UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA
INGENIERÍA MECÁNICA Y AUTOMOTRIZ

TESIS

Sistema de Evasión de Obstáculos en Robot Humanoide de 18 GDL Mediante Clasificador Basado en RNA

Como parte de los requisitos para obtener el título de

INGENIERO MECÁNICO Y AUTOMOTRIZ

PRESENTA

Oscar David de Jesús Díaz

DIRIGIDO POR

M. en C. Salvador Martínez Cruz

M. en C. Salvador Martínez Cruz (16783) _____
Presidente

Dr. Gerardo Israel Pérez Soto (12950) _____
Secretario

Dr. Rivera Guillen Jesús Rooney (8943) _____
Vocal

Dr. Israel Zamudio Ramírez (16787) _____
Suplente

UAQ, FI, San Juan del Rio,
Querétaro, México.
Julio del 2023

Sistema de evasión de obstáculos en robot humanoide de 18 GDL mediante clasificador basado en RNA

por

Oscar David de Jesús Díaz

DEDICATORIA

Queridos mamá y papá,

Esta tesis no solo representa mi dedicación y esfuerzo, sino también el amor y apoyo incondicional que me han brindado a lo largo de mi vida. Gracias por ser mis pilares, mis guías y mi mayor inspiración. Sin ustedes, este logro no habría sido posible. Los amo con todo mi corazón.

Con cariño,

Davo

AGRADECIMIENTOS

Agradezco al equipo LYNXBots por su generosidad al prestarnos el material necesario para llevar a cabo este proyecto. Su apoyo ha sido fundamental y ha permitido que nuestras ideas se conviertan en realidad.

Mi más sincero agradecimiento a todas las personas que contribuyeron en la revisión y corrección de este documento. Vuestras sugerencias y comentarios han mejorado significativamente la calidad de este trabajo.

Deseo expresar mi gratitud al M. en C. Salvador Martínez Cruz por su inestimable dirección y apoyo a lo largo de este proyecto. Su experiencia y confianza han sido un pilar fundamental en mi camino hacia el éxito académico.

Por último, pero no menos importante, quiero agradecer a toda mi familia por su apoyo incondicional. Vuestro aliento constante y el impulso que me han brindado han sido mi motor durante todo este proceso. No hay palabras suficientes para expresar mi gratitud por el amor y la confianza que han depositado en mí.

A todos y cada uno de ustedes, mi más profundo agradecimiento. Sin su colaboración, este logro no habría sido posible.

RESUMEN

En la última década, ha habido un creciente interés en la investigación de robots que utilizan sensores para lograr una correcta identificación y comprensión del entorno en el que operan. Estos trabajos de investigación son de suma importancia, ya que permiten a los robots interactuar de manera más efectiva con su entorno y realizar tareas complejas de manera autónoma.

En el presente trabajo, se propone la implementación del sensor de visión HaVimo2.0 en un robot humanoide Bioloid Premium con el objetivo de lograr la detección y evasión de obstáculos. El sensor de visión HaVimo2.0 es conocido por su capacidad para capturar imágenes de alta resolución y su capacidad de procesamiento incorporada, lo que lo convierte en una elección adecuada para este propósito. La implementación conlleva el desarrollo de software de bajo nivel C embebido que permite establecer una comunicación funcional y adecuada entre el sensor HaVimo2.0 y el controlador del robot humanoide, CM-530. Este enfoque permite aprovechar al máximo las capacidades del sensor y el controlador al mismo tiempo. Para el procesamiento que será llevado a cabo por parte del controlador, se utiliza como método clasificador de las diferentes imágenes capturadas por parte del sensor, la implementación de una Red Neuronal Artificial (RNA) entrenada por Aprendizaje Maquina, mejor conocido como *Machine Learning*.

El actual desarrollo propone la utilización de una RNA aplicada a un robot humanoide Bioloid Premium, para que, en conjunto con el resto de los elementos del sistema, éste tenga la capacidad y rendimiento que tienen robots de la misma empresa Robotis© de gama alta que ya incluyen la capacidad de hacer el reconocimiento y evasión de obstáculos de fábrica, pero que sin embargo, lo realizan teniendo un valor 6 u 8 veces mayor al que alcanza el sistema del presente proyecto en conjunto con todos sus elementos.

ÍNDICE

Índice de figuras

Índice de tablas

1. Introducción

Se predice que los robots de servicio van a transformar en buena medida la vida de las personas en un futuro próximo, ya que realizaran una gran cantidad de tareas en beneficio de la humanidad. Estas van desde tareas cotidianas como cortar pasto, labores que requieren de cierto entrenamiento especializado como cuidar personas de avanzada edad o enfermas, trabajo que involucran un alto grado de exactitud, como intervenciones en cirugía de alta precisión, hasta tareas que involucran riesgos elevados, como explorar una mina terrestre o marítima, o desactivar una bomba (Morales & Sucar, 2022).

La revolución robótica va a detonar un importante avance científico y tecnológico en diversas áreas de la mecánica, control, electrónica y computación; pero además ayudará a resolver problemas sociales de salud y seguridad, entre otros; y creará una nueva industria con importantes beneficios económicos (Morales & Sucar, 2022).

En su conjunto, la robótica impulsará el desarrollo de los países que inviertan y se coloquen a la vanguardia en esta tecnología.

Dentro de la gran gama de posibles robots que se pueden desarrollar, sin lugar a duda, los robots humanoides son los que ofrecen mayores beneficios. Un robot humanoide es un robot móvil que tiene una apariencia similar a la de un ser humano, incluyendo la capacidad de manipulación (brazos) y locomoción (ya sea bípedo o con llantas). Por un lado, se espera que un robot humanoide de tamaño natural tenga la capacidad de utilizar todas las herramientas que el hombre ha desarrollado a lo largo de su historia, sin que se tengan que adaptar ambientes especiales (Morales & Sucar, 2022).

El uso de robots humanoides ha incrementado en los últimos años en diferentes áreas y sin duda uno de los mayores desafíos que se enfrentan los desarrolladores de este tipo de sistemas es la correcta identificación y evasión de obstáculos de parte del robot durante su deambulación.

Comúnmente se usan sensores en el robot para la identificación correcta del entorno, pueden ser sensores con un funcionamiento trivial como un sensor de proximidad hasta sensores de alta complejidad computacional y de alto costo, sin

embargo, los trabajos más recientes se esmeran en lograr la correcta identificación del entorno con la implementación de la nueva tecnología que podría ser de bajo costo buscando la eficiencia mediante las diversas técnicas de procesamiento de señales.

1.1. Antecedentes

En esta sección, se muestra parte de la literatura revisada para llevar a cabo el presente proyecto de Tesis. Iniciando con los trabajos relacionados a Robots Humanoides, los referentes a la competencia de la Federación de la Asociación Internacional de Robot-Soccer, mejor conocido como *Federation of International Robot-Soccer Association* (FIRA), trabajos en los que se tiene por objetivo evadir obstáculos con robots y finalmente el uso de redes neuronales en robots en la última década.

1.1.1. Robots humanoides

En los últimos años, se desarrollaron trabajos con robots humanoides en diferentes áreas de la ingeniería, por ejemplo, (Valero Carvajal et al., 2021) presentaron el desarrollo de algoritmos de procesamiento de imágenes implementados en el robot humanoide InMoov. Por su parte, (Segura et al., 2017) llevaron a cabo el desarrollo de un sistema robótico Bioloid Premium que enseña pasos de la cultura maya siendo controlado remotamente a través de una interfaz gráfica. Por otro lado, (Lobos-Tsunekawa et al., 2018) propusieron un sistema de navegación visual sin mapa para robot humanoides bípedos que extrae información de imágenes en color para derivar movimientos utilizando Aprendizaje de Refuerzo Profundo, mejor conocido como *Deep Reinforcement Learning* (DRL). Además, (Fanello et al., 2017) tuvieron el objetivo de mejorar las capacidades de percepción visual de un robot haciendo una restricción en el escenario de adquisición, construyendo y analizando un sistema que puede entrenarse rápidamente para incorporar nuevas evidencias a través del esquema de codificación Mejores Entradas de Código, mejor conocido como *Best Code Entries* (BCE), y un nuevo operador de agrupación Clasificacion de Pesos de Nivel Medio, mejor conocido como *Mid-Level Classification Weights* (MLCW). Asimismo, (Fritsche et al., 2013) proporcionaron un entorno asequible e intuitivo para tele operar robots humanoides en primera persona utilizando realidad aumentada, reduciendo la complejidad para completar tareas en entornos que no son accesibles. Continuando, (Al-Shuka et al.,

2016) ofrecieron una presentación sistemática de controladores de equilibrio multinivel para la estabilización y recuperación del equilibrio de robots humanoides basados en el Punto de Momento Cero, mejor conocido como *Zero Moment Point* (ZMP). Sin embargo, el desarrollo e impulsión en la investigación en los robots humanoide no abarca solamente los últimos años, sino que es un proceso del cual se tiene registro desde ya bastante tiempo atrás, y es posible darse una idea con (Hirose & Ogawa, 2007) quienes hablaron en su artículo sobre los inicios de la industria Honda en el desarrollo de la robótica humanoide con el robot ASIMO en el año 1986, con el cual, buscaban la mejora del funcionamiento del mismo para lograr que compartiera espacio con los humanos y mejorar la calidad de vida. De igual manera, (Bogue, 2020) proporcionó detalles técnicos de los desarrollos de robots humanoides desde la década de 1970 hasta la actualidad para considerar sus posibles aplicaciones y perspectivas. A su vez, (Kuffner et al., 2001) crearon un algoritmo para planificar estrategias de navegación seguras para robots bípedos que se mueven en entornos llenos de obstáculos. A partir de un conjunto discreto de movimientos plausibles de un solo paso estáticamente estables, se utiliza un enfoque de programación dinámica directa para calcular una secuencia de ubicaciones de pasos factibles. Adicionalmente, (Kim et al., 2007) plantearon un algoritmo de control en línea que considera las inclinaciones locales y globales del suelo mediante las cuales un robot humanoide bípedo puede adaptarse a las condiciones del suelo. Además, (Martínez Cruz et al., 2016) desarrolló e implementó un software para la programación de movimientos en robot humanoide Bioloid Premium de la empresa Robotis©.

El presente trabajo de Tesis contribuirá con el desarrollo de los robots humanoides en el área de investigación y el crecimiento de la misma, aportando a la literatura previamente revisada, un sistema de evasión de obstáculos en un robot humanoide de 18 Grados De Libertad (GDL) usando servomotores como articulaciones y un sensor de visión HaVimo2.0 como retroalimentación, el método de clasificación está basado en una Red Neuronal Artificial (RNA).

1.1.2. Competencia FIRA

Los usos y aplicaciones que conllevan a los robots humanoides son bastantes numerosos, y el continuar con la mejora de este tipo de sistemas es sumamente importante para el perfeccionar de los mismos, es por esto que se impulsa a los jóvenes

a desarrollarse en estas áreas, y un ejemplo claro de un evento de este tipo es la competencia internacional llevada a cabo por FIRA, en donde por sus diferentes categorías y áreas, lleva a gente de todo el mundo a el intento de superación de sus mismos sistemas, como por ejemplo, (Hu et al., 2010) quienes diseñaron e implementaron un robot humanoide basado en visión para HuroCup FIRA, específicamente para competencias de maratón y tiro penal, instalaron una placa llamada RoBoard con un procesador x86 en el robot humanoide para que sea su sistema de control, que se utiliza para capturar imágenes, procesar las mismas, tomar decisiones y controlar los movimientos del robot. Similarmente, (Su et al., 2008) desarrollaron un sistema de estrategia de control e imagen basado en un Sistema en un Chip Programable, mejor conocido como *System on a Programmable Chip* (SOPC) para un robot humanoide, constaba con un sistema visual para el reconocimiento de patrones. La estrategia de control se ajusta para el evento de tiro penal y otros eventos de la competencia FIRA. Mientras tanto, (Kuo et al., 2014) utilizaron la arquitectura de nodo de sensor inalámbrico basada en la red de Petri para controlar un robot humanoide para competencias de levantamiento de pesas y sprint en la liga FIRA HuroCup.

Y no solamente el software que los sistemas poseen es importante, sino que también la cinemática lo es, para ejemplificar se tiene a (Tu et al., 2020) realizaron el diseño e implementación de un robot humanoide capaz de participar en tiro con arco en HuroCup de FIRA 2018, imitando una postura de tiro especial bajo una estructura de brazo restringida y una planificación del movimiento de ambos brazos para obtener más torque para jugar con el arco profesional. Añadiendo, (Suarez-Rivera et al., 2017) presentaron modificaciones realizadas a un robot Bioloid tipo A para participar en la competencia de levantamiento de pesas HuroCup de la FIRA. Los cambios se basan en un análisis estático en las posiciones y los elementos diseñados eran simulados antes de la construcción utilizando un software de elementos finitos.

El desarrollo del presente trabajo será una innovación más que se podría apreciar durante una competencia FIRA, dado que el sistema que se realizará, basado en una red neuronal y el uso del sensor de visión HaVimo2.0 es algo que no ha apreciado anteriormente y podría ser un sistema que mejore la funcionabilidad de los sistemas vistos hasta el momento para la competencia.

1.1.3. Evasión de obstáculos en robots

Aunque los avances científicos en los sistemas robóticos avanzan de una manera parecida, se destaca que las tareas para las que se preparan cada uno de ellos puede volver completamente diferente el procesamiento y características que cada uno conlleva, y también hay acciones que deben incluir para poder llegar a su objetivo, una de ellas podría ser la evasión de obstáculos, en donde aún se sigue intentando perfeccionar el desempeño aplicando diferentes herramientas. Por ejemplo, (Yoo et al., 2014) presentaron un simulador para el procesamiento de la visión y planificación de tareas para un robot humanoide pequeño (HSR-VII). Se introdujeron el HSR-VII y el algoritmo de generación de patrones de caminata modificables. La simulación fue diseñada para la categoría ObstacleRun en FIRA. Continuando, se tiene a (Rath et al., 2018) quienes desarrollaron un controlador de navegación para un humanoide mediante el uso de lógica difusa como un algoritmo inteligente para evitar los obstáculos presentes en el entorno y alcanzar la posición deseada de forma segura. A su vez, (Regier et al., 2019) entrenaron una Red Neuronal Convolucional, mejor conocida como *Convolutional Neural Network* (CNN) para el desarrollo de un nuevo enfoque para la navegación humanoide a través de entornos abarrotados que explota el conocimiento sobre diferentes clases de obstáculos y selecciona las acciones de robot apropiadas. En cambio, (Nobile et al., 2021) exploraron la posibilidad de usar información de profundidad de una cámara Rojo Verde Azul-Digital, mejor conocida como *Red Green Blue-Digital* (RGB-D) móvil montada en la cabeza del robot e investigaron, en particular, estrategias de control activo para escanear el entorno de manera efectiva. Añadiendo, (Delfin et al., 2018) presentaron un esquema de navegación humanoide basado en un mapa topológico normalmente llamado Memoria Visual, aunque mejor conocido como *Visual Memory* (VM), el cual está compuesto por un conjunto de imágenes clave adquiridas offline mediante una fase de enseñanza supervisada con guía humana. De manera más general, (Muni et al., 2020) examinaron la técnica basada en reglas para la dirección del robot humanoide en entornos caóticos y desarrollaron varias reglas basadas en la dirección del movimiento, la distancia entre el humanoide y el objetivo, la distancia entre el humanoide y los obstáculos, y el ángulo entre el robot y los obstáculos adyacentes, para llegar a la meta sin chocar con los obstáculos.

Con el desarrollo del presente trabajo de Tesis, se contribuye significativamente en el área de evasión de obstáculos con robots humanoides, esto se obtiene mediante la implementación de la RNA en para le clasificador de obstáculos.

1.1.4. Inteligencia artificial en robots

La inteligencia artificial tiene de igual manera un auge en los últimos años, lo que lleva al intento de aplicación de esta rama en los diferentes sistemas encontrados en el mundo, y la robótica no es la excepción, por ejemplo (B & Estivill-castro, 2019) propusieron una arquitectura de RNA eficiente, a través del *Deep Learning* (Aprendizaje profundo, traducido del idioma inglés), para el problema de detección de objetos relevantes en entornos de fútbol robótico, reduciendo aproximadamente 35 veces en el tiempo de ejecución en comparación con el modelo Tiny YOLO de última generación. Continuando, (Fernández Calderón, 2018) empleó algoritmos de *Deep Learning*, en concreto, RNA, para la clasificación de imágenes, construyendo un robot móvil capaz de recorrer espacios cerrados evitando su colisión con los obstáculos que encuentre, usando un único sensor visual. Además, (Garrido et al., 2020) implementaron algunas variantes de *Deep Q-Learning* para entrenar a un robot móvil en la tarea de navegar hacia las personas que detecta (a través de una cámara RGB) en simulación. Y con un sistema un poco diferente, (Badenes Villena, 2022) presentó la implementación de una RNA mediante aprendizaje supervisado con el lenguaje de programación Python, entrenando la red neuronal en un pc de sobremesa para que aprenda a conducir el coche y una vez logrado dicho cometido llevar a cabo el trasladado de la red neuronal al robot creado.

En el presente trabajo de Tesis, se implementa una RNA en un robot humanoide de 18 GDL, haciendo uso de Machine Learning en el sistema, buscando una buena eficiencia del robot para la tarea de la evasión de obstáculos.

1.2. Descripción del problema

En la actualidad existe una vasta cantidad de aplicaciones para los robots del tipo humanoide, por ejemplo; robots de rescate, robots de servicio, robots para rehabilitación en personas con capacidades diferentes, robots instructores, entre otros. Sin embargo, una de las mayores dificultades que enfrentan los desarrolladores de este tipo de

sistemas es la correcta identificación y evasión de obstáculos durante la deambulación. Esto impide el correcto desplazamiento del robot hacia su lugar de destino y afecta directamente su desempeño, por lo que el propósito para el que fue diseñado no se cumple. Además, con base en la literatura revisada, no se ha desarrollado un sistema de evasión de obstáculos en robot humanoide usando un clasificador con RNA y el sensor HaVimo2.0.

1.3. Justificación

Hoy en día, se invierte tiempo y recursos económicos a nivel global para la investigación y desarrollo de sistemas autónomos, tales como; automóviles autónomos, Vehículos Aéreos no Tripulados, mejor conocidos como *Unmanned Aerial Vehicle* (UAV), robots industriales, brazos robóticos, entre otros.

Los robots humanoides no están rezagados en este sentido, ya que en la última década se publicaron artículos científicos y tesis donde se pretende obtener una mayor independencia en este tipo de robots. En estos trabajos, principalmente se realiza la toma de decisiones usando un solo parámetro de entrada, el cual es una señal que proviene directamente de un sensor sin una etapa de preprocesamiento. La inteligencia artificial es una buena alternativa para evadir obstáculos en un robot humanoide, sin embargo, hasta la investigación realizada por el autor del presente trabajo de tesis, no se presenta un trabajo de inteligencia artificial en el que se use particularmente un sensor HaVimo2.0 y un robot humanoide conformado por servomotores.

En este trabajo de Tesis, se propone un sistema que permita la evasión de obstáculos en un robot humanoide de 18 GDL y un sensor HaVimo2.0 aplicando un clasificador basado en RNA. Al usar esta tecnología se tiene un sistema con la misma eficiencia que los trabajos mencionados anteriormente, pero con un costo de hardware hasta diez veces menor y se reduce significativamente el nivel de complejidad computacional. Además, con este desarrollo, se pretende participar en competencias de robótica nacionales e internacionales que seguirán poniendo en alto el nombre de la Universidad Autónoma de Querétaro, del estado de Querétaro y del país.

1.4. Hipótesis

Es posible, mediante el entrenamiento e implementación de una RNA, desarrollar un clasificador de obstáculos en un robot humanoide de 18 GDL, usando como retroalimentación el sensor de visión Havimo2.0, y probar su efectividad usando obstáculo de distintas formas y colores.

1.5. Objetivos

En esta sección se presentan los objetivos planteados en el presente proyecto de Tesis, desde el objetivo general hasta los particulares.

1.5.1. Objetivo general

Desarrollar e implementar un sistema de evasión de obstáculos en un robot humanoide de 18 GDL mediante un clasificador basado en RNA usando como retroalimentación un sensor de visión HaVimo2.0.

1.5.2. Objetivos específicos

- Ensamblar el robot de 18 GDL en la configuración Tipo A, siguiendo los lineamientos de fábrica de la empresa Robotis© para implementar la metodología propuesta.

- Ejecutar rutinas de movimientos previamente programadas usando lenguaje C embebido para la toma de decisiones del clasificador.

- Capturar una base de datos usando el sensor de visión HaVimo2.0 para el entrenamiento de la red neuronal.

- Diseñar y entrenar la RNA usando la base de datos obtenida del sensor de visión HaVimo2.0.

- Implementar la RNA en el sistema embebido para la realización de las pruebas.

- Realizar pruebas con el sistema bajo diferentes circunstancias para medir el grado de exactitud en la evasión de obstáculos.

- Analizar y discutir los resultados obtenidos mediante métodos estadísticos para validar el funcionamiento del método propuesto.

2. Fundamentación teórica

En esta sección se presentan las herramientas de software y hardware que serán usadas en el presente proyecto de Tesis.

2.1. Robot Bioloid Premium

Bioloid Premium es un kit educativo de construcción robótica que utiliza servo-bloques de Corriente Continua, mejor conocida como *Direct Current* (DC) modulares especiales, un ejemplo de este robot se muestra en la Figura 2.1.

Figura 2.1. Ilustración del kit Bioloid Premium.

Estos servos inteligentes controlados en serie permiten al usuario construir múltiples tipos de robots, como un humanoide, un cachorro, una araña o un automóvil. Además de un excelente hardware, las utilidades de software basadas en una Interfaz de Usuario Gráfica, mejor conocida como *Graphic User Interface* (GUI) se desarrollaron para facilitar el control del servo.

2.1.1. Componentes y funciones

A continuación, se enlistan los componentes del robot humanoide usado en este trabajo.

- Dynamixel AX-12A

El actuador de robot de la serie Dynamixel mostrado en la Figura 2.2, es un actuador modular inteligente que incorpora un reductor de engranajes, un motor de CC de precisión y un circuito de control con funcionalidad de red.

Figura 2.2. Dynamixel AX-12A.

A pesar de su tamaño compacto, puede producir un par elevado y está fabricado con materiales de alta calidad para proporcionar la resistencia y la resiliencia estructural necesarias para soportar grandes fuerzas externas. Algunas de sus características se muestran en la Tabla 2.1.

Tabla 2.1. Caracteristicas de los Servomotores Dynamixel.

Control de precisión	La posición y la velocidad se pueden controlar con una resolución de 1024 pasos.
Cumplimiento de conducción	El grado de cumplimiento se puede ajustar y especificar en la posición de control.
Realimentación	Se encuentran disponibles comentarios para la posición angular, la velocidad angular y el par de carga.
Sistema de alarmas	La familia de actuadores robóticos Dynamixel puede alertar al usuario cuando los parámetros se desvían de los rangos definidos por el usuario (por ejemplo, temperatura interna, torque, voltaje, etc.) y también puede manejar problemas automáticamente, por ejemplo, torque off).
Comunicación	El cableado es fácil con la conexión en cadena y admite velocidades de comunicación de hasta 1M BPS.
Control distribuido	La posición, la velocidad, el cumplimiento y el par se pueden configurar con un solo paquete de comando, lo que permite que el procesador principal control de muchas unidades Dynamixel incluso con muy pocos recursos.

Plástico de ingeniería	El cuerpo principal de la unidad está fabricado con plástico de ingeniería de alta calidad que le permite manejar cargas de alto par.
Cojinete del eje	Se utiliza un rodamiento en el eje final para garantizar que no se degrade la eficiencia con cargas externas elevadas.
LED de estado	El LED puede indicar el estado de error al usuario.
Marcos	Se incluyen un marco de bisagra y un marco de montaje lateral.

Los servomotores Dynamixel son fabricados por Robotis© y se podría agregar que destaca su capacidad de comunicación bidireccional. Esto significa que no solo reciben comandos de control, sino que también pueden enviar información de estado y retroalimentación al controlador (Robotis, 2006a).

- CM-530

El controlador CM-530 de Robotis© es una placa de control diseñada específicamente para la serie de robots Bioloid de Robotis©, misma que se puede visualizar en la Figura 2.3. Es una unidad de control que permite programar y controlar el movimiento y la interacción de los robots Bioloid.

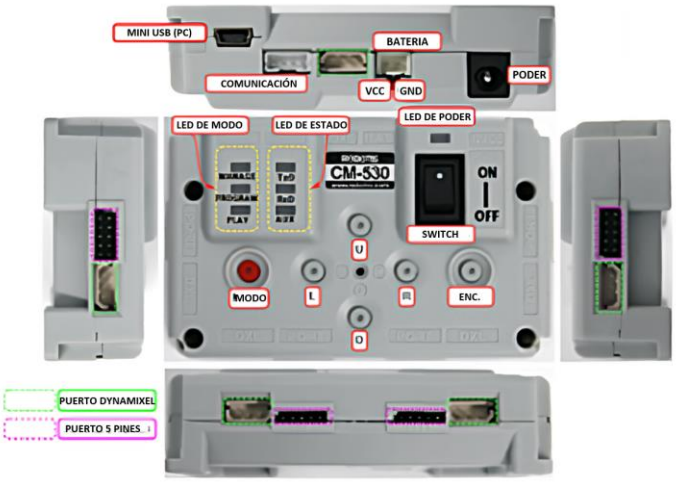

Figura 2.3. CM-530.

