Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Control de un sistema Bola Balancín

Autor:

Maestre Rodríguez, Héctor Manuel

Tutor:

García Ruiz, Francisco Javier

Departamento de Ingeniería de Sistemas y Automática

Valladolid, julio de 2019

Resumen

En el presente proyecto se desarrolla el control tanto teórico como real, de un sistema bola-balancín. Se entiende este sistema como el conjunto de una plataforma con una bola que puede rodar por ella y que puede alcanzar diferentes posiciones entre sus extremos.

Se va a diseñar un programa en Matlab-Simulink con un controlador PID que lea la posición de la bola en tiempo real y que varíe el ángulo de la plataforma mediante un sensor de distancia y un servomotor, respectivamente. Todo ello lo podremos visualizar en una LCD.

La comunicación de la planta con el ordenador se implementará en Arduino.

Palabras clave

PID, control, Ball & Beam, Simulink, Arduino

Abstract

In the present project we are going to develop the theoretical and real control of a ball and beam system. This system is understood as the set of a platform with a ball wich can roll through it and can reach different positions between its ends.

A program in Matlab-Simulink wiil be designed with a PID controller that will read the position of the ball in real time and will vary the angle of the platform by means of a distance sensor and a servomotor, respectively. All this we can see on a LCD.

The communication of the plant with the computer will be implemented in Arduino.

Keywords

PID, control, Ball & Beam, Simulink, Arduino

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

El control por realimentación tiene una larga historia que comenzó con el deseo primordial de los seres humanos de dominar los materiales y las fuerzas de la naturaleza en su provecho.

El reloj de agua, inventado por ktesibios de Alejandría (Egipto, 260 a.c.) fue posiblemente el primer sistema realimentado creado por el hombre. Su finalidad es mantener el nivel de agua de un tanque constante.

Para controlar el nivel se emplea un flotador o válvula flotante. Cuando el nivel del agua disminuye, dicha válvula favorece un aumento del caudal que alimentaba a este depósito, posibilitando la subida del nivel del mismo, y cuando el nivel de agua aumentaba, la válvula limita el caudal. Así se conseguía que el nivel del tanque permanezca constante. De esta forma se creó también el primer controlador, la Válvula de flotador.

Hoy en día, cualquier planta moderna tiene innumerables y sofisticados controladores para regular sus procesos, pero esto no sería posible sin los avances del pasado.

No podemos hablar de lo que conocemos actualmente como un regulador sin presentar el "regulador centrífugo de Watt", que tuvo un impacto determinante en la revolución industrial.

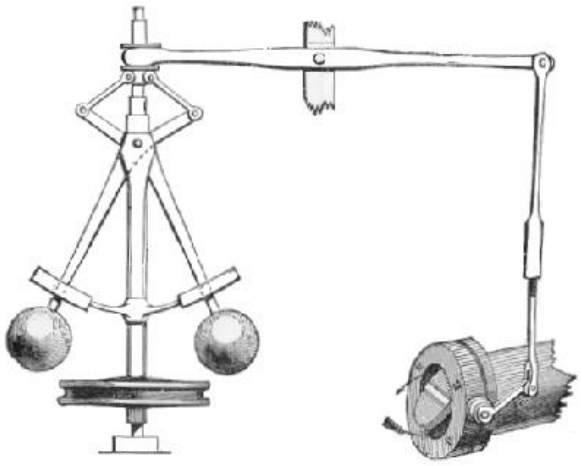


Figura 1. Regulador centrífugo de Watt

El control se hace muy importante a partir de la Revolución Industrial. En Europa supuso la aparición de las primeras máquinas que se movían solas: molinos, hornos, calderas, máquinas de vapor... Estos dispositivos no se podían controlar manualmente así que surgió la necesidad de nuevos dispositivos y de esta forma se desarrollaron los primeros reguladores de temperatura, presión, velocidad y nivel.

