UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO-SALAMANCA

DIVISIÓN DE INGENIERÍAS

"Navegación de un robot móvil mediante una red pulsante con aprendizaje por refuerzo"

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA

PRESENTA:

Ing. Alvaro Javier Patiño Saucedo

DIRECTORES:

Dr. Horacio Rostro González

Dr. Erick Israel Guerra Hernández

Salamanca, Gto. Mayo, 2022

Salamanca, Gto., a 5 de mayo del 2022

M.I. HERIBERTO GUTIÉRREZ MARTÍN
COORDINADOR DE ASUNTOS ESCOLARES
PRESENTE

Por medio de la presente, se otorga autorización para proceder a los trámites de impresión, empastado de tesis y titulación al alumno Alvaro Javier Patiño Saucedo del **Programa de Maestría en Ingeniería Eléctrica (Instrumentación y Sistemas Digitales)** y cuyo número de NUA es: 147351, del cual somos directores de tesis. El título de la Tesis es: *Navegación de un robot móvil mediante una red pulsante con aprendizaje por refuerzo.*

Hago constar que hemos revisado dicho trabajo y tenido comunicación con los dos sinodales asignados para la revisión de la tesis, por lo que no hay impedimento alguno para fijar la fecha de examen de titulación.

ATENTAMENTE

DR. HORACIO ROSTRO GONZÁLEZ
DIRECTOR DE TESIS
SECRETARIO

DR. ERICK ISRAEL GUERRA HERNÁNDEZ
DIRECTOR DE TESIS

DR. MARIO ALBERTO IBARRA MANZANO
PRESIDENTE

DR. DAVID CAMARENA MARTÍNEZ
VOCAL

Resumen

En este trabajo se implementa una estrategia de aprendizaje por refuerzo (RL, por sus siglas en inglés) para la navegación de un robot móvil en un ambiente estructurado. Este trabajo fue implementado en un hardware integrado por una tarjeta de procesamiento Raspberry Pi y un robot móvil construido con el kit Lego Mindstorms EV3.

La estrategia de aprendizaje se conoce como Spike-Timing-Dependent-Plasticity modulado por recompensa (R-STDP) y es aplicada para el entrenamiento de una red neuronal pulsante (SNN por sus siglas en inglés) la cual controla la navegación del robot móvil en el entorno estructurado.

La implementación del método fue llevada a cabo en el lenguaje de programación Python con ayuda de la librería VREP. El programa desarrollado se carga y se ejecuta en la Raspberry Pi, la comunicación entre ésta y el Brick EV3 se da a través del entorno de desarrollo integrado Matlab.

La validación de la implementación se hizo efectiva utilizando un sistema retroalimentado, compuesto principalmente por Raspberry Pi, el Brick EV3 y los sensores ultrasónicos. El experimento de validación consiste en que el agente (robot móvil) obtiene información del ambiente de desarrollo a través de sus sensores, estos datos son recibidos por la red neuronal pulsante el cual se hace cargo de analizar, procesar y tomar una decisión sobre qué acción ejecutar (derecha, izquierda, adelante) de acuerdo a la experiencia obtenida en el entrenamiento. El criterio que maneja la red para la toma de decisiones es el de desplazarse en dirección hacia donde los sensores no detecten obstáculos o estén más lejanos, obteniendo así una mayor recompensa por sus acciones.

Dedicatoria

A mis queridos padres: Sr. Alvaro Patiño Vanegas y Sra. Libia del Carmen Saucedo Uribe, por su apoyo incondicional.

A mis queridos hermanos: Janns Patiño y Alvaro José Patiño, quiénes han sido mi guía en este camino.

A mi amada pareja, Maria Alejandra Ortega, porque siempre ha creído en mí.

A mi primo, Alberto Patiño, por su apoyo y consejos.

A mis tíos: Jorge Patiño, Alberto Patiño, John Patiño y Juan Saucedo, por estar presente en cada paso que doy.

A mis abuelos y tíos: Franco Patiño †, Juan Saucedo Vargas †, Juana Vanegas †, Alfonso Patiño † y Carlos Saucedo †, siempre presentes en mi corazón.

Agradecimientos personales

Al Dr. Horacio Rostro por su confianza, apoyo, orientación y formación desde mi llegada, siendo un guía importante para mi crecimiento profesional durante la maestría.

Al Dr. Erick Guerra por su apoyo, orientación y dirección durante el desarrollo de este trabajo.

Al Dr. David Camarena por la oportunidad brindada, su apoyo, amistad y confianza.

Al Dr. Juan Manuel Vilardy que aún en la distancia conté con su dirección y apoyo.

Al M.I. Daniel Zambrano por todo su apoyo y consejos.

A todas aquellas personas que de una u otra forma aportaron en la realización de este trabajo de Tesis.

Agradecimientos institucionales

Expreso mi más sincera gratitud hacia la Universidad de Guanajuato, especialmente a la División de Ingenierías Campus Irapuato-Salamanca (DICIS) por la formación y por el apoyo que he recibido. A todos los profesores mis agradecimientos por sus enseñanzas en estos dos años que me permitieron obtener el título de Maestría en Ingeniería Eléctrica.

A mi Alma mater, la Universidad Popular del Cesar (Colombia), base sólida de mi formación académica.

Este trabajo de tesis fue realizado gracias al apoyo recibido a través del Consejo Nacional de Ciencia y Tecnología CONACYT de México, bajo el número de becario 763911 y CVU 1046013.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Introducción

En la actualidad, los robots móviles son instrumentos usados en distintas aplicaciones, tales como exploración, transporte, vigilancia, procesos agrícolas, limpieza, operaciones militares o de búsqueda y rescate, entre otros (Gai et al., 2021; Lozada et al., 2020; Ha et al., 2019). En situaciones como las mencionadas donde el ser humano tenga limitantes o simplemente desee automatizar algún proceso, se le debe dar a los robots móviles capacidades como la inteligencia y autonomía. En el caso de la navegación autónoma por ejemplo, para moverse en un ambiente del mundo real sin la conducción o ayuda humana y ejecutar las tareas establecidas como la locomoción, detección, localización, planificación de rutas y evasión de obstáculos (Siegwart et al., 2004).

Por otro lado, la aparición de las redes neuronales artificiales y el aprendizaje por refuerzo en conjunto con otras técnicas de inteligencia artificial (AI, por sus siglas en inglés) está produciendo avances tecnológicos importantes en el campo de la navegación robótica móvil, debido a ciertas características como la capacidad de aprender con el ejemplo y la alta velocidad de procesamiento de datos en paralelo (Zou et al., 2006). En los últimos años se ha demostrado buen rendimiento en la navegación de robots tanto simulada, con

múltiples objetivos (navega a diferentes metas) (Shantia et al., 2021), en entornos dinámicos y entre agentes autónomos externos (otros agentes) (Zhang et al., 2020), como experimentos robóticos reales (Wang et al., 2021). En este trabajo se quiere hacer uso de las ventajas que nos ofrecen el aprendizaje por refuerzo y la inteligencia artificial para implementar una red neuronal pulsante entrenada con una estrategia de aprendizaje autónomo en un robot móvil para su correcta navegación de un punto a otro sin colisionar con los obstáculos encontrados en un ambiente estructurado.

1.2. Antecedentes

El ser humano es capaz de realizar tareas cognitivas como reconocer, aprender, memorizar y recordar de forma eficiente, esto nos permite relacionarnos con el mundo a nuestro alrededor. En 1906 el médico y citólogo italiano Camillo Golgi y el científico español Santiago Ramón y Cajal recibieron el premio nobel de Medicina y Fisiología, su teoría neuronal tomó lugar como pieza fundamental en la organización del sistema nervioso (Torres-Fernández, 2006). Ellos establecieron que éste se constituye por células independientes, llamadas neuronas. Estas procesan la información mediante pequeños pulsos eléctricos en el tiempo, llamados potenciales de acción, donde gracias a un elemento neuronal conocido como sinapsis, estos sirven de comunicación entre dendritas y soma.

En las últimas decadas, han surgido modelos inteligentes que emulan el comportamiento de un sistema nervioso biológico, como lo son las redes neuronales artificiales (RNA) y se pueden distinguir tres tipos de RNA (Maass, 1997). Entre las décadas de 1950 y 1960 el científico Frank Rosenblatt creó el Perceptrón, y con él nació la primera generación de RNA (Rosenblatt, 1960). La segunda generación amplió las posibles salidas de las redes neuronales mediante diferentes funciones de activación continuas, como la función sigmoide, abriendo así su campo de aplicación (Salazar Triana and Sánchez Moreno, 2018). A diferencia del Perceptrón, este modelo tiene capacidad para resolver problemas no lineales (por ejemplo, XOR) mediante el uso de una sola capa de neurona (Hernández-Becerra and Mejía-Lavalle, 2016).

Aunque las redes neuronales artificiales han sido un gran éxito, en algunos ámbitos como el procesamiento de voz y el reconocimiento de patrones, la exigencia en la capacidades

de procesamiento de datos durante el entrenamiento y de energía, así como la propia implementación están necesitando una de mayor recursos por el incremento en la información a procesar. Es aquí donde se ha encontrado una posible solución, esto es, reemplazando este tipo de neuronas artificiales con una neurona pulsante (Davidson and Furber, 2021), las cuales son conocidas como de tercera generación, estas se asemejan más a las redes neuronales biológicas, son redes neuronales artificiales más realistas, con rápido procesamiento de la información y eficiencia energética. Se da a conocer que el modelo de la tercera generación de redes neuronales tiene un poder de cómputo mayor que los modelos mencionados en las dos primeras generaciones. Pese a que los modelos de neuronas pulsantes existen desde inicios del siglo XX, es muy reciente su aplicación en modelos para el reconocimiento de patrones (Hernández-Becerra and Mejía-Lavalle, 2016), también ha sido de gran provecho para el proceso eficiente de imágenes médicas (Mejia-Lavalle et al., 2019) gracias a su poder de cómputo en paralelo. Actualmente, se encuentran implementaciones de sistemas neuro-inspirados usados para el diseño de sistemas de control y procesamiento de señales (Jiménez Fernández, 2010) y del mismo modo para el procesamiento de la información basados en estos tipos de modelos (neuronales pulsantes) (Cerezuela Escudero, 2015).

A pesar de que ha sido poco el avance sobre la construcción de sistemas neuronales en hardware, unas de las aplicaciones que estas redes neuronales pulsantes ofrecen es la segmentación y reconocimiento de imágenes (Sanchez et al., 2019; Ruge Ruge and Alvarado, 2013). Hay investigaciones que nos muestran el largo camino por recorrer en el diseño de redes neuronales pulsantes en hardware (Zhang et al., 2021). Redes tan grandes como estas parecen ser inmanejables en implementación digital, algunos estudios demuestran que el uso de FPGA para su implementación, da ventajas tanto en el diseño como en la disminución del tiempo de desarrollo, aumenta el rendimiento del sistema y reduce los costos de fabricación, y lo más importante es que permite un hardware reconfigurable sin necesidad de grandes cambios en el diseño (Schrauwen, Benjamin and D'Haene, Michiel and Verstraeten, David and Van Campenhout, Jan, 2008).

Por otro lado, el condicionamiento clásico que fue adoptado por la rama de la psicología, es una teoría conductista del aprendizaje donde los experimentos más conocidos son del fisiólogo ruso Pavlov (1997). En la situación cuando un agente se relaciona con el ambiente y ejecuta acciones que lo cambian de estado a medida que pasa el tiempo, el entorno da una recompensa que puede ser positiva o negativa, mientras va ejecutando estas

acciones se debe determinar si el estado actual lo acerca o aleja del resultado que se quiere obtener (Dietterich, 2017).

En el aprendizaje por refuerzo (RL, por sus siglas en inglés) el tema de la asignación de créditos (Minsky, 1961) es un problema a resolver, a esto se suma la dificultad de recompensar a medida que transcurre el tiempo, ya que si un agente logra ese estímulo puede estar en un estado diferente cuando sea recompensado, a este fenómeno se le conoce como recompensa distal. Este problema ha sido muy estudiado en los últimos años en búsqueda de una estructura neuronal más cercana a lo biológicamente conocido (Taylor et al., 2012). Un equipo de investigación que pertenece al Instituto Médico Howard de California, conducido por la investigadora Schuman descubrió la manera en que se puede usar la dompamina (molécula importante que facilita la comunicación entre las neuronas en el cerebro) su función es estimular la síntesis de proteínas durante ciertos procesos neuronales (Smith et al., 2005). La función que cumple la dopamina en el aprendizaje por refuerzo es dependiente del tiempo en el que sucedan las acciones con la recompensa, sabiendo que la dopamina fortalece las conexiones sinápticas entre las neuronas (Nitz et al., 2007), lo importante que es para los circuitos del cerebro donde se incluyen los responsables de la habilidad para aprender del ser humano y el reconocer si las acciones nos llevan a consecuencias negativas o positivas ante las respuestas del ambiente. Algunos estudios muestran como se puede resolver el problema de recompensa distal con una red neuronal pulsante que emula el método spike-timing-dependent-plasticity (STDP) modulado por dopamina. Son patrones de disparos que coinciden en corto tiempo, a escala de los milisegundos. El autor se enfoca en lo preciso que son los disparos en la dinámica cerebral y propone una señal de refuerzo que ayuda a la sinapsis en el momento correcto, usándolo para solucionar dicho problema (Izhikevich, 2007).

Los resultados obtenidos en estudios recientes demuestran que las redes neuronales pulsantes con la regla de aprendizaje Spike-Timing-Dependent-Plasticity son muy eficientes en el entrenamiento de controladores para robots móviles, tanto en aprendizaje supervisado (Bing et al., 2019) como en aprendizaje autónomo (Lu et al., 2021) logran desarrollar funciones específicas (como la evasión de obstáculos) bajo distintos estados y ambientes después de finalizar su respectivo proceso de aprendizaje.

1.3. Justificación

Una tarea que aún sigue en estudio es como el cebrebro humano en conjunto con el sistema nervioso tienen esa alta eficiencia transportando grandes cantidades de información y de forma paralela en pulsos eléctricos, se desconoce si implementa patrones o sigue alguna codificación. Aunque no sea fácil de descifrar, en la actualidad, se han obtenido buenos resultados no imitando de una forma auténtica el sistema nervioso pero con el uso de las neuronas pulsantes artificiales en sistemas neuromórficos nos puede representar ciertas ventajas tecnológicas desde la perspectiva de la neurobiología, la neurociencia computacional y la ingeniería neuromórfica acercándose a este efectivo comportamiento de la naturaleza (George et al., 2020). El estudio de modelos neuronales pulsantes representa actualmente un campo de vanguardia ya que con estos modelos se busca tener entes artificiales más eficientes en términos de consumo energético, velocidad de procesamiento y precisión. Estos modelos han inspirado el desarrollo de un nuevo campo en la ingeniería, el desarrollo de sistemas neuromórficos los cuales buscan ir más allá del cómputo tradicional. Estos sistemas se caracterizan por ser altamente eficientes al momento de procesar información, razón por la cual IBM, Intel y la Universidad de Manchester han apostado por desarrollar esta tecnología. De manera específica, un sistema neuromórfico es todo hardware configurado con modelos biológicamente inspirados tal como una red neuronal pulsante implementada en una neuroprótesis para recuperar la funcionalidad en personas con discapacidad neurológica (Buccelli et al., 2019).

En este sentido se considera la posibilidad de implementar y validar una red neuronal pulsante (SNN) sobre una arquitectura de hardware donde se busca entrenar a la red con una estrategia de aprendizaje por refuerzo ya que en conjunto con la inteligencia artificial son el presente y futuro en cuanto a la optimización de procesos, y ante el requisito de lograr optimizar el controlador del robot móvil usado para la observación y extracción de la información relevante acerca de cada elemento del entorno estructurado para que con estos datos realice una navegación y evasión de obstáculos exitosa.

1.4. Objetivo General

El objetivo principal del presente trabajo es desarrollar una estrategia de aprendizaje por refuerzo que permita la navegación de un robot móvil en un entorno estructurado.

1.5. Objetivos Específicos

1. Implementar una red neuronal pulsante con ayuda de la librería VREP.
2. Aplicarle a la red una regla de aprendizaje por refuerzo enfocada a la navegación de un robot móvil.
3. Una vez implementada y validada la red en software, programarla en una tarjeta Raspberry pi con el fin de que se pueda embeber a un robot.
4. Configuración de la plataforma robótica, esto se trata de adaptar sensores y la tarjeta de procesamiento (Raspberry pi).
5. Validación de la red en la plataforma robótica.
6. Corrección de parámetros, esto se contempla ya que normalmente a la hora de implementar el modelo en una plataforma real algunos parámetros deberán ser calibrados.
7. Escritura de la tesis.

1.6. Organización de la tesis

La tesis consta de cinco capítulos que se resumen a continuación:

Capítulo 1: presenta la introducción y los objetivos de la tesis.

Capítulo 2: muestra la definición de conceptos teóricos básicos sobre los principios de los robots móviles, modelo cinemático, métodos para su navegación, redes neuronales artificiales y el aprendizaje por refuerzo, todo esto necesario para comprender el siguiente

trabajo.

Capítulo 3: describe cada una de las herramientas que se necesitó para la exitosa ejecución del trabajo. Primero un amplio repaso sobre la arquitectura del Raspberry pi y su funcionamiento. A continuación, una descripción detallada del funcionamiento del sensor ultrasónico (HC-SR04). Y por último, se mostrará una breve descripción del Brick Lego Mindstorms EV3 usado para la construcción del robot móvil.

Capítulo 4: presenta la implementación y validación del sistema completo. Donde paso a paso se describe cómo es la adaptación de los sensores a la Raspberry pi y Además, de cómo se da la comunicación entre Raspberry pi y el Brick EV3 a través del entorno de desarrollo integrado Matlab.

Capítulo 5: consiste en la redacción de los resultados obtenidos. Se concluye el aporte del proyecto y los trabajos futuros.