El CM-530 se basa en un microcontrolador ARM Cortex-M3 de 32 bits que proporciona un rendimiento potente y una capacidad de procesamiento rápida. Está

equipado con 18 puertos de expansión que permiten conectar varios servomotores Dynamixel y sensores adicionales para ampliar las capacidades del robot.

Proporciona una interfaz de programación fácil de usar, utilizando el software RoboPlus© de Robotis©, que permite a los usuarios programar el comportamiento y la secuencia de movimientos de los robots. También es compatible con el lenguaje de programación C, lo que brinda flexibilidad adicional a los programadores más avanzados.

Además, el CM-530 cuenta con una serie de sensores integrados, incluidos un acelerómetro de tres ejes, un giroscopio y un sensor de temperatura, que proporcionan información importante sobre el entorno y el estado del robot (Robotis, 2022).

2.1.2. Configuración del robot Bioloid tipo A

El robot Bioloid tipo A, mostrado en la Figura 2.4, es un humanoide miniatura de 18 GDL (12 GDL en piernas y 6 GDL en brazos), que es empleado por la comunidad científica para realizar diversos estudios relativos a humanoides; se utiliza, por ejemplo, en investigaciones sobre estrategias de control de marcha bípeda basadas en redes neuronales, y en técnicas evolutivas. Así mismo, el Bioloid se ha aplicado en el análisis de la locomoción y auto-localización de agentes de equipos de futbol soccer (J. Javier et al., 2015).

Figura 2.4. Robot humanoide Bioloid tipo A.

La altura del Bioloid es de 39,7 cm. y su peso es de 1.7 Kg, todos sus eslabones son de plástico. La cadena cinemática del robot y la ubicación de las articulaciones se muestran en la Figura 2.5.

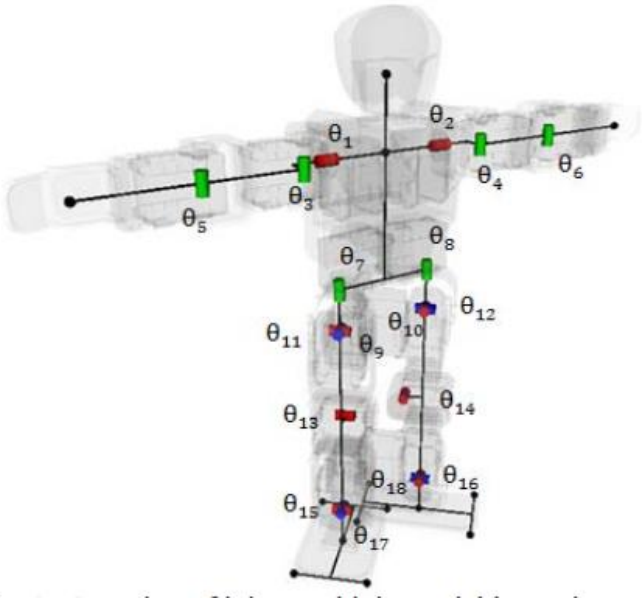

Figura 2.5. Ubicación de articulaciones y asignación de variables de articulación.

Para esta postura, todos los motores están en la posición 0°. Los ángulos de cada articulación se denotan como θ_i con $i \epsilon \{1\ 2 \dots 18\}$ (Nunez et al., 2012).

2.1.3. Protocolo de comunicación

El controlador principal se comunica con las unidades Dynamixel enviando y recibiendo paquetes de datos, ver Figura 2.6. Hay dos tipos de paquetes; el "Paquete de instrucciones" (enviado desde el controlador principal a los actuadores Dynamixel) y el "Paquete de estado" (enviado desde los actuadores Dynamixel al controlador principal).

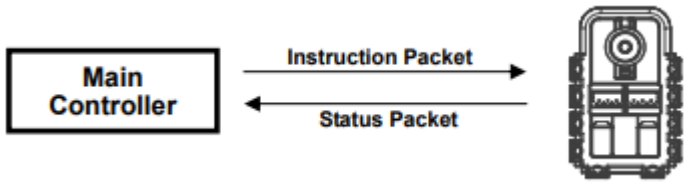

Figura 2.6. Ilustración de unidades enviadas y recibidas por los actuadores Dynamixel.

Para la conexión del sistema a continuación mostrado en la Figura 2.7, si el controlador principal envía un paquete de instrucciones con la ID configurada en N, solo la unidad Dynamixel con este valor de ID devolverá su paquete de estado respectivo y ejecutará la instrucción requerida.

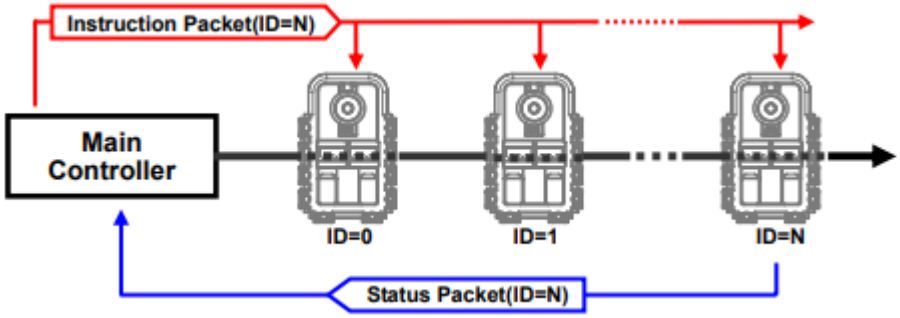

Figura 2.7. Ilustración de ID en actuadores Dynamixel.

Si varias unidades Dynamixel tienen el mismo valor de ID, varios paquetes enviados simultáneamente chocan, lo que genera problemas de comunicación. Por lo tanto, es imperativo que ninguna unidad Dynamixel comparta la misma ID en un nodo de red.

Los actuadores Dynamixel se comunican a través de comunicación serial asíncrona con 8 bits, 1 bit de parada y sin paridad (Robotis, 2006b).

2.2. Sensor de visión HaVimo2.0

El sensor de visión Havimo2.0, desarrollado por la empresa Robotis©, es un módulo compacto de visión compatible con plataformas y microcontroladores pequeños como se observa en la Figura 2.8. Este sensor se utiliza frecuentemente en competencias de

robótica. Con una resolución de 2 megapíxeles, el Havimo2.0 tiene la capacidad para capturar imágenes detalladas y procesarlas mediante algoritmos de visión artificial

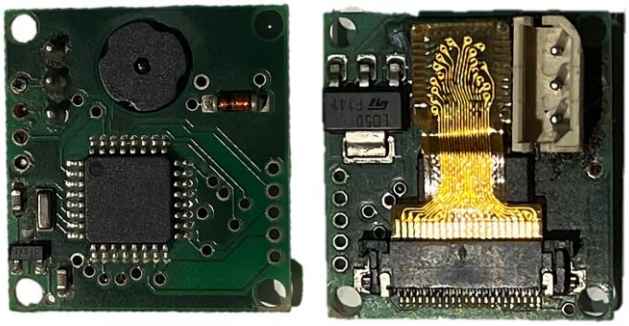

Figura 2.8. Sensor de visión HaVimo2.0.

El Havimo2.0 presenta características tanto en su hardware como en su software que lo hacen una opción versátil y potente para aplicaciones de visión en robótica. En cuanto al hardware, el sistema cuenta con un sensor de imagen OV2640 de 2 megapíxeles, así como una unidad de procesamiento principal basada en el ARM Cortex M3. Además, incluye un estado LED que indica la funcionalidad del módulo y un nuevo factor de forma que se ajusta al protocolo de comunicación de los servomotores Dynamixel.

El Havimo2.0 se comunica mediante una comunicación UART dúplex completo, lo que facilita su conexión a diferentes plataformas de microcontrolador. Esto lo convierte en una opción flexible para integrarse en proyectos de robótica y automatización (Hugo et al., 2021).

2.3. Descripción de la competencia FIRA

La FIRA, fundada por el Prof. Jong-Hwan Kim, KAIST, Corea en 1996, es la competencia de fútbol de robots más antigua del mundo. Desde sus humildes comienzos, FIRA ha crecido hasta convertirse en una importante competencia de robótica con el objetivo de utilizar los deportes como problemas de referencia para la investigación de vanguardia en robótica y otras áreas relacionadas. FIRA también incluye la competencia FIRA *Air* para robots voladores autónomos, FIRA Challenge para la investigación en robótica con grandes beneficios sociales, como robots de búsqueda y rescate urbanos,

y FIRA *Youth* para la próxima generación de investigadores. En 2018, FIRA RoboWorld Cup se celebró en Tai Chung, Taiwán y atrajo a más de 1200 participantes. Y en 2019 se llevó a cabo la FIRA *RoboWorld Cup* en Changwon, Corea del Sur, ver Figura 2.9, y al año siguiente, debido a la situación del covid-19 en el mundo, la FIRA *RoboWorld Cup* tuvo múltiples eventos en línea (FIRA, 2022).

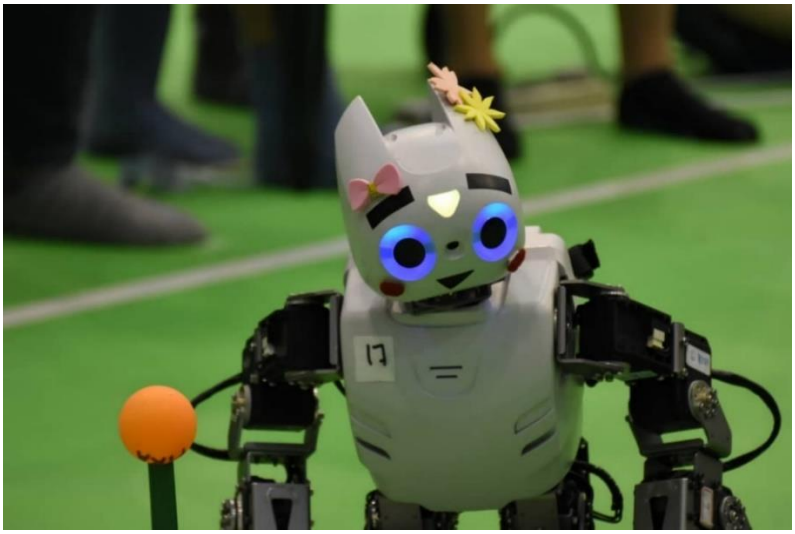

Figura 2.9. Fotografía tomada durante competencia HUROCUP FIRA 2019.

2.3.1. Obstacle Run - HuroCup

En el desafío de la carrera de obstáculos, el robot debe moverse de un extremo al otro del campo de juego lo más rápido posible. Hay tres tipos de obstáculos en el entorno: (a) obstáculos de agujeros que simulan agujeros en el suelo, (b) obstáculos de pared que no se pueden superar y (c) obstáculos de puerta que un robot suficientemente móvil puede superar arrastrándose por debajo de ellos, un diagrama de esta pista utilizada en competencia se muestra en la Figura 2.10.

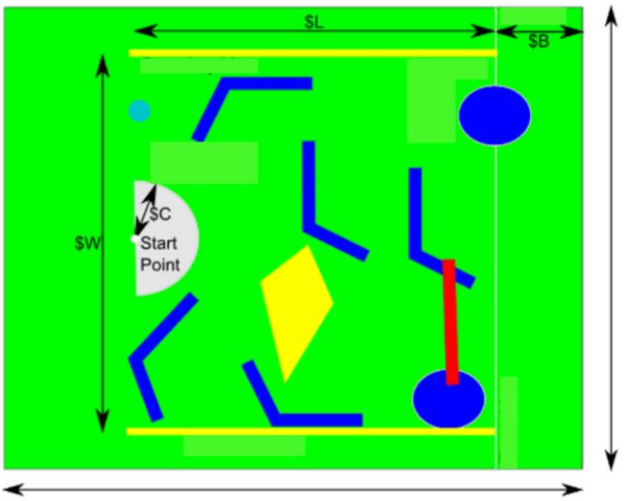

Figura 2.10. Campo de desempeño para competencia de carrera de obstáculos.

En los eventos de HuroCup, la puntuación de una sola ronda en un evento se basa en el desempeño del robot, así como en su clasificación entre los demás robots en una competencia (Karla Anhel Camarillo Gomez et al., 2021).

2.4. Redes neuronales

Las actividades de investigación desarrolladas en torno al estudio de RNA, simplemente redes neuronales o neuro-redes, están motivadas en modelar la forma de procesamiento de la información en sistemas nerviosos biológicos. Especialmente, por la forma de funcionamiento del cerebro humano, que es completamente distinta al funcionamiento de un computador digital convencional. El cerebro humano corresponde al de un sistema altamente complejo, no-lineal y paralelo. En términos sencillos lo anterior equivale a decir que puede realizar muchas operaciones simultáneamente a diferencia de los computadores comunes que son de tipo secuencial, o sea, realizan sólo una operación a la vez. En este sentido, una neuro-red es un procesador de información, de distribución altamente paralela, constituido por muchas unidades sencillas de procesamiento llamadas neuronas. La neuro-redes se caracterizan principalmente por:

- Tener una inclinación natural a adquirir el conocimiento a través de la experiencia, el cual es almacenado, al igual que en el cerebro, en el peso relativo de las conexiones interneuronales.

- Tienen una altísima plasticidad y gran adaptabilidad, son capaces de cambiar dinámicamente junto con el medio.

- Poseen un alto nivel de tolerancia a fallas, es decir, pueden sufrir un daño considerable y continuar teniendo un buen comportamiento, al igual como ocurre en los sistemas biológicos.

- Tener un comportamiento altamente no-lineal, lo que les permite procesar información procedente de otros fenómenos no-lineales.

Nuestro cerebro es un procesador de información muchísimo más eficiente que un computador. La clave de esto se encuentra en la inmensa plasticidad del cerebro, existen tareas cotidianas para el cerebro que sería impensable realizar mediante computación tradicional. Un ejemplo de esto es la capacidad reconocer a una persona en un tiempo de 100 a 200 ms. En ese breve lapso, el cerebro es capaz de procesar un patrón de información tridimensional, por ejemplo, de una persona que quizás ha cambiado de aspecto (luce distinto o simplemente envejeció) en un paisaje cambiante (que puede contener muchos otros rostros). En la actualidad, tareas mucho más simples consumen días de trabajo de los computadores más veloces. La plasticidad se percibe también en la capacidad de responder de forma correcta frente a un estímulo nunca recibido. Esa capacidad hace que cuando nos presentan por primera vez a alguien, sepamos automáticamente que es una persona y no un objeto u otro ser biológico. Debido a estas características y muchas otras, las neuro-redes se han convertido en una gran ayuda en el procesamiento de datos experimentales de comportamiento complejo. Además, su comportamiento iterativo no lineal las une de modo natural al caos y teorías de la complejidad. De hecho, las posibilidades son tan amplias que se empieza a hablar de un nuevo campo, aparte de la Biología, la Matemática y la Física: las Neurociencias. Como ya lo dijimos, lo que se desea inicialmente es imitar, al menos parcialmente, el funcionamiento del cerebro (Izaurieta & Saavedra, 1999).

2.4.1. Modelo matemático

El primer modelo matemático de una neurona artificial, creado con el fin de llevar a cabo tareas simples, fue presentado en el año 1943 en un trabajo conjunto entre el

psiquiatra y neuroanatomista Warren McCulloch y el matemático Walter Pitts. Un ejemplo de modelo neuronal con dos entradas x e y es representado en la Figura 2.11.

El mismo consta de:

- Las entradas x e y
- Los pesos sinápticos w1 y w2 correspondientes a cada entrada
- Un término aditivo b
- Una función de activación f
- Una salida z

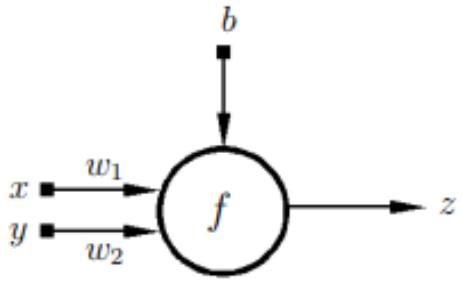

Figura 2.11. Modelo de McCulloch-Pitts para una neurona artificial.

Las entradas *x* e *y* son el estímulo que la neurona artificial recibe del entorno que la rodea, y la salida *z* es la respuesta a tal estímulo. La neurona se adapta al medio circundante y aprende de él modificando el valor de sus pesos sinápticos *w1* y *w2* y su término aditivo *b*. Estos son conocidos como los parámetros libres del modelo, dado que los mismos pueden ser modificados y adaptados para realizar una tarea determinada

En este modelo, la salida neuronal *z* está dada por (1).

$$z = f(w_1 x + w_2 y + b) \tag{1}$$

La función de activación *f* es seleccionada de acuerdo con la tarea realizada por la neurona (C. Javier et al., 2009).

De una manera general, si se requiere obtener la ecuación de una RNA, se sigue el siguiente modelo mostrado en la ecuación (2)

$$a_i^L = \sigma\left[\sum_{i=0}^{n-1} a_i^{L-1} w_i^{L-1} + b_L\right] \tag{2}$$

Donde a es la salida de la neurona en i en la capa L, σ es la función de activación a_i de $L-1$ es la salida de la neurona i en la capa anterior $L-1$, w_i de $L-1$ es el peso i de la capa anterior $L-1$ y b_L es el valor de Bias en la capa actual.

2.4.2. Entrenamiento

Un algoritmo de aprendizaje es un proceso computacional que utiliza datos de entrada para lograr una tarea deseada sin estar literalmente programado (es decir, "codificado de forma rígida") para producir un resultado particular.

Estos algoritmos son, en cierto sentido, "codificados por software" en el sentido de que alteran o adaptan automáticamente su arquitectura a través de la repetición, es decir, la experiencia, para que sean cada vez mejores en el logro de la tarea deseada. El proceso de adaptación se denomina capacitación, en el que se proporcionan muestras de datos de entrada junto con los resultados deseados. Luego, el algoritmo se configura de manera óptima para que no solo pueda producir el resultado deseado cuando se le presenten las entradas de entrenamiento, sino que también pueda generalizarse para producir el resultado deseado a partir de datos nuevos nunca vistos. Esta capacitación es la parte de “aprendizaje” del Machine Learning. El entrenamiento no tiene por qué limitarse a una adaptación inicial durante un intervalo finito. Al igual que con los humanos, un buen algoritmo puede practicar el aprendizaje “permanente” a medida que procesa nuevos datos y aprende de sus errores. Hay muchas formas en que un algoritmo computacional puede adaptarse en respuesta al entrenamiento. Los datos de entrada se pueden seleccionar y ponderar para proporcionar los resultados más decisivos. El algoritmo puede tener parámetros numéricos variables que se ajustan mediante optimización iterativa. Puede tener una red de posibles vías computacionales que organiza para obtener resultados óptimos. Puede determinar distribuciones de

probabilidad a partir de los datos de entrada y utilizarlos para predecir resultados (Naqa & Murphy, 2015).

2.4.3. Machine Learning

El ideal del Aprendizaje Máquina mejor conocido como *Machine Learning*, por sus siglas en inglés, es emular la forma en que los seres humanos (y otras criaturas sensibles) aprenden a procesar señales sensoriales (entrada) para lograr un objetivo. Este objetivo podría ser una tarea de reconocimiento de patrones, en la que el aprendiz quiere distinguir manzanas de naranjas. Cada manzana y naranja es única, pero aún podemos (generalmente) distinguir una de la otra. En lugar de codificar una máquina con muchas, muchas representaciones exactas de manzanas y naranjas, se puede programar para aprenda a distinguirlos a través de la experiencia repetida con manzanas y naranjas reales. Este es un buen ejemplo de aprendizaje supervisado, en el que cada ejemplo de entrenamiento de datos de entrada (color, forma, olor, etc.) se empareja con su etiqueta de clasificación conocida (manzana o naranja). Le permite al aprendiz lidiar con similitudes y diferencias cuando los objetos a clasificar tienen muchas propiedades variables dentro de sus propias clases, pero aún tienen cualidades fundamentales que los identifican. Lo más importante es que el aprendiz exitoso debe poder reconocer una manzana o una naranja que nunca ha visto.

Un segundo tipo de aprendizaje automático es el llamado algoritmo no supervisado. Esto podría tener el objetivo de intentar lanzar un dardo a la diana. El dispositivo (o humano) tiene una variedad de GDL en el mecanismo que controla la trayectoria del dardo. En lugar de intentar programar exactamente la cinemática a priori, el alumno practica el lanzamiento del dardo. Para cada ensayo, los GDL cinemáticos se ajustan para que el dardo se acerque más y más a la diana. Esto no está supervisado en el sentido de que el entrenamiento no asocia una configuración de entrada cinemática particular con un resultado particular. El algoritmo encuentra su propio camino a partir de los datos de entrada de entrenamiento. Idealmente, el lanzador de dardos capacitado podrá ajustar la cinemática aprendida para adaptarse, por ejemplo, a un cambio en la posición del objetivo.

Un tercer tipo de Machine Learning es el aprendizaje semisupervisado, en el que una parte de los datos está etiquetada y otra parte no está etiquetada. En tal escenario, la parte etiquetada se puede usar para ayudar al aprendizaje de la parte no etiquetada. Este tipo de escenario se presta a la mayoría de los procesos en la naturaleza y emula más de cerca cómo los humanos desarrollan sus habilidades.

Hay dos características particularmente importantes para un algoritmo exitoso. En primer lugar, puede sustituir el laborioso y repetitivo esfuerzo humano. En segundo lugar, y lo que es más importante, puede aprender potencialmente patrones más complicados y sutiles en los datos de entrada que los que puede hacer el observador humano promedio. Ambas ventajas son importantes para la radioterapia. Por ejemplo, el contorno diario de tumores y órganos en riesgo durante la planificación del tratamiento es un proceso lento de reconocimiento de patrones que se basa en la familiaridad y experiencia del observador con la apariencia de la anatomía en las imágenes de diagnóstico. Sin embargo, esa familiaridad tiene sus límites y, en consecuencia, hay incertidumbre y variabilidad entre observadores en los contornos resultantes.

Es posible que un algoritmo para el contorno pueda captar sutilezas de textura o forma en una imagen o incorporar simultáneamente datos de múltiples fuentes o combinar la experiencia de numerosos observadores y así reducir la incertidumbre en el contorno (Naqa & Murphy, 2015).

3. Metodología

Se comprende una metodología como aquel conjunto de métodos o procedimientos que se llevan a cabo con el objetivo de obtener un resultado determinado. Dentro del actual proyecto de Tesis se propone el proceso mostrado en el diagrama de la Figura 3.1 para alcanzar todos los objetivos planteados.1

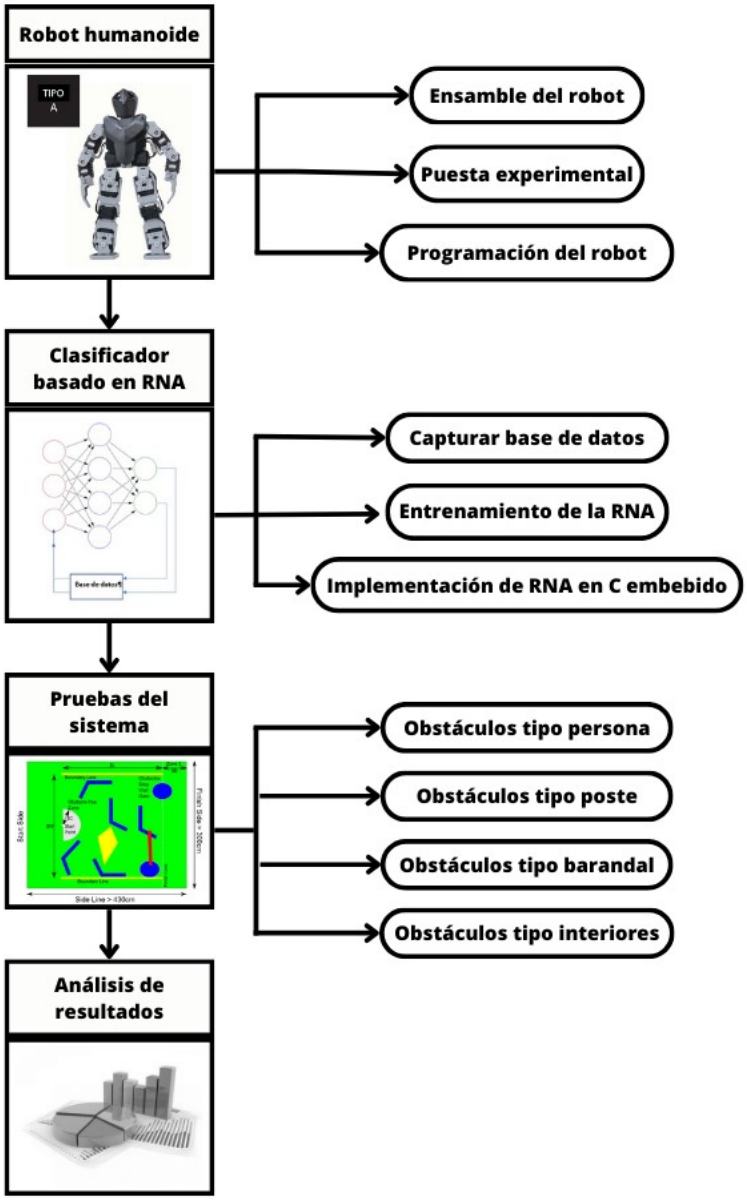

Figura 3.1. Diagrama de la metodología propuesta para llevar a cabo el presente proyecto de Tesis.

