

UTN- Regional Buenos Aires Teoría de

Control - 2024



Propuesta Trabajo Práctico: Robot Velocista

Apellido, Nombre	Legajo
Martinez, Guido	1680882
Dyzenchauz, Theo	2035832

Fecha Presentación: 16/12/2024

Curso: K4571

Docente: Omar Oscar Civale

Índice

Índice.....	2
Introducción.....	3
Objetivo.....	3
Alcance.....	4
Contexto.....	4
Entrada.....	4
Elemento de Medición.....	4
Controlador.....	4
Variable Controlada.....	4
Variable de Referencia o Set Point.....	4
Señal de Error.....	4
Error en Estado Estable (ess).....	5
Factores clave que afectan el error en estado estable:.....	5
Elemento de Control.....	6
Parámetros del controlador.....	6
Constantes.....	7
Proceso.....	7
Perturbaciones.....	7
Impacto de las perturbaciones.....	8
Diagrama de Bloques del Sistema.....	8
Componentes.....	9
Modelización y Simulación.....	13
Modelo de Simulación.....	13
Parámetros Variables.....	13
Resultados a Comparar.....	13
Implementación.....	13
Escenario 1.....	14
Interpretación.....	14
Escenario 2.....	15
Interpretación.....	16
Escenario 3.....	16
Interpretación.....	17
Escenario 4.....	17
Interpretación.....	18
Conclusiones.....	18
Anexos - Hojas de datos.....	19
Bibliografía.....	21

Introducción

En este trabajo práctico, se presenta el desarrollo de un robot velocista programado con Arduino para competir en una carrera donde el objetivo es que el robot siga una pista guiado por una línea predeterminada de la forma más rápida posible. El robot cuenta con sensores de seguimiento de línea que le permiten detectar la posición de la misma y realizar correcciones en su trayectoria para mantener la línea en el centro.

La competencia implica que el robot pueda adaptarse a diferentes trayectorias, como rectas, zigzags y curvas cerradas, utilizando un sistema de control en lazo cerrado para ajustar su rumbo en tiempo real.

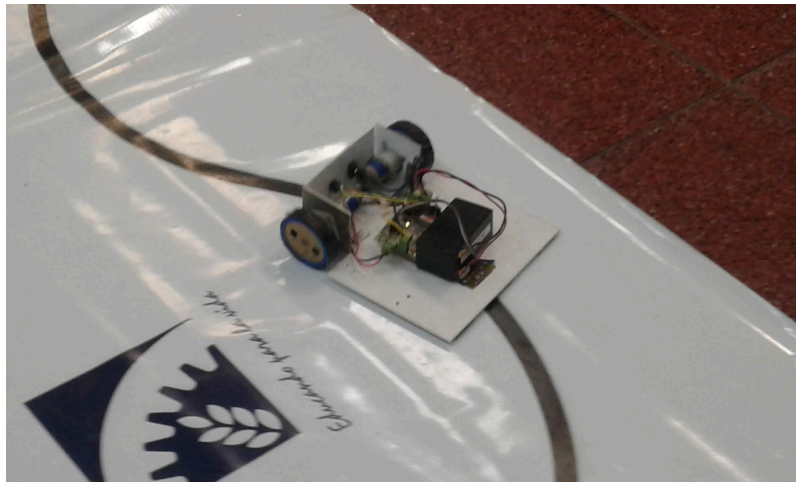


Imagen 1: Robot velocista en una carrera

Objetivo

Diseñar e implementar un robot velocista capaz de seguir una línea predeterminada, manteniendo la máxima velocidad posible y corrigiendo su trayectoria en tiempo real mediante un controlador PID.

- Implementar un sistema de detección de línea utilizando sensores infrarrojos (IR).
- Corregir la trayectoria del robot si la línea se desplaza hacia un lado.
- Mantener la máxima velocidad posible sin perder precisión en el seguimiento de la línea.
- Analizar las perturbaciones presentes y sus efectos sobre el sistema.

Alcance

Contexto

El robot opera sobre una pista blanca con una línea negra como guía. La tarea principal es que el robot mantenga la línea en el centro de su trayectoria utilizando sensores de posición infrarrojos (IR) y un controlador programado en Arduino que ajuste el movimiento de los motores de las ruedas.

Entrada

Es aquello que hace que la respuesta sea constante, en nuestro caso de estudio es el ancho de la línea la cual tiene que seguir el sistema de sensores.

Elemento de Medición

El robot cuenta con una serie de sensores infrarrojos (**QTR-8RC**) colocados en la parte inferior que detectan los colores del piso. Estos sensores proporcionan señales al microcontrolador Arduino, que procesa la información y determina la posición relativa de la línea respecto al centro del robot.

Controlador

Nuestro controlador es un microprocesador Arduino. Este controlador tiene como entrada una serie de colores proveniente de los sensores, con lo que procesa la posición de la línea, y comparándolo con el centro del robot, envía una señal a los motores para que ajusten la posición del robot

Variable Controlada

La variable controlada es la posición del robot con respecto a la línea. Si la línea se encuentra desplazada de su posición central, el robot debe ajustar su dirección para recentrar.

Variable de Referencia o Set Point

La referencia es la posición "0" del sensor, dicho de otra forma, el centro del sensor. Cualquier desviación significativa de esta referencia genera una señal de error que se utilizará para realizar las correcciones necesarias.

