

Balancín 1D. Versión 2

Control en Tiempo Discreto

Román Garnica Cortés

8170

IECSA08

Academia de Electrónica
Universidad Aeronáutica en Querétaro
México
19 de Abril, 2022

Índice

Índice de figuras	2
1. Objetivo	3
2. Introducción	3
3. Marco teórico	3
3.1. Físicas del sistema	3
3.2. Control PID	5
3.2.1. Control Proporcional	5
3.2.2. Control Derivativo	5
3.2.3. Control Integral	5
3.2.4. Ecuación del Controlador	5
3.3. Servomotor	6
3.3.1. Servo de rotación posicional	6
3.4. Sensor de distancia	6
3.4.1. Sensor de distancia infrarrojo	7
4. Desarrollo	8
4.1. Calendario de actividades	8
4.2. Materiales	8
4.3. Diseño de la estructura	10
4.3.1. Diseño 1	10
4.3.2. Diseño 2	10
4.3.3. Diseño Final	11
4.4. Diagramas de cuerpo libre	11
4.5. Adquisición de señales	12
4.5.1. Retroalimentación	12
4.5.2. Señal de entrada	13
4.6. Controlador	13
4.7. Código	14
4.7.1. Código para el sensor	14
4.7.2. Código para la referencia	15
4.7.3. Código para el control	15
5. Resultados	17
6. Conclusiones	19
7. Trabajos futuros	19
8. Sugerencias	19
8.1. Para mí	19
8.2. Para el profesor	19
9. Anexos	20
9.1. Código	20
Bibliografía	25

Índice de figuras

1.	Diagrama del sistema	4
2.	Servomotor MicroSG90	6
3.	Sensor SHARP	7
4.	Calendario de actividades hecho para el proyecto	8
5.	Hercules RM42	9
6.	Primer diseño de la maqueta	10
7.	Segundo diseño de la maqueta	10
8.	Diseño final del proyecto	11
9.	Diagrama de cuerpo libre del sistema a 0 grados	11
10.	Diagrama de cuerpo libre del sistema a 45°	12
11.	Diagrama de cuerpo libre del sistema a -45°	12
12.	Filtro analógico aplicado al sensor para limpiar la señal	13
13.	Señal de entrada del sistema con POT de 1KΩ	13
14.	Diagrama del controlador final con ganancias	14
15.	Gráfica del sistema sin ganancias, se ve en la señal cómo la pelota parte de la referencia y se aleja hasta un extremo	17
16.	Gráfica del sistema con control proporcional, se ve en la señal cómo la pelota osciló sobre la referencia, quiere decir que ya es posible estabilizar el sistema agregando control derivativo	17
17.	Gráfica del sistema con control PD, en la señal se ve cómo se estabilizó el sistema pero no se alcanzó un error igual a cero, pero sí un error cercano.	18
18.	Gráfica del sistema funcionando con control PID, aquí el sistema se estabilizó en 0 o al menos muy cerca de 0.	18

1. Objetivo

Diseñar un sistema que controle la posición de una pelota dentro de un balancín con ayuda de un control PID.

2. Introducción

Este proyecto es una versión un poco más complicada de un sistema ball & beam regular. El objetivo es prácticamente el mismo, lograr que una pelota se quede en la posición que nosotros le digamos, el problema es que la pelota nunca va a estar quieta, pues el balancín no es recto, es un arco. Esta clase de proyectos servirán como introducción al control discreto para después aplicarlas a otras áreas de la ingeniería como la robótica, la automatización, robótica, etc.

El hecho que sea un arco complica mucho las cosas, pues el sistema difícilmente estará en un estado estacionario en el que nuestro control deje de reaccionar, todo el tiempo estará moviendo el balancín porque la pelota estará en movimiento todo el tiempo.

3. Marco teórico

3.1. Físicas del sistema

Para obtener las físicas del sistema se tienen que tomar en cuenta todas las variables involucradas en el sistema. Estas variables son las siguientes:

Símbolo	Descripción	Valores
m	Masa de la pelota	0.11kg
R	Radio de la pelota	40mm
d	Brazo del servo	75mm
g	Gravedad	$9.8 \frac{m}{s^2}$
L	Longitud del balancín	300mm
J_b	Momento de inercia de la pelota	$\frac{2mR^2}{5Kgm^2}$
r	Posición de la pelota	
α	Ángulo del balancín	
θ	Ángulo del servo	

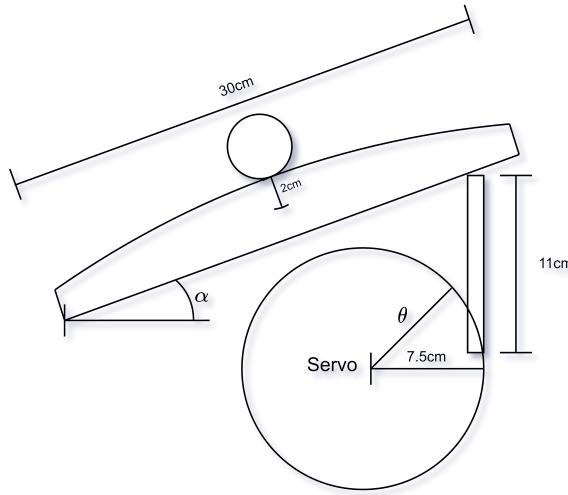


Figura 1: Diagrama del sistema

La dinámica de la pelota está sujeta a la gravedad, a fuerzas iniciales y centrífugas. La ecuación Lagrangiana simplificada para el movimiento de la pelota es:

$$\left(m + \frac{J_b}{R^2}\right)\ddot{r} + \frac{J_b}{R\ddot{\alpha}} - mr\dot{\alpha}^2 + mg\sin\alpha = 0 \quad (1)$$

Linealizando (1) sobre el ángulo del balancín, $\sin\alpha = \alpha$ para α pequeños, nos da la siguiente aproximación lineal del sistema

$$\left(m + \frac{J_b}{R^2}\right)\ddot{r} = mg\alpha \quad (2)$$

Al igualar la distancia del arco, la ecuación que relaciona el ángulo del balancín con el ángulo del servo se puede aproximar como lineal y se escribe como

$$\alpha = \left(\frac{d}{L}\right)\theta \quad (3)$$

Sustituyendo (3) en (2) obtenemos

$$\left(m + \frac{J_b}{R^2}\right)\ddot{r} = mg\left(\frac{d}{L}\right)\theta \quad (4)$$

Tomando la transformada de Laplace de (4), la ecuación en términos de entrada y salida es

$$\left(m + \frac{J_b}{R^2}\right)s^2r(s) = mg\left(\frac{d}{L}\right)\theta \quad (5)$$

Reordenando la ecuación, la función de transferencia de la posición de la pelota ($r(s)$) respecto al ángulo del balancín ($\theta(s)$) es representada como

$$\frac{r(s)}{\theta(s)} = \frac{mgd}{L\left(m + \frac{J_b}{R^2}\right)s^2} \quad (6)$$

3.2. Control PID

Un controlador PID es un dispositivo que permite controlar un sistema en lazo cerrado para que alcance un estado deseado. Está compuesto por tres elementos que proporcionan una acción proporcional, integral y derivativa.