Como se puede apreciar en la Figura 3.1, la metodología planteada se compone de 4 secciones principales: robot humanoide, clasificador basado en RNA, pruebas del sistema y análisis de resultados. Además, dentro de estas secciones se pueden encontrar subsecciones que ayudan a especificar en que consiste cada una de ellas y dividen la metodología en partes más pequeñas.

A continuación, se enlista cada una de las secciones y subsecciones con la descripción de lo que se llevó a cabo durante su etapa respectiva.

3.1. Robot humanoide

El kit Bioloid Premium de la empresa Robotis©, es un artículo comercial manejado y distribuido por la misma empresa alrededor de todo el mundo, el mismo incluye dentro de sí 18 servomotores de la serie Dynamixel AX-12A, un controlador CM-530, una batería recargable para el uso de los diferentes robots de manera autónoma, un sensor de infrarrojos para la detección de obstáculos, un sensor de distancia, un sensor giroscopio y la posibilidad de comunicación a través de mismos infrarrojos. Con este kit se pueden armar tres tipos diferentes de robots humanoides, un dinosaurio, un perro, un escorpión y diversos hexápodos. Este robot fue el utilizado para el desarrollo del presente proyecto de Tesis como portador del sistema para la evasión de obstáculos basado en RNA.

3.1.1. Ensamble del robot

La importancia de un buen armado y conexión de cualquier robot va mucho más allá de cuestiones estéticas, dado que está directamente relacionado al funcionamiento que tiene el sistema, su buen desempeño o incluso puede derivar directamente en fallas catastróficas para el hardware del mismo sistema, esto se debe a que usualmente las fallas ocurren cuando un defecto se presenta en el entorno de funcionamiento del robot, lo que conlleva a datos o resultados no esperados y producen una ejecución inadecuada, lo que se considera una falla que muchas veces en sistemas tan complejos como lo es un robot, puede producir daños al hardware irreversibles. Debido a lo anterior, a continuación, se explica el armado del robot utilizado durante el desarrollo del proyecto a detalle.

Como se mencionó anteriormente, el kit Bioloid Premium de la empresa Robotis© nos ofrece la posibilidad de armar tres diferentes robots humanoides; humanoide tipo A, humanoide tipo B y humanoide tipo C, también se pueden encontrar en el manual de usuario administrado por la misma empresa y que es recomendado utilizar al adquirir un kit correspondiente, los modelos previamente mencionados se pueden apreciar a continuación en la Figura 3.2, en conjunto con sus características generales, dicha imagen es parte del mismo manual de usuario mencionado.

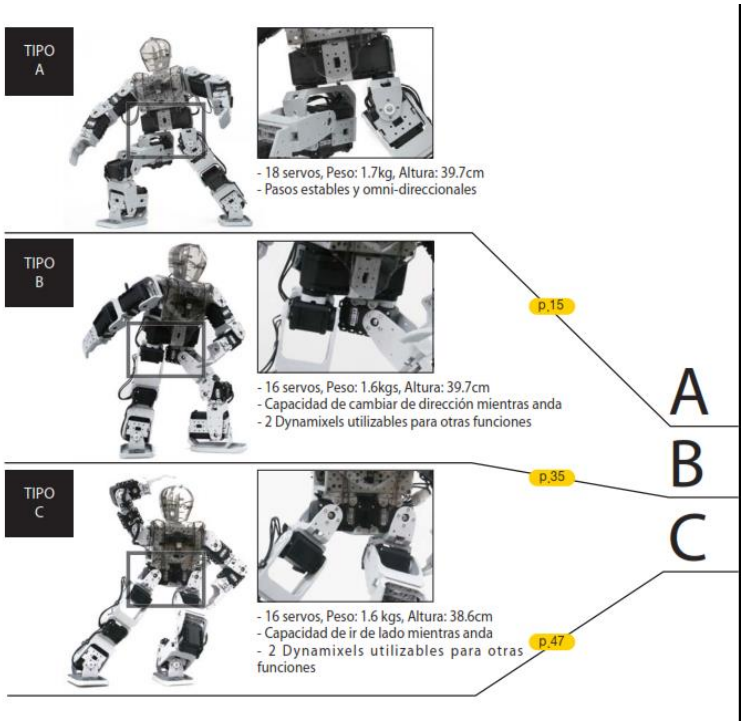

Figura 3.2. Posibles armados para el robot humanoide con el kit Bioloid Premium.

El modelo seleccionado de armado de robot humanoide seguido en base al manual de usuario previamente mencionado es el Tipo A, apreciable en la Figura 3.2, tal y como se puede ver, el armado conlleva la utilización de los 18 servomotores incluidos en el kit, mismos servomotores le brindan al robot 18 GDL para el movimiento del sistema, mencionando otras características, el robot tendría 39.7cm de altura y 1.7kg de peso total. Los motivos para utilizar este modelo de robot humanoide sobre los otros dos, son los 2 GDL más que ofrece el armado, además de la estabilidad que proporciona y la

capacidad de moverse durante su caminado en cualquier dirección, dichas características también son encontradas en el manual de usuario.

Iniciando con el armado del robot humanoide Tipo A, se procede a seguir los 30 pasos incluidos en el manual en la sección de montaje para dicho armado, durante los primeros 4 pasos, se lleva a cabo el ensamble de uno de los brazos del robot, cada uno de los pasos de esta sección muestra la posición en la que deben colocarse las estructuras y servomotores, además de la misma tornillería que será utilizada para fijar cada una de las partes incluidas, en la Figura 3.3 se pueden apreciar un par de pasos que son encontrados en el manual para el montaje del robot y que son pertenecientes específicamente a la construcción de uno de los brazos del mismo.

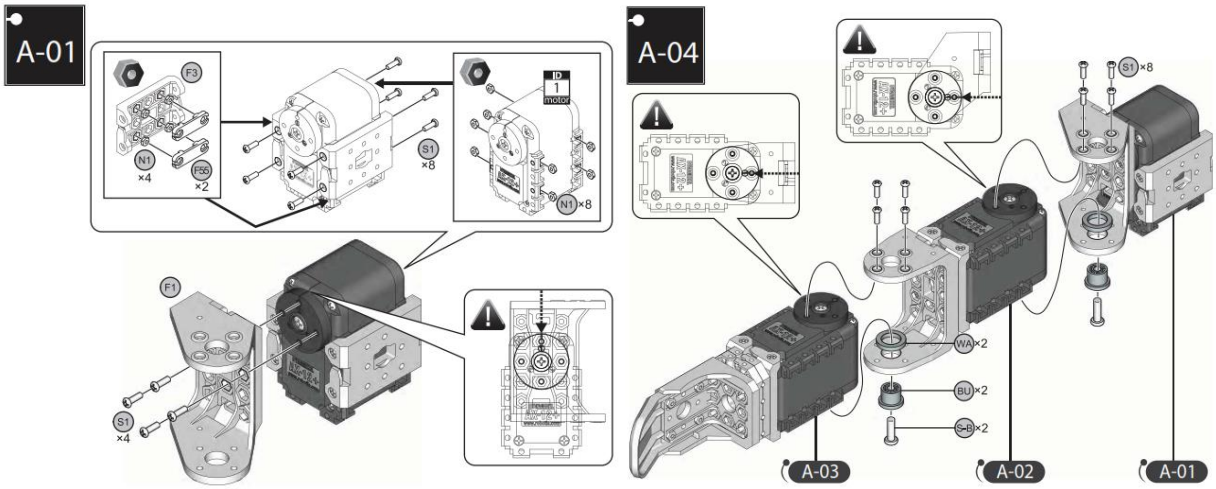

Figura 3.3. Pasos 1 y 4 incluidos en manual de montaje del kit Bioloid Premium.

De una manera bastante parecida y deducible, siguiendo con el armado del robot, los siguientes 4 pasos del manual de ensamble (pasos 5 al 8), contienen los pasos para llevar a cabo el armado del otro brazo del robot, con los cuales ya se tendrían los brazos izquierdo y derecho del armado final, solamente faltaría la unión del torso con los mismos. La Figura 3.4, muestra un par de pasos incluidos en el manual de ensamble para el otro brazo previamente mencionado.

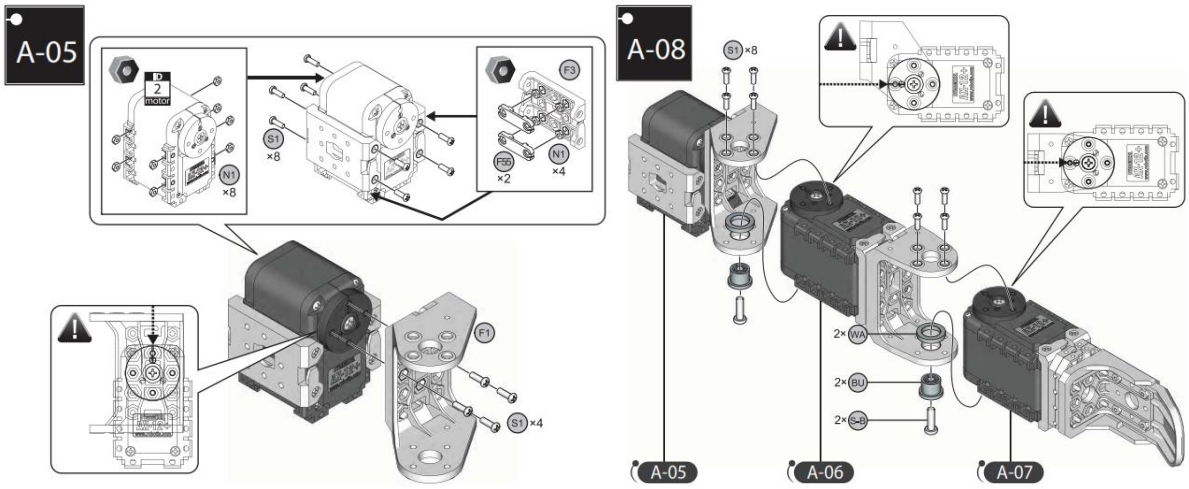

Figura 3.4. Pasos 5 y 8 incluidos en manual de montaje del kit Bioloid Premium.

Siguiendo con los pasos, esta vez se siguen los pasos 9 al 20, en donde se describe la construcción de ambas piernas del robot, así como la unión de estas, como primera parte se muestra en la Figura 3.5 la construcción de la pierna izquierda y derecha (pasos 9 al 14).

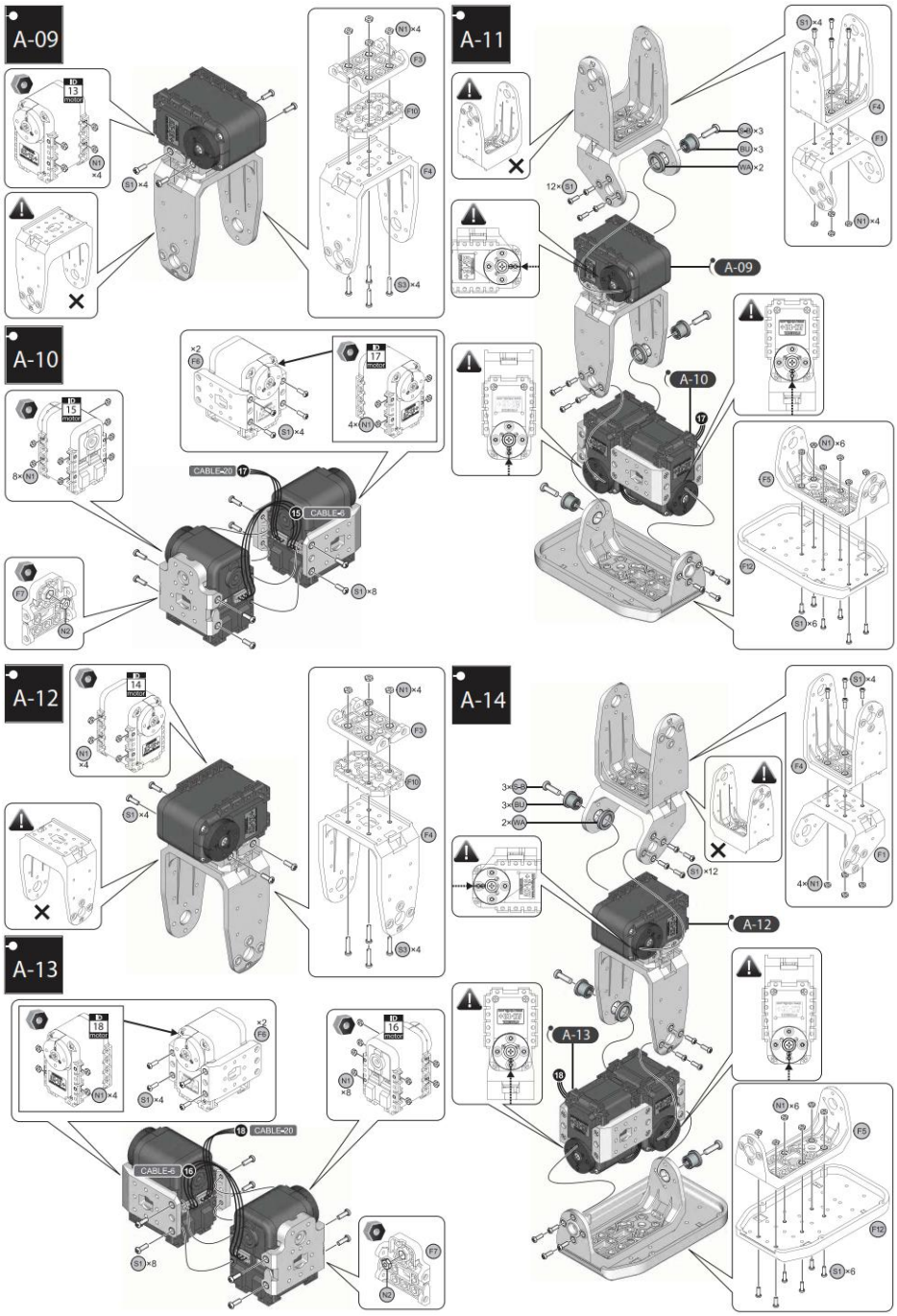

Figura 3.5. Pasos 9-14 incluidos en manual de montaje del kit Bioloid Premium.

Posterior a tener ambas piernas armadas, se procede a la construcción de la parte baja del torso para la unión de estas, esto se muestra en la Figura 3.6.

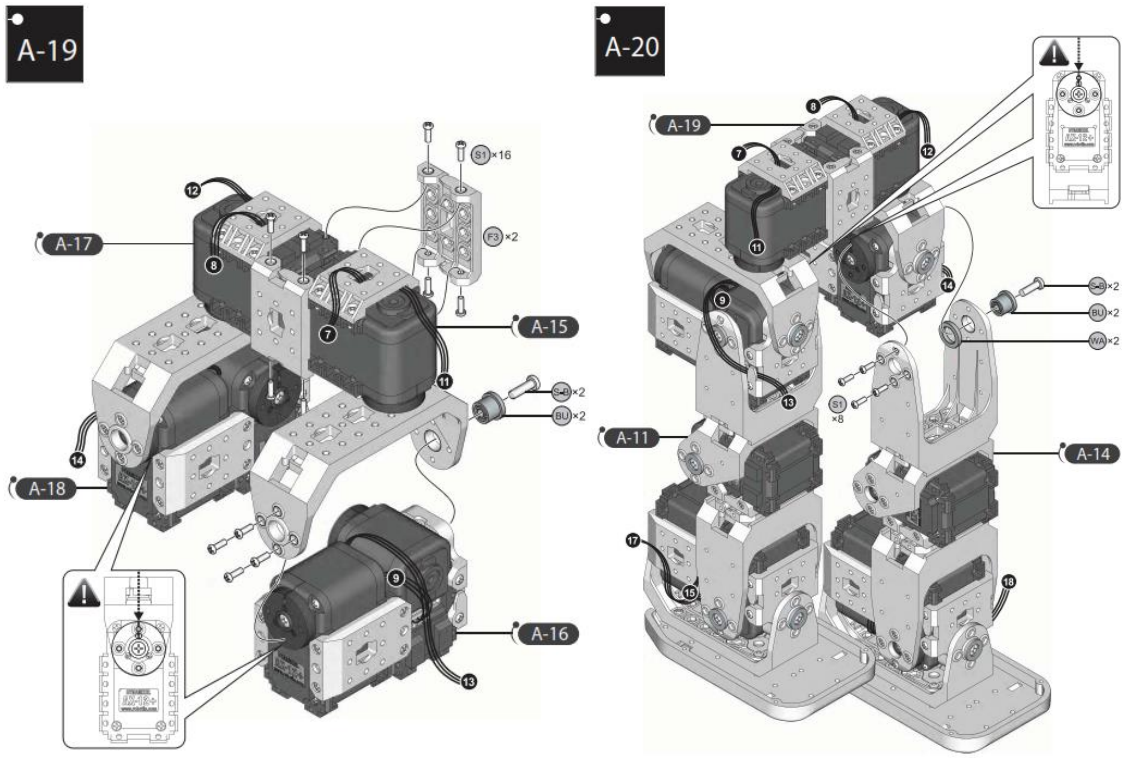

Figura 3.6. Pasos 19-20 incluidos en manual de montaje del kit Bioloid Premium.

Una vez que se tiene la parte baja del robot armado y la unión de cada una de sus partes, se procede a continuar con la construcción del torso para llevar a cabo la unión de los brazos, esta parte del robot se muestra en la Figura 3.7.

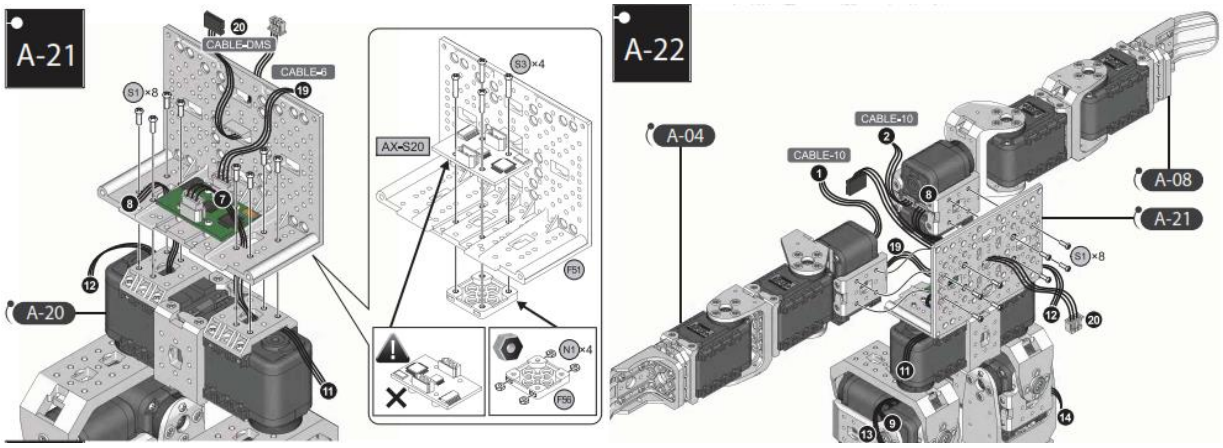

Figura 3.7. Pasos 21-22 incluidos en manual de montaje del kit Bioloid Premium.

La misma parte del torso, cuya construcción se inició en la Figura 3.6 ayuda en su construcción total como soporte para el controlador CM-530 que será utilizado durante el desarrollo del proyecto para el control del sistema, esta continuación de pasos se muestra en la Figura 3.8.

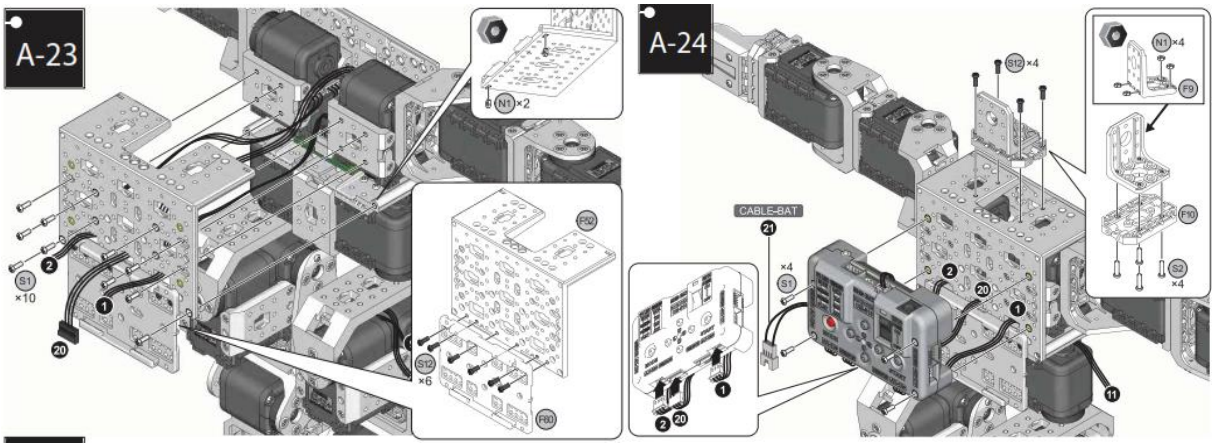

Figura 3.8. Pasos 23-24 incluidos en manual de montaje del kit Bioloid Premium.

Como una de las ultimas partes para el ensamble del robot humanoide, se lleva a cabo la instalación y conexión de cada uno de los cables que ayudan para la comunicación entre motores y a la vez, la comunicación de estos con el controlador CM-530, la Figura 3.9, ejemplifica uno de estos pasos previamente mencionados, en el cual se puede observar la descripción de cada cable que se utiliza y el lugar o elementos que conecta.

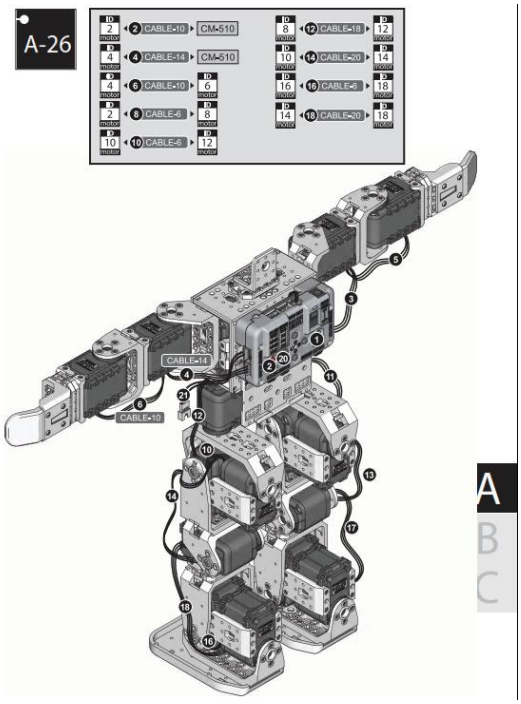

Figura 3.9. Paso 26 incluido en manual de montaje del kit Bioloid Premium.

Finalmente se tienen un par de pasos que muestran la colocación de un par de piezas estéticas, mismas simulan una cabeza y un pecho en el robot, aunque no afectan en el funcionamiento del mismo, estos pasos se muestran en la Figura 3.10.

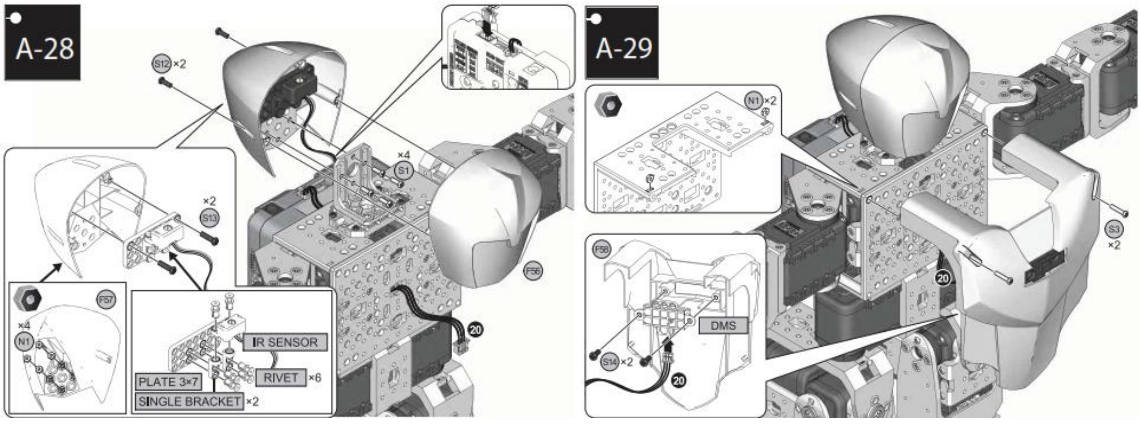

Figura 3.10. Pasos 28-29 incluidos en manual de montaje del kit Bioloid Premium.

Si se siguen cada uno de los 30 pasos que se incluyen en el manual de ensamble del robot humanoide, mismos que son descritos anteriormente, se tiene como resultado el robot Bioloid Premium en su Tipo A, que contiene 18 servomotores y posee las

características mencionadas al principio de esta sección. A continuación, en la Figura 3.11 se muestra el resultado del robot armado por el usuario para el desarrollo del presente proyecto.