Señal de Error

Llamamos a la diferencia entre la posición objetivo (el centro de la línea) y la posición medida del error. (que tan lejos de la línea se encuentra el centro del sensor)

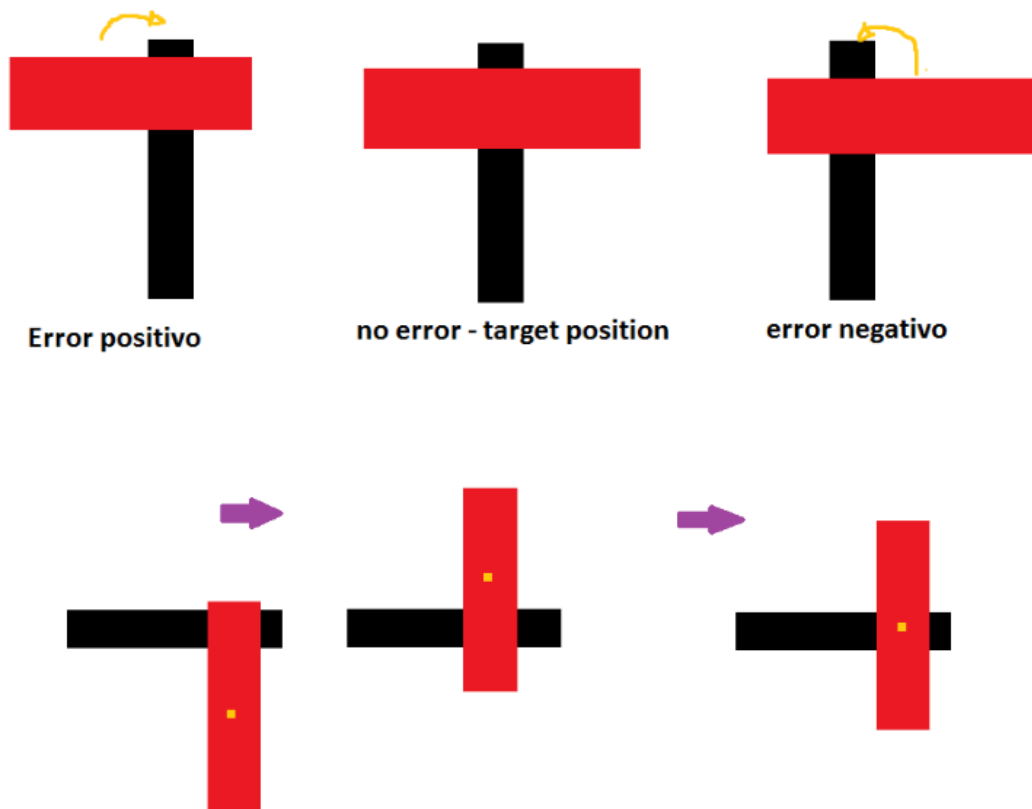


imagen 2: Corrección de error

Set point o Target Position: se refiere al punto o trayectoria deseada que el robot debe seguir durante la simulación. El setpoint sirve como referencia para el sistema de control PID, el cual ajusta dinámicamente la posición del robot para minimizar el error entre su posición actual y la posición objetivo

Banda de error: Define un rango alrededor del setpoint que permite cierta tolerancia en el seguimiento.

Error: Calculado como la diferencia entre la posición actual del robot y la posición objetivo.

Error en Estado Estable (ess)

El error en estado estable es una métrica importante en teoría de control y describe la diferencia persistente entre la posición deseada (setpoint) y la posición real del sistema cuando este alcanza un estado estable (es decir, después de que el sistema se ha estabilizado). Este error ocurre debido a factores como limitaciones en las ganancias del controlador, dinámicas del sistema o perturbaciones no completamente compensadas.

Factores clave que afectan el error en estado estable:

1. Parámetros del controlador PID:

- **Proporcional (Kp):** Incrementar Kp reduce el error en estado estable, ya que aumenta la respuesta del sistema al error. Sin embargo, un valor excesivo puede causar oscilaciones o inestabilidad.
- **Integral (Ki):** Este componente es fundamental para eliminar el error en estado estable, ya que acumula el error a lo largo del tiempo y lo corrige. Un Ki demasiado alto puede inducir oscilaciones indeseadas.

- **Derivativo (Kd):** Aunque Kd no afecta directamente el error en estado estable, mejora la estabilidad al contrarrestar cambios bruscos en el error.

Elemento de Control

El controlador PID (control proporcional, integral y derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener (setpoint/ target position), para aplicar una acción correctora que ajuste el proceso.

En el caso del robot velocista, el controlador PID (que es una rutina basada matemáticamente) procesa los datos del sensor, y lo utiliza para controlar la dirección (velocidad de cada motor), para de esta forma mantenerlo en curso.

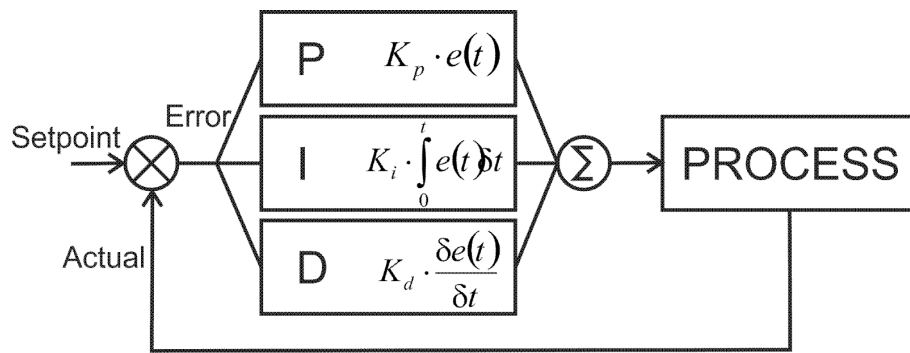


Imagen 3: Circuito del controlador PID

Parámetros del controlador

Proporcional: Es la respuesta al error que se debe entregar a los motores de manera inmediata, es decir, si nos encontramos en el centro de la línea, los motores tendrán en respuesta una velocidad de igual valor, si nos alejamos del centro, uno de los motores reducirá su velocidad y el otro aumentará.

$$\text{Proporcional} = \text{posición} - \text{setpoint}$$

Integral: La respuesta integral es la sumatoria de los errores acumulados, su propósito es disminuir y eliminar el error en estado estacionario, de forma que si el robot velocista se encuentra mucho tiempo alejado del centro (ocurre muchas veces cuando se encuentra en curvas), la acción integral se acumulará y disminuirá el error hasta llegar al valor nominal deseado.

$$\text{Integral} = \text{Integral} + \text{proporcional_pasado}$$

Derivativo: Es la derivada del error, su función es corregir al error de forma instantánea, con una velocidad correlativa al tiempo en el que surgió el error, de forma que el error no aumente rápidamente.

$$\text{Derivativo} = \text{Derivativo-proporcional_pasado}$$

Constantes

Factor (Kp): Es un valor constante utilizado para aumentar o reducir el impacto de proporcional. Si el valor es excesivo, el robot tenderá responder inestablemente, oscilando excesivamente. Si el valor es muy pequeño, el robot respondera muy lentamente, tendiendo a salirse de las curvas

Factor (Ki): Es un valor constante utilizado para aumentar o reducir el impacto de la Integral, El valor excesivo de este provocará oscilaciones excesivas, Un valor demasiado bajo no causará impacto alguno.

