Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería Mecatrónica

Desarrollo de un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE.

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura

Carlos Eduardo Jara Vásquez

Cartago, 11 de marzo 2024

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.

Carlos Jara

Cartago, 11 marzo de 2024.

Carlos Eduardo Jara Vásquez

Céd. 1-1730-0218

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el 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 Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: Carlos Eduardo Jara Vásquez

Proyecto: Desarrollo de un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE.

CRESPO MARIÑO JUAN LUIS - 78790530M

Firmado digitalmente por CRESPO MARIÑO JUAN LUIS - 78790530M
Fecha: 2024.03.18 08:47:58 -06'00'

Dr. Ing. Juan Luis Crespo Mariño

Asesor

Cartago, 11 marzo de 2024.

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) 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: Carlos Eduardo Jara Vásquez

Proyecto: Desarrollo de un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE.

CARLOS ADRIAN SALAZAR GARCIA (FIRMA)
Firmado digitalmente por CARLOS ADRIAN SALAZAR GARCIA (FIRMA)
Fecha: 2024.03.14 09:41:48 -06'00'

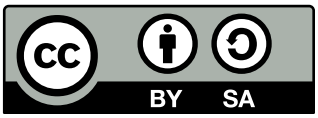
Miembros del jurado evaluador

Digitally signed by FELIPE GERARDO MEZA OBANDO (FIRMA)
Date: 2024.03.14 22:30:23 -06'00'

Ing. Carlos Salazar García	Ing. Felipe Meza Obando
Jurado	Jurado

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

Cartago, 11 marzo de 2024.



Resumen

Este proyecto se desarrolla en el GII y consiste en un robot modular capaz de aprender la representación de su propia morfología mediante su modelo de mundo para realizar pruebas a la arquitectura cognitiva desarrollada por el grupo, la e-MDB. En este informe se explican los conceptos necesarios para comprender correctamente el proceso de diseño, implementación y decisiones tomadas durante el desarrollo. Se sigue la metodología de Ulrich-Eppinger adaptada para utilizarse en un proyecto de investigación. El modelo de mundo esta representado mediante una red neuronal artificial y el sistema robótico se desarrolla con base en los módulos EMERGE. Para la implementación del sistema robótico se rediseña el módulo base del sistema y se le añaden 2 sensores: un sensor infrarrojo de distancia y una IMU, y para el método de aprendizaje se utilizan redes neuronales para lo que se desarrollan múltiples programas en Python: generación de datos, control del sistema, y aprendizaje del modelo de mundo. Los principales resultados obtenidos son: un sistema capaz de aprender modelos de mundo para un robot de hasta 3 módulos en cadena con un porcentaje de error menor a 0,01 %, y el módulo base fabricado, que permite la conexión de hasta 16 módulos en configuraciones de manipuladores, cuadrúpedos y bípedos. Por lo tanto, ahora el GII cuenta con un sistema robótico reconfigurable, modular y escalable para realizar gran variedad de pruebas a la e-MDB. Adicional al proyecto, se realizan pruebas a una parte del e-MDB mediante la cual, utilizando el modelo de mundo aprendido, el sistema robótico implementado es capaz de aprender a llegar a un objetivo por su propia cuenta, además de la redacción de un articulo científicos para dos conferencias distintas.

Palabras clave:

Robótica modular, EMERGE, modelos de mundo, arquitecturas cognitivas, e-MDB, redes neuronales artificiales

Abstract

This project is developed in the GII and consists of a modular robot capable of learning the representation of its own morphology through its world model to perform tests to the cognitive architecture developed by the group, the e-MDB. This report explains the concepts necessary to properly understand the design process, implementation and decisions made during development. It follows the methodology of Ulrich-Eppinger adapted for use in a research project. The world model is represented by an artificial neural network and the robotic system is developed based on EMERGE modules. For the implementation of the robotic system, the base module of the system is redesigned and 2 sensors are added: an infrared distance sensor and an IMU, and for the learning method, neural networks are used for which multiple programs are developed in Python: data generation, system control, and learning of the world model. The main results obtained are: a system capable of learning world models for a robot with up to 3 modules in a chain with an error rate of less than 0,01 %, and the base module manufactured, which allows the connection of up to 16 modules in manipulator, quadruped and biped configurations. Therefore, now the GII has a reconfigurable, modular and scalable robotic system to perform a wide variety of tests to the e-MDB. In addition to the project, tests are carried out on a part of the e-MDB by which, using the learned world model, the implemented robotic system is able to learn to reach a goal on its own, as well as the writing of a scientific paper for two different conferences.

Keywords:

Modular robotics, EMERGE, world models, cognitive architechtures, e-MDB, artificial neural networks

Agradecimiento

Primero que nada, quiero agradecer a mi madre Anabelle Vásquez y a mi padre Carlos Jara, por estar siempre de manera incondicional, asegurándose que no me falte nada y dándolo todo por mí. A mi hermana Catalina Jara que al igual que mis padres me apoyó a lo largo de toda mi carrera, especialmente en los últimos años que vivimos juntos, en los cuales estuvo a mi lado todo el tiempo y ha sido parte esencial de mi vida. El apoyo inquebrantable de mi familia ha sido mi mayor motivación y la fuerza impulsora detrás de mi éxito. Agradecer a mi gran amiga América Rojas, que estuvo conmigo durante todo este recorrido, me apoyó, me motivó a superarme y a buscar mejores oportunidades, sin ella esto no hubiese sido lo mismo.

Agradecer también a mis amigos y compañeros André, Chris, Celimo y Kenneth, con los que tuve la suerte de compartir la mayor parte de la carrera, que me impulsaron a mejorar mi desempeño y a dar lo mejor de mi en los estudios, además de que hicieron de esta experiencia algo inolvidable. También a Itnan e Ian, amigos que han llegado a formar parte importante de mi vida y últimos años de carrera, que igualmente me apoyaron, ayudaron y mejoraron mi vida. Además, quiero agradecer a Francella, cuyos aportes fueron vitales en el desarrollo de este proyecto y mejoró considerablemente esta experiencia. En general a mis amigos, con quienes he compartido risas, desafíos y momentos inolvidables, les agradezco por su amistad y por ser un pilar de apoyo en los momentos más difíciles.

Al profesor Juan Luis Crespo, que atendió todas mis consultas con toda disposición a lo largo de este proyecto y sin el cual este no hubiese sido posible. También a todos los miembros del GII que me ayudaron y mejoraron la experiencia al realizar este trabajo, especialmente a Martín Naya y Alejandro Romero que dedicaron parte de su tiempo para resolver dudas y ayudarme en lo que necesitara. De la misma forma, agradecer a todas las personas de la ESN y Erasmus que me dieron su amistad y apoyo durante estos meses.

Finalmente, agradecer a todas las personas que de una u otra forma impactaron esta experiencia, a familia, compañeros, profesores y demás. Gracias a todos ustedes por haber sido parte de este viaje.

Lista de Contenidos

Lista de Figuras

Lista de Tablas

Lista de Abreviaciones

UDC Universidade da Coruña

GII Grupo Integrado de Ingeniería

e-MDB *epistemic Multilevel Darwinist Brain*

LOLA *Lifelong Open-ended Learning Autonomy*

RNA Redes Neuronales Artificiales

NA Neurona Artificial

API *Application Programming Interface*

FDM *Fused Deposition Modeling*

CAD *Computer Assisted Design*

UE Ulrich-Eppinger

SD Solución Diseñada

EMERGE *Easy Modular Embodied Robot Generator*

IMU *Inertial Measurement Unit*

dps *Degrees per second*

IR Infrarrojo

R+ 3.0 Roboplus 3.0

SGD *Stochastic Gradient Descent*

MSE *Mean Square Error*

MAE *Mean Absolute Error*

PC *Personal Computer*

IWINAC *International Conference on the Interplay between Natural and Artificial Computation*

ALIFE *Conference on Artificial Life*

1 Introducción

En este capítulo se presentan el entorno, definición del problema y objetivos del proyecto, a modo de introducción de forma que se comprenda de manera clara la razón por la que se desarrolla el proyecto y lo que se busca conseguir. Se busca que el lector tenga una idea general de donde y porque se realiza el proyecto, al igual que la importancia que representa para el cliente. Este es el capitulo inicial del proyecto, seguido de esto se presenta el marco teórico, metodología, proceso de diseño, procedimientos para la implementación, resultados y análisis, y finalmente conclusiones y recomendaciones. Al final del documento se adjuntan los anexos, donde se incluyen los archivos importantes del desarrollo de este proyecto.

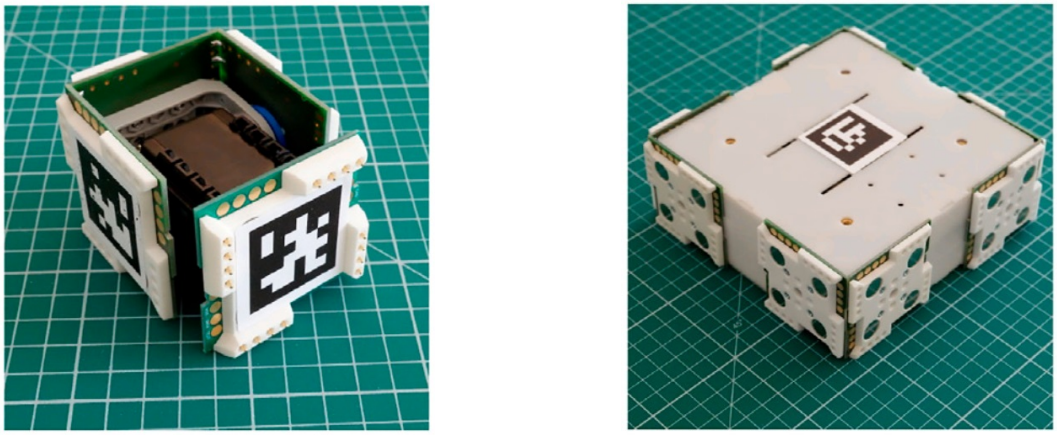
1.1. Entorno

El proyecto se desarrolla en el Grupo Integrado de Ingeniería (GII) que fue creado en 1999 y se ubica en la Universidade da Coruña (UDC), Campus de Ferrol. El GII es un grupo de investigación aplicada integrado por profesionales de múltiples disciplinas y cuyo objetivo es el desarrollo de nuevos productos y generación de conocimiento científico [1].

El GII mantiene fuertes lazos con las Escuelas de Ingeniería del centro educativo, lo que ha provocado que los proyectos desarrollados vayan por la línea de las diferentes especialidades en ingeniería que se ofrecen en la UDC, por lo que los proyectos se desarrollan principalmente en las áreas de: Ingeniería Naval, Dinámica de Fluidos, Inteligencia Computacional, Robótica, Sistemas de Medida y Sensorización y Organización Industrial [2].

El proyecto que se desarrolla en este informe se centra en las áreas de sensorización y robótica. En estas áreas el GII tiene un amplio historial en el desarrollo de proyectos, con inicios únicamente en investigación y actualmente en el desarrollo de proyectos aplicados. El

enfoque actual de los proyectos es la fabricación de estructuras robóticas con capacidades singulares, con el objetivo general de desarrollar sistemas autónomos que sean capaces de adaptarse a los cambios en el medio y establecer nuevas formas de cumplir con la tarea para la cual fueron diseñados, es ahí donde entra el proyecto del desarrollo de un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE que se muestran en la Fig. 1.1 [2].



(a) Módulo EMERGE. (b) Base EMERGE (sin modificaciones).

Figura 1.1. Componentes EMERGE [3].

Para el desarrollo del proyecto se contó con las siguientes tecnologías: módulos reconfigurables EMERGE desarrollados por el *REAL Lab* de la *IT University of Copenhague*, software de diseño de circuitos impresos EAGLE, software de diseño e impresión en 3D, lenguaje de programación Python y el simulador CoppeliaSim.

1.2. Definición del problema

1.2.1. Generalidades

Como se menciona previamente, el desarrollo del proyecto se realiza en el GII en el Campus de Ferrol de la UDC. Se trabaja con el GII específicamente en las áreas de robótica y cognición, áreas en las cuales se ubica el desarrollo de la *epistemic Multilevel Darwinist Brain* (e-MDB), una arquitectura cognitiva con el objetivo de que los robots sean capaces de aprender por ellos mismos a largo plazo, independientemente del entorno en el que opere, tarea a realizar, morfología, entre otros.

En la actualidad dicha arquitectura ha sido probada únicamente en tareas y dominios limitados por características físicas de los robots como el robot Baxter que tiene una morfología fija e invariable, con el cual se han realizado múltiples pruebas con la e-MDB únicamente en tareas de alcance y agarre pues son las únicas posibilidades que el robot ofrece [4].

1.2.2. Justificación

El GII ha desarrollado la arquitectura e-MDB en busca de generar la capacidad en los robots de aprender su morfología y posibilidades de forma independiente y sin importar las condiciones que le rodean o características de este. Sin embargo, al igual que todo proceso de desarrollo e investigación, se deben realizar pruebas al sistema, en este caso las únicas pruebas que se han realizado se han visto limitadas por el único robot con el que cuentan el cual carece de capacidades de reconfiguración y modificación por lo cual no se han podido satisfacer las necesidades de validación. Asimismo, el GII colabora con el *REAL Lab* de la *IT University of Copenhague* en el desarrollo de este proyecto, en el cual el *REAL Lab* abarca los campos de robótica modular y evolutiva, y el GII de la robótica cognitiva y del desarrollo morfológico. En este proyecto se utilizan los módulos EMERGE que desarrolla el *REAL Lab* y acondicionan para que se puedan utilizar en un robot capaz de identificar una representación de su morfología.

A lo largo de los últimos años, con el crecimiento de la industria 4.0, la robótica ha tomado un papel importante en la industria puesto que permite automatizar gran cantidad de tareas, lo que facilita y optimiza procesos. La variedad de tareas es tan grande que un robot modular, reconfigurable y capaz de aprender su morfología y capacidades, representa una herramienta excelente para la industria debido a la amplia gama de posibilidades que este ofrece, lo que apoya la investigación y desarrollo de arquitecturas como e-MDB y robots modulares capaces de aprovechar dicha arquitectura.

Otro factor por tomar en cuenta es que actualmente no se cuenta con una opción en el mercado que satisfaga las necesidades específicas que se requieren para las pruebas de la e-MDB.

Es por lo mencionado a lo largo de esta sección que, utilizando los módulos robóticos EMERGE del *REAL Lab* como base, se desarrolló un robot modular que sea capaz de identificar su morfología, y para esto se modifican los módulos EMERGE y mejoran sus capacidades sensoriales.

1.2.3. Síntesis del problema

Necesidad del GII de un robot modular reconfigurable capaz de aprender su morfología para evaluar la arquitectura e-MDB en múltiples dominios y tareas.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un robot modular capaz de aprender su morfología, mediante la adaptación de módulos robóticos EMERGE.

1.3.2. Objetivos Específicos

1. Evaluar soluciones para aumentar la capacidad de sensorización del robot modular.
2. Rediseñar los diferentes módulos que conforman el robot reconfigurable.
3. Diseñar una solución que permita aprender la propia representación de la morfología del robot modular.
4. Verificar el funcionamiento de la técnica de aprendizaje de representación morfológica implementada.

2

Marco Teórico

En este capítulo se desarrollan los temas necesarios para la comprensión de este informe. Se incluyen los siguientes temas:

- *epistemic Multilevel Darwinist Brain* [1]
- Robótica Modular
- Representación de la Morfología
- Redes Neuronales Artificiales
- Breve descripción de los actuadores utilizados
- Manufactura aditiva por *Fused Deposition Modeling*

Estos temas se utilizan durante el desarrollo del proyecto, para justificar las decisiones de diseño y consideraciones durante la implementación, es por esto que es recomendable la lectura de este capitulo para una mejor comprensión. Asimismo, se referencia las fuentes utilizadas en caso de que se desee obtener información mas detallada de algún tema.

2.1. *Epistemic Multilevel Darwinist Brain*

La e-MDB es una arquitectura cognitiva que busca otorgar a los robots un mayor nivel de autonomía de forma que sean capaces de llevar a cabo aprendizaje abierto (*open-ended*

[1] Se conservan algunos términos en ingles debido a que no tienen traducción o pierden el sentido al traducirse.

learning) a lo largo de toda la vida (*lifelong*). Por lo tanto es una arquitectura diseñada para alcanzar *Lifelong Open-ended Learning Autonomy* (LOLA), en robots reales.

La versión más reciente de la arquitectura esta integrada por cinco componentes principales:

- *Motivational system*: consiste en un sistema que gestiona las motivaciones del robot, permite seleccionar y definir nuevos objetivos, de manera que permite el aprendizaje abierto al generar motivaciones independientes del dominio.
- *Learning system*: es el sistema que se encarga de los mecanismos de aprendizaje *online* de los nodos de conocimiento necesarios por la arquitectura.
- *Long-Term Memory system*: este sistema conforma la el elemento central de la arquitectura, en él se almacenan y relacionan los diferentes nodos de conocimiento.
- *Representational system*: sistema responsable de aprender las representaciones espaciales y funciones para obtener los espacios de estado con los que los componentes de la arquitectura utilizan.
- *Episodic Memory*: espacio donde se guardan los denominados *episodes*, conjuntos de percepciones y acciones para una acción y dominio determinado. Permite el aprendizaje de modelos, policies y funciones de representación.

En la memoria de la arquitectura se almacenan los siguientes nodos de conocimiento y se relacionan de la forma que se muestra en a Fig. 2.1:

- *Perceptions* (P_n): las percepciones, sensación u observaciones corresponden a la información externa e interna de los sensores.
- *Perceptual re-descriptions* (RP_n): corresponden a las funciones que mapean las percepciones reales (valores provenientes de los sensores) en términos de como son utilizadas internamente.
- *P-nodes* (pN_q): representan las clases perceptuales, donde se discretiza el espacio perceptual continuo.
- *Goals* (G_s): son las áreas en el espacio de estados (conjunto de percepciones) para las cuales, una vez alcanzadas, se reduce el valor de al menos uno de los impulsos en el *Motivational system* del robot. Los *goals* dependen del dominio.

- *C-nodes* (cN_u): representan las clases contextuales. Relacionan *P-nodes* (percepciones iniciales), los *goals* (percepciones finales), y los *utility models* o policies necesarios para llegar de las percepciones iniciales a las finales para un dominio dado (representado por el *world model* o modelo de mundo).
- *World Models* ($W\ M_v$): o modelos de mundo, son las funciones que describen el comportamiento del dominio donde opera el robot.
- *Policies* (π_w): una *policy* es una estructura de decisiones reactivas para determinar la mejor acción para un conjunto de percepciones P_t, y usualmente están relacionadas con alcanzar un *goal* especifico G_s.
- *Utility Models* ($U\ M_t$): los *utility models* están asociados a un *goal* especifico G_s y son funciones que determinan la probabilidad de alcanzar G_s a partir de una percepción P_t.
- *Actuation re-description* (RA_x): corresponden a las funciones que transforman la representación interna de la actuación en representación real de la actuación del robot.
- *Actions* (A_y): son las acciones básicas de la arquitectura, pueden actuar sobre los actuadores del robot y en la arquitectura.

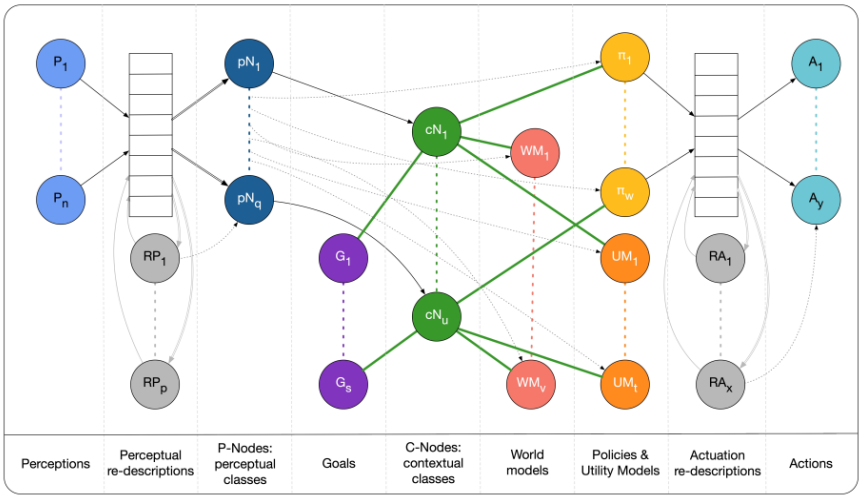


Figura 2.1. Nodos de conocimiento del e-MDB y su relación dentro de la memoria [4].

Los nodos de conocimiento mencionados anteriormente se obtienen de forma autónoma a través de la interacción del robot con el entorno.

En este proyecto lo que concierne son principalmente los modelos de mundo, *world models* ($W\ M_v$), debido a que representa uno de los temas centrales de este proyecto. Como se menciona anteriormente, los modelos de mundo son funciones que describen el comportamiento del dominio dentro del cual se desenvuelve el robot. En otras palabras los modelos de mundo predicen para un conjunto de percepciones (P_t) y acciones (A_y) aplicadas en un tiempo t las percepciones para un tiempo $t+1$ [4].

Los modelos de mundo son utilizados en la arquitectura para proceso deliberativo que se muestra en la Fig. 2.2, este es el proceso mediante el cual la arquitectura determina para un objetivo determinado y conjunto de acciones aleatorias, cual de estas es la mejor para cumplir el objetivo. El funcionamiento consiste en que utilizando el modelo de mundo, para un estado actual del sistema, *Current state*, y un conjunto de acciones, se predice el estado siguiente para cada acción; y con el modelo de utilidad, *Utility Model* se evalúan los estados siguientes para determinar cual es el mejor para cumplir con el objetivo dado.