Para que el sistema se estabilice necesitaremos una señal de referencia y una señal de error. La señal de referencia indica el estado que se desea conseguir y la señal de error indicará la diferencia que existe entre la referencia y el estado real del sistema.

3.2.1. Control Proporcional

La acción proporcional multiplica la señal de error por una constante K_p que determina la cantidad de acción proporcional que tendrá el controlador.

Modificar la ganancia proporcional tiene los siguientes efectos:

- Aumenta la velocidad de respuesta del sistema
- Disminuye el error del sistema
- Aumenta la inestabilidad del sistema

3.2.2. Control Derivativo

La acción derivativa será la encargada de ajustar la velocidad a la que el sistema se acerca al error. Cuando el sistema se mueve a altas velocidades, el sistema se puede pasarse por la inercia, para controlar esto, el control derivativo debe reconocer la velocidad y frenarla antes de que se acerque a la referencia deseada.

Aumentar la ganancia derivativa tiene los siguientes efectos:

- Aumenta la estabilidad del sistema controlado
- Disminuye la velocidad del sistema

Un problema que presenta el control derivativo es que amplifica las señales que varían rápidamente, como el ruido. Es deseable aplicar filtros para que esto no pase y el sistema se estabilice con éxito.

3.2.3. Control Integral

La acción integral es la encargada de acumular la señal de error, con esto se consigue reducir el error del sistema en el régimen permanente. La desventaja es que hace más inestable al sistema.

Aumentar la ganancia integral tiene los siguientes efectos:

- Disminuye el error en régimen permanente del sistema
- Aumenta la inestabilidad del sistema
- Aumenta un poco la velocidad del sistema

3.2.4. Ecuación del Controlador

La ecuación del controlador se puede obtener con la siguiente fórmula:

$$c(t) = K_p \cdot e(t) + K_i \int e(t) dt + K_d \cdot \frac{\delta e(t)}{\delta t}$$

3.3. Servomotor

Un servomotor es un actuador rotativo que permite un control preciso en términos de posición angular, aceleración y velocidad. La señal de control es la entrada, ya sea analógica o digital.

Estos circuitos utilizan un encoder para la retroalimentación de posición. La posición final se informa al controlador y se compara con la entrada de posición inicial.

Los servos están compuestos por tres componentes:

- Motor: Pueden ser CA o CC, los CA funcionan mejor para aplicaciones de posición y velocidad constante y los CC para aplicaciones de velocidad variable.
- Amplificador: Amplifica la señal del controlador para proporcionar potencia suficiente al motor.
- Mecanismo de retroalimentación: Mecanismo que analiza la velocidad, aceleración y posición del motor para que se ejecute el movimiento deseado por el sistema.

3.3.1. Servo de rotación posicional

Para este proyecto se usará este tipo de servo. Gira 180° y cuenta con topes físicos en el mecanismo para evitar que se sobrepase de la posición deseada.

Se usará un Micro Sg90. Tiene las siguientes especificaciones:

- Torque: 1.6KG/cm
- Velocidad de reacción: 0.12-0.13s/60°
- Temperatura de operación: -30°C a 60°C
- Ángulo máximo: 180°C
- Voltaje de operación: 3.5V a 6V



Figura 2: Servomotor MicroSG90

3.4. Sensor de distancia

Un sensor de distancia es un dispositivo que nos permite medir el espacio entre un objeto o una superficie y nuestro sensor. Este tipo de sensores también se pueden configurar como sensores de presencia y movimiento.

3.4.1. Sensor de distancia infrarrojo

Estos sensores funcionan emitiendo ondas infrarrojas y calculando el ángulo de reflexión, a esto se le llama principio de triangulación. Este tipo de sensores se utiliza en aparatos electrónicos en los que se pueda usar un control, como una TV o un proyector, en sistemas de seguridad como alarmas y también para monitoreo de aplicaciones que requieran control.

Una de sus mayores ventajas es que se puede usar con o sin luz en el espacio en el que está, lo que lo hace apto para aplicaciones nocturnas, también pueden medir distancia de objetos con superficies más complejas a diferencia de los sensores ultrasónicos.

Para este proyecto se utilizará uno de estos sensores. Es el sensor Sharp Gp2y0a21. Sus características son las siguientes:

- Distancia de detección: 10cm a 80cm
- Voltaje de alimentación: -0.3V a +7V
- Voltaje de salida: -0.3V a Vcc+0.3V
- Temperatura de operación: -10°C a +60°C
- Voltaje de alimentación recomendado: 4.5V a 5.5V



Figura 3: Sensor SHARP

4. Desarrollo

4.1. Calendario de actividades

TITULO: Balancín 1D												
Actividades	semana 1	semana 2	semana 3	semana 4	semana 5	semana 6	semana 7	semana 8	semana 9	semana 10	semana 11	Entregable
Elegir proyecto y hacer cronograma												Cronograma
Investigación sobre lo necesario para llevar a cabo el proyecto (Materiales, características de componentes, control, linealización)												Parte del reporte: Portada, marco teórico y bibliografía
Diseñar estructura												Plano de la estructura final
Conseguir materiales necesarios												Materiales
Prueba de sensores y motores												Excel con rango de salidas del sensor.
Ajuste de sensor												Agregar al excel las salidas del sensor linealizado
Armar maqueta												Maqueta armada
Escribir lista de requerimientos del programa												Lista de requerimientos
Diagrama de flujo del código												Diagrama de flujo
Código base												Semana 7: Código donde obtengamos información del sensor y poder establecer la referencia con el POT. Semana
Obtención del modelo matemático del sistema												Modelo matemático del sistema
Código dos												Código que mueva el servo con la distancia del sensor
Código completo												Proyecto funcionando por completo.
Reporte												Subir reporte de proyecto final realizado en Latex.

