



# **Robot Educativo STEM de Código Abierto para Escuelas**

**CURSO:** Intelig. Artif. Y Sist. Expertos

**NRC:** 11952

**DOCENTE:** Robert Freddy Ore Galvez

<sup>#1</sup>Aguilar Abel, <sup>\*2</sup>Millones Eduardo, <sup>†3</sup>Sanchez Miguel., <sup>‡4</sup>Silva Anderson, <sup>‡5</sup>Rivera Joseph.

*<sup>#1, \*2, †3, ‡4 and ‡5 Departamento de Ingeniería Mecatrónica, Universidad Privada del Norte</sup>*

*Los Olivos, Lima, Perú*

<sup>1</sup>N00316060@upn.pe

<sup>2</sup>N00337782@upn.pe

<sup>3</sup>N00335857@upn.pe

<sup>4</sup>N00296032@upn.pe

<sup>5</sup>N00317738@upn.pe

# Índice

<b>CAPÍTULO I: INTRODUCCIÓN</b>	<b>3</b>
1. SITUACIÓN PROBLEMÁTICA	3
2. FORMULACIÓN DEL PROBLEMA	3
2.1. PROBLEMA GENERAL	3
2.2. PROBLEMAS ESPECÍFICOS	3
3. JUSTIFICACIÓN DEL PROYECTO	3
4. OBJETIVOS DE LA INVESTIGACIÓN	3
4.1. OBJETIVO GENERAL	3
4.2. OBJETIVOS ESPECÍFICOS	3
<b>CAPÍTULO II: MARCO TEÓRICO</b>	<b>4</b>
1. ANTECEDENTES	4
2. BASES TEÓRICAS	4
3. GLOSARIO	4
<b>CAPÍTULO III: PROCESO</b>	<b>5</b>
1. IDENTIFICACIÓN DE PROCESOS	5
2. IDENTIFICACIÓN DE SENsoRES Y ACTUADORES	5

## RESUMEN EJECUTIVO

El presente proyecto desarrolla un Robot Educativo STEM de código abierto, diseñado para facilitar el aprendizaje de programación y conceptos básicos de robótica en estudiantes de colegios de bajos recursos. A diferencia de los robots Minisumo tradicionales —enfocados en el combate— esta plataforma se transforma en una herramienta pedagógica, segura y amigable, orientada a que los niños comprendan cómo las instrucciones modifican el comportamiento del robot. El sistema incorpora un servidor web integrado, desde el cual los estudiantes pueden interactuar con botones asociados a conceptos lógicos como “Bucle”, “Condicional” y “Variable”, permitiendo que experimenten de manera directa cómo estos elementos estructuran un programa. El robot utiliza tres sensores ultrasónicos para la detección de distancia y dos sensores infrarrojos TCRT5000 para el reconocimiento de superficies o líneas. Estos elementos permiten reemplazar el antiguo modo de ataque por un MODO\_SÍGUEME, basado en lógica difusa (Fuzzy Logic), que habilita al robot a seguir la mano del estudiante manteniendo una distancia segura y estable. De esta forma, el robot se vuelve un dispositivo didáctico, no agresivo y adecuado para ambientes escolares. Como aporte técnico y social, el proyecto adopta un enfoque open source, brindando documentación completa en GitHub que permite a cualquier docente replicar el robot con un costo aproximado menor a 20 dólares. Esta iniciativa busca reducir la brecha digital mediante una plataforma accesible, económica y adaptable a diversos centros educativos.

### 1. INTRODUCCIÓN

El acceso a la educación tecnológica sigue siendo limitado en muchas instituciones educativas, especialmente en zonas rurales o de bajos recursos, donde la falta de infraestructura y herramientas de aprendizaje dificulta el desarrollo de habilidades STEM (Ciencia, Tecnología, Ingeniería y Matemáticas). En respuesta a esta problemática, el proyecto ROBO-EDU propone el diseño y construcción de un robot educativo autónomo de bajo costo, basado en hardware open source y orientado a la enseñanza práctica de programación, electrónica y control inteligente.

El sistema está desarrollado alrededor de un ESP32, un microcontrolador moderno y accesible que ofrece conectividad WiFi, procesamiento dual-core y capacidades de interfaz ideales para la educación. El robot incorpora una arquitectura sensorial compuesta por tres sensores ultrasónicos HC-SR04, encargados de medir distancias frontales y laterales, así como dos sensores infrarrojos TCR 5000, utilizados para la detección de superficies y líneas. Estos sensores permiten que el robot interactúe de forma segura con su entorno, convirtiéndose en una herramienta confiable dentro de un aula.

El rasgo distintivo de ROBO-EDU es la implementación de un sistema de lógica difusa, que otorga al robot la capacidad de tomar decisiones graduales basadas en incertidumbre, imitando el razonamiento humano. En lugar de reaccionar únicamente con valores binarios (0 o 1), el robot evalúa niveles intermedios como “cerca”, “medio” o “lejos”, ajustando su velocidad y comportamiento de manera continua. Este enfoque permite enseñar a los estudiantes conceptos avanzados como “fuzzy logic”, control inteligente y sistemas expertos mediante ejemplos visuales y comportamientos observables.

Además, ROBO-EDU integra una interfaz web educativa donde los estudiantes pueden interactuar con el robot, visualizar sus sensores en tiempo real y activar funciones didácticas como “condicional”, “bucle” o “variable”, facilitando el aprendizaje de programación de manera intuitiva. Gracias a su diseño modular, económico y documentado públicamente, cualquier docente puede replicarlo por menos de 20 USD, reduciendo así la brecha digital y promoviendo una educación inclusiva y moderna.

En conjunto, este proyecto no solo constituye una plataforma robótica funcional, sino también una herramienta pedagógica accesible, segura y replicable, capaz de transformar la forma en que los estudiantes aprenden tecnología desde edades tempranas.

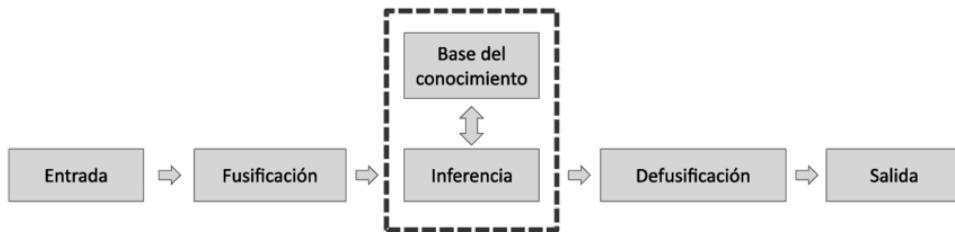


Ilustración 2-1. Estructura de un sistema de lógica difusa

### **Objetivos Generales :**

1. Desarrollar un sistema de control inteligente basado en lógica difusa para mejorar la toma de decisiones del robot minisumo frente a entornos cambiantes y ruidos
2. Implementar un módulo completo de detección y acción en ESP32, integrando sensores ultrasónicos, infrarrojos y un puente H para el control dinámico de los motores.
3. Optimizar el comportamiento del robot para que pueda atacar, evadir o corregir su trayectoria de manera autónoma y estable utilizando reglas difusas en tiempo real.

