UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:

DESARROLLO DE UN ROBOT MÓVIL DIFERENCIAL CONTROLADO MEDIANTE UN ALGORITMO DE BÚSQUEDA CON REDES NEURONALES.

AUTORES:
JONATHAN PATRICIO AGUILAR CAMPOS
HÉCTOR ISIDRO SANDOVAL SUQUILLO

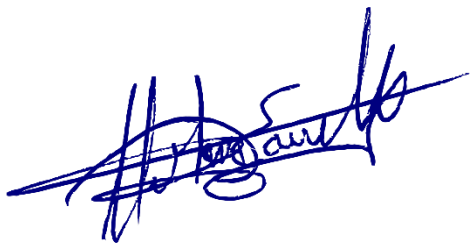
TUTORA:
CARMEN JOHANNA CELI SÁNCHEZ

Quito, agosto del 2018

CESIÓN DE DERECHOS DE AUTORES

Nosotros Jonathan Patricio Aguilar Campos y Héctor Isidro Sandoval Suquillo, con documento de identificación N° 1715740369 y N° 1718232646 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: "DESARROLLO DE UN ROBOT MÓVIL DIFERENCIAL CONTROLADO MEDIANTE UN ALGORITMO DE BÚSQUEDA CON REDES NEURONALES", mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.



Jonathan Patricio Aguilar Campos
C.I: 1715740369

Héctor Isidro Sandoval Suquillo
C.I: 1718232646

Quito, agosto 2018

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación "DESARROLLO DE UN ROBOT MÓVIL DIFERENCIAL CONTROLADO MEDIANTE UN ALGORITMO DE BÚSQUEDA CON REDES NEURONALES" realizado por Jonathan Patricio Aguilar Campos y Héctor Isidro Sandoval Suquillo, obteniendo un producto que cumple con todos los requerimientos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, agosto 2018

Carmen Johanna Celi Sánchez
C.I: 1717437808

DEDICATORIA

Dedico este trabajo de titulación a mi madre, por ese gran apoyo brindado a lo largo de estos 5 años de carrera, por toda la confianza depositada en mi persona para poder culminar mi carrera profesional; por ser padre y madre a la vez, por demostrarme día a día que el esfuerzo de hoy tiene sus frutos mañana.

Dedico a mis tíos y tías queridas por ese gran apoyo a lo largo de todo este tiempo, por estar ahí en los momentos difíciles de la vida y por dar ese consejo sabio para seguir adelante y no desmayar, gracias por todo ese apoyo incondicional.

Jonathan Patricio Aguilar Campos

Este proyecto de titulación va dedicado a cada persona que tuvo confianza en mí, a aquellos que alguna vez supieron compartir parte de su tiempo, conocimientos y paciencia conmigo, a aquellas personas que me incentivaron siempre a buscar algo más y que en base a su cariño hoy estoy en este camino, a mi madre, que es mi gran apoyo, y a mi padre que hoy ya no está aquí.

He tenido la suerte de conocer a varias personas, que para mí han sido más que mentores, de los cuales aprendí que toda meta es alcanzable con trabajo duro y constancia, para todos ellos va dedicado este trabajo de titulación.

Héctor Isidro Sandoval Suquillo

AGRADECIMIENTOS

Mi profundo agradecimientos a la Universidad Politécnica Salesiana en especial a la Facultad de Ingeniería Electrónica por su acogida durante estos 5 años de carrera y por el apoyo de un gran equipo de docentes quienes día a día han impartido sus conocimientos y herramientas necesarias para nuestra formación profesional. A la Ingeniera Johanna Celi en su papel de directora de tesis, por sus importantes sugerencias, criticas, aportes y desarrollo de este trabajo de titulación.

Jonathan Patricio Aguilar Campos

El presente trabajo de tesis culmina gracias al esfuerzo de un grupo de personas a las que por medio del siguiente párrafo brindaran agradecimientos
En primera instancia agradecemos a la UNIVERSIDAD POLITÉCNICA SALESIANA, específicamente a la Facultad de Ingeniería Electrónica y a su equipo de docentes por brindarnos las herramientas necesarias para nuestra formación. A la Ingeniera Johanna Celi en su papel de directora de tesis quien brindo su apoyo, tiempo, conocimientos, visión crítica y experiencia para la culminación del proyecto de grado.

Héctor Isidro Sandoval Suquillo

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el siguiente proyecto técnico se detalla el diseño, el ensamblaje y la programación de la autonomía de un robot móvil diferencial destinado a movilizarse en ambientes semiestructurados basado en redes neuronales y un algoritmo de mapeo con el fin de que el desplazamiento del mismo sea eficiente en referencia a la distancia recorrida entre dos puntos A y B.

En primera instancia el robot mapea el entorno identificando todos los nodos por lo que se moviliza, siendo estos zonas con más de una posible ruta como por ejemplos bifurcaciones, cruces totales, cruces en T, cabe recalcar que para este proyecto no se toman en cuenta como nodos los giros obligatorios como por ejemplo el camino sin salida con el fin de optimizar el tiempo de procesamiento de la red neuronal.

Una vez que el robot mapea todo su entorno envía una matiz de nodos a la red neuronal para proseguir con el algoritmo de aprendizaje. Para la convergencia de la red neuronal, se utilizó la red de Hopfield la cual es de aprendizaje no supervisado en su forma discreta.

Mediante el algoritmo de recuperación la red neuronal identifica patrones de entrada, para finalmente producir los respectivos desplazamientos del robot. Al final selecciona la ruta más corta entre la salida y la llegada.

ABSTRACT

The following technical project details the design, assembly and programming of the autonomy of a mobile diagnostic robot was mobilized in semi-structured environments based on neural networks and a mapping algorithm in order that the displacement of the same is efficient in reference to the distance traveled between two points A and B.

In the first instance the robot maps the environment identifying all the nodes for what is mobilized, being these areas with more than one possible route as for example bifurcations, total crosses, T-crosses, it should be noted that for this project is not taken into account as nodes the obligatory turns such as the dead-end path in order to optimize the processing time of the neural network.

Once the robot maps all its surroundings, it sends a nuance of nodes to the neural network to continue with the learning algorithm for the convergence of the neural network, the Hopfield red which is unsupervised learning in its discrete form.

Through the recovery algorithm the neural network identifies input patterns, to finally produce the next displacements of the robot. At the end, select the shortest route between departure and arrival.

INTRODUCCIÓN

Los argumentos para este trabajo se encuentran detallados en los siguientes cuatro capítulos comenzando por el capítulo uno donde se describe el problema a ser tratado, los objetivos planteados y su respectiva justificación. En el capítulo dos se redacta el marco teórico utilizado para la implementación de la red neuronal de Hopfield el modelo matemático de la odometría del robot, así como los conceptos básicos tales como señores, locomoción, diseño electrónico, calibración, controladores PID y ON/ OFF.

En el capítulo tres se redacta la implementación de la red de Hopfield y las ecuaciones odométricas utilizando el software Matlab, se muestra los pasos necesarios para la instalación de los programas necesarios para grabar dicho algoritmo en el robot laberinto. Se describe todo el diseño eléctrico y electrónico para llegar al prototipo final así como los elementos utilizados, finalmente se muestran los flujogramas que describen la programación implementada para que el robot se desplace mapeando el laberinto y lo resuelva dando como solución la ruta optima del punto de partida A al punto de llegada B.

En el capítulo cuatro de describen los resultados obtenidos durante de todas las pruebas realizadas al prototipo para verificar que la red neuronal y la calibración de los controladores PID funcionen satisfactoriamente, se muestra la importancia de que el robot tenga un buena tracción sin derrapes y finalmente de muestran todos los errores obtenidos durante estas pruebas tales como que problema de que las iteraciones de la red no sean necesarias para que esta aprenda.

CAPÍTULO 1

ANTECEDENTES

1.1 PLANTEAMIENTO DEL PROBLEMA

Luego de haber participado en diversos concursos de robótica tanto a nivel interno como externo se pudo constatar que, los robots móviles diferenciales cuentan únicamente con una programación tradicional la cual depende netamente de los sensores. Al no contar con una realimentación en las ruedas el robot no sigue una línea recta y esto provoca que una rueda se frene y otra avance en un intento de estabilizarse lo que ocasiona que el robot pierda velocidad y en otros casos colisione en la pared. Además, de pendiendo de la programación y del tipo de sensores en algunos casos el robot no encuentra la salida y se pierde en el entorno.

1.2 JUSTIFICACIÓN

Actualmente, el campus Quito-Sur de la Universidad Politécnica Salesiana cuenta con un Club de Robótica, el cual no ha desarrollado un robot diferencial eficiente al balancear velocidad y precisión en entornos semiestructurados. Es así que, en los concursos realizados por otras universidades, nuestra facultad no ha conseguido la notoriedad e importancia deseada en comparación a otras universidades que han destinado mayores recursos para estos eventos.

1.3 OBJETIVOS

1.3.1 Objetivo General

- Desarrollar un robot móvil diferencial controlado mediante un algoritmo de búsqueda de la ruta óptima con redes neuronales.

1.3.2 Objetivos Específicos

- Diseñar el sistema eléctrico, electrónico y mecánico del proyecto mediante aplicación de métodos viables para crear un prototipo funcional en un entorno semiestructurado.
- Programar la autonomía del robot en la tarjeta ARM a través de redes neuronales y algoritmos de búsqueda con el fin de hallar la ruta óptima.

- Desarrollar un algoritmo capaz de registrar la trayectoria del robot y que permita visualizarla utilizando la interfaz de Matlab.
- Ejecutar pruebas de funcionamiento en diferentes entornos para disminuir errores y aplicar correcciones

1.4 METODOLOGÍA

De todas las redes neuronales existentes y después de un estudio de diferentes redes se estudió y se utilizó la red de Hopfield ya que esta red además de memorizar la trayectoria permite obtener el camino óptimo de un punto de partida hacia un punto de llegada, esto hizo que se minimizara el código al no utilizar otro algoritmo para encontrar la ruta óptima.

La odometría al ser el cálculo de posicionamiento del robot resolvió los problemas de que distancia recorrió y el ángulo que giro el robot. En un principio se pensó en utilizar un giroscopio pero este ayudaba únicamente con el ángulo girado pero no con la distancia, así que la odometría fue la solución para responder a la pregunta ¿Dónde está el robot?.

Debido a que la red neural conlleva un tiempo de convergencia y que este tiempo no es el mismo para cualquier número de nodos, se realizó diferentes pruebas en diferentes entornos para de esta manera saber cuántas iteraciones necesita en cada entorno. Con la información obtenida se tendrá un número de iteraciones aproximadas para un número de nodos.

Una óptima calibración de los sensores ayuda a disminuir posibles errores de zonificación durante todo el recorrido del robot, esto se logró después de múltiples pruebas en la pista.

1.5 BENEFICIARIO

La beneficiaria de este robot diferencial es la Universidad Politécnica Salesiana campus Sur, la cual generalmente concursa en los distintos torneos de robótica que se llevan a cabo cada año, para los cuales contará con un prototipo de calidad tanto en hardware como software.

CAPÍTULO 2
FUNDAMENTACIÓN

En el presente proyecto se realiza el desarrollo y la implementación de un robot diferencial, por ello se escogerá hardware de calidad adecuada para el correcto funcionamiento del dispositivo y se programará una red neuronal mediante la cual el robot optimizara la trayectoria, finalmente se realizarán pruebas en un laberinto para corroborar su funcionamiento.

2.1 INTRODUCCIÓN

Para un concurso de robótica es necesario tener un prototipo competitivo tanto en hardware como en software además de una estrategia, con ese objetivo se ha planteado realizar un robot diferencial cuyo sistema de navegación no esté basada en programación tradicional, por lo cual se propone implementar el uso de una red neural, para resolver la trayectoria escogiendo la ruta más corta entre el punto de salida y el punto de llegada, hipótesis que se corroborara experimentalmente.

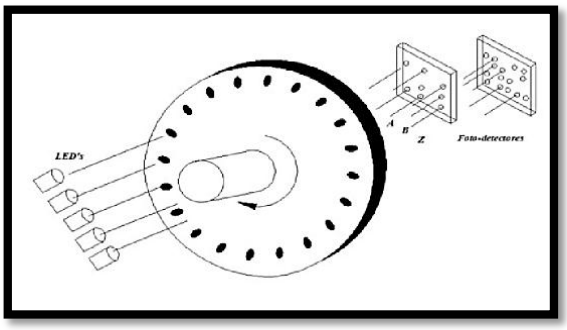
El uso de redes neuronales en un robot brinda un mejor desempeño en diferentes actividades, en el caso particular de un robot diferencial, el software comúnmente usado es un controlador PID, que conjuntamente con sensores de distancia, intentan combinar la evasión de obstáculos y la velocidad, entrando muchas veces en caminos sin salida, reflejándose esto en la acumulación del tiempo de llegada a la meta, la ventaja de utilizar redes neuronales recae en que después de realizar el mapeo del entorno por primera vez, la red neuronal converge, seleccionando los nodos que en conjunto forman la ruta más corta desde el punto de salida hasta el punto de llegada optimizando el tiempo en pista.

2.2 ENCODER

Este tipo de dispositivo es ampliamente utilizado en la robótica, con este se puede medir posición, aceleración y velocidad del rotor de un motor. En un robot diferencial se utilizan para obtener datos de velocidad de las ruedas y realizar retroalimentación con el fin de controlar el desplazamiento del robot, el cual debe ser paralelo a las paredes del entorno, además de calcular la distancia recorrida; los más utilizados son los encoders incrementales cuyo esquema se puede observar en la

figura 2.1, estos muestran dos señales desfasadas 90 grados eléctricos, lo cual permite conocer el sentido de giro del rotor. Para obtener la velocidad y posición es necesario únicamente una señal. (Requena, 2009)

Figura 2.1: Encoder incremental

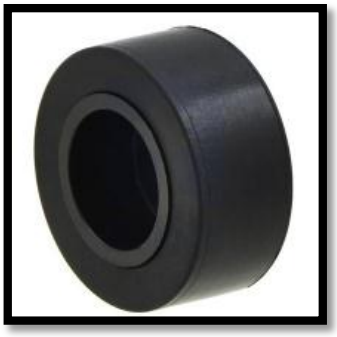


Encoder óptico incremental, Fuente: (Cortés, 2011)

2.3 RUEDAS

Uno de los aspectos importantes en un robot diferencial son las ruedas, el tamaño estas guarda una relación proporcional con la resolución del encoder, ya que a un mayor diámetro de rueda un pulso en el encoder representa una mayor distancia de desplazamiento y por ende un mayor error de medición, además el material del que están constituidas las mismas debe balancear tracción y peso; la rueda debe ser blanda para tenga mejor tracción con el suelo, usualmente se utilizan llantas de goma similares a las de la figura 2.2, las cuales tienen la tracción suficiente si excederse en su peso, esto brinda una mejor estabilidad cuando el robot se desplaza a velocidades altas en pista y ayuda a que el robot no derrape en las curvas. (Ollero, 2001)

Figura 2.2: Llanta de goma pololu



Rueda de goma de alta tracción, Fuente: (Pololu, 2017)

2.4 MICRO MOTOR

Estos motores cumplen un rol importante en el robot , están provistos de una caja reductora la cual permite desplazar diferentes pesos, de acuerdo a relación que tenga la caja, según la página oficial de Pololu la relación de la caja reductora va desde 5:1 hasta 1000:1 como se puede ver en la tabla 2.1; el torque de cada motor está en función de la relación que tenga la caja reductora; estos motores deberán ser escogidos de acuerdo a la velocidad que se requiera y lo más importante de acuerdo al peso del robot, generalmente para robots laberintos la relación usada es de 30:1. (Pololu, 2017)

Tabla 2. 1: Micromotores de alta potencia

6 V	High-power (HP)	1600 mA	6000 RPM	2 oz-in	5:1 HP 6V
			300 RPM	4 oz-in	10:1 HP 6V
			1000 RPM	9 oz-in	30:1 HP 6V
			625 RPM	15 oz-in	50:1 HP 6V
			400 RPM	22 oz-in	75 HP 6V
			320 RPM	30 oz-in	100:1 HP 6V
			200 RPM	40 oz-in	150:1 HP 6V
			140 RPM	50 oz-in	210:1 HP 6V
			120 RPM	60 oz-in	250:1 HP 6V
			100 RPM	70 oz-in	298:1 HP 6V
			32 RPM	125 oz-in	1000:1 HP 6V

Tabla de velocidades del micro motor según la relación que tenga la caja reductora, Fuente: (Pololu, 2017)

2.5 SENSORES

Actualmente en el mercado existen una gran gama de sensores a la disposición del usuario, estos deberán ser escogidos de acuerdo a las necesidades a cubrir. Los más usados en las competencias para robot diferenciales son los sensores ópticos y ultrasónicos. Los señores ópticos Sharp como el de la figura 2.3, funcionan mediante luz mientras que los sensores ultrasónicos usan ondas de sonido. Si el sensor tiene una respuesta rápida, la velocidad y precisión en el desplazamiento del robot aumenta, caso contrario la probabilidad de rozar las paredes es mayor. (Pololu, 2017)

Figura 2.3: Sensor Sharp de 2 a 15 cm

Sensor óptico Sharp Gp2y0a21yk, Fuente: (Pololu, 2017)

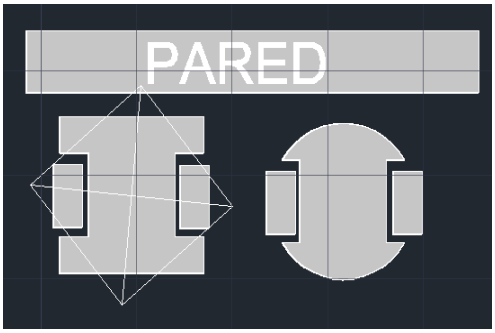
2.6 DIMENSIONES

Las dimensiones del robot están limitadas por el reglamento de la categoría respectiva del concurso, que para este caso en particular establece, que las dimensiones no deben sobrepasar los 15 x 15 cm. de largo-ancho y los 20 cm. de alto. La distancia de pared a pared es de 25 cm., por esta razón se ha optado por realizar un robot de dimensiones inferiores a las especificaciones, para facilitar los giros, especialmente los de 180 grados.

