UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

OPTIMIZACIÓN DE TRAYECTORIAS EN PLATAFORMAS ROBÓTICAS MÓVILES USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Trabajo de titulación modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Andrés David Soto Rodríguez

TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

Ambato – Ecuador

agosto - 2023

APROBACION DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: OPTIMIZACIÓN DE TRAYECTORIAS EN PLATAFORMAS ROBÓTICAS MÓVILES USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Andrés David Soto Rodríguez, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, agosto 2023.

Firmado electrónicamente por:
VICTOR SANTIAGO
MANZANO VILLAFUERTE

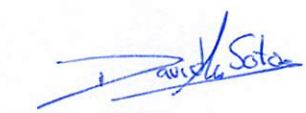
Ing. Víctor Santiago Manzano Villafuerte, Mg.

TUTOR

AUTORÍA

El presente trabajo de titulación titulado: OPTIMIZACIÓN DE TRAYECTORIAS EN PLATAFORMAS ROBÓTICAS MÓVILES USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL, es absolutamente original, autentico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efecto legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto 2023.



Andrés David Soto Rodríguez

CC. 1805143201

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este Trabajo de Titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, agosto 2023.

Andrés David Soto Rodríguez

CC. 1805143201

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Andrés David Soto Rodríguez, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación titulado OPTIMIZACIÓN DE TRAYECTORIAS EN PLATAFORMAS ROBÓTICAS MÓVILES USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, agosto 2023.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.

PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
ANA PAMELA CASTRO MARTIN

Ing. Ana Pamela Castro Martin, Mg.

PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARLON ANTONIO SANTAMARIA VILLACIS

Ing. Marlon Antonio Santamaría Villacís, Mg.

PROFESOR CALIFICADOR

DEDICATORIA

A mi abuelito que está en el cielo, Galo Arcenio Rodríguez Jarrín, quiero que sepa que lo logré y cada día que pasa le extraño mucho.

Con todo mi amor a mi mamita, Susana María Rodríguez Cevallos, que supo instruirme en mi camino para que hoy, pese a todo, no me haya apartado de el (Proverbios 22:6).

A mi hermana, Andrea Vanessa Soto Rodríguez, quien desde muy temprana edad nos ha ayudado a salir adelante sin darse por vencida, trabajando duro día a día, quiero que sepas que te quiero muchísimo.

A mi tío, Juan Pablo Rodríguez Cevallos, quien siempre creyó en mí ayudándome a forjar mi carácter desde muy pequeño, quiero que sepa que lo admiro mucho y lo considero como mi ejemplo a seguir por todo lo que tuvo que luchar para salir adelante, espero que el tiempo nos reúna pronto.

Andrés David Soto Rodríguez

AGRADECIMIENTO

Agradezco a Dios por derramar sus bendiciones, en mí y en mi familia día tras día, comprobando cada mañana que él es y siempre será fiel.

Andrés David Soto Rodríguez

ÍNDICE GENERAL DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El crecimiento de la industria robótica a medida que avanza el tiempo es de manera exponencial y las empresas que buscan automatizar sus procesos lo hacen con el fin de optimizar los recursos como el tiempo empleado.

Una de las soluciones actuales para la optimización de varios procesos es el uso de la inteligencia artificial (IA), al permitir aprendizajes en entornos supervisados, donde la instrumentación robótica al tener implementaciones de algoritmos de IA tiende a minimizar el margen de error a futuro.

En el presente proyecto se expone una solución para la optimización de trayectorias utilizando como apoyo un algoritmo de IA de aprendizaje por refuerzo con redes neuronales implementado en la plataforma robótica omnidireccional del robot KUKA youBot para desplazarse de un punto a otro, esquivando obstáculos presentados en su camino.

El algoritmo de IA utilizado para el aprendizaje es Deep Q Network (DQN), este algoritmo consta redes neuronales profundas para maximizar alguna noción de recompensas de forma acumulativa, donde por medio de un sensor de movimiento lidar Hokuyo, colocado en la parte frontal de la plataforma robótica, se adquieren muestras de datos del entorno, que son procesadas en el algoritmo para ser reconocidas como colisiones o recompensas. A medida que las recompensas aprendidas por el algoritmo son mayores, la posibilidad de colisión ante un obstáculo disminuye, desplazando a la plataforma robótica hacia una zona libre de obstáculos.

El lenguaje de programación de este algoritmo DQN es basado en Python 2, este lenguaje trabaja junto con ROS (sistema operativo robótico) y permite conocer, de manera entendible, cómo se realiza la ejecución del movimiento por medio de la publicación y suscripción a los tópicos correspondientes de la plataforma robótica, facilitando se esta manera la calibración de los parámetros utilizados en la misma.

Palabras clave: KUKA youBot, Python, inteligencia artificial, optimización de trayectorias.

ABSTRACT

As time progresses, the growth of the robotics industry is exponential and companies that seek to automate their processes do so to optimize resources, such as the time spent.

The use of artificial intelligence is one of the current solutions for the optimization of various processes, by allowing learning in supervised environments, where robotic instrumentation tends to minimize the margin of error in the future thanks to implementations of AI algorithms.

In the present project, a solution for the optimization of trajectories is exposed using as support an AI algorithm of reinforcement learning with neural networks implemented in the omnidirectional robotic platform of the KUKA youBot robot to move from one point to another avoiding obstacles presented in its path.

The AI algorithm used for learning is Deep Q Network (DQN), this algorithm consists of deep neural networks to maximize some notion of rewards in a cumulative way, whereby means of a Hokuyo lidar motion sensor, placed in the front part of the robotic platform, they are acquired. You sample data from an environment, which is processed in the algorithm to be recognized as collisions or rewards. As the rewards learned by the algorithm are greater, the possibility of collision with an obstacle decreases, moving the robotic platform towards an obstacle-free zone.

The programming language of this DQN algorithm is based on Python 2, this language works together with ROS (robotic operating system) and allows to know, in an understandable way, how the execution of the movement is carried out through the publication and subscription to the topics corresponding to the robotic platform, thus facilitating the calibration of the parameters used in it.

Keywords: KUKA youBot, Python 2, artificial intelligence, trajectory optimization.

CAPITULO I

MARCO TEÓRICO

1.1. Tema de investigación

OPTIMIZACIÓN DE TRAYECTORIAS EN PLATAFORMAS ROBÓTICAS MÓVILES USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

1.1.1. Planteamiento del problema

En la actualidad a nivel mundial, la mayoría de las empresas fabricantes requieren procesos de automatización, debido al crecimiento exponencial que tiene el mercado en algunos ámbitos, intentando de esta manera aumentar la eficiencia y optimizar en lo posible costes de producción, tal es el caso que se espera que la inversión en automatización para el año 2030 sea de $412,80 billones a comparación con el año 2022 que es de $213,49 billones [1].

Una de las necesidades que llevan a las empresas a invertir en procesos de automatización, es la optimización del tiempo, ya que las tareas manuales, especialmente las tareas repetitivas, son mucho más susceptibles al error humano, en pocas palabras automatizar procesos significa reducir los pasos de producción manual [2].

El transporte de material de un sitio a otro dentro de una empresa se ha convertido en una necesidad para la optimización de tiempo y no solo el transporte de material si no también la realización de otras actividades guiadas, esta necesidad ha llevado a un incremento en el mercado de la demanda de vehículos guiados automatizados en América Latina tal es el caso que en 2020 este mercado se valoró en $177.5 millones y se espera que para el año 2026 se valore en $963.8 millones [3].

En el Ecuador la industria muestra un crecimiento continuo en la implementación de procesos automatizados, ubicándose así el país en el tercer puesto de los porcentaje de trabajadores en ocupaciones con alto riesgo de ser reemplazados por robots con un 69%, uno de los factores para que se presente esta situación es mejorar el transporte de material de un punto a otro, lo cual conlleva una pérdida de tiempo ya que los propios empleados de las empresas realizan esta acción y las pocas empresas que tienen una automatización implementada, esta es tradicional en donde los vehículos guiados automáticamente solo se basan en desplazarse por una trayectoria previamente establecida [4].

1.2. Antecedentes investigativos

Elmer Barahona Guamani, en su trabajo de investigación titulado, "Navegación autónoma basada en maniobras bajo estimación de posturas humanas para un robot omnidireccional KUKA youBot", en la Universidad Técnica de Ambato en el año 2019. Incursionó en la implementación de un sistema de navegación con visión artificial encargado de enviar ordenes de rutas al robot KUKA youBot y realizar maniobras aplicando el reconocimiento de posturas humanas. Esto se logró mediante la utilización de softwares libres como OpenCV, creando un espacio de descriptores en un plano 2D, eligiéndose las posturas mediante algoritmos Surf, este robot logró reconocer estas posturas gracias a la utilización de una tarjeta NVIDIA jetson nano, la cual recibe las imágenes para posteriormente aplicar el algoritmo de IA [5].

Viridiana Yatzen Hernández Márquez, en su trabajo de investigación titulado "Generación de trayectorias para tareas de navegación autónoma y mapeo en ambientes interiores" en la Universidad Politécnica de Tulancingo en el año 2018 en México. Implementó un algoritmo que genera rutas por medio de creación de mapas autónomos con la utilización de un robot móvil. Este robot utiliza la plataforma de código abierto llamado TurtleBot, juntamente con un algoritmo de implementación de nodo ROS (Robot Operating System) para el funcionamiento de programas de mapeo y odometría, logrando así un mapeo de rutas de un área cerrada teniendo en cuenta la generación de una trayectoria subóptima elegida en algún punto la cual afectaría a procesos posteriores [6].

Clelio Endara Sumba, Emerson Maigua Yánez, en su trabajo de investigación titulado "Desarrollo de un algoritmo de trayectoria para un robot seguidor de línea destreza de competencia mediante visión e inteligencia artificial" en la Universidad Politécnica Salesiana, sede Quito en el año 2020. Incursionaron en el reconocimiento de trayectorias al utilizar una red neuronal basada en inteligencia artificial, donde el robot sigue una ruta establecida. La efectividad lograda al poner el robot un escenario controlado específicamente para competencias es alta, gracias a la aplicación de una red conformada por 12 neuronas juntamente con una tarjeta idónea para el manejo de esta red neuronal la cual es OpenMV Cam H7 con núcleo ARM Cortex A8 Sitara AM335x. La IA introducida en el mismo logra en un número de intentos tener un porcentaje de pista recorrida del 100%, teniéndose tiempos promedios comparados entre 5 neuras con 64,897 segundos, con 10 neuronas con 59,042 segundos y con 12 neuronas con un tiempo de 54,851 segundos y a partir de esta cantidad de neuronas el tiempo mejora tan solo en un 0,1 segundo [7].

Juan Manuel León Muñoz, en su trabajo de investigación titulado "Detección de señales y planificación de trayectorias en un robot móvil", en la Escuela Técnica Superior de Ingeniería Universidad de Sevilla, en el año 2021, desarrollo un sistema de detección de señales de tráfico las cuales reaccionan ante las mismas, esto gracias al uso de un robot móvil Rosbot 2.0. La idea que se describe es la utilización de una cámara RGBD junto con un sensor LIDAR para la detección de estas señales, asimismo recoger la información extraída en el reconocimiento de imágenes gracias a un algoritmo desarrollado que utiliza principalmente nodos "Find_Object" y "Gmapping" para el mapeo del entorno, logrando de esta manera que el robot generar un mapa del entorno que rodea al mismo y trazar trayectorias para posteriormente recorrerlas [8].

D. Vázquez Lucero, E. Luna Taylor, I. Santillán, C. Higuera, en su trabajo de investigación titulado "Robot móvil autónomo con reconocimiento y navegación hacia botellas de plástico", en la Universidad Autónoma del Estado de Hidato en México, en el año 2022, diseñó un prototipo de robot autónomo capaz de reconocer botellas de plástico, aplicando una estrategia semántica por medio de IA con la creación de redes

neuronales convolucionales. Dentro de los experimentos realizados se logró un 96.6% de precisión en el reconocimiento de botellas de plástico. Este prototipo de robot autómata utiliza una tarjeta de NVIDIA jetson Nano B01, la cual es programa en base al lenguaje de programación Python, para recibir las imágenes de la cámara incorporada del robot. En las imágenes se aplica un modelo entrenado de red neuronal para identificar los contornos de las botellas y obtener el centroide de estas. Luego mediante la utilización de una tarjeta Arduino se recibe el área y centroide y gracias un sistema de inferencia difusa se calcula la potencia impresa en cada motor correspondiente del robot [9].

1.3. Fundamentación teórica

1.3.1. Robots Autónomos

Se conocen como vehículos guiados automáticamente o por sus siglas en inglés (AGV), son sistemas robóticos móviles. Por lo general, siguen una trayectoria que está señalada en el piso de la nave. También existen AGV autónomos que no necesitan una trayectoria cableada. Asimismo, hay características adicionales, por ejemplo, la capacidad de movimiento omnidireccional, a diferencia de las ruedas convencionales de dos grados de libertad (DOF) que se usan en automóviles y otros AGV, proporcionan tres DOF; es decir, estos AGV también pueden moverse lateralmente. Los AGV se usan asimismo en hospitales en enfermería, en vigilancia y otras aplicaciones [10].

1.3.2. Plataforma robótica móvil

Una plataforma robótica móvil es esencialmente un robot que contiene todo lo necesario para su pilotaje y movimiento (potencia, control y sistema de navegación). Con el paso del tiempo, el incremento de posibilidades con que la movilidad dota a un robot y las dificultades especiales que surgen al abordar esta capacidad, han originado que los robots móviles hayan sido y sean motivo de interés de numerosas investigaciones [11].

1.3.3. Robot omnidireccional autónomo KUKA youBot

Conocido como el pequeño gran robot, KUKA youBot está compuesto por una plataforma móvil y un brazo robótico. Para el accionamiento de estos componentes se emplean sistemas motores brushless de maxon motor. KUKA youBot es un pequeño robot móvil que ha sido desarrollado como una plataforma de código abierto para la investigación y la enseñanza científicas [12].

Figura 1. Robot omnidireccional autónomo KUKA youBot [12].

Especificaciones de hardware:

- **Pinza youBot:** pinza desmontable de 2 dedos unida al brazo (carrera de 10 mm/dedo = carrera de apertura de 20 m; diferentes montajes permiten sujetar objetos de hasta 70 mm de diámetro).
- **Plataforma youBot:** plataforma móvil omnidireccional con 4 KUKA omniWheels (530x360x106 mm, 15 mm de espacio libre, 20 kg de peso total, 20 kg de carga útil, velocidad mínima 0.01m/s, velocidad máxima 0.8 m/s, comunicación EtherCAT, 24 V).
- **Suministro de energía:** un juego de dos baterías (dos baterías recargables de plomo-ácido de 12 V, 5 Ah, libres de mantenimiento; autonomía aproximada: 90 minutos) integradas en la plataforma móvil.
- **PC integrada:** factor de forma de placa mini ITX (CPU integrada, refrigeración pasiva, 512 MB de RAM, Compact Flash de 4 GB, WLAN, USB), integrada en la plataforma móvil [13].

Especificaciones del software:

- **Controlador de robot de código** abierto con interfaces abiertas (posición, velocidad y control de corriente)
- **BRIDE** (BRics Integrated Development Environment) basado en Eclipse para simplificar el desarrollo de aplicaciones
- **BROCRE** (BRICS Open Code Repository) ofrece interfaces interoperables y bibliotecas de algoritmos de mejores prácticas: BRICS_MM para manipulación móvil, BRICS_3D para percepción y modelado 3D, BRICS_RN para navegación robusta [13].

1.3.4. Plataforma omnidireccional KUKA youBot

La base del robot KUKA youBot, es una plataforma móvil omnidireccional con cuatro ruedas mecánicas. Se encuentra en el centro de odometría. Los valores positivos para la rotación sobre el eje z dan como resultado un movimiento en sentido contrario a las agujas del reloj, como lo indica la flecha azul en la Figura 2, así como la numeración de las ruedas [14]. Se debe tener muy en cuenta como trabaja el eje de coordenadas en la plataforma del robot ya que este mismo eje se muestra la simulación en Gazebo.

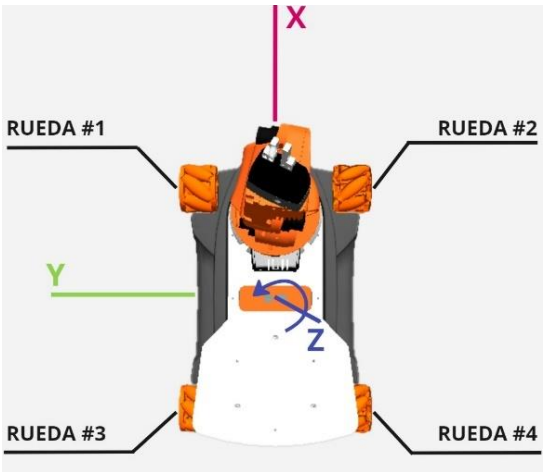


Figura 2. Coordenadas y ruedas plataforma KUKA youBot.

Fuente: Investigador.

A continuación, se tiene la plataforma omnidireccional del robot KUKA youBot simulado en Gazebo, donde se aprecia los ejes de coordenadas en cada una de sus ruedas.

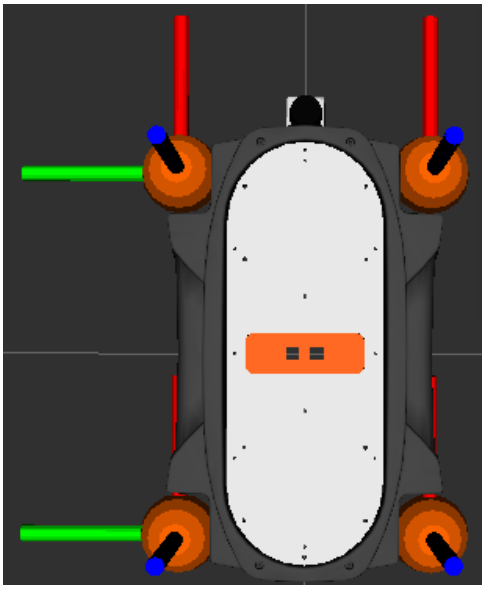


Figura 3. Simulación plataforma robot KUKA youBot Gazebo, ejes de coordenadas

Fuente: Investigador.

La plataforma omnidireccional del robot KUKA youBot cuentan con especificaciones de medidas en su diseño, las cuales se presentan en la Tabla 1.

Tabla 1. Medidas de la estructura de la plataforma KUKA youBot [15].

Características	Detalles
Longitud	580mm
Ancho	376mm
Altura	140mm
Distancia con respecto al suelo	30mm

Para poder apreciar mejor las especificaciones de proporción de tamaño de los componentes que conforman la plataforma se presenta en la siguientes Figuras 4 y 5.

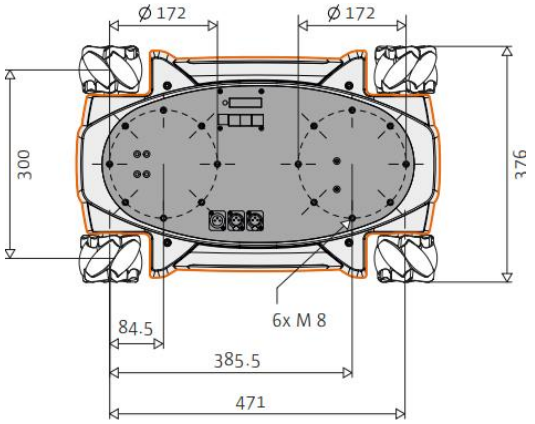


Figura 4. Medidas plataforma KUKA youBot vista superior [15].

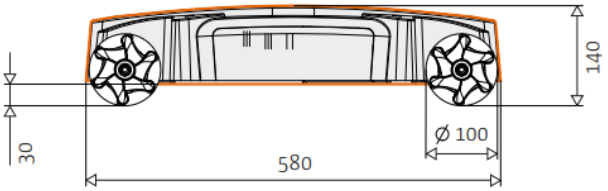


Figura 5. Medidas plataforma KUKA youBot vista lateral [15].

1.3.5. OmniRueda

El Omniwheel u OmniRueda en español, consiste en rodillos convexos no accionados, que están montados entre 2 llantas en un ángulo de aproximadamente 45°. Los rodillos están dispuestos en de tal manera que repliquen un círculo. Las ruedas no tienen mecanismo de dirección. Su velocidad de traslación, rotación y el centro de rotación del vehículo están determinados esencialmente por el sentido de giro de las ruedas en relación unos con otros [16].