Capítulo 2

Marco Teórico

2.1. Introducción

Los algoritmos basados en información incompleta o reactivos es una de las técnicas más interesantes para que un robot móvil se pueda mover en un entorno y la más efectiva para implementaciones en la vida real. Es una alternativa a la planificación de rutas o en casos de que no se cuente con un mapa del entorno sobre el cual se navegará, la implementación de estos algoritmos le otorgarán comportamientos reactivos al robot móvil. En este capítulo se hará una introducción a la navegación en robots móviles, sus principios y técnicas más importantes. Además, se definirán conceptos teóricos sobre las redes neuronales artificiales y la estrategia del aprendizaje por refuerzo.

2.2. Conceptos previos de movilidad y navegación robótica

Definición 2.2.1 (locomoción). Es el proceso en el cual se desplaza un vehículo autónomo o robot, y se aplican fuerzas con el fin de producir este movimiento.

Definición 2.2.2 (rueda directriz). Son las ruedas de direccionamiento de orientación

controlable.

Definición 2.2.3 (ruedas locas o ruedas de castor). Son las ruedas orientables no controladas.

Definición 2.2.4 (restricciones no holónomas). Significa que el robot puede moverse instantáneamente adelante o atrás pero no lateralmente por el deslizamiento de ruedas, se le conoce como problema del estacionamiento paralelo.

Definición 2.2.5 (misión). Es cuando el robot móvil ejecuta un desplazamiento y en conjunto interactúa con los distintos elementos del entorno cumpliendo con una serie de objetivos de una tarea específica.

2.3. Robótica

La robótica es una de las ciencias que ha tenido un crecimiento exponencial en los últimos años, se trata de la construcción de dos piezas fundamentales: Hardware y Software. Software es la parte que se encarga de razonar a través de las desiciones que toma, sean guiadas o totalmente autónomas, y con esto darle vida al Hardware, que es la parte electromecánica dotada de sensores y actuadores (Ortiz Arroyave et al., 2018; Angleraud et al., 2021; Cuan, 2021).

Se puede decir que la robótica es el área de estudio encargada de implementar sistemas electromecánicos «autónomos» que son útiles para la humanidad, se usan para múltiples disciplinas y con una gran variedad de aplicaciones. No es fácil señalar un comienzo de esta a lo largo de la historia, pero podemos encontrar hechos transcendentes como el pato de Vaucanson (Wood, 2002) y el legado de autómata de Da Vinci (Rosheim, 2006), es lo más cercano a mecanismos que realizan una tarea de forma programada. Cuando se habla de este tema es necesario destacar dos campos, el área industrial centrada en el estudio de robots manipuladores y un segundo bloque donde la robótica se enfoca en diseñar robots móviles autónomos. El área industrial lleva siendo estudiada desde los años 50 (Murphy, 2000), y en el presente hace parte de toda la estructura industrial en los países desarrollados, se ha expandido en la industria automovilística, eléctrica y electrónica.

2.4. Principios de los robots móviles

Los robots móviles, no son simplemente una recopilación de algoritmos en prueba sino también estructuras físicas que operan en el mundo real. De manera específica, son sistemas que se caracterizan por poder desplazarse de un lugar a otro de manera sistematizada o autónoma cumpliendo con el desarrollo de una tarea puntual (Murphy, 2000).

La capacidad a resaltar es la autonomía, la cual busca que estos sistemas operen de forma eficiente en entornos donde no puedan ser guiados o tengan la menor intervención humana posible (Sánchez and Rodriguez, 2021), para esto cuenta con un sistema sensorial, un sistema de locomoción, un sistema de control y de alimentación. Con los sensores recolecta información propia del robot (velocidad de ruedas, posicionamiento del sistema) e información del ambiente (posicionamiento de objetos, distancia de obstáculos) el cual lo ayudará a tomar decisiones lógicas para su movimiento y percepción del entorno. El sistema de locomoción depende del medio en que se desplacen, estos se pueden clasificar en: terrestres, marinos y aéreos. Para el terrestre normalmente el sistema de locomoción usado son las ruedas, patas u oruga. El sistema sensorial obtiene información del ambiente y así se da una idea de los cambios hechos por los actuadores, con el sistema ultrasónico se obtienen mediciones de distancia de los objetos encontrados que nos permite tomar desiciones acertadas para la navegación.

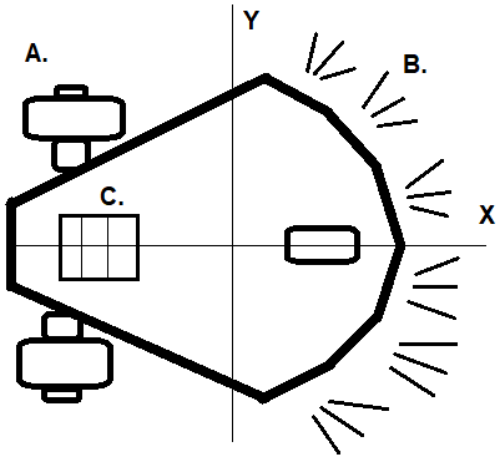


Figura 2.1: A. Sistema de locomoción. B. Sistema sensorial. C. Sistema de alimentación (Gutiérrez, 2009).

Con esto se ha cubierto parte de la misión de los robots móviles, se tiene el sistema de actuadores necesario para el desplazamiento y se cuenta con el sistema de sensores requerido para percepción o descripción del entorno estructurado a navegar. Conociendo esto, se amplía el problema hacía uno de interpretación y actuación, un comportamiento autónomo del robot móvil ya que se cuenta con la capacidad de percibir el entorno, el robot debe estar en condiciones de procesar y ejecutar movimientos a través de su sistema de locomoción para recorrer una región que pueda ser válida para navegar y no colisionar con el ambiente, cumpliendo con el objetivo de desplazarse por una ruta libre, tal como se muestra en la figura 2.2.

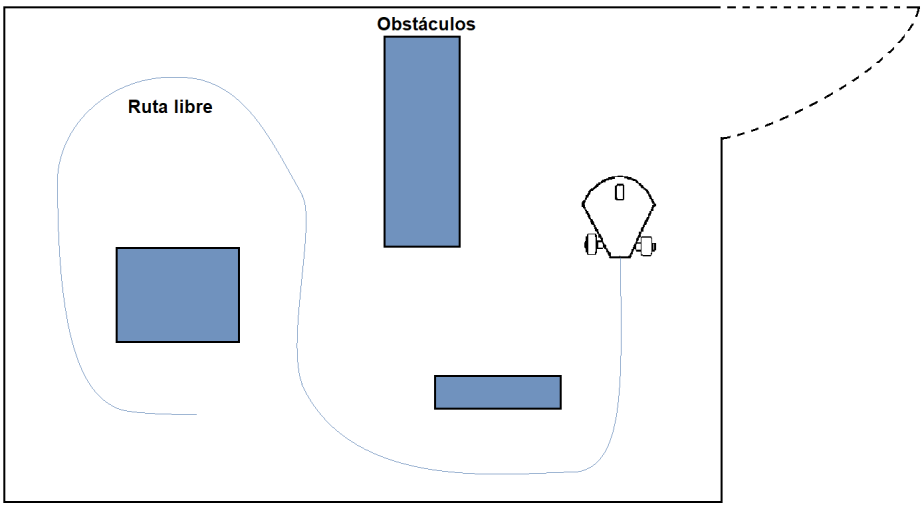


Figura 2.2: Navegación por ruta libre de un robot móvil autónomo (Gutiérrez, 2009).

Sin embargo, hace falta quien tiene el control para que los objetivos de la tarea a desarrollar se cumplan de la mejor manera. Aquí tenemos al software, que ante la información de lo sensado y la capacidad de la locomoción, es el encargado de tomar decisiones lógicas para obtener resultados ligados a la estrategia de aprendizaje que se quiere desarrollar y cumplir con los objetivos propuestos.

Por último, dependiendo de qué tan autónomo o las tareas a ejecutar, su sistema de control aumentará en complejidad y su sistema de alimentación que es el responsable de la energía del robot, será mayor a medida que aumente su autonomía (Pastor and Rodríguez, 2006; Franchi et al., 2020). Con esto, los robots móviles permiten experimentar con mayor facilidad los conceptos teóricos y algoritmos en el mundo real, aunque la mayoría de trabajos

sean experimentales hay aportes importantes en la industria y el comercio.

2.4.1. Modelado cinemático en robots móviles

Específicamente, este trabajo se enfoca en un mecanismo de tracción diferencial, es quizás el mecanismo más simple que existe para un robot móvil de contacto con el suelo. Consta de dos ruedas estándar montadas sobre el mismo eje, controladas por motores separados y una tercera rueda giratoria que mantiene el balance del vehículo, como se muestra en la figura 2.1. El mecanismo de accionamiento diferencial frecuentemente se utiliza en interiores y algunas ventajas son su fácil implementación y bajo costo, sin embargo son dificiles de controlar, y para trayectorias rectas se requiere de un control de precisión.

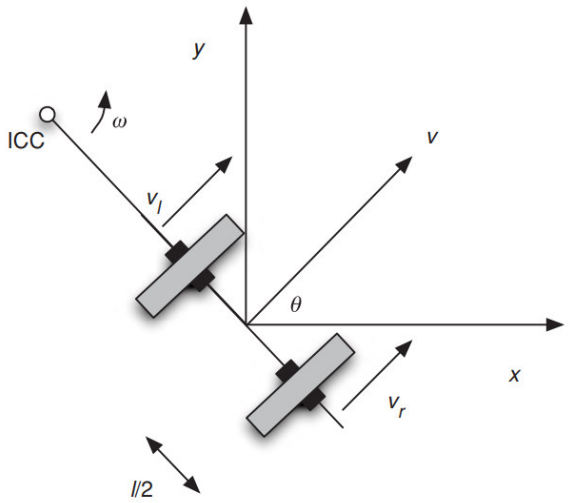


Figura 2.3: Modelo cinemático de un sistema de locomoción diferencial sobre el eje (Dudek and Jenkin, 2010).

En este tipo de locomoción no hay ruedas directrices, la traslación y rotación se da por el movimiento independiente de las ruedas de tracción al variar la velocidad relativa de estas, cambiando su punto de rotación y tomando diferentes trayectorias. En cada paso, se debe asegurar que el robot gire alrededor de su centro instantáneo de curvatura (ICC, por sus siglas en inglés), a la misma velocidad angular w y por lo tanto tenemos las siguientes definiciones:

$$w(R + \frac{l}{2}) = V_r \tag{2.1}$$

$$w(R - \frac{l}{2}) = V_l \tag{2.2}$$

donde l es la distancia entre los centros de las dos ruedas, la rueda derecha se mueve con velocidad V_r y la izquierda con velocidad V_l, R es la distancia desde el punto medio entre las dos ruedas y el ICC. Ya que V_l, V_r, w y R son funciones dependientes del tiempo, la resolución de R y w puede llegar a lo siguiente

$$R = \frac{l}{2}\frac{(V_l + V_r)}{(V_l - V_r)} \tag{2.3}$$

$$w = \frac{V_r - V_l}{l} \tag{2.4}$$

Se resaltan unos casos especiales, si $V_l = -V_r$, segun la ecuación (2.3) el radio es cero, esto hace que el robot gire sobre su propio eje. Si $V_l = V_r$, entonces el radio es infinito y el robot se desplazará en linea recta. Estas características hacen que la implementación de los sistemas mecanicos diferenciales sean de interés para navegar en espacios reducidos.

Para los casos restantes, donde V_l y V_r toman otros valores, el robot sigue una trayectoria curva rodeando un punto a cierta distancia R de su centro, cambiando la orientación y posición del mismo. Este tipo de locomoción tiene unas restricciones no holónomas, lo que significa que no existen valores para V_l y V_r tal que el vehículo pueda moverse a lo largo del eje común de las ruedas. Un vehículo con este tipo de mecanismo es sensible a la velocidad relativa de las ruedas, cualquier diferencia con la velocidad requerida puede dar como resultado una trayectoria totalmente distinta a la esperada. Para mantener el equilibrio se usa una tercera rueda giratoria, esto limita su uso a entornos estructurados, ya que son sensibles a cualquier perturbación en el suelo (Dudek and Jenkin, 2010).

2.5. Navegación en robots móviles

La navegación es una de las tareas más importantes para un robot móvil, con ella se trata de realizar una serie de desplazamientos a través de un ambiente para interactuar con

distintos elementos sin permitirle colisionar, transportándolo de forma segura a su destino final. Podemos encontrar tres categorías de navegación móvil: la navegación global es donde el robot móvil se provee previamente de información del entorno, posición de los obstáculos con respecto al eje de referencia y posición de la meta, es decir, de un entorno totalmente conocido. En la navegación local el robot móvil se encarga de identificar características dinamicas en el entorno, el enfoque de esta categoría es un enfoque reactivo ya que tiene la capacidad de controlar y ejecutar tareas de forma autónoma, puede navegar en entornos desconocidos y parcialmente conocidos. Por último, la capacidad de relacionar la posición de los diferentes elementos del entorno entre sí, teniendo en cuenta la posición actual del robot, es la navegación personal. Hace muchos años, en la navegación móvil se suele encontrar los siguientes tipos de problemas:

- Percepción del entorno: se da a través de sensores externos, banco de datos de un modelo o mapa del entorno donde se realizará la navegación.
- Planificación de la trayectoria: encuentra trayectorias sin colisionar con el ambiente ejecutando una serie de pasos que se calculan usando el mapa del entorno y realizando un procedimiento estratégico que lo lleven desde su posición inicial a su destino final.
- Seguimiento de la ruta: realiza el cambio de posición de un punto de origen a un punto destino siguiendo el camino generado efectuando el debido control a los actuadores del vehículo.

El éxito de la robótica autónoma móvil es desarrollar una estructura física que pueda desplazarse sin necesidad de intervención humana en el mundo real. Actualmente, uno de los retos más grandes para los investigadores en robótica móvil es el desarrollo de algoritmos que solucionen estos problemas de la navegación autónoma (Hong et al., 2012). Anteriormente, se han desarrollado prototipos robóticos y entornos estructurados, sin embargo esto aumenta los costos, reduce la autonomía y normalmente no son aplicables al mundo real. Para este tipo de desafios existen dos grandes grupos de algoritmos que pueden ser usados para la navegación de un entorno, estos son aquellos basados en un enfoque clásico y los de enfoque reactivo. Ambos serán desarrollados en la siguiente sección.

2.6. Métodos para la navegación de un robot móvil

Los algoritmos basados en información completa o con mapas del entorno a recorrer son llamados de enfoque clásico. Siendo lo más usados cuando aún no se habían desarrollado las técnicas de inteligencia artificial, ya que obtienen una ruta óptima si esta existe o confirma que no hay una solución posible. Los algoritmos basados en información incompleta son llamados de enfoque reactivo, son muy útiles ya que uno de los problemas más frecuentes es que no se cuenta con un mapa del ambiente que se va a navegar, suelen encontrar una mejor solución en menor tiempo aunque no siempre encuentren un resultado.

Por lo tanto, es muy importante elegir adecuadamente el método de navegación en la planificación de la trayectoria de un robot sea para implementar en un ambiente simple o complejo. A continuación, se examinan algunos ejemplos de los diferentes enfoques.

1. Enfoque clásico
    - Descomposición de celdas
    - Campo pontencial artificial
2. Enfoque reactivo
    - Algoritmo de Luciérnagas (FA)
    - Redes Neuronales Artificiales
    - Algoritmo Genético (AG)
    - Lógica Difusa (LD)

2.6.1. Descomposición de celdas

El método de descomposición de celdas se basa en estrategias geométricas de segmentación de imágenes en celdas o regiones, es de las técnicas más usadas en la planificación de trayectorias. La idea principal es definir una ruta óptima y sin colisiones para el desplazamiento de un robot móvil desde un punto inicial hasta el destino final, a través de un espacio libre de obstáculos en el entorno de navegación.

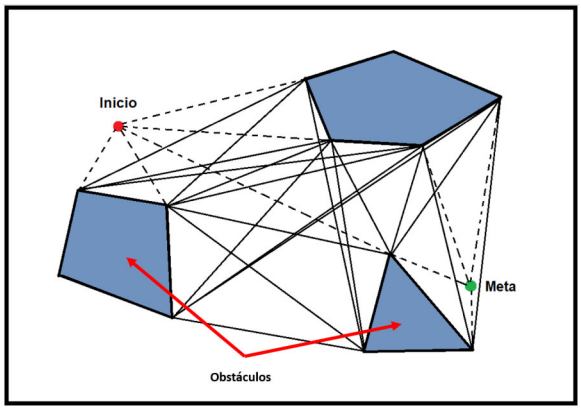


Figura 2.4: Gráfico del cálculo de distancias entre obstáculos (Patle et al., 2019).

En la figura 2.4 se puede observar que cada nodo señala regiones en el espacio libre y calculando la distancia euclidiana entre cada punto se selecciona la trayectoria más óptima. Con esto, el algoritmo identifica a partir del sistema sensorial los vértices de los obstáculos lo que le permite bordearlos satisfactoriamente, define las regiones y subdivide el espacio en regiones mas pequeñas llamadas celdas.

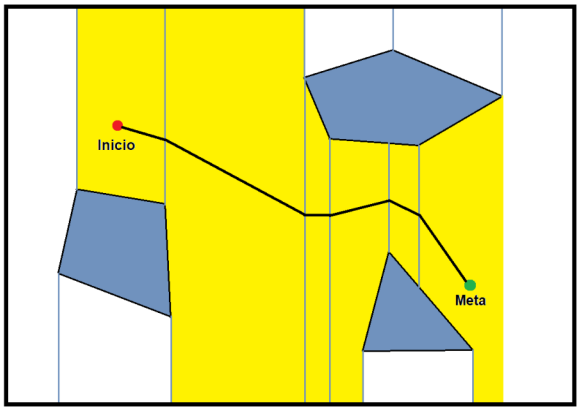


Figura 2.5: Trayectoria generada por la descomposición de celdas (Montiel et al., 2015).