Factor (Kd): Es un valor constante utilizado para aumentar o reducir el impacto de la Derivada. Un valor excesivo provocará una sobre amortiguación, provocando inestabilidad.

$$\text{Salida_pwm} = (\text{proporcional} * Kp) + (\text{derivativo} * Kd) + (\text{integral} * Ki)$$

Elemento de Corrección

Los motores de las ruedas reciben las señales de corrección del controlador. Si la línea está a la izquierda, se reduce la velocidad de la rueda izquierda y se aumenta la velocidad de la rueda derecha para corregir la dirección.

Proceso

El proceso es el seguimiento de línea mediante el uso de sensores **QTR-8RC** que detectan la posición de la línea y ajustan la velocidad de las ruedas para mantener el robot en la trayectoria correcta.

Perturbaciones

Las posibles perturbaciones entrantes al sistema incluyen:

- Imperfecciones en la superficie de la pista (desniveles, irregularidades).
- Interferencias eléctricas en los motores.
- Errores en la detección de la línea debido a interferencias externas como variaciones en la iluminación.

Las mismas se modelizan con un impulso, porque son de carácter transitorio que afectan momentáneamente el sistema. Se atenúan ajustando Kp y usando materiales absorbentes de luz en los sensores.

- **Senoidal**
 - Simula una perturbación cíclica, como el efecto de viento que varía periódicamente.

- **Aleatoria**

- Representa perturbaciones impredecibles, como pequeñas variaciones en el terreno.

Impacto de las perturbaciones

Esta modificación hará que el robot tenga que lidiar con desviaciones simuladas, lo que permitirá evaluar cómo el controlador PID ajusta los motores en respuesta a estas perturbaciones. Esto es especialmente útil para optimizar el diseño del controlador frente a condiciones reales

Diagrama de Bloques del Sistema

Posición de referencia respecto al centro de la línea (0o): representa la distancia del robot al centro de la línea.

Sensor infrarrojos QTR-8RC (S): Sensores Infrarrojos que detecta los colores del suelo.

Arduino: Encapsula el bloque suma y el controlador.

Controlador (C): En base a la señal de error del bloque suma, asigna un voltaje a los motores.

Micro motores Pololu (A): Motores que corrigen la trayectoria en función de la señal del arduino.

Posición de la línea (0i): posición relativa del robot respecto a la línea de referencia.

Carga (L): El movimiento físico del robot sobre la pista.

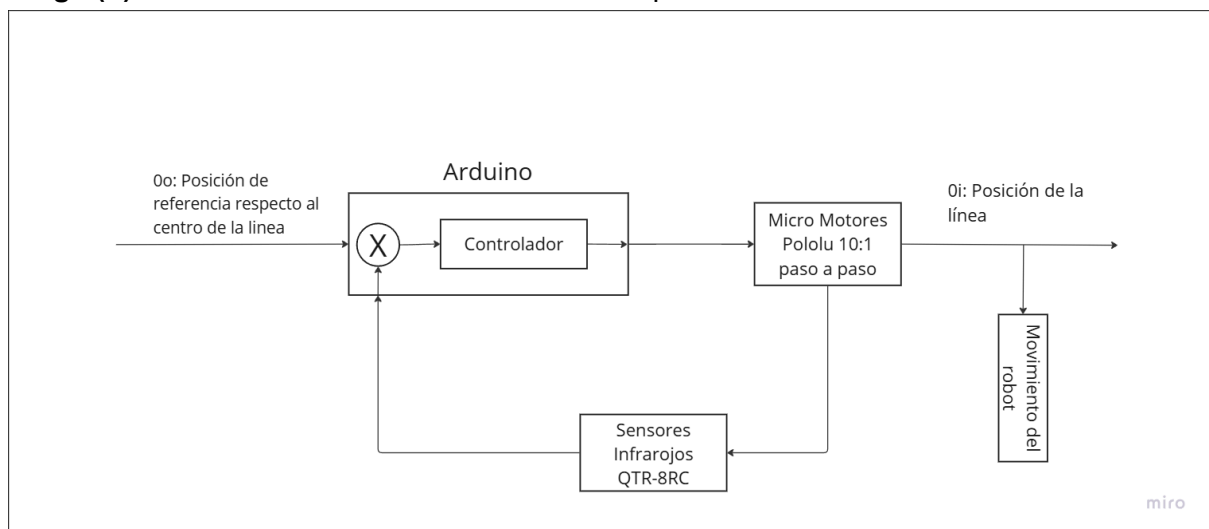

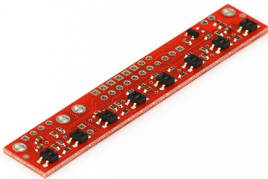
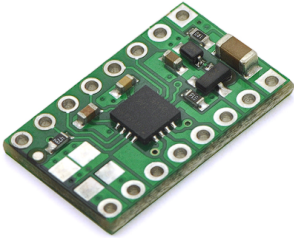
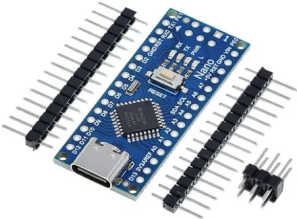