Figura 6. Disposición de rodillos en Omnirueda [16].

1.3.6. Concepto de dirección

La dirección de viaje se puede definir, por ejemplo, dividiendo el ángulo en un círculo: 0º corresponde a la dirección de viaje "hacia delante", 90º corresponde al sentido de marcha "derecha", 180º corresponde al sentido de marcha “hacia atrás” y así sucesivamente [16].

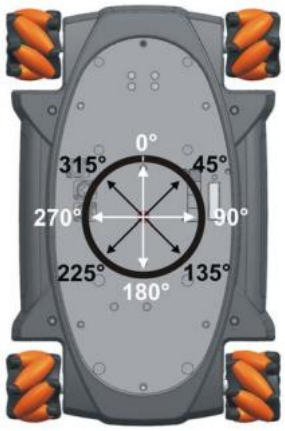
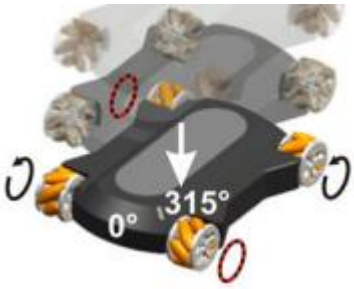
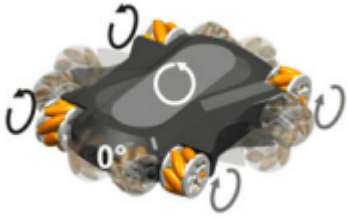


Figura 7. Definición de dirección de movimiento plataforma KUKA youBot [16].

La ejecución de movimiento que realiza la plataforma del robot en todas las direcciones se describe en la Tabla2.

Tabla 2. Ejemplos de la relación entre la dirección de recorrido y sentido de giro de las ruedas [16].

Dirección de movimiento	Descripción
Movimiento en línea recta 	Las ruedas de la plataforma se mueven a la misma velocidad y en una sola dirección.

Movimiento hacia los lados 	Cada rueda se mueve en dirección opuesta a su rueda de al lado. Las ruedas mantienen a la misma velocidad en la ejecución de movimiento realizado.
Movimiento en diagonal 	Cada par de ruedas a lo largo de una diagonal se mueve en la misma dirección y a la misma velocidad. Aquí el par de ruedas diagonalmente opuesto (representado con un círculo de rojo y negro) permanecen estacionarias.
Rotación sobre el eje central 	Las ruedas de un lado giran en dirección opuesta a las ruedas del otro lado. Teniendo en cuenta que las ruedas giran en la misma velocidad

Para el movimiento en curvas, cada par de ruedas a lo largo de una diagonal se mueve en la misma dirección y a la misma velocidad. El par de ruedas diagonalmente opuestas también se mueve en la misma dirección, pero a una velocidad más rápida o lenta, mientras existe más diferencia de velocidad, más cerrada es la curva.

1.3.7. Sensor Lidar

Lidar significa detección de luz y alcance, similar al sonar y al radar con ondas de sonido y radio respectivamente. La tecnología Lidar utiliza haces de luz para detectar el alcance de un objeto [17].

Existen gran variada de sensores Lidar, de los cuales se pueden definir tres categorías importantes las cuales son [17]:

- 1D - Mide la distancia a un solo punto. Estos se utilizan a menudo como "cintas métricas digitales".
- 2D: Mide muchos puntos en un plano de exploración. Estos se utilizan a menudo en robots móviles para crear un plano de planta para navegar. Los modelos varían en frecuencia, rango (distancia máxima), resolución y campo de visión horizontal. Estos a veces se denominan "escáneres láser".
- 3D: En lugar de un solo plano de escaneo 2D, estos lidars ven en tres dimensiones. Algunos modelos pueden tratarse efectivamente como cámaras 3D, devolviendo una imagen con una distancia medida a cada píxel.

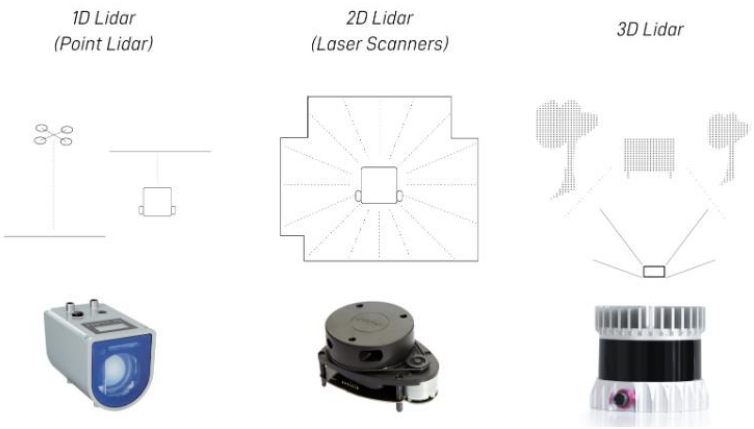


Figura 8. Clasificación por categoría de sensores Lidar [3].

1.3.8. Inteligencia Artificial

Dentro de la inteligencia artificial se encuentran ramas que la componen, teniéndose:

- Lógica difusa
- Redes neurales artificiales
- Algoritmos genéticos

Lógica Difusa

La lógica difusa es una rama de la IA que le permite a una computadora analizar información del mundo real en una escala entre lo falso y verdadero. Los matemáticos dedicados a la lógica en la década de 1920 definieron un concepto clave: todo es cuestión de grado. La lógica difusa manipula conceptos vagos como “caliente” o “húmedo” y permite a los ingenieros construir televisores, acondicionadores de aire,

lavadores y otros dispositivos que juzgan información difícil de definir. Los sistemas difusos son una alternativa a las nociones de pertenencia y lógica que se iniciaron en la Grecia antigua [18].

Redes neurales artificiales

Las redes neurales se basan en generalizar información extraída de datos experimentales, las cuales son validadas previamente por expertos. Dichas redes neurales toman en cuenta las entradas (corriente, voltaje) y como salidas las señales del sistema (velocidad, temperatura, torque). La red neural utilizada es una red multicapa de diez neuronas en la capa de entrada, diez neuronas en la capa oculta y cinco neuronas en la capa de salida. Por lo tanto, se tienen 250 pesos ajustables mediante un control retroalimentado o de lazo cerrado. En la Figura 9 que se muestra a continuación se presenta un diagrama de red neural. Los parámetros de inicialización se obtuvieron mediante un conjunto de datos experimentales y una base de datos [18].

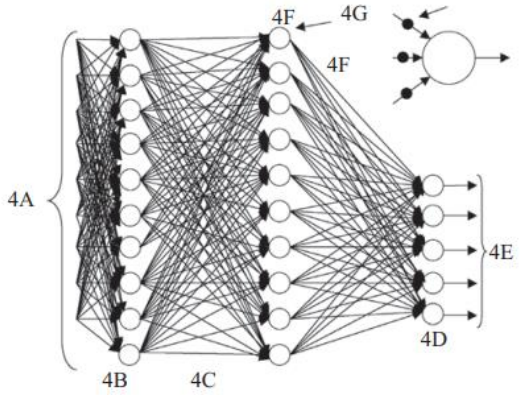


Figura 9. Esquema de la red neural multicapas [18].

Algoritmos genéticos

Un algoritmo genético (AG) es una técnica de búsqueda iterativa inspirada en los principios de selección natural. Los AG no buscan modelar la evolución biológica sino derivar estrategias de optimización. El concepto se basa en la generación de poblaciones de individuos mediante la reproducción de los padres [18].

En los AG las cadenas están compuestas por características, que toman diferentes valores. Estas características se localizan en distintas posiciones de la cadena [18].

Los componentes de un algoritmo genético son [18]:

- Una función que se desea optimizar.
- Un grupo de candidatos para la solución.
- Una función de evaluación que mida cómo los candidatos optimizan la función.
- Función de reproducción.

1.3.9. Machine Learning

Conocido en español como aprendizaje automático, tiene como objetivo para clasificación (o aprendizaje supervisado) en inducir una aproximación (modelo o hipótesis) h de una función desconocida f definida desde un espacio de entrada X hacia un espacio discreto y desordenado Y = 1, . . . ,K, dado un conjunto de entrenamiento S [19].

El conjunto de entrenamiento $S = \{(x_1, y_1), \ldots., (x_n, y_n)\}$ contiene n ejemplos de entrenamiento, que corresponden a pares (x,y) donde x ∈ X e y = f (x). La componente x de cada ejemplo es un vector $x = (x_1, \ldots, x_n)$ de atributos con valores continuos o discretos que describen la información relevante o propiedades del ejemplo. Los valores del espacio de salida Y asociados a cada ejemplo son las clases del problema. Así, cada ejemplo de entrenamiento queda totalmente caracterizado por un conjunto de pares atributo–valor y una etiqueta de clase [19].

1.3.10. Deep Learning

El aprendizaje profundo o "deep learning" se refiere a una clase bastante amplia de técnicas y arquitecturas de aprendizaje automático o "machine learning", con el sello de utilizar muchas capas de procesamiento de información no lineal que son de naturaleza jerárquica. Se basa en la creación y entrenamiento de redes neuronales artificiales profundas para analizar y aprender de los datos. Dependiendo de cómo se pretenda usar las arquitecturas y técnicas, por ejemplo: síntesis/generación o reconocimiento/clasificación; se puede categorizar ampliamente la mayor parte del trabajo en tres clases principales [20]:

- **Redes profundas para el aprendizaje no supervisado o generativo**, el objetivo de esta categoría es capturar una correlación de alto orden de los datos observados o visibles para fines de análisis o síntesis de patrones cuando no hay información disponible sobre las etiquetas de clase objetivo. Cuando se usa en el modo generativo, también puede estar destinado a caracterizar distribuciones estadísticas conjuntas de los datos visibles y sus clases asociadas cuando estén disponibles y se traten como parte de los datos visibles.
- **Redes profundas para el aprendizaje supervisado**, estas redes tienen como objetivo proporcionar directamente poder de discriminación para fines de clasificación de patrones, a menudo caracterizando las distribuciones posteriores de clases condicionadas a los datos visibles.
- **Redes profundas híbridas**, el objetivo de esta red es la discriminación que es asistida, a menudo de manera significativa, con los resultados de redes profundas generativas o no supervisadas. El objetivo también se puede lograr cuando se utilizan criterios discriminativos para el aprendizaje supervisado para estimar los parámetros en cualquiera de las redes profundas generativas o profundas no supervisadas.

1.3.11. Algoritmos preestablecidos de inteligencia artificial en base al reconocimiento de trayectorias en plataformas robóticas.

El aprendizaje por refuerzo se basa en la administración de pares de estado-acción, manteniendo una recompensa a una acción para determinar la política óptima.

En lugar de utilizar una tabla se utilizan redes neuronales para predecir valores para acciones [21].

A continuación, en las siguientes tablas 3 y 4 se detallan los algoritmos utilizados para el aprendizaje profundo involucrando redes neuronales, partiendo del algoritmo Q-Learning siendo la base de donde se derivan los mismos.

Tabla 3. Descripción de los tipos de algoritmos de aprendizaje por reforzado profundo.

Algoritmo	Descripción
Q- Learning	Es un algoritmo de aprendizaje para buscar la mejor acción a tomar dependiendo del estado actual, este algoritmo aprende de las acciones que están fuera de la política en la que se encuentre, es decir realizar acciones aleatorias sin la necesidad de utilizar una política, buscando de esta manera maximizar la recompensa total [22].
Q – Learning con redes neuronales (DQN)	Se considera un algoritmo de aprendizaje por refuerzo profundo que combina el algoritmo Q-Learning con redes neuronales profundas, este algoritmo es aplicado a la industria de videojuegos interviniendo también la robótica [23]. La idea de este algoritmo es utilizar redes neuronales profundas para representar la red Q y entrenar esta red para predecir la recompensa total así mismo [24].
Double DQN	Este algoritmo se compone de dos redes neuronales de aprendizaje profundo las cuales son Deep Q Network (DQN) y Target Network (es objetivo) [25]. Estas dos redes funcionan en conjunto, en donde la red neuronal principal, DQN, decide la mejor acción entre todas las posibles, y luego la red objetivo, Targen Network, evalúa esa acción para conocer su valor-Q [26].
Dueling DQN	Este algoritmo, divide la red neuronal en dos partes, consideradas funciones; una función es encargada de reflejar el valor de la recompensa desde el primer estado, mientras que la segunda función refleja cuánto mejor es una acción respecto a las demás [26].

Tabla 4. Ventajas y desventajas de los tipos de algoritmos de aprendizaje reforzado profundo.

Algoritmo	Ventajas	Desventajas
Q- Learning	• Funciona en un entorno simple • Se puede representar en un tabla o matriz con cantidad de valores moderados [22].	• Al existir datos de mayor volumen para el aprendizaje, la tabla para la visualización de estos no es viable [22].
Q – Learning con redes neuronales (DQN)	• Permite que RL (aprendizaje por refuerzo) funcione en entornos complejos de gran dimensión [23]. • Corrige errores en los cuales los algoritmos RL son propensos a sobre ajustarse en los modelos de aprendizaje por refuerzo [24].	• Sobreestima las recompensas, en donde los valores Q que aprende, interpreta que va a recibir una recompensa mayor, la cual ya está previamente definida [26].
Double DQN	• Reduce las sobreestimaciones en las recompensas producidas en el algoritmo DQN.	• DDQN requiere una mayor cantidad iteraciones de realizar acciones en diferentes estados y obtener recompensas, lo cual hace que en rendimiento en ocasiones no tenga unas mejoras [27].
Dueling DQN	Es eficiente en entornos donde hay muchas acciones para elegir permitiendo que la red diferencie mejor las acciones entre sí mejorando el aprendizaje [28].	• La división de la red neuronal ayuda en algunos casos, ya que a veces no es necesario saber exactamente al valor de cada acción [26].

1.4. Objetivos

1.4.1. Objetivo General

Implementar la optimización de trayectorias en plataformas robóticas móviles usando técnicas de inteligencia artificial.

1.4.2. Objetivos Específicos

- Analizar algoritmos preestablecidos de inteligencia artificial en base al reconocimiento de trayectorias en plataformas robóticas.
- Implementar un algoritmo de inteligencia artificial para la optimización de trayectorias en la plataforma del robot KUKA youBot.
- Diseñar una aplicación para la optimización de trayectorias en la plataforma del robot KUKA youBot.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

2.1.1. Gazebo

Gazebo brinda la capacidad de simular poblaciones de robots en entornos interiores y exteriores complejos de manera precisa y eficiente. Actualmente, Gazebo es más adecuado para ser utilizado en sistemas basados en Ubuntu [29].

Figura 10. Logo Gazebo [29].

2.1.2. RViz

RViz es una plataforma de visualización 3D en ROS. Por un lado, realiza la visualización gráfica de información externa y, por otro envia información de control a un objeto a través de rviz, realizando el seguimiento y control de un robot [30].

Figura 11. Entorno RViz Linux [30].

2.1.3. Sistema Operativo Robótico ROS

Robot Operating System (ROS) es un middleware robótico, es decir, una colección de frameworks para el desarrollo de software de robots. ROS provee los servicios estándar de un sistema operativo a pesar de no serlo como lo es la abstracción del hardware, el control de dispositivos de bajo nivel, la implementación de funcionalidad de uso común, el paso de mensajes entre procesos y el mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. Una de las características principales que brinda ROS son sus librerías orientadas para un sistema UNIX (Ubuntu -Linux), además estas se están adaptando a otros sistemas operativos como Fedora, Mac OS X, Arch, Gentoo, OpenSUSE, Slackware, Debian o Microsoft Windows, considerados hoy como 'experimentales' [31].

2.1.4. Lenguaje de programación para inteligencia artificial Python

Existen diversos lenguajes de programación para inteligencia artificial, destacándose en la actualidad el uso del lenguaje Python.

Se tienen otros lenguajes de programación entre los cuales incluyen Java, C++ y JavaScript, pero es probable que Python sea la mejor opción para el desarrollo de IA [32]. Dentro de las características que tiene Python se destacan:

Fácil de aprender. La sintaxis de Python es extremadamente flexible y el lenguaje tiene muchas características de calidad de vida y facilidad de uso. Incluso los no programadores encontrarán que Python es intuitivo. Esta baja barrera de entrada es importante porque muchos científicos y analistas de datos que trabajan con IA no tienen experiencia en programación.

Bien integrado. Los programadores no necesitan reinventar la rueda. Muchos marcos, bibliotecas y plataformas de IA ya se han desarrollado en Python y están disponibles como proyectos de código abierto.

Bien documentado. Es aún más fácil aprender Python porque hay muchos tutoriales, proyectos y campamentos de entrenamiento en línea. Es posible que los idiomas menos populares no tengan tantos ejemplos disponibles.

Simple y fácil de leer. El código que es más fácil de leer es más fácil de desarrollar. Python produce código corto extremadamente legible, especialmente en comparación con lenguajes como Java.

Plataforma independiente. Python puede ejecutarse en prácticamente cualquier plataforma, desde Windows hasta Unix. No tiene que ser compilado porque es un lenguaje interpretado.

Excelentes herramientas de visualización. Python tiene una amplia selección de bibliotecas de visualización de datos, esenciales para el desarrollo de IA. Los científicos de datos pueden crear gráficos atractivos y legibles por humanos con bibliotecas como Matplotlib [32].

2.1.5. Sensor Lidar URG-04LX-UG01

El sensor físico utilizado propiamente para el robot KUKA youBot junto con su caja se muestra en la Figura 12.

Figura 12. Sensor Lidar URG-04LX-UG01

Fuente: Investigador.

Es un sensor láser bidimensional es decir 2D, la fuente de luz de este sensor es un láser infrarrojo con longitud de onda de 785nm. El área de escaneo es un semicírculo de 240º con un radio máximo de 4000 mm. El ángulo de inclinación es de 0, 36º y el sensor emite la distancia medida en cada punto (683 pasos) [33].

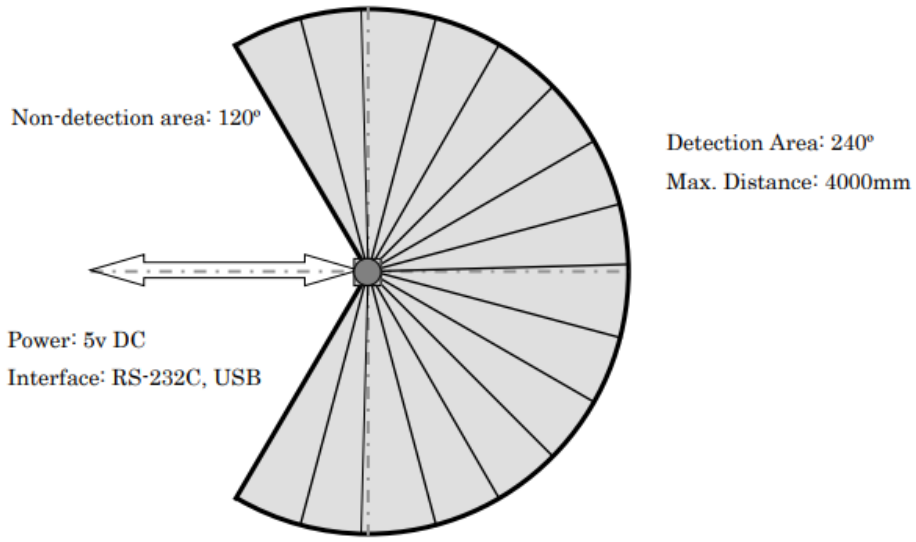


Figura 13. Área detectable sensor Lidar URG-04LX-UG01 [33].

El diámetro del rayo láser se encuentra en un rango inferior de 20 mm a 2000 mm con una divergencia máxima en un rango de 40 mm a 4000 mm. El principio de la medición de distancia se basa en el cálculo de la diferencia de fase, por lo que es posible obtener una medición estable con una influencia mínima del color del objeto y reflectancia, este sensor está diseñado bajo los estándares JISC8201-5-2 e IEC60947-5-2 para aplicaciones industriales [33].