2.7 FORMA

La forma del chasis está determinada por la disposición de los elementos y la misma que puede ser cuadrada, circular, rectangular, etc., como se observa en la figura 2.4. Se debe ubicar los elementos distribuyendo el peso simétricamente, el elemento más pesado es la batería, la misma que se debe ubicar en el centro del robot, sobre la línea del eje de los motores, para que el robot tenga una mejor tracción. En este proyecto se optó por la forma circular, ya que permite que el robot gire libremente con un menor riesgo de colisiones, siendo la forma circular más eficiente en la distribución del espacio, peso y movilidad.

Figura 2.4: Influencia del chasis en el comportamiento del robot



Influencia del chasis en el giro del robot, Autores: Jonathan Aguilar y Héctor Sandoval.

2.8 ROBOTS AUTÓMATAS

La autonomía de un robot es la propiedad que permite al robot acatar órdenes dadas por una programación previa; se habla que un robot es autónomo cuando para su movimiento no es necesario la intervención de una persona basta con accionar un botón de prender para que el robot empiece a moverse solo. (Samaniego, 2009)

2.9 COMUNICACIONES INALÁMBRICAS

Se llama comunicación inalámbrica porque carece de cables que conecten el emisor con el receptor, la comunicación entre ambas se da a través de la modulación de ondas electromagnéticas en el aire. Para este proyecto se escogió utilizar el bluetooth para enviar datos del robot hacia la el ordenador ya que es un dispositivo popular y fácil de utilizar dispone de 4 hilos dos de polarización y dos de datos. El módulo bluetooth se puede observar en la imagen 2.5. (Pietrosemoli, 2013)

Figura 2.5: Bluetooth HC – 05

Pines de conexión para el bluetooth HC 05, Fuente (core-electronics, 2015)

2.10 ODOMETRÍA

Dentro de los métodos de medida de posición relativa se encuentran métodos tales como el giróscopo basado en un sensor capaz de medir la orientación con respecto al plano de la tierra, la navegación inercial, cuyo enfoque va dirigido a realizar mediciones tanto de la orientación y aceleración del robot, y el método odométrico el cual fue usado en la realización de este proyecto y será detallado a continuación.

La odometría es un sistema sensorial el cual busca dar solución a la pregunta básica ¿Dónde Estoy? en el desarrollo de robots móviles. Se centra en la utilización de encoders para medir la rotación de las ruedas y la orientación del robot, siempre que este se desplace en planos horizontales. Este método en especial no requiere de

ningún factor externo, a diferencia del método giróscopo, sin embargo mientras se realiza el desplazamiento, se produce la acumulación de error de medida, por lo que puede resultar ineficaz en trayectorias largas, eventualmente serán necesarias calibraciones recurrentes las cuales serán subyacentes al entorno.

Entran en consideración errores tales como la resolución de los encoders, diferencias en el diámetro de las ruedas, desalineamiento en los ejes, los cuales pueden considerarse sistemáticos, mientras que errores tales como la uniformidad del suelo, tracción de las ruedas y objetos inesperados en la trayectoria pueden considerarse como no sistemáticos. (Somolinos Sánchez, 2002)

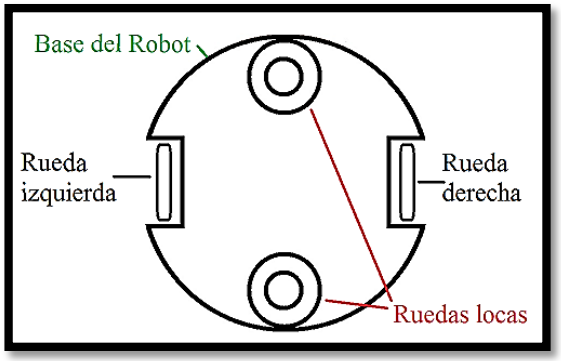
2.11 ESTIMACIÓN DE POSICIÓN POR ODOMETRÍA

La única forma de obtener información del entorno en el que se ubica el robot es por medio de sensores, cuyas mediciones jamás soportarán precisiones absolutas, lo que conllevo a que no se pueda conocer la posición del robot con un 100% de exactitud. Las posiciones aproximadas a partir del método aplicado son relativas al punto de inicio. Cave recalcar que dichas mediciones jamás estarán exentas de errores. El punto de inicio está definido como el punto desde el cual se empieza a calcular la posición del robot, siendo este un punto escogido arbitrariamente en el entorno.

2.12 CINEMÁTICA DIFERENCIAL

La configuración cinemática por la que se optó, se denomina diferencial debido a que cuenta con dos ruedas, donde cada una de estas tiene su propio motor.

Figura 2.6: Esquema del Robot



Esquema del Robot, Autores: Jonathan Aguilar y Héctor Sandoval.

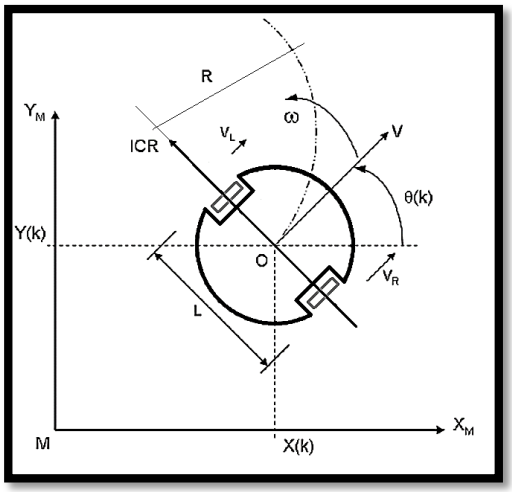
El robot se diseñó con una base circular como se observa en la figura 2.6, además consta de dos ruedas motrices, cuyos ejes se encuentran centrados el uno respecto al otro. Cada rueda posee un motor y un encoder propio. Se añadió dos ruedas locas con el fin de mantener la estabilidad del robot. En si el conjunto que conforma el robot móvil diferencial está conformado por dos encoders, uno a la derecha y uno a la izquierda, los cuales realizan mediciones en unidades de pulsos/revolución referente a cada eje. La resolución del encoder es proporcional a la cantidad de pulsos necesarios para concretar una revolución de tal forma que la distancia entre cada pulso representa un valor angular concreto. Para el desarrollo del proyecto fue un factor clave estimar la posición del robot, mediante el vector (x, y, θ).

2.13 MODELAMIENTO CINEMÁTICO

La línea que contiene el eje de ruedas es el Centro de Rotación Instantáneo o IRC la cual se muestra en la figura 2.7, y está definida por el sistema de coordenadas:

$$IRC = (x - R\ sin\theta;\ x + R\ cos\theta) \qquad \text{Ec. (2.1)}$$

Figura 2. 7: Esquema Cinemático del Robot.



Cinemática del robot, Fuente: (Bermúdez Bohórquez, 2003)

Donde R es el radio de curvatura de la trayectoria y θ es el ángulo de orientación de la base del robot respecto al plano. Considerando que el IRC puede ser variable a lo largo del tiempo, es posible describir cualquier tipo de trayectoria plana siendo la cinemática de su movimiento descrita a partir del sistema de ecuaciones:

$$\omega(t) = \frac{V_R(t)}{R + \frac{L}{2}} = \frac{V_L(t)}{R - \frac{L}{2}} \quad \text{Ec. (2.2)}$$

$$V(t) = R * \omega(t) \quad \text{Ec. (2.3)}$$

Donde

ω: Es la velocidad angular.

V: La velocidad lineal del robot.

R: El radio de curvatura.

V_R y V_L : Las velocidades de las ruedas.

L : La longitud.

Desarrollando las ecuaciones (2.2) y (2.3) se obtienen las siguientes relaciones entre las velocidades lineales de cada rueda respectivamente y la velocidad angular.

$$\omega(t) = \frac{V_R(t) - V_L(t)}{L} \quad \text{Ec. (2.4)}$$

$$V(t) = \frac{V_R(t) - V_L(t)}{2} \quad \text{Ec. (2.5)}$$

$$R = \frac{L}{2}\frac{(V_R(t) + V_L(t))}{(V_R(t) - V_L(t))} \quad \text{Ec. (2.6)}$$

El desplazamiento sobre el plano estará dado por $\bar{x} = [x, y, \theta]$, planteadas estas condiciones, se puede presentar un modelo cinemático referenciado en el mismo robot, considerando que los grados de libertad serán únicamente 2, ya que para el eje Y coincidente con el ICR del robot la capacidad de movimiento es nula.

$$V(t) = \frac{V_R(t) + V_L(t)}{2} = \frac{r(\omega_R(t) + \omega_L(t))}{2} \quad \text{Ec. (2.7)}$$

$$\omega(t) = \frac{V_R(t) + V_L(t)}{L} = \frac{r(\omega_R(t) + \omega_L(t))}{L} = \dot{\theta}(t) \quad \text{Ec. (2.8)}$$

Donde

ω_R y ω_L: Las velocidades angulares de cada rueda.

r: El radio de las ruedas.

El modelo obtenido no es aun útil para la implementación de trayectorias o sistemas de navegación inteligentes, por lo cual se requiere referenciar el modelo a un punto espacial conocido. Este nuevo modelo se define mediante las ecuaciones:

$$\dot{x}(t) = V(t)\cos\theta(t) \qquad \text{Ec. (2.9)}$$

$$\dot{y}(t) = V(t)\ \sin\theta(t) \qquad \text{Ec. (2.10)}$$

$$\dot{\theta}(t) = \omega(t) \qquad \text{Ec. (2.11)}$$

De donde remplazando los valores de $V(t)$ y $\omega(t)$ se obtiene:

$$\dot{x}(t) = \frac{r(\omega_R(t) + \omega_L(t))}{2}\cos\theta(t) \qquad \text{Ec. (2.12)}$$

$$\dot{y}(t) = \frac{r(\omega_R(t) + \omega_L(t))}{2}\sin\theta(t) \qquad \text{Ec. (2.13)}$$

$$\dot{\theta}(t) = \frac{r(\omega_R(t) + \omega_L(t))}{L} \qquad \text{Ec. (2.14)}$$

2.14 REDES NEURONALES RECURRENTES

Este tipo de tecnología trata de reproducir el proceso usado por el cerebro para la resolución de problemas, aplicando el conocimiento y experiencia adquirida a nuevos problemas de manera que se pueda constituir en un sistema de toma de decisiones. Dentro de las características de las redes neuronales se tiene:

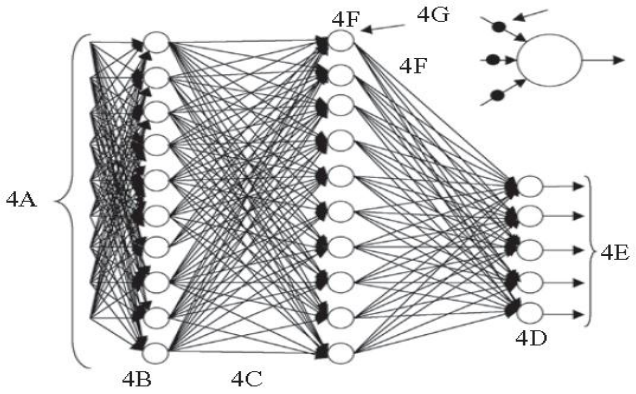
- Pueden sintetizar algoritmos a través de un proceso de aprendizaje.
- La solución de problemas no lineales.
- Son robustas.
- Se deben entrenar para cada problema.
- Necesitan muchos datos.

- Representan un aspecto complejo para un observador externo que desee realizar cambios.

Las redes neurales como la que se observa en la figura 2.8 generalizan información obtenida del proceso de experimentación, bases de datos, determinadas por humanos, estas se entrenan mediante un algoritmo de retro propagación, actualizando los pesos, además de un conjunto ordenado de entradas - salidas deseadas y la comparación entre la salida actual y la salida deseada de la red neural. Otro método de entrenamiento es un clasificador de forma binaria: sólo existe la posibilidad de ser parte de un grupo A o B; funciona con sistemas lineales.

Las redes neuronales recurrentes, tienen como característica principal estar exentas de restricciones en lo referente a la conectividad, lo que permite conexiones entre neuronas creando ciclos o bucles, a diferencia del perceptrón.

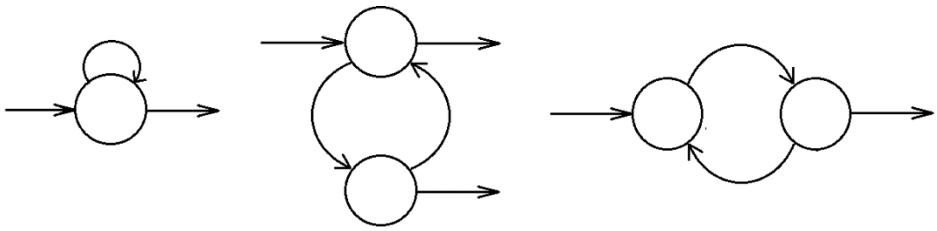
Figura 2. 8: Esquema de la red neural multicapas



Red Neuronal multicapas, Fuente:(Ponce Cruz, 2010)

El uso de conexiones recurrentes en una red neuronal implica el aumento de pesos o parámetros ajustables en la red esto se puede ver en la figura 2.9, por lo que la información gana más representación, sin embargo este efecto tiene consecuencias directas en el proceso del aprendizaje, como la variación del tiempo de convergencia de la red.

Figura 2. 9: Esquemas de redes neuronales recurrentes



Redes neuronales recurrentes, Fuente:(Isasi Viñuela, Pedro; Galván León, Inés;, 2004)

Para el modelo de una red neuronal recurrente es necesario considerar la variable tiempo en la activación o estado de una neurona.

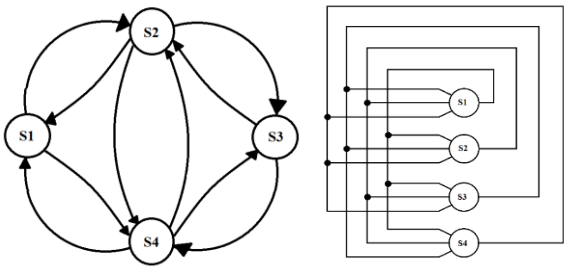
$$a_i(t+1) = f_i\left(\sum_j \omega_{ji}\, a_j(t)\right) \qquad \text{Ec. (2.15)}$$

Donde j varía en el conjunto de todas las neuronas conectadas a la neurona i. Las neuronas recurrentes facilitan el manejo de la información temporal, es decir toda aquella información que depende del tiempo cuyo valor en un instante actual está ligado a los valores anteriores en instantes pasados de tiempo.

2.15 RED DE HOPFIELD

Esta red fue desarrollada por John Hopfield en la década de los ochenta, siendo un modelo de red neuronal no lineal como se puede ver en la figura 2.10, se la clasifica como recurrente. Esta red esta presentada como un modelo de memoria asociativa de patrones, siendo capaz de recuperar patrones almacenados a partir de información incompleta. Las neuronas están conectadas con todas las demás, y actúa como una memoria asociativa, procesando patrones en los que generalmente no interviene la variable tiempo, siendo estos estáticos.

Figura 2.10: Representación de la Red de Hopfield para n=4



Red de Hopfield, Fuente:(Isasi Viñuela, Pedro; Galván León, Inés;, 2004)

Las neuronas en una red de Hopfield están conectadas a todas las demás neuronas menos a ellas mismas. Las conexiones de la red forman una matriz cuadrada $W = (\omega_{ji})$ donde ω_{ji} es el peso de la conexión entre la neurona i y la neurona j. Las características más relevantes de esta matriz son:

- Es simétrica, de tal forma que el peso de la conexión de la neurona i hacia la neurona j es igual al peso de la conexión de la neurona j hacia la neurona i.
- Los elementos de la diagonal principal siempre serán igual a cero.

En general las neuronas de la red de Hopfield poseen dos estados, siendo estos -1 y 1, el estado de la neurona i para un tiempo $t + 1$ se denota como:

$s_i(t+1) = sgn\big(v_i(t+1)\big)$ para $i = 1,2,\dots,$ Ec. (2.16)

Donde

$sgn\big(v_i(t+1)\big) = 1 \quad si \quad v_i(t+1) > 0$ Ec. (2.17)

$sgn\big(v_i(t+1)\big) = -1 \quad si \quad v_i(t+1) < 0$ Ec. (2.18)

Mientras que el nivel de activación $v_i(t+1)$ en la neurona i se presenta como:

$$v_i(t+1) = \sum_{j=1}^{n} \omega_{ji} s_j(t) - u_i \quad para \quad i = 1,2,\dots,n \qquad \text{Ec. (2.19)}$$

Donde s_j es el estado de la neurona en un instante anterior, y u_i es un umbral fijo. De ser $v_i(t+1) = 0$ se asume que el estado de la neurona no ha cambiado.

Una red de Hopfield con n neuronas se deferiría como:

$s_i(t+1) = [s_1(t+1), s_2(t+1), \dots, s_n(t+1)]^t$ Ec. (2.20)

2.16 APRENDIZAJE EN LA RED DE HOPFIELD

Se distinguen dos fases de operación, siendo estas la fase de almacenamiento y la fase de recuperación. En la primera fase se determinaran los valores a tomar por los pesos dela red y la segunda fase consiste en un mecanismo para recuperar la información almacenada a partir de información incompleta.