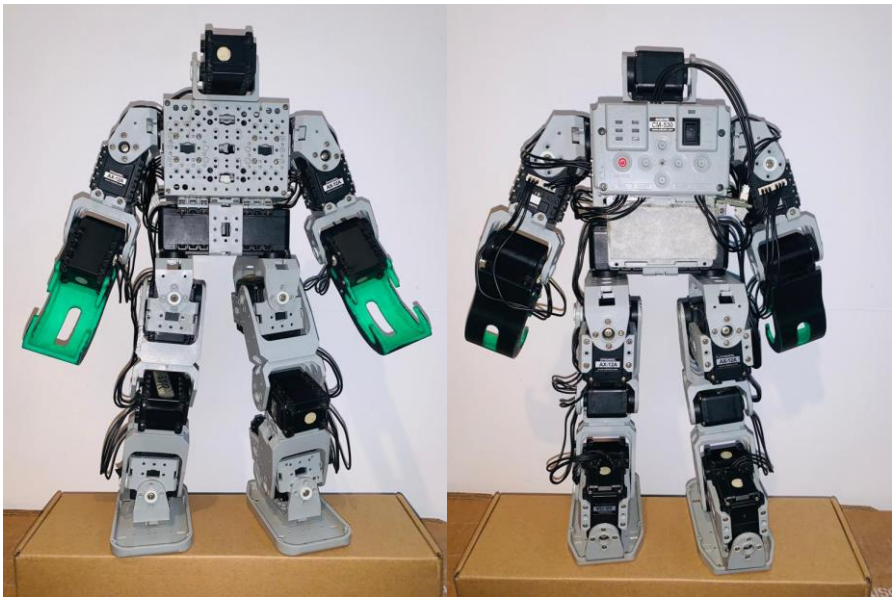

Figura 3.11. Resultado del robot humanoide Tipo A del kit Bioloid Premium.

Se observa en la figura anterior que las estructuras que son solamente para cuestiones estéticas no fueron incluidas, y se recalca que esto no tiene alguna repercusión en el funcionamiento del robot.

3.1.2. Puesta experimental

Para la creación de la pista de obstáculos que será utilizada para las pruebas del sistema que está siendo desarrollado durante el presente proyecto, se basa en obstáculos parecidos a los que se encuentran o han formado parte de la competencia de evasión de obstáculos llevada a cabo en FIRA HuroCup. Cada uno de los obstáculos fabricados representa algún obstáculo (según la forma) que puede ser encontrado en la vida real, dado que el objetivo es simular lo mayor posible el comportamiento del humanoide a un robot de tamaño humano real. Por ello, se eligieron 4 obstáculos, los cuales se describen a continuación.

Obstáculo tipo poste:

Este obstáculo, como su nombre lo dice, simula la interacción del humanoide con postes de la vida real, los cuales tienen las características de ser altos y angostos, por ello se creó un obstáculo con una altura de 40cm y una anchura de 12cm aproximadamente, mismo obstáculo fue caracterizado con el color verde y se puede apreciar en la Figura 3.12 en conjunto con el objeto del que se planteó simular.

Figura 3.12. Objeto tipo poste diseñado y ejemplo de objeto aparentado.

En la Figura 3.12, del lado izquierdo se observa el obstáculo usado para las pruebas en este proyecto de Tesis y del lado derecho se observa el obstáculo que representa de la vida real.

Obstáculo tipo interiores

Al intentar involucrar a los robots en la vida diaria de los humanos, es imposible no considerar la presencia de obstáculos que sean similares a los interiores que se pueden encontrar en casas, oficinas, tiendas, supermercados, etc. Estos pueden ser muebles, mesas, sillas, camas, entre algunos otros, por lo cual se caracterizó un obstáculo que simula las propiedades de estos cuerpos, y que normalmente son bastante anchos y considerablemente altos. Este obstáculo posee una altura 40cm y un ancho de 35cm, además de poseer el color azul y se puede apreciar en la Figura 3.13.

Figura 3.13. Objeto tipo interiores diseñado y ejemplo de objeto aparentado.

En la Figura 3.13 mostrada anteriormente, del lado izquierdo se puede observar el obstáculo usado para las pruebas en este proyecto de Tesis y del lado derecho se observa el obstáculo que representa en la vida real.

Obstáculo tipo barandal

De igual manera se deben considerar los obstáculos tipo barandal, que normalmente se pueden encontrar marcando algún límite con respecto a alguna zona en la que se deba tener cuidado o simplemente a la cual no se deba acceder por seguridad propia, normalmente este tipo de obstáculos son horizontales, bastante alargados en este sentido y son angostos verticalmente. Por ello se caracterizó un objeto que simule obstáculos de esta clase, el obstáculo creado posee un ancho de 40cm y un alto de 12cm, de igual manera fue levantado por un soporte para que no estuviera a ras de suelo, y finalmente fue caracterizado por el color amarillo y se puede apreciar en la Figura 3.14 en conjunto con el objeto del que se planteó simular.

Figura 3.14. Objeto tipo barandal diseñado y ejemplo de objeto aparentado.

En esta Figura 3.14, podemos apreciar de manera similar a los obstáculos anteriores, del lado izquierdo el objeto a utilizar en el presente proyecto y del lado derecho el objeto al que se le asemeja de la vida real.

Obstáculo tipo persona

El cuarto y último obstáculo fue diseñado en base a la idea de que el incluir a los robots en un futuro en la vida cotidiana de los seres humanos representa una completa interacción con los mismos, es por ello por lo que es indispensable que el robot reconozca a una persona de manera efectiva. De esta manera se caracterizó al obstáculo tipo persona con un círculo debido a que es muy normal que la mayor masa de las personas se encuentre en su centro y las extremidades vayan reduciéndose en cantidad de esta. Este obstáculo posee un radio de 12cm y fue caracterizado por el color rojo, se puede apreciar en la Figura 3.15.

Figura 3.15. Objeto tipo persona diseñado y ejemplo de objeto aparentado.

Nuevamente, con este último obstáculo de los que serán utilizados en el actual proyecto de Tesis, a la izquierda de la Figura 3.15 se muestra el obstáculo que se utilizará en las pruebas y a la derecha se muestra un posible objeto al que representa de la vida real.

Una vez que se tienen todos los obstáculos caracterizados y diseñados se procede a comenzar con la programación del controlador para el funcionamiento del robot en conjunto con el sensor de visión HaVimo2.0, para la detección de obstáculos.

3.1.3. Programación del robot

De una manera muy parecida a como la empresa Robotis© administra el manual de usuario para el ensamble del robot humanoide que se sigue para el desarrollo del presente proyecto de Tesis, la misma empresa también administra una guía de ayuda para tus primeros pasos en cuanto a la programación del humanoide, dado que brinda un software para la interacción con el mismo, RoboPlus©. En esta sección se hace mención y descripción del proceso seguido para la programación de movimientos y/o rutinas del humanoide.

Para tener acceso a la guía inicial de la empresa Robotis© basta con acceder a la página principal de la misma e ir a la sección Robotis Premium, tal y como se espera, las

primeras instrucciones en esta sección caracterizan y describen el procedimiento a seguir para el ensamble del robot, según sea el armado que el usuario decida, posteriormente a tener tu robot armado, se procede a programarlo. Para ello se instala, como primera parte, el software RoboPlus©, un software libre que se puede descargar desde la misma página de la empresa en donde se encuentra la guía (ROBOTIS, 2023), este software es un entorno donde se centralizan todos los programas para el uso y monitoreo del robot, cada uno de estos programas se pueden observar en la Figura 3.16 en conjunto con una breve descripción de estos.

Figura 3.16. Interfaz principal de software RoboPlus© con descripción.

Seguido de tener el software instalado y el robot armado, se procede a averiguar si las conexiones que se llevaron a cabo durante el ensamble del mismo fueron acertadas, para ello se accede al programa RoboPlus Manager© y se conecta el humanoide (desde el controlador) al ordenador, en esta sección del software se observan los elementos conectados y detectados por parte del microcontrolador tal y como se puede observar en la Figura 3.17, específicamente dentro del recuadro señalado en color verde, en esta ocasión se muestra la detección de parte del programa con el controlador, así como los 18 servomotores de la serie Dynamixel AX-12A que están siendo utilizados para el armado del humanoide especifico del proyecto.

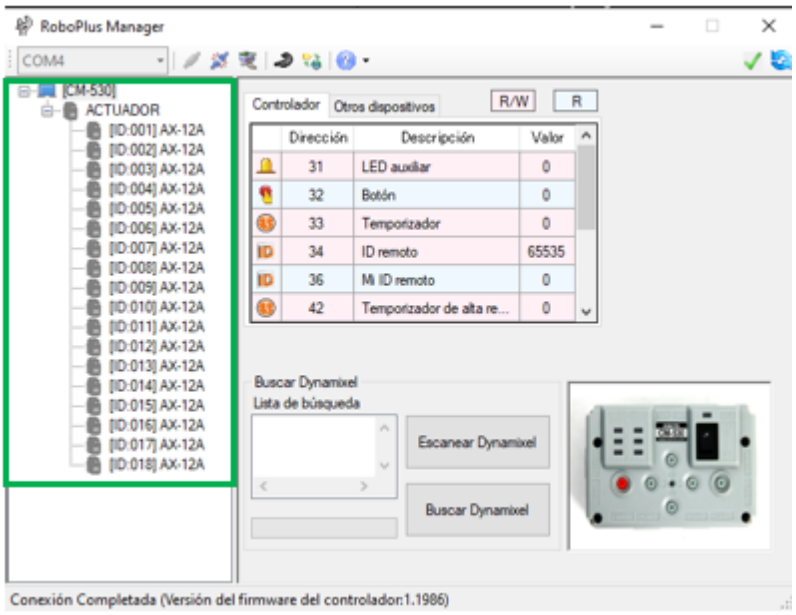

Figura 3.17. Interfaz del programa RoboPlus Manager©.

Dentro del mismo programa del software RoboPlus© se puede visualizar el estado de cada uno de los elementos conectados, en este caso, el estado de cada uno de los servomotores. Dentro de las características que pueden ser monitoreadas se tiene la posición actual, velocidad, carga, tensión, temperatura, límites de movimiento, conexión, entre algunas otras, permitiendo el completo seguimiento del comportamiento durante el funcionamiento de estos. Como ejemplo, se muestra la Figura 3.18, en donde se puede observar uno de los 18 servomotores usados durante el desarrollo del proyecto correspondiente, en conjunto con algunas de sus características, incluyendo su dirección, descripción y valor de estas.

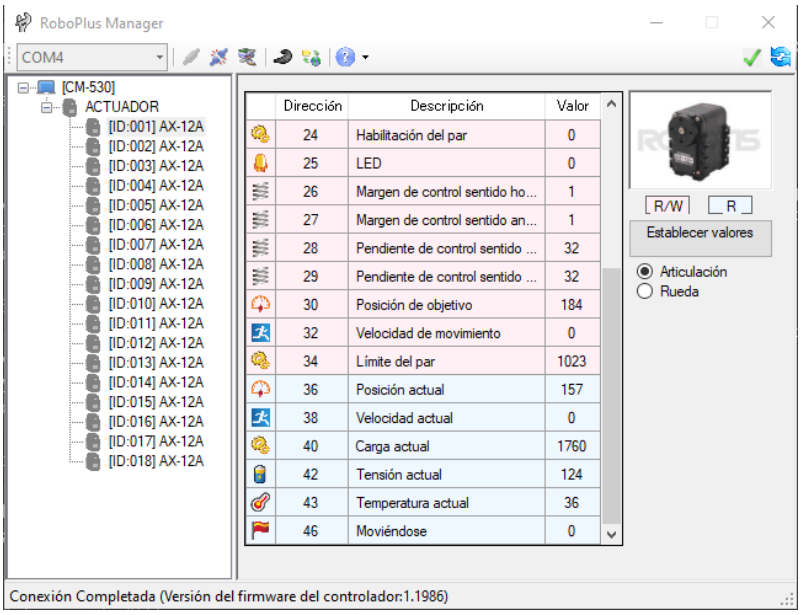

Figura 3.18. Características mostradas para monitorio del servomotor en RoboPlus Manager©.

Una vez que se comprueban las conexiones del sistema armado durante la etapa de ensamble del robot y que cada uno de los elementos que componen al mismo se encuentran en buen estado y con una correcta identificación de parte del controlador, se procede con la programación del humanoide considerando que como consecuencia del buen desarrollo del ensamble no se tendrá ningún problema o error que pueda ocasionar una falla catastrófica durante la etapa de programación de movimientos y/o rutinas.

De parte de la empresa Robotis©, al igual que se maneja la guía para acceder al software de programación, también existen algunas rutinas que son manejadas por la misma empresa para que se termine de comprobar el funcionamiento del humanoide y de igual manera se puedan utilizar para el proyecto para el cual se adquiere el kit robótico. Estas rutinas son completamente descargables de manera libre y pueden ser leídas y ejecutadas con el programa RoboPlus Motion©, que de igual manera que RoboPlus Manager©, es un programa que pertenece al software RoboPlus©, pero en lugar de funcionar para el monitoreo de los elementos que se tengan conectados al controlador, éste funciona para la interacción con los servomotores conectados en cuanto a la posición que poseen, este programa permite la programación de diferentes posiciones para los diversos servomotores conectados que se pueden enviar al controlador y a la vez a los servomotores. Al reproducirse diferentes posiciones de

manera continua a través del tiempo se puede obtener un movimiento continuo de los mismos, lo que permite a este programa la capacidad de almacenar y ejecutar rutinas que sean de ayuda para el desarrollo del proyecto para el cual fue adquirido el kit Bioloid Premium. La Figura 3.19 mostrada a continuación muestra la interfaz principal del programa previamente mencionada y enseguida una breve descripción de cada una de sus secciones, cabe destacar que de igual manera se muestran los movimientos de fábrica que fueron mencionados.

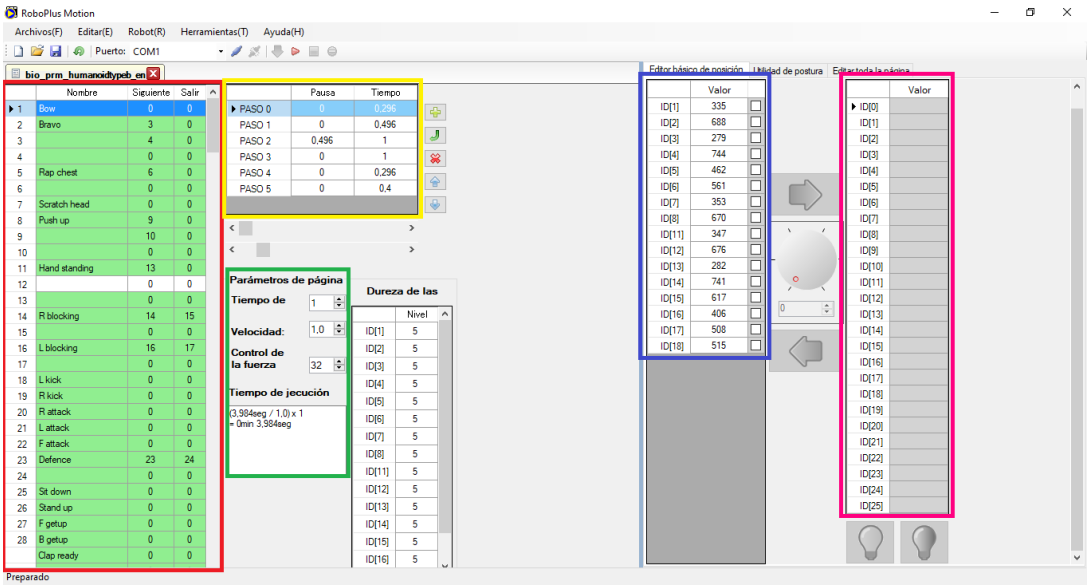

Figura 3.19. Interfaz del programa RoboPlus Motion©.

De acuerdo con la Figura 3.19, a continuación, se describe cada una de las secciones de la interfaz señaladas en cada uno de los recuadros.

- El área encerrada en un recuadro color rojo representa las hojas del archivo de rutinas, cada archivo puede tener hasta 255 hojas cada uno y pueden ser almacenadas y enviadas al controlador CM-530.

- El área encerrada en un recuadro color amarillo representa los pasos que contiene cada una de las hojas que se conservan en los archivos, cada hoja puede tener hasta 7 pasos.

- El área encerrada en un recuadro color verde representa las propiedades que caracterizan las rutinas que son enviadas, se pueden asignar diferentes propiedades a cada una de las hojas en conjunto, sin embargo, no a cada uno de los pasos de manera individual.

- El área encerrada en un recuadro color azul representa las posiciones para cada uno de los motores que son controlados en el sistema, aquí se introducen los valores que se desean enviar o modificar en los servomotores del humanoide desde el computo.

- El área encerrada en un recuadro color rosa representa los valores que posee el sistema que está conectado de manera actual, dichos valores pueden ser leídos, interpretados, copiados y hasta guardados según la conveniencia o utilización del usuario.

Al obtenerse las rutinas que son proporcionadas por la empresa Robotis© se procede a la búsqueda de estas que puedan ser utilizadas para la evasión de obstáculos, para lo cual se encontraron 6 rutinas principales que se describen a continuación:

Caminado hacía delante: Principal movimiento que caracteriza la deambulación del humanoide, este movimiento se mantiene hasta que sea encontrado algún obstáculo.

Caminado hacia atrás: Caracteriza el movimiento inverso del caminado hacia delante, que puede ser utilizado para la corrección del camino.

Lateral derecho: Caracteriza el desplazamiento hacia uno de los lados del robot de manera lateral, puede ser utilizado para evadir obstáculos que no son tan largos.

Lateral izquierdo: Caracteriza el desplazamiento similar al lateral derecho, solo que hacía el otro lado,

Giro sobre su propio eje en sentido de las manecillas del reloj: Caracteriza el giro del robot, permitiendo evadir obstáculos largos en combinación con el caminado hacía delante.

Giro sobre su propio eje en sentido contrario de las manecillas del reloj: Caracteriza el giro similar al giro sobre su propio eje en sentido de las manecillas del reloj, solo que, en sentido contrario, permitiendo evadir obstáculos hacia cualquiera de los dos lados.

Cada una de las rutinas anteriores son probadas en el robot para corroborar el correcto funcionamiento del humanoide, de cada una de las conexiones y a su vez, de las mismas rutinas. Cada una de estas posee 4 hojas con 7 pasos cada una, lo que arroja un total de 28 posiciones para cada uno de los servomotores que son comprendidas durante la rutina completa.

Para pasar estas rutinas a C embebido, que será la manera de programar el humanoide de manera definitiva para el proyecto debido a su mejor procesamiento por encima de RoboPlus©, se extrae cada uno de los valores de las posiciones para cada uno de los servomotores y se forma una matriz de 18x28, en donde las 18 filas representa cada uno de los servomotores y el valor que será enviado a cada uno de ellos, y las 28 columnas son cada posición que se emitirá a cada uno de los servomotores durante la rutina. Después de tener cada una de las 6 matrices correspondientes a cada rutina que será utilizada, son declaradas en C embebido, la Figura 3.20 muestra un ejemplo de una de estas matrices.

```
/*MATRIZ DE CAMINADO HACIA ADELANTE*/
int WF [NoMot][NMWF] = {    {   300 ,   303 ,   302 ,   298 ,   290 ,   280 ,   268 ,   254 ,   239 ,   22
        {   853 ,   856 ,   855 ,   851 ,   843 ,   833 ,   821 ,   807 ,   792 ,   776 ,   762 ,   748 ,
        {   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,   279 ,
        {   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,   744 ,
        {   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,   462 ,
        {   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,   561 ,
        {   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,   358 ,
        {   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,   666 ,
        {   501 ,   496 ,   490 ,   482 ,   473 ,   467 ,   465 ,   467 ,   473 ,   482 ,   490 ,   496 ,
        {   510 ,   505 ,   504 ,   510 ,   519 ,   527 ,   530 ,   527 ,   519 ,   510 ,   504 ,   505 ,
        {   363 ,   372 ,   373 ,   361 ,   341 ,   319 ,   299 ,   287 ,   283 ,   286 ,   291 ,   293 ,
        {   721 ,   715 ,   709 ,   702 ,   696 ,   691 ,   688 ,   686 ,   685 ,   684 ,   683 ,   680 ,
        {   254 ,   262 ,   262 ,   245 ,   222 ,   203 ,   196 ,   203 ,   222 ,   245 ,   262 ,   262 ,
        {   777 ,   775 ,   771 ,   766 ,   761 ,   757 ,   756 ,   757 ,   761 ,   766 ,   771 ,   775 ,
        {   675 ,   675 ,   677 ,   682 ,   685 ,   681 ,   669 ,   649 ,   627 ,   607 ,   595 ,   596 ,
        {   402 ,   398 ,   395 ,   394 ,   393 ,   392 ,   390 ,   387 ,   382 ,   376 ,   369 ,   363 ,
        {   501 ,   496 ,   491 ,   486 ,   482 ,   480 ,   479 ,   480 ,   482 ,   486 ,   491 ,   496 ,
        {   510 ,   505 ,   500 ,   496 ,   493 ,   491 ,   490 ,   491 ,   493 ,   496 ,   500 ,   505 ,
        {   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,
        {   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,   512 ,
```

Figura 3.20. Fragmento de matriz de posiciones de movimiento para caminado hacía delante en C embebido.

Posteriormente a tener la matriz, se genera una función para la reproducción y envío de cada uno de los movimientos a los diferentes motores, en el caso del movimiento para caminado hacia delante, su función se puede apreciar en la Figura 3.21.

```
void WalkForward(int c)
{
    for( i=0; i<NoMot; i++ )
            {
                    id[i] = i+1;
                    }
    a=0;
    while(a<c)
        {
        for(j=0; j<NMWF; j++)
        {
            dxl_set_txpacket_id(BROADCAST_ID);
            dxl_set_txpacket_instruction(INST_SYNC_WRITE);
            dxl_set_txpacket_parameter(0, P_GOAL_POSITION_L);
            dxl_set_txpacket_parameter(1, 4);
            for( i=0; i<NoMot; i++ )
            {
                dxl_set_txpacket_parameter(2+5*i, id[i]);
                dxl_set_txpacket_parameter(2+5*i+1, dxl_get_lowbyte(WF[i][j]));
                dxl_set_txpacket_parameter(2+5*i+2, dxl_get_highbyte(WF[i][j]));
                dxl_set_txpacket_parameter(2+5*i+3, dxl_get_lowbyte(velocidad[i]));
                dxl_set_txpacket_parameter(2+5*i+4, dxl_get_highbyte(velocidad[i]));
            }
            dxl_set_txpacket_length((4+1)*NoMot+4);
                        dxl_txrx_packet();
            mDelay(tiempo);

            for(i=0; i<NoMot; i++)
            {
            if(WF[i][j+1]<WF[i][j])
                difPos[i]=WF[i][j]-WF[i][j+1];
            else
                difPos[i]=WF[i][j+1]-WF[i][j];


            for(i=0; i<NoMot; i++)
            {
            if(WF[i][j+1]<WF[i][j])
                difPos[i]=WF[i][j]-WF[i][j+1];
            else
                difPos[i]=WF[i][j+1]-WF[i][j];

                velocidad[i]= 494*difPos[i]/tiempo;
            if(velocidad[i]<13)
                velocidad[i]=13;
            if(velocidad[i]>1023)
                velocidad[i]=1023;
            }
        }
        a++;
        }
}
```

Figura 3.21. Función de reproducción para el movimiento de caminado hacia delante.

Una vez que está diseñada la función mostrada en la Figura 3.21 para cada uno de los movimientos, basta con que sean llamadas en el código principal para su ejecución y permitan ser evaluadas observando el movimiento del humanoide.

3.2. Clasificador basado en RNA

En esta sección se describe el proceso de diseño, entrenamiento e implementación de la RNA utilizada para el cumplimiento de los objetivos planteados.

Tal y como se revisó en el desarrollo de los robots humanoides a lo largo de la historia, la implementación de nuevas tecnologías y métodos para mejorar la captura y procesamiento de datos se lleva día tras día en busca de mejorar el desempeño que este tipo de sistemas presentan durante la ejecución de sus rutinas y/o tareas. Por otro lado, el desarrollo de RNA en los últimos años las ha demostrado como un método clasificador bastante eficiente, rápido y de poca carga computacional, por lo cual, los antiguos métodos han sido buscado ser sustituidos por estas mismas en busca de mejorar las tareas que desempeñan. Considerando esto,se busca en el presente proyecto, la implementación de una RNA basada en *machine learning* para completar la tarea de identificación y evasión de obstáculos en un robot humanoide, con la espera de lograr la eficiencia de sistemas que conllevan una mayor carga computacional a la que una RNA podría generar.

3.2.1. Captura de base de datos

Para llevar a cabo un entrenamiento de una RNA a través de *machine learning*, es necesario realizar la captura de una base de datos. En esta sección se describe el procedimiento seguido para la captura de la base de datos utilizando el sensor de visión HaVimo2.0.

Para la utilización del sensor de visión HaVimo2.0, lo primero que se tiene que realizar es la conexión de este al controlador CM-530 y el acople al humanoide para que la captura de datos que realice el sensor se lleve a cabo mientras el humanoide está ejecutando su rutina.

Como fue explicado previamente en este documento, el sensor de visión HaVimo2.0 cuenta con un puerto de conexión del estilo para el CM-530, y tal y como los servomotores de la serie Dynamixel AX-12A, también es posible realizar la conexión de éste en serie, incluso con los mismos servomotores.

En cuando a la posición que tendrá el sensor, será en la posición en donde iría una supuesta cabeza del humanoide según la similitud que tiene este con el cuerpo humano. La conexión y posición que asume el sensor es mostrado en la Figura 3.22.

