

22.85 - Sistemas de Control

Implementación de un sistema de balanceo de bola mediante control PID

Matías Bergerman

Phillipe Dutriez

Francisco Ledesma

Xi Lin

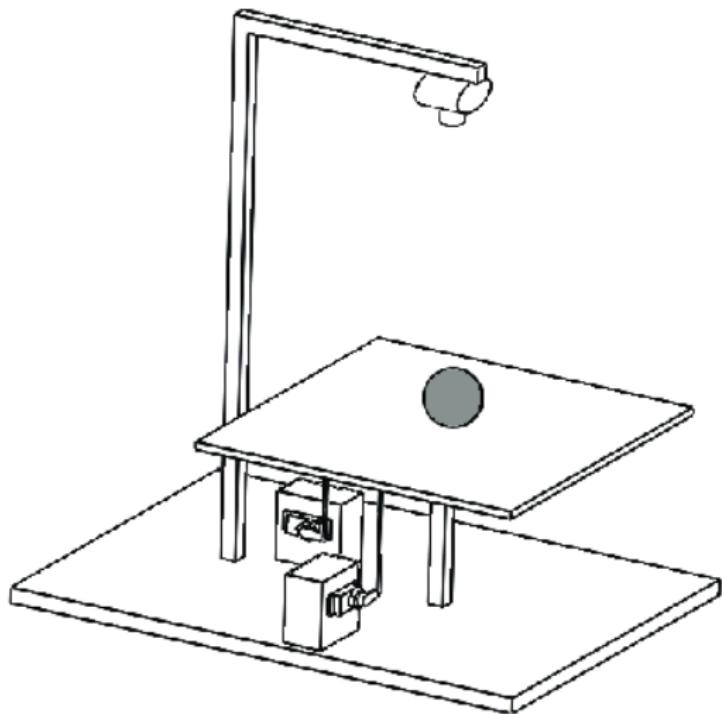
Pablo Smolkin

Instituto Tecnológico de Buenos Aires

14 de marzo de 2022

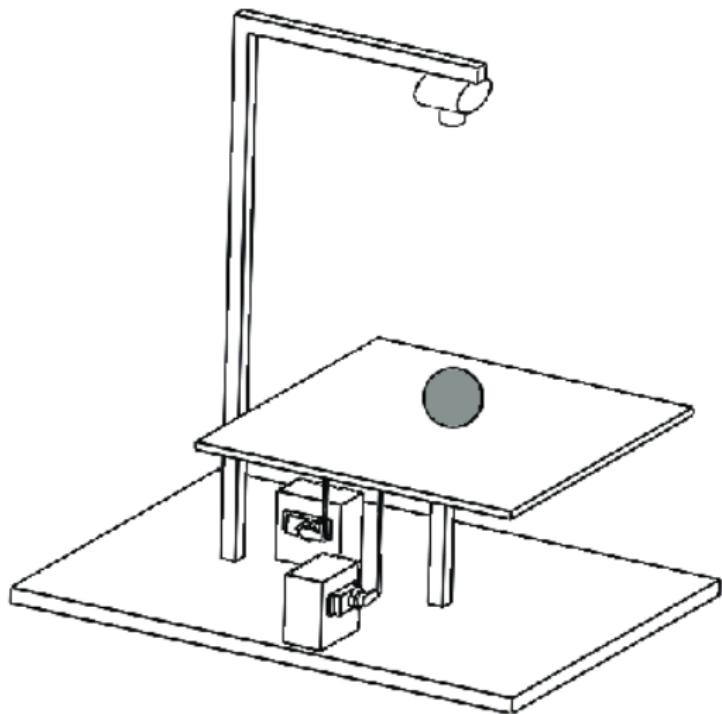
Introducción

El objetivo del proyecto es balancear una bola sobre una plataforma plana regulando la inclinación de la misma respecto de dos ejes ortogonales.