### **Objetivos Específicos :**

1. Diseñar las funciones de pertenencia y las reglas difusas que permitan interpretar las lecturas de los sensores en estados como cercano/lejos y centrado/desviado.
2. Integrar y calibrar los sensores ultrasónicos e infrarrojos con el ESP32, asegurando la adquisición precisa de datos y su conversión en acciones mediante el puente H.

### **Justificación**

El proyecto ROBO-EDU se justifica porque proporciona una herramienta educativa accesible y de bajo costo que permite enseñar robótica, programación y control inteligente en instituciones con recursos limitados. Su diseño basado en hardware económico (ESP32, sensores ultrasónicos y TCR5000) y su carácter open source facilitan que cualquier docente pueda replicarlo sin necesidad de equipamiento especializado. Además, la implementación de lógica difusa permite que los estudiantes comprendan conceptos avanzados mediante comportamientos visibles del robot, fortaleciendo el aprendizaje práctico. Gracias a su enfoque seguro, didáctico y adaptable, el robot contribuye directamente al desarrollo de habilidades STEM y reduce las barreras de acceso a la tecnología en el entorno educativo.

### **Importancia**

El proyecto ROBO-EDU se justifica porque proporciona una herramienta educativa accesible y de bajo costo que permite enseñar robótica, programación y control inteligente en instituciones con recursos limitados. Su diseño basado en hardware económico (ESP32, sensores ultrasónicos y TCR5000) y su carácter open source facilitan que

cualquier docente pueda replicarlo sin necesidad de equipamiento especializado. Además, la implementación de lógica difusa permite que los estudiantes comprendan conceptos avanzados mediante comportamientos visibles del robot, fortaleciendo el aprendizaje práctico. Gracias a su enfoque seguro, didáctico y adaptable, el robot contribuye directamente al desarrollo de habilidades STEM y reduce las barreras de acceso a la tecnología en el entorno educativo.

## **Alcance**

El proyecto incluye:

1. Definición del entorno sensorial del robot educativo
2. Diseño del sistema de lógica difusa para el movimiento seguro
3. Simulación del sistema experto difuso
4. Implementación completa en ESP32 usando C++
5. Desarrollo de una interfaz web pedagógica

## **2. IDENTIFICACIÓN Y APLICACIÓN DE PRINCIPIOS DE INGENIERÍA**

### **2.1. IDENTIFICACIÓN Y APLICACIÓN DE PRINCIPIOS DE INGENIERÍA**

- 2.1.1. Control en lazo cerrado: El robot mide continuamente distancias con los 3 sensores ultrasónicos y el estado del piso con los TCR5000, toma decisiones y ajusta la velocidad de sus motores para mantener un comportamiento seguro.
- 2.1.2. Sistemas embebidos: Se implementó una arquitectura optimizada para ESP32 que gestiona WiFi, interfaz web, PWM para motores, lectura de sensores y lógica difusa sin comprometer memoria ni tiempo de respuesta.
- 2.1.3. Lógica Difusa (Fuzzy Logic): El sistema reemplaza decisiones rígidas por grados de verdad (cerca, medio, lejos), permitiendo un comportamiento más humano, suave y educativo.
- 2.1.4. Interacción Hombre-Robot: La interfaz web convierte acciones en conceptos lógicos (bucle, condicional, variable), integrando enseñanza de programación con control físico del robot.
- 2.1.5. Diseño para Seguridad: Los sensores infrarrojos del piso actúan como un mecanismo de protección que evita caídas sin detener el flujo educativo.

## 2.2. APLICACIÓN DE LOS PRINCIPIOS

- 2.2.1. Los sensores entregan valores continuamente; la lógica del robot decide acciones (acerarse, detenerse, retroceder, explorar) y ajusta los motores en tiempo real.
- 2.2.2. Se definieron funciones de pertenencia “Cerca”, “Medio” y “Lejos”, y se aplicó un proceso de fusificación → reglas → desfusificación para calcular la velocidad final.
- 2.2.3. Se usaron PWM por hardware, interrupciones eficientes y un WebServer ligero para permitir que la lógica, los motores y la interfaz web funcionen simultáneamente.
- 2.2.4. Si los TCR5000 detectan un borde, el robot retrocede y corrige automáticamente su ruta sin detener el modo de aprendizaje.
- 2.2.5. Cada botón del servidor web activa un concepto lógico real (bucle, condicional, lógica difusa, variable), traducido a una acción física del robot.

## 3. MODELADO MATEMÁTICO Y ANÁLISIS TÉCNICO

El robot ROBO-EDU toma decisiones utilizando un **Sistema de Lógica Difusa (Fuzzy Logic)**, el cual permite transformar mediciones imprecisas del entorno en acciones suaves y continuas. Este modelo reemplaza la lógica binaria tradicional por un proceso matemático más cercano al razonamiento humano.

### 3.1. Variable de Entrada: Distancia Difusa

$$d \in [0,60] \text{ cm}$$

En lugar de interpretarse como “*objeto cerca / objeto lejos*”, se transforma en tres clases difusas:

$$u(d) = 0 \quad x \leq a$$

$$\text{Cerca} \rightarrow u_{cerca}(d) = \text{fuzzyGrade}(d, -1, 0, 25)$$

$$u(d) = \frac{x - a}{b - a} \quad a < x \leq b$$

$$\text{Media} \rightarrow u_{media}(d) = \text{fuzzyGrade}(d, 15, 30, 45)$$

$$u(d) = \frac{c - x}{c - b} \quad b < x \leq c$$

$$\text{Lejos} \rightarrow u_{lejos}(d) = \text{fuzzyGrade}(d, 35, 60, 80)$$

$$u(d) = 0 \quad x \geq c$$

### 3.2. Base de Conocimiento: Reglas Difusas

.Las reglas son del tipo:

R1: Si distancia es cerca → velocidad = 0

R2: Si distancia es media → velocidad = 120

R3: Si distancia es lejos → velocidad = 180

Matemáticamente, la activación de cada regla es:

$$R_i = u_i(d)$$

Es decir, cuanto más “verdadera” es una condición, mayor influencia tiene su acción sobre la salida.

