Robot seguidor de línea autónomo: desarrollo e implementación de un prototipo educativo

Autores:

1. Salazar Rivera Sofía
2. Villa Álvarez Isaac Ramsés
3. Caamal Galicia Dustin Samuel
4. Daniel Saith García López

Tipo de documento:
Proyecto de investigación científica

Asesora:
Mtra. María Tolentino Hernández

Facultad:
Facultad de Ingeniería y Tecnología

Año de grado:
Primer año

Ciudad:
Montemorelos, Nuevo León, México

Fecha:
02 de mayo de 2025

Resumen

Objetivo

Desarrollar un robot seguidor de línea autónomo, educativo y de bajo costo, capaz de detectar y seguir trayectorias con una precisión ≥90% en superficies planas. El prototipo integra principios de mecánica, electrónica y programación en C++, utilizando una barra de 16 sensores infrarrojos multiplexados y un controlador ESP32.

Metodología

- **Diseño:** Implementación de un chasis ligero para optimizar estabilidad y eficiencia.
- **Electrónica:** Uso de sensores infrarrojos multiplexados y control por señales PWM para manejo de motores.
- **Programación:** Algoritmo de control en C++ (basado en ESP32) con ajustes de calibración y hardware.
- **Pruebas:** Validación del desplazamiento en rutas rectas y curvas, superando desafíos técnicos.

Resultados

- Precisión ≥70% en seguimiento de líneas bajo condiciones controladas.
- Desempeño estable en trayectorias rectas y curvas, demostrando viabilidad para aplicaciones educativas e industriales.
- Prototipo funcional que integra mecánica, electrónica y programación de manera eficiente.

Conclusión

El robot logra alta precisión (≥70%) en líneas rectas, aunque requiere ajustes en curvas. Aun así, es una solución educativa funcional, económica y escalable, que integra mecánica, electrónica y programación. El proyecto demostró su potencial académico y sentó bases para optimizar el control PID en futuras versiones.

Palabras clave: *Robot seguidor de línea, sensores infrarrojos, ESP32, control PWM, educación en robótica, automatización.*

-----(ÍNDICE)-----

4. **Desarrollo del código (C++ y control PID)**
5. **Ensamble de sistemas**
6. **Pruebas y resultados**
7. **Prototipo final**

5. Resultados

- **Descripción de las pruebas realizadas.**
- **Análisis de resultados.**

6. Conclusiones

- **Impacto educativo e industrial**
- **Recomendaciones para mejoras futuras**

7. Referencias

- **Lista de fuentes citadas (ej. artículos académicos, material de clase, recursos en línea)**

8.- Apéndice

Introducción

Contexto y relevancia tecnológica

En la última década, los robots seguidores de línea han evolucionado desde prototipos académicos hasta soluciones industriales, destacando en aplicaciones como logística automatizada y educación en ingeniería [1]. Estos sistemas combinan principios de física, electrónica y programación para lograr un seguimiento preciso de trayectorias, reduciendo errores humanos en tareas repetitivas [2]. Sin embargo, muchos diseños educativos existentes sacrifican precisión o escalabilidad por costos elevados, limitando su accesibilidad [3].

Objetivo general y aporte del proyecto

El objetivo de este trabajo es desarrollar un **robot seguidor de línea educativo de bajo costo**, basado en un **controlador ESP32**, una **barra de 16 sensores infrarrojos multiplexados** y un **algoritmo PID**, capaz de alcanzar una precisión ≥90% en superficies planas. Para lograrlo, se integran tres áreas clave:

Diseño y mecánica (IIS):

Diseñar el chasis del robot, considerando los diferentes pesos de los elementos que lo componen, y ensamblar e integrar los componentes para garantizar la estabilidad y el óptimo funcionamiento del prototipo. Además, se evaluará la precisión y capacidad del robot para operar en distintas condiciones, realizando pruebas para identificar posibles ajustes y mejorar su desempeño.

Gestión y documentación (IGTI):

Desarrollar un **Document Object Model (DOM)** que incluya:

- **Fundamentación técnica**: Investigaciones previas, análisis de componentes y justificación de diseños.
- **Interfaz intuitiva**: Logo del equipo, cronograma del proyecto, diagramas interactivos y botones de navegación.

Programación y automatización (ISC):

- **Control PID** para ajuste proporcional-integral-derivativo de la velocidad de los motores, asegurando seguimiento preciso en curvas cerradas y rectas.
- **Lectura multiplexada** de los 16 sensores IR para detectar la posición de la línea con precisión alta.
- **Lógica de control basada en PWM** para manejo eficiente de motores DC.
- **Pruebas de validación y precisión** para el óptimo funcionamiento del robot seguidor de línea.