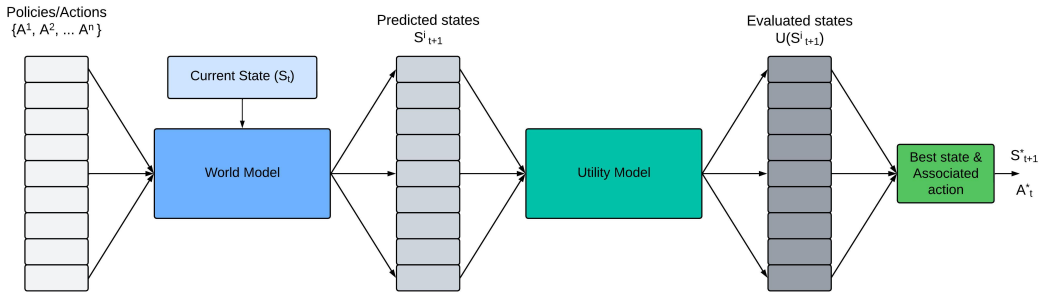


Figura 2.2. Proceso deliberativo del e-MDB [4].

El proceso deliberativo se desarrolla en paralelo en el proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares.

2.2. Robótica Modular

La robótica multicomponente, sistemas robóticos compuestos por múltiples robots sencillos, surge como una aproximación para la automatización de tareas en entornos dinámicos y no estructurados. Tiene como objetivo obtener conjuntos de robots sencillos que colaboren entre si para realizar tareas y evitar la necesidad del desarrollo de un único robot capaz de desempeñar todas las operaciones requeridas. Los sistemas multirobots tienen tres propiedades principales: tienen gran redundancia, son distribuidos y generan muchas sinergias entre robots. Los sistemas multicomponente se dividen en tipos: distribuidos, ligados y modulares.

Esta sección se enfoca en los robots modulares debido a que en este proyecto se emplean únicamente robots multicomponente de este tipo.

Los sistemas robóticos modulares se basan en dispositivos con capacidades de actuación limitadas pero capaces de combinarse entre si para generar sistemas robóticos con una mayor complejidad y capacidad de actuación. Como su nombre indica estos sistemas están compuestos por módulos, unidad elemental de los robots modulares. Los módulos se definen como piezas con capacidades de actuación y sensorización reducidas. En la imagen de la Fig. 2.3 se muestra un ejemplo de sistema robótico modular, el sistema robótico *Easy Modular Embodied Robot Generator* (EMERGE), y algunas de las configuraciones que se pueden obtener al combinar sus módulos. Las configuraciones que se muestran fueron utilizadas en el desarrollo de un articulo sobre el desarrollo de sistemas robóticos evolutivos, se muestran las configuraciones iniciales hasta llegar a una configuración de utilidad como el cuadrúpedo que se muestra en la imagen superior derecha [3].

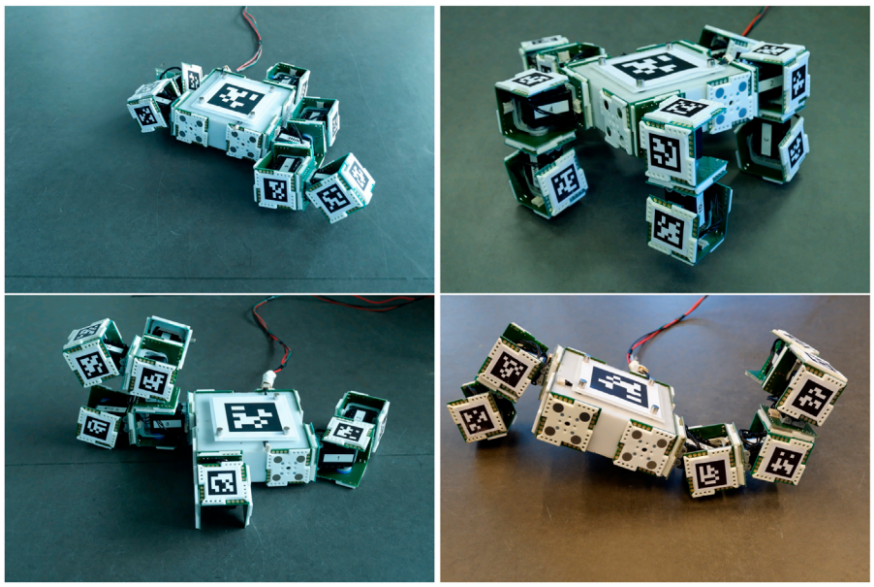


Figura 2.3. Ejemplos de configuraciones del robot modular e-MDB [3].

La morfología del robot depende de como se acoplen los módulos entre sí, lo que permite que según la tarea y entorno en el que el sistema se desenvuelva se selecciona la configuración de los módulos, es por lo tanto que la reconfiguración es considerada una propiedad intrínseca de los robots modulares. Lo que ofrece una gran versatilidad a la hora de resolver tareas en diferentes entornos. Sin embargo, a costa de una mayor versatilidad se sacrifica rendimiento, complejidad mecánica y de control. Es por esto que el rendimiento de un robot modular

siempre sera inferior al de un robot diseñado específicamente para desarrollar una tarea en concreto. Los verdaderos beneficios de este tipo de robots se aprecia al considerar una amplia gama de tareas que en otro caso requerirían de múltiples sistemas robóticos diferentes o cuando se debe realizar el diseño de un sistema robótico sin haber establecido claramente una tarea y entorno.

Un aspecto clave de la robótica modular es la homogeneidad y heterogeneidad de los sistemas, por lo tanto, este tipo de sistemas tienden a ser más homogéneos que heterogéneos. En otras palabras, un sistema modular puede tener diversos tipos de módulos, manteniendo una menor cantidad de tipos de módulos que la cantidad de módulos del sistema. Algunos de los tipos de módulos más comunes son: actuadores, sensores, batería o alimentación y control. Esto se evidencia en la idea detrás de la robótica modular, que consiste en emplear un solo modulo actuador, y aumentar la versatilidad y funcionalidad del sistema al combinar un gran numero de módulos del mismo tipo. De forma que en caso de que un modulo falle, este pueda se remplazado por cualquier otro modulo del sistema puesto que son iguales. Asimismo, la facilidad de reconfiguración se incrementa, debido a que no es necesario tener módulos específicos en posiciones especificas, sino que cualquier modulo puede ubicarse en las posiciones dadas [5].

2.3. Representación de la Morfología

En la realidad el entorno de trabajo donde se desenvuelve un robot esta en constante cambio y hay propiedades que no pueden ser modeladas de previamente, y aun si el entorno es predecible, dotar al robot de este conocimiento no necesariamente es posible o deseable. Es por esto que para que un robot sea realmente autónomo debe ser capaz de de aprender y adaptar sus modelos de mundo a la realidad [6].

De la misma forma que los humanos y los animales, los robots utilizan modelos internos para predecir los efectos que se producen al realizar diferentes acciones sin necesidad de realmente ejecutarlas, de ahí la necesidad de la representación computacional de los modelos internos de la morfología de los robots y la habilidad para simular su representación física. Las predicciones obtenidas mediante estos modelos usualmente son utilizadas en la toma de decisiones con respecto a acciones futuras. Un punto fuerte de estos modelos es que al ser adquiridos correctamente pueden ser utilizados con diferentes propósitos, de manera que son de suma utilidad para el aprendizaje a lo largo de toda la vida en robots [7].

2.3.1. Morfología

La morfología de un robot se define como el conjunto de vínculos (**L**), articulaciones (**J**), que pueden ser actuadas o no; y sensores (**S**). La morfología esta definida para toda la vida útil del robot. Asimismo, la morfología incluye las propiedades asociadas a la posición y orientación de cada uno de los elementos que conforman los conjuntos de vínculos ($\mathbf{L_p}$), articulaciones ($\mathbf{J_p}$) y sensores ($\mathbf{S_p}$) [8].

En la Fig. se muestran los elementos de una morfología, encerrados en círculos de color purpura las articulaciones (**J**) y marcados con lineas de color naranja y rojo los vínculos (**L**). El sensor (**S**) en este ejemplo es una cámara esta encerrado en un rectángulo de color amarillo.

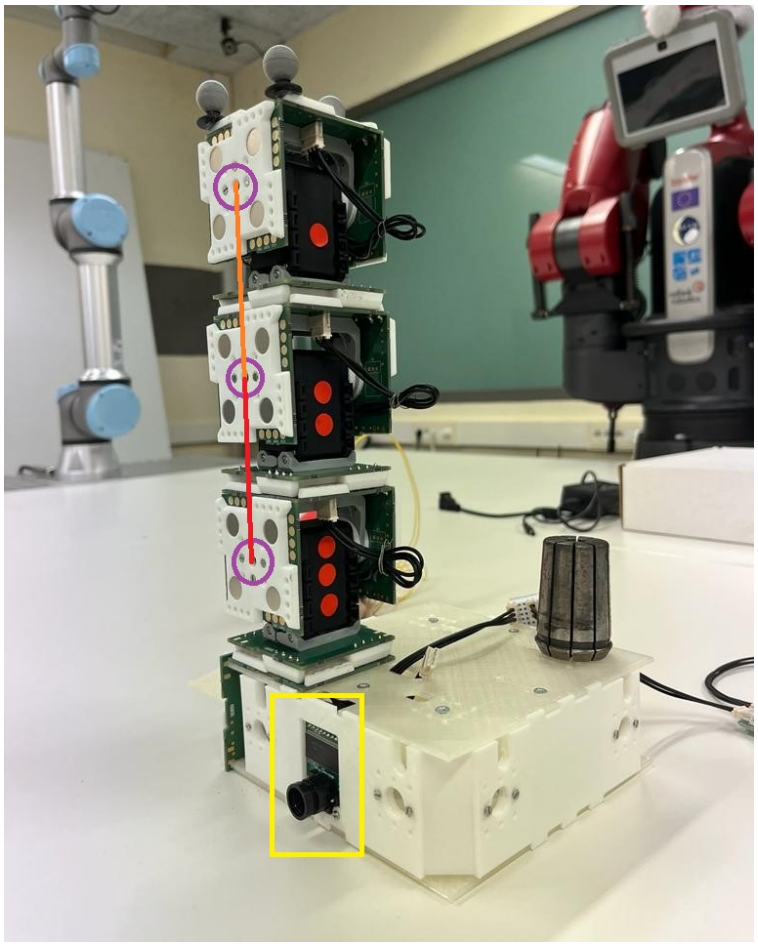


Figura 2.4. Ejemplo de elementos de la morfología.

2.3.2. Representación interna

Los humanos desarrollan un modelo mental de mundo basados en las percepciones; y a partir de este modelo se determinan las decisiones y acciones que se realizan. Los modelos mentales de mundo se definen como la imagen del mundo que los rodea. Para representar este modelo los humanos no toman en consideración todos los detalles del mundo sino los conceptos y relaciones entre estos. De forma que para que esta información sea manejable su cerebro aprende de forma abstracta del los aspectos espaciales y temporales de esta información.

La evidencia indica que lo que en realidad perciben los humanos en un determinado momento esta determinado por una predicción de nuestro cerebro basado en el modelo de mundo interno. Los modelos de mundo internos, más que predecir el futuro, predicen las percepciones futuras para un conjunto de acciones dado [9].

En sistemas robóticos la definición es la misma, con la diferencia de que las percepciones vienen dadas por los sensores del robot y no los sentidos. Por lo tanto, como se mencionó en este capítulo Sección 2.1 el modelo interno de mundo predice las percepciones futuras para un conjunto de percepciones y acciones dado.

La idea principal de estos modelos internos es controlar la interacción de un robot con el entorno en donde se desenvuelve, sin embargo, la relación entre la actuación del robot y los efectos que estas acciones tengan sobre si mismos y el entorno no es $1:1$. Para describir esta relación se utilizan los modelos directos, en inglés *forward models*, que para un estado actual y percepciones dadas, estos predicen las percepciones y/o estado producto de la ejecución de dichas acciones; y los modelos inversos, *inverse models* en inglés, que como su nombre indica de forma inversa a los modelos directos, estos dan para un objetivo determinado las acciones o movimientos que el robot debe ejecutar para alcanzarlo [6].

2.4. Redes Neuronales Artificiales

El cerebro humano se asemeja a una computadora compleja, no lineal y paralela, debido a que es capaz de completar tareas como de reconocimiento de patrones, percepción y control de habilidades motoras mucho más rápido que cualquier computador, a pesar de la gran diferencia en la velocidad entre las compuertas de silicio, que conmutan en nanosegundos, y el sistema neuronal, que trabaja en el orden de los milisegundos. Es por estas capacidades y más que se inicia la búsqueda de un algoritmo para modelar los sistemas neuronales biológicos, lo que actualmente se conoce como Redes Neuronales Artificiales (RNA).

El cerebro humano es sumamente complejo, cuenta con aproximadamente al rededor de 10-500 billones de neuronas en tan solo el corte, con hasta 60 trillones de sinapsis, todo esto únicamente en una parte del cerebro. Es por lo mencionado anteriormente que un modelo completo y realista del cerebro humano es una tarea muy compleja y que requiere una cantidad muy alta de potencia computacional. Debido a que la potencia computacional aumenta de manera proporcional a la complejidad y tamaño de la RNA, lo más común es modelar redes neuronales más pequeñas para resolver un problema específico a la vez, a diferencia del cerebro humano que puede resolver múltiples problemas complejos de forma paralela.

Una Neurona Artificial (NA) es, como su nombre indica, un modelo de una neurona biológica. De forma similar a como funciona el cerebro humano, cada NA recibe señales de su entorno y otras neuronas, para luego transmitirlas a las neuronas conectadas. Estas señales recibidas son amortizadas o potenciadas dependiendo de los pesos asociados a las conexiones entre NAs. La salida de cada NA se controla mediante una función conocida como función de activación. En la Fig. 2.5 se muestra el modelo de una NA, esta recibe todas las señales de entrada y calcula una sola entrada neta determinada por la suma de cada una de las señales de entrada individual multiplicada por el peso correspondiente. La señal entrada total se evalúa en la función de activación para así obtener la salida de la NA.

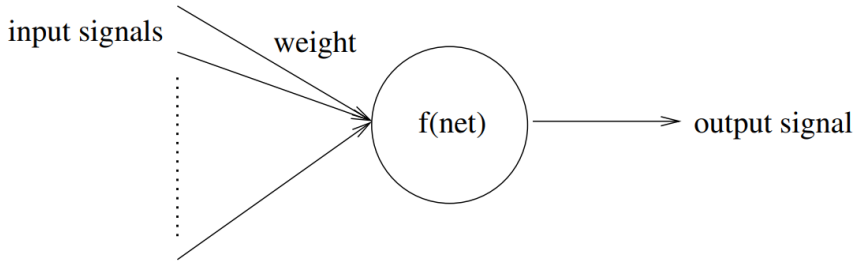


Figura 2.5. Neurona Artificial [10].

En pocas palabras una RNA es un conjunto de capas formadas por NAs. La forma mas común de RNA cuenta con una capa de entrada, una capa de salida y una o múltiples capas ocultas. Como se mencionó, cada una de estas capas esta formada por NAs, y las neuronas entre capas pueden estar parcial o completamente conectadas. La estructura típica de una red neuronal se muestra en la Fig. 2.6.

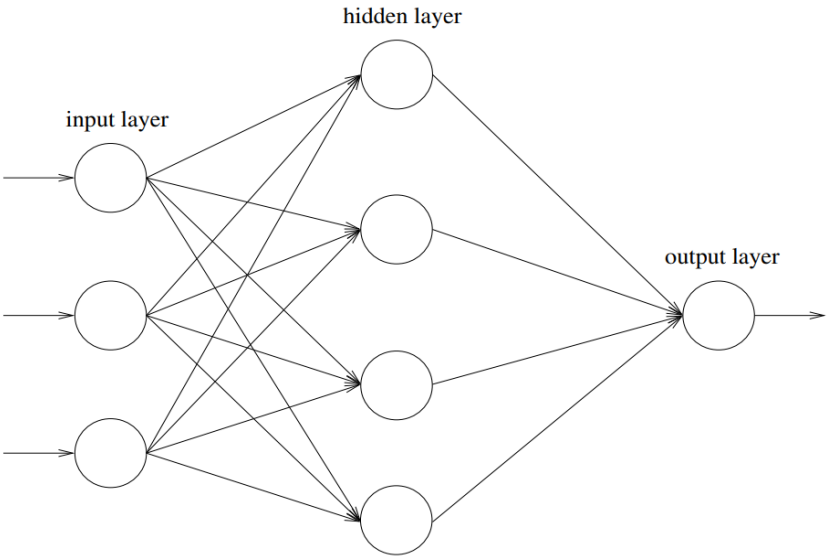


Figura 2.6. Red Neuronal Artificial [10].

Un ejemplo de RNA son las redes *Feedforward* o perceptrón multicapa, ambos términos se utilizaran indistintamente a lo largo de este trabajo [10].

Redes Neuronales *Feedforward*

Una RNA *Feedforward*, perceptrón multicapa de aquí en adelante, en su forma más común se muestra en la Fig. 2.7 y cuenta con tres capas: una capa de entrada, una capa oculta y una capa de salida, sin embargo puede tener múltiples capas ocultas.

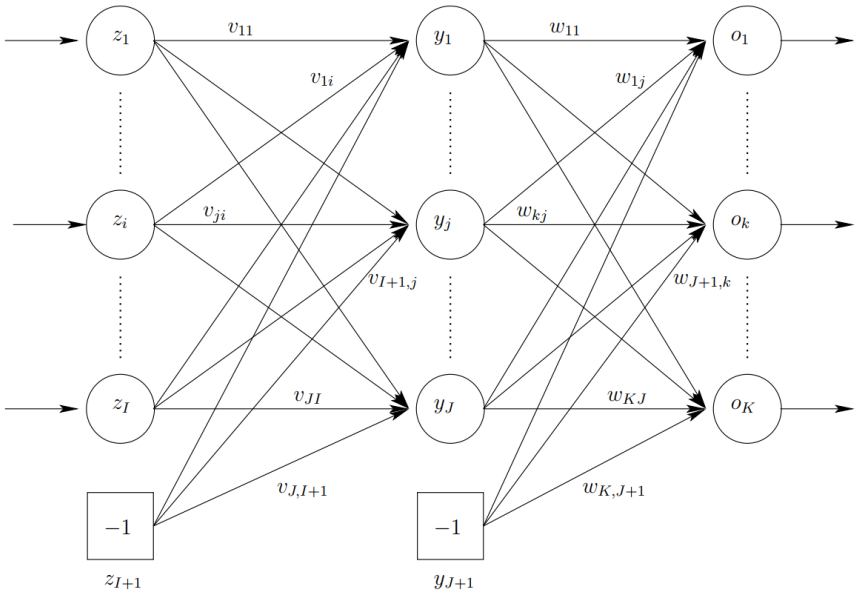


Figura 2.7. Red Neuronal Perceptrón Multicapa [10].

La salida de la red que se muestra en la Fig. 2.7 para un vector de entrada z_p se obtiene al realizar una evaluación directa sobre la red. De forma que para una salida o_k se tiene el comportamiento que se muestra en la Ec. 2.1, donde f_{o_k} y f_{y_j} corresponden a las funciones de activación para la neurona de salida o_k y neurona de la capa oculta y_j, y los términos w_{kj} y v_{ji} corresponden a los pesos entre capas [10].

$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj}\ f_{y_j}\left(\sum_{i=1}^{I+1} v_{ji}\ z_{i,p}\right)\right) \tag{2.1}$$

2.4.1. Aprendizaje supervisado

Una parte importante al trabajar con redes neuronales artificiales es el entrenamiento para poder obtener salidas tan exactas como se desean. Para esto debe realizarse un aprendizaje y entrenamiento, en los que se utilizan algunos algoritmos ya establecidos.

El objetivo del aprendizaje es que utilizando un conjunto de datos (D) que contiene pares de entradas (X) y salidas (y), se pueda aproximar una función (μ_z) para la cual al evaluar una entrada se pueda obtener la salida que corresponda.

Para el aprendizaje, el conjunto de datos se divide en 3 subconjuntos: conjunto de entrenamiento, conjunto de validación y conjunto de prueba. La aproximación de la función μ_z se realiza utilizando los datos de entrenamiento, la memorización se determina mediante los

datos de validación y se determina la precisión mediante el conjunto de validación.

En la optimización del aprendizaje se busca reducir al mínimo el error y para esto existen múltiples métodos y algoritmos que se dividen en: optimizadores locales y globales. Algunos optimizadores locales comúnmente utilizados son el: *Stochastic Gradient Descent* (SGD) y *Scaled conjugate gradient*, y entre los optimizadores locales más utilizados están: *Leap-Frog*, *simulated annealing* y algoritmos evolutivos. Cabe destacar que se pueden combinar algoritmos de ambos tipos para conseguir un entrenamiento híbrido.

Una forma de ver el aprendizaje es el ajuste de los pesos en la red hasta lograr un error aceptable, y para esto existen dos tipos de aprendizaje supervisado:

- Aprendizaje en linea o estocástico: en el que se ajustan los pesos cada vez que se introduce un vector de entradas y salidas, de forma que los vectores de entrada se seleccionan de manera aleatoria.

- Aprendizaje *offline* o por lotes: en el cual se ajustan los pesos únicamente después de que todos los vectores de entrenamiento de un lote han sido analizados [10].

2.4.2. Hiperparámetros

En el aprendizaje automático con RNA se utiliza la biblioteca TensorFlow y su *Application Programming Interface* (API) Keras para la implementación y entrenamiento de modelos. Para la implementación y entrenamiento se deben definir múltiples parámetros que determinan el rendimiento del modelo aprendido, conocidos como hiperparámetros. En la compilación del modelo se presentan los siguientes hiperparámetros:

- Capas ocultas y neuronas: corresponde a la cantidad de capas ocultas que conforman la RNA y las neuronas que conforman cada una de estas capas.