### 3.3. Agregación y Desfusificación:

Para determinar la velocidad final del robot, se combinan todas las reglas activadas mediante un método tipo Centro de Gravedad (COG):

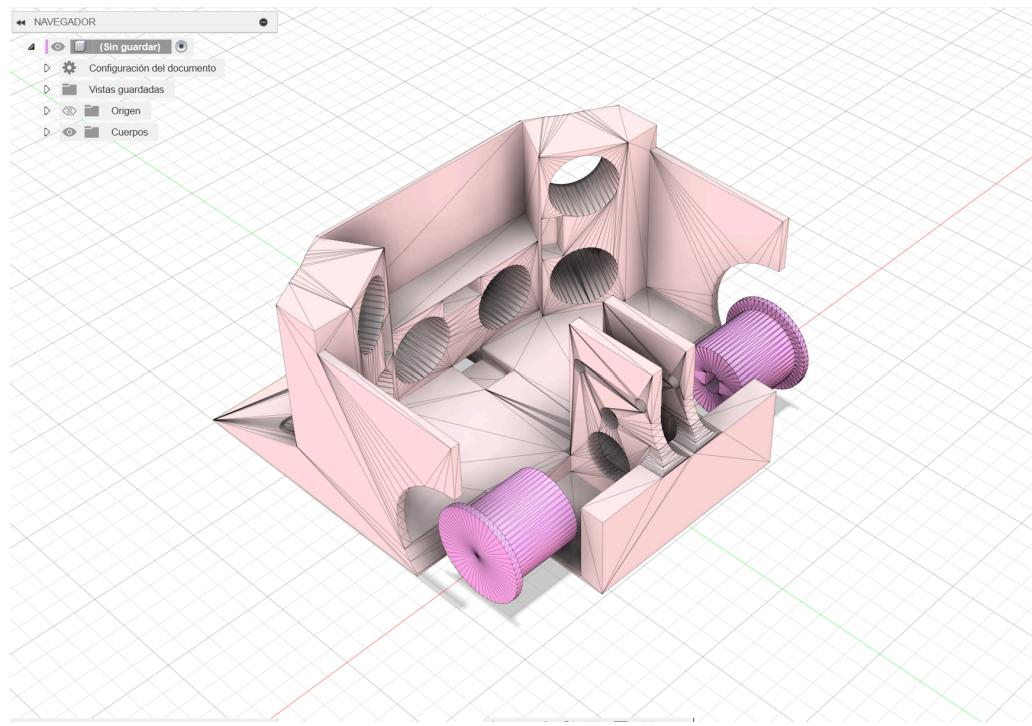
$$V = \frac{R_1 \cdot v_1 + R_2 \cdot v_2 + R_3 \cdot v_3}{R_1 + R_2 + R_3}$$

### 3.4. Justificación técnica

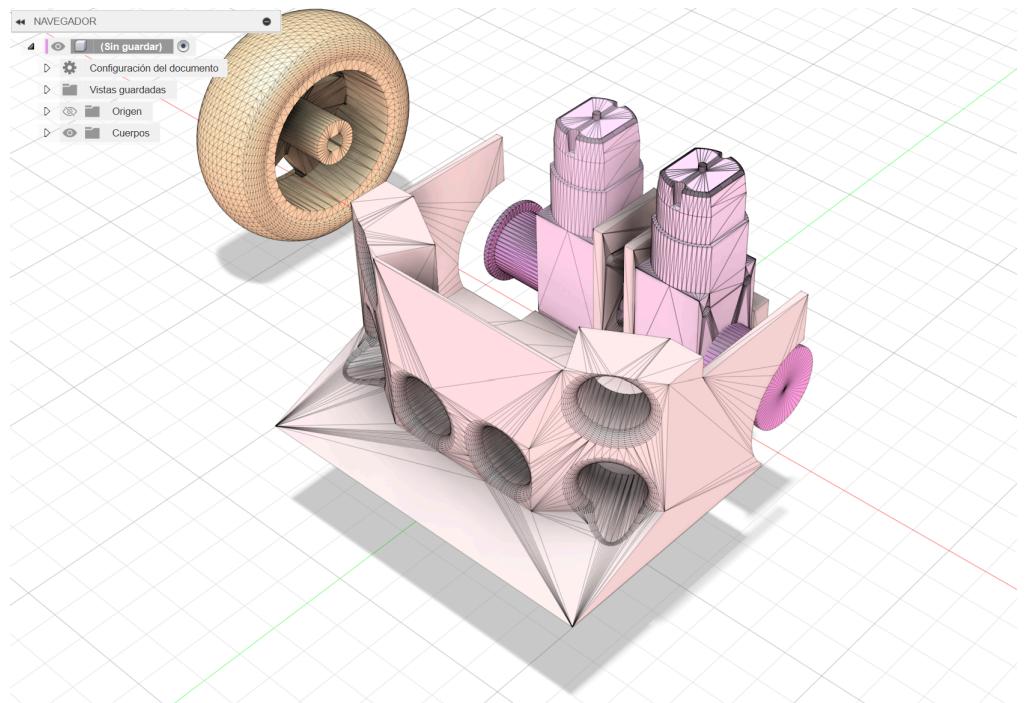
La lógica difusa se eligió como técnica principal debido a su capacidad para tomar decisiones en entornos inciertos, como sucede en un minisumo donde las lecturas de sensores cambian constantemente y no siempre son exactas. A diferencia de un control tradicional basado en umbrales rígidos, la lógica difusa permite interpretar valores intermedios (distancias, direcciones, intensidad del blanco) y convertirlos en acciones suaves y más inteligentes. El diseño se basa en funciones de pertenencia que modelan comportamientos reales del robot (cercano/lejos, centrado/desviado) y en reglas difusas que combinan estos estados para determinar acciones como atacar, girar o retroceder. Esto permite que el robot responda de forma adaptable sin necesidad de cálculos pesados. Se justifica técnicamente porque el modelo difuso reduce complejidad computacional, es estable ante ruido sensorial, y permite implementar una toma de decisiones continua y robusta en un microcontrolador como el ESP32.