Diagrama 1: Diagrama de bloques del sistema

Componentes

Dispositivo	Especificaciones	Justificación
Micro Motores Pololu MP 10:1 	<ul style="list-style-type: none"> ● Relación de reducción: 10:1 (10 vueltas del motor equivalen a 1 vuelta del eje de salida). ● Voltaje nominal: 6 V (pueden operar en un rango de 3-9 V, pero con variaciones en el rendimiento). ● Velocidad sin carga: <ul style="list-style-type: none"> ○ A 6 V: ~2200 RPM. ● Torque máximo: <ul style="list-style-type: none"> ○ A 6 V: Aproximadamente 0.4 kg·cm (kilogramo-centímetro). ● Corriente: <ul style="list-style-type: none"> ○ Sin carga: ~70 mA. ○ Máxima (bloqueo del motor): ~1.5 A. ● Dimensiones: <ul style="list-style-type: none"> ○ Longitud: ~26 mm. ○ Diámetro: ~10 mm ● Peso: ~9 g. ● Ejes: <ul style="list-style-type: none"> ○ Diámetro: 3 mm. 	<p>Son motores paso a paso de corriente continua (DC) con caja de engranajes, diseñados para aplicaciones de robótica. La relación de reducción 10:1 implica que el motor tiene una salida de alta velocidad y torque moderado. Tamaño compacto, ideal para robots pequeños. Alta eficiencia y baja fricción en el engranaje. Velocidad de salida rápida, adecuada para aplicaciones como robots velocistas.</p>
Sensores QTR-8RC 	<ul style="list-style-type: none"> ● Cantidad de sensores: 8 emisores/receptores reflectivos. ● Voltaje de operación: 3.3 V o 5 V (compatible con Arduino y otros microcontroladores). ● Salida: Señal digital (cada sensor genera un pulso cuya duración depende de la reflectancia detectada). ● Distancia efectiva: 	<p>Es un módulo de sensores reflectivos con 8 detectores (fotodiodos y LEDs infrarrojos). Cada sensor mide la reflectancia del suelo, útil para detectar líneas negras o blancas. Fácil integración con microcontroladores como Arduino. Capacidad de ajuste de sensibilidad</p>

	<ul style="list-style-type: none"> ○ Detecta superficies a distancias de 3 a 9 mm. ○ Sensibilidad óptima: ~6 mm. ● Tiempo de respuesta: <1 ms (ideal para aplicaciones rápidas como robots velocistas). ● Dimensiones: <ul style="list-style-type: none"> ○ Largo: 70 mm. ○ Ancho: 12.7 mm. ○ Altura: ~3.2 mm. ● Consumo de corriente: ~100 mA con todos los LEDs encendidos. ● Configuración: <ul style="list-style-type: none"> ○ LEDs pueden desactivarse para ahorrar energía mediante un pin de control. ○ Cada sensor tiene una resistencia pull-up interna ajustable. ● Uso común: Seguimiento de líneas o detección de bordes en superficies contrastantes. 	<p>según la distancia del sensor al suelo. Salidas digitales (pulsos) que se pueden leer directamente.</p>
<p>Driver para motores DRV8833</p> 	<ul style="list-style-type: none"> ● Voltaje de operación: 2.7 V a 10.8 V (compatible con baterías LiPo de 2S). ● Corriente: <ul style="list-style-type: none"> ○ Continua: 1 A por canal. ○ Pico: 2 A por canal (por breves periodos). ● Puentes H: 2 canales bidireccionales (puede controlar 2 motores DC o un motor paso a paso). ● Entradas: 	<p>Es un controlador de motores de doble canal, diseñado para manejar motores DC pequeños o motores paso a paso. En este caso, controla los micro motores Pololu, asegurando una respuesta precisa y suave en aceleración, frenado y cambio de dirección. Soporta corrientes de hasta 1 A por canal (pico de 2 A). Protección contra</p>

	<ul style="list-style-type: none"> ○ Compatible con señales lógicas de 3.3 V o 5 V. ○ Control PWM para ajustar velocidad y dirección. ● Protecciones integradas: <ul style="list-style-type: none"> ○ Sobrecalentamiento. ○ Sobrecorriente. ○ Subvoltaje. ○ Cortocircuitos. ● Frecuencia PWM máxima: Hasta 250 kHz (más que suficiente para un control suave en motores pequeños). ● Dimensiones: <ul style="list-style-type: none"> ○ ~18 mm x 22 mm. ○ Ideal para robots pequeños donde el espacio es limitado. 	<p>sobrecarga, cortocircuitos y sobrecalentamiento. Control bidireccional de dos motores con señales PWM, permitiendo ajuste fino de velocidad y dirección.</p>
<p>Arduino Nano</p> 	<ul style="list-style-type: none"> ● 14 pines digitales (6 PWM) y 8 pines analógicos. ● Alimentación flexible (USB o entrada externa). ● Programación sencilla en el IDE de Arduino. ● Microcontrolador: ATmega328P. ● Velocidad del reloj: 16 MHz. ● Memoria: <ul style="list-style-type: none"> ○ Flash: 32 KB (2 KB ocupados por el bootloader). ○ SRAM: 2 KB. ○ EEPROM: 1 KB. ● Voltaje de operación: <ul style="list-style-type: none"> ○ Lógica: 5 V. ○ Entrada de alimentación: 7-12 V (regulador interno). ● Entradas/Salidas: 	<p>Microcontrolador compacto basado en el ATmega328P, ideal para proyectos pequeños como robots. Actúa como la "cerebro" del robot, procesando señales de los sensores QTR-8RC, calculando ajustes basados en la teoría de control (PID u otros) y enviando comandos al driver DRV8833 para controlar los motores</p>

	<ul style="list-style-type: none"> ○ Digitales: 14 pines (6 con soporte PWM). ○ Analógicas: 8 entradas. ○ Corriente máxima por pin: 40 mA. ● Conexión: Puerto mini-USB para programación y comunicación. ● Dimensiones: <ul style="list-style-type: none"> ○ Largo: ~45 mm. ○ Ancho: ~18 mm. ● Consumo de corriente: ~19 mA a 5 V (sin periféricos). ● Interfaz de comunicación: <ul style="list-style-type: none"> ○ UART, SPI, I2C (útil para expandir funcionalidades). 	
--	--	--

Modelización y Simulación

Modelo de Simulación

Crearemos un modelo que imite, para cada componente:

- **Sensores QTR-8RC:** Simularemos lecturas reflejando la posición de una línea en la pista.
- **Motores Pololu:** Reproduciremos su comportamiento en términos de velocidad y respuesta.
- **PID:** Calcularemos la salida para ajustar el control de los motores.