- Función de activación: algunas opciones son las funciones lineal, ReLU, sigmoide, *softmax* y tanh.

- Optimizador: Es el algoritmo utilizado para actualizar los pesos durante el entrenamiento del modelo. En Keras existen múltiples optimizadores entre estos: Descenso de gradiente (*Gradient descent*), Adam, RMSprop, Adagrad y Adadelta.

- Tasa de aprendizaje (*Learning rate)*: La tasa de aprendizaje es un parámetro que permite ajustar el optimizador seleccionado y comúnmente el valor óptimo es aproximadamente 0,001.

- Función objetivo o de perdida: es utilizada por el optimizador para el ajuste de los pesos. Algunas funciones objetivo son: *MSE*, *Binary cross-entropy* y *Categorical cross-entropy*.

En la etapa de entrenamiento de la RNA se deben tomar en cuenta los siguientes hiperpárametros:

- *Epochs*: Es el número de de veces a las que se expone el modelo al conjunto de datos de entrenamiento. Luego de cada iteración el optimizador ajusta los pesos con base en la función objetivo.
- *Batch size*: Numero de instancias de entrenamiento ejecutadas antes de que el optimizador actualice los pesos de la red.

La optimización de estos parámetros se denomina *hyperparameter tuning*, y consiste en encontrar la combinación óptima de hiperparámetros que maximice o minimice una métrica como la precisión, perdida o el error. La optimización de hiperparámetros se realiza mediante prueba y error, ajustando los valores hasta obtener el resultado deseado [11].

2.5. DYNAMIXEL AX-12A

DYNAMIXEL es una marca de servomotores de la compañía ROBOTIS, especialmente diseñado para ser utilizado en sistemas robóticos. Estos servomotores son actuadores inteligentes todo en uno, debido a que están pensados para requerir únicamente de alimentación y una señal de comunicación para su funcionamiento. Es por esto mismo que la compañía ofrece múltiples herramientas y código fuente para un fácil manejo de los motores. Los actuadores DYNAMIXEL ofrecen una gran versatilidad y compatibilidad, de manera funcionan con cualquier sistema o controlador.

Estos actuadores existen en una amplia gama de modelos y ofrecen diferentes modos de funcionamiento y control: de velocidad, posición, torque, entre otros. Además de que cuentan con realimentación en tiempo real para la posición, velocidad, torque, temperatura, voltaje, estado y demás. Adicionalmente incluyen un control de posición y velocidad por PID integrado [12].

En este proyecto se utiliza específicamente el modelo AX-12A y el protocolo de comunicación DYNAMIXEL 1.0. Este modelo cuenta con una tabla de control donde se almacenan

datos del estado y control del dispositivo. Se puede acceder a estos datos utilizando instrucciones de lectura y escritura mediante el uso de una dirección en memoria única para cada registro. La tabla de control se divide en dos áreas: EEPROM y RAM. En la Tabla 2.1 se muestran los datos almacenados en la EEPROM y RAM con su respectiva dirección de acceso [13].

Tabla 2.1. Tabla de control DYNAMIXEL AX-12A [13].

Dirección	**Datos**
EEPROM	
0	Numero de modelo
2	Versión de *firmware*
3	DYNAMIXEL ID
4	*Baud rate*
5	Retraso en tiempo de respuesta
6	Ángulo limite en sentido del reloj
8	Ángulo limite en sentido contra reloj
11	Temperatura máxima
12	Voltaje de entrada mínimo
13	Voltaje de entrada máximo
14	Troque máximo
16	Nivel del paquete de estado
17	LED de alarma
18	Apagado/desconexión
RAM	
24	Encendido/Apagado del torque del motor
25	Encendido/Apagado del LED de estado
26	Margen de posición en sentido reloj
27	Margen de posición en sentido contra reloj
28	Pendiente de posición en sentido reloj
29	Pendiente de posición en sentido contra reloj
30	Posición objetivo
32	Velocidad de movimiento
34	Limite de torque
36	Posición actual
38	Velocidad actual
40	Carga actual
42	Voltaje actual
43	Temperatura actual
44	Estado de la instrucción
46	Estado de movimiento
47	Bloquear EEPROM
48	Corriente mínima

2.6. Manufactura aditiva por *Fused Deposition Modeling*

La manufactura por *Fused Deposition Modeling* (FDM), es un metodo utilizado en impresión 3D, que a su vez es un tipo de manufactura aditiva. FDM consiste en derretir el material base, filamento, y extruirlo para conformar nuevas piezas. El filamento se ubica en bobinas y se toma mediante un motor, luego se introduce en una boquilla de temperatura controlable y se calienta hasta llegar a un estado semilíquido para ser extruido finalmente por la boquilla. El filamento que se extruye es guiado por la boquilla en capas ultrafinas siguiendo el modelo creado utilizando un programa de diseño asistido por computador, por sus siglas en inglés *Computer Assisted Design* (CAD) [14].

3 Metodología

En este capitulo se presenta la metodología seguida para el desarrollo del proyecto, la metodología de Ulrich-Eppinger (UE). En el desarrollo del proyecto se siguió una adaptación de la metodología descrita en el libro "Diseño y desarrollo de productos" por Ulrich y Eppinger, en adelante metodología UE o simplemente UE.

Esta metodología consta de los siete pasos que se muestran en la Fig. 3.1 y está enfocada en el diseño de productos por lo que algunos de estos pasos se adaptan para un proyecto enfocado en investigación como lo es este.

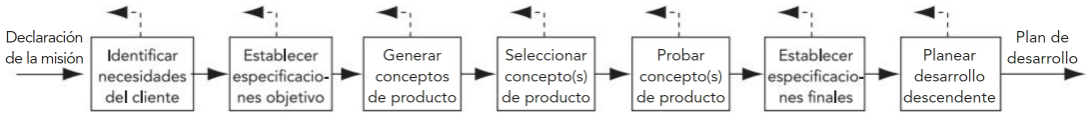


Figura 3.1. Proceso de diseño [15].

De estos siete pasos, por alcance del proyecto únicamente se realizan y adaptan los primeros cinco pasos:

1. Identificar necesidades del cliente
2. Establecer especificaciones objetivo
3. Generar conceptos del producto
4. Seleccionar concepto(s) del producto
5. Probar concepto(s) del producto

La principal adaptación se observa en el paso cuatro, puesto que para algunos de los subproblemas la selección del concepto involucra realizar pruebas.

3.1. Identificar necesidades del cliente

En la metodología descrita por Ulrich y Eppinger el primer paso consiste en identificar las necesidades del cliente, para esto se debe iniciar recopilando los datos e información del cliente, la recolección de datos se realiza mediante múltiples entrevistas y reuniones con los clientes, en este caso con los encargados del proyecto: Richard José Duro Fernández y Martín Naya Varela, con el fin de obtener la mayor cantidad de información sin procesar. Seguido, se procesan los datos obtenidos y se identifican las posibles necesidades. El procesado consiste en identificar las necesidades y establecer su importancia. Para procesar la información se realizan múltiples reuniones posteriores a la definición de necesidades donde se aclaran dudas sobre las necesidades y organización de estas [15]. La importancia de las necesidades se realiza utilizando la escala numérica que va de 1 a 5, que representan respectivamente un atributo poco deseable y un atributo con importancia critica o esencial. De manera que para una futura definición de métricas, la importancia permite establecer una mayor tolerancia conforme menor sea la importancia.

3.2. Establecer especificaciones objetivo

El paso siguiente consiste en establecer las especificaciones objetivo. Ulrich y Eppinger indican que una especificación debe incluir dos partes: la métrica, que consiste en una descripción cuantitativa de un atributos de la Solución Diseñada (SD); y un valor o valores objetivo, que definen el rango de valores para los cuales se considera que se cumple con la necesidad descrita por la métrica correspondiente. El establecimiento de especificaciones se inicia elaborando la lista de métricas, mínimo una por necesidad. Para fijar los valores objetivo se realiza una búsqueda de información, pero principalmente se definieron con los clientes debido a que la totalidad de estos valores dependen del cliente para este proyecto. Con la información recabada se establecen los valores objetivo [15]. Debido a la naturaleza investigativa de este proyecto, se realizaron entrevistas y reuniones periódicas donde las necesidades y principalmente las especificaciones pasaron a través de múltiples iteraciones conforme el proyecto avanzaba, de manera que se ajustara con el progreso del proyecto y los objetivos del GII.

3.3. Generar conceptos del producto

Una vez se han identificaron las necesidades y establecidas las especificaciones, se procede con el tercer paso, la generación de conceptos de producto. En el libro de Ulrich y Eppinger se presentan cuatro actividades principales para la generación de conceptos: el estudio y subdivisión del problema, la búsqueda de conceptos utilizando fuentes tanto internas, lluvia de ideas, como externas, búsqueda de patentes, libros, artículos científicos o en el mercado; luego está la exploración sistemática que consiste en la clasificación y combinación de conceptos [15].

3.3.1. Subdivisión del problema

El objetivo de esta subdivisión es dividir el problema o problemas principales en subproblemas más pequeños que faciliten su resolución y se puedan trabajar uno a uno. El problema que se busca resolver en este proyecto es la necesidad del GII de un robot modular reconfigurable capaz de aprender su morfología, para evaluar la arquitectura e-MDB en múltiples dominios y tareas. Este problema se divide en dos principales subproblemas:

- Sistema robótico
- Método de aprendizaje de la representación de la morfológica

Para esto se utilizan diagramas de caja negra de primer y segundo nivel para los principales subproblemas listados previamente. En el diagrama de primer nivel se pretenden representar las entradas y salidas de cada subproblema, y en el de segundo nivel mostrar los problemas en los que se divide cada uno de los subproblemas, y la relación entre entradas y salidas del sistema.

3.3.2. Búsqueda de conceptos

El siguiente paso en la generación de conceptos, consiste en realizar una búsqueda interna y externa de posibles conceptos para cada uno de los subproblemas definidos. Para esto inicialmente se realiza una lluvia de ideas, que posteriormente se complementa con una investigación para definir los posibles conceptos para el sistema robótico y el método de aprendizaje.

Seguido de la generación de conceptos, se realiza la exploración sistemática en donde se realiza la clasificación y combinación de conceptos. Con los conceptos generados se realizan

múltiples combinaciones para cada subdivisión del problema principal y se obtienen los posibles conceptos por desarrollar. Para esto se consultaron diversas fuentes externas como paginas web, tiendas, hojas de datos y el criterio experto. Adicionalmente se consultaron artículos científicos y libros para los subproblemas relativos al método de aprendizaje.

3.3.3. Combinación de conceptos

Luego de que se tiene el conjunto de conceptos para cada subproblema se procede a realizar múltiples combinaciones de conceptos para generar los posibles candidatos para la SD. Para esto se utilizan tablas donde se listan cada uno de los conceptos y las combinaciones propuestas.

3.4. Selección de concepto(s) del producto

Seguido de haber obtenido los posibles conceptos para la SD se realiza la selección del concepto que se desarrollará mediante el uso de matrices de selección y evaluación en donde primero se filtran los conceptos generados para dejar únicamente los mejores conceptos o los potenciales conceptos por desarrollar. Luego de la etapa del filtrado se utiliza una matriz de evaluación para finalmente obtener el concepto por desarrollar [15].

Debido a que este es un proyecto de investigación la etapa de filtrado y selección se modifica para algunos subproblemas, para los cuales en lugar de utilizar las matrices de evaluación se prueban los diferentes conceptos en busca de la combinación que presente los mejores resultados, de manera que se combinan y ajustan los conceptos hasta obtener el óptimo. Esta adaptación de la etapa se realizó principalmente a los conceptos que estaban relacionados al método de aprendizaje, hiperparámetros y entrenamiento, debido a que el ajuste de los mismos es iterativo y en algunos casos los hiperparámetros pueden tomar una muy amplia gama de valores.

En el paso de selección de conceptos, como se menciona previamente, se realizaron múltiples iteraciones para los conceptos del método de aprendizaje en donde se ajustaron y combinaron nuevamente los conceptos iniciales, siempre basados en el criterio experto, artículos científicos o libros, en busca de la combinación que satisfaga las necesidades del cliente.

3.5. Probar concepto(s) del producto

El paso final consiste en realizar pruebas al concepto seleccionado de producto en donde se analizan los resultados obtenidos para dicho concepto. Se deben realizar diferentes pruebas y comprobar que se cumpla con las métricas y especificaciones establecidas previamente. Se debe asegurar que las necesidades del cliente hayan sido satisfechas. Para este paso se definen las pruebas que se van a realizar a los conceptos ganadores. El éxito en las pruebas se define con respecto a los objetivos, métricas y requisitos tanto del proyecto como del GII. En el caso en que los resultados sean positivos y se cumpliera de forma satisfactoria las pruebas el desarrollo del proyecto se da por finalizado [15].

3.5.1. Sistema Robótico

Se proponen múltiples pruebas de concepto para cada uno de los principales subproblemas. Para el sistema robótico se propone realizar pruebas de sensorización y control.

Pruebas de sensorización

El objetivo de estas pruebas es validar el correcto funcionamiento de los sensores implementados en la base, una *Inertial Measurement Unit* (IMU), por sus siglas en inglés, y un sensor de distancia Infrarrojo (IR).

- Prueba de la IMU: La IMU es un sensor digital y se utilizó para de manera indirecta obtener la orientación de la base, al utilizar el giroscopio para medir la velocidad angular en grados por segundo, en inglés *Degrees per second* (dps). Se propone caracterizar este sensor de manera que se obtenga la exactitud y precisión de cada uno de los ejes del giroscopio. Para esto se fabrica un montaje experimental sobre un actuador AX-12A, donde se fija el sensor en tres posiciones diferentes para tomar mediciones de los tres ejes del giroscopio individualmente. Se aprovecha del funcionamiento de los motores y la capacidad de fijar una velocidad de movimiento que se utiliza de referencia para las mediciones de la IMU. Se prueba con 17 velocidades diferentes y se realiza cada medición 30 veces. En la Tabla 3.1 se muestran los factores de influencia de en esta prueba, velocidad y eje de medición, las combinaciones para realizar esta prueba se realizan tal que para cada eje del giroscopio se realizan mediciones a todas las velocidades de la tabla. Debido a que es una prueba funcional se considera éxito

la correcta conexión con el sensor y medición de los datos, de manera que el resultado final sea determinar la exactitud y precisión de cada eje.

Tabla 3.1. Factores de influencia en prueba de la IMU.

Eje del Giroscopio	**Velocidad (dps)**
X	15
	25
	35
	45
	55
	65
Y	75
	85
	95
	105
	115
	125
Z	135
	145
	155
	165
	175

- Prueba del IR: A diferencia de la IMU, este sensor es analógico, y se utiliza para medir la distancia entre la base y el suelo. Similar a como se realiza con la IMU, con el infrarrojo se busca obtener su curva característica para poder realizar mediciones con el mismo, por lo que se realizan mediciones a múltiples distancias, para poder determinar la relación entre la salida del sensor, en voltios y la distancia medida. Para esto se fabrica un montaje experimental en el cual se utiliza una referencia diseñada en *AutoCAD* y se comprueban las medidas utilizando un *Vernier*. Se prueban 12 distancias diferentes y se realizan 30 mediciones por distancia. Los puntos de medición fueron definidos arbitrariamente y con distancias diferentes entre unos y otros. En la Tabla. 3.2 se muestran las distancias medidas.

Tabla 3.2. Factores de influencia en prueba del IR.

Distancia (mm)
20
22
25
30
40
55
80
105
120
130
140
145
150

Pruebas del control

Estas pruebas se realizar para verificar la implementación del control y diseño de la base y módulos EMERGE. Como parte de la implementación de la base se creó una librería de Python para el control de los motores y sensores del sistema robótico, de forma que también se busca validar las funciones de la librería. En estas pruebas se conectan los módulos a la base, y se ejecutan las funciones para el control de los motores, recolectando datos y comprobando que se ejecuten correctamente. El éxito de esta prueba se define únicamente como que el sistema sea capaz de ejecutar las funciones. Para las pruebas se van a conectar uno y 3 módulos en cadena, y se van a ejecutar todas las funciones principales implementadas. Los factores de influencia en la prueba de los módulos se muestran en la Fig. 3.3. En las configuraciones el orden de los módulos se muestra de abajo hacia arriba donde A, D e I significan arriba, derecha e izquierda respectivamente y corresponde a la posición del modulo con respecto al modulo anterior. De manera que se prueben todas las funciones para 5 combinaciones aleatorias, 3 con módulos independientes, con 20 repeticiones por módulo para un total de 60, y 2 con combinaciones de 3 módulos, y 30 repeticiones por combinación para obtener 60 repeticiones.

Tabla 3.3. Factores de influencia en prueba de los módulos.

Módulos	**Configuración**	**Función**
1	independiente	*getJointPosition*
2	3, 2A, 1A (A)	*setJointTargetPosition*
3	2, 3D, 1D (B)	*getJointList*
		loadEMERGE

3.5.2. Método de aprendizaje de la representación de la morfología

El método de aprendizaje utilizado son Redes Neuronales Artificiales, la cual fue optimizada para realizar predicciones en las salidas con un porcentaje de error menor a 0,005 % para la morfología que se muestra en la Fig. 3.2, una cadena de 3 módulos alineados.

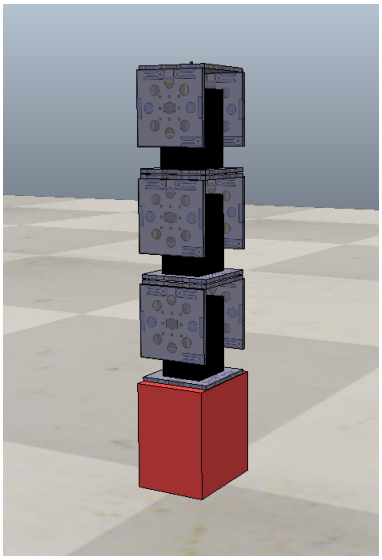


Figura 3.2. Morfología para optimización del método de aprendizaje.

Como parte de la validación de la solución se plantean cinco morfologías diferentes para las cuales se va a realizar el aprendizaje seis veces por cada una, para un total de 30 replicas, en busca de obtener un error en todas las salidas menor al 0,01 % para más del 90 % de las morfologías. La generación de datos se realiza mediante simulación debido a que la generación de datos utilizando el sistema real toma una cantidad de tiempo con la que no se cuenta en este proyecto. Además de que para tener la ubicación del extremo de la cadena requiere de la implementación de un sistema de visión que está fuera del alcance del proyecto.

4 Propuesta de Diseño

En este capítulo se desarrolla brevemente la metodología seguida que se explica en el Capítulo 3. Se describe a profundidad el proceso mediante el cual se completaron los cinco pasos de la metodología UE, todas las decisiones tomadas y el razonamiento detrás de cada una de estas. En el desarrollo de cada uno de los pasos se hace referencia a los recursos utilizados: tablas, figuras y fuentes citadas.

4.1. Identificar necesidades del cliente

El primer paso de la metodología seguida consiste en la identificación de las necesidades, para esto primero se debe entender la problemática que se desea resolver, en este caso la necesidad del GII de un robot modular reconfigurable capaz de aprender su morfología para evaluar la arquitectura e-MDB en múltiples dominios y tareas, por lo tanto, se necesita comprender el funcionamiento del e-MDB, los dominios y tareas que se desean probar. Para entender estos temas se realizan múltiples reuniones y entrevistas. Además, se revisan diversos artículos relacionados con los temas de robótica modular y representaciones de la morfología. Una vez se tiene un mayor conocimiento sobre la problemática se realizan reuniones adicionales donde se recolecta información, se establecen las necesidades y la importancia correspondiente. Los resultados de este paso se que se presentan en la Tabla 4.1.

Tabla 4.1. Necesidades del proyecto.

#	Necesidad	Importancia
1	La SD debe ser modular y reconfigurable.	5
1.1	La SD utiliza únicamente módulos EMERGE.	5
1.2	La SD implementa los sensores únicamente sobre la base del robot.	3
1.3	La SD ofrece múltiples opciones adicionales para conectar los módulos.	4
1.4	La SD cuenta con múltiples módulos.	4
1.5	La SD cuenta con base.	3
1.6	La SD requiere modificaciones únicamente sobre la base del robot.	4
2	La SD aprende su propia morfología.	5
2.1	La SD identifica sus propios módulos.	4
2.2	La SD utiliza redes neuronales para el aprendizaje de la morfología.	5
2.3	La SD utiliza CoppeliaSim para las simulaciones.	5
2.4	La SD se implementa utilizando el lenguaje de programación Python.	5
2.5	La SD es capaz de aprender su propia morfología para cadenas de módulos y sus variantes.	3
2.6	La SD funciona para superficies planas.	5
3	La SD contempla materiales y elementos con alta disponibilidad.	4
4	La SD se mantiene dentro del presupuesto.	4

Las necesidades que se muestran en la Tabla 4.1 se comentan con los investigadores para confirmar que sean correctas y no se estén omitiendo necesidades.

4.2. Establecer especificaciones objetivo

Luego de que las necesidades están debidamente establecidas se definen las especificaciones objetivo. Debido a que las necesidades fueron confirmadas por el cliente se definen las métricas que se busca cumplir al finalizar el proyecto. Las métricas se definen en conjunto con el cliente durante reuniones, se proponen valores iniciales basados en la información previa y se ajustan para cumplir con los requisitos del cliente durante estas reuniones.