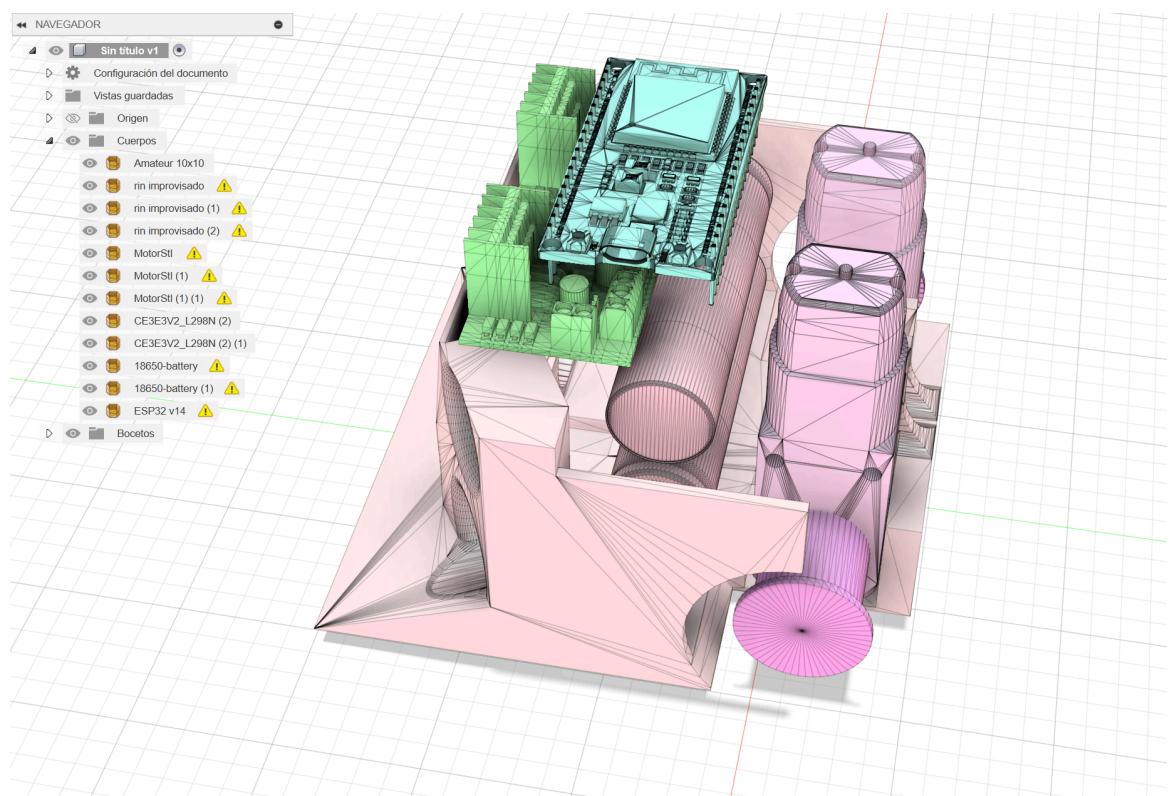
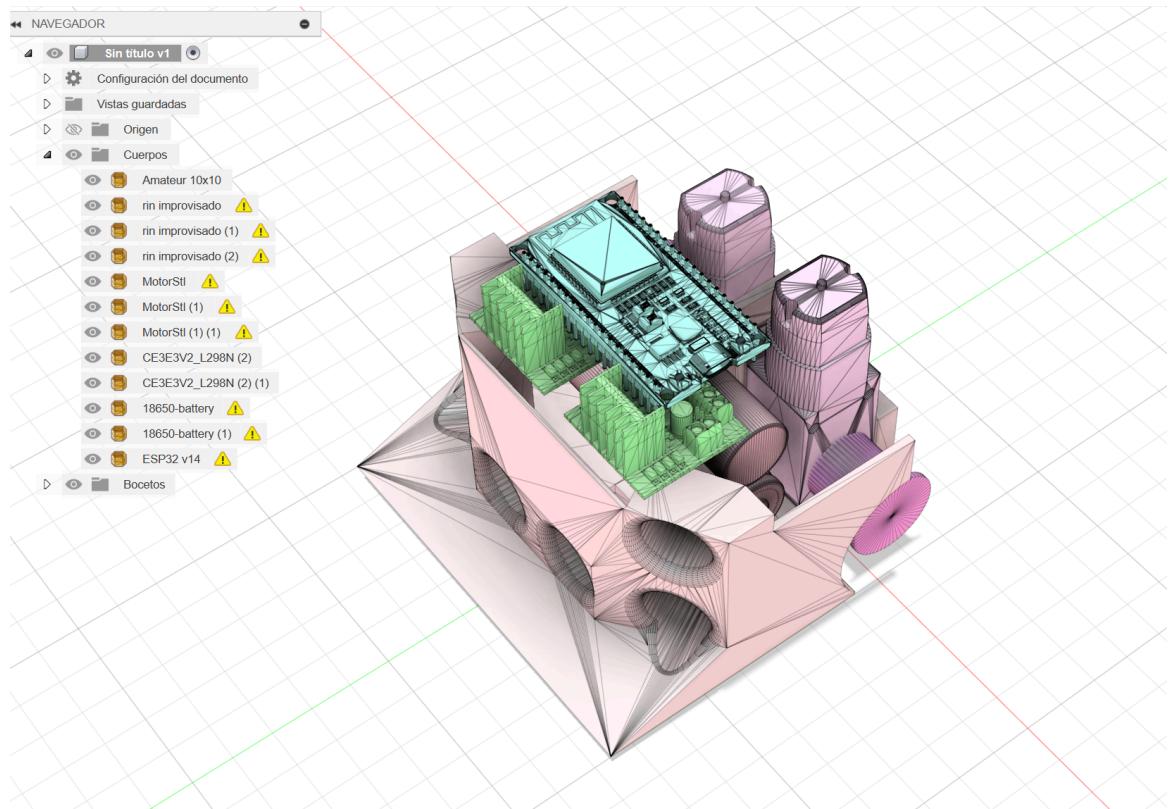
## 4. PROCESO DE CREACIÓN DEL ROBOT

### 4.1. Diseño en 3D



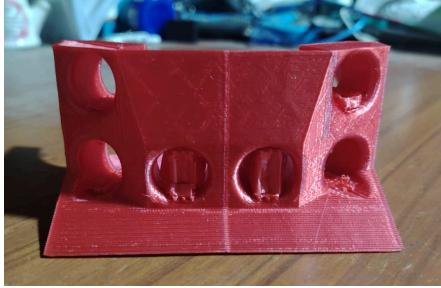
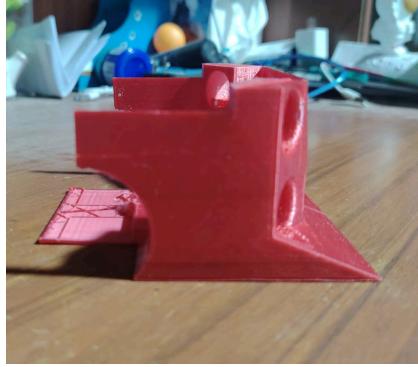
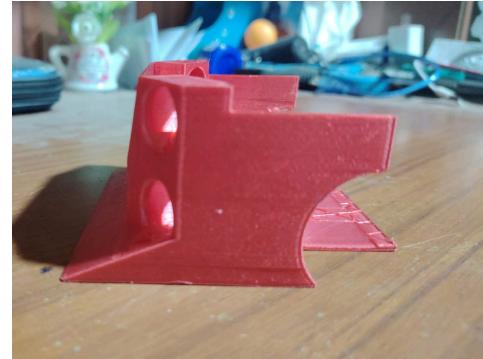
### 4.2. Ensamble en 3D





#### 4.3. PROTOTIPO IMPRESO 3D

Para iniciar el proceso de entrenamiento de nuestro robot sumo, fue necesario realizar la impresión del prototipo y evaluar los aspectos del diseño que podían optimizarse. Esto incluyó revisar tolerancias, ajustar márgenes y verificar el correcto encaje de cada pieza según las vistas desarrolladas en Inventor. Gracias a esta etapa pudimos identificar mejoras esenciales antes del ensamblaje final.