Luego señala los puntos centrales de los límites entre regiones anticipando que el espacio de la ruta permita el paso del robot y selecciona ciertas celdas que cumplan con el criterio de menor distancia para facilitar la navegación. El entorno será observado en su totalidad por parte de la unidad central de control permitiendo comunicarle al robot la

ruta de navegación.

2.6.2. Campos potenciales

El método campos potenciales, introducido por primera vez como un método de evasión de obstáculos por Khatib O. (Khatib, 1986) resuelve de forma práctica los problemas de generación de trayectorias para robots móviles; propone en modelar a un robot con una carga eléctrica del mismo signo que el de los obstáculos con el objetivo de ocasionar una repulsión entre ellos, mientras que la meta se asocia con una carga eléctrica de signo opuesto para cumplir con la tarea de atraer al robot al punto destino.

2.6.3. Enfoques reactivos

Una de las principales cualidades para explorar entornos no estructurados de forma eficiente es la capacidad de reacción ante situaciones inesperadas, con la aplicación de estos algortimos lo que se busca es que el robot móvil muestre cierto grado de reactividad como el de un ser vivo.

Los enfoques reactivos, también nombrados algoritmos híbridos ya que son usados para mejorar los algoritmos de enfoques clásicos. En la actualidad, son los métodos más desarrollados, comunmente usados e investigados en el campo de la navegación robótica móvil (Sánchez and Rodriguez, 2021; Pal and Kar, 1996).

2.6.4. Algoritmo de Luciérnagas (FA)

Propuesto por Xin-She Yang (Yang, 2009), se inspira en el comportamiento intermitente de las luciérnagas. El autor plantea que los agentes de búsqueda son una población de luciérnagas que interactuan entre ellas, donde la atracción está relacionada con su brillo y la distancia de una luciérnaga con respecto a las otras. Este tipo de algortimos metaheurísticos son empleados para la optimización y búsqueda de propósito general. Exploran apropiadamente un espacio determinado siguiendo ciertos métodos de forma iterativa y usando diferentes conceptos de exploración, solucionando problemas como la trayectoria

más corta entre dos coordenadas. En cada iteración las luciérnagas cambian de posición, regidas por la siguientes reglas:

- Todas las luciérnagas son unisex, para que una luciérnaga se sienta atraída por otra independientemente de su sexo.
- La atracción de una luciérnaga es directamente proporcional a su brillo, y disminuye a medida que se aleja de otra luciérnaga, de tal forma que la de menor brillo se moverá hacia la de mayor brillo. Si no hay una luciérnaga más brillante o más atractiva que otra en particular, ésta se moverá aleatoriamente.
- El brillo de una luciérnaga está determinado por el valor de la función objetivo dada por el problema.

Es reciente el uso de esta herramienta de optimización y su aplicación se ha extendido en muchas ramas de la ingeniería, hasta la navegación de robots móviles. El diagrama de flujo de la figura 2.6, muestra como el algoritmo de luciérnagas es usado en la navegación de un robot móvil. Lo han implementado en simulación para encontrar con éxito la ruta óptima en entornos dinámicos (Brand and Yu, 2013), también con obstáculos estáticos (Hidalgo-Paniagua et al., 2017), resolviendo así los problemas presentados en la navegación: la longitud, suavidad y seguridad de la trayectoria.

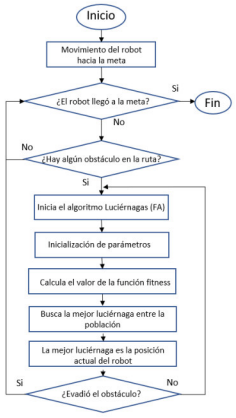


Figura 2.6: Algortimo Luciérnagas (FA) en navegación de robots móviles (Patle et al., 2019).

2.6.5. Redes Neuronales Artificiales

Cada autor da una definición propia sobre las redes neuronales artificiales y tiende a variar según el contexto en el que son usadas. Se definen como sistemas inteligentes conformados por estructuras simples e interconectadas para transmitir señales, también como modelos matemáticos desarrollados para emular el cerebro humano. En general, las redes neuronales artificiales tratan de modelar la forma de procesar la información del cerebro para comprender su funcionamiento. Hay cuatro elementos básicos en un modelo de red neuronal artificial:

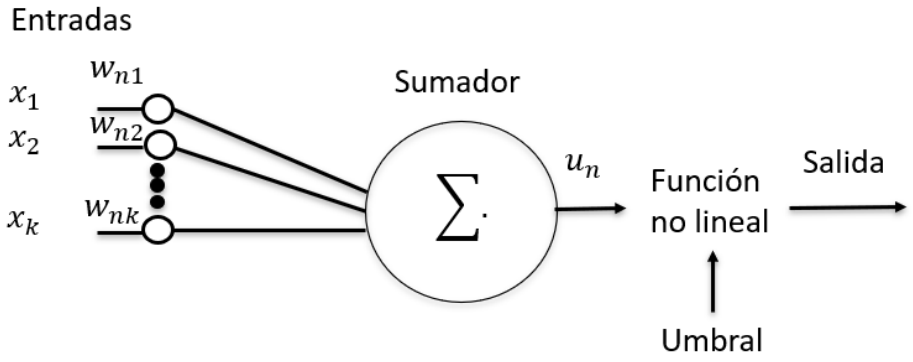


Figura 2.7: Esquema general de una neurona artificial Cuevas-Arteaga and Rostro-Gonzalez (2017).

Un conjunto de enlaces, sinapsis o pesos que pueden incrementar o inhibir el estado de activación de las nueronas adyacentes. Un segundo elemento llamado sumador, se encarga de sumar todas las entradas que han sido multiplicadas por estos pesos. Si esta suma supera un parámetro llamado Umbral la neurona se activa, en caso contrario se mantiene en reposo. En la salida de la neurona se encuentra una función no-lineal que impone un limite a la amplitud antes de extenderse a la siguiente neurona, se conoce como función de activación.

Las redes neuronales artificiales en lugar de ser programadas realizan un aprendizaje automático, ajustan sus pesos a medida que resuelven una y otra vez problemas basados en el conocimiento experimental, y así proporcionar soluciones óptimas. Se ocupan de tareas cognitivas como el reconocimiento, aprendizaje, toma de decisiones y ejecutar acciones que son los principales problemas que se dan en la navegación. Con la aplicación de una red

neuronal artificial, un robot móvil desarrolló una trayectoria sin colisiones en un entorno parcialmente desconocido evitando el obstáculo más cercano (Janglova, 2004). Además han logrado introducir una lazo de control neuronal con un enfoque reactivo para la navegación con robots móviles en tiempo real con sensores ultrasónicos (Medina-Santiago et al., 2014; Pal and Kar, 1996).

2.6.6. Algoritmos Genéticos (AG)

Los Algoritmos Genéticos fueron introducidos al campo de la informática por John Holland en 1962 (Holland, 1992), es una herramienta de optimización inspirada en la mecánica de la selección natural y la genética para encontrar las regiones más óptimas en los espacios de búsquedas y así, se encarga de optimizar problemas donde se tiene una función objetiva con parámetro específicos donde logre maximizar o minimizar su valor. El AG es un método de enfoque reactivo que requiere de poca información sobre el entorno de búsqueda, por estas características tiene un amplio uso en la navegación robótica para suavización de rutas y evasión de obstáculos (Xiao et al., 1997).

2.6.7. Lógica Difusa (LD)

Lotfi A. Zadeh, fue quien definió este concepto por primera vez en 1965 para luego ser utilizado en problemas donde hay alta complejidad y no linealidad. La lógica Difusa ha sido muy importante en el reconocimiento de patrones, clasificación de datos y toma de decisiones en el control de robots móviles, haciendo que estos comportamientos difusos se puedan integrar en un conjunto de condicionales IF-THEN para obtener el equivalente matemático de los resultados. Este método tiene características importantes para solucionar problemas en la navegación de robots autónomos, normalmente estos controladores son diseñados para solucionar una tarea a la vez, como evitar obstáculos o seguir trayectorias (Sugeno and Nishida, 1985). Actualmente, se ha unido con el uso de sensores para la navegación (Carelli and Oliveira Freire, 2003) y mejorar el aprendizaje en entornos desconocidos y así obtener una trayectoria deseada para una situación sin salida.

2.7. Redes Neuronales Pulsantes

En la tercera generación se encuentran los modelos de neuronas generadoras de impulsos, estas muestran ventajas sobre las redes neuronales artificiales tradicionales como mayor velocidad en el procesamiento de la información, menor consumo de energía y según su descripción matemática modela con más realismo a una neurona biológica. La diferencia más notoria con respecto a las redes neuronales clásicas, es que los modelos de neuronas pulsantes incluyen el concepto de tiempo tanto en sus entradas como en su salida, en lugar de vectores numéricos.

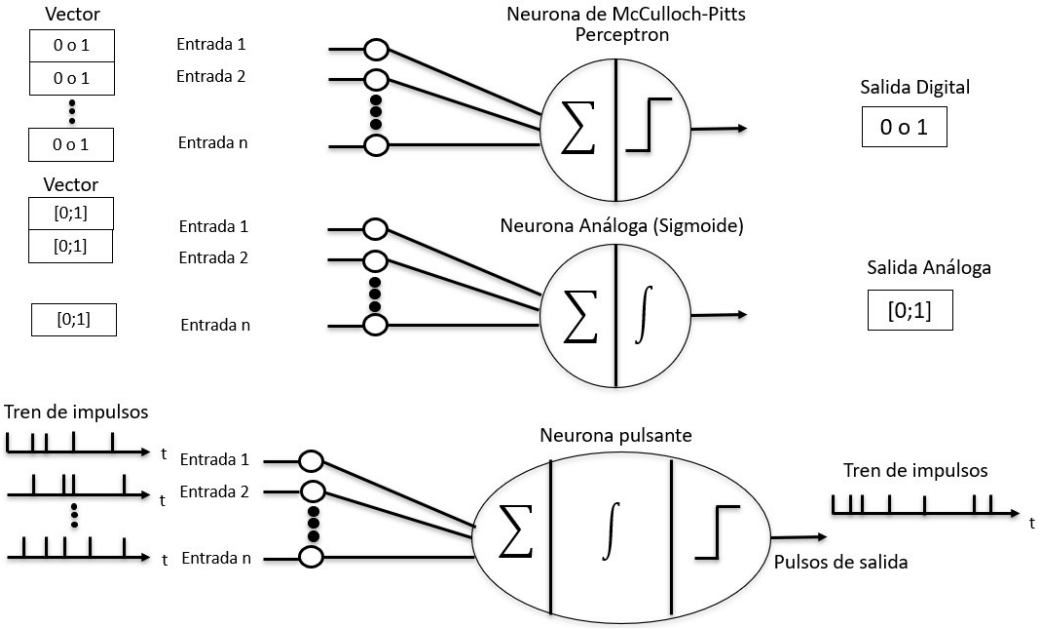


Figura 2.8: Generaciones de redes neuronales artificales (Davies, 2013).

Además cabe resaltar que este tipo de neuronas solo reciben impulsos como entrada, para codificar esta se debe tener en cuenta la cantidad de impulsos y su tiempo de ingreso. De esta manera nacen dos métodos de codificación para trenes de impulsos:

- Codificación basada en frecuencia: donde impulsos muy cercanos con respecto al tiempo representan una variable de entrada con valor alto y en la ausencia de impulsos o cuando estos se presenten esporádicamente respresenta valores bajos en dicha variable.

- Codificación basada en tiempo: la magnitud de la variable de entrada es codificada en función de la distancia en tiempo que pueda existir entre un impulso y otro.

2.8. Aprendizaje por refuerzo

Aprender a través de la interacción es la base fundamental del aprendizaje por refuerzo, la idea está inspirada en la manera en que los seres vivos aprenden. Su principio consiste en realizar una acción y esperar su reacción: si el resultado es bueno, se tiende a repetir el comportamiento y si el resultado no conviene, simplemente se evita.

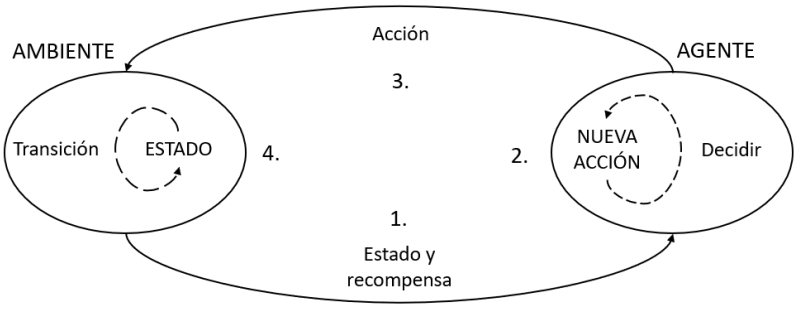


Figura 2.9: Modelo del aprendizaje por refuerzo (Torres.AI, 2021).

El modelo implementado en este aprendizaje es el de un agente que explora un entorno (como se muestra en el paso 1 de la figura 2.9) y ante una situación determinada, que la establece su estado actual y la recompensa, decide la siguiente accion a realizar (paso 2) debido a la experiencia adquirida a través del ensayo y error, ejecuta la acción favorable para el cumplimiento de sus objetivos (paso 3), esto ocasiona un efecto en su entorno situándolo en un nuevo estado y cáculando una nueva recompensa (paso 4). El aprendizaje autónomo se lleva a cabo gracias a las penalizaciones y recompensas que obtiene de sus acciones, debido a esto debe explorar todas las acciones posibles para así aprender cual es la más conveniente. Por lo cual, el agente actúa y desarrolla la mejor estrategia para llegar a la meta recibiendo la mayor recompensa posible.

Esta estrategia es usada para que un robot aprenda de sí mismo: sobre el funcionamiento de sus sensores, la precisión de sus actuadores y sobre el comportamiento propio a medida que pasa el tiempo. Para que aprenda sobre su espacio: el robot puede trazar

una trayectoria por el entorno o aprender una ruta que lo lleve a su destino, intentando coincidir con zonas que le den mayor recompensa (Lozada et al., 2020). Esto le permitirá cumplir de mejor forma sus objetivos dándole una mano al sistema de control del robot. El aprendizaje por refuerzo le permitirá al robot adaptarse a los cambios inesperados por sí solo y en conjunto reducir la programación ya que no es fácil preveer para el programador situaciones en entornos desconocidos.

Capítulo 3

Herramientas

Como se mencionó en el incio de este trabajo, la implementación de nuestro sistema de procesamiento «neuromórfico» será en la tarjeta Raspberry pi. Por lo tanto, en este capítulo se hará una descripción detallada de esta arquitectura y del funcionamiento de las herramientas necesarias para que el robot móvil realice una navegación exitosa. Inicialmente, se describe el sensor ultrasónico HC-SR04 que hace posible la exploración del entorno y luego, el Brick EV3 que se encarga de recibir los valores de salida y accionar los actuadores del sistema.

3.1. Raspberry Pi

Es una herramienta desarrollada por la Fundación Raspberry Pi (Universidad de Cambridge) para aprender de programación y electrónica, que fue lanzado al mercado en febrero de 2012 con el objetivo de que los niños puedan conseguir un ordenador portable propio, gracias a su bajo precio y así, llegar a entender el funcionamiento básico de las computadoras de una forma que no se torne aburrida. La fundación Raspberry Pi fomenta el aprendizaje de Python; es un lenguaje de programación de alto nivel con sintaxis sencilla muy útil para temas de educación, y algunos otros como C y Tiny BASIC, lo que ha permitido implementar sistemas de control y redes neuronales para robots autónomos (Urrea and Kern, 2021; Pecolt et al., 2021) con la ayuda de la visión artificial en tiempo real

(Anand and Kumawat, 2021) en la Raspberry Pi, ayudándole a esto su tamaño reducido y gran poder de cómputo.

3.1.1. Arquitectura de la tarjeta

La Raspberry Pi es un ordenador que carece de accesorios, lo cual deja al descubierto todos sus componentes sin que esto afecte el funcionamiento para el cual está diseñado. Es un dispositivo de placa única que está equipado para llevar a cabo las funciones del robot móvil (exploración, detección y evasión de obstáculos). Es el encargado de la toma de decisiones, esta pequeña computadora, cuyas dimensiones son 85.6x53.98x17mm como se observa en la figura 3.1, es de fácil transporte. Esta unidad de procesamiento es el "cerebro" de nuestro robot móvil.

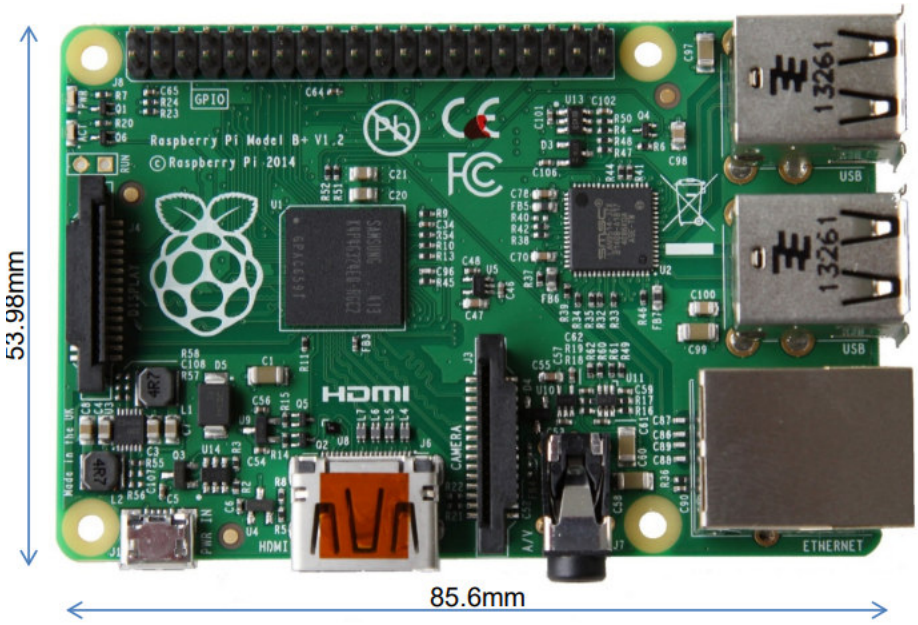


Figura 1 Raspberry PI B+

Figura 3.1: Raspberry Pi B+ (Areiza et al., 2016).