Figura 4: Calendario de actividades hecho para el proyecto

Se creó este calendario para desarrollar el proyecto por pasos, incluyendo tareas de investigación, programación, documentación, construcción de la base, entre otras. El tiempo está dividido en semanas y ayudó a dar un mejor seguimiento a cada una de las fases.

4.2. Materiales

- Hercules RM42

Placa de desarrollo de Texas Instruments basada en la arquitectura ARM Cortex-R4F y diseñado para aplicaciones de seguridad crítica en sistemas de control industrial.

Esta placa de desarrollo cuenta con un puerto de depuración integrado y es compatible con el software de desarrollo TI como Code Composer. También cuenta con una interfaz de depuración a través de una conexión USB para facilitar la programación.

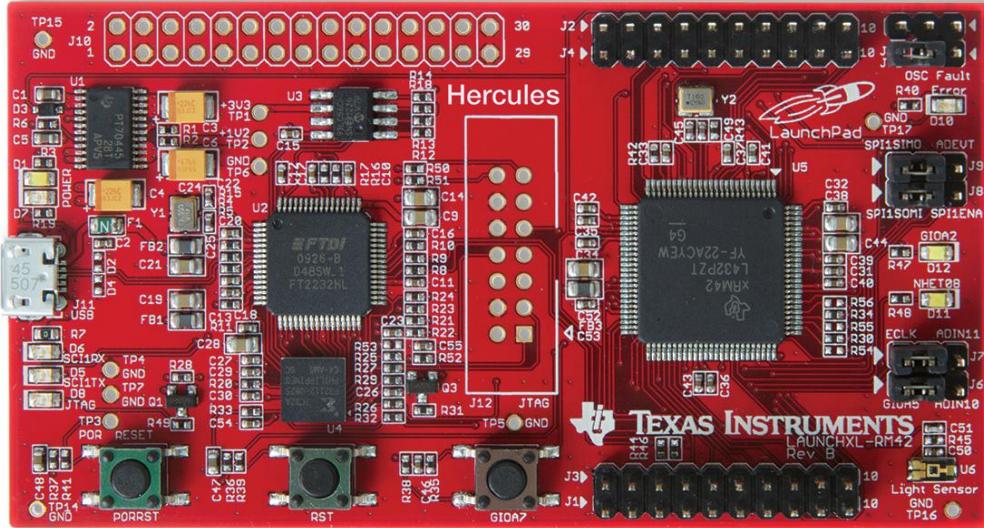


Figura 5: Hercules RM42

■ Componentes electrónicos

- Servomotor SG90
 - Sensor SHARP GP2Y0A21YK
 - Potenciómetro $1\text{k}\Omega$
 - Capacitor cerámico 220nF
 - Resistencia $5.6\text{k}\Omega$
 - Protoboard
 - Cable

■ Estructura

- Cartón
 - Palos de plástico
 - Articulaciones impresas
 - Tornillos 6mm
 - Cabeza para tornillo 6mm
 - Base de madera

Estos materiales fueron seleccionados para usar el menor presupuesto posible y hacer la estructura muy ligera para poder cargarla con un servo tan básico. No hay un número para el presupuesto final, pues varias de estas cosas ya las tenía en casa.

Lo más caro de todo fue la impresión 3D. Un estimado para todo el proyecto en general sería menos de \$500 sin contar el launchpad y la protoboard.

4.3. Diseño de la estructura

Para el diseño se pidió una superficie con treinta centímetros de largo con un arco de altura de 2cm. El arco debe estar recargado en una articulación que le permita girar fácilmente con el empuje del servo. El sensor óptico estará ubicado en uno de los costados del arco para medir la distancia de la pelota fácilmente.

El modelo fue hecho en tinkercad, que es un software de diseño muy básico para novatos en diseño 3D.

4.3.1. Diseño 1

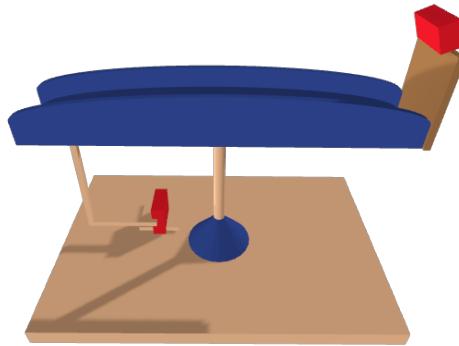


Figura 6: Primer diseño de la maqueta

Este diseño ya no fue usado porque se necesitaba más libertad de movimiento para el puente y la posición en medio de la maqueta no era la óptima para esto.

4.3.2. Diseño 2

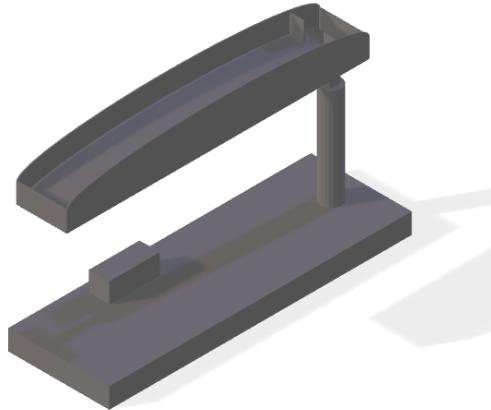


Figura 7: Segundo diseño de la maqueta

En este diseño se cambio la posición del poste para el puente para que así tenga más libertad de movimiento. Igual se acortó el tamaño de la base para reducir costos. Este modelo no fue usado porque

la articulación no permitía que el puente se moviera en un buen rango.

4.3.3. Diseño Final

Para el diseño final, se modelaron tres piezas para que funcionen como articulaciones. Se elevó más el balancín con ayuda de los palos de plástico, esto para que tuviera más grados de movimiento ya que no era posible mover la pelota en la posición anterior.

