BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

COMPLEJO REGIONAL CENTRO - SAN JOSÉ CHIAPA

Desarrollo e implementación de un control de distancia en un carro eléctrico a escala mediante lógica difusa

T E S I NA

Que para obtener el título de

Licenciado en Ingeniería en Automatización y Autotrónica

P R E S E N T A

Yehoshua Bahena Nava

DIRECTOR DE TESINA

M.C. Miguel Ángel Ortega Palacios

Puebla, Pue., agosto 2022

Agradecimientos

Me gustaría comenzar agradeciendo primero a la Benemérita Universidad Autónoma de Puebla, que me permitió ser parte de la comunidad y me brindo siempre un apoyo constante en todos los aspectos. Me permitió ser un alumno más y tener la oportunidad de adquirir una educación de calidad con buenos resultados.

Agradezco a los profesores y directivos de la cede San José Chiapa, puesto que siempre me brindaron su apoyo y ayuda cada que la pedí.

A mis padres abuelos Bernardo Bahena Mendoza y María Luisa Nava Cruz que me apoyaron hasta el final en todos los sentidos en este viaje que inicie, así mismo a mis padres biológicos y a mis abuelos paternos quienes siempre me apoyaron y recordaron lo importante que es la educación y el hacer lo correcto en la vida.

A mis hermanos y amigos que me mostraron calidez en los momentos adecuados, y a mi novia que fue mi compañera en estos años de estudio.

Agradezco a mi asesor de tesis M.C. Miguel Ángel Ortega Palacios del cual sin su ayuda y conocimientos no habría sido posible este trabajo.

Finalmente agradezco a mis compañeros (roomies) y amigos que hice en este tiempo, esperando que el trabajo presentado sume información al tema y pueda ser de utilidad a los estudiantes y a todo aquel que lo lea.

Resumen

El siguiente proyecto retomó un controlador difuso de Distancia implementado en un carro eléctrico a escala, con la intención de arreglar y resolver las fallas presentadas, pero, sobre todo, mejorarlo en algunos aspectos con la intención de volverlo más autónomo.

El controlador fue desarrollado en una placa de programación Arduino UNO, haciendo uso del entorno de programación del mismo nombre. Se utilizaron componentes electrónicos como un sensor ultrasónico HC-SR04, una pantalla LCD de 16x2 con I2C, un módulo TB6612FNG entre otros. Todos con la finalidad de montar una planta para un sistema de control con retroalimentación.

Se establecieron cambios importantes, desde el prototipo hasta el programa principal, logrando buenos resultados. Además de implementar una conexión Bluetooth para el ingreso de datos externos al programa además de remplazar algunos componentes.

También se hicieron varias pruebas físicas, comparando diferentes valores para los conjuntos difusos de las variables de entrada y salida anotando las observaciones para poder contrastar los resultados.

Finalmente se obtuvo un prototipo funcional con más autonomía respecto al anterior, que resolvió muchos de los fallos anteriores.

Contenido

I. Lista de Figuras

II. Lista de Tablas

1. Introducción

El auge actual por los automóviles eléctricos es creciente en los países desarrollados, tratando de implementar sistemas más ecológicos y autónomos, ofreciendo lo que se considera el siguiente paso para la sociedad moderna. Para conseguir ese objetivo se ha trabajado con prototipos a escala o también, se han desarrollado robots móviles en los cuales se ponen en funcionamiento sistemas de control con diferentes fines, desde trayectorias programadas, hasta la propia conducción automática.

Con este contexto surgió el interés por desarrollar un controlador que pudiera regular la distancia en un carro eléctrico a escala (o, estrictamente hablando, un robot móvil), ya que un robot en movimiento representa muchos más retos y obstáculos a superar a comparación de uno estático, comprendiendo con esto que la tendencia de robots va más enfocada hacia los móviles y a aquellos que se enfrentan a los entornos y fenómenos físicos, por ejemplo, los populares drones.

Para este trabajo se optó por mejorar un controlador basado en lógica difusa (Control Difuso) que fue desarrollado previamente y, tomando en cuenta que no hay muchos trabajos con este concepto debido a que es reciente y no tan conocido, se plantearon varias mejoras para obtener un mejor prototipo, así como recabar datos de este.

El controlador se diseñó en el software Matlab, donde se simuló el funcionamiento con una variable de entrada “Error con respecto a la distancia”, y una variable de salida “Voltaje de salida” con tres conjuntos difusos para cada variable y, posteriormente, pudo ser adaptado al entorno de programación Arduino IDE.

En paralelo también se diseñó una sencilla aplicación móvil para celular con la finalidad de enviar el valor de la distancia deseada por el usuario, al programa.

1.1 Planteamiento del problema

La propuesta tiene la misión de retomar un proyecto desarrollado previamente, el cual consistía en un controlador difuso para un mini carro eléctrico a escala.

En este proyecto previo se alcanzó el objetivo general, el cual consistía en desarrollar un control difuso para un mini carro eléctrico, que permitiera mantener una posición determinada, de manera tal que cuando un obstáculo estuviese más cerca retrocediera, y cuando más lejos avanzara, con la intención de mantener la distancia deseada.

La motivación principal para continuar este proyecto es resolver complicaciones y problemas surgidos en este, puesto que aún existe un gran margen de mejora. El diseño del carro, la interacción y modificación del programa en tiempo real y el tiempo de respuesta son algunos de los apartados que pueden mejorarse.

1.2 Objetivos

Objetivo General.

Mejorar un controlador difuso de distancia en un carro eléctrico a escala, que permita mantener una distancia deseada, mediante lógica difusa.

Objetivos Específicos.

- Mejorar físicamente el diseño del prototipo del mini carro eléctrico, con la finalidad de superar las fallas que presenta.
- Desarrollar una aplicación móvil que permita al usuario modificar la distancia deseada.
- Disminuir el margen de error en las pruebas y aumentar la precisión.
- Desarrollar la etapa de Fusificación para las variables de entrada y salida.
- Proponer la base de reglas del control difuso.
- Emplear el método de Mamdani para definir la conclusión de las reglas.
- Emplear el método de centroide para la etapa de Defusificación.

- Analizar los resultados de las pruebas físicas.

1.3 Hipótesis

Es posible implementar y mejorar un controlador difuso de distancia en un carro eléctrico a escala.

1.4 Justificación

Actualmente la automatización esta dominando los avances en la tecnología de producción y con justa razón ya que los beneficios suponen un amplio margen de mejora en la calidad final del producto. Con esto tenemos que el control o los controladores representan una de las partes más importantes de la automatización, junto con otras herramientas como la Inteligencia Artificial. Esta es una de las razones por las que tener el conocimiento sobre sistemas de control es importante.

Se escogió el control difuso debido a la versatilidad y facilidad para ajustarlo a diferentes situaciones y sistemas, puesto que permite adaptarlo de acuerdo con su o sus variables de entrada. Esto representa un menor tiempo en cuanto a su montaje, a comparación de un PID en el cual si las condiciones externas se modifican radicalmente se tiene que volver a modelar.

Para el proyecto se decidió controlar la distancia en un carro eléctrico a escala debido a que en la industria existen nuevos dispositivos y robots móviles que han tenido buenos resultados transportando material con rutas programadas en la industria, desde materia prima, piezas, herramientas, hasta productos terminados. Estos han representado un avance en la autonomía de un robot móvil por lo que desarrollar controladores permite tener más conocimiento enfocado a la automatización.

Al retomar este proyecto se tiene la oportunidad de analizar las fallas y poder resolverlas haciendo uso de los conocimientos adquiridos en estos años, generando un conocimiento bastante útil, ya que el poder mejorar y resolver fallas es importante para un ingeniero.

Este trabajo tiene la misión principal de abonar al conocimiento de los controladores difusos y sus posibilidades de uso. Esto significa que, al aportar nuevos conocimientos y resultados, los futuros estudiantes tendrán a la mano más conocimiento con el que podrán experimentar y desarrollar cosas nuevas.

1.5 Alcances y limitaciones

El trabajo tiene un potencial para seguir siendo desarrollado, en el caso de que se quiera dar uso a un dispositivo que pueda desplazarse a una distancia deseada. También podría ser implementado en un prototipo más grande o incluso en un auto en caso de tener un resultado sobresaliente. Las posibilidades para desarrollar algún prototipo de movimiento automático o también un dispositivo que transporte carga, son bastante interesantes.

En cuanto a las limitaciones actuales, es importante reconocer que se trabaja con dispositivos electrónicos de bajo coste por lo que los resultados no son 100% exactos, pero si satisfactorios. En ese sentido es importante reconocer que las principales limitaciones son el recurso monetario y el tiempo.

1.6 Organización de la Tesina

Este proyecto fue pensado para un periodo semestral es por eso por lo que se planificaron las actividades en todo el plazo.

Inciso	Nombre de la actividad	Inicio	Semanas	Fin
A	**Revisión de temas a investigar y requisitos**	24/01/2022	1	29/01/2022
B	**Investigación de temas**	30/01/2022	2	12/02/2022
C	**Revisión de Diseño general**	13/02/2022	2	26/02/2022
D	**Modificaciones a coche**	27/02/2022	2	12/03/2022
E	**Modificaciones a conjuntos difusos**	13/03/2022	2	26/03/2022
F	**Implementación de cambios**	27/03/2022	2	09/04/2022
G	**Comparación y análisis**	10/04/2022	2	23/04/2022
H	**Documentación**	24/04/2022	4	21/05/2022

Tabla 1: Actividades de Tesina
Fuente: Propia

2. Marco Histórico

2.1 Antecedentes del proyecto

El proyecto comenzó previamente como un trabajo de prácticas profesionales, el cual consistía en desarrollar un controlador de distancia para un carro eléctrico a escala. Este se conformó de dos variables, una de entrada y una de salida, con tres conjuntos cada una.

Se desarrollo un prototipo que permitía reaccionar conforme se acercaba o se alejaba un obstáculo, todo especificado por la Distancia Deseada.

A pesar del funcionamiento del prototipo este presento algunas fallas, por ejemplo, la rueda giratoria cambiaba la dirección del carro cuando avanzaba o retrocedía en una misma prueba. Otra falla importante era la lenta respuesta del programa que reaccionaba tarde a impactos dando respuestas no muy prácticas.

Los motores también presentaban una respuesta desigual en cuanto a tensión, esto se debió principalmente al módulo puente H L298N, el cual no distribuía eficientemente el voltaje de la placa Arduino UNO.

Por estos fallos se optó por resolver y terminar el proyecto, para obtener un conocimiento adecuado, además de experiencia resolviendo problemas.

2.2 Estado del arte

El desarrollo del control ha significado una de las bases más importantes de la automatización, permitiendo modernizar y desarrollar mejores sistemas de producción de bienes y servicios, obteniendo mejores resultados. Ante este panorama de continuo avance surgen nuevos tipos de control e incluso combinaciones para tareas específicas, entre estos surgió el control difuso, basado en lógica difusa o fuzzy logic.

Las aplicaciones del control difuso son recientes y han obtenido buenos resultados tomando en cuenta que se han utilizado para sistemas de llenado, sistemas de control

de temperatura, aplicaciones robóticas, entre muchas otras. Su uso tiene posibilidades interesantes y todo se remonta a la lógica difusa.

La lógica difusa se le atribuye a Lotfi Asker Zadeh quien desarrollo los conceptos de conjuntos difusos, “son un tipo de objetos con grados de pertenencia. Dicho conjunto se caracteriza por una función de pertenencia (característica) que asigna a cada objeto un grado de pertenencia que oscila entre cero y uno”, (Zadeh, 1965, pp 338). En el artículo de Zadeh definió los conceptos principales de los conjuntos difusos (fuzzy sets), estableciendo que surgieron debido a los sistemas no lineales que existen en los fenómenos físicos. En este artículo también se establecieron definiciones como conjuntos difusos y funciones de membresía.

Estos conceptos tuvieron un importante impulso debido a que representaban los diferentes comportamientos de los fenómenos físicos, ya sea temperatura, presión, velocidad, etc. Siendo capaz de entender diferentes rangos entre términos como “frio, tibio, caliente, muy caliente” dando oportunidad a una respuesta de control menos brusca y más precisa.

En 1974 Ebrahim Mamdani aplico los conceptos de lógica difusa en el control de procesos y desarrolló el primer controlador difuso para la regulación de un motor de vapor (D. Guzmán, 2006) en conjunto con Assilian del Queen Mary College (Inglaterra). Primero Mamdani público “Application of Fuzzy Algorithms for Control of Simple Dynamic Plant” (Mamdani, 1974), en este artículo se describió el esquema en el que un algoritmo difuso es aplicado al control de una máquina de vapor de laboratorio. Posteriormente, los resultados se publicaron en: “An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller” (E. Mamdani, 1975). Ellos concluyeron que el controlador difuso era fácil de construir y que funcionaba muy bien.

Para el año 1978 Lauritz Peter Holmblad y JensJurgen Ostergaard, en Dinamarca, desarrollaron el primer controlador difuso para el control industrial de un horno de cemento (L. Wang, 1997).

En 1985 Takagi y Sugeno aportan a la teoría del control difuso un nuevo método llamado Takagi-Sugeno-Kang (TSK), como alternativa del método Mamdani.

En febrero de 1992, la primera Conferencia Internacional sobre sistemas difusos se realizó en San Diego, California, EE. UU, por el Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers o IEEE). Este evento simbolizó la aceptación de la teoría difusa por la IEEE.

Tras la aceptación de los controladores difusos por parte de occidente, se comenzaron a ver cada vez más trabajos en diferentes aplicaciones, entre las que destacan se encuentra “Controlador difuso para problemas de navegación en presencia de obstáculos fijos”, la cual fue publicada en el 2005 por el Instituto de Microelectrónica de Sevilla, en España. En este artículo se describieron sistemas de control difuso que se usaron para aplicaciones de navegación en robots móviles. Se utilizaron herramientas como Xfuzzy 3 e IMSE. Para el sistema de control se estableció una base de reglas que mejoraron la respuesta obteniendo eficiencia en el controlador (M. Brox, 2005).

Otro artículo destacado proviene de Colombia, con el nombre de “Robot Oruga detector y esquivador de obstáculos con Fuzzy Logic I.A.”. Este articulo detalla el desarrollo de un robot tipo oruga con la finalidad de detectar y esquivar obstáculos. El robot conto con una programación lógica difusa y una adaptación de un módulo Arduino. El robot que fue nombrado “Rover 5” cumplió con los parámetros fuzzy logic, siendo capaz de visualizar los objetos y esquivarlos dependiendo la posición, con excepción de los puntos ciegos, (Andrés Rodríguez, 2015).

Con respecto a México los sistemas difusos comenzaron a ser notorios a partir de los años 2000. Entonces se publicaron libros interesantes que abordaban la lógica difusa, como “Sistemas con Lógica Difusa” (Juan Carlos García Infante, 2009), y “El algoritmo de sintonización simple de controladores difusos (ASSCD)” (Eduardo Gómez Ramírez, 2017). Estos explican y ejemplifican la teoría y los conceptos de conjuntos difusos, explicando las funciones de pertenencia y los diferentes tipos (Mamdani y Sugeno).

También hubo muchas publicaciones útiles, como “Desarrollo de un sistema de navegación para robots móviles mediante diferentes patrones de comportamientos”, en este trabajo se utilizó lógica difusa para responder a diferentes comportamientos. Estos comportamientos controlaban el sistema de navegación implementado en un robot móvil. En el trabajo se logró demostrar que el control difuso logro resultados del 98% en las pruebas y simulaciones (Ubaldo Villaseñor, 2010).

Otra publicación interesante y reciente se encontraba en “Research in Computing Science”, por parte del Instituto Politécnico Nacional. El nombre de la publicación es “Sistema de navegación y evasión de obstáculos aplicando un sistema de control difuso en una placa Arduino UNO” y como el nombre lo indica trata la creación de un sistema de control difuso para navegación y evasión de obstáculos (David Segura, 2019). Para el controlador se implementó un control difuso tipo Mamdani que utiliza el método del centro de máximos (COM) como defusificación en una placa de programación Arduino UNO, demostrando que es posible implementar controles difusos en sistemas de poco poder de procesamiento.

También es importante señalar que ya ha habido trabajos de tesis donde se ha intentado probar el uso de Controladores difusos para diferentes fines, de estos, los más interesantes son los relacionados con robots móviles como la tesis “Seguimiento de ruta mediante control difuso para el estacionado tipo paralelo de un vehículo autónomo a escala” por parte de Luis Enrique Ballina Bermúdez del Instituto Politécnico Nacional (L. Bermúdez, 2018). En este trabajo de tesis se desarrolló de un algoritmo de control difuso para el estacionado de un vehículo autónomo a escala. El objetivo que se buscó fue que el vehículo lograra estacionarse en paralelo a una pared y tras la simulación se obtuvieron buenos resultados.

Por parte de la BUAP hay tres tesis sobre controladores difusos con enfoque en robots, las cuales son: “Desarrollo e implementación de software para mantener horizontal una plataforma ante movimientos externos basado en lógica difusa” de Alan Osorio Orduña. La intención principal de la tesis es el desarrollo de un programa de control

para el problema de la cinemática inversa de un robot paralelo de tres grados de libertad, haciendo uso de teorías de conjuntos y control difusos, además de programación (Orduña, 2014); “Simulación del movimiento de un vehículo terrestre para trayectorias programadas aplicando control difuso” de Eduardo Valadez Campos. La tesis propone un modo de implementar el seguimiento de trayectorias en una ruta sobre un mapa, todo con la intención de volverlo autónomo. Para esto se usan checkpoints en un sistema de control difuso para el control de la dirección de un UGV (Campos, 2018); Y “Control de un robot móvil para mapeo con lógica difusa” de David Luna Ramos. En la tesis se plantea el control de un robot móvil Scribbler, un robot que emplea muchos sensores, por lo que idealmente se escogió el control difuso para su operatividad (Ramos, 2021).

Finalmente, el libro “Sensores y Actuadores: Aplicaciones con Arduino” propuso un ejercicio de aplicación para Arduino, que consistía en la creación de un control difuso de velocidad en un motor DC (Leonel G. Corona Ramírez, 2014, pp 290). Este ejemplo mostraba buenos resultados más sin embargo simplificaba casi todo su proceso con librerías para Arduino, razón por la que su proceso no era definido.

3. Marco Teórico

Un controlador difuso es bastante útil para resolver sistemas complejos o controlar fenómenos físicos con sistemas no lineales, que requieren de una mayor versatilidad en el manejo de las variables. A su vez es importante comprender por qué utilizarlo en robots móviles es importante para la independencia y automatización del futuro. Es por esto por lo que es necesario explicar los conceptos principales de la lógica difusa y su estructura principal.

3.1 Lógica Difusa

Lotfi Asker Zadeh es considerado el padre de la teoría de la posibilidad, esto debido a que introdujo los conjuntos difusos y, como tal, la lógica difusa. En el artículo “Fuzzy sets” de 1965 Zadeh definió los conceptos principales de los conjuntos difusos, estableciendo que surgieron debido a que los sistemas no lineales, eran difíciles de controlar con el control clásico y que muy por el contrario la lógica difusa se podía ajustar a las variables complejas.

3.1.1 Conjuntos Difusos (Fuzzy Sets)

En la teoría de control “un conjunto difuso se define, como una agrupación, clase o colección de objetos donde una función de pertenencia que enlaza o empareja estos elementos de un dominio con elementos del intervalo [0,1], donde cuando más cerca se esté de 1, mayor será la pertenencia del objeto al conjunto mientras que cuando más cerca se esté de 0, el elemento no pertenecerá en lo absoluto al conjunto” (Eduardo Gómez Ramírez, 2017, pág. 16). Esto nos muestra que un conjunto difuso no tiene una delimitación rígida si no una transición gradual que consiste en pertenecer o no a un conjunto en un rango de [0, 1].

Matemáticamente se expresa al conjunto difuso como:

$$A = \{(x, A(x)) | x \in X\} \qquad (1)$$

Donde $A(x): X \rightarrow [0,1]$ es la función de pertenencia y X es el dominio.

3.1.2 Funciones de membresía

La función de pertenencia, también conocida como función de membresía, es la que determina el grado de pertenencia de un elemento, definiéndola con fórmulas matemáticas (Juan Carlos García Infante, 2009).

Existen diferentes tipos de funciones de membresía con diferentes formas las cuales abarcan el mapeo de los datos ingresados.