Para este proyecto se usa el modelo B plus de Raspberry Pi 3, es el último producto de esta gama, no es comparable con las computadoras actuales ya que esta es completamente básica pero muy útil para sistemas de navegación autónoma. Este dispositivo ofrece un

buen rendimiento gracias a que cuenta con un procesador ARM Cortex A53 con cuatro núcleos de 64 bits a 1,4 GHz, conexión Bluetooth 4.2 y LAN inalámbrica de banda dual 2.4/5 GHz. También uncluye puertos USB, HDMI, ETHERNET y cuenta con 40 pines de propósito general (GPIO, por sus siglas en inglés), que otorgan la posibilidad de controlar sistemas externos, donde la mayoría de ellos pueden ser personalizados a través del software como entrada/salida.

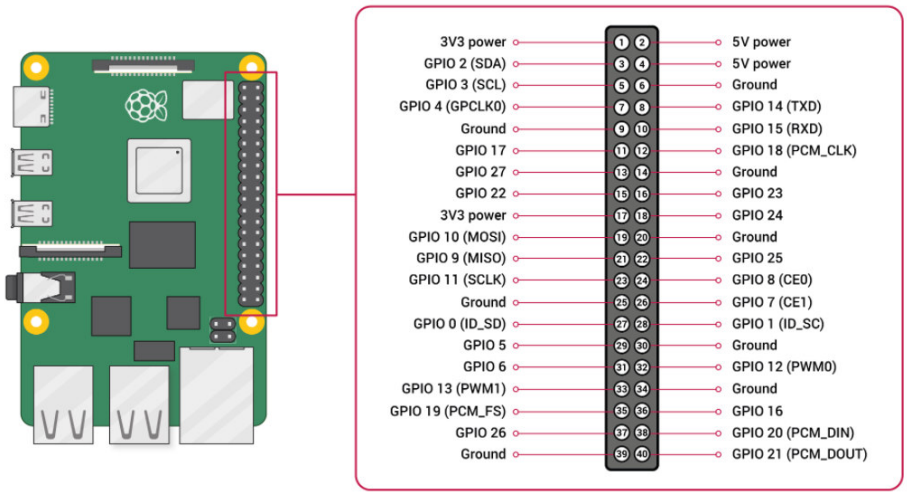


Figura 3.2: Pines de propósito general.

- Pines de alimentación: En la figura 3.2 se observa que el modelo B+ consta de dos pines de 3.3V, dos pines de 5V y 9 pines de 0v (GND) que no se pueden configurar de otra forma. Tanto los pines de 3.3V como los de 5V pueden ser usados para proporcionar energía estable a dispositivos externos como probar un LED o utilizar un sensor. Todos estos voltajes se miden respecto al pin GND.

- Pines de entrada/salida: un pin configurado como *entrada* cumple con la tarea de leer la señal recibida por la Raspberry Pi, si el voltaje de la señal enviada por el dispositivo externo es menor a 1.8V su lectura será como nivel BAJO y si está en el rango de 1.8V y 3.3V se leerá como nivel ALTO. Para un pin configurado como *salida* cumple con la tarea de enviar una señal de 3.3V para indicar nivel ALTO y de 0V para nivel BAJO. Para evitar daños en la placa se recomienda no ingresar un voltaje mayor a 3.3V en sus GPIO.

Además de las funciones básicas de entrada y salida que cumplen los pines, también se puede configurar unos pines específicos para desarrollar funciones más complejas como son:

- Pines modulación por ancho de pulso (PWM, por sus siglas en inglés): como se muestra en la figura 3.2, físicamente están disponibles los pines GPIO12, GPIO13, GPIO18 y GPIO19 para PWM y son los encargados de generar salidas analógicas (valores de 25 %,50 %,75 % del total de la salida) desde pines digitales, frecuentemente usado para controlar motores de robots, luces, entre otros.

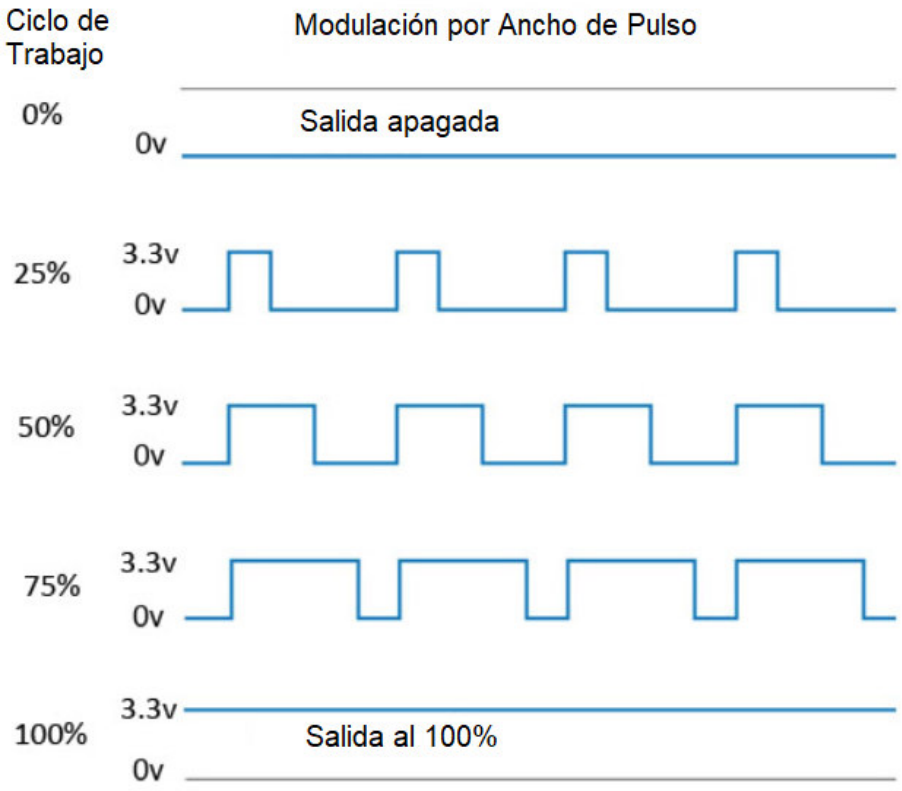


Figura 3.3: Ciclo de trabajo de PWM.

- Pines Universal Asynchronous Receiver/Transmitter (UART): proporciona una comunicación en serie con otros dispositivos. El pin RX(GPIO15) puede ser configurado para recibir datos serie y el pin TX(GPIO14) para transmitir datos serie que han sido enviados de dispositivos series externos.

- Pines I2C: es un protocolo que le permite a la Raspberry Pi comunicarse con un dispostivo compatible con el ciruito integrado. Este estandar exige a los dispositivos tomar roles de maestro-esclavo y se trata de un bus de solo dos hilos de comunicación con baja velocidad, una línea de reloj (SCLk, por sus siglas en inglés) y una linea de datos (SDA, por sus siglas en inglés). Funciona transmitiendo datos en ambas direcciones donde la velocidad de transferencia de datos es controlada por el pin SCLk.

- Pines Serial Peripheral Interface (SPI): es un protocolo síncrono que permite que uno o más dispositivos periféricos se comuniquen con la Raspberry Pi recibiendo y transmitiendo información al mismo tiempo usando dos canales diferentes en el mismo cable. Además, como el I2C es un protocolo serie utilizado para la comunicación maestro-esclavo, siendo SPI usada cuando se necesita mayor velocidad. Cuando la Raspberry Pi envía datos a un dispositivo se utiliza el pin Master Out Slave In (MOSI) para el caso contario, donde el dispositivo se quiere comunicar con la Raspberry Pi se utilizará el pin Master In Slave Out (MISO). Los datos se sincronizan mediante una señal de reloj (SCLK), el pin GPIO11 se configura para esta tarea. A continuación los pines que se necesitan para la comunciación SPI:
  • Serial Clock (SCLK): señal del reloj, todas las señales se sincronizan respecto a esta.
  • Master Out Slave In (MOSI): Este pin es utilizado para la salida de datos del maestro, entrada de datos del esclavo.
  • Master In Slave Out (MISO): Este pin es utilizado para la entrada de datos del maestro, salida de datos para el esclavo.
  • Chip Enable (CE): se necesita configurar un pin CE para cada esclavo (o dispositivos periféricos) a usar en el momento, por ejemplo: para transmitir o recibir datos del primer dispositivo se activa la señal CE1, para el segundo dispositivo se activa la señal CE2, etc. Por defecto se tiene dos pines CE, pero se puede configurar como CE algun GPIO disponible.

Para ser usado como una computadora común hay que instalar y ejecutar su sistema operativo, en este caso se usará *Raspbian*. A diferencia de las computadoras convencionales, que usan sistemas operativos creados en un ambiente reservado usando técnicas patentadas; software de código cerrado, las Raspberry Pi están diseñadas para ejecutar una versión

ligera de un sistema operativo basado en Linux y específicamente compilados para el procesador ARM; este sistema operativo es de código abierto, donde su código fuente puede ser descargado y modificado, lo que ha permitido ser adaptado para la Raspberry Pi. Como se había mencionado anteriormente, el sistema creado por la fundación para este tarjeta de procesamiento es la versión *Raspbian*, una actualización del sistema operativo *Debian.*

3.1.2. Instalación del Sistema Operativo

Para empezar con la instalación del Sistema Operativo se debe contar con los sigientes accesorios externos a la Raspberry Pi:
Una fuente de alimentación 5V/2.5A con un cable micro USB de buena calidad, muy importante su estabilidad para el correcto funcionamiento de la placa y cada uno de sus periféricos. Este modelo cuenta con detección para baja alimentación y sobrecalentamiento; el circuito de alimentación de las Raspberry Pi no permite tensiones por debajo de la establecida, cuando cae por debajo de 4.65V se muestra un cuadro pequeño con un aviso de "Batería baja" en una de las esquinas del escritorio y el LED de alimentación se apaga, también hay una señal similar en pantalla cuando la temperatura de la placa supera los 85°C. Un monitor de computadora o televisión que cuente con entrada HDMI para mejores resultados. Un mouse y teclado USB estándar o inalámbricos. Por último una tarjeta micro SD, se recomienda de almacenamiento un mínimo de 8GB y el uso del programa *Raspberry Pi Imager* para instalarle el sistema operativo.

Una vez se haya bajado el sistema operativo a la tarjeta, se inserta al puerto micro SD de la Raspberry Pi y esta se conecta a la corriente para empezar la instalación. La forma más práctica es realizar el proceso desde la Raspberry Pi con un monitor conectado por HDMI, pero también se puede hacer desde otro ordenador conectándose por el protocolo SSH a la Raspberry Pi (activo por defecto). Para lograr la conexión SSH por primera vez, la Raspberry Pi debe estar conectada vía Ethernet con el router.

Lo primero en la instalación es la configuración de la raspberry Pi como se muestra en la figura 3.4, se puede hacer en el instante o después a través del comando raspi-config. Así configurar un teclado (si es necesario), cambiar de contraseña si desea mayor seguridad, por defecto el usuario es "Pi" y contraseña "raspberry". Además, permite la habilitación de opciones de interfaces (para visualizar Raspberry Pi desde otro ordenador), configuración

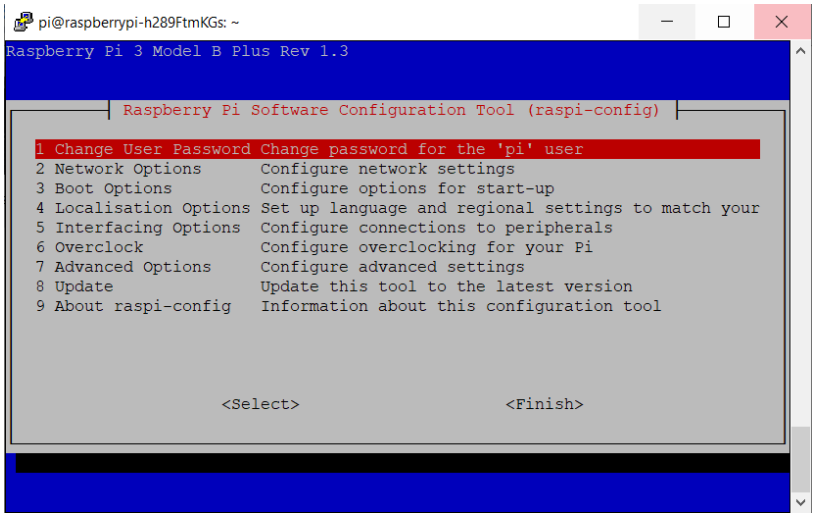


Figura 3.4: Configuración inicial de Raspberry Pi 3 Model B plus (Areiza et al., 2016).

de zona horaria y modificar idioma. Por último, realiza una actualización de Raspbian y los paquetes con que está integrado, para que estos sean descargados es necesaria la conexión vía Ethernet de la Raspberry Pi con el router.

3.1.3. Entorno Gráfico

Raspbian viene con el entorno de escritorio gráfico Lightweight X11 Desktop Environment (LXDE) instalado (figura 3.5); es un entorno de código abierto disponible para plataformas como Linux. Su mejores cualidades son la alta velocidad y el bajo consumo de energía, gracias a esto se puede ejecutar en CPUs de bajo rendimiento como es el caso de la Raspberry Pi. LXDE permite la ejecución rápida de aplicaciones con el uso del internet y soporta muchas arquitecturas de procesadores como Intel y ARM.

3.1.4. Python y Raspberry Pi

Python es uno de los programas que hacen parte de Raspbian, así como IDLE; un entorno de desarrollo integrado útil para programar en lenguaje Python. Aunque este

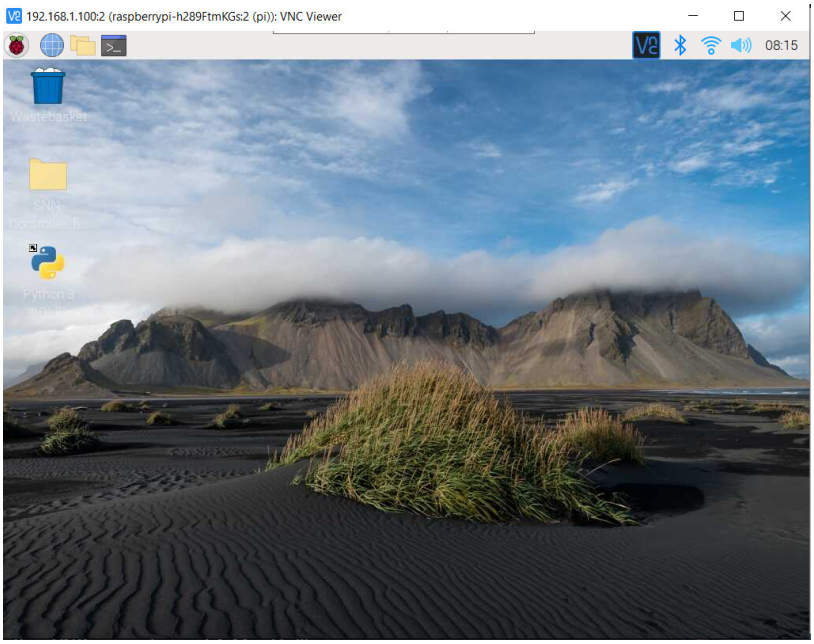


Figura 3.5: Entorno gráfico del Raspberry Pi.

lenguaje de programación tenga fama de fácil comprensión no se le debe restar importancia, es usado en múltiples aplicaciones de cálculos científicos, análisis de datos (Guedes et al., 2022), resolver problemas de optimización (Kuroki, 2021) y con su ámplia colección de bibliotecas se puede realizar muchos proyectos propuestos.

3.2. Herramientas para la validación de resultados

A continuación, se describe el funcionamiento de dos herramientas importantes para la implementación del robot móvil autónomo, una de ellas es el sensor ultrasónico HC-SR04 y la otra herramienta es el Lego Mindstorm EV3 (Brick, motores large) utilizadas para validar los resultados.

3.2.1. Sensor ultrasónico HC-SR04

El sonido está compuesto por ondas que viajan a través de un medio (el aire para este caso) y la cercanía de estas ondas entre cada una de ellas definen la frecuencia. El sonido que está por debajo de la frecuencia audible para el ser humano (menores a 20Hz) se conoce como "infrasonido" y los sonidos de alta frecuencia (mayores a 20kHz) se conocen como "ultrasonido". El sensor ultrasónico, también conocido como transductor ultrasónico básicamente se trata de un transmisor, receptor y un circuito de control que están diseñados para detectar la proximidad de un objeto que puede estar a pocos centímetros o a varios metros, empleando la reflexión del pulso ultrasónico. Se utilizan ondas ultrasónicas porque son inaudibles para el oído humano y son relativamente precisas a cortas distancias. También, se podrían usar ondas a menor frecuencia para cumplir con esta tarea pero precisamente lo que no se quiere es causar pertubaciones, se tendría un robot sonando cada cierto tiempo.

