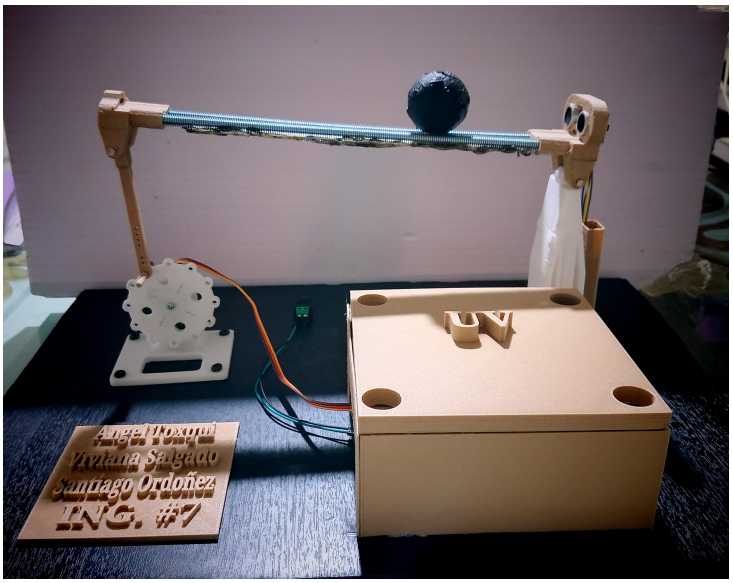
Manual de Usuario

Sistema de Control Ball and Beam



Proyecto de Ingenieria 2 y Control

Universidad del Valle - Sede Cartago

Ingeniería Electrónica

Autores:

Angel Toxqui Muñoz

Anyi Viviana Salgado Perez

Santiago Ordoñez Osorio

1 de julio de 2025

Índice

1. Descripción del Sistema

El "Ball and Beam"(bola y viga) es un sistema de control automático cuyo objetivo principal es mantener una bola en una posición deseada sobre una viga que puede inclinarse, a pesar de que el sistema es inherentemente inestable (la bola tiende a rodar hacia los extremos por gravedad).

1.1. Componentes Clave

1. **Viga y Bola:** La parte física donde la bola se mueve.
2. **Sensor Ultrasónico (HC-SR04):** Mide la posición actual de la bola sobre la viga. Este es el .ºjo"del sistema.
3. **Servomotor (MG995) con Driver PCA9685:** Actúa como el "músculo"del sistema. Inclina la viga para mover la bola.
4. **Arduino Uno:** Es el çerebro"que procesa la información y toma decisiones.
5. **Controlador PID:** Es el algoritmo de control implementado en el Arduino. Calcula cuánto debe inclinarse la viga para corregir la posición de la bola, basándose en el error entre la posición deseada y la medida.

1.2. ¿Cómo Funciona?

El sistema opera en un lazo cerrado de control:

1. El sensor ultrasónico mide dónde está la bola en ese momento.
2. Esta información se envía al Arduino.
3. El controlador PID en el Arduino compara la posición actual de la bola con la posición que se desea (el "setpoint").
4. Si hay un error (la bola no está donde debería), el PID calcula una acción de control.
5. Esta acción se traduce en una señal para el servomotor, que inclina la viga.
6. La inclinación de la viga hace que la bola ruede hacia la posición deseada, y el ciclo se repite continuamente para mantenerla estable.

1.3. Propósito

El proyecto sirve como una plataforma educativa y experimental para validar conceptos fundamentales de la ingeniería de control, como la estabilidad de sistemas, la retroalimentación, el modelado matemático y la implementación de algoritmos de control digital en hardware de bajo costo.

