Universidad de las Américas Puebla

Escuela de ingeniería

Departamento de computación, electrónica y mecatrónica

UDLAP®

Reporte Final

Clave materia sección: LRT3082

Número de equipo: 3

176907 Denisse Alvarado Palacios
178261 José Rodrigo Apodaca Álvarez
159807 Sebastián Trejo Barrera
173856 Emilio Márquez Jiménez

A 15/Noviembre/2024, San Andrés Cholula, Puebla

Índice

1. Resumen

El proyecto "Ball & Beamçonsiste en diseñar un sistema de control que estabilice la posición de una bola sobre un riel utilizando un controlador implementado en un Arduino UNO. El sistema requiere un modelado dinámico del sistema físico, análisis de estabilidad y diseño de controladores basados en técnicas de control clásico como PID. El uso de MATLAB y Simulink para la simulación y análisis permitirá obtener un prototipo controlable en tiempo real. El objetivo es mantener la bola en una posición deseada mediante ajustes del ángulo del riel, abordando tanto el análisis temporal como frecuencial para asegurar un desempeño óptimo.

Keywords: *Ball and Beam System, Control de sistemas inestables, Control PID, Arduino UNO, Modelado matemático*

2. Introducción

El sistema "Ball & Beam"(bola y riel) es un experimento clásico en la teoría de control, ampliamente utilizado para la enseñanza de control automático debido a su naturaleza inherentemente inestable y su similitud con sistemas de control más complejos, como los sistemas de guiado de misiles y estabilización de aviones. El problema consiste en controlar la posición de una bola que se desplaza libremente sobre un riel mediante la inclinación de dicho riel. La bola tiende a rodar hacia los extremos del riel debido a la gravedad, por lo que se necesita un controlador para estabilizar su posición en un punto deseado.

El propósito del presente proyecto es diseñar un sistema de control basado en un microcontrolador Arduino UNO que permita mantener la bola en una posición específica mediante el ajuste dinámico del ángulo del riel. Para ello, se utilizará un enfoque que combine el modelado matemático, simulación en MATLAB/Simulink, y control en tiempo real con un Arduino.

2.1. Descripción del Sistema Ball & Beam

El sistema "Ball & Beam" consta de los siguientes componentes:

- **Una bola** que puede rodar libremente sobre un riel.
- **Un riel horizontal** que puede inclinarse en un ángulo θ controlado por un actuador (como un servo).
- **Sensores** que miden la posición de la bola a lo largo del riel.
- **Un controlador** (implementado en Arduino) que ajusta el ángulo del riel para estabilizar la posición de la bola.

El principio de operación del sistema se basa en la gravedad. Cuando el riel se inclina, una componente de la fuerza gravitacional actúa sobre la bola, haciéndola rodar hacia abajo. El objetivo es ajustar el ángulo del riel de manera que la bola se mantenga en una posición deseada, compensando los efectos de la gravedad.

2.2. Propósito y Aplicaciones

El sistema "Ball & Beam"se utiliza para demostrar principios fundamentales de control como:

- **Estabilidad:** Se supone que el sistema es inherentemente inestable, si no se controla, la bola continuará rodando hasta el borde del riel.
- **Control retroalimentado:** Se requiere un controlador que ajuste continuamente el ángulo del riel basado en la posición actual de la bola.
- **Sistemas de segundo orden:** El sistema puede ser modelado matemáticamente como un sistema de segundo orden, lo que lo convierte en un excelente ejemplo para el análisis de estabilidad, respuesta en el tiempo y respuesta en frecuencia.

Este sistema tiene aplicaciones prácticas en el desarrollo de sistemas de control avanzados, como la estabilización de aeronaves, vehículos autónomos, y sistemas de control de precisión.

2.3. Modelado Matemático del Sistema

Para diseñar un controlador, es necesario modelar la dinámica del sistema "Ball & Beam". A continuación, se presenta el modelado paso a paso.

2.3.1. Ecuaciones de Movimiento

La dinámica del sistema se puede describir mediante las leyes de la mecánica clásica. Considerando una bola de masa m que rueda sin deslizarse sobre un riel de longitud L, la fuerza neta que actúa sobre la bola es la componente de la gravedad que depende del ángulo θ del riel.

donde:

- x es la posición de la bola a lo largo del riel.
- $\ddot{x}$ es la aceleración lineal de la bola.
- g es la aceleración debido a la gravedad.
- θ es el ángulo de inclinación del riel.
- b es el coeficiente de fricción.

2.3.2. Relación entre el Ángulo y la Aceleración de la Bola

La aceleración angular del riel afecta la aceleración lineal de la bola. Si modelamos el sistema como un conjunto de cuerpo rígido con la bola rodando sin deslizarse, obtenemos una relación entre el momento de inercia J, el radio de la bola r, y la masa m

2.4. Diseño del Controlador

Para estabilizar la posición de la bola, se implementará un controlador PID (Proporcional-Integral-Derivativo). El controlador ajustará el ángulo del riel basado en la diferencia entre la posición deseada y la posición actual de la bola. La función de transferencia del controlador PID se expresa como:

$$PID(s) = K_p + \frac{K_i}{s} + K_d s \tag{1}$$

donde:

- K_p es el término proporcional.
- K_i es el término integral.
- K_d es el término derivativo.

El sistema cerrado, que incluye el controlador y la planta, se modelará en Simulink para analizar su respuesta y ajustar los parámetros del controlador para optimizar la estabilidad y el tiempo de respuesta.

2.5. Implementación

3. Planteamiento del Problema

El sistema "Ball & Beam"presenta un desafío significativo en el campo del control automático debido a su naturaleza inherentemente inestable. En este sistema, una bola se desplaza libremente sobre un riel horizontal que puede inclinarse mediante un actuador. Cuando el riel se inclina, la bola rueda hacia un extremo o el otro, impulsada por la gravedad. Sin embargo, a diferencia de otros sistemas, el "Ball & Beam"no tiene un punto de equilibrio natural: la bola continuará moviéndose indefinidamente hacia los extremos si no se controla adecuadamente.

El problema radica en diseñar un sistema de control que permita mantener la bola en una posición deseada sobre el riel mediante la inclinación del mismo. Esto requiere el desarrollo de un controlador que ajuste de manera continua y precisa el ángulo del riel en función de la posición actual de la bola, con el objetivo de estabilizar el sistema en un punto fijo. En ausencia de un controlador, cualquier perturbación, por pequeña que sea, resultará en que la bola ruede hacia el borde del riel y caiga.

3.1. Desafíos Técnicos

El"Ball & Beam" es un ejemplo de un sistema de segundo orden con características no lineales, lo que añade complejidad al diseño del controlador. Algunos de los principales desafíos incluyen:

- **Inestabilidad inherente:** La bola no puede permanecer en una posición fija sin un control activo. Si el sistema no se ajusta continuamente, la bola se desplazará rápidamente hacia los extremos del riel.

- **Tiempo de respuesta:** El sistema debe ser capaz de responder rápidamente a los cambios en la posición de la bola para evitar que se desplace demasiado lejos del punto de equilibrio deseado.
- **Ruido y perturbaciones:** Los sensores que miden la posición de la bola y los actuadores que controlan el ángulo del riel están sujetos a ruido y errores de medición, lo que puede afectar negativamente el rendimiento del sistema de control.
- **Linealización del sistema:** Para simplificar el diseño del controlador, se asume que el ángulo de inclinación es pequeño, lo que permite aproximar $\sin(\theta) \approx \theta$. Sin embargo, esta aproximación es válida solo para ángulos pequeños, lo que limita el rango de operación del sistema.

3.2. Objetivos del Proyecto

El objetivo principal del presente proyecto es diseñar e implementar un sistema de control que mantenga la bola en una posición deseada sobre el riel, utilizando un Arduino UNO para el control en tiempo real. Esto implicará:

- Modelado dinámico del sistema "Ball & Beam"para obtener las ecuaciones que describen su comportamiento.
- Análisis de estabilidad y diseño de un controlador PID que ajuste el ángulo del riel de forma continua.
- Implementación del sistema en un entorno físico utilizando el Arduino UNO junto con simulaciones en MATLAB y Simulink para validar el diseño.
- Evaluación del desempeño del sistema en términos de estabilidad, tiempo de respuesta y robustez frente a perturbaciones.

3.3. Justificación del Estudio

El "Ball & Beam.[es] un sistema que, aunque simple en su estructura, plantea retos complejos en términos de control, lo que lo convierte en un excelente caso de estudio para desarrollar y validar técnicas de control automático. Además, los conceptos aprendidos a partir de este proyecto son aplicables a sistemas de mayor complejidad en la industria, como la estabilización de plataformas, el control de drones y el balanceo de robots bípedos.