VISTA FRONTAL 3D	VISTA LATERAL DERECHA 3D
	
VISTA ATRAS EN 3D	VISTA LATERAL IZQUIERDO 3D
	

## 5. ENTRENAMIENTO CON LÓGICA DIFUSA

El robot funciona como un sistema experto compuesto por:

Entrada: Sensores (ultrasonidos e infrarrojos)

Procesamiento: Lógica Difusa (fuzzy logic), Reglas de inferencia, Defuzificación

Salida: Movimiento mediante DRV8833 (PWM y dirección)

Esto simula un “cerebro suave”, donde las decisiones no son 0 o 1, sino grados de certeza.

### Proceso de toma de decisiones:

- Percepción del entorno :

El robot mide la distancia con:

```
long dist = getDist(Trigger2, Echo2);
```

La lectura se satura a **60 cm**, para mantener estabilidad:

```
if (dist > 60) dist = 60;
```

- Fusificación (convertir la distancia en conceptos lingüísticos):

```
float fuzzyGrade(float val, float low, float peak, float high);
```

Esta función convierte un número (cm) en un grado de pertenencia, es decir, qué tan verdadero es un concepto como: Muy Cerca, Distancia Media, Lejos.

$$\text{Cerca} \rightarrow u_{cerca}(d) = \text{fuzzyGrade}(d, -1, 0, 25)$$

$$\text{Media} \rightarrow u_{media}(d) = \text{fuzzyGrade}(d, 15, 30, 45)$$

$$\text{Lejos} \rightarrow u_{lejos}(d) = \text{fuzzyGrade}(d, 35, 60, 80)$$

- Reglas Inferencia (Sistema Experto):

```

float ruleCaution = isClose;
float ruleObserve = isMedium;
float ruleExplore = isFar;

```

Situación Difusa	Acción
Cerca	Frenar con cuidado
Media	Avanzar lento y observar
Lejos	Explorar con libertad

- Defuzificación (convertir razonamiento en movimiento):

```

float num = (ruleCaution * 0) + (ruleObserve * 120) + (ruleExplore * 180);
float den = ruleCaution + ruleObserve + ruleExplore;
int speed = (den > 0) ? (int)(num / den) : 100;

```

Representa un promedio ponderado, que genera una velocidad suave, ejemplo:

Si está muy cerca → 0

Si está medio → 120

Si está lejos → 180

## Resultado

Una vez definidas las funciones de pertenencia y las reglas del sistema difuso, estas fueron implementadas en el ESP32 mediante funciones matemáticas ligeras y condicionales simples. Todo el proceso de razonamiento de fusificación, activación de reglas y desfuzificación se ejecuta directamente en el microcontrolador sin necesidad de cálculos complejos ni librerías externas, lo que garantiza una respuesta rápida en tiempo real. De este modo, la lógica difusa se integra como un

sistema experto embebido, permitiendo que el robot tome decisiones suaves y continuas basadas en la distancia del objeto detectado, optimizando el movimiento de los motores sin recurrir a procesamiento intensivo o modelos de inteligencia artificial.

Para el desarrollo usamos los siguientes softwares y frameworks:

Herramienta	Uso
Arduino IDE	<p>Entorno principal donde se desarrolló y cargó el código en el ESP32.</p> <p>Permite programar en C++ y manejar las librerías para sensores ultrasónicos, infrarrojos y el driver DRV8833.</p>
EasyEDA	<p>Diseño del circuito electrónico del robot.</p> <p>Se utilizó para esquematizar las conexiones del ESP32, sensores HC-SR04, sensores IR y el driver de motores.</p>
Fuzzy Logic Designer (Simulador Web)	<p>Herramienta utilizada para diseñar las funciones de pertenencia y validar las reglas difusas del sistema.</p> <p>Permite visualizar fusificación, activación de reglas y desfusificación antes de llevar la lógica al ESP32</p>

Tabla 1-3. Herramientas utilizadas. Creación Propia

Hardware:

- 1) ESP32 Dev Kit: Placa de desarrollo principal
- 2) Driver de Motor: DRV8833(o similar)
- 3) Sensores de Línea: 2x sensores TCRT5000
- 4) Fuente de Alimentación: Batería LiPo o Li-ion adecuada

## PREPARACIÓN DEL ENTORNO:

En esta fase se integró el sistema de lógica difusa dentro del microcontrolador ESP32, el cual actúa como el cerebro del robot educativo. Para ello se configuraron los pines correspondientes a los tres sensores ultrasónicos HC-SR04 y a los dos sensores infrarrojos TCRT5000, los cuales permiten medir distancia frontal y detectar superficies cercanas al suelo.

Estas lecturas proporcionan las variables necesarias para la lógica difusa:

Distancia (Cerca, Medio, Lejos)

Seguridad del piso (Derecha segura / no segura, Izquierda segura / no segura)

También se definieron las funciones de control para los motores —avanzar, detenerse, girar suavemente o reducir velocidad— utilizando el driver DRV8833, encargado de manejar la dirección y el PWM de cada rueda.

Durante la ejecución, el ESP32 realiza un ciclo continuo donde:

Lee los sensores en tiempo real.

Fusifica la distancia detectada en grados de pertenencia (ej. 20% Cerca, 33% Medio).

Evalúa las reglas difusas diseñadas para mantener una distancia segura del niño (modo “Sígueme”).

Desfusifica el resultado para obtener un valor PWM suave y progresivo.