Función de membresía triangular

La función triangular se compone de tres parámetros {a, b, c}, los cuales representan cada lado del respectivo triangulo. También pueden ser representados como {a, m, b}.

$$Triangular\{x; a, b, c\} = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ \dfrac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \geq x \end{cases} \qquad (2)$$

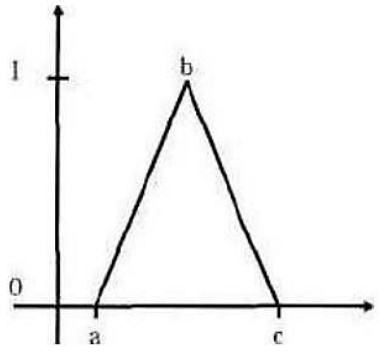

Figura 1: Función de Membresía Triangular
Fuente: (Juan Carlos García Infante, 2009, pág. 26)

Función de membresía trapezoidal

La función trapezoidal se compone de cuatro parámetros {a, b, c, d}, los cuales representan cada lado del respectivo trapecio.

$$Trapezoidal\{x; a, b, c, d\} = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{c-x}{c-b}, & c \leq x \leq d \\ 0, & d \geq x \end{cases} \quad (3)$$

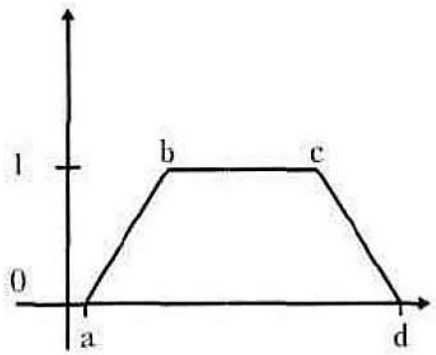

Figura 2: Función de Membresía Trapezoidal
Fuente: (Juan Carlos García Infante, 2009, pág. 27)

Función de membresía de campana

La función de campana se compone de tres parámetros {a, b, c} donde b es comúnmente positivo, de lo contrario la campana es inversa.

$$Campana\{x; a, b, c\} = \frac{1}{1\left|\frac{x-c}{a}\right|^{2b}} \quad (4)$$

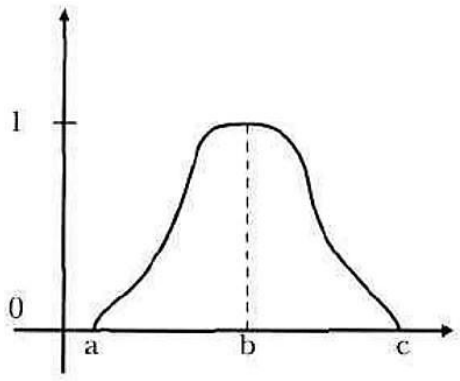

Figura 3: Función de Membresía de Campana
Fuente: (Juan Carlos García Infante, 2009, pág. 28)

Función de membresía S

La función S es especial por su forma no recta y por un valor medio *m* el cual se puede obtener fácilmente. Se compone de los parámetros {a, m, c}.

$$A(x;a,m,c)=\begin{cases}0, & x\le a\\ 2\left\{\frac{x-a}{c-a}\right\}^2, & a<x\le m\\ 1-2\left\{\frac{x-c}{c-a}\right\}^2, & m<x<c\\ 1, & x\ge c\end{cases} \qquad m=\frac{a+c}{2} \qquad (5)$$

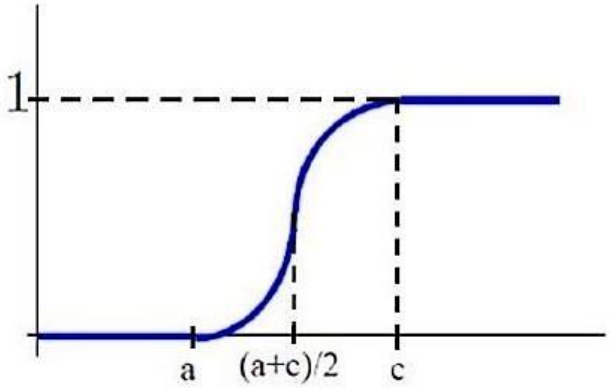

Figura 4: Función de Membresía S
Fuente: (Palacios, 2021, pág. 16)

Función de membresía Z

La función Z es la inversa de la función S e igualmente se compone de tres parámetros {a, m, c}.

$$\mu Z\{x; a, m, c\} = \begin{cases} 1, & x \leq a \\ 1 - 2\left(\frac{a-x}{c-a}\right)^2, & a < x \leq m \\ 2\left(\frac{c-x}{c-a}\right)^2, & m < x < c \\ 0, & x \geq c \end{cases} \qquad m = \frac{a+c}{2} \qquad (6)$$

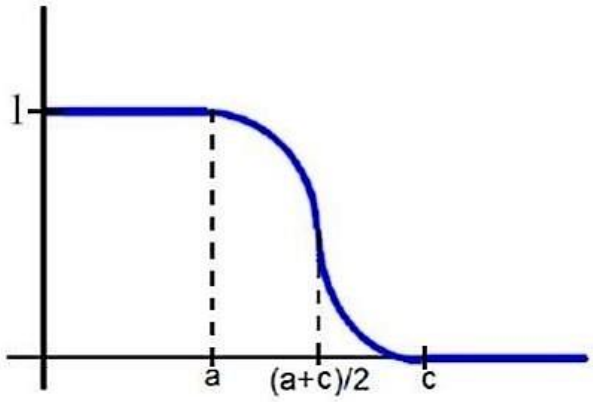

Figura 5: Función de Membresía Z
Fuente: (Palacios, 2021, pág. 17)

3.1.3 Operaciones con conjuntos difusos

Las funciones de pertenencia definen al conjunto difuso es por eso por lo que se utilizan operaciones bastante útiles que abordan la interacción de las variables dentro de la función de membresía. Entre todas estas las más importantes son:

Unión: $(A \cup B)X = A(X) \cup B(X) = \max\{A(X), B(X)\}$ (7)

Intersección: $(A \cap B)X = A(X) \cap B(X) = \min\{A(X), B(X)\}$ (8)

Complemento a uno o negación: $\bar{A}(X) = 1 - A(X)$ (9)

3.1.4 Variables lingüísticas

La lógica clásica trabaja con verdades absolutas, por ejemplo, encendido/apagado, verdadero/falso, lleno/vacío. No obstante, la vida real y el universo es más complejo que eso y existe una escala o rango para las variables, en donde sabemos que, si bien existe el blanco y el negro, también existe el gris y una larga gama de tonos en medio. Ahí es donde surge la lógica difusa, en la cual las variables de entrada o salida se clasifican como variables lingüísticas.

Una variable lingüística se compone de diferentes conjuntos difusos (funciones de pertenencia). Estos a su vez se pueden etiquetar lingüísticamente y deben estar ubicadas en su rango o universo (Eduardo Gómez Ramírez, 2017).

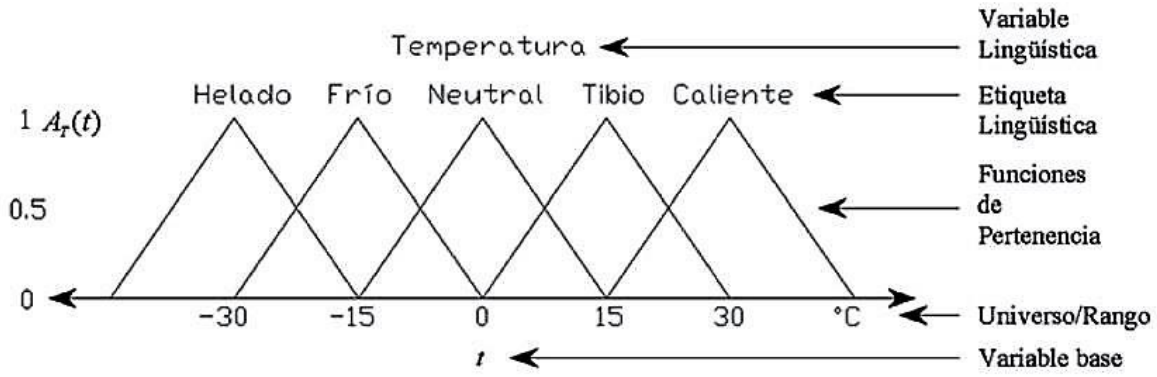

Figura 6: Estructura de variables lingüísticas de temperatura
Fuente: (Eduardo Gómez Ramírez, 2017, pág. 24)

3.1.5 Reglas Difusas

Las reglas difusas o reglas de inferencia son parámetros que establecen como serán las interacciones entre conjuntos difusos estableciendo de una manera más natural una respuesta en el control. Las reglas también son consideradas IF-THEN o Si-Entonces ya que siguen un antecedente y una consecuencia:

SI como un antecedente, ENTONCES como una consecuencia

En estas reglas se combinan uno o más conjuntos de entrada (antecedentes o premisas) y se asocian a un conjunto de salida (consecuencia).

R1: Si X es A; entonces Y es B

Donde R1 es la Regla 1, X es una variable de entrada y A es su conjunto difuso, Y es la variable de salida y B es su conjunto. Con esta estructura también se pueden aplicar reglas cuando se tienen más variables de entrada, por ejemplo:

$$R1: Si\ X_0\ es\ A_1\ \wedge\ Y_0\ es\ B_1\ entonces\ Z\ es\ C_1$$

$$R2: Si\ X_0\ es\ A_2\ \wedge\ Y_0\ es\ B_2\ entonces\ Z\ es\ C_2$$

$$\vdots$$

$$R_n: Si\ X_0\ es\ A_n\ \wedge\ Y_0\ es\ B_n\ entonces\ Z\ es\ C_n$$

Para una respuesta precisa y clara se crea una base de reglas donde se expresa cada combinación posible, no obstante, cuando son varias es mejor representarlas en una tabla de reglas, también conocida como memoria asociativa difusa o FAM (Fuzzy Associative Memory). Estas son matrices que representan gráficamente cada consecuencia de cada premisa o antecedente. Por ejemplo:

R1: Si X es P y Y es P; entonces Z es G

R2: Si X es M y Y es P; entonces Z es M

⋮

R9: Si X es G y Y es G; entonces Z es P

Y\X	P	M	G
P	G	M	P
M	M	M	P
G	P	M	P

Tabla 2: Base de reglas
Fuente: Propia

3.2 Control Difuso

El control es un proceso importante que engloba gran parte de la tecnología moderna, ha facilitado y simplificado procesos que han ayudado al ser humano, desde regular temperatura en un lugar hasta automatizar procesos industriales, todo ha permitido un gran manejo de energía y recursos. Con esto se ha comprendido que existen muchos

procesos donde no es tan fácil obtener un modelo matemático para controlar una planta o un sistema, por lo que los controladores difusos tienen una ventana de oportunidad importante para resolver y automatizar procesos más complejos con variables que están en constante cambio.

Ebrahim H. Mamdani fue un importante ingeniero eléctrico y matemático, quien desarrollo el primer controlador difuso para una máquina de vapor y su trabajo se tomó como modelo a seguir para desarrollar los siguientes controladores, bautizando el método como el método de Mamdani (Mamdani, 1974).

El control difuso puede aplicarse fácilmente a procesos donde las variables cambian de valor drásticamente, por ejemplo, en el control de un ventilador con respecto a la temperatura del ambiente, en este caso solo habría una entrada que sería la temperatura y la salida seria la planta o sistema a funcionar en este caso el ventilador.

3.2.1 Estructura de Control Difuso

Un controlador difuso se compone de cuatro elementos los cuales son: las reglas, la Fusificación, el mecanismo de inferencia y la Defusificación.

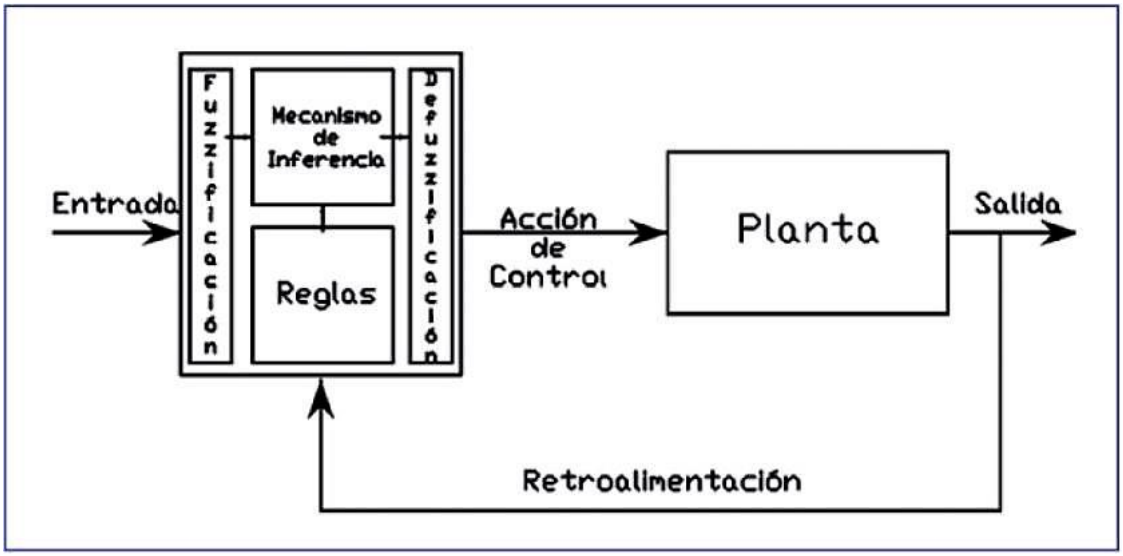

Figura 7: Esquema básico de control difuso
Fuente: (Eduardo Gómez Ramírez, 2017, pág. 29)

Tomando en cuenta esto, es importante resaltar que para diseñar un controlador difuso se tiene que escoger correctamente las variables de entrada y de salida, ya que puede tener varias, y representar los parámetros en reglas que podamos establecer.

3.2.2 Método de Inferencia

El método de inferencia tiene la finalidad de obtener una función de membresía **μ(y)** con respecto de las reglas difusas. Esta función de membresía representaría la salida. Sin embargo, para esto es necesario tener en cuenta procesos que ocurren.

Por ejemplo, si tenemos reglas de una variable de entrada y una de salida:

$$x\ es\ A'$$

$$si\ x\ es\ A_1, entonces\ y\ es\ B_1$$

$$si\ x\ es\ A_2, entonces\ y\ es\ B_2$$

$$\vdots$$

Se buscará obtener la variable de salida B' de modo que su interpretación matemática es:

$$R' = A' \quad \rightarrow \quad \mu_{A'}(x)$$

$$R_1 = A_1 \times B_1 \quad \rightarrow \quad \mu_{R_1}(x, y)$$

$$R_2 = A_2 \times B_2 \quad \rightarrow \quad \mu_{R_2}(x, y)$$

$$\vdots$$

Para simplificar es necesario aplicar las operaciones con conjuntos difusos de unión e intersección de forma que la función de membresía que se obtendría seria:

$$\mu_{B'_i} = Max_x\left[\min\left[\mu_{A'}(x), \mu_{R_i}(x, y)\right]\right] = \bigvee_x\left[\mu_{A'}(x) \wedge \mu_{R_i}(x, y)\right] \qquad (10)$$

Lo que hace el método es una comparación entre funciones de membresía, para eso obtenemos los mínimos entre A' que está en el dominio de (x) y cada una de las reglas que están en dominio de (x, y). Posteriormente buscamos el máximo.

Este proceso se llama Fusificación y tiene la finalidad de obtener un dato entre la intersección y unión en donde busca solo obtener un área específica en *B* de cada regla y de esa forma armar la función de membresía de salida.

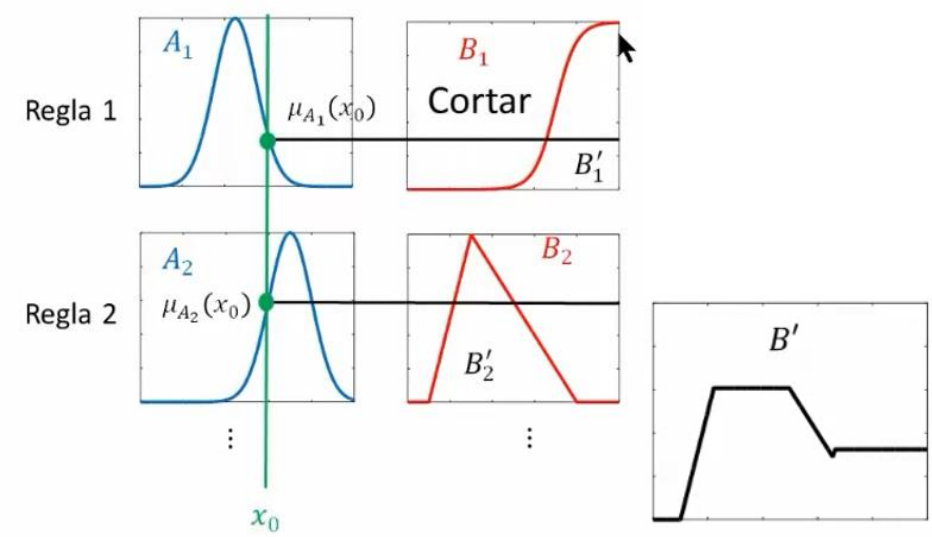

Figura 8: Fusificación
Fuente: (Palacios, 2021, pág. 24)

3.2.3 Defusificación

Debido a que, al obtener la función de membresía de salida, en realidad obtenemos una serie de datos, no podemos utilizarlos como salida debido a que para la respuesta del controlador se necesita un dato a la vez, es por eso por lo que se necesita el proceso de defusificación.

La defusificación busca obtener un numero de la función de membresía de salida, el cual sería el verdadero dato final del controlador. Existen diferentes métodos de defusificación:

- Centroide: considera la función de membresía de salida como si fuera una función de distribución de masa, buscando su centro. Su representación es:

$$y_0 = \frac{\sum y\mu(y)}{\sum \mu(y)} \qquad (11)$$

- Bisectriz: y_0 divide el área en dos regiones iguales.
- Máximo Central (MOM): y_0 es el promedio de los máximos.
- Máximo más pequeño (SOM): y_0 es el valor más pequeño.
- Máximo más grande (LOM): y_0 es el valor más grande.

3.2.4 Método de Mamdani

El método de Mamdani es a grandes rasgos el método de inferencia que se desarrolló basándose en el primer controlador difuso de Mamdani y se ha utilizado para la mayoría de los controladores difusos. Su desarrollo fue bastante importante, tanto que los otros métodos (Sugeno y Tsukamoto) derivan de este.

3.3 Robots Móviles

Existen varios proyectos de robots móviles en donde se aplican controladores difusos con la intensión de manejar diferentes variables, esto debido a que los obstáculos son bastantes, es por esto por lo que cada proyecto escoge pocas variables y busca desarrollarlas lo mejor posible. Ejemplos:

Robot oruga (Andrés Rodríguez, 2015)

El robot consta de dos pares de llantas que están unidas por una banda, a su vez el motor maneja solo a dos de las cuatro llantas. Su utilidad es buena debido a que puede moverse en cualquier tipo de terreno lo que lo hace ideal para muchas operaciones.

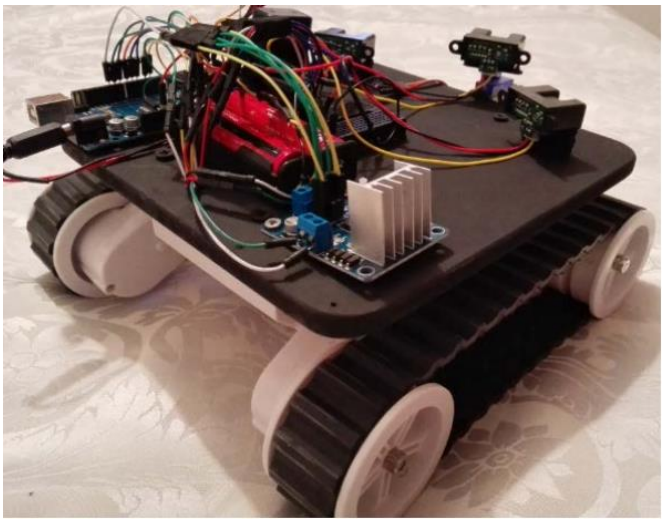

Figura 9: Robot Oruga
Fuente: (Andrés Rodríguez, 2015, pág. 374)

El controlador que se desarrollo tuvo la intención de evitar obstáculos a través de sensores ultrasónicos y siendo manejado por una tarjeta de programación Arduino.

Robot Scribbler (Ramos, 2021)

El robot tiene una forma circular baja, se compone de dos llantas que transportan del dispositivo y una o dos llantas pequeñas para equilibrio. Además, el mismo cuenta con varios sensores por lo que la cantidad de variables a tratar es amplia.

Figura 10: Robot Scribbler
Fuente: helenemartin.com

El controlador desarrollado tiene la intención de controlar completo el robot, principalmente su navegación, desarrollando un mapa en simulador sobre su ruta.

Carro a escala

El robot móvil que se escogió para este proyecto fue un carro a escala, esto debido a la accesibilidad de materiales que hay en el mercado, además de haber un antecedente en el cual se diseñó un controlador para la navegación y evasión de obstáculos para un carrito (David Jesús Segura Cristino, 2019).

El proyecto implemento un controlador difuso de tipo Mamdani con el método de defusificación máximo central (MOM), en una plataforma con la placa de programación Arduino UNO. Todo con la intención de demostrar que es posible aplicar un controlador de este tipo en un sistema de poco poder computacional.

3.4 Controlador para robot móvil

Los controladores y los sistemas de control en general son una parte importante para la robótica, desde el diseño de un brazo robot, hasta sencillas puertas automáticas, utilizan sistemas de control que dictan sus acciones de acuerdo con variables que entran de forma externa. Esto ha derivado en prototipos y proyectos cada vez más eficientes y autónomos que han sido capaces de sustituir trabajos humanos, obteniendo resultados similares o incluso mejores.

Es por esto por lo que el desarrollo de estos ha avanzado bastante ya que los beneficios se traducen en robots más autónomos y eficientes, que significan un ahorro de dinero para la industria.

Con este panorama, con tantos tipos de robots, los móviles son los que presentan un mayor reto a comparación de los estáticos, ya que la cantidad de obstáculos y variables a las que se enfrentan son bastantes lo cual se vuelve complicado a procesar. Para este proyecto se desarrolló un controlador difuso el cual se compone de dos partes importantes el software y el hardware.