Debido a la gran componente de investigación con la que cuenta este proyecto, las especificaciones objetivo pasaron a través de múltiples iteraciones, donde se ajustaron tanto valores como métricas. Una de las principales razones por las que se modificaron las especificaciones es debido a una aplicación especifica con la que el cliente deseaba probar la SD, el proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares que se desarrolló en paralelo, en el que se aprovecha el modelo de mundo aprendido y el sistema robótico implementado. Por lo mencionado anteriormente, en la Tabla 4.2 se muestran las especificaciones finales, es decir, la iteración final de las especificaciones objetivo, y se van a utilizar para evaluar el concepto final desarrollado.

Tabla 4.2. Métricas y especificaciones finales.

#	Métrica	Imp.	Unidad	V. Marginal	V. Ideal
1	Cantidad de módulos disponibles	5	módulos	> 3	3
2	Cantidad de bases disponibles	5, 3	bases	1	≥ 1
3	Morfologías que puede adoptar	5	lista	*Lista 1	**Lista 2
4	Tipo de módulos utilizados	5	lista	EMERGE	EMERGE
5	Cantidad de sensores adicionales fuera de la base	3	sensores	≤ 2	0
6	Cantidad de conexiones adicionales en la base	4	conexiones	≥ 4	8
7	Modificaciones fuera de la base	4	mods.	4	< 4
8	Morfologías que es capaz de aprender	5, 3	lista	*Lista 1	**Lista 2
9	Error con el que aprende las morfologías	5, 3	%	0,01	$\leq 0{,}005$
10	Cantidad de módulos que es capaz de identificar	4	módulos	2	≥ 3
11	Topologías utilizadas para el aprendizaje	5	lista	RNA	RNA
12	Software utilizado para la simulación de la solución	5	lista	CoppeliaSim	CoppeliaSim
13	Lenguajes de programación donde se implementa la solución	5	lista	Python	Python
14	Superficies sobre las cuales aprende su propia morfología	5	lista	plana	plana
15	Disponibilidad	4	lista	corto plazo	inmediata
16	Presupuesto utilizado	4	€	4200	4200

*Lista 1: cadena de 3 módulos alineados, cadena de 2 módulos alineados
**Lista 2: cadena de 3 módulos alineados, cadena de 2 módulos alineados, combinación de 3 módulos, combinación de 2 módulos

En la Tabla 4.2 se observan las métricas, importancia asociada a la(s) necesidad(es) que responde(n) y el valor marginal e ideal con sus unidades correspondientes. Los valores fueron definidos de manera cercana con el cliente debido la arquitectura del e-MDB y su compatibilidad, que en conjunto con el uso de módulos EMERGE, son las necesidades que principalmente acotaron las métricas.

4.3. Generar conceptos del producto

Esta etapa se inicia con el análisis y división de la problemática principal en subproblemas más fáciles de abordar, de forma que se buscan soluciones para cada subproblema de forma independiente. Luego se buscan posibles soluciones a cada subproblema, conceptos. Con los conceptos generados se definen combinaciones para pasar por un proceso de selección. En el proceso de selección se establecen criterios de selección para determinar cual combinación de conceptos se adapta mejor a las necesidades del cliente para finalmente ser desarrollada.

4.3.1. Subdivisión del problema

En la generación de conceptos se realiza inicialmente la subdivisión del problema. Como se menciona en la Sección 3.3, la problemática que se trata en este proyecto es la necesidad del GII de un robot modular reconfigurable capaz de aprender su morfología, para evaluar

la arquitectura e-MDB en múltiples dominios y tareas. Este problema principal se divide en dos subproblemas:

- el desarrollo de un sistema robótico
- y la implementación de un método de aprendizaje de la representación de la morfología.

Por lo tanto se van a abordar ambos problemas por separado, tomando en cuenta que tienen cierto grado de relación el uno con el otro. Para la división de estos subproblemas se utiliza el diagrama de caja negra de primer nivel y segundo nivel, para representar las entradas y salidas de cada problema con mayor facilidad y la relación entre estas y los subproblemas.

Lo primero que se hace es realizar el diagrama de caja negra de primer nivel para ambos subproblemas. En la Fig. 4.1 se muestra el diagrama del sistema robótico con sus entradas y salidas correspondientes. Se observa que cuenta con tres entradas: alimentación o energía, la de la señal de control y las percepciones; y como salidas se tienen el movimiento del sistema y el conjunto de señales provenientes de los sensores.

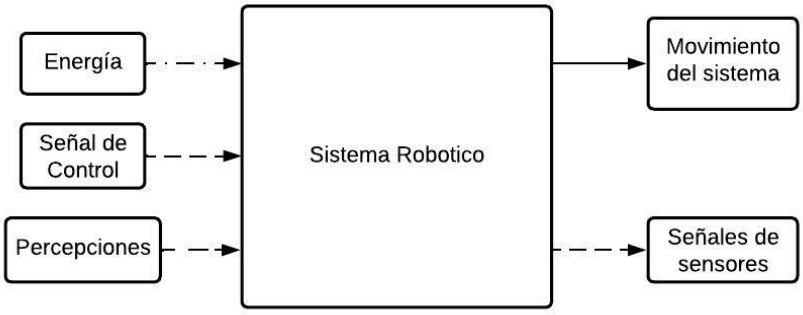


Figura 4.1. Diagrama de caja primer nivel del sistema robótico.

El diagrama del método de aprendizaje se muestra en la Fig. 4.2. Este, a diferencia del sistema robótico cuenta únicamente con una entrada y una salida, las señales de los sensores y el modelo o representación de la morfología respectivamente.

Figura 4.2. Diagrama de caja primer nivel del método de aprendizaje.

Con base en los diagramas de las Figs. 4.1 y 4.2 se realiza la subdivisión de los subproblemas. El problema del diseño y desarrollo del sistema robótico se subdivide en múltiples subproblemas, sin embargo, en este proyecto los subproblemas relativos a la actuación, alimentación, comunicación, conexión entre módulos, entre otros; ya vienen definidos previamente por el sistema base utilizado, el sistema robótico EMERGE. Es por esto por lo que en la búsqueda de conceptos únicamente se abarcan los subproblemas de control y sensorización. En la Fig. 4.3 se muestra el diagrama de caja de segundo nivel del sistema robótico, se observan los dos subproblemas en los que se subdivide y la relación entre entradas, salidas y subproblemas.

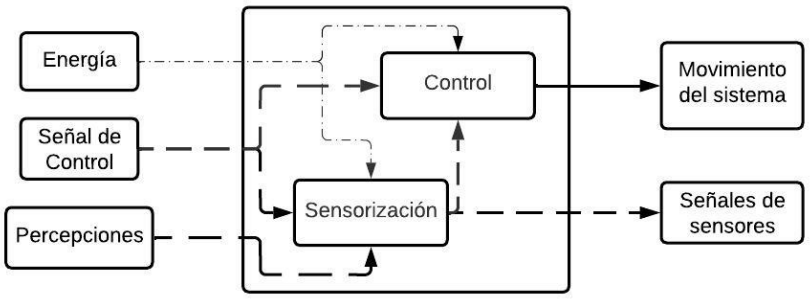


Figura 4.3. Diagrama de caja segundo nivel del sistema robótico.

Para el control se debe seleccionar el método o dispositivo mediante el cual se van a controlar los motores, y se divide en hardware y software. Debido a que en el EMERGE se utilizan los actuadores AX-12A de DYNAMIXEL se debe seleccionar un hardware y software compatible.

El problema de la sensorización del sistema se analiza primero desde el punto de vista de la generación de datos del modelo y se detalla en el Anexo A.1 donde se desarrolla todo

el proceso de sensorización y adaptación de la base, el cual se explica a profundidad más adelante. Inicialmente el problema se abordó con el objetivo de contar con la sensorización necesaria para que un robot cuadrúpedo fuera capaz de aprender su propia morfología, por lo que los sensores y diseño de la base del sistema se realizo con este objetivo en mente, sin embargo, conforme avanzo el desarrollo del proyecto se decidió enfocarse en un brazo robótico, o manipulador, de hasta tres módulos en serie. Con base en las simulaciones realizadas, se determina que para el aprendizaje del propio modelo interno de un robot cuadrúpedo las percepciones que se requieren son la orientación de la base y la altura, cosas que se obtienen fácilmente de la simulación, que para obtenerlas en la realidad se deben seleccionar sensores que permitan obtener estas percepciones.

El método de aprendizaje está compuesto por dos subproblemas: los hiperparametros del método de aprendizaje, que debido a los requisitos del proyecto debe hacerse en Python y utilizar RNA; y la generación de datos para el entrenamiento de la RNA. En la Fig. 4.4 se observan ambos subproblemas y su relación, debido a que la RNA puede ser entrenada con los datos obtenidos mediante los sensores del robot real o mediante los datos generados mediante simulación dependiendo de lo que se desee. Para este proyecto, por términos de tiempo y alcance, no se puede realizar la recolección de datos con el robot real, por lo que se utiliza el simulador para generar los datos de entrenamiento.

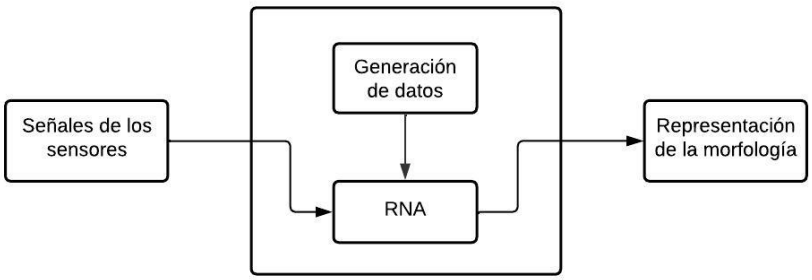


Figura 4.4. Diagrama de caja segundo nivel del método de aprendizaje.

El subproblema de la generación se datos se divide en el simulador que se va a utilizar y la interfaz de control del simulador, que en este proyecto debido a que se van a utilizar RNA se debe controlar el simulador mediante Python y por necesidad del cliente el simulador debe ser CoppeliaSim. Por otro lado, en el subproblema de la RNA se deben ajustar los siguientes hiperparámetros: cantidad de capas ocultas y de NA por capa, función de activación, optimizador, taza de aprendizaje, función de perdida, *epochs* y *batch size*.

4.3.2. Búsqueda de conceptos

Para la búsqueda de conceptos se realiza inicialmente una lluvia de ideas de posibles soluciones para cada subproblema, de forma que se generan conceptos iniciales que posteriormente se desarrollan y se especifican recurriendo a fuentes externas como artículos científicos y proveedores para finalmente obtener un conjunto de conceptos para combinar y seguir con el paso de la selección de conceptos.

Sistema robótico

Como se muestra en el diagrama de la Fig. 4.3 se deben generar conceptos para el control y sensorización del sistema robótico. El control del sistema depende directamente de los actuadores utilizados, AX-12A, por lo tanto, ambos hardware y software deben ser compatibles el uno con el otro, al igual que funcionar correctamente con los actuadores. La serie de actuadores AX, de la cual forma parte el actuador utilizado, es compatible con los controladores: CM-5, CM-510, CM-530, CM-700, CM-900, OpenCM9.04 + OpenCM 485 EXP, OpenCR y OpenRB-150. Adicionalmente se pueden controlar los actuadores mediante una interfaz de comunicación, como el USB2AX o U2D2, en conjunto con una computadora personal, en inglés *Personal Computer* (PC). No todas las opciones mencionadas anteriormente se encuentran disponibles en la tienda oficial de ROBOTIS o en otras tiendas, por lo tanto en la Tabla 4.3 se muestran las opciones de hardware disponibles para el control. Por otro lado, también se debe seleccionar el software de control que se utiliza en conjunto con el hardware. El software debe ser compatible con los actuadores y el controlador o interfaz de comunicación seleccionada, por ende, los software disponibles son: Roboplus 3.0 (R+ 3.0), Dynamixel SDK y Arduino IDE. Los conceptos generados de software y hardware se muestran en conjunto en la Tabla 4.3 [16]-[18].

Tabla 4.3. Conceptos generados para el control del sistema robótico.

Hardware	***Software***
CM-530	R+ 3.0
OpenRB-150	Dynamixel SDK
OpenCR	Arduino IDE
OpenCM9.04 + OpenCM 485 *Expansion Board*	
PC + USB2AX	
PC + D2U2	

La sensorización del sistema se debe realizar tal que se pueda obtener su orientacion

y altura. Para la orientación se considera el uso de una IMU, un giroscopio o sistema de visión OptiTrack; y para la altura se considera el uso de un sensor de distancia por IR, por ultrasonido, o el sistema de visión OptiTrack. En la selección de estos conceptos se debe considerar la compatibilidad y el control de los sensores, para esto se dispone de un Arduino y un PC. La compra de los sensores se debe realizar mediante el GII y con una tienda aprobada por el cliente, por lo tanto las opciones de compra para cada sensor dependen de la tienda BricoGeek, una tienda de electrónica, robótica, Arduino y Raspberry Pi, cofinanciada por la Xunta, gobierno autónomo, de Galicia [19]. En cuanto al sistema de visión OptiTrack, el GII cuenta ya con este sistema instalado en uno de sus laboratorios. Los conceptos generados para la sensorización del sistema robótico se muestran en la Tabla 4.4.

Tabla 4.4. Conceptos generados para la sensorización del sistema robótico.

Orientación	**Altura**	**Control de Sensores**
IMU (ICM-20948)	IR (Sharp GP2Y0A51SK0F)	Arduino (MEGA 2560 rev3)
Giroscopio (MPU-6050)	Ultrasonido (HC-SR04)	PC
OptiTrack	OptiTrack	

En la Tabla 4.4 se muestra una opción para cada uno de los conceptos de sensorización, estas son las únicas opciones disponibles para su compra. A continuación se detalla la razón por la cual se consideran satisfactorias estas opciones [19]:

- La IMU ICM-20948 es muy versátil puesto que, incluye un giroscopio y acelerómetro, de 3 ejes cada uno, de rango programable por lo que puede medir diversos rangos de velocidad angular y aceleración, además de ser compatible con Arduino y otros controladores mediante I^2C y SPI. Adicionalmente cuenta con magnetómetro que puede ser aprovechado en aplicaciones futuras.

- El módulo MPU-6050, consiste en un giroscopio y un acelerómetro de 3 ejes a un precio muy accesible y ampliamente utilizado en proyectos de robótica. Se comunica mediante I^2C.

- El sensor de distancia IR Sharp GP2Y0A51SK0F es un sensor analógico que entrega en su salida un voltaje inversamente proporcional a la distancia medida. El detalle mas importante de este sensor es su rango de funcionamiento, de 2 a 15 cm, el cual es suficiente puesto que el robot no va a llegar a alturas mayores a 10 cm.

- El sensor de distancia por ultrasonido HC-SR04, este sensor de distancia se destaca principalmente por su rango de funcionamiento de 2 a 400 cm, más que suficiente para el sistema, y su bajo coste.

- Se selecciona el Arduino MEGA 2560 rev3 para el control de los sensores puesto que está disponible de forma inmediata, y al ofrecer gran cantidad pines digitales y analógicos permite que la base sea escalable en caso de querer aumentar las capacidades del robot. Una versión más sencilla de Arduino también hubiese funcionado, a costa de sacrificar la escalabilidad que ofrece el Arduino MEGA 2560 rev3, que a pesar de que no es necesaria es algo que se aprecia especialmente en aplicaciones de LOLA.

Un detalle a destacar es que el cliente previamente adquirió algunas de estas opciones puesto que una de las necesidades más importantes del proyecto es la disponibilidad de los componentes, y debido a que la adquisición de algunos de ellos puede tardar mucho, las opciones seleccionadas para la IMU, el IR, ultrasonido, y Arduino se compraron previo al desarrollo de este proyecto, sin embargo, si alguna de estas opciones no hubiese sido adecuada siempre existió la posibilidad de adquirir otra.

Método de aprendizaje de la representación de la morfología

Al utilizar RNA se debe realizar el estudio de hiperparametros relativos a la arquitectura de la RNA y su entrenamiento. Para la generación de datos hay muy poca flexibilidad debido a que las necesidades del proyecto son muy claras: para la simulación se debe utilizar CoppeliaSim en conjunto con Python, al igual que para el control del sistema real. Por lo tanto, la generación de conceptos para el método de aprendizaje se enfoca únicamente en los hiperparámetros de la RNA.

La generación de los conceptos iniciales se realiza mediante una investigación en múltiples artículos sobre optimización de redes neuronales para regresión polinomial, en conjunto con el criterio experto. Se generan conceptos para los siguientes hiperparámetros [11], [20], [21]:

- Cantidad de capas ocultas y NA: En cuanto a la cantidad de capas ocultas y neuronas se definen cantidades arbitrarias para realizar las pruebas. Usualmente tres capas ocultas son suficiente para resolver problemas de regresión con redes neuronales.

- Función de activación: Al trabajar con redes neuronales para regresión las funciones de activación mas comunes son la lineal, sigmoide, tanh y ReLU.

- Optimizador: Los optimizadores SGD, RMSprop y Adam han mostrado buenos resultados en problemas de regresión con redes neuronales artificiales, especialmente Adam debido a su capacidad de adaptación, eficiencia y bajo consumo de memoria. Por lo tanto, estos se proponen como conceptos para las combinaciones del método de aprendizaje.

- Taza de aprendizaje: La taza de aprendizaje suele tomar valores menores a 1 y mucho mayores que 0. El valor estándar recomendado para la taza de aprendizaje es de 0,001, sin embargo, también se prueba con dos valores adicionales 0,1 y 0,4.

- Función de perdida: Al trabajar con redes neuronales en TensorFlow existe una gran variedad de posibles funciones de perdida, las mas comunes son *Mean Square Error* (MSE) y *Mean Absolute Error* (MAE). Adicionalmente se prueba con Huber.

- *Epochs* y *batch size*: Los *epochs* y el *batch size* son hiperparametros que pueden tomar una amplia gama de valores, por lo que se establecen valores arbitrarios para realizar pruebas iniciales y posteriormente ajustarlos hasta obtener el mejor concepto para el método de aprendizaje.

Un detalle a considerar es que el conjunto total de datos de entrenamiento se divide en dos subconjuntos: uno con los datos para pruebas y otro para entrenamiento y validación. Este último, se debe dividir en dos, de los cuales una parte se utiliza únicamente para entrenamiento y la otra para validar el entrenamiento. Basado en el criterio de un experto, se recomienda utilizar al menos un 70 % del total del conjunto de datos únicamente para el entrenamiento, por lo que se deben definir los porcentajes en los que se divide el conjunto de datos. Se determina que si se realiza la división tal que un 15 % del conjunto total se utiliza para pruebas y un 17 % del restante para validación, se tiene aproximadamente un 70,55 % del total de los datos para entrenamiento, de manera mantiene sobre el valor recomendado.

Debido a que algunos de los hiperparámetros pueden tomar una mayor cantidad de valores que otros, puesto que son numéricos y las opciones son relativamente infinitas, se van a separar en conceptos numéricos y conceptos categóricos, que solo pueden tomar valores dentro de una pequeña lista de opciones. En las Tablas 4.5 y 4.6 se muestran respectivamente los conceptos numéricos y categóricos generados para el método de aprendizaje.

Tabla 4.5. Conceptos numéricos generados para el método de aprendizaje.

Capas	**NA**	**Taza de Aprendizaje**	***Epochs***	***Batch Size***
1	9	0,001	30	16
2	18	0,1	40	20
3	27	0,4	50	32
	36		75	

Tabla 4.6. Conceptos categóricos generados para el método de aprendizaje.

Función de Activación	**Optimizador**	**Función de Perdida**
Lineal	SGD	MSE
Sigmoide	RMSprop	MAE
Tanh	Adam	Huber
ReLU		

En la Tabla 4.7 se muestran el conjunto de conceptos generados para el método de aprendizaje que son utilizados para generar las combinaciones de conceptos en el paso siguiente.

Tabla 4.7. Conceptos generados para el método de aprendizaje.

Capas	**NA**	**T. Aprend.**	***Epochs***	***B. Size***	**F. Act.**	**Opt.**	**F. Perd.**
1	9	0.001	30	16	Lineal	SGD	MSE
2	18	0.1	40	20	Sigmoide	RMSprop	MAE
3	27	0.4	50	32	Tanh	Adam	Huber
	36		75		ReLU		

4.3.3. Combinación de conceptos

Utilizando los conceptos generados para cada subproblema se realizan combinaciones de estos considerando su compatibilidad. Las combinaciones generadas, en la siguiente etapa son evaluadas y filtradas para así seleccionar el concepto ganador.

Sistema robótico

Al combinar los conceptos del sistema robótico de las Tablas 4.3 y 4.4, se generan las combinaciones de la Tabla 4.8. La generación de estas combinaciones se realizó de manera que los conceptos generados fueran compatibles entre si.

Tabla 4.8. Combinación de conceptos para el sistema robótico.

Comb.	Control		Sensorización		
	Hardware	**Software**	**Orientación**	**Altura**	**Ctrl. Sens.**
1A	CM-530	R+ 3.0	IMU	Ultrasonido	Arduino
2B	PC + USB2AX	Dynamixel SDK	IMU	IR	Arduino
3C	*OpenCM9.04	Arduino IDE	Giroscopio	IR	Arduino
4D	OpenCR	Dynamixel SDK	IMU	Ultrasonido	Arduino
5E	PC + D2U2	Dynamixel SDK	Giroscopio	Ultrasonido	Arduino
6F	*OpenCM9.04	R+ 3.0	OptiTrack	OptiTrack	PC

*OpenCM9.04 es compatible con el AX-12A y R+ 3.0 únicamente si se utiliza en conjunto con la OpenCM 485 *Expansion Board*.

Método de aprendizaje de la representación de la morfología

Similar al como se trabajó con el sistema robótico, en la Tabla 4.9 se muestran las combinaciones generadas para el método de aprendizaje. Para este subproblema las combinaciones fueron generadas de manera arbitraria para probar las diferentes combinaciones, y en la siguiente etapa realizar una comparación y determinar la mejor combinación para posteriormente optimizarla para que cumpla con las necesidades del cliente.