El proyecto no solo permitirá profundizar en el diseño de controladores PID, sino también en el uso de herramientas como MATLAB, Simulink y Arduino para la simulación y control en tiempo real, fomentando una comprensión práctica de la teoría de control y su aplicación en sistemas físicos.

4. Marco Teórico

El sistema "Ball & Beam.[es] un sistema mecánico que se utiliza ampliamente en la enseñanza de teoría de control debido a su complejidad intrínseca y su comportamiento inestable. En esta sección se realizará el modelado paso a paso de su dinámica para obtener un modelo matemático que permita diseñar un controlador que estabilice la posición de la bola.

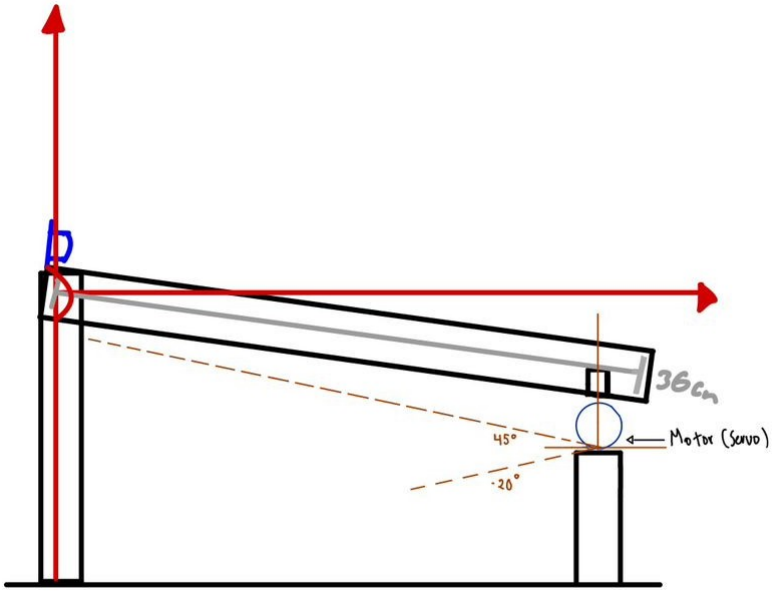

Figura 1: Diagrama de Cuerpo LIbre del sistema.

4.1. Modelado del Sistema Dinámico

El sistema "Ball & Beam.[es] un sistema que consiste en un riel que puede inclinarse mediante un actuador y una bola que se desplaza libremente sobre dicho riel. El objetivo de este sistema es controlar la inclinación del riel para mantener la bola en una posición deseada. A continuación, se detalla el modelado del sistema y la obtención de su función de transferencia.

4.1.1. Ecuaciones de Movimiento

Para simplificar el análisis, consideraremos los siguientes supuestos:

- La bola rueda sin deslizarse sobre el riel.
- El ángulo de inclinación θ es pequeño, por lo que $\sin(\theta) \approx \theta$ y $\cos(\theta) \approx 1$.
- La fricción entre la bola y el riel es despreciable.
- El sistema es simétrico y no hay efectos de fuerzas externas.

La dinámica del sistema se modela utilizando las leyes de la mecánica de Newton. La ecuación diferencial que describe el movimiento de la bola se deriva de las siguientes consideraciones.

4.2. Descripción del Sistema y Variables

El sistema **Ball & Beam** consiste en un riel (haz) inclinado controlado por un motor, con una bola que puede rodar sobre el riel. Al cambiar el ángulo $\theta(t)$ del haz, la bola se mueve debido a la gravedad. A continuación, se definen las variables y parámetros:

- $R(s)$: posición de la bola (en metros).
- $\Theta(s)$: ángulo del haz (en radianes).
- m: masa de la bola (en kg).
- g: aceleración debida a la gravedad (9.81m/s^2).J: $momentodeinerciadelabola$.
- R: radio de la bola.
- L: longitud total del haz (en metros).
- d: distancia desde el centro del haz hasta el punto donde se aplica el torque (en metros).

4.3. Dinámica del Movimiento

4.4. Segunda Ley de Newton para la Bola

La aceleración de la bola sobre el haz está dada por la componente de la gravedad en la dirección del haz. Si el haz se inclina con un ángulo θ, la fuerza neta sobre la bola es:

$$F = mg\sin(\theta) \tag{2}$$

Para ángulos pequeños ($\theta \approx \sin(\theta)$), se puede aproximar como:

$$F \approx mg\theta \tag{3}$$

4.5. Relación entre Aceleración y Fuerza

La aceleración de la bola, a, está relacionada con la fuerza y la masa de la bola:

$$m\ddot{x} = mg\theta \tag{4}$$

$$\ddot{x} = g\theta \tag{5}$$

4.6. Considerando el Momento de Inercia y Rotación

Si la bola rueda sin deslizarse, su aceleración angular α y su aceleración lineal a están relacionadas por:

$$a = R\alpha \tag{6}$$

Usando la ecuación de rotación:

$$J\alpha = RF \tag{7}$$

Sustituyendo $\alpha = \frac{a}{R}$, tenemos:

$$J\frac{a}{R} = Rmg\theta \tag{8}$$

$$a\left(\frac{J}{R^2} + m\right) = mg\theta \tag{9}$$

Despejando la aceleración a:

$$\ddot{x} = \frac{gR^2\theta}{J + mR^2} \tag{10}$$

4.7. Considerando el Efecto del Haz

El efecto del haz en la bola se introduce mediante el factor $\frac{d}{L}$, donde d es la distancia desde el pivote al centro de masa de la bola y L es la longitud del haz:

$$\ddot{x} = \frac{gd}{L} \cdot \frac{R^2\theta}{J + mR^2} \tag{11}$$

4.8. Función de Transferencia en Lazo Abierto

Tomando la transformada de Laplace de la ecuación de movimiento para obtener la función de transferencia, tenemos:

$$s^2R(s) = \frac{gd}{L} \cdot \frac{R^2}{J + mR^2}\Theta(s) \tag{12}$$

Despejamos para obtener la función de transferencia en lazo abierto:

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2} + m\right)} \cdot \frac{1}{s^2} \tag{13}$$

4.9. Función de Transferencia Final

La función de transferencia detallada del sistema **Ball & Beam** en lazo abierto es:

$$P(s) = -\frac{mgd}{L\left(\frac{J}{R^2} + m\right)} \cdot \frac{1}{s^2} \tag{14}$$

5. Interpretación de la Función

Esta función indica que la salida $R(s)$ (posición de la bola) tiene una relación inversamente proporcional al cuadrado de la variable s con respecto a la entrada $\Theta(s)$ (ángulo del haz). El factor de ganancia está influenciado por:

- La masa de la bola (m).
- La gravedad (g).
- La distancia desde el pivote (d).
- La longitud del haz (L).

- El momento de inercia (J) y el radio (R) de la bola.

Esto muestra que el sistema tiene un comportamiento *integrador de segundo orden*, lo que significa que necesita un controlador adecuado (por ejemplo, un **PID**) para estabilizar la posición de la bola en un punto deseado en el haz.

6. Análisis Temporal y Frecuencial del Sistema

A continuación, se presenta el análisis temporal y frecuencial del sistema *Ball & Beam* utilizando un modelo en *Simulink*. El sistema se ha modelado utilizando un controlador PID para estabilizar la posición de una bola sobre un haz basculante.

