuará, ya que este lo aprenderá por medio de la interacción con el dominio en el que se encuentra. Algunos ejemplos de este nivel de autonomía se ve presente en los robots bípedos o en los vehículos autónomos [7].

#### Nivel 3: Tareas y dominios desconocidos

En este caso el robot se enfrentará a dominios o tareas para los que no fue entrenado al momento de su diseño. En este caso el robot debe descubrir por medio de su propia exploración la tarea que puede o debe realizar, así como las habilidades que puede utilizar para realizar esa tarea. Además, el robot se debe adaptar al dominio cada vez que este cambie. Los robots que se encuentran en este nivel son totalmente autónomos, dado que se pueden adaptar a dominios menos estructurados, como son algunos escenarios de la vida cotidiana. Por lo tanto, a través de toda su vida el robot deberá seguir aprendiendo y adaptando sus habilidades para actuar en dominios desconocidos al momento del diseño. Este problema es considerado *el problema de aprendizaje abierto* que se enfoca en la capacidad de un robot para adaptarse a una secuencia ilimitada de tareas y dominios desconocidos a priori [8]. Esto implica retos como la detección de cambios en las tareas sin indicaciones explícitas, la ordenación de la adquisición de conocimientos y la resolución de tareas, y la identificación y transferencia de conocimientos relevantes para construir nuevas representaciones adaptadas. Por lo tanto, para esto es necesario el autonomía de aprendizaje abierto (OLA) [8].

#### Nivel 4: Reutilización del conocimiento

Finalmente, en el nivel más avanzado actualmente se toma en consideración que se facilita el proceso de aprendizaje y adaptación de un robot cuando este puede retener la información que aprendió en el pasado y reutilizar esos conocimientos a lo largo de sus vidas y eso es conocido como *aprendizaje a lo largo de la vida* y es esencial para que los robots autónomos logren operar de manera robusta sin supervisión [1].

## 2.2 Autonomía de aprendizaje abierto a lo largo de la vida

De la sección anterior se concluyó que para aumentar el nivel de autonomía en los robots es necesario que estos tengan la capacidad de manejar la variabilidad en el dominio y en las tareas a realizar, que no fueron consideradas al momento del diseño.

Es importante mantener claro que se trata de dos temas separados que en conjunto lograrán obtener los mayores niveles de autonomía en los robots. A continuación se describirán cada tema individualmente.

### 2.2.1 Autonomía de aprendizaje abierto

La autonomía de aprendizaje abierto en robots se refiere a la capacidad de estos para aprender una secuencia ilimitada de tareas desconocidas en dominios no explorados pre-

viamente [8]. Esto es un desafío importante, porque el robot a priori no conoce qué tareas puede realizar, dado que el diseñador no lo programó directamente para reconocerlas. Entonces el robot primeramente debe descubrir, por sí mismo, qué objetivos puede alcanzar y aprender a cómo hacerlo.

Esta exploración del robot en su entorno es importante para mantener la autonomía, puesto que el robot recolectará sus propios conocimientos de esta interacción y los podrá utilizar para alcanzar sus objetivos. Para lograr esto, es esencial dotar a los robots de mecanismos que le permitan descubrir y seleccionar sus objetivos. Estos mecanismos se denominan mecanismos motivacionales.

Los mecanismos motivacionales se han estudiado a partir de los comportamientos humanos y se han mostrado en literatura psicológica y educativa [9]. De esta literatura se destaca que el comportamiento humano se menaja a través de impulsos o motivaciones, un ejemplo es el hambre o la sed, ya que, el ser humano una vez que siente hambre se motiva a buscar una fuente de alimentación para sentir la satisfacción de saciar el hambre.

Estos enfoques se han aplicado para mejorar la exploración del espacio de estados, la detección de novedades, la recopilación de conocimientos, y el aprendizaje autónomo de habilidades. Sin embargo, estas propuestas han permanecido mayormente como experimentos de laboratorio y enfrentan el desafío de su aplicación en escenarios operativos reales. Los sistemas motivacionales se explicarán con mayor detalle más adelante.

### 2.2.2 Autonomía de aprendizaje a lo largo de la vida

Para complementar la autonomía del aprendizaje abierto, se utiliza la autonomía del aprendizaje a lo largo de la vida, esto implica que el robot podrá accesar a conocimientos que adquirió en el pasado. Esto facilita el proceso de adaptación del robot, puesto que no sería necesario aprender todo desde cero nuevamente, cada vez que se enfrenta a un escenario nuevo.

Para esto la literatura sugiere que una posible solución para alcanzar el aprendizaje de por vida en robots podría ser una estructura de memoria organizada similar a la cognición humana, usando arquitecturas cognitivas que integren y coordinen componentes de aprendizaje y motivacionales [1].

Por lo tanto, la integración de ambos temas lleva a la utilización del término *autonomía de aprendizaje abierto a lo largo de la vida, LOLA*.

### 2.2.3 Arquitecturas cognitivas para LOLA

Una arquitectura cognitiva es una estructura que manifiesta los requisitos de un sistema cognitivo, sus componentes y la dinámica de cómo estos están relacionados entre sí. Por lo tanto, las arquitecturas cognitivas incluyen componentes importantes de la cognición tales como la atención, memoria, resolución de problemas, toma de decisiones y el aprendizaje

[10].

En resumen la arquitectura cognitiva se encarta de implementar la cognición de manera artificial y estas se clasifican dependiendo del tipo de representación que pueden manipular [11]:

- **Simbólica**: Estas arquitecturas utilizan símbolos para representar información y reglas predefinidas para manipular estos símbolos. Son muy buenas en tareas de planificación y razonamiento lógico. Sin embargo, tienen dificultades para conectar estos símbolos con objetos y situaciones del mundo real, lo que limita su capacidad de adaptación. Algunos ejemplos son las versiones iniciales de ACT-R [12] y SOAR [13].
- **Emergente**: Este tipo de arquitecturas se desarrollan mediante la interacción directas con el entorno. El conocimiento adquirido es representado en estructuras como redes neuronales, permitiéndoles una mejor adaptación y superación de los problemas de conexión con el mundo real. Sin embargo, esta aproximación es más compleja y toma mucho más tiempo en aprender de esta manera. MDB [14] y GRAIL [15] son ejemplos de este tipo de arquitecturas, enfocándose en el aprendizaje autónomo y la adaptación.
- **Híbrida**: Combinan lo mejor de los tipos anteriores, utilizando representaciones simbólicas para procesos de alto nivel y enfoques conexionistas para tareas de percepción y adaptación a nivel bajo. Aunque esta combinación promete solucionar problemas de ambos enfoques, la transición de información entre los niveles simbólicos y sub-simbólicos sigue siendo un desafío. OpenCogPrime [16] y MLECOG [17] son ejemplos de intentos de arquitecturas tipo híbridas.

Es fundamental que una arquitectura para LOLA tenga las siguientes componentes: un sistema motivacional, un sistema de memoria basado en memoria contextual y un sistema de aprendizaje. El **sistema motivacional** es el encargado de motivar al robot a explorar y buscar nuevos objetivos en el espacio en el que se encuentra. Luego, el **sistema de memoria contextual** será el encargado de almacenar el conocimiento adquirido durante la exploración y decidir el momento adecuado para utilizar estos conocimientos para mantener el conocimiento a lo largo de la vida. Finalmente, un **sistema de aprendizaje** es importante, que además sea en línea, con el objetivo de adquirir conocimiento sobre cómo alcanzar los objetivos [1].

## 2.3 Arquitectura cognitiva e-MDB

Epistemic Multilevel Darwinist Brain, es el significado de las siglas e-MDB, la cual corresponde a la arquitectura cognitiva que fue desarrollada en el GII como parte de la evolución de la arquitectura cognitiva pre-existente MDB (Multilevel Darwinist Brain)

[14]. El diseño de esta nueva arquitectura fue parte de la tesis doctoral del Dr. Alejandro Romero investigador del GII, por lo que, las siguientes descripciones de este capítulo provienen de tal tesis [1].

Esta arquitectura cognitiva tiene como objetivo abarcar el problema de LOLA en plataformas de robots reales [1]. Por lo tanto, en esta sección se detallarán los principales componentes implementados por esta arquitectura para lograr LOLA y como estos se relacionan entre sí.

### 2.3.1 Componentes principales de la e-MDB

Primeramente, la e-MDB se desarrolló con la intención de añadir componentes que hacían falta en la MDB. Componentes fundamentales para lograr LOLA, que también son aplicables en otras arquitecturas de aprendizaje autónomo:

Un **Sistema Representacional** encargado de aprender la representación del espacio-estado del robot, así como re-escribir para que puedan ser utilizados por el resto de la arquitectura. Una **Memoria Episódica** Esta se encarga de almacenar conjuntos de percepciones y acciones (episodios) que se relacionan directamente con la tarea y el dominio del robot. Un **Sistema motivacional** que sería el núcleo del aprendizaje abierto y continuo en robots, ya que dirige sus motivaciones. Ayuda al robot a establecer y priorizar objetivos basándose en las necesidades e impulsos, así como los tareas accesibles en el dominio en el que opera el robot. Un **Sistema de aprendizaje** que proporciona mecanismos para el aprendizaje de conocimiento necesarios para operar en dominios variados. Incluye modelos de utilidad para alcanzar objetivos, modelos del mundo que representan el comportamiento de los dominios en los que se encuentra el robot, políticas como representaciones reactivas de habilidades y funciones de representación que facilitan la gestión de la información del sistema y los procesos de aprendizaje. Finalemnte un **Sistema de Memoria a Largo Plazo** que consiste en una central operacional de la arquitectura. El sistema de memoria se encarga de almacenar los objetivos, los modelos (de mundo y de utilidad), así como toda el conocimiento generado.

La manera en que se relacionan todos estos componentes en la arquitectura cognitiva se puede ver representada en la figura 2.1, y también se identifica la operación del robot, es decir, el ciclo que se cumple desde que se obtiene la percepción hasta que se realiza la actuación. Este ciclo se detalla a continuación:

1. **Percepción:** Este primer bloque se refiere a la información obtenida por medio de los sensores del robot, ya sean físicos o por medio de simulación. Ya que, el robot percibe su dominio por medio de sus sensores.

2. **Sistema de Representación:** Este bloque se encarga de re-describir las percepciones del robot, es decir, la información que proviene de los sensores. En algunas ocasiones esto puede significar una codificación de la información para lograr adaptar la información a las necesidades del procesamiento. Este bloque también es

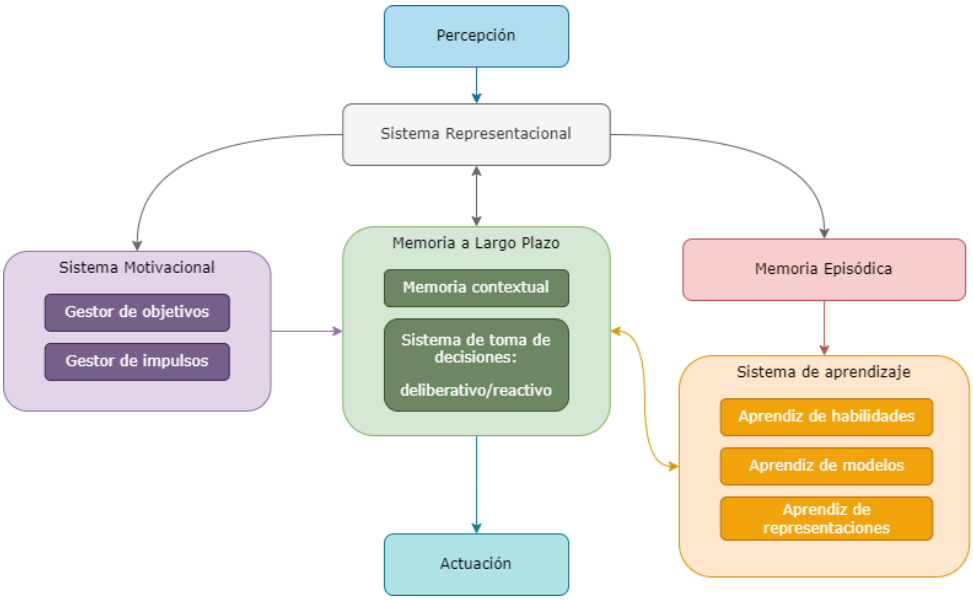



**Figura 2.1:** Componentes principales de la arquitectura e-MDB. Adaptado de [1]

de filtrado de información. Por lo tanto, la salida del bloque de representación corresponde al estado actual del robot.

3. **Memoria Episódica:** En la memoria episódica se almacenan episodios experimentados por el robot en durante su operación. Los episodios corresponden a los pares percepción-acción en el instante $t$ y el estado resultado de aplicar esta acción en el instante *t-1*. El contenido de este bloque de memoria se utiliza en el sistema de aprendizaje.

4. **Sistema de motivación:** Este bloque recibe como entrada el estado actual del robot y este es de suma importancia para la operación de la e-MDB, ya que involucra dos procesos principales.

   El gestor de impulsos corresponde al primer proceso importante, ya que se encarga de gestionar las necesidades y los impulsos del robot, y la forma en la que se satisfacen.

   El segundo proceso, principal corresponde al gestor de objetivos, el cual se encarga de identificar qué objetivos están presentes en el instante $t$ para así determinar qué necesidades o impulsos del robot se deben satisfacer.

5. **Sistema de aprendizaje:** Este bloque se encarga de generar el aprendizaje del sistema, este aprende modelos de mundo, modelos de utilidad, políticas y funciones de representación, por eso sus entradas corresponden a los episodios almacenados en la memoria episódica.

6. **Sistema de memoria a largo plazo:** En este bloque se obtienen como entradas los objetivos determinados en el sistema motivacional, los modelos aprendidos del sistema de aprendizaje, y además, del sistema representacional recibe la percepción del robot. Además, este bloque contiene la memoria contextual y el sistema de toma de decisiones que se encarga de decidie qué acción realizar en el instante dado.

7. **Actuación** En el proceso de actuación se envían las señales a los actuadores del robot para que la acción seleccionada por el sistema de toma de decisiciones sea aplicado.

### 2.3.2 Nodos de conocimiento de la arquitectura

Dentro de la memoria de la arquitectura es donde se almacenan los nodos de conocimiento, que corresponden a todas las estructuras de conocimiento generadas por el sistema de aprendizaje o proporcionadas por el diseñador. En todo caso, la representación de estos nodos de conocimiento en la arquitectura se encuentran en la figura 2.2 y serán descritas a continuación basandose en la dsecripción dada por A. Romero en su tesis doctoral [1]:

- **Percepciones** ($P_n$): Corresponde a la información obtenida de los sensores del robot.
- **Re-descripción de las percepciones** ($RP_p$): Son las funciones de normalización y representación de los valores de percepciones que vienen de los sensores.
- **Nodos-P** ($pN_q$): Corresponden a las clases perceptuales.
- **Objetivos** ($G_s$): Los objetivos se definen como el área en el espacio de estados que al ser alcanzada se satisface alguno de los impulsos o necesidades del robot. Además, los objetivos agregan una utilidad asociada a las acciones que realiza el robot para satisfacer las necesidades e impulsos.
- **Nodos-C** ($cN_u$): Corresponden a las clases contextuales. Estos nodos generan un enlace entre las percepciones y el objetivo a alcanzar, así como con los modelos de utilidad que permiten alcanzar este objetivo, además asocia el modelo de mundo (que describe el dominio en el que se encuentra el robot). Por lo tanto, generan los enlaces correspondientes del contexto en que se encuentra el robot.
- **Modelos de mundo** ($WM_v$): Los modelos de mundo son una función que representa el comportamiento del robot en el dominio, porque estas predicen los estados futuros del robot. Es decir, en función de la percepción acutal del robot $P_t$ y una acción candidata $a_i$, se predice la percepción futura $P_{t+1}$, como se muestra en la ecuación (2.1).

  Para cuestiones de este proyecto, los modelos de mundo vienen dados para el robot y el entorno. Estos son producto del trabajo realizados por el estudiante Carlos Jara en su proyecto final de graduación que se realizó en paralelo con este pryecto en el GII.

$$P_{t+1} = WM(P_t, a_i) \tag{2.1}$$

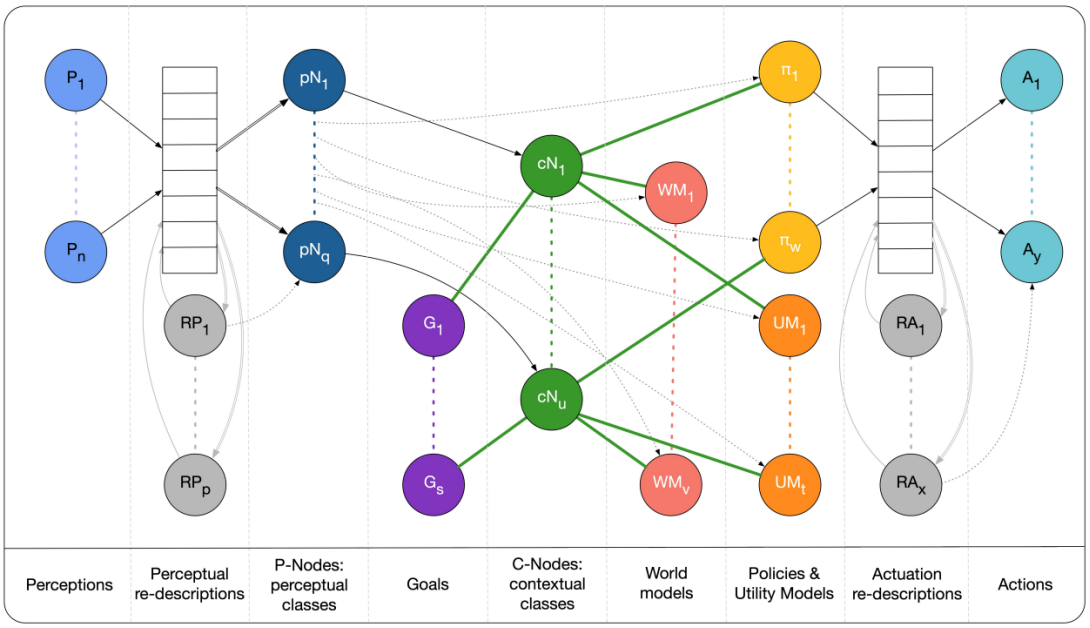



**Figura 2.2:** Nodos de conocimiento y su relación dentro de la estructura de conocimiento de la e-MDB [1]

- **Políticas** ($\pi_w$): Una política corresponde a una estructura de toma de decisiones reactiva para elegir la acción candidata óptima a ser aplicada dada la percepción $P_t$, como se muestra en la ecuación (2.2).

$$a_i = \pi_w(P_t) \tag{2.2}$$

- **Modelos de Utilidad** ($UM_t$): Un modelo de utilidad es una función que provee la utilidad esperada $\hat{u}$, de una percepción $P_t$ con respecto a el objetivo que se desea alcanzar. Por lo tanto, es una función que permite determinar la probabilidad de alcanzar el objetivo gracias a la utilidad que se le asigna a la percepción. Los modelos de utilidad son esenciales para tomar decisiones sobre qué acciones realizar, entonces más adelante se describirán con mayor detalle, para entender su funcionamiento y cómo se crean.

- **Re-descripción de la actuación** ($RA_x$): Son funciones que realizan re-escriben la actuación de la representación interna a la representación que utilizarán los actuadores del robot.

- **Actuación** ($A_y$): Representa la acción que aplica el robot y la acción utilizada dentro de la arquitectura cognitiva.

Ahora, una vez representada la arquitectura desde dos puntos de vista, los bloques que la componene y los nodos de conocimiento que posee, queda definir cómo se construyen estos nodos de manera autónoma por el robot para que no sea el diseñador el que incluya las funciones necesarias, si no que sea el mismo robot, que incentivado por medio de

las motivaciones logre generar suficiente conocimiento y almacenarlo para lograr tomar las decisiones necesarias para alcanzar los objetivos que se proponga. Por lo tanto, en el siguiente capítulo se comentará sobre las motivaciones intrínsecas y cómo estas son utilzadas para alcanzar LOLA.

## 2.4 Motivaciones intrínsecas

En esta sección se definirá el concepto de motivaciones intrínsecas y se explicaran los diferentes tipos de motivaciones intrínsecas epistémicas: novedad, effectance y reutilización del conocimiento. También, se explorarán los sistemas motivacionales utilizados en la exploración de los robots en la actualidad y su implementación.

Actualmente se define una motivación intrínseca como *un impulso que complementa los impulsos asociados a recompensas externas basadas en tareas para construir estados y espacios de acción adecuados, así como habilidades motrices* [8].

### 2.4.1 ¿Por qué las motivaciones intrínsecas?

La motivación es un elemento esencial en la inteligencia de un agente, porque es el que lo lleva a descubrir su entorno y crear nuevo conocimiento. En la literatura se menciona que la función de las motivaciones extrínsecas en la robótica consiste en completar los objetivos que le asigna el usuario, mientras que las motivaciones intrínsecas se relacionan más con la adquisición de conocimientos y habilidades (relacionadas a acciones) [4].

Es esta necesidad de crear conocimiento y explorar el entorno que es clave para generar LOLA, porque el robot no necesita que un agente externo le motive a realziar una tarea, si no que por él mismo pueda tener la necesidad de aprender, explorar y encontrar las tareas que desee realizar. Y así aumenta su autonomía.

A continuación se describirán los componentes motivacionales que incluye la arquitectura e-MDB que le permite iniciar sus operaciones previo a que la memoria a largo plazo tenga almacenada información alguna [1].

### 2.4.2 Motivación basada en Novedad

Esta motivación incita al robot a explorar zonas de su espacio de estados que no haya explorado aún y así poder mapear de manera eficiente su espacio de estados. Esta función de novedad implica que el robot cuando se encuentre en un estado conocido, quiera moverse a un estado que no haya visitado antes.

El cálculo de la novedad se realiza a partir de un buffer de trayetoria que contiene todos los estados por los que ha transcurrido el robot. Entonces la ecuación compara la distancia entre el estado candidato con respecto a los estados del buffer de trayectorias para

determinar cuan novedoso es el estado candidato. Por lo tanto, el valor de novedad para el $k$-ésimo estado candidato, denominado $s_{c,k}$, se calcula mediante la ecuación (2.3).

$$Nov = 1 - \frac{1}{M} \sum_{i=1}^{M} dist(s_{c,k}, s_i)^n, k \in [1, N] \tag{2.3}$$

En la ecuación (2.3) la variable $s_i$ corresponde al $i$-ésimo estado del buffer de trayectorias. $n$ corresponde al coeficiente que regula el balance entre la relevancia de la distancia entre estados cercanos. Finalmente, $N$ corresponde a la cantidad de estados candidatos que se toman en consideración para calcular su novedad.

La implementación de novedad es a través de un proceso de toma de decisiones deliberativo, el cual se describirá más adelante.

### 2.4.3 Motivación basada en Effectance

La motivación basada en Effectance también se considera una motivación de curiosidad. Por lo tanto, esta motivación está diseñada para activarse cada vez que suceda un efecto nuevo en el estado de percepciones. Existen dos tipo de efectos que se pueden producir:

1. **Efectos en el ambiente:** como la activación de un sensor, al tocar un botón o agarrar un objeto.

2. **Efectos dentro de la propia arquitectura:** cuando se activan elementos de los nodos de conocimiento de la arquitectura, y son interesantes de aprender y recordar.

### 2.4.4 Motivación basada en Competencia

La motivación de competencia es un complemento a la motivación de curiosidad, ya que le permite al robot de activar esos conocimientos nuevos aprendidos y recordados, para que el robot pueda mejorar su competencia en ellos y satisfacer así el impulso.

### 2.4.5 Motivación basada en Reutilización del Conocimiento

En la literatura, se menciona el uso de únicamente las tres motivaciones anteriores [18], sin embargo en la e-MDB se creó un investigó sobre el uso de una cuarta motivación que se relaciona con la reutilización del conocimiento [1].

La reutilización del conocimiento, como se ha mencionado en otras secciones es de suma importancia en LOLA, para facilitar el proceso de aprendizaje de nuevos conocimientos basado en experiencias del pasado.

El objetivo de esta motivación es generar nuevos nodos cognitivos (nodos-C) y relacionarlos con el dominio actual para facilitar la reutilización del conocimiento. Esto se logra cuando el conocimiento se reutiliza con éxito, lo cual satisface este impulso.

## 2.5 Modelos de Utilidad

Una vez reconocidas las motivaciones intrínsecas, se procede a explicar qué son los modelos de utilidad ($UM$) y cómo se implementan en la arquitectura e-MDB, además de su relación con las motivaciones y por qué estas son necesarias en el proceso de aprendizaje y toma de decisiones de los robots.

### 2.5.1 ¿Qué son los Modelos de Utilidad?

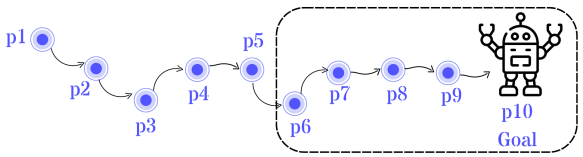
Con anterioridad se explicó brevemente la definición de un modelo de utilidad. Este consiste en una función que evalúa el estado en el que se encuentra el robot con respecto a la ubicación de un objetivo en el espacio de estados. De esta manera la utilidad esperada de cada estado representaría un camino a seguir para alcanzar un objetivo

### 2.5.2 Aprendizaje de Modelos de Utilidad

Las funciones de valor, que proporcionan una estimación precisa de la utilidad de cada punto del espacio de estados, son utilizadas por la e-MDB para desarrollar Modelos de Utilidad [19]. El conocimiento que el robot tiene del objetivo, o de los caminos que recorrió para llegar a él en el espacio de estados, debe servir de base para el entrenamiento de estos modelos. Para reunir esta información, el robot utiliza un método de exploración, en este caso impulsado por la motivación basada en novedad.

El robot almacena todos los puntos del espacio de estados que ha visitado durante la fase de exploración en orden cronológico desde un punto de partida aleatorio hasta que alcanza uno de sus objetivos. Esta trayectoria se conoce como una traza, y a cada paso de la traza se le asigna un valor de utilidad esperada. El proceso para asignar el valor de utilidad se realiza de la siguiente manera: se le asigna un valor de utilidad real de 1 al estado en el que se encuentra el robot una vez que alcanza el objetivo y este valor va disminuyendo hasta 0 a lo largo de la traza, en forma de utilidad esperada para cada punto. Un ejemplo de esto se muestra en la figura 2.3, donde se selecciona una parte de la traza almacenada en la memoria episódica y a estos se les asigna el valor de utilidad esperada para usarlos para el entrenamiento de la función de valor.

Producir una cantidad significativa de trazas con una variedad de puntos de partida para ser empleadas como base de datos para entrenar correctamente una red neuronal artificial (RNA), que servirá como Modelo de Utilidad, es esencial para reducir las ambigüedades o sesgos en el proceso de aprendizaje [1].




| Memoria Episódica | |
|---|---|
| **Percepción** | **Utilidad** |
| p10 | 1.0 |
| p9 | 0.8 |
| p8 | 0.6 |
| p7 | 0.4 |
| p6 | 0.2 |

**Figura 2.3:** Diagrama representativo de la asignación de utilidades en una traza. Adaptado de [1]

.

## 2.6 Redes Neuronales Artificiales

Con el fin de entender el funcionamiento del modelo de utilidad, se estudian las redes neuronales artificiales (RNAs). Las redes neuronales se desarrollaron con el objetivo de realizar tareas que realizan los seres humanos o los animales pero son muy difíciles de computar por medios clásicos. Entre estas tareas están: la clasificación, que predice la clase a la que pertenece un determinado vector; regresión, con el fin de predecir la función que relaciona una entrada con la salida deseada; optimización, para predecir valores óptimos para un vector de entrada; compleción de patrones, que predice partes faltantes de un vector de entrada.