Parámetros Variables

Simularemos distintos valores de:

- **Constantes del PID (K_p , K_i , K_d):**
 - **K_p (Proporcional):** controla rapidez de la respuesta. Si es muy alto, el robot podría sobrepasar la línea, si es muy bajo, la corrección podría ser lenta
 - **K_i (Integral):** ayuda a corregir errores acumulados a lo largo del tiempo. Un valor bajo, puede dejar pequeños errores persistentes, pero uno muy alto puede generar oscilaciones
 - **K_d (Derivativo):** ayuda a frenar las oscilaciones. Un valor bajo puede resultar en un control menos estable, y uno alto puede hacer que el sistema reaccione demasiado rápido a los cambios, produciendo oscilaciones.

Resultados a Comparar

- **Precisión:** Desviación respecto al centro de la línea.
- **Estabilidad:** Cómo recuperarse de errores.
- **Error en estado estable:** desviación respecto al objetivo deseado (setpoint).

Implementación

Se utilizó el lenguaje Python junto con la librería de gráficos **Matplotlib** para visualizar las trayectorias. Creamos el modelo de simulación y comparamos los resultados con distintos valores de parámetros.

Escenario 1

Realizamos la simulación sin la presencia de perturbaciones y con los siguientes valores del controlador PID:

Kp	Ki	Kd
3.8	0.9	0.2

Se obtuvieron los siguientes resultados:

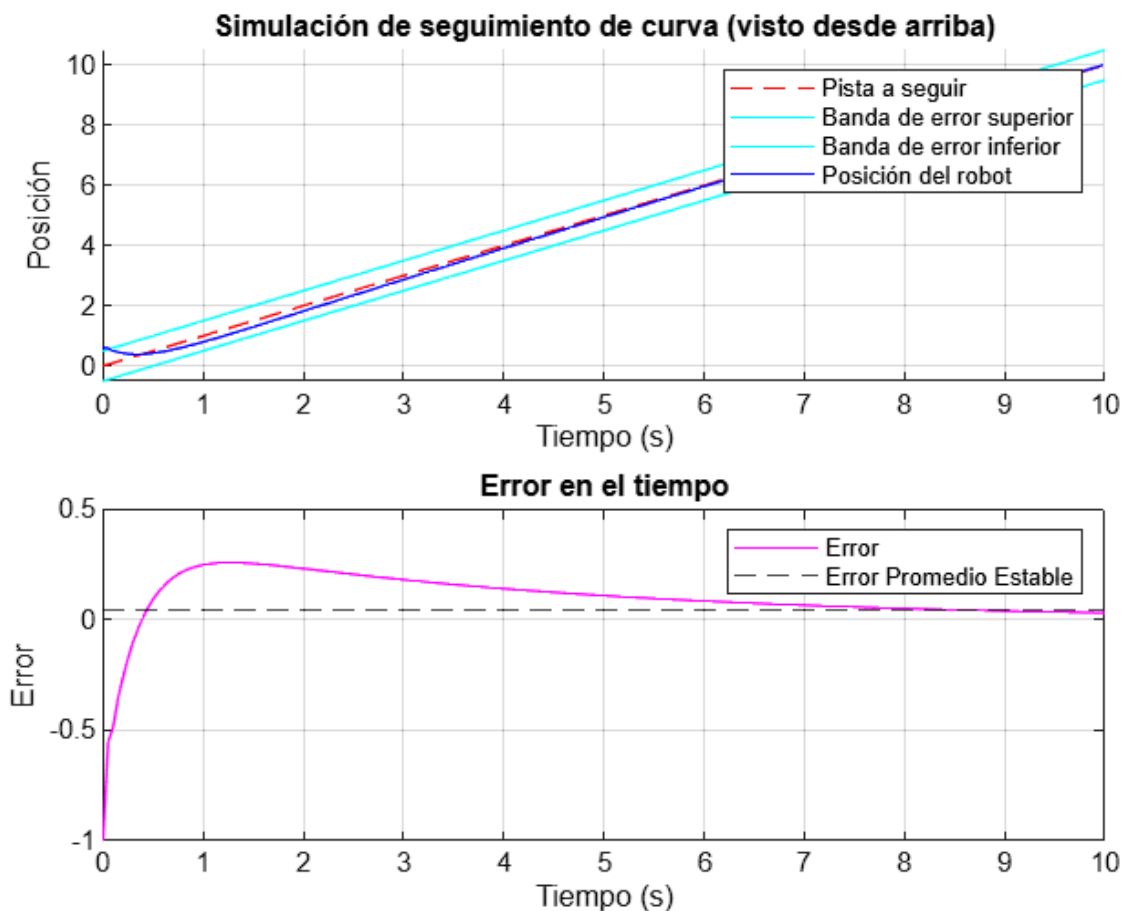


Gráfico 1 y 2 : Simulación inicial sin presencia de perturbaciones

Interpretación

Gráfico 1: Posición del robot y pista

Este gráfico muestra la evolución de la **posición del robot** en comparación con la pista que representa la trayectoria que el robot debe seguir. Para simular el estado transitorio, se puso al robot alejado del centro de la pista, de forma que tenga que alcanzar el centro mediante

múltiples correcciones. Además, se puede observar que el error no excede los valores admisibles de la banda de error, los cuales establecimos entre 1 y -1.

Gráfico 2: Error en cada instante de tiempo

El gráfico del **error en base al tiempo** muestra cómo cambia el error en función del tiempo. Ya que el PID está funcionando correctamente, se observa que el error se reduce gradualmente y se logra estabilizar cerca de cero.

Se puede observar que se alcanza el estado estable del sistema a partir del segundo 8 de forma aproximada, ya que el error del sistema a partir de ese punto es cercano a 0.