Envía esa velocidad a los motores, permitiendo comportamientos naturales como acercarse lentamente, mantenerse estable o retroceder si se acerca demasiado. Este proceso se repite de manera continua, permitiendo que el robot responda de forma amigable, segura y fluida, sin movimientos bruscos, cumpliendo su función educativa de mostrar cómo una “máquina piensa” mediante lógica difusa.

```
EditMode currentMode = IDLE;

// --- 3. HARDWARE MAPPING (DRV8833) ---
#define MOT_A1 25
#define MOT_A2 27
#define MOT_B1 4
#define MOT_B2 13

// SENSORES (EN PISO Y ULTRASONIDOS)
#define FLOOR_L 32
#define FLOOR_R 33

const int Echo1 = 18; const int Trigger1 = 19; // Izq
const int Echo2 = 16; const int Trigger2 = 17; // Centro
const int Echo3 = 14; const int Trigger3 = 15; // Der
#define RD_SYS 2

// CONFIG PWM
const int pwmFreq = 5000;
const int pulses = 8;
const int ch_A1a0, ch_A2a1, ch_B1a2, ch_B2a3;

// --- 4. INTERFAZ WEB EDUCATIVA (HTML/CSS) ---
const char Index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <title>ROBO-EDU LAB</title>
</head>
<body>
    <h1>ROBO-EDU LAB</h1>
    <p>Este es un prototipo de robot educativo desarrollado en Arduino y Python.</p>
    <p>Características principales:</p>
    <ul>
        <li>Control remoto inalámbrico (Bluetooth).</li>
        <li>Sensores de proximidad (ultrasonidos y infrarrojos).</li>
        <li>Motor paso-a-paso con driver DRV8833. </li>
        <li>Interfaz web para monitoreo y control remoto. </li>
    </ul>
</body>
</html>
)rawliteral";
```

Visual Studio Code

*Creación propia (2025)*

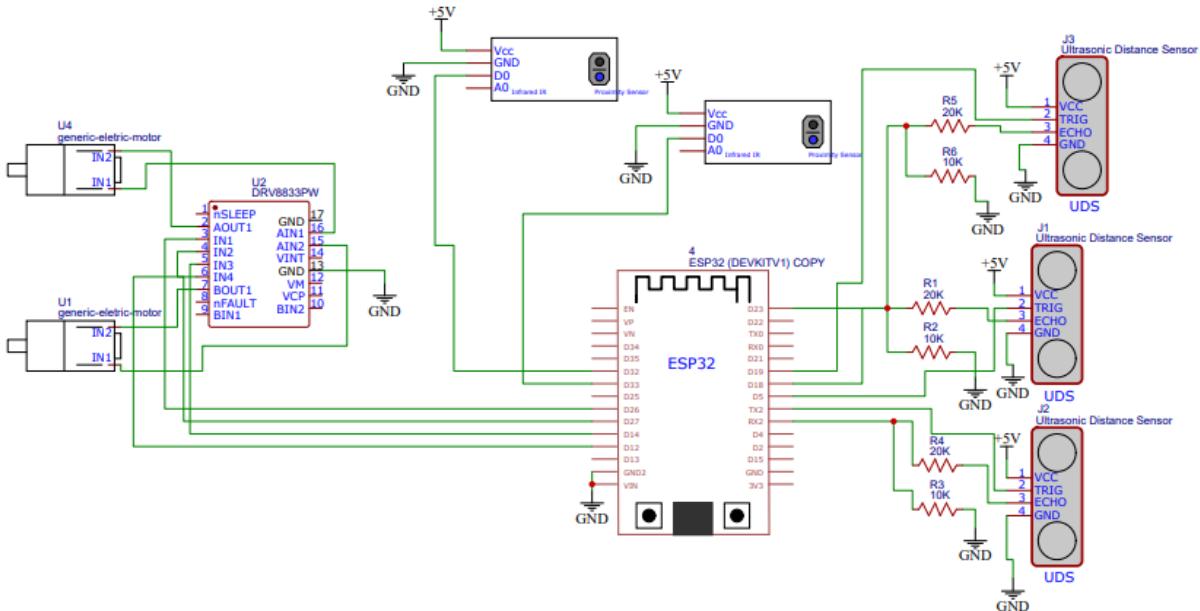
## 6. PLANO ELECTRÓNICO

### 1. IDENTIFICACIÓN DE SENSORES Y ACTUADORES

Componente	Función
Microcontrolador (cerebro)	ESP32
Actuadores (movimiento)	Dos motores DC (con ruedas) y un Driver L298N.
Sensores de Borde	Sensores infrarrojos de reflexión (x2) para detectar la línea blanca.
Sensor de Oponente	Sensor de proximidad (ultrásónico o IR) para medir la distancia frontal.

Tabla 1-2. Componentes. Creación Propia

### 2. PLANO ELECTRÓNICO



## CONCLUSIONES

- Se logró diseñar y construir un robot educativo funcional, cumpliendo el objetivo de crear una plataforma accesible, reproducible y de bajo costo basada en ESP32, permitiendo que estudiantes comprendan conceptos básicos de robótica sin requerir hardware costoso.
- La implementación de lógica difusa cumplió el objetivo específico de dotar al robot de un comportamiento suave, seguro y entendible, permitiendo al sistema mantener distancia con el usuario de forma progresiva y no brusca, demostrando que este tipo de control es adecuado para entornos educativos.
- El sistema sensorial compuesto por sensores ultrasónicos y TCRT5000 respondió adecuadamente, logrando el objetivo de ofrecer un robot que interprete su entorno de forma confiable y actúe en consecuencia, reforzando el aprendizaje sobre lectura de sensores y toma de decisiones.
- El robot facilitó el aprendizaje interactivo de programación y algoritmos, cumpliendo el objetivo general de servir como herramienta pedagógica: los estudiantes pueden visualizar cómo las reglas difusas y las entradas de sensores se traducen en movimientos reales.
- El proyecto demostró que es posible crear soluciones tecnológicas educativas open-source con impacto social, cumpliendo el objetivo de ofrecer un recurso replicable por docentes y estudiantes, contribuyendo a reducir brechas tecnológicas en instituciones con recursos limitados.

## **Recomendaciones**

- Ampliar el conjunto de reglas difusas para permitir más modos educativos (seguir luz, evitar obstáculos complejos, control de velocidad variable), aumentando el valor pedagógico.
- Mejorar la interfaz web educativa, permitiendo a los estudiantes modificar parámetros en tiempo real (ganancias, reglas, umbrales), reforzando el aprendizaje basado en experimentación.
- Agregar una guía docente estructurada, con prácticas, actividades y retos progresivos, fortaleciendo el objetivo de servir como herramienta educativa formal.
- Expandir la modularidad del hardware, usando conectores estándar que permitan a los estudiantes integrar fácilmente nuevos sensores o actuadores sin rehacer el diseño.
- Incorporar métricas de aprendizaje, como mostrar gráficamente las entradas y salidas del controlador difuso, para mejorar la comprensión visual del funcionamiento interno del robot.

## VII. REFERENCIAS

- [1] Watkins, C. J. C. H. (1989). Learning from delayed rewards. PhD thesis, Cambridge University. (Introducción a Q-Learning).
- [2] Documentación oficial de ESP32 y el framework Arduino.
- [3] Documentación de la librería NumPy.