Como las redes neuronales se inspiran en tareas que realizan los humanos, estas también se inspiran en el funcionamiento del cerebro humano. El cerebro humano se compone de un componente principal: la neurona. Las neuronas en el cerebro se organizan en redes neuronales de conexiones increíblemente complejas, puesto que deben realizar muchísimas tareas constantemente. Por lo tanto las redes neuronales artificiales intentan replicar el funcionamiento de las redes neuronales biológicas, y como estas intentan resolver una menos cantidad de tareas que el cerebro humano entonces son más simples (menos o más conexiones, dependiendo de la complejidad del problema) [20].

### 2.6.1 La Neurona Artificial

Una neurona artificial tiene como objetivo realizar un mapeo de las señales que tiene como entradas (porque puede tener múltiples señales de entradas, no solo una). A cada una de estas señales se le asigna un peso $v_i$ para reforzar o disminuir el valor de la entrada $z_i$. A la salida se utiliza una función de activación para computar la salida que depende del valor del sesgo $\theta$. En la figura 2.4 se muestra la relación de estos componentes.

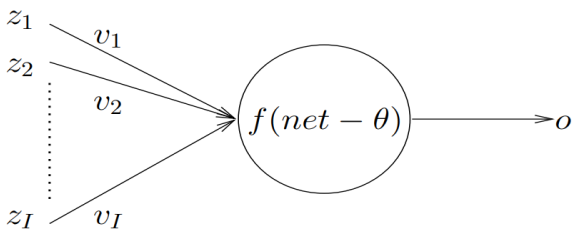



**Figura 2.4:** Neurona Artificial y sus componentes [20].

### La Función de Activación

La función de activación es importante, porque esta es la que recibe la señal de entrada, la evalúa y produce la salida. Existen múltiples funciones de activación utilizadas en redes neuronales artificiales: Función lineal, Función escalón, Función rampa, Función sigmoide, Tangente hiperbólica, Función gaussiana [20].

## 2.6.2 Conexión de las neuronas

La estructura más común de conexión de neuronas para crear una red neuronal se encuentra en la figura 2.5, acá la red posee una capa de neuronas de entrada, que usualmente corresponden a la cantidad de datos en el vector de entrada, en este caso si habían tres datos en el vector, se diseñaron tres neuronas de entradas. Luego, sigue la capa oculta, donde puede configurar cuantas neuronas habrán, o cuantas capas ocultas se pueden añadir. Las neuronas en las capas ocultas pueden estar densamente conectadas (redes neuronales densas), como se muestra en la figura 2.5, o pueden estar parcialmente conectadas. Finalmente se tiene la capa de salida donde se obtiene el resultado de la red.

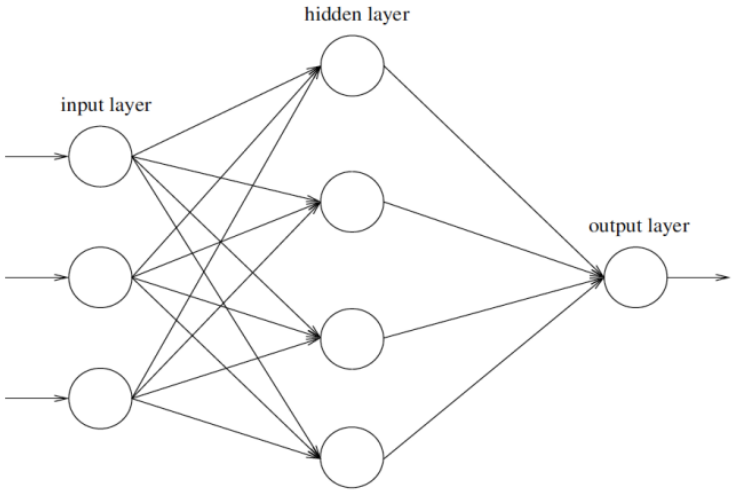



**Figura 2.5:** Red neruronal artificial básica. [20].

**Tipos de aprendizaje**

Las variables de los pesos $v_i$ y del sesgo $z_i$ se pueden calcular en el caso de problemas sencillos donde se tenga conocimientos previos sobre la función a analizar, pero en los casos donde no se tiene este conocimiento previo y únicamente se dispone de datos de entrada y de salida, surge la interrogante de cómo calcular estas dos variables $v_i$ y $z_i$. Estas variables se ajustan durante el proceso de aprendizaje de la red neuronal, que existen tres tipos [20]:

- **Aprendizaje supervisado:** en este caso se tiene una base de datos de entrenamiento, de datos de entrada y datos de salida, que corresponden a los resultados esperados de la red. Entonces durante el proceso de aprendizaje la red ajusta los pesos dependiendo del error que esta genere entre el valor real (de los datos de entrenamiento) y el valor predicho, hasta que este error se minimice. Para este proyecto se utilizó este tipo de aprendizaje, ya que, es esperaba que el robot generase una base de datos propia y con esto realizar el entrenamiento de la red.
- **Aprendizaje no supervisado:** este tipo de aprendizaje no obtiene datos de ninguna fuente, entonces la red debe aprender a identificar patrones.
- **Aprendizaje por refuerzo:** este tipo de aprendizaje premia a la red por generar buenos resultados, o de lo contrario, la castiga.

### 2.6.3 Hiperparámetros de la red neuronal

Para este proyecto se utilizó la librería de Tensorflow. La cual incluye una lista de hiperparámetros que se pueden configurar de manera que le permitan tener el mejor rendimiento a la red neuronal. Estos hiperparámetros se describen a continuación [21]:

1. Cantidad de capas ocultas que conforman la red y el tipo (Dense, Sequential, Flatten)
2. Número de neuronas de cada capa oculta.
3. Función de activación.
4. Optimizador, utilizado para la actualización de los pesos.
5. Epochs, indica la cantidad de iteraciones completas sobre los datos de entrenamiento.
6. Función de pérdida, las recomendadas para problemas de regresión corresponden a: Huber, MAE (suma absoluta del error), MSE (suma cuadrática del error).
7. Taza de aprendizaje.

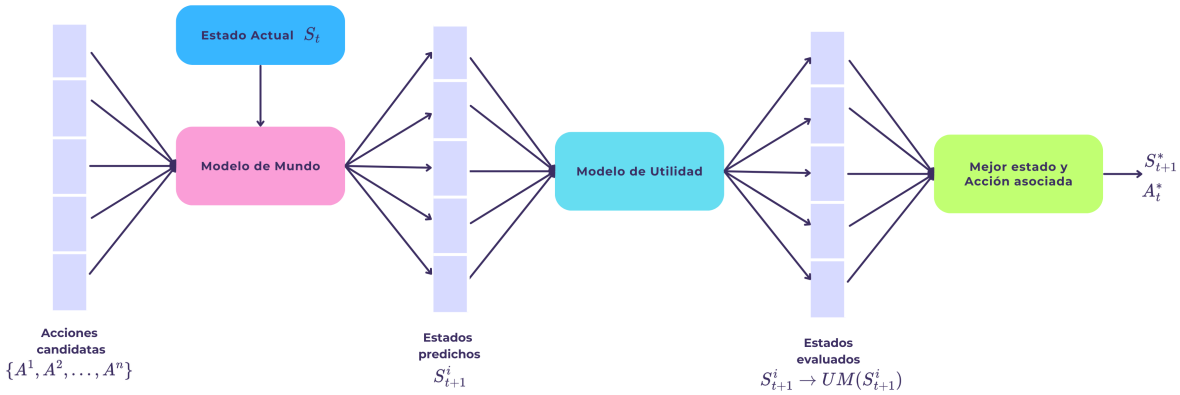



**Figura 2.6:** Modelo Deliberativo. Adaptado de [1]

## 2.7 Toma de decisiones

Una vez que se utilizan las motivaciones intrínsecas para incentivar la exploración en el espacio y la generación de conocimienos del robot a partir de los modelos de utilidad. Sin embargo, queda pendiente definir cómo se toman decisiones dentro de la arquitectura cognitiva e-MDB [1].

La toma de decisiones ocurre de dos maneras dentro de la arquitectura, a través de un proceso de decisiones reactivas denominadas políticas, o por medio de un proceso deliberativo de toma de decisiones. A continuación se describirán ambos procesos.

### 2.7.1 Proceso Deliberativo de Toma de Decisiones (Deliberative Skills)

El proceso deliberativo de toma de decisiones, propuesto en la e-MDB, se aprovecha del uso de los modelos de utilidad ($UM$), en conjunto con los modelos de mundo ($WM_m$) para generar un proceso de deliberación. En donde a partir del estado actual del robot $S_i$ y una serie de acciones candidatas, el $WM_m$ predice el estado futuro que podría alcanzar el robot al aplicar cada acción candidata. Partiendo de esta lista de estados predichos por el $WM_m$, el $UM$ evalúa cada estado en base a qué tan útil sería para alcanzar el objetivo deseado. De esta manera el proceso deliberativo elige la acción asociada al estado predicho más útil, para ser aplicada por el robot, generando decisiones "pensadas" y no reactivas. Una ilustración de este proceso se encuentra en la figura 2.6.

Al utilizar el proceso deliberativo de toma de decisiones es necesario tomar en cuenta que es un proceso más lento que la toma de decisiones reactivas, porque debe pasar por el proceso de predecir y el proceso de evaluar de cada uno de los modelos. Además, se deben tener modelos representativos del dominio, si el robot no ha tenido la oportunidad de generar conocimientos suficientes y estos modelos son mal entrenados entonces el sistema de toma de decisiones deliberativo no será veráz. Por esto es importante que el robot explore el dominio y aduiera suficiente información para mejorar los modelos.

Por otro lado, una de las ventajas que tiene este proceso es que se adapta a situaciones en donde el robot aún no tiene conocimientos, como en el caso de los robots nuevos que aún no han definido ni qué objetivos pueden alcanzar. Ya que, para generar el proceso deliberativo se deben explorar y entrenar los modelos mencionados. Otra ventaja de este proceso es que el robot puede tomar decisiones de manera deliberada durante el proceso de adquisición de datos para lograr explorar el dominio de una forma más eficiente y generar una base de datos suficiente para entrenar el modelo de utilidad.

### 2.7.2 Decisiones reactivas (políticas)

Como se explicó anteriormente tomar decisiones de manera deliberativa toma más tiempo en implementar una acción, y a veces se presentan casos en los que es necesario actuar más rápido o de manera más eficiente y para esto se crearon las políticas. Por eso se crearon las políticas, para que estas generen una acción con simplemente utilizar de entrada el estado actual del robot $S_i$ y el objetivo a alcanzar.

Ahora, para mantener los requisitos básicos de LOLA, el aprendzaje de las políticas se realiza en base a los conocimientos que ya posee el robot, estas políticas no las puede definir el diseñador, por ejemplo: no pueden ser ecuaciones explícitas. En la e-MDB se propone utilizar los modelos aprendidos para generar el proceso delibertivo para re-estructurar el conocimiento de manera que se puedan aprender políticas reactivas. Además, se propone utilizarlas en los nodos-C que es donde se realiza el vínculo entre el dominio y los modelos aprendidos para alcanzar los objetivos.

El proceso de aprender políticas parte de tener una población de políticas candidatas $\Pi$, estas políticas corresponden a RNAs. Inicialmente, estas corresponden a RNAs aleatorias que se van entrenando, por medio del proceso de toma de decisiones deliberativo. Durante este proceso cada política candidata $\pi_i$ debe generar una acción $a_{ik}$ a partir de una percepción $p_k$ del grupo de percepciones $P*$: $a_{ik} = \pi_i(p_k)$. Luego, esta acción generada pasa por el modelo deliberativo de la figura 2.6 hasta que se obtiene la evaluación de esta acción del modelo de utilidad y este valor de utilidad se le es asignado a la política en prueba $\pi_i$. Una vez que se prueban todas las percepciones de $P*$, se selecciona la política con el promedio de utilidades más alta para asociarla al nodo-C.

## 2.8 Robots Modulares

### 2.8.1 Introducción a los Robots Modulares

Los robots modulares son sistemas compuestos por unidades intercambiables que permiten una variedad de configuraciones y funciones. Estos sistemas son capaces de adaptarse a diferentes tareas y entornos, lo que los hace ideales para aplicaciones en entornos dinámicos y no estructurados.


**Figura 2.7:** Módulos de EMERGE: a) Módulo de un grado de libertad y cuatro caras de conexión, b) base, c) enlace de conexión [23].

#### Importancia en la robótica actual

La modularidad en la robótica ofrece soluciones flexibles y escalables, esenciales para enfrentar los retos actuales en industrias y áreas de investigación. La capacidad de estos robots para reconfigurarse en función de las necesidades específicas reduce significativamente los costos y tiempos de desarrollo, además de facilitar el mantenimiento y la actualización de los sistemas [22].

### 2.8.2 Robot Modular EMERGE

EMERGE (Easy Modular Embodied Robot Generator) es un sistema robótico modular diseñado para la rápida adaptación y evolución de robots en entornos variados. Este enfoque permite crear configuraciones personalizadas que pueden optimizarse para tareas específicas, potenciando la investigación y el desarrollo en robótica modular.

El propósito de EMERGE es superar las limitaciones de los sistemas robóticos tradicionales mediante la flexibilidad y adaptabilidad que ofrece la modularidad. Este sistema busca simplificar el proceso de diseño de robots, haciéndolo más accesible y eficiente para una variedad de aplicaciones prácticas y de investigación [22].

#### Componentes y características del diseño

EMERGE se caracteriza por su conjunto de módulos intercambiables que facilita la configuración de robots modulares. Esta herramienta promueve la innovación en el diseño de robots, permitiendo la exploración de nuevas formas y funcionalidades de manera eficiente.

El módulo EMERGE actuado tiene cuatro caras de conexión y un grado de libertad rotacional controlado por un servomotor. Los módulos de enlace y base no tienen grados de libertad y tienen dos caras de conexión. Conectando módulos a través de sus caras de conexión, se pueden construir varios robots rápidamente. En la figura 2.7 se muestran los tres tipos de módulos de EMERGE. Para este proyecto solo estaban disponibles el módulo de un grado de libertad y la base [23].

Cada módulo contiene lo siguiente:

- Los conectores magnéticos mantienen unidos mecánicamente los módulos, usando partes impresas en 3D con imanes de neodimio y PCBs para contactos eléctricos.
- Conectores masculinos y femeninos con polaridades opuestas, que soportan hasta 58N y 1.2 Nm de torque antes de desconectarse por completo.
- Los conectores masculinos utilizan pines de resorte para transmitir energía y comunicaciones entre módulos, asegurando una conexión efectiva incluso ante pequeñas desalineaciones o vibraciones.

En la figura 2.8, se muetra una vista explosionada de los diferentes componentes de EMERGE y como estos están conectados entre sí.

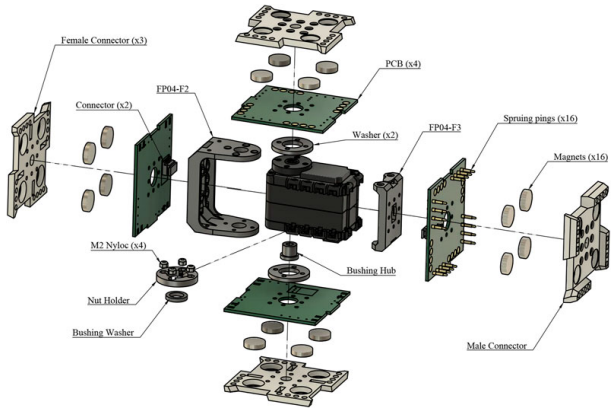



**Figura 2.8:** Vista explosionada de un módulo EMERGE [23].

## 2.9 Discusión Final

En resumen, la exploración de la autonomía en los robots, a través de LOLA, a través de la implementación de la arquitectura e-MDB, promueve la adaptabilidad y la eficiencia en entornos dinámicos y en la resolución de problemas complejos. Al incorporar estos elementos. Este enfoque integral hacia la robótica autónoma nos lleva a considerar el siguiente paso en la evolución de los sistemas robóticos: el desarrollo de **robots modulares**. Estos sistemas prometen ofrecer una flexibilidad en el diseño y la funcionalidad, permitiendo la creación de máquinas que pueden cambiar de forma para satisfacer una amplia gama de tareas y desafíos. De esta forma al integrar LOLA a los robots modulares se mejoraría su capacidad para aprender y adaptarse de manera autónoma, además de generar una interacción más intuitiva con el entorno y sus diferentes morfologías posibles.

# Capítulo 3

# Metodología

El enfoque sistemático y estructurado de K. Ulrich y S. Eppinger del libro *Diseño y Desarrollo de Productos* [24] es muy útil para la creación de soluciones innovadoras en el campo de la robótica cognitiva. Este proceso destaca por integrar la identificación de las necesidades del cliente, la creación y selección de conceptos basados en criterios técnicos y el uso de decisiones deliberativas en el diseño.

## 3.1 Descripción general

A continuación se realizará una descripción general de esta metodología de diseño y como esta se adaptó para la generación de una solución para este proyecto, ya que, este fue un proyecto de tipo investigativo y no para un producto comercial. En la Fig. 3.1 se muestran las etapas propuestas en el libro para llevar la fase de desarrollo del concepto. Es importante mencionar que las flechas de líneas interrumpidas muestran que el avance en el desarrollo de la metodología no siempre es secuencial y existen momentos en los que es necesario detenerse, regresar a una etapa anterior y repetir la actividad antes de continuar [24].

- **Identificación de las necesidades del cliente:** En esta primera etapa del proyecto se deben documentar cuidadosamente las necesidades del cliente, para ello se deben realizar entrevistas para asegurarse de obtener toda la información pertinente. Una vez realizado esto, las necesidades se organizan en una lista jerárquica según la importancia. En el caso de esta investigación, esta etapa de mantiene.

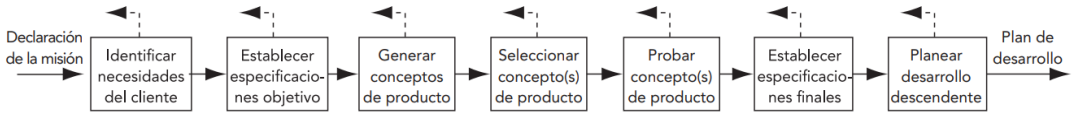



**Figura 3.1:** Diversas etapas iniciales que comprenden la fase de desarrollo del concepto, según Ulrich y Eppinger [24]

- **Establecimiento de las especificaciones:** En esta etapa se realiza una traducción técnica de lo que expresó el cliente en las necesidades, en términos técnicos. De esta sección se obtienen especificaciones objetivo, con una métrica asociada y los respectivos valores marginales e ideales. En esta etapa de especificaciones. Como parte de la investigación esta etapa también se mantiene, porque es de suma importancia entender correctamente los términos de la e-MDB, cómo se correlacionan y cómo se implementan.

- **Generación y selección de conceptos:** La generación de conceptos es una etapa en la que se exploran todos las ideas y posibilidades, por medio de búsquedas externas y búsquedas internas, así como la solución creativa de problemas. Esta etapa es muy importante en este proyecto de investigación porque se dividió el problema principal en subproblemas críticos y se realizaron pruebas experimentales para lograr enfrentarlos de manera más eficiente y enfocada.

  Además, la solución de los subproblemas se fue realizando conforme se avanzaba en las etapas del proyecto. De esta manera se generaban conceptos para cada subproblema de manera enfocada, se seleccionaba de manera deliberativa, se aplicaba, se probaba y se seguía con la siguiente etapa.

- **Prueba del concepto:** La etapa de prueba de concepto fue de suma importancia, puesto que, consistió en validar el concepto final seleccionado para determinar si cumple con las especificaciones establecidas. En la investigación esta etapa es de suma importancia, para probar el funcionamiento correcto de las diferentes ideas propuestas, así como llevar a implementar las soluciones en el robot real.

- **Establecer especificaciones finales:** Antes de finalizar el equipo debe identificar las limitaciones del concepto del producto, como las limitaciones del modelado técnico y de su desempeño, para así definir nuevamente las especificaciones finales de acuerdo al rendimiento real del producto.

- **Planeación del proyecto:** En esta última etapa el equipo crea un programa de desarrollo. Sin embargo, para este proyecto que no es para un producto comercial, esta etapa no se lleva a cabo como una etapa de desarrollo, si no más como una etapa de seguimiento de la investigación. Donde se definen nuevos pasos y trabajos futuros a ser realizados basándonse en el trabajo realizado en este proyecto.

Es importante mencionar que a lo largo de cada etapa es importante construir y probar modelos y prototipos.

## 3.2 Adaptación de la metodología

La metodología se modificó, principalmente porque el resultado esperado de este proyecto era la implementación de componentes de la arquitectura e-MDB en el robot modular

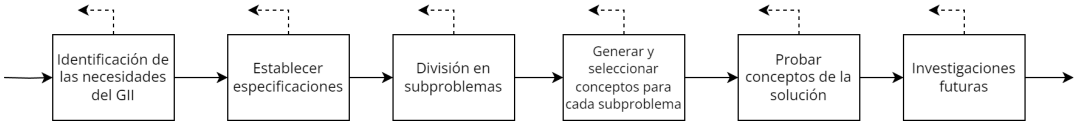



**Figura 3.2:** Adaptación de la metodología de diseño planteada para este proyecto investigativo. Elaboración propia.

EMERGE. Por lo tanto, la solución ya tenía forma, sin embargo se debían generar adaptaciones a lo largo del proceso de diseño para que se pudiese adaptar al robot EMERGE.

Por lo tanto, este proyecto se decidió trabajar de manera que por cada subproblema se generasen soluciones y se utilizara un proceso de generación y selección de conceptos en el momento en que estos problemas ocurrían. Entonces, se decidió seguir el enfoque de solución, utilizar las principales fuentes de investigación propuestas y comenzar a desarrollar la solución.

Por lo tanto, la adaptación de estas etapas para este proyecto se identifican en la Fig. 3.2. En donde se enfatiza la etapa de experimentación durante la generación de conceptos, para seleccionar el concepto ideal para cada subproblema. Así como la etapa final de planeación se definió más bien como una etapa de seguimiento a la investigación realizada, ya que este robot modular no tiene el objetivo de ser implementado en el laboratorio ni nada por el estilo.

## 3.3 Identificación de las necesidades

Durante la primera etapa de esta metodología se identificaron las necesidades de parte del GII. Se obtuvo una primera versión de las necesidades por medio escrito, de donde se desarrollaron preguntas para ser planteadas en una entrevista a los profesores del GII: Dr. Richard José Duro Fernández, director del proyecto; Dr. Martín Naya Varela, encargado de los módulos robóticos EMERGE; y Dr. Alejandro Romero Montero, desarrollador de la arquitectura cognitiva e-MDB. La presencia de los tres investigadores generó una discusión donde se pudo identificar las necesidades de cada una de las diferentes ramas del proyecto, así como las zonas confusas de la integración de ambas temáticas en el ámbito del proyecto en general. Las necesidades discutidas durante esta sesión se muestran en la tabla 3.1.

## 3.4 Especificaciones y métricas

En esta etapa de establecimiento de las especificaciones y las métricas, se intentó traducir en palabras técnicas las necesidades que se obtuvieron de la sección anterior.

Para determinar las especificaciones, se inició con la creación de una relación de métricas alineadas específicamente con cada necesidad, para que cada una tuviese al menos una

**Tabla 3.1:** Necesidades del cliente.

| ID | Enunciado de la necesidad |
|---|---|
| 1 | La solución se simula en CoppeliaSim |
| 2 | La solución permite al robot alcanzar objetivos en el entorno |
| 3 | La solución utiliza redes neuronales en Python que representan bien el conocimiento |
| 4 | La solución es integabre a distintas morfologías |
| 5 | Los componentes de la e-MDB son integrables en la solución |
| 6 | La solución elige acciones para realizar en dominios nuevos |
| 7 | La solución almacena los conocimientos adquiridos por el robot |
| 8 | La solución es integrable al robot real |
| 9 | La solución cuenta con las documentaciones necesarias para su uso |

métrica relacionada. Posteriormente, se consultaron las preferencias de los investigadores para asignar valores a dichas métricas. Se establecieron así los valores ideales y los valores marginales o aceptables. Los valores específicos para cada métrica se presentan en la tabla 3.2.

**Tabla 3.2:** Especificaciones y métricas.

| ID | Métrica | Unidades | Valor ideal | Valor marginal |
|---|---|---|---|---|
| 1 | Utiliza CoppeliaSim | Binaria | Cumple | Cumple |
| 2 | Porcentaje de objetivos alcanzados por el robot en simulación | Porcentaje | 90% | 70% |
| 3 | Pérdida en la red neuronal | Porcentaje | 2.5% | 5% |
| 4 | Cantidad de morfologías | Morfologías | 3 | 2 |
| 5 | Utilización de componentes de la e-MDB | Binaria | Cumple | Cumple |
| 6 | Presencia de un proceso de toma de decisiones | Binaria | Cumple | Cumple |
| 7 | El robot reutiliza sus conocimientos | Binaria | Cumple | Cumple |
| 8 | Porcentaje de objetivos alcanzados por el robot real | Porcentaje | 70% | 50% |
| 9 | Documentación del diseño | Binaria | Cumple | Cumple |

La lista de especificaciones tuvo un nivel de complejidad para determinar, puesto que el cliente no especificó necesidades relacionadas al precio del producto solución, ni a la precisión que debería tener el sistema, ni el tiempo en que debería alcanzar los objetivos el robot. Ya que, esto no representaba problemas para el GII en este proyecto de investigación. Por esta razón, muchas de las necesidades del cliente son de tipo binarias, puesto que las necesidades del GII recaían en que se incluyeran componentes de la e-MDB dentro de la solución. Entre estas están las métricas 1, 5, 6, 7, y la 9.

La métrica 1, simplemente corresponde a la necesidad que representaba utilizar el simulador CoppeliaSim donde se habían realizado previamente los diseños de los módulos del robot EMERGE. Luego, la utilización de componentes de la e-MDB en la métrica 5, sucede porque esta fue la arquitectura cognitiva desarrollada en el GII y representaba una necesidad para ellos comenzar a implementar diferentes componentes de esta arquitectura en específico (nodos de conocimiento y sus conexiones en la sección 2.3.2), para así demostrar el funcionamiento y la implementación de esta en diferentes tipos de robots. De igual manera las métricas 6 y 7 tienen que ver con la implementación de los nodos de conoci-

miento de la e-MDB, específicamente sobre almacenamiento del conocimiento adquirido por el robot (y su reutilización, en la métrica 7), y un proceso de toma de decisiones para que el robot pueda elegir las acciones que desea realizar y que no se les fueran dadas por la diseñadora. Finalmente, la métrica 9, corresponde a la documentación del diseño, que viene de la necesidad de poder reutilizar estos códigos para futuras investigaciones, así como para añadirlas al repertorio de códigos del GII. Por otro lado la métrica 9, de la misma necesidad 9, incluye que el GII pidió la recopilación de manuales de uso de las diferentes herramientas utilizadas en el laboratorio para la implementación de los robots, para que estas mismas herramientas pudiesen ser utilizadas también por el personal del GII, como parte de las diferentes investigaciones.

Ahora las métricas 2 y 8 tienen que ver con que el robot alcance sus objetivos. Estas métricas se basaron en las necesidades 2 y 8 respectivamente. Porque la idea del cliente era que una vez que el robot lograse alcanzar objetivos dentro del entorno de simulación entonces que este se pudiera implementar en el robot real, de manera que este también pudiese alcanzar objetivos en la realidad. Entonces, la métrica 2 que se refiere a alcanzar objetivos dentro de un entorno simulado, según el criterio de los investigadores del GII, el valor marginal y el valor ideal se cumplirían mejorando el entrenamiento de las redes neuronales, o añadiendo más componentes de la e-MDB, puesto que esta situación se mejoró de esta forma durante los experimentos con el robot Baxter durante las pruebas de la e-MDB [1]. Por otro lado, en la métrica 9, de igual manera se obtuvo el valor por criterio de los investigadores porque no existían pruebas estandarizadas para realizar sobre los módulos del EMERGE.