Figura 3.22. Conexión y posición del sensor HaVimo2.0 en el humanoide.

Una vez colocado el sensor en la posición adecuada y asignada dentro del sistema, además de ser comprobada su correcta identificación dentro del programa RoboPlus Manager© (previamente explicado su funcionamiento), se procede a llevar a cabo la calibración del sensor. Para esto, se descarga la interfaz HaViMoGUI©, el cual es un software libre para que aquellos que adquieran un sensor de la marca, puedan hacer uso de él, la Figura 3.23 muestra la interfaz principal de HaViMoGUI©.

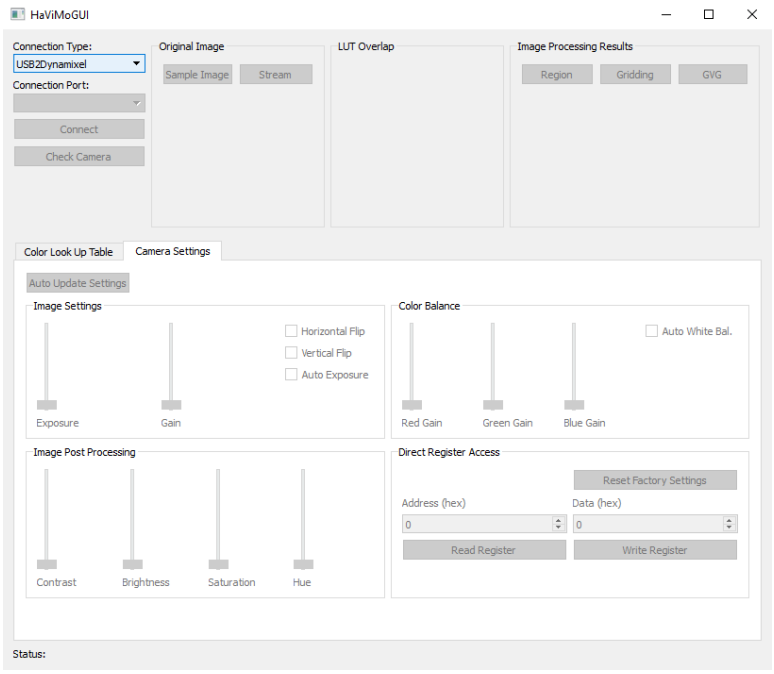

Figura 3.23. Interfaz gráfica principal del software HaViMoGUI©.

Una vez funcionando el software HaViMoGUI©, se siguen los siguientes pasos para la calibración del sensor:

a) Conectar el controlador CM-530 al computador, mismo al que está conectado el sensor.

b) Seleccionar el tipo de controlador y puerto que se desean identificar, posteriormente dar “conectar”.

c) Identificado el sensor, se toma una imagen de muestra algún obstáculo que se estará identificando, en este caso, se capturan los obstáculos presentados en el apartado “puesta experimental”.

d) Capturada la imagen, se selecciona el número de color al que será asignado al identificarlo, y se selecciona en la imagen capturada el color que se está identificando.

e) Una vez almacenados los colores a identificar en la memoria del sensor, en cada captura se llevará a cabo el procesamiento de la misma.

f) La misma interfaz es capaz de mostrar la imagen post procesamiento.

En la Figura 3.24 se muestra la misma interfaz gráfica de HaViMoGUI©, pero esta vez después de realizar la calibración de registro de uno de los colores que fueron asignados a los respectivos obstáculos utilizados en el proyecto.

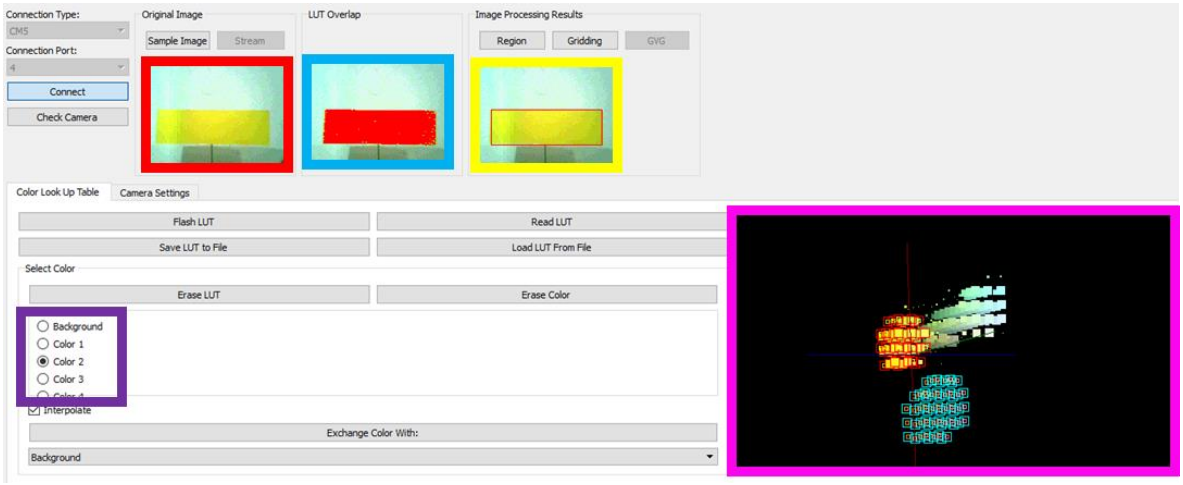

Figura 3.24. Calibración de color amarillo en la interfaz HaViMoGUI©.

De acuerdo con la Figura 3.24, a continuación, se describe cada una de las secciones de la interfaz señaladas en cada uno de los recuadros.

- El área encerrada en un recuadro color rojo representa la imagen capturada por el sensor, misma en la cuál será seleccionado el color que se desea calibrar.

- El área encerrada en un recuadro color morado, muestra el número de registro en el cual será almacenado el color seleccionado en la imagen encerrada en el recuadro color rojo.

- El área encerrada en un recuadro color azul muestra la misma imagen capturada por el sensor, solo que esta vez remarcando los colores que se están seleccionando dentro de la imagen para el reconocimiento.

- El área encerrada en recuadro rosa muestra una imagen en pixeles 3D de la imagen capturada por el sensor, la misma también puede ayudar a la identificación y marcaje de los colores en sus diferentes tonos a identificar.

- El área encerrada en el recuadro amarillo representa la imagen post procesamiento, en donde se reconocen las áreas ocupadas dentro de la imagen capturada por el sensor que contienen alguno de los colores almacenados en la memoria.

Para el sensor HaVimo2.0 existen dos tipos de post procesamiento, uno que muestra la región dentro de un recuadro en la que son reconocidos los colores almacenados, y otro en el que la imagen se muestra como una rejilla de pixeles en la que se destacan los pixeles que contienen los colores a reconocer. Ambos tipos de post procesamientos se muestran en la Figura 3.25.

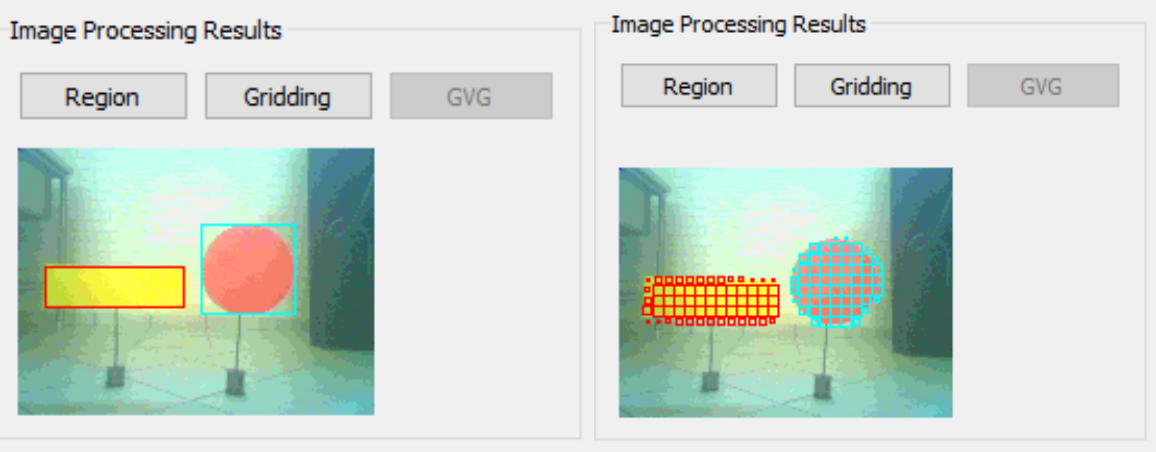

Figura 3.25. Post procesamiento de imagen capturada por el sensor en HaViMoGUI©.

Una vez calibrado el sensor HaVimo2.0 con todos los colores que se estarán utilizando en los diferentes tipos de obstáculos y verificado el post procesamiento en cada uno de los mismos, el sensor de visión está listo para trabajar de manera continua dentro del sistema, capturando imágenes, identificando los colores de los obstáculos y con ayuda del post procesamiento, calculando una relación de tamaño de estos que serviría posteriormente para el entrenamiento de la RNA. En la Figura 3.26, se puede apreciar al robot humanoide frente a los diferentes obstáculos para la captura de datos.

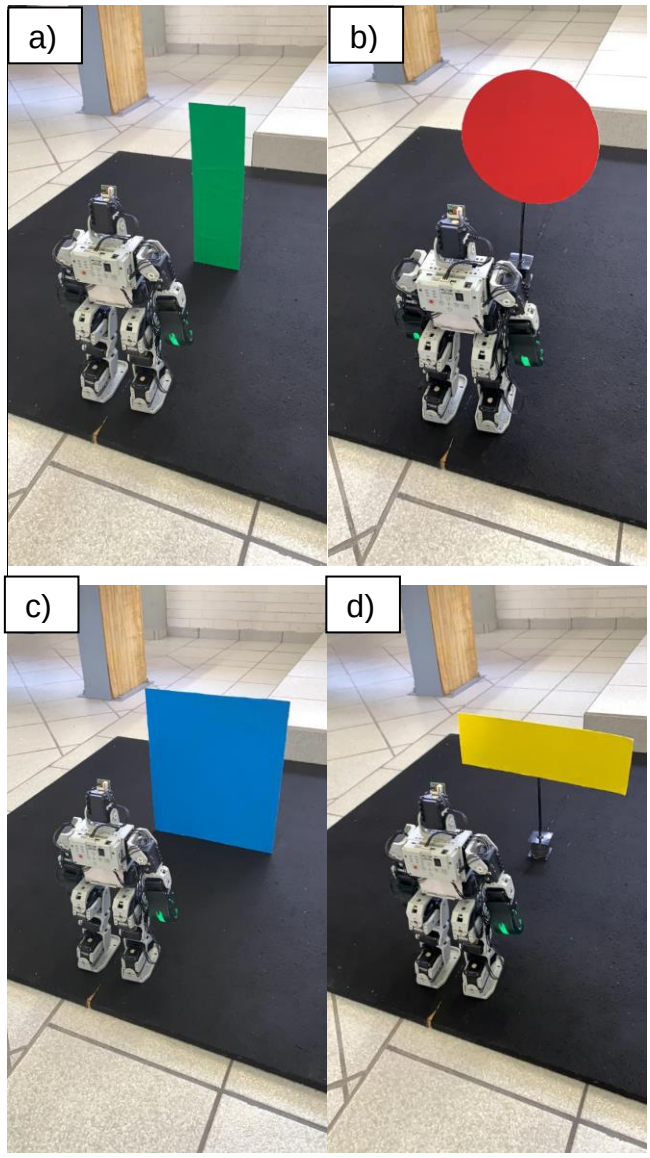

Figura 3.26. Robot durante captura de base de datos frente los diferentes obstáculos: a) Robot frente al obstáculo tipo poste; b) Robot frente al obstáculo tipo persona; c) Robot frente a obstáculo tipo interiores; d) Robot frente a obstáculo tipo barandal.

Además de colocar cada uno de los obstáculos frente al robot, cada uno de los obstáculos fue cambiando su posición en cada captura para variar los valores y abarcar las posibilidades en las que los mismos se le puedan presentar al humanoide durante la deambulación, esto con el objetivo de lograr un entrenamiento de la RNA más robusto. Los valores de cada una de las capturas que fue tomada (30 por cada obstáculo), se muestran a continuación en la Figura 3.27, la cual es una captura de la tabla de la base de datos que posteriormente será utilizada para el entrenamiento de la RNA.

E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
tipo poste							tipo barandal								tipo interiores								tipo persona					
Entrada		Salida					Entrada		Salida						Entrada		Salida						Entrada		Salida			
Color	R	N1	N2	N3	N4		Color	R	N1	N2	N3	N4			Color	R	N1	N2	N3	N4			Color	R	N1	N2	N3	N4
1	3.3	1	0	0	0		2	0.38	0	1	0	0			3	1.32	0	0	1	0			4	1	0	0	0	1
1	3.41	1	0	0	0		2	0.23	0	1	0	0			3	1.41	0	0	1	0			4	1.07	0	0	0	1
1	3.36	1	0	0	0		2	0.38	0	1	0	0			3	1.38	0	0	1	0			4	1.1	0	0	0	1
1	3.35	1	0	0	0		2	0.25	0	1	0	0			3	1.49	0	0	1	0			4	0.99	0	0	0	1
1	3.3	1	0	0	0		2	0.37	0	1	0	0			3	1.41	0	0	1	0			4	1.05	0	0	0	1
1	3.44	1	0	0	0		2	0.36	0	1	0	0			3	1.48	0	0	1	0			4	1.06	0	0	0	1
1	3.36	1	0	0	0		2	0.22	0	1	0	0			3	1.36	0	0	1	0			4	1.05	0	0	0	1
1	3.14	1	0	0	0		2	0.2	0	1	0	0			3	1.34	0	0	1	0			4	1.11	0	0	0	1
1	3.13	1	0	0	0		2	0.38	0	1	0	0			3	1.24	0	0	1	0			4	1.13	0	0	0	1
1	3.43	1	0	0	0		2	0.31	0	1	0	0			3	1.39	0	0	1	0			4	1.03	0	0	0	1
1	3.48	1	0	0	0		2	0.39	0	1	0	0			3	1.3	0	0	1	0			4	1.07	0	0	0	1
1	3.23	1	0	0	0		2	0.17	0	1	0	0			3	1.38	0	0	1	0			4	0.99	0	0	0	1
1	3.47	1	0	0	0		2	0.14	0	1	0	0			3	1.44	0	0	1	0			4	1.08	0	0	0	1
1	3.41	1	0	0	0		2	0.38	0	1	0	0			3	1.4	0	0	1	0			4	1.07	0	0	0	1
1	3.14	1	0	0	0		2	0.4	0	1	0	0			3	1.23	0	0	1	0			4	1	0	0	0	1
1	3.38	1	0	0	0		2	0.23	0	1	0	0			3	1.45	0	0	1	0			4	1.05	0	0	0	1
1	3.18	1	0	0	0		2	0.34	0	1	0	0			3	1.41	0	0	1	0			4	1.09	0	0	0	1
1	3.1	1	0	0	0		2	0.39	0	1	0	0			3	1.21	0	0	1	0			4	1.02	0	0	0	1
1	3.11	1	0	0	0		2	0.33	0	1	0	0			3	1.43	0	0	1	0			4	1.12	0	0	0	1
1	3.26	1	0	0	0		2	0.33	0	1	0	0			3	1.37	0	0	1	0			4	1.12	0	0	0	1
1	3.34	1	0	0	0		2	0.39	0	1	0	0			3	1.29	0	0	1	0			4	1.04	0	0	0	1
1	3.41	1	0	0	0		2	0.15	0	1	0	0			3	1.25	0	0	1	0			4	1.04	0	0	0	1
1	3.34	1	0	0	0		2	0.34	0	1	0	0			3	1.4	0	0	1	0			4	1.04	0	0	0	1
1	3.29	1	0	0	0		2	0.23	0	1	0	0			3	1.29	0	0	1	0			4	1.02	0	0	0	1
1	3.33	1	0	0	0		2	0.28	0	1	0	0			3	1.24	0	0	1	0			4	0.99	0	0	0	1
1	3.16	1	0	0	0		3	0.34	0	1	0	0			3	1.41	0	0	1	0			4	1.09	0	0	0	1
1	3.45	1	0	0	0		4	0.11	0	1	0	0			3	1.34	0	0	1	0			4	1.13	0	0	0	1

Figura 3.27. Captura de la base de datos.

Para una mejor observación de la dispersión que tienen los valores de la relación en la base de datos, se tiene la Figura 3.28, en donde se puede apreciar a través de un gráfico tipo ANOVA, los valores en los que se concentran la relación alto/ancho en cada uno de los obstáculos.

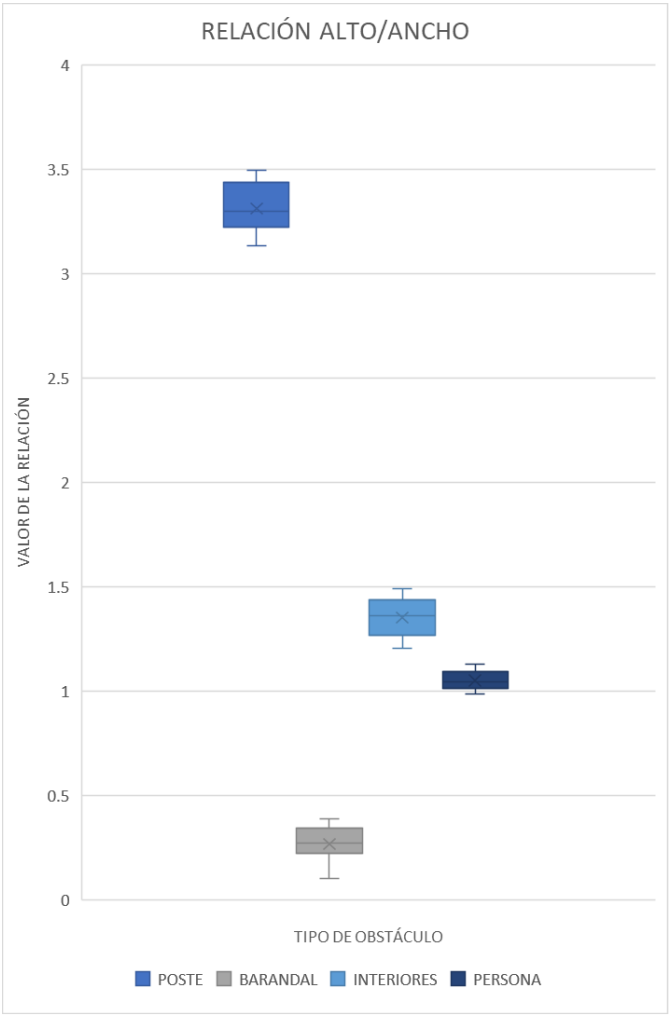

Figura 3.28. Gráfico ANOVA de la dispersión de la relación alto/ancho en base de datos.

En base al gráfico ANOVA mostrado anteriormente, en donde se muestra con un color de caja diferente el rango en donde se encontraron la mayor proporción de valores en la toma de la base de datos y a la vez se puede observar los valores extremos para cada tipo de obstáculo, se puede confirmar que los valores de la relación alto/ancho para cada uno de los obstáculos se encuentran en un rango adecuado no muy disperso y que a la vez no se enciman valores de la relación de un obstáculo con otro.

3.2.2. Normalizar las señales

La normalización de los datos de entrada es el proceso en el cual se reduce el rango en las señales de entrada a los rangos [0,1] o [-1,1]. Si no se realiza la normalización los datos de entrada tendrán un efecto adicional sobre la neurona, dando lugar a decisiones incorrectas o trabajo computacional excesivo. En esta sección se

explica el proceso seguido para la normalización de las señales utilizadas en el presente proyecto para el entrenamiento de la red neuronal.

Una vez obtenidos los valores de la base de datos, mismos que fueron mostrados en la sección anterior, en la Figura 3.27, se procede a normalizarlos, para lo cual se sigue la siguiente ecuación (3).

$$y = \frac{(x - x_{min})(d_2 - d_1)}{x_{max} - x_{min}} + d1 \tag{3}$$

En donde:

$$x\ es\ el\ valor\ a\ normalizar$$
$$[x_{min}\ , x_{max}]\ es\ el\ rango\ de\ los\ valores\ en\ la\ señal\ de\ entrada$$
$$[d_1\ , d_2]\ es\ el\ rango\ al\ que\ serán\ reducidos\ los\ valores$$

A continuación, en la Tabla 3.1, se muestra una captura de la base de datos para uno de los obstáculos en cuestión, después de haber sido normalizada.

Tabla 3.1. Señal normalizada para el entrenamiento de la RNA.

Color	R	N1	N2	N3	N4
0.25	0.94366141	1	0	0	0
0.25	0.92768153	1	0	0	0
0.25	0.92321012	1	0	0	0
0.25	0.93287712	1	0	0	0
0.25	0.94504431	1	0	0	0
0.25	0.93437311	1	0	0	0
0.25	0.88786406	1	0	0	0
0.25	1	1	0	0	0
0.25	0.88936122	1	0	0	0

0.25	0.9595012	1	0	0	0
0.25	0.8921803	1	0	0	0
0.25	0.89358809	1	0	0	0
0.25	0.95436411	1	0	0	0
0.25	0.93068825	1	0	0	0
0.25	0.98085496	1	0	0	0
0.25	0.92552308	1	0	0	0
0.25	0.92525265	1	0	0	0
0.25	0.9608027	1	0	0	0
0.25	0.88917182	1	0	0	0
0.25	0.97012905	1	0	0	0
0.25	0.93744723	1	0	0	0
0.25	0.98876335	1	0	0	0
0.25	0.93735943	1	0	0	0
0.25	0.9351202	1	0	0	0
0.25	0.99609845	1	0	0	0

El cambio que existe en los valores de la base de datos antes y después de la normalización es completamente proporcional, para mostrar lo anteriormente dicho, se tiene la Figura 3.29, en donde se muestra la gráfica de los valores de la relación alto/ancho de tamaño de uno de los objetos utilizados con y sin la normalización.

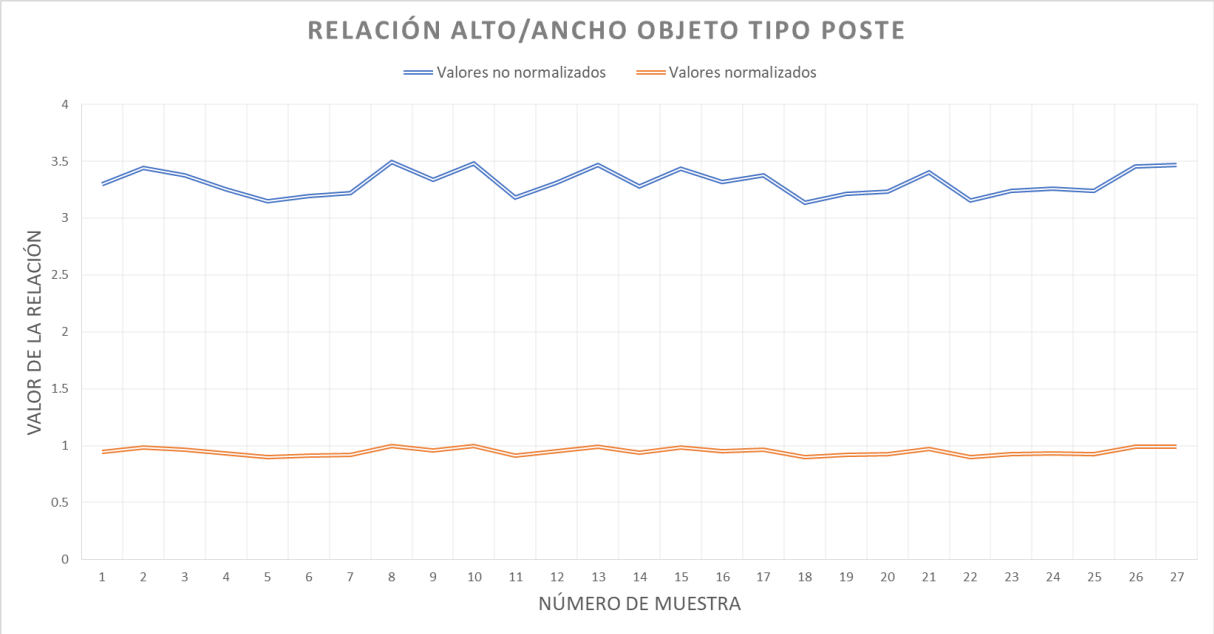

Figura 3.29. Gráfica de valores normalizados y no normalizados para el objeto tipo poste.

En la gráfica mostrada en la Figura 3.29, se puede observar que, pese a que ambas líneas, en donde la línea azul representa los valores no normalizados y la línea naranja los valores normalizados, se encuentran en un rango de valores diferentes, dado que en una ya se encuentran los valores con un límite en el valor 1, ambas líneas tienen la misma forma, dado que como se había comentado anteriormente, el cambiarlas de rango, es un cambio completamente proporcional.

Este procedimiento es repetido para toda la base de datos, y con estos valores encontrados entre el 0 y el 1, es que será entrenada la RNA, a continuación, se muestran el resto de las señales normalizadas en comparación con su señal antes de normalizar, como primera parte se tiene la Figura 3.30 con el objeto tipo barandal, en donde la línea azul y la línea naranja, siguen representando los valores normalizados y no normalizados al igual que en la Figura 3.29.