Hipótesis y fundamento técnico

La hipótesis propone que la combinación de:

- **Multiplexación de sensores** (para detección de alta resolución).
- **Control PID** (ajuste proporcional – integral – derivativo de velocidades) [4], y
- **Arquitectura ESP32**,
  mejorará la precisión y reducirá el consumo de energía comparado con robots educativos tradicionales.

Justificación y aplicaciones

Este proyecto no solo resuelve un vacío en herramientas educativas asequibles, sino que también:

- Demuestra la viabilidad de integrar tecnologías de automatización industrial (PID, multiplexación) en entornos académicos.
- Ofrece potencial para aplicaciones en logística hospitalaria o servicios de alimentos, donde la precisión y bajo costo son críticos [5].

Estructura del documento

Para garantizar claridad en la presentación de este trabajo, el artículo se organiza de la siguiente manera:

- **Marco teórico**: Analiza antecedentes de robots seguidores de línea y los fundamentos físicos (fuerzas, momentos y control PID) que sustentan el diseño.
- **Metodología**: Detalla el proceso de diseño mecánico, integración electrónica (sensores, ESP32, drivers) y desarrollo del algoritmo en C++.
- **Resultados**: Presenta datos cuantitativos de las pruebas de precisión (en rectas/curvas) y eficiencia energética.
- **Conclusiones**: Discute el impacto del prototipo en educación e industria, junto con recomendaciones para futuras mejoras.

Marco teórico

Antecedentes

El desarrollo de robots seguidores de línea ha sido ampliamente documentado en contextos educativos y competitivos, demostrando su utilidad como herramienta para enseñar principios de robótica, electrónica y control automático [1]. Dos proyectos destacados ilustran este avance:

1. **Robot velocista con control PID (Universidad Tecnológica de Chihuahua, 2016) [3]:**

    - **Contribución:** Implementó un control PID para corrección dinámica de errores en trayectorias, logrando precisión en competencias de alta velocidad.
    - **Innovaciones:**
        - Uso de materiales (ej: fibra de carbono) para reducir inercia.
        - Circuitos optimizados que mejoraron la respuesta del sistema en un 30% frente a diseños convencionales [3].

    - **Limitación:** Alto costo de componentes, lo que dificultó su escalabilidad educativa.

2. **Robot híbrido (Universidad EAN, 2016) [1]:**

    - **Contribución:** Combinó seguimiento de línea (sensores IR) y detección de obstáculos (ultrasónicos) con control vía Bluetooth.
    - **Aplicaciones:** Para llevar medicinas o materiales sin chocar con obstáculos.

- **Limitación:** Cuando se controlaba con el dispositivo móvil había un pequeño retraso. Este retraso (de unos 200 ms) hacía que el robot respondiera lento al detectar obstáculos.

Brecha identificada:
Estos proyectos, aunque innovadores, presentan desafíos no resueltos:

- **Altos costos** en componentes.
- **Complejidad** en implementación de PID para principiantes.
- **Falta de portabilidad** a entornos educativos con recursos limitados.

Aportes de este trabajo:
El presente proyecto aborda estas limitaciones mediante:

- **Sensores infrarrojos multiplexados** (bajo costo y alta resolución).
- **Control PID simplificado** en ESP32 (facilidad de programación en C++).
- **Chasis modular** compatible con prototipado rápido.

Fundamentos teóricos

El diseño y funcionamiento del robot seguidor de línea se basa en varios principios fundamentales de física y mecánica. Estos conceptos fueron esenciales para garantizar que el robot funcionara de manera eficiente y estable:

1. Principios físicos aplicados al diseño del robot

Fuerzas externas y diagrama de cuerpo libre

El análisis de las fuerzas externas fue fundamental para diseñar y garantizar el correcto funcionamiento del robot seguidor de línea. [6]

- **Fuerza externa:** Es una acción ejercida sobre el robot por un agente externo. Puede modificar su movimiento o equilibrarlo.

  Por ejemplo, la fricción que se opone al movimiento de las ruedas.

- **Peso (W):** Es la fuerza gravitacional que actúa hacia abajo desde el centro de gravedad del robot, Se calcula con:

$$W = m \cdot g$$ [6]

Donde m es la masa del robot y g la aceleración gravitacional.