3.4.1 Software

El software siendo una parte importante para casi cualquier proyecto de ingeniería, juega un papel importante en este proyecto, siendo tres los necesarios para este trabajo.

Matlab

Es el software más completo en cuanto a operaciones y simulaciones matemáticas, permitiendo modelar sistemas matemáticos complejos y por supuesto sistemas de control. Su importancia es destacable por sus grandes capacidades además de poder graficar y simular los comportamientos de sistemas matemáticos.

La operatividad es familiar debido a su estructura de funcionamiento basada en algoritmos, donde es sencillo plasmar operaciones matemáticas y ejecutarlo como cualquier código tipo C con la simplicidad de BASIC.

Las ventajas principales es que traduce las sentencias plasmadas en un código, permitiendo observar y o graficar los diferentes cálculos y sistemas especificados (Antonio Souto Iglesias, 2013).

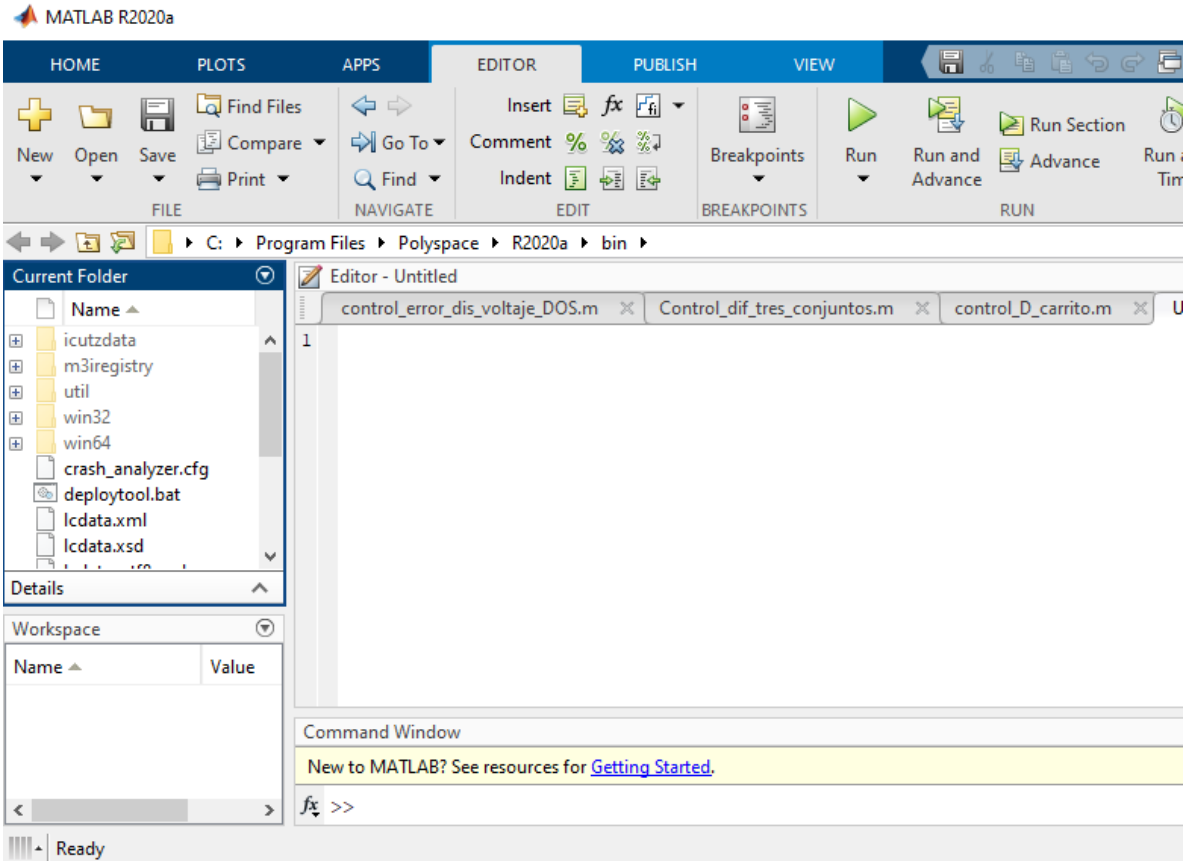

Figura 11: Interfaz MATLAB
Fuente: Propia

Arduino IDE

Arduino engloba un sistema que permite desarrollar proyectos de electrónica. Este sistema se compone de la tarjeta de programación Arduino, en sus diferentes variantes, y el software de programación Arduino IDE. Entre las ventajas principales de utilizar Arduino está el hecho de que podemos programar proyectos de electrónica, varias veces y estos a su vez, se ejecutan una y otra vez mientras la placa este alimentada.

Los proyectos en Arduino son variados, van desde robótica hasta domótica y cualquier otro de carácter electrónico.

El software utiliza un lenguaje tipo C, bastante interactivo, puesto que permite utilizar librerías, además poder crear las tuyas o acceder a una comunidad bastante amplia con conocimientos variados.

La estructura del software es sencilla se compone de un **void setup** que ejecuta lo establecido una vez y de un **void loop** que repite infinitamente el código escrito en el, además de tener la posibilidad de crear funciones **void** personalizadas y llamarlas de ser necesario.

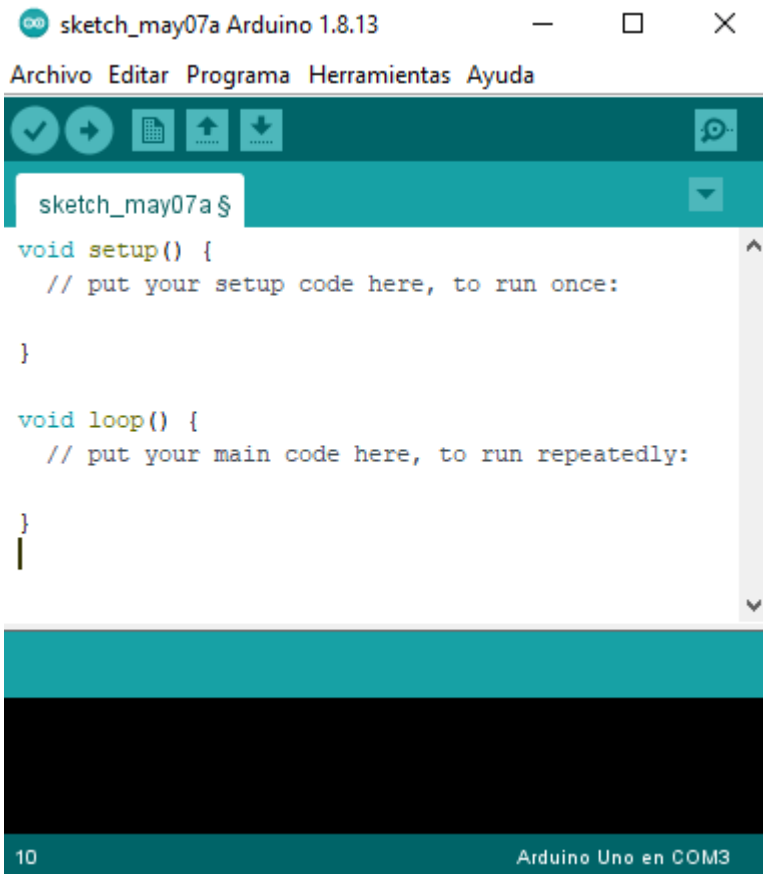

Figura 12: Interfaz Arduino IDE
Fuente: Propia

MIT App Inventor

MIT App Inventor es un entorno de programación visual desarrollado por MIT (Instituto Tecnológico de Massachusetts) para la creación interactiva de aplicaciones para teléfonos Android, iPhone y Tabletas Android/iOS. Los programas codificados son

basados en bloques en donde se especifican funciones, desde control, conectividad, etc.

Para manejar el entorno y poder crear una aplicación móvil es necesario crear una cuenta en el sitio web de app inventor.

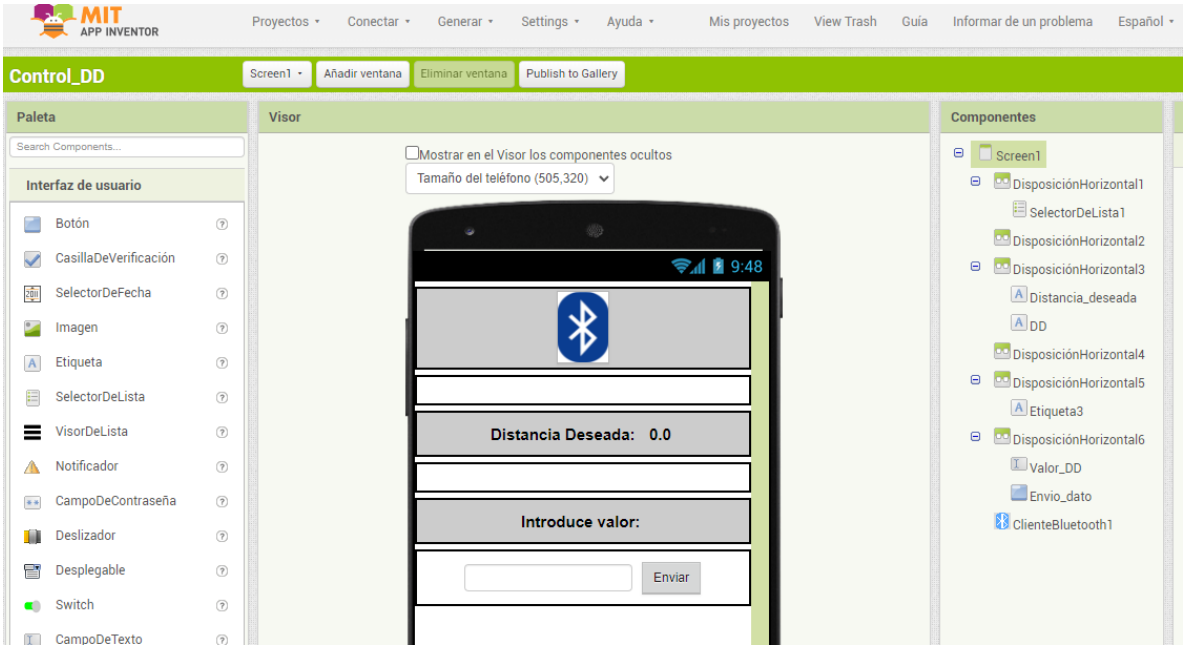

Figura 13: Entorno de MIT App Inventor
Fuente: Captura propia de appinventor.mit.edu

3.4.2 Componentes electrónicos

Arduino UNO

Arduino Uno es una placa compuesta de un microcontrolador, generalmente ATMEGA 328P de 8-bits, pines digitales y analógicos, un conector USB, conexión a alimentación, un botón reset, conectores ICSP y algunos indicadores LED.

Figura 14: Esquema Arduino UNO
Fuente: (López, 2016, pág. 45)

Todo este sistema permite introducir datos para procesarlos y obtener como resultados en tensión (voltaje) salidas ya sea digitales o analógicas (López, 2016). Además, esta placa permite unir diferentes componentes con los pines ya sea para introducir datos o recibirlos de la misma.

Para este proyecto el Arduino UNO que se utilizo tiene las siguientes especificaciones:

- Microcontrolador: ATMEGA 328P-PU marca ATMEl
- Pines Digitales: 14 (6 con la capacidad de emitir PWM).
- Pines Analógicos: 6 (como entrada).
- Otros pines: 1 de 5V, 1 de 3.3V, 3 de GND, 1 de RESET, 1 de IOREF y 1 de AREF.
- Intensidad por pin: 40 mA
- Alimentación: 7-12V (recomendada).

Es importante resaltar que, aunque es posible alimentar el Arduino por el cable USB y/o también por un adaptador de corriente alterna, es recomendable alimentarlo con baterías, debido a que se busca un sistema con movilidad y cierta autonomía.

Sensor Ultrasónico HC-SR04

El sensor HC-SR04 mide la distancia mediante un método curioso, cuenta el tiempo que tarda una señal ultrasónica de 40 kHz enviada (**Echo**) desde el transmisor, rebota sobre el objeto y regresa (**Trig**) al receptor. Además, es sencillo de alimentar, basta un voltaje de 5V.

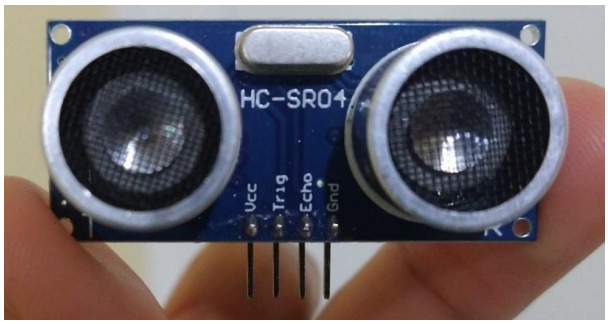

Figura 15: Sensor Ultrasónico HC-SR04
Fuente: Captura propia

Su funcionamiento consiste en enviar una señal en alto a través del pin que tenga asignado **Trig** con una duración de 10 microsegundos. Es entonces cuando se envía una señal en alto en el pin asignado para **Echo** que durara hasta que reciba la señal de regreso.

Modulo Bluetooth HC-06

Este módulo es un dispositivo electrónico que permite el intercambio de datos a través de una conexión Bluetooth estándar 2.0. Es muy usado para proyectos de robótica móvil porque permite dar movilidad al quitar cables y permitir manejos remotos. Este dispositivo trabaja a través de la conexión serial de Arduino y opera con los comandos AT.

Una característica importante es que al módulo solo se le pueden ordenar, es decir está configurado solo como esclavo (slave), por lo que su utilidad se limita a eso, pero es suficiente para las necesidades del proyecto.

Figura 16: Modulo Bluetooth HC-06
Fuente: Captura propia

El dispositivo cuenta solo con cuatro pines, dos son para establecer la conexión, TX y RX, y dos para funcionar, es decir Vcc y GND.

- Vcc: Entrada de voltaje positivo de entre 3.3V y 6V.
- GND: Tierra.
- TX: Pin de transmisión de datos.
- RX: Pin de recepción de datos.

Otras especificaciones técnicas:

- Voltaje de operación: 3.3V y 5V.
- Corriente de operación: 40mA
- Chip: BC417143
- Bluetooth: V2.0+EDR
- Frecuencia: Banda ISM de 2.4GHz
- Alcance: 10m.
- Interfaz de comunicación: Serial UART TTL.

Driver Puente H TB6612FNG

El módulo puente H TB6612FNG es, a grandes rasgos, un circuito integrado que permite el manejo de dos motores, desde cambiar su dirección hasta regular su velocidad, esto se consigue debido a que en realidad son dos puentes H, es decir, un arreglo de transistores MOSFET que direcciona el paso de la energía según se permita.

La forma de operarlo es mediante PWM para cada motor, además de escoger la dirección de acuerdo con dos pines de entrada para cada motor, esto quiere decir que cuando un pin de entrada está en alto (HIGH) y el otro en bajo (LOW), el motor trabajara en un sentido y si cambian el estado de los pines, cambiara el sentido. Esto aplica para los dos motores.

Este modelo cuenta con un Pin extra que no tienen otros módulos como el L298N, este es STBY, el cual como su nombre sugiere al ponerlo en alto lo pone en standby.

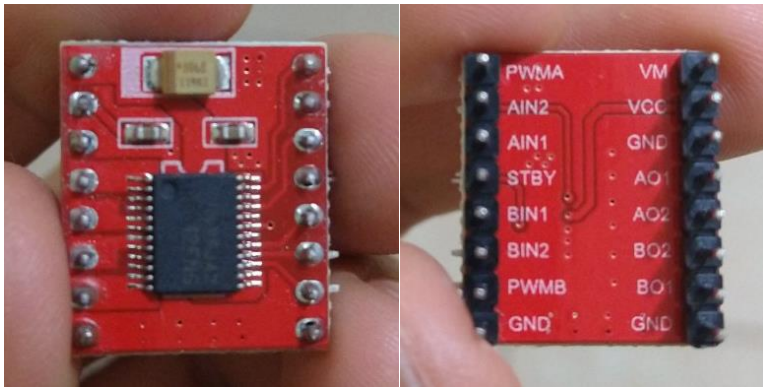

Figura 17: Modulo puente H TB6612FNG
Fuente: Captura propia

Entre los pines témenos los siguientes:

- PWMA: Recibe la señal PWM que controlara al motor A.
- PWMB: Recibe la señal PWM que controlara al motor B.
- AIN1 y AIN2: Pines de entrada que definirán el sentido del motor, para esto uno deberá estar en HIGH y el otro en LOW.

- BIN1 y BIN2: Pines de entrada que definirán el sentido del motor, para esto uno deberá estar en HIGH y el otro en LOW.
- STBY: Pin que al estar en HIGH entra en standby.
- GND: Tierra.
- VM: Voltaje que manejara el o los motores, va de 5V a 15V.
- Vcc: Voltaje que permite el funcionamiento del módulo generalmente de 5V.
- A01 y A02: Pines se salida al motor A.
- B01 y B02: Pines de salida al motor B.

Otras especificaciones técnicas son:

- Canales: 2 (soporta dos motores DC o uno PAP)
- Voltaje de potencia (VMOT): 5V-15V.
- Voltaje de operación (Vcc): 2.7V-5.5V.
- Capacidad de corriente: 1.2A (picos de hasta 3A).
- Posee diodos internos de protección.

Display LCD 16x2

Una pantalla o display LCD (También conocida como Liquid Crystal Display) es un dispositivo electrónico que transforma las señales eléctricas en información visual, lo que la convierte en un elemento útil para tener datos en tiempo real que están siendo trabajados en la placa de Arduino.

La pantalla utilizada es alfanumérica de 16x2, lo que le da la posibilidad de reproducir letras y números en dos filas con 16 columnas, lo que implica que trabaja con sistemas de matrices (como la mayoría de las pantallas).

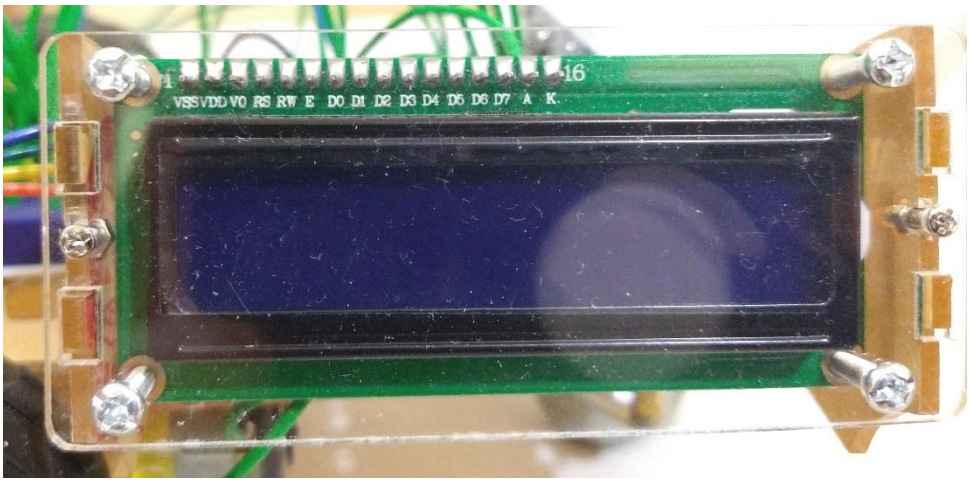

Figura 18: Pantalla LCD alfanumérica 16x2
Fuente: Captura propia

Los pines con los que opera son bastantes y es importante conocerlos.

Pin No	Función	Nombre
1	Tierra (0V)	Ground
2	Voltaje de Operación; 5V (4.7V – 5.3V)	Vcc
3	Ajuste de contrastes: mediante un potenciometro (no incluido, puede ser de 1k-10k)	V_{EE}
4	Selección del registro, para 0 es comandos y en 1 es para datos	Register Select
5	Estado bajo para escribir y estado alto para leer el registro	Read/write
6	Envia datos a los pines de datos cuando recibe un flanco de bajada	Enable
7	Pines de datos 8-bit	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Tierra (0V)	Led-

Tabla 3: Pines de Display LCD
Fuente: web naylampmechatronics.com

Especificaciones:

- Voltaje de operación: 5V.
- Color de texto/numero: Blanco.
- Color de fondo: Azul.
- Filas: 2.
- Columnas: 16.
- Interfaz de comunicación: 4 u 8 bits.

I2C

Para este trabajo se optó por utilizar un módulo adaptador llamado I2C, el cual simplifica el trabajo de operar 16 pines a solo 2 cuando se trabaja con un Display LCD, lo que representa un ahorro importante en cables y sobre todo pines en Arduino.

I2C es un módulo que trabaja con dos pines (SDA y SCL), aparte de la alimentación (Vcc y GND). Adicionalmente también es importante resaltar que ya posee integrado el potenciómetro que es usado comúnmente para regular la iluminación de la pantalla.

Este módulo trabaja con algunas librerías de Arduino lo que simplifica mucho también su operación dentro de la programación.

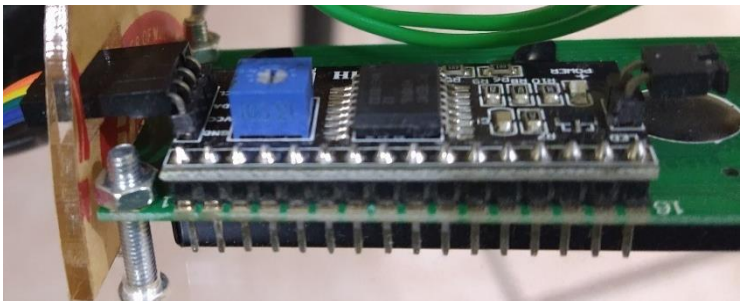

Figura 19: Modulo I2C
Fuente: Captura propia

Especificaciones:

- Voltaje de operación: 5V.
- Compatible con protocolo I2C.
- Potenciómetro para ajuste de contraste.
- Chip controlador: PCF8574.