6.1. Función de Transferencia del Sistema

La función de transferencia en lazo abierto para el sistema *Ball & Beam* es:

$$P(s) = \frac{R(s)}{\Theta(s)} = -\frac{mgd}{L\left(\frac{J}{R^2} + m\right)} \cdot \frac{1}{s^2} \tag{15}$$

La constante del sistema K está dada por:

$$K = \frac{mgd}{L\left(\frac{J}{R^2} + m\right)} \tag{16}$$

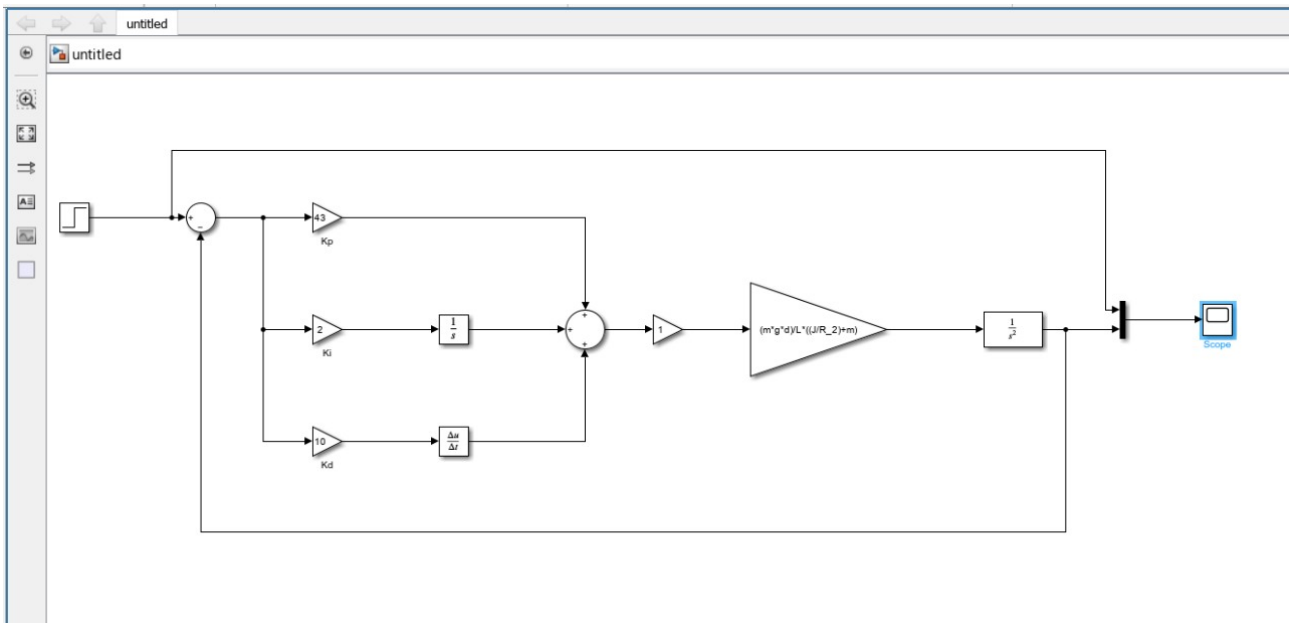

Figura 2: Implementación del sistema en Simulink.

Figura 3: Respuesta del sistema.

6.2. Análisis Temporal del Sistema

El análisis temporal se realiza para observar la respuesta del sistema en el dominio del tiempo. En *Simulink*, se ha utilizado un controlador PID para ajustar la respuesta del sistema ante una entrada en escalón.

6.2.1. Configuración del Modelo en Simulink

El modelo de *Simulink* incluye los siguientes elementos:

- Un bloque de entrada tipo `Step` que genera una entrada en escalón.
- Un controlador PID con valores ajustables de K_p, K_i, y K_d.
- Una función de transferencia que representa la dinámica del sistema $Ball\&Beam$.
- Un bloque `Scope` para visualizar la salida.

Para determinar los valores de K_p, K_i, y K_d utilizamos la tabla de prueba y error proporcionada en clase. Este método consistió en probar diferentes valores para cada constante y observar el comportamiento del sistema en cada caso, ajustándolos gradualmente para optimizar la respuesta.

Después del análisis realizado y basándonos en los resultados observados en el comportamiento del sistema, decidimos adoptar los siguientes valores para las constantes del controlador:

- K_p=43
- K_i=2
- K_d=10

Respuesta	Tiempo de levantamiento	Sobrepaso	Tiempo de estabilización	Error estado est
Kp	Disminuye	Aumenta	Poco Influencia	Disminuye
Ki	Disminuye	Aumenta	Aumenta	Elimina
Kd	Poco Influencia	Disminuye	Disminuye	Poco Influencia

Figura 4: Tabla Prueba y Error

6.2.2. Parámetros Analizados

En base al valor de la constante de tiempo obtenido en la simulación, donde $\tau = 1{,}23$ segundos, se procede a realizar el análisis temporal del sistema Ball & Beam. Utilizando las fórmulas correspondientes para sistemas de primer orden, se obtienen los siguientes parámetros:

Al observar la respuesta del sistema en el `Scope`, se pueden analizar los siguientes parámetros:

- **Tiempo de establecimiento**: El tiempo de establecimiento es el tiempo que tarda la salida del sistema en mantenerse dentro de un margen del 2 % del valor final deseado. Se define como:

$$t_s = \text{Tiempo en el que } |y(t) - y_{\text{final}}| \leq 0{,}02 \times y_{\text{final}} \tag{17}$$

$$t_s \approx 4\tau \tag{18}$$

  Sustituyendo el valor de $\tau = 1{,}23$ segundos:

$$t_s \approx 4 \times 1{,}23 = 4{,}92 \text{ segundos} \tag{19}$$

- **Tiempo de subida**: El tiempo de subida es el tiempo que tarda la respuesta en pasar del 10 % al 90 % del valor final. Matemáticamente se expresa como:

$$t_r = t_{90\,\%} - t_{10\,\%} \tag{20}$$

  donde $t_{10\,\%}$ y $t_{90\,\%}$ son los tiempos en los que la respuesta alcanza el 10 % y el 90 % del valor final, respectivamente.

$$t_r \approx 2{,}2\tau \tag{21}$$

  Sustituyendo $\tau = 1{,}23$ segundos:

$$t_r \approx 2{,}2 \times 1{,}23 = 2{,}71 \text{ segundos} \tag{22}$$

- **Sobreimpulso**: El sobreimpulso indica cuánto se excede la salida por encima del valor de referencia y se expresa como un porcentaje:

$$\text{Overshoot} = \left(\frac{y_{\max} - y_{\text{final}}}{y_{\text{final}}} \right) \times 100\,\% \tag{23}$$

donde $y_{\max}$ es el valor máximo

alcanzado por la respuesta.

- **Error en estado estacionario**: El error en estado estacionario se calcula como la diferencia entre el valor final alcanzado por la salida y el valor deseado:

$$e_{ss} = y_{\text{final}} - y_{\text{ref}} \tag{24}$$

$$e_{ss} = 1{,}0072 - 0{,}9998 \tag{25}$$

Si el sistema se estabiliza en el valor de referencia, $e_{ss} = 0{,}009$.

6.3. Análisis Frecuencial del Sistema

El análisis frecuencial se utiliza para evaluar la estabilidad y el comportamiento del sistema en función de la frecuencia.

6.3.1. Diagrama de Bode

El diagrama de Bode permite observar la respuesta en frecuencia del sistema, en términos de ganancia y fase. Para un sistema dado, la función de transferencia $G(s)$ se representa en un diagrama logarítmico:

$$G(s) = \frac{Y(s)}{U(s)} \tag{26}$$

A partir del diagrama de Bode, se puede calcular:

- Margen de ganancia: Frecuencia en la que la fase es $-180°$.
- Margen de fase: Frecuencia en la que la ganancia es 0 dB.

6.3.2. Diagrama de Nyquist

El diagrama de Nyquist se utiliza para evaluar la estabilidad del sistema en lazo cerrado. El criterio de estabilidad de Nyquist indica que el número de encierros alrededor del punto $(-1, 0)$ determina la estabilidad del sistema.

6.4. Simulación en Simulink

A continuación, se presentan los resultados obtenidos en Simulink para el sistema *Ball & Beam*. La respuesta ante una entrada escalón se muestra en la Figura 3.

6.5. Cálculo de Parámetros a partir de la Simulación

Basado en la simulación realizada:

- **Tiempo de establecimiento**: Aproximadamente $t_s = 3$ segundos.
- **Tiempo de subida**: Aproximadamente $t_r = 0{,}7$ segundos.
- **Sobreimpulso**: No se observó sobreimpulso significativo.
- **Error en estado estacionario**: $e_{ss} = 0$.

7. Análisis y explicación del código

En esta sección se desglosa el funcionamiento del código implementado para el control del sistema *Ball & Beam*, detallando las funciones principales, las configuraciones iniciales, y el propósito de cada bloque de código.

7.1. Configuración inicial

El código comienza con la importación de la biblioteca `Servo.h`, necesaria para controlar el servo. Además, se definen constantes importantes:

- `Umax` y `Umin`: Establecen los límites del ángulo de inclinación del beam en grados (66 y -66, respectivamente).
- `Umax_rad` y `Umin_rad`: Límite equivalente en radianes.
- `T`: Periodo de muestreo del sistema, en segundos.

Estas constantes fueron determinadas mediante un proceso de prueba y error para garantizar que los límites del movimiento del servo fueran seguros y adecuados para mantener la estabilidad de la pelota dentro del riel.

7.2. Variables del controlador

Se declaran las variables necesarias para implementar el control PID:

- `setpoint` y `setpoint_prec`: Representan la posición deseada de la pelota en el riel.
- `y` y `y_prec`: Representan la posición actual de la pelota.
- `error`: Diferencia entre la posición deseada (`setpoint`) y la posición actual (`y`).
- `P, I, D, U`: Componentes proporcional, integral y derivativa del controlador PID, junto con la acción de control (`U`).