Así Watt combina la máquina de vapor con un regulador y patenta un prototipo adecuado para funcionar en entornos industriales, marcando esta fecha el comienzo de la Revolución industrial.

Watt no fue el primero en introducir reguladores con realimentación en Europa, en España existen patentes de Jerónimo de Ayanz con el molino de eje vertical que utilizaba la realimentación para mantener la orientación óptima.

Durante los años se han ido creando nuevas tecnologías que han ido ligadas a los mayores periodos de desarrollo de la sociedad, pero los fundamentos y conceptos principales no son diferentes a los que se pensaron hace dos milenios.

Objetivos

La finalidad es lograr un sistema que funcione de forma estable en estado estacionario, entendiendo esto como la situación en que la bola se encuentra en el centro de la plataforma en posición horizontal, o en su lugar, que logre estabilizarse en el valor indicado previamente por el usuario en la referencia. Esto se deberá realizar lo más rápido posible.

En el proceso estudiaremos los tipos de sistemas de segundo orden, la forma de controlarlos, el modelado y control teórico y la incidencia de los diferentes parámetros en un controlador PID en el sistema real.

En nuestra planta deberemos adaptar tanto el sensor para medir la distancia, el servomotor, como la LCD. Lo realizaremos mediante la interfaz de Simulink y la tarjeta Arduino. Los componentes los protegeremos en una caja que situaremos en la plataforma. Aprenderemos a configurar las herramientas con las que trabajaremos y a implementar un sistema real.

Nos enfrentaremos a los problemas ligados a trabajar con Hardware y trataremos de obtener un resultado, además de funcional, lo más estético y limpio posible.

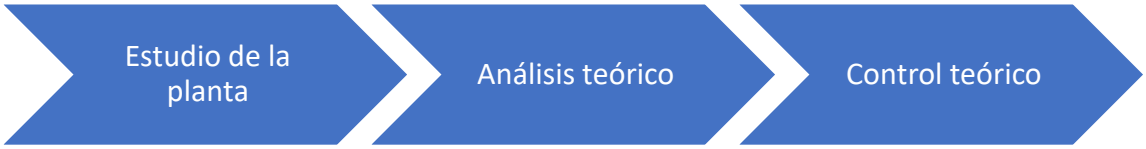
Alcance

El presente proyecto pretende derivar en una aplicación didáctica para alumnos de la escuela de ingenierías industriales de la Universidad de Valladolid y como apoyo para el departamento de ingeniería de sistemas y automática y sus laboratorios, para explicar de forma práctica conceptos como el control en lazo, función de transferencia o PID.

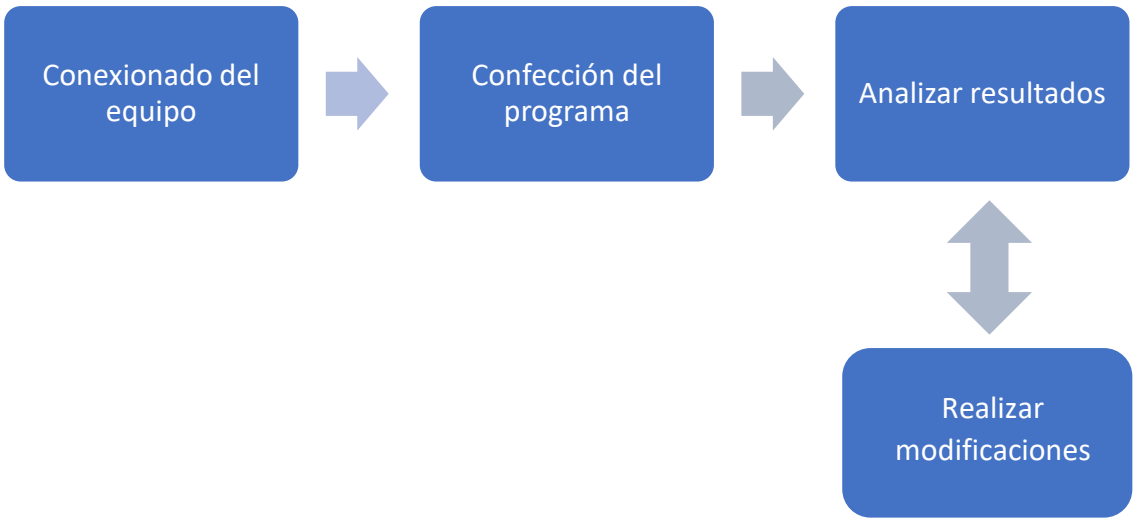
Por ello, el proyecto se va a desarrollar de una manera especialmente explicativa y visual, para que se puedan comprender todos sus conceptos y la metodología seguida.