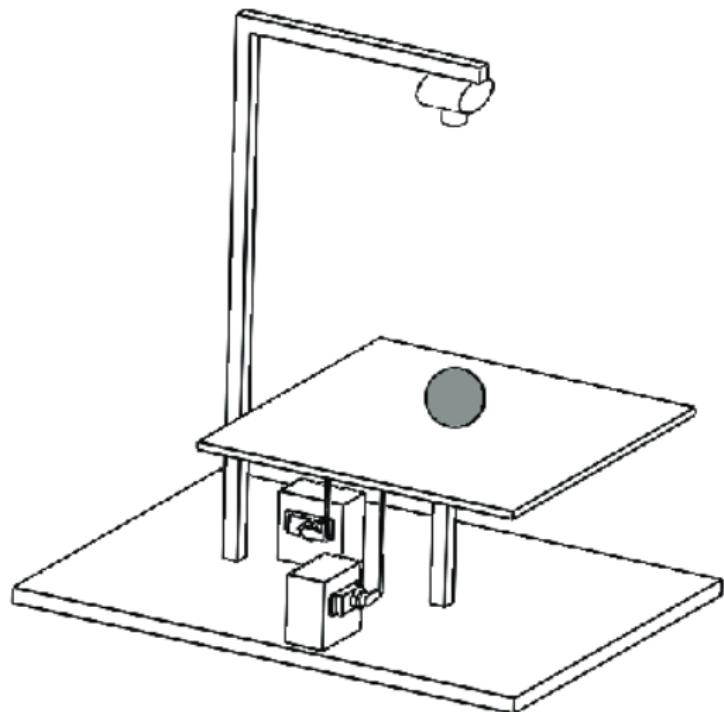
Introducción

La posición de la bola es determinada mediante una cámara, la cual se encuentra colocada sobre la plataforma, cuya imagen se procesa utilizando una micro-computadora.



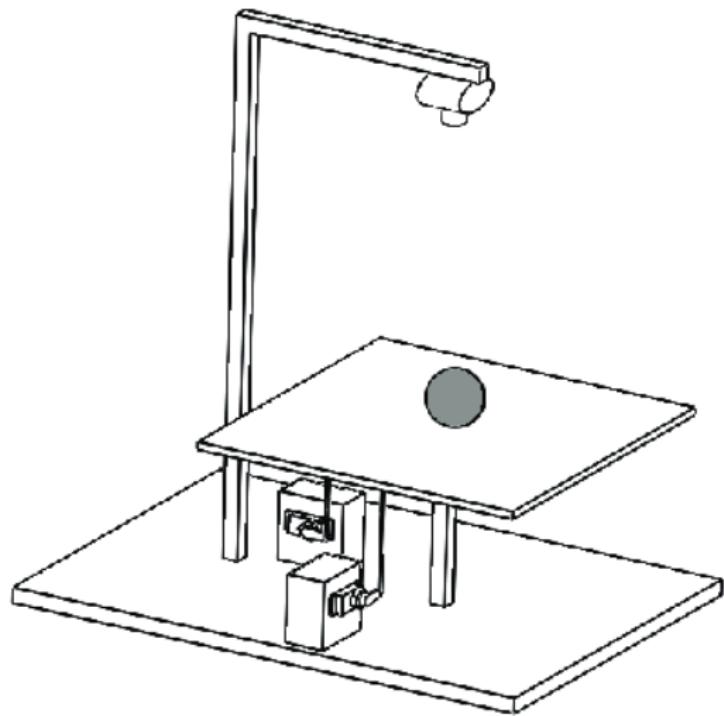
Introducción

A partir de la posición medida de la bola y la posición deseada, se determina la inclinación apropiada de la plataforma.