Las ganancias `Kp`, `Ki` y `Kd` del controlador fueron ajustadas empíricamente mediante prueba y error para obtener un comportamiento estable del sistema.

7.3. Inicialización del sistema

En la función `setup()`, se configura el sistema:

- Se inicializa la comunicación serial para depuración.
- Se configuran los pines de entrada y salida para los sensores ultrasónicos (`trigPin1`, `echoPin1`, `trigPin2`, `echoPin2`) y se adjunta el servo al pin 9.
- Se calibra la posición inicial del beam (90 grados) y se toman mediciones iniciales de la posición de la pelota (`y_prec`) y la posición deseada (`setpoint_prec`).

7.4. Ciclo principal (`loop()`)

En el ciclo principal, se ejecutan las siguientes tareas:

Medición de posiciones Las funciones `measure_1()` y `measure_2()` se utilizan para medir la posición actual de la pelota y la posición deseada respectivamente, usando sensores ultrasónicos. Estas funciones aplican filtros digitales para suavizar las señales y reducir el ruido.

Cálculo del error El error se calcula como la diferencia entre la posición deseada (`setpoint`) y la posición actual (`y`). Este valor es la entrada al controlador PID.

Implementación del controlador PID El controlador PID se implementa según la ecuación:

$$U = P + I + D$$

donde:

- `P` es proporcional al error.
- `I` acumula el error en el tiempo, a menos que el sistema esté saturado.
- `D` depende del cambio en la posición actual (`y`) respecto al tiempo.

La acción de control (`U`) se satura dentro de los límites establecidos para evitar que el servo exceda su rango seguro.

Control del servo La función `move_servo()` convierte la acción de control (`U`) a un ángulo que puede interpretar el servo, ajustando la inclinación del beam para estabilizar la pelota.

7.5. Comentarios sobre el armado físico

El sistema físico incluye:

- Un riel de 40 cm para la pelota.
- Un sensor ultrasónico para medir la posición de la pelota.

- Un servo MG996R para inclinar el riel.

Los límites de los ángulos del servo y las ganancias PID fueron determinados experimentalmente mediante prueba y error, ajustando los valores para obtener una respuesta estable y minimizar oscilaciones.

7.6. Análisis del funcionamiento esperado

El código espera que el sistema pueda estabilizar la pelota en una posición deseada dentro del riel, compensando perturbaciones mediante ajustes en la inclinación del beam. La acción del controlador PID debe asegurar que el error tiende a cero, y que las oscilaciones sean mínimas.

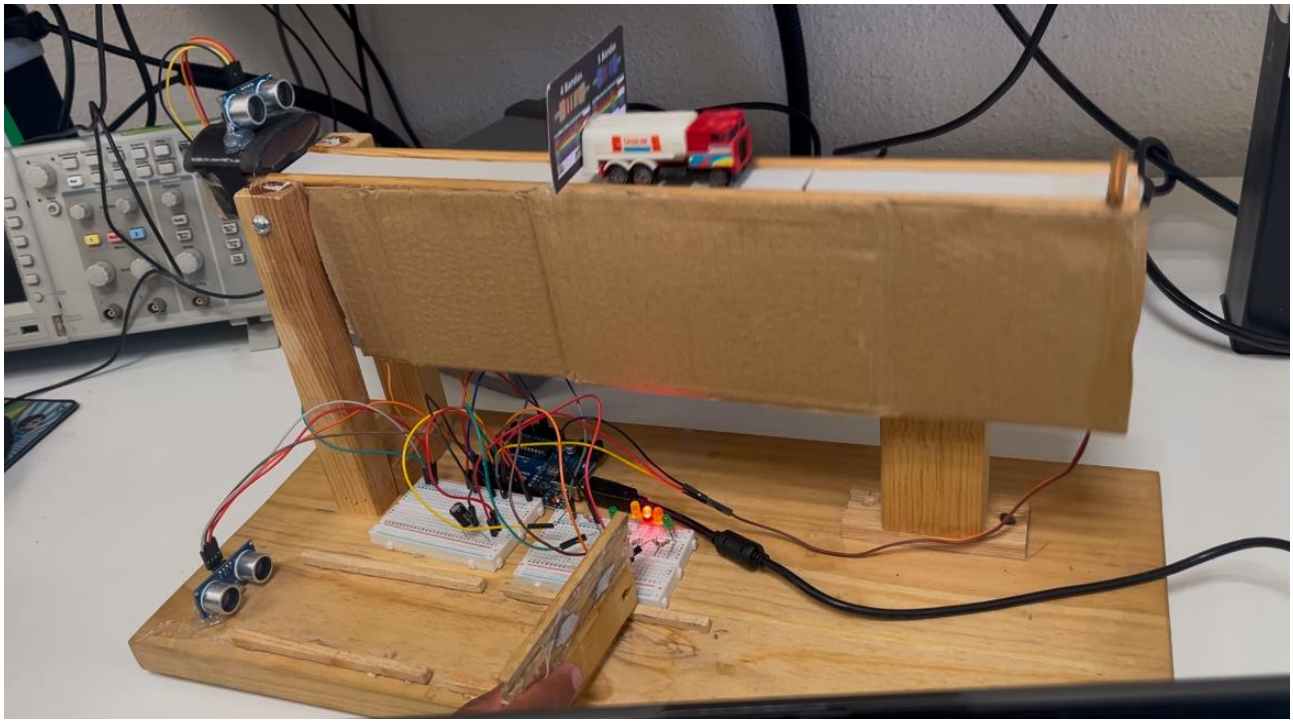

Figura 5: Esta imagen muestra el armado ideal para la maqueta y que funcione con el código previamente explicado

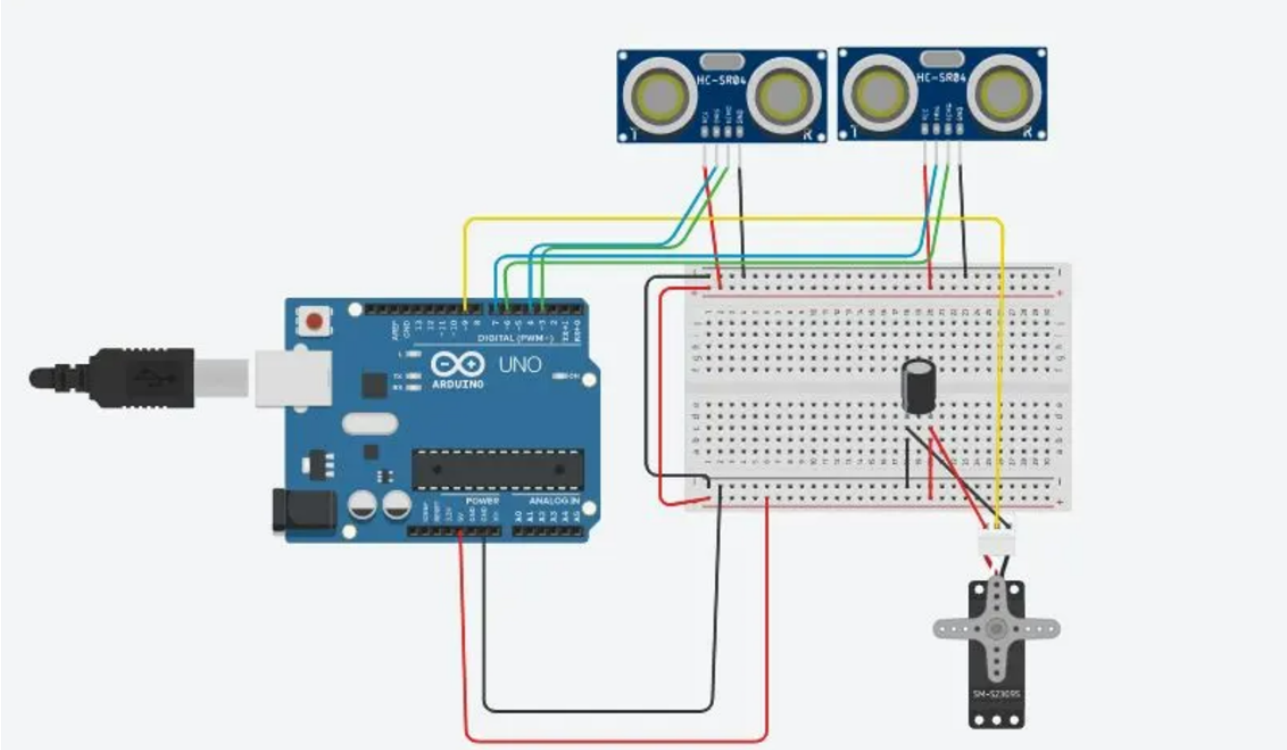

Figura 6: Esta imagen muestra las conexiones realizadas en arduino, en este caso se conecta un condensador al servo para que tenga más corriente
[1]

8. Propuesta de la Estructura del Controlador (Denisse)

Se propone rediseñar el controlador actual utilizando una **estructura PD con compensación integral por calibración externa**. Este enfoque simplifica el diseño al reducir la acumulación de errores históricos y permite ajustes más específicos en situaciones de comportamiento lento o inestabilidad sin depender completamente de la acción integral. La propuesta incorpora elementos novedosos y alternativas para evaluar su desempeño.