Luego, la métrica 3, está relacionada a la métrica 2, porque redes neuronales entrenadas correctamente demuestran una mejoría en los resultados del robot, esto implica un error bajo en las predicciones, donde es aceptable un máximo de 5% de error. Además, los investigadores recalcaron que una de las necesidades de ellos era utilizar las redes neuronales, ya que, se habían utilizado antes, y mostraron buenos resultados. Sin embargo, esto se analizará más adelante para determinar si sería útil usarlas en este caso.

Para la métrica 4, al ser pruebas en un robot modular (ver la sección 2.8), pues era y es importante que se realizaran pruebas en más de una morfología. Sin embargo, como es un proceso de carácter investigativo, y la cantidad de morfologías también dependía del avance del proyecto del compañero Carlos Jara se decidió implementar al menos 2 morfologías e idealmente 3. Además, esta métrica se refiere a que la misma solución debe ser utilizada para las diferentes morfologías, ya que la necesidad 4 especifica que la solución es integrable a distintas morfologías.

## 3.5 Generación de conceptos

Para la generación de conceptos se intentó seguir el orden sugerido en el libro de K. Ulrich y S. Eppinger, sin embargo, algunas de las búsquedas internas incluyeron experimentaciones

y a continuación se detallará por qué y cómo.

### 3.5.1 Aclaración del problema

Para comenzar se utilizó la premisa del objetivo general del proyecto que dice que menciona que el robot debe descubrir y aprender las posibilidades de actuación en función de la morfología y según la necesidad 7 de la tabla 3.1, la solución diseñada debe elegir acciones para que el robot realice en nuevos dominos. Entonces, esto implica que se debe elegir un proceso para toma de decisiones del robot. Según la teoría mencionada en le sección 2.7, para robots que están iniciando su proceso de aprendizaje y no tienen conocimientos aún de su entorno o sus posibilidades de actuación, el método utilizado para comenzar a recopilar información corresponde al proceso deliberativo de toma de decisiones. A parte, entre las necesidades del cliente no está que el sistema de decisiones sea rápido, por ende se descarta el uso de las políticas. Por lo tanto se elige el proceso deliberativo de toma de decisiones (Deliberative Skills).

### 3.5.2 Descomposición en subproblemas y enfoque de solución

Ahora una vez que se identifica que el robot debe tomar decisiones y el método a utilizar, se propone la descomposición en subproblemas. Para esto, se planteó un diagrama general mostrada en la Fig. 3.3, que corresponde con el diagrama de Deliberative Skills de la Fig. 2.6. Es importante mencionar que para este bloque se establecieron las mismas entradas que posee el modelo deliberativo. Además, se planteó un diagrama de enfoque de solución, que se divide en subproblemas a solucionar y se muestran en la Fig. 3.4

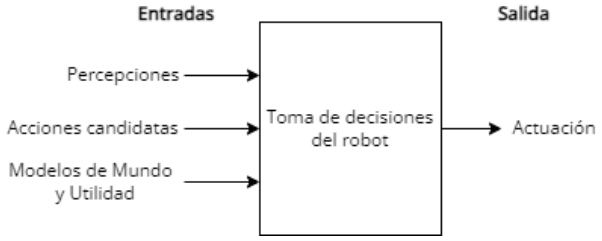



**Figura 3.3:** Diagrama general de la implementación del sistema de toma de decisiones del robot. Elaboración propia.

Este enfoque de solución se puede relacionar con el procedimiento a seguir para lograr alcanzar los objetivos específicos del proyecto. En la primera etapa, se planteaba determinar problemas relacionados a la morfología del robot y las percepciones que este podría tener a través de sensores así como las posibles acciones que este podría realizar, que se relacionan con el primer objetivo específico de este proyecto. Le definición de las posibles actuaciones y sensores que utilizaría el robot se determinó en conjunto con el compañero Carlos Jara dado que las percepciones que predice el modelo de mundo que él entrenó debían tener las mismas unidades y forma que las percepciones del estado actual

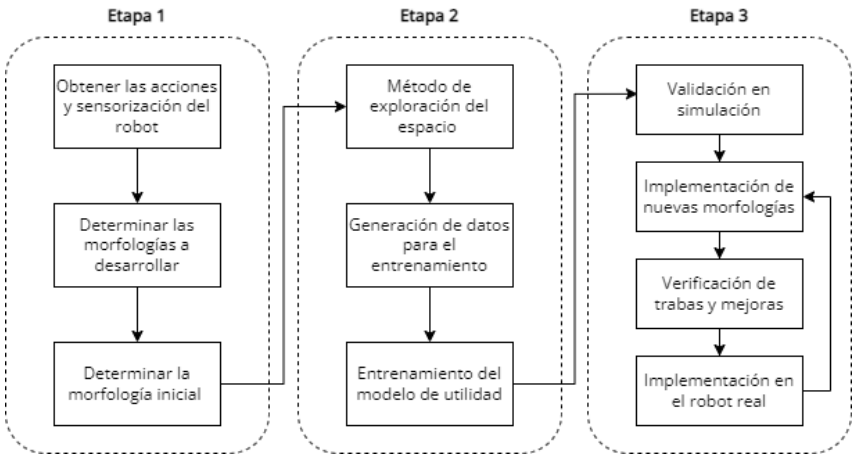



**Figura 3.4:** Enfoque de solución. Elaboración propia.

del robot. Además, se determinó en conjunto las morfologías a desarrollar en el proceso de aprendizaje del robot, así como la morfología inicial para hacer las pruebas con los robots.

Para la segunda etapa se plantearon subproblemas a solucionar para la generación del modelo de utilidad que corresponde al nodo de conocimiento que indicaría cuán útil sería cada acción candidata evaluada. Para esto era necesario entrenar una red neuronal (indicada en las especificaciones) y por ende conseguir datos para entrenarla. Por esto se planteó como un problema el método de exploración que tendría el robot en el entorno y la generación de datos para el entrenamiento de la red neuronal. Este método de exploración se relaciona en el segundo objetivo específico de que el robot tenga la posibilidad de descubrir su dominio y sus posibilidades de actuación. Luego, al utilizar un modelo de utilidad y el proceso deliberativo de toma de decisiones se incluyen componentes de la e-MDB y esto se relaciona con el tercer objetivo específico.

La tercera etapa corresponde a una de validación en simulación primeramente, donde se identificaron trabas y mejoras para las morfologías implementadas. Luego de aplicar cambios, estas se implementaron en el robot real, para así cumplir con el cuarto objetivo específico.

### 3.5.3 Investigación

Para este proyecto las principales fuentes de consulta fueron usuarios líderes que ya hayan inventado soluciones para satisfacer las necesidades de este proyecto. Es por esto que, para temas de la arquitectura e-MDB se consultó directamente con el investigador a cargo (Alejandro Romero) ya que, la publicación de su tesis doctoral sucedió en setiembre de 2022, por lo tanto, los artículos científicos publicados hasta el momento habían sido de él, además, él era el encargado de otros proyectos en robótica que involucraban la arquitectura que se estaban desarrollando en el GII durante el periodo en que se desarrolló este proyecto.

Por otro lado, para la temática del robot EMERGE se consultó directamente con el investigador a cargo (Martín Naya) el cual estaba familiarizado con los módulos robóticos, ya que, él había trabajado con ellos anteriormente, así como consultas relacionadas a la implementación de la simulación.

Esta metodología permite una exploración profunda de las posibilidades técnicas, fomentando la innovación y la adaptabilidad en soluciones robóticas al centrarse más en el aspecto investigativo. Este método fomenta la colaboración entre disciplinas, lo que permite que el proyecto se desarrolle con una base sólida en investigación y desarrollo técnico, lo que es esencial para enfrentar los desafíos complejos de la ingeniería mecatrónica.

## 3.6 Implementación en el robot real

Para la implementación en la realidad se pensaron en dos alternativas: crear un sistema de visión para imitar la identificación del punto en el espacio que corresponde a la percepción de las coordenadas, o la implementación del sistema de captura de movimiento de cámaras OptiTrack, disponible en el laboratorio de investigación del GII.

El criterio de selección que tuvo más peso para elegir la herramienta fue la disponibilidad del sistema de cámaras OptiTrack, porque este se asemeja más al funcionamiento de CoppeliaSim. Además aprovechar este recurso implicaba poder generar una documentación tipo manual de usuario que sería de mucha ayuda para el GII y cumple con la necesidad 9, de generar documentaciones.

# Capítulo 4

# Propuesta de Diseño

En este capítulo crucial del proyecto, se adentrará en el detallado proceso de adaptación de los nodos de conocimiento de la e-MDB al diseño y funcionamiento del robot modular EMERGE. A través de un enfoque sistemático y riguroso, se delinearán los pasos seguidos para integrar eficazmente estos componentes, asegurando que el robot cumpla con las especificaciones establecidas. Este capítulo no solo refleja el esfuerzo conjunto y la sinergia entre los conocimientos teóricos adquiridos durante la formación académica y su aplicación práctica, sino que también marca un punto de inflexión en la forma en que los sistemas robóticos modulares pueden ser adaptados y optimizados para enfrentar los desafíos del futuro.

## 4.1 Acciones y percepciones del robot

Esta primera fase de diseño fue esencial para definir la manera en que el robot percibiría el dominio por medio de los sensores y como interactuaría con este por medio de sus acciones. En esta sección se tomaron en cuenta las posibilidades disponibles por medio de la simulación con la herramienta CoppeliaSim, así como las posibilidades del robot real para su implementación más adelante. Además, era crucial tomar decisiones en conjunto con el compañero Carlos Jara para mantener la coherencia entre proyectos e implementaciones.

### 4.1.1 Herramienta CoppeliaSim

El uso de la herramienta CoppeliaSim fue una de las necesidades del cliente, además del uso de la programación por medio de Python. Ahora, como parte del proceso de diseño se verificó que CoppeliaSim fuese la herramienta ideal para este proceso y a continuación se detalla por qué, además de qué versión se decidió utilizar.

De la tabla 4.1 se identifica que la versión educativa, CoppeliaSim Edu es la que ofrece las

**Tabla 4.1:** Versiones disponibles de CoppeliaSim y sus características [25]

| Versión | Funciones completas de simulación | Funciones completas de edición | Uso comercial | Gratuito |
|---|---|---|---|---|
| **Player** | Sí | No | Sí | Sí |
| **Pro** | Sí | Sí | Sí | No |
| **Edu** | Sí | Sí | No | Sí |

mejores características para este caso, porque incluye todas las funcionalidades de edición y simulación, además tiene la ventaja de ser gratuita. Lo único que no ofrece es el uso comercial, pero para este caso eso no es relevante.

### Descripción de la herramienta

CoppeliaSim Edu, que anteriormente se llamaba V-REP, es una plataforma de simulación robótica avanzada y versátil destinada a la educación, la investigación y el desarrollo en robótica y áreas relacionadas. La capacidad de modelar, simular y controlar sistemas robóticos complejos y dinámicos de entornos en tiempo real o de manera acelerada es una de las características más destacadas de esta herramienta.

- **Interfaz Gráfica Intuitiva:** permite a los usuarios construir y personalizar sus modelos robóticos y entornos de simulación mediante una interfaz gráfica de usuario (GUI)
- **Simulación de Sensores y Actuadores:** soporta una amplia gama de sensores y actuadores simulados, proporcionando una aproximación realista a la interacción con el mundo físico.
- **Scripting y Programación:** ofrece opciones flexibles de programación

CoppeliaSim Edu utiliza motores de física de alto rendimiento, como Bullet, ODE (Open Dynamics Engine) y Vortex, para proporcionar simulaciones realistas, colisiones y otras interacciones físicas. Estos motores permiten simular con precisión el comportamiento de los objetos en un entorno tridimensional, incluyendo la gravedad, la fricción y las fuerzas de impacto, entre otros aspectos. EL motor de física utlizado fue el Bullet (V2.78) recomendado por Martín Naya, ya que es el que han utilizado en otros proyectos en el GII.

### Conexión mediante Remote API con Python

Una de las funcionalidades más potentes de CoppeliaSim Edu es su capacidad para conectarse con Python a través de la ZeroMQ Remote API. Esto es muy útil para la implementación de algoritmos de inteligencia artificial, el control de las acciones del robot y la obtención de las percepciones.

La conexión mediante la ZeroMQ Remote API a Python se realizó siguiendo las instrucciones proporcionadas directamente por Coppelia Robotics en su manual de usuario [26].

Tambien, se utilizó el tutorial *BubbleRob* para aprender a utilizar funcionalidades de CoppeliaSim Edu [27].

### 4.1.2 Módulos Robóticos para simulación

Ahora, otra de las ventajas que posee el simulador CoppeliaSim es que los módulos robóticos ya fueron creados y listos para utilizarlos en la simulación. Estos se encuentran en la página oficial del proyecto EDHMOR (Evolutionary Designer of Heterogeneous Modular Robots), que entrena robots modulares como el EMERGE a través de algoritmos evolutivos en el lenguaje de programación Java [28].

En el directorio edhmor/edhmor/models/edhmor/emergeModules se encuentran las escenas y los modelos correspondientes de los módulos robóticos de EMERGE y la base a utilizar [28]. En la figura 4.1 se muestra unos ejemplos del módulo robótico y la base cuadrada disponibles de EMERGE.

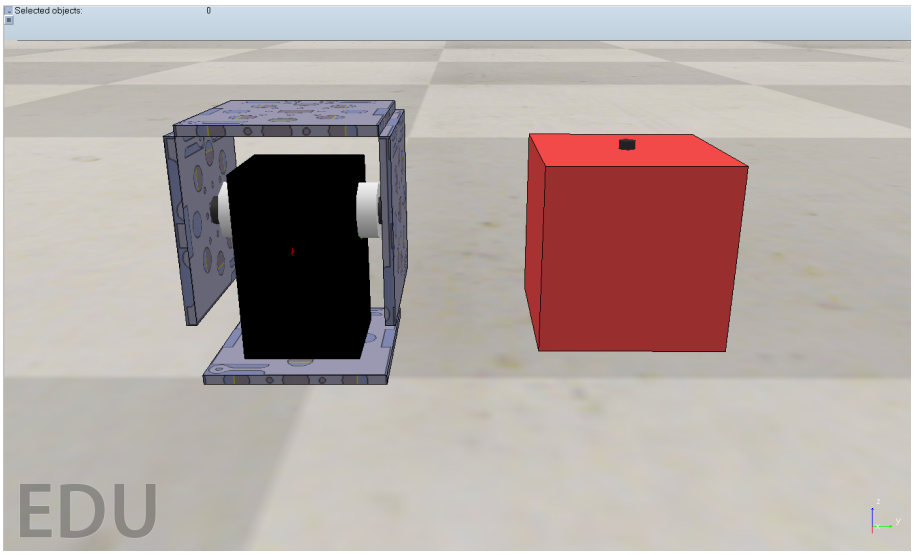



**Figura 4.1:** Escenas del módulo (izquierda) y la base cuadrada (derecha) de EMERGE. Elaboración propia.

### 4.1.3 Actuadores disponibles

Dado que cada módulo EMERGE posee únicamente un actuador, que corresponde a su motor Dynamixel AX-12 [29], se determinó que las acciones se debían definir a partir de este actuador. A continuación se estudian las posibilidades a nivel de simulación y la realidad.

### Mediante simulación

En la simulación cada uno de los módulos posee un motor (revolute joint). En la figura 4.2 se muestra el eje de referencia de estos, donde la posición vertical del motor corresponde a la posición de 0° y cuando el motor llega al final de su trayectoria y se encuentra en una posición horizontal, el máximo valor al que llega es a 90° positivos o negativos. En la figura 4.3 se muestra cómo se ve este movimiento en la simulación al utilizar los módulos.

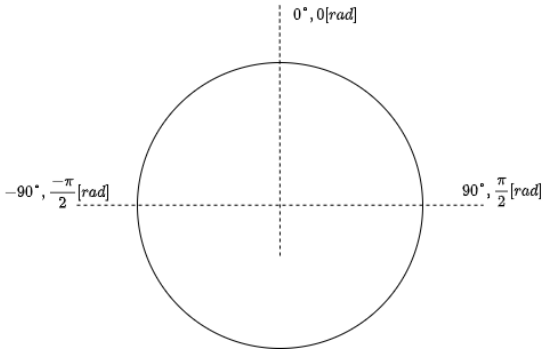



**Figura 4.2:** Eje de referencia del movimiento de los motores en simulación.

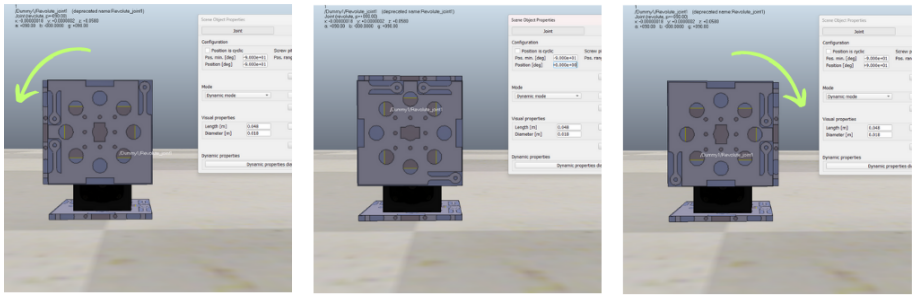


**Figura 4.3:** Demostración del eje de referencia de los motores en simulación. Elaboración propia.

### En el robot real

En cuanto al robot real, los motores más bien tienen un rango de movimiento de hasta 300° como se muestra en la figura 4.4. Por lo tanto, para que fuese coherente el sistema de referencia de los motores en simulación con los motores de la realidad, se decidió realizar una adaptación mediante código para que los ejes de referencia fuesen iguales en ambos dominios.

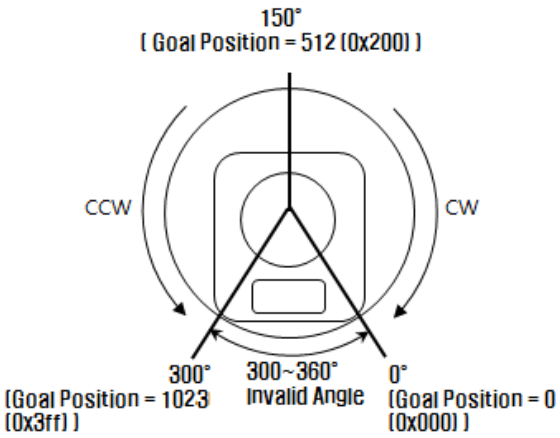



**Figura 4.4:** Eje de referencia del movimiento de los motores Dynamixel AX-12 [29].

### 4.1.4 Sensores disponibles

#### Mediante simulación

En simulación existe una variedad muy grande de sensores disponibles para el uso, que se le pueden agregar al robot dependiendo del uso que se le quisiera dar. Se tenían disponibles sensores de proximidad, sensores de visión y hasta cámaras, así como el mismo eje de referencia de la simulación que permite obtener las coordenadas de la posición de los elementos del robot medir distancias entre elementos.

Por otro lado, en la simulación también se obtiene la medición del ángulo en el que se encuentra el motor del robot.

#### En el robot real

Los sensores disponibles para la implementación en el robot real dependían de la implementación del proyecto del compañero Carlos Jara. También, como sensores se podía utilizar el sistema de captación de movimiento de las cámaras OptiTrack disponibles en el laboratorio, para obtener las coordenadas en el espacio de la posición del robot y la posición de los objetivos.

Otro sensor disponible para la obtención de la percepción y el estado actual del robot correspondía a la misma medida que toman los motores del robot sobre la posición actual en que se encuentran.

### 4.1.5 Elección final

Por lo tanto, las acciones disponibles se definieron como el delta, es decir, la variación a aplicar, en radianes, determinada en función del estado actual del motor. Las acciones pueden ser variaciones positivas o variaciones negativas mientras que se mantengan en el

rango de $\frac{-\pi}{2}$ a $\frac{\pi}{2}$. La longitud del buffer de acciones depende de la cantidad de módulos presente en la morfología del robot. En la ecuación 4.1 se muestra como se definieron las acciones en el código.

$$action = [\Delta_{\theta 1}, \Delta_{\theta 2}, ..., \Delta_{\theta n}], \theta \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] [rad] \tag{4.1}$$

En cuanto a las percepciones del robot, a partir de los sensores disponibles se elige utilizar dos: las coordenadas en el espacio, y la posición de cada motor.

Las medición de las coordenadas generan versatilidad ya que se puede calcular distancia entre objetos, porque a partir de las posiciones se puede determinar las distancias. Esta percepción se mide en milímetros, puesto que, se adecúan mejor al tamaño del robot, que es pequeño y hacer las mediciones en metros implica menos precisión en el sistema. Además, como el robot se mueve en el plano *X, Z* del simulador se decide poner simplemente estas dos variables, porque si se le añadía el eje $Y$ entones este simplemente estaría añadiendo ruido al sistema porque no proporcionaba información útil para la posición del robot.

Por otro lado, el sensor de la posición del motor es muy útil también para dar apoyo a la percepción anterior, así juntas disminuyen las ambigüedades o confusiones para el robot.

$$perception = [x, z, \theta_1, \theta_2, ..., \theta_n], \theta \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] [rad] \tag{4.2}$$

## 4.2 Selección de Morfologías

Para el desarrollo de este proceso de investigación se eligieron utilizar cuatro morfologías. Cada morfología posee un nivel de dificultad mayor a la anterior de manera que se fuera desarrollando poco a poco el programa en código, así como los arprendizajes adquiridos.

Inicialmente se propusieron utilizar las morfologías que se muestran en la figura 4.5. De izquierda a derecha en el orden de implementación mencionado: Modular01, Modular02, Modular03 y Modular02b. Donde el Modular01 posee un único módulo que corresponde a la configuración más sencilla, y en las demás se va añadiendo un módulo extra en forma de brazo robótico, para determinar puntos de ambigüedades y error. Estas morfologías se implementaron en dos dimensiones, por lo que se decidió agregar una cuarta morfología ubicada para funcionar en tres dimensiones para añadirle un nivel más de dificultad.

Cabe resaltar que las decisiones sobre las morfologías a desarrollar se tomaron en conjunto con el equipo de trabajo (Carlos, Martín y Alejandro).

1. **Modular01:** Esta morfología corresponde a la configuración más sencilla que se puede lograr, ya que consiste de un único módulo. Se eligió trabajar con esta morfología primero para poder avanzar en la programación del código, mientras el

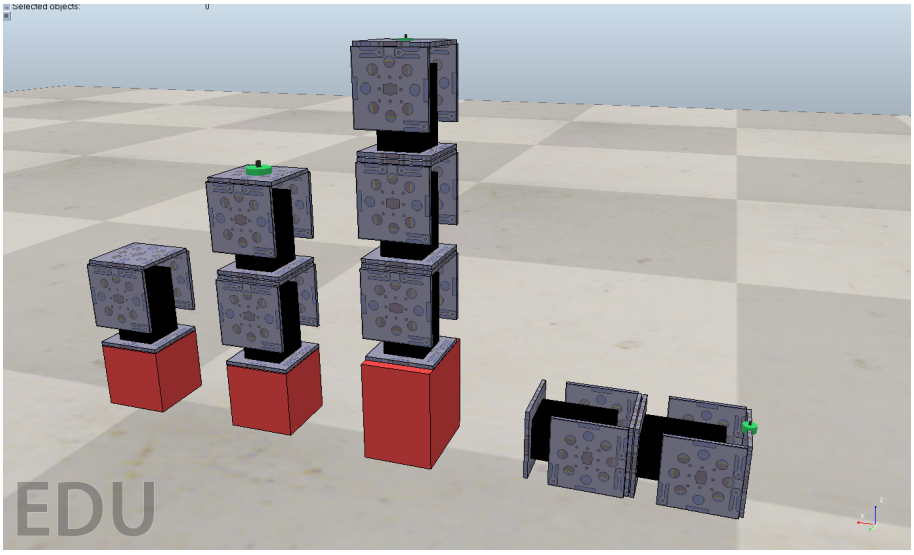



**Figura 4.5:** Morfologías seleccionadas para la implementación. Elaboración propia.

compañero Carlos Jara avanzaba con su proyecto de aprendizaje de los modelos de mundo de las demás morfologías. Principalmente porque para esta morfología es más sencillo obtener las ecuaciones de cinemática directa que describen su movimiento y así generar un modelo de mundo por medio de ecuaciones para así ir desarrollando el proyecto.

2. **Modular02:** Esta morfología se eligió para añadirla un nivel más de dificultad a la anterior. De igual manera se eligió trabajar en dos dimensiones para avanzar poco a poco. En este caso ya se utilizaron ecuaciones en vez del modelo de mundo porque aún este no estaba listo. Se aprovechó el aumento de módulos para adaptar el código modular, de manera que se convitió en un código genérico para todas las morfologías.

3. **Modular03:** Para este caso, se añadió un módulo extra y final porque según la literatura del EMERGE este mecanismo puede soportar hasta 3 módulos en serie. Esta morfología sí representaba mayor dificultad al generar las ecuaciones que representasen al modelo de mundo, por lo que se utilizó únicamente cuando ya estaba listo el modelo de mundo entrenado.

4. **Modular02b:** Finalmente, esta morfología representaba un salto a tres dimensiones y un mayor nivel de dificultad para el entrenamiento de los modelos de mundo y el uso de los modelos de utilidad.

La implementación de cada morfología se desarrollará en las siguientes secciones, así como las trabas enfrentadas y resueltas para cada una de ellas. Es importante mencionar que la morfología inicial corresponde a la Modular01.

## 4.3 Exploración del espacio de estados

Como primer paso antes de implementar el proceso deliberativo de toma de decisiones como tal, se decidió comenzar a hacer experimentos para la exploración del espacio de estados del robot, utilizando la morfología inicial (Modular01). Donde se implementaran diferentes variables para construir la mejor base de datos para el entrenamiento del modelo de utilidad. Primero, se realizó un diagrama de flujo para identificar el procedimiento a realizar, este se muestra en la figura 4.6.

### Diagrama de Flujo

En el diagrama de flujo se establece que primero hay que asignar un objetivo para que el robot lo encuentre en el espacio. Esto es importante para que el robot determinara el final de cada traza. Luego, como las acciones son cambios con respecto a la posición actual del robot, entonces primero se debe leer ese estado actual. Luego, se elige una acción para aplicar y se aplica. Una vez realizado esto se almacenan los valores del estado actual leído anteriormente y de la acción aplicada para ir creando la base de datos del espacio de estados. Luego, si se alcanzó el objetivo entonces se termina la traza, si no, entonces se vuelve a obtener el estado actual y repite el proceso. Una vez alcanzado el objetivo se almacena el último estado del robot (donde alzancó el objetivo) y a este se le asigna una utilidad de 1, y para los demás pasos se les asigna una utilidad esperada de manera retro-pagada. Luego se revisa si se alcanzaron la cantidad de trazas establecidas para la base de datos, si no, se continúa con una nueva posición aleatoria para la siguiente traza y se repite todo el proceso.

Es clave mencionar que se determinó que para validar la generación de datos se entrenó un modelo de utilidad, que por medio de sus gráficas de predicción de valores y la pérdida se pudiese corroborar el buen funcionamiento requerido del modelo.

### Generación de conceptos

Para este caso, se generaron algunos conceptos para algunas variables mostrados en la tabla de combinación de conceptos que se muestra en la figura 4.7 y se explican a continuación:

- **La asignación del objetivo:** en este caso se realizó una lluvia de ideas porque el objetivo podría ser asignado por medio de coordenadas o que el robot alcanzara una posición angular específica.
- **El estado actual o percepción del robot:** si bien es cierto, anteriormente se determinó qué percepciones se podrían utilizar del robot, en este caso estas percepciones dependerían del objetivo a desarrollar. Puesto que si el objetivo elegido sería alcanzar una posición angular no serían necesarias las coordenadas en el espacio.