Metodología

Podemos separar la metodología en tres etapas. Por un lado, el estudio, análisis y control teórico del sistema.



Por otro lado, la construcción y conexionado de los componentes para, a continuación, la realización de la aplicación del sistema real, analizar los resultados y la iteración de pruebas y de los parámetros para obtener el resultado deseado.



A lo que hay que añadir una última etapa de modificaciones estéticas, confección de la caja envolvente y puesta a punto para que el sistema funcione, se pueda transportar, quede protegido y sea lo más visual tanto para el futuro alumno como para un técnico de laboratorio.

Capítulo 2
Estudio de la planta

Lo primero, para realizar el proyecto, es estudiar y analizar el diseño de la planta para conocer a lo que nos enfrentamos.

Figura 2. Planta del sistema Bola-Balancín ya construido y con todo implementado

Nuestro sistema se compone de una viga de aluminio reforzada con un armazón metálico con dos soportes.

Un soporte fijo que va a permanecer inmóvil y va a impedir el movimiento de traslación de la viga en sentido vertical, pero permite el giro en torno a él. En el otro extremo un soporte móvil que va a permitir la traslación de la viga en sentido vertical y que va a provocar la inclinación de esta para la correspondiente rodadura de la bola.

Estos soportes se han realizado con una impresora 3D que garantiza una buena sujeción y fiabilidad, además de un bajo coste y la posibilidad de realizarlas con un diseño personalizado para nuestros requisitos.

Sobre la viga se va a situar la bola, que en nuestro caso se ha elegido de billar, la cual podrá rodar de un extremo a otro por el carril en función de la inclinación de este. En los extremos se han colocado dos topes, para que la esfera no se salga de los límites del sistema y a su vez, no dañe el sensor de posición como los demás elementos del equipo.

En un extremo se ha colocado una tubería y un tapón de PVC donde se sitúa el sensor de posición que mide la distancia en la que se encuentra la bola.

En el otro extremo se sitúa la parte móvil del servomotor que será el encargado de variar el ángulo de la viga para provocar el movimiento de la bola.

A su vez, tanto servomotor como sensor de posición están conectados mediante cables con una placa Arduino Mega 2560, que se encuentra alojada en una caja en el centro de la estructura. Esta tarjeta será la encargada de facilitar la comunicación

entre el ordenador y planta y poder así transmitir las órdenes que enviemos a través de Matlab-Simulink, tanto recibir la posición de la bola como enviar la señal para controlar el servomotor. Todo ello en tiempo real.

A mayores, se ha colocado un display 16x2 para poder visualizar la posición de la bola, una fuente de alimentación para alimentar el servomotor, un interruptor eléctrico y un pequeño led, todo ello alojado en la caja de 190x115x75.

Figura 3. Caja en detalle donde se aprecia la conexión de alimentación y de USB al PC

Capítulo 3
Fundamentos teóricos

Para realizar el presente proyecto es necesario introducir unos conceptos teóricos con los que iremos profundizando más adelante y que nos ayudarán a comprender el sistema y su forma de controlarlo.