2. Componentes y Materiales

Componente	Cantidad	Especificaciones
Arduino Uno	1	Microcontrolador ATmega328P
Sensor Ultrasónico HC-SR04	1	Rango: 2cm - 400cm
Servomotor MG995	1	Torque: 13kg/cm (6V)
Driver PCA9685	1	16 canales PWM, I2C
Viga metálica	1	30cm de largo, 8mm de ancho
Bola metálica	1	Diámetro apropiado para la viga
Fuente de alimentación	1	5V - 1A mínimo
Cables jumper	Varios	Macho-macho, macho-hembra
Protoboard	1	Para conexiones temporales

Cuadro 1: Lista de componentes necesarios

3. Conexiones y Alimentación

3.1. Sensor Ultrasónico HC-SR04

- **VCC** → 5V de Arduino
- **GND** → GND de Arduino
- **TRIGGER** → Pin digital 8 de Arduino
- **ECHO** → Pin digital 9 de Arduino

3.2. Servomotor MG995 (a través del PCA9685)

El servomotor no se conecta directamente al Arduino, sino al driver PCA9685, que permite controlar varios servos fácilmente y proporciona la señal PWM adecuada.

3.2.1. MG995:

- **Cable rojo (VCC):** Alimentación (conecta a 5V - 1A)
- **Cable marrón (GND):** Tierra (GND)
- **Cable naranja (Señal):** Conecta a uno de los canales de salida del PCA9685 (canal 0)

3.2.2. PCA9685:

- **VCC:** 5V de Arduino
- **GND:** GND de Arduino
- **SCL:** Pin A5 de Arduino (I2C)
- **SDA:** Pin A4 de Arduino (I2C)

- **V+ (alimentación servos):** Fuente de alimentación para los servos (5V - 1A)
- **GND (alimentación servos):** GND de la fuente externa

3.3. Arduino Uno

- **Pin 8:** Trigger del HC-SR04
- **Pin 9:** Echo del HC-SR04
- **Pin A4:** SDA (I2C, para PCA9685)
- **Pin A5:** SCL (I2C, para PCA9685)

3.4. Notas de Seguridad

- Nunca conectar servomotores potentes directamente a los pines del Arduino
- Usar fuente de alimentación externa para el servomotor
- Conectar todas las tierras (GND) en común
- Verificar polaridad antes de conectar componentes

4. Lectura de Datos para Control

4.1. ¿De dónde tomar los datos del sensor ultrasónico y el servomotor?

En el sistema Ball and Beam implementado en Arduino, los datos se obtienen y manipulan principalmente en el código Arduino.

4.1.1. Sensor ultrasónico (HC-SR04)

Lectura de datos: El sensor mide la posición de la bola usando la función `sonar.ping_cm()`, que devuelve la distancia en centímetros.

```
Input = sonar.ping_cm();
if (Input == 0) {
    Input = Setpoint;
}
```

Aquí, `Input` es la variable que almacena la posición actual de la bola, medida por el sensor ultrasónico.

4.1.2. Servomotor (MG995)

Control del servomotor: El servomotor se controla a través del módulo PCA9685, usando la función `pwm.setPWM()`. El valor que se le envía depende de la salida del PID.

```
int servoPulse = map(Output, PID_OUTPUT_MIN, PID_OUTPUT_MAX,
                     SERVO_MIN_PULSE, SERVO_MAX_PULSE);
pwm.setPWM(SERVO_CHANNEL, 0, servoPulse);
```

Aquí, `Output` es la salida calculada por el PID, que se mapea al rango de pulsos que acepta el servomotor.

4.2. ¿Cómo ver su comportamiento o aplicar control?

4.2.1. Monitoreo

El código imprime por el puerto serie los valores de Setpoint, Input, Error y Output en cada ciclo del loop. Puedes ver estos datos en tiempo real usando el monitor serie de Arduino IDE:

```
Serial.print("Setpoint: "); Serial.print(Setpoint);
Serial.print(" cm | Posicion (Input): "); Serial.print(Input);
Serial.print(" cm | Error: "); Serial.print(error);
Serial.print(" cm | Salida PID (Output): ");
Serial.println(Output);
```

Así puedes graficar o analizar el comportamiento de la bola y la acción del servo.

4.2.2. Aplicar control

El control se realiza automáticamente en el loop principal, donde el PID calcula la salida en función del error entre la posición deseada (Setpoint) y la medida (Input). Si quieres cambiar el setpoint (la posición objetivo), puedes enviarlo por el monitor serie y el sistema lo actualizará.