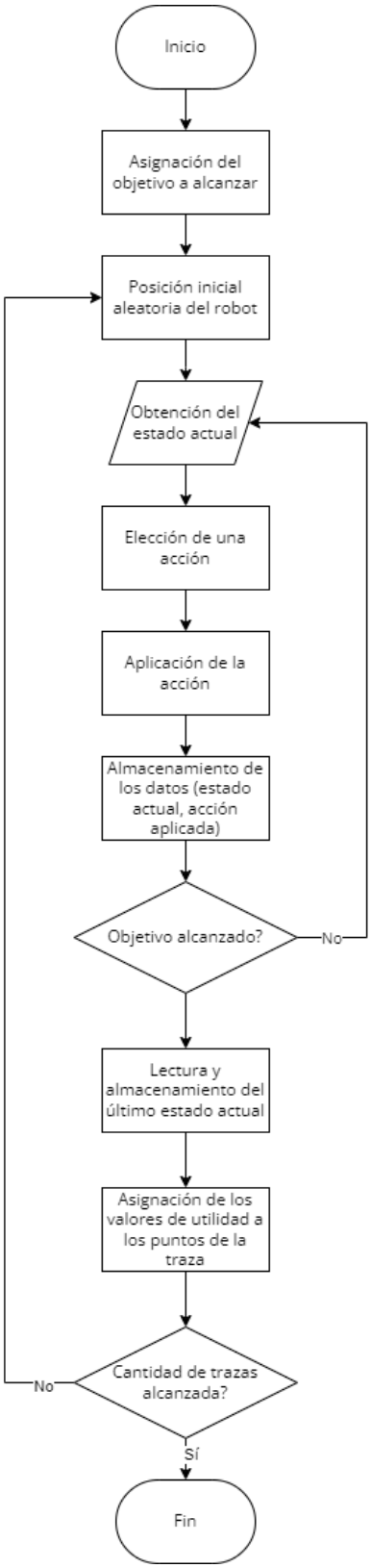



**Figura 4.6:** Diagrama de flujo del algoritmo de exploración del robot. Elaboración propia.

- **Método de elección de las acciones:** según la literatura estas podrían elegir por medio de la motivación intrínseca de novelty, donde se elige la acción más novedosa para el robot. Sin embargo, también se podrían elegir acciones aleatorias como una alternativa.

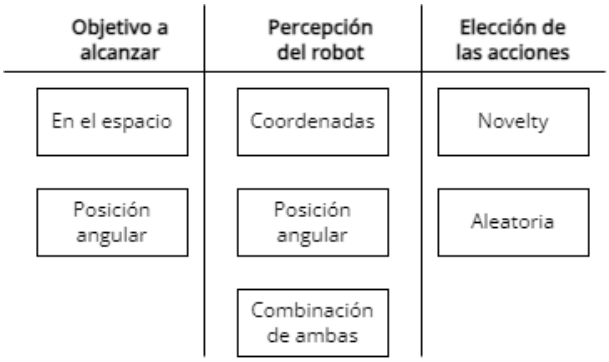



**Figura 4.7:** Tabla de combinación de conceptos para la exploración del espacio. Elaboración propia.

#### Planeamiento de experimentos

Para este caso, a partir de los conceptos desarrollados se prefirió realizar un proceso de experimentación donde el progreso se realizara de manera incremental. Para esto se debe comenzar desde la opción más simple y a partir de ahí ir añadiendo características al sistema. Por esto se elige primero asignarle al robot un objetivo a alcanzar que consistiese en una posición angular, y por ende su percepción también.

También, se podría haber elegido el método de selección de acciones por medio de la motivación intrínseca de novelty para implementar de una vez LOLA según la teoría explicada en la sección 2.4. Sin embargo, para probar que novelty de verdad es una solución factible para este proyecto se decidió realizar primero una exploración aleatoria y luego una exploración por novelty para comparar los resultados, y finalmente seleccionar el mejor método. Además, durante este proceso de experimentación se encontraron trabas inesperadas y generaron mejoras que se explicarán más adelante.

Cabe destacar que estas experimentaciones se realizaron siguiendo el proceso de validación de conceptos donde se definió un objetivo, determinación de valores de prueba y la presentación de resultados, y finalmente el análisis de los resultados. Para esta etapa específicamente.

### 4.3.1 Exploración aleatoria

Para esta primera exploración se planteó el entrenar la red neuronal artificial del modelo de utilidad con una cantidad grande de datos, para asegurar que se tengan datos de todo

el espacio de estados. El objetivo asignado era que el robot alcanzara la posición angular de 50° en su motor.

### Valores de prueba

Para definir la cantidad de pasos por trace mostrado en la tabla 4.2, se determina porque el robot avanza un máximo de 7.5° por cada time step de simulación. Entonces si el robot se encuentra en el extremo de -90° tiene que moverse 140° hasta llegar al objetivo de 50°. Sin embargo, si las acciones aleatorias son muy bajas de menos de 2° entonces el robot avanzaría muy lento hasta llegar al objetivo. Entonces para casos como este es que se elige esta cantidad de pasos. También, son 5000 datos para tener al menos 50 trazas para entrenar.

**Tabla 4.2:** Valores de prueba para el experimento de exploración aleatoria

| **Método de exploración** | **Cantidad de pasos por trace** | **Cantidad de datos a generar** | **Método de asignación de valores de utilidad** |
|---|---|---|---|
| Aleatoria | 100 | 5000 | Retropropagación |

### Datos obtenidos de la exploración

Una vez lanzado el experimento siguiendo el diagrama de la figura 4.6 se obtienen las trazas que se muestran en la figura 4.8 como resultado. De estas trazas se puede ver como se le asignaron valores de utilidad un poco desordenados, porque en la traza 10 se le asigna la utilidad de 0.0 a la posición angular de 57°, mientras que en la traza 11 se le asigna el valor de utilidad 0.0 a la posición angular de 80°. Por lo tanto, se decidió no entrenar un modelo de utilidad con estos datos por la ambigüedad que presentaron al asignar los valores de utilidad esperada.

Más bien se propuso recortar las trazas al promedio de pasos realizados en este experimento que sería de 20 pasos por traza, para de esta manera no generar valores tan ambiguos al asignar la utilidad a la percepción obtenida.

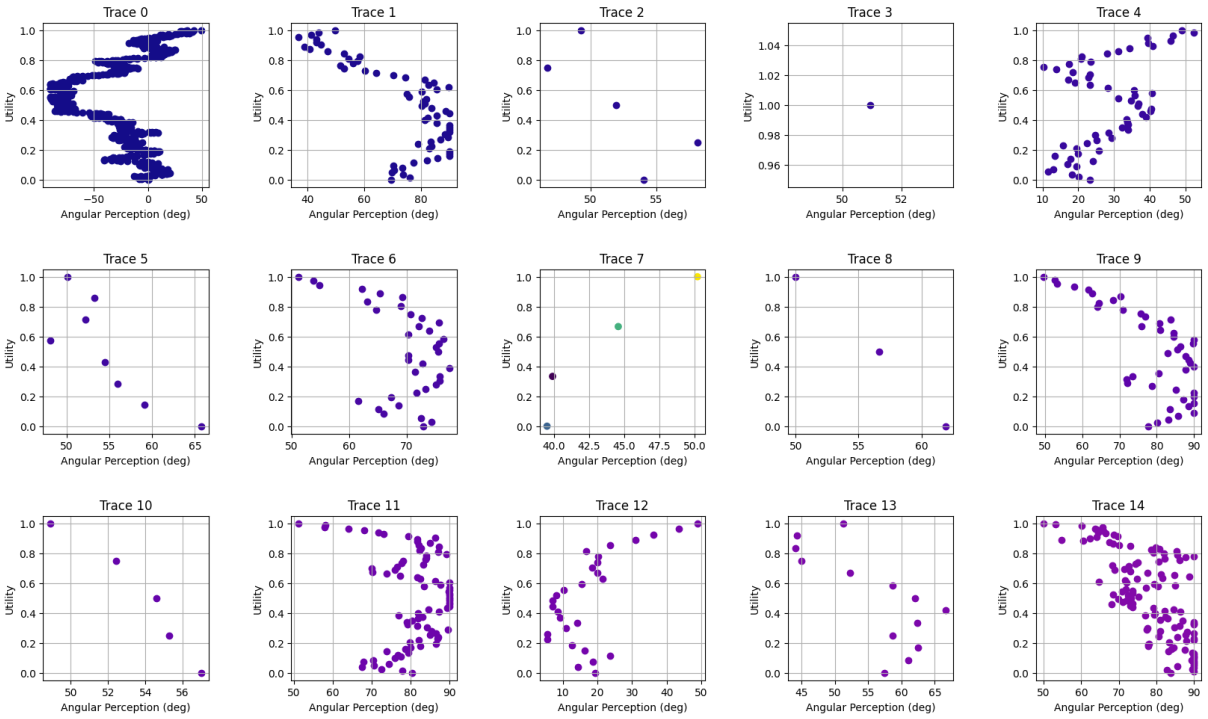



**Figura 4.8:** Trazas obtenidas por exploración aleatoria. Elaboración propia.

### 4.3.2 Exploración aleatoria con trazas recortadas

#### Valores de prueba

Este experimento posee el mismo propósito del anterior pero esta vez se cambiaron los valores de prueba, que se muestran en la tabla 4.3. Donde se eligió crear una memoria tipo FIFO (first in first out) donde se almacenasen únicamente los últimos 20 pasos que haya realizado el robot en su exploración.

**Tabla 4.3:** Valores de prueba para el experimento de exploración aleatoria y memoria FIFO

| Método de exploración | Cantidad de pasos por trace | Cantidad de datos a generar | Método de asignación de valores de utilidad |
|---|---|---|---|
| Aleatoria | 20 | 400 | Retropropagación |

#### Datos obtenidos de la exploración

Para los datos obtenidos de esta exploración se mostró una mejora. Ya no habían valores de utilidad 0.0 distribuidos a lo largo de todo el espacio de percepciones. Sin embargo, esta vez en la figura 4.10 se muestra como solo se le asignaron valores de utilidad esperada a una porción del espacio de estados, ubicada entre 0° y 90°. Se decidió igualmente

entrenar un modelo de mundo para este caso, puesto que una red neuronal de regresión podría solucionar este problema de falta de datos en las percepciones negativas.

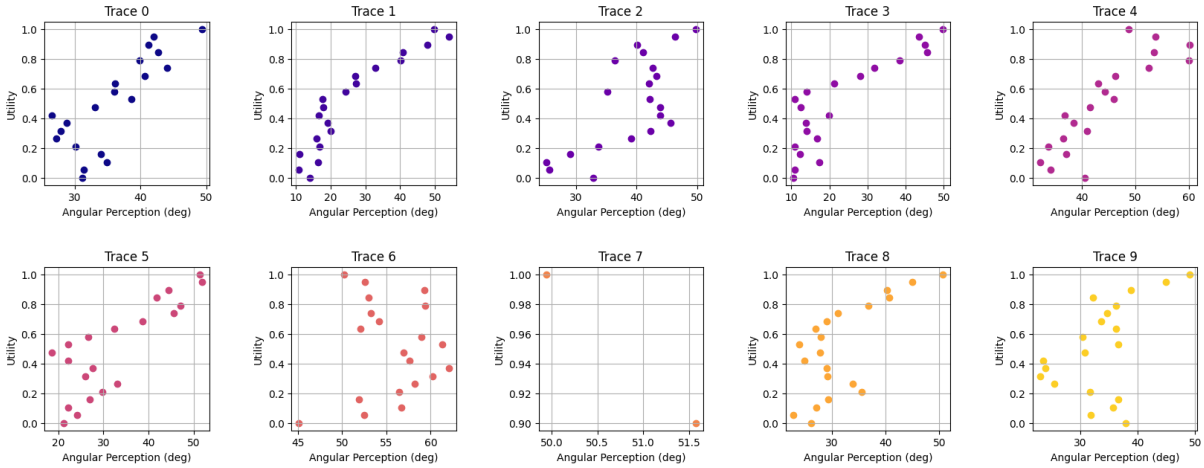



**Figura 4.9:** Trazas individuales obtenidas por exploración aleatoria y la memoria de 20 pasos. Elaboración propia.

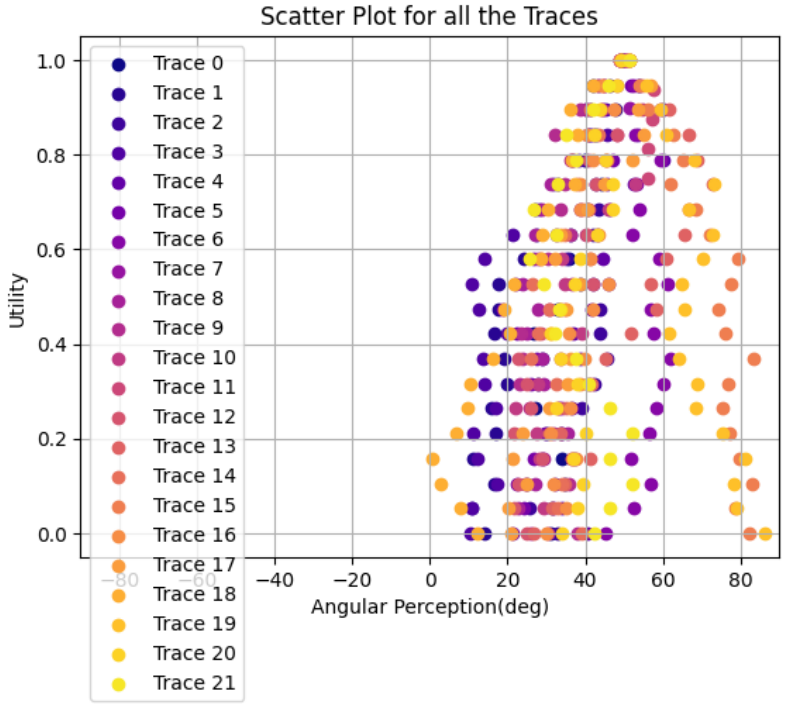



**Figura 4.10:** Grupo de trazas obtenidas por exploración aleatoria y la memoria de 20 pasos. Elaboración propia.

### Entrenamiento de la red neuronal

Entonces, se eligió una red neuronal de tipo regresión para que esta pudiese generar una aproximación para las zonas desconocidas del espacio de estados.

Se realizaron dos iteraciones para el entrenamiento de esta red neuronal. Esta red neuronal debía contener una neurona como entrada donde entra el valor de la percepción, luego la estructura de neuronas implementada y finalmente como salida debía dar un único valor de regresión que representase la utilidad dada de la percepción a analizar, como se muestra en el diagrama de la figura 4.11.


**Figura 4.11:** Diagrama de red neuronal deseada. Elaboración propia

Para el estudio de hiperparámetros primero se eligió la función de pérdida: huber, porque es menos sensible a los valores atípicos en los datos en comparación con MSE, debido a que para errores grandes se comporta como MAE, lo cual previene que los errores grandes tengan demasiado peso, y para errores pequeños se comporta como MSE, proporcionando una solución más estable y suave durante la optimización [30].

Por otro lado, la función de activación elegida para las capas ocultas fue la sigmoide porque transforma los valores de entrada en un rango entre 0 y 1, y la función de activación de la salida sería la lineal.

Luego, los hiperparámetros estudiados se encuentran en la tabla 4.4. Cabe resaltar que en esta etapa de prueba no se estaba probando el modelo de utilidad como tal, si no, la combinación adecuada de variables para la generación de trazas, con el objetivo de generar una exploración eficiente del espacio de estados. Por lo que no se adentrará en el entrenamiento de la red neuronal en esta etapa aún (más adelante se entrena el oficial).

**Tabla 4.4:** Estudio de hiperparámetros para el entrenamiento de la red de exploración aleatoria

| Iteración | Número de capas | Cantidad de neuronas por capa oculta | Optimizador | Learning Rate | Epochs |
|---|---|---|---|---|---|
| 1 | 2 | Capa1: 3, Capa2: 3 | Adagrad | 0.001 | 50 |
| 2 | 2 | Capa1: 6, Capa2: 6 | Adam | 0.01 | 75 |

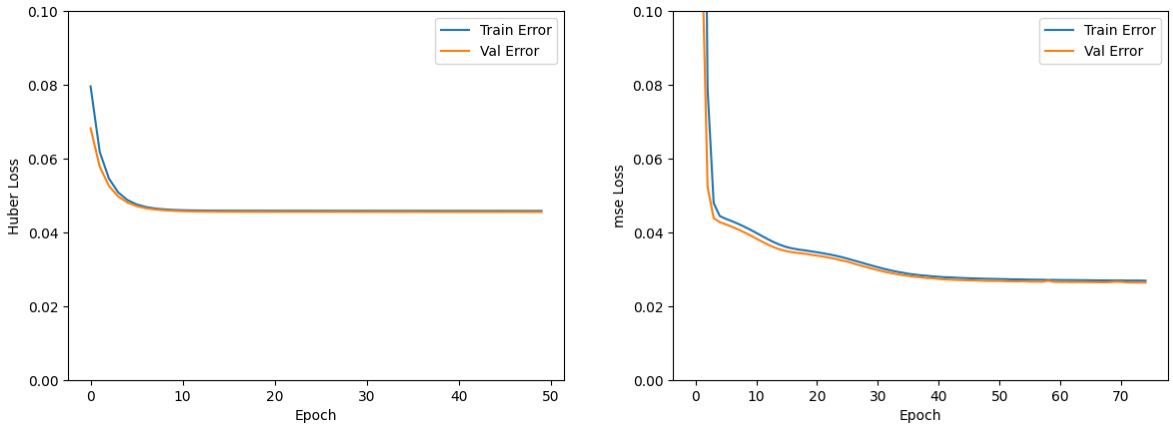
De el entrenamiendo en la primera iteración se obtuvo una pérdida del 20%, por lo que se descartó esta iteración. En la segunda iteración, donde se añadieron más neuronas a las capas ocultas y se aumentó la taza de aprendizaje se obtuvo una pérdida del 4%, lo que representa que la red neuronal logra realizar predicciones más acertadas de acuerdo a los datos que se le dieron. Entonces se elige la red neuronal entrenada en la segunda iteración para visualizar las predicciones que realizaría este modelo de mundo

### Modelo de Utilidad resultante

El modelo resultante del entrenamiento anterior, se probó de manera que se le introdujeron posiciones angulares entre -90° y 90°, en incrementos de 1° para obtener la utilidad predicha por el modelo de utilidad, y estas predicciones se graficaron en la figura 4.13.

De esta gráfica se obtiene que un modelo de mundo que predice las utilidades bien para únicamente para el rango de valores que conoce, que fue mencionado anteriormente (entre 0° y 90°). Porque existe una pendiente que le va asignando más utilidad conforme se acerque a la posición angular de 50°. Sin embargo, en las posiciones angulares negativas el modelo de utilidad tiene una pendiente con una inclinación levemente invertida, donde la utilidad disminuye en cuanto se acerca a la posición de 0°. Por lo tanto, este modelo de utilidad no representaba correctamente el espacio de utilidades y se tenía que mejorar.

Para arreglar esto, se decidió utilizar la motivación intrínseca de novelty para generar una exploración del espacio de estados más eficiente. Porque esta gráfica demostró que la red neuronal aprende los datos que conoce. Así que no se puede entrenar sin datos lo suficientemente representativos.




**(a)** Iteración 1 **(b)** Iteración 2

**Figura 4.12:** Gráficas de pérdida del entrenamiento de la red de exploración aleatoria. Elaboración propia.

## 4.3.3 Exploración por Novelty

Para mejorar la exploración mediante motivación intrínseca de novedad, se aplicó una limitación en la retropropagación de los valores de utilidad esperada. Esto permite un entrenamiento más preciso al asignar valores de utilidad, reconociendo que posiciones cercanas al objetivo deben tener una utilidad alta. Por lo tanto, se sugiere escalonar la utilidad en 20 pasos desde 0.0 hasta 1.0, aumentando en 0.05 por paso, para reflejar con mayor exactitud el valor de cada posición en la trayectoria hacia el objetivo.

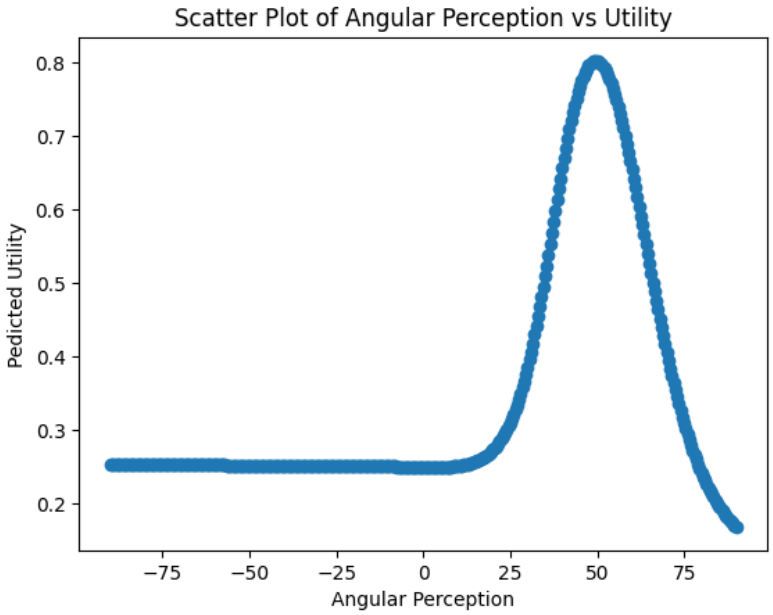



**Figura 4.13:** Modelo de utilidad obtenido de una exploración aleatoria. Elaboración propia.

### Propósito

Entrenar la red neuronal artificial del modelo de utilidad por medio de una exploración por novelty para que el robot alcance una posición angular de 50° en el motor.

### Valores de prueba

**Tabla 4.5:** Valores de prueba para el experimento de exploración por novelty

| Método de exploración | Cantidad de pasos por trace | Cantidad de datos a generar | Método de asignación de valores de utilidad |
|---|---|---|---|
| Novelty | 20 | 10000 | Retropropagación restringida |

### Método de implementación de la exploración

En la teoría mencionada en la sección 2.4, se explica como la forma de implementar novelty es mediante el modelo deliberativo, el cual se adaptó de la siguiente manera para ser utilizado en la elección de las acciones por medio de novelty, y porque no se tiene un modelo de mundo entrenado.

En la figura 4.14 se muestra la adaptación realizada en el modelo deliberativo para ser adaptado a este caso. Primero, se obtiene el estado futuro que para este caso era simplemente sumarle la acción al estado actual y obtener el estado futuro. Luego, este estado futuro se pasa a través del algoritmo de novelty que utiliza el buffer de la trayectoria del

robot y compara qué estado futuro es más novedoso con respecto a la trayectoria que ya ha recorrido el robot. Como recordatorio la ecuación de novelty corresponde a la ecuación 2.3. Para finalmente elegir la acción relacionada con el estado más novedoso y así asegurar que el robot explore de manera más eficiente.

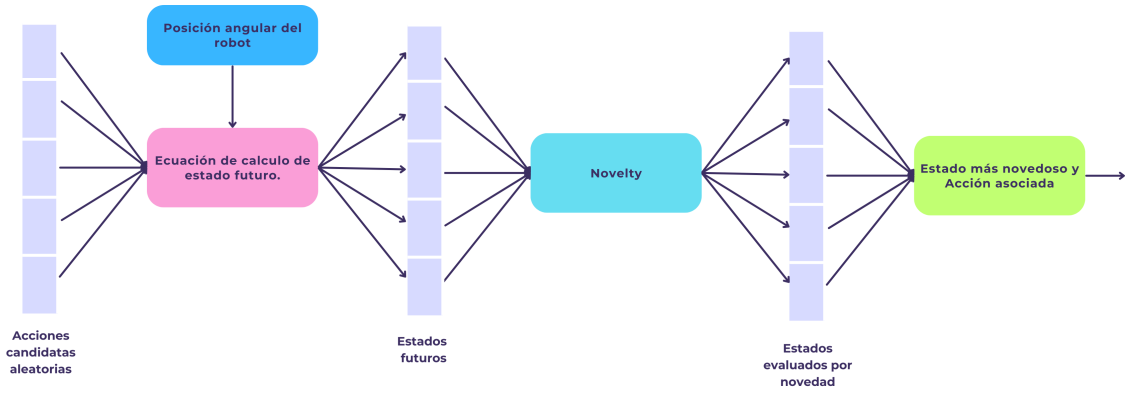



**Figura 4.14:** Implementación del Modelo Deliberativo para la exploración por Novelty. Elaboración propia.

### Datos obtenidos de la exploración

En la figura 4.15 se muestra como se obtuvieron trazas mucho más ordenadas que en los experimentos anteriores. Las trazas muestran como el robot avanza en una dirección establecida y a menos de que se enfrente con una traba, como lo es alcanzar los extremos (-90° y 90°), el robot sigue para adelante sin devolverse generando conocimiento menos "confuso". Luego, en la traza 111 y la traza 29 se muestra como los valores de utilidad se asignan con la retropropagación restringida y son más adecuados para el estado en que se encuentra el robot.

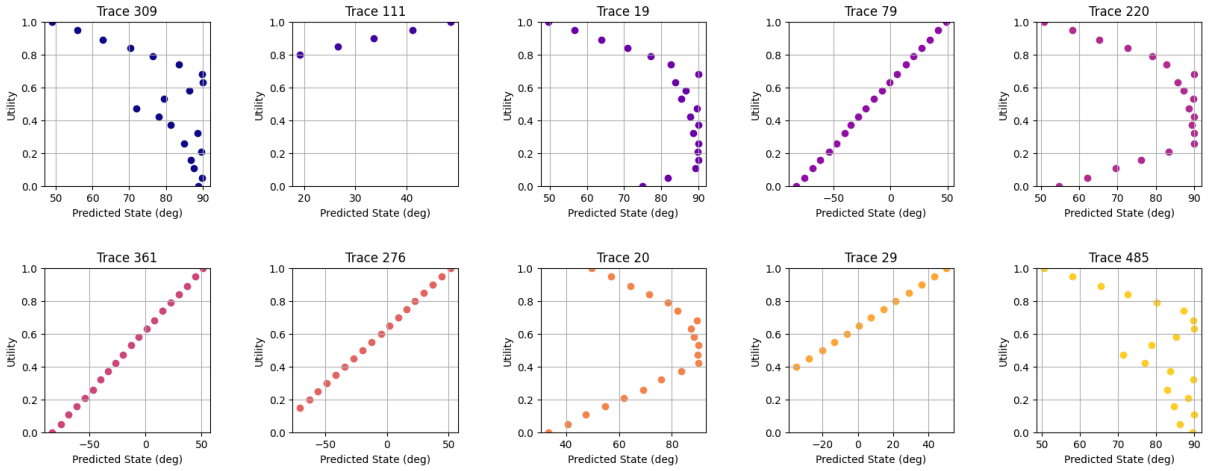


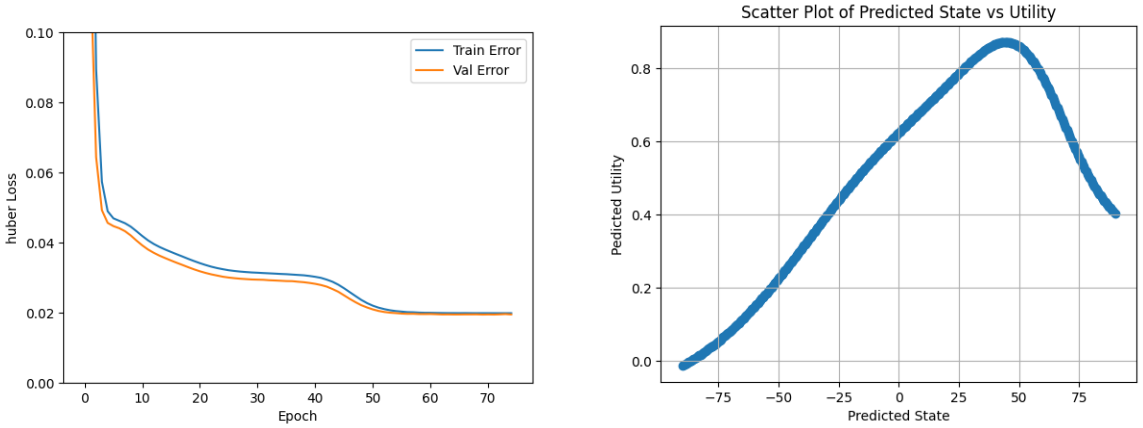

**Figura 4.15:** Trazas obtenidas de la exploración por novelty. Elaboración propia.

### Entrenamiento de la red neuronal

En este caso se elige utilizar la misma red neuronal diseñada en el experimento anterior. Sin embargo, esta vez se obtiene una pérdida del 2%, lo cual mejora al modelo de utilidad entrenado en el experimento anterior.

