ISBN: 978-99961-39-01-7 (Impreso)
ISBN: 978-99961-39-02-4 (E-Book)

INFORME FINAL DE INVESTIGACIÓN

DISEÑO DE PROTOTIPO DE VEHÍCULO AUTÓNOMO UTILIZANDO REDES NEURONALES

Aplicación para el Transporte de Materiales

DOCENTE INVESTIGADOR PRINCIPAL:
ING. MORRIS WILLIAM DÍAZ SARAVIA

DOCENTE CO-INVESTIGADOR:
ING. JOSÉ MANUEL TREJO PERAZA

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ITCA-FEPADE SEDE CENTRAL

ENERO 2019

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN
ITCA EDITORES

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

ISBN: 978-99961-39-01-7 (Impreso)
ISBN: 978-99961-39-02-4 (E-Book)

INFORME FINAL DE INVESTIGACIÓN

DISEÑO DE PROTOTIPO DE VEHÍCULO AUTÓNOMO UTILIZANDO REDES NEURONALES

Aplicación para el Transporte de Materiales

DOCENTE INVESTIGADOR PRINCIPAL:
ING. MORRIS WILLIAM DÍAZ SARAVIA

DOCENTE CO-INVESTIGADOR:
ING. JOSÉ MANUEL TREJO PERAZA

ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ITCA-FEPADE SEDE CENTRAL

ENERO 2019

GOBIERNO DE EL SALVADOR
MINISTERIO DE EDUCACIÓN
ITCA EDITORES

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA-FEPADE
DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL
SANTA TECLA, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

629.229 3
D542d Díaz Saravia, Morris William, 1968 -
Diseño de prototipo de vehículo autónomo utilizando
SV redes neuronales : aplicación para el transporte de materiales [recurso electrónico] / Morris William Díaz Saravia, José Manuel Trejo Peraza, coaut. -- 1ª ed. – Datos electrónicos (1 archivo : 19000 kb). – Santa Tecla, La Libertad, El Salv. : ITCA Editores, 2019.
1 recurso en línea : col.
Forma de acceso : World Wide Web. URL: https://www.itca.edu.sv/produccion-academica/
Titulo tomado de la pantalla de presentación
Datos publicados también en forma impresa
ISBN : 978-99961-39-01-7 (Impreso)
ISBN : 978-99961-39-02-4 (E-Book)

1. Carros eléctricos para la industria - Diseño. 2. Redes neurales (computadores). 3. Manejo de materiales - Automatización. 4. Sensores remotos. I. Trejo Peraza, José Manuel, coaut. II. Título.

Autor
Ing. Morris William Díaz Saravia

Co Autor
Ing. José Manuel Trejo Peraza

Tiraje: 13 ejemplares
Año 2019

Este documento técnico es una publicación de la Escuela Especializada en Ingeniería ITCA–FEPADE; tiene el propósito de difundir la Ciencia, la Tecnología y la Innovación CTI, entre la comunidad académica y el sector empresarial, como un aporte al desarrollo del país. Para referirse al contenido debe citar el nombre del autor y el título del documento. El contenido de este Informe es responsabilidad de los autores.

Escuela Especializada en Ingeniería ITCA-FEPADE
Km 11.5 carretera a Santa Tecla, La Libertad, El Salvador, Centro América
Sitio Web: www.itca.edu.sv
TEL: (503)2132-7423
FAX: (503)2132-7599

CONTENIDO

1. INTRODUCCIÓN

Este proyecto consiste en el diseño experimental e implementación de un vehículo autónomo que transporte mercancías o materias primas en el interior de una industria, oficinas o comercio. Mediante un conjunto de sensores, será capaz de detectar los objetos en su entorno y basados en ellos, la red neuronal decide la mejor ruta a seguir para alcanzar su destino.

Según las especificaciones planteadas en el diseño, el vehículo será capaz de transportar un peso de hasta 30 Kg; para su transporte, la carga debe estar debidamente embalada y sujeta al vehículo, las tareas de carga y descarga serán realizadas por un operador humano. Debido a los componentes electrónicos a bordo del vehículo, es importante evitar cualquier derramamiento de líquidos que pueda contener la carga, además se recomienda su operación en ambientes secos y una superficie plana.

El nivel de autonomía del vehículo, se refiere a la operación del vehículo: transportar la carga de un punto a otro sin acción humana directa durante su desplazamiento. El operador debe indicar el punto de partida y el punto de llegada y cuando se dé la orden de transportar, el vehículo se moverá autónomamente hacia el punto destino. Entre los campos de aplicación, se puede considerar en una compañía de logística, en ambientes industriales, para el transporte de materias primas, transporte de herramientas, transporte de componentes electrónicos, telas, alimentos enlatados, etc.

El objetivo de este proyecto es incursionar en tecnologías innovadoras de inteligencia artificial y redes neuronales, controladas por sensores ultrasónicos, infrarrojos y LIDAR, aplicado a un diseño experimental de prototipo de vehículo guiado de forma autónoma AGV.

2. PLANTEAMIENTO DEL PROBLEMA

2.1. DEFINICIÓN DEL PROBLEMA

Los vehículos guiados autónomos para ambientes industriales existen, pero se han desarrollado para trabajar en rutas preestablecidas, y con baja capacidad de tomar decisiones ante obstáculos o modificaciones en la ruta, poseen poca flexibilidad y poca o nula capacidad de resolver situaciones no previstas, produciendo su bloqueo o la necesidad de intervención humana. Se pueden agregar nuevas rutas, pero implica altos costos en su programación, además en un entorno con personas hay posibilidad de producir colisiones y accidentes. Es necesario elaborar un sistema que corrija los inconvenientes antes mencionados.

2.2. ANTECEDENTES / ESTADO DE LA TÉCNICA

Las redes neuronales son una tecnología bioinspirada, que, a diferencia de las computadoras actuales, se fundamentan en simular el cerebro humano con neuronas y sus respectivas conexiones, aunque se plantearon sus bases teóricas en los 60’s, en ese momento, la capacidad de cómputo reducía en gran medida el desarrollo de una lógica compleja. En la actualidad, las redes neuronales han superado en gran medida sus expectativas, sistemas basados en ellas son capaces de aprender, entrenarse y hacer las operaciones mejor que un operador humano, han creado nuevas áreas como el aprendizaje profundo, lógica difusa, algoritmos genéticos llegando a crear una inteligencia artificial “real”, capaz de tomar decisiones complejas en tiempo real.

Además, la técnica SLAM (Simulateneous Localization And Mapping) se encuentra en su máximo desarrollo, siendo una técnica robótica que, mediante sensores y odometría de las ruedas, es capaz de situar el vehículo en un mapa del entorno creado por el mismo vehículo.

Actualmente se encuentran en prueba, por diversas empresas como Uber, Google y Tesla, una serie de vehículos autónomos para el transporte de personas en los centros urbanos, basándose en sensores, visión artificial, GPS, lógica difusa y redes neuronales. En la actualidad, las empresas dedicadas a los vehículos guiados autónomamente son pocas, en España se encuentra ASTI Mobile Robotics (http://asti.es), especializada en soluciones de logística a la medida, en Alemana se encuentra Kuka Industries (www.kuka.com/es-es), ambos proveen vehículos guiados autónomamente, robustos y ampliamente utilizados, no cuentan con una inteligencia artificial autónoma, se basan en un equipo central que coordina las actividades de todos los vehículos autónomos en circulación, y aunque posee elaboración de mapas SLAM y determinación de la ruta de forma dinámica, está limitado a las instrucciones dadas por el ordenador central.

2.3. JUSTIFICACIÓN

El desarrollo de este vehículo permitirá la automatización del transporte de materias primas, herramientas, equipos, piezas y cualquier objeto que sea necesario transportarse en un entorno industrial o de almacén, reduciendo costos en cuanto al aprendizaje de nuevas rutas en nuevos entornos.

Con este proyecto se reducirán los tiempos de transporte de materiales. El propósito de la investigación, además del vehículo autónomo, es implementar una red neuronal capaz aprender para llegar de un punto a otro de la forma más eficiente, en ambientes dinámicos. Se pueden beneficiar, las industrias aeronáuticas, fabricación y reutilización de baterías, y cualquiera otra industria que maneje grandes cantidades de materia primas/productos terminados en sus arcas industriales o comercios.

Actualmente este tipo de aplicación se encuentra a un alto costo, a limitados sectores, con este trabajo de investigación se busca hacer accesible dicha tecnología a un menor costo, con ambientes flexibles para su aplicación, además, al ser automatizado se implementará mecanismos de seguridad para evitar daños, tanto a los equipos o materiales transportados, como a otros vehículos y personas circundantes.

Se justifica la necesidad de esta investigación al tomar en cuentas los siguientes factores:

Educativo: los alumnos de las carreras de la escuela de Ingeniería Eléctrica y Electrónica, así como ramas afines, podrán aplicar sus conocimientos practicando en el proyecto en las áreas de mecánica, robótica y programación. Cuando esté finalizado se podrán analizar y mejorar los sistemas.

Social: puede ser aplicado en tareas repetitivas o realizar tareas en zonas de difícil acceso para el ser humano, evitando riesgos innecesarios, como puede ser su uso en situaciones de catástrofes, para búsqueda de personas, etc.

Empresarial: los vehículos autónomos para ambientes industriales, es un rubro poco explotado en el mundo, alguna empresa nacional puede desarrollar esta tecnología y exportarla al mundo, generando empleos y exportación a nivel local, regional y mundial.

3. OBJETIVOS

3.1. OBJETIVO GENERAL:

Desarrollar un vehículo autónomo guiado para el transporte de mercancías y materias primas en un ambiente industrial, siendo capaz de tomar decisiones en tiempo real y evitando obstáculos estáticos y en movimiento, basándose en redes neuronales, otorgándole la capacidad de aprendizaje profundo que lo hará adaptable a múltiples ambientes.

3.2. OBJETIVOS ESPECÍFICOS:

- Investigar los principios tecnológicos que conforma un vehículo autónomo y los tipos de algoritmos de una red neuronal.
- Diseñar los diferentes sistemas mecánicos y electrónicos que componen el AGV (del inglés Automated Guided Vehicle: Vehículo Autónomo Guiado).
- Construir el chasis que permita realizar las tareas solicitadas al AGV.
- Ajustar y configurar los diferentes elementos que conforman el AGV.
- Desarrollar algoritmos para la construcción de mapas de localización de ambientes industriales y comerciales
- Realizar pruebas de funcionamiento mecánico
- Realizar pruebas del sistema de la red neuronal, aprendizaje profundo.

4. HIPÓTESIS

¿Aumentará la productividad y disminuirá los tiempos de transporte al ser utilizado un sistema AGV para el transporte materias primas, herramientas, equipos y piezas al interior una planta industrial?

Para pruebas de campo en ambiente real, se harán en Industrias Tejano, Armenia, Sonsonate, después de funcionar haciendo las tareas para las cuales se diseñó el vehículo, se obtendrá las recomendaciones y validación del proyecto, utilizando un instrumento de evaluación.

5. MARCO TEÓRICO

5.1. BREVE HISTORIA

En 1925, el inventor Francis Houdina muestra al mundo un automóvil controlado por radio, el cual se maneja por las calles de Manhattan sin nadie al volante. Según el New York Times, el vehículo controlado por radio puede encender su motor, cambiar de marcha y hacer sonar el pito “como si una mano fantasma estuviese al volante”.

Como anécdota, hay que decir que el nombre de Houdina sonaba lo suficiente parecido al famoso ilusionista Harry Houdini, tanto que mucha gente pensó que todo se trataba de un truco del gran mago.

En 1969, John McCarthy, conocido como uno de los padres de la inteligencia artificial, describe algo similar al vehículo autónomo moderno en un ensayo titulado "Computer Controlled Cars". McCarthy se refiere a un "chófer automático", capaz de navegar por una vía pública a través de una "entrada de cámara de televisión que usa la misma entrada visual disponible para un controlador humano".

Él escribe que los usuarios deberían poder ingresar un destino usando un teclado, lo que induciría al automóvil a llevarlos inmediatamente hacia allí. Comandos adicionales permitirían a los usuarios cambiar el destino, detenerse en una sala de descanso o restaurante, reducir la velocidad o acelerar, en caso de una emergencia. No se construye ningún vehículo de este tipo, pero el ensayo de McCarthy se convierte en una especie de "misión" en la que otros investigadores trabajarán.

A principios de 1990, el investigador de Carnegie Mellon, Dean Pomerleau, escribe una tesis doctoral que describe cómo las redes neuronales podrían permitir que un vehículo autónomo captara imágenes en bruto de la carretera, enviando órdenes a los controles de dirección en tiempo real. Pomerleau no es el único investigador que trabaja en carros autodirigidos, pero su uso de redes neuronales demuestra ser mucho más eficiente que los intentos alternativos de dividir manualmente las imágenes en categorías "viales" y "no viales".

En 1995, Pomerleau y su colega investigador, Todd Jochem, llevan su sistema de auto-conducción Navlab a la carretera. Su minivan autónoma (ellos tienen que controlar la velocidad y el frenado) recorre 2,797 millas de costa a costa, desde Pittsburgh, Pennsylvania hasta San Diego, California, en un viaje que la pareja llama "No Hands Across America", que traducido sería "Sin Manos A Través de América".

En 2002, DARPA anuncia su Gran Desafío, ofreciendo a los investigadores de las principales instituciones de investigación un premio de $1 millón de dólares si pueden construir un vehículo autónomo capaz de conducirse por 142 millas a través del desierto de Mojave.

Cuando el desafío comienza, en 2004, ninguno de los 15 competidores puede completar el curso. El "ganador" consigue recorrer sólo ocho millas en varias horas, todo antes de incendiarse. Es un duro golpe para todos aquellos que buscan construir verdaderos carros sin conductor.

Si bien los vehículos autónomos aún parecen ser un sueño muy futuro en los primeros años del 2000, comienzan a surgir sistemas de estacionamiento automático, los cuales demuestran que los sensores y las tecnologías de carreteras autónomas se están preparando para escenarios del mundo real.

El vehículo híbrido Prius, del fabricante japonés Toyota, ofrece asistencia automática de estacionamiento en paralelo desde 2003, mientras que Lexus pronto agrega un sistema similar para su Lexus LS sedán. Ford incorpora Active Park Assist en 2009 y BMW le sigue un año más tarde con su propio asistente de estacionamiento en paralelo.

A partir de 2009, Google comienza en secreto a desarrollar su proyecto de vehículo autónomo (ahora llamado Waymo). El proyecto está liderado inicialmente por Sebastian Thrun, ex director del Laboratorio de Inteligencia Artificial de Stanford y co-inventor de Google Street View.

Dentro de unos años, Google anuncia que sus carros autónomos se han conducido colectivamente por 300,000 millas bajo control de computadoras sin que haya ocurrido ni un solo accidente. En 2014, se presenta un prototipo de un automóvil sin conductor y sin volante, acelerador o pedal de freno, lo que lo hace 100 por ciento autónomo. A fines del año pasado, más de 2 millones de millas habían sido ya recorridas por los carros autónomos de Google.

Para 2013, las principales compañías automotrices, incluidas General Motors, Ford, Mercedes Benz, BMW y otras, están trabajando en sus propias tecnologías autónomas. Nissan se la juega incluso con una fecha, al anunciar que lanzará varios autos sin conductor para el año 2020.

Otros autos, como el Mercedes Clase S 2014, agregan características semiautónomas como la dirección automática, la capacidad de permanecer dentro de carriles, evitar accidentes y más. Los gustos de Tesla y Uber también comienzan a explorar activamente la tecnología de conducción autónoma, mientras que se rumorea que Apple lo estaría también haciendo.

Triste, pero inevitablemente, se produce la primera víctima mortal en la historia de la conducción autónoma. El incidente ocurre en Florida, cuando un Tesla Model S está en modo de piloto automático. El ocupante humano del Tesla muere cuando el automóvil choca contra un remolque de 18 ruedas, no alcanzando a frenar a tiempo después de que el remolque pasó frente a él.

La muerte provoca un renovado debate sobre los autos sin conductor, así como de algunos de los problemas técnicos y éticos que rodean este camino. Es un retroceso, pero que subraya el hecho de que, les guste o no, los vehículos autónomos están aquí, ya de verdad. Y no parece que vaya a haber una vuelta atrás.

El vehículo autónomo guiado cuenta con los siguientes subsistemas:

1. Sistema mecánico: la parte mecánica de un vehículo determina la forma del mismo y las tareas de transporte que puede desarrollar.
2. Sistema de navegación: es el sistema cuya función es posicionar al vehículo en las rutas determinadas para llegar a cabo su función.
3. Sistema de seguridad: es el sistema encargado de que el vehículo realice su tarea de manera segura, con respecto al resto de elementos que le rodean.
4. Sistema de baterías y recarga: sistema que garantiza que los vehículos cuenten con autonomía para moverse.
5. Sistema de comunicación: sistema que permite la comunicación entre los elementos en planta, tanto con software como entre máquinas. Recoge los datos y los gestiona para la generación de órdenes y el control de tráfico

5.2. TECNOLOGÍAS

a. ROBÓTICA INDUSTRIAL

Existen ciertas dificultades a la hora de establecer una definición formal de lo que es un robot industrial. La primera de ellas surge de la diferencia conceptual entre el mercado japonés y el euro-americano de lo que es un robot y lo que es un manipulador. Así, mientras que para los japoneses un robot industrial es cualquier dispositivo mecánico dotado de articulaciones móviles destinado a la manipulación, el mercado occidental es más restrictivo, exigiendo una mayor complejidad, sobre todo en lo relativo al control. En segundo lugar, y centrándose ya en el concepto occidental, aunque existe una idea común acerca de lo que es un robot industrial, no es fácil ponerse de acuerdo a la hora de establecer una definición formal. Además, la evolución de la robótica ha ido obligando a diferentes actualizaciones de su definición.

La definición más comúnmente aceptada posiblemente sea la de la Asociación de Industrias Robóticas (RIA), según la cual:

Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas, o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas.

Esta definición, ligeramente modificada, ha sido adoptada por la Organización Internacional de Estándares (ISO) que define al robot industrial como:

Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

Se incluye en esta definición la necesidad de que el robot tenga varios grados de libertad. Una definición más completa es la establecida por la Asociación Francesa de Normalización (AFNOR), que define primero el manipulador y, basándose en dicha definición, el robot:

Manipulador: mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivo lógico.

Robot: manipulador automático servo-controlado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material.

Por último, la Federación Internacional de Robótica (IFR) distingue entre robot industrial de manipulación y otros robots:

Por robot industrial de manipulación se entiende una máquina de manipulación automática, reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento.

En esta definición se debe entender que la reprogramación y la multifunción se consiguen sin modificaciones físicas del robot.

Común en todas las definiciones anteriores es la aceptación del robot industrial como un brazo mecánico con capacidad de manipulación y que incorpora un control más o menos complejo. Un sistema robotizado, en cambio, es un concepto más amplio. Engloba todos aquellos dispositivos que realizan tareas de forma automática en sustitución de un ser humano y que pueden incorporar uno o varios robots, siendo esto último lo más frecuente.