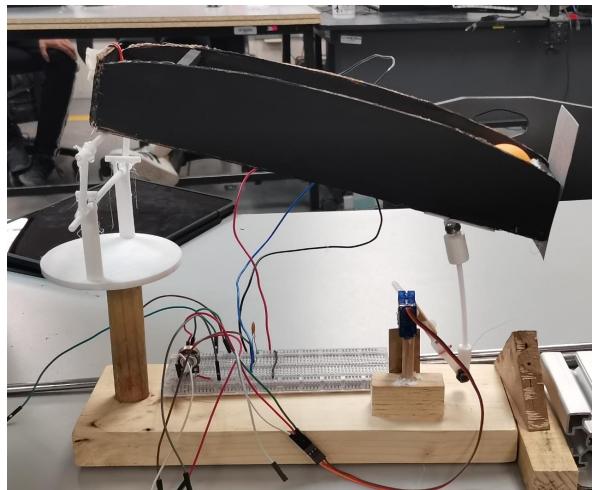


Figura 8: Diseño final del proyecto

4.4. Diagramas de cuerpo libre

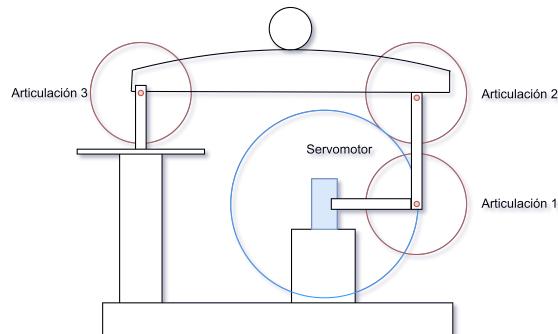


Figura 9: Diagrama de cuerpo libre del sistema a 0 grados

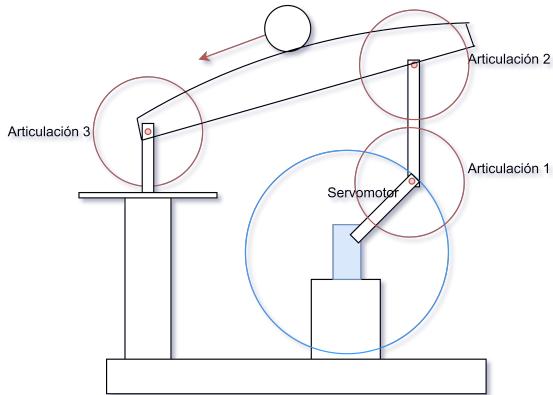


Figura 10: Diagrama de cuerpo libre del sistema a 45°

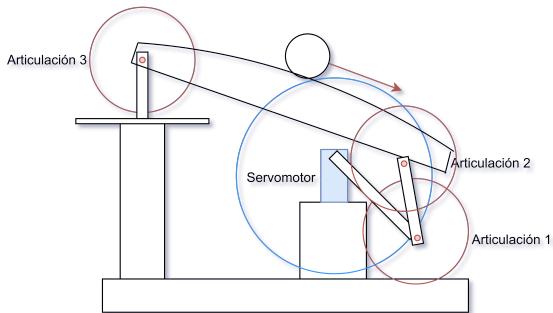


Figura 11: Diagrama de cuerpo libre del sistema a -45°

En los diagramas mostrados en las figuras 7 y 8 se usan 45° y -45° porque esas serán las ganancias máximas que tendrá nuestro sistema para poder mover la pelota de forma exitosa sin forzar las articulaciones del sistema ni el rango del servo.

4.5. Adquisición de señales

Para el proyecto se necesitaban adquirir dos señales. Una para la posición del elemento a controlar (Esta señal funciona como retroalimentación) y otra para la referencia (Esta señal funciona como entrada del sistema).

4.5.1. Retroalimentación

La señal de retroalimentación se adquirió con el sensor SHARP ya mencionado antes. Este sensor se conectó a una fuente de 5V. La salida del sensor se conectó a un filtro analógico RC con la resistencia de $5.6\text{k}\Omega$ y el capacitor de 220nF .

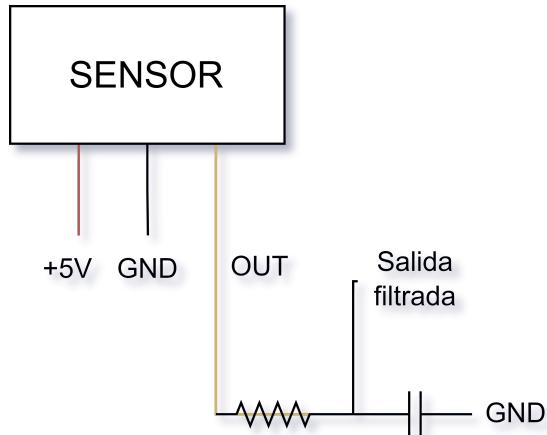


Figura 12: Filtro analógico aplicado al sensor para limpiar la señal

Este es un filtro pasivo pasa bajas que funciona con la señal que sale del sensor. La salida se limpia gracias al capacitor, que guarda el voltaje de salida y limpia los picos o las caídas de voltaje que pueda haber.

Después de aplicar este filtro a mi sensor, se limpió la señal, pero no por completo, aún después se tuvo que volver a filtrar para evitar el ruido por completo.

La salida del filtro se conectó al AD0 de la Hércules.

4.5.2. Señal de entrada

Para la señal de entrada se usó un resistor variable de $1k\Omega$ conectado a una fuente de 3V y su salida se conectó al AD1 de la Hércules. No fue necesario filtrar el POT, pues su parte no lineal no afectaba realmente al sistema.

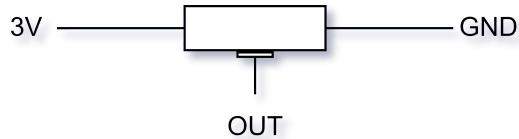


Figura 13: Señal de entrada del sistema con POT de $1K\Omega$

4.6. Controlador

Se implementó un control PID para poder equilibrar la pelota en donde esté la referencia. Aunque el sistema era aceptable solo con control PD, se le agrego la parte integral para que este fuera más preciso, aunque también complicó encontrar las ganancias necesarias para el sistema.

Para adquirir las ganancias adecuadas para el sistema, primero se buscó una ganancia proporcional con el que el sistema reaccionara lo suficientemente para que oscilara cerca de la referencia. Nuestra ganancia proporcional fue de 2.6, o sea que nuestro error sería multiplicado por la ganancia para amplificar el error. Las ganancias fueron relativamente bajas en comparación con otros proyectos ya que el error del sistema es milimétrico.

La siguiente ganancia a encontrar fue la derivativa, que es la que hace al sistema reaccionar a la velocidad de la pelota. Esta fue la más difícil de encontrar ya que si era muy alta, el sistema se hacia

inestable porque la pelota, al ser muy ligera, comenzaba a rebotar sobre el balancín, por lo que tuve que variar por centésimas este número hasta encontrar la más adecuada. La ganancia final fue de 0.23.