### Modelo de Utilidad resultante

Finalmente, en este experimento se obtuvo un modelo de utilidad que representa el espacio de estados con una utilidad adecuada para cada estado. En la figura 4.16 se muestra como este modelo de utilidad posee una pendiente tanto del lado de los estados negativo, así como de los estados positivos, lo cual le indica al robot que los estados se vuelven más llamativos en cuanto se acerca al objetivo.




**(a)** Pérdida de la red **(b)** Predicciones del modelo de utilidad entrenado

**Figura 4.16:** Modelo de utilidad obtenido de una exploración por Novelty. Elaboración propia.

## 4.3.4 Discusión

La fase experimental de este proyecto de graduación reveló descubrimientos clave para la futura selección de variables, destacando significativamente las siguientes tres variables:

- **Exploración por novelty:** La motivación intrínseca de novelty sí representa una mejoría en comparación con una exploración aleatoria, proporcionando una estrategia más eficaz y orientada al aprendizaje, de manera que el robot pudo descubrir las posibilidades de actuación mediante una asociación de las percepciones del dominio con sus posibilidades de actuación.
- **Asignación de los valores de utilidad esperada:** La correcta determinación de los valores de utilidad, ajustados al contexto específico del robot, resulta esencial para prevenir incoherencias en los datos de entrenamiento, optimizando así la calidad y la relevancia de la información recopilada.

- **Memoria FIFO para almacenar la trayectoria del robot:** Esta memoria representa una solución efectiva para el almacenamiento de las trayectorias del robot. Este enfoque garantiza la actualización constante de los datos, limitando el número de pasos por secuencia.

Esta etapa de experimentación representó una etapa de hallazgos importantes dentro de este proyecto de graduación, de donde se obtuvieron las bases para entender e implementar las siguientes etapas.

## 4.4 Diseño y obtención del Modelo de Utilidad

En esta sección se presenta el proceso de diseño de la segunda etapa del proyecto donde se debía implementar la generción de datos para el entrenamiento oficial del modelo de utilidad utilizado. Además, para esta etapa ya se tenía un modelo de mundo entrenado por el compañero Carlos Jara, para la morfología Modular02.

### 4.4.1 Generación de datos para el entrenamiento

Antes de comenzar con esta fase era importante mencionar que el modelo de mundo entrenado que como entradas recibe acciones de la forma mostrada en la Ec.4.1, estado actual de la forma mostrado en la ecuación 4.2 y los estados predichos en el mismo formato. En esta morfología se tienen 2 módulos entonces se evalúan dos posiciones angulares.

Para esta primer fase se utilizaron los principios aprendidos en la sección anterior: método de exploración por novelty, la retro propagación restringida y una memoria FIFO para la recolección de los datos, sin embargo más adelante estas se prueban para verificar que sean compatibles con este problema.

#### Asignación del objetivo a alcanzar

Se propuso utilizar una pelota en el aire como objetivo a alcanzar del robot, porque el simulador CoppeliaSim permite añadir objetos en el espacio de esta forma y, además, porque de esta forma se aprovecha el hecho de que el robot se mueve en el plano y el objetivo se añade dentro de su rango de movimiento, como se muestra en la figura 4.17.

El punto utilizado para medir la percepción del robot corresponde al force sensor (color verde) colocado en la parte superior del robot.

En la figura 4.18 se muestran los diferentes objetivos en ubicados en el espacio. Con el mismo formato. Nótese que para las primeras tres morfologías la ubicación es en el plano $XZ$, pero para la última morfología esta tendría capacidad de moverse por el espacio en tres dimensiones $XYZ$.

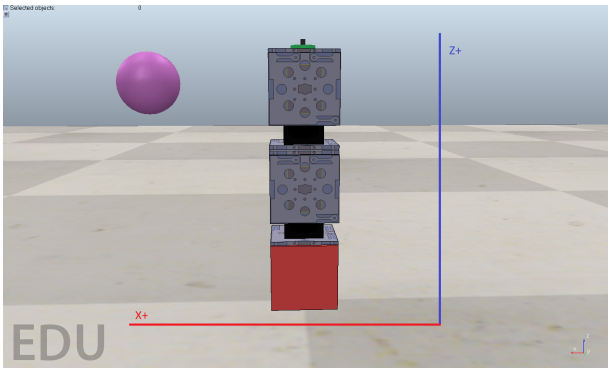
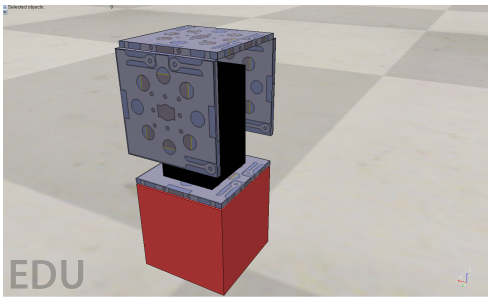
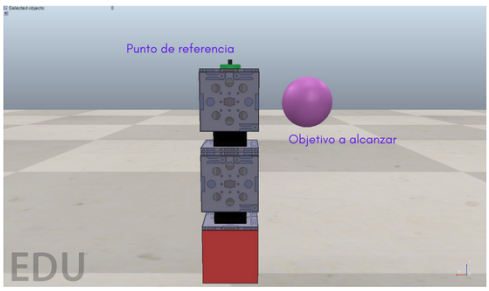


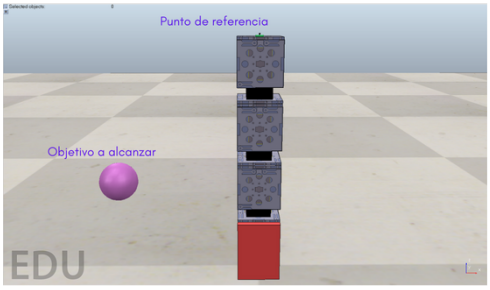

**Figura 4.17:** Ubicación en el espacio del objetivo y la morfología Modular02. Elaboración propia.



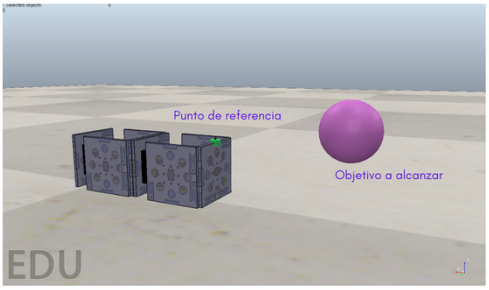

**(a)** Modular01




**(b)** Modular02




**(c)** Modular03




**(d)** Modular02b

**Figura 4.18:** Designación del identificador de las morfologías y los objetivos a alcanzar en el espacio. Elaboración propia.

### Análisis de la cantidad de pasos por traza necesarios

En el caso de la morfología Modular01 se obtuvo que 20 pasos eran suficientes para generar datos representativos del espacio. Pero, esta variable se probó para determinar si representaría al espacio de estados correctamente. En la figura 4.19 se muestra un ejemplo de una traza de 20 pasos y sus utilidades esperadas asignadas mediante retro-propagación restringida.

En la figura 4.19 se obtuvo una representación equivoca del espacio de estados. El objetivo a alcanzar del robot se encontraba en el círculo rojo mientras que los pasos dados por el robot y sus utilidades se representan mediante el esquema de colores. Los estados visitados por el robot se les asignaron una utilidad en orden, entonces se puede ver como

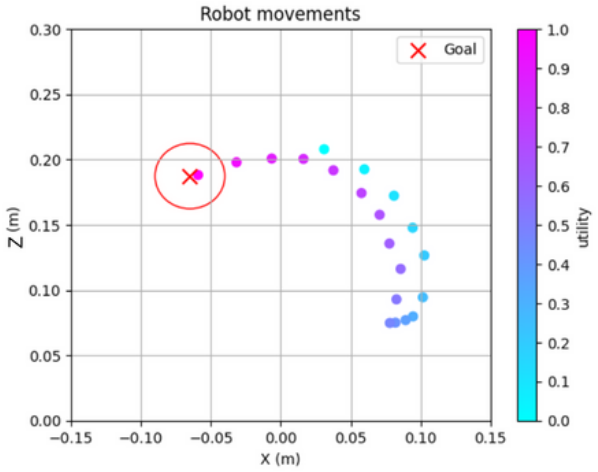



**Figura 4.19:** Utilidades asignadas una traza de 20 pasos. Elaboración propia.

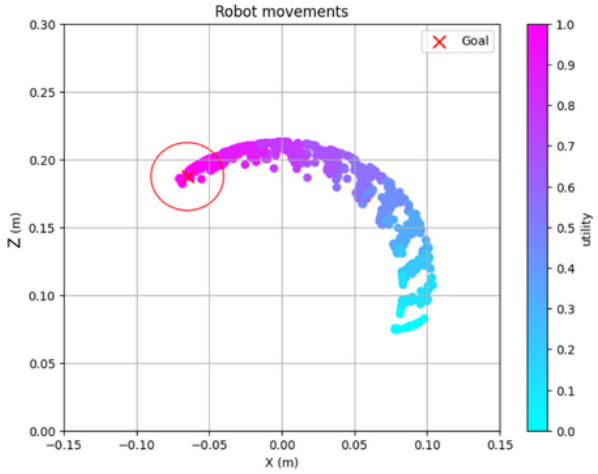



**Figura 4.20:** Utilidades asignadas en trazas de 10 pasos. Elaboración propia.

la utilidad más alta fue el último paso, y la utilidad más baja corresponde al primer paso.

Se ve como se le asignaron utilidades bajas entre 0 y 0.2 a los estados en color cian se encuentran relativamente cerca al objetivo y por ende deberían tener utilidades mayores, como los estados cercanos identificados en colo rosado, con utilidades entre 0.9 y 0.7.

Por lo tanto a partir de esta gráfica se determinó que la cantidad de pasos ideal sería de **10 pasos** en la memoria FIFO, porque se ve como el robot en el décimo paso (utilidad 0.5) se encuentra en la posición más lejana al objetivo y por ende los pasos anteriores se deberían descartar.

En la figura 4.20 se muestra un ejemplo de 10 trazas con posiciones iniciales aleatorias (únicamente del lado derecho del objetivo para hacer una mejor comparación) y de un máximo de 10 pasos por traza. En esta gráfica mediante el esquema de colores se puede ver como los espacios de utilidad se asignaron. Es decir, el espacio de estados se representó con mayor coherencia.

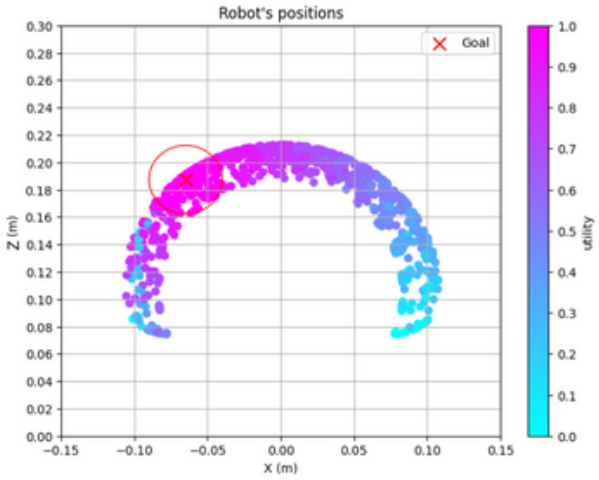



**Figura 4.21:** Trazas de entrenamiento y zonas de utilidad esperada. Elaboración propia.

### 4.4.2 Generación oficial de trazas

Con esto se definieron los parámetros para la generación de trazas oficial para el entrenamiento del modelo de utilidad, mostrados en la tabla 4.6, siguiendo el algoritmo de exploración de la figura 4.6.

**Tabla 4.6:** Valores de prueba para el experimento de exploración por novelty

| Método de exploración | Cantidad de pasos por traza | Cantidad de trazas a generar | Método de asignación de valores de utilidad |
|---|---|---|---|
| Novelty | 10 | 150 | Retropropagación restringida |

En las figuras 4.21 y 4.22 se muestran las trazas generadas por el robot en su proceso de exploración del espacio de estados. Mediante este método diseñado el robot logró dimensionar su espacio de estados categorizando los diferentes estados mediante una utilidad esperada. En ambas gráficas hay estados evaluados incorrectamente, como la traza 121 de la figura 4.22 que podrían generar ruido al momento del entrenamiento de la red neuronal. Entonces, para evitar esto y pensando en reutilizar este modelo de utilidad para otras morfologías, se propone un proceso de re-descripción de los datos, para que no se representen en coordenadas, si no en distancias normalizadas entre cero y uno medidas desde el centro del objetivo hasta el force sensor de la parte superior del robot. De esta manera la percepción del robot se generaliza para ser aplicado a más casos.

#### Normalización de los valores de la distancia

La traducción de las percepciones del robot a distancias normalizadas entre cero y uno, se representa en la gráfica de la figura 4.23. El valor de normalización se obtuvo de la mayor distancia posible que podría estar el objetivo del robot.

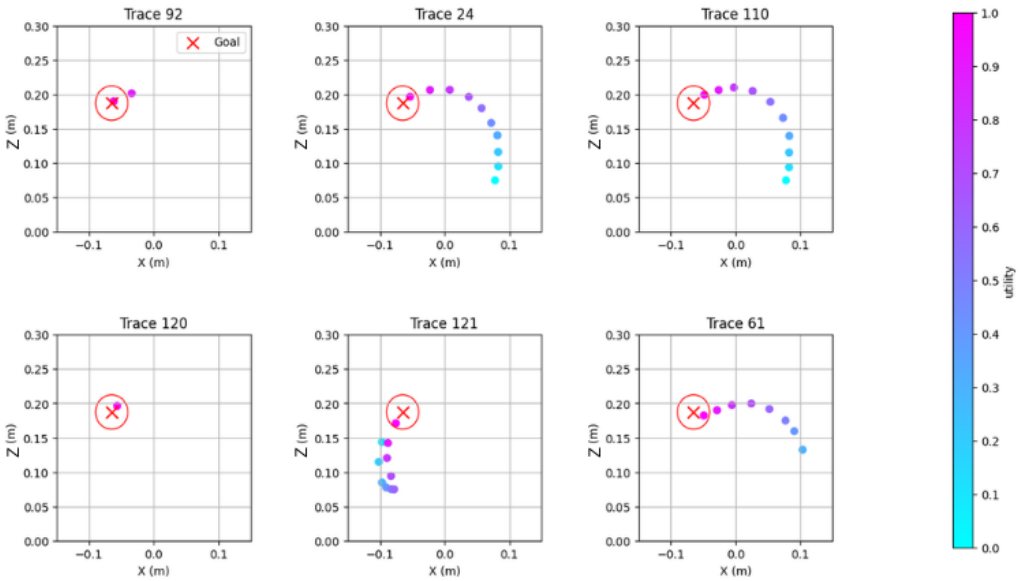



**Figura 4.22:** Muestra de trazas de entrenamiento y sus zonas de utilidad esperada. Elaboración propia.

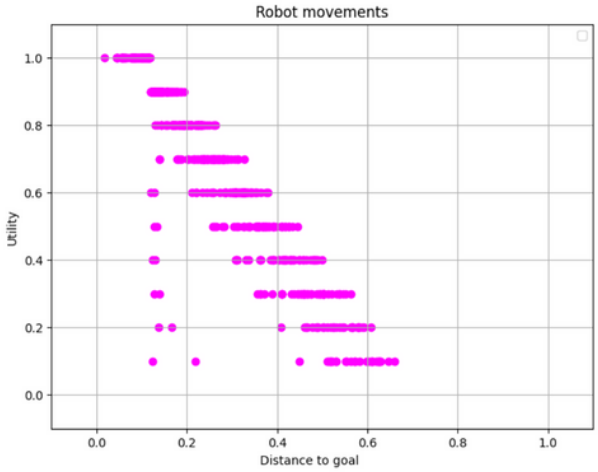



**Figura 4.23:** Trazas de entrenamiento con distancias normalizadas. Elaboración propia.

Estos datos aún tienen outliers, que no se le asignaron correctamente los valores de utilidad, y se analizan en la siguiente sección.

### 4.4.3 Preparación de los datos de entrenamiento

Primero se identifica la cantidad de datos que hay por valor de utilidad esperada, estos se muestran en la tabla 4.7. Cada categoría de utilidad tiene registros de distancia asociados, lo cual es un resultado favorable ya que ninguna categoría carece de datos.

Por otro lado en la figura 4.24 se analizan los datos outlier que se mencionaron anteriormente. En las gráficas de la utilidad de 0.1, 0.2, 0.3, 0.4, 0.5 y 0.6, es donde los datos

**Tabla 4.7:** Cantidad de datos de entrenamiento presentes por cada valor de utilidad

| Valor de utilidad | Cantidad de datos |
|---|---|
| 1.0 | 88 |
| 0.9 | 77 |
| 0.8 | 71 |
| 0.7 | 64 |
| 0.6 | 56 |
| 0.5 | 49 |
| 0.4 | 41 |
| 0.3 | 41 |
| 0.2 | 31 |
| 0.1 | 28 |

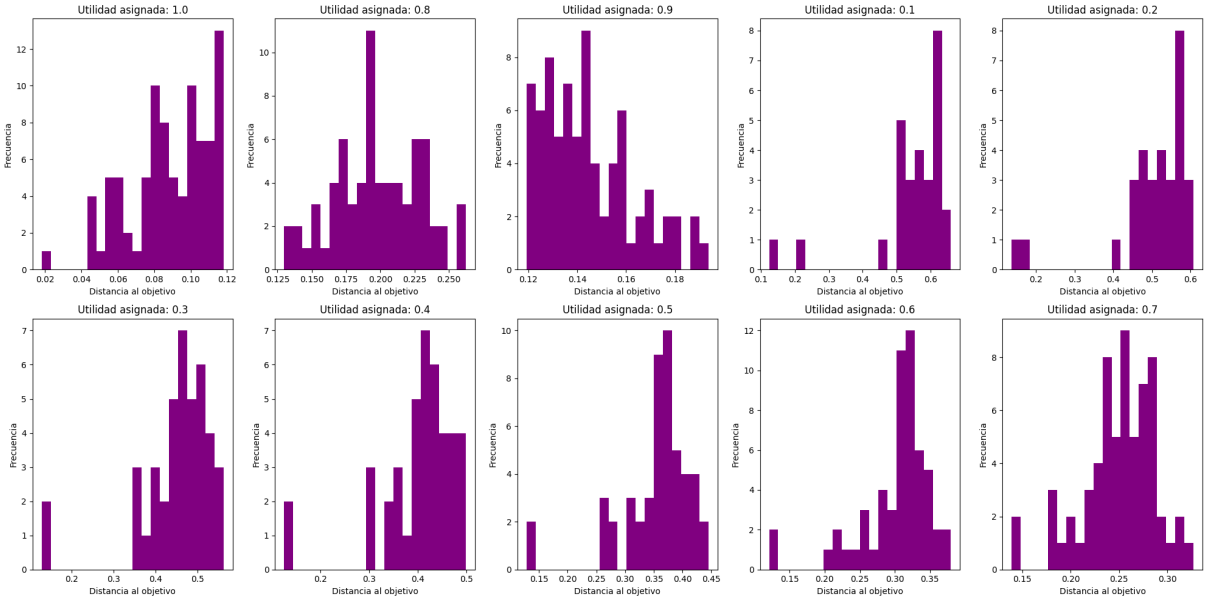



**Figura 4.24:** Histogramas de los datos de entrenamiento. Elaboración propia.

outliers son más prominentes. Sin embargo, en cada gráfica se muestra que solo hay 2 datos outlier máximo. Esto quiere decir que en el peor de los casos esto representa un 7.4% del total de la categoría de 0.1, la cual no es tan importante que sea tan precisa porque representa la posición más lejana al objetivo. Es importante que el modelo de utilidad prediga con mayor precisión entre más cerca se encuentre el robot del objetivo, entre más lejos es importante que simplemente sirva como indicador para llevar al robot a posiciones cada vez más cercanas al objetivo. Los valores de utilidad de 0.7, 0.8, 0.9 tienen menos datos outlier lo cual es favorable. Sin embargo se decidió eliminarlos estos datos outliers para evitar sesgos.

Otra cosa importante es que se obtuvieron únicamente 88 trazas para el entrenamiento porque se cayó el experimento en el servidor (múltiples veces), entonces se tuvo que trabajar con lo que había disponible.

**Tabla 4.8:** Hiperparámetros del modelo de utilidad

| | |
|---|---|
| Función de pérdida | Huber |
| Función de activación | Sigmoide |
| Función de activación de la capa de salida | Lineal |
| Número de capas ocultas | 2 |
| Neuronas de la primer capa | 3 |
| Neuronas de la segunda capa | 3 |
| Optimizador | Adam |
| Learning Rate | 0.01 |
| Epochs | 100 |

### 4.4.4 Entrenamiento de la red neuronal

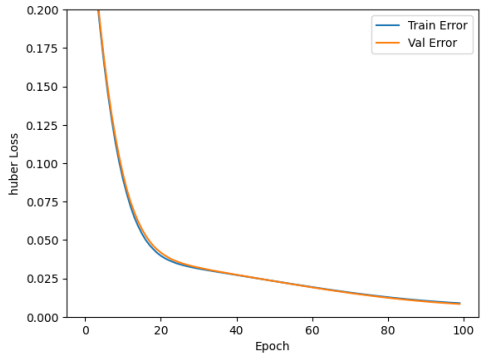
Como se había mencionado anteriormente ese era un problema de regresión, porque dado un valor de distancia se debía predecir un valor de utilidad entre cero y uno. Para este caso se utilizó como base la red neuronal entrenada en los experimentos iniciales. Pero, se decidió disminuir la cantidad de capas ocultas para la primera iteración. Por lo que los hiper parámetros se encuentran en la tabla 4.8.

Del entrenamiento se obtuvieron los resultados mostrados en las figuras 4.25. En la función de pérdida se ve como la perdida bajó progresivamente hasta llegar a un valor de 1%. Por otro lado, para la generación de la gráfica de predicciones se generó un array de datos de distancias entre cero y uno con incrementos de 0.001, esto con el fin de probar todas las posibles distancias que podría percibir el robot y probar la predicción del modelo de utilidad. En la gráfica de los valores de predicción del modelo se muestra como este mantiene una curva que aumenta de manera improporcional a la distancia, es decir que en cuanto disminuye la distancia al objetivo la utilidad aumenta, y justo este es el comportamiento que se esperaba del modelo de utilidad.

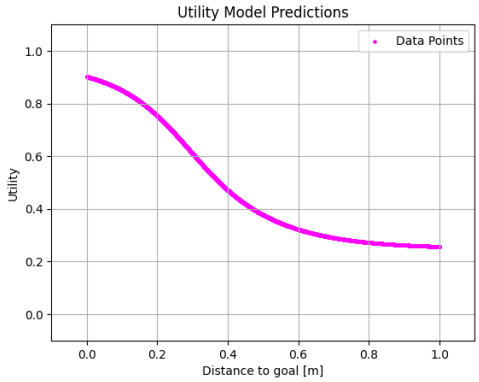
En consecuencia, la primera ronda de pruebas de hiperparámetros arrojó resultados muy buenos. Aunque se exploraron configuraciones adicionales, estas no mejoraron los resultados obtenidos, los resultados se encuentran en el anexo B. Por lo tanto, se decidió adoptar esta primera configuración dada su eficacia demostrada.

## 4.5 Implementación del proceso deliberativo de toma de decisiones

Una vez entrenado el modelo de utilidad se procedió a aprovechar el recurso e implementar la primer versión del proceso deliberativo de toma de decisiones. Para implementarlo se utilizaron las primeras versiones del modelo de mundo de las morfologías Modular02 y Modular03 del compañero Carlos Jara.




**(a)** Función de pérdida del modelo de utilidad




**(b)** Valores de predicción del modelo de utilidad

**Figura 4.25:** Resultados del modelo de utilidad entrenado. Elaboración propia.

Al implementar el modelo de utilidad y en el modelo deliberativo fue una manera de probar su funcionamiento, y coherencia en el proyecto, más que únicamente quedarse con el resultado de las predicciones demostrado en la sección anterior. Además, implementar en ambas morfologías representó la ventaja de probar y corregir errores en el código para que este pudiera adaptarse a múltiples morfologías.

Para esta implementación lo esencial es que el robot logre tomar la decisión de qué acción es la que lo va a llevar a acercarse a su objetivo y alcanzarlo. Cabe destacar que el modelo de mundo fue entrenado para predecir estados futuros a partir de acciones candidatas en un rango de $[-7.5, 7.5]$.

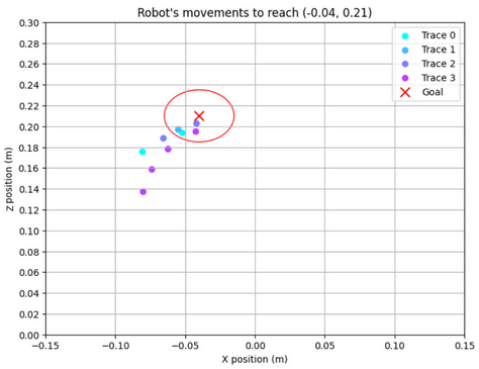
Cabe destacar que estas implementaciones se realizaron previas a las validaciones oficiales que esas se explican en la sección de Resultados de Análisis 5. De este procedimiento se destacó la integración de los componentes de la arquitectura e-MDB para su utilización con cada una de las morfologías establecidas.

### 4.5.1 Implementación en morfología Modular02

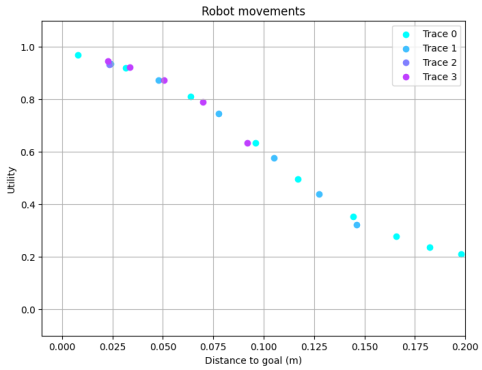
En esta implementación se programaron tres objetivos aleatorios y que el robot alcanzara este objetivo tres veces diferentes, para probar que funcionase con posiciones iniciales aleatorias. En la figura 4.26 se muestra como los 3 experimentos funcionaron correctamente el robot logró tomar la decisión de qué acción aplicar para acercarse poco a poco al objetivo. En estos experimentos se almacenaron valores extras como: la posición en coordenadas del objetivo aleatorio y coordenadas de las trazas del robot, para realizar estas gráficas.