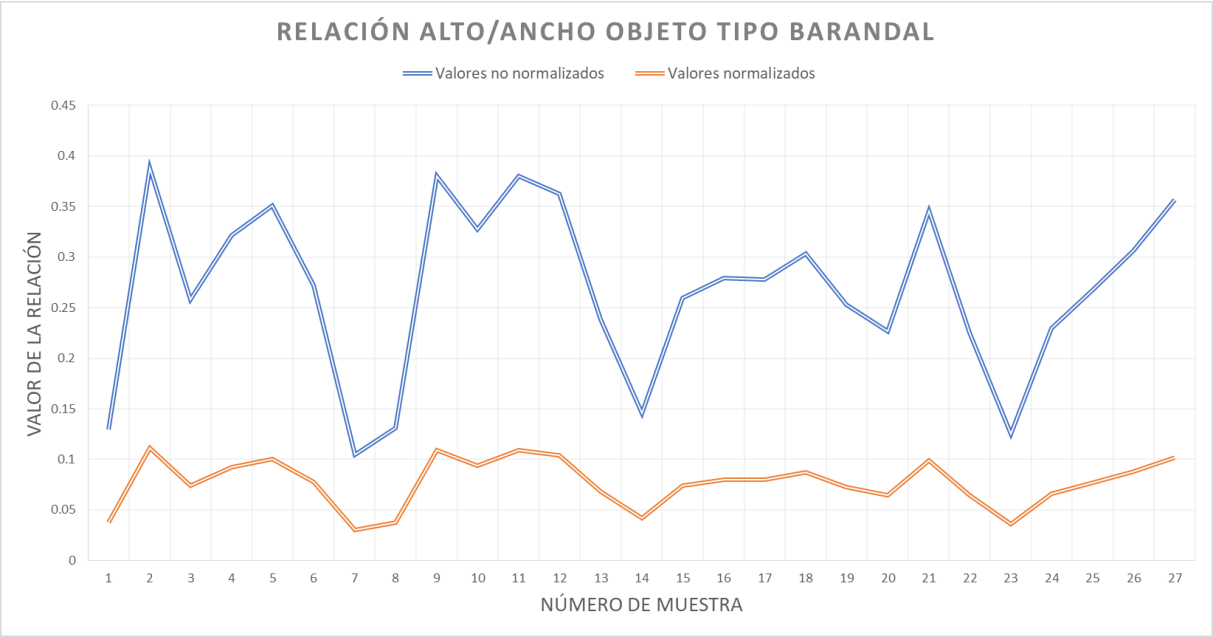

Figura 3.30. Gráfica de valores normalizados y no normalizados para el objeto tipo barandal.

Buscando que la tendencia se siga cumpliendo con los diferentes objetos, se continúa con el objeto tipo persona, dicha gráfica se puede observar en la Figura 3.31, en donde, al igual que con los obstáculos anteriores, la línea azul representa los valores no normalizados y la naranja los valores después de normalizar.

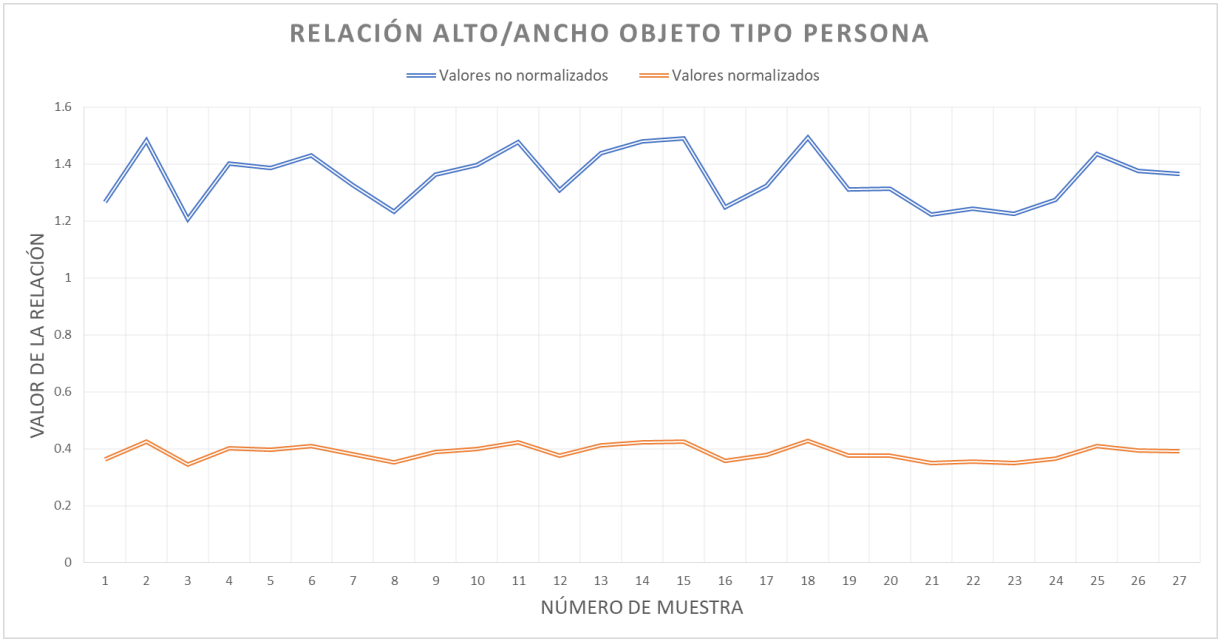

Figura 3.31. Gráfica de valores normalizados y no normalizados para el objeto tipo persona.

Finalmente, se tiene el objeto tipo interiores en la Figura 3.32, siguiendo la tendencia, se puede observar solamente un cambio proporcional entre la línea azul (valores no normalizados), y la línea naranja (valores normalizados).

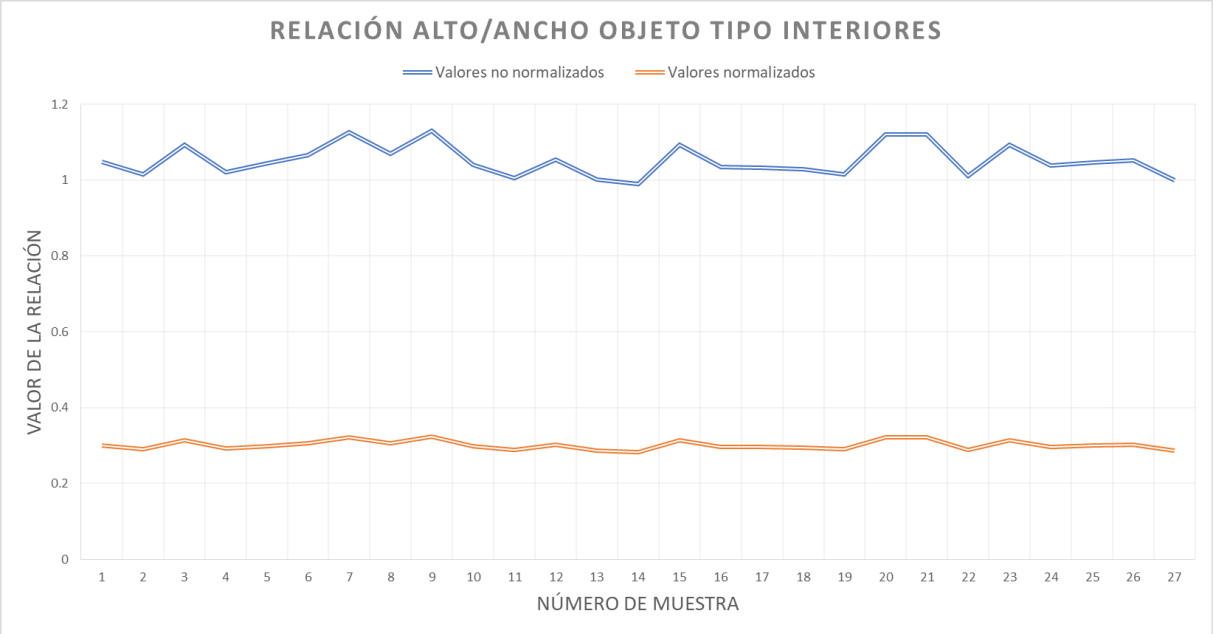

Figura 3.32. Gráfica de valores normalizados y no normalizados para el objeto tipo interiores.

De igual manera, los valores que se tienen para la identificación de los colores de los objetos y que son los numero enteros encontrados en el conjunto [1, 4], son normalizados, debido a que también serán usados para el entrenamiento de la RNA y pueden ocasionar el mismo problema en caso de que no se lleve a cabo el proceso. A continuación, en la Figura 3.33, se observa la normalización de las señales del color de todos los obstáculos después de seguir el mismo proceso que con la relación alto/ancho y que nos permite observar solamente un cambio en el rango de los valores completamente proporcional, pues se mantienen las mismas formas de línea. En este caso, la línea de color azul representa los datos originales capturados, mientras que, la línea naranja representa los datos capturados pero después de realizar la normalización.

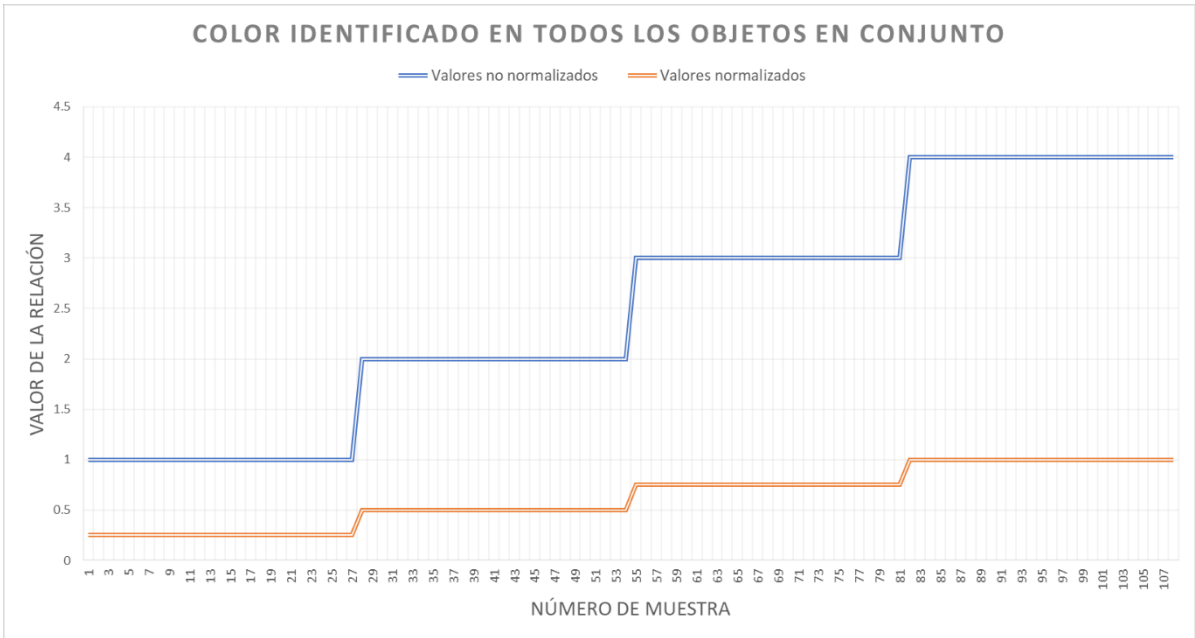

Figura 3.33. Gráfica de valores normalizados y no normalizados de las señales de identificación de color de todos los objetos en conjunto.

Para terminar de ver la comparativa de la base de datos antes y después de normalizarla, se tiene en la Figura 3.34 una gráfica estilo ANOVA que muestra la dispersión que se tiene después de la normalización en cada uno de los valores en la relación alto/ancho para los obstáculos. Nuevamente el gráfico muestra con un color de caja diferente el rango en donde se encontraron la mayor proporción de valores en la toma de la base de datos y a la vez se puede observar los valores extremos para cada tipo de obstáculo. Se puede apreciar que la única diferencia en comparación con la Figura 3.28, en donde se muestra el gráfico de los valores antes de normalizar, es el rango en el que se encuentran los valores en cuanto al valor de la relación.

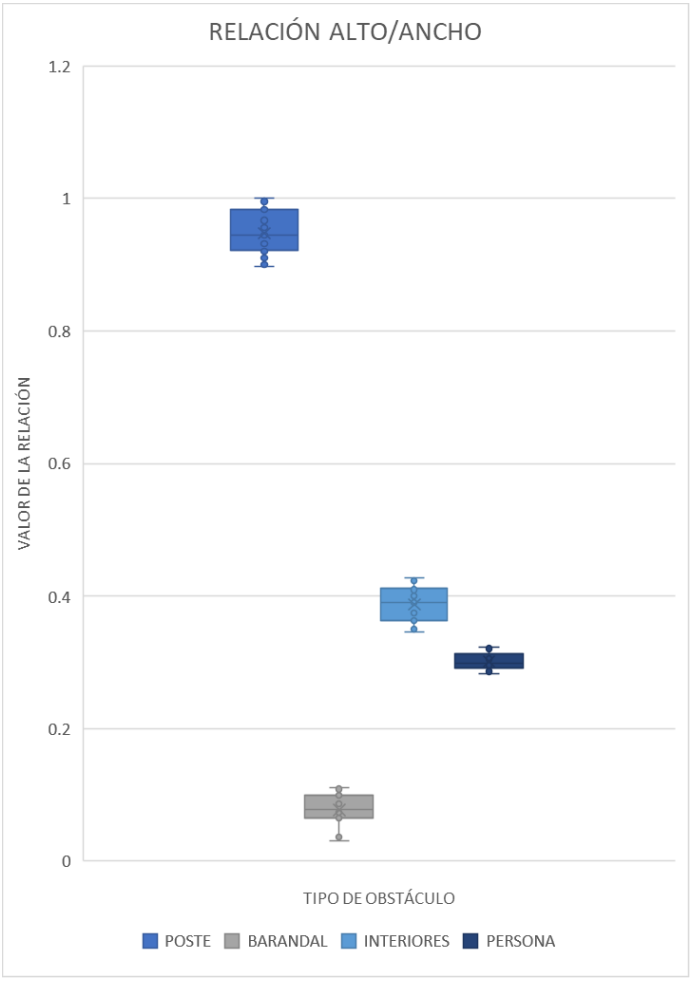

Figura 3.34. Gráfico ANOVA de la dispersión de la relación alto/ancho en base de datos después de la normalización.

Una vez que se tienen normalizadas todos los valores en las señales que serán utilizadas, se puede proseguir con el entrenamiento de la RNA.

3.2.3. Entrenamiento de la RNA

En esta sección, se describe el método seguido para el entrenamiento de la RNA.

Una vez que se tiene la base de datos con las señales para el entrenamiento normalizadas se procede a realizar el entrenamiento de la RNA. El algoritmo usado para llevar a cabo el entrenamiento es el Algoritmo de retro-propagación, presentado por (Rojas, 1996), también se hace uso de la herramienta de software matemático MATLAB©, parte del código usado en este trabajo se muestra a continuación.

```
while(1)
      %Evaluacion
      h1       = sigmoide((In*Wi')'+Bi);
      h2       = sigmoide((h1'*Wh)'+Bh);
      Outp     = sigmoide(h2'*Wo+Bo);
      %error
      d=Outp - Out;
      %Calculo del error función de coste
      C=1/2*d.^2;
      %Actualizar pesos
      Wo = Wo-nu*d*dsig(h2);
      Wh = Wh-nu*d*dsig(h1*Wo');
      Wi = Wi-nu*d*dsig(i*Wh*Wo);
      %Actualizar bias
      Bi = Bi-nu*d*dsig(sum(Wh*Wo));
      Bh = Bh-nu*d*dsig(sum(Wo));
      Bo = Bo-nu*d;
      err(i)=d;
      cos(i)=C;
          if (C<goal)||(i>10000)
              break;
          end
      i=i+1;
      tt(i) = i;
end
```

En la sección de código mostrado anteriormente, se observa la implementación del algoritmo de retro-propagación, se engloba todo el entrenamiento en un ciclo *while* en el cual se actualizan en cada iteración los pesos y los bias de la RNA, la función de activación utilizada en este algoritmo se describe a continuación.

$$\sigma(t) = \begin{cases} -1 & , \quad t < -1 \\ t & , \quad -1 \leq t \leq 1 \\ 1 & , \quad t > 1 \end{cases} \tag{5}$$

Esta función de activación le da un valor de -1 a la salida cuando la entrada es menor a -1, y le da un valor igual al de la entrada cuando está entre el intervalo de -1 a 1, cuando es mayor que 1 le da un valor de 1 a la salida como se muestra en la Figura 3.35.

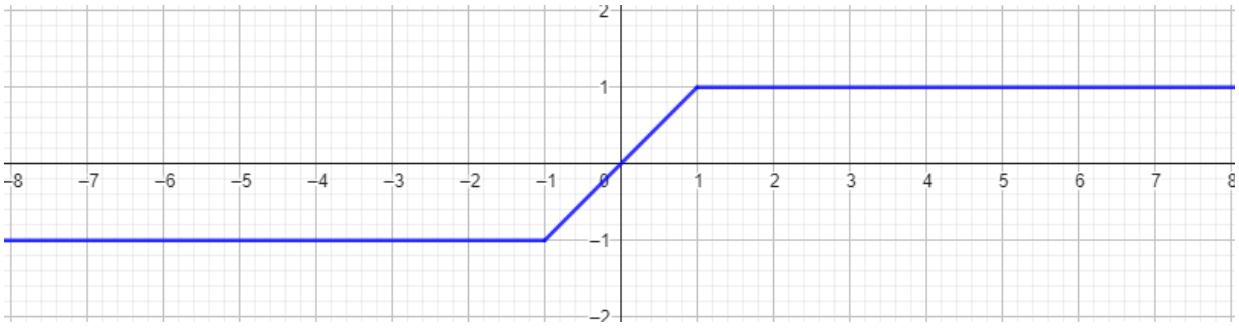

Figura 3.35. Función de activación usada para el entrenamiento.

Parte del entrenamiento de la RNA usando el algoritmo de retro-propagación utiliza la derivada de la función de activación por lo que también es necesario calcularla, la gráfica de la Figura 3.36 muestra esta derivada.

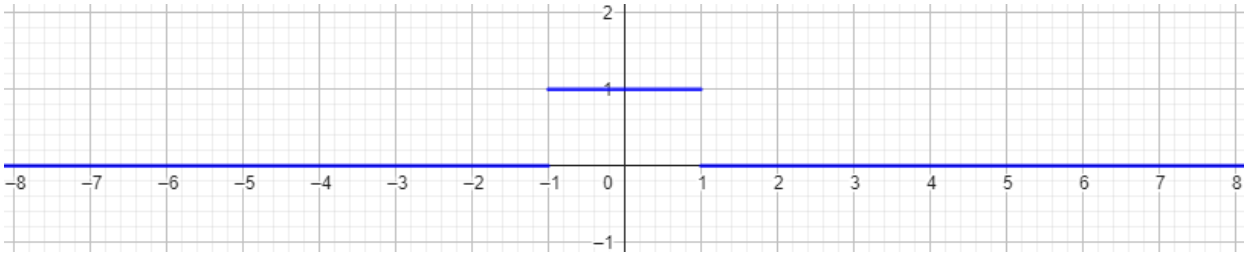

Figura 3.36. Derivada de la función de activación.

En este algoritmo, el orden en que se ejecuta la actualización de los pesos y los bias es contrario al orden en que se ejecuta la RNA, en este caso va de derecha a izquierda, como se muestra en la Figura 3.37.

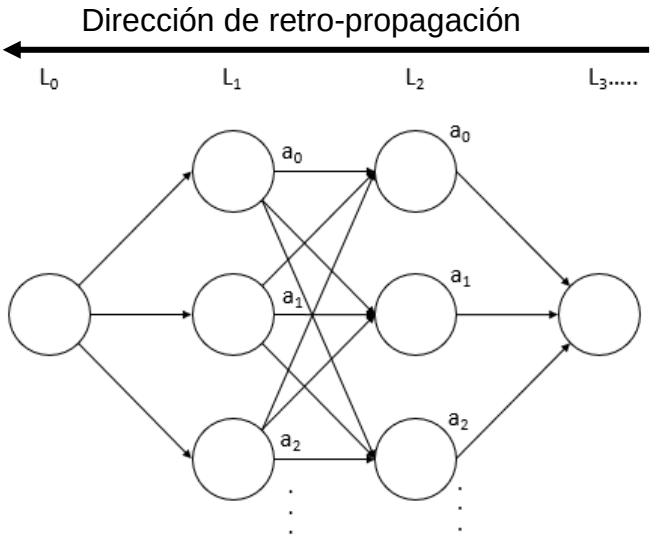

Figura 3.37. Diagrama de retro-propagación en una RNA.

Una vez implementado el algoritmo, se obtienen las gráficas de error y de coste de entrenamiento mostradas en la Figura 3.38.

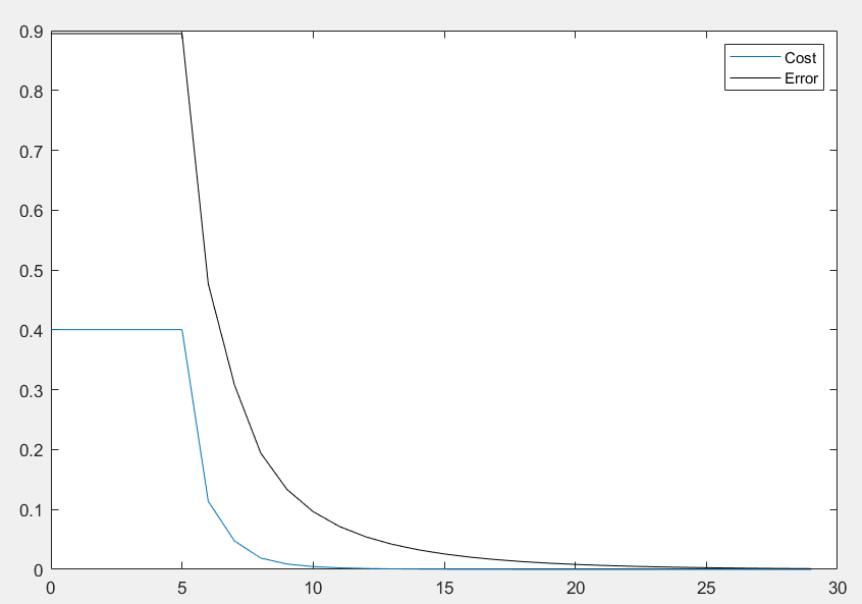

Figura 3.38. Gráfica de error y de coste en el entrenamiento.

En la Figura 3.38 se observa que la gráfica de coste llega a un valor muy cercano a cero desde la iteración 10, mientras que la gráfica de error llega casi a cero desde la iteración 20, eso representa un entrenamiento efectivo y rápido en la RNA.

3.2.4. Extraer pesos y bias

Para el desarrollo del presente proyecto, y específicamente para el entrenamiento de la RNA, se sigue por un entrenamiento fuera de línea, y es necesario realizar una extracción de pesos y bias desde el software donde se entrenó la RNA, en este caso MATLAB©, para posteriormente llevar a cabo una implementación.

El entrenamiento fuera de línea de una RNA consiste en entrenar la red en una etapa separada de la fase de ejecución. Durante el proceso de entrenamiento, la red neuronal ajusta sus parámetros para minimizar la función de pérdida y mejorar su capacidad de clasificación precisa. Luego de este proceso, se evalúa la red neuronal en un conjunto de datos de validación distintos a los que se usaron en el entrenamiento y se prueba para determinar su capacidad de clasificación.

Como se mencionó en la sección anterior, el diseño de la RNA usado en el presente proyecto es de 2 capas ocultas y de 5 neuronas cada una, también tiene dos entradas y cuatro salidas. por lo que los pesos y bias extraídos se muestran como sigue en la Tabla 3.2.

Tabla 3.2. Extracción de pesos y bias.

Pesos Capa de entrada	**Pesos Capa oculta**	**Pesos Capa de salida**	**Bias**
Wi =	Wh =		
0.7697	0.2024 -0.1090		
0.6443	0.3590 0.5232	Wo =	
0.5709	-0.0087 -0.0726	0.0556	Bi = 0.1472
0.2987	-0.1626 -0.1513	-0.0062	Bh = 0.3450
	0.7333 -0.0482	0.3503	Bo = -0.0807
	0.7165 0.6458	0.4256	
	-0.1326 0.4063		
	0.5140 0.7197		

Cuando se utiliza el modelo en un entorno fuera de línea, como en una aplicación o sistema en producción, no se realiza un entrenamiento adicional. En su lugar, se carga el modelo previamente entrenado y se utiliza para generar respuestas en función de las

entradas recibidas. Esto permite que el sistema funcione de manera autónoma, sin necesidad de conectarse a un servidor externo o realizar el entrenamiento en tiempo real.

3.2.5. Implementación en C embebido

Para implementar la RNA previamente calculada, se deben ingresar los valores de pesos, bias y la matriz de valores de prueba en el código de C embebido, luego se procede a la ejecución de la RNA usando estos valores. Una parte del código implementado se muestra a continuación.

```
for (int i = 0; i < L; i++)
        for (int j = 0; j < n; j++){
                A[j][i] = 0;
                for (int k = 0; k < n; k++){
                        A[j][i] = A[j][i] + A[j][i-1]W[j][i-1];
                        A[j][i] = A[i][i] + B[i];
                        A[j][i] = AF(A[j][i]);
                }
        }
```

En la sección de código mostrada anteriormente se observa que todo se ejecuta en dos ciclos *for* anidados, esto para recorrer la matriz de valores de entrada y cada valor de salida de cada neurona se multiplica por el correspondiente peso de la capa anterior, al final cuando se obtiene la suma, esta entra a la función de activación “AF” y se calcula finalmente el valor de salida de la nueva neurona, cada valor se almacena de forma organizada en la matriz “A”.