Por último la ganancia integral, que fue la más fácil, pues solo buscaba una que reaccionara lo suficiente para mover la pelota en caso de que no estuviera en la referencia. Esta tenía que ser baja, pues entre más alta era, más lento era el sistema. La ganancia final fue de 0.06.

El controlador final quedó de la siguiente manera:

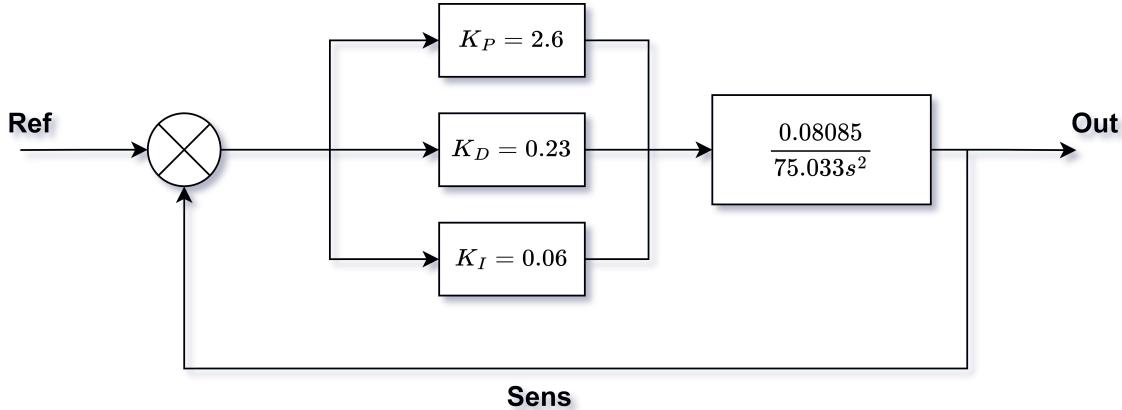


Figura 14: Diagrama del controlador final con ganancias

4.7. Código

En esta sección explicaré cada una de las partes importantes del código, para evitar poner las líneas de comentarios de Code Composer, el código completo está en la sección de Anexos.

4.7.1. Código para el sensor

```

adc_sen[0]=adc_sen[1];
adc_sen[1]=adc_sen[2];
adc_sen[2] = adc_data[0].value;

for(i=0;i<3;i++){
    med[i]=adc_sen[i];
}

while(med[0]>med[1] || med[1]>med[2]){
    if(med[0]>med[1]){
        sto=med[0];
        med[0]=med[1];
        med[1]=sto;
    }
    else{
        sto=med[1];
        med[1]=med[2];
        med[2]=sto;
    }
}

sens=ema_filter((float)med[1]);

```

```

sens_mv=((int32_t)sens*3100)/4096;
mvoltos=(double)sens_mv;
sens_mm=(int)(10*pow((147737/(mvoltos*10)),1.2134));
if(sens_mm>230){
    sens_mm=230;
}

```

Para el sensor fue necesario obtener tres datos y sacar la mediana de estos para filtrar la señal. Después de esto se aplicó un filtro EMA (Exponential Moving Average), es un filtro que suaviza la señal con ayuda de una variable (ALPHA), que es el tamaño del suavizado que requerimos, que va de 0 a 1.

La función del filtro es la siguiente:

```

float ema_filter(float a)
{
    float result;

    result = ALPHA * a + (1-ALPHA)*previous_val;
    previous_val=a;
    return result;
}

```

Posteriormente se usó la siguiente ecuación del manual del sensor para linealizarlo y obtener el valor de la distancia medida en milímetros.

$$Distancia(mm) = 10 * \left(\frac{147737}{VoltajeDeSalida(mV) * 10} \right)^{1.2134}$$

4.7.2. Código para la referencia

```
POT_mm=50+((181*adc_data[1].value)/4096);
```

A la función para obtener el valor en milímetros primero se le sumó 50 porque el balancín tenía un tope para la pelota a 5cm del sensor. Después de esto se dividió todo el rango del potenciómetro para que vaya hasta 230, que era el rango máximo que podía alcanzar la pelota por su diámetro de 40mm.

4.7.3. Código para el control

```

e[4]=e[3];
e[3]=e[2];
e[2]=e[1];
e[1]=e[0];

e[0]=(float)(POT_mm-sens_mm);

Up=kp*e[0];
Ui=Ui+ki*T*e[0];
Ud=(kd/T)*(e[0]-e[4]);

limitControl(-100,100,&Ui);

```

```
Uc=Up+Ui+Ud;  
limitControl(-400,400,&Uc);  
miPWM.duty=650+(int32_t)Uc;  
pwmSetSignalMIO(hetRAM1,pwm0,miPWM);
```

Para el control se inicia con un corrimiento de errores, este corrimiento nos servirá después para el control derivativo. El error lo obtendremos restando la señal del sensor a la referencia.

Después obtenemos los valores proporcionales, derivativos e integrales. Up multiplicará nuestro error por nuestra ganancia proporcional. Ui multiplicará el error por nuestra ganancia intergal y nuestro tiempo de muestreo, a esto se le sumará el Ui anterior para ir aumentando el valor integral hasta que nuestro error sea cero. Ud comparará el error más reciente con uno anterior para multiplicarlo por nuestra ganancia derivativa.

Posteriormente se aplican límites para el control integral y el derivativo. Al integral le aplicamos un control de 100, que nunca fue alcanzado pero solo era por si este valor llegaba a ser muy grande. El límite de Uc es el límite para el valor de la suma de nuestros valores anteriores, con este límite se estableció el rango de movimiento del servo, el 650 sumado al PWM.duty es donde el servo estaba centrado en el rango.

Esta es la función para limitar el control:

```
void limitControl(int32_t MIN, int32_t MAX, float *x){  
    float *aux=x;  
    if(*aux<MIN){*aux=MIN;}  
    if(*aux>MAX){*aux=MAX;}  
}
```

5. Resultados

Para validar el sistema, se llevaron a cabo pruebas moviendo la referencia a los extremos del balancín, se desestabilizó el sistema con perturbaciones, se hizo un recorrido de la pelota por todo el balancín moviendo la referencia poco a poco y se dejó el sistema funcionando por un periodo largo para ver si se mantenía controlado en todo ese tiempo. Todas estas pruebas fueron pasadas.

Se graficó el error del sistema sin control, después se agregó control proporcional, posteriormente derivativo y al final el control PID funcionando, controlando el sistema con éxito.

