



## Instituto Tecnológico de Buenos Aires

22.85 - SISTEMAS DE CONTROL

## PROYECTO FINAL

Autores:

57568 Ledesma, Francisco Daniel

59009 Dutriez, Philippe

59523 Smolkin, Pablo

60831 Bergerman, Matías

59780 Lin, Xi

# Contenido

<b>1. Introducción</b>	<b>1</b>
<b>2. Modelado</b>	<b>1</b>
2.1. Modelo físico . . . . .	2
2.2. Controlabilidad . . . . .	3
<b>3. Hardware</b>	<b>3</b>
<b>4. Software</b>	<b>5</b>
4.1. Detección de la bola . . . . .	5
4.2. Control de motores . . . . .	5
4.3. PID . . . . .	6
4.4. Transmisión de datos en tiempo real . . . . .	7

## 1. Introducción

Como proyecto final se eligió realizar un sistema de balanceo de bola mediante un control del tipo Proporcional, Integrativo y Derivativo (PID) como el que puede visualizarse de manera conceptual en la figura 1.1. El objetivo del mismo es balancear una bola sobre una plataforma plana regulando la inclinación de la misma respecto de dos ejes ortogonales.

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 (la cual se detallará en la sección de [Hardware](#)). A partir de la posición de la bola medida y la posición deseada se determina cómo debe inclinarse la plataforma mediante dos servomotores que modifican el ángulo de la misma en sus dos ejes ortogonales de forma independiente. Por otro lado, la micro-computadora se encuentra conectada mediante internet a un servidor MQTT<sup>1</sup> de forma tal que envía constantemente datos en tiempo real sobre la posición de la bola medida y la posición deseada. Estos datos pueden ser analizados desde cualquier terminal conectada al mismo servidor, y ser graficados en tiempo real mediante un programa Python.

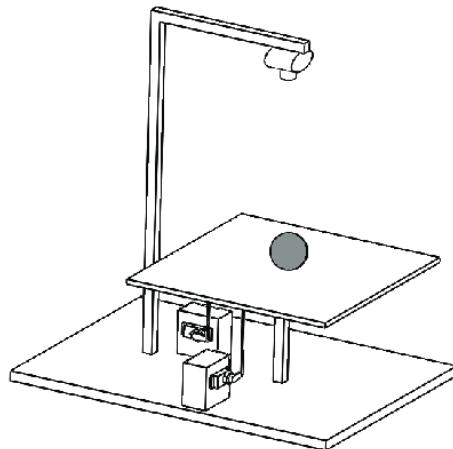


FIGURA 1.1: Boceto del sistema implementado.

## 2. Modelado

La plataforma de balanceo es simétrica respecto de sus ejes ortogonales, es por esto que se puede asumir que el funcionamiento del sistema de los ejes  $x$  e  $y$  son independientes. Por lo tanto, es posible modelar al sistema como dos sistemas de “ball and beam” independientes. Inicialmente buscamos un modelo que describa la posición de la bola respecto del ángulo del servomotor que controla cada eje.

---

<sup>1</sup><https://mqtt.org/>

## 2.1. Modelo físico

El esquema del modelo físico para cada eje es el de la fig. 2.1. Se puede ver entonces que la sumatoria de fuerzas que actúan sobre la bola son la fuerza translacional  $F_{x,t}$  y la fuerza generada por la inercia  $F_{x,r}$ .

$$m_b \ddot{x}_b(t) = \sum F = F_{x,t} + F_{x,r} \quad (2.1)$$

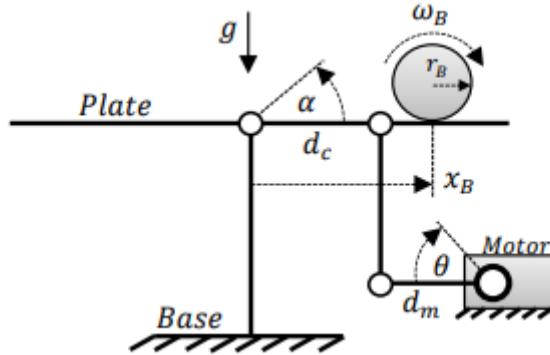


FIGURA 2.1: Modelo físico de los ejes

Donde la fuerza de translación esta definida como:

$$F_{x,t} = m_b g \sin(\alpha(t)) \quad (2.2)$$

Y la fuerza de rotación es:

$$F_{x,r} = I_b \frac{\ddot{x}_b(t)}{r_b^2} \quad (2.3)$$

Quedando entonces la aceleración de la bola como:

$$\ddot{x}_b(t) = \frac{m_b g \sin(\alpha(t)) r_b^2}{m_b r_b^2 + I_b} \quad (2.4)$$

Definiendo la altura del codo de la unión del servomotor respecto de la posición inicial como  $h(t)$ , queda entonces la relación entre la inclinación del servomotor  $\sin(\theta(t)) = h(t)/d_m$  y la inclinación de la plataforma  $\sin(\alpha(t)) = h(t)/d_m$  como la expresión 2.5.