El transmisor emite un sonido, enviando ráfagas cortas de ondas ultrasónicas, donde estas se reflejan en un objeto sólido y el eco producido por este rebote es detectado por el receptor del sensor, como se muestra en la figura 3.6. El circuito de control se encarga de convertir esta señal de retorno (ondas sonoras) a señal eléctrica nuevamente, calcular el tiempo transcurrido desde que se envío por el transmisor hasta que lo captó el receptor, para posteriormente utilizarlo junto con el valor estimado de la velocidad del sonido en el cálculo de la distancia del objeto. El sensor ultrasónico HC-SR04 cuenta con unas dimensiones 45mmx20mmx15mm, tamaño conveniente para la implementación en robots móviles autónomos para navegar en una plataforma robótica (Pecolt et al., 2021) o en terrenos difíciles de explorar (Oltean, 2019). Cuenta con tecnología "sin contacto" (no necesita contacto físico con el objeto a medir), tiene una frecuencia de funcionamiento de

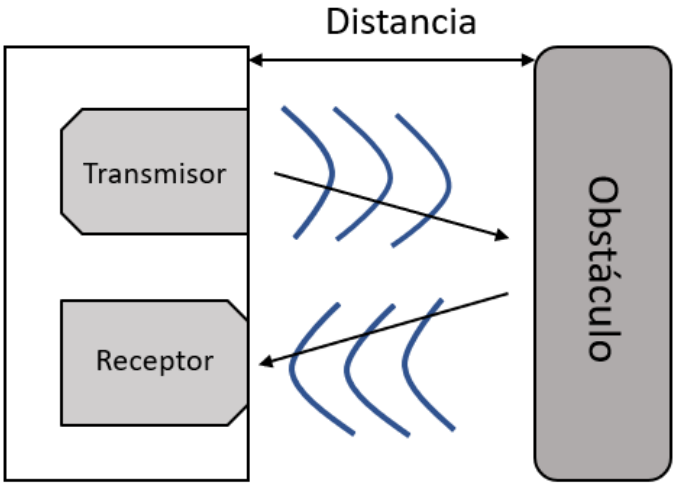


Figura 3.6: Descripción gráfica del funcionamiento del sensor HC-SR04.

4MHz detectando objetos dentro de un ángulo de 30° a una distancia mínima de 3cm y máxima de 400cm aproximadamente, consumiendo un poco menos de 15mA y consta de los siguientes 4 pines:

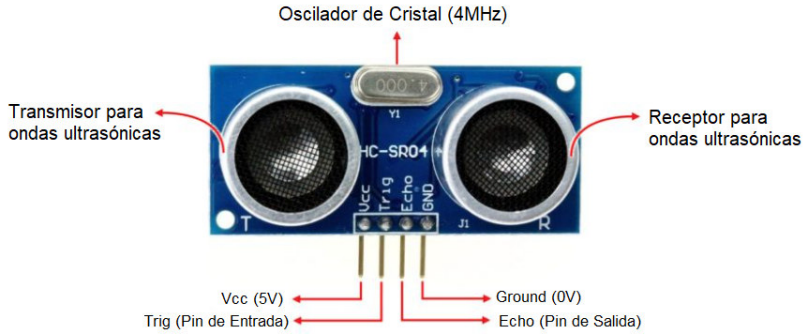


Figura 3.7: Descripción de las partes del sensor HC-SR04.

- VCC: fuente de alimentación del módulo que idealmente funciona a 5V.
- Pin Trig: es quien inicia la medición enviando un pulso ultrasónico.
- Pin Echo: se establece en alto (5V) durante el periodo de tiempo que las ondas salen del transmisor, rebotan y llegan al receptor.
- GND: pin conectado a 0V, con respecto a este pin se miden los 5V de VCC.

3.2.2. Lego Mindstorms EV3

A continuación, se puede observar en la figura 3.8 el robot móvil construido con el Brick y dos motores grandes de Lego utilizado para la implementación y validación del sistema de navegación autónoma.

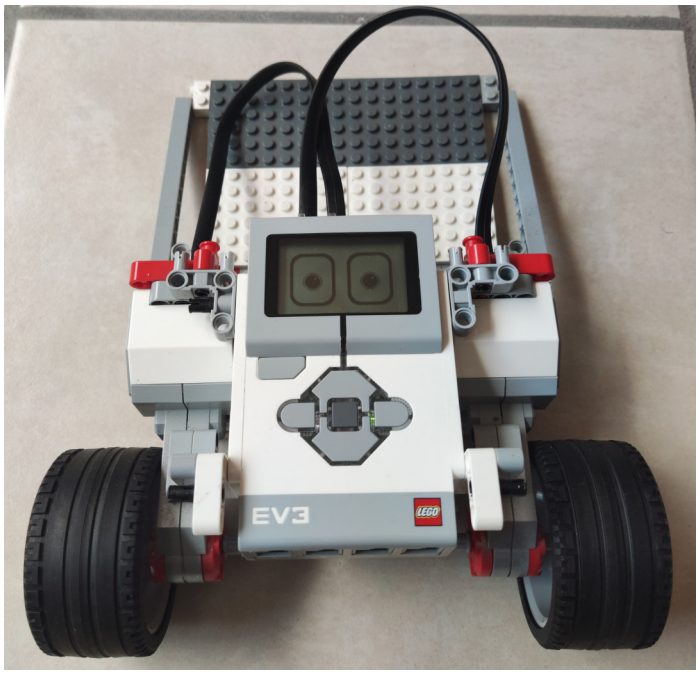


Figura 3.8: Robot móvil utilizado para la validación del sistema.

El Brick EV3 es un ordenador programable compacto usado en este proyecto como centro de control y fuente de potencia del robot con el cual se va a controlar dos de sus servomotores (paso a paso) grandes. Tiene disponible cuatro puertos de salida rj12 donde dos se han usado para conectar los motores al Brick EV3 y cuatro puertos de entrada para conectar cada uno de sus sensores (luz, giro, de contacto, ultrasónico, temperatura) si fuese necesario. Cuenta con un procesador ARM9, es un producto de hardware y software integrado pero con las herramientas correctas se puede ingresar al firmware de Lego para usar otras interfaces y lenguajes de programación externos como Java y C++. Tiene distintas formas de recibir datos desde una computadora u ordenador externo: un puerto mini USB, WIFI (usando un dongle) y Bluetooth. Además, puede ser programado de modo *directo*; es cuando los programas son ejecutados desde una computadora externa y sus acciones son

enviadas al Brick EV3 vía USB, Bluetooth o WIFI. Tiene como ventaja una comunicación que no se vea interrumpida y así poder ver los cambios realizados de forma rápida en la ejecución del robot. Cuando los programas son descargados al Brick EV3 y luego ejecutados se está programando con el modo *compilado*. Cuenta con la ventaja de no tener retraso en cada línea de codigo ejecutada por la comunicación con la computadora, es importante para proyectos donde la velocidad de respuesta es prioridad.

Figura 3.9: Brick del Lego Mindstorms EV3.

El Brick EV3 tiene una fuente de alimentación de 9V entonces con respecto a este voltaje será la potencia total con la que trabajan sus motores. Cuando se recibe un valor numérico de 100, este corresponde a que el motor seleccionado trabaje con la potencia total, ya que se hace un escalamiento. Por lo tanto, si se necesita que el motor use toda su potencia pero que gire en sentido contrario debe recibir el valor negativo de 100. Es una herramienta que ha sido usada para la comprensión de algoritmos de optimización clásicos y metaheurísticos (Zaldivar et al., 2021), el aprendizaje de lenguajes de programación, sistemas de control (Zhang and Wan, 2020), e inteligencia artificial a nivel educativo.

En el siguiente capítulo se describe como fueron utilizadas las herramientas mencionadas (Raspberry Pi, sensor HC-SR04, Lego Mindstorms EV3) para la validación del sistema de navegación autónoma.

Capítulo 4

Implementación y validación

Este capítulo consta de la descripción del sistema implementado, cómo se adaptaron los sensores a la Raspberry Pi y la comunicación de esta con el Brick EV3 a través de Matlab. Además de, las pruebas necesarias para la validación del sistema de navegación autónoma.

4.1. Descripción del sistema

El robot móvil está compuesto por varios subsistemas como se puede observar en la figura 4.1, donde se encuentra un sistema de visión, un sistema de control, un sistema de comunicación y un sistema de manejo. El sistema de visión se encarga de la recolección de datos sobre el entorno donde se encuentra. El sistema de control recibe las lecturas de los sensores, procesa la información a través de la red neuronal pulsante y genera las velocidades correctas para el siguiente movimiento. El sistema de comunicación obtiene la salida del sistema de control mediante Wifi, traduce la información a instrucciones compatibles con el Brick EV3 y las envía vía Bluetooth. Por último, el sistema de manejo recibe y ejecuta estas instrucciones con el fin de darle una nueva posición al robot móvil.

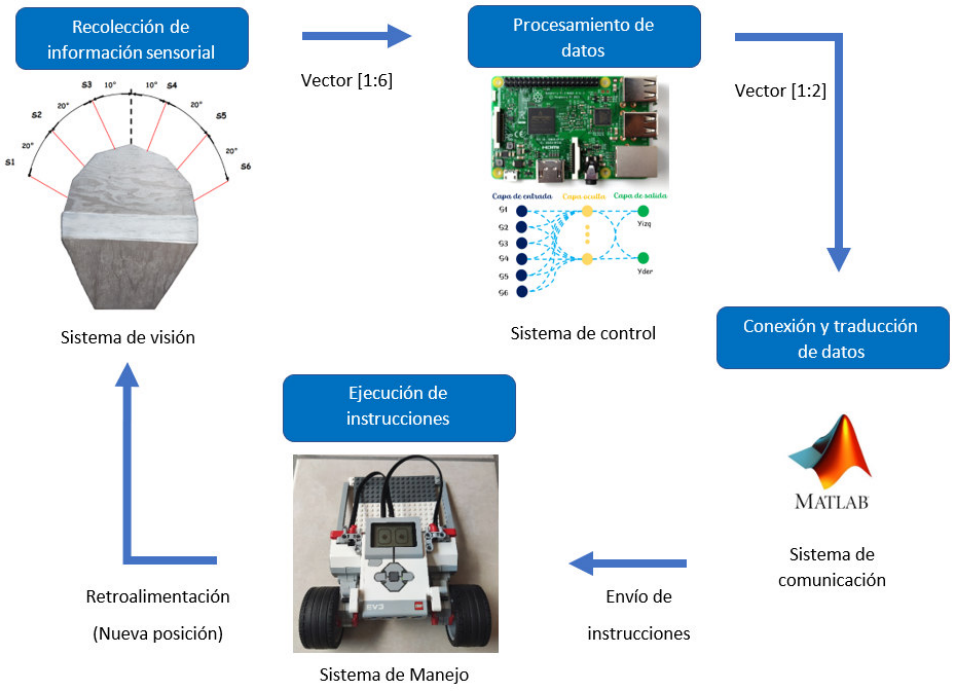


Figura 4.1: Diagrama general del sistema.

4.1.1. Topología de la Red Neuronal Pulsante

La tarea a cumplir de la red es decidir cúal es la dirección que debe tomar el robot móvil para evitar la colisión con el obstáculo detectado. Para esto, se tiene una capa de entrada con seis neuronas asignadas para la lectura de cada sensor como se muestra en la figura 4.2. El requisito inicial es obtener los valores de los seis sensores, estos son normalizados en escala de 0 a 1 lo cual representa la distancia a la que se encuentra el obstáculo, siendo 1 la cantidad que representa un obstáculo más lejano y 0 para un obstáculo más cercano.

Estos datos se procesan en nuestra red neuronal pulsante entrenada, en las neuronas de salida $Yizq$, $Yder$ se obtiene valores entre 0-100 que corresponde a la velocidad que será enviada a cada motor (ver figura 4.3). Por último, se recibe esta información en el workspace de Matlab; quien realiza la comunicación entre el "cerebro" del robot (Raspberry Pi) y el Brick EV3, para el accionamiento de los motores.

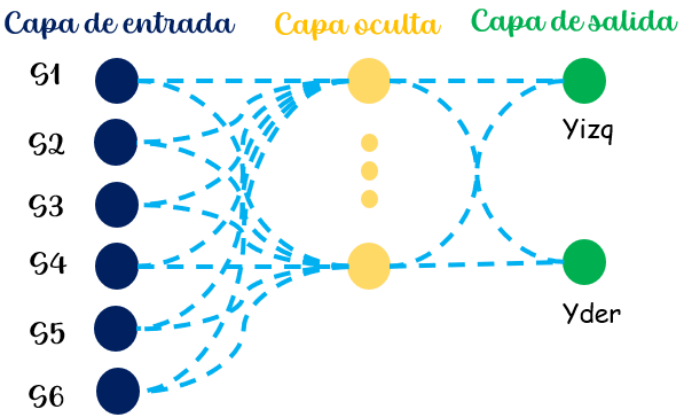


Figura 4.2: Topología de la red (Bing et al., 2019).

```
pi@raspberrypi-h289FtmKGs:~/Desktop/SNN-Controller_final/Controller $ /usr/bin/p
ython3 /home/pi/Desktop/SNN-Controller_final/Controller/controllerSNN_final.py
[1.0, 0.28049707412719727, 0.3574497699737549, 1.0, 1.0, 1.0]
[73.18022870012324, 65.95817662290533]
```

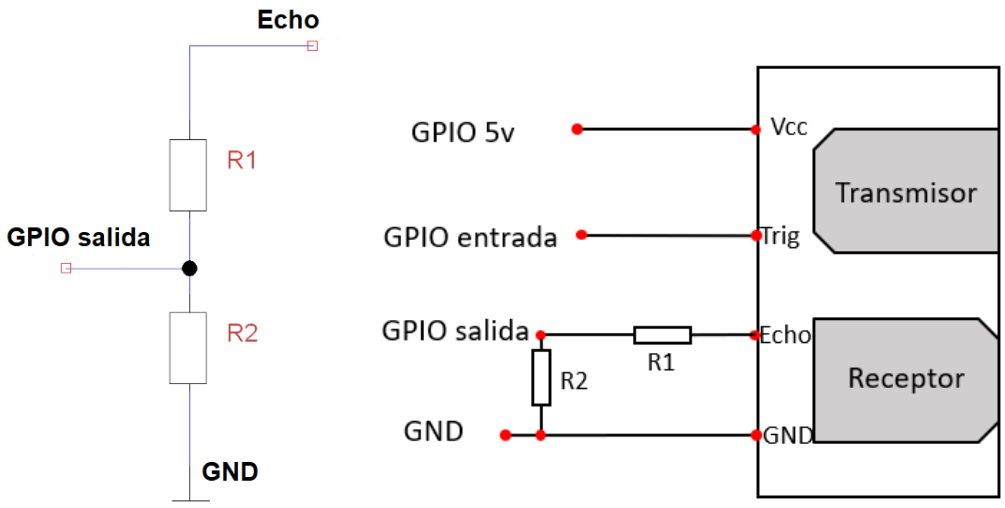
Figura 4.3: Datos de entrada y salida de la red tomadas en Python.

4.1.2. Regla de aprendizaje por refuerzo enfocada a la navegación del robot móvil

El objetivo que se quiere cumplir al aplicar la regla de aprendizaje R-STDP (donde STDP traduce una acumulación más rápida del potencial de la neurona postsináptica, lo que ocasiona que se dispare antes, cuando recibe un pico presináptico), es el de calcular una recompensa de acuerdo a la experiencia, que representa la diferencia de la salida obtenida con la salida deseada después de cada lectura de los sensores, a diferencia de otras reglas, la implementada le asigna una recompensa individual a cada sinapsis y no de forma global. Esta recompensa es usada para fortalecer las conexiones sinápticas de la red, también se define como un ajuste que determina si la salida de la red neuronal pulsante debe aumentarse o reducirse para obtener la salida deseada.

4.1.3. Adaptación de sensores a la Raspberry Pi

A continuación, veremos la adaptación de los sensores HC-SR04 a la Raspberry Pi. Antes de realizar las conexiones se debe tener en cuenta que, como vimos en el capítulo anterior el sensor ultrasónico funciona a 5 V a diferencia de la Raspberry Pi que funciona a 3.3 V, por esta razón se debe realizar un convertidor de nivel con algunas resistencias de valores específicos (1k y 2k para este caso) a la salida del pin Echo como se muestra en la figura 4.4, para ser leído por el pin que sea configurado como entrada en la Raspberry Pi.



(a) Convertidor. (b) Conexiones con el sensor.

Figura 4.4: Convertidor de nivel implementado.

La Raspberry Pi proporciona una señal de disparo en el Pin Trig de 10 μs de duración a un nivel HIGH. Eventualmente, el modulo transmitirá ocho ráfagas ultrasónicas de 40 KHz como se observa en la figura 4.5, si hay algún obstáculo interrumpiendo el viaje de las ondas ultrasónicas estas se reflejarán en él para así, recibir la señal que regresa por el pin Echo durante un tiempo estipulado.

Se obtiene la diferencia entre el tiempo de transmisión y recepción de la señal, considerando que la señal recorre dos veces la distancia a medir y el valor de la velocidad del sonido (343 m/s), con la siguiente ecuación se puede saber la distancia entre el sensor y el obstáculo.

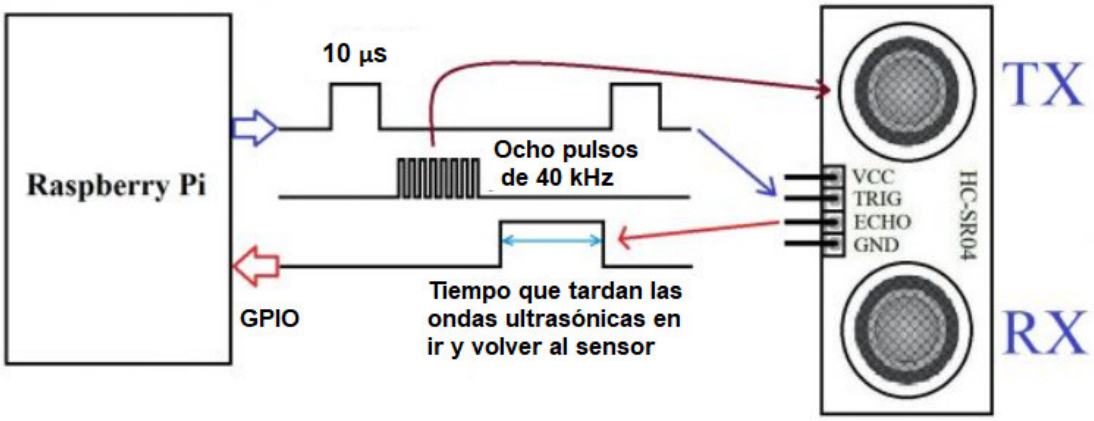


Figura 4.5: Funcionamiento Raspberry Pi y Sensor HC-SR04.

$$Distancia = \frac{VelocidadSonido * Tiempo}{2} \tag{4.1}$$

Se deben tener ciertas consideraciones al medir distancias con este tipo de sensores, como el sonido naturalmente se desplaza en todas las direcciones, cuán más lejos se haya desplazado el pulso desde el origen se habrá extendido en una zona más amplia, como sucede con el haz de luz de la linterna. Por esta razón, este tipo de sensores no se clasifican para un área en general sino para un ángulo de acción, 30° para el sensor ultrasónico HC-SR04 (ver figura 4.6).