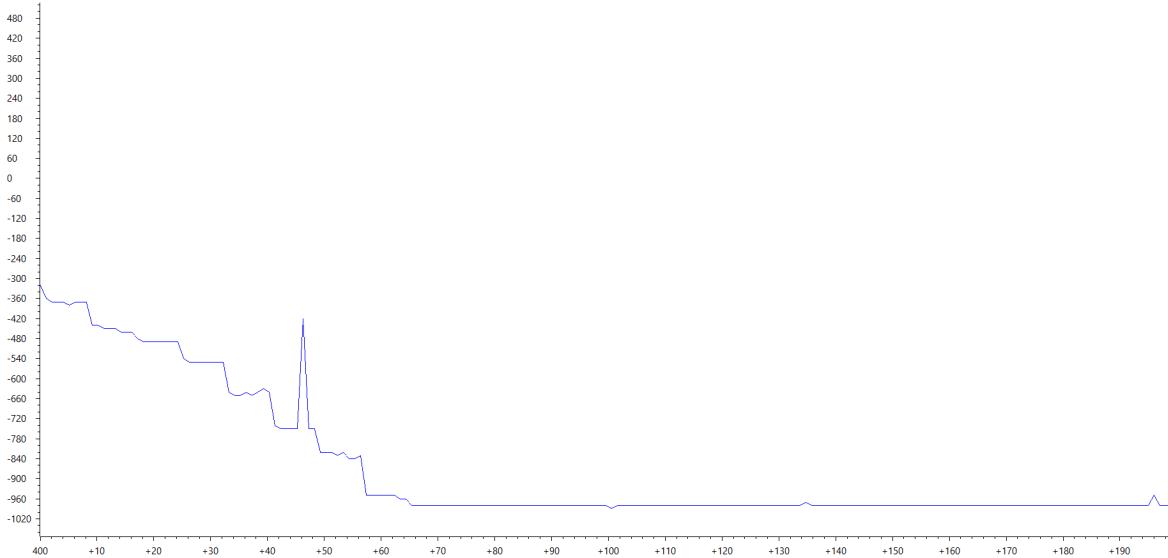


Figura 15: Gráfica del sistema sin ganancias, se ve en la señal cómo la pelota parte de la referencia y se aleja hasta un extremo

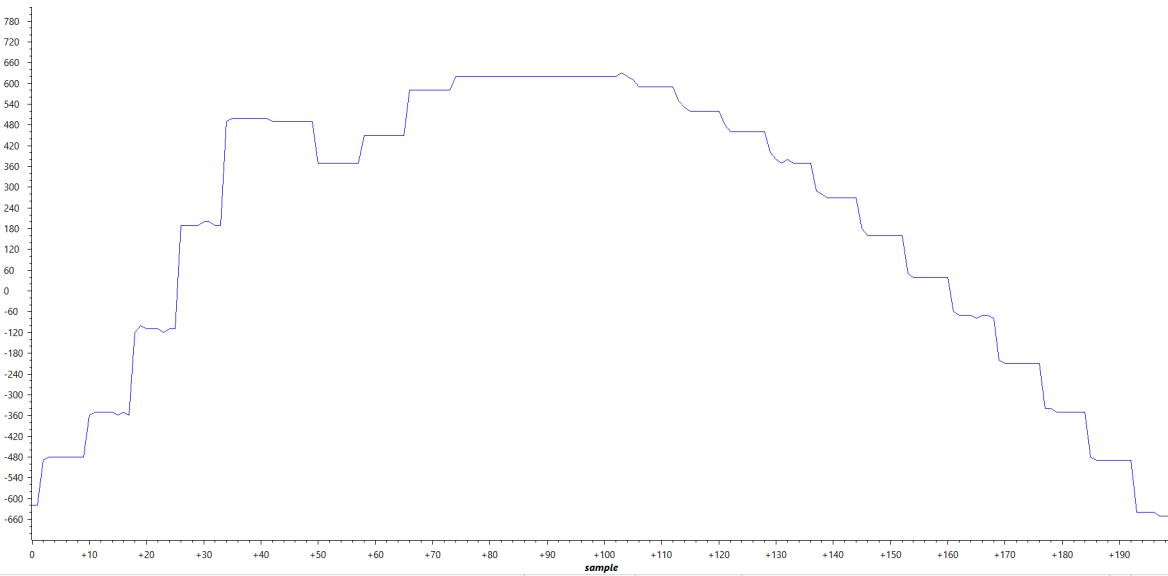


Figura 16: Gráfica del sistema con control proporcional, se ve en la señal cómo la pelota osciló sobre la referencia, quiere decir que ya es posible estabilizar el sistema agregando control derivativo

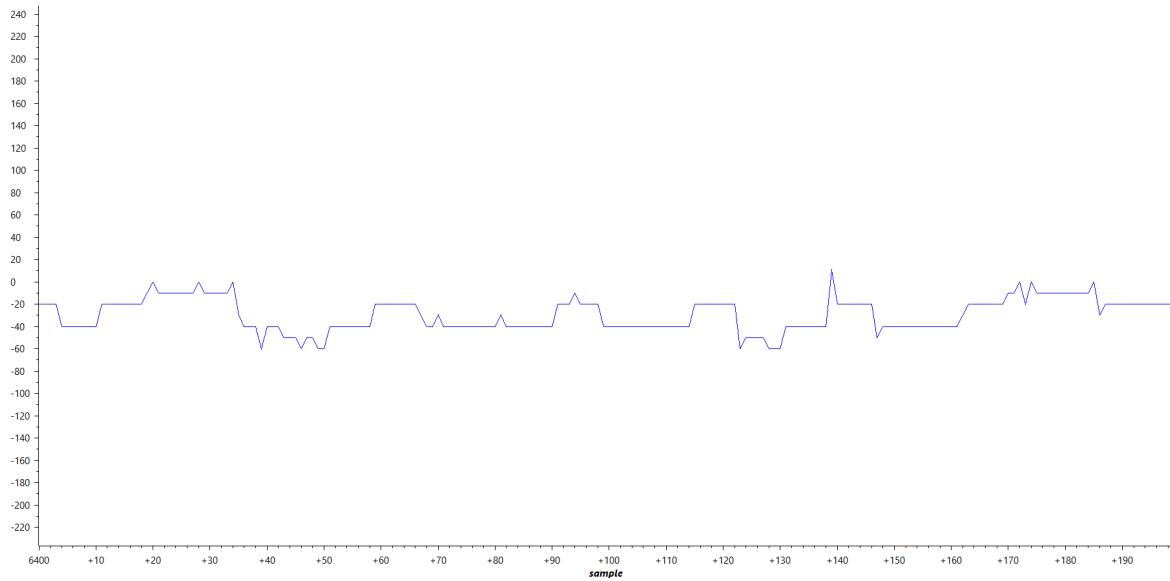


Figura 17: Gráfica del sistema con control PD, en la señal se ve cómo se estabilizó el sistema pero no se alcanzó un error igual a cero, pero sí un error cercano.