Controlador PID

Introducción

Un controlador PID es un elemento de control simultáneo por realimentación comúnmente usado en sistemas de control industrial. Su misión es eliminar el error estacionario entre la señal de referencia y la señal de control a la vez que tratamos de obtenerla lo más rápido posible y sin oscilaciones.

En la actualidad más del 95% de los lazos de control emplean un PID.

Es verdaderamente significativo ajustar adecuadamente los tres parámetros de control para conseguir los mayores beneficios.

Estructura del PID

Las tres componentes que forman el controlador son la acción proporcional, la acción integral y la acción derivativa.

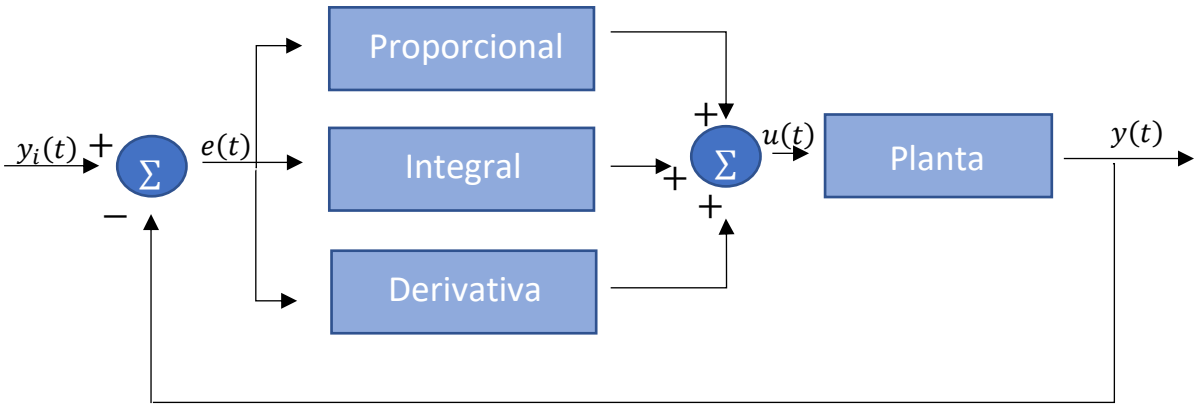


Figura 4. Diagrama de bloques de una estructura PID en una planta

La ecuación del controlador con las tres acciones implementadas se muestra a continuación:

$$u(t) = \boxed{k_p e(t)} + \boxed{\frac{k_p}{T_i}\int_0^t e(t)\,dt} + \boxed{k_p T_d \frac{de(t)}{dt}}$$

$P \qquad I \qquad D$

Que también podemos expresar en función de transferencia, donde se relacionará el error e(s) con la salida u(s).

$$C_{PID}(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad (1)$$

A continuación, se van a explicar las acciones que combina un controlador PID por separado.

Acción proporcional

La finalidad de esta acción es conseguir que el error en estado estacionario sea aproximadamente cero respecto a una referencia fija. Es decir, la salida de esta es proporcional al error de la siguiente manera: $u(t) = k_p \cdot e(t)$. La función de transferencia será controlable únicamente mediante una ganancia ajustable.

$$C_p(s) = K_p \qquad (2)$$

Un controlador proporcional puede controlar cualquier planta estable, pero en régimen permanente su capacidad es limitada.

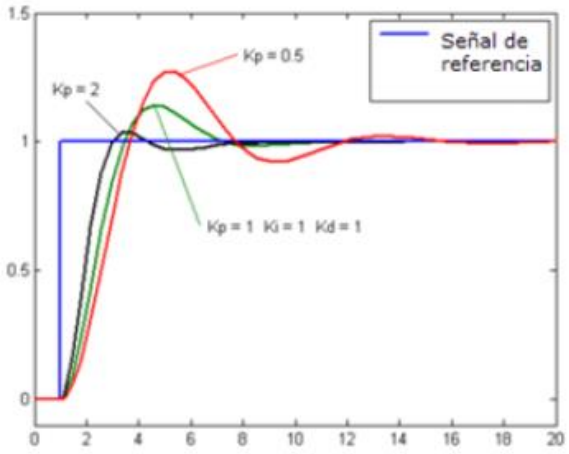


Figura 5. Respuesta de un sistema ante diferentes valores Kp

El error estacionario disminuye en el orden que la constante proporcional va aumentando, a la vez que la velocidad también es mayor pero las oscilaciones aumentan, tardando más en estabilizarse.