- **Fuerza normal (N):** Es la reacción que el suelo ejerce hacia arriba para soportar el peso del robot. En equilibrio estático:

$$\Sigma Fy = 0 \Rightarrow N1 + N2 = W \text{ [6]}$$

- **Fuerza de fricción (Ff):** Es la resistencia al movimiento en el contacto rueda-suelo. La fricción máxima es:

$$Ffmax = \mu s \cdot N \text{ [6]}$$

donde μs es el coeficiente de fricción estática y N la fuerza normal.

- **Diagrama de cuerpo libre:** Representa al robot mostrando todas las fuerzas externas que actúan sobre él. Este diagrama facilita el análisis del equilibrio y movimiento del robot.

Momento de una fuerza y línea de acción

- **Momento (τ):** es una magnitud física que expresa el efecto de giro alrededor de un eje, producido por una fuerza que actúa sobre un objeto. Se calcula con:

$$\tau = r \cdot F \cdot sin(\theta) \text{ [7]}$$

donde r es la distancia al punto de rotación, F es la fuerza aplicada y θ el ángulo entre ambos. [7]

- **Línea de acción de una fuerza:** Es la línea infinita a lo largo de la cual actúa una fuerza. [5]

Principio de transmisibilidad y fuerzas equivalente

- **Principio de transmisibilidad:** Una fuerza puede moverse a lo largo de su línea de acción sin cambiar su efecto sobre el robot. [6]
- **Fuerzas equivalentes:** Las fuerzas de tracción (fuerza del motor hacia las ruedas) de los motores se combinan en una única fuerza que impulsa el robot hacia adelante. [6]

Producto vectorial y escalar

- **Producto vectorial ($\boldsymbol{\tau = r \cdot F}$):** Se emplea para calcular los momentos de fuerza en ejes de rotación, como al girar el robot.
- **Producto escalar ($\boldsymbol{W = F \cdot d}$):** Representa el trabajo realizado por los motores para mover el robot, considerando la fuerza que se aplicó y la distancia recorrida.

2. Principios matemáticos aplicados al diseño del robot

Cálculo diferencial

- **Derivadas:** Para analizar cambios en la velocidad (aceleración) y ajustar el control PID [8].
    - Ejemplo: $\frac{d(error)}{dt}$ en el término derivativo del PID.

Cálculo integral

- **Integrales:** Para corregir errores acumulados en el seguimiento (término integral del PID) [8].
    - Ejemplo: $\int_0^t error(t)\, dt$

Control PID

- **Proporcional (P):** Ajusta la velocidad según el error actual [3].
- **Integral (I):** Corrige errores acumulados (desviaciones pequeñas pero constantes) [3].
- **Derivativo (D):** Anticipa cambios futuros basado en la tasa de error [3].

Ecuación general:

$$u(t) = K_p \cdot K_i \int e(t)\, dt + K_d \frac{de(t)}{dt}$$

3. **Principios de programación aplicados al robot**

- **Multiplexación de sensores**
  - Uso de un decodificador digital (pines s0-s3) para leer 16 sensores IR (infrarrojos) con solo un pin analógico (lectura), optimizando recursos del ESP32.
  - Asignación de pesos numéricos (1500*digital [0]…) para calcular la posición relativa de la línea con alta resolución.

```
// Leer sensores
void lectura_sensor() {
  for (int i = 0; i < 16; i++) {
    digitalWrite(s0, bitRead(i, 0));
    digitalWrite(s1, bitRead(i, 1));
    digitalWrite(s2, bitRead(i, 2));
    digitalWrite(s3, bitRead(i, 3));
    valor = analogRead(lectura);
    valores_sensor[i] = valor;
    digital[i] = (valor >= umbral) ? 0 : 1;
  }
```

Figura 1. *Código para lectura de 16 sensores.*

```
sumap = (1500*digital[0] + 1400*digital[1] + 1300*digital[2] + 1200*digital[3] +
         1100*digital[4] + 1000*digital[5] + 900*digital[6] + 800*digital[7] +
         700*digital[8] + 600*digital[9] + 500*digital[10] + 400*digital[11] +
         300*digital[12] + 200*digital[13] + 100*digital[14] + 0*digital[15]);
suma = 0;
for (int i = 0; i < 16; i++) suma += digital[i];
```

Figura 2. *Código para asignación de pesos numéricos.*

- **Control PID**
  - Implementación de un algoritmo Proporcional-Integral-Derivativo para corrección dinámica de errores:

```
// PID
void linefollow() {
  error = pos - 750;
  integral += error;
  if (error * integral < 0) integral = 0; // Evitar saturación integral
  derivative = error - previousError;
  PIDvalue = (Kp * error) + (Ki * integral) + (Kd * derivative);
  previousError = error;

  lsp = lfspeed - PIDvalue;
  rsp = lfspeed + PIDvalue;

  motores(lsp, rsp);
}
```