Clasificación del robot industrial

La maquinaria para la automatización rígida dio paso al robot con el desarrollo de controladores rápidos, basados en el microprocesador, así como un empleo de servos en bucle cerrado, que permiten establecer con exactitud la posición real de los elementos del robot y establecer el error con la posición deseada. Esta evolución ha dado origen a una serie de tipos de robots, que se citan a continuación:

- Manipuladores
- Robots de repetición y aprendizaje
- Robots con control por computador
- Robots inteligentes
- Micro-robots

b. ROBÓTICA MÓVIL

Tradicionalmente las aplicaciones de la robótica estaban centradas en los sectores manufactureros más desarrollados para la producción masiva: industria del automóvil, transformaciones metálicas, industria química, etc. aunque en la última década el peso de la industria manufacturera ha disminuido.

A principios de los años sesenta se introducen en la industria, de modo significativo, los robots manipuladores como un elemento más del proceso productivo. Esta proliferación, motivada por la amplia gama de posibilidades que ofrecía, suscitó el interés de los investigadores para lograr manipuladores más rápidos, precisos y fáciles de programar. La consecuencia directa de este avance originó un nuevo paso en la automatización industrial, que flexibilizó la producción con el nacimiento de la noción de célula de fabricación robotizada.

Los trabajos desarrollados por los robots manipuladores consistían frecuentemente en tareas repetitivas, como la alimentación de las distintas máquinas componentes de la célula de fabricación robotizada. Ello exigía ubicarlas en el interior de un área accesible para el manipulador, caracterizada por la máxima extensión de sus articulaciones, lo cual podría resultar imposible a medida que la célula sufría progresivas ampliaciones. Una solución a este problema se logra al desarrollar un vehículo móvil sobre rieles para proporcionar un transporte eficaz de los materiales entre las distintas zonas de la cadena de producción. De esta forma, aparecen en los años ochenta los primeros vehículos guiados automáticamente (AGV's). Una mejora con respecto a su concepción inicial estriba en la sustitución de los rieles como referencia de guiado en la navegación por cables enterrados, reduciéndose, con ello, los costes de instalación.

La posibilidad de estructurar el entorno industrial permite la navegación de vehículos con una capacidad sensorial y de razonamiento mínimas. De este modo, la tarea se ordena en una secuencia de acciones en la que a su término el vehículo supone que ha alcanzado el objetivo para el que está programado. Ante cualquier cambio inesperado en el área de trabajo que afecte el desarrollo normal de la navegación, el sistema de navegación del vehículo se encontrará imposibilitado para ejecutar acciones alternativas que le permitan reanudar su labor. Sin embargo, por sus potenciales aplicaciones fuera del ámbito industrial, donde resulta costoso o imposible estructurar el entorno, se les dotó, en la búsqueda de un vehículo de propósito general apto para desenvolverse en cualquier clase de ambiente, de un mayor grado de inteligencia y percepción. Así en los años noventa surgen el robot móvil. Una definición correcta de robot móvil plantea la capacidad de movimiento sobre entornos no estructurados, de los que se posee un conocimiento incierto, mediante la interpretación de la información suministrada a través de sus sensores y del estado actual del vehículo.

Esta evolución mecánica, sensorial y racional de los robots móviles no fue así de estricta ya que, sin tener una finalidad específica, a lo largo de la historia existieron algunos desarrollos que fueron fuente de inspiración para la construcción de los robots móviles actuales. Algunos de ellos son el primer robot humanoide de Leonardo Da Vinci a mediados de los noventa, del siglo XV, la Máquina Speculatrix de W. Walter Grey en los años cincuenta y Shakey del Stanford Research Institute en los setenta, ambos del siglo XX.

c. NAVEGACIÓN BASADA EN SLAM

El problema de la localización y mapeado simultáneos consiste en descubrir si es posible para un robot móvil navegar a través de un entorno desconocido y construir, de manera incremental, un mapa consistente del mismo mientras que determina, al mismo tiempo, su posición dentro de éste

Una solución a este problema se ha considerado "el Santo Grial" por la comunidad de especialistas en robótica móvil, ya que dotaría a los robots con las herramientas necesarias para ser completamente autónomos.

La "solución" del problema del SLAM ha sido uno de los sucesos más notables para la comunidad de los últimos años. SLAM ha sido formulado y resuelto como problema teórico en distintas formas. SLAM ha sido, también, implementado en diversos campos de la robótica móvil, como los robots de interiores, en exteriores, robots subacuáticos y robots voladores. A un nivel teórico y conceptual, SLAM puede considerarse un problema resuelto. Sin embargo, aún quedan varios problemas en la aplicación práctica de soluciones más generales de SLAM, y particularmente en la utilización y construcción de mapas perceptualmente ricos como parte de un algoritmo de SLAM.

5.3. REDES NEURONALES

La mayoría de autores coinciden en que el primer modelo neuronal artificial, fue planteado por Alan Turing en 1936, siendo la primera implementación de una red fue lanzada por Warren McCulloch y Walter Pitts en 1943. Este sistema consistía en una red neuronal basada en circuitos eléctricos. En de 1951, Marvin Minsky y Dean Edmonds implementar la primera máquina de redes neuronales, compuesta por 300 tubos al vacío y un piloto automático de un bombardero B-24 (Herrera 1998). Después de su Red Neuronal, Minsky escribió su tesis doctoral acerca de ésta. En ella describía "cerebros mucho mayores", exponiendo que, si se realizaba este proyecto a gran escala, con miles o millones de neuronas más y con diferentes sensores y tipos de retroalimentación la máquina podría ser capaz de razonar. Minsky sabía que la implementación de la Red Neuronal así descrita, era imposible con la tecnología de su época. En 1957, Frank Rosenblatt desarrolló un modelo de red neuronal que contenía una sola neurona y tenía un aprendizaje basado en la regla de Hebb, siendo considerada ésta la red más antigua y que todavía sigue siendo utilizada en su forma evolucionada que involucra capas múltiples. Entre sus características básicas se pueden mencionar que era capaz de generalizar, es decir reconocer o identificar patrones que no se le hubieran enseñado, utilizar "n" número de entradas y una sola salida. Esta red tenía ciertas limitaciones, no podía clasificar patrones que no tuvieran una separación lineal, si se pudieran ubicar los patrones a reconocer en un plano cartesiano, éstos podrían ser separados por una línea recta, por lo que el plano quedaría dividido en dos partes que representarían las dos clasificaciones que podría realizar el Perceptrón.

Conceptos básicos.

¿Qué es una red neuronal? [1]

Existen diferentes definiciones del término red neuronal:

- Una forma de computación inspirada en modelos biológicos (bioinspirada).
- Elemento computacional de procesamiento, como unidades similares a las del cerebro humano, que tienen la capacidad de procesar información y aprender de ella.
- Un sistema de computación que consiste en un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

La mayoría de tipos de redes neuronales han sido creadas con el objetivo de emular en cierto grado la inteligencia artificial, lo que provoca que la mayoría de los diseños generados, sea orientada a algoritmos que puedan ser fácilmente implementados en cualquier lenguaje de programación, aunque también pueden ser puestos en funcionamiento en forma de hardware.

Una red neuronal artificial (ANN) es un esquema de computación distribuida inspirada en la estructura del sistema nervioso de los seres humanos. La arquitectura de una red neuronal es formada conectando múltiples procesadores elementales, siendo éste un sistema adaptivo que posee un algoritmo para ajustar sus pesos (parámetros libres) para alcanzar los requerimientos de desempeño del problema basado en muestras representativas.

Por lo tanto, podemos señalar que una ANN es un sistema de computación distribuida caracterizada por:

- Un conjunto de unidades elementales, cada una de las cuales posee bajas capacidades de procesamiento.
- Una densa estructura interconectada usando enlaces ponderados.
- Parámetros libres que deben ser ajustados para satisfacer los requerimientos de desempeño.
- Un alto grado de paralelismo.

¿Qué partes componen una neurona?

Como puede verse en la Figura 1, una neurona artificial está compuesta de tres partes: Entradas, núcleo y salidas. Las entradas reciben los datos o parámetros que le permiten decidir a la neurona si estará activa o no, normalmente se representan como $x_1, x_2, ..., x_n$. Entre la entrada y el núcleo aparecen los pesos ($w_1, w_2, ..., w_n$), que representan la memoria de la red. Las salidas devuelven la respuesta de la neurona, es decir si esta activa o no, representadas comúnmente como $y_1, y_2, ..., y_n$.

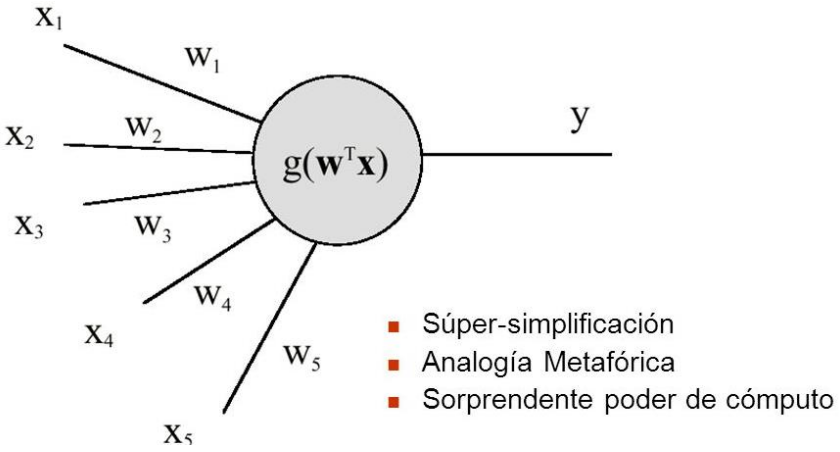


Fig. 1: Neurona artificial simple

En el núcleo se realizan todas las operaciones necesarias para determinar la salida de la neurona; el proceso que se realiza en el núcleo varía dependiendo de la red neuronal que se esté trabajando. Es importante mencionar que la salida del núcleo es pasada por una función de transferencia o activación.

Ventajas de las redes neuronales

- Para el procesamiento de la información, se hace de manera local, al estar compuesto por unidades individuales de procesamiento, cada neurona realiza su activación, dependiendo de sus entradas y pesos, y todas las neuronas de una capa trabajan en forma paralela y proporcionan una respuesta al mismo tiempo.
- Se ajustan los pesos ($w_1, w_2, ..., w_n$) basándose en la experiencia, por lo tanto, se tiene que enseñar a la red lo que necesita saber antes de ponerla en funcionamiento (entrenamiento).
- Los pesos definen si una función es de activación o inhibición, son parte importante en la determinación de si una neurona está activa o no.
- Las neuronas son tolerantes a fallos, si parte de la red no trabaja, sólo dejará de funcionar la parte para la que dicha neurona sea significativa; el resto tendrá su comportamiento normal.
- Las neuronas pueden reconocer patrones que no han sido aprendidos, sólo deben tener cierto parecido con el conocimiento previo que tenga la red. En otras palabras: si la entrada presenta alguna alteración la red podrá identificarla siempre y cuando se mantenga cierto grado de similitud entre lo aprendido y lo mostrado en la entrada a la red.
- Operación en tiempo real, los algoritmos de las redes neuronales son eficientes una vez que la red ha sido entrenada, por lo que su respuesta es casi inmediata.

Desventajas de las redes neuronales

- En grandes tareas, hay un alto grado de complejidad en el aprendizaje, cuantas más cosas se necesite que aprenda una red, más complicado será enseñarle.

- Elevado tiempo de aprendizaje. Hay dos factores que afectan el tiempo: 1. Si se incrementa la cantidad de patrones a identificar o clasificar y, 2. Si se requiere mayor flexibilidad o capacidad de adaptación de la red neuronal para reconocer patrones que sean sumamente parecidos, se deberá invertir más tiempo en lograr que la red converja a valores de pesos que representen lo que se quiera enseñar.
- No se interpreta lo que se ha aprendido, la red por sí sola proporciona una salida, un número, que no puede ser interpretado por ella misma, sino que se requiere de la intervención del programador y de la aplicación en sí para encontrarle un significado a la salida proporcionada.
- Elevada cantidad de datos para el entrenamiento, cuanto más flexible se requiera que sea la red neuronal, más información tendrá que enseñársele para que realice de forma adecuada la identificación.

Red con aprendizaje de Hebb (o Regla de aprendizaje de Hebb)

Se basa en el modelo de la figura 1, y está definido por los siguientes pasos:

a. Definir los patrones a enseñar a la red

b. Definir la cantidad de entradas

c. Definir las salidas que se desea en la red

d. Los pesos comienzan con un valor de cero

e. Poner en las entradas de la red un patrón a ser aprendido

f. Calcular la salida de la neurona usando la fórmula

$$y = \sum_{i=0}^{n} x_i \times w_i$$

Las variables están definidas así:

y es la salida de la neurona,

x_i son las entradas,

w_i son los pesos y

n la cantidad de entradas que tiene la neurona.

g. Se pasa la salida por una función de transferencia escalón unitaria unipolar o bipolar, se iguala ***y*** a la salida deseada para el patrón de entrada se modifican los pesos basándose en la formula

$$w_{new} = w_{i\ old} + x_i \cdot y$$

h. Se verifica si han pasado todos los patrones que se requiere la red aprenda. Si la respuesta es no, se retrocede al literal e, y si la respuesta es sí se termina el entrenamiento.

5.4. VEHÍCULO AUTÓNOMAMENTE GUIADO (AVG)

Los AGV's son vehículo guiados autónomamente o automáticamente, fueron inventados en 1953 por Berret Electronics, es un "robot móvil controlado por computador utilizado para el transporte de materiales".

Un sistema AVG, es un conjunto de estos vehículos con capacidad de programación de destino, selección de trayectoria y posicionamiento, permitiendo el manejo de materiales que se caracteriza por ser altamente flexible, inteligente y versátil, transportando materiales desde diversos puntos de carga a lo largo de las

instalaciones de una planta o entrono.

Diseño de un sistema AVG

El diseño de una AVG es complejo, hay muchas variables a tomar en cuenta que impacta en el funcionamiento del mismo y qué son difíciles de predecir. Las decisiones que se tomen están ligadas a otras variables y estas decisiones tendrán implicaciones importantes en el diseño. Los principales aspectos a tomar en cuenta para el diseño son:

- Las trayectorias.
- Manejo del tráfico de los AGV.
- El número de estaciones de carga y descarga de materiales.
- Los requerimientos de los vehículos.
- Enrutamiento de los vehículos.
- Planificación de uso de los vehículos.
- Posiciones ideales para los vehículos.
- Manejo de baterías.
- Manejo de fallas.

Las trayectorias definen por donde se debe mover el AGV, el manejo del trafico incluye todas las estrategias de seguridad que se deben tener para que los vehículos no colisionen entre sí o con algún otro objeto u o persona, los requerimientos de los vehículos determinan que características deben tener los AGV para poder encajar a la perfección con el sistema (tamaño, forma, sensores, sistema de guiado, programación), la planificación se refiere a que debe hacer cada robot en que momento, las posiciones ideales se refieren a los sitios más seguros de tránsito, el manejo de las baterías está ligado a que baterías se deben tener, cuál va a ser su forma de carga, en que parte de la planta se van a cargar y por último el manejo de fallas se refiere a como se va a manejar las fallas en el sistema.

Componentes de un sistema AGV

- **El vehículo (AGV):** Encargado de cumplir las órdenes realizando movimientos de materiales dentro de un sistema.
- **La trayectoria guía:** Dirige el vehículo para que se mueva a lo largo de la trayectoria establecida.
- **La unidad de control:** Supervisa y dirige los procedimientos y actividades del sistema.
- **La interface de computadora:** Se comunica con el sistema y las computadoras.

Ventajas de los Sistemas AGV

- Flexibilidad: Estos sistemas son los más flexibles comparados con otros sistemas utilizados en el manejo de materiales. Entre sus ventajas se puede nombrar la alta flexibilidad en los cambios que se pueden hacer en sus recorridos lo cual permite un mejor uso del espacio.
- Mayor confiabilidad: Son los más confiables a comparación de otros tipos de sistemas utilizados para el

manejo de materiales. Fácilmente el AGV puede ser sustituido por otro en caso de inconvenientes sin causar un paro en el sistema de manufactura.

- Ahorro de Inversión: El costo de operación de los AGV es menor que el de otros sistemas de manejo de materiales ya que solo se necesita una persona para su programación. Es fácil de realizarle mantenimiento, consume menos potencia y es muy raro que falle logrando que el tiempo muerto del sistema sea bajo gracias a su funcionamiento continuo.
- Movimientos libres: La trayectoria a seguir asegura suavidad, flexibilidad y confiabilidad en las operaciones. Al ser un robot sus movimientos van a estar completamente estandarizados, y si la programación es la adecuada el manejo de materiales se hará de forma segura.
- Aplicaciones de los AGV: Estos vehículos cuentan con varias aplicaciones, entre las más importantes existe el transporte, carga y descarga de materiales y limpieza.

Sistemas de seguridad

El AGV consta de múltiples sensores para tener un ambiente más seguro, scanner laser, cubriendo el perímetro del área de trabajo, disminuyendo su velocidad en cuanto detecta una presencia en su entorno, hasta el punto de detenerse si la presencia es muy cercana evitando un choque. Hay múltiples tipos de sensores utilizados para evitar los obstáculos: sensores infrarrojos, ultrasónicos, magnéticos, inductivos, etc. También, como parte del sistema de seguridad, poseen bumpers que evitan el daño en el AGV.

También cuenta con distintos sistemas de seguridad para el manejo de la carga, que detectan el estado de la carga para evitar accidentes en la carga, descarga y transporte de material. Sistema de diagnóstico, es uno de los más importantes, ya que permite analizar el funcionamiento completo del AGV en cualquier momento, este reduce el tiempo de resolución de incidencias y se pueden resolver problemáticas del funcionamiento de este con anticipación.

Sistema de energía

Una gran mayoría de estos vehículos utilizan baterías para obtener la energía para su funcionamiento, hay diferentes tecnologías de baterías, por ejemplo, las de níquel cadmio o níquel metal hidruro son recomendadas donde el tiempo de carga son mínimos, para no tener interrupciones en el vehículo. Las formas de cambio de la batería pueden ser:

1. Cambio manual: un operario sustituye la batería descargada por una cargada
2. Cambio automático: el AGV se dirige a una estación de recambio automático cuando está baja de carga la batería
3. Carga en caliente: el sistema se acerca a un punto predeterminado en donde se conecta automáticamente al cargado sin ayuda.

Tipos de AGV's

Los tipos de AGV`s que existen son utilizados para satisfacer diferentes tareas determinadas por el cliente. A continuación, se nombrarán algunos de los tipos de AGV utilizados:

- **Vehículos de arrastre:** Vehículo guiado automáticamente al que se le pueden acoplar diferentes remolques los cuales pueden servir para transportar materiales distintos. Son útiles para movilizar grandes volúmenes con peso de 1.500 kg a grandes distancias.