Si aumentamos k -> más rápido -> mayor sensibilidad al ruido -> peor estabilidad

Acción integral

Esta acción tiene como propósito disminuir y llegar a eliminar el error en estado estacionario, los cuales no pueden ser corregidos por el control proporcional. Es proporcional al error acumulado, por lo que será de respuesta lenta. La ecuación de un controlador con esta acción y su correspondiente función de transferencia se muestra a continuación.

$$u(t) = k_i \int_0^t e(\tau)\, d\tau \qquad\qquad c_i(s) = \frac{k_i}{s} \qquad (3)$$

En la figura 6 se ven los efectos que tiene esta acción integral en la estabilidad de un sistema.

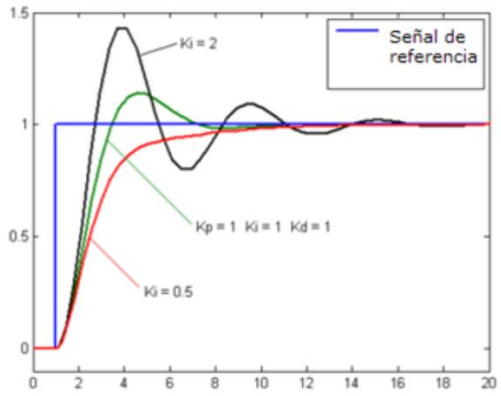


Figura 6. Respuesta de un sistema ante diferentes valores de Ki

Donde también podemos representar $T_i = \frac{1}{K_i}$. Siendo T_i el tiempo requerido para que la acción integral contribuya a la salida del controlador en una cantidad igual a la acción proporcional.

Acción derivativa

Esta acción actúa cuando hay un cambio absoluto del error. Al no corregir el error en la etapa transitoria, no se aplica ella sola. Es una acción con la capacidad de predecir por lo que será de acción rápida.

Su finalidad es corregir la señal de error antes de que se haga demasiado grande. La predicción se realizar mediante la extrapolación del error de control en la dirección de la tangente a la curva correspondiente, como se muestra en la figura 7.

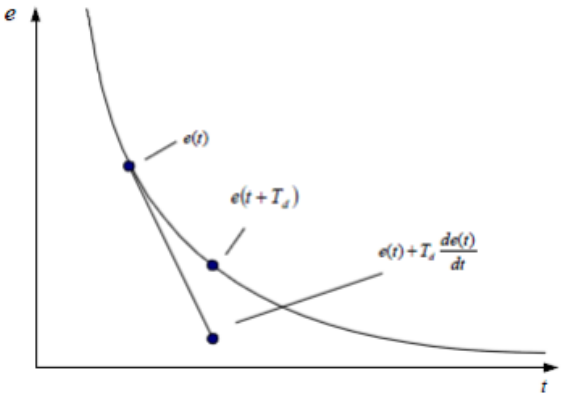


Figura 7. Extrapolación del error de control

En contrapartida, esta acción amplía las señales de ruido, lo que puede provocar saturación en el controlador. Puede emplearse en sistemas con tiempo de retardo considerables, porque permite una repercusión rápida de la variable después de presentarse una perturbación en el proceso.

En la figura 8 se observan las características de esta acción derivativa donde se representa un control PID en el que se varía la acción derivativa, ya que como hemos dicho antes, esta acción no se puede aplicar por ella misma.