Motorreductores DC

Un motorreductor DC es un motor pequeño de corriente continua en una caja reductora de plástico amarillo la cual está diseñada para portar una llanta de plástico. Son muy útiles para robots móviles o robots sumos.

Su caja reductora opera con una serie de engranajes que reducen las RPM (revoluciones por minuto) y operan usualmente entre 6V y 12V.

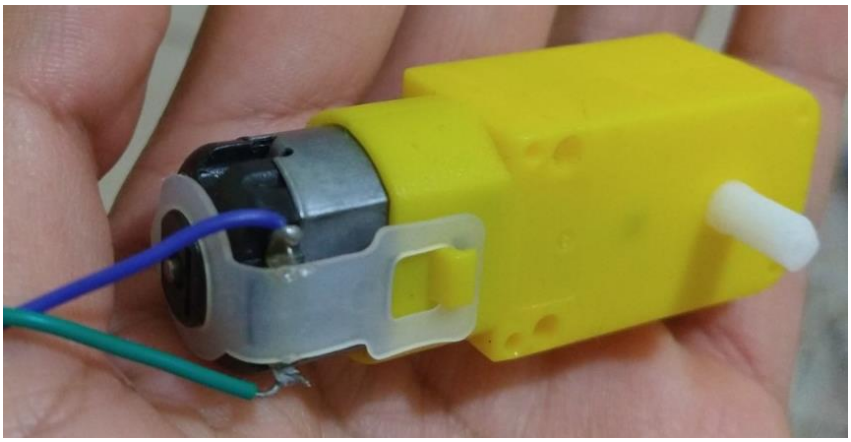

Figura 20: Motorreductor
Fuente: Captura propia

4. Controlador de Distancia Difuso

La metodología que se determinó para el proyecto es experimental con un enfoque cuantitativo, esto tomando en cuenta que el trabajo plantea desarrollar un prototipo físico para probar la hipótesis.

El proyecto comenzó con la investigación de fuentes de información sobre el tema de lógica y controladores difusos. Posteriormente se diseñó el prototipo con los respectivos materiales y dispositivos electrónicos.

4.1 Trabajo previo

En un trabajo anterior se desarrolló un controlador difuso con la intención de manejar la distancia en un carrito a escala, sin embargo, el proyecto mostro deficiencias tanto en el programa como en el prototipo.

El control difuso contó con dos variables, una de entrada y otra de salida, con tres conjuntos difusos triangulares cada una. La variable de entrada es el error con respecto a la distancia (ED), la cual se obtiene restando la distancia actual (D) de la distancia deseada (DD). La distancia deseada (DD) es la ubicación a la que se desea posicionar el coche y la distancia actual (D) es la medida en tiempo real que va proporcionando el sensor ultrasónico.

$$ED = DD - D \qquad (12)$$

El rango de distancia que se utilizo fue de 0cm a 150cm por lo que para tener una distancia deseada de la mitad (75cm), se definió un rango de -75cm a 75cm para el error con respecto a la distancia.

$$ED = 75cm - 150cm = -75cm \qquad (13)$$

$$ED = 75cm - 0cm = 75cm \qquad (14)$$

Los conjuntos difusos por su parte se definieron como:

- EDN = Error de Distancia Negativo
- EDM = Error de Distancia Medio
- EDP = Error de Distancia Positivo

La salida se definió como el Voltaje (V) de salida que iba hacia los motorreductores y se mantuvo en un rango de -8V a 8V. Sus conjuntos se definieron como:

- VN = Voltaje Negativo
- VM = Voltaje Medio
- VP = Voltaje Positivo

Para las reglas lingüísticas se asignaron tres:

- Si el error de distancia es EDN, entonces el voltaje de salida es VP
- Si el error de distancia es EDM, entonces el voltaje de salida es VM
- Si el error de distancia es EDP, entonces el voltaje de salida es VN

4.1.1 Simulación

Para demostrar los conceptos se simularon primero en MATLAB implementando los modelos matemáticos para los conjuntos difusos triangulares y se mandaron a graficar con los valores asignados en las variables de entrada y salida respetando los rangos de -75cm a 75cm de ED y -8V a 8V de V.

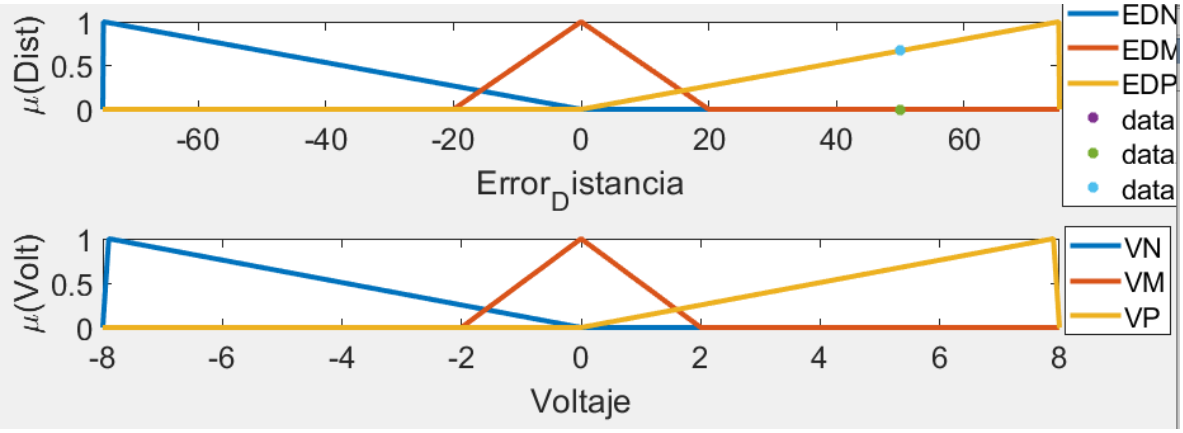

Figura 21: Graficas de las variables ED y V
Fuente: Captura propia Matlab

Demostrando el comportamiento de las funciones de membresía se asignó un valor a **T** buscando un valor (**d0**) en el rango de ED. Este valor es usado para el método de inferencia sacando el mínimo comparando las funciones de membresía de acuerdo con las reglas lingüísticas, utilizando del valor de **T** en el error de distancia. Después se obtuvo el máximo entre las reglas y con ello la función de membresía de **R**.

```
76 -    d0 = 50; % Error Distancia leida marcada en puntos
77 -    T = find(ED==d0);
78 -    subplot(3,1,1), hold on,
79 -    plot(d0,EDN(T),'*',d0,EDM(T),'*',d0,EDP(T),'*','LineWidth',3),
80 -    hold off
81
82      % Fusificacion Error Distancia por metodo Mandani
83 -    R1 = min(EDN(T),VP);
84 -    R2 = min(EDM(T),VM);
85 -    R3 = min(EDP(T),VN);
86 -    R = max(R1,max(R2,R3));
87 -    subplot(3,1,3), plot(V,R,'LineWidth',3)
88 -    set(gca,'FontSize',18), legend('V')
89 -    axis([-8 8 0 1])
```

Figura 22: Método de inferencia en Matlab
Fuente: Captura propia de Matlab

Para el método de defusificación se utilizó una función llamada **defuzz** con el rango de V y la función de membresía **R** en la que se especificó el método por el centroide para posteriormente ser graficado. De esta forma obtenemos el comportamiento y la intersección en la función de membresía.

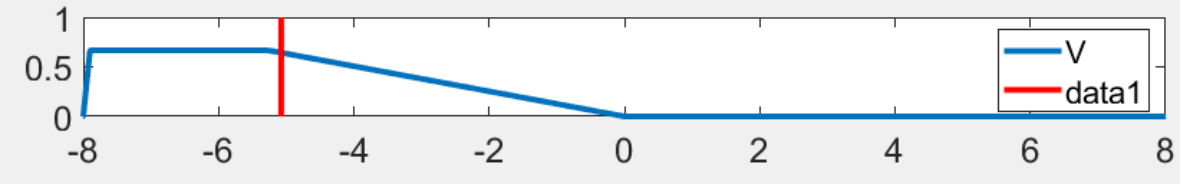

Figura 23: Grafica de función de membresía con método de centroide
Fuente: Captura propia Matlab

Se opto por el método del centroide para la defusificación debido a que es el método más utilizado, mostrando en general mejores resultados. Por contrario tanto el método del máximo más grande como el máximo más pequeño, presentan una respuesta más brusca.

4.1.2 Primer prototipo

El primer prototipo se montó en una base de acrílico, llevando los motorreductores atornillados y como soporte una llanta giratoria. Se utilizo un sensor ultrasónico HC-SR04 para determinar el valor de la distancia, un Display LCD con I2C para revelar los datos procesados, un módulo L298N para manejar los motorreductores y la placa Arduino UNO. Cabe mencionar que en las pruebas se utilizó una fuente de 12V para alimentar a los motores.

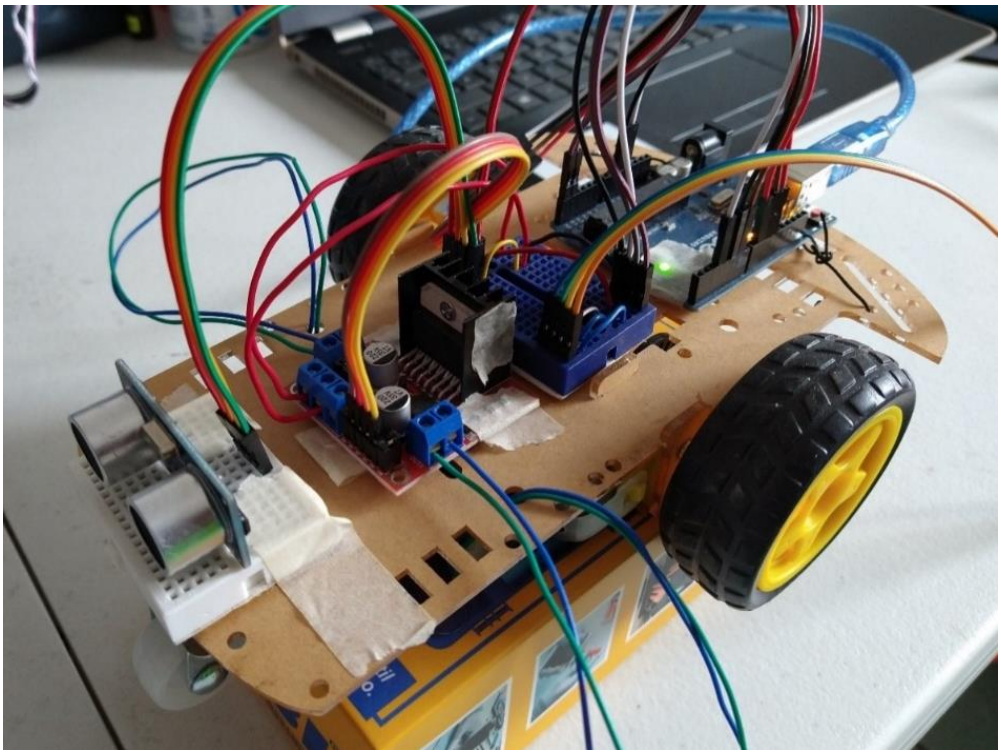

Figura 24: Primer prototipo de carro eléctrico a escala
Fuente: Captura propia

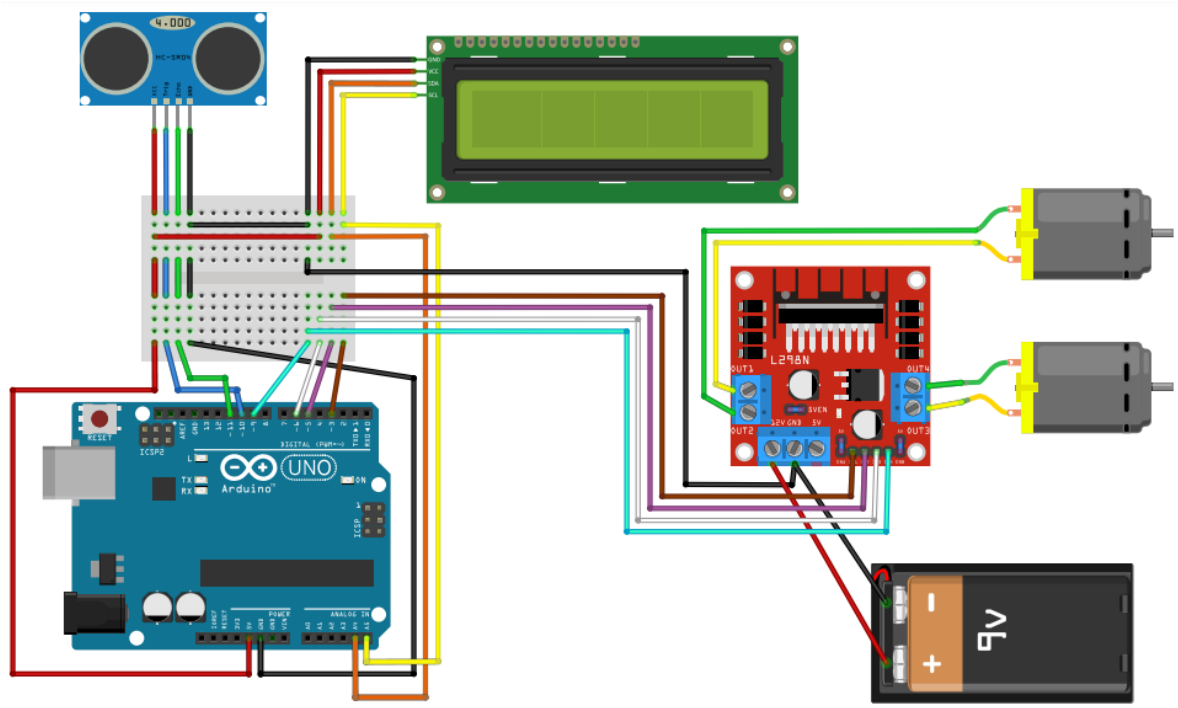

Figura 25: Esquema de conexión del prototipo 1
Fuente: Captura propia Fritzing

4.1.3 Primer controlador

Para adaptar la simulación a un programa de Arduino fue necesario entender que, a comparación de Matlab, Arduino obtiene una respuesta física por parte de los pines de salida y que también, el programa se ejecuta en un ciclo infinito. Es por esto por lo que en el programa se comenzó estableciendo librerías para manejar la pantalla LCD además de declarar todas las variables para el sensor, el módulo L298N, las variables ED y V con sus conjuntos, además de algunas otras variables de Arduino tipo **float**, que fueron necesarias para procesos posteriores.

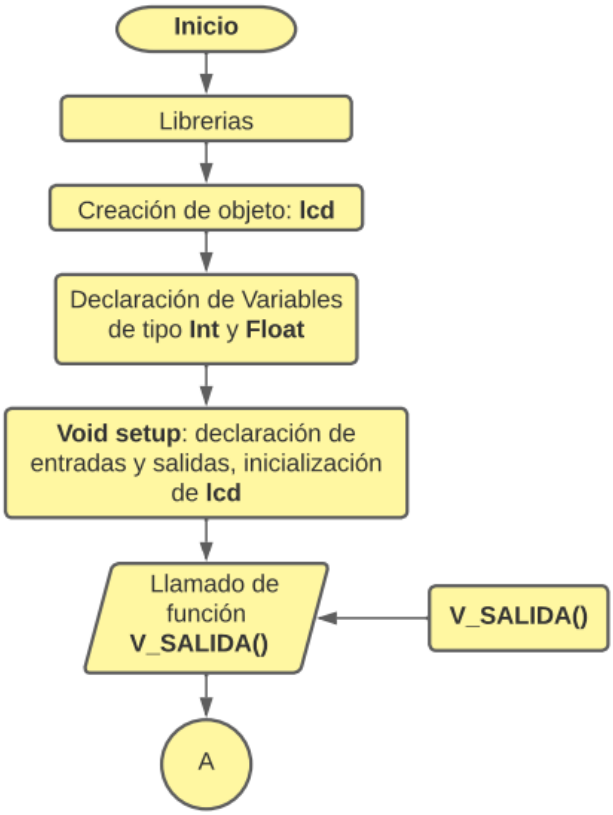

Figura 26: Diagrama de flujo 1 del primer controlador
Fuente: Creación propia

En el **void setup** se declararon las variables de Arduino para los pines de salida (OUTPUT) y entrada (INPUT), así como se mandó a encender la luz de la pantalla LCD inicializándola. También se mandó a llamar la función **V_SALIDA()** la cual en realidad es un repaso de los conjuntos de salida de V, en donde, con un ciclo for, se inicia un conteo con el cual se evalúa el valor en cada función de membresía triangular iniciando por VN y continuando con VM y VP. El conteo inicia en cero en la variable **i** para terminar en nueve, permitiendo con cada número seleccionar la columna a la que corresponde el mismo en el vector de **V[ ]**. Este ciclo permite evaluar el valor seleccionado en el vector con respecto a todos los conjuntos difusos de la salida.

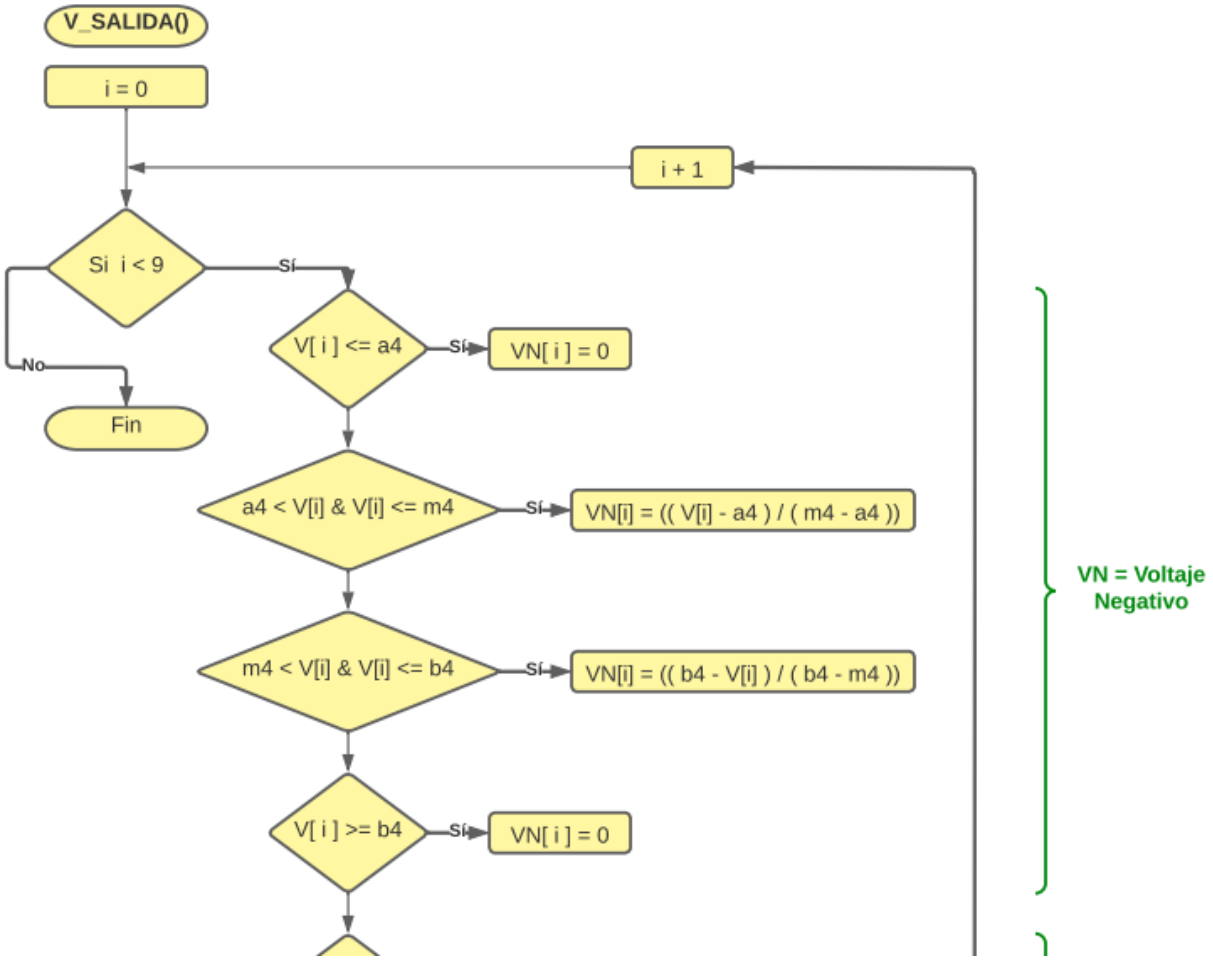

Figura 27: Parte superior del diagrama de flujo de la función V_SALIDA
Fuente: Creación propia

Para la parte del **void loop** se comienzo ejecutando el funcionamiento del sensor en donde primero se desactiva **TRIG** por dos microsegundos y se reactiva por 10 para volver a desactivarse. Con la función **pulseIn()** se hace un conteo del tiempo en que **ECO** esta activo y se asigna el valor en **DURACION**. Se reasigna el valor nuevamente, esta vez a D (Distancia actual).