2.2. Métodos

2.2.1. Modalidad de investigación

El presente proyecto se sustentó en una Investigación Aplicada, ya que se empleó los conocimientos adquiridos durante el transcurso de la carrera juntamente con investigaciones que sustentaron la realización del proyecto; estas investigaciones se lo

obtuvieron tanto en las bases de datos como en las bibliotecas virtuales, los mismos que son proporcionados por la Universidad.

Además, se hizo una Investigación Experimental, debido a que en el desarrollo del proyecto se utilizó diversos escenarios predeterminados para el entrenamiento de la inteligencia artificial en una misma plataforma robótica para optimizarla en el reconocimiento de rutas y objetos.

2.2.2. Recolección de información

La recolección de información fue mediante el uso de libros, revistas, fuentes online y proyectos desarrollados, así mismo se tomó como referencia la programación con la que cuenta actualmente el robot KUKA youBot para el levantamiento del nodo maestro y ejecutar un modelo IA entrenado en la plataforma móvil.

2.2.3. Procesamiento y análisis de datos

En el procesamiento y análisis de datos se realizó los siguientes pasos:

- Análisis de todas las fuentes de información documental que tengan relación con el uso de trayectorias en plataformas robóticas móviles usando técnicas de inteligencia artificial.
- Estudio de las propuestas de solución planteadas para la implementación de algoritmos de inteligencia artificial en el uso trayectorias en plataformas robóticas móviles.
- Planteamiento de la propuesta de solución.
- Control y verificación de los datos obtenidos mediante pruebas de detección y corrección de errores en el robot KUKA youBot.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

La implementación de uno de los algoritmos estudiados para el aprendizaje por refuerzo con redes neuronales profundas (DQN) en la plataforma robótica móvil del robot KUKA youBot instaura un desarrollo significativo en el estudio de la inteligencia artificial (IA) enfocada a la robótica de carácter industrial. Este proyecto optimiza el desplazamiento realizado por la plataforma de un punto a otro con la presencia o ausencia de obstáculos, sean estos móviles o estáticos. El entrenamiento realizado en la plataforma generó un modelo de aprendizaje adecuado que utiliza estrategias efectivas al momento de evadir obstáculos; para este modelo se desarrolló una aplicación, como una interfaz gráfica de sistema de control y monitoreo, que permite guiar al usuario en todos los pasos que se requieren para el entrenamiento de la plataforma.

3.2. Desarrollo de la propuesta

3.2.1. Requerimientos del sistema

La industria enfocada en diferentes ámbitos de procesos va en desarrollo continuo de manera exponencial, las empresas buscan optimizar los procesos llevados a cabo en las mismas, mejorando el tiempo que conlleva cada uno de ellos en su realización.

El transporte de equipos de un punto a otro, situados estratégicamente dentro de una empresa, busca mantener un estándar para el cumplimiento de procesos, donde un AGV lleva a cabo este proceso cumpliendo tiempos preestablecidos. Para optimizar estos tiempos, se utilizan técnicas de aprendizaje por refuerzo para que el robot busque la mejor ruta y pueda llegar a su destino de manera más rápida.

3.2.2. Esquema del sistema

En la Figura 14 se aprecia el esquema general del sistema dividido en etapas para la implementación del algoritmo de entrenamiento de refuerzo con redes neuronales tanto para plataforma del robot KUKA youBot simulado como para la plataforma real.

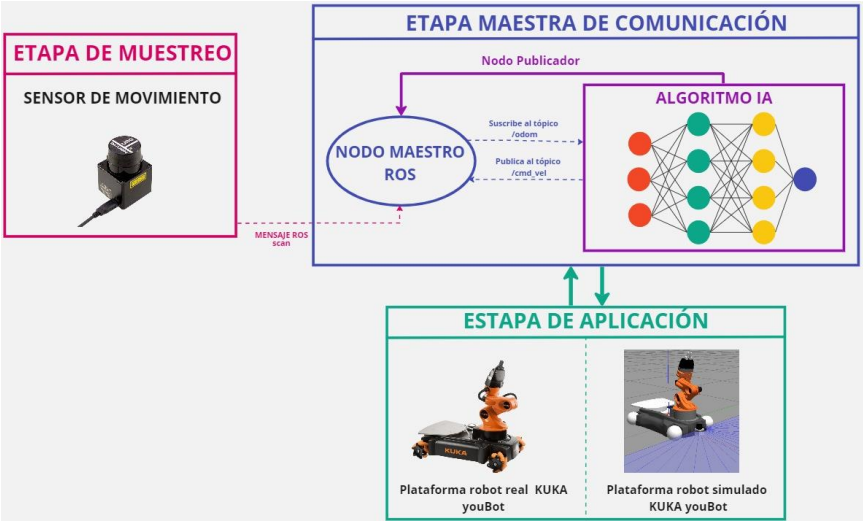


Figura 14. Esquema general del sistema.

Fuente: Investigador.

Etapa maestra de comunicación

En esta etapa se levanta el nodo maestro de ROS; en el robot real se ejecuta el archivo controlador interno de la PC incorporada y en el robot simulado como un archivo lanzador mostrando al robot en Gazebo con su controlador incorporado. El algoritmo realiza la función de nodo publicador, donde se suscribe el tópico de odometría (/odom) el cual almacena la estimación de la posición y publica el tópico de velocidad (/cmd_vel).

Etapa de muestreo

En esta etapa se realiza la adquisición de datos por medio del sensor lidar Hokuyo, este sensor es lanzado como un nodo suscriptor al nodo maestro de ROS, recopilando datos

de su entorno mediante muestreo y enviándolos a manera de mensaje de ROS al algoritmo.

Etapa de aplicación

En esta etapa se aplica los datos procesados por el algoritmo como valores hacia los tópicos correspondientes de la plataforma omnidireccional para la ejecución su movimiento, esto es llevado a cabo primero en la simulación para el entrenamiento del algoritmo y luego en el robot real con un modelo entrenado.

La secuencia de funcionamiento se muestra en la Figura 15. El usuario inicia el nodo maestro de ROS de manera principal en donde se suscribe el nodo del sensor lidar Hokuyo, los nodos levantados son verificados y de no existir errores se muestra al usuario. El sensor lidar capta datos muestreados del escenario de entrenamiento, los almacena en una matriz y los envía a manera de mensaje de ROS al algoritmo, este realiza la función de nodo publicador hacia el nodo maestro de ROS y levanta los movimientos respectivos en las 4 ruedas de la plataforma omnidireccional del robot KUKA youBot sea este simulado o real.

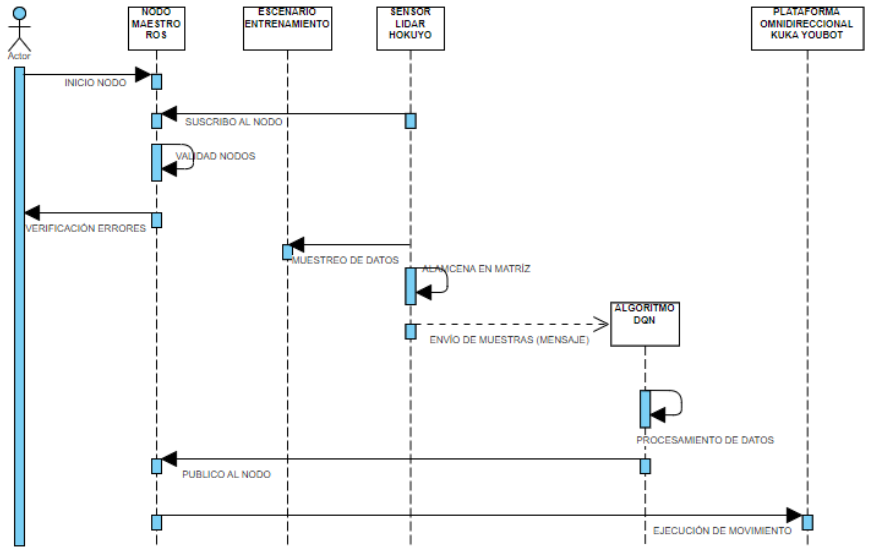


Figura 15. Diagrama de secuencia de funcionamiento general.

Fuente: Investigador.

3.2.3. Etapa maestra de comunicación

3.2.3.1. Selección del algoritmo de entrenamiento por refuerzo

De los algoritmos mostrados en la Tabla 2, los más aptos para trabajar con ROS en su versión Melodic o posteriores son: DQN, Double DQN y Dueling DQN. Sin embargo, el robot KUKA youBot trabaja con las versiones de ROS Groovy, Hydro, Indigo, Jade y Kinetic. Esta última versión de ROS posee soporte para los paquetes de Maching Learning del robot TurtleBot en su última versión 3 para fines educativos, esencialmente este robot es dedicado al estudio de la tecnología de localización y mapeo simultáneo de objetos (SLAM), Navegación y Manipulación, siendo en navegación donde este robot tiene entrenamientos realizados y probados con el algoritmo DQN en laboratorios.

Analizado los criterios anteriores y teniendo en cuenta compatibilidad completa de los paquetes de ROS Kinetic con el robot KUKA youBot se realiza la elección de este algoritmo (DQN) a ser implementado en la plataforma robótica móvil del robot, junto con los requerimientos esenciales de instalación para el aprendizaje de refuerzo, descritos en la Tabla 5.

Tabla 5. Requerimientos para implementación algoritmo DQN.

Requerimientos	**Versión**
Ubuntu	16.04 LTS
ROS	Kinetic
Gazebo	7.16.1
Python	2.7.15
Anaconda2	5.0.2
Tensorflow	1.8.0
Keras	2.1.5

Fuente: Investigador basado en [34].

Este algoritmo de aprendizaje por refuerzo DQN, consta de parámetros ajustables preestablecidos, conocidos como hiperparámetros que permiten controlar el proceso de entrenamiento del modelo. Las configuraciones de estos hiperparámetros son descritas en la Tabla 6.

Tabla 6. Hiperparametros algoritmo DQN.

Hiperparámetro	**Por defecto**	**Descripción**
episode_step	6000	El paso de tiempo de un episodio.
target_update	2000	La tasa de actualización de la red de destino.
discount_factor	0.99	Este factor representa cuánto pierden su valor los eventos futuros según la distancia.
learning_rate	0.00025	Es la velocidad de aprendizaje. Se considera que, si el valor es demasiado grande, el aprendizaje no funciona bien, y si es demasiado pequeño, el tiempo de aprendizaje es largo.
epsilon	1.0	Este valore representa la probabilidad de elegir una acción aleatoria.
epsilon_decay	0.99	Es la tasa de reducción de épsilon. Cuando termina un episodio, épsilon se reduce.
epsilon_min	0.05	Es el valor mínimo de épsilon considerado.
batch_size	64	Tamaño de un grupo de muestras de entrenamiento.
train_start	64	Este valor indica que se comience a entrenar si el tamaño de la memoria de reproducción es superior a 64.
memory	1000000	Es el tamaño de la memoria de reproducción.

Fuente: Investigador basado en [34].

3.2.3.2. Implementación del algoritmo de entrenamiento por refuerzo con redes neuronales DQN en el robot KUKA youBot

Para el entrenamiento del algoritmo DQN en el robot KUKA youBot tanto real como simulado, se ejecuta mediante un archivo lanzador o ".launch", este archivo llama al nodo publicador del algoritmo DQN según el número de escenario, este nodo a su vez llama al archivo de configuración ".py" de parámetros del nodo suscriptor del sensor lidar para la recopilación de muestras por medio de mensaje de ROS y realiza la publicación de los datos procesados en el nodo publicador DQN en el tópico "/cmd_vel" para la velocidad lineal y angular como la suscripción al tópico "/odom" de odometría para recopilar la información de la posición estimada del robot, esta recopilación de información de la posición estimada del robot es comparada a la información obtenida en base a la posición de la recompensa del archivo "respawnGoal", donde se configura la posición inicial y aleatoria de la recompensa.

En la Figura 16 se aprecia de mejor manera en el cuadro de referencia como se realiza la interacción del nodo publicador DQN con el archivo de configuración ".py" y el archivo de recompensa, a su vez como este cuadro de referencia interacciona con el nodo maestro de ROS y el nodo del sensor lidar Hokuyo, previamente lanzados.

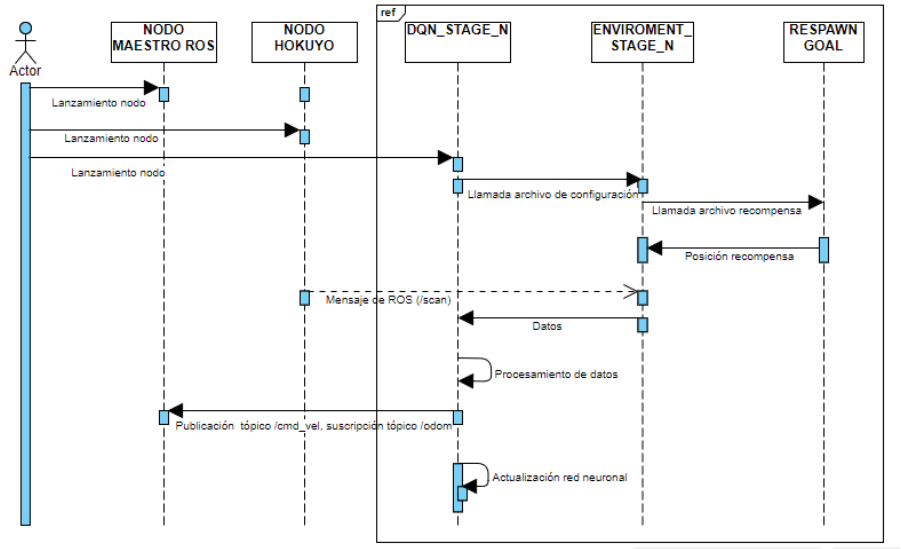


Figura 16. Diagrama de secuencia lanzamiento nodo publicador DQN

Fuente: Investigador.

Acoplamiento algoritmo DQN según parámetros robot KUKA youBot.

El lanzamiento del nodo publicador DQN se lo realiza mediante un archivo lanzador o ".launch" denominado "LAUNCH DQN_STAGE_N", donde se introduce el paquete y el número del escenario para el entrenamiento.

El paquete para el entrenamiento es el nodo denomina "DQN_STAGE_N" donde se almacena las muestras de datos proporcionados del sensor lidar Hokuyo, estas muestras deben ser reacondicionadas a las proporcionadas por sensor real para almacenarlas en la matriz requerida en este nodo. Cada 10 episodios de entrenamiento, se genera un modelo con la red neuronal entrenada, este modelo puede ser llamado en este nodo para ejecutar dicho entrenamiento o seguir entrenándose a partir del mismo.

El número del escenario para el entrenamiento es un archivo de configuración ".py" denominado "ENVIROMENT_STAGE_N". En este archivo la velocidad lineal y angular es calibrada, teniendo en cuenta la velocidad máxima y mínima dada por el fabricante (velocidad mínima 0.01m/s, velocidad máxima 0.8 m/s [13]), así mismo se calibra la distancia para la colisión y para la obtención de la recompensa.

Para la posición inicial de la recompensa se establece en archivo de configuración ".py" denominado "RESPWN GOAL", así mismo se configura el rango que cubrirá la recompensa por la superficie de entrenamiento del escenario de manera randomica.

Cada parte que conforma el lanzamiento del nodo publicador DQN cumple una dependencia directa ya que no se puede lanzar el nodo sin todas las partes que lo conforma como se muestran en la Figura 17.

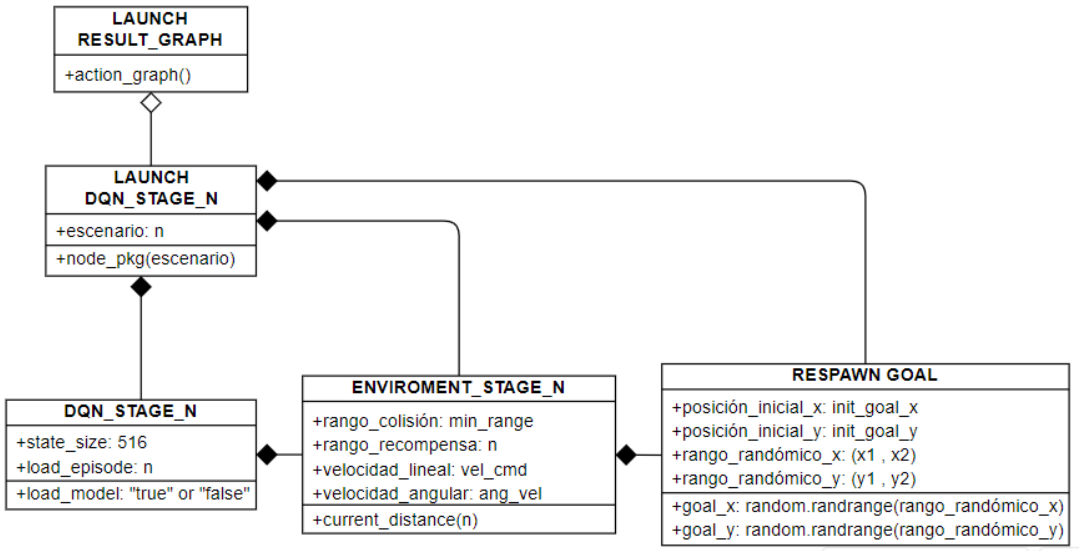


Figura 17. Configuración de parámetros para lanzamiento algoritmo DQN.

Fuente: Investigador.

El nodo para la obtención de resultados por medio de gráficas del estado de aprendizaje del robot denominado "LAUNCH RESULT_GRAPH" no tiene una dependencia directa con el nodo publicador DQN, este puede lanzarse independientemente del escenario de aprendizaje escogido.

3.2.4. Etapa de muestreo

3.2.4.1. Calibración del sensor lidar Hokuyo

El sensor lidar Hokuyo para el entrenamiento se configura con el número de muestras que utiliza el sensor real, 512 muestras, estas están distribuidas en grupo de datos de 64 muestras para entrenar el algoritmo DQN acoplándose a un escenario de entrenamiento real, es importante primero levantar muestras de datos del sensor real hacia el escenario de simulación, para lo cual se debe quitar el sensor del robot KUKA youBot simulado, como se especifica en la Figura 18, para que se tome las muestras del tópico del sensor real esto para calibrar las muestras arrojadas por el sensor en caso de existir variaciones.

Para el entrenamiento del algoritmo DQN se realiza en el entorno de simulación Gazebo, por lo cual conocidas la cantidad de muestras requeridas por el sensor real se configura el archivo ".xacro" del sensor de Gazebo a 512 muestras, posteriormente en el archivo del nodo publicador del escenario de entrenamiento elegido, se configura a 516 muestras, 4 muestras más a la cantidad de muestras requeridas por el sensor, este requerimiento es arrojado en los mensajes de errores al momento de lanzar el nodo de entrenamiento, este cambio puede variar de acorde a la necesidad de configuración mostrada por el nodo.

El siguiente diagrama de clase en la Figura 18, se detalla el número de muestras que se debe insertar tanto para el escenario de entrenamiento del algoritmo y el sensor lidar para llevar a cabo el entrenamiento en el entorno simulado, teniendo en cuenta primeramente quitar el sensor simulado del robot, tomar muestras desde el sensor real hacia la simulación para calibrar las muestras requeridas para el entrenamiento del algoritmo.

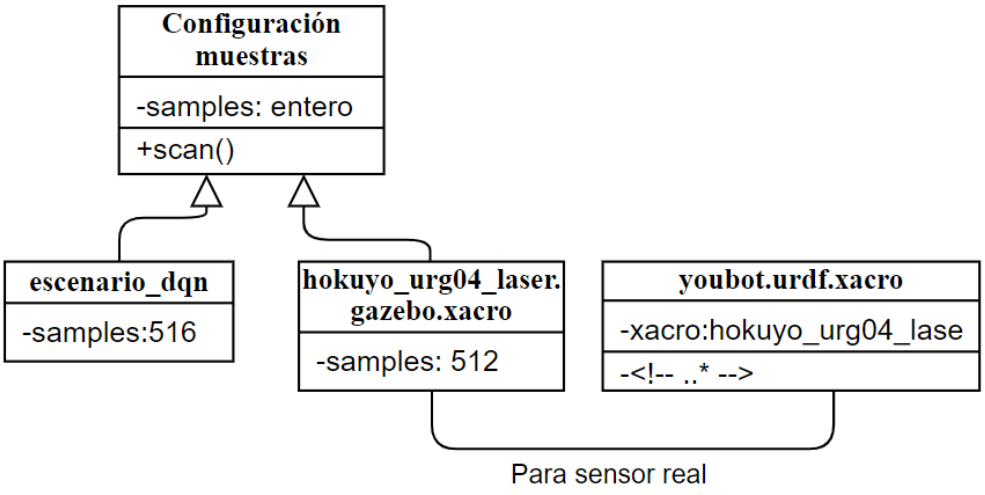


Figura 18. Diagrama de clase calibración del sensor lida Hokuyo.