Escenario 2

Se agregó la presencia de perturbaciones de forma aleatoria, simulando un ruido que varía de -0.1 a 0.1, usando los mismos valores de K_p , K_i y K_d del escenario anterior

Se obtuvieron los siguientes resultados:

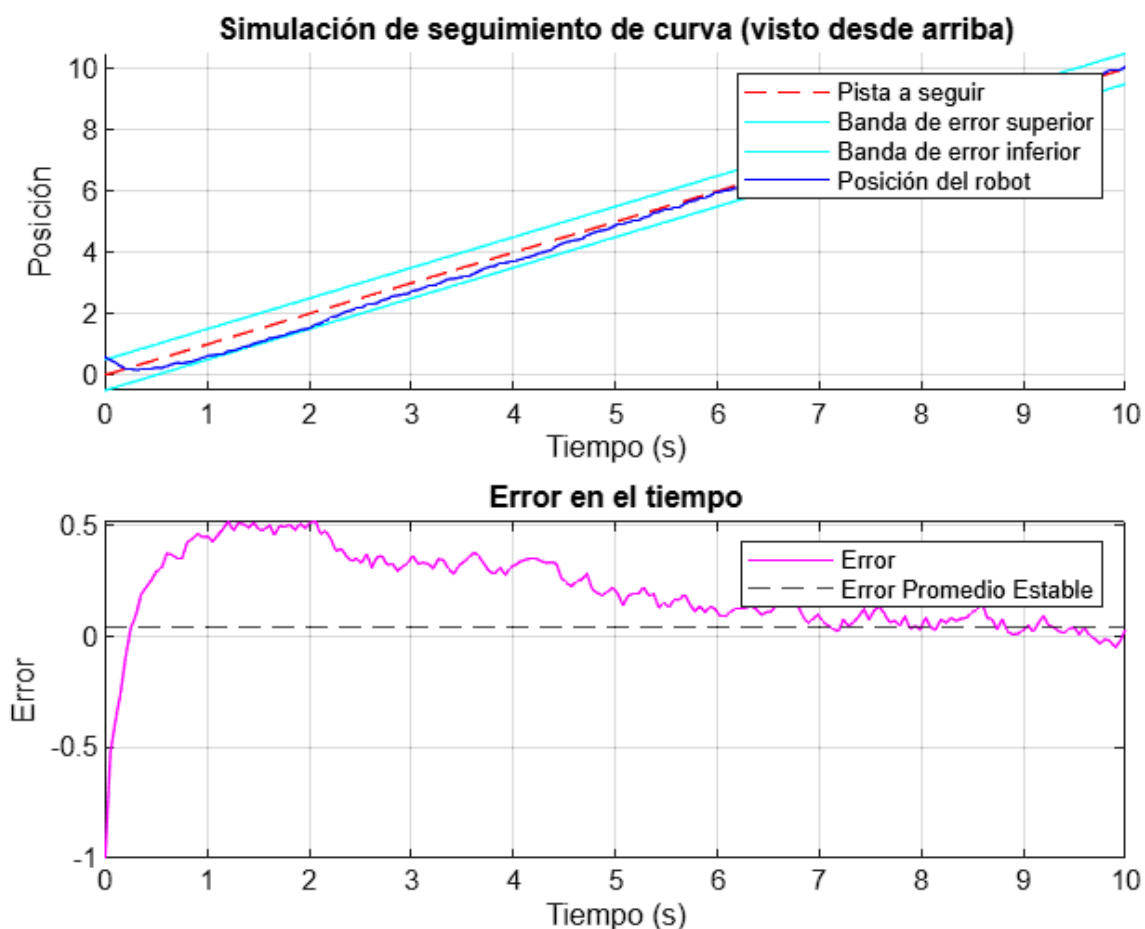


Gráfico 3 y 4: Simulación inicial con presencia de perturbaciones

Interpretación

Debido a la influencia de perturbaciones en el sistema, se observa que el estado estable se alcanza de manera más tardía. Además, el error en estado estable no logra reducirse completamente a cero, ya que las perturbaciones continúan afectando la dinámica del sistema incluso después de alcanzar una aparente estabilidad. Esto evidencia la sensibilidad del sistema ante perturbaciones externas, las cuales generan fluctuaciones residuales en el error.

Escenario 3

Utilizando los siguientes valores

K_p	K_i	K_d
10	0.9	0.2

Se obtuvieron los siguientes resultados:

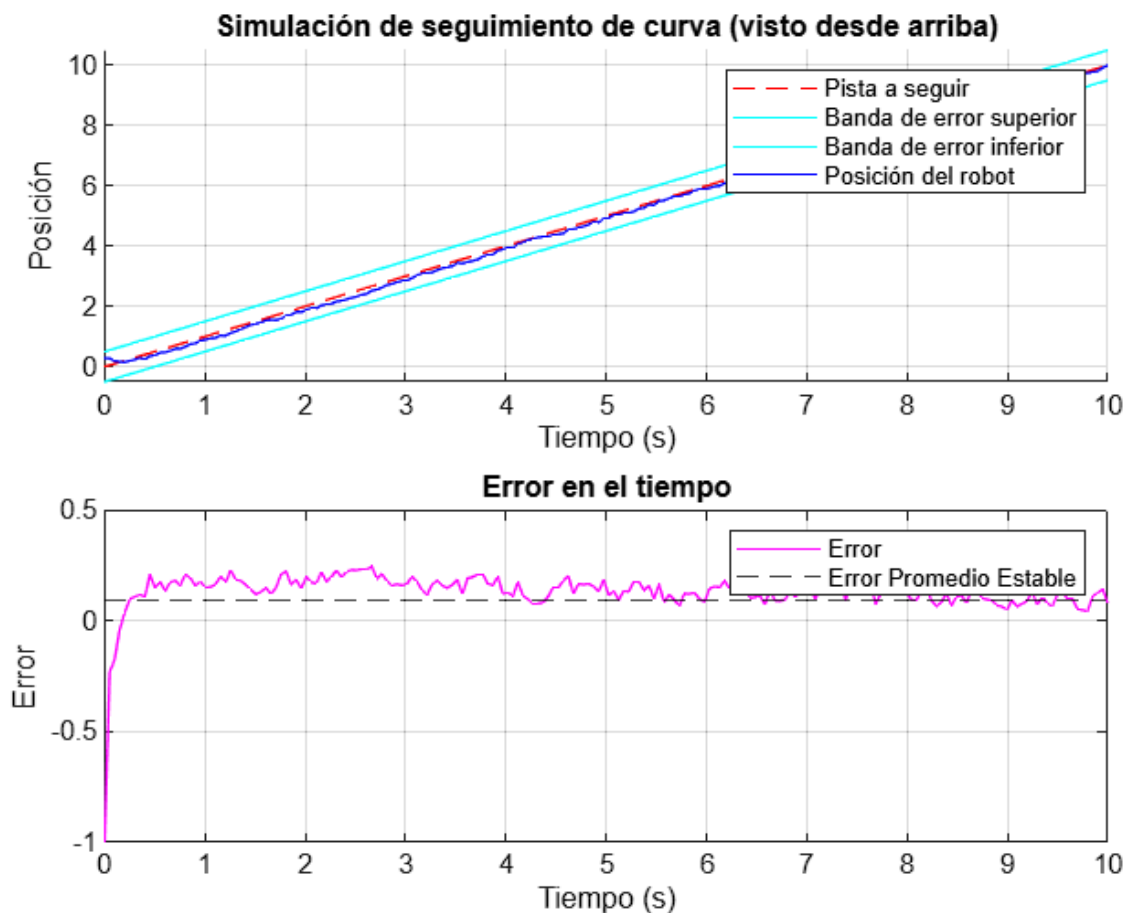


Gráfico 5 y 6: Simulación con mayor valor de K_p

Interpretación

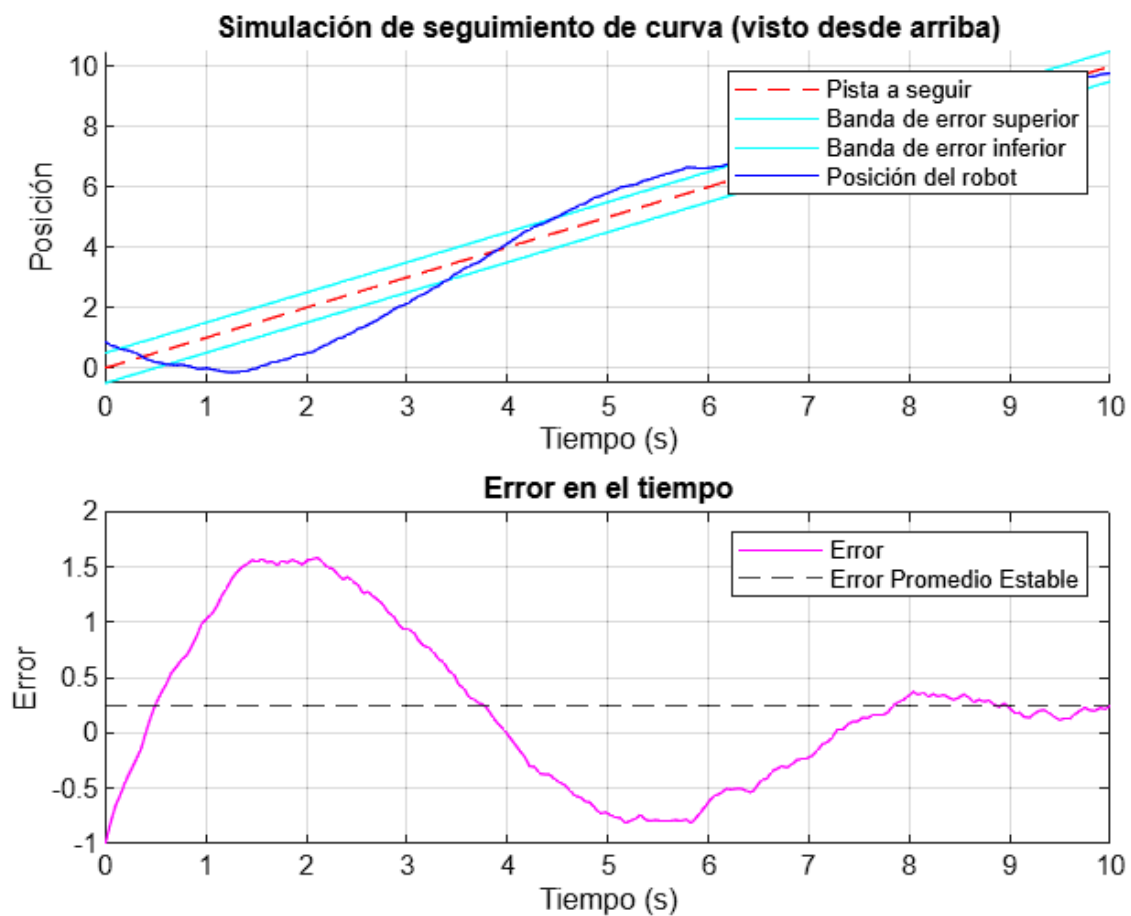
En este escenario, se incrementó el valor de K_p para intensificar la respuesta proporcional del sistema. Esto permitió que las correcciones fueran más rápidas frente a errores iniciales significativos, ayudando al robot a acercarse al centro de la línea con mayor velocidad. Como resultado, el sistema logró centrarse y alcanzar el estado estable en un tiempo considerablemente menor en comparación con otros casos.

Escenario 4

Utilizando los siguientes valores

K_p	K_i	K_d
0.5	0.9	0.05

Se obtuvieron los siguientes resultados:



Interpretación

En este escenario, se redujeron los valores de las ganancias del controlador PID, lo que disminuyó la intensidad de las correcciones aplicadas al error. Como consecuencia, el sistema tardó más en alcanzar el estado estable y mostró un comportamiento menos preciso, con mayores fluctuaciones alrededor de la posición objetivo. Esto evidencia la relación directa entre los parámetros del PID y la rapidez y estabilidad del sistema en su respuesta.

Conclusiones

El desarrollo de este trabajo práctico permitió aplicar conceptos fundamentales de la teoría de control en un sistema real, integrando sensores, actuadores y algoritmos en un entorno dinámico. Desde un enfoque teórico, se destacan los siguientes aspectos:

1. Aplicación del Control en Lazo Cerrado:

- El uso de un sistema en lazo cerrado permite garantizar que el robot se ajuste continuamente a las desviaciones detectadas en la trayectoria. Este principio es esencial para sistemas que operan en ambientes con incertidumbre o perturbaciones externas.

2. Modelización de Perturbaciones:

- Las perturbaciones externas, como cambios en la iluminación o irregularidades en la superficie, fueron modeladas, permitiendo simular de forma más cercana a la realidad. Esto permitió analizar cómo afectan la estabilidad y la respuesta del sistema, evidenciando la importancia de un diseño robusto del controlador derivativo en especial y de los sensores.