Una vez teniendo el valor de D es sencillo definir ED restando DD a D. Es importante aclarar que el dato DD se asigna desde el principio.

Los datos son llamados a imprimir en el Display LCD. Esto es importante porque de esa forma se puede observar el proceso que lleva a cabo el programa mostrando los datos más importantes, los cuales son D, DD, ED y vo.

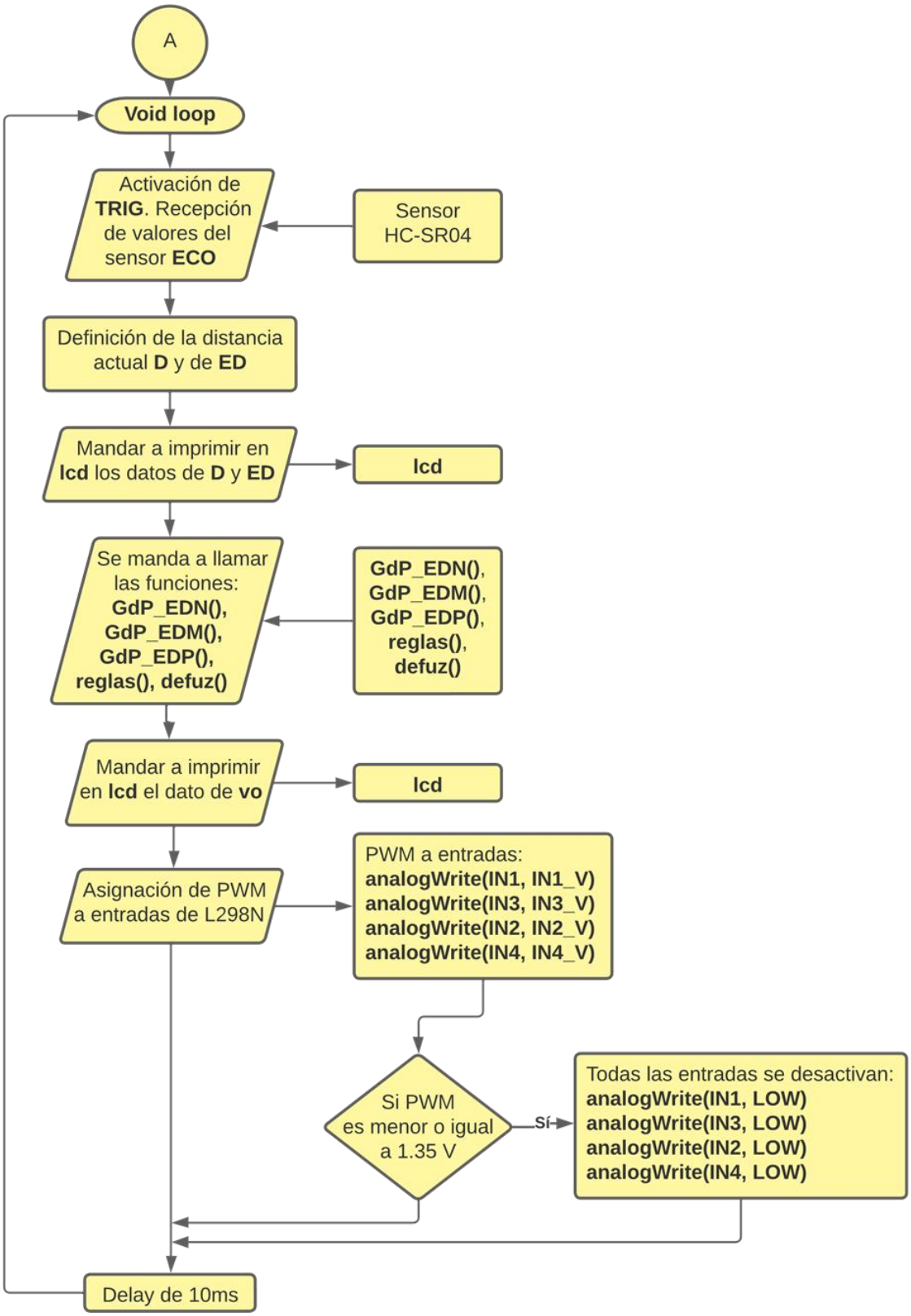

Figura 28: Diagrama de flujo 2 del primer controlador
Fuente: Creación propia

Por su parte las funciones **GdP_EDN()**, **GdP_EDM()** y **GdP_EDP()** son conjuntos difusos dentro de funciones **void** y cada una evalúa los datos de la función de membresía del conjunto difuso al que representa por ejemplo **GdP_EDN** representa a **EDN** (Error de la Distancia Negativo).

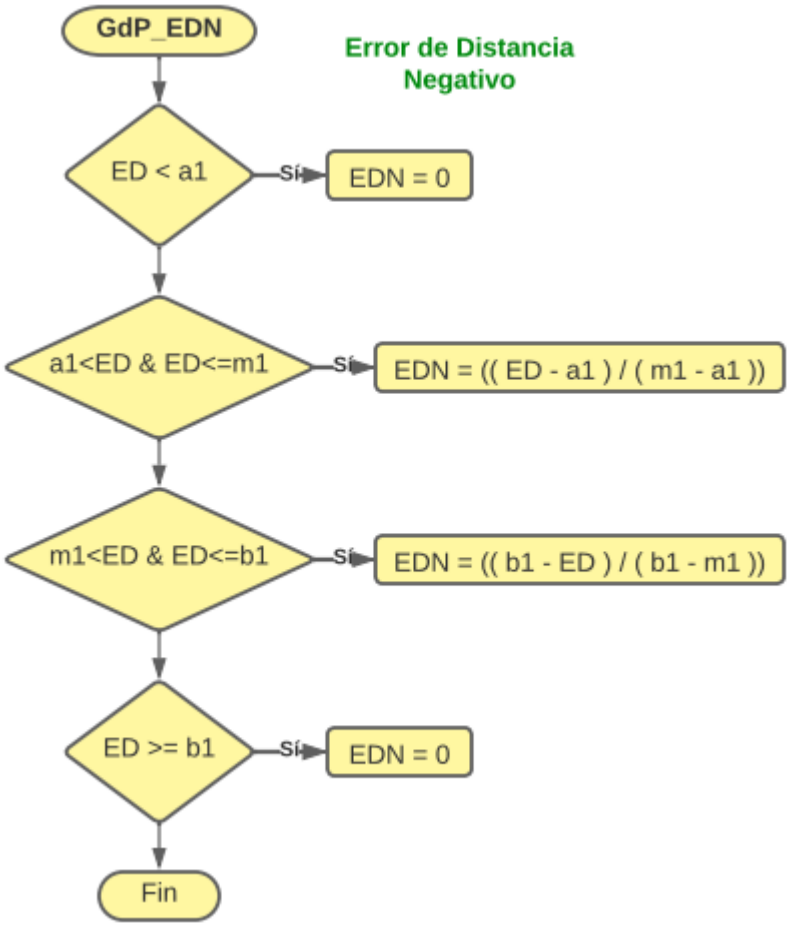

Figura 29: Diagrama de flujo de GdP_EDN
Fuente: Creación propia

Finalmente las funciones **reglas()** y **defuz()** se encargaron del método de inferencia y la defusificación respectivamente. En el proceso de inferencia de Mamdani, a través de un ciclo for se evalúan ocho veces las tres reglas, obteniendo el mínimo entre los conjuntos de entrada y salida para después conseguir el máximo de todo y finalmente, el valor es guardado en **T** siguiendo el diseño de MATLB. Para la defusificación se obtuvo el valor de **vo** y se asignó a los PWM de salida.

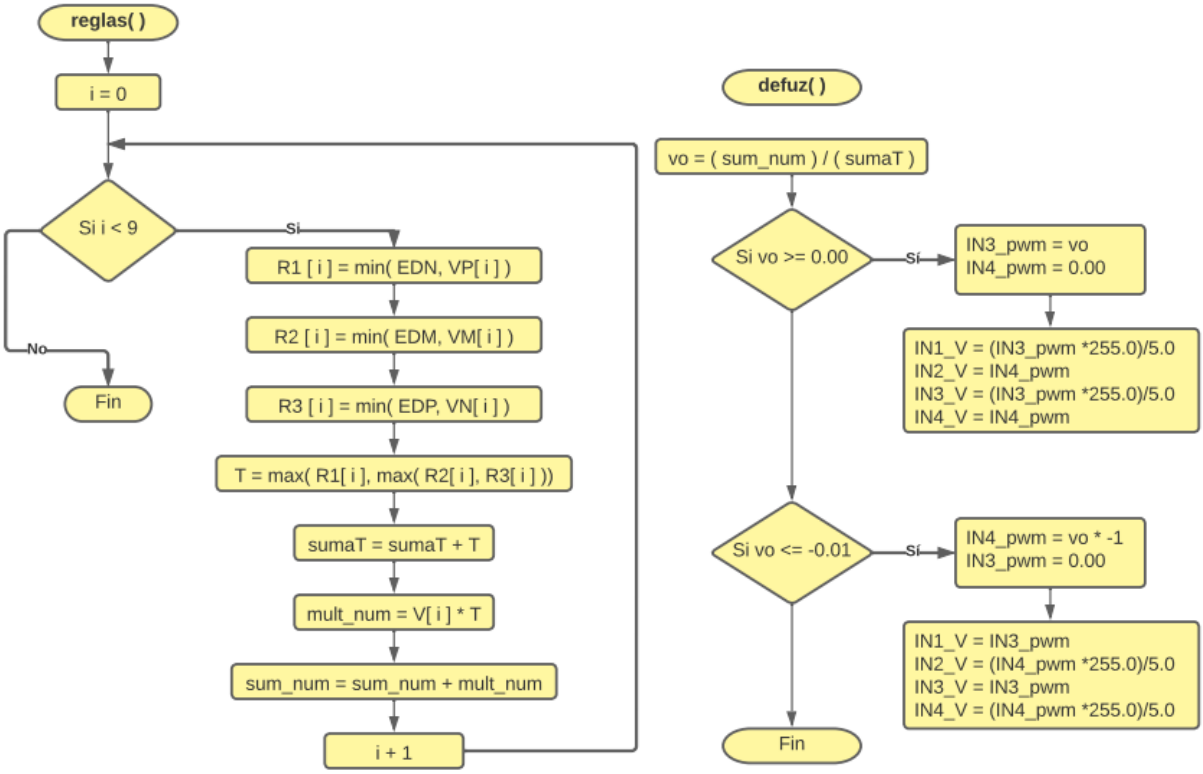

Figura 30: Diagramas de flujo de las funciones reglas() y defuz() del primer controlador
Fuente: Creación propia

Los PWM fueron asignados a las entradas del módulo L298N, respetando las condiciones que indicaban la dirección de giro de los motores, especificando que, cuando el voltaje final es menor a 1.35V se deben detener, esto tomando en cuenta que con un voltaje menor los motores no funcionan por lo que solo se activan cuando el voltaje es funcional.

Se realizaron pruebas con los motores sin llantas para ver su desempeño. A pesar de que el voltaje que asignaba el programa era equivalente en todos los pines de la placa Arduino UNO, el módulo L298N no distribuía la tensión al mismo tiempo. Esta es otra razón por la que se condiciono al control a operar cuando, el valor de **vo** dentro el control, fuera superior a 1.35V.

Al final el control permitió al carro avanzar cuando la distancia entre un obstáculo y el sensor era más grande que la deseada. Así mismo retroceder cuando el obstáculo se encontraba más cerca.

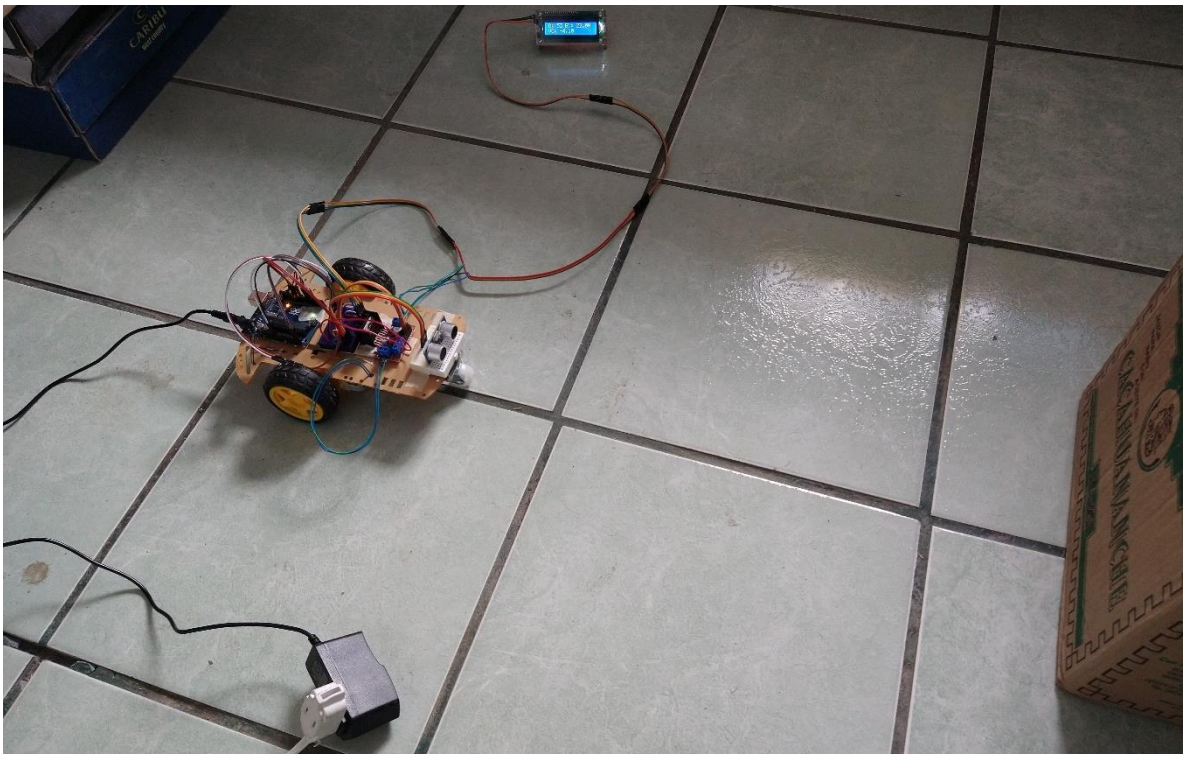

Figura 31: Pruebas con el prototipo 1
Fuente: Captura propia

4.1.4 Fallos del prototipo 1

Un inconveniente a la hora de probar el prototipo fue que la respuesta se alentaba con respecto al tiempo transcurrido, lo cual era más evidente cuando se pasaba rápidamente de un valor positivo a uno negativo en los valores de ED. También sucedía que, al ser probado varias veces sin reiniciarlo, disminuía la respuesta.

La precisión también era un fallo común debido a que esta variaba de 10cm o más, debido a que existe una zona entre los voltajes -8 y 8 en la que los motores no funcionan.

Otro inconveniente es que debido al diseño de dos motorreductores y una sola llanta móvil de soporte, esta última podía cambiar la dirección del carro al girar con brusquedad y alterar completamente la respuesta del control al modificar la dirección donde apunta el sensor.

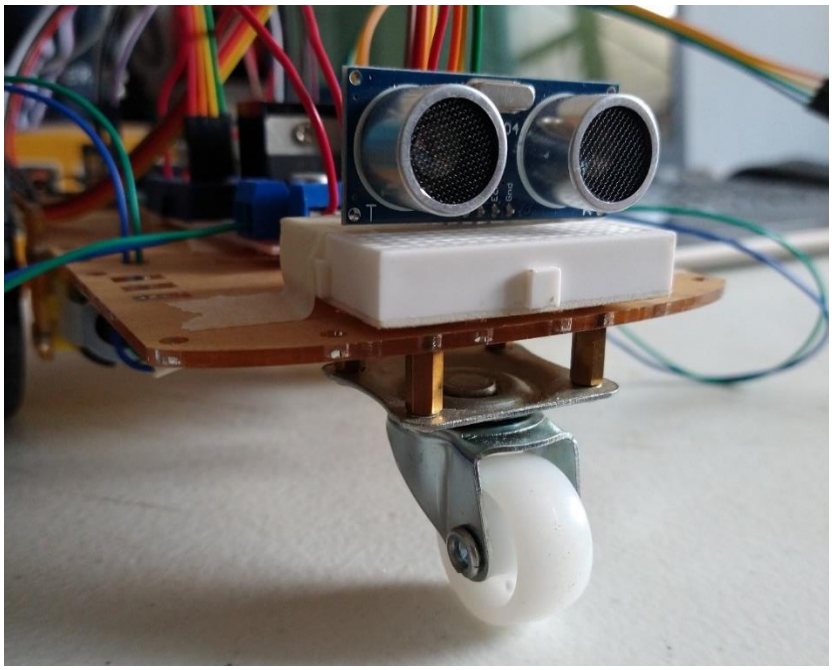

Figura 32: Llanta giratoria del prototipo 1
Fuente: Captura propia

El último fallo, es con respecto al módulo L298N, el cual no distribuye equitativamente la tensión en las salidas hacia los motores, razón por la cual en voltajes menores un motor responde antes que el otro con una diferencia de tensión de 4.04V de uno con respecto a otro.

Figura 33: Voltaje de L298N prototipo 1
Fuente: Captura propia

4.2 Segundo prototipo

El primer cambio que se realizo fue físico, buscando cambiar la llanta giratoria que presentaba inconvenientes al alterar la dirección del carro involuntariamente. Para resolver esto se escogió una “rueda loca” que no presenta dirección y por el contrario se somete al camino señalado por las ruedas de los motores.

4.2.1 Distancia deseable con bluetooth

El segundo cambio que se realizo fue permitir modificar el dato de la distancia deseable (DD) con intención de poder escoger, en tiempo real, la distancia a la cual mantener el carro, reajustando el código al momento. Para lograr esto se optó por modificarlo a distancia a través de conexión bluetooth, volviendo más practico el proceso al no añadir cables ni estar cargando el código a la placa a cada momento.

El añadido fue el módulo Bluetooth HC-06 el cual permitía una conexión con la placa de Arduino UNO a otro dispositivo con la conexión bluetooth. Para esto se desarrolló una aplicación en el entorno de MIT App Inventor, para teléfono Android.

Figura 34: Aplicación móvil de MIT App Inventor
Fuente: Captura propia

La App permite mandar un dato tipo **string** al establecer una conexión bluetooth con el destino. Su programación es intuitiva al utilizar bloques que funcionan como condicionantes y declaraciones, permitiendo establecer que sucede cuando se selecciona un bloque y como se ejecuta.

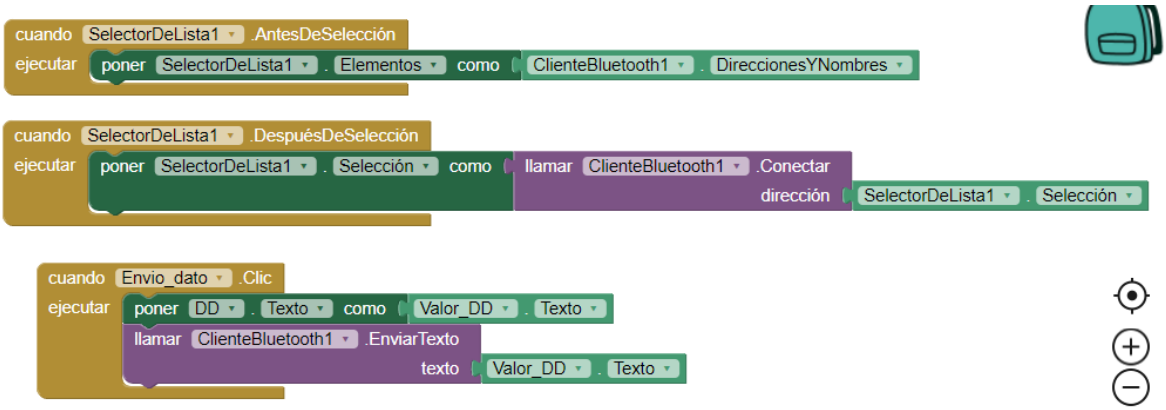

Figura 35: Programación tipo bloques de MIT App Inventor
Fuente: Captura propia de MIT App Inventor

Para incluir el módulo HC-06 en el programa se abrió una nueva conexión serial con la librería **SoftwareSerial.h**, esta fue llamada “bluetooth” y se definieron los pines 12 y 13 como RX y TX respectivamente. Esto se hace con la intención de dejar libre la conexión serial por defecto en los pines 0 y 1, tomando en cuenta que es importante tenerlos libres cuando se compila el programa.