3.3. Pruebas del sistema

En esta sección se describe en qué consiste cada una de las pruebas en las que será evaluado el robot humanoide.

Para la realización de las pruebas con cada uno de los obstáculos descritos con anterioridad, se usa una pista cuadrangular con medidas1.2×1.2 m. El robot partiría de

tres diferentes posiciones para evaluar el efectivo reconocimiento del obstáculo en cuestión, dichas posiciones se etiquetan como; posición A, posición B y posición C, y se pueden observar en la Figura 3.39. El cambio de posición del humanoide durante las pruebas con cada uno de los obstáculos se debe a la intención de abarcar diferentes escenarios posibles en los que en un caso real se le puedan presentar estos obstáculos al robot durante su deambulación y no considerar únicamente una trayectoria recta y directa hacia los mismos.

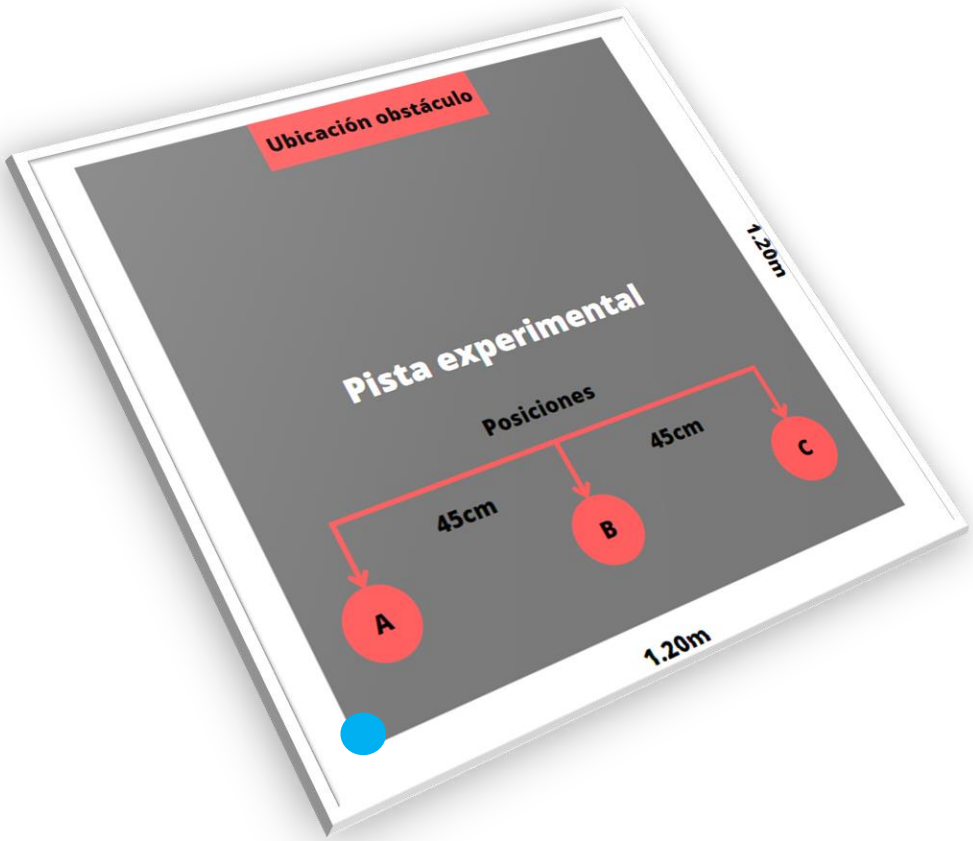

Figura 3.39. Ilustración de la pista experimental y la ubicación de las diferentes posiciones iniciales del robot humanoide y el obstáculo.

Haciendo un análisis de la pista experimental mostrada en la Figura 3.39, se puede observar la ubicación del obstáculo, así como la disposición grafica de las posiciones descritas A, B y C. Sin embargo, para tener una mejor perspectiva de cada una de las mismas, a continuación, se muestra la Tabla 3.3, la cual contiene las coordenadas de las

posiciones con medidas en centímetros y tomando como origen el punto azul de la Figura 3.39.

Tabla 3.3. Coordenadas de las posiciones iniciales del robot humanoide durante las pruebas.

Posición	Coordenadas (cm)
A	(15, 20)
B	(60, 20)
C	(105, 20)

Todas las posiciones iniciales del humanoide durante las pruebas comenzarán a una distancia de 100 cm o 1 m del obstáculo, a esta distancia el robot comienza el caminado con dirección al obstáculo, en el cual, en todo momento estará analizando las imágenes capturadas por el sensor de visión HaVimo2.0. En el momento en el que, en base al procesamiento del controlador, se determine la detección de un obstáculo, el robot detendrá por completo su movimiento y ejecutara una serie de movimientos para la evasión del obstáculo, en la Figura 3.40 se muestran los movimientos a realizar para la evasión de acuerdo con cada tipo de obstáculo.

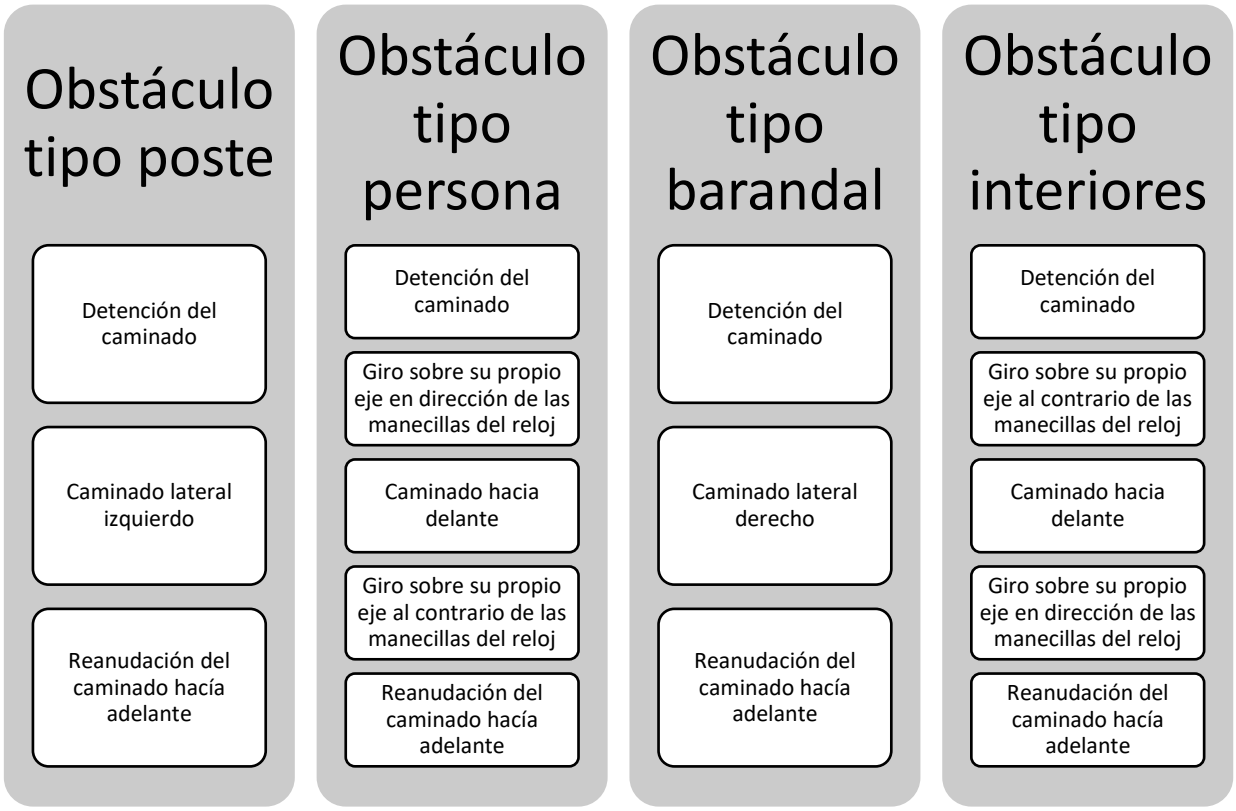

Figura 3.40. Movimientos para la evasión de cada tipo de obstáculos.

La distancia a la que se detecta el obstáculo será capturada para el análisis de los resultados. A continuación, en la Figura 3.41 se muestra un diagrama de flujo que describe el procedimiento seguido en cada una de las pruebas.

Figura 3.41. Ilustración de la pista experimental y la ubicación de las diferentes posiciones iniciales del robot humanoide y el obstáculo.

Para cada uno de los cuatro obstáculos, se realizarán 15 pruebas, de las cuales serán cinco arrancando en cada una de las posiciones iniciales, es decir, 5 serán desde la posición A, 5 desde la posición B y 5 desde la posición C, siguiendo el procedimiento mostrado en el diagrama de la Figura 3.41.

4. Resultados

En esta sección se presentan los resultados obtenidos en las pruebas realizadas con cada uno de los obstáculos descritos anteriormente.

4.1.1. Resultados para obstáculo tipo persona

Para este tipo de obstáculo se llevaron a cabo 15 pruebas en distintas posiciones, siguiendo el procedimiento descrito en la sección 3.3 del presente documento, los resultados obtenidos para el obstáculo tipo persona se muestran en la Tabla 4.1, en donde se puede observar el número de prueba, la posición inicial, un valor indicativo acerca de si el obstáculo fue detectado o no (siendo 1 el valor en caso de ser detectado y 0 en caso de no serlo), y la distancia a la que el obstáculo fue detectado en caso correspondiente.

Tabla 4.1. Resultados de las pruebas realizadas para el obstáculo tipo persona.

No. Prueba	Posición Inicial	Detectado (0-1)	Distancia (cm)
1	A	1	61
2	A	1	69
3	A	1	61
4	A	1	71
5	A	1	64
6	B	1	78
7	B	1	79
8	B	1	73
9	B	1	78
10	B	1	75
11	C	1	69
12	C	1	69

11	C	1	61
14	C	1	64
15	C	1	79

La Tabla 4.1 muestra que para el caso del obstáculo tipo persona el robot tuvo una efectividad del 100% haciendo el reconocimiento de este sin importar la posición en la que fue colocado y solamente variando la distancia a la que lo detectaba. Para una mejor percepción de los valores de distancia a la que el robot identificó el obstáculo de forma efectiva se muestra en la Figura 4.1 una gráfica de los valores mencionados en cada una de las pruebas.

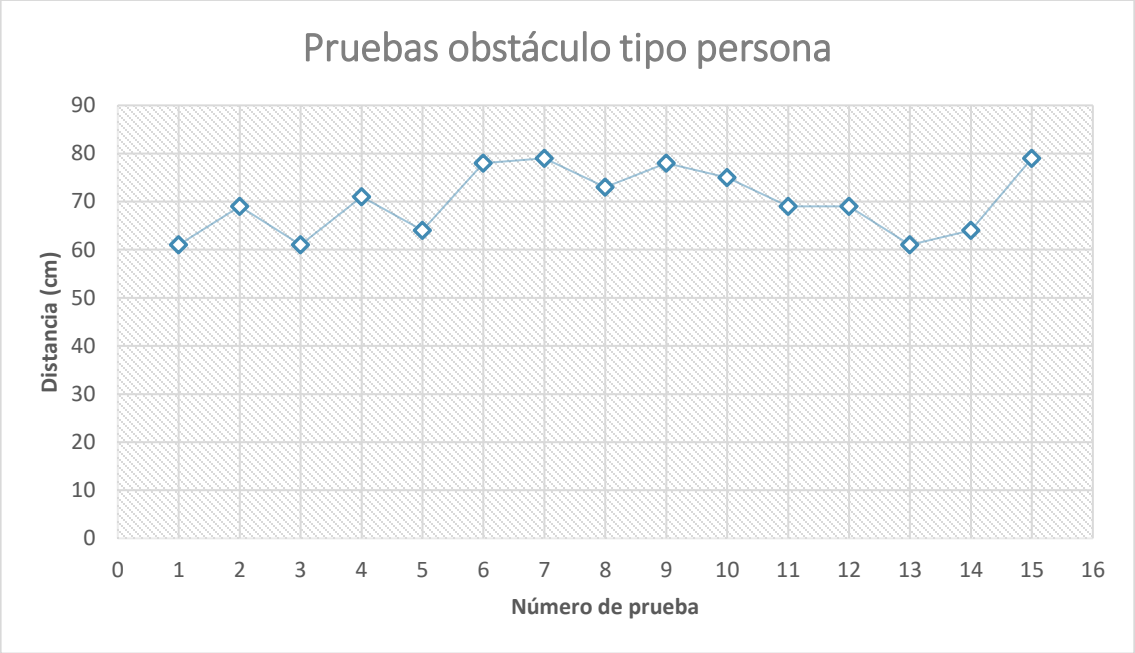

Figura 4.1. Gráfico de distancia hacia el obstáculo tipo persona cuando es detectado por el robot.

Al observar los valores a los que fue detectado el obstáculo en cada una de las pruebas, se encontró que ninguno se presenta tan alejado de acuerdo a la tendencia que presentan, pues se obtiene un valor promedio de detección de obstáculo de 70.07 cm y el valor más lejano encontrado en las 15 pruebas presenta una variación de solamente 9 cm, sumado a que el objeto fue detectado en cada una de las pruebas, se puede

mencionar que el sistema funciona de manera efectiva al presentársele un objeto tipo persona durante su deambulación.

4.1.2. Resultados para obstáculo tipo poste

Siguiendo el procedimiento descrito en la sección 3.3 del presente documento y ya utilizado con el obstáculo tipo persona, se muestran en la Tabla 4.2 los resultados de las 15 pruebas realizadas con el obstáculo tipo poste.

Tabla 4.2. Resultados de las pruebas realizadas para el obstáculo tipo poste.

No. Prueba	Posición Inicial	Detectado (0-1)	Distancia (cm)
1	A	0	---
2	A	0	---
3	A	1	71
4	A	1	63
5	A	1	76
6	B	1	66
7	B	1	77
8	B	1	61
9	B	1	73
10	B	1	70
11	C	1	70
12	C	1	72
11	C	1	65
14	C	1	71
15	C	1	78

La Tabla 4.2 muestra que para el caso del obstáculo tipo poste el robot tuvo una efectividad del 86.66% haciendo el reconocimiento de este, el único par de errores que se presentaron en este obstáculo se presenciaron durante las pruebas en la posición A, esto mismo podría deberse a la iluminación encontrada en el lugar de las pruebas y el reflejo que pudo afectar directamente a la percepción del sensor HaVimo2.0. Para una mejor percepción de los valores de distancia a la que el robot identifico el obstáculo se muestra en la Figura 4.2 una gráfica de los valores mencionados en cada una de las pruebas, las líneas con la etiqueta No Detectado (ND) representa las pruebas en donde el obstáculo no fue identificado, por lo cual, no se capturó una distancia.

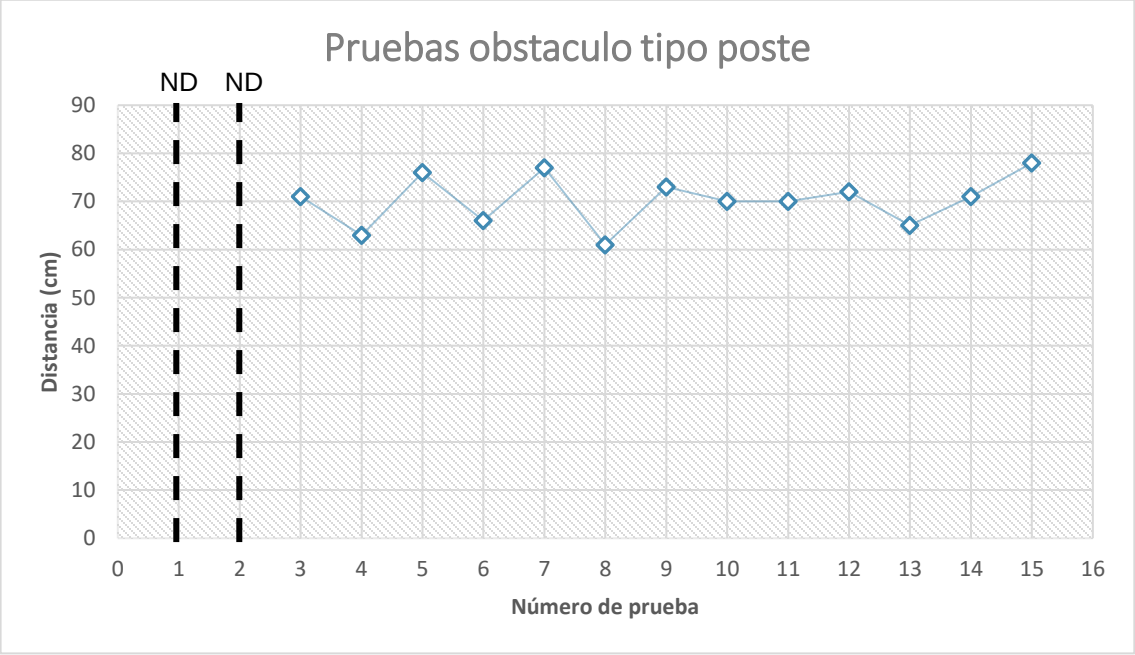

Figura 4.2. Gráfico de distancia hacia el obstáculo tipo poste cuando es detectado por el robot.

Con los valores de la distancia obtenidos en las pruebas para el obstáculo tipo poste, nos encontramos con que cada uno de ellos se presenta en un rango cercano al valor promedio de los mismos (70.23), agregado a que el objeto fue detectado en un buen número de pruebas de manera efectiva, se puede mencionar que el sistema funciona correctamente al presentársele un objeto tipo poste durante su deambulación.

4.1.3. Resultados para obstáculo tipo barandal

Continuando con el procedimiento efectuado en las pruebas con los obstáculos tipo persona y tipo poste, esta vez se realiza para el obstáculo tipo barandal, del cual se muestran los resultados de las pruebas realizadas para cada una de las diferentes posiciones de inicio en la Tabla 4.3.

Tabla 4.3. Resultados de las pruebas realizadas para el obstáculo tipo barandal.

No. Prueba	Posición Inicial	Detectado (0-1)	Distancia (cm)
1	A	1	68
2	A	1	78
3	A	1	61
4	A	1	65
5	A	1	72
6	B	1	79
7	B	1	76
8	B	1	64
9	B	1	64
10	B	1	76
11	C	1	67
12	C	1	71
11	C	1	63
14	C	1	79
15	C	1	60

La Tabla 4.3 muestra que, para el caso del obstáculo tipo barandal, el robot tuvo un comportamiento similar al presentado con el obstáculo tipo persona, debido a que se alcanzó una efectividad del 100% y solamente presentando variaciones en la distancia a la que el obstáculo fue detectado en cada una de las pruebas. Para una mejor percepción de los valores de distancia a la que el robot identifico el obstáculo de forma efectiva se muestra en la Figura 4.3 una gráfica de los valores mencionados en cada una de las pruebas.

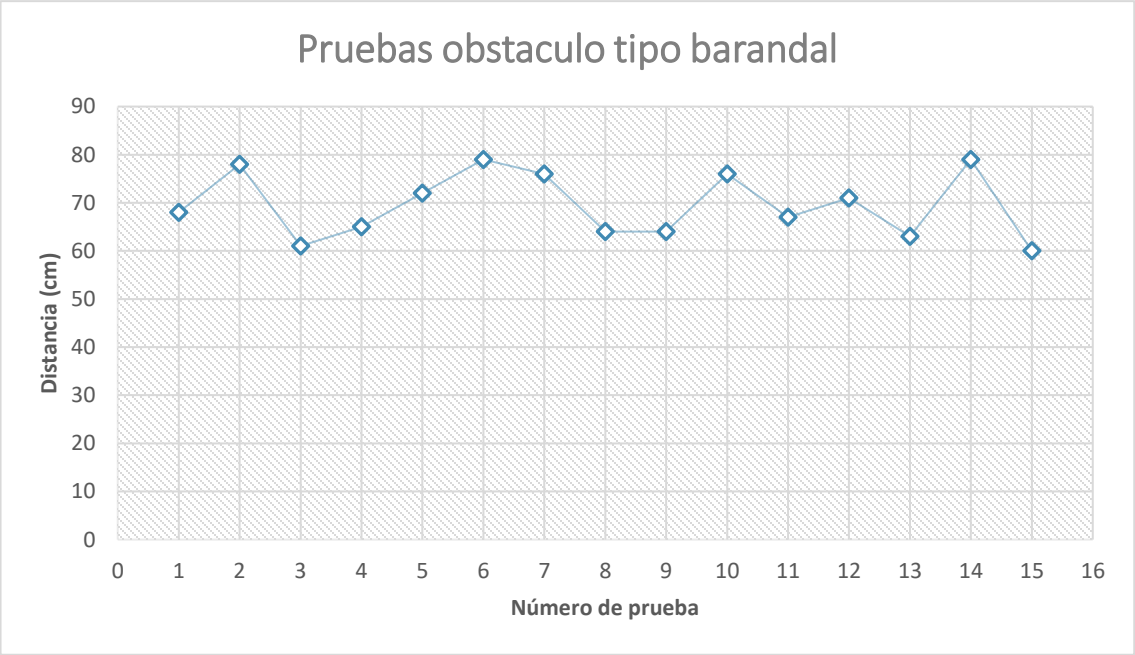

Figura 4.3. Gráfico de distancia hacia el obstáculo tipo barandal cuando es detectado por el robot.

En el caso del obstáculo tipo barandal, el promedio obtenido al realizar todas las pruebas con inicio desde las diferentes posiciones es de 69.53 cm, con esto, añadido a que el objeto fue detectado en cada una de las 15 pruebas, se puede mencionar que el sistema funciona de manera efectiva al presentársele un objeto tipo barandal durante su deambulación.

4.1.4. Resultados para obstáculo tipo interiores

Para la realización de las pruebas con el último de los obstáculos, se mantuvo el procedimiento descrito en la sección 3.3 y seguido en las pruebas de los tres obstáculos

cuyos resultados fueron presentados con anterioridad, los resultados obtenidos se muestran en la Tabla 4.4.

Tabla 4.4. Resultados de las pruebas realizadas para el obstáculo tipo interiores.

No. Prueba	Posición Inicial	Detectado (0-1)	Distancia (cm)
1	A	0	---
2	A	0	---
3	A	0	---
4	A	1	70
5	A	1	77
6	B	1	74
7	B	1	63
8	B	1	68
9	B	1	75
10	B	1	70
11	C	1	75
12	C	1	65
11	C	1	71
14	C	1	72
15	C	1	65

La Tabla 4.4 muestra que para el caso del obstáculo tipo interiores el robot no obtuvo una efectividad completa, sino que fue del 80% haciendo el reconocimiento del mismo, los únicos errores que se presentaron en este obstáculo se presenciaron durante las pruebas en la posición A, siendo un caso parecido al presentado en los resultados

del obstáculo tipo poste, esto mismo refuerza la teoría mencionada sobre que los errores podrían deberse a la iluminación encontrada en el lugar de las pruebas y el reflejo que pudo afectar directamente a la percepción del sensor HaVimo2.0 cuando se iniciaba de la posición A. Para una mejor percepción de los valores de distancia a la que el robot identifico el obstáculo se muestra en la Figura 4.4 una gráfica de los valores mencionados en cada una de las pruebas, las líneas con la etiqueta ND representa las pruebas en donde el obstáculo no fue identificado, por lo cual, no se capturó una distancia.

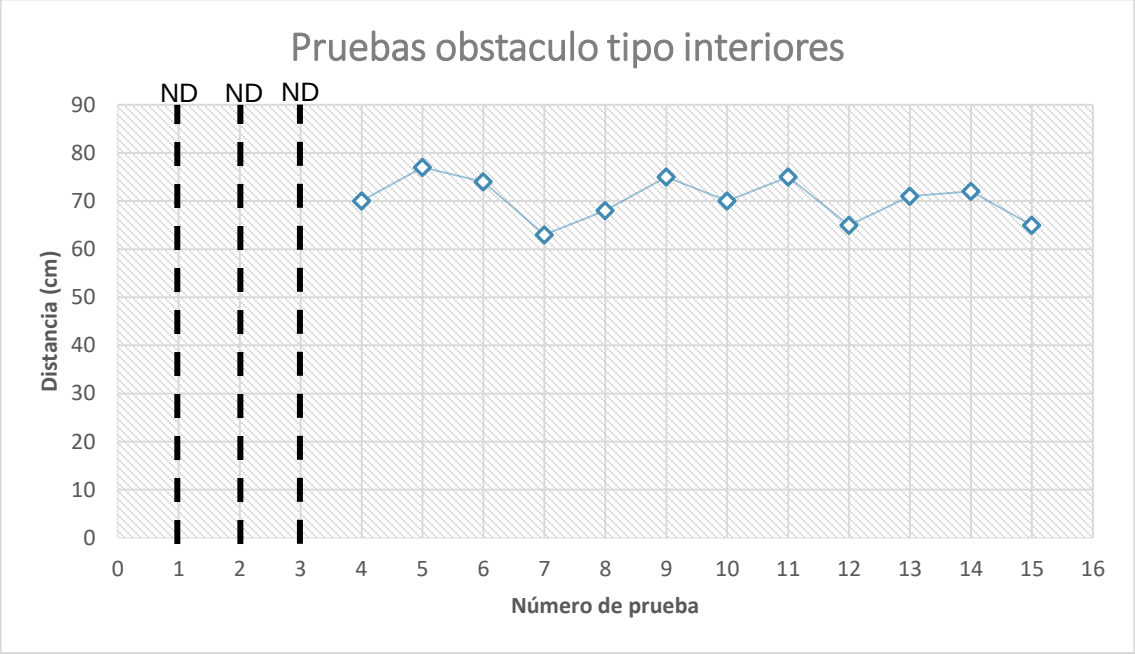

Figura 4.4. Gráfico de distancia hacia el obstáculo tipo barandal cuando es detectado por el robot.