Para almacenar patrones el peso de la conexión entre la neurona j y la neurona i se define como:

$$\omega_{ji} = \sum_{k=1}^{p} x_j(k)\, x_i(k) \qquad \forall \qquad i \neq j \qquad \text{Ec. (2.21)}$$

Donde $x(k)$ es un vector de n dimensiones perteneciente al conjunto de patrones, cuyas componentes toman valores binarias, -1 o 1.

En la fase de recuperación, teniendo un patrón de prueba $x = (x_1, x_2, \dots, x_n)$, se inicializan los estados de las n neuronas de la red utilizando dicho patrón.

$$s_i\,(0) = x_1 \quad para\ i = 1,2,\dots,n \qquad \text{Ec. (2.22)}$$

Posteriormente se calcularan los estados de la red, mediante las ecuaciones 2.16, 2.17, 2.18 y 2.19, hasta llegar a un momento en el que los estados de la neurona se mantengan invariantes respecto al tiempo. Este estado representa el patrón recuperado.

$$s_i\,(t+1) = s_i(t) \qquad \forall \quad i = 1,2,\dots,n \qquad \text{Ec. (2.23)}$$

2.17 FUNCIÓN DE ERROR EN LA RED DE HOPFIELD

Para una red de Hopfield, con n neuronas y con conexiones $W = (\omega_{ij})$ siendo W una matriz simétrica, con ceros en su diagonal, la función de error asociado es:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq 1}^{n}\omega_{ij}s_i s_j + \sum_{i=1}^{n} u_i s_i \qquad \text{Ec. (2.24)}$$

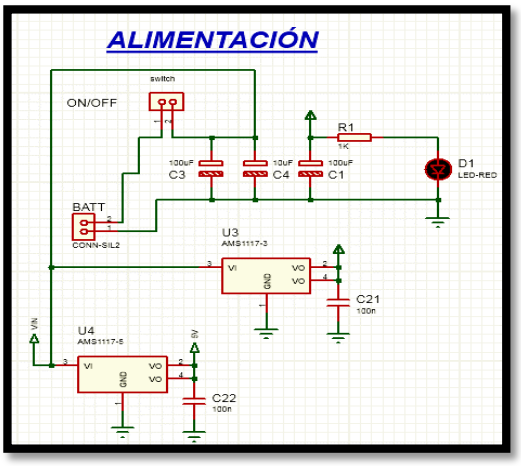
CAPÍTULO 3
DESARROLLO DISEÑO ELÉCTRICO Y ELECTRÓNICO

En este capítulo se detalla el procedimiento realizado para poner en funcionamiento el robot diferencial, se detalla el diseño eléctrico y electrónico realizado en el software Proteus 8.6 Profesional. Además se indica todos los dispositivos utilizados así como las conexiones que se hizo con los diferentes dispositivos electrónicos, además, se indicara el procedimiento mecánico.

3.1. ALIMENTACIÓN

La alimentación del sistema será provista por una batería LIPO de 7.4 voltios, esta batería proporciona 600 mAh, energía suficiente para que el robot funcione alrededor de unos 60 minutos continuos, debido a que el microprocesador funciona con 3.3V, es necesario regular el voltaje para evitar sobrecargas, los sensores funcionan a 5 voltios por lo que necesitan que el voltaje se regule esto se puede observar en la figura 3.1.

Figura 3.1: Alimentación del sistema



Sistema de alimentación para el funcionamiento del robot, Autores: Jonathan Aguilar y Héctor Sandoval.

Se utiliza el botón (Switch) para prender el robot y poner en marcha todo el sistema a continuación se colocan los capacitores C3 y C4 para eliminar picos y estabilizar el voltaje, a la salida se colocan dos reguladores con tecnología SMD, el primer capacitor es el AMS1117-3 el cual limita el voltaje a 3.3V y con este se alimenta el

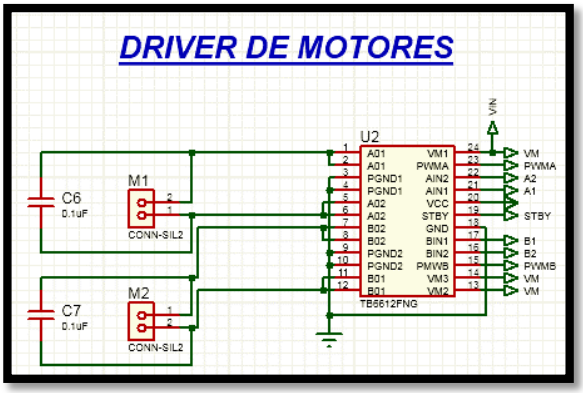
micro procesador y las entradas del mismo, de igual manera se coloca un capacitor C21 para estabilizar el sistema, seguidamente se coloca un segundo regulador AMS1117-5 este limita el voltaje a 5V para los sensores, drivers para control de motores y los dos motores, en el prototipo se incluye un led que indica que el robot está listo para su funcionamiento.

3.2. DRIVE PARA CONTROL DE LOS MOTORES

Para el control de los motores se utiliza el drive TB6612FNG cuyas características principales son: una frecuencia de conmutación de hasta 100 kHz, permite controlar dos motores a la vez y utiliza tecnología CMOS la cual minimiza las pérdidas de energía haciéndolo ideal para obtener una rápida respuesta de los motores.

Este drive tiene dos entradas de voltaje, una entrada destinada a la alimentación de los motores VM, que va desde los 3V hasta los 13V, y la segunda alimentación VCC de 5V destinada al funcionamiento del drive. El motor M1 se conecta a las salidas A01 y A02 en paralelo con un capacitor C6, para eliminar los sobre picos de voltaje, el motor M2 se conecta a las salidas B01 y B02 de igual manera se coloca un capacitor en paralelo C7, también dispone de las entradas PMWA y PWMB para el control de velocidad de los motores M1 y M2 respectivamente, las entradas A1 y A2 controlarán el sentido de giro del motor M1 siempre y cuando la primera entrada se encuentre alto y la otra en bajo, si las dos están en alto, el motor frenará y si las dos están en bajo el motor no girará, análogamente las entradas B1 y B2 controlan en sentido de giro del motor M2, la entrada STBY permite habilitar el drive, se deberá poner en alto para que este funcione.

Figura 3.2: Drive TB6612FNG

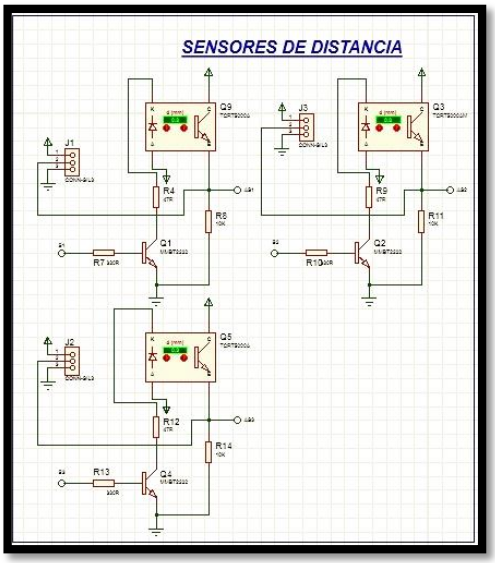


Conexiones del drive para control de los motores, Autores: Jonathan Aguilar y Héctor Sandoval.

3.3. SENSORES DE DISTANCIA

Para este proyecto se utilizaron 3 sensores de distancia Sharp los cuales tienen un rango de medición de 2 a 15 cm y un periodo de actualización de 16,5 $\pm$5 ms siendo adecuados para el prototipo, este sensor utiliza tres hilos dos de alimentación y uno de datos, en la figura 3.3 se puede observar el esquema de conexión de los sensores J1, J2, J3; para efectos de simulación se utilizó el sensor TCR5000A debido a que Proteus 8.6 no dispone del modelo exacto del sensor a ser usado.

Figura 3.3: Sensores Sharp



Conexiones del sensor Sharp de distancia óptico, Autores: Jonathan Aguilar y Héctor Sandoval.

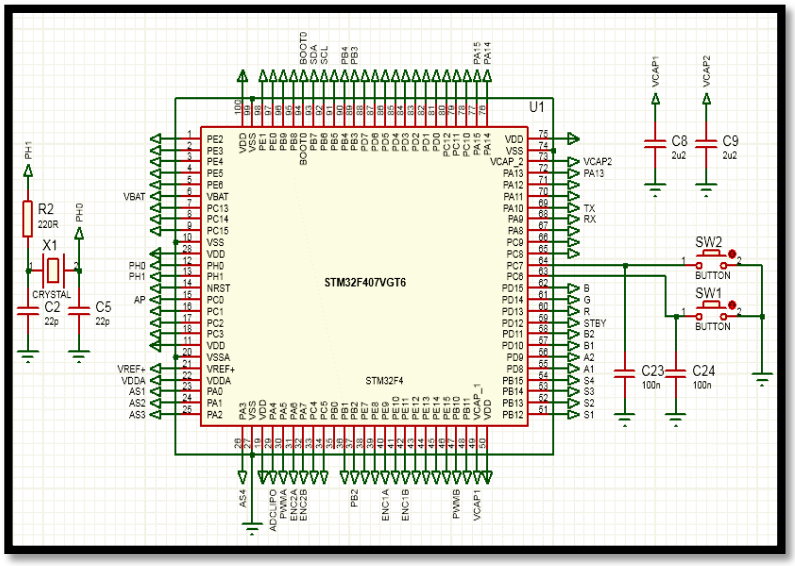
3.4. BLUETOOTH

Se usa para comunicar el microcontrolador con el ordenador, posee una comunicación serial a 4 hilos dos de polarización y dos de datos (Rx y Tx) los cuales van conectados a los pines 68 y 69 respectivamente.

3.5. MICROCONTROLADOR STM32F407

Como se observa en la figura 3.4, descontando los pines tanto de VDD y VSS, se usaron 28 pines del microcontrolador. En la tabla 3.1 se especifican los pines usados y la función que desempeñan dentro del desarrollo del prototipo.

Figura 3.4 Microcontrolador STM32F407

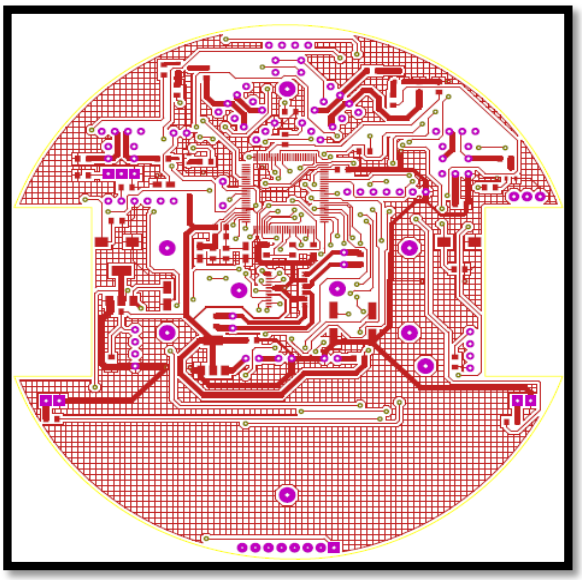


Conexiones del microcontrolador para su funcionamiento, Autores: Jonathan Aguilar y Héctor Sandoval.

3.6. DISEÑO DE LA PLACA DEL ROBOT EN PROTEUS 8.6

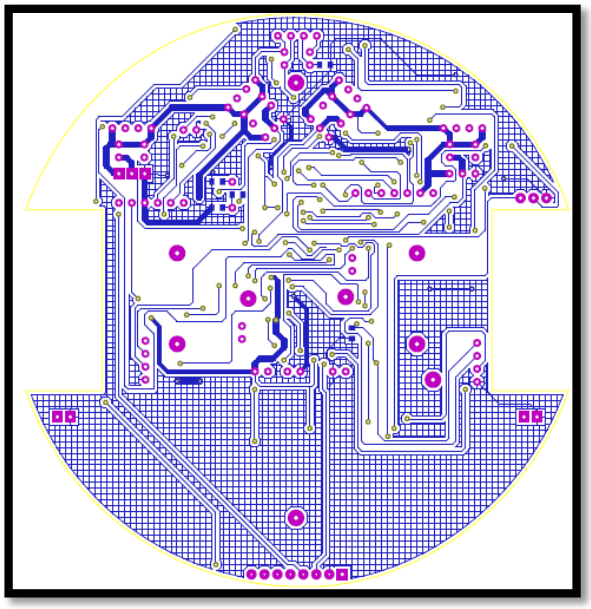
El diseño de la placa del robot es a doble capa. En la capa superior (véase la figura 3.5) van los sensores, bluetooth, pulsadores, microcontrolador ARM, batería lipo, resistencias, leds y comunicación USB (para cargar el programa al micro). En la figura 3.6 se muestra el diseño de la placa inferior en la cual van algunas resistencias y capacitores además de los motores y rueda loca.

Figura 3.5: Vista parte superior de la placa



Pistas de la vista superior de la placa, Autores: Jonathan Aguilar y Héctor Sandoval.

Figura 3.6: Vista parte inferior de la placa

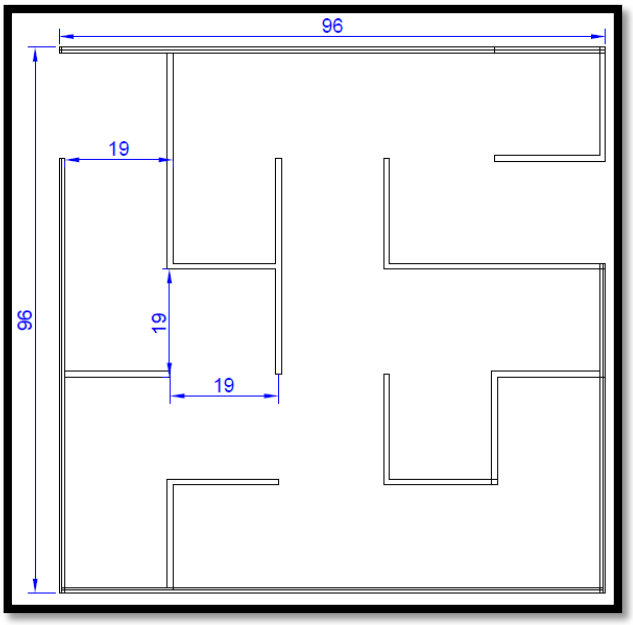


Pistas de la vista inferior de la placa, Autores: Jonathan Aguilar y Héctor Sandoval.

3.7. DISEÑO DEL LABERINTO

Se diseñó el laberinto para que este no sea fijo sino modular, las dimensiones se pueden ver en la figura 3.7, la madera que se utilizó es MDF con un espesor de 1 cm y alto de 15 cm, las paredes pintadas de color blanco mate.

Figura 3.7: Diseño de la pistas



Dimensiones de la pista, Autores: Jonathan Aguilar y Héctor Sandoval.

Tabla 3 1: Descripción de pines de entrada y salida de la STM32

PINES	DESCRIPCIÓN	ACCIÓN
TB6612FNG		
pinPD12	STBY	Permite activar o desactivar el Drive
pinPA5	PWMA	Control velocidad motor A
pinPD11	B2	Giro Derecha
pinPD10	B1	Giro Izquierda
pinPB10	PWMB	Control velocidad motor B
pinPD9	A2	Giro Derecha
pinPD8	A1	Giro Izquierda
VDD	VDD	Alimentación 5 v
VSS	VSS	Alimentación GND
ENCODER		
pinPA6	ENC2A	Conteo de pulsos
pinPA7	ENC2B	Dirección
pinPE9	ENC1A	Conteo de pulsos
pinPE11	ENC1B	Dirección
BLUETOOTH		
pinPA10	TX	Transmisión
pinPA9	RX	Recepción
VDD	VDD	Alimentación 5 v
VSS	VSS	Alimentación GND
SWITCH		
pinPC7	SW1	Inicia funcionamiento del robot
pinPC6	SW2	Extra
CRISTAL		
pinPH0	PH0	CRISTAL 8MHz
pinPH1	PH1	CRISTAL 8MHz
POLARIZACIÓN DEL MICROCONTROLADOR		
VDD	VDD	Alimentación 5 v
VSS	VSS	Alimentación GND
ACTIVACIÓN DE SENSORES SHARP		
pinPD12	S1	Activación
pinPB10	S2	Activación
pinPB14	S3	Activación
PINES DE LOS SENSORES SHARP		
pinPA0	AS1	Sensor Izquierda
pinPA1	AS2	Sensor Frente
pinPA2	AS3	Sensor Derecha
LEDS INDICADORES		
pinPD13	R	Indica Procesamiento de red neuronal
pinPD14	G	Indica presencia de nodos
pinPD15	B	Indica que se ha conectado la batería
PROGRAMADORA		
pinPA13	SWDIO	Datos entrada / salida

pinPA14	SWCLK	Reloj
VCC	3.3V	Alimentación 3.3 v
GND	GND	Alimentación GND

Distribución de pines del microcontrolador para su funcionamiento, Autores: Jonathan Aguilar y Héctor Sandoval.

3.8. REQUISITOS PARA QUE FUNCIONE CORRECTAMENTE EL PROGRAMA

El programa final fue diseñado y compilado en MATLAB R2013a, motivo por el que se requiere instalar un compilador C compatible con la versión de MATLAB, el cual es Visual Studio 2010 y deberá estar licenciado, caso contrario no funcionará, una vez instalado el Visual Studio 2010, mediante el uso del comando MEX –SETUP en la ventana principal se selecciona el compilador C, como se puede observar en la figura 3.8, en el caso de que no se haya instalado el compilador C en MATLAB, no se podrá cargar el programa en el microcontrolador STM32.