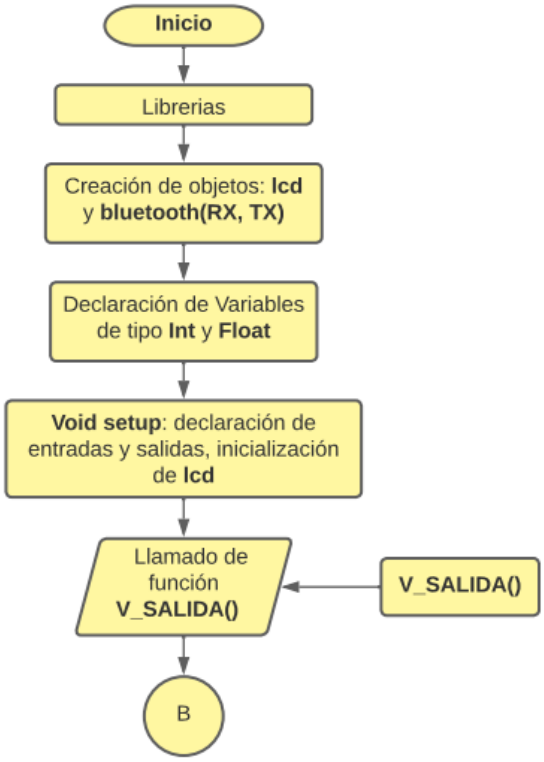

Figura 36: Diagrama de flujo 1 del segundo controlador
Fuente: Creación propia

Para verificar la conexión entre el módulo y el teléfono celular se utilizó la función **bluetooth.available(),** la cual es condicionada dentro de un IF en donde se establece que el valor proporcionado es mayor a cero. Esto se hace debido a que la función es necesaria para mantener el dato proporcionado dentro del ciclo mientras no sea modificado.

Al recibir el dato por parte del teléfono Android fue requerida la función **parseFloat()**, la cual hace la conversión del dato **string** a **float** (flotante). Adicionalmente se asigna el dato a DD, así como se manda a HIGH la variable STBY.

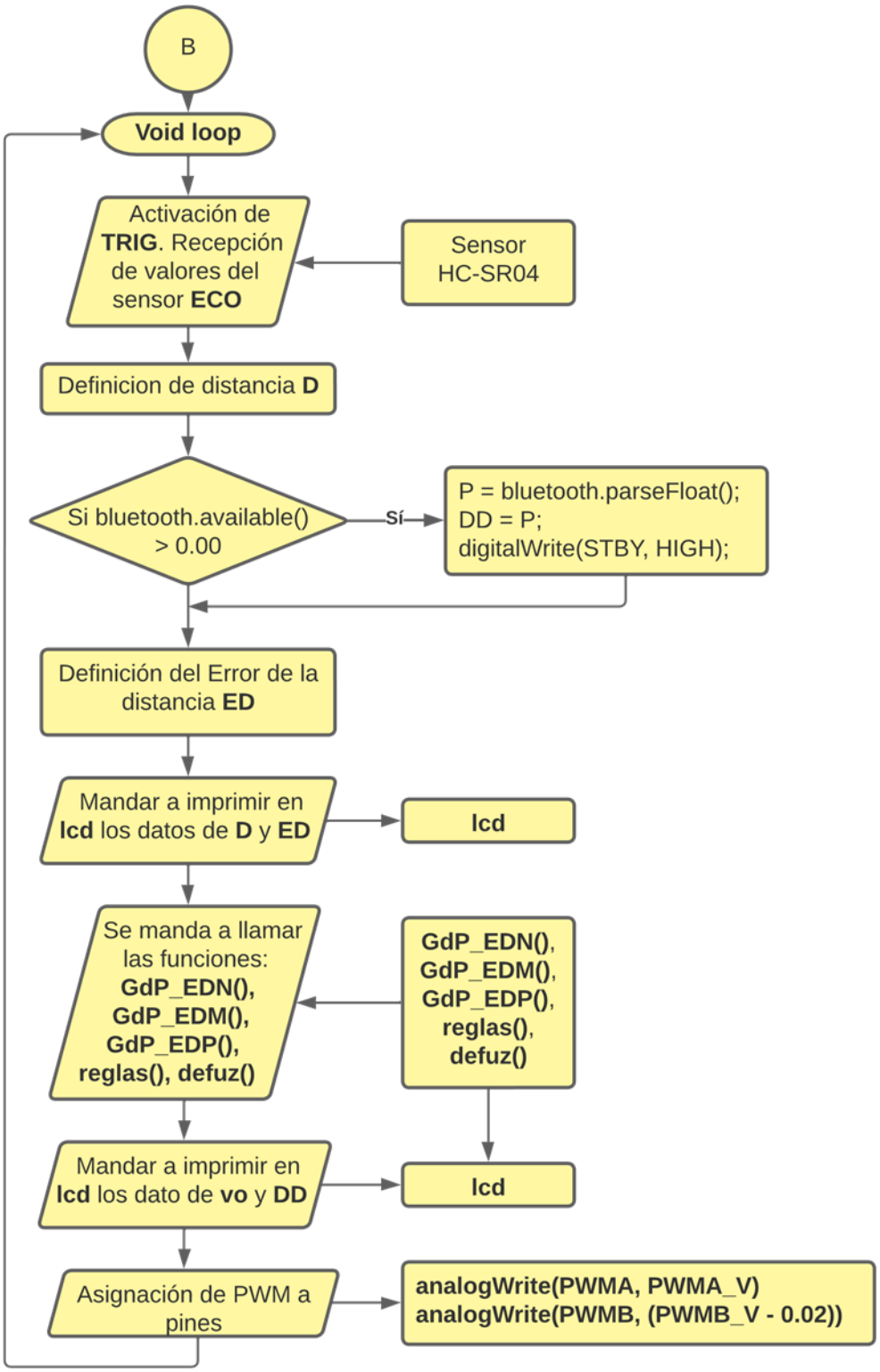

Figura 37: Diagrama de flujo 2 del segundo controlador
Fuente: Creación propia

4.2.2 Modulo TB6612FNG

Como tercer cambio importante tenemos el reemplazo del módulo L298N, el cual, en pruebas sin motores, mandaba una diferencia de voltaje bastante amplia en las terminales del módulo, la cual era más evidente cuando el PWM era de un valor bajo y la diferencia disminuía al subir el valor.

Para esto se seleccionó el módulo puente H TB6612FNG, el cual se presenta como una alternativa actualizada al L298N, teniendo en su composición transistores MOSFET en lugar de los BJT. En cuanto al voltaje de salida del módulo se presentó mucho más equilibrado mostrando pequeñas diferencias de aproximadamente 0.02V.

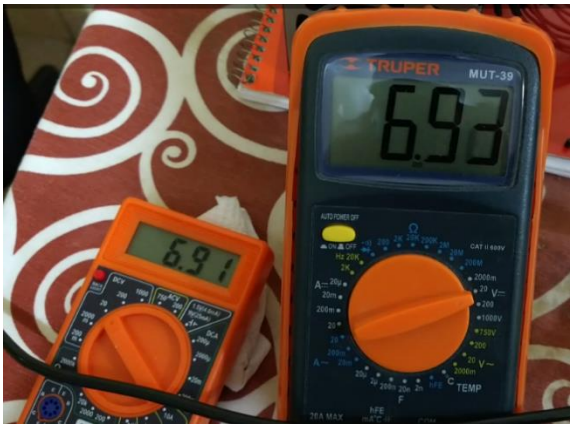

Figura 38: Voltaje de TB6612FNG en el programa 2
Fuente: Captura propia

Para introducirlo al código principal se modificaron las variables de entrada a los motores, añadiendo tres de estas, dos que corresponden a los PWM y una que corresponde a STBY. También se declararon en sus respectivos pines de salida.

En cuanto a su funcionamiento dentro del programa, el voltaje de salida (**vo**) es asignado a las variables de **PWMA_V** y **PWMB_V** dentro de condicionales IF en donde se define de acuerdo con si el voltaje final es positivo o negativo. También se estableció el sentido de giro del motor poniendo en alto (HIGH) y en bajo (LOW) las entradas de cada motor (AIN1, AIN2, BIN1 y BIN2) dentro de cada IF.

El ultimo cambio fue quitar los condicionales IF en donde se desactivaban los motores a un cierto voltaje de salida, esto debido a que se hicieron cambios en los valores de los conjuntos de las variables de entrada y salida. Estos cambios permitieron mejorar la precisión y disminuir la zona en donde el voltaje no era funcional.

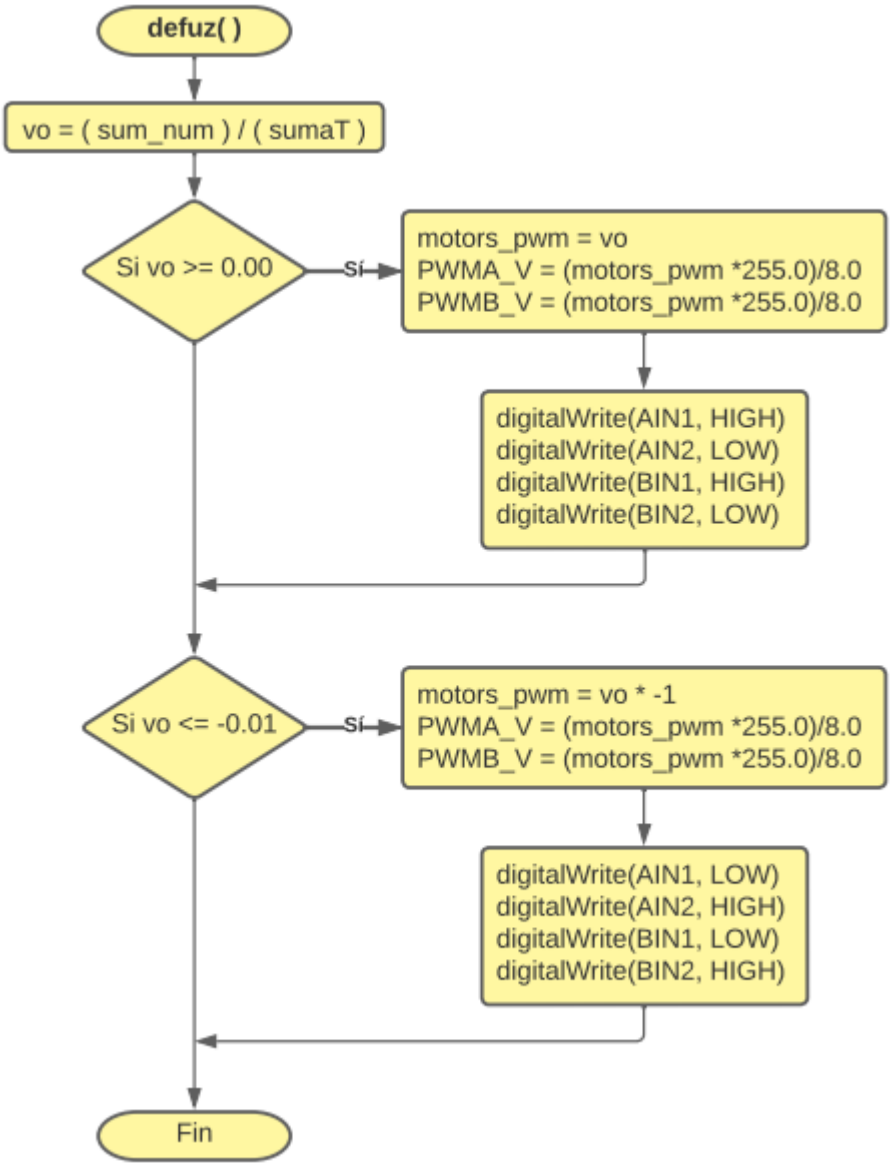

Figura 39: Diagrama de flujo de la función defuz() del segundo controlador
Fuente: Creación propia

Finalmente, los PWM son mandados a las salidas, restándole al motor B, 0.02V que es la pequeña diferencia de voltajes (véase figura 37).

4.2.3 Cambio en función reglas()

Como cuarto cambio se eliminó el ciclo For dentro de la función **reglas()**. Este se encargaba de evaluar y obtener los mínimos, para después, obtener los máximos de cada grupo de reglas con respecto a cada dato del vector **V[ ]** y finalmente sumar todo.

Este cambio es debido a que retrasaba el procesado en el tiempo de ejecución del programa lo que retardaba la respuesta y reacción del carro.

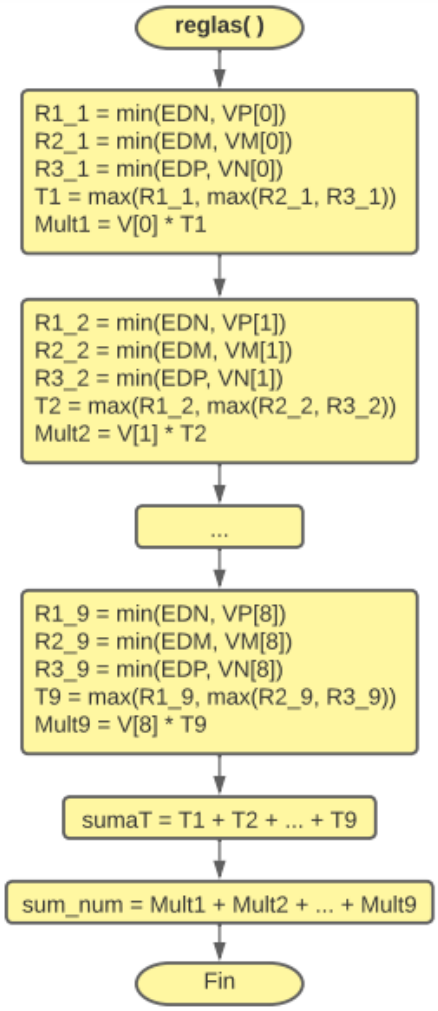

Figura 40: Diagrama de flujo de función reglas() del segundo controlador
Fuente: Creación propia

En lugar del ciclo For, se optó por ejecutar el proceso de cada regla y al final sumar cada resultado. Esto realmente ayudo bastante en mejorar la respuesta del programa.

4.2.4 Cambios en conjuntos difusos de ED y V

Finalmente, un cambio importante se realizó en los valores que conformaban las funciones de membresía de los conjuntos difusos de ED y V (las variables de entrada y salida). Esto con la intención de suavizar la respuesta del Voltaje de salida, debido a que el tiempo de reacción del programa era bastante rápido, lo cual provocaba que la respuesta del sensor tardara en llegar, causando algunos impactos en el prototipo al hacer pruebas.

El sensor Ultrasónico posee un tiempo de respuesta variante, esto debido a la distancia que tiene que recorrer la onda que envía **TRIG**. Esto quiere decir que, a una mayor distancia, el tiempo para recibir el dato es ligeramente mayor.

Tras esto se optó por aumentar la respuesta del voltaje de salida, para reaccionar de acuerdo con el sensor y a su vez para disminuir la zona del voltaje en donde el motor no funciona. Para esto es necesario mantener el voltaje mínimo mayor a 1.2V.

Los nuevos valores serían los siguientes:

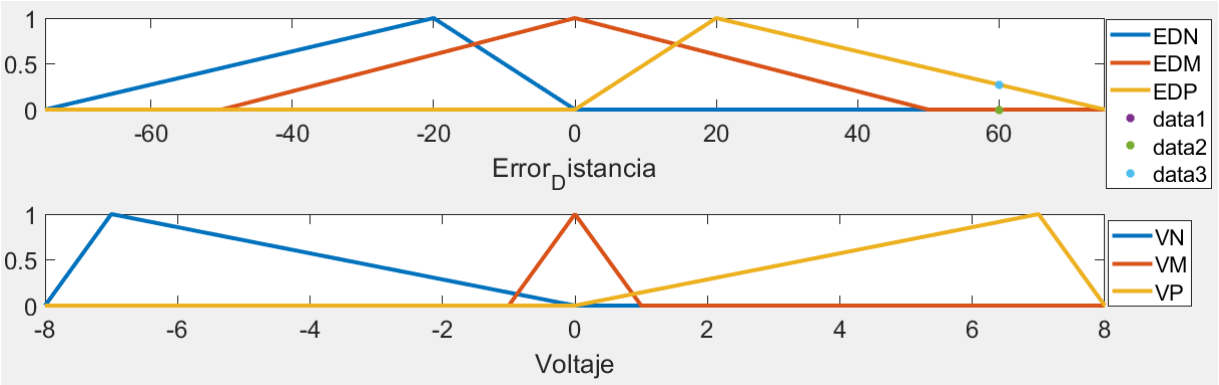

Figura 41: Graficas de los conjuntos difusos de ED y V en el controlador 2
Fuente: Creación propia Matlab

4.3 Resultados

Durante este trabajo se buscó que el segundo prototipo fuera más compacto, cargando con el Display LCD a un costado y montando cables más cortos para no ser tan voluminoso. Además, se propuso que funcione con tres baterías, una de 9V para

alimentar el Arduino UNO y las otras dos recargables de 3.7V cada una, para alimentar los motorreductores. De esta forma se consiguió una mejor autonomía.

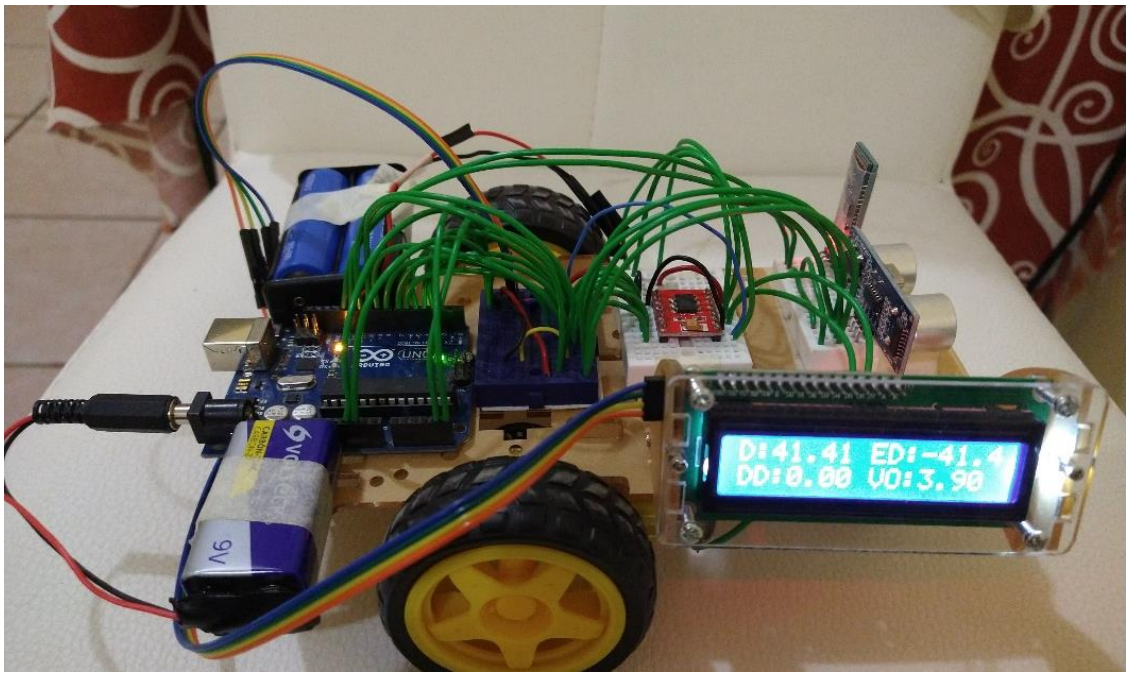

Figura 42: Prototipo 2
Fuente: Captura propia

En cuanto a las pruebas realizadas el carro respondió más rápido y, al pasar el tiempo, el programa se mantuvo con una respuesta optima a comparación del anterior, que cuando hacía un cambio rápido de distancia el controlador se tardaba en responder, provocando impactos.

El margen de error disminuyo satisfactoriamente llegando de entre 1cm a 0.25cm aproximadamente. Esto permitiendo llegar a la distancia deseada.

No obstante, un inconveniente importante es que el motor B sigue mostrando una respuesta un poco mayor, provocando un ligero cambio de dirección en el carro. Esto se debe posiblemente al módulo TB6612FNG.

El módulo TB6612FNG funciono mejor que el L298N, repartiendo de una mejor forma el voltaje en ambos Motorreductores, no obstante, al realizar algunas pruebas más se notó que existe un voltaje mínimo que llega al motor B aun cuando todas las entradas

están en bajo (LOW), lo que altera un poco el funcionamiento, es por esto por lo que se decidió incluir una resta de 0.02V al PWM del motor B, sin embargo, aún se nota el cambio de dirección.

La rueda loca resolvió el problema que presentaba la anterior. Esta ya no afecta ni altera la dirección del carro cuando retrocede, dependiendo únicamente de los motorreductores para la misma.