Tabla 4.9. Combinación de conceptos para el método de aprendizaje.

Comb.	**Capas**	**NA**	**T. Aprend.**	***Epochs***	***B. Size***	**F. Act.**	**Opt.**	**F. Perd.**
1A	1	9	0.001	30	20	Lineal	Adam	Huber
1B	1	9	0.001	30	20	Sigmoide	SGD	MSE
2C	3	27	0.4	50	32	Sigmoide	RMSprop	MAE
1D	1	9	0.001	30	20	Tanh	SGD	Huber
2E	3	27	0.4	50	32	Tanh	Adam	MSE
1F	1	9	0.001	30	20	ReLU	RMSprop	MSE
2G	3	27	0.4	50	32	ReLu	Adam	Huber
3C	3	27	0.001	50	32	Sigmoide	RMSprop	MAE
3E	3	27	0.001	50	32	Tanh	Adam	MSE
3G	3	27	0.001	50	32	ReLu	Adam	Huber

Con la combinaciones de la Tabla 4.9 se busca inicialmente determinar la mejor combinación de conceptos categóricos para luego optimizar los conceptos numéricos a partir de los valores iniciales establecidos.

4.4. Selección de concepto(s) del producto

En esta etapa es donde se aprecian las principales adaptaciones de la metodología UE y se va a abordar de forma diferente para cada subproblema. Para el sistema robótico se trabaja como lo establece la metodología, se definen criterios de selección basándose en las necesidades y especificaciones del proyecto, que se utilizan en las matrices de filtrado y evaluación para determinar el concepto ganador. Y para el método de aprendizaje, este paso se realiza de manera similar con la diferencia de que el filtrado y evaluación se realizan en conjunto y sin el uso de matrices para la comparación.

4.4.1. Sistema robótico

Para el filtrado y selección del conceptos del sistema robótico se utilizan los criterios que se listan a continuación, en donde se hace referencia a la necesidad que responden de la Tabla 4.1:

- Sensorización en la base: Este criterio responde a la necesidad 1.2 y requiere que los sensores estén principalmente en la base del robot.
- Compatibilidad con Python: Con este criterio se desea evaluar la compatibilidad con el lenguaje de programación establecido en la necesidad 2.4, Python.
- Disponibilidad: Se busca caracterizar la disponibilidad de la combinación de conceptos y que la combinación satisfaga la necesidad 3.
- Se mantiene dentro del presupuesto: Este criterio es bastante explicito y contempla que la combinación no incumpla con la necesidad 4.

Luego de establecer los criterios se procede con el filtrado de conceptos, arbitrariamente se define como referencia la combinación 1A y se compara con las demás, puntuando con $+$ si es mejor o $-$ si es peor, se suman los puntos y se determina la evaluación y lugar de cada combinación. En la Tabla 4.10 se muestra la matriz de filtrado para el sistema robótico.

Tabla 4.10. Matriz de filtrado para los conceptos del sistema robótico.

Criterio de selección	**1A**	**2B**	**3C**	**4D**	**5E**	**6F**
Sensorización en la base	0	0	0	0	0	−
Compatibilidad con Python	0	+	+	+	+	0
Disponibilidad	0	+	−	0	−	0
Se mantiene dentro del presupuesto	0	+	−	0	−	0
Evaluación	0	3	-1	1	-1	-1
Lugar	3	1	4	2	4	4
¿Continúa?	No	Sí	No	Mejorar	Combinar	Combinar

En la Tabla 4.10 observa claramente como el mejor concepto después del filtrado es el 2B, esto es debido a que el Dynamixel SDK ofrece la mayor compatibilidad con Python, los conceptos seleccionados para sensorización y control tienen una alta disponibilidad y pueden ser instalados en la base. Se observa potencial en algunos de los conceptos con posibilidad de mejora del concepto 4D y la combinación de los conceptos 5E y 6F. Estos resultados no son los finales puesto que falta realizar la etapa de evaluación donde la comparación se realiza de manera cuantitativa. En la Tabla 4.11 se muestran los conceptos finalistas para el sistema robótico producto de conservar la combinación 2B y la mejora y combinación de distintos conceptos.

Tabla 4.11. Conceptos finalistas para el sistema robótico.

Comb.	**Control**		**Sensorización**		
	Hardware	**Software**	**Orientación**	**Altura**	**Ctrl. Sens.**
2B	PC + USB2AX	Dynamixel SDK	IMU	IR	Arduino
7D	PC + USB2AX	R+ 3.0	IMU	Ultrasonido	Arduino
5F	PC + D2U2	Dynamixel SDK	OptiTrack	OptiTrack	PC

Con los nuevos conceptos generados en la Tabla 4.11, se busca corregir algunas de las debilidades de los conceptos previos. En el 7D se busca aumentar la disponibilidad del concepto 4D cambiando de controlador, y probar utilizando un software distinto. Con la combinación de los conceptos 5E y 6F se toman las mejores características de ambos y se unen en busca de un potencial concepto ganador.

La evaluación de los conceptos se realizó utilizando la matriz de la Tabla 4.12. Se utilizan los mismos criterios que en el filtrado pero esta vez se asigna una calificación de entre 1 y 5, donde 1 uno es el mínimo y 5 el máximo. Se determina un peso para cada criterio, basado en la importancia de la necesidad correspondiente. Al multiplicar la calificación por el peso

se tiene la evaluación de cada criterio que luego se suma para obtener la evaluación final de cada concepto y con esta seleccionar el concepto ganador que será desarrollado.

Tabla 4.12. Matriz de evaluación para los conceptos del sistema robótico.

Criterio de selección	**Peso**	**2B**		**7D**		**5F**	
		Calif.	**Eval.**	**Calif.**	**Eval.**	**Calif.**	**Eval.**
Sensorización en la base	**10 %**	5	0,5	5	0,5	1	0,1
Compatibilidad con Python	**50 %**	5	2,5	3	1,5	5	2,5
Disponibilidad	**20 %**	5	1	5	1	4	0,8
Se mantiene dentro del presupuesto	**20 %**	5	1	5	1	5	1
Evaluación (%)		100		80		88	
Lugar		1		3		2	
¿Continúa?		Desarrollar		No		No	

Con base en la evaluación realizada que se muestra en la Tabla 4.12 se concluye que el concepto ganador es el 2B, debido a:

- La gran compatibilidad que tiene el USB2AX, el Dynamixel SDK y el control de los sensores mediante Arduino con Python.
- Los sensores seleccionados son altamente compatibles con el Arduino y pueden ser instalados sobre la base sin problema alguno.
- La disponibilidad de todos los componentes es inmediata y no se requiere una mayor inversión.
- *A priori* se cumplen todos los requisitos de sensorización y control del sistema.

También se observa que los demás conceptos cuentan con muy buenas características debido a que la diferencia entre evaluaciones es relativamente poca. El concepto en segundo lugar 5F no se descarta completamente pues en caso de que el 2B tenga algún problema se puede utilizar para generar un concepto nuevo e incluso desarrollarse en un futuro si las necesidades del proyecto cambiaran.

4.4.2. Método de aprendizaje de la representación de la morfología

La selección del método de aprendizaje es diferente de como se realiza con el otro subproblema. El estudio de hiperparámetros de la RNA se realiza en la etapa de selección. Para esto se definieron las combinaciones iniciales que se muestran en la Tabla 4.9 se utilizan para

determinar de manera preliminar las mejores combinaciones, principalmente de parámetros categóricos que pueden tomar un pequeño rango de valores, y posteriormente optimizar los valores de los demás hiperparámetros. Para determinar las mejores combinaciones se utiliza una morfología fija, que se selecciona basándose en las necesidades inmediatas del proyecto, de ser utilizado para el desarrollo y pruebas del proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares, es por esto que se utiliza la morfología que se muestra en la Fig. 3.2.

Se genera un conjunto de datos representativo mediante simulación, 120 000 pares de datos y el criterio de evaluación se define con base en la métrica 9 de la Tabla 4.2, utilizando el valor ideal, un porcentaje de error $\leq 0{,}005$ en cada salida. El porcentaje de error (%e) se calcula utilizando el MSE de cada salida y el rango de valores de la salida correspondiente (Ec. 4.1).

$$Rango = y_{max} + y_{min} \tag{4.1}$$

La Ec. 4.2 muestra el calculo del porcentaje de error que se utiliza para comparar el error entre las distintas salidas. Lo que se busca con este error es determinar de alguna forma que tan representativo es este error en términos relativos a cada salida para poder compararlas, debido a que las salidas varían tanto en el orden de magnitud como en las unidades.

$$\%e = \frac{MSE}{Rango} * 100 \tag{4.2}$$

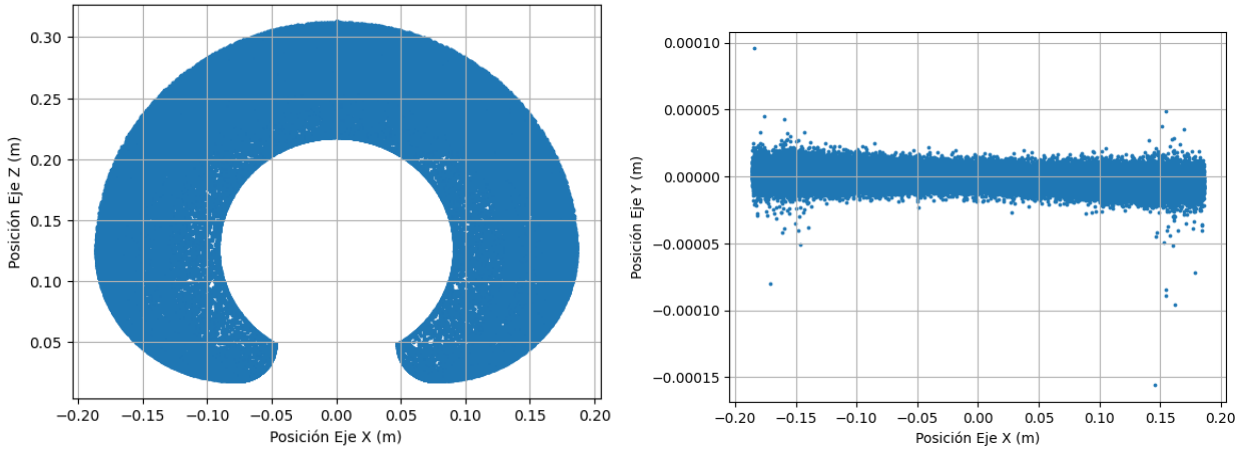
De esta forma utilizando el %e como criterio de evaluación, se procede de la siguiente forma: para cada una de las combinaciones iniciales se realiza el entrenamiento hasta un máximo de 4 replicas para obtener un promedio, en caso de que el error sea muy alto o el entrenamiento produzca resultados indeseados se realiza una única replica. De esta forma se obtiene el %e para las salidas de cada combinación. En el Anexo B.1 se puede acceder a los resultados obtenidos durante los entrenamientos. De esto se determina que las dos mejores combinaciones son la 3E y 3G, que de seis salidas, cada una de ellas presentó el menor error en tres de estas salidas. Debido a esto se generan dos combinaciones adicionales, 3H y 3I, y se generan al cambiar la función de perdida de la 3E por la de 3G y viceversa. En la Tabla 4.13 se muestran las mejores combinaciones iniciales y las nuevas.

Tabla 4.13. Combinaciones intermedias de conceptos para el método de aprendizaje.

Comb.	**Capas**	**NA**	**T. Aprend.**	***Epochs***	***B. Size***	**F. Act.**	**Opt.**	**F. Perd.**
3E	3	27	0.001	50	32	Tanh	Adam	MSE
3G	3	27	0.001	50	32	ReLu	Adam	Huber
3H	3	27	0.001	50	32	Tanh	Adam	Huber
3I	3	27	0.001	50	32	ReLu	Adam	MSE

Se procede a realizar el entrenamiento de las nuevas combinaciones, se realizan 4 replicas para obtener resultados más representativos. Destacar que estas son comparaciones preliminares antes de realizar la optimización de los hiperparámetros.

Es importante aclarar que para estas comparaciones se desprecia el error en el eje Y, esto debido a que si se observa con atención la Fig. 3.2, la morfología del robot es planar, es decir, se mueve únicamente sobre dos ejes, X y Z. Debido a esto, los datos obtenidos del eje Y tienen un comportamiento en simulación semejante al ruido y producen valores al menos 2 000 veces menores que los demás ejes, aproximadamente cero; por lo que la RNA al intentar aprender el comportamiento del eje Y produce un error muy alto, esto sucede únicamente para morfologías planares donde un eje se mantiene muy cerca de 0. En la Fig. 4.5 se muestra el conjunto de datos para entrenamiento graficado, el plano XZ (Fig. 4.5a) que representa el área de trabajo del robot, y el plano XY (Fig. 4.5b) donde se observa que el movimiento sobre el eje Y es mínimo y aproximadamente constante contrario a los demás ejes.



(a) Plano XZ (b) Plano XY

Figura 4.5. Conjunto de datos para optimización del método de aprendizaje.

En la Tabla 4.14 se muestran los resultados obtenidos, se determina que las mejores combinaciones son 3E y 3H, y entre estas la única diferencia es la función de perdida.

Tabla 4.14. Porcentaje de error de las combinaciones intermedias para el método de aprendizaje.

Combinación	**%e pos. X**	**%e pos. Z**	**%e pos. A1**	**%e pos. A2**	**%e pos. A3**	**%e prom.**
3E	0,007	0,009	0,006	0,006	0,007	0,007
3G	0,020	0,021	0,004	0,006	0,006	0,012
3H	0,007	0,009	0,006	0,006	0,005	0,007
3I	0,021	0,030	0,006	0,005	0,006	0,013
Valor mínimo	0,007	0,009	0,004	0,005	0,005	0,007
Mejor comb.	3E	3E	3G	3I	3H	3H

Al realizar una comparación entre ambas combinaciones se observa que para el %e en los ejes X y Z 3H produce un error un 3 % mayor que el resultante con 3E, sin embargo, produce un %e en la articulación 3 y promedio aproximadamente un 13 % menor que al utilizar 3E. Con base en estos resultados, se determina que la combinación categórica H es la óptima para este problema, y se procede con la optimización del resto de los hiperparámetros. Con el fin de realizar la optimización del resto de hiperparametros utilizando el criterio y recomendaciones de un experto se generan cuatro combinaciones numéricas adiciones y en conjunto con la combinación categórica H se generan las combinaciones finales de la Tabla 4.15.

Tabla 4.15. Conceptos finalistas para el método de aprendizaje.

Comb.	**Capas**	**NA**	**T. Aprend.**	***Epochs***	***B. Size***	**F. Act.**	**Opt.**	**F. Perd.**
4H	3	27	0.001	100	21	Tanh	Adam	Huber
5H	3	27	0.001	500	42	Tanh	Adam	Huber
6H	3	27	0.001	500	21	Tanh	Adam	Huber
7H	3	27	0.001	100	42	Tanh	Adam	Huber

Debido a que los conceptos restantes son los finalistas, la comparación en este caso debe ser más ardua, por esto se realizan 10 replicas por combinación, en lugar de las cuatro que se realizaron previamente. En las gráficas de las Figs. 4.6 y 4.7 se muestra el porcentaje de error de las diferentes salidas para cada concepto de la Tabla 4.15. En cada grafica se muestra una barra por salida y el conjunto de barras por combinación, de manera que se puede realizar la comparación entre conceptos de manera mas sencilla. Claramente se observa como el error que presenta el concepto 5H para todas las salidas es menor que los demás conceptos.

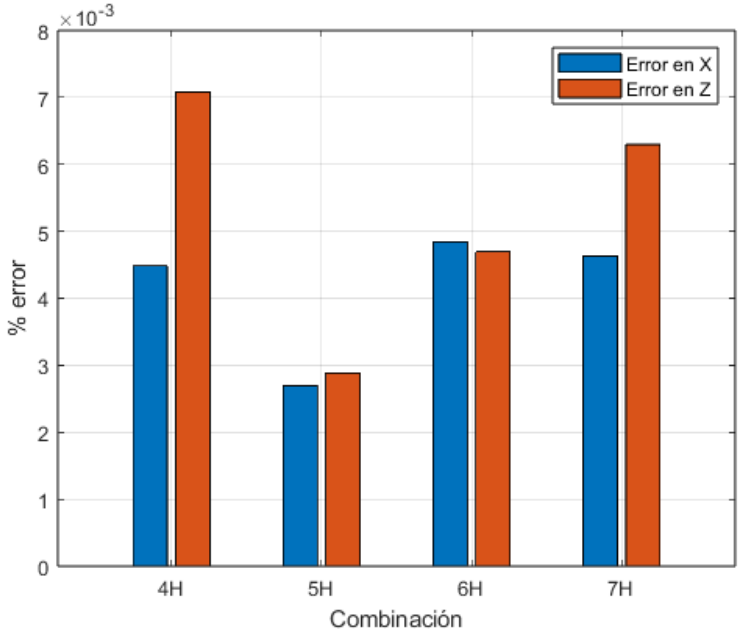


Figura 4.6. Porcentaje de error de la posición en los ejes X y Z para combinaciones finalistas.

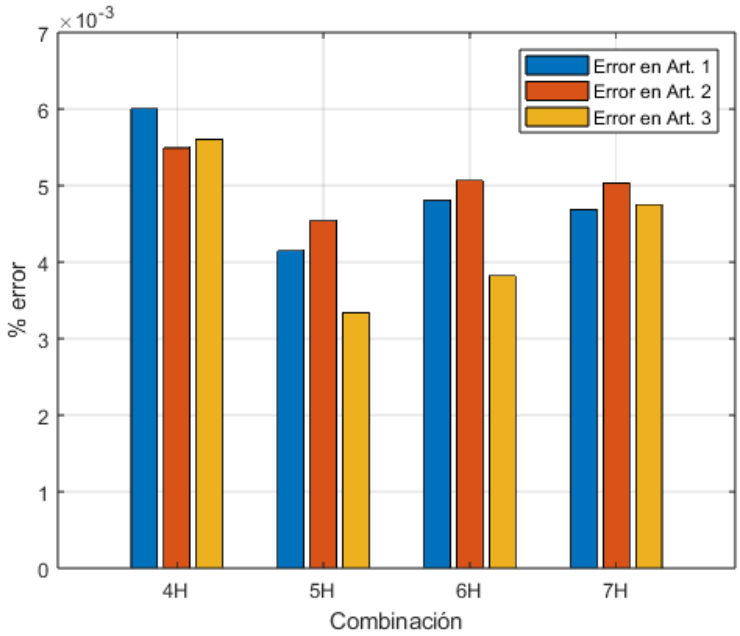


Figura 4.7. Porcentaje de error de la posición de las articulaciones para combinaciones finalistas.

Adicionalmente, se toma el %e promedio de cada salida y se compara con el error promedio

de la combinación inicial 3H de manera que se determine el decremento o aumento del error y determinar el concepto ganador, un numero positivo indica que el error del concepto es menor y uno negativo que el error aumenta. En la Tabla 4.16 se muestran los resultados obtenidos luego de las 10 replicas con cada concepto.

Tabla 4.16. Decremento en el porcentaje de error de las combinaciones finales con respecto a la combinación 3H para el método de aprendizaje.

Combinación	**Decr. X**	**Decr. Z**	**Decr. A1**	**Decr. A2**	**Decr. A3**	**Decr. prom.**
4H	39,33	22,54	6,80	3,83	-23,28	9,85
5H	63,57	68,42	35,64	20,56	26,61	42,96
6H	34,45	48,64	25,44	11,30	16,01	27,17
7H	37,31	31,11	27,25	11,88	-4,56	20,60
Valor máximo	63,57	68,42	35,64	20,56	26,61	42,96
Mejor modelo	5H	5H	5H	5H	5H	5H

De los resultados obtenidos en la Tabla 4.16 se determina que el mejor concepto es el 5H con una disminución del %e de más del 40 %. Por lo tanto, se selecciona 5H como el concepto ganador y se realizan 20 replicas adicionales para tener un total de 30 replicas, una muestra representativa que compruebe que se cumple la métrica correspondiente. En la gráfica de la Fig. 4.8 se muestra el error de entrenamiento y validación en función de las *epochs*, en esta grafica se recortan las primeras 20 *epochs* puesto que el error inicia en un valor muy alto y conforme avanza disminuye de forma significativa, por lo tanto para poder apreciar correctamente el efecto del aumento de la cantidad de *epochs* en el error, se grafica de esta forma. El comportamiento mostrado del error en la Fig. 4.8 justifica el valor utilizado para este hiperparámetro.

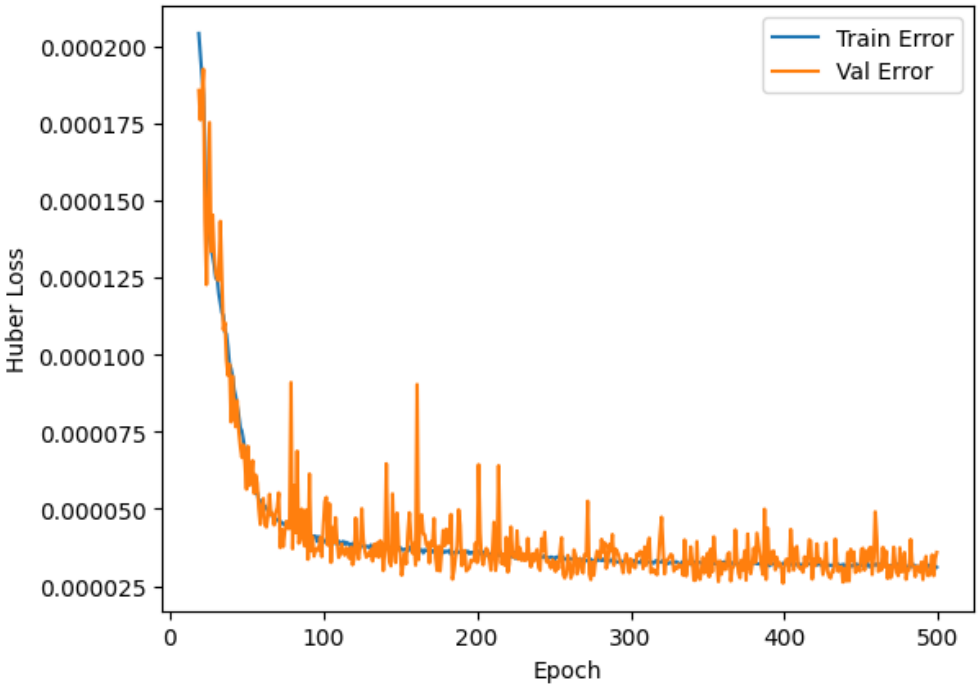


Figura 4.8. Gráfica de error de entrenamiento y validación en el entrenamiento del concepto ganador.

