"Año de la recuperación y consolidación de la economía peruana"

UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Escuela Profesional de Ciencia de la Computación

Robótica

Implementación de un Controlador PID para un Robot Seguidor de Línea en el Simulador Webots

---

Estudiante:
Katherine Nikole Béjar Román

Docente:
Percy Maldonado Quispe

16 de octubre de 2025

Índice

1. Introducción

En este proyecto se implementó un **robot seguidor de línea** en el simulador **Webots R2025a**, utilizando el robot **E-puck** y un controlador en lenguaje **C**. El objetivo fue diseñar un sistema de control que permita al robot seguir una línea negra sobre un fondo blanco, haciendo uso de sensores de distancia infrarrojos y aplicando un **control PID** para mejorar la estabilidad y precisión del movimiento.

Este tipo de robot tiene aplicaciones en la industria, la logística y la robótica educativa, ya que ejemplifica el principio de navegación autónoma mediante realimentación sensorial y control retroalimentado.

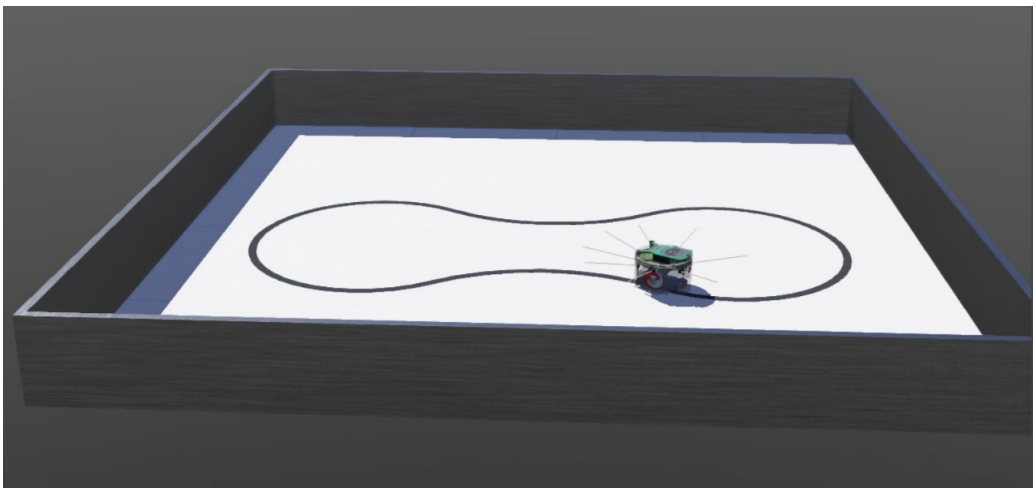

Figura 1: Simulación del robot E-puck siguiendo la línea negra en Webots.

2. Entorno de simulación y configuración inicial

El entorno de simulación se desarrolló en **Windows 11** con el software Webots R2025a. Se creó un proyecto llamado `my_project_line_follower`, compuesto por los siguientes elementos:

- **RectangularArena:** superficie blanca que representa el piso.
- **TrackOne.proto:** pista negra corregida, reemplazando el nodo `Solid` por `Transform` para evitar errores de compilación.
- **E-puck:** robot con sensores infrarrojos y motores diferenciales.
- **Controlador en C:** archivo `my_controller_line_follower.c`.

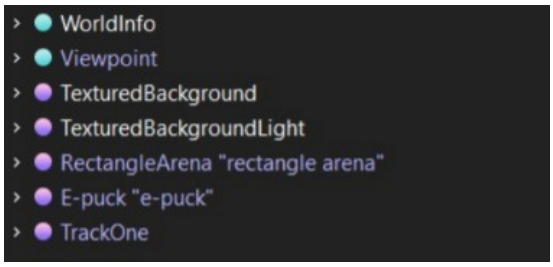

Figura 2: Estructura del mundo en Webots con los componentes principales.

3. Configuración del robot y sensores

El robot E-puck utiliza dos **motores diferenciales** y múltiples **sensores infrarrojos**. En este proyecto se emplearon los sensores `left_sensor` y `right_sensor`, configurados para detectar la diferencia de reflectancia entre la línea negra y el fondo blanco. Cuando un sensor detecta menor reflectancia (valor más bajo), significa que está sobre la línea negra.