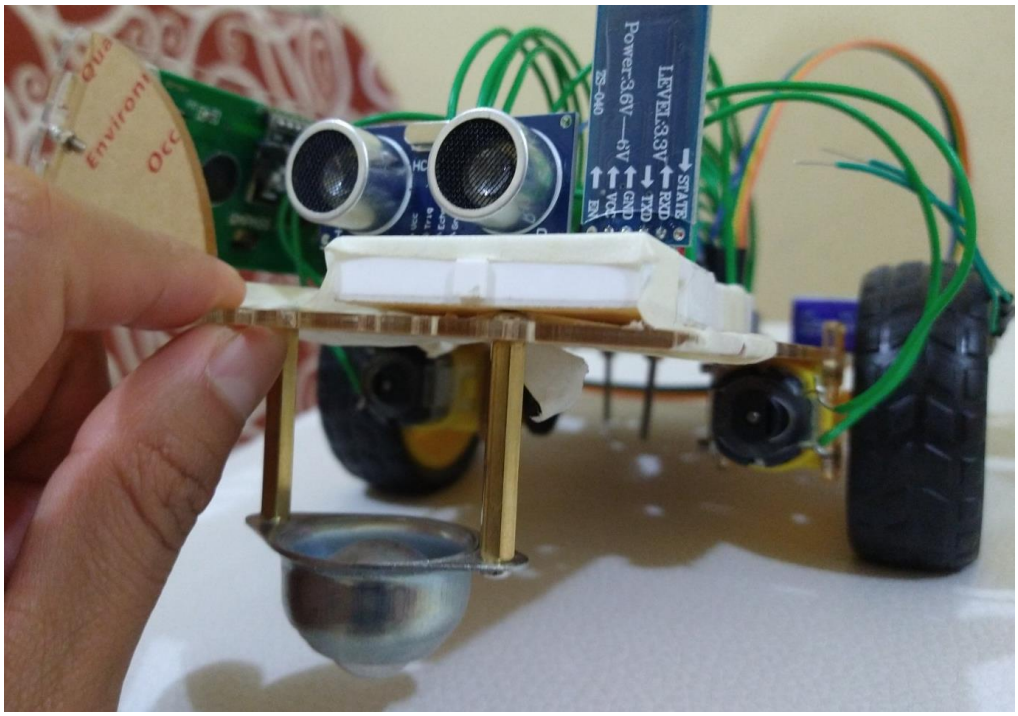

Figura 43: Carro con "rueda loca"
Fuente: Captura propia

Por parte del envió de datos por bluetooth funciono adecuadamente, permitiendo modificar la distancia en cualquier momento, además de iniciar el funcionamiento de los motorreductores en cuanto se introduce la distancia deseada gracias a la función STBY (véase, figura 37).

4.3.1 Pruebas con Conjuntos Difusos

Tras mejorar la respuesta del controlador, viviéndolo más ágil y con un tiempo de reacción decente, surgieron algunos inconvenientes, principalmente pequeños impactos al hacer las pruebas físicas. Para tratar de resolver esto se optó por probar cambiar los valores de los conjuntos difusos.

Lo que se pretendía era que la transición entre voltaje negativo a positivo fuera suave. Esto quiere decir que se mantuviera en un voltaje bajo mientras el carro buscaba llegar a la distancia deseada, pero a su vez, que el voltaje funcional se mantenga lo suficiente para llegar a la distancia deseada, es decir, que el margen de error sea mínimo.

Esto último es importante ya que el voltaje funcional es aquel que es capaz de hacer funcionar a los motores, puesto que cuando el voltaje es menor a 1.2V los motores están inactivos o no funcionan. En algunas combinaciones de conjuntos difusos, se alcanza el voltaje funcional cuando la diferencia entre D y DD es grande. Es por esto por lo que los conjuntos difusos deben poder permitir que el voltaje se mantenga arriba de 1.2V.

Error de Distancia	Voltaje
EDN [-75, -75, 0]	VN [-8, -8, 0]
EDM [-20, 0, 20]	VM [-2, 0, 2]
EDP [0, 75, 75]	VP [0, 8, 8]

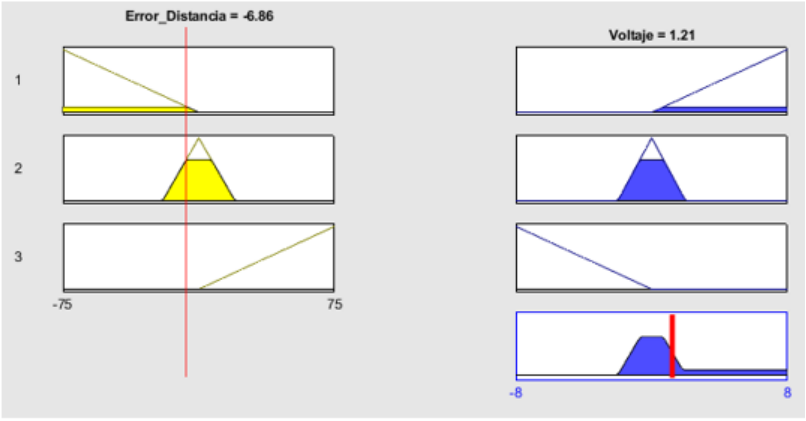

Figura 44: Prueba con simulación en Matlab con los valores anteriores
Fuente: Captura propia de Matlab Toolbox

Tras una serie de pruebas con varios valores se definieron los presentados en el controlador final debido a que, aunque no obtuvieron la respuesta más suave, permitieron un margen de error mínimo logrando la mejor precisión, disminuyendo esa zona no funcional en la que los motores no responden.

Las combinaciones se fueron probando y se recopilaron observaciones.

Error de Distancia	Voltaje	Respuesta
EDN [-75, -75, 0] EDM [-20, 0, 20] EDP [0, 75, 75]	VN [-8, -8, 0] VM [-2, 0, 2] VP [0, 8, 8]	Estos son los conjuntos del control anterior los cuales presentaban una respuesta rápida cuando la variable ED se acercaba a cero, pero más lenta cuando se acercaba a cualquier extremo ya sea positivo o negativo.
EDN [-75, -75, -50] EDM [-60, 0, 60] EDP [50, 75, 75]	VN [-8, -8, 0] VM [-2, 0, 2] VP [0, 8, 8]	Con esta combinación la respuesta es muy lenta, llegando a permitir un voltaje funcional hasta casi los extremos de ED, es decir con una diferencia de D y DD muy grande.
EDN [-75, -75, -50] EDM [-60, 0, 60] EDP [50, 75, 75]	VN [-8, -8, -5] VM [-6, 0, 6] VP [5, 8, 8]	La respuesta se mantuvo lenta, permitiendo un voltaje funcional casi a los extremos de ED.
EDN [-75, -75, 0] EDM [-20, 0, 20] EDP [0, 75, 75]	VN [-8, -8, -5] VM [-6, 0, 6] VP [5, 8, 8]	Presenta un cambio drástico, cuando el valor está cerca de 20cm (positivo o negativo) a 0 es suave, cuando pasa ese límite la respuesta crece bastante.
EDN [-75, -75, 0] EDM [-50, 0, 50] EDP [0, 75, 75]	VN [-8, -8, -5] VM [-6, 0, 6] VP [5, 8, 8]	La respuesta es suave e ideal, sin embargo, el voltaje funcional se alcanza a la mitad de ED, es decir por los 37cm aproximadamente.
EDN [-75, -75, 0] EDM [-50, 0, 50] EDP [0, 75, 75]	VN [-8, -8, 0] VM [-6, 0, 6] VP [0, 8, 8]	La respuesta se mantiene suave, pero de igual forma el voltaje funcional se alcanza a la mitad de ED. Es decir, el margen de error sigue siendo alto

EDN [-75, -75, 0] EDM [-40, 0, 40] EDP [0, 75, 75]	VN [-8, -8, 0] VM [-2, 0, 2] VP [0, 8, 8]	Para esta prueba se obtienen voltajes funcionales de una manera rápida y la respuesta es un poco más suave a comparación del control anterior, pero el controlador tarda en llegar a la DD y el margen de error es de 10cm.
EDN [-75, -75, 0] EDM [-40, 0, 40] EDP [0, 75, 75]	VN [-8, -6, 0] VM [-2, 0, 2] VP [0, 6, 8]	En esta prueba se obtuvo una respuesta más suave a comparación lo visto anteriormente, pero el controlador tarda en llegar a DD. El margen de error es el mismo.
EDN [-75, -70, 0] EDM [-40, 0, 40] EDP [0, 70, 75]	VN [-8, -6, 0] VM [-2, 0, 2] VP [0, 6, 8]	En esta prueba se obtuvo una respuesta similar a la anterior con el mismo margen de error.
EDN [-75, -70, 0] EDM [-50, 0, 50] EDP [0, 70, 75]	VN [-8, -6, 0] VM [-2, 0, 2] VP [0, 6, 8]	La prueba resulto mejor, con buena respuesta y obteniendo a buen tiempo el voltaje funcional. El margen de error es de 7cm.
EDN [-75, -70, 0] EDM [-60, 0, 60] EDP [0, 70, 75]	VN [-8, -6, 0] VM [-2, 0, 2] VP [0, 6, 8]	La respuesta resulto buena, sin embargo, el voltaje funcional se recorre de tal forma que inicia cuando la distancia deseada es más lejana, es decir, cuando hay una diferencia grande entre DD y D.
EDN [-75, -40, 0] EDM [-60, 0, 60] EDP [0, 40, 75]	VN [-8, -7, 0] VM [-1, 0, 1] VP [0, 7, 8]	La respuesta fue buena y redujo bastante el margen de error a 5cm, pero comenzó a ser un poco menos suave.

EDN [-75, -20, 0] EDM [-50, 0, 50] EDP [0, 20, 75]	VN [-8, -7, 0] VM [-1, 0, 1] VP [0, 7, 8]	La respuesta fue rápida. El margen de error se disminuyó entre 1cm y 0.25cm aproximadamente. Fueron los valores seleccionados debido a que la precisión aumento mucho y se acercaba a la distancia deseada.

Tabla 4: Comparación de valores en conjuntos difusos
Fuente: Propia

Las pruebas fueron bastante útiles porque permitían observar el comportamiento de las reglas establecidas y aunque el resultado final no fue muy suave, si fue más preciso en comparación de los conjuntos anteriores logrando disminuir el margen de error, aproximándose a la distancia deseada.

5. Conclusiones y trabajo a futuro

5.1 Conclusión

A lo largo de este proyecto se buscó siempre tratar de probar la hipótesis, sobre poder implementar y mejorar un controlador difuso de distancia en un carro eléctrico a escala y en su mayoría se logró. Se pudo demostrar que con el tiempo suficiente se pueden solucionar los fallos y lograr un mejor prototipo.

Durante el trabajo se mostró que existen muchas herramientas para poder simplificar procesos que se realizan en un sistema, la principal fue el intercambio de datos por conexión Bluetooth, permitiendo modificar la distancia deseada en tiempo real, lo que ahorraba bastante tiempo al momento de hacer pruebas físicas, evitando tener que cargar el programa en la placa de Arduino cada vez que se necesitara modificar la distancia deseada.

Así mismo es necesario notar que los módulos encargados de manejar los voltajes hacia los motores presentan ciertos desfases de tensión que, aunque lleguen a ser pequeños, se llegan a notar en las pruebas físicas. No obstante, es importante señalar que el módulo TB6612FNG es mejor que el L298N, presentando una diferencia de voltaje mucho menor a comparación, además del modo standby que ayuda a iniciar el funcionamiento cuando se introduce la Distancia Deseada.

Otro cambio importante para notar es que los ciclos For, al hacer mucho procesamiento dentro de un mismo proceso, ralentiza todo el programa, siendo esta la razón por la cual el anterior controlador disminuía su respuesta al pasar el tiempo.

Los conjuntos difusos por su parte han significado cambios importantes, comprendiendo que la respuesta del controlador depende mucho de estos, siendo que lo más importante que se busca es una mayor aproximación de la distancia actual (D) con la distancia deseada (DD), es decir una mejor precisión. Para lograr esto fue necesario disminuir la zona de voltaje no funcional modificando los valores de los

conjuntos difusos. En estas pruebas se observó que, al acercar los picos triangulares de las funciones de membresía de ED a cero, el voltaje aumentaba en un tiempo corto permitiendo a los motores operar un poco más cuando el margen de error disminuía. Esto al final permitió un margen de error de 1cm a 0.25cm logrando una buena precisión.

Finalmente, aunque no se puede afirmar que se pueda sustituir al control clásico, es mucho más sencillo modificar un control difuso si se comprenden y se establecen las reglas lingüísticas adecuadas. Con esto podemos considerar que los controladores difusos tienen bastante campo de exploración puesto que existen muchas áreas donde se pueden ajustar los parámetros más fácilmente de acuerdo con las necesidades humanas o las necesidades de la industria.

5.2 Resultados de la investigación

Por último, de este proyecto se obtiene un prototipo mejorado y un controlador difuso de distancia para Arduino en el cual se puede asignar una distancia deseada a través de una aplicación móvil.

5.2.1 Diagrama de conexión del segundo prototipo

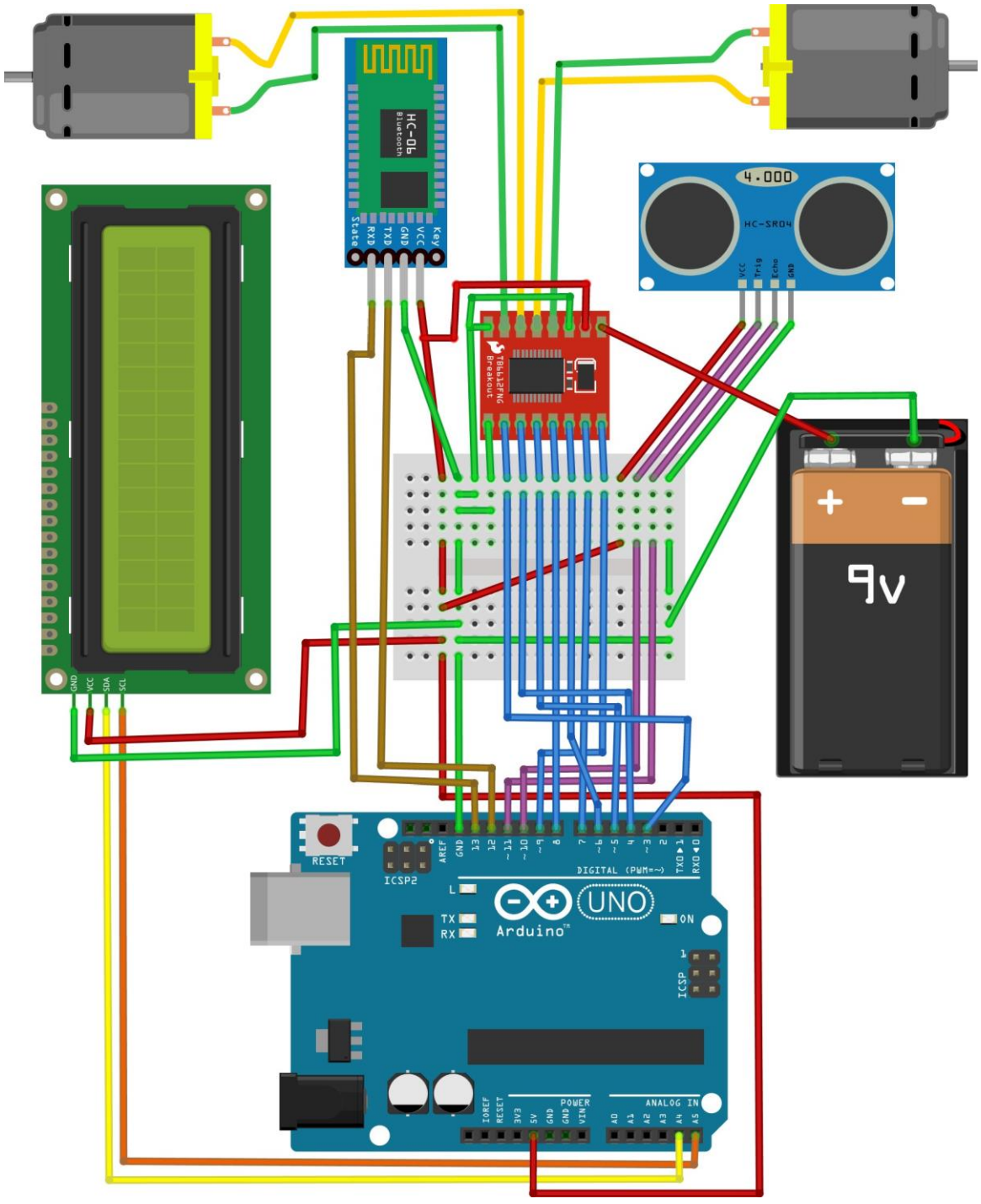

Figura 45: Esquema de conexión del prototipo 2
Fuente: Captura propia Fritzing

5.2.2 Código del segundo controlador (programa)

```
// Librerias
#include <SoftwareSerial.h>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

//se crean objetos, bluetooth y lcd
SoftwareSerial bluetooth(12,13); //RX,TX
LiquidCrystal_I2C lcd (0x27,2,1,0,4,5,6,7);

//Se Declaran Variables para Arduino
// Sensor ultrasonico
int TRIG = 10;
int ECO = 11;
int DURACION;
float DISTANCIA;

// Vector de Voltaje de salida
float V[] = {-8,-6,-4,-2,0,2,4,6,8};
// Conjuntos Difusos de V
float VN[9]={};
float VM[9]={};
float VP[9]={};

//Variables del modulo TB6612FNG
float motors_pwm;
float PWMA_V;
float PWMB_V;
int STBY = 8;
int AIN2 = 7;
int AIN1 = 6;
int BIN1 = 5;
int BIN2 = 4;
int PWMA = 9;
int PWMB = 3;

// Variables para conjuntos de entrada ED
float D;  // valor de distancia actual
float ED; // Error de distancia
float DD; // Distancia Deseable
float P;

//Variables para los conjuntos difusos (Distar
float EDN;
float EDM;
float EDP;

//valores de los conjuntos de ED
float a1 = -75; //Error Distancia Ne
float m1 = -20;
float b1 = 0;

float a2 = -50; //Error Distancia Me
float m2 = 0;
float b2 = 50;

float a3 = 0;   //Error Distancia Po
float m3 = 20;
float b3 = 75;

//valores de los conjuntos del Volta
float a4 = -8;  //Voltaje Negativo
float m4 = -7;
float b4 = 0;

float a5 = -1;  //Voltaje Medio
float m5 = 0;
float b5 = 1;

float a6 = 0;   //Voltaje Positivo
float m6 = 7;
float b6 = 8;

//REGLAS
float R1_1; //1
float R2_1;
float R3_1;

float R1_2; //2
float R2_2;
float R3_2;

float R1_3; //3
float R2_3;
float R3_3;

float R1_4; //4
float R2_4;
float R3_4;
```

```
float R1_5; //5
float R2_5;
float R3_5;

float R1_6; //6
float R2_6;
float R3_6;

float R1_7; //7
float R2_7;
float R3_7;

float R1_8; //8
float R2_8;
float R3_8;

float R1_9; //9
float R2_9;
float R3_9;

// Varibles extra
int i;
float T1;
float T2;
float T3;
float T4;
float T5;
float T6;
float T7;
float T8;
float T9;
float sumaT;
float vo;     //Voltaje de salida
float Mult1;
float Mult2;
float Mult3;
float Mult4;
float Mult5;
float Mult6;
float Mult7;
float Mult8;
float Mult9;
float sum_num;
```

```
void setup()
{
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);
  bluetooth.begin(9600);
//  Serial.begin(9600);
  lcd.setBacklightPin(3,POSITIVE);  //
  lcd.setBacklight(HIGH);   //habilita
  lcd.begin(16, 2);     //16 columnas
  lcd.clear();      // limpia pantalla
  pinMode(STBY, OUTPUT);
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);
  pinMode(PWMB, OUTPUT);
  V_SALIDA(); //llama a funcion voltaj
}



void loop()
{
  // Comunicacion con Sensor HC-SR04
  digitalWrite(TRIG, LOW);  // desacti
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH); // activa
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);  // desacti
  // la funcion pulseIn cuenta el tiem
  DURACION = pulseIn(ECO, HIGH); // as
  DISTANCIA = (DURACION * 0.034)/2; //
  D = DISTANCIA;

// Recepcion de datos Bluetooth
  if (bluetooth.available() > 0){
    P = bluetooth.parseFloat();
    DD = P;
    digitalWrite(STBY, HIGH);
  }

  ED = DD - D;
```

```
  //Manda a imprimir D y ED en LCD
  lcd.setCursor(0, 0);
  lcd.print("D:");
  lcd.print(DISTANCIA);
  lcd.print(" ED:");
  lcd.print(ED);

  // funciones EDN, EDM, EDP, reglas y defuz
  GdP_EDN();
  GdP_EDM();
  GdP_EDP();

  reglas(); // Metodo de inferencia, fusificaci
  defuz(); // defusificacion y señal de salida

  // imprime en pantalla vo y DD
  lcd.setCursor(0, 1);
  lcd.print("DD:");
  lcd.print(DD);
  lcd.print(" VO:");
  lcd.print(vo);

  // manda el valor del PWM a PWMA y PWMB
  analogWrite(PWMA, PWMA_V);
  analogWrite(PWMB, (PWMB_V - 0.02));
}

  // Funciones de ED
void GdP_EDN() {
  if (ED<a1){
    EDN = 0;
  }
  if (a1<ED && ED<=m1){
    EDN = ((ED-a1)/(m1-a1));
  }
  if (m1<ED && ED<=b1){
    EDN = ((b1-ED)/(b1-m1));
  }
  if (ED>=b1){
    EDN = 0;
  }
}

void GdP_EDM() {
  if (ED<a2){
    EDM = 0;
  }
  if (a2<ED && ED<=m2){
    EDM = ((ED-a2)/(m2-a2));
  }
  if (m2<ED && ED<=b2){
    EDM = ((b2-ED)/(b2-m2));
  }
  if (ED>=b2){
    EDM = 0;
  }
}

void GdP_EDP() {
  if (ED<a3){
    EDP = 0;
  }
  if (a3<ED && ED<=m3){
    EDP = ((ED-a3)/(m3-a3));
  }
  if (m3<ED && ED<=b3){
    EDP = ((b3-ED)/(b3-m3));
  }
  if (ED>=b3){
    EDP = 0;
  }
}

void V_SALIDA(){
  for (int i = 0; i < 9; i = i + 1){
    // Grados de pertenencia de Volta
    // Voltaje Negativo
    if (V[i]<=a4){
      VN[i] = 0;
    }
    if (a4<V[i] && V[i]<=m4){
      VN[i] = ((V[i]-a4)/(m4-a4));
    }
    if (m4<V[i] && V[i]<=b4){
      VN[i] = ((b4-V[i])/(b4-m4));
    }
    if (V[i]>=b4){
      VN[i] = 0;
    }
```

```
    // Voltaje Medio
    if (V[i]<=a5){
      VM[i] = 0;
    }
    if (a5<V[i] && V[i]<=m5){
      VM[i] = ((V[i]-a5)/(m5-a5));
    }
    if (m5<V[i] && V[i]<=b5){
      VM[i] = ((b5-V[i])/(b5-m5));
    }
    if (V[i]>=b5){
      VM[i] = 0;
    }

    // Voltaje Positivo
    if (V[i]<=a6){
      VP[i] = 0;
    }
    if (a6<V[i] && V[i]<=m6){
      VP[i] = ((V[i]-a6)/(m6-a6));
    }
    if (m6<V[i] && V[i]<=b6){
      VP[i] = ((b6-V[i])/(b6-m6));
    }
    if (V[i]>=b6){
      VP[i] = 0;
    }
  }
}

void reglas(){
    //Metodo de inferencia o fuzzificacion
    R1_1 = min(EDN, VP[0]);
    R2_1 = min(EDM, VM[0]);
    R3_1 = min(EDP, VN[0]);

    T1 = max(R1_1, max(R2_1, R3_1));
    Mult1 = V[0] * T1;

    R1_2 = min(EDN, VP[1]);
    R2_2 = min(EDM, VM[1]);
    R3_2 = min(EDP, VN[1]);

    T2 = max(R1_2, max(R2_2, R3_2));
    Mult2 = V[1] * T2;

    R1_3 = min(EDN, VP[2]);
    R2_3 = min(EDM, VM[2]);
    R3_3 = min(EDP, VN[2]);

    T3 = max(R1_3, max(R2_3, R3_3));
    Mult3 = V[2] * T3;

    R1_4 = min(EDN, VP[3]);
    R2_4 = min(EDM, VM[3]);
    R3_4 = min(EDP, VN[3]);

    T4 = max(R1_4, max(R2_4, R3_4));
    Mult4 = V[3] * T4;

    R1_5 = min(EDN, VP[4]);
    R2_5 = min(EDM, VM[4]);
    R3_5 = min(EDP, VN[4]);

    T5 = max(R1_5, max(R2_5, R3_5));
    Mult5 = V[4] * T5;

    R1_6 = min(EDN, VP[5]);
    R2_6 = min(EDM, VM[5]);
    R3_6 = min(EDP, VN[5]);

    T6 = max(R1_6, max(R2_6, R3_6));
    Mult6 = V[5] * T6;

    R1_7 = min(EDN, VP[6]);
    R2_7 = min(EDM, VM[6]);
    R3_7 = min(EDP, VN[6]);

    T7 = max(R1_7, max(R2_7, R3_7));
    Mult7 = V[6] * T7;

    R1_8 = min(EDN, VP[7]);
    R2_8 = min(EDM, VM[7]);
    R3_8 = min(EDP, VN[7]);

    T8 = max(R1_8, max(R2_8, R3_8));
    Mult8 = V[7] * T8;

    R1_9 = min(EDN, VP[8]);
    R2_9 = min(EDM, VM[8]);
    R3_9 = min(EDP, VN[8]);
```

```
    T9 = max(R1_9, max(R2_9, R3_9));
    Mult9 = V[8] * T9;

    sumaT = T1 + T2 + T3 + T4 + T5
    + T6 + T7 + T8 + T9;
    sum_num = Mult1 + Mult2 + Mult3
    + Mult4 + Mult5 + Mult6 + Mult7
    + Mult8 + Mult9;
}

void defuz(){
  vo = (sum_num)/(sumaT);
  // motor hacia adelante
  if (vo >= 0.00){
    motors_pwm = vo;
    PWMA_V = (motors_pwm *255.0)/8.0; //Conversion para 8V
    PWMB_V = (motors_pwm *255.0)/8.0; //Conversion para 8V
    digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);
    digitalWrite(BIN1, HIGH);
    digitalWrite(BIN2, LOW);
  }

  // motor hacia atras
  if (vo <= -0.01){
    motors_pwm = vo * -1;
    PWMA_V = (motors_pwm *255.0)/8.0; //Conversion para 8V
    PWMB_V = (motors_pwm *255.0)/8.0; //Conversion para 8V
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
    digitalWrite(BIN1, LOW);
    digitalWrite(BIN2, HIGH);
  }
}
```