Fuente: Investigador.

3.2.4.2. Esquema de conexión para el levantamiento del sensor lidar Hokuyo en el robot real

En el robot real KUKA youBot se configura su archivo ".bashrc" para que, dentro de un mismo segmento de red, cumpla la función de nodo maestro ROS y poder ejecutar el movimiento en la plataforma real.

Primero se configura el robot real como nodo maestro ROS y luego la laptop como una PC remota. Siendo en esta donde se ejecuta el algoritmo de redes neuronales y se conecta el sensor lidar Hokuyo por medio de USB 2.0 para poder levantar el nodo del sensor el cual se suscribe al nodo maestro, su funcionamiento se observa en el Anexo 02 donde al ejecutar Rviz se muestra el comportamiento del sensor.

En la Figura 19 se describe de mejor manera la conexión realizada para la ejecución de movimiento en la plataforma real con adquisición de muestras del escenario físico por medio del sensor.

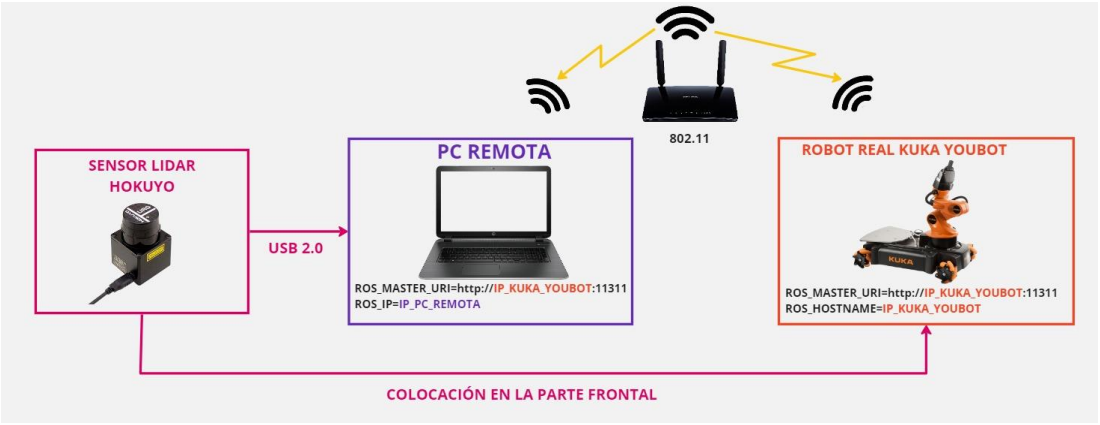


Figura 19. Esquema de conexión sensor lidar Hokuyo y KUKA YouBot real.

Fuente: Investigador.

3.2.4.3. Adquisición de muestras del sensor lidar Hokuyo URG-04LX-UG01

El sensor lidar Hokuyo, trabaja en conjunto con el escenario de entrenamiento creado en Gazebo, este es lanzado con el robot simulado KUKA youBot, aquí el sensor

captura muestras de datos del escenario de entrenamiento en el que se encuentra, estos datos son enviados por medio de mensaje de ROS al algoritmo a través del nodo maestro ROS, estos datos son procesados en el algoritmo principalmente para colisiones ya que el punto de recompensa trabaja únicamente como una ubicación de llegada predeterminada. Procesados estos datos en el algoritmo son enviados, a manera de nodo publicador, a la plataforma del robot para que esta haya evitado colisionar.

En el diagrama de la Figura 20 se observa 4 bloques, donde partiendo del escenario creado, el sensor capta las muestras de este, las envía al bloque de componentes central hacia el algoritmo por medio de la comunicación que permite el nodo maestro ROS y finalmente estos datos procesados en el algoritmo los envía para la ejecución del movimiento en la plataforma.

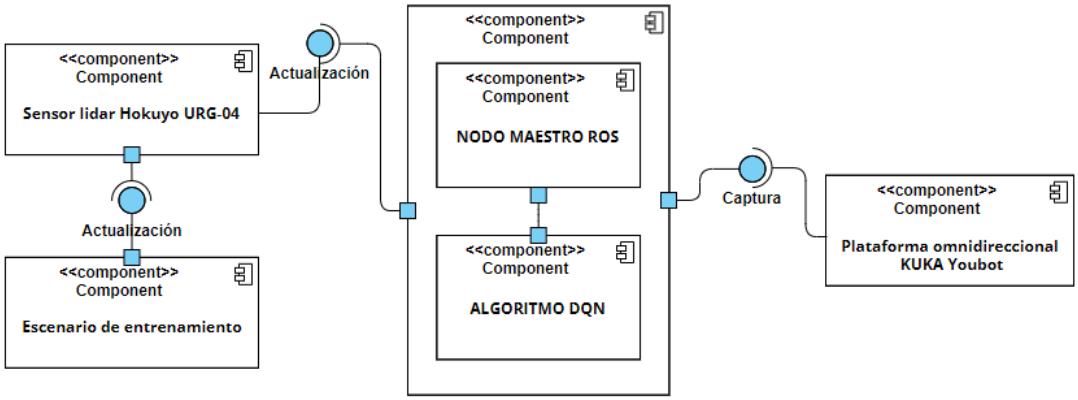


Figura 20. Diagrama de conexión de los componentes del sistema.

Fuente: Investigador.

En este apartado el sensor lidar Hokuyo trabaja bajo el tópico de ROS "/scan", esto tanto para la simulación como para el sensor real, ya que por defecto el sensor lidar del robot KUKA youBot viene configurado el tópico de ROS "/base_scan".

3.2.5. Etapa de aplicación

Las fases para el entrenamiento del robot KUKA youBot se realizan gracias a la adaptación de ciertos escenarios existentes que vienen incluidos en los paquetes de instalación del algoritmo DQN.

3.2.5.1. Adaptación de escenario para fase uno de entrenamiento

Para esta fase, se elige el escenario cuyo mundo no consta de obstáculos y su modelo para el entrenamiento es amplio, las medidas de este modelo se muestran en la Figura 21 y en su archivo lanzador se introduce al modelo del robot KUKA youBot bajo el nombre "youbot", como se muestra en la Figura 23.

En la salida de ejecución del archivo lanzador, en su simulación en Gazebo, se mostrará como en la Figura 21, donde el comportamiento del sensor mostrado de manera azul abarcando toda la zona al no existir obstáculos y solo es delimitado por el área cerrada del escenario.

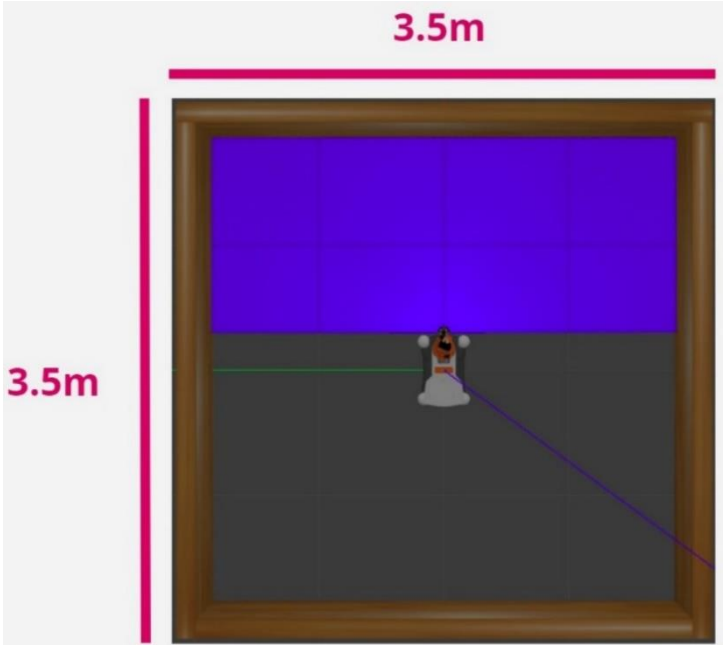


Figura 21. Lanzamiento del escenario para fase 1 de entrenamiento.

Fuente: Investigador.

3.2.5.2. Creación de escenario para fase dos de entrenamiento

Para realizar el entrenamiento tanto para el robot real KUKA youBot como para la simulación, se utiliza el entorno de Gazebo, donde se crea un modelo para el escenario de entrenamiento, cuyas medidas son replicadas en un escenario real, este modelo corresponde a las paredes que conforman el área cerrada para la colisión de 2.5m x 2.7m como se muestra en la Figura 24, una vez creado el modelo, el archivo será ".sdf", este archivo es almacenado en la carpeta de simulación de modelos de Gazebo. En la carpeta generada del modelo creado se introduce el modelo de recompensa "goal_box" para llamar en un solo conjunto a estos dos modelos, en la Figura 22 se muestra la secuencia para la creación del archivo sdf de este modelo.

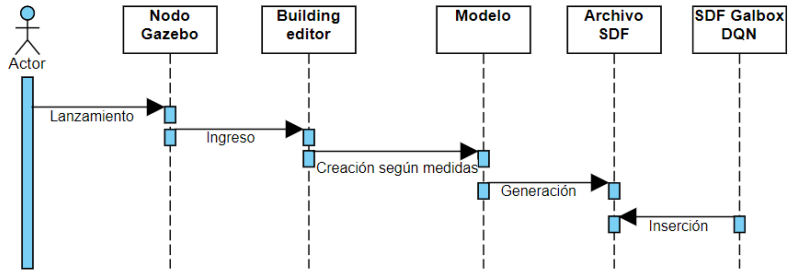


Figura 22. Diagrama de secuencia creación modelo del escenario.

Fuente: Investigador.

El modelo creado se incorpora en el archivo ".world" del escenario de entrenamiento escogido, como se muestra en la Figura 23. Los paquetes del algoritmo DQN constan de 4 escenarios previamente configurados, en este paso lo que se pretende es reacondicionar un escenario de entre los 4, para acoplarse a las medidas de un escenario real, así mismo sus obstáculos.

La elección para el reacondicionamiento del escenario, de entre los 4 existentes, se realiza de manera acertada, teniendo en cuenta los obstáculos que están ubicados en el mundo de los escenarios y configurados previamente.

La ejecución del robot simulado KUKA youBot se lleva a cabo bajo creación de un archivo lanzador ".launch", este archivo llama al mundo reacondicionado del

escenario escogido ".world" y al modelo del robot bajo el nombre "youbot", como se muestra en la Figura 23. Se puede en el archivo lanzador configurar la posición inicial del robot en cualquiera de los ejes x, y o z, teniendo en cuenta las medidas del modelo creado.

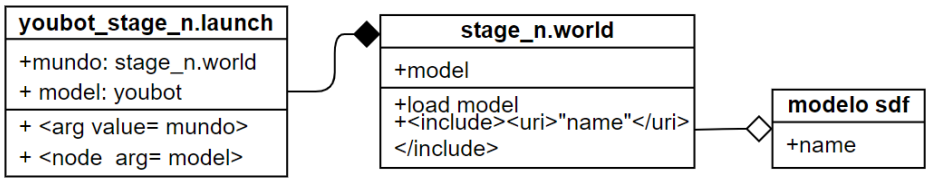


Figura 23. Diagrama de clase lanzamiento escenario creado.

Fuente: Investigador.

En la salida de ejecución del archivo lanzador, en su simulación en Gazebo, se mostrará como en la Figura 24, donde el comportamiento del sensor mostrado de manera azul abarca toda la zona cercana a su alrededor, detectando los dos obstáculos de color blanco ubicados tanto la parte frontal y lateral izquierda del escenario, tomando en cuenta la posición inicial del robot. Los espacios conservados en color gris representan las zonas huecas donde el sensor no llega a detectar.

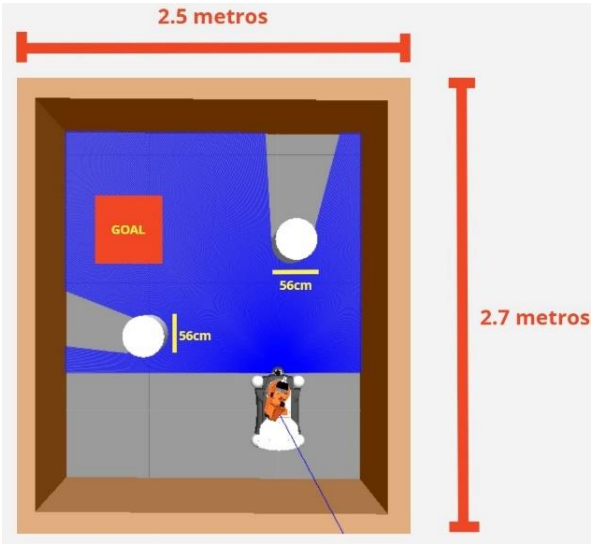


Figura 24. Lanzamiento del escenario para fase 2 de entrenamiento.

Fuente: Investigador.

Las medidas utilizadas en la creación del modelo sdf son replicadas en un entorno real juntamente con el robot real KUKA youBot como se muestra en la Figura 25.

Figura 25. Recreación del mundo del escenario simulado en un entorno real.

Fuente: Investigador.

3.2.5.3. Adaptación de escenario para fase tres de entrenamiento

Se reacondiciona otro mundo de los escenarios existentes que cuenta de obstáculos que se mantendrán en constante movimiento desplazándose de manera circular, en contra de las manecillas del reloj alrededor del robot. De igual manera en su archivo lanzador se introduce al modelo del robot KUKA youBot bajo el nombre "youbot", como se muestra en la Figura 23.

La variación de movimiento en los obstáculos es ejecutada por un archivo adicional a la configuración de parámetros para lanzamiento del nodo publicador de algoritmo

DQN mostrados en la Figura 17. Este archivo se aprecia en el diagrama de clase de la Figura 26.

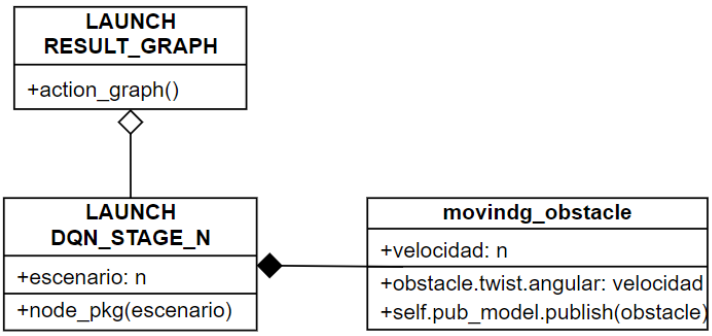


Figura 26. Configuración de parámetros para lanzamiento algoritmo DQN con obstáculos móviles.

Fuente: Investigador.

En Figura 27 se muestra la ejecución del archivo lanzador en su simulación en Gazebo.

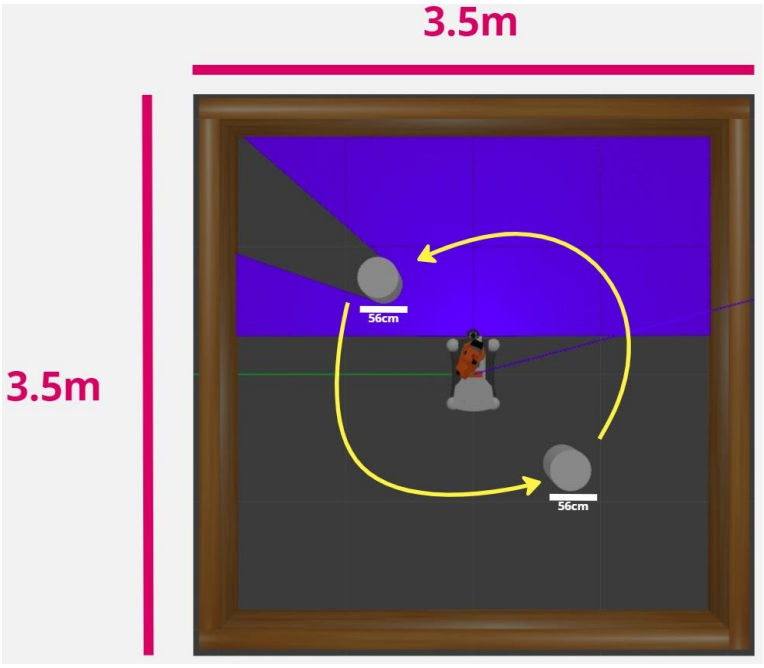


Figura 27. Lanzamiento del escenario para fase 3 de entrenamiento.

Fuente: Investigador.

3.2.5.4. Robot KUKA youBot junto con el sensor lidar Hokuyo

El modelo del robot KUKA youBot ya existe juntamente con los paquetes adicionales para su simulación dentro del entorno Gazebo como se muestra en la Figura 28, este modelo se toma como base para la implementación del sensor Hokuyo URG-04LX-UG01 en el robot real, ubicándolo en la parte frontal de la plataforma del robot.

El lanzamiento del robot simulado KUKA youBot puede realizarse independientemente de cada parte que lo conforma, por defecto se puede lanzar el robot entero, el brazo o la base. Para quitar o agregar componentes como por ejemplo el sensor lidar Hokuyo, se editar el archivo ".urdf.xacro" del robot completo, donde se llama cada parte por separado para completar el lanzamiento de todo el robot en conjunto.

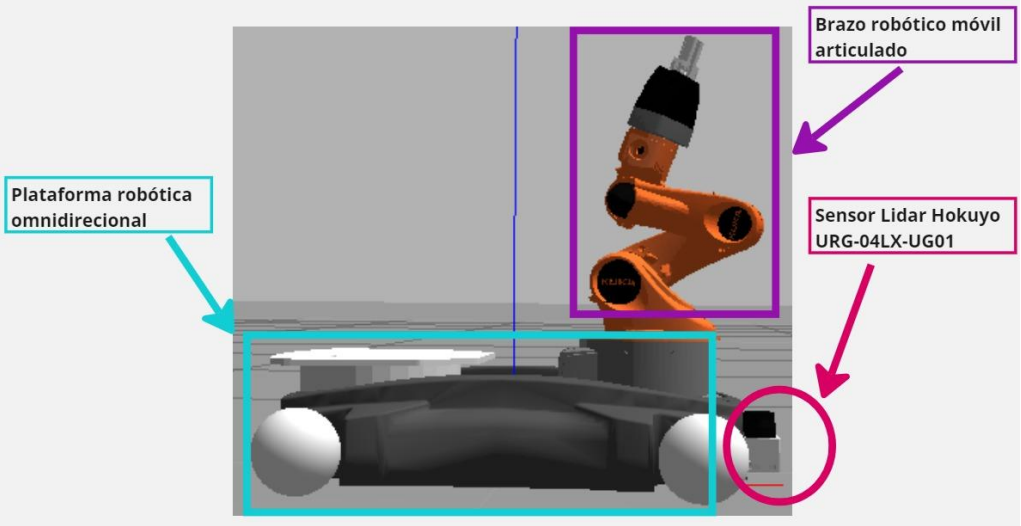


Figura 28. Robot KUKA youBot simulado con sensor lidar Hokuyo.

Fuente: Investigador.

Para la incorporación del sensor Hokuyo en el robot real, se debe colocar de igual manera en la parte frontal de la plataforma omnidireccional, como el sensor lidar es conectado a una laptop por medio de USB 2.0 para la adquisición de las muestras, esta laptop es colocada en un espacio encima de la plataforma teniendo cuidado de que el

cable que va conectado al sensor no se enrede en alguna de sus ruedas, mostrándose en la Figura 29.

Figura 29. Robot KUKA youBot real simulado con sensor lidar Hokuyo.

Fuente: Investigador.

3.2.5.5. Diagramas de funcionamiento del sistema por nodos

El lanzamiento de todos los procesos que se involucran para iniciar el sistema completo de entrenamiento de la plataforma móvil del robot, son registrados en ROS por medio de un gráfico de cálculo conocido como "rqt_graph", este gráfico muestra como el sistema trabaja bajo una arquitectura de nodos y tópicos.