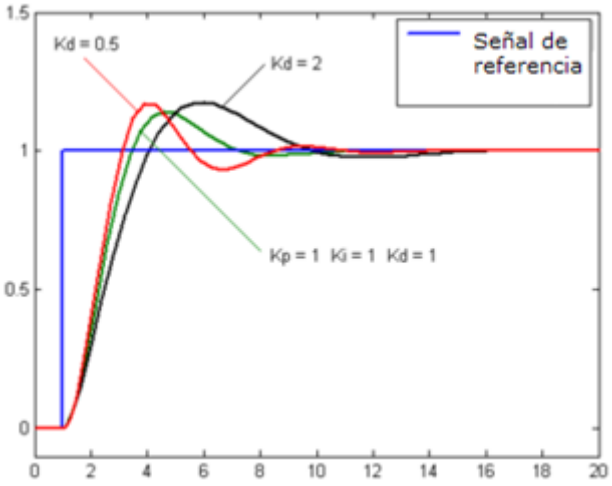


Figura 8. Respuesta de un sistema para diferentes valores de Kd

Sintonizar el controlador

La finalidad del sistema es conseguir que la variable del proceso sea igual a la referencia. Esto se logra cuando se ha elegido de forma satisfactoria los valores de las ganancias proporcional, integral y derivativa. Este procedimiento se conoce como sintonizar el controlador PID.

Esto suele ser un proceso iterativo donde se van variando los valores de las ganancias individualmente observando la respuesta del sistema. Sin embargo, hay software especializado que obtienen los valores adecuados o métodos que se basan en la teoría de control, como es el ejemplo de Zieglers y Nichols

Ziegler y Nichols

Primer método

Aplicar a la planta sólo control proporcional con una ganancia K_p pequeña. Posteriormente aumentar el valor de K_p hasta que el sistema empiece a oscilar. Estas oscilaciones deben ser lineales y regulares. Registramos la ganancia crítica $K_p = K_c$ y el período de oscilación P_c en la salida del controlador.

Con estos datos ajustamos los parámetros del controlador PID siguiendo la tabla adjunta.

	K_p	T_i	T_d
P	0.5 K_c	∞	0
PI	0.45 K_c	$\frac{P_c}{1.2}$	0
PID	0.6 K_c	0.5 P_c	0.125 P_c

Tabla 1. Parámetros de un controlador PID según el método de oscilación de Ziegler y Nichols

Segundo método

En este segundo método se emplea las características de la respuesta de la planta ante una entrada escalón unitario obtenida de manera experimental. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta tiene una forma de 'S'.

Esta curva se caracteriza por el tiempo de retardo 'd' y una constante de tiempo 'τ'. El tiempo de retardo y la constante de tiempo se hayan trazando la tangente en el punto de inflexión de la curva con forma de 'S' y determinando las intersecciones de esta recta con el eje de tiempo y la línea c(t)=k, como se aprecia en la figura 9.

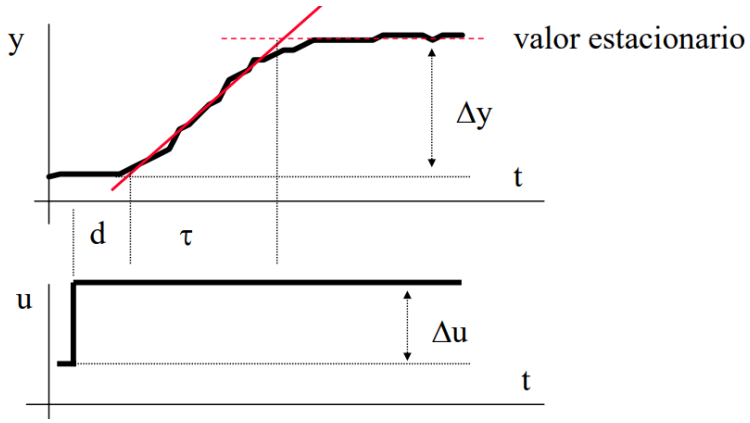


Figura 9. Respuesta del sistema en forma de 'S' ante una entrada escalón unitario