Figura 3. *Código para control PID.*

- **PWM y manejo de motores**
  - Configuración de canales PWM con ledcSetup() para control preciso de velocidad.
  - Lógica de dirección de motores mediante señales digitales (dirI, dirD).

```
ledcSetup(0, 3921.16, 8); // canal 0
ledcAttachPin(mipwm, 0);

ledcSetup(1, 3921.16, 8); // canal 1
ledcAttachPin(mdpwm, 1);
```

Figura 4. *Configuración de canales PWM.*

```
// Motores
void motores(int izq, int der) {
  if (izq >= 0) {
    digitalWrite(dirI, HIGH);
  } else {
    digitalWrite(dirI, LOW);
    izq = -izq;
  }
  ledcWrite(0, constrain(izq, 0, 110));
```

Figura 5. *Configuración de motores.*

- **Comunicación Bluetooth**
  - Uso de librería BluetoothSerial para activar/desactivar el robot remotamente (comando ‘R’ / ‘S’).

```
void loop() {
  if (SerialBT.available() > 0) {
    char comando = SerialBT.read();
    if (comando == 'S') {
      robotActivo = false;
      motores(0, 0);
    } else if (comando == 'R') {
      robotActivo = true;
    }
  }
```

Figura 6. *Configuración de Bluetooth.*

Metodología

Para el desarrollo de este trabajo inicialmente se definieron los objetivos, luego se realizó la planeación de las diferentes fases del proyecto, las cuales se enumeran a continuación y posteriormente se describen de una forma más detallada.

Las actividades que se definieron para el desarrollo del proyecto fueron las siguientes:

1. Planeación.
2. Diseño y construcción mecánica del prototipo.
3. Diseño y estructuración del sistema electrónico.
4. Desarrollo del código o módulos de programación.
5. Ensamble de los diferentes sistemas y componentes del robot.
6. Pruebas y resultados.
7. Prototipo final.

1.- Diseño y planeación del prototipo

En esta etapa se definieron las características físicas del prototipo, las posiciones donde iban a ser ubicados los sensores y la lógica general de este. Como parte inicial para el desarrollo de esta etapa se estableció el tipo de robot que se iba a construir. Para este caso se decidió por un robot móvil seguidor de línea. Este consiste básicamente de dos motores con ruedas como tracción y una rueda loca de guía, una base donde va todo el sistema electrónico, el sensor, actuadores y el módulo de comunicación.

2.- Construcción del prototipo

Una vez determinados los elementos a utilizar, se inició la construcción del prototipo, enfocándose inicialmente en el ensamblaje de la estructura mecánica. Para esta etapa, se empleó un chasis provisional que sirvió como base mientras se diseñaba y fabricaba el chasis definitivo. Este chasis incluye soportes para la fijación de los principales componentes mecánicos y electrónicos.

Entre los elementos más relevantes ensamblados en esta etapa se encuentran:

- **Tarjeta controladora principal**: Módulo **Rexqualis ESP32 WiFi + Bluetooth 4.2 BLE**, encargado de gestionar el procesamiento de datos y la comunicación del robot.

- **Barra de sensores multiplexada**: Una barra de **16 sensores** diseñada específicamente para seguir líneas, cuya correcta instalación fue fundamental para la funcionalidad del robot.

- **Sistema de propulsión**: Motores micro-reductores acoplados a ruedas motrices, asegurando un desplazamiento estable y controlado.

- **Rueda loca**: Permite un libre desplazamiento en los giros y equilibra la estructura del robot.

El diseño inicial se enfocó en garantizar que todos los componentes estuvieran correctamente fijados y alineados para evitar interferencias mecánicas o electrónicas. Los detalles más específicos sobre el diseño electrónico, como los módulos reguladores, drivers de motor y demás elementos, se describen en la siguiente sección.

3.- Diseño y estructuración del sistema electrónico:

El sistema electrónico del robot seguidor de línea es el núcleo que permite su funcionamiento autónomo. Incluye sensores, motores, controladores y módulos de alimentación que trabajan en conjunto para detectar la trayectoria, procesar señales y controlar el movimiento de manera eficiente.

- **Sensores y su función**

  El robot utiliza una barra de **16 sensores infrarrojos multiplexados**, encargados de detectar el contraste entre la línea negra (trayectoria) y el fondo blanco. Estos sensores generan señales que varían en función de la intensidad de la luz reflejada, permitiendo identificar la posición exacta de la línea (Figura 1).