$$\sin(\alpha(t)) = \sin(\theta(t)) \cdot \frac{2d_m}{L} \Rightarrow \alpha(t) = \theta(t) \cdot \frac{2d_m}{L} \quad (2.5)$$

Por último, reemplazando la aproximación paraxial y escribiendo la transferencia de Laplace, la transferencia del modelo es:

$$G(s) = \frac{X(s)}{\Theta(s)} = \frac{2m_b r_b^2 d_m g}{L(m_b r_b^2 + I_b)} \cdot \frac{1}{s^2} = \frac{K_b}{s^2} \quad (2.6)$$

## 2.2. Controlabilidad

La definición de la controlabilidad para cada eje esta definida de la siguiente manera:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{2.7}$$

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:

$$x(t) = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad y(t) = \begin{bmatrix} y \end{bmatrix} \quad u(t) = \begin{bmatrix} \theta \end{bmatrix} \tag{2.8}$$

Donde  $x$  es la posición de la bola y  $\dot{x}$  es la aceleración en su eje correspondiente. Por lo tanto el espacio estado queda definido de la siguiente manera:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ K_b \end{bmatrix} \Theta \tag{2.9}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{2.10}$$

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

## 3. Hardware

Para el armado del proyecto se utilizó una plataforma de madera de 25cm x 25cm que fue sujetada mediante un cardán modelado según la figura 3.1 e impreso en 3D. En las figuras 3.2 y 3.3 se pueden ver modelos del sistema construido.

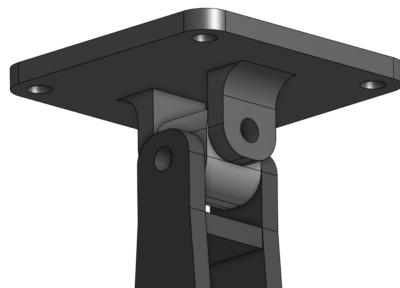


FIGURA 3.1: Modelo del Cardán

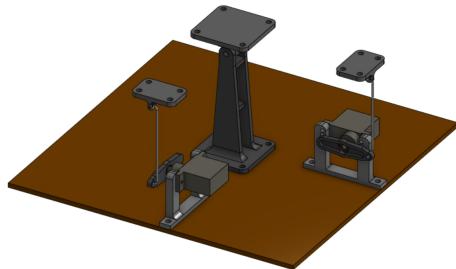


FIGURA 3.2: Vista superior del sistema sin la plataforma de balanceo.



FIGURA 3.3: Vista inferior del sistema sin su base de apoyo.

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 misma.

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.



FIGURA 3.4: Dispositivo construido.

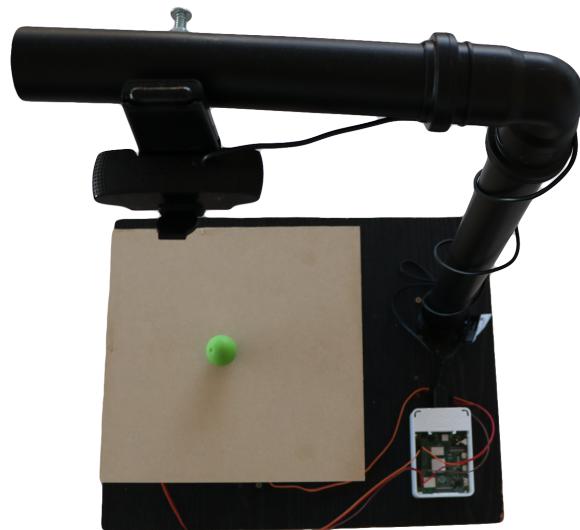


FIGURA 3.5: Dispositivo construido.

Se utilizó una [Raspberry Pi 4B](#) para el control de los servomotores y la adquisición y procesamiento de la información de la cámara como muestra el esquemático de la figura 3.6.

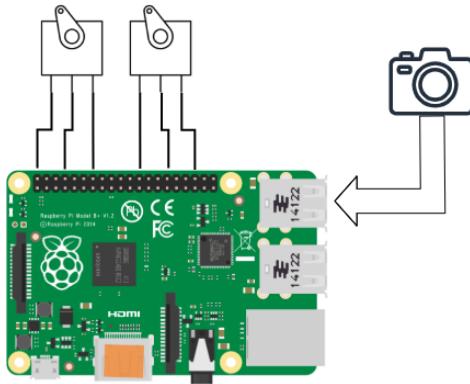


FIGURA 3.6: Esquemático de conexión

## 4. Software

### 4.1. Detección de la bola

Para lograr obtener la posición de la bola, las imágenes capturadas por la cámara web son analizadas por un programa en Python que utiliza OpenCV<sup>2</sup>. Para esto se adaptó un popular código ejemplo que detecta objetos circulares del color indicado en una imagen.

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.

Dado que este es un procedimiento conocido en el ámbito del procesamiento de imágenes y que no es el foco de este trabajo, no será detallado en profundidad.