- **Transportadores unitarios de cargas:** Contiene rodillos mecánicos o no mecánicos permitiendo transportar una carga unitaria a bordo del vehículo. Estos vehículos son utilizados en sistemas de almacenamiento y distribución. Las trayectorias son normalmente cortas y pueden transportar de volumen grande.
- **Carretillas de tarimas:** Utilizados en funciones de distribución. Diseñados para el transporte, elevación de cargas y maniobrar cargas en tarimas.
- **Montacargas de horquilla:** Relativamente nuevos. Capacidad de recoger y descargar en tarimas. Utilizado en sistemas que requieren total automatización.
- **Transportadores de carga ligera:** Transporte de carga liviana, capacidades menores de 230 Kg. Utilizados para distancias moderadas entre los almacenamientos y diferentes estaciones de trabajo. Diseñado para espacios limitados.
- **Vehículo de línea de ensamble:** Utilizado para transportar motores, transmisiones y/o vehículos que hacen parte de una línea de ensamble. Estos vehículos sirven para llevar las partes de una estación de trabajo a la otra o para manipular los productos terminados.
- **Vehículos de carga trilateral:** Vehículos con capacidad de trabajar en pasillos estrechos en donde se tiene que hacer la carga y descarga de materiales en poco espacio.
- **Vehículos transportadores de bobinas:** Especializados en el transporte de bobinas. Estas se pueden manejar horizontalmente o verticalmente dependiendo de la necesidad que se tenga.
- **Vehículos contrapesados:** Vehículos a los cuales están diseñados para soportar grandes pesos de más de 1.500 kg. Estos vehículos se usan principalmente como apiladores y cuentan con un contrapeso en su base la cual ayuda a que el peso del material no desequilibre el robot.
- **Vehículos para hospitales:** Diseñados para mejorar todos los procesos logísticos de los hospitales como apoyo en cocina, apoyo en lavandería, reparto de medicamentos u organización de historiales médicos.

6. METODOLOGÍA DE INVESTIGACIÓN

6.1. MATRIZ OPERACIONAL DE LA METODOLOGÍA

OBJETIVOS ESPECÍFICOS	ACTIVIDADES A EJECUTAR	RESULTADOS ESPERADOS	MATERIALES
Investigar los principios tecnológicos que conforma un vehículo autónomo y los tipo de algoritmo de una red neuronal	-Investigar sobre los principios de funcionamiento y los tipos de tecnologías para reconocimiento -Investigar los principios de los algoritmos utilizados en una red neuronal.	- Obtener información de tecnología utilizada para realizar un reconocimiento de espacios. -Algoritmo capaz de cumplir las necesidades del proyecto	-Búsqueda de información - Computadora
Diseñar los diferentes sistemas mecánicos y electrónicos que componen el AGV.	-Dibujar en un software de diseño mecánico los elementos necesarios para brindar movimiento y estabilidad al vehículo	-Simulación de comportamiento de vehículo autónomo	- Software de dibujo mecánico -Computadora
Construir el chasis que permita realizar las tareas solicitadas al AGV.	Maquinado de piezas para construcción de chasis de vehículo autónomo	-Chasis que cumpla con los usos estimados para el vehículo autónomo	- Piezas mecánicas -Soldador -Maquinas herramientas
Ajustar y configurar los diferentes elementos que conforman el AGV	-Elegir los componentes primarios para el control, la fuerza y el sensado del vehículo autónomo	-Elementos funcionales capaz de realizar las tareas que competen al vehículo autónomo	-Elementos electrónicos de control
Desarrollar algoritmos para la construcción de mapas de localización de ambientes industriales y comerciales	-Programar los diferentes sistemas para el buen funcionamiento del vehículo autónomo	-Vehículo autónomo capaz de moverse en diferentes ambientes cerrados	- Laptop -Controladores
Realizar pruebas de funcionamiento mecánico	Verificar el funcionamiento mecánico en ambientes cerrados	Una buena Movilidad del AGV	-AGV
Realizar pruebas del sistema de la red neuronal, aprendizaje profundo.	- Realizar pruebas de rutas dinámicas con las redes neuronales	- Carro autónomo capaz de evitar obstáculos y de aprender sobre ambientes dinámicos	- Laptop -AGV

6.2. DEFINICIÓN DEL CONCEPTO DEL VEHÍCULO – SELECCIÓN DEL TIPO

El primer paso consistió en la investigación bibliográfica, de la que partimos para definir que antes de hacer cualquier diseño, es indispensable determinar el ambiente, industria o comercio en el cual se utilizará el vehículo, ya que esto determinará el tipo de vehículo a construir, impactando en su diseño y todos los demás factores involucrados. Se contactó con la Industria Texano, y se hizo una visita técnica a su planta, ubicada en Armenia, Sonsonate.

Según su sitio web (http://www.industriastexano.com), Industrias Texano es una empresa que se dedica a la fabricación y comercialización de jeans, pantalones casuales y jackets tanto para hombres como para mujeres, actualmente se comercializan en Centro América las marcas Sergio Valente y Euro Silver. En su misión dice "Nuestra misión es brindar al mercado latinoamericano una variedad de prendas de vestir de calidad y a un precio accesible, basados en el buen servicio al cliente, la honestidad, la integridad y el respeto por los demás".

De la visita se obtuvo la siguiente información:

1. La construcción es de 600 m^2 de área, distribuidos en dos plantas: en la primera planta, la sala de venta, almacenaje, máquina cortadora de patrones y máquina bordadora. En la segunda planta tiene las máquinas ranas, es donde se elaboran los pantalones, se llevan las piezas cortadas a los diferentes puestos de trabajo y luego los pantalones ya elaborados, son transportados al área de almacenaje temporal. El peso máximo en Kg es de 25 Kg que se mueve entre las estaciones de trabajo. En la Fig. 2 se muestran los ambientes de la segunda planta.
2. Se selecciona la segunda planta como el ambiente de trabajo para el vehículo autónomo, no es posible el transporte del AGV entre la primera planta y la segunda planta, no cuenta la edificación con ascensor y la escalera es reducida.
3. Basados en la información obtenida, y las imágenes de la Fig. 2 y Fig. 3, se selecciona un transportador de carga ligera con capacidad de hasta 30 Kg.

Fig. 2: Ambientes de Industria Texano (2ª planta)



Fig. 3: Ambientes de Industria Texano (1ª planta)

En la Fig. 3, además del área de bodega (almacenamiento de pantalones), se puede observar un carro de 4 rodos que es el utilizado para transportar los pantalones y piezas de lona entre las diferentes áreas de trabajo. Este carro tiene las dimensiones exactas para transitar por todos los pasillos de la planta con materiales, sus dimensiones, largo, ancho y alto son de 82 cm × 50 cm × 60 cm.

6.3. ELABORACIÓN DEL DISEÑO DEL AGV.

Mecánico

Mediante el paquete de diseño CAD 3D, Solidworks, se elaboró el diseño de un vehículo con medidas: longitud 80 cm, ancho 50 cm y de alto 24 cm, con una depresión interna a 16cms de altura, para colocar en ella toda la electrónica y procesamiento necesario para que el vehículo realice su función, así como tener espacio para las baterías.

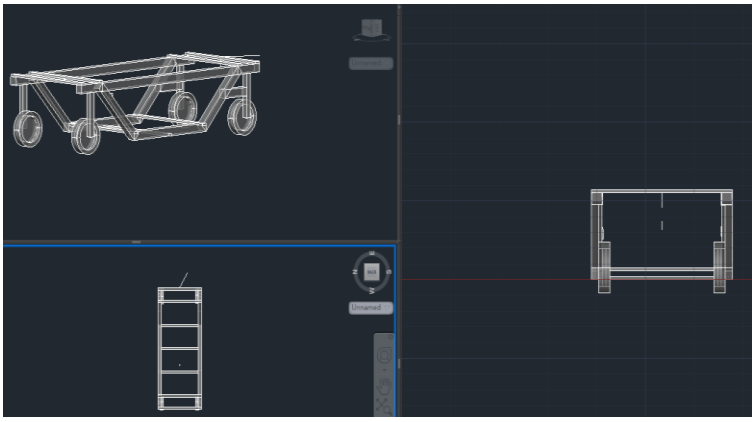


Fig. 4: Elaboración de diseño AGV en Solidworks

En la Fig. 5 se muestra el vehículo en 3D con mayor detalle.

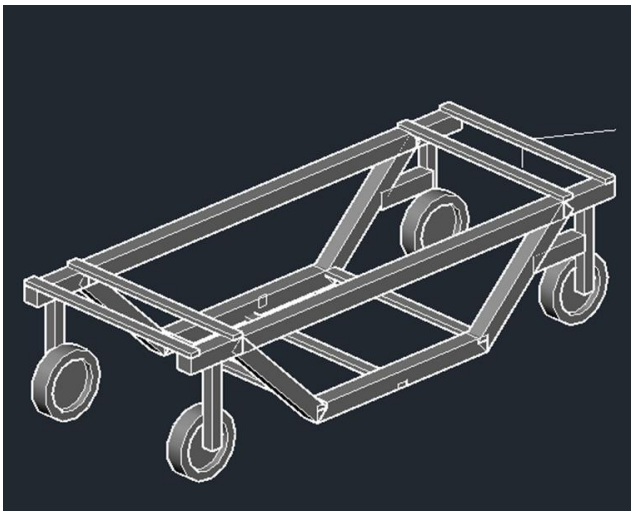


Fig. 5: Diseño de AGV en Solidworks

Construcción

Se elaboró el chasis del vehículo con tubo estructural cuadrado de 1 ½“ de hierro y con lámina de 1/16”, se consideró un espacio de 8 cm de altura por 27 cm de ancho por 50 cm de longitud para colocar el resto de componentes, a detallar más adelante. Las dos ruedas delanteras son de hule sólidas de 160 mm × 20 mm, y las traseras son rodos giratorios de 160 mm de diámetro (ruedas locas). Se utilizó soldadura MIG para soldar todos los componentes que conformarían el chasis. En la Fig. 6, 7 y 8 se muestra el vehículo terminado.

Tracción

Para la tracción, se escogió tracción delantera, proporcionada por dos motores DC de 12 V cada uno, con una potencia máxima de hasta 80 W por motor. Los motores utilizados, son los mismos motores de limpiaparabrisas que utilizan los vehículos marca Toyota modelo Yaris 2005, en base a la experiencia en proyectos previos de las carreras de Electrónica y Mecatrónica, en años anteriores, donde han tenido un magnífico desempeño.

Fig. 5: Motores DC a 12 V, 80 W

Fig. 6: Parte trasera del chasis

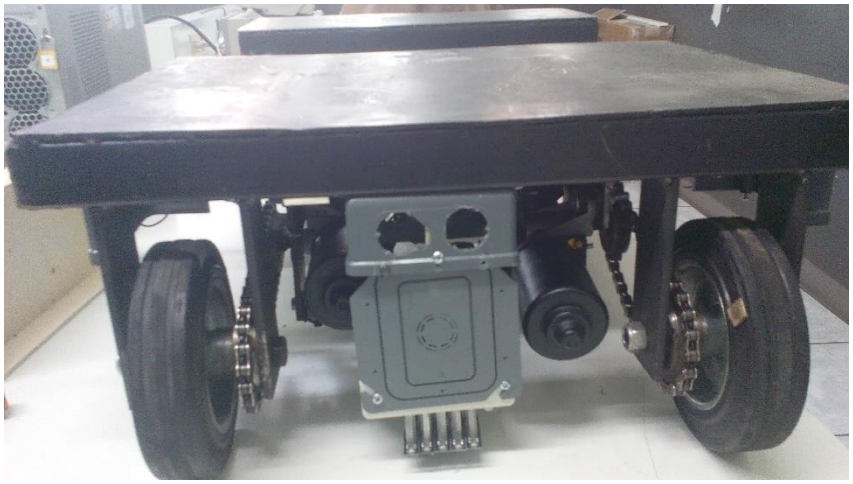


Fig. 7: Parte delantera del chasis

Fig. 8: Área central del chasis, espacio para demás componentes

6.4. ELABORACIÓN DE ETAPA ELECTRÓNICA DEL AGV.

Etapa de potencia

Los motores DC utilizados requieren hasta 8 A, en principio, el vehículo debe ser capaz de moverse hacia atrás y hacia adelante, para lograr dicha inversión de giro se requiere de una etapa electrónica que sea capaz de invertir la polaridad del voltaje en los terminales del motor. También deben ser capaces de variar su velocidad de giro de forma programable.

En la Fig. 9, se observa un motor alimentado por una fuente DC, por ejemplo 12 V con dos interruptores.

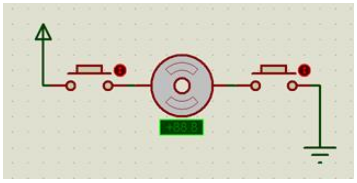


Fig. 9: Polarización motor DC

En esta configuración, un interruptor interrumpe la conexión al voltaje de alimentación de uno de los terminales del motor, y otro interruptor interrumpe la conexión a tierra del otro terminal del motor, si se activan ambos, el motor gira en un sentido.

Aprovechando este tipo de conexiones podríamos armar dos circuitos separados que compartan el mismo motor, así, si activamos un par de botones podemos hacer que gire en un sentido (1-4), y activando el otro par de botones (2-3), el motor gira en el sentido contrario, como se muestra en la Fig. 10.

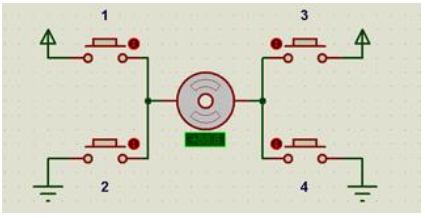


Fig. 10: Polarización motor DC

Es importante mencionar, que la lógica de activación debe evitar ciertas combinaciones de conexiones: no se pueden activar los interruptores 1 y 2, porque provocarían un cortocircuito en la fuente, así como la activación simultánea de 2 y 4, que también producen un cortocircuito en la fuente. También, otra combinación que debe ser evitada, es activar todos los interruptores al mismo tiempo, produciendo un cortocircuito de la fuente. Las combinaciones de activación que son permitidas son:

1-4: Gira en sentido horario el motor

2-3: Gira en sentido antihorario

2-4: Al estar girando el motor, detendrá el motor, no importa el sentido en que gire

Puente H con relés

Para poder controlar los interruptores, se sustituyen los interruptores por relés, tenemos el circuito mostrado en la Fig. 11.

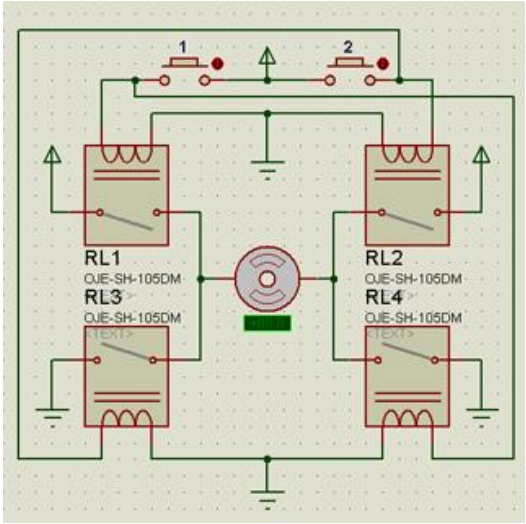


Fig. 11: Puente H con relés

Para que la etapa de potencia pueda ser controlada por un microcontrolador, como el Arduino, es necesario que cada interruptor pueda ser manipulado por los pines de salida del Arduino, para ello se colocó un transistor para manipular cada relé, el pin de salida del Arduino alimenta la base del transistor, el cual activa o desactiva el relé dependiendo del estado del pin, como se muestra en la Fig. 12.

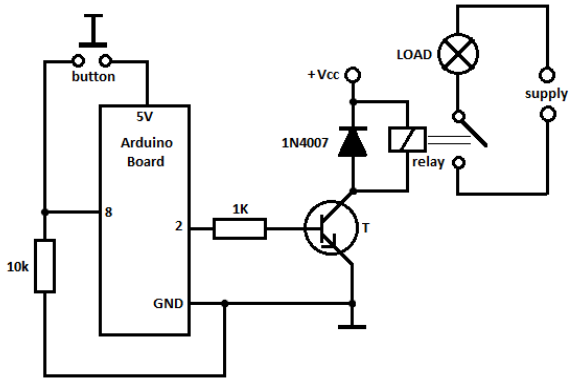


Fig. 12: Conexión de salida de Arduino a relé

Con el circuito modificado, se puede hacer girar el motor en un sentido o en otro, controlado por un programa desde Arduino, pero aún no se puede controlar la velocidad de giro. Uno de los métodos para controlar la velocidad de giro de los motores DC, es enviar un tren de pulsos al motor, entre 0 y 12 V, modulando el tiempo que está en 12 V dicha señal, podemos aumentar o disminuir la velocidad de giro, mayor tiempo en 12V, mayor velocidad, menor tiempo en 12 V, disminuye la velocidad. A este sistema se le denomina, control PWM, y el Arduino cuente con pines que incorporan PWM a 500 Hz, frecuencia que no soportan los relés. Para que el circuito pueda modularse con PWE, SE modificó el circuito, colocando un MOSFET de potencia canal N, modelo IRFZ44N, en serie con cada uno de los relés que conecta a tierra el terminar del motor, estos MOSFETs son controlados por los pines del Arduino que generan PWM, obteniéndose el control de velocidad deseado.

Puente H con MOSFET[2]

El circuito anterior tiene una seria desventaja, el uso de relé, producirá que, a corto o mediano plazo, el relé se dañe por la continua conmutación que hace para cambiar de giro el motor, o al pasar del estado de reposo a movimiento en el motor. Fue necesario sustituir todos los relés por MOSFET, utilizándose el IRFZ44N que

había demostrado un buen desempeño, pudiendo manejar hasta 50 A cada uno. En la Fig. 13 se muestra el circuito completo con MOSFET.