### 1. GLOSARIO

- 1.1. Dohyo: El ring circular utilizado en la competición de Sumo, delimitado por una línea blanca.
- 1.2. Q-Table: Matriz de valores que almacena la calidad o utilidad de tomar una acción específica en un estado dado. Es el resultado final del entrenamiento Q-Learning.
- 1.3. ESP32: Microcontrolador de bajo costo con Wi-Fi y Bluetooth, utilizado como el cerebro del robot.
- 1.4. Aprendizaje por Refuerzo (RL): Paradigma de Machine Learning donde un agente aprende a tomar decisiones interactuando con un entorno para maximizar la recompensa.
- 1.5.  $\epsilon$ -greedy: Estrategia utilizada en RL que decide si el agente debe explorar el entorno o explotar el conocimiento aprendido.
- 1.6. Proceso de Decisión de Markov (MDP): Marco matemático para modelar la toma de decisiones donde los resultados son parcialmente aleatorios y parcialmente bajo el control de un decisor.

## ANEXO:

A continuación, se presentan los códigos fuente de la simulación de la IA y la implementación en el microcontrolador.  
Tabla del código de ESP32

Este es el repositorio, donde se encuentra recopilada toda la información del proyecto junto con el código de prueba para su correcto funcionamiento: <https://github.com/edumillones/ROBO-EDU-ESP32>.”

```
1  #include <WiFi.h>
2  #include <WebServer.h>
3
4  // -----
5  // PROYECTO: ROBO-EDU "LOGIC LAB"
6  // OBJETIVO: PLATAFORMA EDUCATIVA STEM (ODS 4: EDUCACIÓN DE CALIDAD)
7  // HARDWARE: ESP32 + DRV8833 + SENSORES
8  // -----
9
10 // --- 1. CONFIGURACIÓN DE RED ---
11 const char* ssid = "TU_WIFI";      // <--- CAMBIAR
12 const char* password = "TU_CLAVE"; // <--- CAMBIAR
13
14 WebServer server(80);
15
16 // --- 2. MODOS EDUCATIVOS ---
17 enum EduMode {
18     IDLE,           // En espera
19     AUTO_LOGIC,    // Lógica Difusa (El robot "piensa")
20     MODE_FOLLOW,   // Seguimiento (Antes Ataque)
21     MODE_ZONE,     // Respetar límites (Antes Defensa)
22     MODE_DEMO      // Demostración de movimientos
23 };
24
25 EduMode currentMode = IDLE;
26
```

6

```
27 // --- 3. HARDWARE MAPPING (DRV8833) ---
28 #define MOT_A1 25
29 #define MOT_A2 27
30 #define MOT_B1 4
31 #define MOT_B2 13
32
33 // SENSORES (IR PISO y ULTRASONIDOS)
34 #define FLOOR_L 32
35 #define FLOOR_R 33
36
37 const int Echo1 = 18; const int Trigger1 = 19; // Izq
38 const int Echo2 = 16; const int Trigger2 = 17; // Centro
39 const int Echo3 = 14; const int Trigger3 = 15; // Der
40 #define LED_SYS 2
41
42 // CONFIG PWM
43 const int pwmFreq = 5000;
44 const int pwmRes = 8;
45 const int ch_A1=0, ch_A2=1, ch_B1=2, ch_B2=3;
46
```

```

47 // --- 4. INTERFAZ WEB EDUCATIVA (HTML/CSS) ---
48 const char index_html[] PROGMEM = R"rawliteral(
49 <!DOCTYPE html>
50 <html lang="es">
51 <head>
52   <meta charset="UTF-8">
53   <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
54   <title>ROBO-EDU LAB</title>
55   <style>
56     :root {
57       --primary: #4A90E2; /* Azul Educativo */
58       --success: #7ED321; /* Verde Lógica */
59       --warning: #F5A623; /* Naranja Acción */
60       --danger: #D0021B; /* Rojo Stop */
61       --bg: #F0F4F8;
62       --card: #FFFFFF;
63     }
64     body {
65       background-color: var(--bg);
66       color: #333;
67       font-family: 'Verdana', sans-serif;
68       text-align: center;
69       margin: 0; padding: 20px;
70       display: flex; flex-direction: column; align-items: center;
71     }
72     h1 { color: var(--primary); margin-bottom: 5px; }
73     p { color: #666; font-size: 0.9rem; margin-top: 0; }
74
75     .status-card {

```

```

76       background: var(--card);
77       padding: 15px;
78       border-radius: 15px;
79       box-shadow: 0 4px 6px rgba(0,0,0,0.1);
80       width: 96%; max-width: 400px;
81       margin-bottom: 20px;
82       border-left: 5px solid var(--success);
83     }
84     #status-text { font-weight: bold; font-size: 1.2rem; color: var(--primary); }
85
86     .grid-container {
87       display: grid; grid-template-columns: 1fr 1fr; gap: 15px;
88       width: 100%; max-width: 400px;
89     }
90
91     button {
92       border: none; border-radius: 12px;
93       padding: 20px; font-size: 1rem; font-weight: bold;
94       cursor: pointer; transition: transform 0.1s;
95       color: white; box-shadow: 0 4px 0 rgba(0,0,0,0.2);
96     }
97     button:active { transform: translateY(4px); box-shadow: none; }
98
99     .btn-logic { background-color: var(--success); grid-column: span 2; }
100    .btn-action { background-color: var(--primary); }
101    .btn-zone { background-color: var(--warning); }
102    .btn-stop { background-color: var(--danger); grid-column: span 2; margin-top: 10px; }
103
104    .footer { margin-top: 30px; font-size: 0.8rem; color: #999; }

```

```

</style>
</head>
<body>
    <h1>ROBO-EDU LAB</h1>
    <p>Aprendizaje de Lógica y Robótica</p>

    <div class="status-card">
        <small>ESTADO DEL SISTEMA:</small><br>
        <span id="status-text">LISTO PARA APRENDER</span>
    </div>

    <div class="grid-container">
        <button class="btn-logic" onclick="cmd('auto')">▶ EJECUTAR LÓGICA DIFUSA</button>
        <button class="btn-action" onclick="cmd('follow')">MODO "SIGUEME"</button>
        <button class="btn-zone" onclick="cmd('zone')">TEST "ZONA SEGURA"</button>
        <button class="btn-action" onclick="cmd('demo')">DEMO MOTORES</button>

        <button class="btn-stop" onclick="cmd('stop')">■ DETENER PROGRAMA</button>
    </div>

    <div class="footer">
        PROYECTO DE RESPONSABILIDAD SOCIAL<br>
        Ingeniería Mecatrónica - UPN
    </div>

    <script>
        function cmd(mode) {
            // Feedback visual educativo
            const txt = document.getElementById('status-text');
            const msgs = {
                'auto': 'PENSANDO (Fuzzy Logic)...',
                'follow': 'EJECUTANDO: if(obj < 20)',
                'zone': 'VALIDANDO LÍMITES...',
                'demo': 'TEST DE HARDWARE',
                'stop': 'PROGRAMA DETENIDO'
            };
            txt.innerText = msgs[mode] || mode.toUpperCase();

            // Envío al servidor
            fetch('/api?mode=' + mode);
        }
    </script>
</body>
</html>
)rawliteral";
150
151 // --- DECLARACIONES ---
152 void setMotorA(int v); void setMotorB(int v);
153 void stopMotors(); void move(int l, int r);
154 long getDist(int t, int e);
155 void runEducationalFuzzy();
156 void runFollowMode();
157
158 ▾ void setup() {
159     Serial.begin(115200);
160
161     // Configuración de Pines
162     pinMode(FLOOR_L, INPUT); pinMode(FLOOR_R, INPUT);
163     pinMode(Trigger1, OUTPUT); pinMode(Echo1, INPUT);
164     pinMode(Trigger2, OUTPUT); pinMode(Echo2, INPUT);
165     pinMode(Trigger3, OUTPUT); pinMode(Echo3, INPUT);
166     pinMode(LED_SYS, OUTPUT);