### 4.2. Control de motores

Los servomotores utilizados cuentan con terminales de alimentación independientes de la señal de datos. Es por esto que los mismos pueden ser controlados por la Raspberry Pi sin la necesidad de utilizar hardware adicional para el control de los motores.

---

<sup>2</sup><https://opencv.org/>

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.

### 4.3. 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)
```

Sin embargo, se observaron inconvenientes con esta implementación que debieron ser resueltos. El principal problema es que la medición de la posición de la bola tiene presente ruido significativo, debido principalmente a la vibración del dispositivo, la precisión de la cámara, la latencia variable del sistema y las pequeñas inconsistencias en el algoritmo de detección. Este ruido de alta frecuencia se amplifica por la componente derivativa, y como consecuencia la señal enviada a los servomotores es ruidosa provocando que el sistema tenga vibraciones notorias. 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”.

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.

El diagrama resultante se puede observar en la figura 4.1.

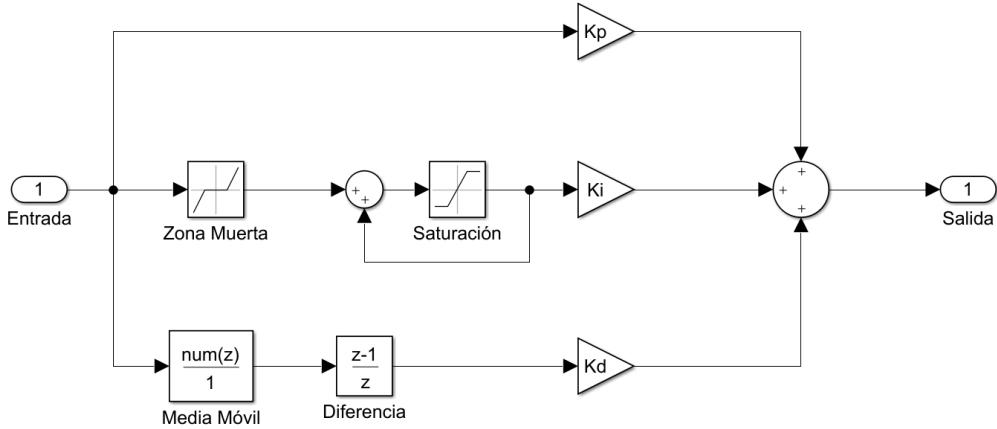


FIGURA 4.1: Diagrama del algoritmo PID modificado.

#### 4.4. Transmisión de datos en tiempo real

Para permitir la 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. De esta forma, 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. En la figura 4.2 podemos observar la aplicación desarrollada para este propósito. En la figura de la izquierda podemos visualizar la posición actual de la bola registrada por la cámara sobre la plataforma por un círculo verde y la posición deseada señalizada por una cruz. Además se puede observar el recorrido previo de la bola representada por una estela gris.

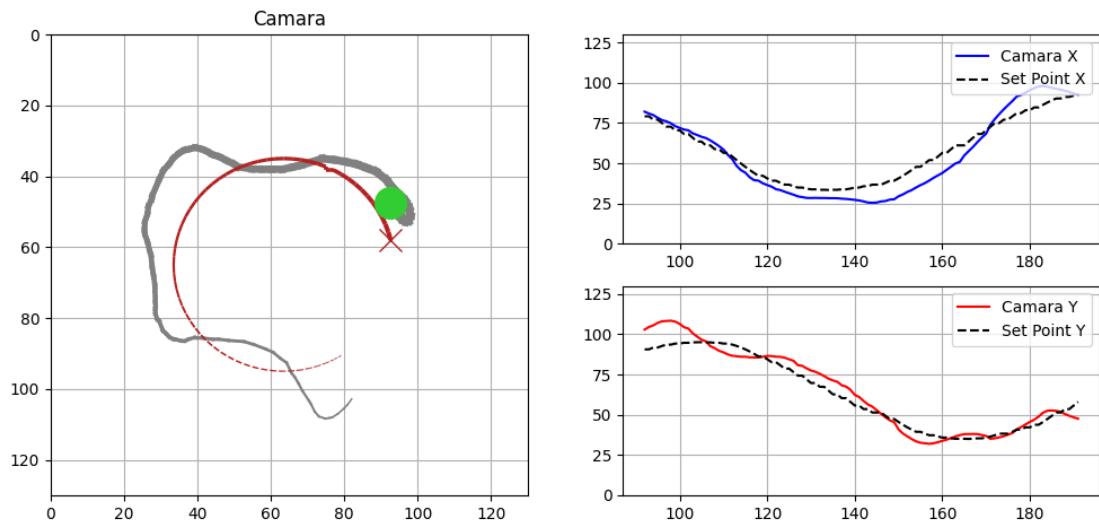


FIGURA 4.2: Gráficos de la información recibida en el tiempo

Luego, en las figuras de la derecha podemos observar la misma información pero en cada eje por separado y en el tiempo. En este ejemplo, la bola intenta describir una circunferencia con su trayectoria. Esto se ve representado como una señal senoidal para cada eje.