Figura 3.8: Instalación del compilador C en MATLAB

```
>> mex -setup

Welcome to mex -setup.  This utility will help you set up
a default compiler.  For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2013a/win64.html

Please choose your compiler for building MEX-files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Microsoft Visual C++ 2010 in c:\Program Files (x86)\Microsoft Visual Studio 10.0

[0] None

Compiler: 1

Please verify your choices:

Compiler: Microsoft Visual C++ 2010
Location: c:\Program Files (x86)\Microsoft Visual Studio 10.0

Are these correct [y]/n? y
```

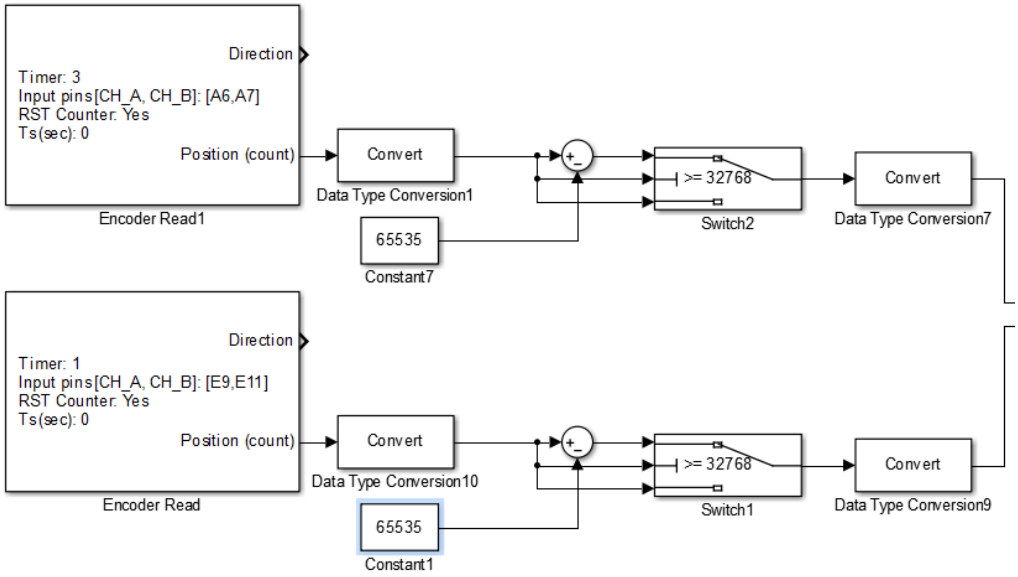
Comando mex-setup, Matlab, Autores: Jonathan Aguilar y Héctor Sandoval.

3.9. PROGRAMACIÓN DE LA ODOMETRÍA

Los bloques encoder Read sirven para la lectura de los pulsos enviados por los encoders, en el tiempo de muestreo establecido, que es de 0.01 segundos, si el

desplazamiento es en avance, se suma un pulso y si es en retroceso se resta un pulso, una vez realizada la lectura, se cuentan los pulsos y la variable de acumulación vuelve a cero para contar los pulsos en el siguiente tiempo de muestreo, el número de pulsos se resta de 65535 ya que el PWM de la STM32 es de 16 bits, este valor ingresa al bloque Switch y se compara, si el valor ingresado es mayor a 32768 es positivo caso contrario es negativo, entonces a la salida del bloque se tiene un conteo de pulsos con signo. El bloque Convert convierte el dato de ingreso en un dato compatible con el bloque siguiente, el código se puede observar en la figura 3.9.

Figura 3. 9 Lectura de pulsos de los encoders

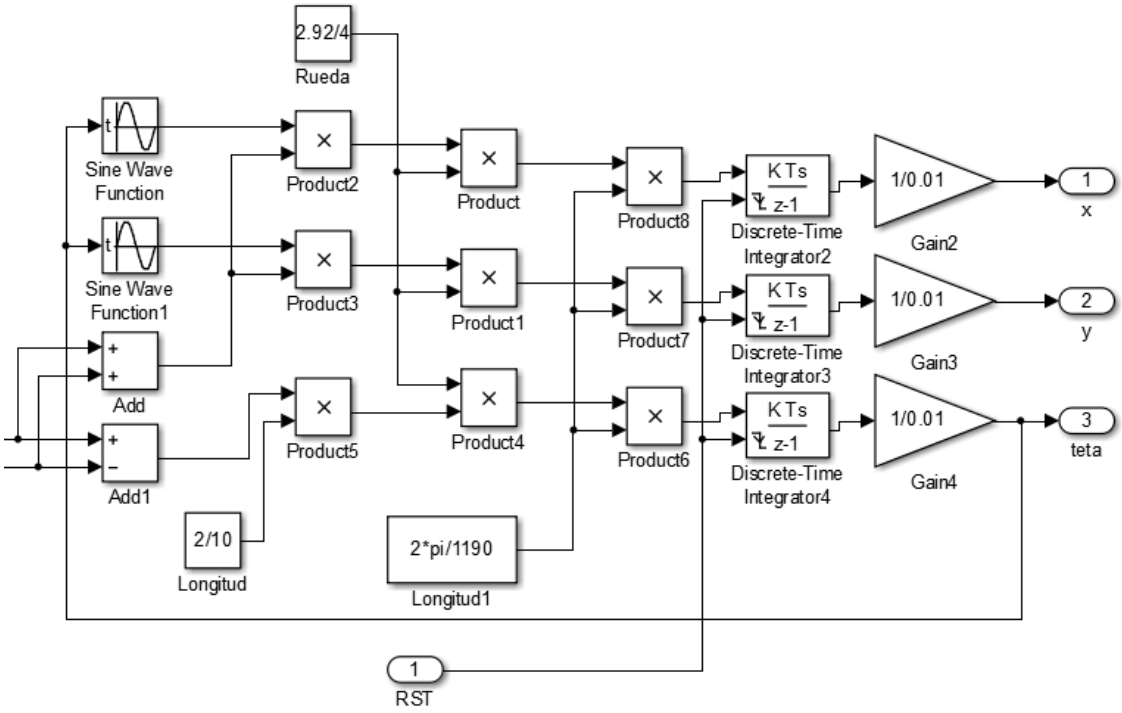


Bloques en SIMULINK, Lectura de encoders, Matlab, Autores: Jonathan Aguilar y Héctor Sandoval.

En la figura 3.10 se tiene la aplicación de las ecuaciones de odometría, con el objetivo de transformar los pulsos de cada encoder en velocidad angular utilizando la ecuación (25), este valor obtenido se ingresa en las ecuaciones (12), (13) y (14) para obtener la posición en X, Y y el ángulo θ.

$$\mathrm{w} = \frac{2\pi * \#\text{pulsos}}{\text{pulsos totales}} \left[\mathrm{rad}/\mathrm{s}\right] \qquad \text{Ec. (3.1)}$$

Figura 3.10: Ecuaciones de odometría

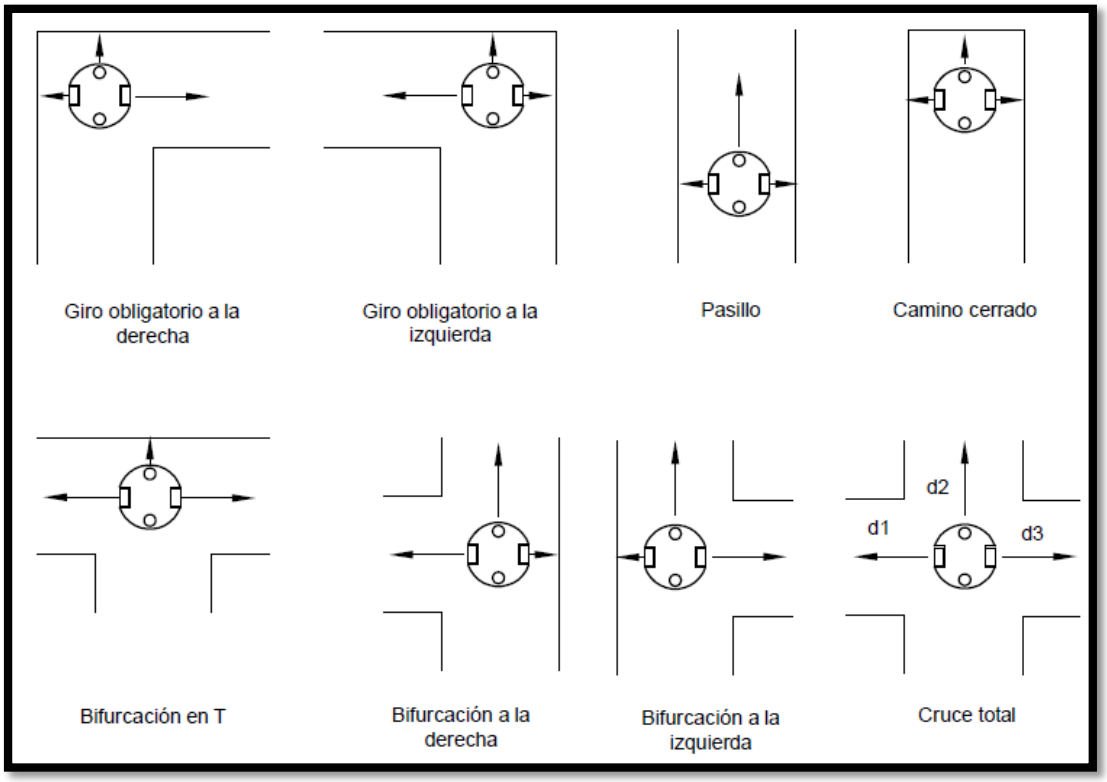


Bloques en SIMULINK, ecuaciones odométricas, Autores: Jonathan Aguilar y Héctor Sandoval.

3.10. ZONIFICACIÓN

Se han considerado 8 zonas posibles dentro del entorno en el que se desplazara el robot, los cuales serán identificados dependiendo de los rango de mediciones entregados por los tres sensores Sharp como se observa en la figura 3.9. Estos datos analógicos son enviados a un ADC y digitalizados para posteriormente ser transformados a distancias mediante un bloque Matlab function el cual utiliza la ecuación de linealización, la cual se obtuvo experimentalmente para los sensores Sharp. La zonificación ayudará a generar nodos los cuales se guardarán en una matriz para posteriormente ser enviados a la red neuronal, las zonas que generen nodos serán las que tengan más de dos opciones de giro tales como la bifurcación en T, bifurcación derecha, izquierda y cruce total, las demás no generan nodos ya que tienen un solo giro obligatorito, la zonificación se puede observar en la figura 11.

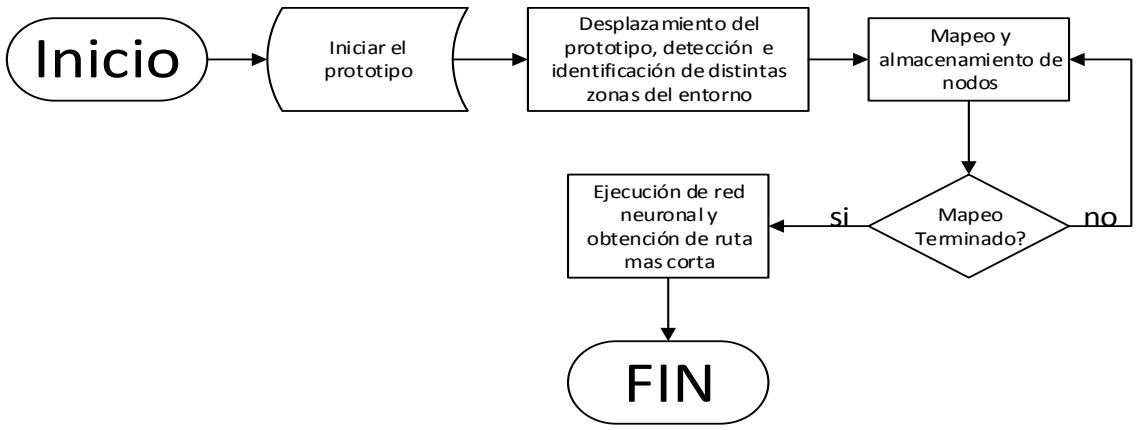
Figura 3.11: Detección de zonas



Detección zonas para el giro del robot, Autores: Jonathan Aguilar y Héctor Sandoval.

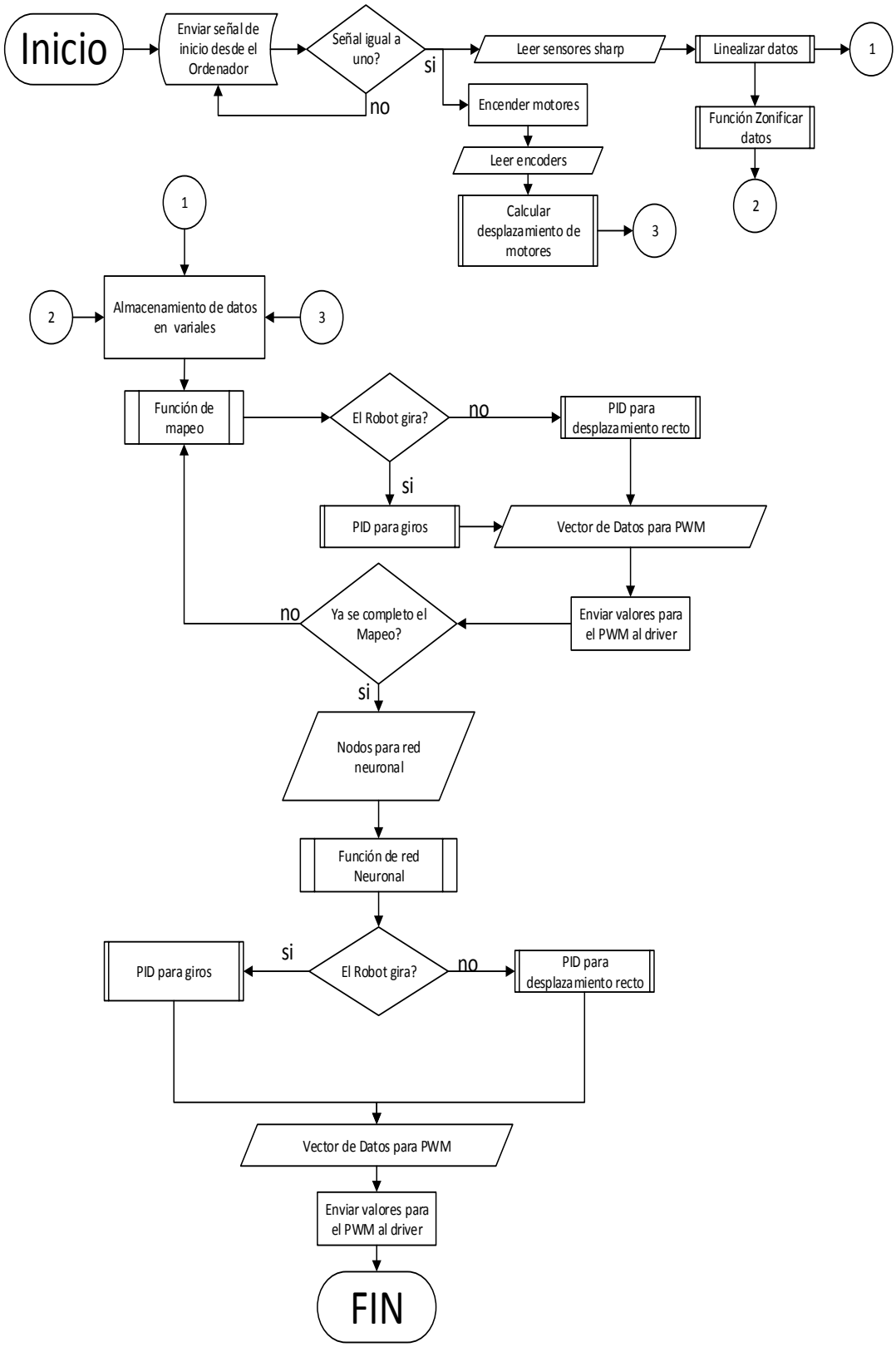
3.11. FLUJOGRAMA GENERAL DE LA APLICACIÓN

El siguiente flujograma representa el funcionamiento general del dispositivo.



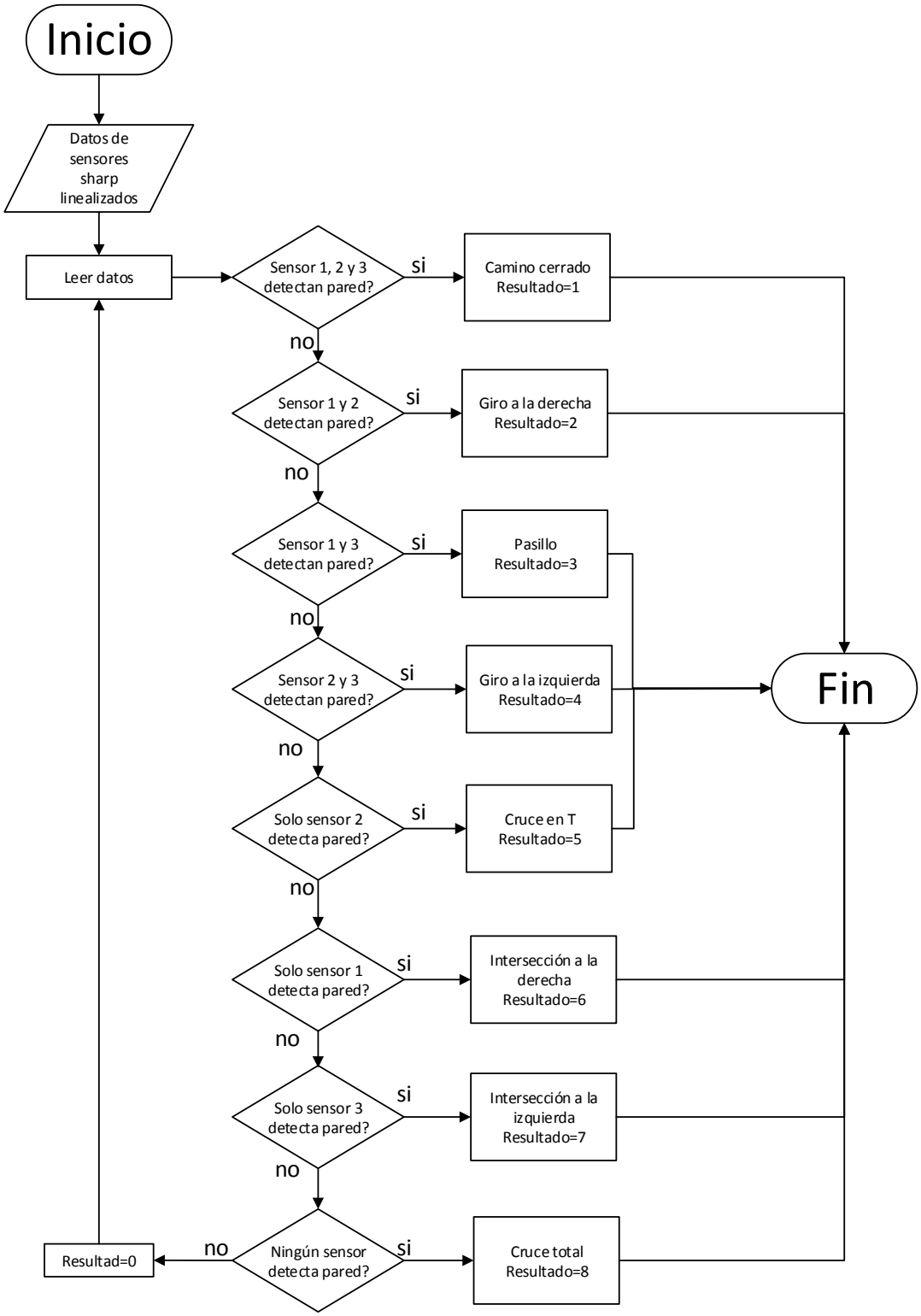
3.12. FLUJOGRAMA ODOMETRÍA

El siguiente flujograma representa el proceso Odométrico, zonificación y mapeo.