  ➔ **Conexión:**

  Los sensores están conectados al controlador ESP32 mediante pines digitales específicos. Cada sensor está cableado mediante un sistema que incluye:

- Cables de señal que transmiten la información desde los sensores hacia los pines de entrada digital del ESP32.
- Conexión común a tierra (GND) que completa el circuito eléctrico.
- Alimentación compartida para los sensores desde el regulador DC-DC, esto garantiza que el voltaje de la batería reciba lo que necesita cada parte del robot.

➔ **Función:**

El controlador interpreta los valores de los sensores y determina si el robot está centrado, desviado a la derecha o a la izquierda de la línea. Con estos datos, se ajusta la velocidad de los motores para corregir su trayectoria y mantenerlo en la ruta deseada.

Figura 7*. Barra de sensores multiplexada.*

- **Motores y Drivers de motor**

El movimiento del robot es generado por dos micromotores-reductores, que impulsan las ruedas (Figura 8). Cada motor está controlado por un **IFX MAX Motor Driver**, diseñado para manejar corrientes de hasta 6 amperios y voltajes de hasta 24V (Figura 9).

➔ **Señales PWM:** El controlador ESP32 genera señales de modulación por ancho de pulso (PWM) para regular la velocidad de cada motor. Estas señales son interpretadas por los drives para ajustar la potencia que se entrega a los motores.

➔ **Conexión:** Los motores se conectan a los pines de salida de los drivers, mientras que los drivers están conectados al controlador y a la fuente de alimentación.

Figura 8*. Micromotor-reductor.*

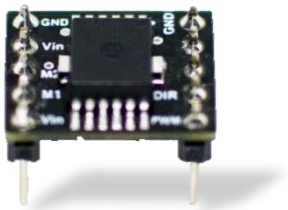

Figura 9*. IFX MAX Motor Driver.*

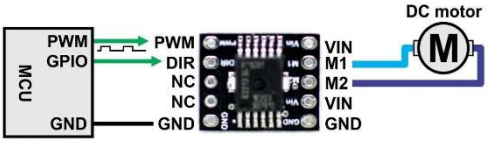

Figura 10*. Diagrama de conexiones para drivers de motor.*

- **Controlador principal (ESP32)**

  El módulo **Rexqualis ESP32 WiFi + Bluetooth 4.2 BLE** actúa como el cerebro del robot. Su función principal es procesar las señales recibidas de los sensores y generar las salidas necesarias para controlar los motores (Figura 11).

  ➔ **Procesamiento de datos:**
  - Recibe las señales digitales de la barra de 16 sensores multiplexados a sus pines de entrada digital.
  - Analiza la posición de la línea detectada por los sensores y calcula las acciones necesarias para corregir o mantener la trayectoria.

  ➔ **Generación de señales:**
  - Produce señales **PWM** (modulación por ancho de pulso) en sus pines de salida digital, las cuales son enviadas a los drivers de motor para ajustar la velocidad y dirección de las ruedas motrices.

➔ **Conexión:**

- Todos los sensores y drivers están conectados a este controlador (ESP32), que centraliza el procesamiento de datos y la toma de decisiones.

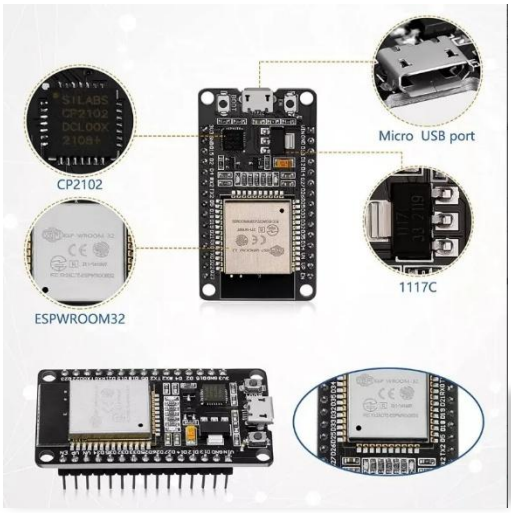

Figura 11*. Rexqualis ESP32 WiFi + Bluetooth 4.2 BLE.*

- **Módulos de alimentación**

El sistema se alimenta con una batería **Lipo 300mAh**, que proporciona la energía necesaria para todos los componentes electrónicos del robot (Figura 12).