La función de transferencia se puede aproximar mediante un sistema de primer orden con un retardo determinado.

$$G(s) = \frac{Ke^{-ds}}{\tau s + 1} \qquad (4)$$

Siendo:

$$K = \Delta y / \Delta u \quad (5)$$

Donde se establecen los valores de K_p , T_i y T_d siguiendo la siguiente tabla, como en el caso anterior:

	$\mathbf{K_p}$	$\mathbf{T_i}$	$\mathbf{T_d}$
P	$\frac{\tau}{K\,d}$	∞	0
PI	$0.9\frac{\tau}{K\,d}$	$\frac{\tau}{0.3}$	0
PID	$1.2\frac{\tau}{K\,d}$	2 d	0.5 d

Tabla 2. Parámetros de un controlador PID según el método de la curva de reacción 'S' de Ziegler y Nichols

Sin embargo, los métodos de Ziegler y Nichols en la práctica no suelen ser tan eficientes como se ve teóricamente ya que es necesario llevar al sistema a situaciones a veces no posibles, por lo que en nuestro proyecto iremos ajustando las ganancias manualmente siguiendo unos criterios que se explicarán en su apartado correspondiente.

PID discreto

Durante el proyecto vamos a trabajar teóricamente tanto con el PID continuo como el digital en tiempo discreto. Posteriormente, en la práctica real, se realizará el control únicamente mediante la discretización del PID. Por tanto, aunque comparten la misma filosofía, cabe mencionar las particularidades del PID discreto, que se explicarán a continuación.

Primero de todo, al trabajar en tiempo discreto, necesitamos obtener una señal continua a partir de datos tomados en instantes de tiempo. Para crear esta señal usamos un mantenedor de orden cero (‘zoh’).

Este método permite mantener el último valor de la secuencia hasta encontrar un nuevo valor en el siguiente instante de tiempo. Este retenedor es el usado por defecto por Matlab cuando transformamos una función de transferencia a tiempo discreto y, por tanto, el que usaremos para el control teórico en el capítulo 5.

Las estructuras del PID digital y analógico tienen similitudes, trabajando en el dominio de ‘s’ o en el de su transformada ‘z’. A continuación, se representan sus estructuras:

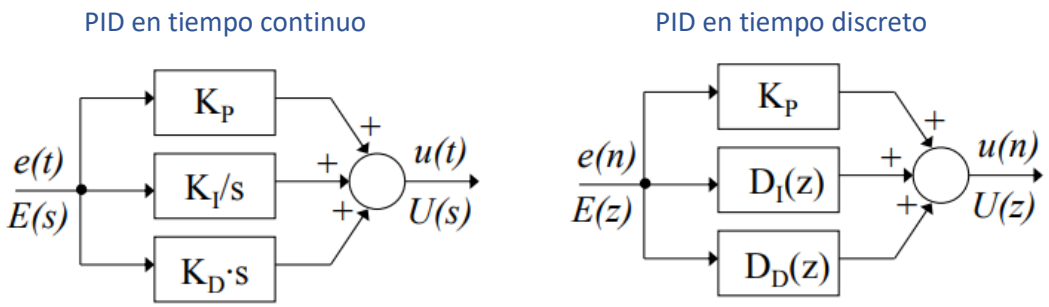


Figura 10. Estructura PID en tiempo continuo y tiempo discreto

La función de transferencia del controlador PID en tiempo discreto es la siguiente:

$$U(z) = K_p E(z) + D_i(z)E(z) + D_D(z)E(z) \quad (6)$$

Donde estamos relacionando la salida en el dominio de 'z', U(z), con el error en la entrada E(z).