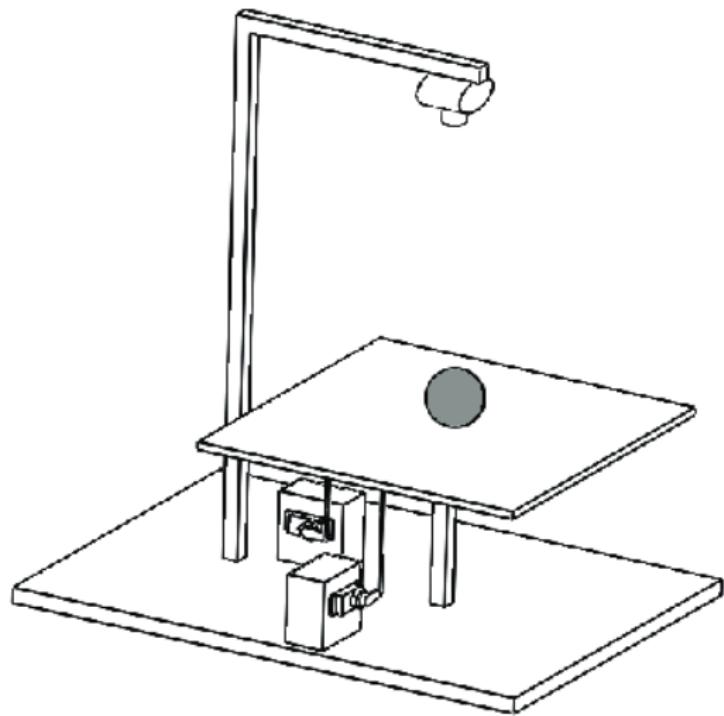
Introducción

La inclinación de la plataforma se consigue mediante dos servo motores que modifican el ángulo de la misma en sus dos ejes ortogonales de forma independiente.



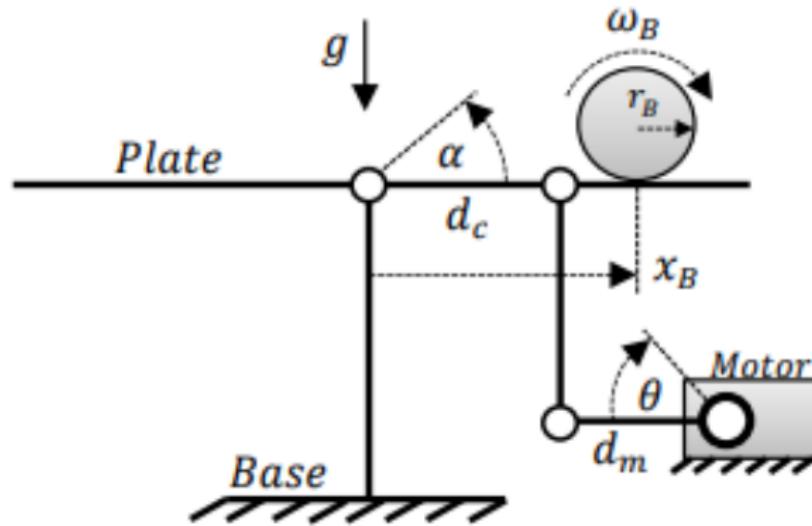
Introducción

La micro-computadora se encuentra conectado mediante internet a un servidor MQTT, envía datos en tiempo real sobre la posición de la bola medida y la posición deseada. Estos datos son graficados en tiempo real mediante un programa Python.



Modelado

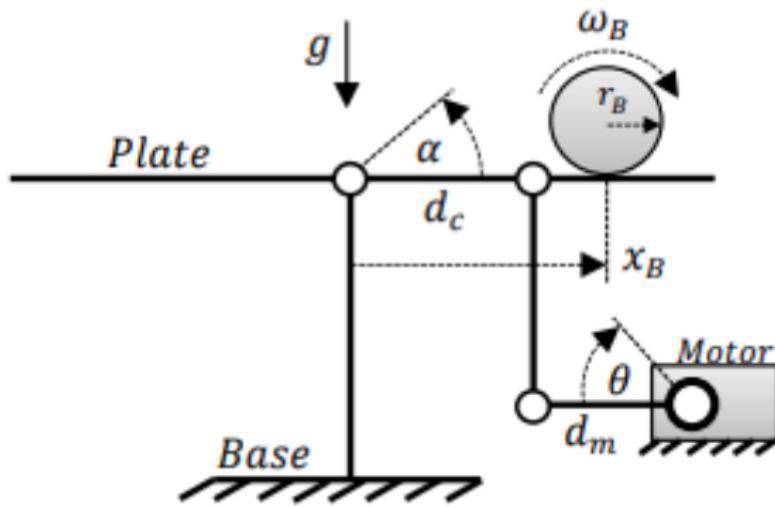
Inicialmente buscamos un modelo que describa la posición de la bola respecto del ángulo del servomotor que controla cada eje.