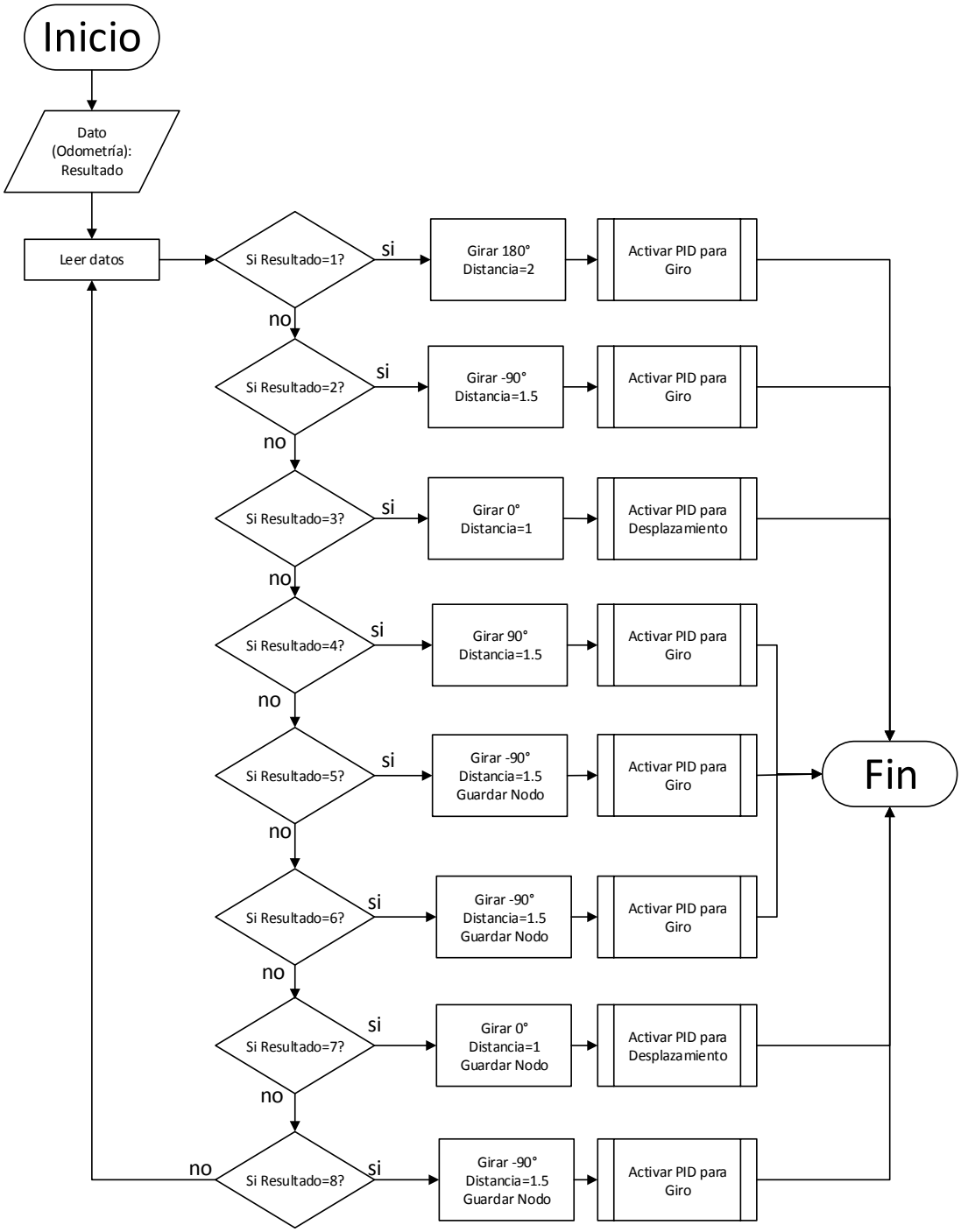
3.13. FLUJOGRAMA ZONIFICACIÓN

El siguiente flujograma detalla el proceso de zonificación, y asignación de un valor a cada posibilidad.



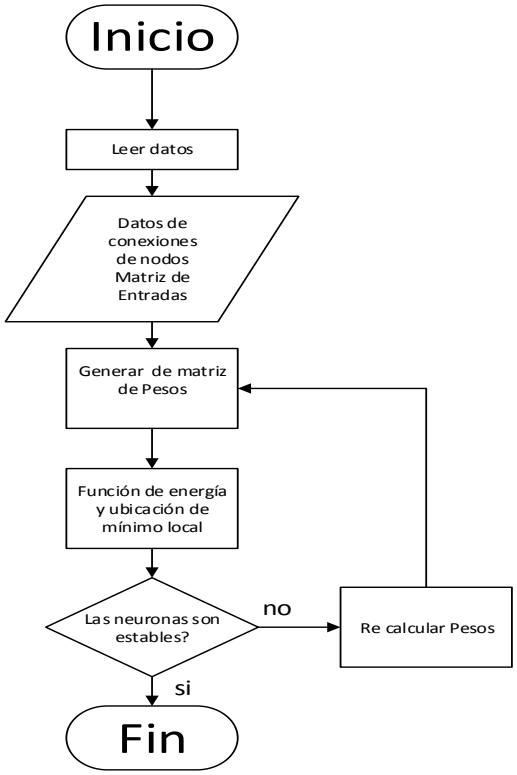
3.14. FLUJOGRAMA MAPEO

El siguiente flujograma detalla el proceso de mapeo, el ángulo de giro y el PID a ser activado.



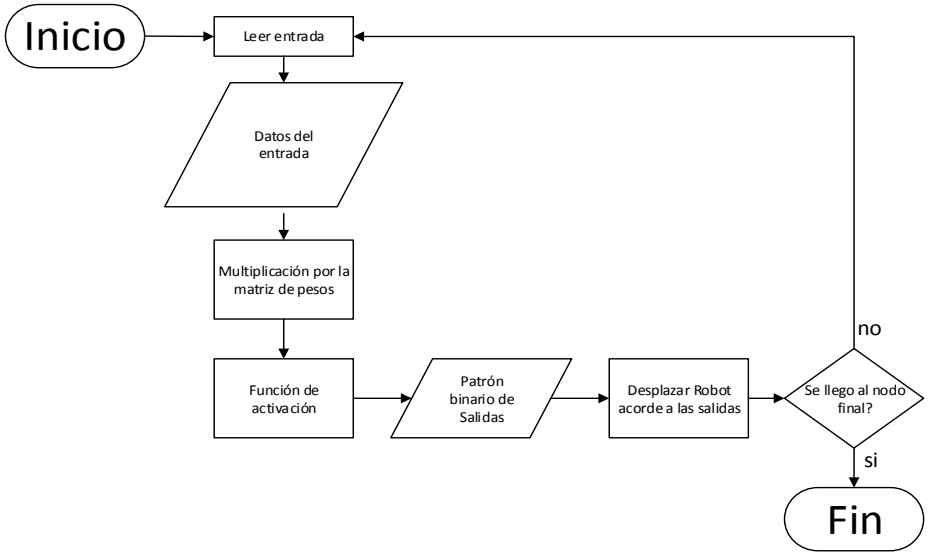
3.15. FLUJOGRAMA DE APRENDIZAJE

El siguiente flujograma detalla el proceso de aprendizaje, realizado por la neurona.



3.16. FLUJOGRAMA DE CONSULTA

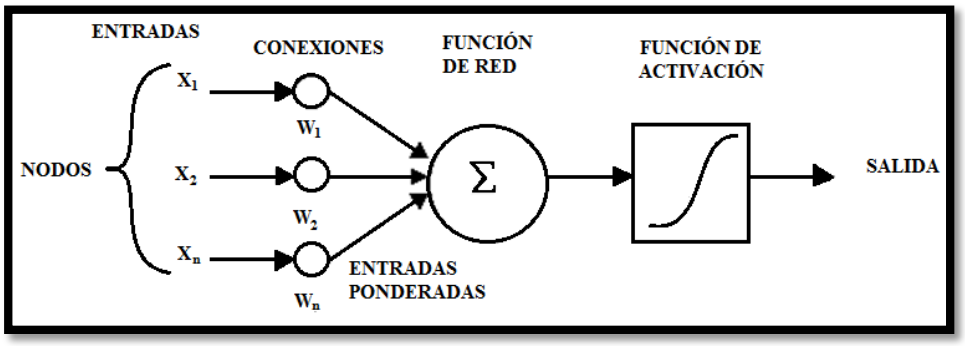
El siguiente flujograma detalla el proceso de consulta, realizado por la neurona.



3.17. RED NEURONAL

La estructura más común de una red neuronal se observa en la figura 3.12; consta de entradas, pesos, función de red, función de activación y salidas.

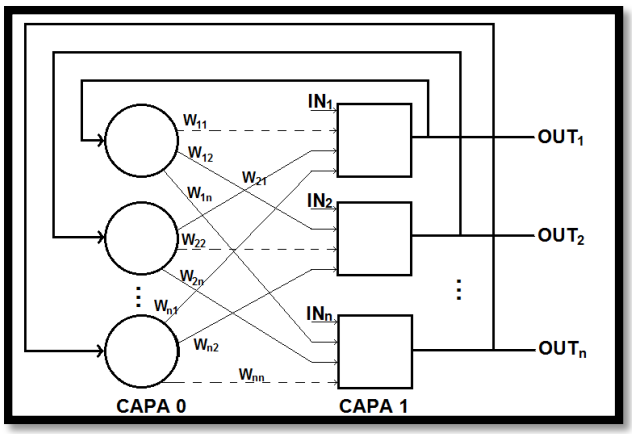
Figura 3.12: Esquema de red Neuronal



Esquema de red Neuronal, Autores: Jonathan Aguilar y Héctor Sandoval.

En una red de retroalimentada, como se observa en la figura 3.13, para una siguiente iteración la primera salida se realimenta como entrada, originado una nueva salida. El aprendizaje para este sistema no conlleva la modificación de los pesos los cuales desde un inicio son fijos, consiste en hallar pesos, en relación a las características del problema.

Figura 3.13: Esquema de red Neuronal



Esquema de red Neuronal, Autores: Jonathan Aguilar y Héctor Sandoval.

La red busca encontrar un camino desde el nodo de origen s al nodo de destino d de manera que el costo total en el camino es mínimo.

Para este caso, la matriz de nodos C, es la matriz en donde cada par ordenado representa la existencia o la no existencia de una conexión entre cada uno de los nodos con todos los demás nodos. Para el caso del nodo uno, existen dos conexiones, tanto con el nodo dos como con el nodo tres, por ende el nodo tres y el nodo dos presentaran una conexión con el nodo uno, dicha conexión se representa con un número uno en la poción $C(1,2)$, $C(1,3)$, $C(2,1)$ y $C(3,1)$.

Cada nodo presenta una conexión inclusive consigo mismo, pero ya que el nodo no se realimenta a sí mismo el valor de esta conexión es cero. Las entradas son valores binarios, cada elemento de la matriz se trata como una neurona, por ende el número de neuronas seria nueve. (Wen Liu, Lipo Wang, 2005)

La red neuronal está dispuesta en una matriz $n \times n$, los elementos de la diagonal han sido suprimidos. En resumen una neurona ubicada en (x, i), describir el enlace entre el nodo x y el nodo i.

$$C(x,i) = \begin{bmatrix} 0&1&1&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0&0 \\ 1&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1 \\ 0&0&1&0&0&1&0&0&0 \\ 0&0&0&0&1&0&1&0&0 \\ 0&0&0&0&0&1&0&1&1 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&1&0&0&1&0&0 \end{bmatrix}$$

La topología de la conexión o indicador de conexión se denotada por $P(x,i)$ y se define por:

$$P(x,y) = \begin{cases} 1 \ ; si\ no\ existe\ conexión \\ 0 \ ; si\ \ existe\ conexión \end{cases}$$

$$P(x,i) = \begin{bmatrix} 0&0&0&1&1&1&1&1&1 \\ 0&0&1&0&1&1&1&1&1 \\ 0&1&0&1&0&1&1&1&1 \\ 1&0&1&0&1&1&1&1&0 \\ 1&1&0&1&0&0&1&1&1 \\ 1&1&1&1&0&0&0&1&1 \\ 1&1&1&1&1&0&0&0&0 \\ 1&1&1&1&1&1&0&0&1 \\ 1&1&1&0&1&1&0&1&0 \end{bmatrix}$$

La diagonal principal dela matriz de topología se mantiene con valores de cero, mientras que la solución de enrutamiento se define mediante la salida final $V(x,i)$ y se define por:

$$V(x,y) = \begin{Bmatrix} 1\ ;\ si\ existe\ una\ conexión\ en\ la\ solucion\ final\ entre\ las\ neuronas \\ 0\ ;\ si\ no\ existe\ una\ conexión\ en\ la\ solución\ final\ entre\ las\ neuronas \end{Bmatrix} \qquad Ec.(3.2)$$

La función de energía consta de un nuevo término para garantizar que el flujo sea unidireccional hasta el destino, pero al mismo tiempo este término hizo que los pesos de conexión de la red neuronal dependieran de la topología de la red de comunicación.

$dim: representa\ el\ número\ de\ neuronas$;

$s: nodo\ salida$;

$d: nodo\ de\ llegada$

$$\mathrm{E} = \frac{u_1}{2}\sum_{x=1}^{dim}\sum_{i=1}^{dim} C(x,i)V(x,i) + \frac{u_2}{2}\left[\sum_{x=1}^{dim}\left(\sum_{i=1}^{dim} V(x,i) - \sum_{i=1}^{dim} V(x,i) - \gamma_x\right)^2\right] + \sum_{x=1}^{dim}\sum_{i=1}^{dim} P(x,i) * V(x,i) \qquad Ec.(3.3)$$

$Donde\ siempre\ \ i \neq x$
$Si\ X = 0\ \ entonces\ E = 0$

$$\gamma_x = \begin{Bmatrix} 1\ ;\ x = s \\ -1\ ;\ x = d \\ 0\ ;\ para\ otro\ caso \end{Bmatrix} \qquad Ec.(3.4)$$

Tanto el término γ_x como los términos u_{i} son valores ponderados, donde el término u_1 minimiza el costo en la ruta mientras término u_2 representa restricciones, siendo estas:

- En cada nodo individual, excepto el origen y el destino, el número de enlaces entrantes debe ser igual al de enlaces salientes.
- Penalizar las neuronas que representan enlaces no existentes en la red de comunicación.

Los valores iniciales recomendados para u_1 y u_2 son 300 y 2500 respectivamente. Ya que solo son dos parámetros, el ajuste de los mismos es más eficiente. Estadísticamente se puede plantear que el sistema estará perturbado por ruidos, de tal forma que puede ser representado mediante la Ec. 3.5.

$$\frac{dU(x,i)(t)}{dt} = \frac{1}{\tau}U(x,i)(t) + \sum_{y=1}^{dim}\sum_{j=1}^{dim}\sum_{x=1}^{dim}\sum_{i=1}^{dim} W(\mathrm{y,j,x,i}) * V(y,j)(t) + I(x,i) \qquad Ec.(3.5)$$

Para el caso práctico de la programación, se implementa la Ec. 3.6, donde $U(x,i)$, es la función de transferencia, cuya salida ingresa a la función de activación, en la Ec. 3.10.

$$U(x,i) = \sum_{y=1}^{dim}\sum_{j=1}^{dim}\sum_{x=1}^{dim}\sum_{i=1}^{dim} w(y,j,x,i) * V(y,j) + I(x,i); \qquad Ec.(3.6)$$

La matriz de pesos W es fija, sus características principales son la simetría y una diagonal nula, se puede deducir que la red no aprende nunca, debido a que W es fija, sin embargo, es en si la formación de la matriz W es lo que permite que se dé el aprendizaje.

$$W(\mathrm{y,j,x,i}) = \sum_{y=1}^{dim}\sum_{j=1}^{dim}\sum_{x=1}^{dim}\sum_{i=1}^{dim} -k2[\delta(x,y) - \delta(x,j) + \delta(i,j) - \delta(i,y)] \qquad Ec.(3.7)$$

Donde:

$$\delta(i,j) = \begin{Bmatrix} 1 & ; i = j \\ 0 & ; i \neq j \end{Bmatrix} \qquad Ec.(3.8)$$

Siendo $\delta(i,j)$ la función delta de Kronecker $\delta(i,j)$, es una función de dos variables.