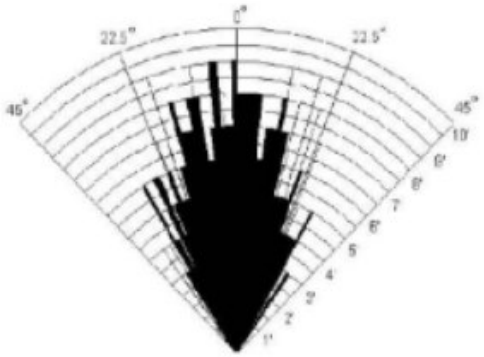


Figura 4.6: Prueba de rendimiento.

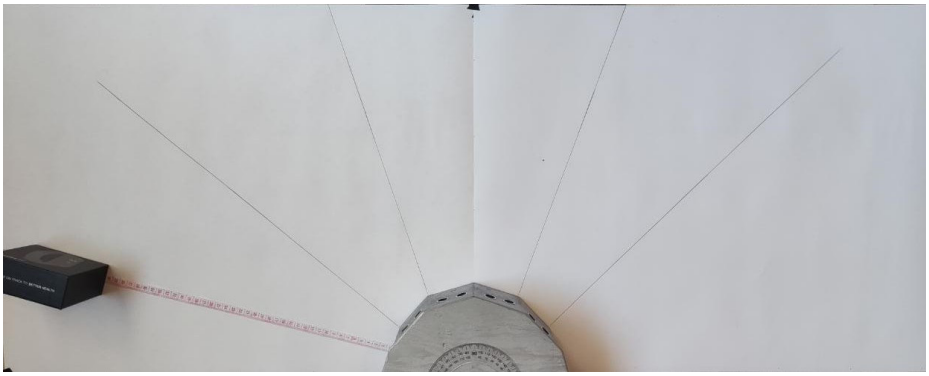
A continuación, se han realizado una serie de pruebas para verificar si los objetos que

se encuentren en el área frontal del robot son detectados de manera exitosa. Además, se necesita saber cuál es la distancia mínima real en la que un objeto es detectado; ya que para la realización de una correcta exploración se necesita que los sensores detecten obstáculos a una distancia entre 5-130 cm.

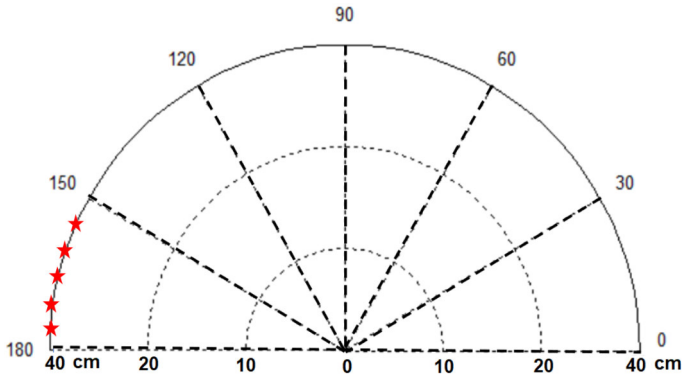
Las pruebas para validar el funcionamiento de los sensores ultrasónicos consiste en ubicar el obstáculo a diferentes distancias y ángulos con respecto a estos, tal como se muestra en las figuras 4.7 a 4.12 y obtener la lectura entregada por cada uno de los sensores. La máxima distancia con la que se trabajó en este experimento fue de 40 cm y el objeto usado como obstáculo es una caja rectangular de base 4x10 cm y altura de 10 cm. En las siguientes figuras *A* 4.7 a 4.12, se muestra el obstáculo dentro del rango de cada sensor (S1 a S6) con diferentes distancias, en las figuras *B* los distintos ángulos y en las tablas 4.1 a 4.6 se entregan los detalles de los resultados obtenidos, el error encontrado en cada una de las mediciones fue calculado con la siguiente ecuación.

$$\%Error = \mid \frac{DistanciaReal - DistanciaMedida}{DistanciaReal} \mid * 100 \tag{4.2}$$

A continuación, se observan seis mediciones, enumerándolos como sensor 1 hasta sensor 6 en el sentido de las manecillas del reloj.



(a) Objeto en el rango de S1.

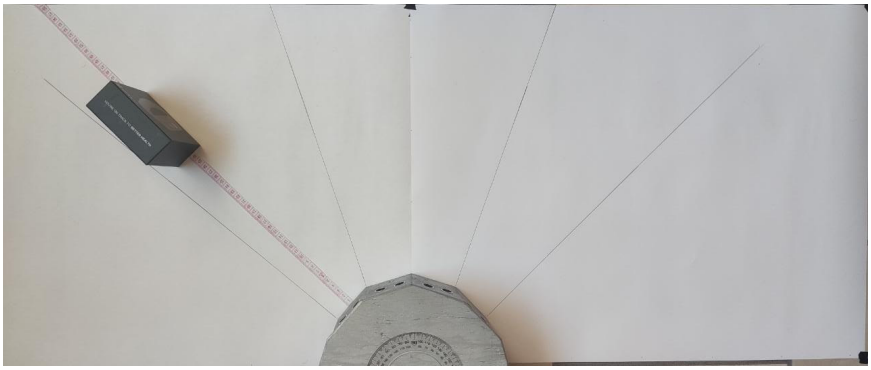


(b) Ángulos de medición para S1.

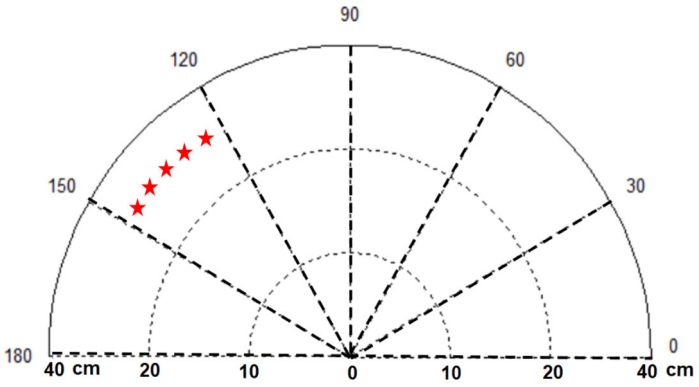
Figura 4.7: Detección del objeto a 40 cm para el sensor 1.

Tabla 4.1: Medición de Obstáculos a 40 cm para sensor 1.

Ángulo	Lectura S1 (cm)	Error (%)
155°	40.927	2.317 %
160°	40.658	1.645 %
165°	39.952	0.12 %
170°	40.570	1.425 %
175°	40.852	2.13 %



(a) Objeto en el rango de S2.

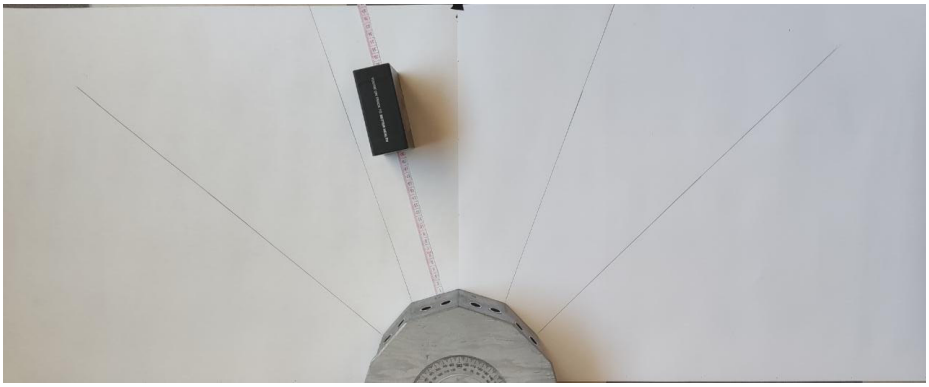


(b) Ángulos de medición para S2.

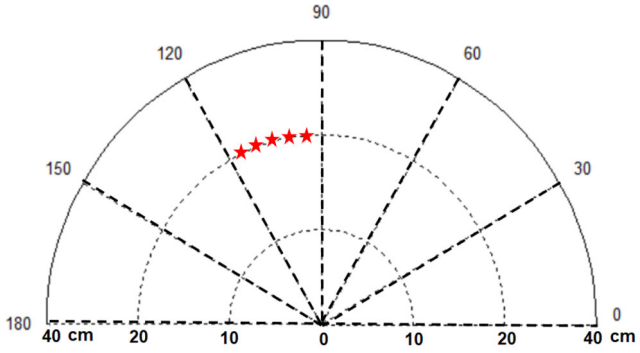
Figura 4.8: Detección del objeto a 30 cm para el sensor 2.

Tabla 4.2: Medición de Obstáculos a 30 cm para sensor 2.

Ángulo	Lectura S2 (cm)	Error (%)
125°	30.965	3.216 %
130°	30.723	2.41 %
135°	30.234	0.78 %
140°	31.570	5.23 %
145°	31.345	4.483 %



(a) Objeto en el rango de S3.

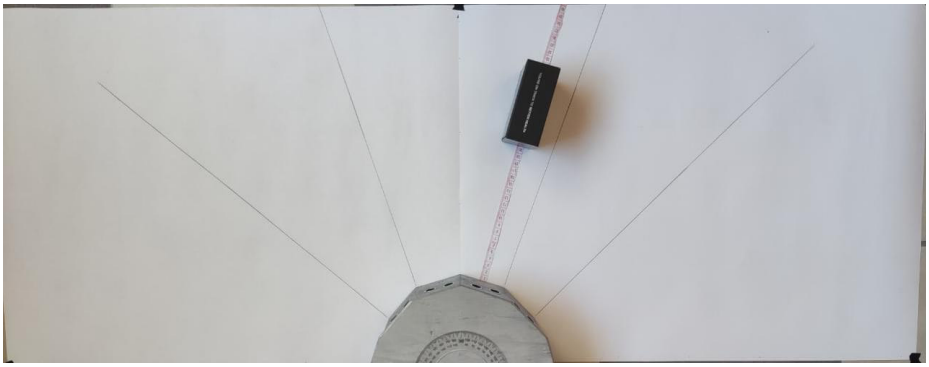


(b) Ángulos de medición para S3.

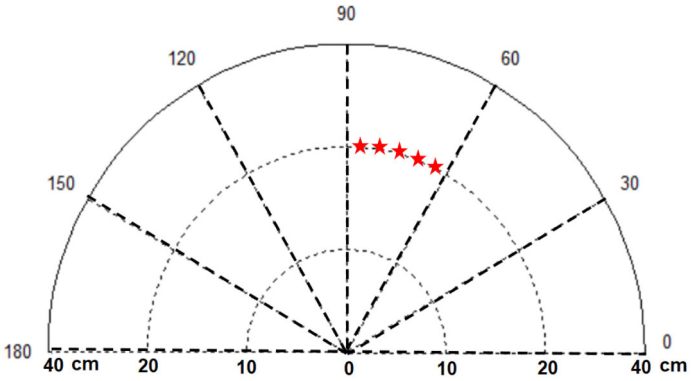
Figura 4.9: Detección del objeto a 20 cm para el sensor 3.

Tabla 4.3: Medición de Obstáculos a 20 cm para sensor 3.

Ángulo	Lectura S3 (cm)	Error (%)
95°	20.456	2.28 %
100°	20.323	1.615 %
105°	19.937	0.315 %
110°	20.437	2.185 %
115°	20.656	3.28 %



(a) Objeto en el rango de S4.

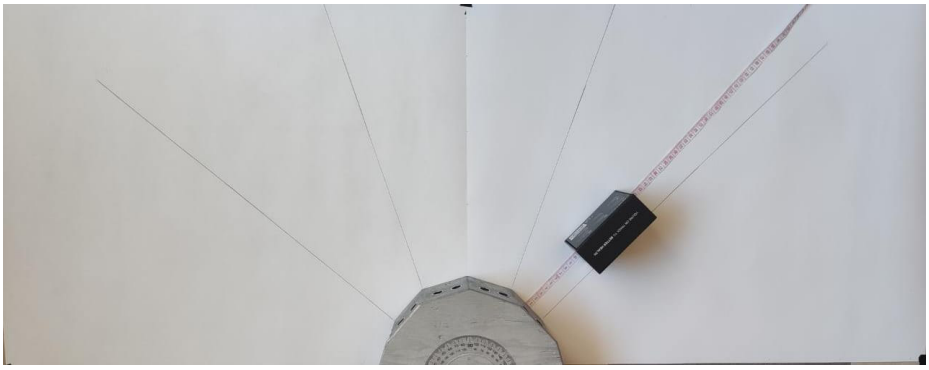


(b) Ángulos de medición para S4.

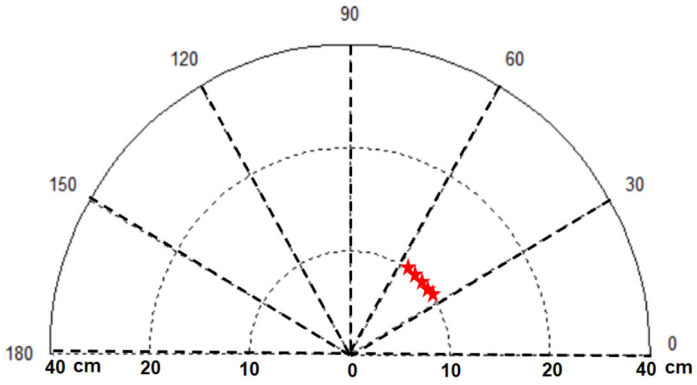
Figura 4.10: Detección del objeto a 20 cm para el sensor 4.

Tabla 4.4: Medición de Obstáculos a 20 cm para sensor 4.

Ángulo	Lectura S4 (cm)	Error (%)
65°	20.453	2.265 %
70°	20.356	1.78 %
75°	20.284	1.42 %
80°	20.401	2.01 %
85°	20.542	2.71 %



(a) Objeto en el rango de S5.

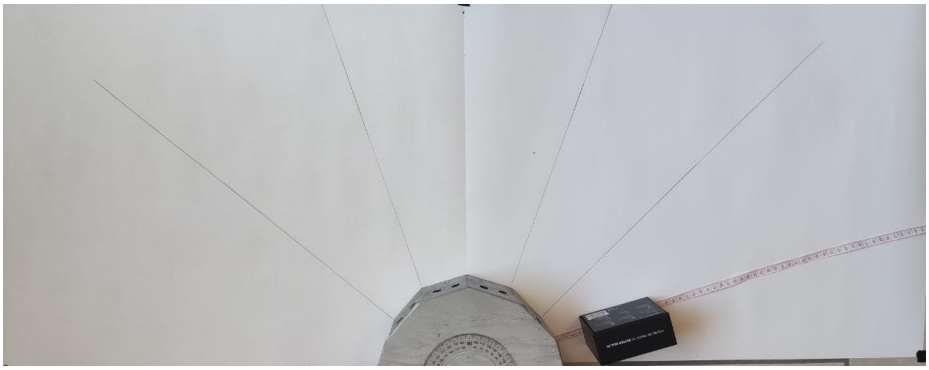


(b) Ángulos de medición para S5.

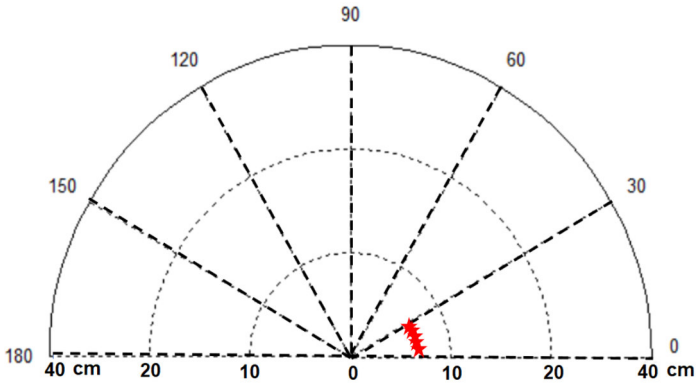
Figura 4.11: Detección del objeto a 10 cm para el sensor 5.

Tabla 4.5: Medición de Obstáculos a 10 cm para Sensor 5.

Ángulo	Lectura S5 (cm)	Error (%)
35°	10.197	1.97 %
40°	10.102	1.02 %
45°	10.051	0.51 %
50°	10.112	1.12 %
55°	10.134	1.34 %



(a) Objeto en el rango de S6.



(b) Ángulos de medición para S6.

Figura 4.12: Detección del objeto a 5 cm para el sensor 6.

Tabla 4.6: Medición de Obstáculos a 5 cm para sensor 6.

Ángulo	Lectura S6 (cm)	Error (%)
5°	5.132	2.64 %
10°	5.083	1.66 %
15°	5.037	0.74 %
20°	5.073	1.46 %
25°	5.179	3.58 %

De igual manera, se hicieron mediciones con el objeto a distancia menores de 5 cm, corroborando lo que nos dice el fabricante del sensor qué su lectura mínima es de 3 cm aproximadamente. Por lo tanto, si el objeto se encuentra a una distancia menor, este enviará un dato fuera de rango.