8.1. Estructura General del Controlador

El controlador se formula como:

$$U(t) = K_p e(t) + K_d \frac{de(t)}{dt} + U_{\text{comp}}$$

Donde:

- $K_p e(t)$: Acción proporcional para corregir el error actual.
- $K_d \frac{de(t)}{dt}$: Acción derivativa para anticipar el error futuro y estabilizar el sistema.
- U_{comp}: Compensación externa predeterminada para corregir desviaciones acumuladas o ajustes en tiempo real.

8.2. Componentes del Controlador

- **Acción Proporcional (P):** Responsable de reducir el error en proporción directa a su magnitud. Ajusta la respuesta rápida del sistema sin generar oscilaciones excesivas.

$$P = K_p e_k$$

- **Acción Derivativa (D):** Introduce un término estabilizador basado en el cambio del error en el tiempo. Esto amortigua oscilaciones generadas por la acción proporcional:

$$D = K_d \frac{e_k - e_{k-1}}{T}$$

- **Compensación Integral Externa (U_{comp}):** En lugar de integrar el error dentro del controlador, U_{comp} es calculado manualmente durante la calibración o con un pequeño algoritmo de ajuste. Esto reduce los efectos de *windup* y permite ajustar manualmente la posición inicial del sistema.

8.3. Saturación de la Acción de Control

Para garantizar la seguridad del hardware y evitar movimientos extremos del servo, la salida U estará limitada. Rango propuesto:

$$U_{\text{servo}} = \text{map}(U, -1{,}2, 1{,}2, 20, 160)$$

8.4. Periodo de Muestreo y Filtros

El periodo de muestreo $T = 0{,}07\,\text{s}$ fue elegido para aumentar la frecuencia de actualización sin saturar el microcontrolador. Se utilizan filtros digitales para suavizar las señales de los sensores:

$$y_k = \beta y_k + (1 - \beta) y_{k-1}, \quad \text{con } \beta = 0{,}65$$

8.5. Análisis del Diseño

8.6. Ventajas

- **Reducción de Complejidad:** Al eliminar la integración directa, el controlador se vuelve más simple y menos propenso a saturaciones.
- **Estabilidad Mejorada:** La acción derivativa y el ajuste manual de U_{comp} permiten una respuesta más precisa frente a cambios rápidos en el sistema.
- **Ajustes Externos:** U_{comp} puede recalibrarse para diferentes condiciones del sistema, como variaciones en la masa de la bola.

8.6.1. Desventajas

- **Dependencia del Ajuste Manual:** Requiere supervisión para mantener U_{comp} óptimo.
- **Menor Corrección Automática:** En sistemas con errores acumulados grandes, podría ser necesario incluir K_i nuevamente.

9. Análisis de la Simulación

9.1. Sistema en Lazo Cerrado

Con esta estructura, se espera:

- Una respuesta más rápida debido al enfoque en K_p y K_d.
- Menor overshoot en comparación con un PID convencional.
- Mayor estabilidad frente a ruidos de los sensores.

9.2. Funcionamiento Esperado del Prototipo

- La pelota debería estabilizarse en el punto de referencia en menos de 5 segundos.
- Los ajustes manuales de U_{comp} permiten corregir errores sistemáticos.

Si se valida el desempeño del diseño, este enfoque podría implementarse como una alternativa más robusta para aplicaciones futuras.

10. Análisis de Lazo Cerrado

10.1. Introducción

El análisis de un sistema en lazo cerrado es fundamental para entender cómo se comporta un sistema de control con retroalimentación. En este caso, el sistema **Ball & Beam** está compuesto por un riel que se mueve para mantener la pelota en una posición deseada. Este sistema utiliza un controlador PID para ajustar el ángulo del riel en función de la retroalimentación de la posición de la pelota.

10.2. Modelo del Sistema

El sistema Ball & Beam puede ser modelado como un sistema de control en lazo cerrado. A continuación, se describen los componentes del sistema.

10.2.1. Planta

La planta está compuesta por el riel (beam) que puede ser inclinado en un ángulo variable y sobre el cual se desplaza la pelota. El comportamiento de la planta depende de las leyes físicas de la dinámica del riel y la pelota. Este sistema puede ser modelado como un sistema de segundo orden, con una entrada (el ángulo del riel) y una salida (la posición de la pelota).

10.2.2. Controlador PID

El controlador PID genera una señal de control que ajusta el ángulo del riel para minimizar el error entre la posición deseada de la pelota y su posición real medida por un sensor. El controlador PID se calcula como sigue:

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

Donde:

- K_p es la ganancia proporcional.
- K_i es la ganancia integral.
- K_d es la ganancia derivativa.
- $e(t)$ es el error entre la posición deseada y la medida.

10.2.3. Diagrama de Bloques del Sistema en Lazo Cerrado

El diagrama de bloques básico del sistema en lazo cerrado se puede describir de la siguiente manera:

$$\text{Entrada deseada} \rightarrow \text{Controlador PID} \rightarrow \text{Planta} \rightarrow \text{Salida medida}$$

La salida medida es la posición de la pelota y el error es la diferencia entre la entrada deseada y la salida medida.

10.2.4. Ecuación del Error

El error $e(t)$ es la diferencia entre la posición deseada $y_d(t)$ y la posición medida $y(t)$:

$$e(t) = y_d(t) - y(t)$$

10.2.5. Ecuación de la Acción de Control

La acción de control $u(t)$ es generada por el controlador PID, como se describió anteriormente:

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

10.2.6. Dinámica de la Planta

La dinámica de la planta describe cómo la posición de la pelota cambia con el tiempo en respuesta a la señal de control. Esta dinámica puede ser modelada como un sistema de segundo orden debido a la bola y el servomecanismo, con una entrada (el ángulo del riel) y una salida (la posición de la pelota).

10.2.7. Estabilidad del Sistema en Lazo Cerrado

La estabilidad del sistema en lazo cerrado depende de las ganancias del controlador PID. Se pueden realizar análisis de estabilidad utilizando el criterio de Nyquist o el criterio de Bode. Estos métodos permiten determinar si el sistema es estable y tiene un margen de estabilidad adecuado.

10.2.8. Respuesta Transitoria y Estabilidad

Los aspectos clave de la respuesta transitoria son:

- **Tiempo de subida**: El tiempo que tarda la pelota en alcanzar la posición deseada.
- **Sobreshoot**: La cantidad de oscilación antes de que el sistema se estabilice.
- **Estabilidad**: El sistema debe ajustarse a los cambios sin volverse inestable.

El sistema debe ser ajustado para minimizar el sobreshoot y obtener una respuesta rápida con un tiempo de subida corto.

10.3. Ajuste de las Ganancias del Controlador PID

El ajuste de las ganancias PID es crucial para que el controlador funcione correctamente. Las ganancias se ajustaron empíricamente para obtener un rendimiento adecuado. A continuación, se describen brevemente las ganancias utilizadas:

- $K_p = 8{,}6$: Establece la respuesta proporcional.
- $K_i = 1{,}1$: Ayuda a eliminar el error acumulado.
- $K_d = 6{,}3$: Mejora la estabilidad y reduce la sobreoscilación.

Estas ganancias fueron ajustadas mediante un proceso de prueba y error, basándose en la observación de la respuesta del sistema físico y simulado.

10.4. Simulación y Prototipo

El comportamiento del sistema fue verificado tanto mediante simulación como en un prototipo físico.

10.4.1. Simulación en Simulink

En Simulink, el modelo matemático del sistema Ball & Beam fue simulado utilizando los parámetros PID. Se observaron las respuestas del sistema ante diferentes entradas (cambios en la posición deseada de la pelota).