I(x, i) es el bias de entrada de la neurona (x,i). Es un parámetro dependiente de la función de salida neuronal. Los costos se interpolan en el bias. La ventaja de este procedimiento consiste en evitar la necesidad de ajustar los parámetros internos de la

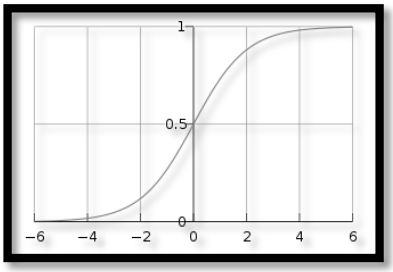
red neuronal para adaptarse a los cambios en el entorno, lo que se desencadena que el implemento de hardware de la red neuronal es general.

$$\mathrm{I(x,i)} = -\frac{k_1}{2} C(x,i)[1-\delta(x,d)\delta(i,s)] + k_2 * \delta(x,d)\delta(i,s) - \frac{k_2}{2} P(x,i)[1-\delta(x,d)\delta(i,s)] \qquad Ec.(3.9)$$

Dentro de la matriz de conexión ninguno de los términos tiene dependencia del costo de los enlaces o la topología, la matriz de conexión es independiente de cualquier cambio en la red de comunicación. La solución al enrutamiento denomina $V(y,j)$, Las funciones de activación más comunes son la función Gaussiana y la función sigmoidal, en este caso se usa una función de activación sigmoide que se puede observar en la figura 3.14.

$$V(y,j) = \sum_{y=1}^{dim} \sum_{j=1}^{dim} \left(\frac{1}{1+e^{-(U(y,j))}} \right) \qquad Ec.(3.10)$$

Figura 3.14: Función Sigmoide



Función Sigmoide, $y = \frac{1}{1+e^{-X}}$, Autores: Jonathan Aguilar y Héctor Sandoval.

Mediante la función $\mathrm{Us(x,i)}$ definida como se observa en el Ec. 3.11, mediante el valor resultante de $\mathrm{V(x,i)}$, se obtienen valores binarios, los cuales representan las conexiones entre los nodos que forman la ruta más eficiente, es decir la ruta de menor costo.

$$\mathrm{Us(x,i)} = \begin{cases} 1 \ ; \mathrm{V(x,i)} \geq 0.5 \\ 0 \ ; \ \mathrm{V(x,i)} \ < \ 0.5 \end{cases} \qquad Ec.(3.11)$$

Es decir, cuando el valor resultante $\mathrm{V(x,i)}$ entre la conexión de un nodo x con un nodo i es mayor o igual a o 0.5, dicha conexión es parte del conjunto de conexiones denominado como la ruta más corta entre un nodo de salida y un nodo de llegada.

Para la matriz $C(x,i)$, cuto nodo de salida era el nodo uno y el de llegada el nodo nueve, la solución se plantea como:

$$Us(x,i) = \begin{bmatrix} 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

Cuya interpretación es:

- El nodo uno presenta una conexión con el nodo dos.
- El nodo dos presenta una conexión con el nodo cuatro.
- El nodo 4 presenta una conexión con el nodo nueve.

De tal forma que las demás conexiones, son suprimidas, obteniendo la ruta más corta.

3.18. DESPLAZAMIENTO POR RUTA MÁS CORTA

Previamente en el mapeo, la conexión entre cada nodo encontrado, tiene una distancia y un giro almacenados, que conjuntamente con la obtención de $Us(x,i)$, permiten que el robot, pueda desplazarse de forma autónoma, en la ruta más corta entre dos nodos.

La matriz resultante $Us(x,i)$, indica las conexiones validas, mientras que la distancia indica si existe una curva o es un desplazamiento en línea recta, para que finalmente, de existir una curva, el dato del ángulo defina si esta será de $90°$ o $-90°$.

CAPÍTULO 4
PRUEBAS Y RESULTADOS

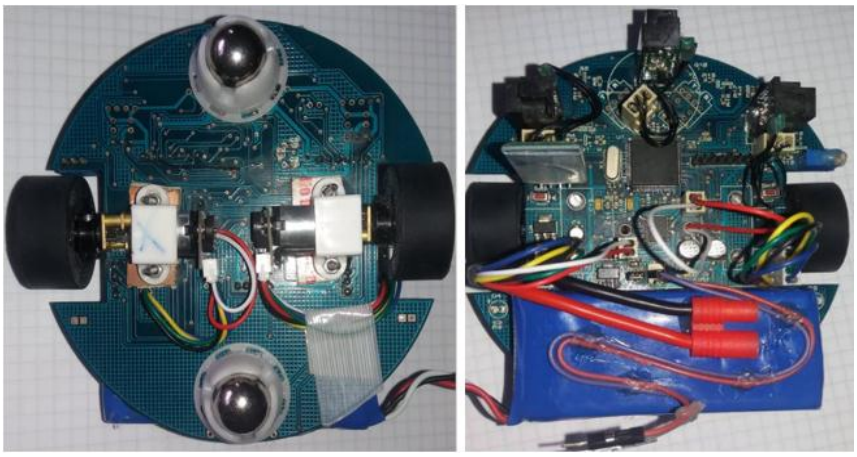
En este capítulo se presentan las pruebas realizadas al prototipo para mostrar su correcto funcionamiento, como parte de las pruebas se muestran los diferentes laberintos en los cuales se ha probado el prototipo.

4.1 PRUEBAS DE HARDWARE

Toda la parte electrónica se ha diseñado en una baquelita de doble lado y con circuitos SMD para minimizar el área que ocupa cada elemento, en dicha placa se encuentran todos los elementos que conforman el robot tales como sensores, pulsadores, leds, batería, microcontrolador STM32, drive para control de motores y bluetooth. Al colocar y soldar los elementos sobre la placa el inconveniente que se tuvo es que al ser tecnología SMD los elementos son muy pequeños y los puntos de suelda tienen que hacerse con mucho cuidado.

Una vez soldado todos los elementos, se realizan todas las conexiones de los motores, los sensores, la batería y el bluetooth con la placa, siguiente a este se realizan pruebas del funcionamiento de los diferentes dispositivos dando como resultado un funcionamiento excelente de acuerdo a lo diseñado, el resultado final se puede observar en la figura 4.1.

Figura 4.1: Prototipo final



Prototipo final colocado todos los dispositivos, Autores: Jonathan Aguilar y Héctor Sandoval.

En la prueba de los sensores Sharp se tuvo un inconveniente especialmente con los sensores laterales ya que en un principio se los colocó con un ángulo de 45 grados respecto de la pared, el inconveniente se debía a que, al estar en esa posición no detectaba bien cuando entra o sale de una zona, para corregir este problema se colocaron todos los sensores perpendiculares con respecto a la pared, de esta manera se pudo eliminar el problema.

Finalmente se realizaron algunas pruebas de deslizamiento con diferente velocidades, se notó que, con la rueda (a) el robot no se mantenía estable sobre la pista esto se debe a que al tener un labrado no existe suficiente adherencia sobre el piso por lo que el robot tiende a derrapar dando un gran problema con la lectura de los encoders, se realizaron pruebas con la rueda (b) dando como resultado una mejor estabilidad del robot, esto se debe a que dicha llanta tiene una mayor área de contacto con el piso, las llantas de prueba se puede observar en la figura 4,2.

Figura 4.2: Ruedas

a) b)

Ruedas utilizadas para pruebas, (Pololu, Solarbotics RW2i Wheel, 2005)

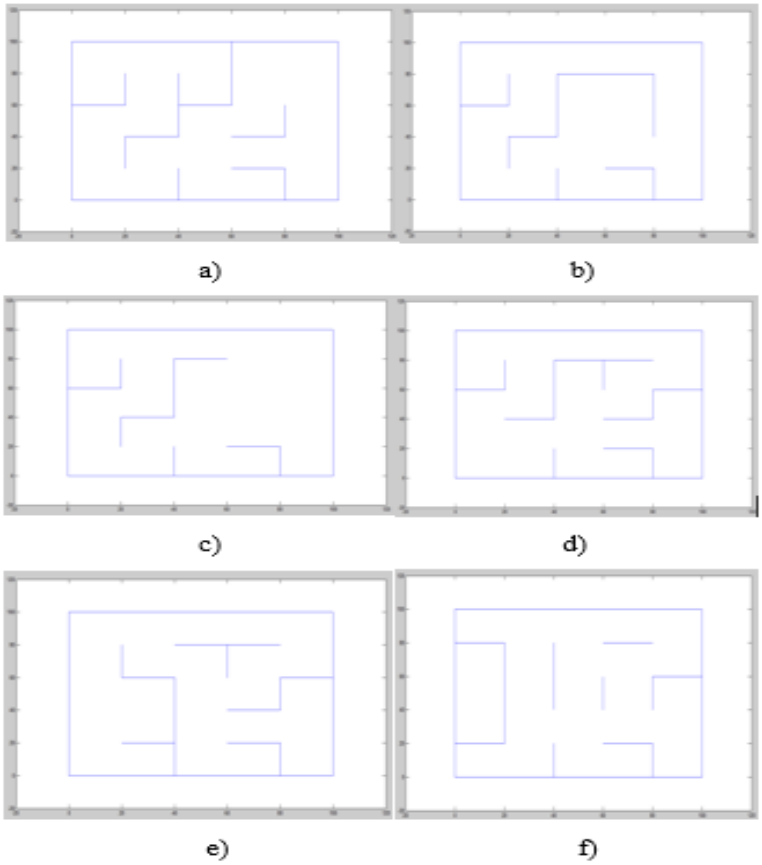
4.2 PRUEBAS DE SOFTWARE

Para mantener el robot en el centro de la pista se realiza un controlador PID tanto para las llantas como para los giros, durante las pruebas realizadas se tuvieron que cambiar los parámetros varias veces hasta obtener como resultado un control aceptable tanto de los giros como del desplazamiento, cabe recalcar que la calibración de los PIDs se realizó con el método de prueba y error.

4.3 PRUEBAS EN PISTA

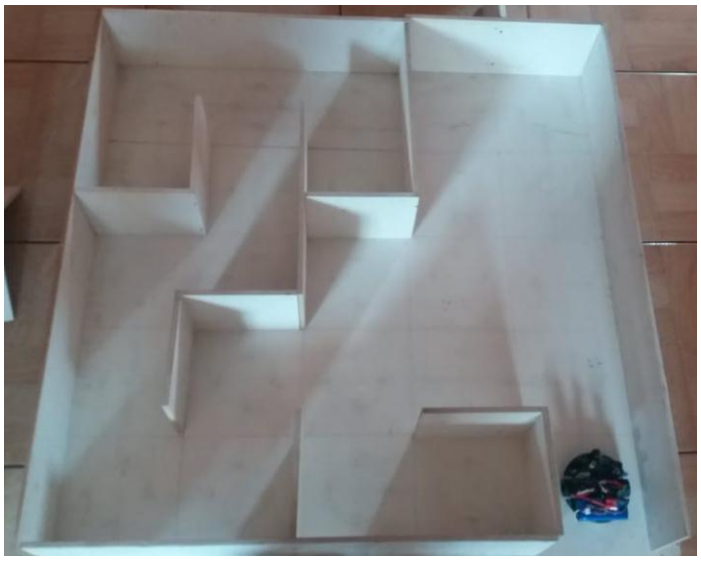
Una vez construido el prototipo y realizado las pruebas necesarias de hardware y software se realizan pruebas sobre diferentes laberintos para verificar el correcto funcionamiento de la red neuronal, el prototipo mapeará el laberinto siguiendo la regla de la mano derecha.

Figura 4.3: Laberintos de prueba



Laberintos implementados para pruebas, Autores: Jonathan Aguilar y Héctor Sandoval.

Figura 4.4: Implementación del laberinto de prueba



Pruebas en laberinto, Autores: Jonathan Aguilar y Héctor Sandoval.

En la figura 4.3 se muestran los diferentes laberintos que se utilizaron para probar que la red neuronal funciona correctamente y en la figura 4.4 muestra la implementación del laberinto (a), el objetivo del robot es salir de un punto inicial y llegar al mismo punto luego de mapear todo el contorno del laberinto identificando los nodos que existen en su trayectoria, cada que detecte un nodo se prendera un led indicador y se apagará en caso de que no haya un nodo, en la tabla 4.1 se muestran los resultados obtenidos.

Tabla 4. 1: Resultados obtenidos

Laberinto	**a**	**b**	**c**	**d**	**e**	**f**
Número de nodos	7	6	9	8	8	9
Número de iteraciones de la neurona	150	300	300	300	600	600 - 3000
Tiempo requerido [s]	6	5	28	17	38	-------

Tabla de valores obtenidos durante las pruebas con el robot, Autores: Jonathan Aguilar y Héctor Sandoval.

Una vez que el prototipo llegue al punto inicial girará 180 grados y empezara el procesamiento de la red para indicar esta acción se prende un led indicador y se apaga cuando ya ha terminado de procesar la red, se prenderá otro led para indicar que el robot va a ejecutar la trayectoria designada por la red neuronal, en la tabla 4.1 a se observa que los tiempos que le lleva a la red aprender y dar la solución son

variados, en primera instancia se observó que a mayor número de nodos más iteraciones necesita la red para llegar a la solución, pero esto no es cierto como se observa en la tabla 4.1, también depende de las distancias entre nodos, se consideran distancias a los puntos que no son nodos, entre mayor distancias entre nodos mayor tiempo le lleva converger a la red. Para un mismo número de nodos pero diferente laberinto a la red le lleva tiempo en converger y en otros casos no llega a la solución como es el caso del laberinto (f), se probó con iteraciones que fueron desde 600 a 3000 iteraciones y no funcionó puede ser debido a que la memoria del micro no es suficiente para procesar toda la información.

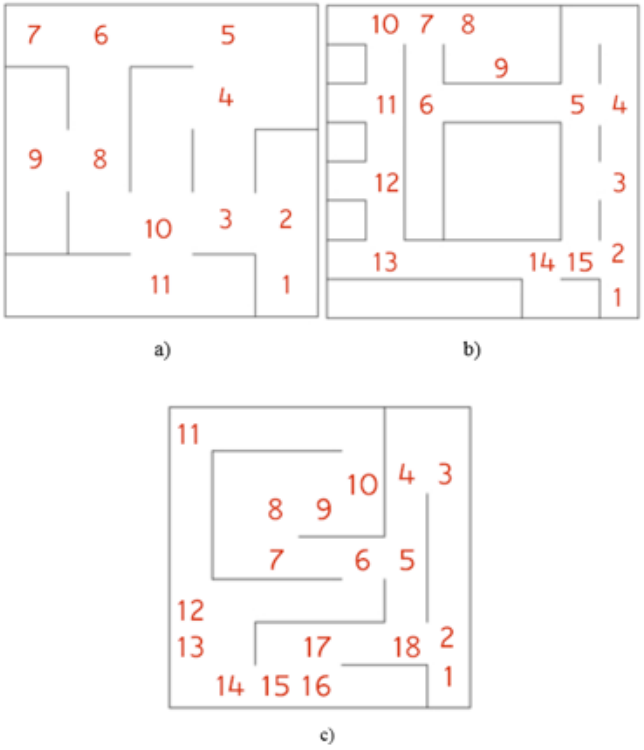
Se utilizó la red neuronal en el computador para resolver laberintos con mayor número de nodos, los laberintos se muestran en la figura 4.5 y los resultados se muestran en tabla 4.2.

Tabla 4. 2: Resultados obtenidos

Laberinto	**a**	**b**	**c**
Número de nodos	11	15	18
Número de iteraciones de la neurona	900	1100	1200
Tiempo requerido [min]	2.26	10	15
Matriz	5 x 5	8 x 8	7 x 7

Tabla de valores obtenidos durante las pruebas con la computadora, Autores: Jonathan Aguilar y Héctor Sandoval.

Figura 4.5 Laberintos de prueba en el ordenador



Diseño de laberintos para pruebas con el ordenador, Autores: Jonathan Aguilar y Héctor Sandoval.

Se observa que con un número de nodos mayor a 10 las iteraciones de la red aumentan y los tiempos en converger también, el mayor tiempo que le lleva a la red en converger fue de 15 minutos en el laberinto (c), si al ordenador le lleva ese tiempo en dar una solución en el micro se demoraría horas y sobre todo la memoria del micro no alcanzaría a procesar.

4.4 CONTROL PID VS CONTROL ON/OFF

En las pruebas realizadas con el controlador ON/OFF se observa que en línea recta el robot presenta balanceos al desplazarse con el fin de mantenerse en el centro de la pista, además en los giros de 90 y 180 grados presenta un error grande como se puede ver en la figura 4.3, mientras que con un controlador PID el error se reduce drásticamente el error y reduce notablemente el balanceo en línea recta.

Tabla 4. 3 Pruebas de un controlador PID vs ON/OFF

Pruebas	Ángulo deseado	Controlador PID	Error [%]	Controlador ON/OFF	Error[%]
1	180	182	1,11	192	6,67
2	90	88	2,22	98	8,89
3	180	181	0,56	172	4,44
4	90	92	2,22	82	8,89

Tabla de valores obtenidos durante ejecución de los controladores, Autores: Jonathan Aguilar y Héctor Sandoval.

4.5 ERRORES

- Si el led indicador del procesamiento de la red se apaga y el robot no se mueve o el robot se mueve a un nodo y se detiene puede deberse a las siguientes circunstancias, las iteraciones de la red son insuficientes para que esta haya llegado a la solución o son demasiadas iteraciones y no llego a la solución.