Modelo físico

Fuerza translacional $F_{x,t}$ y fuerza de inercia $F_{x,r}$.

$$m_b \ddot{x}_b(t) = \sum F = F_{x,t} + F_{x,r} \quad (1)$$



Modelo físico

Fuerza Traslacional:

$$F_{x,t} = m_b g \sin(\alpha(t)) \quad (2)$$

Fuerza de la Inercia:

$$F_{x,r} = I_b \frac{\ddot{x}_b(t)}{r_b^2} \quad (3)$$

Aceleración de la bola:

$$\ddot{x}_b(t) = \frac{m_b g \sin(\alpha(t)) r_b^2}{m_b r_b^2 + I_b} \quad (4)$$

Modelo físico

$$\sin(\alpha(t)) = \sin(\theta(t)) \cdot \frac{2d_m}{L} \Rightarrow \alpha(t) = \theta(t) \cdot \frac{2d_m}{L} \quad (5)$$

La transferencia final del modelo es:

$$G(s) = \frac{X(s)}{\Theta(s)} = \frac{2m_b r_b^2 d_m g}{L(m_b r_b^2 + I_b)} \cdot \frac{1}{s^2} = \frac{K_b}{s^2} \quad (6)$$

Calculo de constantes

Cálculo de las constantes:

$$PID(s) = K \cdot \frac{(s - z_d)(s - z_i)}{s} \quad (7)$$

$$= K \cdot s - K \cdot (z_p + z_i) + K \cdot (z_p * z_i) \cdot \frac{1}{s} \quad (8)$$

$$= K_d s + K_p + K_i \frac{1}{s} \quad (9)$$

Valores calculados usando Matlab: Valores efectivamente utilizados:

$$K_p = 9,69$$

$$K_p = 0,5$$

$$K_i = 0,024$$

$$K_i = 0,3$$

$$K_d = 1,0075$$

$$K_d = 0,3$$

Espacio de Estados

Espacio de estados:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{10}$$

Donde $x(t)$ es el vector de estados e $y(t)$ es el vector de salida y por ultimo $u(t)$ es el vector de entrada:

$$x(t) = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad y(t) = [y] \quad u(t) = [\theta] \tag{11}$$

Controlabilidad y Observabilidad

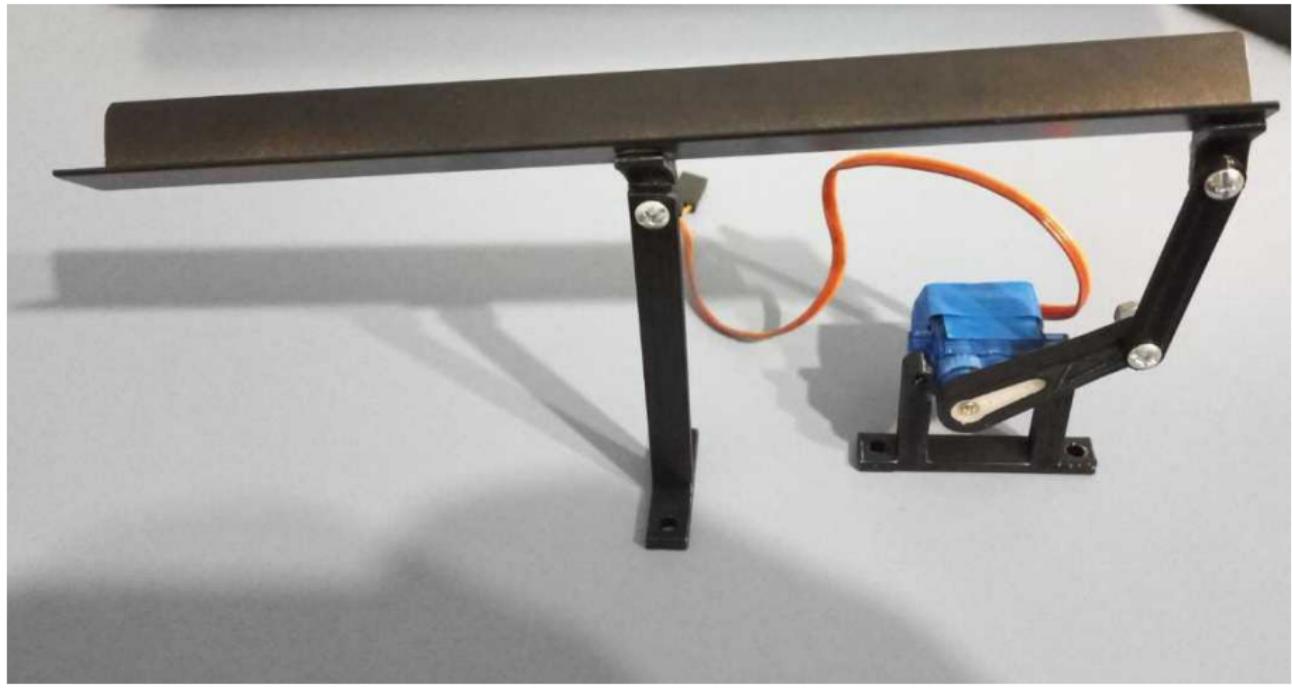
Donde x es la posición de la bola y \dot{x} es la aceleración en su eje correspondiente:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ K_b \end{bmatrix} \Theta \quad (12)$$