De acuerdo con las pruebas realizadas, se concluye qué las lecturas de los sensores dentro de un rango de 30° grados presenta un error menor al 5 % lo cual es un resultado aceptable para la implementación del sistema de navegación autónomo.

4.1.4. Algoritmo utilizado para calcular las velocidades

Para que nuestra red neuronal pulsante entrenada con la regla de aprendizaje R-STDP implementada en la tarjeta de procesamiento cumpla con el propósito de navegar el entorno evadiendo los obstáculos, se basó en unas reglas simples de If-Then. Para esto, se necesita las entradas de la red que, como anteriormente se mencionó son las lecturas de los sensores ultrasónicos (S1 a S6) adaptados en la parte frontal del robot móvil (ver figura 4.13), los dos ángulos de salida α_{Izq}, α_{Der} y el sentido que debe tomar *giro* (Izquierda o derecha).

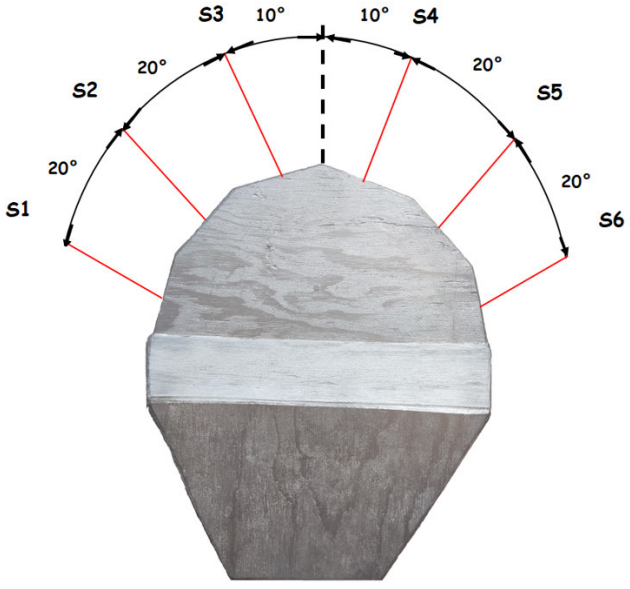


Figura 4.13: Orientación de los sensores en la parte frontal del robot (Bing et al., 2019).

Como se observa en la figura 4.13, los ángulos de orientación de los sensores son ±10°, ±30° y ±50°, para asegurarse de que haya una evasión exitosa en el algoritmo se le agrega 10° más a cada sensor respectivamente (±20°, ±40° y ±60°). Los sensores que tienen prioridad son los más centrales (S3 y S4), ya que son los que detectan los obstáculos con los que se puede colisionar potencialmente. Dado el caso que los sensores del lado derecho (S1, S2 y S3) o los del lado izquierdo (S4, S5 y S6) detecten un obstáculo simultáneamente dentro de su rango de medición, la salida será un ángulo de ±90°, esto corresponde a un giro en sentido contrario al obstáculo. A continuación, se puede observar el algoritmo que consta de estas reglas.

Algorithm 1 Obtención de ángulos y dirección de giro.

Input: $S_1, S_2, S_3, S_4, S_5, S_6$

Output: α_{Izq} , α_{Der} , Giro

$\alpha_{Izq} = 0°$, $\alpha_{Der} = 0°$, $Giro = 0$

$Sum_L \leftarrow S_1 + S_2 + S_3$

$Sum_R \leftarrow S_4 + S_5 + S_6$

if $S_3 == 1$ **then**
    $\alpha_{Izq} = 20°$
else
    if $S_2 == 1$ **then**
        $\alpha_{Izq} = 40°$
    else
        if $S_1 == 1$ **then**
            $\alpha_{Izq} = 60°$
        else
            $\alpha_{Izq} = 90°$
        end if
    end if
end if

Algorithm 2 Continuación del Algoritmo 1.

```
if S_4 == 1 then
    α_Der = 20°
else
    if S_5 == 1 then
        α_Der = 40°
    else
        if S_6 == 1 then
            α_Der = 60°
        else
            α_Der = 90°
        end if
    end if
end if
if α_Izq >| α_Der | then
    Giro = 1
else
    if α_Izq <| α_Der | then
        Giro = 2
    else
        if Sum_L < Sum_R then
            Giro = 1
        else
            Giro = 2
        end if
    end if
end if
```

Cuando el robot navega a través del entorno estructurado, toma lecturas de los sensores y el Algoritmo 1 genera salidas, esto se da cada 900 ms. Los diferentes escenarios garantizan que el robot móvil gire a su derecha o izquierda según lo requiera mientras navega. Eventualmente, la red toma la decisión de en qué sentido girar optando por el ángulo

con el valor absoluto más pequeño, sea α_{Izq} o α_{Der}, como ángulo de salida. Dado el caso en que este valor absoluto sea de igual valor, tomará la dirección contraria (1 para giro a la derecha, 2 para giro a la izquierda) respecto a la posición de los obstáculos.

Hasta el momento solo se han obtenido los ángulos y en qué sentido se hará el giro, con las siguientes ecuaciones se va a traducir en velocidades de motor reales para el Brick EV3 y sus actuadores, donde $V_{forward}$ denota una velocidad mínima inicial del motor para avanzar:

$$V_{left} = V_{forward} - \frac{\Delta V(\alpha)}{2} \tag{4.3}$$

$$V_{right} = V_{forward} + \frac{\Delta V(\alpha)}{2} \tag{4.4}$$

Donde $\Delta V(\alpha)$ es las velocidades del motor (derecho e izquierdo) que se necesitan para dar un giro de un ángulo α en 1 segundo.

También se cuenta con la función *Avanzar*, lo que significa que el robot móvil puede ir hacia adelante sin preocuparnos de que colisione, siempre que los sensores (S1 a S6) devuelven una lectura que está dentro de los siguientes rangos:

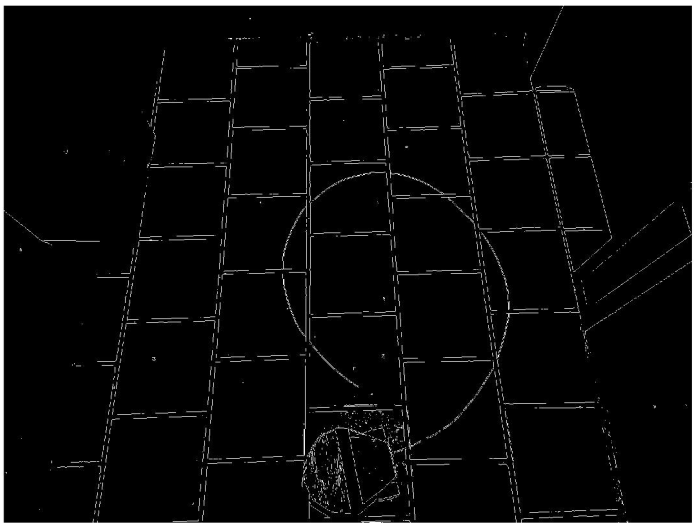
$$\begin{cases} s_1, s_6 \geq 0,01 \\ s_2, s_5 \geq 0,15 \\ s_3, s_4 = 1 \end{cases} \tag{4.5}$$

No obstante, hay un caso especial en el que suele fallar, cuando avanza a la velocidad media constante directamente hacia una curva, debido a esto y a el tiempo entre cada lectura detecta el obstáculo demasiado tarde, lo que no permite que tenga suficiente espacio para un giro y a su vez ocasiona un choque con los limites u obstáculos del entorno. Esto se logra solucionar relacionando la velocidad de avance con las lecturas del sensor (ecuación 4.6), lo que ocasiona que a medida que el robot se acerque a un obstáculo la velocidad disminuya y así, evita colisionar. como se observa en la figura 4.14.

$$V_{forward} = V_{init} * \sum_{i=1}^{6} S_i * \frac{1}{6} \tag{4.6}$$

Esta función se usa para que el robot pueda maniobrar giros en espacios reducidos y le permirte trazar rutas más suaves al evitar obstáculos solucionando este tipo de problemas como se observa en la siguiente figura:

(a) Robot móvil evitando colisión en curva.



(b) Detección de bordes.

Figura 4.14: Validación de función que reduce la velocidad media constante en curva.

4.2. Comunicación entre Raspberry pi y Brick EV3 a través de MATLAB

MATLAB (abreviatura de MATrix LABoratory, «laboratorio de matrices») es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio de alto nivel. Entre sus prestaciones básicas se hallan la manipulación de matrices, exploración, modelamiento, visualización de datos y funciones, el diseño y la implementación de algoritmos, efectúa cálculos a gran escala mediante equipos multinúcleo, GPU y nubes, y la comunicación con programas en otros lenguajes (Python, C/C++, Java) y con otros dispositivos hardware.

4.2.1. Raspberry Pi y MATLAB

Inicialmente, se instala un paquete en Matlab llamado "MATLAB Support Package for Raspberry Pi Hardware" lo que le permite compatibilidad y el poder comunicarse con la tarjeta de procesamiento Raspberry Pi de forma remota desde un ordenador que ejecute Matlab para así, adquirir los datos de los sensores y salidas de nuestra red neuronal entrenada. El adquirir esta herramienta también instala una versión personalizada de Raspbian Linux® en el hardware de Raspberry Pi, la instalación de esta versión nos permite utilizar comandos de Matlab para controlar y recuperar datos de hardware y periféricos Raspberry Pi.

Para comunicarse con la Raspberry Pi, Matlab necesita la dirección IP (dirección única para cada dispositivo) que se le otorga a la tarjeta de procesamiento al conectarse a la red, además de el usuario y contraseña configurada por defecto. Para este caso "192.168.1.84" será la dirección usada para crear una conexión que lleve un nombre específico, por ejemplo:

mypi = raspi('192.168.1.84','pi','raspberry')

Luego de conectar exitosamente, se puede acceder a la interfaz de comandos de Linux en el hardware de Raspberry Pi. La función *system* permite ejecutar archivos de python (.py) en el hardware de Raspberry Pi y así, obtener las salidas de nuestra red entrenada en el workspace de Matlab, esta función utiliza cifrado SSH. Como primer argumento se usa la conexión definida *mypi*, el segundo argumento es para especificar la ruta de acceso

```
mypi =

  raspi with properties:

         DeviceAddress: '192.168.1.84'
                  Port: 18734
             BoardName: 'Raspberry Pi 3 Model B+'
        AvailableLEDs: {'led0'}
  AvailableDigitalPins: [4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
  AvailableSPIChannels: {'CE0','CE1'}
     AvailableI2CBuses: {'i2c-1'}
      AvailableWebcams: {}
           I2CBusSpeed: 100000
```

Figura 4.15: Conexión al Hardware y periféricos de Raspberry Pi.

y el nombre del archivo, tal como se muestra a continuación:

velocidad = system(mypi,'python3/home/pi/Controller/controllerSNN-final.py')

Cada 900 ms, los datos obtenidos son guardados en el vector "velocidad" e inmediatamente enviados al Brick EV3 para ser traducidos en velocidad y a través de los actuadores el robot móvil ejecute un desplazamiento en la dirección deseada.

4.2.2. Brick EV3 y MATLAB

Esta etapa de comunicación consiste en lograr que el robot móvil realice la navegación del entorno, encontrando la ruta libre de obstáculos, sin colisiones. Para realizar tal actividad de forma óptima, es necesaria la traducción de los valores obtenidos de la salida de la red neuronal entrenada y enviados a través de Matlab en instrucciones que el robot pueda interpretar. Para esta traducción, como se hizo anteriormente, lo primero es instalar el paquete "MATLAB Support Package for LEGO MINDSTORMS EV3 Hardware" para agregar compatibilidad con el Brick EV3. Esta herramienta consta de funciones escritas en Matlab para comunicarle al robot las distintas tareas que debe realizar, tales como: configuración y lectura de sensores, encendido y apagado de motores, este último serán los que se usen.

Como se mencionó en el capítulo anterior, hay varias opciones de conexión ordenador host al Brick EV3, en este caso enviaremos los comandos por medio de Bluetooth. Para

esto debemos mantener activa esta opción en ambos extremos, y creando el siguiente objeto *mylego* en el workspace, se conectará el software Matlab al hardware Brick EV3 permitiéndole interactuar con periféricos y objetos, en este caso se usa *motor* para controlar el estado de los motores.

```
mylego = legoev3('bt','0016534d273d');

mymotor1 = motor(mylego, 'C');

mymotor2 = motor(mylego, 'B');
```

Se utiliza la función *legoev*3 para conectarse al Brick EV3, como primer parámetro se especifica el tipo de conexión y luego, el código único asignado al dispositivo. Luego de conectarse exitosamente mediante el uso de la interfaz serial Bluetooth, para controlar la velocidad y la dirección del motor, se le asigna un valor a la propiedad *Speed* del objeto creado para el motor derecho e izquierdo (mymotor1, mymotor2).

```
mymotor1.Speed = velocidad(1);

mymotor2.Speed = velocidad(2);
```

Anteriomente, se mencionó que los valores de entrada para los motores oscilan entre -100 y 100 obteniendo un avance o retroceso total. El valor por defecto de la propiedad *Speed* es 0, este valor puede variar mientras el motor está en marcha o detenido (usando el objeto *start* o *stop*).

Por último, se diseñó un algoritmo con las instrucciones mencionadas en el entorno Matlab para que después de establecer conexión del ordenador con la Raspberry Pi y el Brick EV3, por un tiempo estipulado para la navegación se tome lecturas de los sensores ubicados en la parte frontal del robot móvil, se procesen estos datos en la red neuronal pulsante entrenada con la estrategia de refuerzo R-STDP y obtener las salidas en el Workspace para así, enviarlas con las instrucciones ya mencionadas al Brick EV3 para que el robot realice una serie de desplazamientos cumpliendo con la tarea de navegar los entornos propuestos a continuación sin colisionar con los límites y obstáculos encontrados.

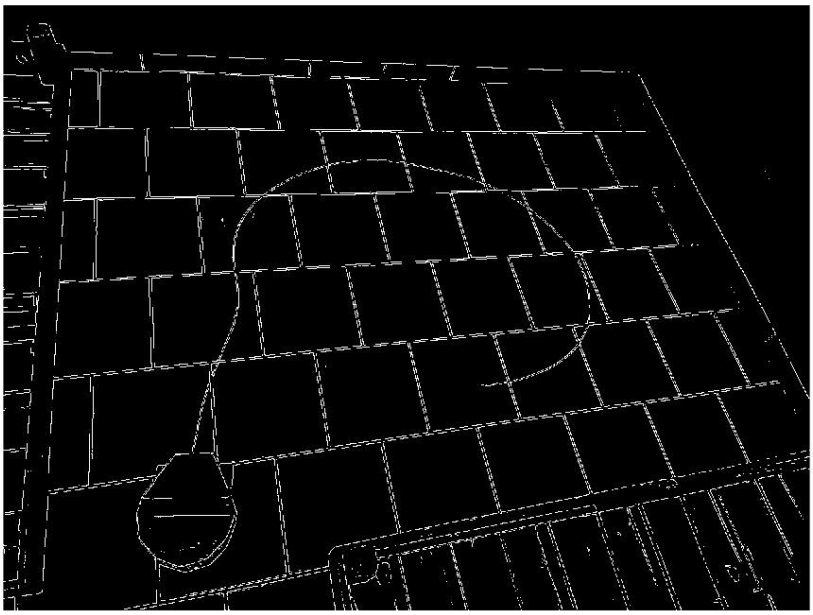
4.3. Validación de la navegación del robot móvil en un entorno estructurado

Para validar el correcto funcionamiento de la implementación de la red neuronal pulsante en la toma de decisiones, la adaptación de los sensores para la captura de la distancia de los obstáculos y la comunicación de la Raspberry Pi con el Brick EV3 para el desplazamiento oportuno, se construyó físicamente un ambiente estructurado de longitud 200x250 cm. Donde el robot móvil conduce con el riesgo de chocar con los límites del entorno y algunos obstáculos de diferentes dimensiones como se muestra en las figuras 4.16, 4.17 y 4.18, esto es fundamental para la demostración de su correcta capacidad reactiva ante el ambiente desconocido.

A continuación, se presentan tres escenarios para observar la conducta del robot donde en la primera escena el robot móvil evita los límites del entorno, para la segunda escena se sumaron dos objetos, iniciando desde un punto diferente al primero. Por último, se agregó un objeto más a la escena dificultando el desplazamiento del robot móvil para conocer sus capacidades de navegación y evaluar los comportamientos deseados para el sistema dentro del entorno, con el fin de documentar en este trabajo.

Se comprobó qué, el robot móvil logró explorar el entorno detectando por medio de los sensores ultrasónicos los limites del espacio y obstáculos propuestos, trazando así una trayectoria libre de colisiones.

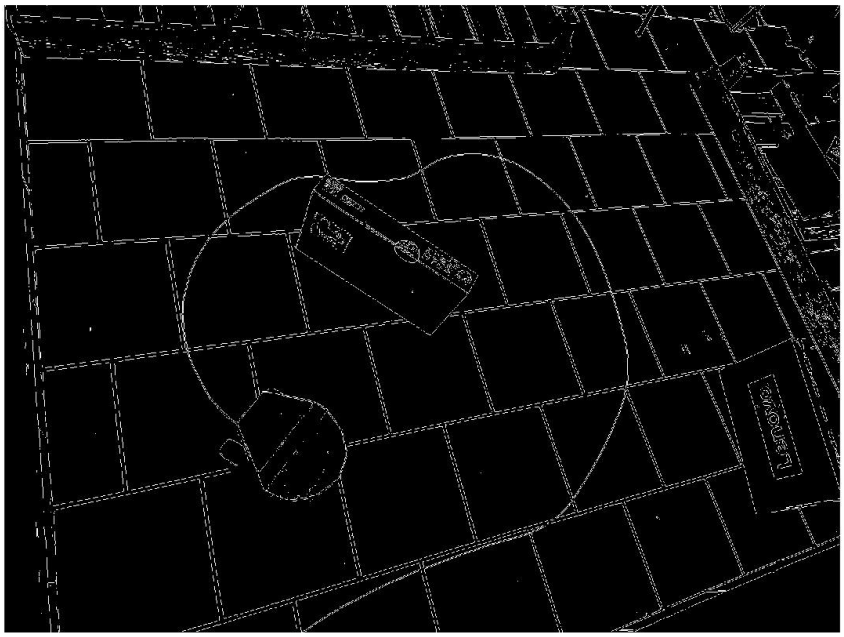
(a) Robot móvil trazando ruta sin obstáculos.