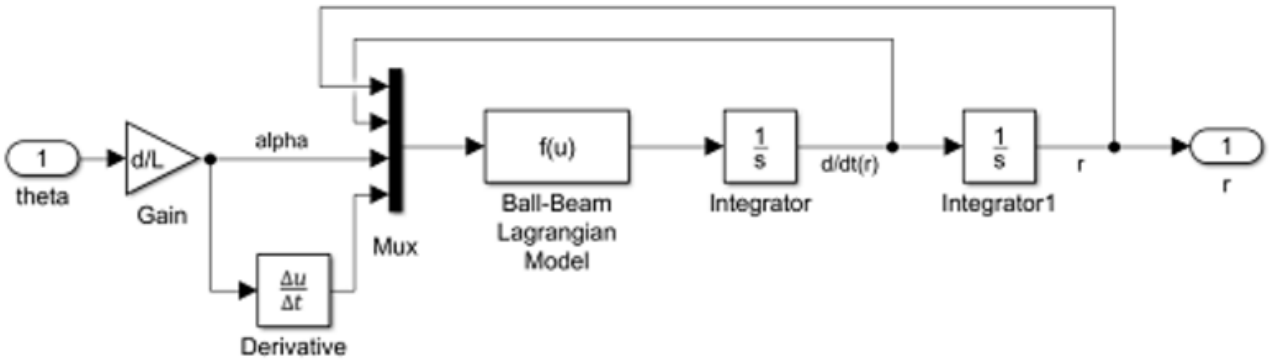

Figura 7: Esta imagen muestra lo que usamos en Lazo abierto

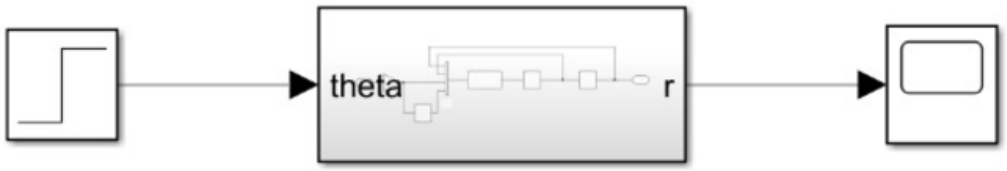

Figura 8: Reducción básica de lazo abierto, para después poder meter eso en lazo cerrado

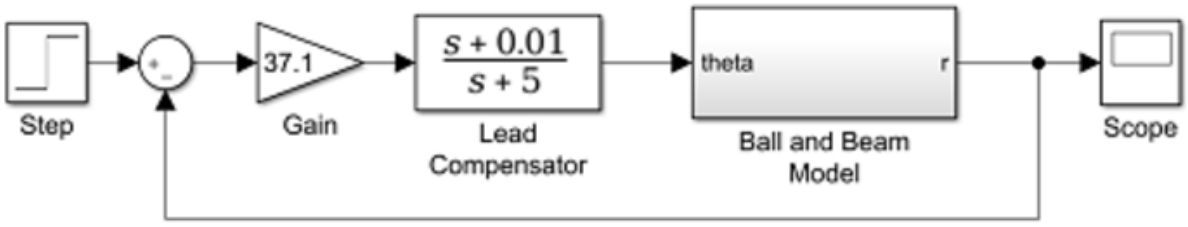

Figura 9: En esta imagen se muestra el lazo cerrado, para después poder hacer el análisis.