Haciendo un análisis de los valores de la distancia a la que el robot realizo el reconocimiento del obstáculo tipo interiores, se encuentra un valor promedio de 70.42 cm y que ningún valor se encuentra en un rango mayor a ±10 cm, considerando el número de pruebas efectivas se puede mencionar que el sistema funciona correctamente al presentársele un objeto tipo interiores durante su deambulación.

4.1.5. Análisis general de los resultados

Una vez obtenidos los resultados de cada una de las pruebas, y posterior a realizar un análisis de los mismos, se puede decir que el sistema propuesto funciona de correcta manera para la clasificación de obstáculos, debido a que, resumiendo los resultados, se

realizaron 60 pruebas, de las cuales en 55 casos se presentó una correcta clasificación con el obstáculo que se estaba trabajando, los resultados de la distancia a la que se detectaron los cuatro obstáculos en conjunto se pueden apreciar gráficamente en la Figura 4.5.

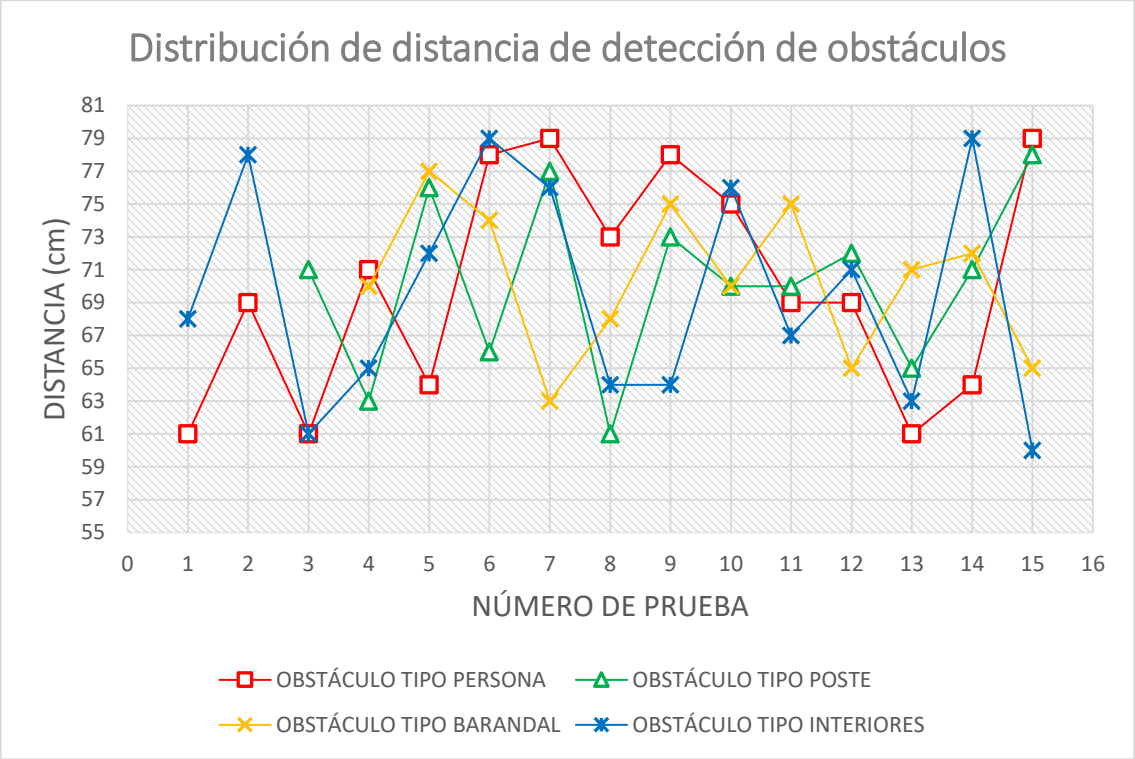

Figura 4.5. Gráfico de distancia hacia los diferentes obstáculos cuando son detectados por el robot.

Para una mejor percepción de la dispersión encontrada en la distancia a la que fueron detectados los obstáculos en cada uno de sus tipos, se tiene la Figura 4.6, un gráfico ANOVA en donde se muestra con un color de caja diferente el rango en donde se encontraron la mayor proporción de valores en las pruebas realizadas y a la vez se puede observar los valores extremos para cada tipo de obstáculo

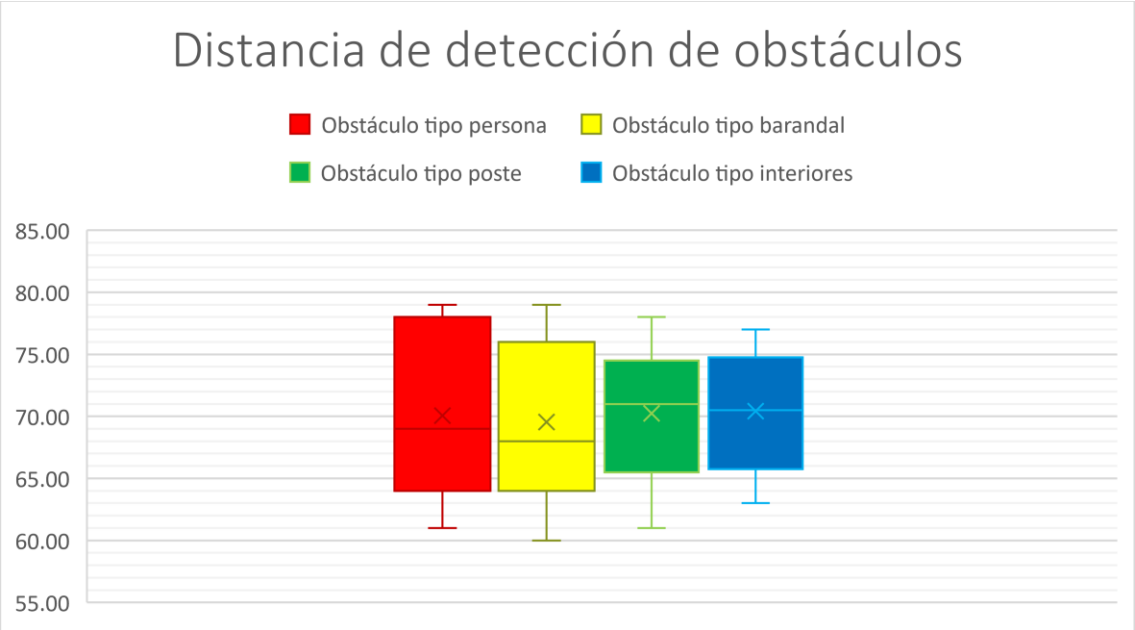

Figura 4.6. Gráfico ANOVA de la distancia a la que fueron detectados los diferentes tipos de obstáculos.

Obteniendo un promedio de los valores de distancia a la que los obstáculos fueron encontrados hablando de las 60 pruebas realizadas con los diferentes tipos de obstáculos y posiciones, se obtiene un valor de 70.04 cm, este valor se muestra en permitiendo una comparación con el promedio obtenido con cada uno de los obstáculos en la Figura 4.7.

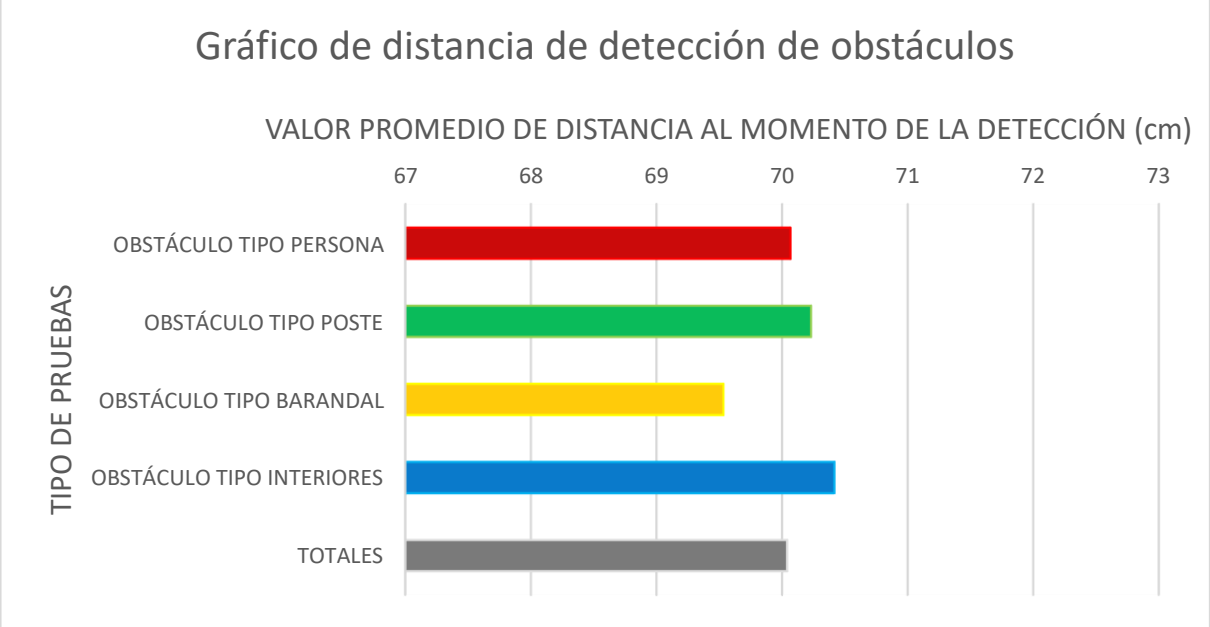

Figura 4.7. Gráfico de promedio de distancia hacia los diferentes obstáculos cuando son detectados por el robot.

En la Figura 4.7, se puede apreciar que, aunque sea pequeña, existe una diferencia en la distancia a la que los diferentes tipos de obstáculos fueron encontrados, por ejemplo, el obstáculo tipo barandal mostrado por la barra amarilla es el que tiene el promedio más bajo de distancia, pues durante las pruebas se encontró a distancias más cercanas a comparación con los otros obstáculos, por otro lado, el obstáculo tipo interiores fue el obstáculo que en promedio, se encontró a mayores distancias.

Si se realiza un análisis de la efectividad del sistema considerando todos los obstáculos en conjunto, se encuentra un valor 91.6%. Un gráfico que permite la percepción de la efectividad con cada uno de los obstáculos y la efectividad total se puede apreciar en la Figura 4.8.

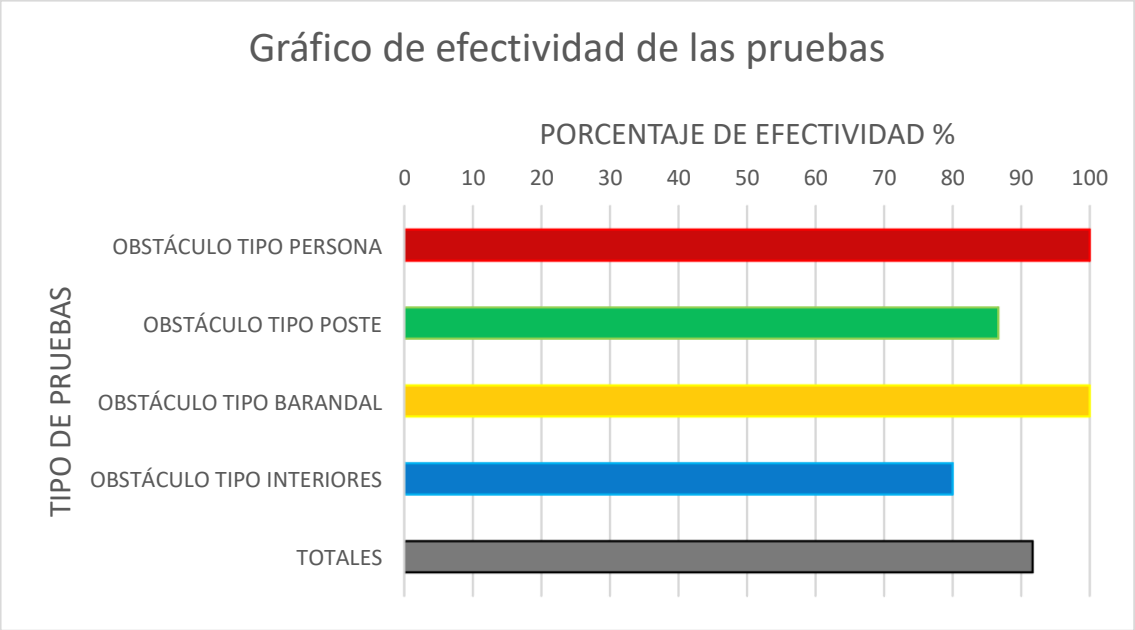

Figura 4.8. Gráfico de efectividad obtenida durante las pruebas.

Por otro lado, en los 5 casos en el que no se presentó de manera adecuada la detección de los diferentes tipos de obstáculos, estos fueron reportados en la posición inicial A del robot.

5. Conclusiones

El desarrollo del presente proyecto de Tesis se enfoca en un sistema de evasión de obstáculos en un robot de 18 GDL, el cual funciona con retroalimentación de un sensor de visión HaVimo2.0 y se elaborara el procesamiento de las imágenes captadas por un clasificador basado en RNA implementado en leguaje C embebido.

Se puede mencionar que el objetivo general y los objetivos particulares se han cumplido tal y como se esperaba, dado que se logró exitosamente el ensamble del robot de 18 GDL en la configuración tipo A de la empresa Robotis© siguiendo los lineamientos de fábrica de la misma empresa para implementar la metodología propuesta, seguido del ensamble, la programación del humanoide no se quedó atrás dado que, se realizó de manera efectiva la ejecución de movimientos tanto en el software de fábrica RoboPlus©, como usando lenguaje C embebido para la toma de decisiones del clasificador, posteriormente de realizar la programación de los movimientos principales del robot, se diseñó y entrenó la RNA con la base de datos previamente capturada en base a imágenes de obstáculos dadas por el sensor HaVimo2.0, permitiendo de esta manera la correcta implementación de la misma RNA en el sistema embebido para la realización de las pruebas del sistema bajo diferentes casos de estudio, después se lleva a cabo un completo análisis y discusión de los resultados que se obtuvieron sobre las pruebas y empleando métodos estadísticos para la validación positiva del método propuesto.

Posteriormente, los resultados de las pruebas realizadas bajo los diferentes casos de estudios descritos en el presente proyecto arrojan en el análisis de estos que de igual manera son descritos, que el método propuesto para la evasión de obstáculos es completamente viable debido a mantener satisfactorios comportamientos del sistema en completo durante la deambulación del humanoide.

A pesar de que hay más métodos o tecnologías implementadas para la realización de la misma tarea, el método propuesto y validado en el presente proyecto de Tesis es viable debido a su bajo costo de implementación, dado que no utiliza nada más que un sensor de visión HaVimo2.0 que en comparación con otros sensores que hay en el mercado, éste tiene un costo significativamente bajo de USD 1610.98, El desarrollo de

este proyecto permite plantear que el robot humanoide tenga la capacidad de participar en competencias nacionales e internacionales como lo es FIRA contra otros robots humanoides que valen 8 veces el valor de éste en el mercado como lo es el robot NAO con un valor de USD 12,990,00 (RobotLAB Group, 2023), y que, sin embargo, se pueda igualar o superar la eficiencia en la identificación y evasión de los obstáculos.

Referencias

Al-Shuka, H. F. N., Corves, B., Zhu, W. H., & Vanderborght, B. (2016). Multi-level control of zero-moment point-based humanoid biped robots: A review. *Robotica*, *34*(11), 2440–2466. https://doi.org/10.1017/S0263574715000107

B, M. S., & Estivill-castro, V. (2019). *ROBO : Robust , Fully Neural Object*. *1*, 309–322.

Badenes Villena, I. J. (2022). *Conceptualización, implementación y desarrollo de un coche autónomo desde cero mediante machine learning*.

Bogue, R. (2020). Humanoid robots from the past to the present. *Industrial Robot*, *47*(4), 465–472. https://doi.org/10.1108/IR-05-2020-0088

Delfin, J., Becerra, H. M., & Arechavaleta, G. (2018). Humanoid navigation using a visual memory with obstacle avoidance. *Robotics and Autonomous Systems*, *109*, 109–124. https://doi.org/10.1016/j.robot.2018.08.010

Fanello, S. R., Ciliberto, C., Noceti, N., Metta, G., & Odone, F. (2017). Visual recognition for humanoid robots. *Robotics and Autonomous Systems*, *91*, 151–168. https://doi.org/10.1016/j.robot.2016.10.001

Fernández Calderón, V. (2018). *Navegación mediante retroalimentación visual de un robot móvil basado en técnicas de aprendizaje automático Deep Learning*.

FIRA. (2022). *ABOUT FIRA*. https://firaworldcup.org/about/

Fritsche, L., Unverzagt, F., Peters, J., & Calandra, R. (2013). *First-Person Tele-Operation of a Humanoid Robot* *. 997–1002.

Garrido, C., Lozano, F., & Higuera, C. (2020). *Sistema de navegación para robot móvil basado en aprendizaje por refuerzo*. 1–11.

Hirose, M., & Ogawa, K. (2007). Honda humanoid robots development. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *365*(1850), 11–19. https://doi.org/10.1098/rsta.2006.1917

Hu, Y.-Y., Huang, K.-H., Chan, H.-M., Hung, C.-H., & Wong, C.-C. (2010). Behaviors design for vision-based humanoid robot. *Proceedings of SICE Annual Conference 2010*, *1*, 18–21.

Hugo, V., Borrego, C., Anhel, K., Gómez, C., Fernando, J., Ramírez, R., Brandom, K., Ramírez, P., Guadalupe, M., & Ramírez, N. (2021). DESARROLLO DE ALGORITMOS PARA LA ESTIMACIÓN DE DISTANCIAS Y LOCALIZACIÓN EN UN ROBOT HUMANOIDE DEVELOPMENT OF ALGORITHMS FOR DISTANCE AND LOCATION ESTIMATION IN A HUMANOID ROBOT. In *Tecnológico Nacional de México en Celaya Pistas Educativas* (Vol. 43, Issue 140). http://itcelaya.edu.mx/ojs/index.php/pistas

Izaurieta, F., & Saavedra, C. (1999). Redes Neuronales Artificiales. *Charlas de Fisica*, 1–15. https://doi.org/10.1016/S0210-5691(05)74198-X

Javier, C., Germ, T., Una, R., & Neuronal, R. (2009). Redes Neuronales Artificiales Introducción Modelo neuronal de McCulloch-Pitts. *Revista De Educación Matemática*, *24*(3), 22–30.

Javier, J., González, T., Cauca, U., Fabian, H., Magé, N., Cauca, U., & Cauca, U. (2015). *SIMULACIÓN DEL CICLO DE MARCHA DEL ROBOT BÍPEDO BIOLOID EN EL ENTORNO VIRTUAL V-REP*. *November*.

Karla Anhel Camarillo Gomez, Jacky Baltes, Meng-Chen Lau, & Kuo-Yang Tu. (2021). HuroCup Laws of the Game Obstacle Run. *HuroCup - RoboWorld Cup*, *0800*.

Kim, J. Y., Park, I. W., & Oh, J. H. (2007). Walking control algorithm of biped humanoid robot on uneven and inclined floor. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *48*(4), 457–484. https://doi.org/10.1007/s10846-006-9107-8

Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., & Inoue, H. (2001). Footstep planning among obstacles for biped robots. *IEEE International Conference on Intelligent Robots and Systems*, *1*, 500–505. https://doi.org/10.1109/iros.2001.973406

Kuo, C. H., Kuo, Y. C., Chen, T. S., Shen, Y. P., & Cheng, C. C. (2014). Petri-net-based implementations for FIRA weightlifting and sprint games with a humanoid robot. *Robotics and Autonomous Systems*, *62*(3), 282–293. https://doi.org/10.1016/j.robot.2013.08.012

Lobos-Tsunekawa, K., Leiva, F., & Ruiz-Del-Solar, J. (2018). Visual navigation for biped humanoid robots using deep reinforcement learning. *IEEE Robotics and Automation Letters*, *3*(4), 3247–3254. https://doi.org/10.1109/LRA.2018.2851148

Martínez Cruz, S., Pérez Soto, G. I., & Morales Hernández, L. A. (2016). *Desarrollo e implementación de software para programación de movimientos a robot Bioloid Premium*.

Morales, E. F., & Sucar, L. E. (2022). *Los Robots del Futuro y su Importancia para Mexico Omnipresencia y dependencia tecnologica*. 1–11.

Muni, M. K., Kumar, P. B., Parhi, D. R., Rath, A. K., Das, H. C., Chhotray, A., Pandey, K. K., & Salony, K. (2020). Path Planning of a Humanoid Robot Using Rule-Based Technique. *Lecture Notes in Mechanical Engineering*, 1547–1554. https://doi.org/10.1007/978-981-15-0124-1_135

Naqa, I. El, & Murphy, M. J. (2015). Machine Learning in Radiation Oncology. *Machine Learning in Radiation Oncology*, 3–11. https://doi.org/10.1007/978-3-319-18305-3

Nobile, L., Randazzo, M., Colledanchise, M., Monorchio, L., Villa, W., Puja, F., & Natale, L. (2021). *Humanoid Robot*. 1–20.

Nunez, J. V., Briseno, A., Rodriguez, D. A., Ibarra, J. M., & Rodriguez, V. M. (2012). Explicit analytic solution for inverse kinematics of bioloid humanoid robot. *Proceedings - 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium, SBR-LARS 2012*, 33–38. https://doi.org/10.1109/SBR-LARS.2012.62

Rath, A. K., Parhi, D. R., Das, H. C., Muni, M. K., & Kumar, P. B. (2018). Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone. *Defence Technology*, *14*(6), 677–682. https://doi.org/10.1016/j.dt.2018.03.008

Regier, P., Milioto, A., Karkowski, P., Stachniss, C., & Bennewitz, M. (2019). Classifying Obstacles and Exploiting Knowledge about Classes for Efficient Humanoid Navigation. *IEEE-RAS International Conference on Humanoid Robots*, *2018-Novem*(1), 820–826. https://doi.org/10.1109/HUMANOIDS.2018.8625036

Robotis. (2006a). Dynamixel AX-12, User's Manual. *Communication*. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Robotis. (2006b). Dynamixel AX-12, User's Manual. *Communication*.

Robotis. (2022). *Robotis e-Manual*. https://emanual.robotis.com/docs/en/edu/bioloid/premium/

ROBOTIS. (2023). *BIOLOID Premium*. https://emanual.robotis.com/docs/en/edu/bioloid/premium/

RobotLAB Group. (2023, July 15). *ROBOT NAO V6 EDICIÓN ESTÁNDAR*. https://www.robotlab.com/latam/store/robot-nao-para-la-educacion

Segura, C., García, M., Díaz, L., Educativas, R. P.-P., & 2017, undefined. (2017). Robot Bioloid Premium Jaranero Controlado Remotamente Por Voz. *Itcelaya.Edu.Mx*, *39*(126), 99–117.

Su, Y. Te, Hu, C. Y., Lu, M. F., Chang, C. M., Lai, S. W., Liu, S. H., & Li, T. T. (2008). Design and implementation of SOPC based image and control system for HuroCup. *Harbin Institute of Technology (New Series)*, *15*(SUPPL. 2), 41–46.

Suarez-Rivera, R. A., Loza-Alvarez, A., Espinosa-Baylon, J. R., Martinez-Cruz, S., Perez-Alvarez, K., Arteaga, D. J., Perez-Soto, G. I., & Camarillo-Gomez, K. A. (2017). Adaptation of robot BIOLOID premium for weightlifting HuroCup-FIRA. *18th Congreso Mexicano de Robotica, COMRob 2016*, 3–5. https://doi.org/10.1109/COMROB.2016.7955156

Tu, K. Y., Lin, H. Y., Li, Y. R., Hung, C. P., & Baltes, J. (2020). First human-robot archery competition: A new humanoid robot challenge. *Knowledge Engineering Review*, *35*, 1–12. https://doi.org/10.1017/S026988892000003X

Valero Carvajal, D. A., Orozco Isaza, J. E., Córdoba Puerto, J. N., Noreña Hemelberg, O. S., Gómez Alarcón, M. A., Jinete Gómez, M. A., & Gutiérrez Alfonso, J. A. (2021). Algoritmos para el procesamiento de imágenes implementados en el Robot Humanoide InMoov. *Revista EIA*, *18*(36), 1–6. https://doi.org/10.24050/reia.v18i36.1495

Yoo, J., Lee, B., & Kim, J. (2014). *Vision simulator of HSR-VII for obstaclerun in HuroCup VISION SIMULATOR OF HSR-VII FOR OBSTACLERUN IN HUROCUP*. *August*.

Apéndices

Código de entrenamiento de la RNA

Instalación de programas para C embebido

Código de programación en C embebido