### 4.5.2 Implementación en la morfología Modular02b

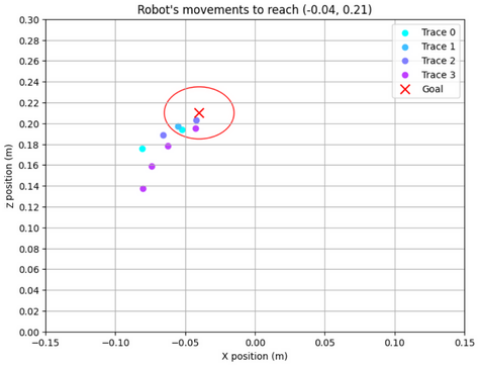
Es importante mencionar que esta morfología se implementó únicamente en simulación, utilizando el método anterior. Esta morfología se implementó para que alcanzase un ob-



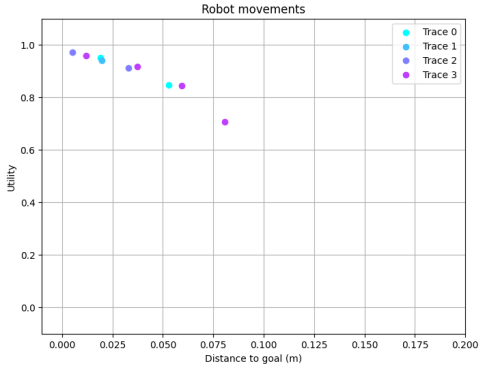

**(a)** Traza del primer objetivo aleatorio



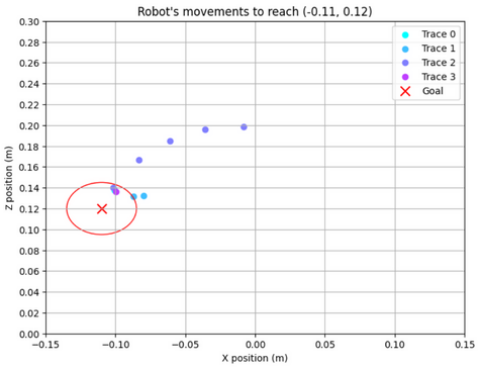

**(b)** Utilidades del primer objetivo aleatorio



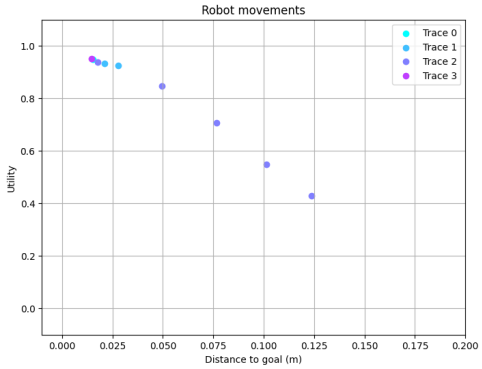

**(c)** Traza del segundo objetivo aleatorio




**(d)** Utilidades del segundo objetivo aleatorio




**(e)** Traza del tercer objetivo aleatorio




**(f)** Utilidades del tercer objetivo aleatorio

**Figura 4.26:** Resultados de implementar el modelo deliberativo en la morfología Modular02. Elaboración propia.

jetivo en el espacio de tres dimensiones. Sin embargo, el modelo de mundo proporcionado para esta prueba tenía muy poco conocimiento del espacio de estados entonces a pesar de que el modelo de utilidad sí predecía qué acción lo llevaría a acercarse al objetivo, se le complicaba moverse por estados desconocidos para los que no se había entrenado previamente.

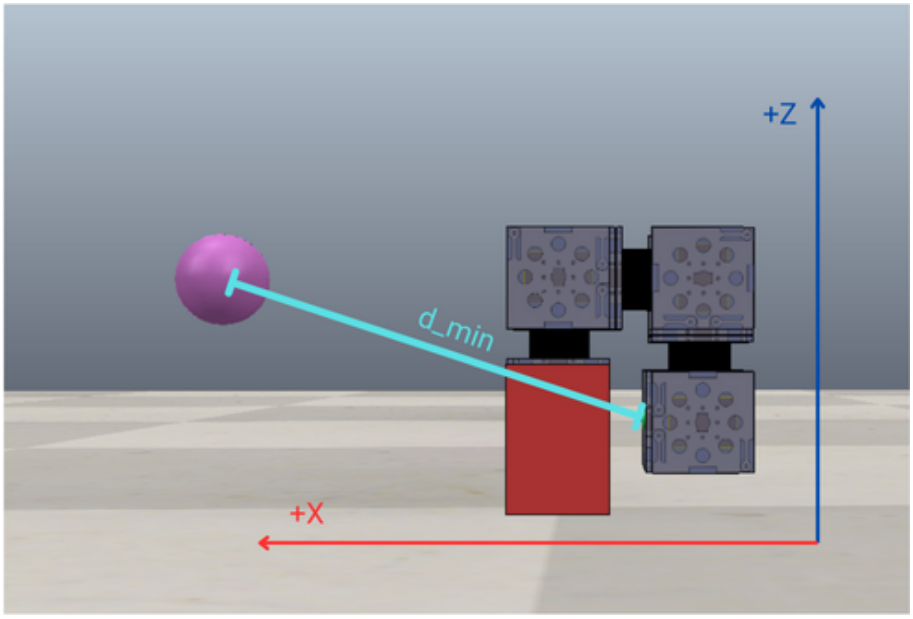



**Figura 4.27:** Morfología Modular03 atascada. Elaboración propia.

### 4.5.3 Implementación en morfología Modular03

Para esta morfología se utilizó la misma propuesta de la implementación anterior. Sin embargo los resultados no fueron buenos, ya que algunos estados sugerían una distancia menor al objetivo, pero las trayectorias resultantes eran impracticables. Este inconveniente se debió a configuraciones que provocan que el robot quede varado debido a la interferencia con su propia estructura, lo que refleja las deficiencias del robot en diferenciar entre rutas factibles y aquellas susceptibles a situaciones de estancamiento operativo. Un ejemplo se muestra en la figura 4.27, donde el robot se queda atascado en un estado de distancia mínima y para salir este debe desenrrollarse, lo cual implica alejarse del objetivo por un rato.

Además se encontraron partes en donde el robot se quedaba atascado porque no reconocía bien el espacio en el que se encontraba. Entonces para esto se propusieron dos soluciones, una de las soluciones era por parte del compañero Carlos Jara que proponía mejorar el modelo de mundo para evitar confusiones en el robot. Sin embargo por mi parte, la solución que se planteó se enfocó más en resolver estos atascos que podría tener el robot por razones como un modelo de mundo mal diseñado, para darle una motivación al robot para seguir explorando y moviéndose a través del espacio para poder salir de estos atascos.

## 4.6 Índice de Frustración

Se denominó *Frustración* a las instancias en las que el robot permanecía estancado, repitiendo acciones ineficaces que no facilitaban su liberación del punto de atasco. Esta terminología refleja la analogía de una sensación de frustración, similar a la experiencia

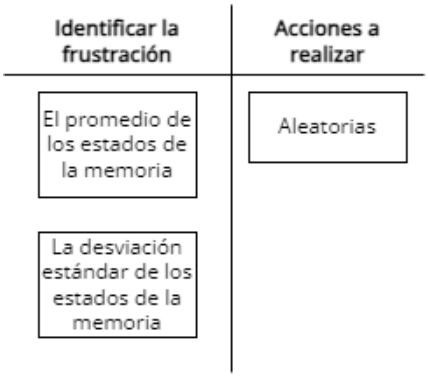



**Figura 4.28:** Diagrama de combinación de conceptos para la frustración. Elaboración propia.

humana de no lograr avanzar a pesar de los repetidos intentos, quedando atrapado en un ciclo constante de esfuerzos sin éxito.

Por lo tanto se debía diseñar una solución para la frustración, que implicaba crear un algoritmo para que el robot identificase que estaba frustrado y a partir de ahí aplicar alguna acción o acciones que lo llevasen a otro punto en el espacio de estados. Había que tomar en cuenta que para esta solución no se podía confiar el modelo de mundo porque se intentaba que este fuera una solución posibles problemas en el modelo de mundo.

Entonces, en la tabla 4.28 se muestra la generación de conceptos para la solución a estos problemas.

Para la selección del concepto se basó más que nada en la parte de identificar la frustración:

1. Inicialmente se sugirió calcular la media de los últimos estados visitados por el robot; sin embargo, esto implicaría que el promedio variaría continuamente, dificultando la definición de un umbral específico para determinar cuando el robot se encuentra en estado de frustración.

2. Posteriormente, se consideró la desviación estándar de los estados, dado que esta métrica, al utilizar los valores normalizados de la distancia al objetivo, facilita la identificación de períodos prolongados de inmovilidad. Esto se debe a que una desviación estándar reducida indica que las últimas distancias registradas son similares entre sí, lo cual sirve como criterio para establecer un umbral que determine la frustración del robot.

Por lo tanto, como la frustración aumenta cuando la desviación estándar de los datos se disminuye, se propuso identificar el índice de frustración con la ecuación 4.3. La ecuación implementa la extracción de la raíz cuadrada de la desviación estándar con el objetivo de amplificar su impacto. Dado que la desviación estándar de las distancias normalizadas es siempre menor que uno, la naturaleza de la función raíz cuadrada incrementa la relevancia de estos valores. Posteriormente, se resta este resultado de uno, de manera que valores de frustración más próximos a uno indican un mayor nivel de frustración.

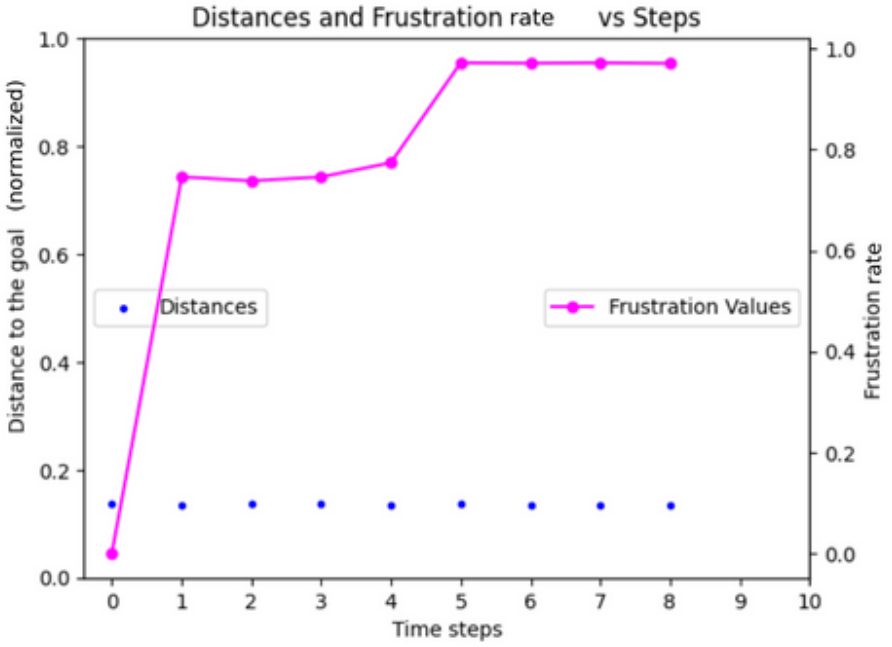



**Figura 4.29:** Gráfica de análisis del índice de frustración. Elaboración propia.

$$F = 1 - \sqrt{\sigma} \tag{4.3}$$

Ahora, la memoria utilizada se eligió para ser de los últimos cinco estados del robot, ya que cinco estados ofrecen una visión comprensiva sin ser excesivamente específicos, además de que si se agregan más estados entonces el robot tardaría más en darse cuenta de que está frustrado. Por otro lado, si se elegía menos entonces el robot podría identificar una frustración falsa porque la desviación entre los datos es muy baja al ser estados continuos.

Para determinar el umbral se llevó al robot a una posición de frustración conocida, la de la figura 4.27 en donde todos los motores del robot se encuentran en 90° y esa es la distancia más corta al objetivo. Entonces, en la gráfica de la figura 4.29 el robot no se muestra frustrado hasta que alcanza el quinto paso y se llena la memoria, y es ahí donde se realiza un pico y el índice de frustración sube a más de 0.9. Antes de alcanzar este valor, el índice de frustración se mantuvo menor a 0.8, entonces se determinó el umbral para definir que el robot está frustrado a partir de que el índice alcanza el valor de 0.8.

En la figura 4.30 se muestra dónde se incluye el análisis de la frustración dentro del algoritmo. Esta se añade después de que se aplica la acción en el proceso delibertivo de toma de decisiones. No se puede calcular la frustración antes de aplicar la acción porque el modelo de mundo podría haber predicho mal el estado futuro de manera que no fuese un estado que generase frustración en el robot, mientras que al aplicar la acción se puede determinar realmente si el robot entró en frustración o no. Cabe destacar que el el índice de frustración sirve como agente motivador para dejar que el robot salga de sus atascos.

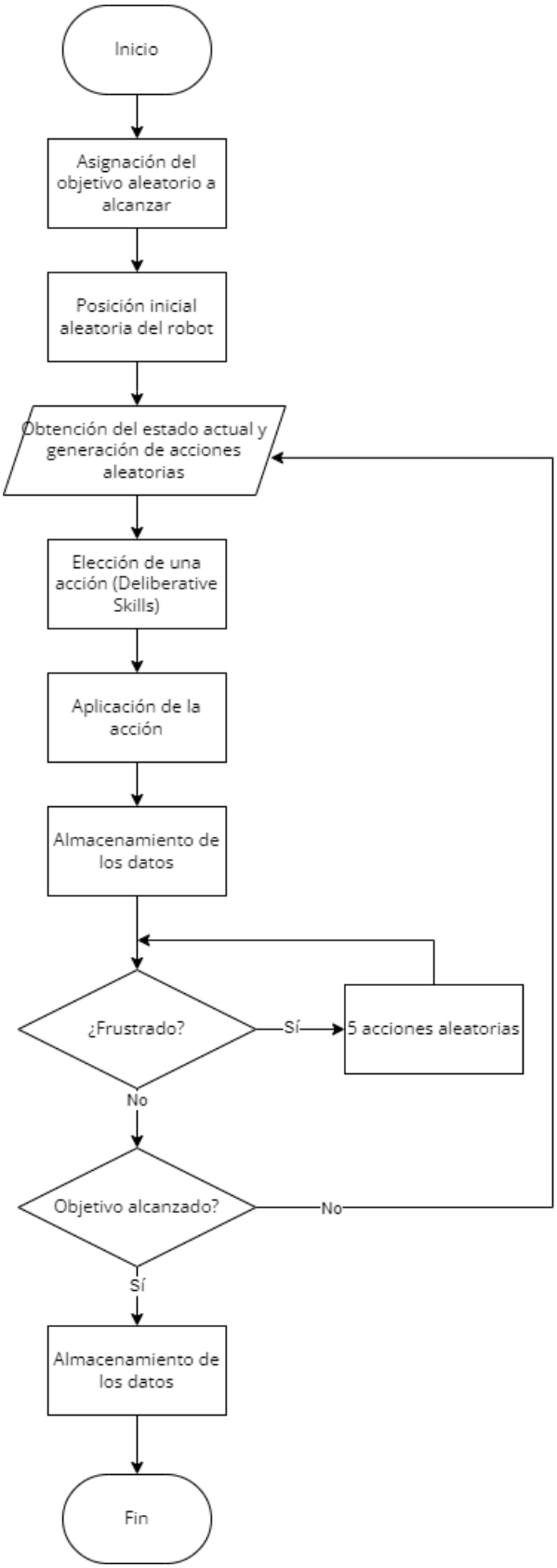



**Figura 4.30:** Diagrama de flujo de la implementación completa de la solución. Elaboración propia.

**Figura 4.31:** Cámaras OptiTrack en el laboratorio. Elaboración propia.

## 4.7 Sistema de captura de movimiento: OptiTrack

Previamente se definió que el sistema a utilizar para implementar el robot real sería el sistema de de captura de movimiento OptiTrack. Para el proceso de implementación del sistema de captura de movimiento por medio de las cámaras Optitrack Prime13, se utilizó el software de Motive Tracker 3.0.0. Para obtener la comunicación entre el Motive y las cámaras se realizó una comunicación por medio de ethernet. En el laboratorio se cuenta cn cuatro cámaras previamente posicionadas y una mesa de trabajo con vista desde todas las cámaras. En la figura 4.31 se muestra el posicionamiento de tres de ellas en el laboratorio. La configuración de las cámaras fue realizada por un estudiante de la UDC como parte de su Trabajo de Fin de Grado y se utilizó como guía para implementar el sistema [31].

### 4.7.1 Comunicación por ROS

Para enviar los datos de la posición del robot obtenida por el sistema de captura de movimiento al código principal se utilizó el protocolo de comunicación ROS (Robot Operation System) [32], y se implementó a través del modelo publicador/subscriptor, donde se definió un nodo encargado de recopilar las coordenadas de la posición del robot, como se muestra en el diagrama de la figura 4.32.

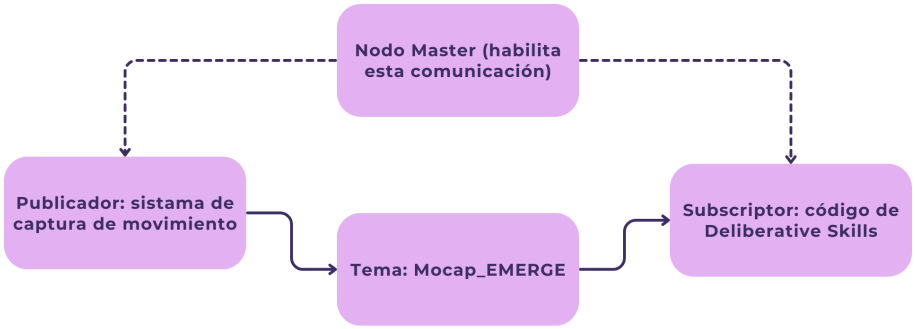



**Figura 4.32:** Diagrama de comunicación por ROS. Elaboración propia.

**Figura 4.33:** Marcador retrorreflectivo para el sistema OptiTrack. Elaboración propia.

### 4.7.2 Calibración del sistema

El proceso de calibración se realizó a partir de la guía oficial de calibración del sistema OptiTrack [33]. Se siguieron los siguientes pasos generales:

1. Preparación del volumen de captura: las optitracks utilizan marcadores retrorreflectantes, de la figura 4.33 que se le colocan a los objetos para identificar su movimiento, por lo que se liberó el espacio de materiales similares para evitar errores en las mediciones.

2. En Motive se aplicó una máscara para ignorar las reflexiones existentes en la vista de las cámaras.

3. Se recogieron muestras de calibración mediante el proceso de *wanding*, como se muestra en la figura 4.34.

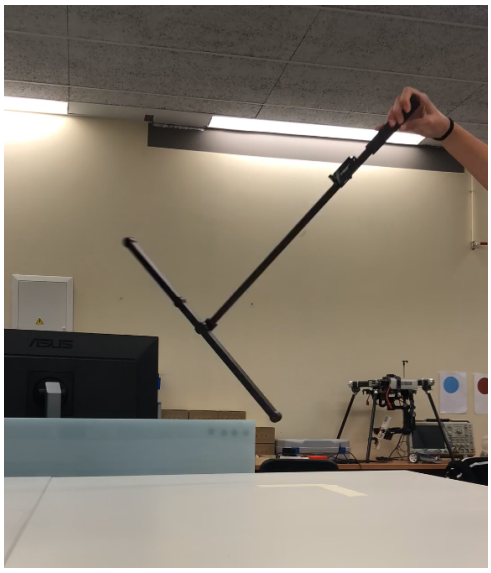
4. Se revisan los resultados del proceso de *wanding* en Motive y se aplica la calibración. Para este caso el error resultante en las mediciones. En la figura 4.35 se muestra como el resultado fue excepcional.

5. Finalmente se establece el plano de referencia con la cuadrícula de referencia, como se muestra en la figura 4.36.

### 4.7.3 Método de implementación del objetivo

En el laboratorio se disponía de una estructura semejante a una grúa que se encontraba en desuso. Esta estructura había sido utilizada en experimentos previos de robótica,

**(a)** CW-500 Calibration Wand



**(b)** Movimiento del *wand*

**Figura 4.34:** Proceso de *Wanding*. Elaboración propia.

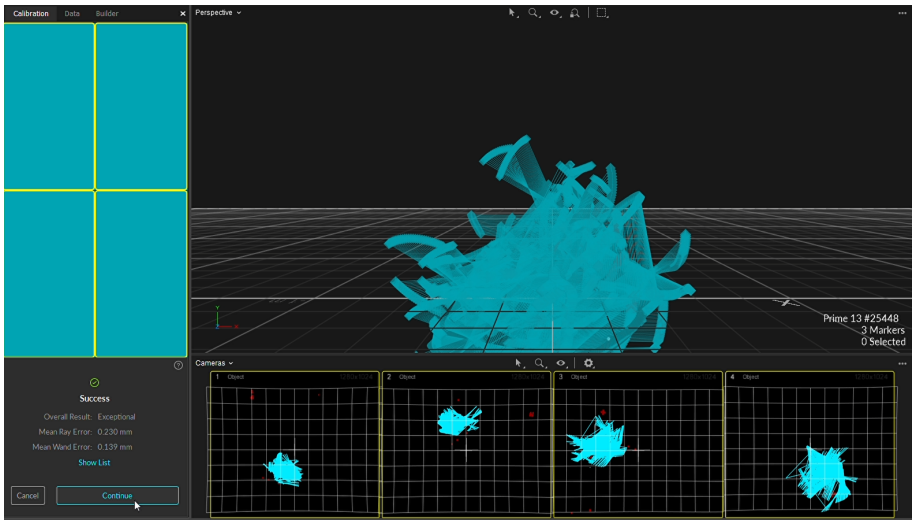
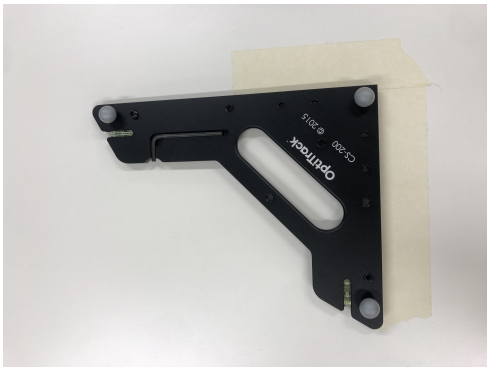



**Figura 4.35:** Resultado de la calibración con el *wand*. Elaboración propia.

funcionando como un medio para posicionar los objetivos del robot en el espacio, ya que permitía colgar una pelota, como se muestra en la figura 4.37. Esta estructura tenía un gancho arriba que permitía colocar cuerdas de diferentes longitudes para colocar el objetivo a diferentes alturas.

### 4.7.4 Creación de los *Rigid Bodies*

Para determinar el movimiento de un objeto primero se debe llenar este de markers y crear un *Rigid Body* asociado en Motive. En la documentación se recomienda que no están ubicados de forma simétrica [33]. Entonces en las figuras 4.38 y 4.39 se muestra como se definió cada objeto en Motive, así como la asignación de los markers respectivamente.




**(a)** CS-200 Calibration Square

**(b)** Ubicando la cuadrícula del plano

**Figura 4.36:** Establecimiento del origen y el plano del suelo. Elaboración propia.

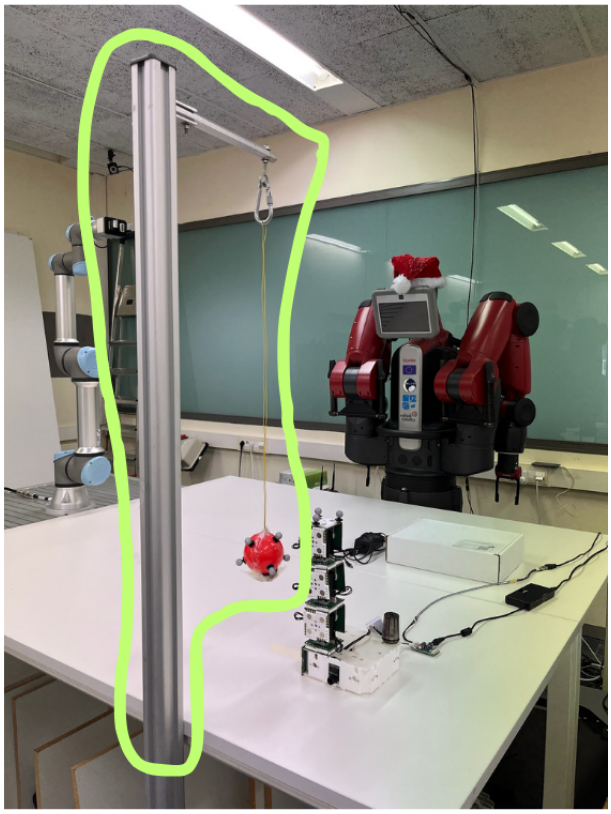
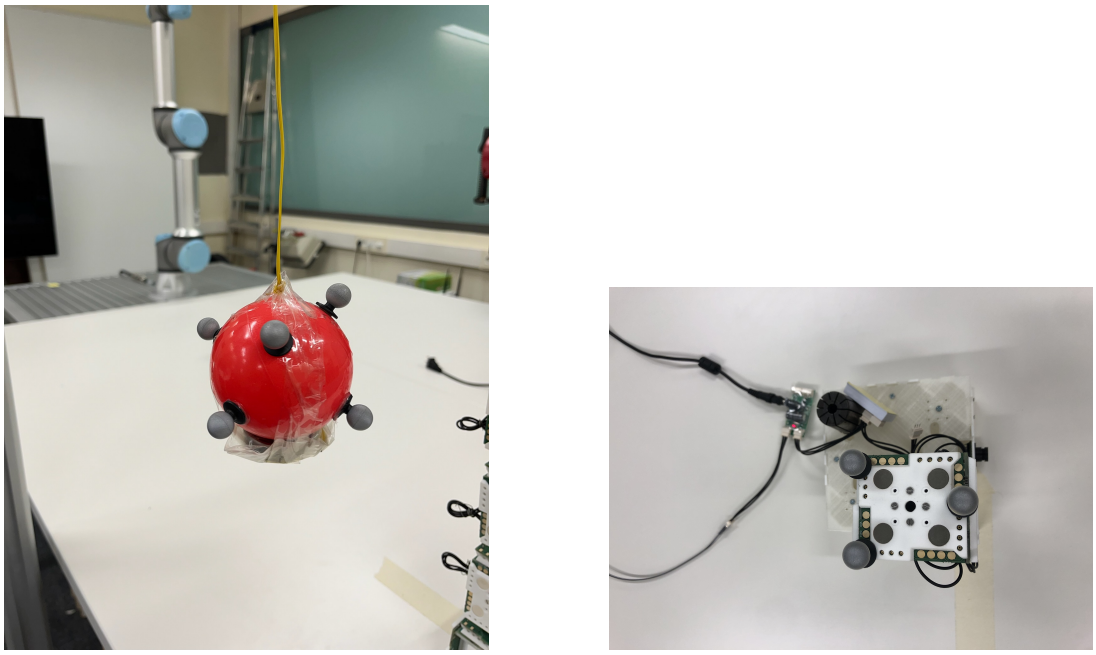


**Figura 4.37:** Estructura tipo grúa para colgar la pelota objetivo. Elaboración propia.

Los markers en el EMERGE se ubicaron en la cara superior del módulo superior porque ahí se tomaban las mediciones durante la simulación. También es importante mencionar que los markers se debían posicionar a al menos 3.5 centímetros de distancia ente ellos porque si no, el sistema de captura de movimiento no los identificaba.



**(a)** Markers en la pelota objetivo **(b)** Markers en el EMERGE

**Figura 4.38:** Ubicación de los marcadores en cada objeto. Elaboración propia.

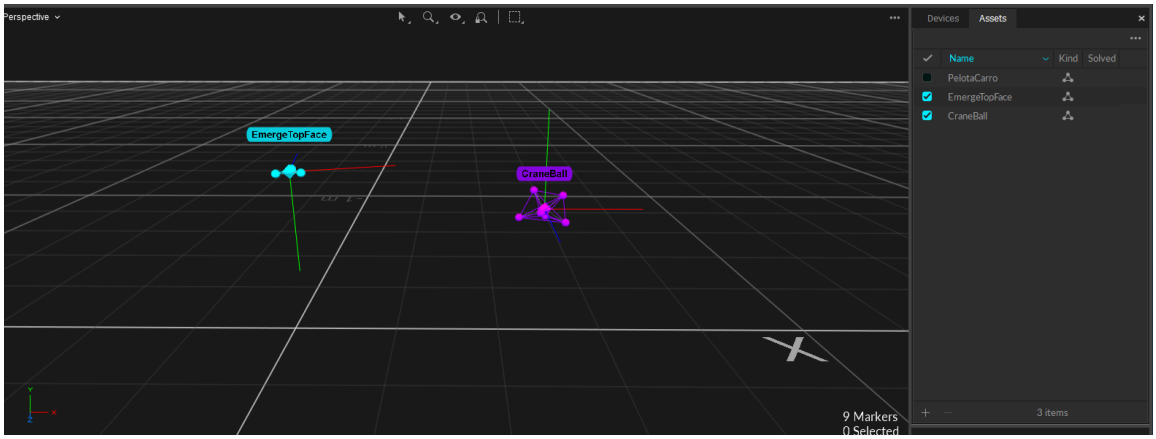



**Figura 4.39:** Establecimiento de los *Rigid Bodies* en Motive: en color azul la cara superior del EMERGE y en color morado el objetivo. Elaboración propia.