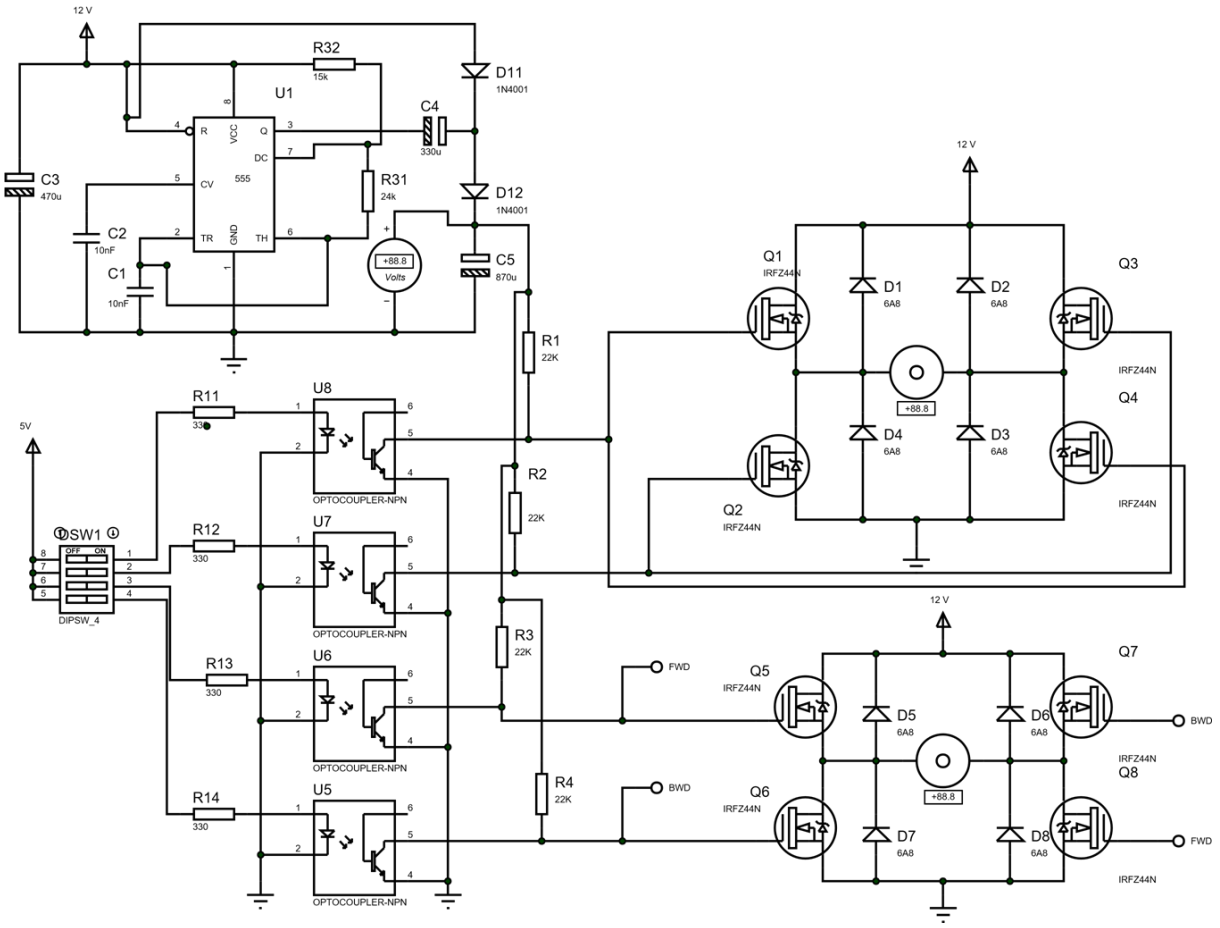


Fig. 13: Puente H con MOSFET

Descripción y funcionamiento

Entradas: las entradas, que están conectadas a SW1, son los pines que se conectan al Arduino, están optoaisladas, para separar la etapa de potencia (MOSFET) de la etapa de control (Arduino). Son bajas activas, por defecto, en reposo deben estar a 5 VDC.

Duplicador de voltaje: la etapa del 555 duplica el voltaje de la fuente, generando a partir de 12 V, un voltaje aproximado de 22.3 V entre los terminales de C5, tal circuito se agregó al observar que los transistores Q1, Q3, Q5 y Q7, necesitaban en el pin Gate más de 12 V para dispararse, y aunque se disparaban con 12 V, pero el voltaje en el motor era de 8.3V, por la caída que necesita G-S para mantenerse activo.

Puente H: las entradas de la etapa de potencia, que denominaremos pines 1, 2, 3 y 4 (aunque en el Arduino son pines 5,6, 10 y 11) son salidas del Arduino y son bajas activas, se activan con 0, y son inactivas en 1. Para hacer girar el motor izquierdo en sentido horario (hacia adelante), se activan Q1 y Q4, colocando los pines 1 en 0V y el pin 2 en 5V. Para hacerlo girar en sentido antihorario, colocamos los pines 1 en 5V y el pin 2 en 0V, activando respectivamente Q2 y Q3. El PWM se logra, en sentido horario con el pin 1, y en sentido antihorario con el pin 2, ambos conectados a salidas PWM del Arduino.

Para el motor derecho, similar: para hacer girar el motor izquierdo en sentido horario (hacia adelante), se activan Q5 y Q8, colocando los pines 3 en 0V y el pin 4 en 5V. Para hacerlo girar en sentido antihorario, colocamos los pines 3 en 5V y el pin 4 en 0V, activando respectivamente Q6 y Q7. El PWM se logra, en sentido horario con el pin 3, y en sentido antihorario con el pin 4, ambos conectados a salidas PWM del Arduino.

Observación: en caso de no tener voltaje en los pines del Arduino, o encontrarse en 0 V, por ejemplo, porque no le llega energía al Arduino, todos los MOSFET se cortocircuitarán (dañando los MOSFET y descargando la batería) porque son bajos activos, para ello se implementó una etapa con relé, que desactiva la fuente de 12 VDC a los MOSFET, y se activa hasta que el Arduino ha desactivado los pines 1, 2, 3 y 4.

En la Fig. 14 se observa el prototipo de la etapa de potencia finalizada:

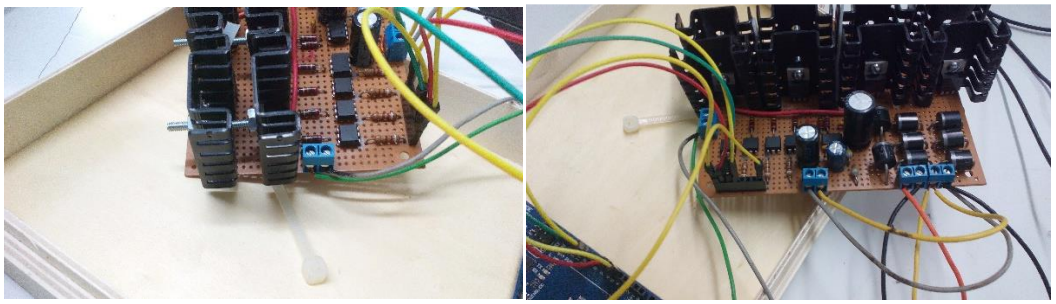


Fig. 14: Etapa de potencia con MOSFET

6.5. Prueba del Prototipo Mecánico y Etapa de Potencia

En este punto se hacen las pruebas iniciales: se conectaron los motores a la etapa de potencia, el Arduino se conectó a las entradas de la etapa de potencia, y por medio de un programa simple se probó que el vehículo realice los movimientos deseados.

Estas son las conexiones del Arduino a la etapa de potencia:

- Pin 5: activa MOSFET Q1 y Q4 del motor izquierdo
- Pin 6: activa MOSFET Q3 y Q4 del motor izquierdo
- Pin 10: activa MOSFET Q5 y Q8 del motor derecho
- Pin 11: activa MOSFET Q6 y Q7 del motor derecho
- Pin 18 y 19: interface bluetooth, de esta forma se conecta la tablet, vía bluetooth.

Funcionamiento: básicamente este código recibe cuatro letras por el puerto Bluetooth: A, B, C, D o E, estás son enviadas por una tablet que ejecuta ArduinoRC. El programa funciona así:

- Si recibe A de la tablet: se mueve hacia adelante el AGV durante 2 segundos y para.
- Si recibe B de la tablet: se mueve hacia atrás el AGV durante 2 segundos y para.
- Si recibe C de la tablet: gira a la izquierda el AGV durante 2 segundos y para.
- Si recibe D de la tablet: gira a la derecha el AGV durante 2 segundos y para.

Se implementó el siguiente código:

```
char DATO_0;
int TIME=500;
int i1 = 5;//Salidas para el motor1
int i2 = 6;//Salida para motor1
int i3 = 10;//Salida para motor2
int i4 = 11;//Salida para motor2
void setup() {
 Serial.begin(9600);
 pinMode(i1, OUTPUT);
 pinMode(i2, OUTPUT);
 pinMode(i3, OUTPUT);
 pinMode(i4, OUTPUT);
 }
void loop() {
 if (Serial.available()){
  DATO_0= Serial.read();
  }
  switch (DATO_0){
  case 0x41:
     forward();
     delay(2000);
     stop();
     break;
  case 0x42:
     backward();
     delay(2000);
     stop();
     break;
   case 0x43:
    turn_left();
     delay(2000);
     stop();
     break;
  case 0x44:
     turn_right();
     delay(2000);
     stop();
     break;
  }
}
void backward(){
 Serial.println("Retroceso-recto...");
 digitalWrite(i1,HIGH);
 digitalWrite(i2,LOW);
 digitalWrite(i3,HIGH);
 digitalWrite(i4,LOW);
 }
 void forward(){
 Serial.println("Avance-recto...");
 digitalWrite(i1,LOW);
 digitalWrite(i2,HIGH);
 digitalWrite(i3,LOW);
 digitalWrite(i4,HIGH);
 }
void stop(){
 Serial.println("Parada...");
 digitalWrite(i1,HIGH);
 digitalWrite(i2,HIGH);
 digitalWrite(i3,HIGH);
 digitalWrite(i4,HIGH);
 }
void turn_left() {
 Serial.println("GiroIzquierda...");
 digitalWrite(i1,LOW);
 digitalWrite(i2,HIGH);
 digitalWrite(i3,HIGH);
 digitalWrite(i4,HIGH);
}
void turn_right(){
 Serial.println("GiroDerecha...");
 digitalWrite(i1,HIGH);
 digitalWrite(i2,HIGH);
 digitalWrite(i3,LOW);
 digitalWrite(i4,HIGH);}
```

Resultados de la primera prueba: el vehículo intentaba moverse, pero las cadenas de los motores se salían de posición normal y ya no se movía el vehículo, solo los motores seguían girando.

Diagnóstico y reparación: ajustar el vehículo para mantener las cadenas en su sitio. Después de ajustar la parte mecánica, colocarle arandelas donde era necesario se hizo la segunda prueba.

Resultados segunda prueba: se mantenía la cadena en su sitio, pero no avanzaba como es debido porque una de las llantas con tracción no tenía contacto con el piso, además una de las llantas tendía a bloquearse en parte del recorrido.

Diagnóstico y reparación: el chasis estaba ligeramente torcido, por eso no asentaba bien la llanta, además la soldadura de una de las llantas con el piñón había quedado abultada, y ese abultamiento topaba con el chasis, provocando que se bloqueará la llanta. Se desarmó totalmente el chasis, se le quitó lo torcido y se limo el abultamiento, listos para la tercera prueba.

Resultados de la tercera prueba: asentaban bien las llantas, pero el vehículo no avanzaba recto, se tendía a girar hacia la izquierda, aunque se le indicará que se fuera recto.

Diagnóstico y reparación: el problema era más serio de lo pensado, lo que sucede es que los motores de limpiaparabrisas no tienen la misma velocidad cuando giran en sentido horario que cuando giran en sentido antihorario, y como se habían instalado -en el caso de avance hacia adelante- para que el motor izquierdo girará en sentido horario y el derecho en sentido antihorario, al girar a diferentes revoluciones, el vehículo tendía irse hacia el lado del motor que giraba más rápido. Solución: la solución más simple parecía que ambos motores giraran en el mismo sentido, pero implicaba desarmar el chasis del vehículo y reconstruirlo desde cero con ese requerimiento, una tarea bastante compleja. Se optó por hacer la corrección en la programación: por medio del programa se ajustó la velocidad de los motores para que giraran al mismo ritmo, y se mantuviera en una línea, el código del programa se muestra a continuación:

```
char DATO_0;
int TIME=500;
int i1 = 5;//Salidas para el motor1
int i2 = 6;//Salida para motor1
int i3 = 10;//Salida para motor2
int i4 = 11;//Salida para motor2
int pwm_izq = 7;
int pwm_der = 8;
int timeAvance = 100;
int velocidad = 175;
int velocidadizq = 158;

void setup() {
 Serial1.begin(9600);
 Serial.begin(9600);
 pinMode(i1, OUTPUT);
 pinMode(i2, OUTPUT);
 pinMode(i3, OUTPUT);
 pinMode(i4, OUTPUT);
}
void loop() {
 velocidad = constrain(velocidad, 125, 255);
 velocidadizq = velocidad * 0.9;
 analogWrite(pwm_izq,velocidadizq);
 analogWrite(pwm_der,velocidad);
 if (Serial1.available()){
  DATO_0= Serial1.read();

  switch (DATO_0){
  case 0x41:
    forward();
    delay(timeAvance);
    break;
  case 0x42:
```

```
      backward();
      delay(timeAvance);
      break;
    case 0x43:
     turn_left();
      delay(timeAvance);
      break;
   case 0x44:
      turn_right();
      delay(timeAvance);
      break;
   case 0x45:
      velocidad = velocidad + 10;
      break;
   case 0x46:
      velocidad = velocidad - 10;
      break;
  }
  }
  else{
   Detener();
   delay(timeAvance);
  }
}
void backward(){
  Serial.println("Retroceso-recto...");
  digitalWrite(i1,LOW);
  digitalWrite(i2,HIGH);
  digitalWrite(i3,LOW);
  digitalWrite(i4,HIGH);

 }
```

```
 void forward(){
  Serial.println("Avance-recto...");
  digitalWrite(i1,HIGH);
  digitalWrite(i2,LOW);
  digitalWrite(i3,HIGH);
  digitalWrite(i4,LOW);

 }
void Detener(){
  Serial.print("Parada...  ");
  Serial.println(velocidad);
  digitalWrite(i1,LOW);
  digitalWrite(i2,LOW);
  digitalWrite(i3,LOW);
  digitalWrite(i4,LOW);
  }

void turn_left() {
  Serial.println("GiroIzquierda...");
  digitalWrite(i1,HIGH);
  digitalWrite(i2,LOW);
  digitalWrite(i3,LOW);
  digitalWrite(i4,LOW);
}

void turn_right(){
  Serial.println("GiroDerecha...");
  digitalWrite(i1,LOW);
  digitalWrite(i2,LOW);
  digitalWrite(i3,HIGH);
  digitalWrite(i4,LOW);
}
```

El vehículo demostró moverse según lo esperado al introducir las ordenes en la tablet, avanzaba recto, retrocedía, giraba a la izquierda y giraba a la derecha durante un tiempo definido por la variable Timeavance. Prueba superada.

6.6. FUNCIONAMIENTO DE LA ETAPA DE SENSORES

Son varios los tipos de sensores considerados para la implementación del AGV: infrarrojos, ultrasónicos, LIDAR. A continuación, se describe brevemente cada uno de ellos:

Infrarrojo: Un sensor de infrarrojo se compone de dos partes, un diodo emisor infrarrojo y un fototransistor que opera en la misma longitud de onda. Estos sensores suelen ser de corto alcance, y cuando hay una superficie suficientemente plana y sólida lo bastante cerca, la señal emitida por el diodo es recibida por el fototransistor.

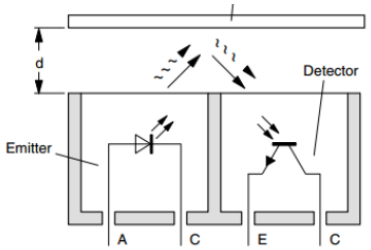


Fig. 15: Estructura interna de un sensor infrarrojo

Son dos los tipos utilizados:

Sensor Infrarrojo Sharp: es un infrarrojo que concentra la luz mediante lentes convexos y es capaz de medir distancias (hasta 50 cm) hacia los objetos en los que rebota la luz del led emisor.

Fig. 16: Sensor infrarrojo tipo SHARP.

Sensor IR Line Tracking: es un sensor infrarrojo usado para hacer el proceso de sigue línea, si el sensor está sobre la línea, genera un nivel lógico 1, y si sale de la línea genera un nivel lógico 0. Este sensor cuenta con un potenciómetro, para ajustar la activación del sensor. Su alcance máximo 1 cm o menos.

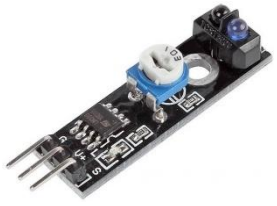


Fig. 16: Sensor infrarrojo Line Tracking

LIDAR: LIDAR es el acrónimo de Light Detection and Ranging, es decir, detección por luz y distancia. Se trata de un sistema láser que permite medir la distancia entre el punto de emisión de ese láser hasta un objeto o superficie. El tiempo que tarda ese láser en llegar a su objetivo y volver del mismo, es lo que nos dice la distancia entre los dos puntos. El resultado es que se puede obtener un mapa en 2D o 3D de alta resolución para conocer el terreno en cuestión.

Este sensor tiene un alcance para distancias de hasta 40 m (~ 131 pies) con una precisión de ± 2,5 cm (~ 1 pulgada) y un tiempo de adquisición inferior a 20 ms. Todo el sensor consume menos de 100 mA a 5 V DC. El dispositivo utiliza una interfaz I2C simple para comunicar las distancias medidas.

Fig. 17: Sensor LIDAR LITE de Garmin

Sensor ultrasónico. Su funcionamiento es simple, envía una señal ultrasónica inaudible y entrega el tiempo que demoro en ir y venir hasta el obstáculo más cercano que detecto. Generalmente están conformados por dos cilindros puestos uno al lado del otro, uno de ellos es quien emite la señal ultrasónica, mientras que el otro es quien la recibe, es un sistema muy simple pero no por eso deja de ser efectivo. El sensor HC-SR04 en particular tiene una sensibilidad muy buena del orden de los 3mm, teniendo en cuenta que la mayoría de las aplicaciones donde este sensor es utilizado es para medir o detectar obstáculos o distancias mayores a varios centímetros, podemos decir que su sensibilidad es muy buena.

Fig. 18: Sensor Ultrasónico HC-SR04

Webcam HD300. Es una cámara para capturar video y transmitirlo de manera remota mediante un procesador Raspberry, de tal manera que la persona que esté a cargo de la supervisión del AGV sea capaz de ver, mediante una tablet o computador, el entorno donde está el robot, y determinar algún posible obstáculo ante el cual el vehículo se ha detenido.

Fig. 19: Cámara Web

Funciones de los sensores

En principio, los sensores infrarrojos serían los detectores de obstáculos atrás y adelante del vehículo, en la fase de aprendizaje, el sensor LIDAR, detectaría el ambiente y generaría un mapa en 2D de la ubicación, para luego, en la fase de ejecución, navegaría en dicho mapa de un punto A un punto B, ubicado dentro del mapa generado anteriormente. Ante cualquier obstáculo, los sensores infrarrojos y ultrasónico lo detectarían y detendrían el avance del vehículo.

Se muestran los ambientes donde el vehículo sería probado en Industrias Texano, Fig. 20.

Fig. 20: Ambientes en industria Texano donde se desempañaría el vehículo

Para efectos de prueba y calibración del sensor, se utilizó el laboratorio de Electrónica de la Institución, con ciertas similitudes, Fig. 21.

Fig. 21: Ambientes usado para prueba de LIDAR en la Institución

Pruebas con sensor LIDAR

En esencia, si se usa el LIDAR, para generar un mapa, ya sea 2D o 3D, debe hacer un barrido, es necesario acoplarle un servomecanismo y un adaptador plástico (impreso en 3D), como se muestra en la Fig. 22. En la Fig. 23 se muestra uno de los sensores acoplado al AGV.

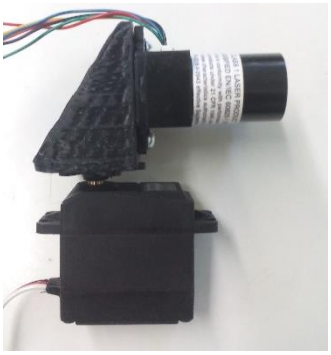


Fig. 22: Sensor LIDAR con Servomecanismo

Fig. 23: Sensor LIDAR instalado a un costado del AGV

En la fig. 24 se muestra la conexión del Arduino a los sensores LIDAR.