$$y = [1 \ 0] \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad (13)$$

Calculando la constante K_b se comprobó por MATLAB que el sistema es **totalmente controlable y observable**.

Hardware: Prototipado



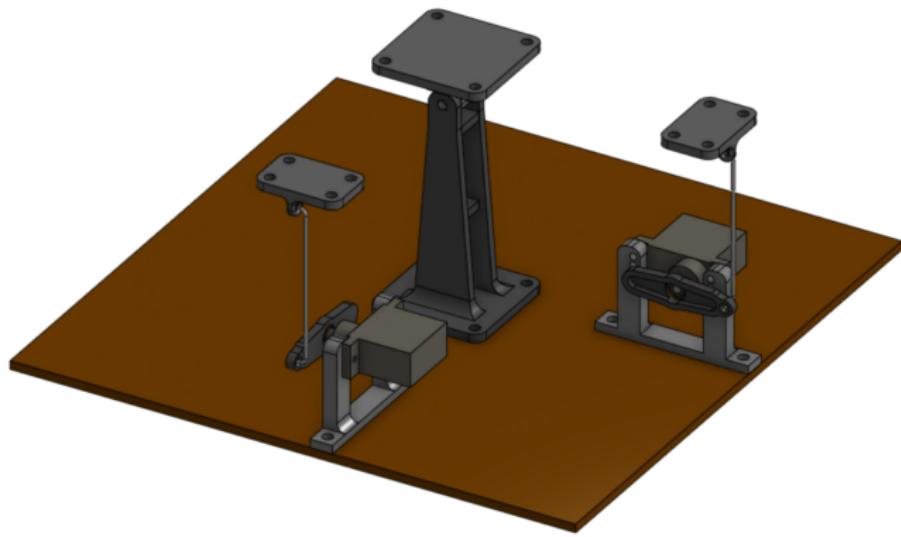
Hardware

Para el armado del proyecto se utilizó una plataforma de madera de 25cm x 25cm que fue sujetada mediante un cardán impreso en 3D:



Hardware

Modelado 3D del sistema completo:



Vista superior del sistema sin la plataforma de balanceo.

Hardware

Modelado 3D del sistema completo:

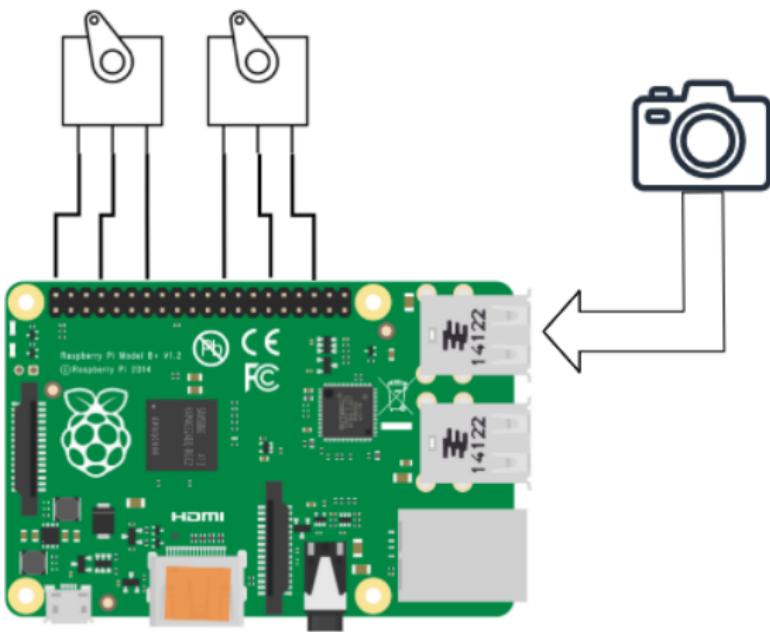


Vista inferior del sistema sin su base de apoyo.

Hardware

- Se utilizó una cámara **Logitech C920 HD PRO** para la adquisición de la posición de la bola mediante conexión USB.
- Se utilizó un tubo de PVC en forma de L para lograr colocar la cámara por encima de la plataforma de forma tal que se encuentre centrada respecto de la plataforma.
- Para el control de la inclinación de la mesa se utilizaron 2 servomotores **MG90S** conectados a la plataforma mediante un alambre rígido para transmitir el movimiento de los motores.
- Se utilizó una **Raspberry Pi 4B** para el control de los servomotores y la adquisición y procesamiento de la información de la cámara.

Hardware



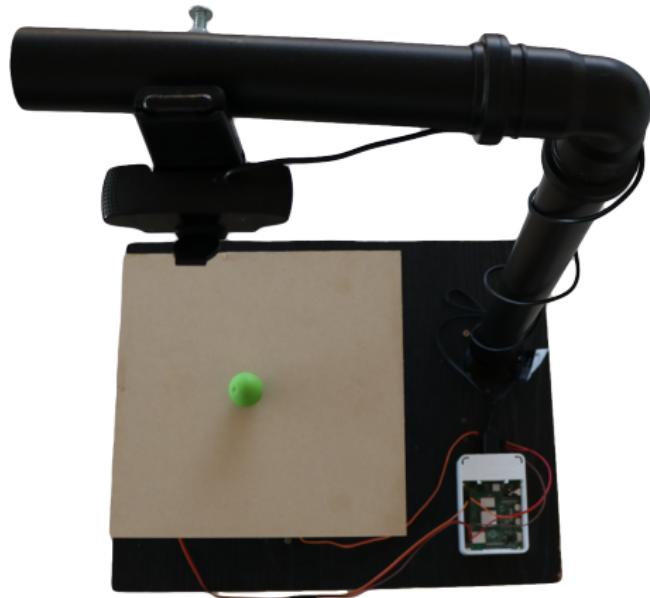
Esquemático de conexión

Hardware



Dispositivo construido.

Hardware



Dispositivo construido.

Software: Detección de la bola

Las imágenes capturadas por la cámara web son analizadas por un programa en Python que utiliza OpenCV.

Los pasos que sigue este algoritmo son:

- Reducir la resolución de la imagen para acelerar el tiempo de procesamiento.
- Aplicar un *gaussian blur* a la imagen.
- Transformar la imagen RGB al espacio de color HSV.
- Aplicar una máscara que preserve únicamente los píxeles cuyos valores HSV se encuentren dentro del rango apropiado del color de la bola.
- Aplicar una serie de “erosiones” y “dilataciones” para remover cualquier pequeña mancha que haya quedado.
- Buscar el contorno más grande que haya quedado en la imagen.
- Encontrar el centroide del círculo envolvente del contorno.

Software: Control de motores

Los servomotores utilizados cuentan con terminales de alimentación independientes de la señal de datos.

Los mismos pueden ser controlados por la Raspberry Pi sin la necesidad de utilizar hardware adicional para el control de los motores.

Para que cada motor se posicione en el ángulo deseado, se debe generar una señal PWM con un *duty-cycle* apropiado. La Raspberry Pi cuenta con controladores de PWM por hardware los cuales fueron empleados para esta tarea.