El %e promedio luego de 30 replicas para cada salida se muestra en la Tabla 4.17, se observa que todas y cada una de las salidas presentan un % de error menor a 0,005.

Tabla 4.17. Porcentaje de error del concepto ganador para el método de aprendizaje.

Salida	**% de error**
Pos. eje X	0,003
Pos. eje Z	0,003
Pos. articulación 1	0,004
Pos. articulación 2	0,004
Pos. articulación 3	0,003
Promedio	0,003

Los resultados obtenidos de la ejecución de todas las replicas realizadas son accesibles mediante el Anexo B.1, incluye los datos en bruto de error, salidas, gráficas de entrenamiento, comparaciones y demás para cada combinación y replica.

4.5. Probar concepto(s) del producto

En esta etapa se prueban los conceptos finalistas para cada subproblema de manera independiente. Para el sistema robótico el concepto 2B de la Tabla 4.11, y el concepto 5H de la Tabla 4.15 para el método de aprendizaje. Se plantean las pruebas necesarias para comprobar el cumplimiento de los objetivos y especificaciones finales del proyecto establecidas en la Tabla 4.2.

4.5.1. Sistema robótico

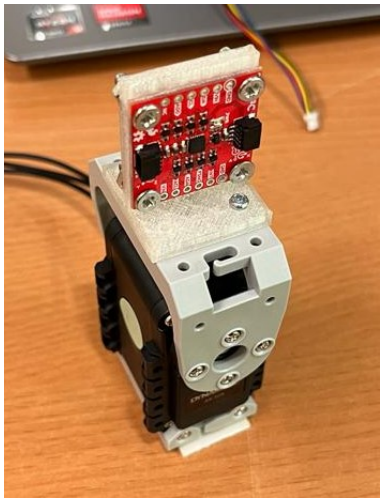
Como se describe en el diagrama de la Fig. 4.3, el sistema robótico se divide en control y sensorización, por lo tanto se deben probar ambos conceptos. Debido a la evolución del proyecto y las especificaciones, finalmente se implementa un manipulador de 3 módulos utilizando la base diseñada que cuenta con los sensores instalados, es por esto que las pruebas de control y sensorización se realizan de forma independiente. Las pruebas de control están enfocadas en probar el funcionamiento del sistema robótico, su base, módulos y método de control. Por otro lado, las pruebas de sensorización tienen como objetivo probar la conexión y funcionamiento de los sensores utilizados.

Sensorización

En las pruebas de sensorización se debe probar el sensor IR y la IMU, en conjunto con su método control, un Arduino MEGA 2560 rev3.

La prueba de la IMU consiste en caracterizar el sensor para obtener la precisión y exactitud de cada eje del giroscopio, al trabajar con un rango de rango de ± 250 dps. Debido a que se realiza de forma experimental, se debe fabricar un montaje que permita realizar mediciones y establecer una referencia para la comparación. El montaje experimental se diseñó en el software de CAD, *Fusion360* y se fabricó mediante FDM, se muestra en la Fig. 4.9. La caracterización del eje X y Y se realiza utilizando el montaje que se observa en la Fig. 4.9a debido a que para cambiar de eje solo se debe girar 90° sobre el eje Z de la IMU y utilizar la misma pieza. Para el eje Z se utiliza una pieza distinta, Fig. 4.9b.

(a) Eje X y Y



(b) Eje Z

Figura 4.9. Montaje experimental de la IMU.

Como se observa en la Fig. 4.9 las piezas del montaje se fijan sobre un actuador AX-12A que se utiliza en los módulos, para aprovechar su capacidad de regular la velocidad. De forma que el actuador se utiliza como referencia y para girar a velocidad constante la IMU sobre el eje que se desea caracterizar. Los factores de influencia en esta prueba se muestran en la Tabla 3.1: el eje que se caracteriza y la velocidad de giro, en dps. Se utilizan 17 velocidades diferentes. Se van a realizar 30 mediciones para cada combinación de eje y velocidad, cantidad de repeticiones suficiente para obtener una muestra representativa [22]. Debido a que el comportamiento de la velocidad del actuador no es un impulso, sino un aumento lineal hasta llegar a la velocidad máxima, la medición se realiza lo más cercano a la mitad del recorrido donde se tiene la velocidad máxima. El movimiento que se realiza es de 180° de manera que se obtienen mediciones del sensor en ambos sentidos, reloj y contra-reloj. Los datos obtenidos de la IMU se comparan con los medidos por el actuador y se determina la exactitud y precisión del sensor, utilizando error promedio y la desviación estándar promedio de cada eje respectivamente. A razón de que esta es una prueba funcional, el éxito de esta prueba se define como la capacidad de caracterizar el sensor de manera experimental, lo que comprueba la correcta conexión e instalación del sensor.

La prueba del sensor IR se enfoca en obtener la curva característica de este sensor, es decir la función que describe el comportamiento de la tensión de salida del sensor (V) en función de la distancia medida (mm). Nuevamente, para obtener datos confiables y validos, se fabrica el montaje experimental de la Fig. 4.10 en conjunto con el marco calibrado para las mediciones que se muestra en la Fig. 4.11. Las lineas que se observan en el marco están

separadas a distancias arbitrarias y diferentes de manera que se obtengan variaciones de distancia distintas entre medidas.

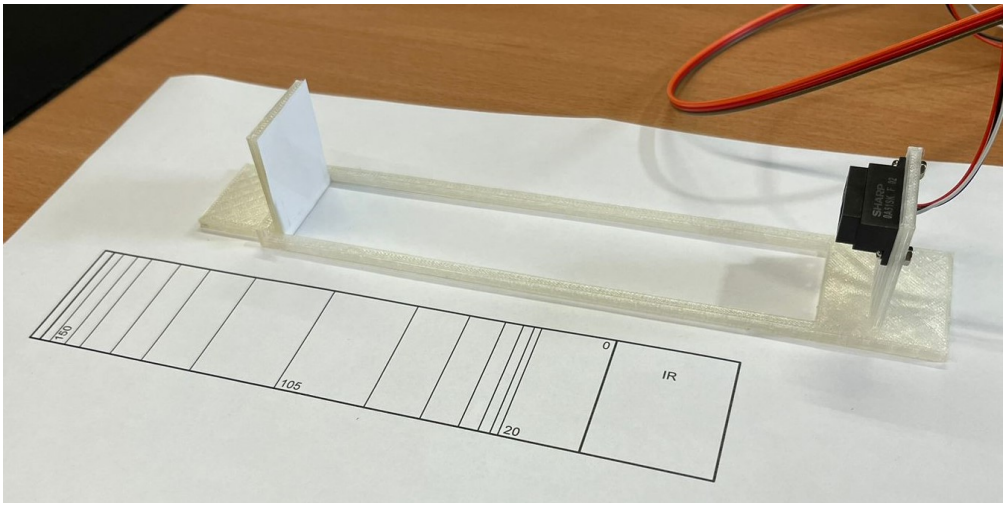


Figura 4.10. Montaje experimental del sensor IR.

Las piezas se diseñan en *Fusion360* y se fabrican mediante FDM. Para comprobar las medidas del marco se diseña en *AutoCAD* y se verifica con un *Vernier*. Las piezas del montaje están diseñadas para que la placa de medición pueda desplazarse y se mantenga paralela al sensor IR, y utilizando el marco de referencia se puedan repetir con facilidad y confianza las mediciones. El montaje final es similar al mostrado con la diferencia que las piezas se alinean con el marco. Se fija un cuadrado de papel blanco en la placa móvil de manera que el material de fabricación no afecte la reflexión del haz de luz.

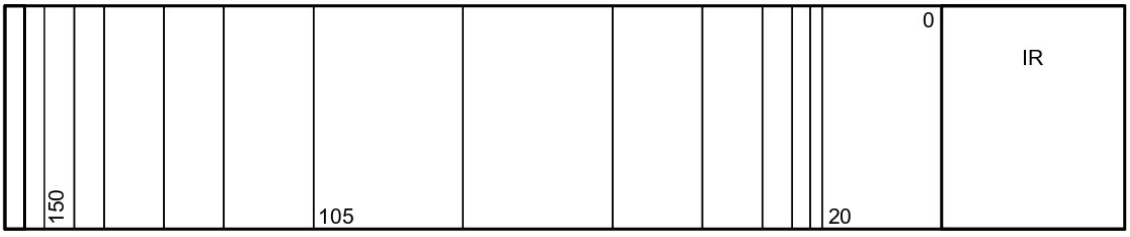


Figura 4.11. Marco para montaje experimental del sensor IR.

Los factores de influencia de esta prueba se muestran en la Tabla 3.2 y consiste únicamente en la distancia de la placa de medición al sensor. Se prueba con 12 distancias diferentes,

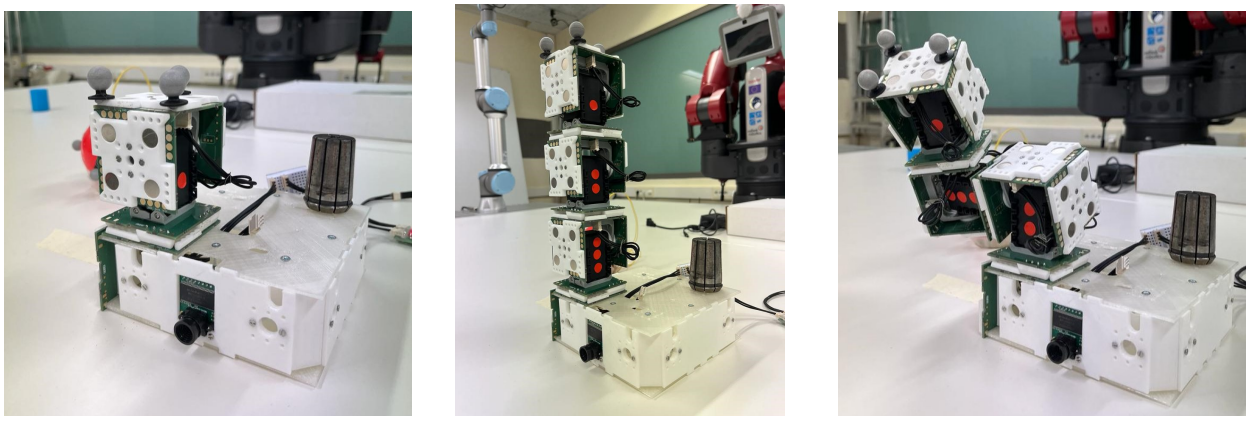
realizando 30 repeticiones por distancia para obtener un conjunto de datos representativo.

Control y módulos

El objetivo de estas pruebas es verificar al implementación del control, base y módulos. Debido a que el control se realiza utilizando Python en conjunto con el Dynamixel SDK se desarrolla una librería con las funciones para acceder a la memoria del actuador, que se muestra en la Tabla 2.1, y para el control del movimiento del sistema. Estas pruebas consisten en realizar la conexión de los módulos a la base, en diferentes configuraciones y cantidad de módulos, para ejecutar las funciones implementadas en la librería. Los factores de influencia de esta prueba son la cantidad de módulos, la configuración de los módulos y las funciones de la librería, se muestran en la Tabla 3.3. Las configuraciones están escritas de izquierda a derecha, de manera que indica el orden de los módulos de abajo hacia arriba, y las letras indican el punto de conexión al modulo anterior, esto debido a que cada modulo cuenta con tres posibles puntos de conexión diferentes: arriba (A), izquierda (I) y derecha (D). Se definen cinco combinaciones de cantidad de módulos y configuración:

- tres combinaciones donde se prueban las funciones en cada modulo individual,
- y dos combinaciones con tres módulos para las configuraciones A y B para probar todas las funciones.

En la Figura 4.12 se muestran las configuraciones utilizadas para las pruebas de control. La Fig. 4.12a ejemplifica la configuración para un módulo individual, específicamente el módulo 1, y en las Figs. 4.12b y 4.12c se muestran las configuraciones de tres módulos.



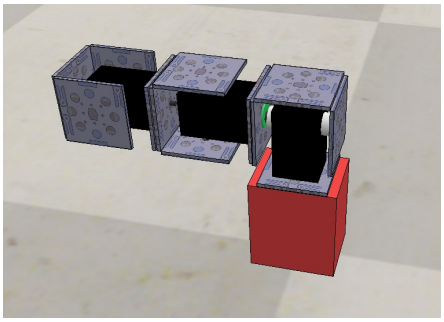
(a) 1 módulo (b) 3 módulos (A) (c) 3 módulos (B)

Figura 4.12. Combinaciones para las pruebas del sistema robótico.

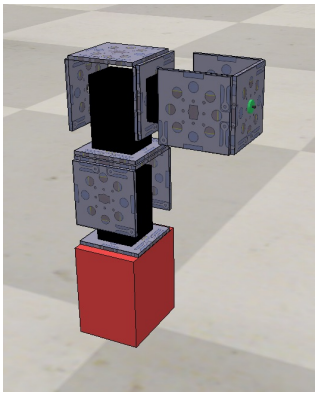
Para las combinaciones con un modulo independiente se realizan 20 repeticiones por combinación, para un total de 60 repeticiones por función, y para las configuraciones de tres módulos se prueba cada función 30 veces por cada una, un total de 60 repeticiones. Se establece la cantidad de repeticiones de tal forma que la muestra obtenida sea representativa, probando cada función y modulo un total de 120 veces, cuatro veces la cantidad mínima recomendada [22]; esta cantidad para probar múltiples configuraciones diferentes con cada módulo, verificando el funcionamiento de la conexión, comunicación y control de los módulos.

4.5.2. Método de aprendizaje de la representación de la morfología

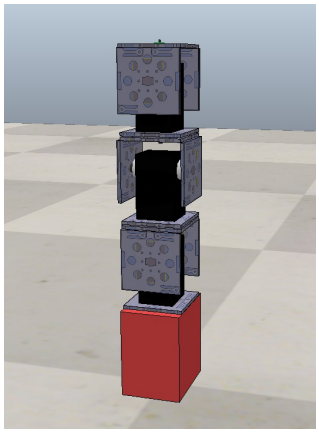
Luego de la optimización de la RNA seleccionada para que en todas sus salidas presente un porcentaje de error $\leq 0{,}005\,\%$ para la morfología de la Fig. 3.2. Se busca probar que sea capaz de aprender el modelo de mundo para otras morfologías con un error $\leq 0{,}01\,\%$ en todas las salidas, idealmente $\leq 0{,}005\,\%$. En esta comprobación se van a generar datos para las cinco morfologías que se muestran en la Fig. 4.13, generadas de manera aleatoria.



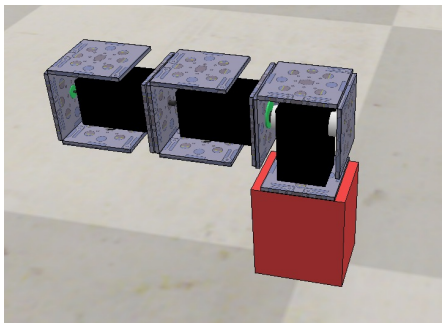
(a) modular03a



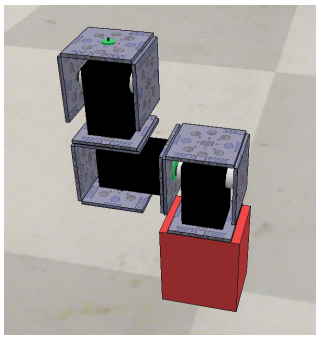
(b) modular03b



(c) modular03c



(d) modular03d



(e) modular03e

Figura 4.13. Morfologías para las pruebas del método de aprendizaje.

La generación de datos se realiza de la misma manera que se hizo para lo optimización de la RNA. Para cada una de las morfologías mostradas en la Fig. 4.13 se genera un conjunto con 100 000 datos para el entrenamiento. Se van a realizar seis replicas por cada morfología, para un total de 30 replicas y se calcula el %e promedio las salidas de cada morfología. Se considera exitosa esta prueba si se obtiene un % de error en todas las salidas $\leq 0{,}01\,\%$ para

un 90 % de las morfologías, 4, 5 morfologías, es decir, cuatro morfologías y al menos tres replicas de una quinta morfología.

5 Procedimientos

En este capítulo se desarrolla el procedimiento de la implementación de los conceptos ganadores. Se explica el proceso mediante el cual se diseñó y fabricó la base del sistema, se realizó la implementación de la librería para el control de los actuadores y sensores, al igual que el desarrollo de los códigos para la generación de datos, aprendizaje y entrenamiento de la RNA. El repositorio con todos los códigos utilizados es accesible desde el vinculo en el Anexo A.2. Asimismo, los modelos de las piezas de la base son accesibles mediante el vinculo del Anexo A.3.

5.1. Generación de datos

Previo a la generación de los datos, se realizó un estudio y pruebas en simulación para determinar cuales percepciones e información es necesaria para el entrenamiento del modelo. Inicialmente se realizo una lluvia de ideas donde se consideraron percepciones como: posición angular, torque y velocidad de los actuadores; posición espacial y orientación de la base. Seguido, se realizaron simulaciones donde se recolectaron datos para cada una de las percepciones y se determinó utilizar únicamente la posición angular de los actuadores, posición en el eje Z y orientación de la base, para un cuadrúpedo. Debido a que las necesidades del proyecto cambiaron, para poder realizar pruebas al proceso deliberativo de la Fig. 2.2 se enfocó el resto del desarrollo en un robot modular con la morfología que se muestra en la Fig. 2.3, y se utilizó posición angular de los actuadores y la posición del extremo de la cadena de módulos, la cual en la realidad se obtiene utilizando el sistema OptiTrack que fue habilitado con este fin, durante el desarrollo del proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares.

La generación de datos para el aprendizaje se debe realizar utilizando CoppeliaSim en conjunto con Python, por lo tanto se debe desarrollar un código capaz de controlar el simulador y almacenar los datos de manera eficiente para que puedan ser cargados para el entrenamiento de la RNA. Se toman en cuenta las limitaciones del robot real en cuanto a movimiento puesto que debe ser lo más cercano a la realidad. El entrenamiento se realiza encadenando múltiples movimientos de $-90°$ a $90°$, determinados de forma aleatoria, de forma que el robot real puede realizar pasos $\leq 90°$. Estas cadenas de movimientos se denominan secuencias, a su vez, los movimientos se denominan pasos. En el diagrama de flujo de la Fig. 5.1 se describe el funcionamiento del código para la generación de datos.

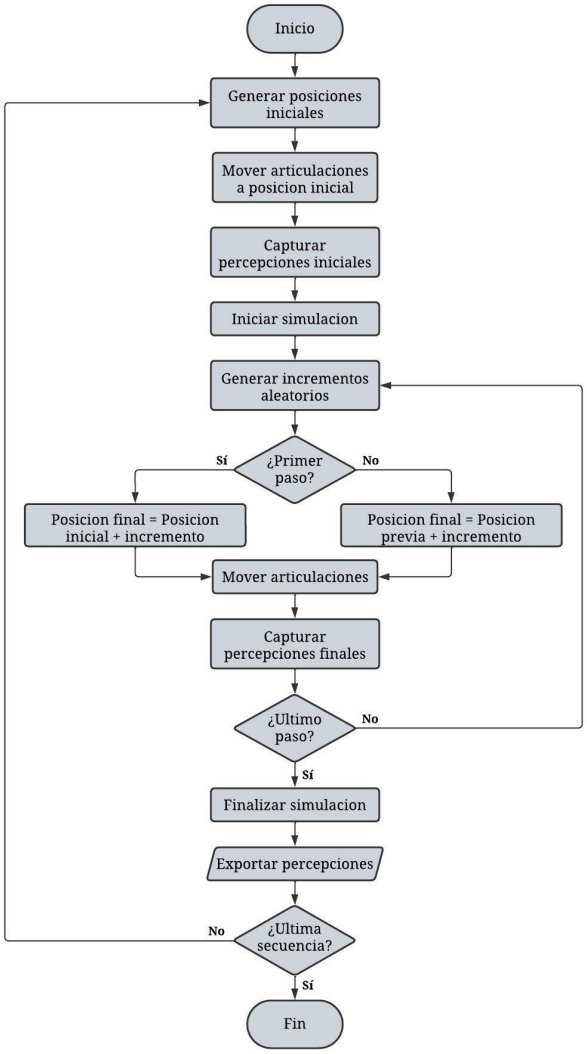


Figura 5.1. Diagrama de flujo del programa para la generación de datos para el aprendizaje.