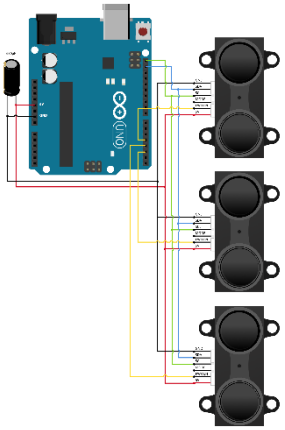


Fig. 24: Conexión de sensores LIDAR a Arduino

Una vez instalado y conectado, se ejecutó el software de prueba, proporcionado por un revendedor del fabricante, cuyo listado se da a continuación.

```
#include <Wire.h>


#define    LIDARLite_ADDRESS   0x62        // Default I2C Address of LIDAR-Lite.
#define    RegisterMeasure     0x00        // Register to write to initiate ranging.
#define    MeasureValue        0x04        // Value to initiate ranging.
#define    RegisterHighLowB    0x8f        // Register to get both High and Low bytes in 1 call.
#include <Servo.h>
Servo myservo;
int pos = 0;       // Position of the servo (degress, [0, 180])
int distance = 0;    // Distance measured
```

```
void setup()
{
 // Serial output
 Serial.begin(9600);
 Serial.println("< START >");
 // Servo control
 myservo.attach(5);
 // LIDAR control
 Wire.begin(); // join i2c bus
}
// Get a measurement from the LIDAR Lite
int LIDARGetRange(void)
{
 int val = -1;
 Wire.beginTransmission((int)LIDARLite_ADDRESS); // transmit to LIDAR-Lite
 Wire.write((int)RegisterMeasure); // sets register pointer to  (0x00)
 Wire.write((int)MeasureValue); // sets register pointer to  (0x00)
 Wire.endTransmission(); // stop transmitting
 delay(20); // Wait 20ms for transmit
 Wire.beginTransmission((int)LIDARLite_ADDRESS); // transmit to LIDAR-Lite
 Wire.write((int)RegisterHighLowB); // sets register pointer to (0x8f)
 Wire.endTransmission(); // stop transmitting
 delay(20); // Wait 20ms for transmit
 Wire.requestFrom((int)LIDARLite_ADDRESS, 2); // request 2 bytes from LIDAR-Lite
 if(2 <= Wire.available()) // if two bytes were received
 {
   val = Wire.read(); // receive high byte (overwrites previous reading)
   val = val << 8; // shift high byte to be high 8 bits
   val |= Wire.read(); // receive low byte as lower 8 bits
 }
 return val;
}
void serialPrintRange(int pos, int distance)
{
   Serial.print("Position (deg): ");
   Serial.print(pos);
   Serial.print("\t\tDistance (cm): ");
   Serial.println(distance);
}
void loop()
```

```
{
  for(pos = 0; pos <= 180; pos += 1)
  {
    myservo.write(pos);
    distance = LIDARGetRange();
    serialPrintRange(pos, distance);
    delay(20);
  }
  for(pos = 180; pos>=0; pos-=1)
  {
    myservo.write(pos);
    distance = LIDARGetRange();
    serialPrintRange(pos, distance);
    delay(20);
  }
}
```

LIDAR utiliza la teledetección activa[3]. Esto significa que el sistema LIDAR envía un pulso de luz y espera a que el pulso regrese. Además, LIDAR es una herramienta de muestreo. Lo que quiero decir con eso es que tiene la capacidad de enviar 160,000 pulsos por segundo. En una nube de puntos LIDAR, crea millones de puntos, los cuales son graficados para representar las superficies y objetos, como en la Fig. 25.

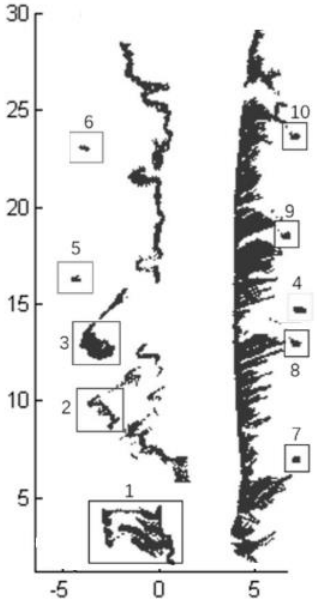


Fig. 25: Resultado de escaneo con LIDAR después de aplicar filtro ETEW

Resultado de prueba de LIDAR

LIDAR es una herramienta unidireccional, es decir que envía la luz a un punto y mide el tiempo de respuesta, al agregar un servomecanismo se convierte en una herramienta 2D, ideal para detectar la superficie de la tierra usando en plataforma áreas, que al irse desplazando generan mapas de puntos 3D, al hacer el barrido perpendicular al vector de avance del vehículo. En un vehículo terrestre se usa para detectar obstáculos y superficies como se muestra en la Fig. 26.

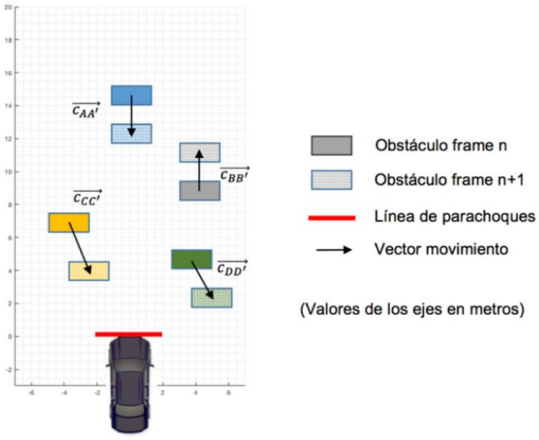


Fig. 26: Detección de objetos con LIDAR

El resultado del sensor, que son ángulos, distancias combinado con la posición del vehículo no son los esperados, no se dibuja el ambiente de la Fig. 21 como se espera. Entre las principales razones a las que atribuimos este hecho, es que sólo se hace un escaneo 2D a una altura predeterminada por la posición del sensor en el vehículo, y la otra razón, es que los objetos de la Fig. 20 (y Fig. 21) no tienen superficie de rebote, solo la estructura de madera.

Es en este punto de la investigación, donde nos encontramos con que el sensor LIDAR no es el más adecuado para la aplicación deseada y tenemos que replantearnos el proyecto.

Funcionamiento sensor ultrasónico

La interfaz del sensor HC-SR04 y Arduino se logra mediante 2 pines digitales: el pin de disparo (trigger) y eco (echo). La función de cada uno de estos pines es la siguiente:

- El pin trigger recibe un pulso de habilitación de parte del microcontrolador, mediante el cual se le indica al módulo que comience a realizar la medición de distancia.
- En el pin echo el sensor devuelve al microcontrolador un puso cuyo ancho es proporcional al tiempo que tarda el sonido en viajar del transductor al obstáculo y luego de vuelta al módulo.

Mediante una fórmula puede estimarse entonces la distancia entre el sensor y el obstáculo si se conoce el tiempo de viaje del sonido, así como la velocidad de propagación de la onda sonora. La siguiente imagen muestra los pulsos recibidos y enviados por el sensor, de acuerdo a la hoja de datos del sensor que colocamos más arriba para su descarga.

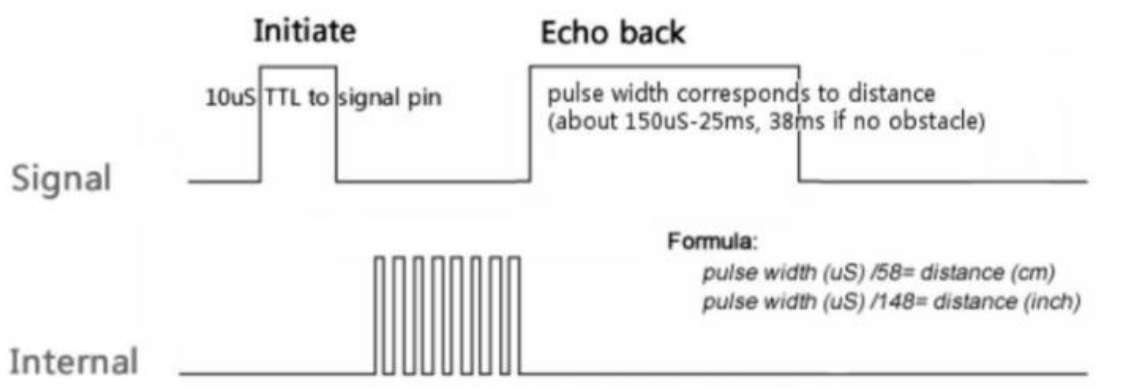


Fig. 27: Sensor ultrasónico emitiendo un pulso de 8450 kHz.

Se puede observar en la Fig. 27, que el HC-SR04 genera un pulso en el pin marcado como “echo” cuya duración es proporcional a la distancia medida por el sensor. Para obtener la distancia en centímetros, solamente se divide el tiempo en microsegundos entre 58 para obtener la distancia en centímetros (148 para pulgadas).

En la Fig. 28 se observa la conexión con Arduino. La conexión del sensor con Arduino es directa al sensor, directamente con alambres. Para lograr que el sensor funcione, son necesarias 4 señales:

- Alimentación de 5 volts.
- Tierra o común del circuito.
- Señal de disparo (trig).
- Señal de eco (echo).

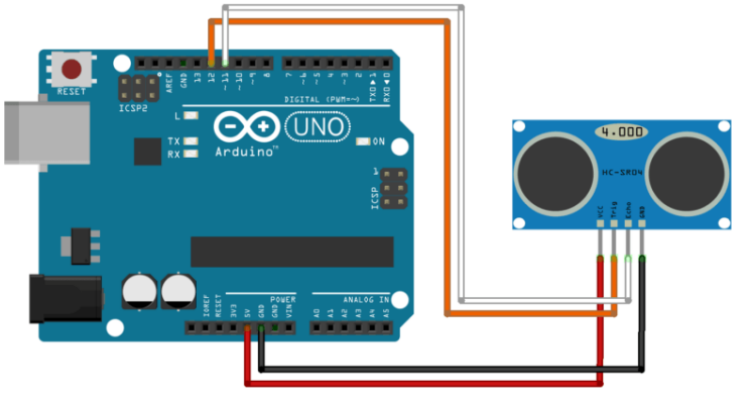


Fig. 28: Conexión de sensor ultrasónico HC-SR04 a Arduino.

Fig. 29: Sensor ultrasónico montado en AGV

En la Fig. 29 se observa uno de los sensores del costado del vehículo, acoplado al chasis.

Código integrado al AGV para el uso de sensores ultrasónicos para detección de obstáculos y paredes.

```
const int Trigger = 2;   //Pin digital 2 para el Trigger del sensor
const int Echo = 3;  //Pin digital 3 para el Echo del sensor
int ledPin =(13); // Add the onboard LED on pin 13.
char DATO_0;
int TIME=500;
int i1 = 5;//Salidas para el motor1
int i2 = 6;//Salida para motor1
int i3 = 10;//Salida para motor2
int i4 = 11;//Salida para motor2
int pwm_izq = 7;
int pwm_der = 8;
int timeAvance = 100;
int velocidad = 175;
int velocidadizq = 158;
void setup() {
  Serial.begin(9600);//iniciailzamos la comunicación
 pinMode(Trigger, OUTPUT); //pin como salida
 pinMode(Echo, INPUT);  //pin como entrada
 digitalWrite(Trigger, LOW);//Inicializamos el pin con 0
 Serial1.begin(9600);
 Serial.begin(9600);
 pinMode(i1, OUTPUT);
 pinMode(i2, OUTPUT);
 pinMode(i3, OUTPUT);
 pinMode(i4, OUTPUT);
 }
void loop() {
 long t; //timepo que demora en llegar el eco
 long d; //distancia en centimetros

 digitalWrite(Trigger, HIGH);
 delayMicroseconds(10);          //Enviamos un pulso de 10us
 digitalWrite(Trigger, LOW);
  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
 d = t/59;             //escalamos el tiempo a una distancia en cm
 Serial.print("Distancia: ");
 Serial.print(d);      //Enviamos serialmente el valor de la distancia
 Serial.print("cm");
 Serial.println();
 delay(100);
```

```
if(d <=30) // If the sensor detects an obstacle less than 30 cm in distance, the LED will start to blink.
{
digitalWrite (ledPin, HIGH);
delay(50);
}
if(d >=30)// If no obstacle is there within 30 cm, the Led should turn off.
{
digitalWrite (ledPin, LOW);
delay(50);
}
  velocidad = constrain(velocidad, 125, 255);
 velocidadizq = velocidad * 0.92;
 analogWrite(pwm_izq,velocidadizq);
 analogWrite(pwm_der,velocidad);
 if (Serial1.available()){
   DATO_0= Serial1.read();
    switch (DATO_0){
   case 0x41:
      forward();
      if(d <=30) // If the sensor detects an obstacle less than 30 cm in distance, the LED will start to blink.
{
digitalWrite (ledPin, HIGH);
delay(3000);
}
      delay(timeAvance);
      break;
   case 0x42:
      backward();
      delay(timeAvance);
      break;
    case 0x43:
     turn_left();
      delay(timeAvance);
      break;
   case 0x44:
      turn_right();
      delay(timeAvance);
      break;
   case 0x45:
      velocidad = velocidad + 10;
      break;
   case 0x46:
      velocidad = velocidad - 10;
      break;
 }
 }
 else{
  Detener();
  delay(timeAvance);
 }
}
void backward(){
 Serial.println("Retroceso-recto...");
 digitalWrite(i1,LOW);
 digitalWrite(i2,HIGH);
 digitalWrite(i3,LOW);
 digitalWrite(i4,HIGH);

 }
 void forward(){
 Serial.println("Avance-recto...");
 digitalWrite(i1,HIGH);
 digitalWrite(i2,LOW);
 digitalWrite(i3,HIGH);
 digitalWrite(i4,LOW);
  }
void Detener(){
 Serial.print("Parada...  ");
 Serial.println(velocidad);
 digitalWrite(i1,LOW);
 digitalWrite(i2,LOW);
 digitalWrite(i3,LOW);
```

```
  digitalWrite(i4,LOW);
  }

void turn_left() {
  Serial.println("GiroIzquierda...");
  digitalWrite(i1,HIGH);
  digitalWrite(i2,LOW);
  digitalWrite(i3,LOW);
  digitalWrite(i4,LOW);
}

void turn_right(){
  Serial.println("GiroDerecha...");
  digitalWrite(i1,LOW);
  digitalWrite(i2,LOW);
  digitalWrite(i3,HIGH);
  digitalWrite(i4,LOW);
}
```

Resultado de las Pruebas

Fueron exitosos, detecta los objetos en el rango previsto, pudiendo calibrarse en el programa y realizando la acción adecuada para cada caso.

Sensores IR Sharp

Estos sensores, en conjunto con los sensores ultrasónicos son detectores de proximidad, ya sean de obstáculos o de objetos circundantes, como mesas, paredes, muebles, etc. que, al aproximarse el vehículo, debe realizar una acción evasiva para evitar hacer contacto o estrellarse con dichos objetos. En la Fig. 30 se puede observar instalado uno de dichos sensores.

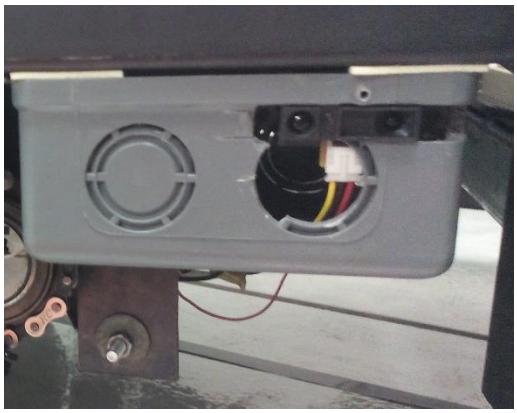


Fig. 30: Sensor IR montado en AGV

Este tipo de sensores envían un resultado analógico, el cual el Arduino lo convierte en un valor digital usando una de sus entradas analógicas, en el cuadro abajo se ve el código, donde se utiliza la función que convierte el dato enviado por el sensor en una distancia en centímetros, observe la fórmula del sensor, la cual se interpolo en base a 3 medidas realizadas con el mismo sensor para distancias conocidas.

```
void setup() {
  // Comunicación seria a 9600 baudios
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}
void loop() {
  long tiempo=millis(); //tiempo antes de iniciar la lectura
  int D_cm=distancia(40); //lectura de distancia
  tiempo=millis()-tiempo; //milisegundos que duró la lectura
// Serial.print("Tiempo de lectura: ");
//  Serial.print(tiempo);
//  Serial.print("ms  Distancia: ");
  Serial.println(D_cm);
//  Serial.println("  cm");
  delay(10);
}


float distancia(int n)
```

```
{
  long suma=0;
  for(int i=0;i<n;i++)
  {
    suma=suma+analogRead(A0);
  }
  float adc=suma/n;
  float distancia_cm = 195113307100 * pow(adc, -4.395507946);
  return(distancia_cm);
}
}
```

Sensores IR Line Tracking.

Como se mencionó en el resultado con las pruebas con LIDAR, este sensor no resultó ser el idóneo para la aplicación deseada. Manteniendo la esencia del proyecto, de que tenga un control basado en red neuronal y que use lógica difusa, se opta por un vehículo seguidor de línea[4], el cual mediante sensores IR Line tracking, sigue una línea en el área de trabajo, pasando por todos los puntos donde se necesita llevar y traer materias primas o herramientas. El control aún utiliza una red neuronal, para tomar las decisiones del vehículo y se utiliza lógica difusa en el software que controla los motores del vehículo, para tener un movimiento suave.

El principio de funcionamiento: se pinta una línea en el piso que contraste con el color del mismo, por donde se desea que el vehículo circule, y éste se mueve sobre la línea. Los sensores (Fig. 31), leen los sensores, y el control neuronal mantiene, mediante la acción adecuada el vehículo sobre la línea, buscando que el sensor central detecte la línea.

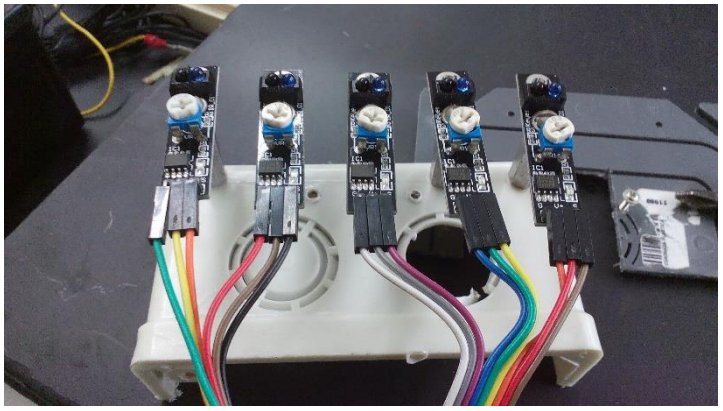


Fig. 31: Sensor IR montado en estructura plástica

La línea: aunque parece simple la idea, en el momento de aplicarla hay que considerar varios aspectos. El contraste del piso con la línea debe asegurar que el sensor se active sobre la línea y se desactive sobre el piso, lo cual no siempre es posible por la variación en el tono que tenga el piso de la instalación, como se muestra en la Fig. 32.