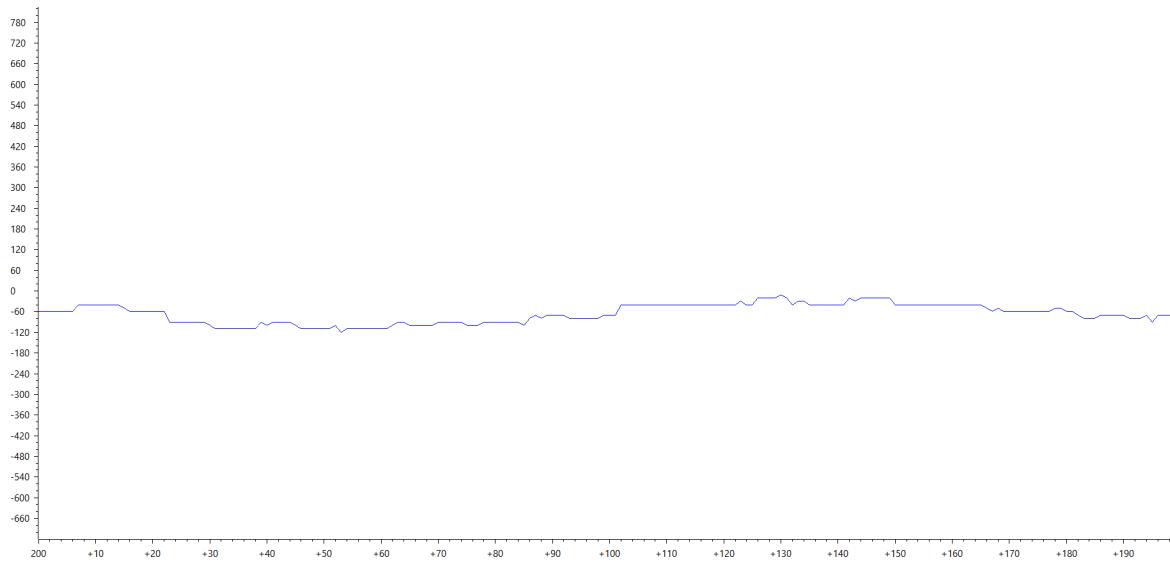


Figura 18: Gráfica del sistema funcionando con control PID, aquí el sistema se estabilizó en 0 o al menos muy cerca de 0.

En las gráficas mientras vamos agregando los elementos del control PID se observa cómo el sistema al final es bastante estable. Esta estabilidad se debe más que nada a que se logró limpiar casi por completo la señal del sensor, por lo que el sistema reaccionaba bastante bien a los movimientos de la pelota por estar sobre una superficie curva.

6. Conclusiones

Un sistema ball and beam puede ser considerado una introducción al control básico, pero por mi experiencia, el hecho de que sea curvo sí lo complica demasiado. En un sistema ball and beam clásico, es fácil llegar a un estado estable, pues sobre una barra recta, la pelota sí puede parar, pero en este caso hay que considerar más cosas, pues en primer lugar, entre más lejos está la pelota del centro, más inestable es, por lo que alcanzar la ganancia derivativa adecuada es todo un reto, pues tiene que frenar la pelota sin tanta fuerza como para hacer que se eleve. Otro reto fue la posición del sensor, pues si no estaba bien posicionado, la pelota se perdía pasando el arco o se iba perdiendo entre más cerca estaba del sensor.

Aún con lo anterior, este sistema sí me pareció adecuado para entender por completo como funcionan cada una de las ganancias del control PID y la importancia que tiene cada una de ellas. Este proyecto logra que haya una comprensión de los principios del control discreto, así como habilidades de programación en el launchpad usado y un poco de electrónica analógica con el filtro aplicado.

7. Trabajos futuros

Usando este trabajo como base, se puede complicar más si se hace el arco más pronunciado y en vez de limitarnos con el sensor infrarrojo, usar una cámara sobre este. Otra posibilidad sería agregar un grado más de libertad, para que también se pueda mover la posición de la pelota en un segundo eje sobre una base.

8. Sugerencias

8.1. Para mí

Este proyecto me dejó bastantes enseñanzas personales, que son aún más valiosas que el conocimiento sobre programación, electrónica, etc, que aprendí haciéndolo. Por hacer las cosas sin esforzarme como debería en un principio, me compliqué mucho más el proyecto al final, simplemente la estructura no era tan estable como requería, se rompía cuando el sistema se hacía muy inestable y tuve que hacer varios ajustes que a final de cuentas costaron el dinero y tiempo que no gasté al principio, por lo que ahora creo que seré más dedicado desde un principio en vez de preocuparme al final.

Otra cosa que me dejó fue ver que aún soy capaz de sacar un proyecto por mi cuenta, como todos los proyectos anteriormente eran en equipos y tengo un equipo muy capaz, realmente la carga de trabajo para mí no era tanto, por lo que nunca di mi máximo y reaccionaba solo cuando era necesario, pero ahora tuve que cargar con que cualquier error era mi responsabilidad y cualquier cosa que fallara la tenía que arreglar yo. Esta clase de cosas me hizo darme cuenta de que tal vez debería tomarme más en serio mis habilidades y en realmente ayudar a hacer las cosas en vez de ayudar solo cuando es necesario.

Fue una buena experiencia para mí hacerlo, me siento bastante conforme con mi desempeño y sobre todo con lo que aprendí.

8.2. Para el profesor

Creo que algo que nos hubiera servido a todo el grupo para que no lo saturáramos tanto al final, fue haber sugerido tomar asesorías para el proyecto, en las que todos los consejos que nos dio al final ya con el estrés y tiempo encima, nos los diera en un ambiente más tranquilo donde realmente escuchemos lo que está diciendo y no solo reaccionemos usando todo lo que podamos para sacar adelante nuestros proyectos.

Igual hubiera servido adelantar un poco más las prácticas que hicimos con la Hercules, sobre todo para poder avanzar con el código del proyecto sin depender de estar esperando a aprender a usar las herramientas que necesitamos.