(b) Detección de bordes.

Figura 4.16: Validación de navegación exitosa para el primer escenario.

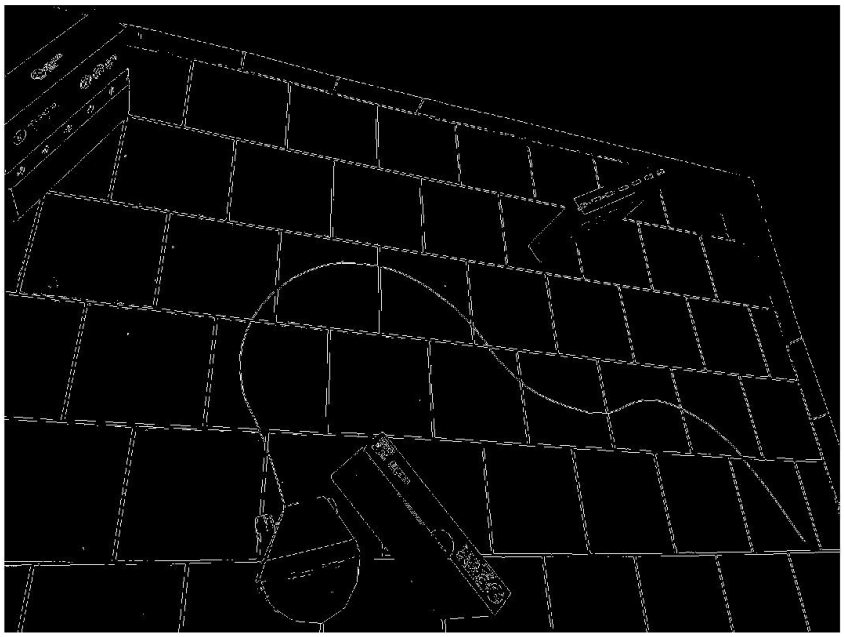
(a) Robot móvil trazando ruta sin obstáculos.



(b) Detección de bordes.

Figura 4.17: Validación de navegación exitosa para el segundo escenario.

(a) Robot móvil trazando ruta sin obstáculos.



(b) Detección de bordes.

Figura 4.18: Validación de navegación exitosa para el tercer escenario.

Capítulo 5

Conclusiones

En este trabajo se presentó un sistema de navegación autónoma para entornos estructurados, usando como medio de comunicación entre la Raspberry Pi y el Brick EV3 a Matlab y con la ayuda de sensores ultrasónicos muestra un grado de reactividad ante una situación inesperada, como la un ser vivo.

Se cumplió con el objetivo de implementar y validar una estrategia de aprendizaje por refuerzo permitiendo la navegación del robot móvil por una trayectoria libre de colisiones en un entorno estructurado.

Se logró detectar los límites y obstáculos en el entorno estructurado real a través del uso de sensores ultrasónicos para la captura de las distancias, lo que permitió que fueran procesados por la red para un exitoso desplazamiento del robot móvil.

Para la realización de este proyecto fue necesaria la adaptación de seis sensores a la Raspberry Pi, los cuales cumplieron con la tarea de detectar y enviar la información de las distancias de los obstáculos en un rango entre 5 cm a 130 cm con un ángulo de acción de 30°.

El uso de la estrategia de aprendizaje por refuerzo favoreció el buen desempeño del robot móvil en la toma de decisiones, ya que al recibir recompensa al dirigirse a donde los obstáculos estaban más lejanos le permitió navegar evitando las colisiones.

La comunicación desarrollada en este sistema puede servir como base en cursos de

robótica y control, aplicando los conocimientos teóricos adquiridos en la investigación y volverlos prácticos al enviarlos en tiempo real ejecutando instrucciones desde un ordenador a través del entorno de desarrollo integrado Matlab, todo esto vía Bluetooth para lograr la tarea propuesta.

5.1. Trabajos futuros

Este trabajo deja abierta las posibilidades de investigaciones para el futuro sobre la navegación reactiva de robots móviles en entornos dinámicos; donde el uso de algoritmos basados en lógica difusa tendría gran ventaja.

Otro campo que puede ser estudiado a futuro con respecto a la investigación desarrollada, es la creación de mapas por parte del sistema autónomo al explorar entornos desconocidos; con los cambios pertinentes en el rango de la comunicación y los tipos de sensores de proximidad usados para el presente trabajo.

Referencias

Anand, G. and Kumawat, A. K. (2021). Object detection and position tracking in real time using Raspberry Pi. *Materials Today: Proceedings*, 47:3221–3226.

Angleraud, A., Mehman Sefat, A., Netzev, M., and Pieters, R. (2021). Coordinating Shared Tasks in Human-Robot Collaboration by Commands. *Frontiers in Robotics and AI*, 8:332.

Areiza, Y., Garzón, Y., and Charry, O. (2016). Raspberry pi b+. revisión técnica: guía de uso y programación.

Bing, Z., Baumann, I., Jiang, Z., Huang, K., Cai, C., and Knoll, A. (2019). Supervised learning in snn via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle. *Frontiers in Neurorobotics*, 13:18.

Brand, M. and Yu, X.-H. (2013). Autonomous robot path optimization using firefly algorithm. In *2013 International Conference on Machine Learning and Cybernetics*, volume 03, pages 1028–1032. ISSN: 2160-1348.

Buccelli, S., Bornat, Y., Colombi, I., Ambroise, M., Martines, L., Pasquale, V., Bisio, M., Tessadori, J., Nowak, P., Grassia, F., Averna, A., Tedesco, M., Bonifazi, P., Difato, F., Massobrio, P., Levi, T., and Chiappalone, M. (2019). A neuromorphic prosthesis to restore communication in neuronal networks. *iScience*, 19:402–414.

Carelli, R. and Oliveira Freire, E. (2003). Corridor navigation and wall-following stable control for sonar-based mobile robots. *Robotics and Autonomous Systems*, 45(3):235–247.

Cerezuela Escudero, E. (2015). Una aportación a los sistemas de procesamiento de la información basados en modelos neuronales pulsantes. Accepted: 2015-09-28T07:10:19Z.

Cuan, C. (2021). OUTPUT: Choreographed and Reconfigured Human and Industrial Robot Bodies Across Artistic Modalities. *Frontiers in Robotics and AI*, 7:218.

Cuevas-Arteaga, B. and Rostro-Gonzalez, H. (2017). *"Implementación de un Generador de Patrones Centrales basado en Neuronas Pulsantes sobre una tarjeta SpiNNaker"*. PhD thesis.

Davidson, S. and Furber, S. B. (2021). Comparison of artificial and spiking neural networks on digital hardware. *Frontiers in Neuroscience*, 15:345.

Davies, S. (2013). Learning in spiking neural networks.

Dietterich, T. G. (2017). Steps toward robust artificial intelligence. *AI Mag.*, 38:3–24.

Dudek, G. and Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge University Press, New York, 2nd ed edition. OCLC: ocn477266771.

Franchi, M., Fanelli, F., Bianchi, M., Ridolfi, A., and Allotta, B. (2020). Underwater Robotics Competitions: The European Robotics League Emergency Robots Experience With FeelHippo AUV. *Frontiers in Robotics and AI*, 7:3.

Gai, J., Xiang, L., and Tang, L. (2021). Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle. *Computers and Electronics in Agriculture*, 188:106301.

George, R., Chiappalone, M., Giugliano, M., Levi, T., Vassanelli, S., Partzsch, J., and Mayr, C. (2020). Plasticity and adaptation in neuromorphic biohybrid systems. *iScience*, 23(10):101589.

Guedes, V. J. C. B., Maciel, S. T. R., and Rocha, M. P. (2022). Refrapy: A Python program for seismic refraction data analysis. *Computers & Geosciences*, 159:105020.

Gutiérrez, J. S. (2009). Trayectorias para robots móviles de navegación autónoma. page 70.

Ha, Q. P., Yen, L., and Balaguer, C. (2019). Robotic autonomous systems for earthmoving in military applications. *Automation in Construction*, 107:102934.

Hernández-Becerra, C. and Mejía-Lavalle, M. (2016). Clasificación de patrones mediante el uso de una red neuronal pulsante. *Research in Computing Science*, 116(1):81–91.

Hidalgo-Paniagua, A., Vega-Rodríguez, M. A., Ferruz, J., and Pavón, N. (2017). Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. *Soft Computing*, 21(4):949–964.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press. Google-Books-ID: 5EgGaBkwvWcC.

Hong, T., Nakhaeinia, D., and Karasfi, B. (2012). *Application of Fuzzy Logic in Mobile Robot Navigation*.

Izhikevich, E. M. (2007). Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling. *Cerebral Cortex*, 17(10):2443–2452.

Janglova, D. (2004). Neural Networks in Mobile Robot Motion. *International Journal of Advanced Robotic Systems*, 1(1):2. Publisher: SAGE Publications.

Jiménez Fernández, F. (2010). Diseño y evaluación de sistemas de control y procesamiento de señales basados en modelos neuronales pulsantes. Accepted: 2014-11-27T11:58:03Z.

Khatib, O. (1986). The Potential Field Approach And Operational Space Formulation In Robot Control. In Narendra, K. S., editor, *Adaptive and Learning Systems: Theory and Applications*, pages 367–377. Springer US, Boston, MA.

Kuroki, M. (2021). Using Python and Google Colab to teach undergraduate microeconomic theory. *International Review of Economics Education*, 38:100225.

Lozada, E. L., Espino, E. R., Azuela, J. H. S., and Ponce, V. H. P. (2020). Selección de acciones para la navegación de un robot móvil basada en fuzzy Q-learning. page 11.

Lu, H., Liu, J., Luo, Y., Hua, Y., Qiu, S., and Huang, Y. (2021). An autonomous learning mobile robot using biological reward modulate stdp. *Neurocomputing*, 458:308–318.

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671.

Medina-Santiago, A., Camas-Anzueto, J. L., Vazquez-Feijoo, J. A., Hernández-de León, H. R., and Mota-Grajales, R. (2014). Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors. *Journal of Applied Research and Technology*, 12(1):104–110.

Mejia-Lavalle, M., Aguilar, K., Sossa, H., Mujica, D., and Magadan, A. (2019). Modelos neuronales pulsantes adaptados para el mejoramiento de luminosidad de imágenes cerebrales de gran resolución. *Research in Computing Science*, 148(7):253–266.

Minsky, M. (1961). Steps toward Artificial Intelligence. *Proceedings of the IRE*, 49(1):8–30.

Montiel, H., Jacinto, E., and MartÃnez, F. H. (2015). Generación de Ruta óptima para Robots a Partir de Segmentación de Imágenes. *Información tecnológica*, 26:145 – 152.

Murphy, R. (2000). Introduction to AI Robotics. page 13.

Nitz, D. A., Kargo, W. J., and Fleischer, J. (2007). Dopamine signaling and the distal reward problem. *NeuroReport*, 18(17):1833–1836.

Oltean, S.-E. (2019). Mobile Robot Platform with Arduino Uno and Raspberry Pi for Autonomous Navigation. *Procedia Manufacturing*, 32:572–577.

Ortiz Arroyave, L. F., Vásquez Carvajal, M., and Muñoz Ceballos, N. D. (2018). Navegación de robots móviles en entornos con discontinuidades: una revisión. *Revista Politécnica*, 14(27):103–115.

Pal, P. K. and Kar, A. (1996). Sonar-based mobile robot navigation through supervised learning on a neural net. *Autonomous Robots*, 3(4):355–374.

Pastor, J. and Rodríguez, F. J. (2006). La Robótica como elemento de motivación del aprendizaje en los alumnos de Ingeniería y potenciación de habilidades profesionales. page 9.

Patle, B. K., Babu L, G., Pandey, A., Parhi, D. R. K., and Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606.

Pavlov, I. P. (1997). *Los reflejos condicionados: lecciones sobre la función de los grandes hemisferios*. Ediciones Morata. Google-Books-ID: tU0e7ox8eQ4C.

Pecolt, S., Blazejewski, A., Królikowski, T., Trzebiatowski, P. Z., and Schulz, B. (2021). Autonomous control of a mobile robot to explore areas with difficult terrain. *Procedia Computer Science*, 192:3467–3476.

Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309.

Rosheim, M. E. (2006). *Leonardo's lost robots*. Springer, Berlin.

Ruge Ruge, I. A. and Alvarado, J. D. (2013). Sistema basado en FPGA para la evaluación de redes neuronales orientadas al reconocimiento de imágenes. *Revista Tecnura*, 17(36):87.

Salazar Triana, J. A. and Sánchez Moreno, J. C. (2018). Diseño e implementación de una red neuronal artificial (RNA) para detección y diagnóstico de fallas de un proceso de control de presión. Accepted: 2018-11-09T21:20:42Z.

Sanchez, G., Madrenas, J., and Cosp-Vilella, J. (2019). LEGION-based image segmentation by means of spiking neural networks using normalized synaptic weights implemented on a compact scalable neuromorphic architecture. *Neurocomputing*, 352:106–120.

Schrauwen, Benjamin and D'Haene, Michiel and Verstraeten, David and Van Campenhout, Jan (2008). Compact hardware liquid state machines on FPGA for real-time speech recognition. *Neural Networks*, 21(2-3):511–523.

Shantia, A., Timmers, R., Chong, Y., Kuiper, C., Bidoia, F., Schomaker, L., and Wiering, M. (2021). Two-stage visual navigation by deep neural networks and multi-goal reinforcement learning. *Robotics and Autonomous Systems*, 138:103731.

Siegwart, R., Nourbakhsh, I., and Scaramuzza, D. (2004). Introduction to Autonomous Mobile Robots.

Smith, W. B., Starck, S. R., Roberts, R. W., and Schuman, E. M. (2005). Dopaminergic Stimulation of Local Protein Synthesis Enhances Surface Expression of GluR1 and Synaptic Transmission in Hippocampal Neurons. *Neuron*, 45(5):765–779.

Sugeno, M. and Nishida, M. (1985). Fuzzy control of model car. *Fuzzy Sets and Systems*, 16(2):103–113.

Sánchez, R. and Rodriguez, F. (2021). Algoritmo de navegación reactiva de robots móviles para tareas bajo invernadero.

Taylor, S. E., Healy, M. J., and Caudell, T. P. (2012). An Autonomous Distal Reward Learning Architecture for Embodied Agents. *Procedia Computer Science*, 13:198–204.

Torres-Fernández, O. (2006). La técnica de impregnación argéntica de Golgi. Conmemoración del centenario del premio nobel de Medicina (1906) compartido por Camillo Golgi y Santiago Ramón y Cajal. *Biomédica*, 26(4):498.

Torres.AI, J. (2021). 1. Introducción al aprendizaje por refuerzo.

Urrea, C. and Kern, J. (2021). Design and implementation of a wireless control system applied to a 3-DoF redundant robot using Raspberry Pi interface and User Datagram Protocol. *Computers & Electrical Engineering*, 95:107424.

Wang, J., Elfwing, S., and Uchibe, E. (2021). Modular deep reinforcement learning from reward and punishment for robot navigation. *Neural Networks*, 135:115–126.

Wood, G. (2002). *Living Dolls: A Magical History of the Quest for Mechanical Life*. Faber & Faber. Google-Books-ID: toO9NAEACAAJ.

Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation*, 1(1):18–28. Conference Name: IEEE Transactions on Evolutionary Computation.

Yang, X.-S. (2009). Firefly Algorithms for Multimodal Optimization. In Watanabe, O. and Zeugmann, T., editors, *Stochastic Algorithms: Foundations and Applications*, Lecture Notes in Computer Science, pages 169–178, Berlin, Heidelberg. Springer.

Zaldivar, D., Cuevas, E., Maciel, O., Valdivia, A., Chavolla, E., and Oliva, D. (2021). Aprender técnicas de optimización clásicas y metaheurísticas mediante el uso de una plataforma educativa basada en robots LEGO. *The International Journal of Electrical Engineering & Education*, 58(2):286–305. Publisher: SAGE Publications Ltd STM.

Zhang, M. and Wan, Y. (2020). Improving learning experiences using LEGO Mindstorms EV3 robots in control systems course. *The International Journal of Electrical Engineering & Education*, page 002072092096587.

Zhang, T., Qiu, T., Pu, Z., Liu, Z., and Yi, J. (2020). Robot Navigation among External Autonomous Agents through Deep Reinforcement Learning using Graph Attention Network. *IFAC-PapersOnLine*, 53(2):9465–9470.

Zhang, X., Lu, J., Wang, Z., Wang, R., Wei, J., Shi, T., Dou, C., Wu, Z., Zhu, J., Shang, D., Xing, G., Chan, M., Liu, Q., and Liu, M. (2021). Hybrid memristor-cmos neurons for in-situ learning in fully hardware memristive spiking neural networks. *Science Bulletin*, 66(16):1624–1633.

Zou, A.-M., Hou, Z.-G., Fu, S.-Y., and Tan, M. (2006). Neural Networks for Mobile Robot Navigation: A Survey. In Wang, J., Yi, Z., Zurada, J. M., Lu, B.-L., and Yin, H., editors, *Advances in Neural Networks - ISNN 2006*, libros Lecture Notes in Computer Science, pages 1218–1226, Berlin, Heidelberg. Springer.