5. Guía de Uso Paso a Paso

5.1. Montaje Físico

1. Ensamblar la estructura mecánica con las piezas 3D impresas
2. Instalar la viga metálica en los soportes
3. Montar el servomotor en su base
4. Conectar el mecanismo de inclinación entre el servo y la viga
5. Posicionar el sensor ultrasónico para medir la posición de la bola
6. Colocar la bola sobre la viga

5.2. Compilar y Hacer Funcionar el Código en Arduino

5.2.1. Pasos para compilar y cargar:

1. Abre el Arduino IDE
2. Conecta el Arduino Uno a la PC mediante el cable USB
3. Copia el código completo (el que controla el servo con PID y permite comunicación serial)
4. Selecciona el puerto y la placa:

- Ve a **Herramientas** → **Placa** → **Arduino Uno**
- Ve a **Herramientas** → **Puerto** y selecciona el correspondiente (ej. COM3, COM8, etc.)

5. Haz clic en el botón **Verificar** para compilar
6. Haz clic en el botón **Subir** (flecha a la derecha)

Si todo está correcto, te mostrará: *Subido correctamente.*

5.3. Interacción con la Aplicación Python

Acción en Python	Efecto en Arduino
Enviar 15.0	Cambia el setpoint a 15.0 cm
Mostrar gráfica de Error y Setpoint	Muestra posición, error, y salida PID
Leer datos seriales	Recibe datos para graficar

Cuadro 2: Interacción Python-Arduino

6. Código Fuente Arduino

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
#include <PID_v1.h>
#include <NewPing.h>

// --- CONFIGURACI N DE COMPONENTES ---
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
#define SERVO_CHANNEL 0
#define SERVO_FREQ 50

#define SERVO_MIN_PULSE 150
#define SERVO_MAX_PULSE 550
#define SERVO_NEUTRAL_PULSE 350

#define TRIGGER_PIN 8
#define ECHO_PIN 9
#define MAX_DISTANCE_CM 35
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE_CM);

// --- CONTROL PID ---
#define TOLERANCE_CM 1.0

double Kp = 25.0;
double Ki = 5.0;
double Kd = 7.0;

double Setpoint;
double Input;
double Output;

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
```

```
32
33 int intervalo_envio_ms = 40; // se puede cambiar desde Python
34
35 void setup() {
36   Serial.begin(9600);
37   pwm.begin();
38   pwm.setPWMFreq(SERVO_FREQ);
39   delay(10);
40   pinMode(TRIGGER_PIN, OUTPUT);
41   pinMode(ECHO_PIN, INPUT);
42
43   pwm.setPWM(SERVO_CHANNEL, 0, SERVO_NEUTRAL_PULSE);
44   delay(1000);
45
46   Setpoint = 15.0;
47   myPID.SetMode(AUTOMATIC);
48   myPID.SetOutputLimits(0, 255);
49   myPID.SetSampleTime(intervalo_envio_ms);
50
51   Serial.println("Sistema iniciado");
52 }
53
54 void loop() {
55   revisarSerial();
56
57   Input = sonar.ping_cm();
58   if (Input == 0) Input = Setpoint;
59
60   double error = abs(Input - Setpoint);
61
62   if (error <= TOLERANCE_CM) {
63     pwm.setPWM(SERVO_CHANNEL, 0, SERVO_NEUTRAL_PULSE);
64   } else {
65     myPID.Compute();
66     int servoPulse = map(Output, 0, 255, SERVO_MIN_PULSE,
67                          SERVO_MAX_PULSE);
68     pwm.setPWM(SERVO_CHANNEL, 0, servoPulse);
69   }
70
71   // Enviar datos a Python para graficar
72   Serial.print("Posici n: "); Serial.print(Input);
73   Serial.print(" cm\tError: "); Serial.print(Setpoint - Input);
74   Serial.print(" cm\tOutput: "); Serial.println(Output);
75
76   delay(intervalo_envio_ms);
77 }
78
79 // === PROCESAR COMANDOS DESDE PYTHON ===
80 void revisarSerial() {
81   if (Serial.available()) {
82     String comando = Serial.readStringUntil('\n');
83     comando.trim();
84
85     if (comando.startsWith("S")) {
86       float nuevoSetpoint = comando.substring(1).toFloat();
87       if (nuevoSetpoint >= 5 && nuevoSetpoint <= 30) {
88         Setpoint = nuevoSetpoint;
89         Serial.print("Nuevo setpoint: ");
```