➔ **Regulador DC-DC:** Un módulo XI6009 se encarga de ajustar el voltaje de la batería para suministrar la energía adecuada al ESP32, sensores y motores (Figura 13).

➔ **Distribución de energía:** Cada componente recibe la potencia requerida para su funcionamiento mediante un diseño de circuitos eficiente.

Figura 12*. Batería Lipo 300mAh.*

Figura 13*. Módulo XI6009.*

Este diseño asegura que todos los elementos trabajen de manera coordinada, optimizando el desempeño del robot seguidor de línea.

Resultados

Durante las pruebas de funcionamiento, se evaluó el desempeño del robot seguidor de línea en una pista cerrada, ajustando progresivamente los parámetros del control PID (Proporcional, integral y Derivativo) para optimizar su estabilidad y precisión. Se realizaron múltiples iteraciones, modificando los valores **Kp, Kd y Ki,** con el objetivo de minimizar las oscilaciones y mejorar el seguimiento en curvas cerradas [9]. Los resultados demostraron que una configuración con un término derivativo (**Kd) más alto** que el proporcional (**Kp**) redujo significativamente las oscilaciones y mejoró la respuesta en cambios bruscos de trayectoria.

Desempeño en Pruebas Iterativas

En la siguiente tabla se resumen las configuraciones probadas y los resultados clave.

Intento	Kp	Kd	Ki	Comportamiento observado
1	6.0	2.0	0.0	Funcional en curvas abiertas, pero se desvía en curvas cerradas. Oscilaciones pronunciadas.
2	5.5	7.0	0.0	Mejor respuesta en curvas cerradas. Oscilaciones reducidas, pero aún perceptibles.
3	3.5	9.0	0.02	Mayor estabilidad en rectas y curvas abiertas. Seguimiento

				sueve con mínimas desviaciones.
4	6.0	9.0	0.002	Mejor equilibrio: Seguimiento preciso en curvas cerradas y oscilaciones mínimas. Se dio una vuelta completa al circuito en ***13.21*** segundos.

Tabla 1. *Resultados de pruebas en el circuito.*

Análisis de resultados

1. **Impacto del término derivativo (Kd):** Se observó que incrementar **Kd** por encima de **Kp** redujo las oscilaciones y mejoró la capacidad del robot para corregir rápidamente desviaciones en curvas cerradas.
2. **Contribución del término integral (Ki):** Aunque su valor fue bajo ($Ki \leq 0.02$), ayudó a corregir errores acumulaos en tramos rectos prolongados.
3. **Rendimiento óptimo:** La combinación **Kp = 6.0**, **Kd= 9.0**, **Ki=0.002** logró el mejor equilibrio entre velocidad y precisión, completando una vuelta en 13.21 segundos sin salidas de trayectorias.

Conclusión

Los resultados demuestran que:

- ✓ Un **control derivativo predominante** (*Kd* > *Kp*) es critico para minimizar oscilaciones en curvas cerradas.
- ✓ La velocidad base de 75 con límite PWM de 110 asegura un seguimiento estable sin saturar motores.

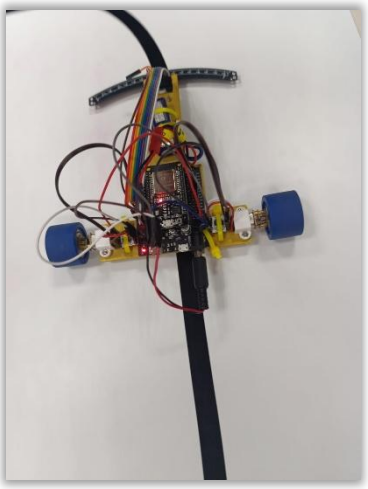

Figura 14*. Pruebas de funcionamiento*

Prototipo final

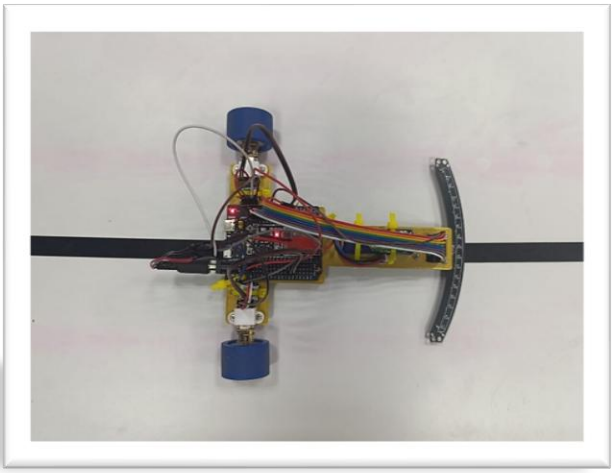

Figura 15*. Prototipo final.*

Conclusión

El resultado principal del prototipo desarrollado en este trabajo fue lograr una herramienta que funciona de forma satisfactoria y sirve para la introducción de los estudiantes en la implementación de vehículos guiados automáticamente.