### 4.7.5 Calibración complementaria

Se llevó a cabo una calibración adicional para ajustar el offset causado por la altura de los marcadores (las coordenadas en simulación se median con respecto al centro de la cara superior de EMERGE) como se puede ver en la figura 4.40. Dicha calibración se incorporó en el código, permitiendo así compensar este offset, de manera que primero se ubicaba al robot en una posición completamente vertical y a partir de ahí se calculaba el offset con respecto a la simulación y se realizaba la compensación. Adicionalmente, debido a que la base sufrió modificaciones y su altura varió, se utilizó este cambio para equilibrar también el incremento en la altura.

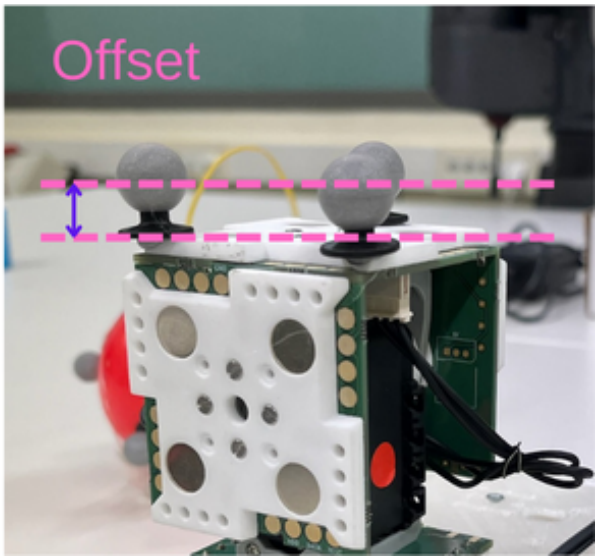



**Figura 4.40:** Offset generado por los markers. Elaboración propia.

Finalmente, para sostener la base del robot a la mesa de trabajo, esta se pegó con silicona caliente y se probó que se podía despegar con facilidad. Y de esta manera quedó listo la configuración para las verificaciones.

Además, en el Anexo A se encuentra un enlace a videos de la calibración tanto del software Motive, como de la calibración del espacio físico. En este mismo anexo, se encuentra un manual de usuario detallado con todos los pasos a seguir para la implementación de la comunicación por ROS, y la puesta en marcha del sistema de captura de movimiento para identificar el movimiento del robot.

# Capítulo 5

# Resultados y análisis

Este capítulo detalla los resultados obtenidos y el análisis correspondiente de las pruebas realizadas tanto en entornos de simulación como en implementaciones con el robot real. La validación en simulación permitió una exploración exhaustiva de los parámetros y comportamientos en un entorno controlado, donde se pudieron simular diversas morfologías y escenarios operativos sin los riesgos o limitaciones físicas inherentes a las pruebas del mundo real.

Posteriormente, las configuraciones y estrategias óptimas identificadas en la simulación se transfirieron al robot EMERGE, lo que proporcionó una plataforma tangible para evaluar la aplicabilidad de los modelos cognitivos y de utilidad en situaciones prácticas. Las pruebas en el robot real no solo confirmaron la efectividad del enfoque propuesto, sino que también revelaron desafíos adicionales y oportunidades de aprendizaje que solo emergen en la interacción con el mundo físico.

Para la verificación del funcionamiento de la propuesta de solución se realizaron dos pruebas de validación, una para verificar el funcionamiento de la motivación intrínseca de frustración y la segunda consistió en una implementación en el robot físico.

## 5.1 Herramientas de medición

Antes de llevar a cabo las pruebas de funcionamiento del sistema en los entornos simulado y real, se verificó la precisión y confiabilidad de los sensores utilizados. En ambos contextos, se evaluaron dos tipos de sensores: el motor del robot, que determina su posición angular, y los sensores encargados de medir las coordenadas dentro del espacio de trabajo.

### 5.1.1 Mediciones del simulador

#### Motores

Para obtener la la precisión en las mediciones del motor del robot en simulación (revolute joint), se utilizó un único módulo para realizar las pruebas de las mediciones de sus posiciones angulares.

En este experimento se optó por realizar 30 réplicas para cada medición de posición angular, seleccionando este tamaño de muestra debido a que se espera la mínima variación en los resultados de las mediciones. Un mayor número de réplicas contribuye a una mejor demostración de la precisión del sistema [34]

En este experimento se hizo un barrido a través de los ángulos del motor, desde -90° hasta 90° y viceversa, realizando incrementos de 1°, hasta generar la cantidad requerida de muestras. Además los valores se midieron en radianes para mantener la coherencia en el proyecto, porque el modelo de mundo fue entrenado en esta unidad.

En la figura 5.1 se puede observar que la precisión de las mediciones aumenta cuando el módulo se encuentra en una posición vertical, la desviación es de aproximadamente 1°. Mientras que cuando se encuentra al rededor de los 85° el valor la desviación de los datos de medición aumenta a aproximadamente 4.58°, lo cual es una desviación bastante alta.

Por otro lado, las mediciones poseen las siguientes características estadísticas:

- Cantidad de mediciones: 5430
- Promedio del error: $-1.751037 \times 10^{-7}$ radianes
- Desviación estándar del error: $5.665724 \times 10^{-2}$ radianes

El error de la herramienta de medición de la posición angular es bastante bajo, esto significa que es bastante confiable. Sin embargo, se le agregó un código extra al simulador, para que llevase al módulo a la posición deseada y leyese la posición, si esta no estaba a $\pm 0.5^o$ de lo deseado entonces que volviera a llevar el módulo a esa posición, hasta que este margen de error en la medición se alcanzara.

#### Sistema de coordenadas

Para probar las mediciones del sistema de coordenadas de CoppeliaSim, se generaron 30 posiciones aleatorias para un *force sensor* que es el objeto utilizado para obtener la medición de la posición del robot. Estas posiciones se repitieron para 30 repeticiones, es decir, que el *force sensor* visitó cada posición 30 veces. En cada posición se realizó una medición y resultó que tanto el error como la desviación estándar en las mediciones era nula, es decir, que cada posición en la que se encontraba el *force sensor* se leía correctamente.

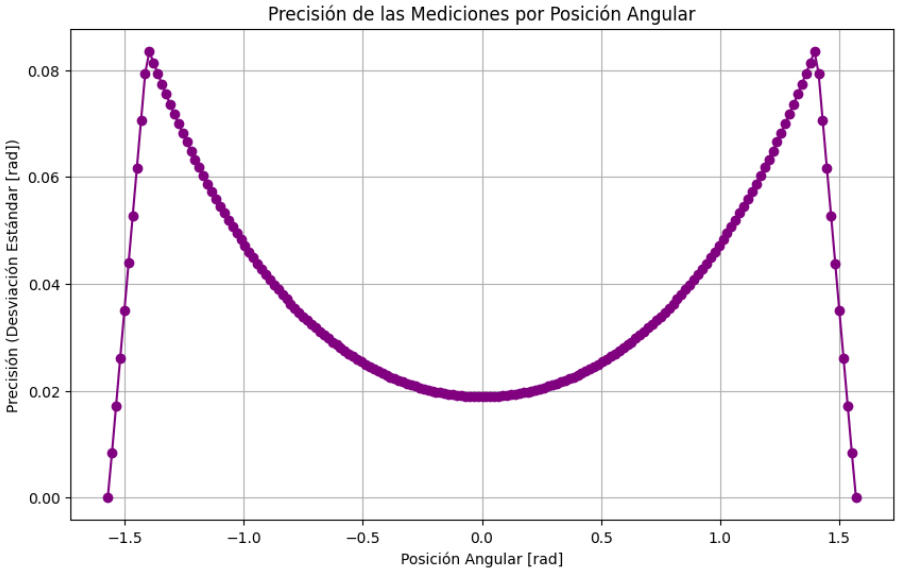



**Figura 5.1:** Precisión en las mediciones de los revolute joint de CoppeliaSim. Elaboración propia.

### 5.1.2 Mediciones en el robot real

A continuación se realizaron mediciones sobre el motor del robot real. Cabe mencionar que montar los módulos y echarlos a andar fue parte de la producción del proyecto de Carlos Jara, también se obtuvo una librería en Python para el control del robot.

#### Servomotores

En este experimento se hizo un barrido a través de los ángulos del motor, desde -90° hasta 90° y viceversa, realizando incrementos de 2°, hasta generar la cantidad requerida de muestras.

Los resultados de calcular la precisión con que realiza las mediciones el sensor del servomotor del robot se muestran en la figura 5.2. Las tablas con las muestras obtenidas se encuentran en el Apéndice A.

De estos resultados se obtiene que el sensor de la posición angular posee una mayor precisión cuando son ángulos más cercanos a cero o la posición vertical, mientras que más inclinación posea el módulo, más se pierde la precisión. Por ejemplo, en los extremos se alcanza una desviación estándar en los datos de medición de hasta 0.2864°, mientras que en posiciones más cercanas a cero, o la posición vertical, el las mediciones se vuelven más precisas, incluso llegan a tener total precisión. Por lo tanto, se establece que la precisión de las mediciones es buena, porque variar máximo 0.2864° no representa un riesgo para que el robot alcance sus objetivos.

Por otro lado, el error de las mediciones tiene las siguientes características estadísticas:

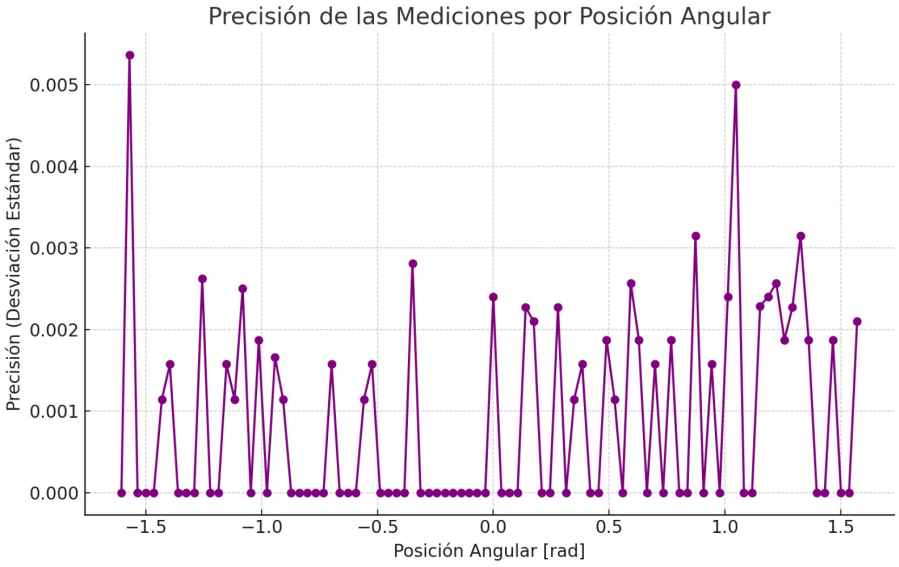



**Figura 5.2:** Precisión en las mediciones de los servomotores AX-12. Elaboración propia.

- Cantidad de mediciones: 1822
- Promedio del error: 0.0095 radianes
- Desviación estándar del error: 0.0044 radianes

Esto se determinó obteniendo la diferencia entre el valor estimado por el sensor y el valor real, luego calculó el promedio de estas diferencias. Realmente el error promedio en las mediciones es bastante bajo, 0.0095 radianes corresponde a 0.5° promedio de error en las mediciones. Como este no representaba una necesidad para el cliente la exactitud ni la precisión en el sistema entonces esto no suma importancia en este aspecto. Sin embargo, esto podía haber afectado al momento de obtener predicciones de los modelos entrenados. Sin embargo, las acciones en el robot real se determinaron de entre -90° y 90° entonces 0.5° es un valor de error promedio que se puede tolerar.

## 5.2 Validación de la solución en simulación

### 5.2.1 Descripción de la prueba

Para comprobar el funcionamiento de la solución diseñada para que el robot alcanzara sus objetivos, se realizaron 5 experimentos en donde se ubicó la pelota objetivo en una posición fija y se generaron 30 posiciones iniciales aleatorias para que el robot pudiese alcanzar la pelota desde allí. En cada experimento se repitieron las posiciones con el objetivo de obtener un promedio de la cantidad de pasos que le tomaría al robot alcanzar el objetivo desde esa posición. Además, en cada experimento se utilizó el proceso deliberativo de toma de decisiones, una versión sin el índice de frustración y una versión con el índice de

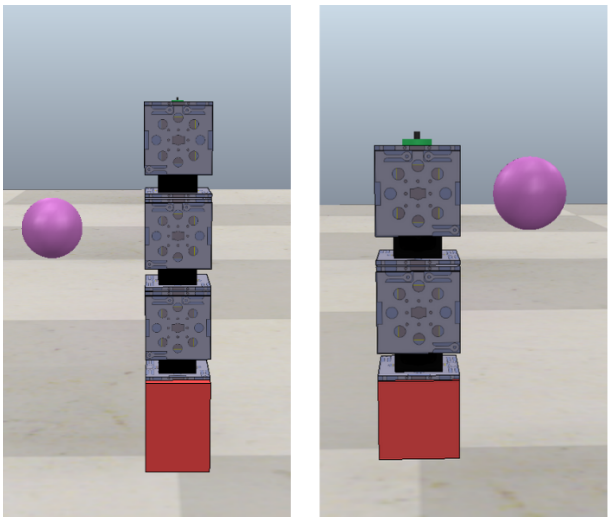


**Figura 5.3:** Morfologías utilizadas para las pruebas en simulación y sus objetivos (pelota rosada). Elaboración propia.

frustración, para demostrar el funcionamiento de esta solución. Para salir de atascos, una vez identificada la frustración se aplicaban 5 acciones aleatorias.

Durante estas pruebas se definió un atasco cuando el robot realizaba más de 100 pasos sin alcanzar el objetivo, porque en estos casos ya no era evidente que el robot no tenía idea como alcanzar el objetivo.

En la figura 5.3 se encuentran las dos morfologías utilizadas para la prueba, a la izquierda se encuentra la morfología Modular03 y a la derecha la morfología Modular02. Con esto se demostraría que el robot puede alcanzar sus objetivos en dominios cambiantes (posiciones iniciales y morfologías variantes).

### 5.2.2 Resultados de la simulación

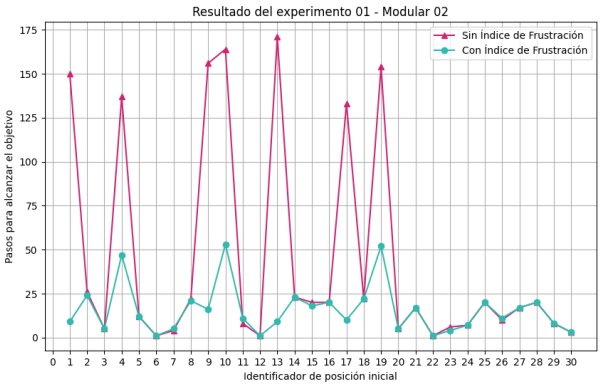
En las figuras 5.4 y 5.5 se muestran los resultados de cada experimento. En los resultados sin el índice de frustración se muestra como muchas veces el robot realizaba más de 100 pasos sin alcanzar el objetivo, porque se atascaba en posiciones "arrolladas" como se explicaba en la sección 4.6, o porque el modelo de mundo de Carlos Jara no tenía la capacidad para predecir correctamente los estados futuros a partir de las posiciones en que se encontraba el robot. Esto generaba que en promedio para la morfología Modular02 (línea color rosada) no se alcanzara el objetivo en 5 de 30 ocasiones, en algunos casos hasta 7 veces de esas 30 iteraciones no se alcazaba el objetivo (en promedio se alcanzabn alrededor de 83% de los objetivos). Lo mismo sucedía con la morfología Modular 03 (línea color amarilla).

En la figura 5.4 si se analiza la posición inicial número 10, sin el índice de frustración (línea color rosada), el robot en el experimento 01 se atascó, no logró llegar al objetivo. Sin embargo en los experimentos 02, 03, 04 y 05, sí los logró alcanzar en un aproximado

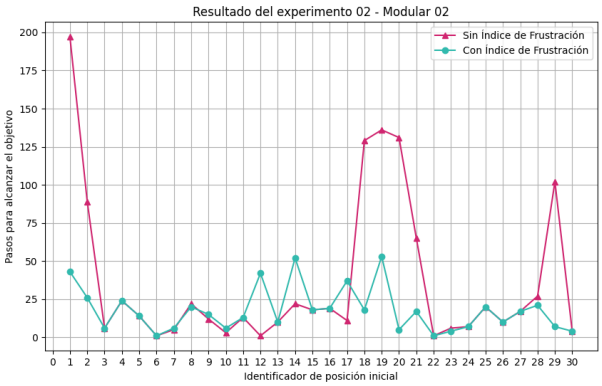
de 10 pasos. Aquí se muestra como sin el índice de frustración el proceso de toma de decisiones deliberativo no era suficientemente robusto para permitirle al robot alcanzar los objetivos de manera constante, más cuando no se puede fiar por completo del modelo de mundo. Mientras que cuando se analiza la mismo posición inicial 10, pero con el índice de frustración incluido (línea color cian), en el experimento 01 se muestra como este llegó a un nivel de atasco, pero logró salir de él y alcanzar el objetivo en aproximadamente 50 pasos.

Lo mismo sucedía con la posición inicial número 19 de la morfología Modular02 que en todos los experimentos generó que el robot se atascara, pero al identificar la frustración y aplicar las acciones aleatorias se logró que el robot saliera de los atascos mucho antes que sin esta solución. También en los resultados de la morfología Modular03, figura 5.5, el índice de frustración mostró ayudar al robot a salir de atascos dadas las siguientes posiciones iniciales: 3, 11, 12, 15, 18 y 23. De esta manera se muestra que el índice de frustración representó una mejora al proceso deliberativo de toma de decisiones diseñado, para ayudarle al robot a identificar cuando se encontraba en atascos y así aplicar acciones para salir del atasco. Finalmente en las figuras 5.6 (línea color cian) y 5.7 (línea color morada) el robot alcanzó el objetivo a partir de todas sus posiciones iniciales para los casos en que se identificó la frustración.

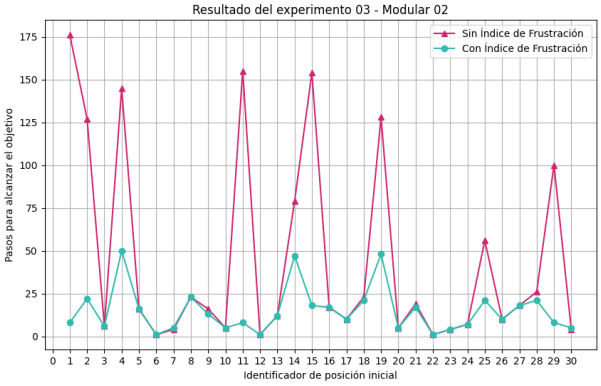
No obstante, en algunos casos al aplicar las acciones aleatorias sí provocaba que el robot realizara varios pasos extra antes de alcanzar el objetivo, en algunos casos llegó a realizar aproximadamente 50 pasos, porque cada vez que entraba en frustración se aplicaban 5 acciones mínimo. Entonces se plantea disminuir la cantidad de pasos una vez identificada la frustración, como parte de investigaciones a futuro.



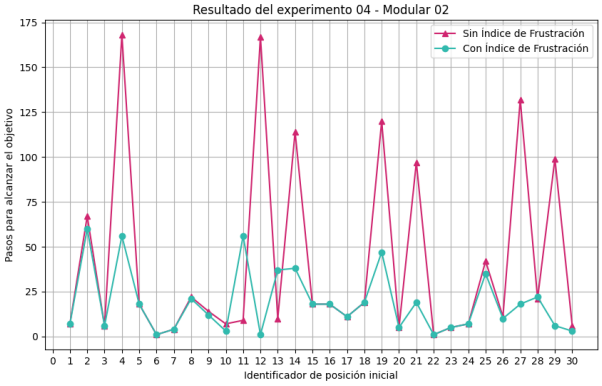

**(a)** Experimento 01 - Modular02



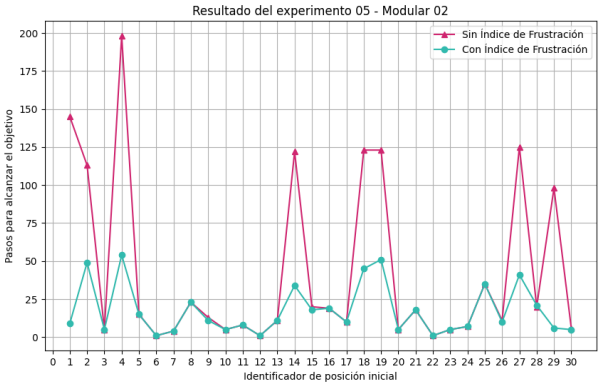

**(b)** Experimento 02 - Modular02




**(c)** Experimento 03 - Modular02

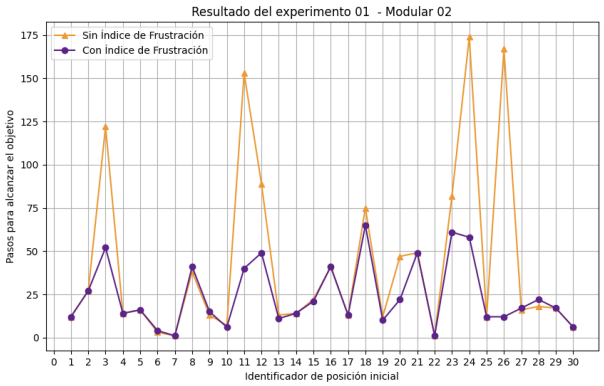



**(d)** Experimento 04 - Modular02

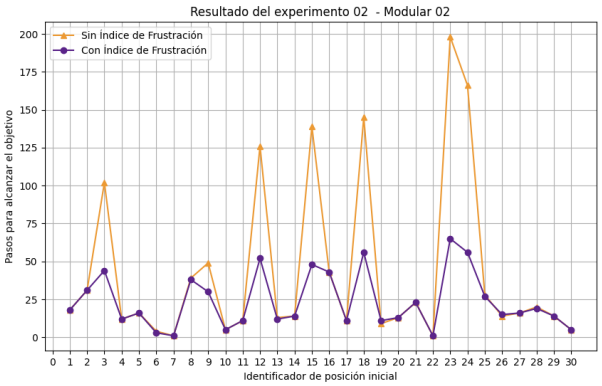



**(e)** Experimento 05 - Modular02

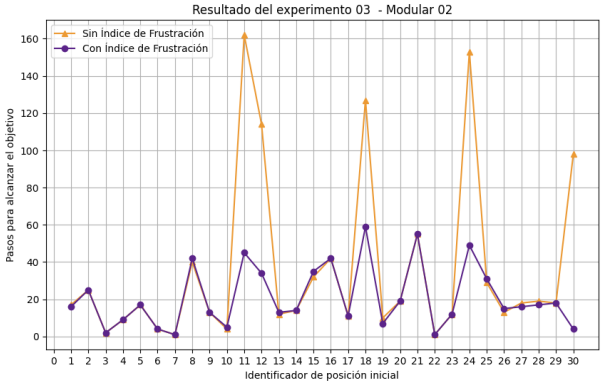
**Figura 5.4:** Resultados de cada experimento de la prueba de validación en simulación para la morfología Modular02. Elaboración propia.



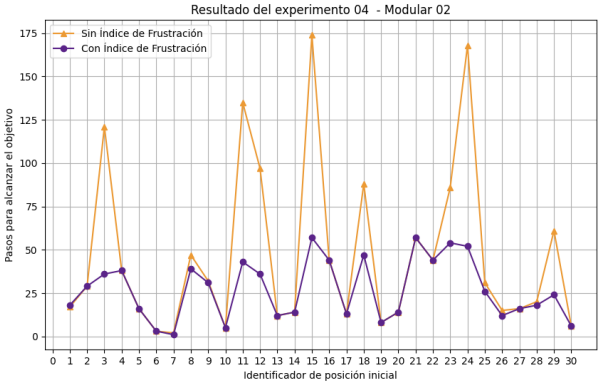

**(a)** Experimento 01 - Modular03



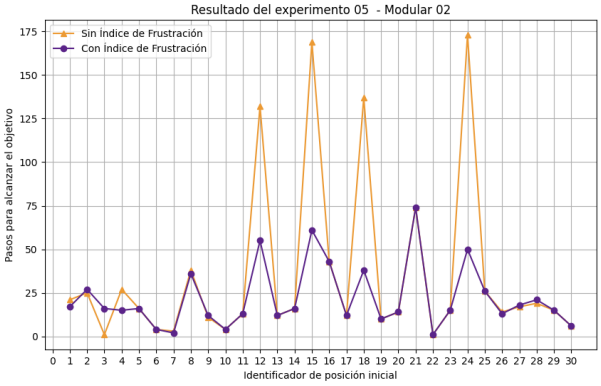

**(b)** Experimento 02 - Modular03




**(c)** Experimento 03 - Modular03




**(d)** Experimento 04 - Modular03




**(e)** Experimento 05 - Modular03

**Figura 5.5:** Resultados de cada experimento de la prueba de validación en simulación para la morfología Modular03. Elaboración propia.

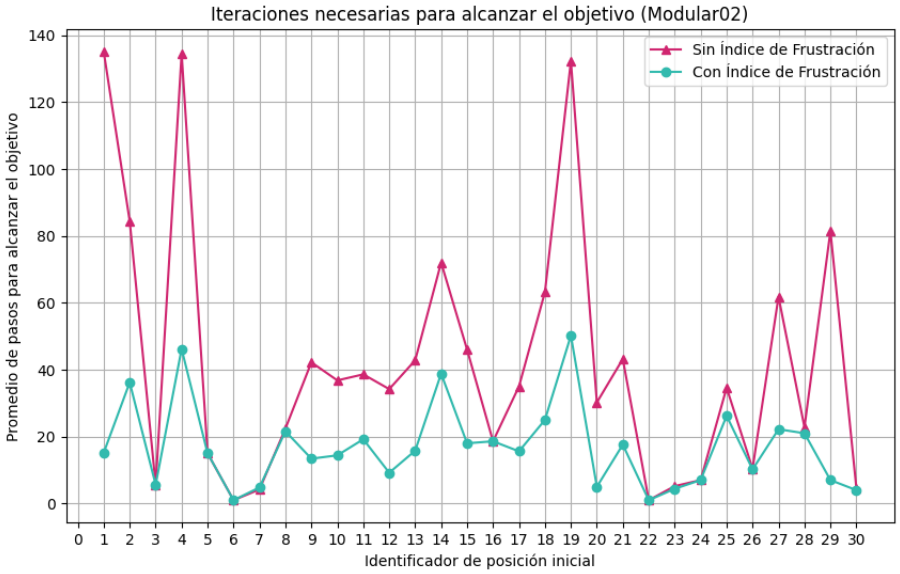



**Figura 5.6:** Resultados promedio de las iteraciones necesarias para alcanzar el objetivo para la morfología Modular02 en simulación. Elaboración propia.