Diagrama rqt_graph de la simulación de sistema para entrenamiento del robot

En el diagrama de nodos rqt_graph de la Figura 30, el nodo publicador DQN administra los datos recibidos del nodo maestro ROS en este caso /gazebo para el entrenamiento, a través de los tópicos suscritos de: /scan para las muestras obtenidas del escenario por medio de mensaje de ROS y /odom para la estimación de la posición del robot. El nodo DQN a su vez publica en el nodo maestro ROS (/gazebo) la velocidad lineal y angular para el movimiento de la plataforma por medio del tópico /cmd_vel y de manera instantánea el nodo /gazebo publica esta velocidad en las

articulaciones (4 omniruedas) de la plataforma móvil en el nodo robot_state_publisher a través del tópico /joint_states.

Completadas las acciones anteriores, para llevar a cabo el entrenamiento de manera continua, el nodo /gazebo publica en el nodo DQN la posición inicial del robot, una vez que este haya colisionado, a través del tópico /gazebo/model_states.

Los datos a medida que avanza el aprendizaje son graficados mediante la publicación del nodo DQN hacia los nodos: /action_graph para el comportamiento que tiene el robot en ese instante, esto a través del tópico /get_action y /result_graph para las recompensas obtenidas y la política de aprendizaje del robot, a través del tópico /result.

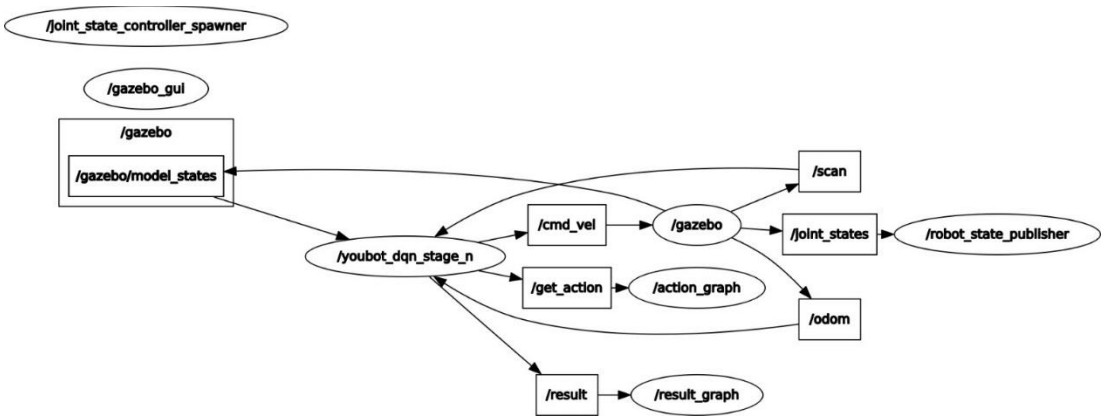


Figura 30. Diagrama rqt_graph de la simulación de sistema.

Fuente: Investigador.

Diagrama rqt_graph del sistema llevado a cabo en el robot real KUKA youBot

El proceso descrito en el diagrama de la Figura 30 se lleva a cabo de manera similar en el diagrama rqt_graph de la Figura 31 con la variación de la existencia del nodo del sensor real "/hokuyo_node" que publica, por medio de mensaje de ROS, las muestras obtenidas del escenario real hacia el nodo DQN a través del tópico /scan y del nodo maestro ROS levantado para el control del robot real "/youbot_driver" (n__youbot_driver) que publica en el nodo robot_state_publisher, a través del tópico

/joint_states, la velocidad en las articulaciones (4 omniruedas) de la plataforma móvil, esta velocidad es obtenida del nodo DQN a través del tópico /cmd_vel.

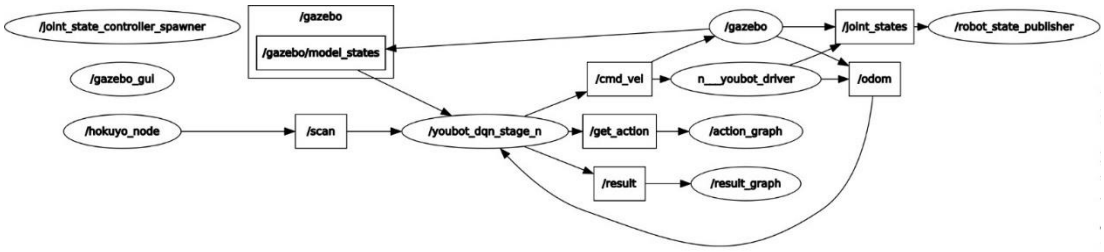


Figura 31. Diagrama rqt_graph del sistema llevado a cabo en el robot real KUKA youBot.

Fuente: Investigador.

3.2.5.6. Diseño de la interfaz gráfica para el control del sistema

Para llevar a cabo el lanzamiento de los archivos requeridos del entrenamiento del algoritmo DQN se presenta una interfaz gráfica para el control realizada en lenguaje programación Python 2; el diseño de su menú principal se muestra en la Figura 32.

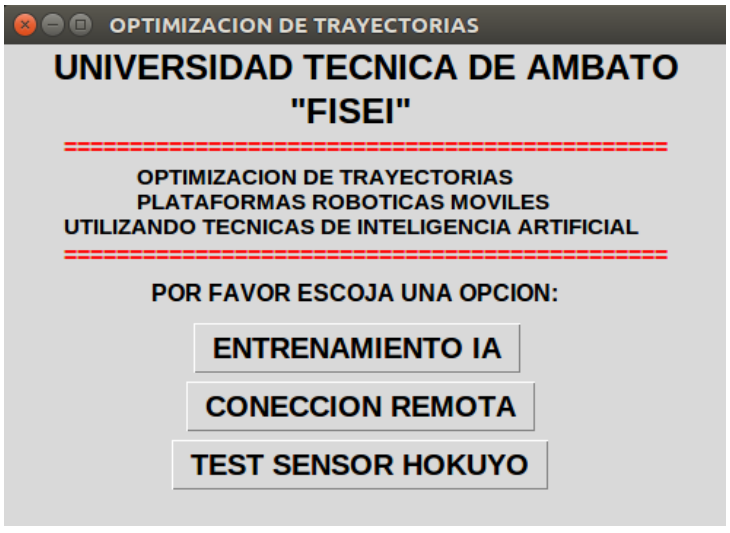


Figura 32. Diseño menú principal interfaz gráfica de control.

Fuente: Investigador.

Cada categoría que conforma el menú principal es descrita a continuación:

Categoría entrenamiento IA

Al elegir la primera opción del menú principal que se muestra en la Figura 32, se despliega un menú de ejecución donde al elegir la primera opción se lanza el escenario de entrenamiento en Gazebo del robot KUKA youBot,

En la opción del lanzamiento del algoritmo de aprendizaje inicial, se realiza el aprendizaje desde cero, es decir el robot deberá aprender estrategias para evitar los obstáculos. En la opción del lanzamiento del algoritmo de aprendizaje optimizado se almacena un modelo de aprendizaje entrenado previamente, donde el robot ya aprendió estrategias efectivas para evitar obstáculos. El diseño de esta interfaz gráfica se muestra en la Figura 33.

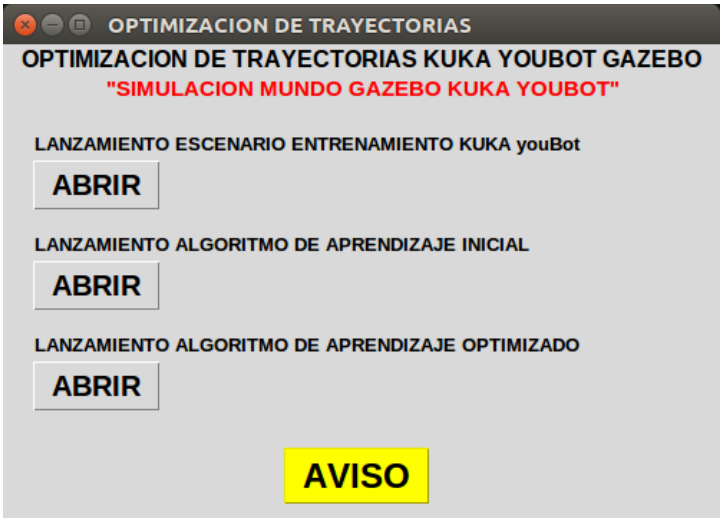


Figura 33. Diseño interfaz categoría entrenamiento IA.

Fuente: Investigador.

Es importante tener en cuenta la opción de AVISO de esta interfaz, ya que al momento de ejecutarla se muestran dos textos informativos. El primer texto indicando que en caso de no abrirse o ejecutarse el mundo del escenario de Gazebo se debe cerrar este

y actualizar el archivo de configuración ".bash", introduciendo en una terminal el comando "source .bashrc". El segundo texto indicando que el programa debe ejecutarse en terminales independientes, como se muestra en la Figura 34.

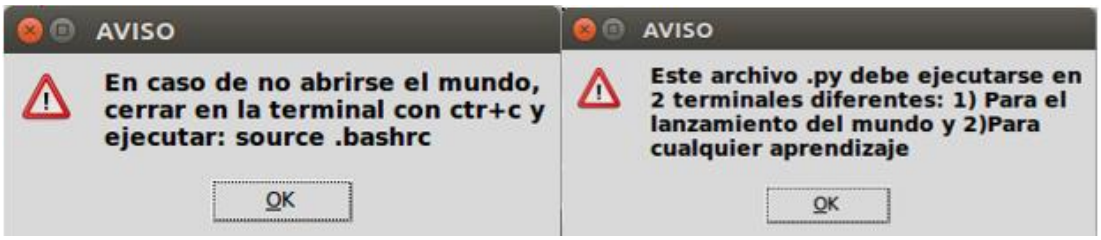


Figura 34. Opción de AVISO menú categoría entrenamiento IA.

Fuente: Investigador.

El lanzamiento de estos archivos principalmente se los debe realiza desde la terminal, en el diagrama de clase de la Figura 37 se muestra de mejor manera como son llamados estos archivos para el lanzamiento de cada uno independientemente.

Categoría conexión remota

En esta categoría se escriben los parámetros de conexión en el archivo ".bashrc" para la ejecución del movimiento en la plataforma del robot real KUKA youBot. Esta configuración se realiza en la PC incorporada del robot real para que cumpla la función de nodo maestro ROS y en la PC remota utilizada, en este caso laptop, para que realice la función de nodo suscriptor hacia el nodo maestro. El diseño de la interfaz gráfica es mostrado en la Figura 35.

Al ejecutar la opción de configuración de PC remota aparecerá una ventana donde se introducirá en el cuadro de texto las ips para: ROS MASTER que es la ip del robot real y la de ROS SUSCRIPTOR que es la ip propiamente de la PC remota, en este caso de la laptop que se utiliza. En la opción de configuración del robot real se introduce en los dos cuadros de texto la ip propiamente del robot real, este programa debe ejecutarse en la PC incorporada del robot KUKA youBot.

Figura 35. Diseño interfaz categoría conexión remota.

Fuente: Investigador.

El ingreso de las ips en cada configuración es mostrado en el diagrama de clase de la Figura 36 donde se describe por numeración que ip ingresar al momento de configurar esta conexión remota.

Categoría datos hokuyo

En esta categoría se muestra una interfaz para el monitoreo del funcionamiento del sensor lidar Hokuyo, donde los datos muestreados son graficados en tiempo real de tres maneras: datos adquiridos en la posición central, en la posición izquierda y en la posición derecha. La posición de cada muestra tendrá variaciones muy pequeñas ya que estas muestras adquiridas tienen variaciones decimales. El diseño de la interfaz gráfica es mostrado en la Figura 36.

Esta interfaz cuenta de monitorea cuando con opciones para graficar los datos adquiridos como también pausar y reanudar la adquisición de muestras.

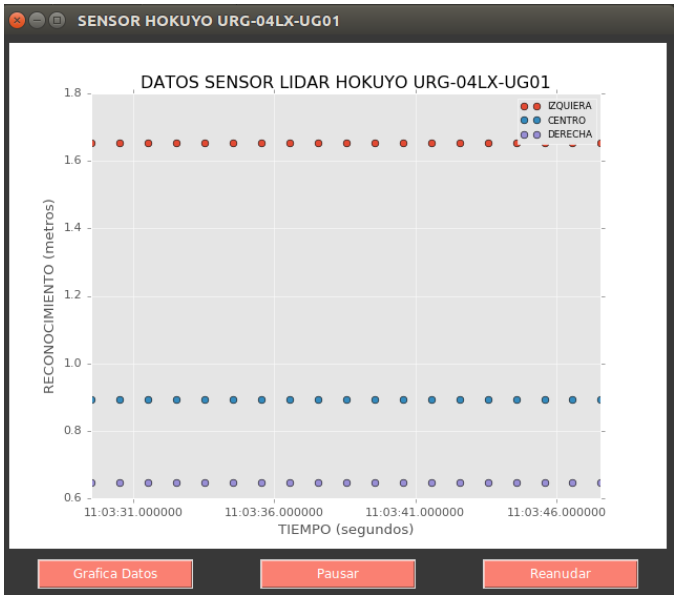


Figura 36. Diseño interfaz de monitoreo datos hokuyo.

Fuente: Investigador.

Como se muestra en la Figura 37, estos datos son extraídos por medio de mensaje de ROS del tópico "/scan" y graficados teniendo en cuenta la actualización del eje "x" en tiempo real.

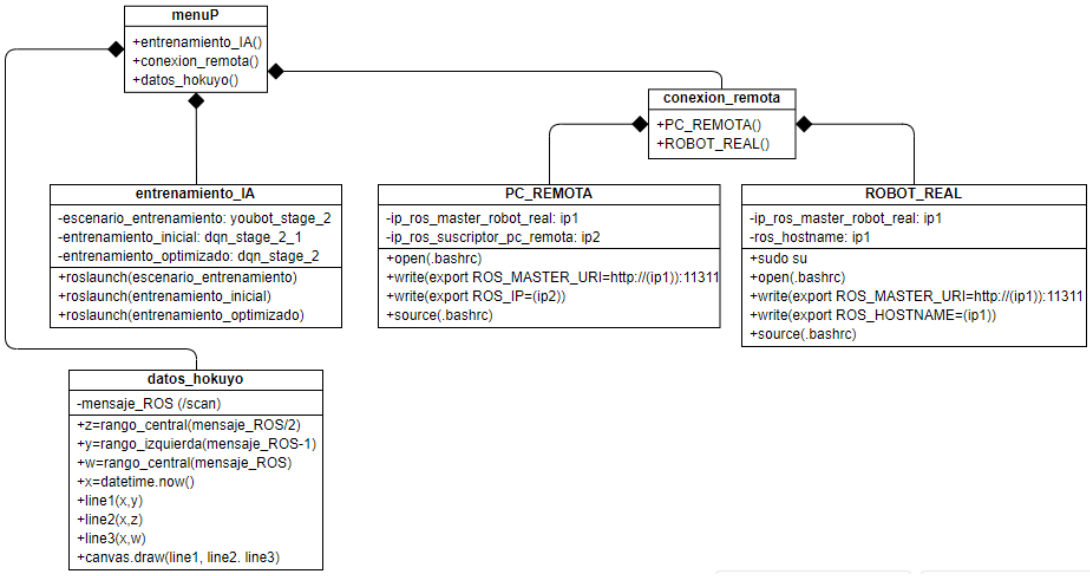


Figura 37. Diagrama de clase interfaz gráfica control del sistema.

Fuente: Investigador.

3.2.5.7. Usabilidad de la interfaz gráfica para el control del sistema

La usabilidad de la interfaz gráfica para el control del sistema es consideraba para cada categoría en las cuales esta divida la interfaz, describiéndose para cada categoría:

Usabilidad categoría entrenamiento IA

El archivo de Python programado en esta categoría se ejecuta 2 veces como si se tratara de la ejecución manual en terminales diferentes ya que para el lanzamiento del escenario de entrenamiento en Gazebo como para el lanzamiento del algoritmo de aprendizaje, se deben ejecutar de manera independiente. El usuario al momento ejecutar el aprendizaje sea este inicial u optimizado solamente deberá escoger uno.

Usabilidad categoría conexión remota

La ejecución del archivo de esta categoría se debe realiza independientemente del caso, sea en el robot real para configurarlo como nodo maestro y en la PC remota para configurarla como nodo suscriptor, esta interfaz se utiliza como una guía para realizar esta conexión ya que esa se debe realizar preferentemente de manera manual en el archivo ".bashrc" y una vez realizadas estas configuraciones, refrescarlas mediante el comando "source .bashrc" directamente en la terminal.

Usabilidad categoría datos hokuyo

Esta categoría es usada al momento de realizar el lanzamiento del sensor lidar Hokuyo preferentemente el sensor real, ya que por medio de la gráfica mostrada en esta interfaz se puede visualizar si los datos muestreados están siendo adquiridos o no.

El diagrama de usabilidad para cada categoría se muestra en la Figura 38.

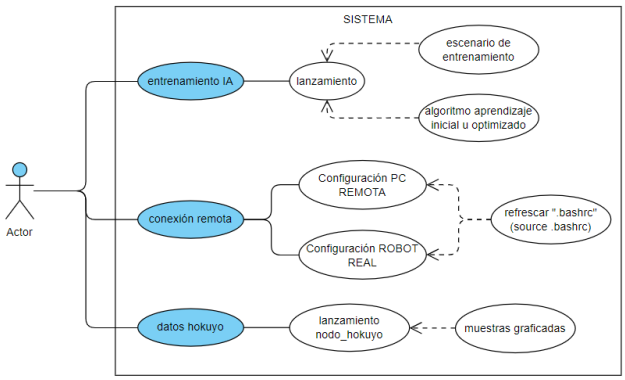


Figura 38. Diagrama de casos de uso de la interfaz gráfica para el control del sistema.

Fuente: Investigador.

3.2.6. Pruebas de funcionamiento

3.2.6.1. Proceso de entrenamiento del algoritmo DQN en la plataforma móvil KUKA youBot

El algoritmo para el entrenamiento por refuerzo de redes neuronales DQN trabaja bajo el aprendizaje especialmente por colisión, a medida que el entrenamiento del robot avanza la posibilidad de colisionar disminuye, aumentando el puntaje positivo recibido en el aprendizaje. El robot al llegar hacia el punto de recompensa denominado goalbox, recibe un puntaje mayor, dependiendo de la asignación, al puntaje de aprendizaje por estado que se obtiene al no colisionar.

El número de episodios de entrenamiento puede ser modificado dentro de la programación del algoritmo, cuando se desea realizar el entrenamiento desde 0 el número de episodios del que parte es 0 y al que termina es un episodio menor al ingresado, es decir si se ingresa 300 episodios el entrenamiento termina en el episodio 299 siendo el último episodio enviado para la actualización de la red neuronal.

La generación de los modelos de entrenamiento escritos cada 10 episodios de entrenamiento, la salida de los modelos son almacenados en formatos de archivos ".h5" y ".json", una vez generados estos modelos se pueden volver a cargar en el

algoritmo según el usuario considere en donde se produjo el mejor aprendizaje posible. Este proceso se muestra en el diagrama de flujo de la Figura 38.

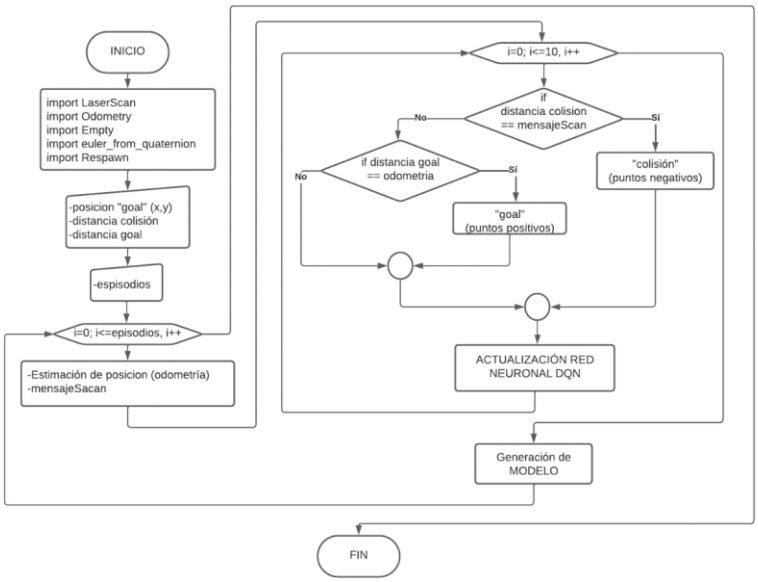


Figura 39. Proceso de entrenamiento del algoritmo DQN.