5.2. Adaptación de la base del sistema robótico

La versión básica del sistema robótico EMERGE incluía una versión inicial de la base, la cual contaba con pocas conexiones para los módulos, de uno a dos conectores en cada lateral de la base, y carecía de sensorización. Es por esto, que debido a las necesidades de este proyecto se modificó la base para poder conectar hasta dos módulos en cada lateral y hasta cuatro módulos tanto en la parte superior como inferior de la base, así como la capacidad de contar con una cámara, una IMU y un sensor IR, en conjunto con un Arduino para su control. Esto conllevó a la modificación de todas las piezas de la base y el diseño de dos piezas adicionales. En esta sección se describen los cambios realizados a cada pieza y se muestran los resultados finales.

En la Fig. 5.2 se muestra la placa lateral de la base inicial. A esta no se le realizaron grandes cambios y se conservó para utilizarse en los laterales.

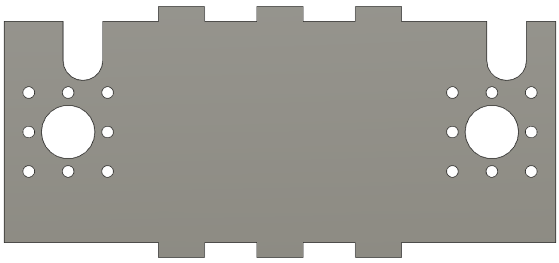


Figura 5.2. Placa lateral de la base.

Con base en la placa lateral, se crearon dos placas adicionales, una frontal y una trasera. La placa frontal se muestra en la Fig. 5.3, para el diseño de esta placa se tomo en cuenta la posibilidad de instalar una cámara para aplicaciones futuras del robot, para esto se realizó un agujero rectangular por donde sale el lente de la cámara y dos agujeros adicionales para fijarla con los tornillos [23].

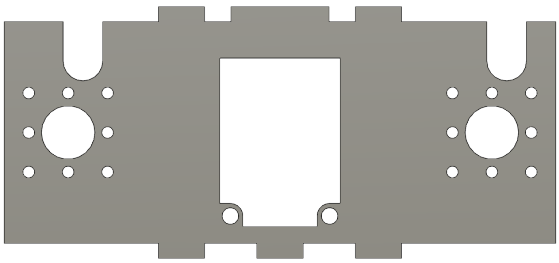


Figura 5.3. Placa frontal de la base.

La Fig. 5.4 muestra la placa trasera de la base, la única modificación que se le realizó a la placa lateral en este caso fue un agujero por el cual se introduce el cable de alimentación del robot.

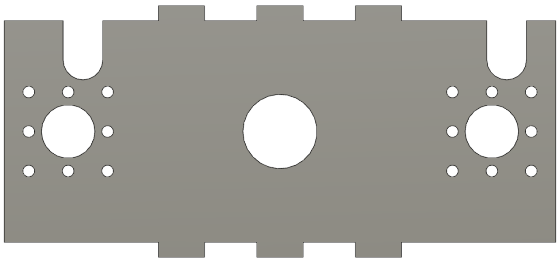


Figura 5.4. Placa trasera de la base.

Como se menciona anteriormente, las placas superior e inferior de la base no tienen la capacidad de fijar conectores para los módulos por lo que se deben modificar. A la placa superior se le añaden los agujeros necesarios para fijar el conector de los módulos, además de un agujero adicional por cada conector para que se pueda conectar el cableado del conector. También se considera la instalación del Arduino en la placa superior, por lo que se hacen agujeros que permitan fijar el Arduino en la parte superior de la placa. Puesto que el Arduino

esta en la parte superior se deben sacar los cables de los sensores, para esto se realiza un agujero adicional en la parte frontal de la placa. En el borde frontal de la placa superior se cambia la forma del borde para dejar espacio para los pines de la cámara y el cableado de esta. La versión final de la placa trasera se presenta en la Fig. 5.5 [24].

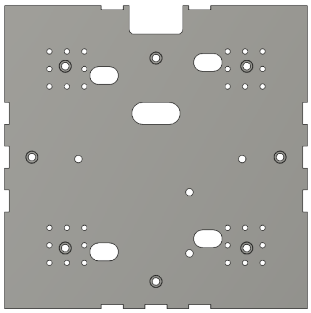


Figura 5.5. Placa superior de la base.

Similar a la placa superior, a la placa inferior se le añaden los mismos agujeros para poner fijar los conectores de los módulos. Además se le realiza un agujero rectangular, centrado en la pieza para que no interfiera con las mediciones del IR. En la Fig. 5.6 se muestra la placa inferior con todos los cambios.

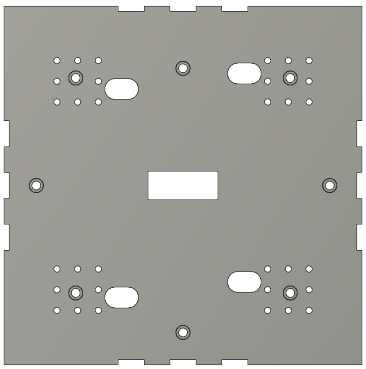


Figura 5.6. Placa inferior de la base.

Para la instalación de los sensores se debe considerar que para obtener las mejores medi-

ciones de orientación y altura, lo óptimo es instalar ambos sensores en el centro de la base, o lo más cerca a este. Para esto se crean dos placas adicionales, una superior y otra inferior, que se fijan a la placa superior e inferior respectivamente mediante cuatro agujeros que se hacen en la parte mas exterior de las placas. Estos agujeros también se hacen sobre las placas superior e inferior.

Debido a que el sensor IR mide la distancia a la superficie, se instala en la placa adicional inferior, y la IMU en la placa adicional superior. Las placas adicionales están separadas aproximadamente a 1 cm de la placa externa que van fijados, por lo que pueden interfieren con el cableado de los conectores de los módulos. Para resolver esto se modifican los bordes de las placas adicionales de manera que calcen con los agujeros para el cableado de los conectores en la placa externa correspondiente. De la misma forma, en cada placa se considera dejar espacio para no interferir con la instalación de la cámara por lo que se observa una hendidura rectangular en el borde.

La placa adicional superior que se muestra en la Fig. 5.7. Además de lo descrito previamente, esta placa tiene los agujeros necesarios para fijar la IMU correctamente y un agujero adicional para el cableado puesto que esta en medio del sensor IR y el Arduino. Las dimensiones de la IMU fueron tomadas de forma experimental utilizando un *Vernier*.

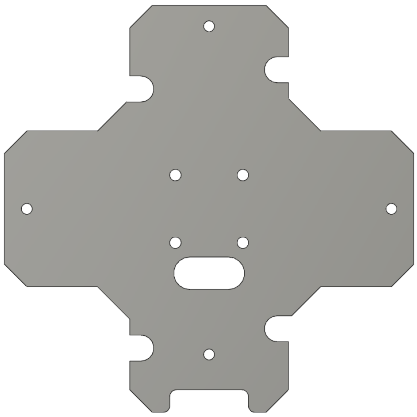


Figura 5.7. Placa adicional superior de la base.

En la Fig. 5.8 se presenta la placa adicional inferior, esta cuenta únicamente con los agujeros necesarios para la instalación del sensor IR, y para ser fijada a la placa inferior [25].

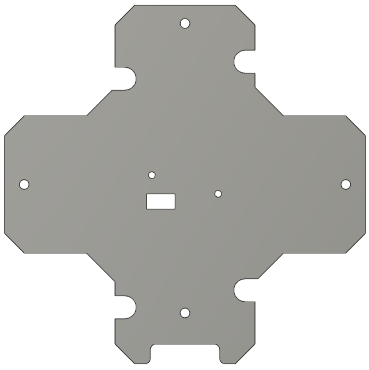


Figura 5.8. Placa adicional inferior de la base.

Las placas externas de la base se unen mediante el uso de una unión o *bracket* que se ubica en cada esquina de la base y donde se fija cada placa. El diseño inicial de esta pieza no contemplaba la instalación de conectores adicionales en la parte superior e inferior de la base, por lo que interfiere con la instalación, para esto se le realizaron dos modificaciones, se realizan dos agujeros en la cara de la diagonal y a su vez se hace una hendidura en la parte superior e inferior de la pieza, donde se fija, esto para que no haya problema al fijar los conectores. En la Fig. 5.9 se observa la unión.

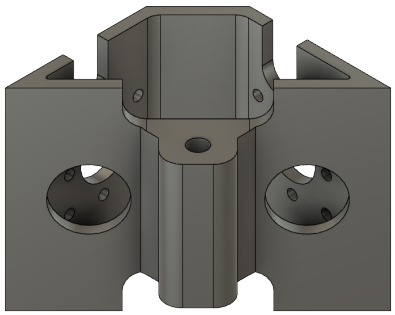


Figura 5.9. Unión para las placas de la base.

Finalmente, en la Fig. 5.10 se muestra el diseño de la base ensamblada con todos los posibles conectores instalados, a excepción de dos de los conectores superiores puesto que no

es posible utilizar estos conectores y el Arduino al mismo tiempo, lo cual de todas formas no es necesario para el cliente.

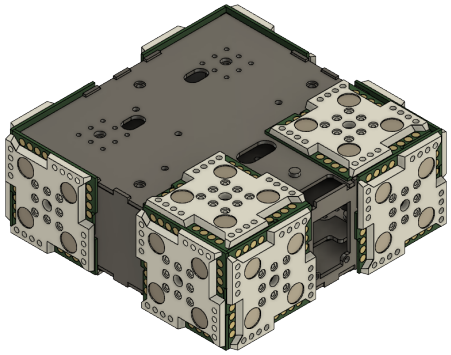


Figura 5.10. Base del sistema con los conectores instalados.

Todas las piezas fueron diseñadas utilizando *Fusion360* y fabricadas mediante Impresión 3D.

5.3. Librería de control

El sistema robótico utiliza los actuadores Dynamixel AX-12A y estos deben ser controlados mediante el lenguaje de programación Python, por lo que el método de software seleccionado es el Dynamixel SDK, una librería que accede a la memoria de los actuadores mediante funciones que administran la comunicación mediante paquetes dependiendo de la cantidad de bytes de cada registro. Como se menciona, las funciones del SDK requieren conocer tanto el tamaño del paquete que se desea enviar y la dirección de memoria, lo cual no es practico, es por esto que se implementa una librería para el control de los actuadores. En la librería se incluyen múltiples archivos con las funciones y clases necesarias para almacenar los datos del actuador. Se inicia con un archivo donde se almacenan las direcciones en memoria y tamaño en bytes de cada registro, y los parámetros necesarios para la comunicación con los actuadores y sensores del sistema.

Luego se desarrolla un archivo adicional, en el cual se definen las funciones para el control de los motores utilizando las funciones incluidas en el Dynamixel SDK. Se crea una función para poder acceder fácilmente a cada uno los registros que se necesitan leer o escribir. A cada

función se le asigno un nombre similar a las funciones utilizadas para controlar la simulación, el objetivo final que utilizando esta librería se pueda controlar el robot real o la simulación con un mismo programa. Las funciones creadas permiten:

- Activar o desactivar el torque en los actuadores.
- Cambiar el rango de movimiento de los actuadores.
- Obtener la posición actual de los actuadores
- Mover los actuadores a una posición dada.
- Cambiar la velocidad de movimiento.
- Cambiar el perfil de velocidad de movimiento, entre un valor constante o un comportamiento trapezoidal.
- Llevar los actuadores a su posición inicial.
- Identificar si un módulo específico esta conectado.
- Identificar la lista de módulos conectados.
- Iniciar y terminar la conexión con los sensores y el Arduino.
- Iniciar y terminar la conexión con los módulos.
- Cargar para los módulos, que ejecuta múltiples de las funcionalidades descritas previamente.
- Obtener la medición del sensor IR.
- Obtener la orientación de la IMU.

Las funciones relacionadas con el control de los sensores mediante el Arduino se implementan en un archivo adicional que administra la conexión al Arduino y la comunicación con lo sensores, en conjunto con un programa para Arduino que controla directamente los sensores. El código cargado en el Arduino aprovecha la librería de *SparkFun* para la IMU seleccionada, con las modificaciones pertinentes para obtener únicamente los datos del giroscopio [26].

Adicionalmente, se desarrolla en otro archivo se implementan funciones equivalentes a las utilizadas para el control del robot real, para controlar la simulación, de manera que con un solo código y utilizando un parámetro se pueda controlar lo que se necesite, simulación o robot real.

Finalmente, se crea un archivo con una función que dependiendo del valor de uno de los argumentos, que determina la plataforma con la que se trabaja, simulación o robot real, devuelve unas u otras funciones, lo que permite que se el código funcione con ambos y requiera mínimos cambios.

6 Resultados y Análisis

En este capítulo se presentan los resultados obtenidos durante la ejecución de este proyecto y el proceso de diseño, en conjunto con el análisis de dichos resultados. Principalmente se presentan los resultados obtenidos en el ultimo paso del proceso de diseño, las pruebas de concepto. Además, se incluye el análisis económico de este proyecto contrastando los recursos generados e invertidos.

6.1. Sistema robótico

Las pruebas del sistema robótico consistieron en caracterizar los sensores utilizados: el sensor IR y la IMU, además de verificar la funcionalidad del método de control, módulos y base implementados.

En la Fig. 6.1 se muestra la curva característica del sensor IR, que muestra un comportamiento similar al esperado, con una tensión de salida (V_{out}) mayor conforme menor sea la distancia al sensor. La tensión de salida va de 0 a 2 (V) para distancias de 20 a 150 (mm). Para determina la curva se realizan 30 mediciones para 12 distancias diferentes.

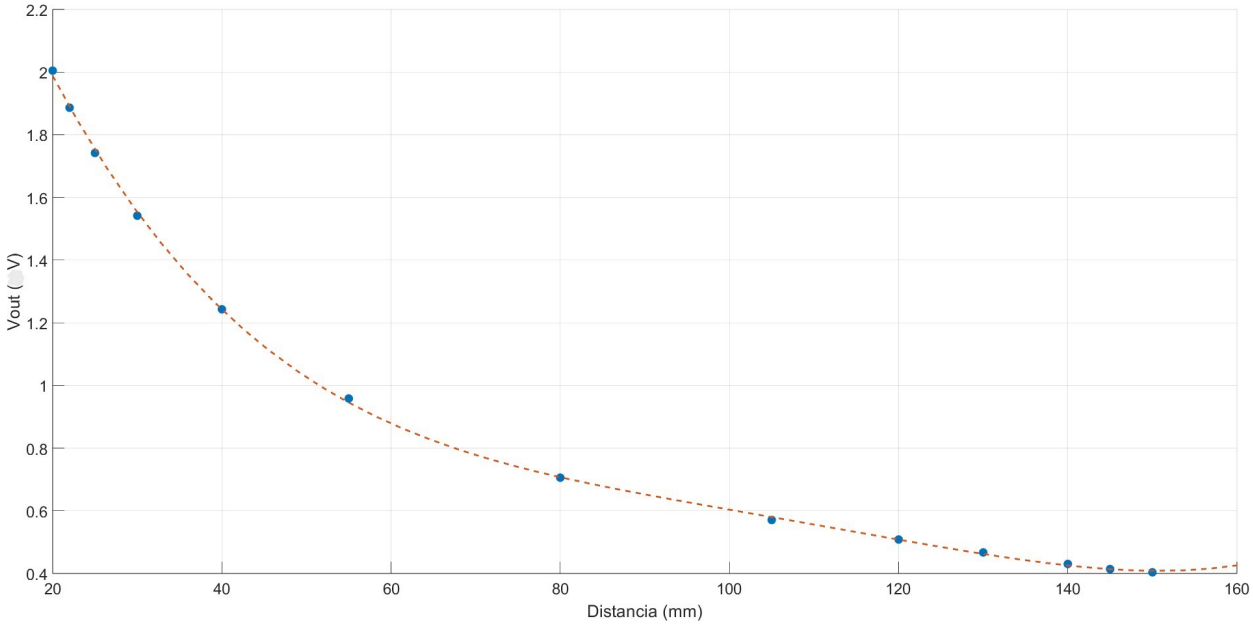


Figura 6.1. Curva característica del sensor IR.

En la Ecuación 6.1 se muestra la función que describe la curva característica del IR, Fig. 6.1.

$$f(x) = 2 * 10^{-8}\, x^4 - 7 * 10^{-6}\, x^3 + 1{,}2 * 10^{-3}\, x^2 - 0{,}09\, x + 3{,}37 \tag{6.1}$$

Los resultados mostrados en la 6.1 comprueban la correcta implementación del sensor IR y en conjunto con la función de la Ecuación 6.1 se implementa la función final de manera que los datos obtenidos del sensor se puedan interpretar y obtener la distancia.

Para caracterizar el giroscopio de la IMU se determina de forma experimental la precisión y exactitud de este se realiza un montaje experimental y se gira el sensor a distintas velocidades utilizando un AX-12A, se miden 17 velocidades diferentes con 30 repeticiones por velocidad, una vez por eje del giroscopio. Las velocidades medidas por el sensor se comparan con la velocidad medida utilizando el actuador, para determinar el error promedio y la desviación estándar de cada eje. En la Tabla 6.1 se muestran los resultados obtenidos para cada eje para un rango de ± 250 dps. La precisión se describe utilizando la desviación estándar de las mediciones y la exactitud con el error promedio.

Tabla 6.1. Precisión y Exactitud de la IMU.

Eje del giroscopio	**Precisión (dps)**	**Exactitud (dps)**
X	3.14	7.57
Y	3.83	7.40
Z	3.37	6.91
Promedio	3.45	7.29

Con los resultados obtenidos de la prueba con la IMU se determina el error de lectura y precisión del sensor de manera que se puedan tomar en cuenta al determinar la orientación de la base. De forma que se puede decir con confianza que la implementación y lectura de los sensores se realizo de forma exitosa.

Seguido, se probó la funcionalidad de los módulos y base implementados para esto, como se menciona previamente, se realizan pruebas a las diferentes configuraciones de módulos que se muestran en la Fig. 4.12 utilizando todas las funciones implementadas que se muestran en la Tabla 3.3. Se realizan 20 repeticiones para cada modulo independiente, y 30 para 2 configuraciones de 3 módulos, de esta forma se obtienen en total 120 repeticiones, cuatro veces el numero recomendado [22].

En estas pruebas se recolectaron datos de los identificadores de cada modulo, la cantidad de módulos, posición objetivo y posición final. El objetivo es verificar el correcto funcionamiento de las funciones implementadas, los módulos y la base en conjunto, por lo que se considera éxito obtener correctamente todos los datos recolectados, así como llegar a la posición objetivo. En el Anexo B.1 se puede acceder a los datos obtenidos, que luego de ser procesados se determina que se ejecutan correctamente todas las funciones y movimientos para cada configuración, se obtiene un éxito en el 100 % de las pruebas funcionales realizadas, con lo que se considera como un éxito la implementación del sistema.

6.2. Método de aprendizaje de la representación de la morfología

En esta sección se muestran los resultados obtenidos al realizar pruebas al método de aprendizaje. Se realizo seis veces el entrenamiento de la RNA para cada una de las morfologías de la Fig. 4.13. Se busca que se obtenga un porcentaje de error $\leq$ 0,01 en cada una de las salidas, idealmente $\leq$ 0,005, después de cada entrenamiento. En las Figs. 6.2 y 6.3 se muestran las gráficas de barras donde se observa el %e de cada salida para las diferentes morfologías

entrenadas.

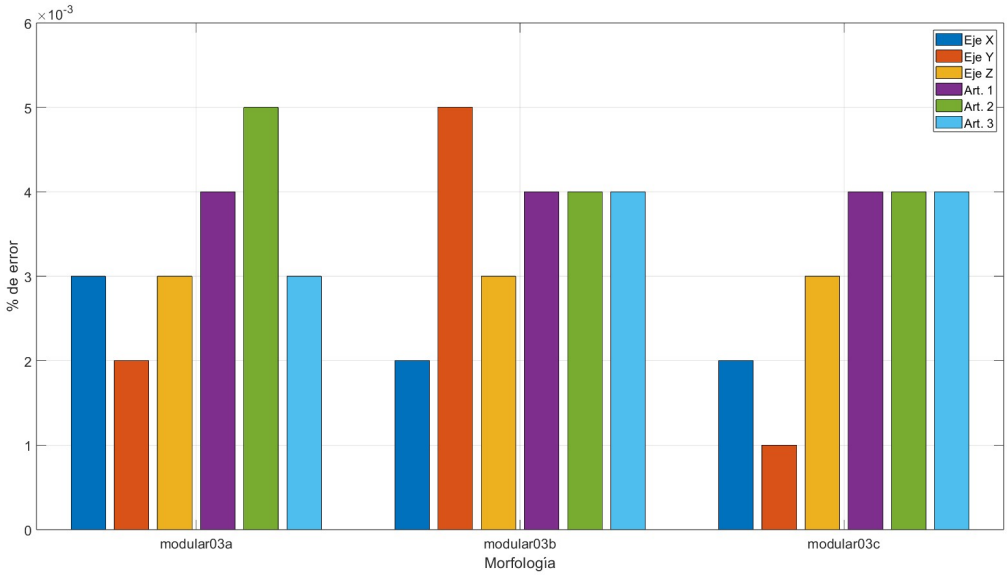


Figura 6.2. Resultados de las pruebas del método de aprendizaje para morfologías a,b y c.

Cada conjunto de columnas en las Figs. 6.2 y 6.3 representa una morfología, con una columna por cada una de las salidas.