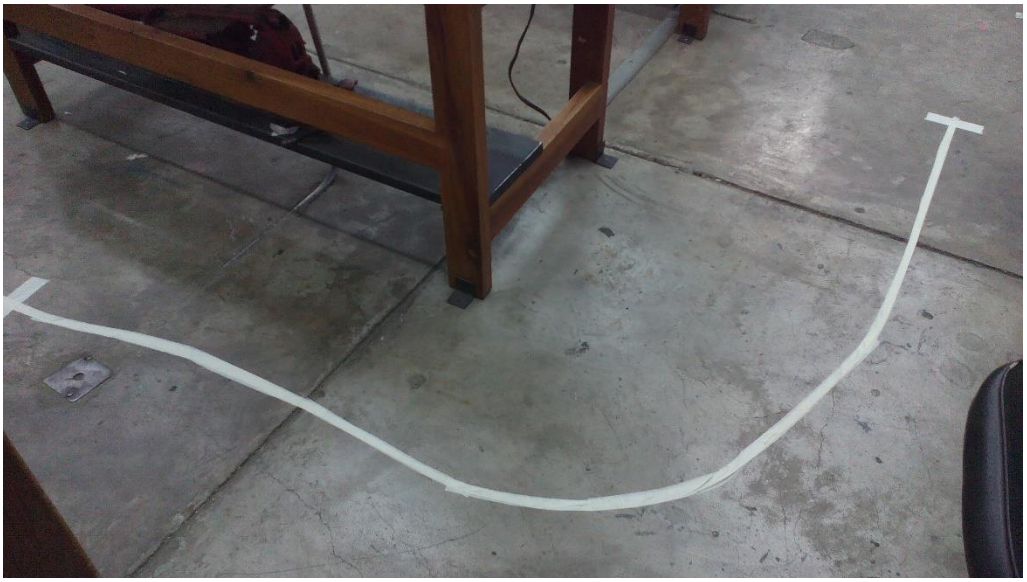


Fig. 32: Primera línea para IR Line tracking

La línea de la Fig. 32 complicó el proceso de calibración e hizo perder tiempo valioso, porque había ciertas zonas en el trayecto en que el sensor se activaba con el piso, y el vehículo continuaba circulando fuera de la línea.

Fig. 33: Segunda línea para IR Line tracking

En la segunda línea, mejoró el contraste, detectaba mejor la línea blanca sobre el plástico negro, pero aún presentaba falsas activaciones. En vista de la limitación que no se podía pintar el piso de blanco sin autorización, para colocar una línea negra, optamos por buscar otro piso, y en ella se colocó la tercera línea, que se muestra en la fig. 34.

Fig. 34: Tercera línea, para IR Line Tracking

De las tres, la línea negra con fondo blanco, es la que mejor funcionó para que circulará el vehículo.

Distribución de los Sensores

En la distribución de los sensores, según la Fig. 31, había un espacio entre ellos, que cuando quedaba la línea entre dos sensores, no detectaba la línea y el vehículo se había programado que se parara bajo esa condición. Se reordenaron los sensores como se muestra en la Fig. 35 y al instalarse, se ajustó la distancia entre el piso y los sensores para que fuera menor a 1 cm.

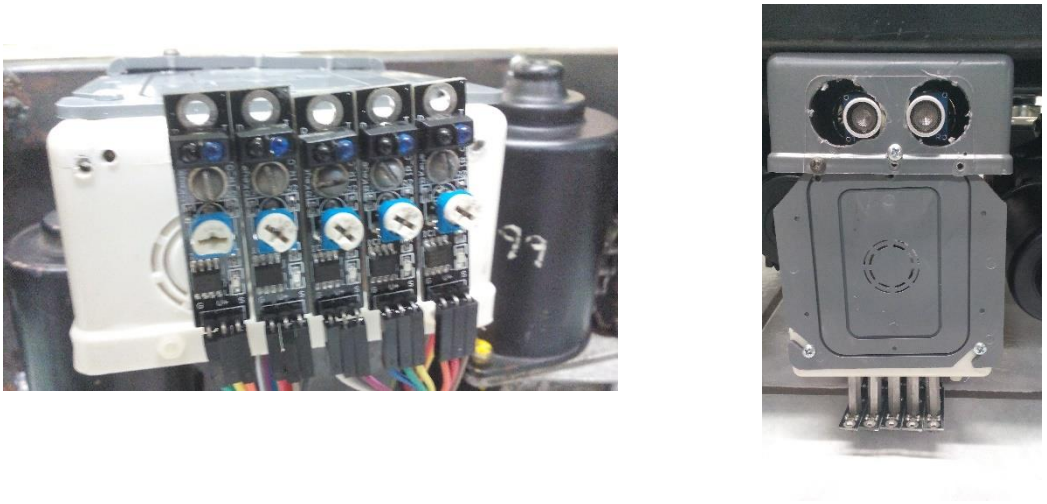


Fig. 35: Ajuste de distribución y altura de sensores IR Line tracking

Calibración de Sensores

Con la nueva línea y los sensores correctamente distribuidos se procedió a la calibración del potenciómetro, cuyo ajuste determinará que el sensor se active en el piso blanco, y se desactive sobre la cinta negra. En la Fig. 36 a 37 se observa el proceso de calibración sensor por sensor.

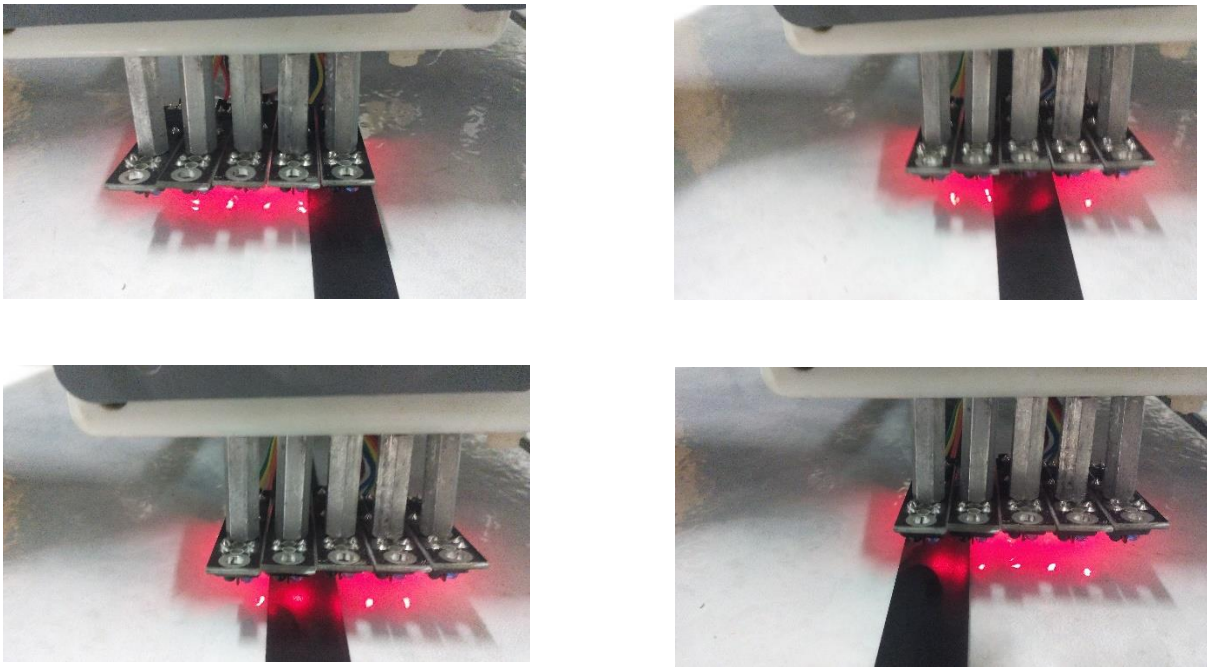


Fig. 36: Ajuste la activación de los sensores IR Line Tracking

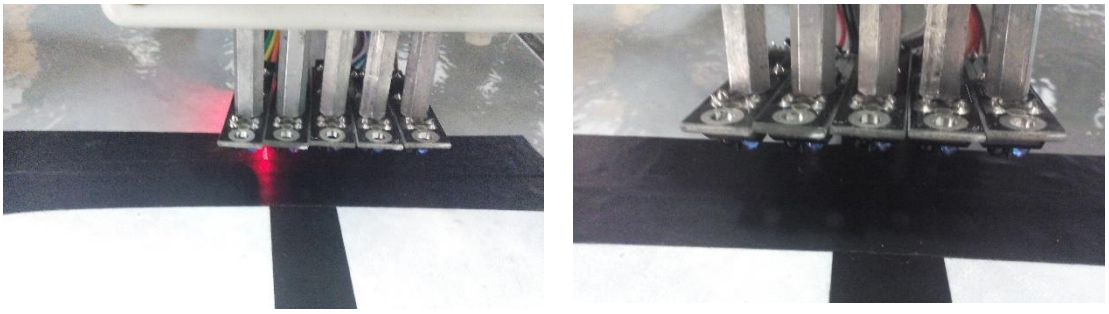


Fig. 37 Sensor desajustado a la izquierda (debería estar apagado), ajustado a la derecha

7. RESULTADOS

1. Diseño e implementación de la red neuronal, que permitirá el aprendizaje continuo para adaptarse a diferentes ambientes.

En esta red neuronal, el sistema obtiene sus entradas de los sensores, que se pueden clasificar en dos tipos: sensores de distancia y sensores seguidores de línea.

Los sensores de distancia son variados: ultrasónico, IR y LIDAR. Los sensores seguidores de línea son los IR Tracking Line.

Los datos de entrada los podemos ordenar en una tabla (Tabla 1), primero los sensores de distancia y posición al objeto detectado. Se clasifica la distancia en 0, si no hay ningún objeto detectado, 0.5 si hay objeto detectado, pero está lejos, y 1 cuando el objeto está a menos de 5 cm del sensor. La posición es definida por el sensor que lo detecta: puede ser -1 si está a la izquierda, 0 si está al centro y 1 si está a la derecha, podrían agregarse otras categorías, ya que se tienen varios sensores de distancia, por ejemplo 0.5 si está a la mitad entre el centro y la derecha y -0.5 si está a la mitad entre el centro y la izquierda.

NOTA: entre más sensores se tienen, mejor es la detección y determinación de la variable involucrada.

Sensor		Distancia al obstáculo	
Distancia	IR, LIDAR o ultrasónico	0	No detecta nada a ningún lado
	IR, LIDAR o ultrasónico	0.5	Aproximándose a un objeto
	IR, LIDAR o ultrasónico	1	El objeto está a menos de 5 cm del vehículo, muy cerca
Posición	IR, LIDAR o ultrasónico	-1	Objeto detectado a la izquierda
	IR, LIDAR o ultrasónico	0	Objeto detectado al centro
	IR, LIDAR o ultrasónico	1	Objeto detectado a la derecha

Tabla 1: Entrada sensores de distancia

En cuanto a los sensores IR Line Tracking, idealmente se activará uno cuando el vehículo se ubique sobre la línea, cuando no esté sobre la línea, se activarán todos los sensores, ya que están sobre el piso blanco, altamente reflejante. Para que el vehículo se detenga en las estaciones se cargarán, se ha colocado a lo largo de la línea, líneas horizontales del mismo color que la línea principal, para que todos los sensores se desactiven, lo que le indica a la red neuronal que debe detenerse para cargar material. Esta lógica se muestra en la tabla 2.

Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Entrada
0	0	0	0	0	Fuera de la línea
1	0	0	0	0	En la línea, sensor 1 activado
0	1	0	0	0	En la línea, sensor 2 activado
0	0	1	0	0	En la línea, sensor 3 activado
0	0	0	1	0	En la línea, sensor 4 activado
0	0	0	0	1	En la línea, sensor 5 activado
1	1	1	1	1	Estación de carga, todos activados

Tabla 2: Entrada sensores IR Line tracking

Las salidas del sistema son la velocidad de los motores, es acá donde se aplicará lógica difusa o control borroso. En el control de un motor, se puede activar (ON) para que gire, y se puede desactivar para que pare (OFF), bajo esta definición, para girar a la izquierda, paramos el motor de la llanta izquierda y activamos el motor de la llanta derecha, y de igual forma, para girar a la derecha, el motor derecho pasa a OFF y el izquierdo a ON. Para avanzar recto, ponemos ambos motores a ON. El problema de este planteamiento es que el giro lo haría abruptamente, ya sea a izquierda o derecha.

En lógica difusa, podemos tener más opciones, por ejemplo, para girar suavemente a la derecha, hacemos que la llanta izquierda gire al 100% (1) y la llanta derecha gire al 90% (0.9), para girar rápidamente a la izquierda, la llanta derecha gira al 100% y la izquierda al 50%, por ejemplo. Este control sobre los motores DC, se logra mediante PWM[5], el cual ajusta la energía que se entrega a cada motor mediante la modulación del ancho de pulso entre los terminales del motor[6]. En la tabla 3 se muestra como sería el control difuso para el manejo de los motores.

Tabla 3: Control difuso de los motores, p. ej. 0.7 es al 70% de la potencia aplicada al motor.

Izquierdo				Derecho		
Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Motor izquierdo	MotorDerecho
0	0	0	0	0	0	0
1	0	0	0	0	0.7	1
0	1	0	0	0	0.85	1
0	0	1	0	0	1	1
0	0	0	1	0	1	0.85
0	0	0	0	1	1	0.7
1	1	1	1	1	0	0

El diseño de la red neuronal se muestra a continuación:

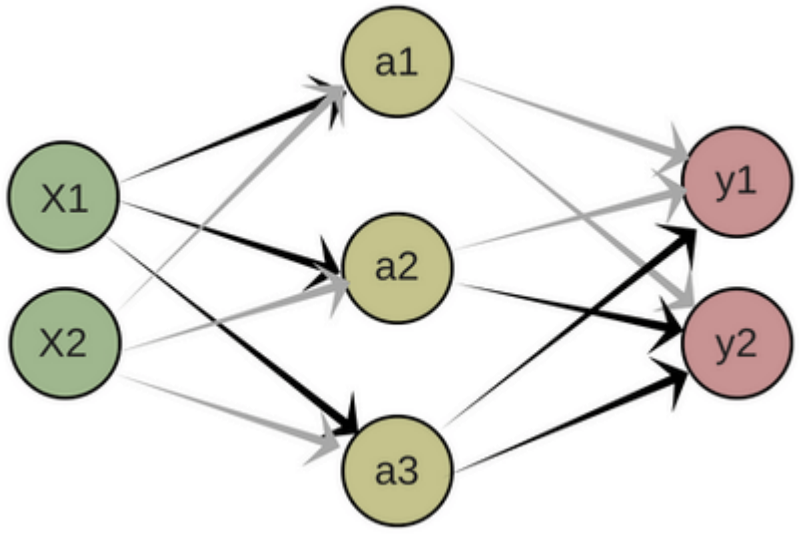


Fig. 38: Diseño de red neuronal para carro autónomo

La notación es la siguiente:

X1, X2: son las entradas de los sensores distancia y de sensores de línea

A1, A2 y A3: activación de la capa 2

Y1, Y2: son las salidas que alimentan a los motores.

2. Algoritmos y programas para el desarrollo del sistema

En esencia, el vehículo debe seguir una línea, para ello se ubica sobre la línea y luego, si la línea se pierde por el sensor derecho, el carro gira a la izquierda, y si la línea se pierde por el sensor izquierdo, el carro gira a la derecha.[7]

Esto se muestra en el Fig. 38.

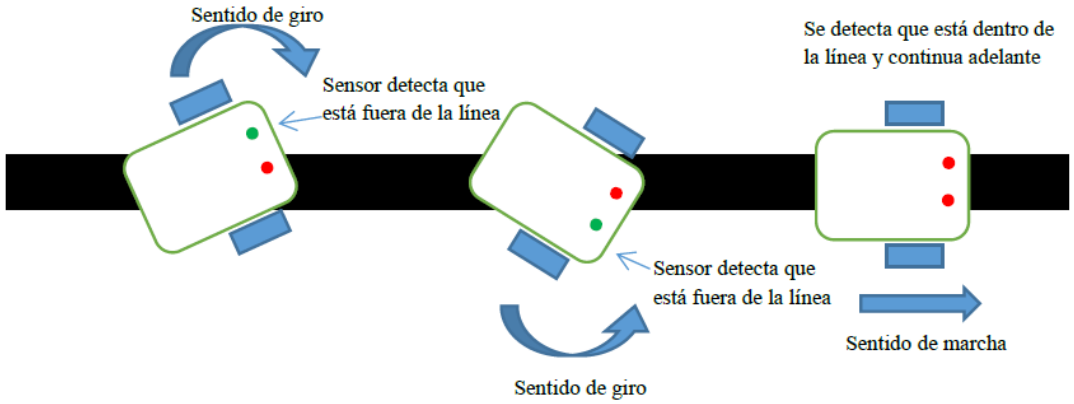


Fig. 38: Funcionamiento del seguidor de línea.

Al aplicar la lógica difusa, hay varios niveles para girar a la izquierda o derecha, depende del sensor que se active, para mantenerse siempre sobre el sensor central en el vehículo.

Declaraciones de los pesos para cada enlace entre neurona y neurona.

- O(j): Los pesos de las conexiones entre neuronas será una matriz que mapea la capa j a la j+1
- Es necesario utilizar una neurona extra en la capa 1 y una neurona extra en la capa 2 a modo de Bias, para mejorar la precisión de la red neuronal, y permitiendo una mayor libertad en el plano cartesiano.

Los cálculos para obtener los valores de activación serán (O^{T1} es la matriz transpuesta de los pesos de las conexiones):

$a(1) = g(O^{T1} \cdot X)$

$a(2) = g(O^{T2} \cdot X)$

$a(3) = g(O^{T3} \cdot X)$

En las ecuaciones, la g es una función Sigmoidea que refiere al caso especial de función logística y definida por la fórmula:

$g(z) = 1/(1+e^{-z})$

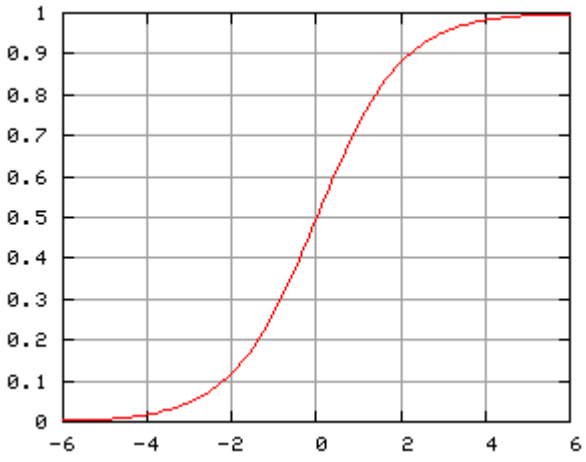


Fig. 39: Función sigmoidea

Feedforward es el recorrido de "izquierda a derecha" que hace el algoritmo de la red, para calcular el valor de activación de las neuronas desde las entradas hasta obtener los valores de salida.