```
90          Serial.println(Setpoint);
91        }
92      } else if (comando.startsWith("F")) {
93        int nuevoDelay = comando.substring(1).toInt();
94        if (nuevoDelay >= 20 && nuevoDelay <= 200) {
95          intervalo_envio_ms = nuevoDelay;
96          myPID.SetSampleTime(nuevoDelay);
97          Serial.print("Nuevo intervalo de muestreo: ");
98          Serial.print(intervalo_envio_ms);
99          Serial.println(" ms");
100       }
101     }
102   }
103 }
```

7. Solución de Problemas

7.1. Problemas Comunes y Soluciones

7.1.1. El sensor ultrasónico no mide correctamente

- Verificar las conexiones VCC, GND, TRIGGER y ECHO
- Asegurar que no hay obstáculos en el rango de medición
- Comprobar que la bola esté dentro del rango del sensor (2-35 cm)

7.1.2. El servomotor no se mueve

- Verificar la alimentación del PCA9685 y del servomotor
- Comprobar las conexiones I2C (SDA y SCL)
- Verificar que el canal del servo esté correctamente configurado

7.1.3. No hay comunicación serial

- Verificar que el puerto COM esté correctamente seleccionado
- Comprobar la velocidad de baudios (9600)
- Reiniciar el Arduino y volver a conectar

7.1.4. El sistema es inestable

- Ajustar los parámetros PID (Kp, Ki, Kd)
- Verificar que la viga esté bien balanceada
- Comprobar que no hay vibraciones externas

7.2. Parámetros Recomendados

Parámetro	Valor Recomendado
Setpoint inicial	15.0 cm
Rango de setpoint	5.0 - 30.0 cm
Kp (Proporcional)	25.0
Ki (Integral)	5.0
Kd (Derivativo)	7.0
Tolerancia	1.0 cm
Intervalo de muestreo	40 ms

Cuadro 3: Parámetros recomendados del sistema

8. Repositorio y Recursos

8.1. Repositorio GitHub

Todos los archivos del proyecto, incluyendo archivos 3D, código Arduino, aplicación Python y este manual de usuario, se encuentran disponibles en:
https://github.com/ToxquiM/Ball-And-Beam-Arduino.git

8.2. Contenido del Repositorio

- Código fuente de Arduino (.ino)
- Aplicación Python para monitoreo y control
- Archivos STL para impresión 3D de las piezas
- Diagramas de conexión
- Manual de usuario (este documento)
- Documentación técnica adicional

8.3. Recursos Adicionales

- Datasheet del sensor HC-SR04
- Datasheet del servomotor MG995
- Documentación de la librería PID de Arduino
- Tutoriales de control PID

9. Anexos

9.1. Anexo A: Especificaciones Técnicas

9.1.1. Sensor Ultrasónico HC-SR04

- Voltaje de operación: 5V DC
- Corriente de trabajo: 15mA
- Frecuencia de trabajo: 40KHz
- Rango máximo: 4m
- Rango mínimo: 2cm
- Ángulo de medición: 15°
- Pulso de trigger: 10µS TTL
- Dimensiones: 45mm x 20mm x 15mm

9.1.2. Servomotor MG995

- Voltaje de operación: 4.8V - 7.2V
- Velocidad (4.8V): 0.20 sec/60°
- Velocidad (6.0V): 0.17 sec/60°
- Torque (4.8V): 10.5 kg/cm
- Torque (6.0V): 13 kg/cm
- Peso: 55g
- Dimensiones: 40.7mm x 19.7mm x 42.9mm

9.2. Anexo B: Comandos de Comunicación Serial

Comando	**Formato**	**Descripción**
Cambiar setpoint	S[valor]	15.0 cambia setpoint a 15.0 cm

Cuadro 4: Comandos disponibles por comunicación serial

9.3. Anexo C: Licencia

Puedes usar, modificar y distribuir el código libremente, manteniendo la atribución a los autores originales.