Software: PID

Para la implementación del PID simplemente se utilizaron las siguientes instrucciones:

```
dt = time() - previous_time
```

```
previous_time = time()
```

```
P = error
```

```
I = I + (error * dt)
```

```
D = (error - previous_error) / dt
```

```
output = (Kp * P) + (Ki * I) + (Kd * D)
```

Software: Problemas con PID

La medición de la posición de la bola tiene presente ruido significativo, debido principalmente a:

- Vibración del dispositivo.
- Precisión de la cámara
- Latencia variable del sistema.
- Pequeñas inconsistencias en el algoritmo de detección.

El problema fue solucionado colocando únicamente a la entrada del bloque derivativo un filtro pasa-bajos para reducir la amplitud del ruido de alta frecuencia que este recibe. Para simplificar la implementación del filtro simplemente se utilizó una media móvil de 5 “taps” .

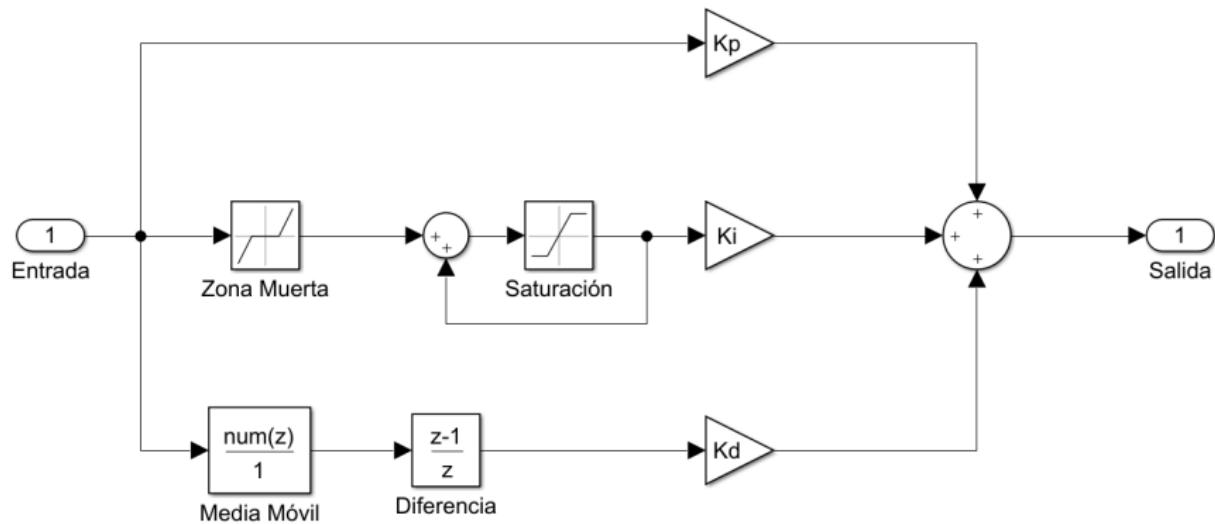
Software: Problemas con PID

Otro problema encontrado es el *windup* del componente integrativo. El valor acumulado del error podía crecer más allá de lo que es razonable, y debía ser saturado. Esto previene que si por alguna razón la bola permanece durante un largo tiempo alejada del *set point*, la integral no crezca ilimitadamente imposibilitando que cuando se libere a la bola esta se estabilice en el objetivo.

Por último, se encontró que existe una “zona muerta” en el sistema. Esto es, si la bola se encuentra detenida se requiere un ángulo de inclinación mínimo para que esta comience a moverse. Esto provoca que pequeños errores se acumulen en la integral y luego de cierto tiempo la bola sea disparada cuando la inclinación de la plataforma supera el umbral mínimo que provoca el movimiento de la bola. Para mitigar este problema se aplicó una “zona muerta” al valor del error que ingresa al bloque integrador, de forma tal que ignore valores muy pequeños del error.

Software: PID

Diagrama resultante:



Software: Transmisión de datos en tiempo real

Se implementó un cliente sobre la Raspberry Pi que utiliza el protocolo MQTT comúnmente usado en el ámbito de IoT. Se envía información a un servidor externo la cual es accesible por cualquier otro cliente MQTT conectado al mismo servidor. Estos datos son recibidos mediante una terminal y graficados en el tiempo utilizando Python.