10.4.2. Prototipo Físico

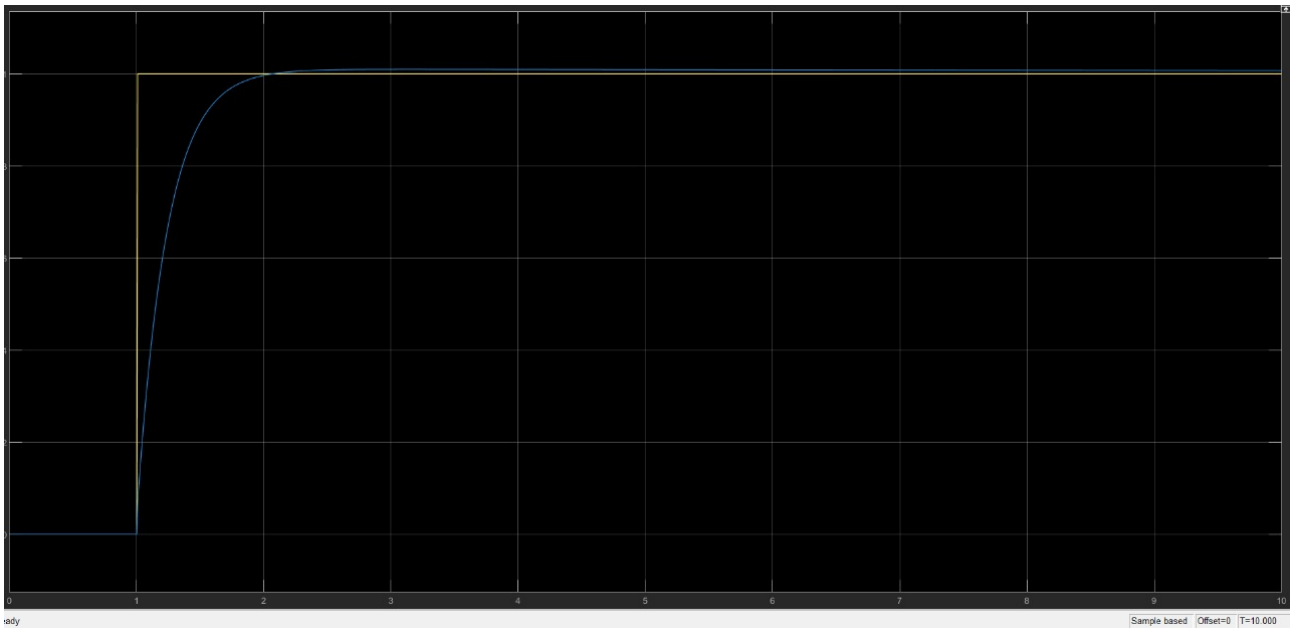

Figura 10: Scope en la simulación

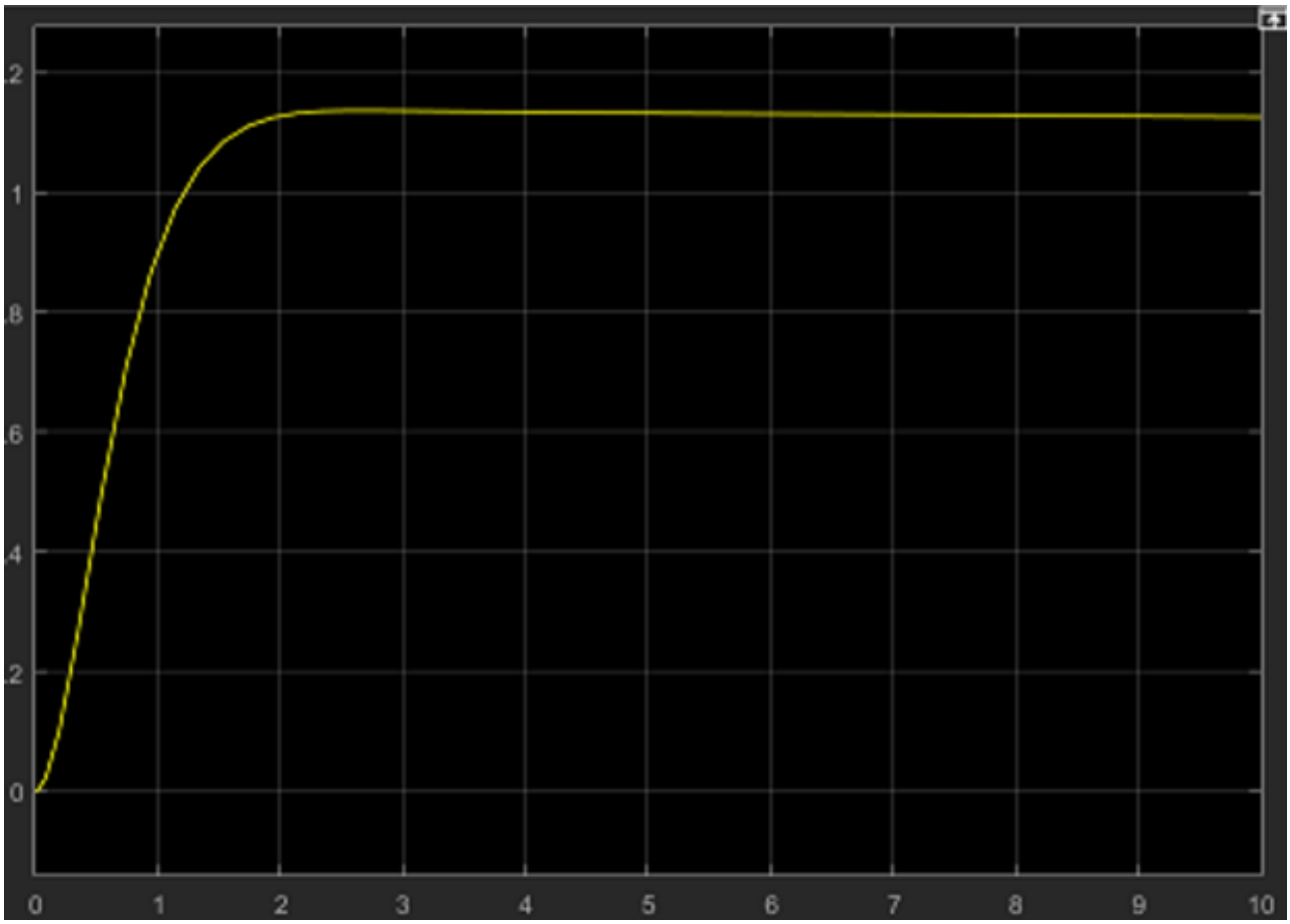

Figura 11: Scoope de Lazo cerrado

10.4.3. Prototipo Físico

El prototipo físico utiliza un servo MG996R para ajustar el ángulo del riel y un sensor ultrasónico para medir la posición de la pelota en el riel. El controlador PID se implementa en un Arduino, que ajusta el ángulo del servo en respuesta a la retroalimentación del sensor.

A través del siguiente link se puede observar el funcionamiento físico del sistema: https://youtu.be/xRiuweFz480?si=nZDVp_I_vt831gNs

Por otro lado, tenemos la siguiente simulación para el sistema físico pero usando simulink.

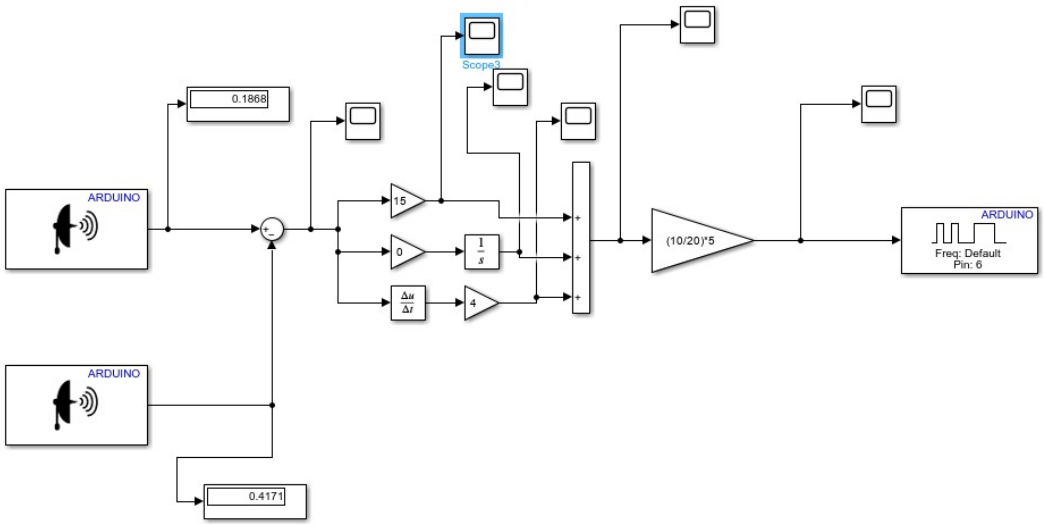

Figura 12: Estructura básica de simulink para correr el prototipo de manera física

10.5. Conclusión del Análisis de Lazo Cerrado

El análisis de lazo cerrado demuestra que el sistema es estable y responde de manera efectiva a las entradas cuando las ganancias PID están bien ajustadas. La respuesta transitoria, como el tiempo de subida y el sobreshoot, depende de los parámetros PID. El sistema físico sigue de cerca el comportamiento simulado, con pequeñas diferencias atribuidas a la no linealidad y la fricción en la planta.

11. Análisis del funcionamiento de su simulación/prototipo

11.1. Introducción

El sistema Ball & Beam es un sistema de control dinámico en el que se busca estabilizar una bola sobre un riel. Para ello, se utiliza un servo motor que ajusta el ángulo del riel y un sistema de control PID para mantener la bola en una posición fija. Este proyecto fue implementado utilizando un Arduino, sensores ultrasónicos para medir la posición de la bola y un servo MG996R para controlar el ángulo del riel.

11.2. Descripción del Sistema

El sistema se compone de los siguientes elementos:

- **Arduino Uno**: Controlador principal que procesa los datos de los sensores y controla el servo.

- **Sensor ultrasónico (Sensor 1 y Sensor 2)**: Se utilizan dos sensores ultrasónicos para medir la distancia entre la bola y las referencias fijas (cubos y carrito). El sensor 1 mide la posición del carrito, mientras que el sensor 2 mide la posición del cubo.
- **Servo MG996R**: Controla el ángulo del riel para mover la bola a la posición deseada.
- **Controlador PID**: Regula el ángulo del riel en función de la diferencia entre la posición medida y la posición objetivo.

El código de control implementa un PID para controlar el servo en función de la posición de la bola. La entrada del controlador es la diferencia entre la posición actual de la bola y la posición de referencia, y la salida es el ángulo al que debe moverse el servo para mantener la bola en la posición deseada.

11.3. Funcionamiento del Código

El código está diseñado para medir la posición de la bola utilizando dos sensores ultrasónicos. Los valores de distancia obtenidos por cada sensor son procesados para obtener la posición de la bola y compararla con la posición de referencia (setpoint).

11.3.1. Inicialización y Configuración

Al inicio del programa, se configuran los pines de los sensores ultrasónicos y el servo. Se realiza una inicialización de las variables necesarias, como las ganancias del controlador PID, y se realiza un ajuste inicial del servo para asegurar que el sistema comience desde una posición conocida. Los sensores se leen en intervalos regulares y las mediciones se filtran digitalmente para reducir el ruido.

11.3.2. Control PID

El controlador PID calcula tres componentes: Proporcional (P), Integral (I), y Derivativo (D). Los términos se calculan de la siguiente manera:

- **Proporcional (P)**: Se calcula como el error entre la posición deseada (`setpoint`) y la posición actual de la bola (`y`).
- **Integral (I)**: Se acumula el error a lo largo del tiempo para compensar el error persistente.
- **Derivativo (D)**: Se calcula como la diferencia de la posición de la bola con respecto a la posición anterior, lo que ayuda a prever cambios rápidos en la dinámica del sistema.

El valor de la salida del controlador (`U`) se obtiene sumando los tres términos, y se asegura que este valor esté dentro de los límites físicos del servo. Si la salida es demasiado grande o pequeña, se realiza una saturación para evitar que el servo se mueva fuera de su rango.

El servo se mueve a la posición calculada (`U`) solo si el error es lo suficientemente grande para justificar un movimiento. De lo contrario, el servo se mantiene en su posición actual para evitar movimientos innecesarios.

11.3.3. Medición de la Posición

Las funciones `measure_1()` y `measure_2()` se encargan de medir la distancia de los sensores ultrasónicos. Estas funciones envían señales de pulso a los pines de disparo de los sensores y miden el tiempo que tarda el pulso en regresar al pin de eco. La distancia se calcula en metros a partir de este tiempo, y los resultados se filtran para reducir el ruido y mejorar la estabilidad del sistema.

11.3.4. Control de Saturación

El sistema tiene en cuenta la saturación del controlador. Si la salida calculada (`U`) excede los límites máximos o mínimos permitidos por el servo, el sistema limita el valor de `U` para evitar que el servo intente moverse a una posición fuera de su rango físico.

11.4. Resultados y Análisis

El sistema muestra una respuesta efectiva al control PID, estabilizando la bola en la posición deseada. Sin embargo, debido a la dinámica no lineal del sistema, el rendimiento del sistema depende de las ganancias PID elegidas y las características físicas del servo y los sensores. Las ganancias PID fueron ajustadas como sigue:

- **Proporcional (Kp)** = 8.6
- **Integral (Ki)** = 1.1
- **Derivativo (Kd)** = 6.3

Estos valores permiten un buen rendimiento en la mayoría de los casos, pero aún pueden mejorarse para optimizar la respuesta del sistema, especialmente en presencia de perturbaciones o cambios en la carga.