Si se usa notación matricial, las ecuaciones para obtener las salidas de la red serán:

$X = [x_0 \quad x_1 \quad x_2]$

$z^{layer2} = O_1 X$

$a^{layer2} = g(z^{layer2})$

$z^{layer3} = O_2 a^{layer2}$

$y = g(z^{layer3})$

Resumiendo: se tiene una red; se tienen 2 entradas, éstas se multiplican por los pesos de las conexiones y cada neurona en la capa oculta suma esos productos y les aplica la función de activación para "emitir" un resultado a la siguiente conexión. Los pesos iniciales se asignan con valores entre -1 y 1 de manera aleatoria. El desafío de este algoritmo, será que las neuronas aprendan por sí mismas a ajustar el valor de los pesos para obtener las salidas correctas.

Backpropagation (cómputo del gradiente)

Al hacer backpropagation es donde el algoritmo itera para aprender. Esta vez, se itera de "derecha a izquierda" en la red para mejorar la precisión de las predicciones. El algoritmo de backpropagation se divide en dos Fases: Propagar y Actualizar Pesos.

Fase 1: Propagar

Esta fase implica 2 pasos:

- Hacer forward propagation de un patrón de entrenamiento (recordemos que es este es un algoritmo supervisado, y conocemos las salidas) para generar las activaciones de salida de la red.
- Hacer backward propagation de las salidas (activación obtenida) por la red neuronal usando las salidas "y" reales para generar los Deltas (error) de todas las neuronas de salida y de las neuronas de la capa oculta.

Fase 2: Actualizar Pesos:

Para cada <<sinapsis>> de los pesos:

- Multiplicar su delta de salida por su activación de entrada para obtener el gradiente del peso.
- Substraer un porcentaje del gradiente de ese peso

El porcentaje que utilizaremos en el paso 2.2 tiene gran influencia en la velocidad y calidad del aprendizaje del algoritmo y es llamado "learning rate" o tasa de aprendizaje. Si es una tasa muy grande, el algoritmo aprende más rápido, pero tendremos mayor imprecisión en el resultado. Si es demasiado pequeño, tardará mucho y podría no finalizar nunca.

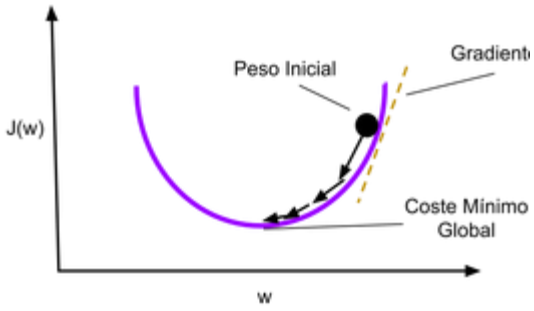


Fig. 40: Minimización del gradiente

En esta gráfica, se observa cómo se utiliza el gradiente paso a paso para descender y minimizar el coste total. Cada paso utilizará la Tasa de Aprendizaje -learning rate- que afectará la velocidad y calidad de la red. Se deben repetir las fases 1 y 2 hasta que el rendimiento de la red neuronal sea satisfactorio.

Si se denota el error en el layer "l" como d(l), para las neuronas de salida en layer 3 la activación menos el valor actual será:

$d(3) = a^{layer3} - y$

$d(2) = O^{T}_{2}\, d(3) \,.\, g'(z^{layer2})$

$g'(z^{layer2}) = a^{layer2} \,.\, (1 - a^{layer2})$

El valor del costo -que es lo que se desea minimizar - de la red será:

$J = a^{layer}\, d^{layer + 1}$

Se usa este valor y se multiplica por el learning rate antes de ajustar los pesos. Esto asegura que se busca el gradiente, iteración a iteración "apuntando" hacia el mínimo global.

El código completo se puede observar en el anexo I.

3. **Vehículo autónomo para ambientes cerrados, reduciendo los costos y aumentando la seguridad en el transporte**.

   Se construyó un vehículo autónomamente guiado, y equipó con sensores LIDAR, infrarrojos, ultrasónicos y webcam, que son controlados por un par microcontroladores Arduino y dos computadoras Raspberry, así como un computador externo Core i7.

   En la Fig. 41 se observa el carro ensamblado.

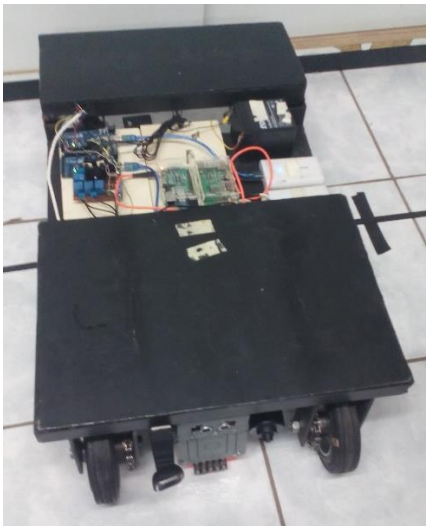


Fig. 41 Vehículo AGV totalmente equipado

Como fuente de energía se cuenta con tres baterías, la del vehículo es la más pequeña que le da una autonomía de 45 min., luego las otras le dan autonomía de 1 hora y 2 horas de acuerdo al tamaño.

Fig. 41 Sistema de alimentación del vehículo

Las principales características del vehículo se detallan a continuación:

Motor:	2 Motores DC
Alimentación:	12 V
Potencia nominal:	96 W cada uno
Peso:	1.5 Kg
Tracción:	Delantera
Driver:	Electrónico con PWM
Dimensiones:	82 cm X 50 cm X 15 cm
Carga máxima:	30 Kg

FUNCIONAMIENTO

El vehículo recorrerá una línea negra con un ancho de 1.5 cm, sobre un fondo blanco, si es necesario, el fondo blanco se puede pintar con un grosor de 15 cm. La línea cubrirá todos los puestos de trabajo donde se desea que llegue el vehículo.

En cada puesto, la línea tiene un ensanchamiento de aproximadamente 10 cm, que le indica al vehículo que se detenga, y activa una alarma que indica que debe cargarse, si no hay nada que carga se presiona el botón de AVANCE en el vehículo y este sigue hasta la próxima estación.

Al principio hay una fase de aprendizaje, hasta que la red neuronal converja en los pesos de las conexiones, y el desempeño sea máximo. Convergerá cuando el carro se mantenga en la línea, puede ser de 15 minutos a algunas horas el proceso de aprendizaje, en la prueba realizada lo hizo en un tiempo de 45 minutos, pero era de 8 m.

8. CONCLUSIONES

- Se ha diseñado el vehículo autónomo cumpliendo las especificaciones iniciales, con un control basado en red neuronal, que utiliza control difuso para manipular sus motores, y logra suavidad en el movimiento, evitando movimientos bruscos, y es una herramienta muy útil, que promueve la Industria 4.0, al automatizar otra tarea más dentro del ámbito del vehículo.
- El aprendizaje profundo requiere tiempo, y en el caso del AGV diseñado, eso significa múltiples recargas de los sistemas de baterías, por tal motivo se modificó el programa para hacer menos iteraciones y que logre un rendimiento muy bueno.
- Al ser utilizado para los campos de aplicación propuestos, industria o comercio, aumenta los índices de desempeño de los usuarios, ya que la materia prima está disponible en su área cercana de trabajo de forma automática, sin necesidad de invertir tiempo en tal tarea. Desde ese punto de vista, los posibles compradores del vehículo, obtendrían un beneficio al utilizar el vehículo autónomo en sus empresas.
- Como autores, se espera que este documento sea utilizado por otros investigadores como base para sus futuras investigaciones acerca de vehículos autónomos.

9. RECOMENDACIONES

- Una mejora a futuro para el vehículo autónomo es dotarlo de un sistema de amortiguamiento en las llantas tractoras, para mantenerlo siempre en contacto de la superficie del piso, a pesar de las irregularidades que tenga el mismo; esto, porque debido a la rigidez de la estructura, cuando el piso no está totalmente a nivel, alguna de las llantas tractoras puede quedar suspendida en el aire, dificultando el movimiento deseado en el vehículo.
- La manipulación del prototipo fue extremadamente dificultosa por el peso del vehículo, una mejora podría ser elaborar un chasis de aluminio o con chapa más delgada, manteniendo las especificaciones del mismo, lo cual le daría más autonomía, ya que las baterías moverían menor peso.
- Se recomienda, seguir investigando en el área, para desarrollar el máximo potencial de los sensores LIDAR, sea para esta aplicación u otras nuevas, ya que, al disminuir el peso de su importancia, se relegó a un simple sensor de distancia, que se puede ver como una subutilización del mismo.

- No se recomienda exceder la carga que puede soportar el vehículo, porque eso puede deformarlo mecánicamente, provocando su mal funcionamiento.
- Investigar y proponer algoritmos y software para enlazar con otros vehículos, para que dos o más vehículos, trabajen en conjunto en su área de aplicación.

10. GLOSARIO

Algoritmo

Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problema. En un contexto informático, se puede definir como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas; o como un conjunto de instrucciones que, realizadas en orden, conducen a obtener la solución de un problema.

Algoritmo genético

Los Algoritmos Genéticos (AG) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Como métodos de búsqueda, imitan la teoría de la evolución biológica de Darwin para la resolución de problemas. Así, partiendo de una población inicial, se seleccionan los individuos más capacitados, para luego reproducirlos y mutarlos, con el fin de obtener la siguiente generación de individuos, que estarán más adaptados que la anterior.

La evolución de las soluciones hacia valores óptimos del problema dependerá en buena medida de una adecuada codificación de las mismas.

Aprendizaje automático

El aprendizaje automático es una rama dentro de la inteligencia artificial, concretamente: “el campo de estudio que proporciona a las computadoras la capacidad de aprender sin haber sido explícitamente programadas para ello” (Arthur Samuel).

Una definición más actual y precisa podría ser la de Tom Mitchell, según la cual:

“Se dice que un programa informático aprende de una experiencia E con respecto a alguna clase de tarea T y una medición del rendimiento P, si su rendimiento en las tareas T, medido como P, mejora con la experiencia E”.

Hay dos tipos de aprendizaje automático: supervisado y no supervisado (ver definiciones).

Aprendizaje no supervisado

Es un tipo de aprendizaje automático en el que el algoritmo no recibe información sobre cómo deben ser los datos de salida. Simplemente se le proporciona como entrada un conjunto de datos no estructurados, en el que él deberá identificar los posibles patrones y relaciones existentes entre ellos, para descubrir, por sí solo, una estructura. En cierto modo, es similar a la forma en la que los seres humanos aprenden de forma natural. Es ideal para resolver problemas en los que desconocemos total o parcialmente cómo debería ser el resultado.

Los algoritmos de agrupación (clustering.) se encuadran dentro del aprendizaje no supervisado.

Aprendizaje por refuerzo (reinforcement learning o RL)

Área del aprendizaje automático inspirada en la psicología conductista, en la que un agente aprende de sus errores y aciertos en la interacción con un entorno, por medio de un sistema de recompensas y castigos.

El sistema es similar al del entrenamiento de las mascotas en el mundo real, en el que si el animal hace algo bien se le recompensa de algún modo y si hace algo mal se le castiga.

En el caso de la inteligencia artificial, se puede premiar al agente con un valor acumulativo positivo en caso de acierto y se le castiga con un valor negativo que le resta puntuación en caso de error, de manera que el agente tratará de desarrollar las estrategias más adecuadas para obtener la mayor recompensa a largo plazo. Se encuadra dentro del aprendizaje supervisado, dado que el proceso de aprendizaje requiere de intervención externa, pero con la diferencia de que no se proporciona al agente la solución o respuesta acertada, sino simplemente se le indica si ha acertado o no, por medio del sistema de recompensa / castigo.

El RL fue uno de los principales tipos de aprendizaje automático utilizados en AlphaGo, de DeepMind, el sistema de IA de Google que venció al campeón del mundo de Go, Lee Sedol, en su disciplina.

Aprendizaje profundo (Deep Learning)

Se incluye también dentro del aprendizaje automático, pero en este caso se trataría de un tipo de aprendizaje no supervisado (sin intervención humana). El aprendizaje profundo trata, en cierto modo, de imitar el funcionamiento del sistema nervioso humano. Para ello, utiliza lo que se conoce como redes neuronales o capas de unidades de procesamiento (neuronas artificiales) que se especializan en identificar características o patrones determinados en objetos o conjuntos de datos no estructurados, sin necesidad de un entrenamiento previo con un conjunto de datos estructurados o etiquetados. Cuando se superponen varias redes neuronales, de manera que la salida de una alimenta la entrada de otra, se denominan redes neuronales profundas.

Aprendizaje supervisado

En este tipo de aprendizaje automático el algoritmo sí recibe previamente información sobre las relaciones existentes entre los datos de entrada y salida y sobre cómo deben ser estos últimos. Para ello, se le proporciona como entrada un conjunto de datos estructurado, en el que la información ha sido organizada y etiquetada. Por ejemplo, si el objetivo es crear un algoritmo para identificar gatos en imágenes, se entrenaría previamente al algoritmo con miles de imágenes de gatos etiquetados como “gato”.

Se encuadran dentro del aprendizaje supervisado los algoritmos de clasificación y regresión, los árboles de decisión o los bosques aleatorios.

El principal problema de este tipo de aprendizaje automático es la gran cantidad de tiempo y trabajo previo que requiere a menudo la preparación de esos datos estructurados.

Autonomía computacional

Capacidad de un sistema para la autogestión adaptativa de sus propios recursos para funciones informáticas de alto nivel sin la intervención del usuario.

Ciencia de datos

Campo interdisciplinario que combina sistemas, procesos y métodos científicos para extraer conocimiento o un mejor entendimiento de los datos en sus diferentes formas, ya sea estructurados o no estructurados.

Utiliza técnicas de disciplinas muy diversas, como las matemáticas, la estadística y el análisis de datos, las ciencias de la información y de la computación, etc.; para analizar y entender fenómenos reales a partir de los datos disponibles sobre ellos.

Entrenamiento

Proceso mediante el cual se forma a un algoritmo con un conjunto de datos.

Ingeniería del conocimiento

Se centra en la construcción de sistemas basados en el conocimiento, incluyendo todos los aspectos científicos, técnicos y sociales de los mismos.

Inteligencia Artificial (IA)

Campo de las Ciencias de la Computación que estudia el desarrollo de un comportamiento aparentemente inteligente en agentes o dispositivos que perciben su entorno y llevan a cabo acciones que maximizan sus oportunidades de éxito en la consecución de algún objetivo.

Inteligencia General Artificial (AGI)

También denominada "Inteligencia artificial fuerte", es aquella que iguala o excede la inteligencia humana promedio. Por el momento es más un objetivo o aspiración que una realidad.

Modelo

Un modelo es un algoritmo de aprendizaje automático que construye su propia comprensión de un tema, o su propio "modelo" del mundo.

Procesamiento del lenguaje natural

Software para entender la intención y las relaciones de las ideas dentro del lenguaje.

Redes neuronales artificiales

Algoritmos construidos para modelar la forma en que el cerebro procesa la información a través de redes de ecuaciones matemáticas conectadas. Los datos dados a una red neuronal se dividen en partes más pequeñas y se analizan por patrones subyacentes miles o millones de veces dependiendo de la complejidad de la red.

Red neuronal profunda

Red neuronal que está formada por redes neuronales superpuestas, de manera que la salida de una red alimenta la entrada de otra. Normalmente, las capas de una red neuronal profunda analizan los datos en niveles de abstracción cada vez más altos, lo que significa que cada una arrojaría datos innecesarios hasta que quedara la representación más simple y precisa de los datos.

Red neuronal convolucional (CNN)

Red neuronal utilizada principalmente para reconocer y comprender imágenes, vídeo y datos de audio, gracias a su capacidad para trabajar con datos densos como millones de píxeles de una imagen o miles de muestras de un archivo de audio.

Red neuronal recurrente (RNN)

Una red neuronal popular para el procesamiento del lenguaje natural que analiza los datos de forma cíclica y secuencial, lo que significa que puede procesar datos como palabras u oraciones, manteniendo su orden y contexto en una oración.

Red de memoria de largo plazo (LSTM)

Una variación de la red neuronal recurrente que pretende retener información estructurada basada en datos. Por ejemplo, una RNN podría reconocer todos los sustantivos y adjetivos en una oración y si se usan correctamente, pero un LSTM podría recordar la trama de un libro.

Red generativa antagónica

Sistema de dos redes neuronales, una que genera una salida y otra que comprueba la calidad de esa salida con respecto a lo que debería ser. Por ejemplo, al tratar de generar una imagen de una manzana, el generador hará una imagen, y el otro (llamado discriminador) hará que el generador vuelva a intentarlo si no puede reconocer una manzana en la imagen.

Sistemas bioinspirados

Son sistemas que están basados en el comportamiento y la forma de actuar de ciertos sistemas biológicos, su principio consiste en resolver problemas observando animales o sistemas que llevan siglos evolucionando.

Estos sistemas han tomado un impuso dentro de la IA (Inteligencia Artificial) debido a que son capaces de minimizar el tiempo de computación de ciertos problemas matemáticos complejos tales como el problema del Viajante (Travelling Salesman Problem) o el de la cena de Filósofos.

Técnicas de búsqueda heurística

Técnicas de búsqueda informada que aprovechan el conocimiento del dominio para encontrar resultados. Deben disponer de alguna información sobre la proximidad de

cada estado a un estado objetivo, lo que les permite explorar en primer lugar los caminos más prometedores. Su uso es ideal en la solución de problemas difíciles para los que una búsqueda exhaustiva necesitaría demasiado tiempo.

Visión por computadora

También conocida como visión computarizada o visión artificial. Área de investigación dentro de la IA que explora el reconocimiento y comprensión de imágenes y vídeo. Se utiliza, por ejemplo, en los vehículos autónomos, la búsqueda de imágenes de Google, el etiquetado automático en Facebook, etc.

11. REFERENCIAS BIBLIOGRÁFICAS

[1] E. Rivera, «Introducción a las Redes Neuronales Artificiales.», *Editorial de Universidad Don Bosco*, 2015.

[2] N. Muñoz-Galeano, J. B. Cano-Quintero, y J. M. López-Lezama, «Enseñando el Funcionamiento de los Inversores Puente H: Análisis del Intercambio de Potencia entre Bobinas y Condensadores», *Form. Univ.*, vol. 9, pp. 117-124, 2016.

[3] F. Scotti *et al.*, «Dual use architecture for innovative LIDAR and free space optical communications.», *Appl. Opt.*, vol. 56, n.º 31, pp. 8811–8815, 2017.

[4] D. Ortiz Martínez, «Robótica para seguimiento de líneas», Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona - Enginyeria Electrònica, Barcelona, 2016.

[5] L. Petru y G. Mazen, «PWM Control of a DC Motor Used to Drive a Conveyor Belt», *Procedia Eng.*, vol. 100, pp. 299-304, 2015.

[6] J.-D. Warren, J. Adams, y H. Molle, «Arduino for robotics», en *Arduino robotics*, Springer, 2011, pp. 51–82.