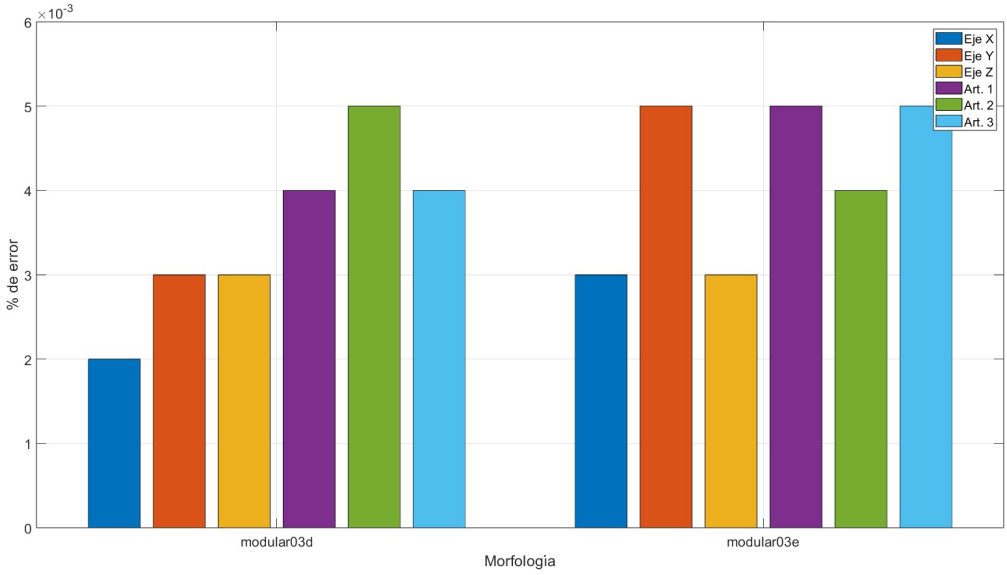


Figura 6.3. Resultados de las pruebas del método de aprendizaje para morfologías d y e.

Se observa que el error se mantiene $\leq 0{,}005\,\%$ en todos los casos, un $100\,\%$ de las pruebas, con lo que se cumple no solo con el valor objetivo, sino también con el valor ideal para estas pruebas. Se considera exitosa la implementación del método de aprendizaje, cumpliendo con las necesidades del cliente. Cabe recalcar, que se demuestra que el error presente en la posición en el eje Y, del extremo del robot, de los resultados para la morfología utilizada en la optimización del método de aprendizaje, se debe a la naturaleza planar de la morfología, y en morfologías no planares, donde hay movimiento sobre el eje en cuestión, el error se mantiene dentro del margen ideal.

6.3. Pruebas adicionales

Adicional a las pruebas descritas en el el Capitulo 4, se realizaron pruebas en conjunto con el proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares, en el cual se utilizó el sistema robótico, base y módulos, y la librería implementada, para que, en pocas palabras, el robot fuera capaz de por cuenta propia, llegar a un objetivo. Se utilizó la misma morfología de la optimización del método de aprendizaje, he ahí la razón de su uso en la selección del concepto ganador y optimización, y un objetivo, ambos se muestran en la Fig. 6.4.

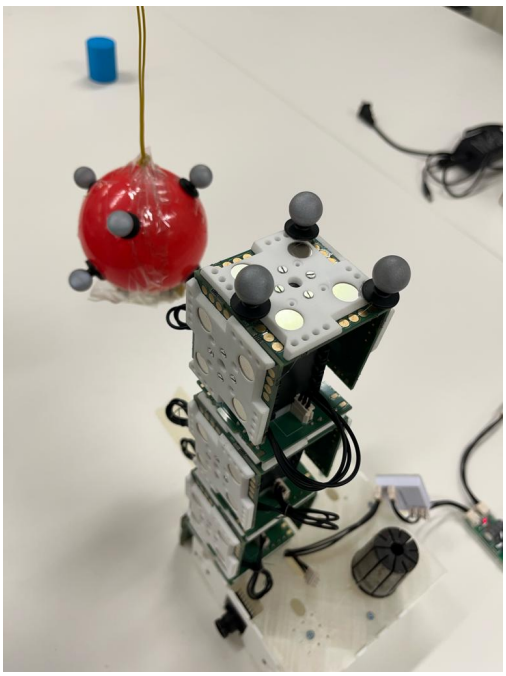


Figura 6.4. Objetivo y morfología para pruebas adicionales.

De las pruebas se grabaron evidencias donde se muestra que el robot es capaz de llegar al objetivo ubicado dentro de su área de trabajo, además de perseguir el objetivo si este se mantiene en movimiento. La verificación del proyecto adicional no hubiese sido posible sin la plataforma robótica desarrollada en este proyecto, por lo tanto, se confirma nuevamente la correcta implementación del sistema robótico y aprendizaje de la representación de la morfología. Además, se debe considerar que los resultados obtenidos son realmente buenos, puesto que el entrenamiento del modelo se realizo con datos de simulación y las pruebas con el sistema real, puesto que aun con la brecha entre la simulación y realidad, los resultados fueron exitosos. Se muestran evidencias en vídeo en el vinculo que se añade en el Anexo B.2.

6.4. Análisis económico

En esta sección se busca realizar un análisis de los recursos generados e invertidos durante la realización de este proyecto. Debido a que es un proyecto de investigación y no tiene como objetivo generar ingresos económicos, la comparación no se enfoca únicamente en los recursos monetarios generados.

Inicialmente se establecen los recursos invertidos que se listan a continuación:

- La compra de los componentes electrónicos y piezas necesarias para montar hasta 12 módulos y 2 bases, representa un gasto de 4 200€. Componentes suficientes para el montaje de manipuladores, cuadrúpedos y bípedos de hasta 12 módulos.
- El tiempo de ingeniería invertido es de aproximadamente 320 horas, que para un ingeniero con el mismo puesto representa 12,5€ por hora, un total de 4 000€. Este proyecto se realiza no remunerado por lo que realmente no representó un gasto para el GII.
- Adicionalmente, se toma en cuenta para posibles imprevistos un presupuesto del 10 % de los gastos en componentes. Que finalmente no fue necesario, sin embargo siempre es importante tomar en cuenta.

En la Tabla 6.2 se muestra el desglose de la inversión total del proyecto, incluyendo el coste de cada uno de los conceptos considerados, es de aproximadamente 8 620€.

Tabla 6.2. Recursos invertidos en la realización del proyecto.

Concepto	**Valor (€)**
Componentes para 12 módulos	4 200
Tiempo de ingeniería	4 000
Imprevistos	420
Total	8 620

En contraste, los recursos generados son los siguientes:

- Se desarrolla un robot modular, reconfigurable y con capacidades sensoriales limitadas, que permite realizar pruebas para un proyecto de suma importancia para el GII, el e-MDB.
- El robot modular implementado es capaz de tomar múltiples configuraciones, funcionar en diversos entornos y es fácilmente escalable. Ahora el GII cuenta con una plataforma de pruebas que puede adoptar morfologías totalmente diferentes: manipuladores, cuadrúpedos, bípedos, entre otros.
- El sistema desarrollado sirve para el desarrollo del proyecto realizado en paralelo: Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares.
- Se redactan dos artículos para enviar a diferentes congresos: uno enfocado en el sistema robótico y método de aprendizaje que se desarrolló de este proyecto; y otro enfocado en el proyecto Integración de distintas componentes de la arquitectura e-MDB en sistemas robóticos modulares, en el cual se utiliza la plataforma desarrollada. A la fecha de redacción de este informe los artículos no han sido enviados ni publicados, y están en desarrollo, pero con certeza serán publicados. Asimismo, no se descarta la posibilidad de publicar artículos adicionales sobre futuras aplicaciones del proyecto.
- Uno de los recursos generados más importantes es la visibilidad que se genera para el GII con este proyecto, ya sea de forma directa con los artículos y una entrevista realizada con un periódico local sobre mi colaboración con el grupo y la universidad, o de forma indirecta al contar con una plataforma reconfigurable y escalable para realizar pruebas con el e-MDB. Esto es debido a que parte de los ingresos del GII provienen de proyectos que realizan para la Unión Europea y empresas externas, de forma que esta visibilidad permite conseguir nuevos proyectos y financiación.

Como se observa, a pesar de que la inversión para el desarrollo de este proyecto es relativamente alta, cerca de 10 000€, se generan múltiples recursos al grupo, a pesar de que no se puede calcular el retorno de inversión de manera monetaria puesto que los recursos generados no pueden medirse en euros. Sin embargo, el cliente se encuentra realmente satisfecho con los resultados obtenidos.

7 Conclusiones y Recomendaciones

En este capitulo se exponen las conclusiones y recomendaciones finales del proyecto. En las conclusiones se busca evidenciar el cumplimiento de los objetivos del proyecto y la satisfacción de las necesidades del cliente. Con las recomendaciones se busca mencionar y explicar los próximos pasos para el proyecto y aplicaciones del mismo, así como aprendizajes útiles para futuros trabajos.

7.1. Conclusiones

Para el desarrollo de este proyecto se plantearon cuatro objetivos específicos con sus respectivos entregables e indicadores. En esta sección se procede a conectar los conceptos desarrollados durante proceso de diseño y procedimientos realizados con el estado final de los objetivos.

Los objetivos se muestran en la Sección 1.3. El primer objetivo consistió en buscar opciones para aumentar las capacidades de sensorización del robot modular, y esto se realizó en el documento del Anexo A.1, en este documento se explica el análisis detrás de la selección de las percepciones finales utilizadas, y los cambios realizados sobre el modulo base del sistema, para implementar conexiones adicionales y las soluciones de sensorización seleccionadas, un sensor de distancia IR y una IMU. Este entregable fue revisado y aceptado por el cliente previo a la implementación de la base adaptada.

El segundo objetivo era referente a la implementación y montaje del sistema robótico, incluyendo los cambios propuestos en el entregable del primer objetivo. Para este objetivo se definen tanto un entregable como indicador, el entregable son los módulos EMERGE fabricados y funcionales, y la manera de verificar los módulos entregados es mediante el

indicador definido, con el cual se busca obtener un 90 % de éxito en las pruebas funcionales realizadas a la sensorización de la base, módulos y cambios realizados. En la Sección 6.1 se presentan los resultados para las pruebas planteadas en el Capítulo 4 para el sistema robótico, su sensorización: sensor IR e IMU, y funcionalidad de los módulos e implementación del control. En la Fig. 6.1 se muestra la curva característica del sensor IR, donde se tuvo éxito en su implementación. Además, para la IMU se realiza la caracterización de los ejes del giroscopio, cuyos resultados se muestran en la Tabla 6.1, donde se determino con éxito la exactitud y precisión del sensor. Finalmente, se realizan pruebas a los módulos fabricados, cambios realizados sobre el módulo base y la librería implementada para su control. Estas pruebas presentan resultados positivos en el 100 % de las ejecuciones, lo que demuestra que la conexión con y entre los módulos, y el control del sistema funciona correctamente. De esta forma, luego de realizar todas las pruebas descritas al sistema mencionado se puede decir con certeza que se cumple con el segundo objetivo, puesto que se obtiene un 100 % de éxito en las pruebas realizadas.

Los objetivos tres y cuatro se enfocan en el desarrollo y verificación del método de aprendizaje. En el objetivo tres se plantea el diseño de una solución que permita aprender la propia representación de la morfología del robot modular, de manera que se entre un programa en Python mediante el cual se pueda aprender la representación de la morfología, en conjunto con un indicador en el que se requiere que el aprendizaje de la morfología se realice con un error menor o igual a 0,01 %, para la cadena de tres módulos que se muestra en la Fig. 3.2. El método de aprendizaje utilizado se ve definido por el cliente puesto que se requiere el uso de RNA, cuando se trabaja con estas redes se debe realizar un estudio de hiperparámetros que va a definir la calidad del modelo aprendido por la RNA, y es mediante este proceso que se realiza el ajuste de los hiperparametros para obtener un modelo que presente un error en todas las salidas $\leq$ 0,005 %, para la morfología establecida. Este proceso se realiza en la selección de conceptos del proceso de diseño que se desarrolla en el Capítulo 4, donde se generan múltiples propuestas, y se selecciona la mejor, que luego se optimiza hasta llegar a cumplir con el objetivo. Para verificar que se cumple con el indicador, se realizan un total de 30 replicas del entrenamiento de la RNA seleccionada, y se calcula el promedio del error en cada salida que se muestra en la Tabla 4.17, se aprecia claramente como el error en todas las salidas fue menor al valor ideal de 0,005 %, menos de la mitad del establecido en el indicador. Por lo tanto, se puede asegurar que se cumple el tercer objetivo del proyecto.

El objetivo restante es el cuarto y como se menciona previamente, este se centra en verificar el funcionamiento de la técnica de aprendizaje implementada, de manera que aprenda

la morfología en mínimo el 90 % de las pruebas realizadas. Para la verificación se plantea la prueba descrita en el Capítulo 4 Sección 4.5 para el método de aprendizaje, donde se establecen cinco morfologías diferentes, para las cuales se generan los datos y realiza el entrenamiento de la RNA, seis replicas por cada morfología para un total de 30 replicas. Se busca que el aprendizaje de la representación de cada morfología presente resultados con un error $\leq 0{,}01\,\%$, idealmente $\leq 0{,}005\,\%$ luego de haber sido optimizada la RNA. Para obtener éxito en al menos un 90 % de las pruebas realizadas, se debe tener un error $\leq 0{,}01\,\%$ en al menos 4,5 de las morfologías, es decir cuatro morfologías y al menos tres replicas de la quinta morfología. En las Figs. 6.2 y 6.3 se muestran los gráficos de barras donde se observa el porcentaje de error para las salidas de cada una de las morfologías aprendidas, donde el error máximo presente es de $0{,}005\,\%$ para cada una de ellas. Con base en los resultados obtenidos se puede concluir que se obtuvo un porcentaje de error que cumple con el valor ideal establecido en las métricas del proyecto, y por ende, se cumple satisfactoriamente con el cuarto y ultimo objetivo del proyecto.

7.2. Recomendaciones

- En proyectos con una gran componente de investigación es recomendable llevar un control de las referencias y fuentes utilizadas, ya sea de manera manual en un archivo de texto, o utilizar gestores de bibliografía como *Zotero* o *Mendeley Reference Manager*. Esto debido a que se revisan muchas fuentes diferentes y no necesariamente todas se utiliza en la versión final, así como para llevar un control de que temas se pueden encontrar en cada fuente.

- La utilización de sensores como el IR y la IMU es muy practico y útil, sin embargo, en algunos casos donde la utilización de un sistema de visión es posible puede ser una opción con mayor flexibilidad y escalabilidad.

- Cuando se trabaja con RNA se debe considerar que comportamientos semejantes al ruido son muy difíciles de aprender por la red, debido a la aleatoriedad de los valores, tal es el caso de la orientación sobre eje Y en la morfología utilizada.

- Para observar con mayor detalle el comportamiento de la cantidad de *epochs* en el error de validación y entrenamiento es recomendable que al graficar se omitan las primeras *epochs* debido a que el error inicia con un valor muy alto y se reduce conforme el entrenamiento avanza.

- Es importante que en proyectos de investigación se definan claramente las metas e hitos de forma que se tenga claro cuando concluyen las tareas, para evitar que el proyecto escale sin control y se salga del alcance establecido.

7.3. Trabajo Futuro

- Continuar el trabajo de este proyecto para un futuro trabajo de fin de máster en colaboración con el GII.
- Realizar más pruebas y continuar el desarrollo del sistema robótico para presentar un articulo el 3 de abril de 2024 en la *Conference on Artificial Life* (ALIFE).
- Realizar pruebas adicionales con la arquitectura e-MDB y el sistema robótico.
- Fabricar módulos adicionales para poder realizar pruebas con cuadrúpedos y bípedos.
- Implementar el sistema de visión *OptiTrack* para obtener la ubicación y orientación del robot con mayor precisión y exactitud.
- Implementar el sistema de visión con la cámara frontal montada sobre la base.
- Ampliar la librería y crear un código que sea completamente compatible con simulación y robot real.
- Publicar múltiples artículos en donde se aprovecha el sistema robótico desarrollado, como por ejemplo uno enviado a la *International Conference on the Interplay between Natural and Artificial Computation* (IWINAC) el 15 de febrero de 2024.
- Continuar colaborando activamente con el GII en este y demás proyectos.

Bibliografía

[1] G. I. D. INGENIERÍA, *Presentación*. dirección: https://gii.udc.es/presentacion.

[2] G. I. D. INGENIERÍA, *Investigación*. dirección: https://gii.udc.es/investigacion.

[3] R. Moreno y A. Faiña, "EMERGE Modular Robot: A Tool for Fast Deployment of Evolved Robots," *Frontiers in Robotics and AI*, vol. 8, 2021, ISSN: 2296-9144. DOI: 10.3389/frobt.2021.699814. dirección: https://www.frontiersin.org/articles/10.3389/frobt.2021.699814.

[4] A. Romero Montero, *e-MDB: A cognitive architecture for lifelong open-ended learning autonomy in robotic systems. PhD program in Naval and Industrial Engineering*, 2022. dirección: http://hdl.handle.net/2183/31717.

[5] A. Faíña Rodríguez-Vila, "Un sistema robótico modular heterogéneo para entornos dinámicos y no estructurados," Tesis doct., 2011.

[6] A. Dearden e Y. Demiris, "Learning forward models for robots," en *IJCAI*, vol. 5, 2005, pág. 1440.

[7] B. Chen, R. Kwiatkowski, C. Vondrick y H. Lipson, "Fully body visual self-modeling of robot morphologies," *Science Robotics*, vol. 7, n.º 68, eabn1944, 2022. DOI: 10.1126/scirobotics.abn1944. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abn1944. dirección: https://www.science.org/doi/abs/10.1126/scirobotics.abn1944.

[8] M. Naya-Varela, A. Faina, A. Mallo y R. Duro, "A study of growth based morphological development in neural network controlled walkers," *Neurocomputing*, vol. 500, págs. 279-294, 2022, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.09.082. dirección: https://www.sciencedirect.com/science/article/pii/S0925231222006385.

[9] D. Ha y J. Schmidhuber, "World Models," *CoRR*, vol. abs/1803.10122, 2018. arXiv: 1803.10122. dirección: http://arxiv.org/abs/1803.10122.

[10] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd. Wiley Publishing, 2007, ISBN: 0470035617.

[11] A. Gulli y S. Pal, *Deep Learning with Keras*. Packt Publishing, 2017, ISBN: 1787128423.

[12] R. Co., *What is DYNAMIXEL?* Dirección: https://www.dynamixel.com/whatisdxl.php.

[13] R. Co., *AX12-A*. dirección: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/.

[14] R. B. Kristiawan, F. Imaduddin, D. Ariawan, Ubaidillah y Z. Arifin, *Open Engineering*, vol. 11, n.º 1, págs. 639-649, 2021. DOI: doi:10.1515/eng-2021-0063. dirección: https://doi.org/10.1515/eng-2021-0063.

[15] K. T. Ulrich y S. D. Steven D. Eppinger, *Diseño y desarrollo de productos (5a. ed.)* México, D.F: McGraw-Hill Interamericana, 2013, ISBN: 6071509440. dirección: https://ebookcentral.proquest.com/lib/[SITE_ID]/detail.action?docID=3214953.

[16] R. Co., *Controller Compatibility*. dirección: https://emanual.robotis.com/docs/en/parts/controller/controller_compatibility/.

[17] R. Co., *PARTS - Controllers*. dirección: https://www.robotis.us/controllers-2/.

[18] R. Co., *SOFTWARE*. dirección: https://emanual.robotis.com/docs/en/software/.

[19] BricoGeek, *Tienda de Electrónica, Robótica, Arduino y Raspberry Pi | BricoGeek*. dirección: https://tienda.bricogeek.com/.

[20] C.-H. Chen, J.-P. Lai, Y.-M. Chang, C.-J. Lai y P.-F. Pai, "A Study of Optimization in Deep Neural Networks for Regression," *Electronics*, vol. 12, n.º 14, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12143071. dirección: https://www.mdpi.com/2079-9292/12/14/3071.

[21] J. Pomerat, A. Segev y R. Datta, "On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression," en *2019 IEEE International Conference on Big Data (Big Data)*, 2019, págs. 6183-6185. DOI: 10.1109/BigData47090.2019.9005674.

[22] H. G. Pulido, R. De la Vara Salazar, P. G. González, C. T. Martínez y M. d. C. T. Pérez, *Análisis y diseño de experimentos*, 2.ª ed. McGraw-Hill New York, NY, USA: 2008.

[23] ArduCAM, *5MP SPI Camera User Guide*. dirección: https://arducam.com/downloads/shields/ArduCAM_Mini_5MP_Plus_OV5642_Camera_Module_DS.pdf.

[24] Arduino, *Arduino MEGA 2560 Rev3 - Product Reference Manual*. dirección: https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf.

[25] SHARP, *GP2Y0A51SK0F*. dirección: https://www.pololu.com/file/0J845/GP2Y0A41SK0F.pdf.

[26] S. Electronics, *SparkFun ICM-20948 Arduino Library*. dirección: https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary.

A

Entregables

A.1. Documento de sensorización

En esta sección se incluye un vinculo al entregable del primer objetivo, archivo donde se evalúan y explican las soluciones de sensorización, y modificaciones sobre el módulo base.

A.2. Librería para el control del sistema robótico

Este anexo incluye el vínculo al repositorio donde se puede acceder a los archivos de la librería implementada y las librerías adicionales utilizadas: *Dynamixel SDK*, para el control de los motores, y *zmqRemoteApi* para el control de CoppeliaSim mediante Python.

A.3. Modelos de las piezas de la base

Este anexo incluye el vínculo a la carpeta con los modelos donde se puede acceder a los archivos .stl de las piezas del módulo base del sistema robótico.

B Archivos adicionales

B.1. Resultados experimentales

En este anexo se añade el vinculo a una carpeta con resultados de las pruebas en Google Drive donde se puede acceder a todos los archivos de Excel donde se almacenan los datos resultantes de todas las pruebas y procedimientos realizados en este proyecto.

B.2. Evidencias adicionales

Vinculo a una carpeta con evidencias en vídeo en Google Drive donde se puede acceder a múltiples archivos donde se observan los resultados de las pruebas adicionales mencionadas en el Capítulo 6.