Fuente: Investigador.

3.2.6.2. Experimentos de fases de entrenamiento

Para verificar la efectivad del algoritmo DQN en la plataforma omnidireccional del robot KUKA youBot se consideran 3 fases de entrenamiento desarrolladas en diferentes escenarios:

- La fase 1 de entrenamiento consta de un escenario donde no existen obstáculos y el robot va desenvolviéndose únicamente por la recolección de muestras y colisionando hacia las paredes del escenario.
- La fase 2 de entrenamiento consta de un escenario con la existencia obstáculos estáticos, las medidas de este escenario mostradas en la figura 25 se acondicionaron para ser replicadas en un escenario real para implementar el modelo entrenado en la plataforma del robot real KUKA youBot.

- La fase 3 de entrenamiento consta de un escenario con obstáculos móviles que se desplazan en contra del sentido de las manecillas del reloj, en esta fase se pretende replicar un escenario donde existan personas en movimiento alrededor del robot y como este reacciona frente a ellas.

Estos tres escenarios son descritos como fases de entrenamiento y mostradas en la Figura 40.

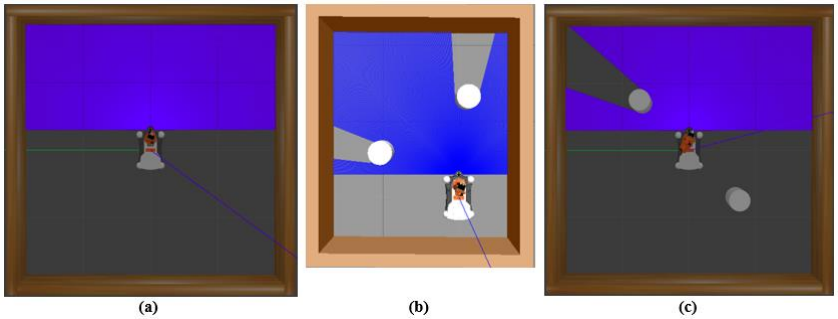


Figura 40. Fases de entrenamientos: (a) fase 1; (b) fase 2; (c) fase 3.

Fuente: Investigador.

Los experimentos se realizaron en una máquina virtual instalando los requerimientos descritos en la Tabla 5, sin gráfica integrada y con la utilización de un procesador Intel(R) Core (TM) i5-10210U CPU, asignándose a la máquina virtual 6 núcleos del procesador, 8 GB de RAM y 100 GB de disco.

El entrenamiento del robot en el entorno de simulación muestra los siguientes resultados: Fase 1 sin obstáculos, duración de 11 horas 3 minutos (162 episodios) el agente aprende una política efectiva a partir del episodio 100; Fase 2 con obstáculos estáticos, duración de 7 horas 32 minutos (733 episodios) el agente aprende una política efectiva a partir del episodio 500, Fase 3 con obstáculos móviles, duración 5 horas 47 minutos y 318 episodios el agente aprende una política efectiva a partir del episodio 150, estos resultados obtenidos se observan en el Anexo 01.

Las recompensas totales obtenidas por cada fase de entrenamiento del robot KUKA youBot son mostradas en la Figura 41, donde los valores negativos son estrategias no efectivas aprendidas mientras que los valores positivos son estrategias efectivas aprendidas por el robot, teniendo en cuenta que cada puntuación obtenida representa el nivel de desenvolvimiento por estado que tuvo el robot para aprender cada estrategia.

Las gráficas en su eje de coordenadas "y" muestra los valores máximos y mínimos obtenidos por las estrategias utilizadas por el robot, mientras que el eje de coordenadas "x" el número de episodios transcurrido por cada fase de entrenamiento. Cada colisión representa un episodio de entrenamiento.

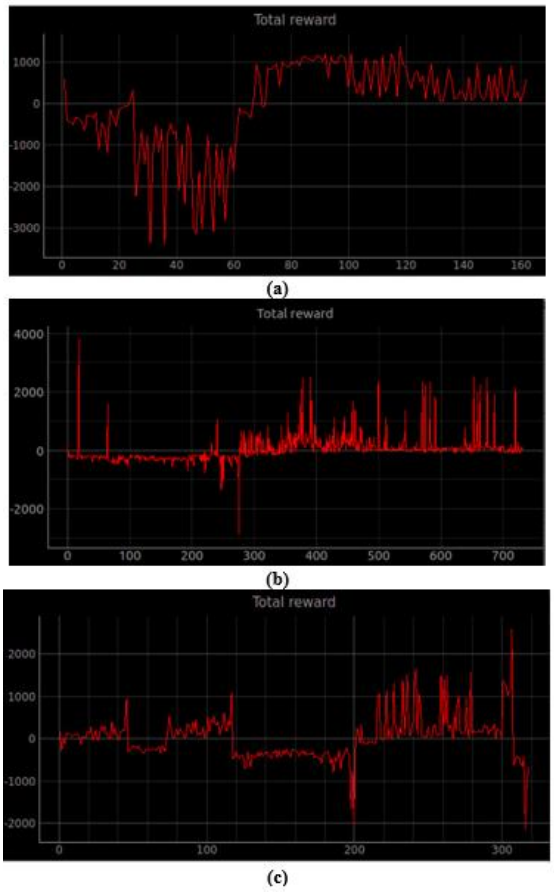


Figura 41. Recompensas totales obtenidas por fases de entrenamiento: (a) fase 1; (b) fase 2; (c) fase 3.

Fuente: Investigador.

La curva promedio de los valores Q máximos del algoritmo DQN por cada fase de entrenamiento se muestran en la Figura 41, donde los valores graficados de Q representan la política de aprendizaje. Los valores positivos graficados y su constante crecimiento muestran el episodio a partir del cual el robot empieza a tener una política efectiva de aprendizaje, mientras que los valores negativos representan que el robot aún no ha empezado a tener una política efectiva de aprendizaje.

Las gráficas en su eje de coordenadas "y" muestra los valores máximos y mínimos Q obtenidos del algoritmo DQN, mientras que el eje de coordenadas "x" el número de episodios transcurrido por cada fase de entrenamiento.

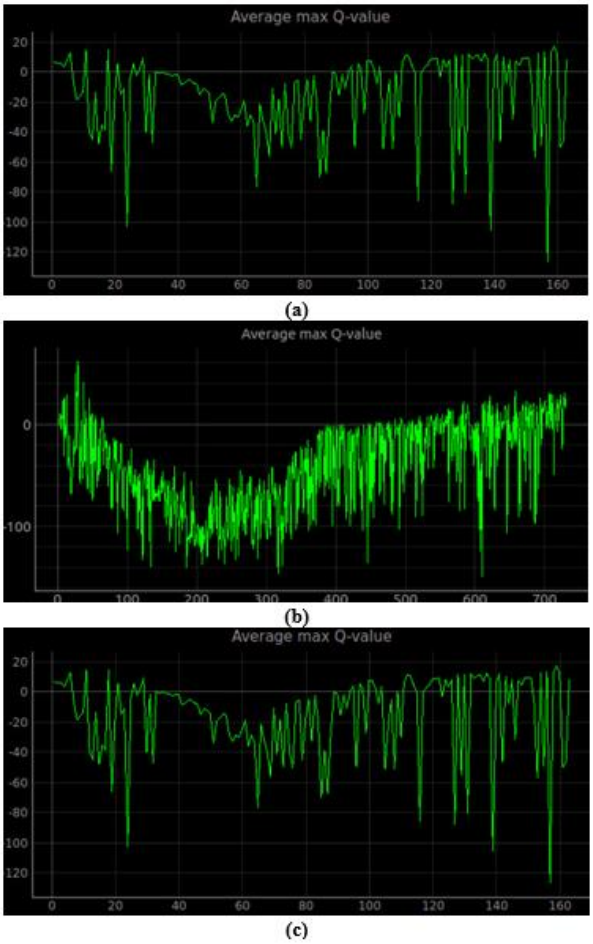


Figura 42. Curva promedio de los valores Q máximos por fases de entrenamiento: (a) fase 1, (b) fase 2; (c) fase 3.

Fuente: Investigador.

3.2.7. Presupuesto

El robot KUKA youBot y el sensor lidar Hokuyo al ser propiedad de la Universidad, no se toman en consideración para el presupuesto.

El presupuesto es detallo en la Tabla 7, donde se especifican los recursos económicos empleados para el desarrollo del estudio.

Tabla 7. Presupuesto de recursos utilizados.

Ítem	**Recursos Económicos**	**Cantidad**	**P. Unit. (USD)**	**P. Total (USD)**
1	Útiles de oficina e impresiones	1	$30,00	$30,00
2	Teclado	1	$12,00	$12,00
3	Mouse	1	$8,00	$8,00
4	Memory Flash	1	$14	$14,00
5	Material para recreación del escenario	9	$0.50	$4.50
6	Horas empleadas en la investigación (ver Anexo 4)	768	(Ver Anexo 4)	$13155
	Subtotal	-	-	$13223,50
	Imprevistos (10%)	-	-	$1322,35
	TOTAL	**-**	**-**	**$14545,85**

Fuente: Investigador.

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Los algoritmos más aptos para trabajar con inteligencia artificial y ROS son: DQN, Double DQN y Dueling DQN. Sin embargo, DQN es el algoritmo seleccionado por la compatibilidad en todos los paquetes de instalación para el correcto funcionamiento del robot KUKA youBot con IA.
- El entrenamiento del algoritmo DQN en la plataforma robótica móvil se realiza en un entorno simulado en tres fases de aplicación: sin obstáculos, con obstáculos estáticos y con obstáculos móviles. Siendo la fase 3 con un escenario con mayor complejidad donde el agente aprende de entrenamiento con obstáculos móviles la que mejor se comporta.
- La interfaz gráfica del sistema consta de tres secciones para el control del levantamiento de procesos involucrados en el entrenamiento de la plataforma móvil del robot KUKA youBot. La sección de monitoreo adquiere los datos del sensor lidar por medio de mensaje de ROS y la gráfica en tiempo real.

4.2. Recomendaciones

- Los parámetros de configuración de la velocidad en la plataforma real omnidireccional del robot KUKA youBot deben ser calibrados de acorde a las especificaciones del fabricante, si se utilizan niveles mayores a los indicados, la plataforma adopta un estado de protección de sus motores, apagándolos de manera inmediata.
- Para la asociación de un modelo o mundo creado en Gazebo hacia un espacio de trabajo se debe actualizar el enlace creado anteriormente “CMakeLists.txt” para ello se ejecuta el comando “catkin_make” dentro de este espacio, así mismo se podrá verificar la existencia de errores.

- El lanzamiento de los procesos especialmente de la simulación para el entrenamiento, así mismo al momento de configurar los parámetros para la conexión “PC remota” tanto en la laptop utilizada como en la PC incorporada en el robot real, en ocasiones no llegan a levantase completamente para ello se debe actualizar las veces que sea necesaria el archivo bashrc insertando el comando “source .bashrc” en la terminal.

BIBLIOGRAFÍA

[1] Equipo Ilabo, «ilabo,» ilabo, 23 11 2022. [En línea]. Available: https:// ilabo.com/blog/growth-of-automation-in-manufacturing/. [Último acceso: 2023 02 03].

[2] H. Pascual, «e-goi,» e-goi, 26 03 2019. [En línea]. Available: https://blog.e-goi.com /benefits-of-automating-processes/. [Último acceso: 03 02 2023].

[3] M. Intelligence, «Mordor Intelligence,» Mordor Intelligence, 2020. [En línea]. Available: https://www.mordorintelligence.com/es/industry-reports/latin-america-automated-guided-vehicles-market. [Último acceso: 03 02 2023].

[4] G. Coba, «Primicias,» Primicias, 05 05 2021. [En línea]. Available: https:// www.primicias.ec/noticias/economia/empleo-ecuador-reemplazo-robots-automatizacion/.
[Último acceso: 03 02 2023].

[5] E. S. B. Guamani, «Navegación autónoma basada en maniobras bajo estimación de posturas humanas para un robot omnidireccional KUKA YouBot,» Universidad Técnica de Ambato, Ambato, 2019.

[6] V. Y. H. Márquez, «Generación de trayectorias para tareas de navegación autónoma y mapeo en ambientes interiores,» Universidad Politécnica de Tulancingo, México, 2018.

[7] E. M. Y. Clelio Endara Sumba, «Desarrollo de un algoritmo de trayectoria para un robot seguidor de línea destreza de competencia mediante visión e inteligencia artificial,» Universidad Politécnica Slesiana Sede Quito, Quito, 2020.

[8] J. M. L. Muñoz, «Detección de señales y planificación de trayectorias,» Escuela Técnica Superior de Ingeniería, Sevilla, 2021.

[9] Vázquez Lucer, Luna Taylor, Santillán y Higuera, «Robot móvil autónomo con reconocimiento y navegación hacia botellas de plástico,» PADI, México, 2022.

[10] S. K. Saha, Introducción a la robótica, India: McGrawHill, 2010.

[11] A. Brrientes, L. F. Peñín , C. Balaguer y R. Aracil, Fundamentos de Robótica, Madrid: McGrawHill, 2007.

[12] M. Motor, «Pequeño gran robot con motores brushless,» [En línea]. Available: https://www.maxongroup.es/medias/sys_master/8815739797534.pdf?attachment=true#:~:text=El%20robot%20est%C3%A1%20compuesto%20por,investigaci%C3%B3n%20y%20la%20ense%C3%B1anza%20cient%C3%ADficas..

[13] M. WAIBEL, «IEEE Spectrum,» IEEE Spectrum, 11 06 2010. [En línea]. Available: https://spectrum.ieee.org/scoop-kukas-youbot.

[14] Locomotec, «KUKA youBot User Manual,» Locomotec, 2013.

[15] K. youBot, «KUKA,» [En línea]. Available: https://www.generationrobots.com/img/Kuka-YouBot-Technical-Specs.pdf.

[16] K. youBot, «Operating and Assembly Instructions,» KUKA Laboratories GmbH, Augsburg, 2012.

[17] J. Newans, «Articulated Robotics,» Articulated Robotics, 02 06 2022. [En línea]. Available: https://articulatedrobotics.xyz/mobile-robot-8-lidar/.

[18] P. P. Cruz, Inteligencia Artificial con apliaciones a la ingeniería, México D.F.: Alfaomega, 2010.

[19] R. Benítez, G. Escudero y S. Kanaan, Inteligencia artificial avanzada, Barcelona: UOC, 2014.

[20] L. Deng y D. Yu, Deep Learning: Methods and Applications, USA: now, 2014.

[21] P. Baheti, «V7Labs,» V7Labs, 31 Agosto 2021. [En línea]. Available: https://www.v7labs.com/blog/deep-reinforcement-learning-guide#neural-networks-and-

deep-reinforcement-learning.

[22] A. Violante, «Medium,» Medium, 18 Marzo 2019. [En línea]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56.

[23] OpenAI, «OpenAI,» OpenAI, 24 Mayo 2017. [En línea]. Available: https://openai.com/research/openai-baselines-dqn.

[24] M. Chaturvedi, «OIM,» OIM, 10 Junio 2021. [En línea]. Available: https://analyticsindiamag.com/what-are-dqn-reinforcement-learning-models/.

[25] Amber, «Medium,» Medium, 26 Abril 2019. [En línea]. Available: https://medium.com/@qempsil0914/deep-q-learning-part2-double-deep-q-network-double-dqn-b8fc9212bbb2.

[26] M. S. Ausin, «Medium,» Medium, 14 Abril 2020. [En línea]. Available: https://markelsanz14.medium.com/introducci%C3%B3n-al-aprendizaje-por-refuerzo-parte-4-double-dqn-y-dueling-dqn-b24ac0a5a46c.

[27] M. Phillips, «Dacademia,» Delta Academy, 1 Septiembre 2022. [En línea]. Available: https://joindeltaacademy.com/blog/why-deepmind-dqn-hard-to-train.

[28] I. A. Lab, «Intel AI Lab,» Intel AI Lab, 2018. [En línea]. Available: https://intellabs.github.io/coach/components/agents/value_optimization/dueling_dqn.html.

[29] cyberithub, «Cyberithub,» Rocket, 14 12 2022. [En línea]. Available: https://www.cyberithub.com/how-to-install-gazebo-on-ubuntu-20-04-lts-focal-fossa/#:~:text=In%20this%20article%2C%20I%20will,outdoor%20environments%20accurately%20and%20efficiently.

[30] E. Robotics, «Elephant Robotics,» Elephant Robotics, [En línea]. Available: https://docs.elephantrobotics.com/docs/gitbook-en/12-ApplicationBaseROS/12.1-ROS1/12.1.4-rivz%E4%BB%8B%E7%BB%8D%E5%8F%8A%E4%BD%BF%E7%94%A8/.

[31] D. O. Delgado, «Openwebinar,» Openwebinar, 21 Septiembre 2017. [En línea].

Available: https://openwebinars.net/blog/que-es-ros/.

[32] S. K. Arora, «hackr.io,» hackr.io, 04 01 2023. [En línea]. Available: https://hackr.io/blog/best-language-for-ai.

[33] HOKUYO, Scanning Laser Range Finder, Japon: HOKUYO AUTOMATIC CO. LTD, 2005.

[34] ROBOTICS, «ROBOTICS,» ROBOTICS Co., Ltd, 2023. [En línea]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/machine_learning/.

[35] A. Benítez, Introducción a la programación en inteligencia artificial, Madrid: Universidad Complutense de Madrid, 2019.

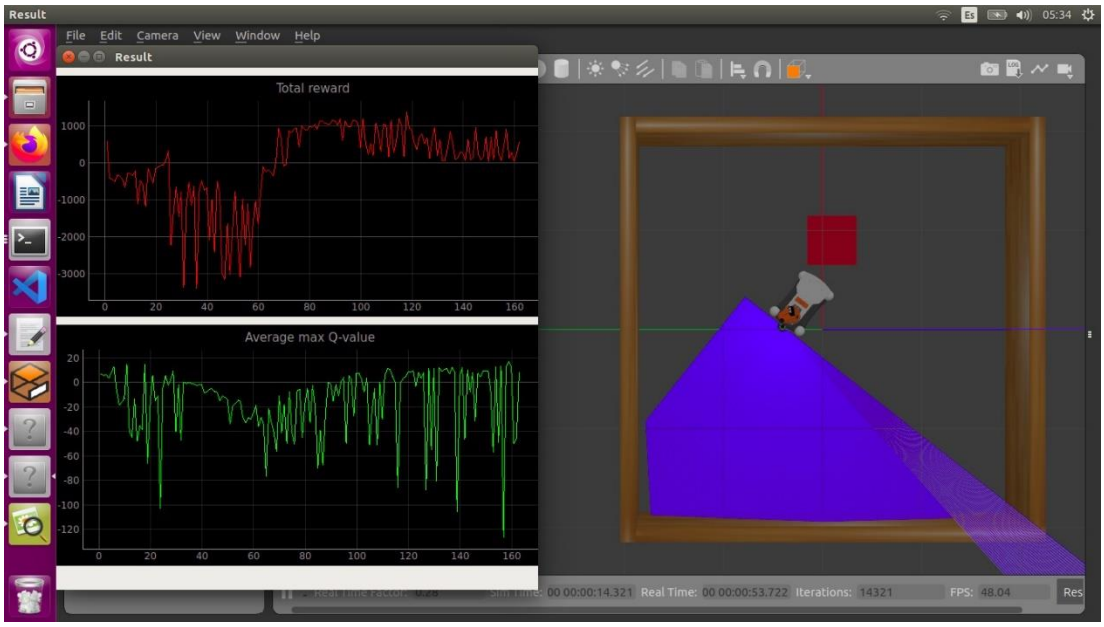
[36] J. Brownlee, «Machine Learning Mastery,» Machine Learning Mastery, 18 04 2016. [En línea].Available: https://machinelearningmastery.com/learning-vector-quantization-for-machine-learning/#:~:text=The%20LVQ%20algorithm%20learns%20the,configurations%20on%20your%20training%20dataset..