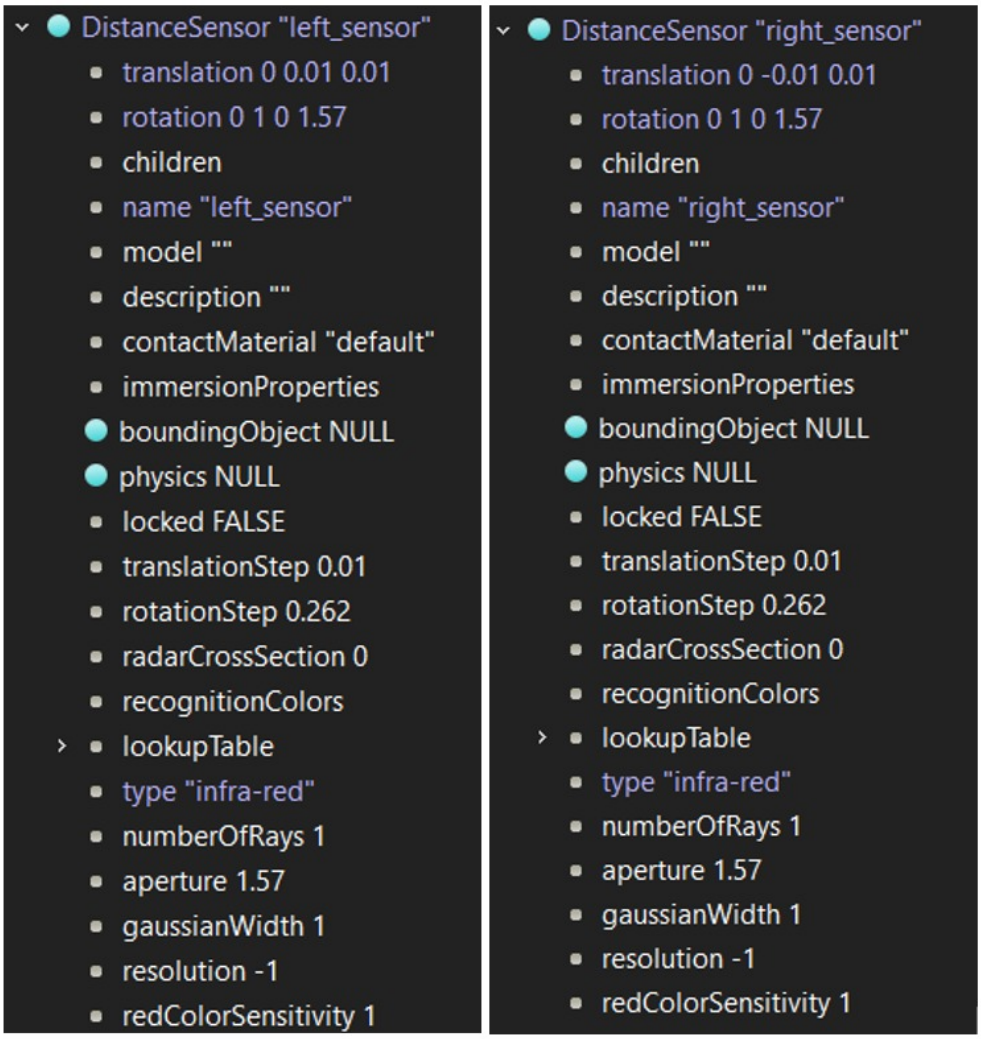

Figura 3: Configuración de los sensores infrarrojos en el robot E-puck.

4. Desarrollo del Controlador PID en C

El controlador PID (Proporcional–Integral–Derivativo) se implementó en lenguaje C para ajustar la velocidad de los motores del robot en función de las lecturas de los sensores IR. El PID calcula un valor de corrección basado en tres componentes:

$$u(t) = K_p \cdot e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{1}$$

donde:

- $e(t)$: error actual (diferencia entre las lecturas de los sensores izquierdo y derecho).
- K_p: ganancia proporcional (corrige el error presente).
- K_i: ganancia integral (corrige el error acumulado en el tiempo).
- K_d: ganancia derivativa (corrige cambios bruscos en el error).

El valor de salida $u(t)$ se usa para ajustar la velocidad de cada motor:

$$v_{izq} = v_{base} - u(t) \qquad v_{der} = v_{base} + u(t) \tag{2}$$

```c
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/distance_sensor.h>
#include <stdio.h>

#define TIME_STEP 64

int main(int argc, char **argv) {
  wb_robot_init();

  WbDeviceTag motor_derecho = wb_robot_get_device("right wheel motor");
  WbDeviceTag motor_izquierdo = wb_robot_get_device("left wheel motor");

  wb_motor_set_position(motor_derecho, INFINITY);
  wb_motor_set_position(motor_izquierdo, INFINITY);

  WbDeviceTag sensor_derecho = wb_robot_get_device("right_sensor");
  WbDeviceTag sensor_izquierdo = wb_robot_get_device("left_sensor");

  wb_distance_sensor_enable(sensor_derecho, TIME_STEP);
  wb_distance_sensor_enable(sensor_izquierdo, TIME_STEP);

  double Kp = 0.4, Ki = 0.01, Kd = 0.2;
  double error = 0.0, error_anterior = 0.0, error_integral = 0.0;
  double v_base = 2.0;

  while (wb_robot_step(TIME_STEP) != -1) {
    double lectura_derecha = wb_distance_sensor_get_value(sensor_derecho
    );
    double lectura_izquierda = wb_distance_sensor_get_value(
    sensor_izquierdo);

    error = lectura_izquierda - lectura_derecha;
```

```
32      error_integral += error;
33      double derivada = error - error_anterior;
34      double salida = Kp * error + Ki * error_integral + Kd * derivada;
35
36      double v_izq = v_base - salida;
37      double v_der = v_base + salida;
38
39      if (v_izq > 6.28) v_izq = 6.28;
40      if (v_der > 6.28) v_der = 6.28;
41      if (v_izq < -6.28) v_izq = -6.28;
42      if (v_der < -6.28) v_der = -6.28;
43
44      wb_motor_set_velocity(motor_izquierdo, v_izq);
45      wb_motor_set_velocity(motor_derecho, v_der);
46
47      printf("Error: %.2f | Salida PID: %.2f\n", error, salida);
48
49      error_anterior = error;
50    }
51
52    wb_robot_cleanup();
53    return 0;
54  }
```

Listing 1: Código en C del controlador PID para seguimiento de línea en Webots.