- Si las paredes del laberinto no están rectas el robot puede colisionar con otra pared y detectar otra zona en la que no está, en ese caso reiniciar el robot e intentar otra vez desde el inicio, caso contrario la red no encontrará la solución ya que los nodos son incorrectos.

- Si al iniciar su trayectoria el robot no se encuentra centrado en la primera casilla puede llegar a un punto donde no detecte bien la zona y se quede dando vueltas.

CONCLUSIONES

Se realizó el diseño eléctrico, electrónico y mecánico acorde a las dimensiones del laberinto, se utilizó componentes SMD puesto que por su reducido tamaño ocupan menos espacio en la placa que un componente de tamaño normal.

En la navegación es necesario saber cuánto se desplazó el robot y que ángulo necesita girar en cada zona, esto se pudo realizar haciendo uso de la odometría que da como resultado posición en X, posición en Y y un ángulo de giro θ.

Para graficar la trayectoria del robot utilizando la interfaz de MatLab fue necesario comunicar el ordenador con el robot utilizando un Bluetooth y las librerías necesarias para la comunicación. La manera más fácil de graficar dicha trayectoria fue dibujar tres zonas con ángulo cero, camino cerrado, giro obligatorio, y pasillo, a medida que el robot avanza ira graficando la zona donde se encuentre con el ángulo que llegue.

Se realizó un algoritmo de mapeo en base a todas las posibilidades de giro que pueda tener el robot en su trayectoria dentro del laberinto, dando como resultado 8 zonas. Con los resultados obtenidos se llenó la matriz de nodos necesaria para la red neuronal.

Gracias a una buena calibración del controlador PID se logró que el robot se mantuviera centrado en su trayectoria y que realizara un giro con mayor precisión que al utilizar un controlador ON / OFF. Por lo que se pudo comprobar que el controlador PID mucho más eficiente.

Se observó que las llantas cumple un rol importante dentro de este diseño ya que al trabajar con encoder y sensores infrarrojos cualquier derrape brusco en la pista provocará primero un error de la zona donde se encuentra el robot y segundo que los giros no serán de acuerdo a los programados ya que el encoder manda los pulsos contados sin tomar en cuenta el derrape de la llanta.

El número de nodos y la distancia entre cada uno de ellos ocasiona que el tiempo que la red converge aumente, además incrementa el número de iteraciones necesarias para llegar a la solución como se confirmó en el capítulo 4.

RECOMENDACIONES

Para realizar este tipo de prototipos los más recomendable desarrollarlo por etapas, primero con el diseño e implementación de la placa de control, luego comprobar que los sensores dan las medidas esperadas, verificar que la comunicación bluetooth funciona y finalmente verificar que el driver que controla los motores funciona correctamente. También el algoritmo se debe desarrollar por etapas para ir verificando que cada una de ellas funcione y en caso de errores corregirlos a tiempo.

Antes de iniciar la comunicación del bluetooth con el ordenador verificar que este tenga el dispositivo integrado luego emparejar con el bluetooth del prototipo.

Al iniciar la navegación del prototipo verificar que la pista esté libre de polvo para evitar un derrape de las ruedas y su mal funcionamiento.

Verificar que la caja de reducción este limpia libre de pelusas ya que la acumulación de esto reduciría el desempeño del motor y provocarían un recalentamiento del motor disminuyendo su vida útil.

Verificar que los encoders estén en buen estado y funcionando caso contrario el robot no cumplirá con sus objetivos, ya que la distancia recorrida así como el ángulo de giro están tomados en base al conteo de pulsos de cada encoder.

Tener mucho cuidado con la conexión de la batería ya que una mala polarización de esta con el circuito provocaría que algún dispositivo deje de funcionar dejando inmóvil al robot.

REFERENCIAS

Bermúdez Bohórquez, G. R. (2003). *Modelamiento cinemático y odométrico de robots móviles: aspectos matemáticos.* Caldas: Universidad Distrital Francisco José de Caldas.

Cortés, F. R. (2011). *Robótica, Control de Robots Manipuladores.* México : Alfaomega.

ElectroniLab. (2010). *Módulo BlueTooth HC-06*. Obtenido de https://electronilab.co/tienda/modulo-bluetooth-hc-06-serial-rs232ttl/

Isasi Viñuela, Pedro; Galván León, Inés;. (2004). *Redes de Neuronas Artificiales un Enfoque Práctico.* Madrid: Pearson.

Ollero Baturone, A. (2001). *Robótica Manipuladores y Robots móviles.* Barcelona: Marcombo.

Pololu. (2005). *Micro Metal Gearmotor LP 6V*. Obtenido de https://www.pololu.com/product/993

Pololu. (2005). *Solarbotics RW2i Wheel*. Obtenido de https://www.pololu.com/product/642

Ponce Cruz, P. (2010). *INTELIGENCIA ARTIFICIAL CON APLICACIONES A LA INGENIERÍA.* México D.F.: Alfaomega.

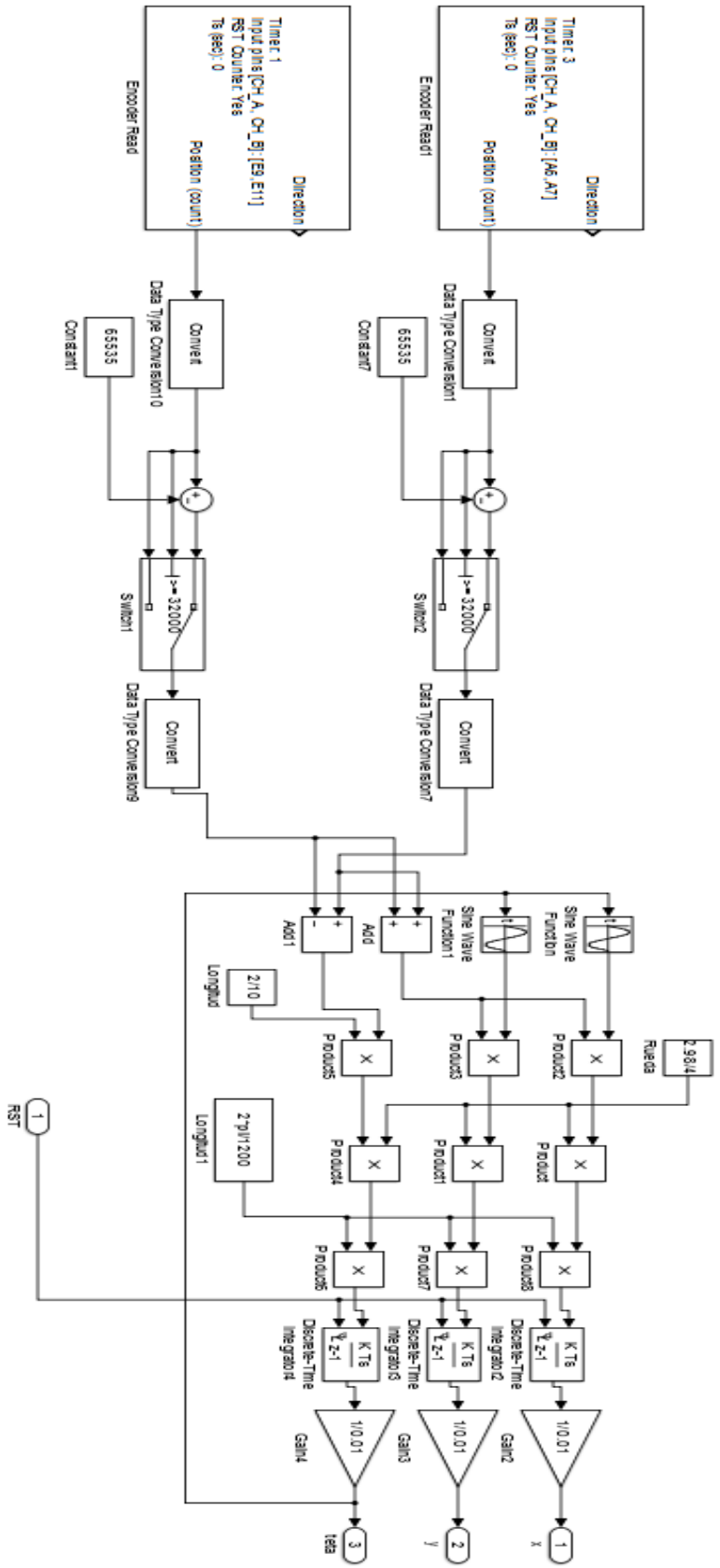
Somolinos Sánchez, J. A. (2002). *Avances en Robótica y Visión por Computador*. Cuenca: Universidad de Castilla - La Mancha.

Venegas Requeme, J. (2009). *Encoders*. Obtenido de http://ramos.elo.utfsm.cl/~elo212/docs/Encoders-jvr-v01.pdf

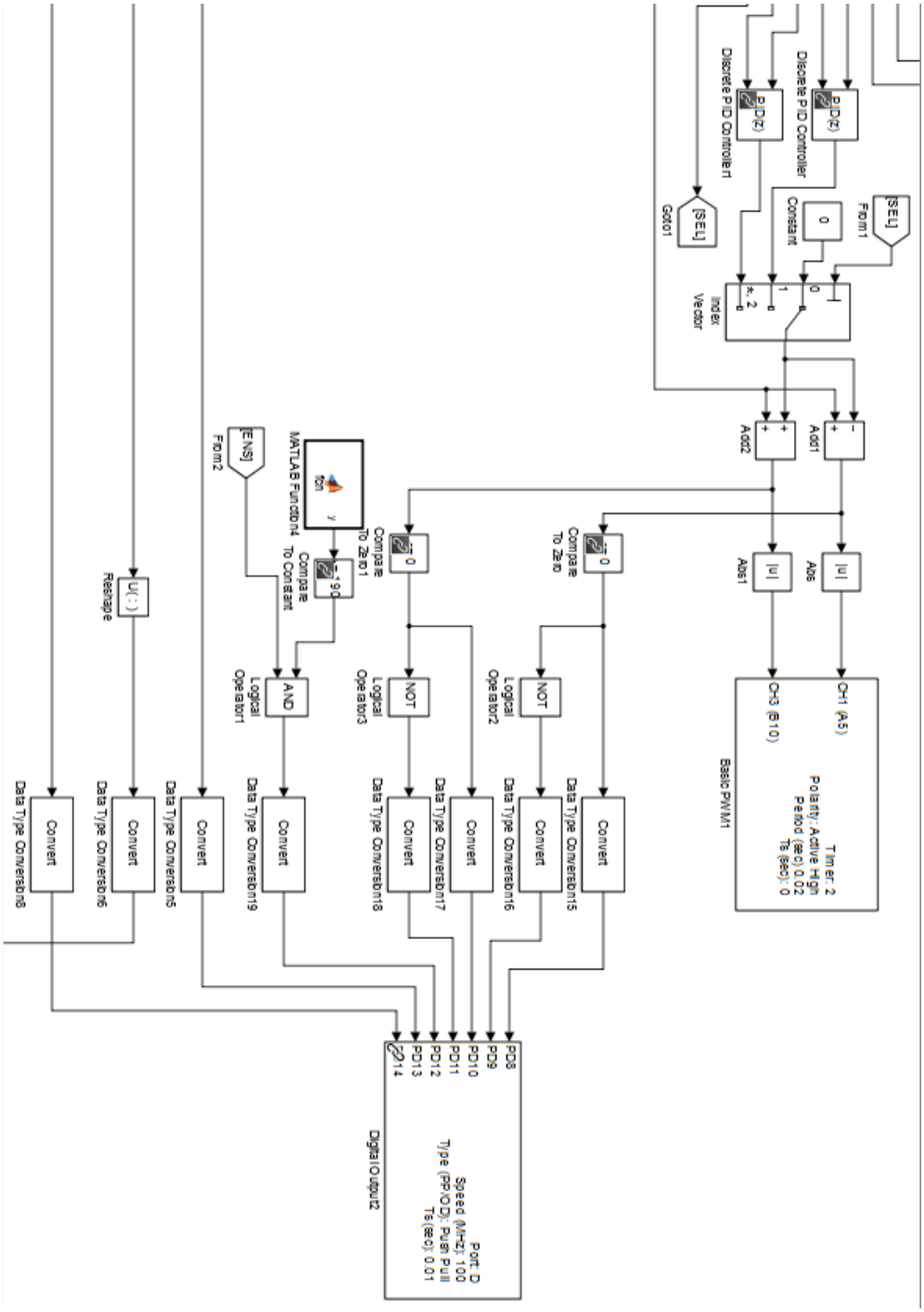
Wen Liu, Lipo Wang. (2005). *Solving the Shortest Path Routing Problem Using Noisy Hopfield Neural.* Singapore: Nanyang Technological University.

ANEXOS

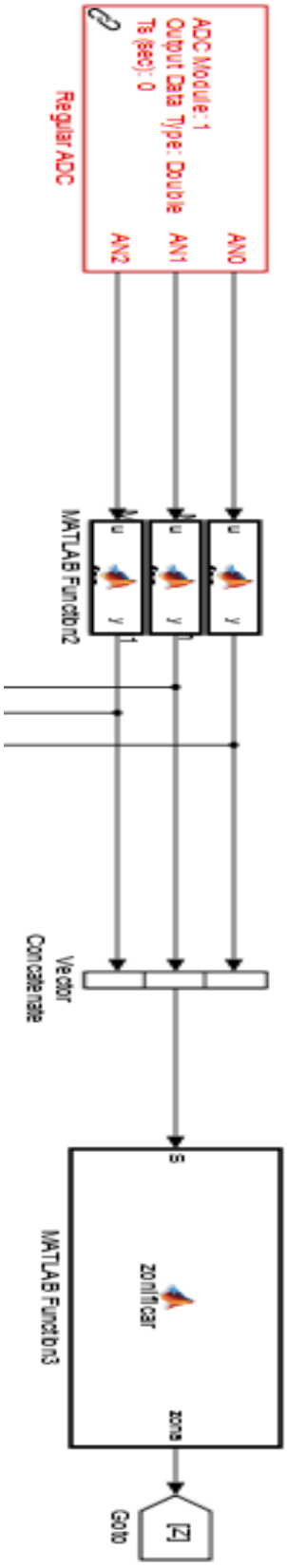
Anexo 1 Bloques que conforman el proceso Odométrico



Anexo 2 PID, PWM, salidas digitales para control de giro del motor.



Anexo 3 Lectura de los sensores y proceso de zonificación.



Anexo 4 Declaración de variables y detección de zonas.

Esta parte del código detalla cómo se realiza la zonificación, y los valores asignados a cada opción.

```
function zona = zonificar(S)

persistent cerrado gDerecha pasillo gIzquierda cruceT intDerecha
intIzquierda cruceTotal
persistent resultado
persistent dcerrado dgDerecha1 dgDerecha2 dpasillo1 dpasillo2
dgIzquierda1 dgIzquierda2 dcruceT1 dcruceT2
persistent dintDerecha1 dintDerecha2 dintIzquierda1 dintIzquierda2
dcruceTotal1 dcruceTotal2

if isempty(cerrado)
    cerrado = [5.5, 5.5, 5.5];
    gDerecha = [5.5, 5.5, 12];
    pasillo = [5.5, 12, 5.5];
    gIzquierda = [12, 5.5, 5.5];
    cruceT = [12, 5.5, 12];
    intDerecha = [5.5, 25, 12];
    intIzquierda = [12, 25, 5.5];
    cruceTotal = [12, 12, 12];

    resultado = 0;

    dcerrado = [4,4,4];
    dgDerecha1 = [4,1.5,1000];
    dgDerecha2 = [4,1.5,1];
    dpasillo1 = [4,1000,4];
    dpasillo2 = [4,1,4];
    dgIzquierda1 = [1000,1.5,5];
    dgIzquierda2 = [1,1.5,5];
    dcruceT1 = [1000,1.5,1000];
    dcruceT2 = [1,1.5,1];
    dintDerecha1 = [3,1000,1000];
    dintDerecha2 = [3,1,1];
    dintIzquierda1 = [1000,1000,3];
    dintIzquierda2 = [1,1,3];
    dcruceTotal1     = [1000,1000,1000];
    dcruceTotal2     = [1,1,1];
end

if (((S'<cerrado+dcerrado) & (S'>=cerrado-dcerrado))*[1;1;1] == 3)
    resultado = 1;
elseif (((S'<gDerecha+dgDerecha1) & (S'>=gDerecha-
dgDerecha2))*[1;1;1] == 3)
    resultado = 2;
elseif (((S'<pasillo+dpasillo1) & (S'>=pasillo-dpasillo2))*[1;1;1]
== 3)
    resultado = 3;
elseif (((S'<gIzquierda+dgIzquierda1) & (S'>=gIzquierda-
dgIzquierda2))*[1;1;1] == 3)
    resultado = 4;
```

```
elseif (((S'<cruceT+dcruceT1) & (S'>=cruceT-dcruceT2))*[1;1;1] == 3)
    resultado = 5;
elseif (((S'<intDerecha+dintDerecha1) & (S'>=intDerecha-
dintDerecha2))*[1;1;1] == 3)
    resultado = 6;
elseif (((S'<intIzquierda+dintIzquierda1) & (S'>=intIzquierda-
dintIzquierda2))*[1;1;1] == 3)
    resultado = 7;
elseif (((S'<cruceTotal+dcruceTotal1) & (S'>=cruceTotal-
dcruceTotal2))*[1;1;1] == 3)
    resultado = 8;
else
    resultado = 0;
end

zona = resultado;
```