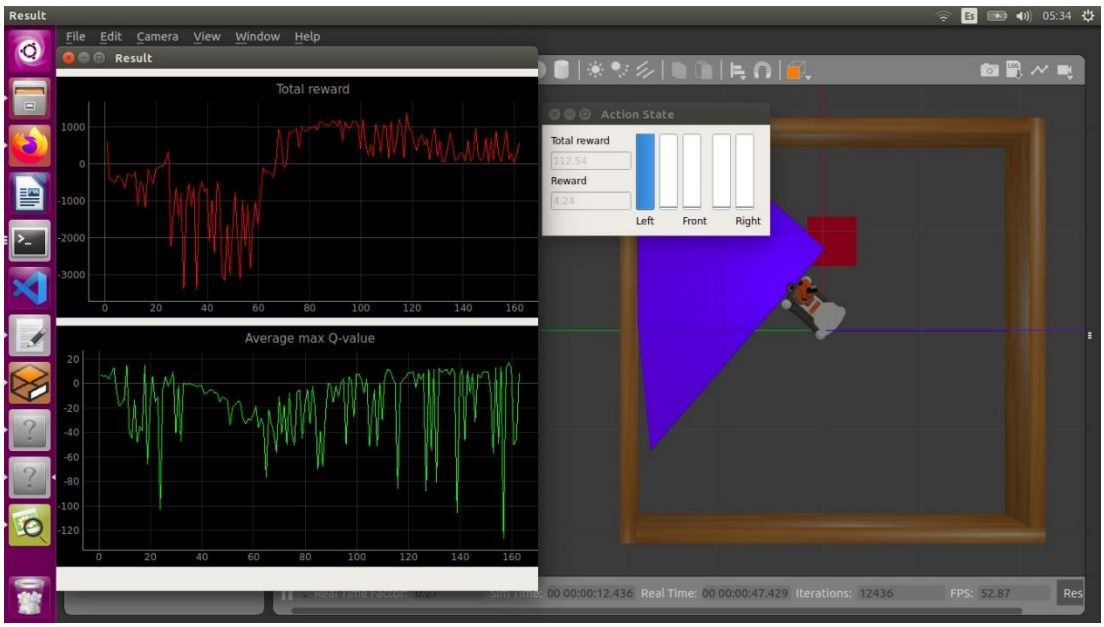
[37] C. Robotics, «CoppeliaSIM,» V-REP, 2022. [En línea]. Available: https://www.coppeliarobotics.com/.

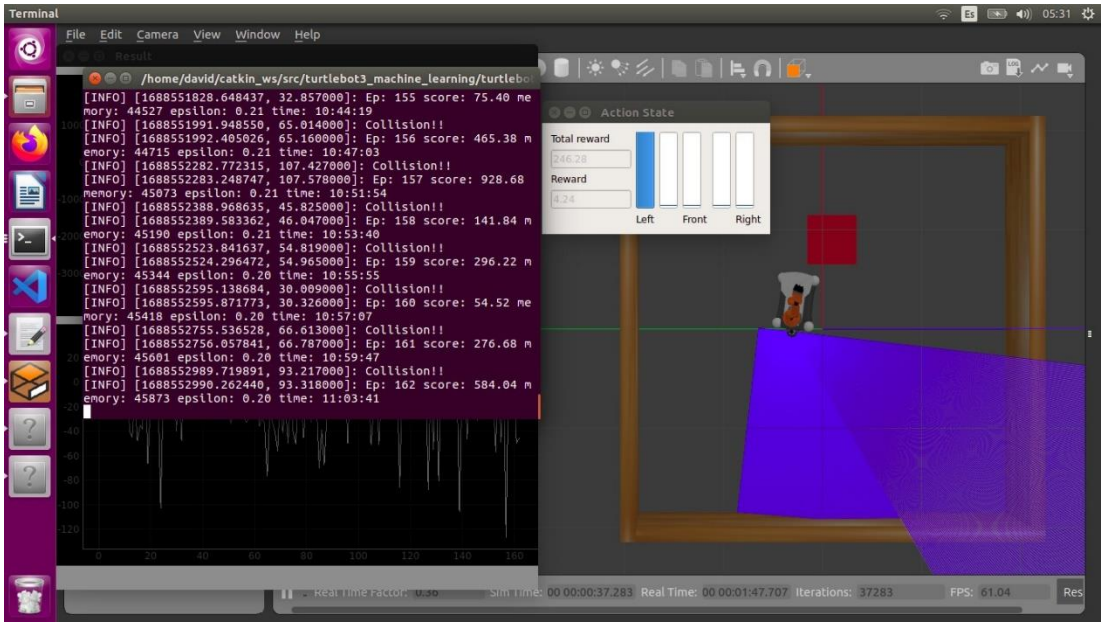
ANEXOS

Anexo 01: Entrenamiento por fases realizado del algoritmo DQN en la plataforma móvil KUKA youBot

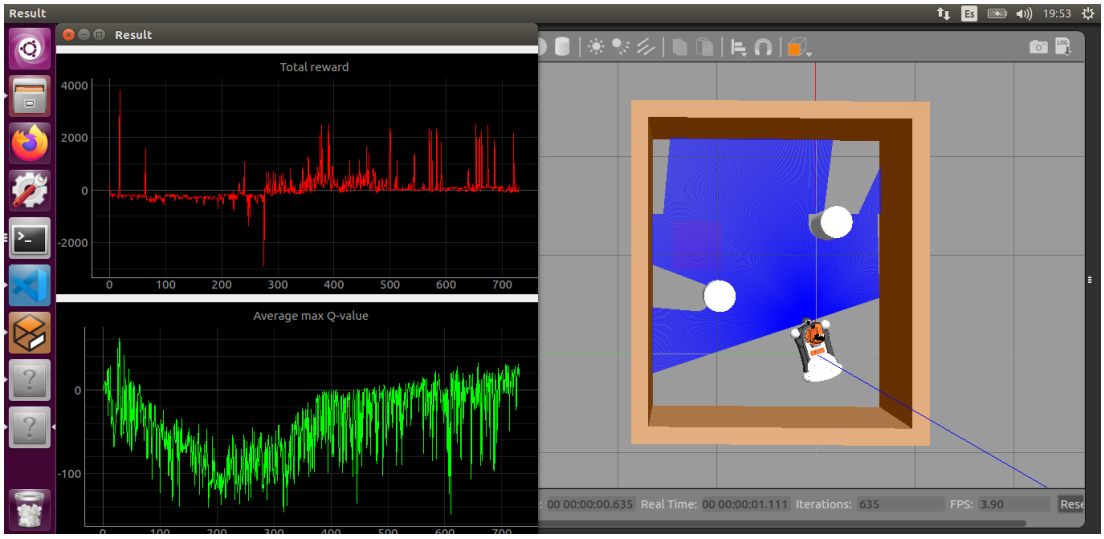
Fase 1

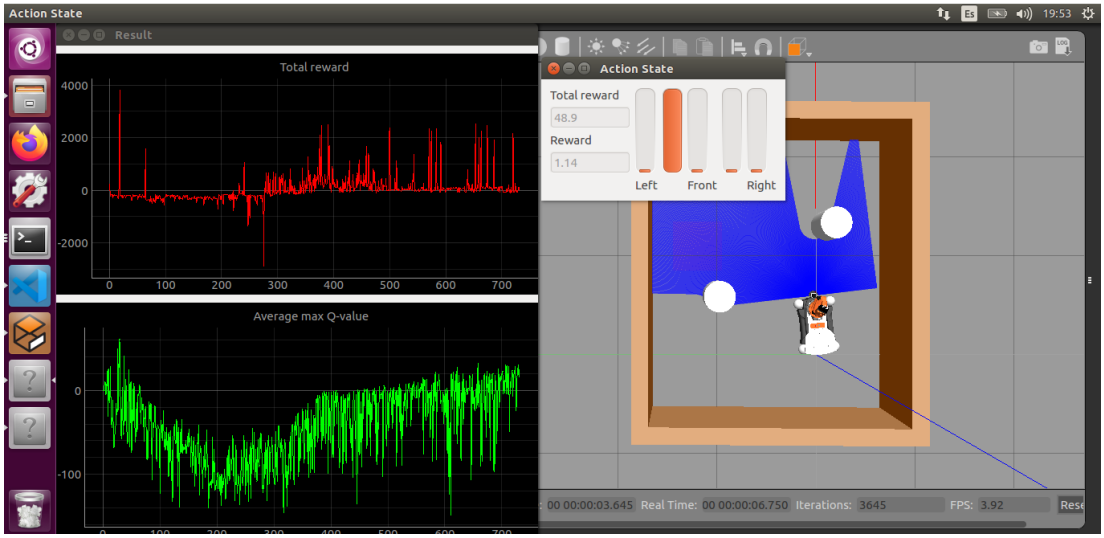


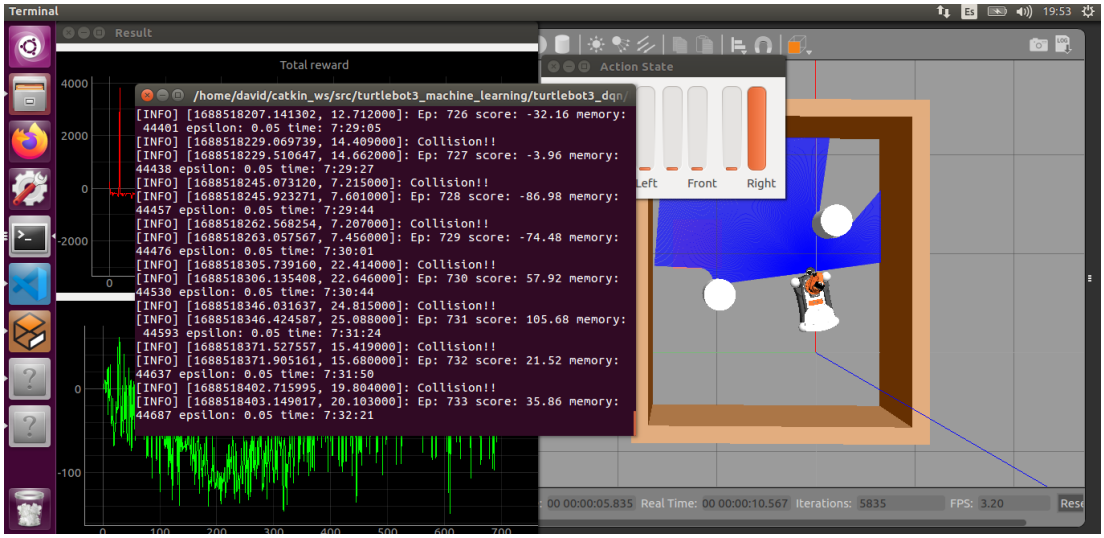



Terminal
File Edit Camera View Window Help
/home/david/catkin_ws/src/turtlebot3_machine_learning/turtlebot
[INFO] [1688551828.648437, 32.857000]: Ep: 155 score: 75.40 me
mory: 44527 epsilon: 0.21 time: 10:44:19
[INFO] [1688551991.948550, 65.014000]: Collision!!
[INFO] [1688551992.405026, 65.160000]: Ep: 156 score: 465.38 m
emory: 44715 epsilon: 0.21 time: 10:47:03
[INFO] [1688552282.772315, 107.427000]: Collision!!
[INFO] [1688552283.248747, 107.578000]: Ep: 157 score: 928.68
memory: 45073 epsilon: 0.21 time: 10:51:54
[INFO] [1688552388.968635, 45.825000]: Collision!!
[INFO] [1688552389.583362, 46.047000]: Ep: 158 score: 141.84 m
emory: 45190 epsilon: 0.21 time: 10:53:40
[INFO] [1688552523.841637, 54.819000]: Collision!!
[INFO] [1688552524.296472, 54.965000]: Ep: 159 score: 296.22 m
emory: 45344 epsilon: 0.20 time: 10:55:55
[INFO] [1688552595.138684, 30.009000]: Collision!!
[INFO] [1688552595.871773, 30.326000]: Ep: 160 score: 54.52 me
mory: 45418 epsilon: 0.20 time: 10:57:07
[INFO] [1688552755.536528, 66.613000]: Collision!!
[INFO] [1688552756.057841, 66.787000]: Ep: 161 score: 276.68 m
emory: 45601 epsilon: 0.20 time: 10:59:47
[INFO] [1688552989.719891, 93.217000]: Collision!!
[INFO] [1688552990.262440, 93.318000]: Ep: 162 score: 584.04 m
emory: 45873 epsilon: 0.20 time: 11:03:41
Action State
Total reward
Reward
Left
Front
Right

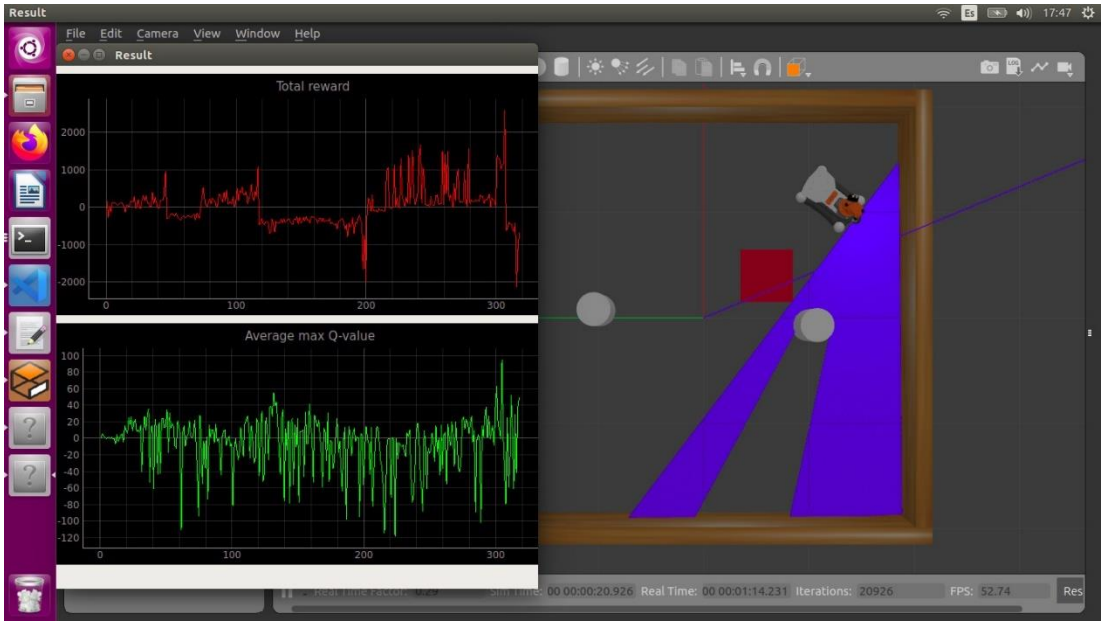
Fase 2

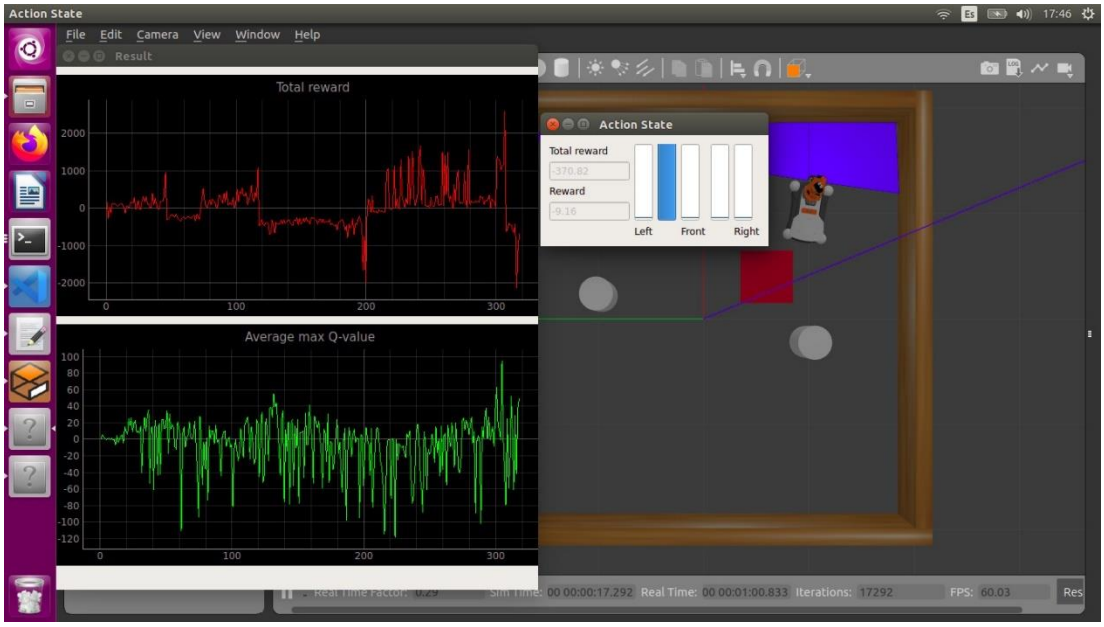

Result
Result
Total reward
Average max Q-value

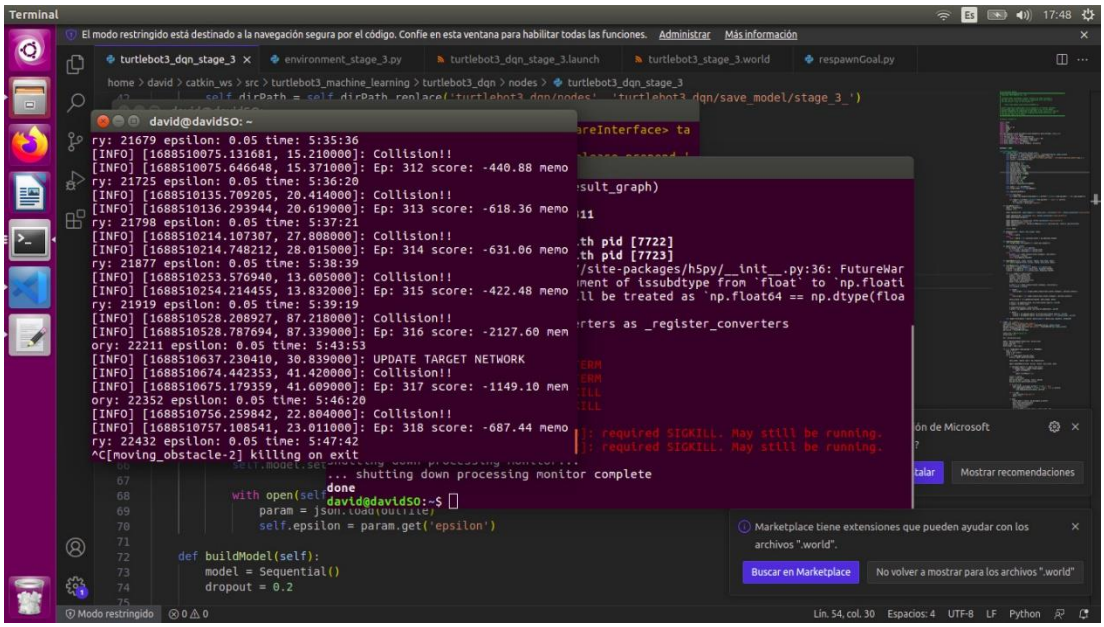

Action State
Result
Total reward
Average max Q-value
Action State
Total reward
Reward
Left
Front
Right