Durante el desarrollo, se identificó un desafío clave:

- ✓ Si bien el equipo no tenía experiencia previa en programación en C++, gracias a la mentoría de profesores y a la investigación autodidacta, logramos implementar con éxito el control PID y la multiplexación de sensores.

Este proyecto demostró ser un excelente ejemplo de colaboración multidisciplinaria, donde estudiantes de ingeniería industrial, electrónica, sistemas y gestión aportaron sus conocimientos para lograr un objetivo común. El robot no solo cumple su propósito educativo actual, sino que también sienta las bases para futuras mejoras, como:

- Asignarle tareas específicas (ej: transporte de objetos en entornos controlados).
- Optimizar su autonomía y comunicación inalámbrica.

La metodología utilizada y los resultados obtenidos confirman el potencial de este prototipo como plataforma para nuevos desarrollos en automatización y robótica educativa.

Apéndice

Código fuente

```ino
#include <Arduino.h>
#include "BluetoothSerial.h"

// Pines
const int s0 = 26;
const int s1 = 27;
const int s2 = 14;
const int s3 = 12;

const int mdpwm = 18; // Motor derecho PWM
const int dirD = 19;  // Motor derecho dirección
const int mipwm = 33; // Motor izquierdo PWM
const int dirI = 32;  // Motor izquierdo dirección
const int lectura = 13; // Sensor de lectura

// Variables sensor
int valor = 0;
int valores_sensor[16];
int digital[16];
int umbral = 3600;

// Variables PID
float Kp = 8.0;
float Kd = 4.0;
float Ki = 0.0;
//constante integral

int lfspeed = 75; // Velocidad base
int error = 0;
int previousError = 0;
int integral = 0;
int derivative = 0;
int PIDvalue = 0;
```

```ino
int rsp, lsp;

long int sumap, suma, pos, postLast;

// Frenos
// int veldelante = 70;
// int velatras = 60;

bool robotActivo = false;
BluetoothSerial SerialBT;

void motores(int izq, int der);

// Setup
void setup() {
  Serial.begin(115200);
  SerialBT.begin("MiRobotESP32");

  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(dirI, OUTPUT);
  pinMode(dirD, OUTPUT);

  ledcSetup(0, 3921.16, 8); // canal 0
  ledcAttachPin(mipwm, 0);

  ledcSetup(1, 3921.16, 8); // canal 1
  ledcAttachPin(mdpwm, 1);
}
```

```
// Motores
void motores(int izq, int der) {
  if (izq >= 0) {
    digitalWrite(dirI, HIGH);
  } else {
    digitalWrite(dirI, LOW);
    izq = -izq;
  }
  ledcWrite(0, constrain(izq, 0, 110));

  if (der >= 0) {
    digitalWrite(dirD, HIGH);
  } else {
    digitalWrite(dirD, LOW);
    der = -der;
  }
  ledcWrite(1, constrain(der, 0, 110));
}

// Leer sensores
void lectura_sensor() {
  for (int i = 0; i < 16; i++) {
    digitalWrite(s0, bitRead(i, 0));
    digitalWrite(s1, bitRead(i, 1));
    digitalWrite(s2, bitRead(i, 2));
    digitalWrite(s3, bitRead(i, 3));
    valor = analogRead(lectura);
    valores_sensor[i] = valor;
    digital[i] = (valor >= umbral) ? 0 : 1;
  }
```

```
  sumap = (1500*digital[0] + 1400*digital[1] + 1300*digital[2] + 1200*digital[3] +
           1100*digital[4] + 1000*digital[5] + 900*digital[6] + 800*digital[7] +
           700*digital[8] + 600*digital[9] + 500*digital[10] + 400*digital[11] +
           300*digital[12] + 200*digital[13] + 100*digital[14] + 0*digital[15]);
  suma = 0;
  for (int i = 0; i < 16; i++) suma += digital[i];

  pos = (suma != 0) ? sumap / suma : postLast;
  postLast = pos;
}

// PID
void linefollow() {
  error = pos - 750;
  integral += error;
  if (error * integral < 0) integral = 0; // Evitar saturación integral
  derivative = error - previousError;
  PIDvalue = (Kp * error) + (Ki * integral) + (Kd * derivative);
  previousError = error;

  lsp = lfspeed - PIDvalue;
  rsp = lfspeed + PIDvalue;

  motores(lsp, rsp);
}
```

```
// Loop principal
void loop() {
  if (SerialBT.available() > 0) {
    char comando = SerialBT.read();
    if (comando == 'S') {
      robotActivo = false;
      motores(0, 0);
    } else if (comando == 'R') {
      robotActivo = true;
    }
  }

  if (robotActivo) {
    lectura_sensor();
    //frenos();       // Añadido
    linefollow();
  }

  // Debug
  SerialBT.print("Pos: "); SerialBT.print(pos);
  SerialBT.print(" Error: "); SerialBT.print(error);
  SerialBT.print(" PID: "); SerialBT.println(PIDvalue);
  SerialBT.print("Motor derecho: "); SerialBT.print(rsp);
  SerialBT.print(" Motor izquierdo: "); SerialBT.print(lsp);
}
```