3. Importancia de la Estabilidad:

- La estabilidad del sistema fue un punto crítico en el diseño del controlador. Los valores adecuados de los controladores proporcional, integral y derivativo fueron esenciales para que el robot se aproximara y mantuviera en la línea sin oscilaciones excesivas y en un tiempo satisfactorio, lo cual resalta la relación entre estabilidad y desempeño en los sistemas de control.

4. Relación entre la Teoría y la Práctica:

- Este trabajo ilustra cómo conceptos teóricos, como la señal de error, las variables controladas y la realimentación, pueden aplicarse directamente a resolver problemas prácticos en robótica móvil. Además, destaca la relevancia de considerar las limitaciones físicas del hardware en la implementación de controladores.

En conclusión, este proyecto fue una excelente oportunidad para poner en práctica los conceptos de teoría de control en un problema real, destacando tanto su utilidad como sus desafíos en aplicaciones concretas. Encontrar el balance adecuado entre simplicidad, estabilidad y rendimiento fue fundamental para lograr un diseño que funcione de manera eficiente y cumpla con los objetivos planteados.

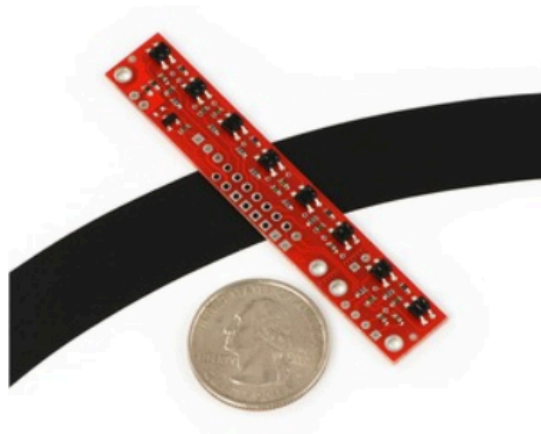
Repositorio:

<https://github.com/GuidoMartinez1/TPFI-Robot-Velocista/tree/main>

Anexos - Hojas de datos

Sensores QTR- 8RC

QTR-8A and QTR-8RC Reflectance Sensor Array User's Guide

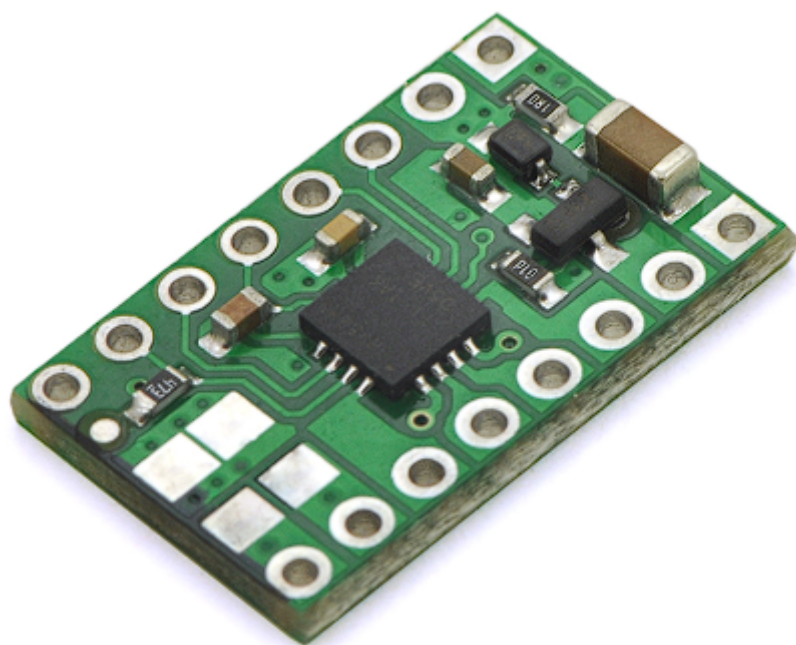


Micromotres Polulu MP 10:1

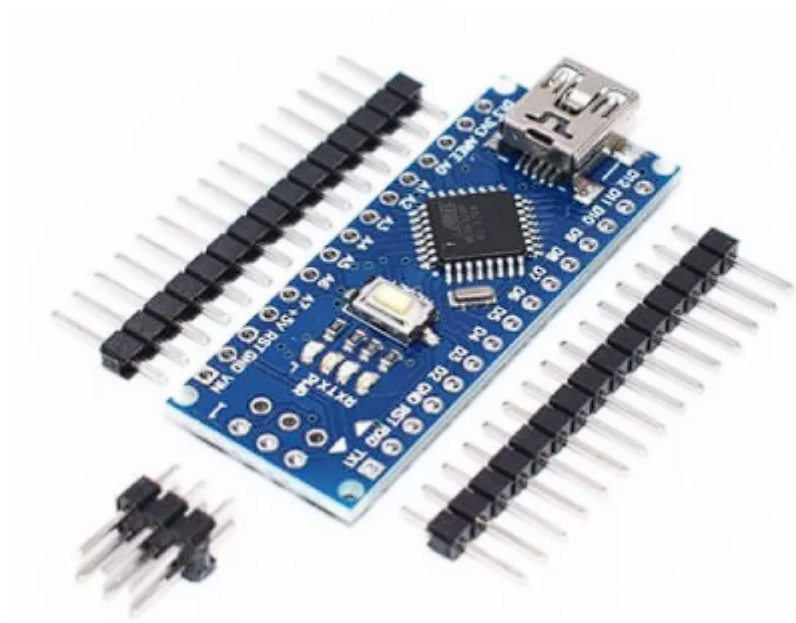
Micro Metal Gearmotors



Driver para motores DRV8833



Arduino Nano



Bibliografía

<https://aprendiendofacilelectronica.blogspot.com/2014/12/robot-velocista-de-competencia-parte.html>

<https://www.pololu.com/docs/0j13/all>

<https://www.alldatasheet.com/datasheet-pdf/pdf/241077/ATMEL/ATMEGA328P.html>

<https://www.alldatasheet.com/datasheet-pdf/pdf/437529/TI1/DRV8833.html>