Explicación del código

El código implementa las tres componentes del PID:

- La parte proporcional (K_p) corrige el error actual.
- La parte integral (K_i) acumula errores pasados para evitar desviaciones persistentes.
- La parte derivativa (K_d) suaviza los movimientos y evita oscilaciones.

Los valores iniciales utilizados fueron:

$$K_p = 0{,}4, \quad K_i = 0{,}01, \quad K_d = 0{,}2$$

Estos parámetros se ajustaron de forma empírica hasta lograr un seguimiento estable de la línea.

5. Resultados y Observaciones

Durante las simulaciones, el robot logró seguir correctamente la línea negra sobre fondo blanco. El uso del PID permitió realizar correcciones suaves y rápidas al detectar curvas o pequeñas desviaciones.

- Con el componente derivativo, se redujeron las oscilaciones en las curvas.
- El término integral ayudó a mantener el robot centrado en trayectorias largas.
- Los ajustes de K_p, K_i y K_d influyen directamente en la precisión y estabilidad.

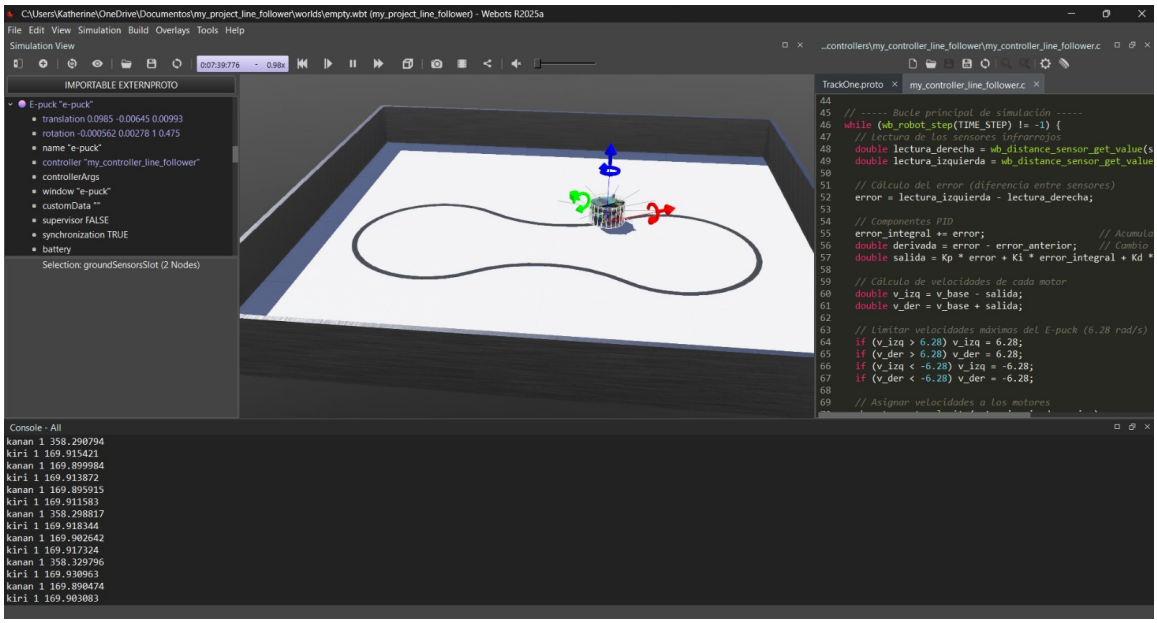

Figura 4: El robot E-puck siguiendo la línea negra utilizando el control PID.

Repositorio del proyecto:
Disponible en: `https://github.com/Katy-Bejar/line-follower-e-puck`

6. Conclusiones

El proyecto permitió comprender la implementación práctica de un **control PID** en un entorno de simulación robótica. Este controlador demostró ser eficaz para mantener el seguimiento de una línea con estabilidad y precisión.

- El control PID mejora significativamente el comportamiento del robot frente a un control proporcional simple.
- Cada componente del PID cumple una función específica: K_p reacciona, K_i corrige, y K_d estabiliza.
- Webots es una herramienta ideal para probar y ajustar algoritmos de control antes de implementarlos en robots reales.

Referencias

[1] Cara Mudah Membuat Line Follower Robot dengan e-puck di Webots Menggunakan Bahasa C. Disponible en: `https://www.youtube.com/watch?v=5BnS4dQL-Ww`