9. Anexos

9.1. Código

```
/** @file sys_main.c
* @brief Application main file
* @date 11-Dec-2018
* @version 04.07.01
*
* This file contains an empty main function,
* which can be used for the application.
*/
/*
* Copyright (C) 2009-2018 Texas Instruments Incorporated - www.ti.com
*
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the
* distribution.
*
* Neither the name of Texas Instruments Incorporated nor the names of
* its contributors may be used to endorse or promote products derived
* from this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
* A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*/
*/
```

```

/* USER CODE BEGIN (0) */
#define Narr 200
/* USER CODE END */

/* Include Files */

#include "sys_common.h"

/* USER CODE BEGIN (1) */
#include "adc.h"
#include "rti.h"
#include "het.h"
#include <stdlib.h>
#include <math.h>
/* USER CODE END */

/** @fn void main(void)
 *  @brief Application main function
 *  @note This function is empty by default.
 *
 *  This function is called after startup.
 *  The user can use this function to implement the application.
 */

/* USER CODE BEGIN (2) */
adcData_t adc_data[2];
int32_t eh[Narr];

//VARIABLES GLOBALES
int32_t sens_mm=0; POT_mm=0,clmin=0,clmax=0,refe=0, j=0, k=0;
float e[5];
float kp=2.6,ki=0.06,kd=0.23,Uc=0,Ui=0,Ud=0,T=0.02,Up=0;

float ALPHA=1,previous_val=1;

hetSIGNAL_t miPWM;
static const uint32 s_het1pwmPolarity[8U] =
{
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
    3U,
};

void pwmSetSignalMIO(hetRAMBASE_t * hetRAM, uint32 pwm, hetSIGNAL_t signal);

float ema_filter(float a)
{

```

```

float result;

result = ALPHA * a + (1-ALPHA)*previous_val;
previous_val=a;
return result;
}

/* USER CODE END */

int main(void)
{
/* USER CODE BEGIN (3) */
    rtiInit();
    adcInit();
    hetInit();
    double mvoltis=0;
    int32_t ch_count;
    int32_t adc_sen[3]={0}, med[3],sto, sens_mv=0;
    uint32_t i;
    float sens=0;
    rtiEnableNotification(rtiNOTIFICATION_COMPARE0);
    rtiStartCounter(rtiCOUNTER_BLOCK0);
    _enable_interrupt_();
    while(1)
    {
        adcStartConversion(adcREG1, adcGROUP1);
        while((adcIsConversionComplete(adcREG1, adcGROUP1)==0));
        ch_count = adcGetData(adcREG1, adcGROUP1, &adc_data[0]);

        adc_sen[0]=adc_sen[1];
        adc_sen[1]=adc_sen[2];
        adc_sen[2] = adc_data[0].value;

        for(i=0;i<3;i++){
            med[i]=adc_sen[i];
        }

        while(med[0]>med[1] || med[1]>med[2]){//FUNCION PARA ORDENAR EL ARREGLO DE LA MEDIANA DE MENOS
            if(med[0]>med[1]){
                sto=med[0];
                med[0]=med[1];
                med[1]=sto;
            }
            else{
                sto=med[1];
                med[1]=med[2];
                med[2]=sto;
            }
        }
        sens=ema_filter((float)med[1]);
    }
}

```

```

sens_mv=((int32_t)sens*3100)/4096;
mvolts=(double)sens_mv;
sens_mm=(int)(10*pow((147737/(mvolts*10)),1.2134));//DISTANCIA EN MM DEL SENSOR
if(sens_mm>230){
    sens_mm=230;
}

POT_mm=50+((181*adc_data[1].value)/4096);
miPWM.period = 20000;

}
/* USER CODE END */

return 0;
}

/* USER CODE BEGIN (4) */
void rtiNotification(uint32 notification)
{
    e[4]=e[3];
    e[3]=e[2];
    e[2]=e[1];
    e[1]=e[0];

    e[0]=(float)(-sens_mm+POT_mm);

    Up=kp*e[0];
    Ui=Ui+ki*T*e[0];
    Ud=(kd/T)*(e[0]-e[4]);

    limitControl(-100,100,&Ui);
    Uc=Up+Ui+Ud;
    limitControl(-400,400,&Uc);
    miPWM.duty=650+(int32_t)Uc;
    pwmSetSignalMIO(hetRAM1,pwm0,miPWM);
}

void limitControl(int32_t MIN, int32_t MAX, float *x){
    float *aux=x;
    if(*aux<MIN){*aux=MIN;}
    if(*aux>MAX){*aux=MAX;}
}

void pwmSetSignalMIO(hetRAMBASE_t * hetRAM, uint32 pwm, hetSIGNAL_t signal)
{
    uint32 action;
    uint32 pwmPolarity = OU;
    float64 pwmPeriod = 0.0F;

    if(hetRAM == hetRAM1)

```

```
{  
    pwmPeriod = (signal.period * 1000.0F) / 640.000F;  
    pwmPolarity = s_het1pwmPolarity[pwm];  
}  
else  
{  
}  
if (signal.duty == 0U)  
{  
    action = (pwmPolarity == 3U) ? 0U : 2U;  
}  
else if (signal.duty >= 10000U)  
{  
    action = (pwmPolarity == 3U) ? 2U : 0U;  
}  
else  
{  
    action = pwmPolarity;  
}  
  
hetRAM->Instruction[(pwm << 1U) + 41U].Control = ((hetRAM->Instruction[(pwm << 1U) + 41U].Control  
hetRAM->Instruction[(pwm << 1U) + 41U].Data = (((uint32)pwmPeriod * signal.duty) / 10000U) << 7U  
hetRAM->Instruction[(pwm << 1U) + 42U].Data = ((uint32)pwmPeriod << 7U) - 128U;  
}  
/* USER CODE END */
```

Bibliografía

- [1] Types of Distance Sensors and How to Select One?, 6 2021. URL <https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/>.
- [2] Sensor de Distancia, 10 2022. URL <https://www.mecatronicalatam.com/es/tutoriales/sensores/sensor-de-distancia/>.
- [3] Aula21. Qué es un Servomotor y para qué sirve, 12 2022. URL <https://www.cursosaula21.com/que-es-un-servomotor/>.
- [4] H. S. Ingeniero. Sistemas de Control: Acciones Básicas (P, I, D, PD, PI, PID), 5 2020. URL <https://www.youtube.com/watch?v=G1UNm0SvMBo>.
- [5] C. Pardo. Controlador PID - Control Automático , 1 2023. URL <https://www.picuino.com/es/control-pid.html>.