Terminal
Result
Total reward
/home/david/catkin_ws/src/turtlebot3_machine_learning/turtlebot3_dqn/
[INFO] [1688518207.141302, 12.712000]: Ep: 726 score: -32.16 memory: 44401 epsilon: 0.05 time: 7:29:05
[INFO] [1688518229.069739, 14.409000]: Collision!!
[INFO] [1688518229.510647, 14.662000]: Ep: 727 score: -3.96 memory: 44438 epsilon: 0.05 time: 7:29:27
[INFO] [1688518245.073120, 7.215000]: Collision!!
[INFO] [1688518245.923271, 7.601000]: Ep: 728 score: -86.98 memory: 44457 epsilon: 0.05 time: 7:29:44
[INFO] [1688518262.568254, 7.207000]: Collision!!
[INFO] [1688518263.057567, 7.456000]: Ep: 729 score: -74.48 memory: 44476 epsilon: 0.05 time: 7:30:01
[INFO] [1688518305.739160, 22.414000]: Collision!!
[INFO] [1688518306.135408, 22.646000]: Ep: 730 score: 57.92 memory: 44530 epsilon: 0.05 time: 7:30:44
[INFO] [1688518346.031637, 24.815000]: Collision!!
[INFO] [1688518346.424587, 25.088000]: Ep: 731 score: 105.68 memory: 44593 epsilon: 0.05 time: 7:31:24
[INFO] [1688518371.527557, 15.419000]: Collision!!
[INFO] [1688518371.905161, 15.680000]: Ep: 732 score: 21.52 memory: 44637 epsilon: 0.05 time: 7:31:50
[INFO] [1688518402.715995, 19.804000]: Collision!!
[INFO] [1688518403.149017, 20.103000]: Ep: 733 score: 35.86 memory: 44687 epsilon: 0.05 time: 7:32:21
Action State
Left
Front
Right
Real Time: 00 00:00:10.567
Iterations: 5835
FPS: 3.20

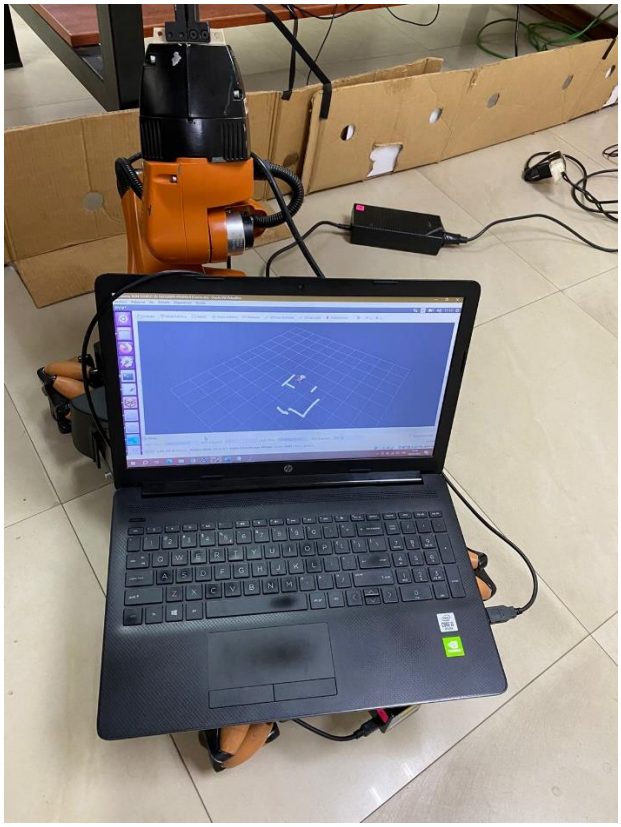
Fase 3

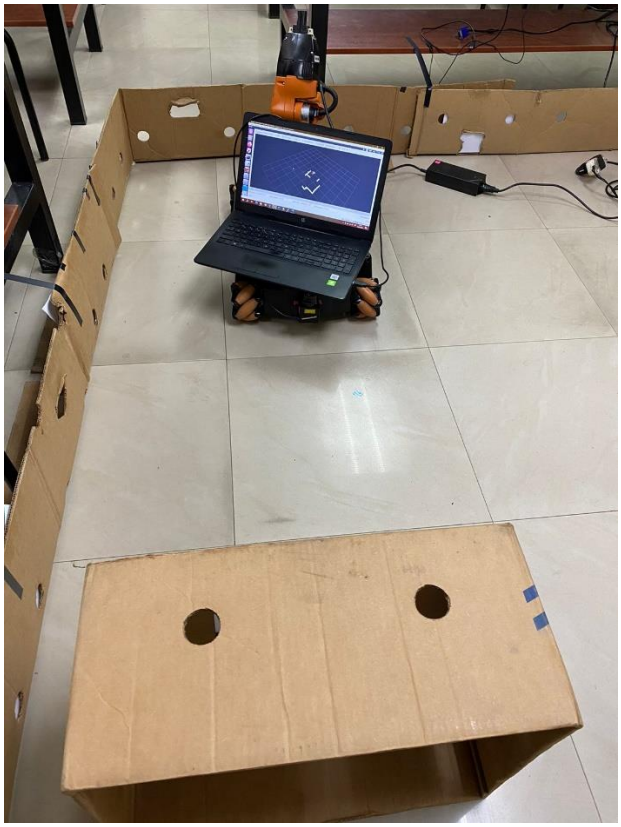

Result
File Edit Camera View Window Help
Result
Total reward
Average max Q-value
Real Time: 00 00:01:14.231
Iterations: 20926
FPS: 52.74


Action State
File Edit Camera View Window Help
Result
Total reward
Average max Q-value
Action State
Total reward
Reward
Left
Front
Right


Terminal
david@davidSO: ~
[INFO] [1688510075.131681, 15.210000]: Collision!!
[INFO] [1688510075.646648, 15.371000]: Ep: 312 score: -440.88 memo
[INFO] [1688510135.709205, 20.414000]: Collision!!
[INFO] [1688510136.293944, 20.619000]: Ep: 313 score: -618.36 memo
[INFO] [1688510214.107307, 27.808000]: Collision!!
[INFO] [1688510214.748212, 28.015000]: Ep: 314 score: -631.06 memo
[INFO] [1688510253.576940, 13.605000]: Collision!!
[INFO] [1688510254.214455, 13.832000]: Ep: 315 score: -422.48 memo
[INFO] [1688510528.208927, 87.218000]: Collision!!
[INFO] [1688510528.787694, 87.339000]: Ep: 316 score: -2127.60 mem
[INFO] [1688510637.230410, 30.839000]: UPDATE TARGET NETWORK
[INFO] [1688510674.442353, 41.420000]: Collision!!
[INFO] [1688510675.179359, 41.689000]: Ep: 317 score: -1149.10 mem
[INFO] [1688510756.259842, 22.804000]: Collision!!
[INFO] [1688510757.108541, 23.011000]: Ep: 318 score: -687.44 memo
^C[moving_obstacle-2] killing on exit
... shutting down processing monitor complete
done
david@davidSO:~$
def buildModel(self):
model = Sequential()
dropout = 0.2

Anexo 02: Pruebas de funcionalidad con Rviz del sensor lidar en el escenario real





Anexo 03: Codificación de la interfaz gráfica de control del sistema

Menú principal interfaz gráfica de control

```
from Tkinter import *
import tkMessageBox
import os
root=Tk()
root.geometry('500x330')
root.title("OPTIMIZACION DE TRAYECTORIAS")
Label(root,text='UNIVERSIDAD TECNICA DE AMBATO',
     font='arial 18 bold').pack()

Label(root,text='"FISEI"',
     font='arial 18 bold').place(x=195, y=30)

Label(root,text='=============================================',
     font='arial 12 bold', foreground ='red').place(x=40, y=60)

Label(root,text='OPTIMIZACION DE TRAYECTORIAS',
     font='arial 11 bold').place(x=90, y=80)
Label(root,text='PLATAFORMAS ROBOTICAS MOVILES',
     font='arial 11 bold').place(x=90, y=97)
Label(root,text='UTILIZANDO TECNICAS DE INTELIGENCIA ARTIFICIAL',
     font='arial 11 bold').place(x=40, y=114)
Label(root,text='=============================================',
     font='arial 12 bold', foreground ='red').place(x=40, y=134)

Label(root,text='POR FAVOR ESCOJA UNA OPCION:',
     font='arial 12 bold').place(x=100, y=160)

def mundo():
    execfile("/home/david/Escritorio/Aplicacion/algoritmos.py")

def conexion():

execfile("/home/david/Escritorio/Aplicacion/menu_comunicacion.py")

def sensor():
    execfile("/home/david/Escritorio/Aplicacion/datos_hokuyo.py")


Button(root, text='ENTRENAMIENTO IA', font='arial 15 bold',
command=mundo).place(x=130, y=190)
Button(root, text='CONECCION REMOTA',font='arial 15
bold',command=conexion).place(x=125, y=230)
Button(root, text='TEST SENSOR HOKUYO',font='arial 15
bold',command=sensor).place(x=115, y=270)

root.mainloop()
```

Interfaz gráfica categoría entrenamiento IA

```
from Tkinter import *
import tkMessageBox
import os
root=Tk()
root.geometry('500x330')
root.title("OPTIMIZACION DE TRAYECTORIAS")
Label(root,text='OPTIMIZACION DE TRAYECTORIAS KUKA YOUBOT GAZEBO',
     font='arial 12 bold').pack()

Label(root,text='"SIMULACION MUNDO GAZEBO KUKA YOUBOT"',
     font='arial 11 bold', foreground ='red').place(x=70, y=20)

Label(root,text='LANZAMIENTO ESCENARIO ENTRENAMIENTO KUKA youBot',
     font='arial 10 bold').place(x=20, y=60)

Label(root,text='LANZAMIENTO ALGORITMO DE APRENDIZAJE INICIAL',
     font='arial 10 bold').place(x=20, y=130)

Label(root,text='LANZAMIENTO ALGORITMO DE APRENDIZAJE OPTIMIZADO',
     font='arial 10 bold').place(x=20, y=200)

#link2=StringVar()
#Label(root,text='IP ROS SUSCRIPTOR (IP PC REMOTA)',
#     font='arial 10 bold').place(x=20, y=100)

def mundo_youbot():
    #os.system("gnome-terminal")
    os.system("roslaunch turtlebot3_gazebo youbot_stage_2.launch")
    #execfile("/home/david/Escritorio/PC-REMOTA.py")

    #execfile("/home/david/Escritorio/PC-REMOTA.py")

    Label(root,text='"CERRADO"',
     font='arial 15 bold', foreground ='blue').place(x=130, y=85)

def aprendizaje_cero():
    os.system("roslaunch turtlebot3_dqn
turtlebot3_dqn_stage_2_1.launch")
    Label(root,text='"ABRIENDO"',
     font='arial 15 bold', foreground ='blue').place(x=130, y=155)

def aprendizaje_270():
    os.system("roslaunch turtlebot3_dqn
turtlebot3_dqn_stage_2.launch")
    Label(root,text='"ABRIENDO"',
     font='arial 15 bold', foreground ='blue').place(x=130, y=225)

def avisos():
    tkMessageBox.showwarning("IMPORTANTE", "En caso de no abrirse
el mundo, cerrar en la terminal con ctr+c y ejecutar: source
.bashrc")
```

```
    tkMessageBox.showwarning("IMPORTANTE", "Este archivo .py debe
ejecutarse en 2 terminales diferentes: 1) Para el lanzamiento del
mundo y 2)Para cualquier aprendizaje")


Button(root, text='ABRIR', font='arial 15 bold',
command=mundo_youbot).place(x=20, y=80)
Button(root, text='ABRIR',font='arial 15
bold',command=aprendizaje_cero).place(x=20, y=150)
Button(root, text='ABRIR',font='arial 15
bold',command=aprendizaje_270).place(x=20, y=220)
Button(root, text='IMPORTANTE!',font='arial 15
bold',bg='red',command=avisos).place(x=170, y=280)

root.mainloop()
```

Interfaz gráfica categoría conexión remota, menú principal

```
from Tkinter import *
import tkMessageBox
import os
root=Tk()
root.geometry('500x230')
root.title("OPTIMIZACION DE TRAYECTORIAS")
Label(root,text='COMUNICACION REMOTA PC - ROBOT REAL',
     font='arial 15 bold').pack()

Label(root,text='POR FAVOR ESCOJA UNA OPCION:',
     font='arial 12 bold', foreground ='red').place(x=100, y=35)

Label(root,text='CONFIGURACION PC REMOTA',
     font='arial 11 bold').place(x=120, y=70)

Label(root,text='CONFIGURACION ROBOT REAL',
     font='arial 11 bold').place(x=120, y=140)

def pc_remota():
    execfile("/home/david/Escritorio/Aplicacion/PC-REMOTA.py")

def robot_real():
    execfile("/home/david/Escritorio/Aplicacion/ROBOT-REAL.py")

Button(root, text='ABRIR', font='arial 15 bold',
command=pc_remota).place(x=190, y=90)
Button(root, text='ABRIR',font='arial 15
bold',command=robot_real).place(x=190, y=160)

root.mainloop()
```

Interfaz gráfica categoría conexión remota, PC remota

```
from Tkinter import *
import tkMessageBox
import os
root=Tk()
root.geometry('500x250')
root.title("CONFIGURACION COMUNICACION REMOTA KUKA YOUBOT")
Label(root,text='CONFIGURACION PC REMOTA KUKA YOUBOT',
     font='arial 15 bold').pack()

Label(root,text='"CONFIGURACION SOLO PARA PC REMOTA"',
     font='arial 11 bold', foreground ='red').place(x=70, y=20)
global link1, link2
link1=StringVar()
Label(root,text='IP ROS MASTER (KUKA YOUBOT REAL)',
     font='arial 10 bold').place(x=20, y=60)
link_enter=Entry(root, width=20,
                 textvariable=link1).place(x=290, y=60)

link2=StringVar()
Label(root,text='IP ROS SUSCRIPTOR (IP PC REMOTA)',
     font='arial 10 bold').place(x=20, y=100)
link_enter=Entry(root, width=20,
                 textvariable=link2).place(x=290, y=100)

def direcciones():
    global link1, link2
    with open('/home/david/.bashrc',"a") as file:
      file.write("\n")
      file.write("export ROS_MASTER_URI=http://")
      file.write(str(link1.get()))
      file.write(":11311\n")
      file.write("export ROS_IP=")
      file.write(str(link2.get()))
      os.open("/home/david/.bashrc",os.O_RDONLY)

    Label(root,text='"GUARDADO"',
     font='arial 15 bold', foreground ='red').place(x=160, y=155)

    tkMessageBox.showwarning("IMPORTANTE", "Una vez guardada la
configuracion, ingresar en la terminal la instruccion: source
.bashrc")

Button(root, text='GUARDAR',
      font='arial 15 bold',
      command=direcciones).place(x=20, y=150)

root.mainloop()
```

Interfaz gráfica categoría conexión remota, robot real KUKA youBot

```
from Tkinter import *
import tkMessageBox
root=Tk()
root.geometry('500x250')
root.title("CONFIGURACION COMUNICACION REMOTA KUKA YOUBOT")
Label(root,text='CONFIGURACION ROBOT REAL KUKA YOUBOT',
     font='arial 15 bold').pack()

Label(root,text='"CONFIGURACION SOLO PARA ROBOT REAL"',
     font='arial 11 bold', foreground ='red').place(x=70, y=20)

link1=StringVar()
Label(root,text='IP ROS MASTER (IP KUKA YOUBOT REAL)',
     font='arial 10 bold').place(x=20, y=60)
link_enter=Entry(root, width=20,
                 textvariable=link1).place(x=290, y=60)

link2=StringVar()
Label(root,text='HOSTNAME ROS (IP KUKA YOUBOT REAL)',
     font='arial 10 bold').place(x=20, y=100)
link_enter=Entry(root, width=20,
                 textvariable=link2).place(x=290, y=100)

def direcciones():
    with open('/home/david/.bashrc',"a") as file: #aqui cambiar
"david" por "youbot"
      file.write("\n")
      file.write("export ROS_MASTER_URI=http://")
      file.write(str(link1.get()))
      file.write(":11311\n")
      file.write("export ROS_HOSTNAME=")
      file.write(str(link2.get()))

    Label(root,text='"GUARDADO"',
     font='arial 15 bold', foreground ='red').place(x=160, y=155)

    tkMessageBox.showwarning("IMPORTANTE", "Una vez guardada la
configuracion, ingresar en la terminal las instrucciones: 1)
export ROS_MASTER_URI=http://DIRECCION IP ROBOT:11311, 2) export
ROS_IP=DIRECCION IP ROBOT, 3) source .bashrc")

Button(root, text='GUARDAR',
      font='arial 15 bold',
      command=direcciones).place(x=20, y=150)

root.mainloop()
```

Diseño interfaz de monitoreo datos hokuyo

```
import rospy
from sensor_msgs.msg import LaserScan
import Tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from matplotlib import pyplot
import matplotlib.animation as animation
from matplotlib.animation import FuncAnimation


plt.style.use('ggplot')
x_data = []
y_data = []
z_data = []
w_data = []

figure = pyplot.figure()
line1, = pyplot.plot_date(x_data, y_data, 'o', linewidth=3)
line2, = pyplot.plot_date(x_data, z_data, 'o', linewidth=3)
line3, = pyplot.plot_date(x_data, w_data, 'o', linewidth=3)
pyplot.legend(('IZQUIERA', 'CENTRO', 'DERECHA'),
prop = {'size': 8}, loc='upper right')
pyplot.xlabel('TIEMPO REAL / s')
pyplot.ylabel('RECONOCIMIENTO')
pyplot.title('DATOS SENSOR LIDAR HOKUYO URG-04LX-UG01')

def scan_callback(msg):

    global range_center, range_left, range_right

    range_center = msg.ranges[len(msg.ranges)/2]
    range_left = msg.ranges[len(msg.ranges)-1]
    range_right = msg.ranges[0]
    #print ("PARAMETROS: Izquierda - %0.1f" %range_left, "Centro-
%0.1f" %range_center, "Derecha - %0.1f" %range_right)

def grafica3(frame):

    global range_center, range_left, range_right

    x_data.append(datetime.now())
    y_data.append(range_left)
    z_data.append(range_center)
    w_data.append(range_right)
    line1.set_data(x_data, y_data)
    line2.set_data(x_data, z_data)
    line3.set_data(x_data, w_data)

    figure.gca().relim()
```

```
    figure.gca().autoscale_view()
    return line1, line2, line3

def iniciar():
        global ani
        ani = animation.FuncAnimation(figure, grafica3,
interval=1000)
        canvas.draw()

def pausar():
        ani.event_source.stop()

def reanudar():
        ani.event_source.start()


ventana = tk.Tk()

ventana.geometry('642x535')
ventana.wm_title('SENSOR HOKUYO URG-04LX-UG01')
ventana.minsize(width=642,height=535)

frame = tk.Frame(ventana,  bg='gray22',bd=3)
frame.pack(expand=1, fill='both')

canvas = FigureCanvasTkAgg(figure, master = frame)
canvas.get_tk_widget().pack(padx=5, pady=5 , expand=1,
fill='both')

tk.Button(frame, text='Grafica Datos', width = 15,
bg='salmon',fg='white', command=iniciar).pack(pady
=5,side='left',expand=1)
tk.Button(frame, text='Pausar', width = 15,
bg='salmon',fg='white', command=pausar).pack(pady
=5,side='left',expand=1)
tk.Button(frame, text='Reanudar', width = 15,
bg='salmon',fg='white', command=reanudar).pack(pady
=5,side='left',expand=1)

rospy.init_node('range_ahead')
scan_sub = rospy.Subscriber('scan', LaserScan, scan_callback)
rospy.spin()
ventana.mainloop()
```

Anexo 04: Cotización de las horas empleadas en la investigación

Se considera 4 meses del desarrollo total del proyecto en representación de la mano de obra, donde se trabaja 8 horas al día y 6 días a la semana. El costo de mano de obra se estima dependiendo de la complejidad de cada parte involucrada.

Ítem	Partes involucradas	Cantidad de horas	P. Hora. (USD)	P. Total (USD)
1	Análisis de los diferentes tipos plataformas omnidireccionales con soporte de ROS	30	$10,00	$300,00
2	Análisis de los componentes compatibles a utilizar en la plataforma	40	$12,00	$480,00
3	Simulación de la plataforma omnidireccional en simulación	80	$15,00	$1120,00
4	Pruebas de movimiento de la plataforma omnidireccional en simulación	80	$15,00	$1120,00
5	Recreación de un entorno real en simulación	105	$18	$1890,00
6	Implementación del algoritmo IA en la plataforma omnidireccional simulada	110	$18	$1980,00
7	Entrenamiento del algoritmo IA en la plataforma omnidireccional en el entorno simulado	180	$18	$3240,00

8	Implementación del algoritmo IA entrenado en la plataforma omnidireccional real	40	$20	$800,00
9	Pruebas de funcionamiento de la plataforma omnidireccional en el entorno real	70	$20	$1400,00
10	Elaboración de documentación	33	$25	$825,00
	TOTAL	**768**	**-**	**$13155**