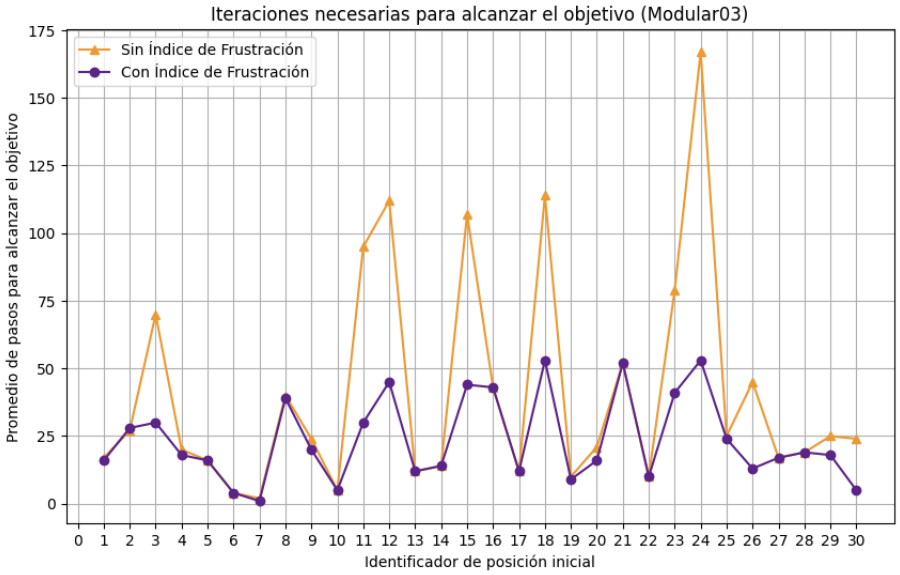



**Figura 5.7:** Resultados promedio de las iteraciones necesarias para alcanzar el objetivo para la morfología Modular03 en simulación. Elaboración propia.

## 5.3 Validación de la solución en el robot real

Como método de validación del modelo de utilidad entrenado para el proceso deliberativo de toma de decisiones, se implementó directamente en el robot real, con una versión mejorada del modelo de mundo de el compañero Carlos Jara que fue entrenado con un rango de acciones aleatorias posibles entre $[\frac{-\pi}{2}, \frac{\pi}{2}]$, es decir entre -90° y 90°. Por cuestiones de disponibilidad de tiempo se implementó directamente en la morfología Modular03, ya que era la que estaba disponible. No obstante, esta configuración presentaba mayores obstáculos; por lo tanto, al llevarla a cabo se evidenciaría su implementación en la modalidad más compleja, alcanzando un resultado satisfactorio. En la figura 5.9 se encuentra la configuración de la implementación real del robot y el objetivo.

### 5.3.1 Descripción de la prueba

Durante esta prueba también se realizaron 30 experimentos, donde se le asignaron posiciones iniciales aleatorias al robot, y se ubicó en el espacio la pelota objetivo para demostrar el funcionamiento del algoritmo. De este experimento se obtuvieron datos del modelo de utilidad y la cantidad de pasos que realizó el robot para alcanzar sus objetivos.

### 5.3.2 Resultados del modelo de utilidad

En la figura 5.8 se muestran resultados del modelo de utilidad al ser implementado en el robot real. La tendencia general indica que, en la mayoría de las trazas, la utilidad aumenta con cada paso. Esto es visible a través de las líneas ascendentes en la gráfica. Esto quiere decir que el modelo de utilidad fue capaz de siempre predecir cuál acción sería más útil que la anterior para alcanzar el objetivo del robot.

En las trazas 10 y 12 se obtuvo un comportamiento menos ascendente, este podría ser porque las acciones aleatorias generadas por el algoritmo no eran suficientes para alcanzar el objetivo, sin embargo el modelo de utilidad permitió siempre elegir la acción más útil. En total, por las 31 trazas, el robot realizó 89 acciones, de estas acciones 6 de ellas no fueron más útiles que la anterior. Por lo tant, se determina que el robot obtuvo un 93% de acciones útiles.

También se puede determinar esto a partir de la cantidad de pasos que se realizaron por cada traza. En muchas ocasiones el robot logró alcanzar el objetivo muy rápido en dos pasos o menos y el máximo de pasos realizados fue de cinco.

También, el robot logró alcanzar todos los objetivos que se le asignaron, eso quiere decir que tanto el código desarrollado, así como la solución diseñada funcionaron como se esperaba.

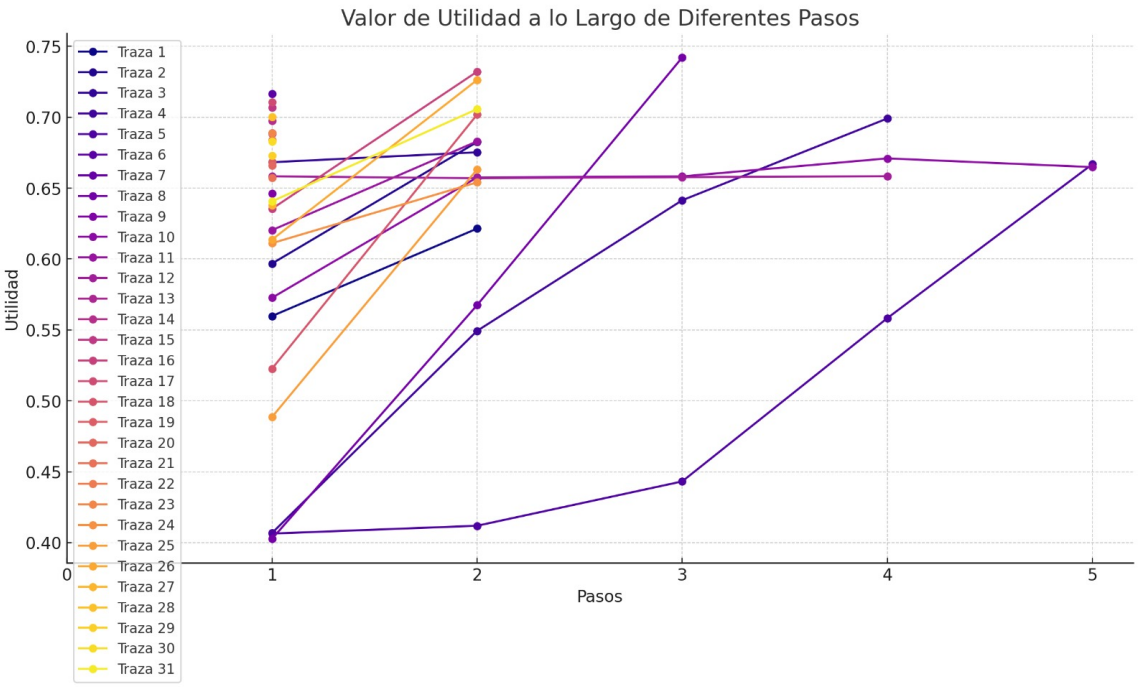



**Figura 5.8:** Utilidades obtenidas por el robot en sus trazas. Elaboración propia.

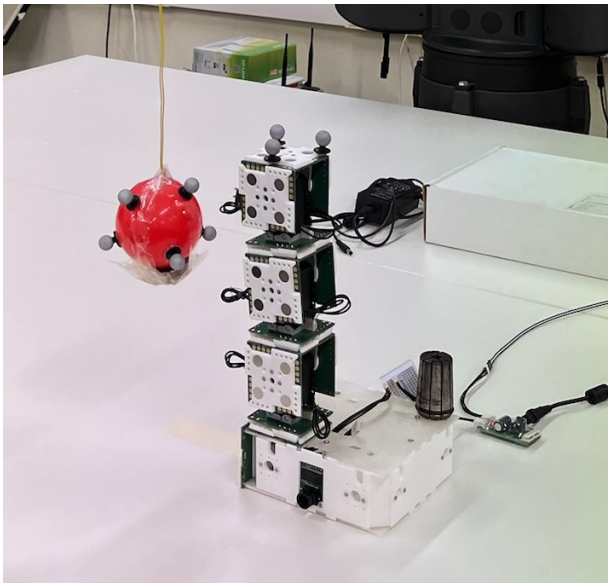


**Figura 5.9:** Implementación en el robot real. Elaboración propia.

## 5.4 Análisis económico

A pesar de que el enfoque del proyecto se orientaba más hacia la investigación que hacia la comercialización, se consideró fundamental realizar un análisis económico. Este proceso no solo permitió comprender con precisión los recursos que se habían invertido y la manera en que estos se habían administrado, sino que también destacó el valor del tiempo ahorrado

durante la investigación y los beneficios tangibles que este trabajo aportó al Grupo de Investigación en Ingeniería (GII).

El proyecto requirió una inversión significativa, tanto en términos de recursos materiales como humanos. Estos aportes se detallaron y se sintetizaron en la tabla 5.1 para proporcionar una referencia rápida y clara.

- **Ordenador:** Para este proyecto se utilizó un ordenador portátil Dell G3, donde se implementaron los programas en Python y las simulaciones en CoppeliaSim. Ambos softwares son gratuitos por lo que no involucraron un costo extra. Además de los sistemas operativos Linux y de comunicación ROS que tampoco involucraron un gasto para este proyecto.
- **Horas de investigación:** Como ingeniera dedicada a la investigación en la UDC, el pago por trabajar una jornada laboral de medio tiempo (20 horas a la semana) correspondía a €1000 mensuales y se trabajó un total de 4 meses en este proyecto. Sin embargo, es importante destacar que el trabajo fue no remunerado, por lo que, no representó un gasto adicional para el GII.
- **Módulos robóticos e imprevistos** Lo invertido por el GII fue de €350 por cada módulo robótio, añadiendo un 10% de presupuesto extra para imprevistos, como impresiones 3D o materiales faltantes (tornillos, arandelas, washers, etc) y herramientas. Como se implementaron 3 módulos y una base, este precio se multiplica por 4 unidades.
- **Sistema de captura de movimiento:** Este recurso ya había sido obtenido por el GII e instalado previamente. Entre los costos analizados se destacó la compra de las 4 cámaras OptiTrack, el software de captura de movimiento Motive, las herramientas de calibración (Wand y Square), así como los materiales utilizados para las conexiones del sistema, los marcadores para hacer la captura del movimiento de los objetos, y los objetos utilizados como objetivos para el robot (el sistema de pelota voladora de Martín).

### 5.4.1 Beneficios para el GII

Como se destacó anteriormente, el enfoque de este proyecto no fue la comercialización, sino la investigación, lo que resultó en un considerable ahorro de horas de investigación para el GII, ya que, el GII se ahorró 20 horas semanales por 16 semanas. Este esfuerzo también ayudó a la innovación, especialmente dentro del contexto del proyecto europeo Pillar Robots, con el cual están colaborando activamente en diversas investigaciones sobre motivaciones intrínsecas y LOLA [2].

Por otro lado, respecto al sistema de captura de movimiento OptiTrack, que previamente se encontraba en desuso, su puesta en marcha y la creación de un manual de uso significaron un eficaz aprovechamiento de este activo. Esto representa una ventaja adicional, ya

**Tabla 5.1:** Recursos asociados al desarrollo del proyecto

| **Recurso** | **Descripción** | **Costo** | **Cantidad** | **Subtotal** |
|---|---|---|---|---|
| Ordenador | Laptop Dell G3 | € 1,000.00 | 1 | € 1,000.00 |
| Ingeniera | Horas investigadora (al mes) | € 1,000.00 | 4 | € 4,000.00 |
| Módulo robótico | Módulos incluyendo la base | € 350.00 | 4 | € 1,400.00 |
| Imprevistos | Impresiones 3D, materiales extra, etc | € 35.00 | 4 | € 140.00 |
| Cámaras OptiTrack | Prime13 | € 2,029.00 | 4 | € 8,116.00 |
| Software de las OptiTracks | Motive Tracker 3.0.0 | € 923.00 | 1 | € 923.00 |
| Calibration Square | CS-200 | € 156.00 | 1 | € 156.00 |
| Calibration Wand | CMW-500 kit | € 323.00 | 1 | € 323.00 |
| Materiales para el SCM | Cables, marcadores, objetivos, etc | € 100.00 | 1 | € 100.00 |
| **Total** | | | | **€ 16,158.00** |

que el sistema ahora puede emplearse en la captura de movimiento de objetos en futuras investigaciones de robótica o ingeniería naval que lleve a cabo el GII.

# Capítulo 6

# Conclusiones

En este proyecto se se presentó una solución innovadora en el ámbito del aprendizaje abierto a lo largo de la vida del robot de los robots modulares, orientada a los módulos robóticos EMERGE. Se demostró un aumento en la autonomía de este robot al integrar componentes de la arquitectura cognitiva e-MDB que proporcionaron un procedimiento estructurado para que el robot pudiese llegar a tomar sus propias decisiones. A través de la motivación intrínseca de novedad el robot logró descubrir sus posibilidades de actuación, mediante una exploración en eficiente del espacio. Mientras que por medio del modelo de utilidad, el robot logró aprender de los conocimientos adquiridos en la exploración para así ponerlos en práctica mediante el proceso deliberativo de toma de decisiones en morfologías de dos y tres módulos, con objetivos cambiantes, y en un dominio físico así como en el dominio simulado. De esta manera se cumplió con el objetivo general del proyecto.

En cuanto al primer objetivo específico del proyecto, se logró obtener las percepciones y acciones en base a los sensores y los actuadores disponibles para cada morfología dada. En la morfología más simple que consistía en un único módulo robótico se probó el primer sensor y el actuador: el servomotor AX-12, que funcionó como sensor para obtener el estado actual de la posición angular de cada módulo, así como actuador para aplicar las acciones, que corresponden a incrementos en esa posición angular (ver ecuación 4.1). Mientras que para más módulos se aprovechó también la posibilidad de medir las coordenadas (ver ecuación 4.2) de la posición en la que se encontraba el robot tanto en simulación (CoppeliaSim) como la realidad (sistema de captura de movimiento OptiTrack). La medición de las coordenadas en el espacio permitió obtener la distancia entre el robot y el objetivo.

Por otro lado, el robot logró descubrir sus posibilidades de actuación por medio de una exploración del espacio de estados basada en la motivación intrínseca de novedad. Esta exploración le permitió asociar sus posibilidades de actuación con sus percepciones de los estados visitados y generar un modelo de utilidad que le proporcionó la utilidad esperada de cada estado en el espacio de estados. El modelo entrenado obtuvo una pérdida muy baja del 1%, pero al probar su predicción con valores del espacio de estados, este predi-

jo correctamente la utilidad esperada para cada estado. Además se comprobó su buen funcionamiento porque le permitió al robot alcanzar un 93% de acciones útiles, aún realizando cambios en el dominio, como mover la pelota objetivo de posición. Cumpliendo así con el indicador del tercer objetivo específico de este proyecto, que requería obtener al menos 90% de acciones útiles.

El modelo de utilidad entrenado, junto con otras componentes de la arquitectura e-MDB, como el proceso deliberativo de toma de decisiones, le permitió al robot alcanzar únicamente un 83% de sus objetivos a nivel de simulación. Sin embargo al implementar la motivación intrínseca de frustración el robot logró alcanzar la totalidad de sus objetivos desde posiciones iniciales aleatorias. Demostrando así que el índice de frustración funcionó como un incentivo para evitar atascos del robot en el dominio y poder cumplir con el indicador del segundo objetivo específico que pedía que el robot alcanzase un 90% de los objetivos asignados en simulación.

No obstante, durante la verificación de funcionamiento en el robot real, este logró alcanzar también la totalidad de objetivos asignados de forma aleatoria. De manera que se validó el resultado del modelo de utilidad entrenado con datos de simulación, así como las acciones y percepciones obtenidas al inicio, de modo que se superó la brecha de simulación-realidad, alcanzando así el cuarto objetivo especifico de este proyecto.

Finalmente, los resultados obtenidos en este proyecto confirmaron la validez de la metodología propuesta, tanto en escenarios reales como simulados, abriendo camino hacia futuros avances en este ámbito. También, se recalca el beneficio que representó este proyecto para el GII al colaborar con investigaciones que permitieron la constante innovación en proyectos de robótica cognitiva. Así como el aprovechamiento de los recursos presentes en el laboratorio como el sistema de captura de movimiento OptiTrack.

## 6.1 Recomendaciones y trabajos futuros

En este proyecto se utilizaron dos sensores: el sistema de captura de movimiento OptiTrack y los servomotores AX-12. Sin embargo sería útil implementar más sensores que permitan obtener más información de la percepción del robot para evitar ambigüedades en las percepciones. Además, se aprovecharía el uso de más sensores para diseñar una nueva forma de salir de atascos cuando el robot se encuentra frustrado, y que le permita al robot salir del atasco más rápido.

Los trabajos a futuro pensados a partir de la realización de este proyecto se detallan a continuación:

- Continuar con el trabajo de investigación realizado, de manera que se aproveche la implementación del robot EMERGE para probar nuevas morfologías y nuevos modelos de mundo con la arquitectura cognitiva e-MDB.
- Publicación de artículos científicos relacionados a la integración del trabajo realizado

en este proyecto junto con el de Carlos. El primer artículo se envió a la conferencia: *International Conference on the Interplay between Natural and Artificial Computation* (IWINAC), y se planea enviar también un artículo a la conferencia llamada *Conference on Artificial Life* (ALIFE).

- La implementación de morfologías en más dimensiones. Esto sería un avance importante entre los siguientes pasos del proyecto, porque implica desarrollar nuevas estrategias para obtener datos de entrenamiento para el modelo de utilidad.

- Montaje de más módulos robóticos EMERGE para implementar morfologías con mayor complejidad.

- Añadir más sensores al sistema robótico para crear escenarios donde el robot pueda generar estrategias más robustas para salir de escenarios donde se encuentra frustrado.

- Diseñar nuevos objetivos para que el alcance que involucren más interacción como: una botonera, un final de carrera o un sensor de proximidad. Esto con el fin de evaluar nuevas habilidades como que el robot se acerque a objetos o que apriete botones.

# Bibliografía


[1] A. Romero Montero. *e-MDB: Unha arquitectura cognitiva para a aprendizaxe autónoma aberta de por vida en sistemas robóticos*. PhD thesis, Universidade da Coruña, 2022.

[2] Pillar robots [online]. 2022. URL: https://pillar-robots.eu/ [cited 19 de marzo de 2024].

[3] Maja J Matarić. *The Robotics Primer*. The MIT Press, 2007.

[4] A. Cangelosi and M. Asada. *Cognitive Robotics*. The MIT Press, 2022.

[5] Thomasnet. All about industrial robots [online]. 2023. URL: https://www.thomasnet.com/articles/automation-electronics/all-about-industrial-robots/ [cited 08 de febrero de 2024].

[6] J. A. Bagnell J. Kober and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotic Research*, vol. 32, no. 11, 2013.

[7] A. S. Polydoros and L. Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, Jun. 2017.

[8] Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M. Roijers, Benoît Girard, Nicolas Perrin, and Olivier Sigaud. Open-ended learning: A conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 12, 2018. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2018.00059, doi:10.3389/fnbot.2018.00059.

[9] C. L. Hull. *Principles of behavior: an introduction to behavior theory*. Appleton-Century, 1943.

[10] D. Vernon. *Artificial Cognitive Systems. A Primer*. The MIT Press, 2014.

[11] F. Bellas y R. J. Duro A. Romero. A perspective on lifelong open-ended learning autonomy for robotics through cognitive architecture. *Sensors*, vol. 23, no.3, 2023. doi:10.3390/s23031611.

[12] C. Lebiere y J.R. Anderson. A connectionist implementation of the act-r production system. In *In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, Boulder, CO, USA*, page 635–640, 1993. doi:10.1109/IJCNN48605.2020.9206931.

[13] A. Newell y P.S. Rosenbloom J.E. Laird. Soar: An architecture for general intelligence. *Artificial Intelligence*, vol. 33, no. 1, pp. 1-64, 1987.

[14] A. Faiña y D. Souto F. Bellas, R.J. Duro. Multilevel darwinist brain (mdb): Artificial evolution in a cognitive architecture for real robots. *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 340-354, Dec. 2013. doi:10.1109/TAMD.2010.2086453.

[15] G. Baldassarre y M. Mirolli V. G. Santucci. Grail: A goal-discovering robotic architecture for intrinsically-motivated learning. *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 3, pp. 214-231, Sept. 2016. doi:10.1109/TCDS.2016.2538961.

[16] B. Goertzel. Opencogprime: A cognitive synergy based architecture for artificial general intelligence. *IEEE International Conference on Cognitive Informatics*, Hong Kong, China, 2009, pp. 60-68. doi:10.1109/COGINF.2009.5250807.

[17] J. A. Starzyk and J. Graham. Mlecog: Motivated learning embodied cognitive architecture. *IEEE Systems Journal*, vol. 11, no. 3, pp. 1272-1283, Sept. 2017. doi:10.1109/JSYST.2015.2442995.

[18] G. Baldassarre y M. Mirolli. *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer-Verlag Berlin Heidelberg, 2013.

[19] Alejandro Romero, Francisco Bellas, Abraham Prieto, and Richard J. Duro. Developmental learning of value functions in a motivational system for cognitive robotics. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi:10.1109/IJCNN48605.2020.9206931.

[20] A.P. Engelbretch. *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition, 2007.

[21] Keras. Model training apis [online]. 2024. URL: https://keras.io/api/models/model_training_apis/ [cited 08 de febrero de 2024].

[22] Andrés Faíña Rodríguez-Vila. *Un sistema robótico modular heterogéneo para entornos dinámicos y no estructurados*. PhD thesis, 2011.

[23] Rodrigo Moreno and Andres Faiña. Emerge modular robot: A tool for fast deployment of evolved robots. *Frontiers in Robotics and AI*, 8, 2021. URL: https://www.frontiersin.org/articles/10.3389/frobt.2021.699814, doi:10.3389/frobt.2021.699814.

[24] Karl T. Ulrich and Steven D. Eppinger. *Product Design and Development*. McGraw-Hill Education, 5th edition, 2012.

[25] Coppelia Robotics. Accelerate your R&D with next-gen automation and robotics prototyping [online]. 2023. URL: https://www.coppeliarobotics.com/ [cited 08 de febrero de 2024].

[26] Coppelia Robotics. Zeromq remote api [online]. 2023. URL: https://manual.coppeliarobotics.com/en/zmqRemoteApiOverview.htm [cited 08 de febrero de 2024].

[27] BubbleRob tutorial. Coppeliasim user manual [online]. URL: https://manual.coppeliarobotics.com/en/bubbleRobTutorial.htm [cited 08 de febrero de 2024].

[28] Andres Faiña. Edhmor: Evolutionary designer of heterogeneous modular robots [online]. URL: https://bitbucket.org/afaina/edhmor/src/master/ [cited 08 de febrero de 2024].

[29] Robotis. AX-12A [online]. 2024. URL: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/ [cited 08 de febrero de 2024].

[30] Frank E. Harrell. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer, New York, 2 edition, 2015.

[31] Alejandro Chao. Implementación y puesta en marcha de un entorno robotizado para la realización de ensayos de interacción dinámica, 2022.

[32] y William D. Smart Morgan Quigley, Brian Gerkey. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, Inc, 2015.

[33] OptiTrack. Optitrack documentation [online]. 2024. URL: https://docs.optitrack.com/motive/calibration [cited 08 de febrero de 2024].

[34] A. C. Carrasco H. G. Pulido, R. de la Vara Salazar and M. O. Sánchez. *Análisis y diseño de experimentos*. McGraw-Hill/Interamericana, 2008.

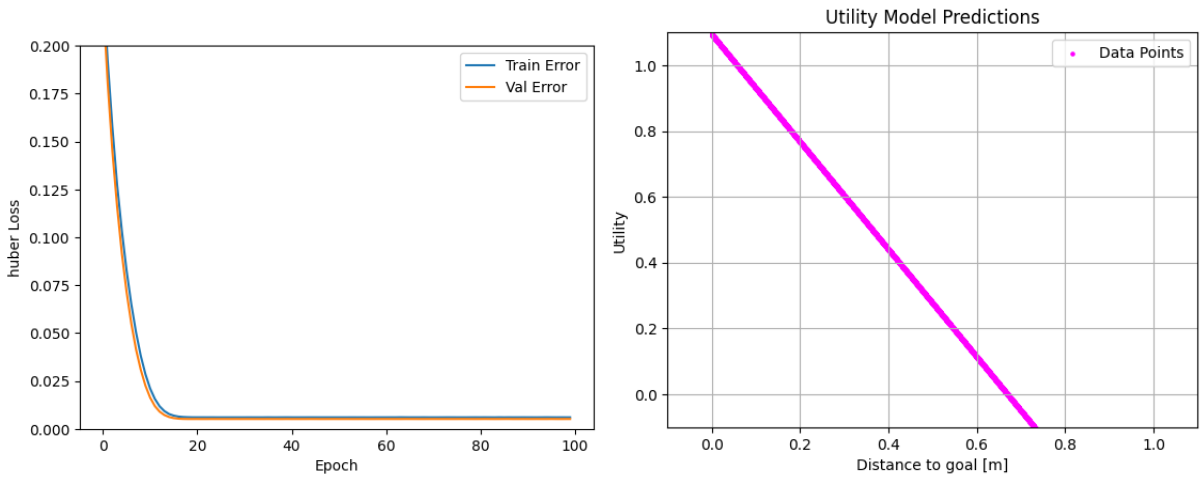
# Apéndice A

# Enlaces a los diferentes materiales

- Enlace a los vídeos de funcionamiento de los robots.
- Enlace a los vídeos de calibración del sistema de captura de movimiento OptiTrack.
- Manual de usuario del sistema de captura de movimiento OptiTrack.
- Enlace a las mediciones realizadas en los sensores del robot

# Apéndice B

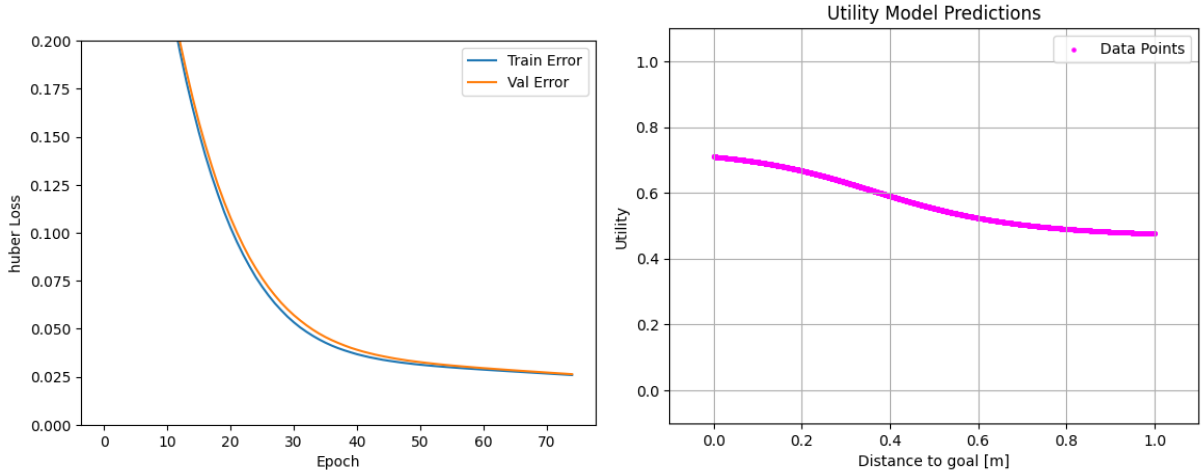
# Imágenes de los resultados de la prueba de hiperparámetros para el modelo de utilidad




**(a)** Pérdida de la red

**(b)** Predicciones de la red

**Figura B.1:** Hiperparámetro cambiado: Función de activación lineal. Descartada por generar utilidades negativas. Elaboración propia.




**(a)** Pérdida de la red

**(b)** Predicciones de la red

**Figura B.2:** Hiperparámetro cambiado: 75 epochs. Descartada por generar una mayor pérdida en la red. Elaboración propia.