Anexo 5 Red Neuronal

El siguiente código contiene la red neuronal y el algoritmo de mapeo.

```
function [ein_1,ein_2,ein0,ein1,ein2,ein3,e, enAvz, e2, enGiro,
selec, vl,led1,led2,mat] = fcn(iz,de, f, xin, yin, teta, z, ens, kd)

persistent tetaAnt giro avanzar xAnt yAnt pos nodo_sum aux2
persistent ini fin matNodos  nodo nodoant dist pos_act2 luz luz2
persistent matrizRed teta_abs coneccion teta_ant teta_actual pos_act
zona_act


persistent dim k2 k1 d s sum w I DU C P dec
persistent U V Us
persistent i j x y t aux sinpc

persistent nodo_act nodo_fin angulo_conec

if isempty (matrizRed)
  matNodos=zeros(5,5);     %Matriz de nodos ubicados en el mapa
  matrizRed=zeros(10,10);   %Matriz de distancias entre nodos
  coneccion=zeros(10,10);   %Matriz de coneciones de angulos entre
nodos
    ini = [5;5];     %Variable para posicion absoluta
    pos = [5;5];     %variable para posicion absoluta
    pos_act=[5;5];  %Variable de posicion enviada en el mapeo
    pos_act2=[5;5]; %Variable de posicion enviada con el robot
entregando la solucion
    tetaAnt = 0;     %Variable de angulo anteriorpara giro
    xAnt = 0;        %Variable de posicion anterior para traslacion
    yAnt = 0;        %Variable de posicion anterior para traslacion
    giro = 0;        %Variable que permite el giro del robot
    avanzar = 0;     %Variable que permite el avance del robot
    fin = 0;         %Variable que permite pasar de los estados de
mapeo a red neuronal y solución
```

```
    nodo=0;          %Variable que guarda el nodo actual
    nodoant=1;       %Variable que guarda el nodo anterior
    dist=0;          %Variable de distancia acumulada en los nodos
    teta_abs=0;      %Variable de angulo absoluto en el laberinto
para determinar posicion
    teta_ant=0;      %Variable de angulo anterior para determinar
matriz de conecciones
    aux=0;           %Variable auxiliar que permite ingresar a la red
neuronal
    teta_actual=0;   %Variable de teta guardado para corregir el
error en mapeo y solucion
    zona_act=0;      %Variable que guarda la zona actual del robot
    luz=0;
    luz2=0;
    matNodos(5,5)=1;
    nodo_sum=1;
    aux2=0;
    dec=0;
    sinpc = 0; % sin la computadorae
end



if isempty (i)
    i=0; j=0; x=0; y=0; t=0;
    dim=0;
    k2=2500;
    k1=100;
    d=0;
    s=1;
    sum=zeros(10,10);
    w=zeros(10,10,10,10);
    I=zeros(10,10);
    U=zeros(10,10);
    V=zeros(10,10);

    DU=zeros(10,10);
    C=zeros(10,10);
    P=zeros(10,10);
    Us=zeros(10,10);
end

if isempty (nodo_fin)
    nodo_act=1;
    nodo_fin=1;
    angulo_conec=0;
end

enAvz = 0;       %Habilita el PID de avance
enGiro = 0;      %Habilita el PID de giro
vl = 0;      %Coloca la velocidad en cero de las ruedas en el avance
selec = 0;       %Selecciona si gira o avanza el robot
led1=luz;        %Apago led
led2=luz2;
```

```
if (fin == 0)
        if (avanzar == 0 && giro == 0)
            if (z==1)
                luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta - pi;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;
                if(pos_act(1)==1 && pos_act(2)==1)
                    if (matNodos(pos_act(1),pos_act(2))==0)
                        nodo_sum=nodo_sum+1;
                        matNodos(pos_act(1),pos_act(2))=nodo_sum;
                    else
                        nodo=matNodos(pos_act(1),pos_act(2));
                    end
                    if (nodoant~=nodo)
                        matrizRed(nodoant,nodo)=dist;
                        coneccion(nodoant,nodo)=teta_ant;
                        coneccion(nodo,nodoant)=teta_abs+180;
                    end
                        teta_ant=teta_abs+180;
                        nodoant=nodo;
                        dist=0;
                end

                dist=dist+2;
                teta_abs=teta_abs+180;
            elseif (z==2)
```

```
luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;

                if(pos_act(1)==1 && pos_act(2)==1)
                    if (matNodos(pos_act(1),pos_act(2))==0)
                        nodo_sum=nodo_sum+1;
                        matNodos(pos_act(1),pos_act(2))=nodo_sum;
                    else
                        nodo=matNodos(pos_act(1),pos_act(2));
                    end
                    if (nodoant~=nodo)
                        matrizRed(nodoant,nodo)=dist;
                        coneccion(nodoant,nodo)=teta_ant;
                        coneccion(nodo,nodoant)=teta_abs+180;
                    end
                        teta_ant=teta_abs-90;
                        nodoant=nodo;
                        dist=0;
                end

                if(dist==0)
                    dist=1;
                else
                   dist=dist+1.5;
                end
                teta_abs=teta_abs-90;
            elseif (z==3)
                luz=0;
                avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;

if((pos_act(1)==1 && pos_act(2)==1))
                    if (matNodos(pos_act(1),pos_act(2))==0)
                        nodo_sum=nodo_sum+1;
                        matNodos(pos_act(1),pos_act(2))=nodo_sum;

                    else
                        nodo=matNodos(pos_act(1),pos_act(2));
                    end
                    if (nodoant~=nodo)
                        matrizRed(nodoant,nodo)=dist;
                        coneccion(nodoant,nodo)=teta_ant;
                        coneccion(nodo,nodoant)=teta_abs+180;
                    end`
                        teta_ant=teta_abs+0;
                        nodoant=nodo;
                        dist=0;
                end

                dist=dist+1;
                teta_abs=teta_abs+0;
            elseif (z==4)
                luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
```

```
pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;

                if(pos_act(1)==1 && pos_act(2)==1)
                    if(matNodos(pos_act(1),pos_act(2))== 0)
                        nodo_sum=nodo_sum + 1;
                        matNodos(pos_act(1),pos_act(2))=nodo_sum;
                    else
                        nodo=matNodos(pos_act(1),pos_act(2));
                    end
                    if (nodoant~=nodo)
                        matrizRed(nodoant,nodo)=dist;
                        coneccion(nodoant,nodo)=teta_ant;
                        coneccion(nodo,nodoant)=teta_abs+180;
                    end
                        teta_ant=teta_abs+90;
                        nodoant=nodo;
                        dist=0;
                end

                if(dist==0)
                    dist=1;
                else
                   dist=dist+1.5;
                end
                teta_abs=teta_abs+90;
            elseif (z==5)
                luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;
                aux2=matNodos(pos act(1),pos act(2));
```

```
if (aux2 == 0)
                    luz=1;
                    nodo_sum=nodo_sum+1;
                    matNodos(pos_act(1),pos_act(2))=nodo_sum;
                end
                    nodo=matNodos(pos_act(1),pos_act(2));
                if (nodoant~=nodo)
                    matrizRed(nodoant,nodo)=dist;
                    coneccion(nodoant,nodo)=teta_ant;
                    coneccion(nodo,nodoant)=teta_abs+180;
                end
                nodoant=nodo;
                dist=1.5;
                teta_abs=teta_abs-90;
                teta_ant=teta_abs;
            elseif (z==6)
                luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;
                aux2=matNodos(pos_act(1),pos_act(2));
                if (aux2 == 0)
                    luz=1;
                    nodo_sum=nodo_sum+1;
                    matNodos(pos_act(1),pos_act(2))=nodo_sum;
```

```
end
                    nodo=matNodos(pos_act(1),pos_act(2));
                if (nodoant~=nodo)
                    matrizRed(nodoant,nodo)=dist;
                    coneccion(nodoant,nodo)=teta_ant;
                    coneccion(nodo,nodoant)=teta_abs+180;
                end
                nodoant=nodo;
                dist=1.5;
                teta_abs=teta_abs-90;
                teta_ant=teta_abs;
            elseif (z==7)
                luz=0;
                avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;
                aux2=matNodos(pos_act(1),pos_act(2));
                if (aux2 == 0)
                    luz=1;
                    nodo_sum=nodo_sum+1;
                    matNodos(pos_act(1),pos_act(2))=nodo_sum;
                end
                    nodo=matNodos(pos_act(1),pos_act(2));
                if (nodoant~=nodo)
                    matrizRed(nodoant,nodo)=dist;
                    coneccion(nodoant,nodo)=teta_ant;
                    coneccion(nodo,nodoant)=teta_abs+180;
                end
                nodoant=nodo;
                dist=1;
                teta_abs=teta_abs+0;
                teta_ant=teta_abs;

elseif (z==8)
                luz=0;
                avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                pos_act=ini;
                teta_actual=teta_abs;
                zona_act=z;
                aux2=matNodos(pos_act(1),pos_act(2));
                if (aux2 == 0)
                    luz=1;
                    nodo_sum=nodo_sum+1;
                    matNodos(pos_act(1),pos_act(2))=nodo_sum;
                end
                    nodo=matNodos(pos_act(1),pos_act(2));
                if (nodoant~=nodo)
                    matrizRed(nodoant,nodo)=dist;
                    coneccion(nodoant,nodo)=teta_ant;
                    coneccion(nodo,nodoant)=teta_abs+180;
                end
                nodoant=nodo;
                dist=1.5;
                teta_abs=teta_abs-90;
                teta_ant=teta_abs;
            end
            if((avanzar == 1 || giro == 1))
```

```
pos=round(pos-[cos(teta_abs*pi/180) -sin(teta_abs*pi/180);
sin(teta_abs*pi/180) cos(teta_abs*pi/180)]*[1;0]);
                ini(1)=pos(1);
                ini(2)=pos(2);
                if (ini(1) > 5 || ini(2) > 5)
                    fin = 1;
                    enAvz = 0;
                    vl = 0;
                    enGiro = 0;
                    selec = 0;
                    ini = [5;5];
                    pos = [5;5];
                    avanzar = 0; xAnt = 0; yAnt = 0;
                    giro = 0; tetaAnt = 0;
                end
            end
        end

        if (avanzar == 1)
            enAvz = 1;
            enGiro = 0;
            vl = 120;  %/************** VELOCIDAD AL MAPEAR
            selec = 1;
            if((sqrt((xin-xAnt)^2+(yin-yAnt)^2) >= 19) || (f < 4.5))
                avanzar = 0;
                enAvz = 0;
                vl = 0;
                selec = 0;
            end
        end

        if (giro == 1)
            enGiro = 1;
            selec = 2;
            if(abs(tetaAnt - teta) < 0.1)
                avanzar = 1; xAnt = xin; yAnt = yin;
                giro = 0;
                enGiro = 0;
                selec = 0;
            end
        end
end

%% <<<<<<<<<<<<<<<<<<    PROGRAMA EJECUTA RED NEURONAL

if (fin == 1)
    if(aux==0)
        giro = 1; tetaAnt = teta - pi;
        teta_abs=0;
        aux=1;
        ini = [5;5];
    end
    if (giro == 1)
            enGiro = 1;
            selec = 2;
        if(abs(tetaAnt - teta) < 0.1)
            giro = 0; avanzar=0;
            enGiro = 0;
            enAvz = 0;
```

```
            selec = 0;
            aux=2;
            teta_abs=0;
            ini = [5;5];
            xAnt = xin; yAnt = yin;
        end
    end
    if(aux==2)
        dec=dec+1;
        luz=1;
        if(dec==5)
            aux=3;
        end
    end
    if(dec==5)
        dim=max(matNodos(:));
        d=matNodos(1,1);
        C=matrizRed;

        for(i=1:dim)
            for(j=1:dim)
                if(j~=i)
                   if(C(i,j)~=0)
                       C(j,i)=C(i,j);
                   end
                end
            end
        end

        for(i=1:dim)
            for(j=1:dim)
                if(j~=i)
                   if(C(i,j)==0)
                       P(i,j)=1;
                   end
                end
            end
        end

         for(y=1:dim)
             for(j=1:dim)
                 for(x=1:dim)
                    for(i=1:dim)
                              w(y,j,x,i)=-k2*(kronDel(x,y)-
kronDel(x,j)+kronDel(i,j)-kronDel(i,y));
                    end
                 end
             end
         end

         for(x=1:dim)
             for(i=1:dim)
                 I(x,i)=-(k1/2)*C(x,i)*(1-
kronDel(x,d)*kronDel(i,s))...
                 -(k2/2)*P(x,i)*(1-kronDel(x,d)*kronDel(i,s))...
                 +(k2/2)*kronDel(x,d)*kronDel(i,s);
             end
         end
```

```
        for(t=1:150)  %<<, numero de iteraciones de la red neuronal
            for(x=1:dim)
                for(i=1:dim)
                    sum(x,i)=0;
                    for(y=1:dim)
                        for(j=1:dim)
                            if(j~=y)
                                V(y,j)=fun_act(U(y,j),1);
                                sum(x,i)=sum(x,i)+w(y,j,x,i)*V(y,j);
                            end
                        end
                    end
                    DU(x,i)=-U(x,i)+sum(x,i)+I(x,i);
                    U(x,i)=U(x,i)+0.01*DU(x,i);
                end
            end
        end
        for(x=1:dim)
                for(i=1:dim)
                    if(V(x,i)>=0.5)
                        Us(x,i)=1;
                    else
                        Us(x,i)=0;
                    end
                end
        end
        dec=0;
        teta_abs=0;
        ini = [5;5];
        luz=0;
        fin=3;
    end
end
```

```
if(fin == 3)
    dec=dec+1;
    luz2=1;
    if(dec==500)
        fin=2;
        luz2=0;
    end
end
if (dec == 500)
        if (avanzar == 0 && giro == 0)
            if (z==1)
                pos_act2=ini;
                avanzar = 0; giro = 1; tetaAnt = teta - pi;
                teta_abs=teta_abs+180;
            elseif (z==2)
                pos_act2=ini;
                avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                teta_abs=teta_abs-90;
            elseif (z==3)
                pos_act2=ini;
                avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                teta_abs=teta_abs+0;
            elseif (z==4)
                pos_act2=ini;
                avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
                teta_abs=teta_abs+90;
```

```
            elseif (z==5)
                pos_act2=ini;
                nodo_act=matNodos(ini(1),ini(2));
                nodo_fin = buscar(Us(nodo_act,:));
                angulo_conec=coneccion(nodo_act,nodo_fin);
                if (round(cos(pi*(angulo_conec-teta_abs)/180))==1)
                    avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                    teta_abs=teta_abs+0;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==1)
                    avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
                    teta_abs=teta_abs+90;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==-1)
                    avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                    teta_abs=teta_abs-90;
                end
            elseif (z==6)
                pos_act2=ini;
                nodo_act=matNodos(ini(1),ini(2));
                nodo_fin = buscar(Us(nodo_act,:));
                angulo_conec=coneccion(nodo_act,nodo_fin);
                if (round(cos(pi*(angulo_conec-teta_abs)/180))==1)
                    avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                    teta_abs=teta_abs+0;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==1)
                    avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
                    teta_abs=teta_abs+90;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==-1)
                    avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                    teta_abs=teta_abs-90;
                end
            elseif (z==7)
                pos_act2=ini;
                nodo_act=matNodos(ini(1),ini(2));
                nodo_fin = buscar(Us(nodo_act,:));
                angulo_conec=coneccion(nodo_act,nodo_fin);
                if (round(cos(pi*(angulo_conec-teta_abs)/180))==1)
                    avanzar = 1; giro = 0; xAnt = xin; yAnt = yin;
tetaAnt = teta;
                    teta_abs=teta_abs+0;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==1)
                    avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
                    teta_abs=teta_abs+90;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==-1)
                    avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                    teta_abs=teta_abs-90;
                end
            elseif (z==8)
                pos_act2=ini;
                nodo_act=matNodos(ini(1),ini(2));
                nodo_fin = buscar(Us(nodo_act,:));
                angulo_conec=coneccion(nodo_act,nodo_fin);
                if (round(cos(pi*(angulo_conec-teta_abs)/180))==1)
```

```
avanzar = 1; giro = 0; xAnt = xin; yAnt = yin; tetaAnt = teta;
                    teta_abs=teta_abs+0;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==1)
                    avanzar = 0; giro = 1; tetaAnt = teta + pi/2;
                    teta_abs=teta_abs+90;
                elseif (round(sin(pi*(angulo_conec-
teta_abs)/180))==-1)
                    avanzar = 0; giro = 1; tetaAnt = teta - pi/2;
                    teta_abs=teta_abs-90;
                end
            end
            if((avanzar == 1 || giro == 1))
                if (pos_act2(1) == 1 && pos_act2(2) == 1)
                    fin = 3;
                    enAvz = 0;
                    vl = 0;
                    giro = 0;
                    enGiro = 0;
                    selec = 0; ini = [5;5];
                    avanzar = 0; xAnt = 0; yAnt = 0;
                    tetaAnt = 0;
                end
                ini=round(ini-[cos(teta_abs*pi/180) -
sin(teta_abs*pi/180);sin(teta_abs*pi/180)
cos(teta_abs*pi/180)]*[1;0]);
            end
        end
```

```
        if (avanzar == 1)
            enAvz = 1;
            enGiro = 0;
            vl = 150;   % >>>>>>>>>>>>>>>   VELOCIDAD CAMINO OPTIMO
            selec = 1;
            if((sqrt((xin-xAnt)^2+(yin-yAnt)^2) >= 19) || (f < 4.5))
                avanzar = 0;
                enAvz = 0;
                vl = 0;
                selec = 0;
            end
        end

        if (giro == 1)
            enGiro = 1;
            selec = 2;
            if(abs(tetaAnt - teta) < 0.15)
                giro = 0;
                enGiro = 0;
                selec = 0;
                avanzar = 1; xAnt = xin; yAnt = yin;
            end
        end
end


e = tetaAnt + (de<8)*kd*(5.5-de) +(iz<8)*kd*(iz-5.5)- teta;
e2 = tetaAnt - teta;

ein_1=dim;
ein_2=d;


ein0=zona_act;
ein1=pos_act(1);
ein2=pos_act(2);
ein3=teta_actual;

mat=matrizRed;
```