5.3 Trabajo a Futuro

El proyecto aún tiene margen de mejora, en donde se podría sustituir el sensor, así como también añadir más conjuntos difusos buscando obtener una respuesta más suave o rápida según lo que se necesite. También se podrían agregar más sensores y variables para realizar un controlador complejo.

Los cambios pueden ser desde los conjuntos difusos hasta la potencia, tomando en cuenta que la corriente que entra a los motores es muy importante, puesto que, sin la corriente necesaria, aunque se tenga el voltaje correcto, los motores no funcionarán.

6. Bibliografía

[1] Andrés Rodríguez, J. C. (2015). Robot Oruga detector y esquivador de obstáculos con Fuzzy Logic I.A. *Envigado, Facultad de Ciencias e Ingeniería, Universidad de Boyacá* , 368-375.

[2] Antonio Souto Iglesias, J. L. (2013). *Curso Básico de Programación en MATLAB.* Madrid: EDITORIAL TÉBAR.

[3] Bermúdez, L. E. (2018). *SEGUIMIENTO DE RUTA MEDIANTE CONTROL DIFUSO PARA EL ESTACIONAMIENTO TIPO PARALELO DE UN VEHÍCULO AUTÓNOMO A ESCALA.* TIJUANA, B.C.: INSTITUTO POLITÉCNICO NACIONAL .

[4] Campos, E. V. (2018). *SIMULACIÓN DEL MOVIMIENTO DE UN VEHÍCULO TERRESTRE PARA TRAYECTORIAS PROGRAMADAS APLICANDO CONTROL DIFUSO.* Puebla, Puebla: Benemérita Universidad Autónoma de Puebla.

[5] D. Guzmán, V. M. (2006). LA LÓGICA DIFUSA EN INGENIERÍA: PRINCIPIOS, APLICACIONES Y FUTURO. *Ciencia y Tecnología* , 87 - 107.

[6] David Jesús Segura Cristino, H. M. (2019). Sistema de navegación y evasión de obstáculos aplicando un sistema de control difuso en una placa Arduino UNO. *Research in Computing Science* , 291 - 303.

[7] E. Mamdani, S. A. (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, 1-13.

[8] Eduardo Gómez Ramírez, J. C. (2017). *El algoritmo de sintonización simple de controladores difusos (ASSCD).* Ciudad de México: Parmenia.

[9] Enrique Onieva, V. M. (2010). Estimación de un Control Lateral Difuso de Vehículos . *Riai*, 91-98.

[10] Garzón, A., Angulo, M., & Domínguez, A. (2005). Control difuso de un motor de inducción . *APLICACIONES INDUSTRIALES*, 32-36.

[11] Juan Carlos García Infante, J. d. (2009). *Sistemas con Lógica Difusa.* México, DF: Instituto Politécnico Nacional.

[12] Leonel G. Corona Ramírez, G. S. (2014). *Sensores y actuadores: Aplicaciones con Arduino.* Mexico DF: GRUPO EDITORIAL PATRIA S.A. de C.V.

[13] López, P. P. (2016). *Robótica y domótica básica con Arduino .* Madrid, España: RA-MA, S.A. Editorial.

[14] Luciano Perez, R. P. (2020). *METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA.* Ituzaingó: Maipue.

[15] M. Brox, A. G.-S. (2005). CONTROLADOR DIFUSO PARA PROBLEMAS DE NAVEGACIÓN EN PRESENCIA DE OBSTÁCULOS FIJOS. *Instituto de Microelectrónica de Sevilla*, 6.

[16] Ma. Jesús García Domínguez, V. M. (2014). *Comentario de Textos .* Las Palmas de Gran Canaria: ULPGC.

[17] Mamdani, E. (1974). Application of Fuzzy Algorithms for Control of Simple Dynamic Plant. *Electrical Engineers, Proceedings of the Institution of* , 121.

[18] Menna, S. H. (2014). Heurísticas y Metodología de la Ciencia. *CIECAS-IPN*, 67-77.

[19] Noriega, Y. R. (2002). La Hermenéutica Aplicada a la Interpretación del Texto. El Uso de la Técnica del Análisis de Contenido . *Revista Ciencias de la Educación* , 9.

[20] Orduña, A. O. (2014). *Tesis: DESARROLLO E IMPLEMENTACIÓN DE SOFTWARE PARA MANTENER HORIZONTAL UNA PLATAFORMA ANTE MOVIMIENTOS EXTERNOS BASADOS EN LÓGICA DIFUSA.* Puebla: Benemérita Universidad Autónoma de Puebla: Facultad de Ciencias de la Electrónica.

[21] Palacios, M. M. (2021). *Control Difuso.* Puebla: BUAP.

[22] Panayotis Tremante, E. B. (2014). Una visión de la teoría difusa y los sistemas difusos enfocados al control difuso. *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, 121-136.

[23] Parra, H., G., L. H., & L., M. B. (2007). Navegación de Robots Móviles mediante comportamientos utilizando Lógica Difusa. *Scientia Et Technica*, 79-84.

[24] Pérez, E. d. (2008). *Control de Procesos: Implementación de una plataforma hardware/software para la experimentación en control digital directo: Controladores PID y Fuzzy.* Tarragona: Pueblicacions URV.

[25] R., H. D., M., R. B., & O., A. S. (1997). Diseño y Ensayo de un Controlador Difuso para un Motor de Inducción Trifasico. *Revista Facultad de Ingenieria, U.T.A. (CHILE)*, 19-25.

[26] Ramos, D. L. (2021). *Tesis: Control de un Robot Móvil para Mapeo con Lógica Difusa.* Puebla: BENEMERITA UNIVERSIDAD AUTÓNOMA DE PUEBLA: Facultad de Ciencias de la Computación.

[27] Romero, A. C. (2014). *Metodologia integral innovadora para planes y tesis: La metodología del cómo formularlos.* México DF: CENGAGE learning.

[28] Sampieri, R. H. (2017). *Metodología de la investigación.* México DF.: McGrawHill Education.

[29] Silvia Argudo, A. P. (2013). *Mejorar las Búsquedas de Información.* Barcelona: UOC.

[30] Ubaldo Geovanni Villaseñor Carrillo, M. A. (2010). Desarrollo de un sistema de navegación para robots móviles mediante diferentes patrones de comportamientos. *CIINDET 2010*, 6.

[31] Wang, L. X. (1997). A Course in Fuzzy Systems and Control . *Prentice Hall*, 0.

[32] ZADEH, L. A. (1965). *Fuzzy Set, Information and Control.* Berkeley, California: University of California.

7. Anexos

7.1 Código de Matlab

```
clc
clear all
close all

paso = 0.1; % muestreo de 0.1
ED = -75:paso:75; %vector de Error distancia
V = -12:paso:12; %vector de voltaje
% funciones de menbresia del Error con respecto la Distancia
% Error Distancia Negativa
a = -75;
m = -70;
b = 0;
f1 = (ED<=a).*(0);
f2 = ((a<ED)&(ED<=m)).*((ED-a)/(m-a));
f3 = ((m<ED)&(ED<=b)).*((b-ED)/(b-m));
f4 = (ED>=b).*(0);
EDN = f1+f2+f3+f4;
% Error Distancia Media
a2 = -50;
m2 = 0;
b2 = 50;
ff1 = (ED<=a2).*(0);
ff2 = ((a2<ED)&(ED<=m2)).*((ED-a2)/(m2-a2));
ff3 = ((m2<ED)&(ED<=b2)).*((b2-ED)/(b2-m2));
ff4 = (ED>=b2).*(0);
EDM = ff1+ff2+ff3+ff4;
% Error Distancia Positiva
a3 = 0;
m3 = 70;
b3 = 75;
fff1 = (ED<=a3).*(0);
fff2 = ((a3<ED)&(ED<=m3)).*((ED-a3)/(m3-a3));
fff3 = ((m3<ED)&(ED<=b3)).*((b3-ED)/(b3-m3));
fff4 = (ED>=b3).*(0);
EDP = fff1+fff2+fff3+fff4;
% Graficación
subplot(3,1,1), plot(ED,EDN,ED,EDM,ED,EDP,'LineWidth',3)
set(gca,'FontSize',18), legend('EDN','EDM','EDP')
xlabel('Error_Distancia'), ylabel('\mu(Dist)')
axis([-75 75 0 1])

% Conjuntos o funciones de menbresia del Voltaje
% Voltaje Negativo
a4 = -8;
m4 = -6;
b4 = 0;
Y1 = (V<=a4).*(0);
Y2 = ((a4<V)&(V<=m4)).*((V-a4)/(m4-a4));
Y3 = ((m4<V)&(V<=b4)).*((b4-V)/(b4-m4));
Y4 = (V>=b4).*(0);
VN = Y1+Y2+Y3+Y4;
```

```
% Voltaje Medio
a5 = -2;
m5 = 0;
b5 = 2;
YY1 = (V<=a5).*(0);
YY2 = ((a5<V)&(V<=m5)).*((V-a5)/(m5-a5));
YY3 = ((m5<V)&(V<=b5)).*((b5-V)/(b5-m5));
YY4 = (V>=b5).*(0);
VM = YY1+YY2+YY3+YY4;
% Voltaje Positivo
a6 = 0;
m6 = 6;
b6 = 8;
YYY1 = (V<=a6).*(0);
YYY2 = ((a6<V)&(V<=m6)).*((V-a6)/(m6-a6));
YYY3 = ((m6<V)&(V<=b6)).*((b6-V)/(b6-m6));
YYY4 = (V>=b6).*(0);
VP = YYY1+YYY2+YYY3+YYY4;
% Graficación
subplot(3,1,2), plot(V,VN,V,VM,V,VP,'LineWidth',3)
set(gca,'FontSize',18), legend('VN','VM','VP')
xlabel('Voltaje'), ylabel('\mu(Volt)')
axis([-8 8 0 1])

d0 = 60; % Error Distancia leida marcada en puntos
T = find(ED==d0);
subplot(3,1,1), hold on,
plot(d0,EDN(T),'*',d0,EDM(T),'*',d0,EDP(T),'*','LineWidth',3),
hold off

% Fusificacion Error Distancia por metodo Mandani
R1 = min(EDN(T),VP);
R2 = min(EDM(T),VM);
R3 = min(EDP(T),VN);
R = max(R1,max(R2,R3));
subplot(3,1,3), plot(V,R,'LineWidth',3)
set(gca,'FontSize',18), legend('V')
axis([-8 8 0 1])

% Defusificacion
vo = defuzz(V,R,'centroid')
hold on, plot(vo*ones(1,3),[0 0.5 1],'r','LineWidth',3)
```

7.2 Código del primer controlador

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd (0x27,2,1,0,4,5,6,7);
// Sensor ultrasonico
int TRIG = 10;
int ECO = 11;
// variables
int DURACION;
int DISTANCIA;

// Control Difuso
float V[] = {-8,-6,-4,-2,0,2,4,6,8}; //Vector de

float VN[9]={};
float VM[9]={};
float VP[9]={};

float IN3_pwm = 0;
float IN4_pwm = 0;
float IN1_V = 0;
float IN2_V = 0;
float IN3_V = 0;
float IN4_V = 0;

int IN1 = 3;
int IN2 = 5;
int IN3 = 6;
int IN4 = 9;

float D = 0;      // valor de la distancia actual
float ED = 0;     // Error de la distancia
float DD = 75;    // Distancia Deseable

//Variables para los grados de pertenencia (Dista
float EDN = 0;
float EDM = 0;
float EDP = 0;

//Conjuntos de ERROR de la distancia
float a1 = -75;
float m1 = -75;
float b1 = 0;

float a2 = -20;
float m2 = 0;
float b2 = 20;

float a3 = 0;
float m3 = 75;
float b3 = 75;

//Conjuntos del Voltaje
float a4 = -8;
float m4 = -8;
float b4 = 0;

float a5 = -2;
float m5 = 0;
float b5 = 2;

float a6 = 0;
float m6 = 8;
float b6 = 8;

//REGLAS
float R1[9], R2[9], R3[9] = {};

int i = 0;
float T = 0;
float sumaT = 0;
float vo = 0;
float mult_num = 0;
float sum_num = 0;

void setup()
{
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);
  Serial.begin(9600);
  lcd.setBacklightPin(3,POSITIVE);
  lcd.setBacklight(HIGH);    // hab
  lcd.begin(16, 2);      // 16 colu
  lcd.clear();       // limpia pant
```

```
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  V_SALIDA(); // voltaje de salida o Volt Out
}
void loop()
{
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  DURACION = pulseIn(ECO, HIGH);
  DISTANCIA = (DURACION * 0.034)/2;
//  Serial.println(DISTANCIA);

  D = DISTANCIA;
  ED = DD - D;

  lcd.setCursor(0, 0);
  lcd.print("D: ");
  lcd.print(DISTANCIA);
  lcd.print(" ED: ");
  lcd.print(ED);

  GdP_EDN();
  GdP_EDM();
  GdP_EDP();

  reglas();
  defuz();

  lcd.setCursor(0, 1);
  lcd.print("VO: ");
  lcd.print(vo);

//  Serial.println("VOLTAJE SALIDA: ");
//  Serial.println(vo);

  analogWrite(IN1, IN1_V);
  analogWrite(IN3, IN3_V);
  analogWrite(IN2, IN2_V);
  analogWrite(IN4, IN4_V);
  if (IN3_pwm<=1.35){
    analogWrite(IN1, LOW);
    analogWrite(IN3, LOW);
  }
  if (IN4_pwm<=1.35){
    analogWrite(IN2, LOW);
    analogWrite(IN4, LOW);
  }
  delay(10);
}
  // Grados de pretenencia
void GdP_EDN() {
  if (ED<a1){
    EDN = 0;
  }
  if (a1<ED && ED<=m1){
    EDN = ((ED-a1)/(m1-a1));
  }
  if (m1<ED && ED<=b1){
    EDN = ((b1-ED)/(b1-m1));
  }
  if (ED>=b1){
    EDN = 0;
  }
}

void GdP_EDM() {
  if (ED<a2){
    EDM = 0;
  }
  if (a2<ED && ED<=m2){
    EDM = ((ED-a2)/(m2-a2));
  }
  if (m2<ED && ED<=b2){
    EDM = ((b2-ED)/(b2-m2));
  }
  if (ED>=b2){
    EDM = 0;
  }
}

void GdP_EDP() {
  if (ED<a3){
    EDP = 0;
  }
  if (a3<ED && ED<=m3){
    EDP = ((ED-a3)/(m3-a3));
  }
```

```
  if (m3<ED && ED<=b3){
    EDP = ((b3-ED)/(b3-m3));
  }
  if (ED>=b3){
    EDP = 0;
  }
}

void V_SALIDA(){
  for (int i = 0; i < 9; i = i + 1){
    // Grados de pertenencia de Voltaje
    // Voltaje Negativo
    if (V[i]<=a4){
      VN[i] = 0;
    }
    if (a4<V[i] && V[i]<=m4){
      VN[i] = ((V[i]-a4)/(m4-a4));
    }
    if (m4<V[i] && V[i]<=b4){
      VN[i] = ((b4-V[i])/(b4-m4));
    }
    if (V[i]>=b4){
      VN[i] = 0;
    }

    // Voltaje Medio
    if (V[i]<=a5){
      VM[i] = 0;
    }
    if (a5<V[i] && V[i]<=m5){
      VM[i] = ((V[i]-a5)/(m5-a5));
    }
    if (m5<V[i] && V[i]<=b5){
      VM[i] = ((b5-V[i])/(b5-m5));
    }
    if (V[i]>=b5){
      VM[i] = 0;
    }

    // Voltaje Positivo
    if (V[i]<=a6){
      VP[i] = 0;
    }
    if (a6<V[i] && V[i]<=m6){
      VP[i] = ((V[i]-a6)/(m6-a6));
    }
    if (m6<V[i] && V[i]<=b6){
      VP[i] = ((b6-V[i])/(b6-m6));
    }
    if (V[i]>=b6){
      VP[i] = 0;
    }
  }
}

void reglas(){
  for (int i = 0; i < 9; i = i + 1){

    R1[i] = min(EDN, VP[i]);
    R2[i] = min(EDM, VM[i]);
    R3[i] = min(EDP, VN[i]);

    T = max(R1[i], max(R2[i], R3[i]));
    sumaT = sumaT + T;

    mult_num = V[i] * T;
    sum_num = sum_num + mult_num;
  }
}

void defuz(){
  vo = (sum_num)/(sumaT);

  if (vo >= 0.00){
    IN3_pwm = vo;
    IN4_pwm = 0.00;
    IN1_V = (IN3_pwm *255.0)/5.0;
    IN2_V = IN4_pwm;
    IN3_V = (IN3_pwm *255.0)/5.0;
    IN4_V = IN4_pwm;
  }
  if (vo <= -0.01){
    IN4_pwm = vo * -1;
    IN3_pwm = 0.00;
    IN1_V = IN3_pwm;
    IN2_V = (IN4_pwm *255.0)/5.0;
    IN3_V = IN3_pwm;
    IN4_V = (IN4_pwm *255.0)/5.0;
  }
}
```