12. Conclusión

El sistema **Ball & Beam**, implementado con un controlador PID y basado en un Arduino Uno, ha demostrado ser eficaz en la estabilización de una bola sobre un riel ajustable. A través de la utilización de dos sensores ultrasónicos para medir la posición de la bola y un servo MG996R para controlar el ángulo del riel, el sistema ha logrado mantener la bola en una posición deseada con una alta precisión, respondiendo adecuadamente a las perturbaciones.

El uso de un controlador PID, con ganancias ajustadas en función de la respuesta dinámica del sistema, ha permitido que el sistema responda de manera estable y eficiente. Las ganancias PID seleccionadas — Proporcional $K_p = 8{,}6$, Integral $K_i = 1{,}1$ y Derivativa $K_d = 6{,}3$ — ofrecen un balance entre rapidez de respuesta y estabilidad, logrando reducir el error entre la posición real de la bola y la posición objetivo. Sin embargo, la naturaleza no lineal del sistema y las limitaciones del servo sugieren que existe un espacio para optimizar aún más las ganancias PID para mejorar la precisión en condiciones de carga variable y perturbaciones externas.

Uno de los puntos clave en este proyecto ha sido el manejo de los sensores ultrasónicos. A pesar de las variaciones inherentes en las mediciones debido a la presencia de ruidos en el entorno, el sistema utiliza un filtro digital que ayuda a suavizar las lecturas, lo que permite una mejor estimación de la posición de la bola. El desafío principal radica en la respuesta física del servo, que, si bien es adecuada para la mayoría de las situaciones, presenta limitaciones en cuanto a rapidez y precisión en movimientos rápidos o cuando el error es grande.

El análisis de saturación en el controlador, que limita la acción de control cuando la salida excede los límites del servo, asegura que el sistema no intente movimientos fuera de su rango físico, lo que contribuye a la seguridad y estabilidad operativa del sistema. Sin embargo, este comportamiento también implica que en ciertos casos el sistema podría no alcanzar la posición deseada con la precisión esperada, especialmente cuando se producen grandes cambios en la referencia.

A nivel físico, el prototipo muestra un comportamiento satisfactorio, con una respuesta rápida a los cambios en la referencia, pero es importante señalar que el control PID podría beneficiarse de una mayor precisión en la estimación del ángulo de inclinación del riel y de mejoras en la dinámica del servo. El sistema puede ser optimizado utilizando un controlador adaptativo o incluso una combinación de control PID con técnicas de control predictivo para mejorar la respuesta ante cambios abruptos o cargas dinámicas.

En resumen, el proyecto **Ball & Beam** demuestra la efectividad del control PID en sistemas dinámicos no lineales y subraya la importancia de un ajuste adecuado de las ganancias del controlador, así como de una correcta integración de los sensores y actuadores. Si bien el sistema ha alcanzado una alta estabilidad y rendimiento en condiciones normales, las futuras mejoras podrían centrarse en la optimización de los algoritmos de control, el perfeccionamiento de la medición de la posición de la bola y la mejora de la respuesta del servo, con el objetivo de lograr un sistema aún más robusto, rápido y preciso. Este proyecto no solo demuestra la viabilidad del sistema en su forma actual, sino que también sienta las bases para exploraciones futuras en el campo de los sistemas de control dinámico y la robótica.

13. Hoja de software

Código 1: Código Arduino para control del sistema Ball & Beam

```
#include <Servo.h>

// Definiciones de l mites para el control del servo
#define Umax 66 // Grados m ximos
#define Umin -66 // Grados m nimos
#define Umax_rad 1.151 // Radianes m ximos
#define Umin_rad -1.151 // Radianes m nimos
#define T 0.09 // Per odo de muestreo (segundos)

// Pines de los sensores ultras nicos
```

```
const int echoPin2 = 4;
const int trigPin2 = 3;
const int echoPin1 = 6;
const int trigPin1 = 7;

Servo servo; // Objeto para el control del servo

// Variables globales para el control PID
double setpoint, setpoint_prec;  // Punto de referencia (en metros)
double y, y_prec;                // Posici n actual (en metros)
double error;                    // Error entre referencia y medici n
double P, I, D, U;               // Componentes del PID y se al de control
double I_prec = 0, U_prec = 0, D_prec = 0; // Valores previos de PID
boolean Saturation = false;      // Bandera de saturaci n

// Ganancias del controlador PID
double Kp = 8.6;
double Ki = 1.1;
double Kd = 6.3;

// Prototipos de funciones
float measure_1(void);
float measure_2(void);
void move_servo(int);

void setup() {
   Serial.begin(9600); // Inicializar comunicaci n serial

   // Configurar pines de los sensores ultras nicos
   pinMode(trigPin2, OUTPUT);
   pinMode(echoPin2, INPUT);
   pinMode(trigPin1, OUTPUT);
   pinMode(echoPin1, INPUT);

   // Configurar y centrar el servo
   servo.attach(9);
   delay(1000);
   move_servo(90);
   delay(2000);

   // Inicializar las mediciones de referencia
   setpoint_prec = measure_2();  // Punto de referencia inicial
   delay(1000);
   y_prec = measure_1();         // Posici n inicial
   delay(1000);
```

```
}

void loop() {
    // Leer el punto de referencia desde el sensor ultras nico 2
    setpoint = measure_2();
    // Aplicar un filtro digital para suavizar la se al
    setpoint = 0.53 * setpoint + 0.47 * setpoint_prec;

    delay(3); // Breve espera

    // Leer la posici n actual desde el sensor ultras nico 1
    y = measure_1();
    // Aplicar un filtro digital a la posici n medida
    y = 0.53 * y + 0.47 * y_prec;

    delay(3); // Breve espera

    // Calcular el error entre la referencia y la posici n actual
    error = round(100 * (y - setpoint)) * 0.01;

    // Controlador PID
    P = Kp * error; // Componente proporcional

    // Actualizar la integral solo si no hay saturaci n
    if (!Saturation)
        I = I_prec + T * Ki * error;

    // Componente derivativa
    D = (Kd / T) * (y - y_prec);
    // Filtrar la derivada para suavizar la se al
    D = 0.56 * D + 0.44 * D_prec;

    // Se al de control
    U = P + I + round(100 * D) * 0.01;

    // Saturar la se al de control en radianes
    if (U < Umin_rad) {
        U = Umin_rad;
        Saturation = true;
    } else if (U > Umax_rad) {
        U = Umax_rad;
        Saturation = true;
    } else {
        Saturation = false;
    }
```

```
    // Convertir la se al de control de radianes a grados
    U = round(U * 180 / M_PI);
    U = map(U, Umin, Umax, 24, 156);

    // Mover el servo si el error es significativo
    if (U < 83 || U > 95 || abs(error) > 0.02)
        move_servo(round(U));

    delay(24); // Espera para estabilidad del sistema

    // Actualizar valores previos
    I_prec = I;
    y_prec = y;
    D_prec = D;
    setpoint_prec = setpoint;
}

// Funci n para medir la posici n actual con el sensor ultras nico 1
float measure_1(void) {
    long elapsed_time = 0;
    float distance = 0;

    // Generar pulso para el sensor
    digitalWrite(trigPin1, LOW);
    delayMicroseconds(10);
    digitalWrite(trigPin1, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin1, LOW);

    // Leer el tiempo del pulso de retorno
    elapsed_time = pulseIn(echoPin1, HIGH);
    distance = (float)elapsed_time / 58.2;

    delay(30); // Espera m nima entre mediciones

    // Limitar el rango de medici n
    if (distance > 42) distance = 43;
    else if (distance < 0) distance = 0;

    return 0.01 * (distance - 1.5 + 0.5); // Conversi n a metros
}

// Funci n para medir la referencia con el sensor ultras nico 2
float measure_2(void) {
```

```
    long elapsed_time = 0;
    float distance = 0;

    // Generar pulso para el sensor
    digitalWrite(trigPin2, LOW);
    delayMicroseconds(10);
    digitalWrite(trigPin2, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin2, LOW);

    // Leer el tiempo del pulso de retorno
    elapsed_time = pulseIn(echoPin2, HIGH);
    distance = (float)elapsed_time / 58.2;

    delay(30); // Espera m nima entre mediciones

    // Limitar el rango de medici n
    if (distance > 42) distance = 43;
    else if (distance < 0) distance = 0;

    return 0.01 * (distance + 2); // Conversi n a metros
}

// Funci n para mover el servo seg n la se al de control
void move_servo(int u) {
    servo.write(u - map(u, 30, 150, 14, 3));
}
```

Referencias

[1] N. Autor, "Código arduino para ball & beam," *Desarrollado para este proyecto*, 2024.

[2] K. Ogata, "Modern control engineering," 2010. [En línea]. Disponible: https://ctms.engin.umich.edu

[2]