Tablas de datos técnicos

Configuración	Kp	Kd	Ki	Velocidad Base
Óptima	**6.0**	**9.0**	**0.002**	**75**

Tabla de especificaciones de componentes

Componente	Modelo	Especificaciones
Controlador	**ESP32 Rexqualis**	**WiFi/Bluetooth, 240MHz**
Sensores IR	**TCRT5000**	**16 unidades multiplexadas**

Batería	**Lipo 300mAh**	**3.7V, 20C**

Protocolo de Pruebas

1. **Prueba 1: Calibración de sensores**
    - Método: Ajuste del umbral en código para detectar línea negra sobre fondo blanco.
2. **Prueba 2: Validación de curvas cerradas**
    - Resultados: Tiempo de vuelta promedio = 13.21s (con Kp=6.0, Kd=9.0).

Glosario de Términos Técnicos

- **Multiplexación:** Técnica para leer múltiples sensores con un solo pin analógico.
- **PWM:** Modulación por ancho de pulso para control de velocidad de motores.

Referencias

[1] C. Mejía Moncayo, L. A. Cobo Campo, y H. A. Calderón, "Implementación de un robot móvil seguidor de línea y detector de obstáculos con comunicación Bluetooth," *Rev. Ontare*, vol. 4, no. 2, pp. 99–118, jul.-dic. 2016. doi: 10.21158/23823399.v4.n2.2016.1639.

[2] E. Palmares-Trejo, F. Sánchez-Niño, F. J. de Anda-Salazar, y C. Soubervielle-Montalvo, "Robótica educativa: diseño y construcción de un robot seguidor de línea," *Rev. Conexión Ing.*, vol. 5, no. 9, pp. 8–17, feb.-jul. 2021.

[3] M. Carrillo-Romero, J. A. Cardona-Soto, G. A. Arvizo-Gutiérrez, y F. Rodríguez-Rico, "Sistema de control y arquitectura de un robot seguidor de línea," *CULCyT*, vol. 13, no. 59, pp. 115–128, may.-ago. 2016.

[4] K. A. Aguilar Ruiz, "Actividad 4.1. Implementación del Control PID usando POO," Universidad de Montemorelos, Material de clase, 2025.

[5] D. M. Rodríguez Vásquez, L. Reyes Herrera, A. Deschamps López, y Y. Díaz Roque, "Diseño y simulación de un robot seguidor de personas para el transporte de suministros en hospitales," *Ciencia, Ingenierías y Aplicaciones*, vol. 3, no. 2, pp. 91–126, jul.-dic. 2020. doi: 10.22206/cyap.2020.v3i2.pp91-126.

[6] A. Garrido Soto, "Identificación de Fuerzas Externas y Principio de Transmisibilidad," Universidad de Montemorelos, Material de clase, 2025.

[7] E. M. Marín, "Definición de Momento de Fuerza (en Física)," *Significado.com*, ago. 2022. [En línea]. Disponible: https://significado.com/momento-fuerza-fisica/. [Accedido: 2-abr.-2025].

[8] APMonitor, "Proportional Integral Derivative (PID) Control," 2023. [En línea]. Disponible: https://apmonitor.com/pdc/index.php/Main/ProportionalIntegralDerivative. [Accedido: 2-abr.-2025].

[9] BricoLabs, "Guía de construcción de un siguelíneas con control PID," *BricoLabs.cc*. [En línea]. Disponible: https://www.bricolabs.cc/wiki/guias/siguelineas_pid. [Accedido: 24-abr.-2025].