```

```

168 // PWM DRV8833
169 ledcAttachChannel(MOT_A1, pwmFreq, pwmRes, ch_A1);
170 ledcAttachChannel(MOT_A2, pwmFreq, pwmRes, ch_A2);
171 ledcAttachChannel(MOT_B1, pwmFreq, pwmRes, ch_B1);
172 ledcAttachChannel(MOT_B2, pwmFreq, pwmRes, ch_B2);
173
174 // Conexión WiFi
175 WiFi.begin(ssid, password);
176 while(WiFi.status() != WL_CONNECTED) delay(500);
177 Serial.println("\n--- ROBO-EDU CONECTADO ---");
178 Serial.println(WiFi.localIP());
179
180 // API
181 server.on("/", [](){ server.send(200, "text/html", index_html); });
182 server.on("/api", [](){
183     String m = server.arg("mode");
184     if(m=="stop") currentMode=IDLE;
185     else if(m=="auto") currentMode=AUTO_LOGIC;
186     else if(m=="follow") currentMode=MODE_FOLLOW;
187     else if(m=="zone") currentMode=MODE_ZONE;
188     else if(m=="demo") currentMode=MODE_DEMO;
189     server.send(200, "text/plain", "OK");
190 });
191     server.begin();
192 }
193
194 ~ void loop() {
195     server.handleClient();

```

```

197 // 1. SEGURIDAD (LECCIÓN: LÓGICA BOOLEANA)
198 // Siempre verificamos si nos salimos de la zona segura (mesa/pista)
199 if (currentMode != IDLE) {
200     if (!digitalRead(FLOOR_L) || !digitalRead(FLOOR_R)) {
201         move(-200, -200); delay(400); // Retroceder
202         move(200, -200); delay(300); // Girar para volver a zona segura
203         // No retornamos, solo corregimos y seguimos la lógica
204     }
205 }
206
207 // 2. CEREBRO DEL ROBOT
208 switch (currentMode) {
209     case IDLE: stopMotors(); break;
210
211     case AUTO_LOGIC:
212         runEducationalFuzzy(); // Tu sistema experto IA
213         break;
214     |
215     case MODE_FOLLOW: // Antes era Ataque, ahora sigue la mano
216         runFollowMode();
217         break;
218
219     case MODE_ZONE: // Modo pacífico, solo evita caerse
220         move(100, 100); // Avanza lento hasta encontrar borde (arriba ya lo gestiona)
221         break;
222
223     case MODE_DEMO:
224         move(150, 150); delay(500);
225         move(-150, -150); delay(500);
226         stopMotors(); currentMode = IDLE;
227         break;

```

```

237    float fuzzyGrade(float val, float low, float peak, float high) {
238        if (val <= low || val >= high) return 0.0;
239        if (val == peak) return 1.0;
240        if (val < peak) return (val - low) / (peak - low);
241        return (high - val) / (high - peak);
242    }
243
244    void runEducationalFuzzy() {
245        long dist = getDist(Trigger2, Echo2);
246        if (dist > 60) dist = 60;
247
248        // FUSIFICACIÓN (Variables Lingüísticas)
249        float isClose = fuzzyGrade(dist, -1, 0, 25); // "Muy cerca"
250        float isMedium = fuzzyGrade(dist, 15, 30, 45); // "Distancia Media"
251        float isFar = fuzzyGrade(dist, 35, 60, 80); // "Lejos"
252
253        // REGLAS (Base de Conocimiento)
254        // 1. Si está CERCA, detenerse suavemente (Precaución)
255        // 2. Si está MEDIO, velocidad de aprendizaje (Observación)
256        // 3. Si está LEJOS, explorar (Curiosidad)
257
258        float ruleCaution = isClose;
259        float ruleObserve = isMedium;
260        float ruleExplore = isFar;
261
262        // DESFUSIFICACIÓN (Cálculo de velocidad PWM)
263        // Cerca -> 0 (Stop), Medio -> 120 (Lento), Lejos -> 180 (Normal)
264        float num = (ruleCaution * 0) + (ruleObserve * 120) + (ruleExplore * 180);
265        float den = ruleCaution + ruleObserve + ruleExplore;
266
267        int speed = (den > 0) ? (int)(num / den) : 100;
268

```

```

269      // ACTUACIÓN
270      if (dist < 10) {
271          stopMotors(); // Parada de seguridad absoluta
272      } else {
273          move(speed, speed);
274      }
275      delay(50);
276  }
277
278  void runFollowMode() {
279      // Lógica Proporcional simple para seguir la mano
280      long d = getDist(Trigger2, Echo2);
281      if (d > 5 && d < 20) {
282          move(150, 150); // Acercarse
283      } else if (d < 5) {
284          move(-150, -150); // Alejarse (Espacio personal)
285      } else {
286          stopMotors(); // Esperar instrucción (mano)
287      }
288  }
289
290  // =====
291 // DRIVERS (DRV8833) & SENSORES
292 // =====
293 void setMotorA(int v) {
294     if(v>0){ ledcWrite(MOT_A1, v); ledcWrite(MOT_A2, 0); }
295     else { ledcWrite(MOT_A1, 0); ledcWrite(MOT_A2, abs(v)); }
296 }
297 void setMotorB(int v) {
298     if(v>0){ ledcWrite(MOT_B1, v); ledcWrite(MOT_B2, 0); }
299     else { ledcWrite(MOT_B1, 0); ledcWrite(MOT_B2, abs(v)); }

```

```

300  }
301  void move(int l, int r) { setMotorA(l); setMotorB(r); }
302  void stopMotors() { move(0,0); }
303
304  long getDist(int t, int e) {
305      digitalWrite(t, LOW); delayMicroseconds(2);
306      digitalWrite(t, HIGH); delayMicroseconds(10);
307      digitalWrite(t, LOW);
308      long d = pulseIn(e, HIGH, 4000);
309      return (d==0)? 99 : d/58.2;
310  }

```