[7] G. Ramírez, «Método de aprendizaje simple para navegación de minirobots móviles rodantes», *Dyna*, vol. 70, n.º 138, pp. 59-66, 2003.

12. ANEXOS

12.1. CÓDIGO FINAL EN ARDUINO PARA EL CONTROL DE LOS MOTORES

```
/*
 IMPORTANTE:
 Para realizar una detención de carga y/o descarga de materiales hay que colocar una linea
 de tape perpendicular, a la línea de seguimmiento, la linea perpendicular debe abarcar a los 5 sensores
 */
byte full_sensor;
int i1 = 5;  //Salida para motor1 en pin 5
int i2 = 6;  //Salida para motor1 en pin 6
int i3 = 10; //Salida para motor2 en pin 10
int i4 = 11; //Salida para motor2 en pin 11
int pwm_izq = 7;  //salida de pwm para motor1 en pin 7
int pwm_der = 8;  //salida de pwm para motor1 en pin 8
int buzzer = 9;   //conexión del buzzer en pin 9
int timeAvance = 100;
int V_derecho = 255;
int V_izquierdo = 255;
int s1 = 22;  //sensor infrarrojo del extremo izquierdo en pin 22
int s2 = 24;  //sensor infrarrojo izquierdo (a un costado del central) en pin 24
int s3 = 26;  //sensor infrarrojo central en pin 26
int s4 = 28;  //sensor infrarrojo derecho (a un costado del central) en pin 28
int s5 = 30;  //sensor infrarrojo del extremo derecho en pin 30
int botonr = 32; //Botón para reiniciar recorrido después de un alto de carga o descarga de materiales, en pin 32
bool sensor1;
bool sensor2;
bool sensor3;
bool sensor4;
bool sensor5;
bool inicio;
int V_min = 140; // 70% V_max
int V_max = 170; // 100%velocidad
int delta_v; // delta_v= (V_max - V_min)/3
void setup() {
//  Serial1.begin(9600);
 Serial.begin(9600);
 pinMode(i1, OUTPUT);
 pinMode(i2, OUTPUT);
 pinMode(i3, OUTPUT);
 pinMode(i4, OUTPUT);
 pinMode(s1, INPUT);
 pinMode(s2, INPUT);
 pinMode(s3, INPUT);
 pinMode(s4, INPUT);
 pinMode(s5, INPUT);
 pinMode(buzzer, OUTPUT);
 //tone(9, 3000, 2000);
 //noTone();
 }
void loop() {
 //V_derecho = constrain(V_derecho, 125, 255);
 V_izquierdo = V_derecho * 0.95;
 delta_v= V_max - V_min;
 //forward ();
 //delay(4000);
//  analogWrite(pwm_izq,V_izquierdo);
//  analogWrite(pwm_der,V_derecho);
   sensor1 = digitalRead(s1);
   sensor2 = digitalRead(s2);
   sensor3 = digitalRead(s3);
   sensor4 = digitalRead(s4);
   sensor5 = digitalRead(s5);
   // se convierte la información de los 5 sensores en un byte de 0-31
   full_sensor = 31 - (byte(sensor1)*16 + byte(sensor2)*8 + byte(sensor3)*4 + byte(sensor4)*2+ byte(sensor5));
   inicio = digitalRead(botonr);
   if ((inicio == 0) and (full_sensor != 31)) {start ();}
```

```
/*   Serial.print(sensor1);
  Serial.print(" ");_
  Serial.print(sensor2);
  Serial.print(" ");
  Serial.print(sensor3);
  Serial.print(" ");
  Serial.print(sensor4);
  Serial.print(" ");
  Serial.print(sensor5);
  Serial.print("      ");
 Serial.println(full_sensor);
  Serial.println(inicio);

 /* ************************************
   if ((BotonMenu/4) == 0){
  MenuAdmin ++;
} */

 switch (full_sensor){ // casos de menu
  case 0:
   accion_a ();
   break;
  case 1:
   accion_h ();
   break;
  case 2:
   accion_g ();
   break;
  case 3:
   accion_g ();
   break;
  case 4:
   accion_e ();
   break;
  case 5:
   accion_d ();
   break;
  case 6:
   accion_f ();
   break;
  case 7:
   accion_f ();
   break;
  case 8:
   accion_d ();
   break;
  case 9:
   accion_d ();
   break;
  case 10:
   accion_e ();
   break;
  case 11:
   accion_f ();
   break;
  case 12:
   accion_d ();
   break;
  case 13:
   accion_f ();
   break;
  case 14:
   accion_e ();
   break;
  case 15:
   accion_f ();
   break;
  case 16:
   accion_b ();
   break;
  case 17:
   accion_d ();
   break;
  case 18:
   accion_d ();
   break;
  case 19:
   accion_d ();
   break;
  case 20:
```

```
 accion_d ();
 break;
case 21:
 accion_d ();
 break;
case 22:
 accion_d ();
 break;
case 23:
 accion_d ();
 break;
case 24:
 accion_c ();
 break;
case 25:
 accion_d ();
 break;
case 26:
 accion_d ();
 break;
case 27:
 accion_d ();
 break;
case 28:
 accion_d ();
 break;
case 29:
 accion_d ();
 break;
case 30:
 accion_d ();
 break;
case 31:
 accion_a ();
 break;
default :
 // para salir
  stop();
  break;
}
}

void start()
{
//  Activa los motores para avanzar
//  analogWrite(pwm_izq,V_izquierdo);
//  analogWrite(pwm_der,V_derecho);
  digitalWrite(i1,HIGH);
  digitalWrite(i2,LOW);
  digitalWrite(i3,HIGH);
  digitalWrite(i4,LOW);
  Serial.println("Activados");
  analogWrite(pwm_izq,V_min);
  analogWrite(pwm_der,V_min);
}
void stop()
{
//  Desactiva los motores, no avanza
  //Serial.println(V_derecho);
  analogWrite(pwm_izq,0);
  analogWrite(pwm_der,0);
  digitalWrite(i1,LOW);
  digitalWrite(i2,LOW);
  digitalWrite(i3,LOW);
  digitalWrite(i4,LOW);
  Serial.println("Stop");
}

void accion_a() // En la estacíon de carga o cuando los
sensores no hayan la línea
{
        stop();
        while(inicio == LOW){
          tone(buzzer, 3000);
          delay(200);
          noTone(buzzer);
          delay(800);
          inicio = digitalRead(botonr);
          }
```

```
        V_derecho = V_max;
        V_izquierdo = 0.95 * V_max;
        accion_e();
        start ();
        delay(500);
        Serial.println("En estación o fuera de la línea");
   }

void accion_b() // giro brusco a la derecha

{

 analogWrite(pwm_izq,0.95*V_max);
 analogWrite(pwm_der,V_min);
 Serial.println("Giro brusco derecha");
  }
void accion_c() // giro mediano  a la derecha
{
 analogWrite(pwm_izq,V_max*0.95);
 analogWrite(pwm_der,V_min+delta_v);
 Serial.println("Giro mediano derecha");
 }

void accion_d() // giro leve a la derecha
{
 analogWrite(pwm_izq,V_max*0.95);
 analogWrite(pwm_der,V_max - delta_v);
 Serial.println("Giro leve derecha");
  }
void accion_e() // maxima velocidad
{
 analogWrite(pwm_izq,0.95*V_max);
 analogWrite(pwm_der,V_max);
 Serial.println("Recto maxima velocidad");
  }
void accion_f() // giro leve a la izquierda

{
 analogWrite(pwm_der,V_max);
 analogWrite(pwm_izq,V_max - delta_v);
 Serial.println("Giro leve izquierda");
  }
void accion_g() // giro mediano  a la izquierda

{
 analogWrite(pwm_der,V_max);
 analogWrite(pwm_izq,V_min+delta_v);
 Serial.println("Giro mediano izq");
 }

void accion_h() // giro brusco a la izq

{
 analogWrite(pwm_der,V_max);
 analogWrite(pwm_izq,V_min);
 Serial.println("Giro brusco izq");
  }

void turn_left()
{
//  Serial.println("GiroIzquierda...");
 analogWrite(pwm_izq,V_izquierdo);
 analogWrite(pwm_der,V_derecho);
 start();
 Serial.println("izquierda");
}
void turn_right()
{
//  Serial.println("GiroDerecha...");
 analogWrite(pwm_izq,V_izquierdo);
 analogWrite(pwm_der,V_derecho);
 start();
 Serial.println("derecha");
}
```

12.2. DECLARACIÓN DE CLASE RED NEURAL (EN PYTHON)

```
1  import numpy as np
2
3  def sigmoid(x):
4      return 1.0/(1.0 + np.exp(-x))
5
6  def sigmoid_derivada(x):
7      return sigmoid(x)*(1.0-sigmoid(x))
8
9  def tanh(x):
10     return np.tanh(x)
11
12 def tanh_derivada(x):
13     return 1.0 - x**2
14
15
16 class NeuralNetwork:
17
18     def __init__(self, layers, activation='tanh'):
19         if activation == 'sigmoid':
20             self.activation = sigmoid
21             self.activation_prime = sigmoid_derivada
22         elif activation == 'tanh':
23             self.activation = tanh
24             self.activation_prime = tanh_derivada
25
26         # inicializo los pesos
27         self.weights = []
28         self.deltas = []
29         # capas = [2,3,2]
30         # rando de pesos varia entre (-1,1)
31         # asigno valores aleatorios a capa de entrada y capa oculta
32         for i in range(1, len(layers) - 1):
33             r = 2*np.random.random((layers[i-1] + 1, layers[i] + 1)) -1
34             self.weights.append(r)
35         # asigno aleatorios a capa de salida
36         r = 2*np.random.random( (layers[i] + 1, layers[i+1])) - 1
37         self.weights.append(r)
38
39     def fit(self, X, y, learning_rate=0.2, epochs=100000):
40         # Agrego columna de unos a las entradas X
```

```
41      # Con esto agregamos la unidad de Bias a la capa de entrada
42      ones = np.atleast_2d(np.ones(X.shape[0]))
43      X = np.concatenate((ones.T, X), axis=1)
44
45      for k in range(epochs):
46          i = np.random.randint(X.shape[0])
47          a = [X[i]]
48
49          for l in range(len(self.weights)):
50                  dot_value = np.dot(a[l], self.weights[l])
51                  activation = self.activation(dot_value)
52                  a.append(activation)
53          # Calculo la diferencia en la capa de salida y el valor obtenido
54          error = y[i] - a[-1]
55          deltas = [error * self.activation_prime(a[-1])]
56
57          # Empezamos en el segundo layer hasta el ultimo
58          # (Una capa anterior a la de salida)
59          for l in range(len(a) - 2, 0, -1):
60              deltas.append(deltas[-1].dot(self.weights[l].T)*self.activation_prime(a[l]))
61          self.deltas.append(deltas)
62
63          # invertir
64          # [level3(output)->level2(hidden)]  => [level2(hidden)->level3(output)]
65          deltas.reverse()
66
67          # backpropagation
68          # 1. Multiplcar los delta de salida con las activaciones de entrada
69          #    para obtener el gradiente del peso.
70          # 2. actualizo el peso restandole un porcentaje del gradiente
71          for i in range(len(self.weights)):
72              layer = np.atleast_2d(a[i])
73              delta = np.atleast_2d(deltas[i])
74              self.weights[i] += learning_rate * layer.T.dot(delta)
75
76          if k % 10000 == 0: print('epochs:', k)
77
78  def predict(self, x):
79      ones = np.atleast_2d(np.ones(x.shape[0]))
80      a = np.concatenate((np.ones(1).T, np.array(x)), axis=0)
81      for l in range(0, len(self.weights)):
```

```
82            a = self.activation(np.dot(a, self.weights[l]))
83        return a
84
85    def print_weights(self):
86        print("LISTADO PESOS DE CONEXIONES")
87        for i in range(len(self.weights)):
88            print(self.weights[i])
89
90    def get_deltas(self):
91        return self.deltas
```

12.3. CÓDIGO DE FUNCIÓN QUE EVITA LOS OBSTÁCULOS (EN PYTHON)

```
1  # funcion Coche Evita obstáculos
2  nn = NeuralNetwork([2,3,2],activation ='tanh')
3  X = np.array([[0, 0],   # sin obstaculos
4                [0, 1],   # sin obstaculos
5                [0, -1],  # sin obstaculos
6                [0.5, 1], # obstaculo detectado a derecha
7                [0.5,-1], # obstaculo a izq
8                [1,1],    # demasiado cerca a derecha
9                [1,-1]])  # demasiado cerca a izq
10
11 y = np.array([[0,1],    # avanzar
12               [0,1],    # avanzar
13               [0,1],    # avanzar
14               [-1,1],   # giro izquierda
15               [1,1],    # giro derecha
16               [0,-1],   # retroceder
17               [0,-1]])  # retroceder
18 nn.fit(X, y, learning_rate=0.03,epochs=15001)
19
20 index=0
21 for e in X:
22     print("X:",e,"y:",y[index],"Network:",nn.predict(e))
23     index=index+1
```

IDENTIDAD INSTITUCIONAL

VISIÓN

Ser una institución educativa líder en educación tecnológica a nivel nacional y regional, comprometida con la calidad, la empresarialidad y la pertinencia de nuestra oferta educativa.

MISIÓN

Formar profesionales integrales y competentes en áreas tecnológicas que tengan demanda y oportunidad en el mercado local, regional y mundial, tanto como trabajadores y como empresarios.

VALORES

EXCELENCIA: *Nuestro diario quehacer está fundamentado en hacer bien las cosas desde la primera vez.*

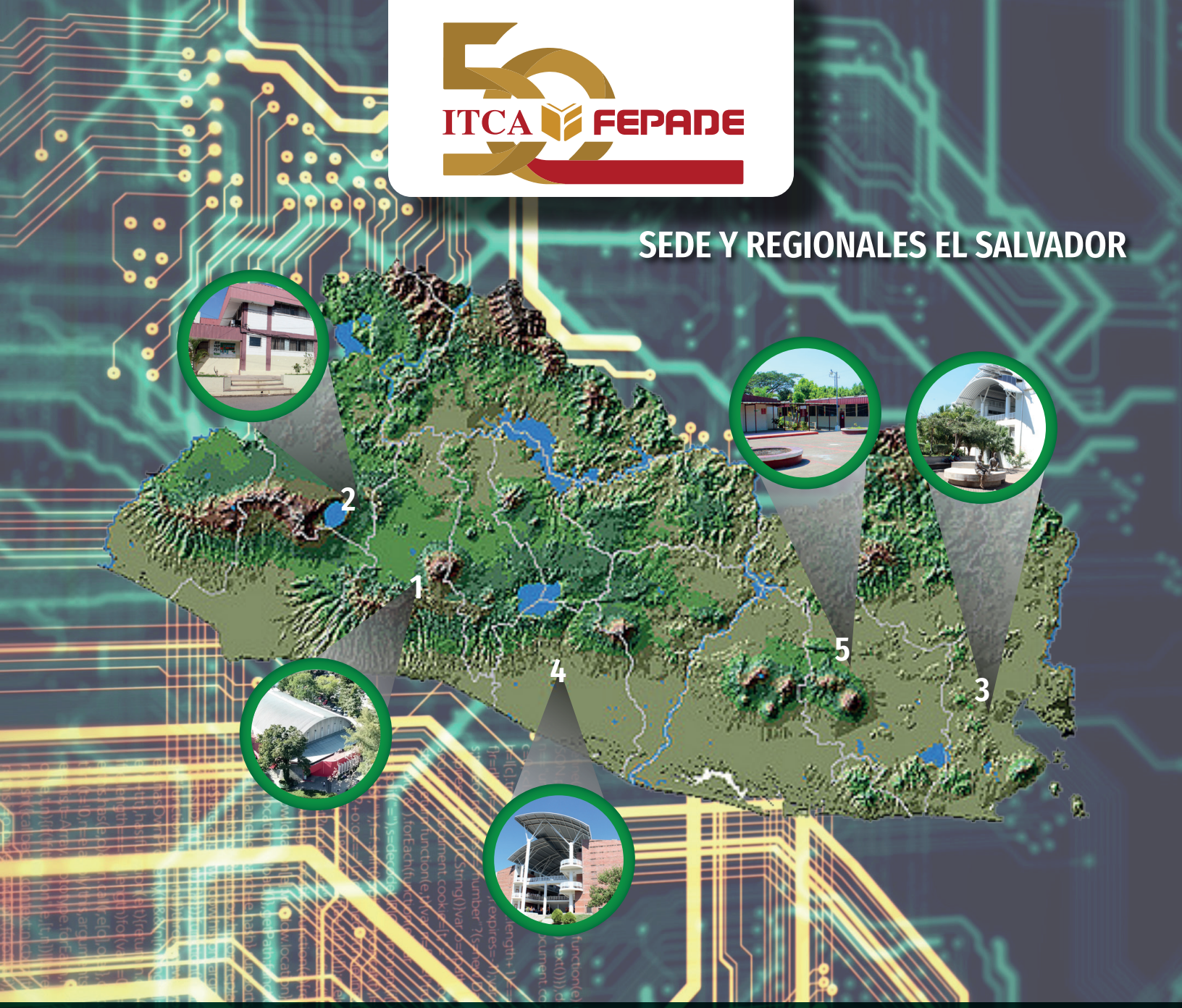
INTEGRIDAD: *Actuamos congruentemente con los principios de la verdad en todas las acciones que realizamos.*

ESPIRITUALIDAD: *Desarrollamos todas nuestras actividades en la filosofía de servicio, alegría, compromiso, confianza y respeto mutuo.*

COOPERACIÓN: *Actuamos basados en el buen trabajo en equipo, la buena disposición a ayudar a todas las personas.*

COMUNICACIÓN: *Respetamos las diferentes ideologías y opiniones, manteniendo y propiciando un acercamiento con todo el personal.*

SEDE Y REGIONALES EL SALVADOR



La Escuela Especializada en Ingeniería ITCA-FEPADE, fundada en 1969, es una institución estatal con administración privada, conformada actualmente por 5 campus: Sede Central Santa Tecla y cuatro Centros Regionales ubicados en Santa Ana, San Miguel, Zacatecoluca y La Unión.

1 SEDE CENTRAL SANTA TECLA
Km. 11.5 carretera a Santa Tecla, La libertad.
Tel.: (503) 2132-7400
Fax: (503) 2132-7599

2 CENTRO REGIONAL SANTA ANA
Final 10a. Av. Sur, Finca Procavia.
Tel.: (503) 2440-4348
Tel./Fax: (503) 2440-3183

3 CENTRO REGIONAL LA UNIÓN
Calle Sta. María, Col. Belén, atrás del Instituto Nacional de La Unión
Tel.: (503) 2668-4700

4 CENTRO REGIONAL ZACATECOLUCA
Km. 64.5, desvío Hacienda El Nilo sobre autopista a Zacatecoluca.
Tel.: (503) 2334-0763 y (503) 2334-0768

5 CENTRO REGIONAL SAN MIGUEL
Km. 140 carretera a Santa Rosa de Lima.
Tel.: (503) 2669-2298
Fax: (503) 2669-0061

www.itca.edu.sv