También conviene representar la función del controlador expresada en el dominio 'z' a partir de su ecuación en tiempo continuo:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \quad (7)$$

Siendo, en términos de z:

$$C(z) = K_p + K_i \frac{z}{z-1} + K_d \frac{z-1}{z} = \frac{(K_p + K_i + K_d)z^2 - (K_p + 2K_d)z + K_d}{z^2 - z} \quad (8)$$

A partir de esta ecuación podemos trabajar con un controlador P, PD o PID discreto, considerando nulas las constantes que nos interesen en cada caso.

Controlador por realimentación de estados

Una alternativa a representar nuestro sistema con la función de transferencia es mediante la realimentación de estados.

Un sistema físico en espacio de estados se define como el conjunto de entradas, salidas y variables de estado declaradas como ecuaciones diferenciales de primer orden que forman un sistema matricial de primer orden.

Las variables de estado son el subconjunto más pequeño de un sistema que tienen la capacidad de representar el estado dinámico completo en un determinado instante. Estas variables de estado son linealmente independientes.

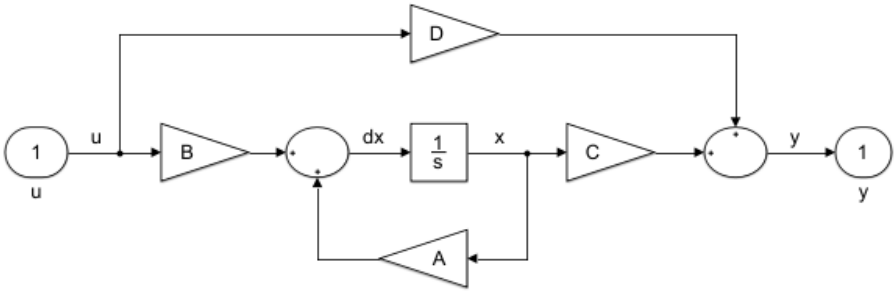


Figura 11. Diagrama de bloques de un sistema en espacio de estados

Donde x es el vector de estados, y el vector de salida, u el vector de entradas, A la matriz de estados, B la matriz de entrada, C la matriz de salida y D la matriz de transmisión directa. Se suele simplificar la matriz D tomándola como matriz nula.

El sistema lineal de p entradas, q salidas y n variables de estado se escribe de la siguiente manera.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t) \qquad (9)$$

La estabilidad puede ser estudiada mediante los autovalores de la matriz A, realizando el determinante de la siguiente expresión.

$$\lambda(s) = |sI - A| \qquad (10)$$

Se conoce $\lambda(s)$ como el polinomio característico y sus raíces, o autovalores, proporcionan los polos en la función de transferencia del sistema.

Cabe destacar que el sistema podría ser estable respecto a sus entradas y salidas aún si es internamente inestable. Esto podrá darse en el caso de que los polos inestables sean cancelados por ceros.

Controlabilidad

La controlabilidad implica que es posible, ante entradas admisibles, dirigir los estados desde cualquier valor inicial a cualquier valor final en un tiempo determinado.

En un modelo en espacio de estados continuo e invariante en el tiempo, se dice si es controlable si cumple:

$$rango[B\ AB\ A^2B\ \dots A^{n-1}B] = n \qquad (11)$$

Observabilidad

Se dice que un sistema es observable si para cualquier entrada y estado inicial, el estado actual puede determinarse usando solamente las salidas en un tiempo finito. Expresado de otra forma, que a partir de las salidas es posible conocer el comportamiento del sistema.

Un sistema continuo e invariante en el tiempo, por tanto, será observable si:

$$rango\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = n \qquad (12)$$

Cabe recordar que el rango de una matriz es el número de filas linealmente independientes.

Realimentación de estados

Una forma de realizar la realimentación de estados es multiplicar la salida por una matriz K y el resultado colocarlo como entrada del sistema:

$$u(t) = Ky(t) \tag{13}$$

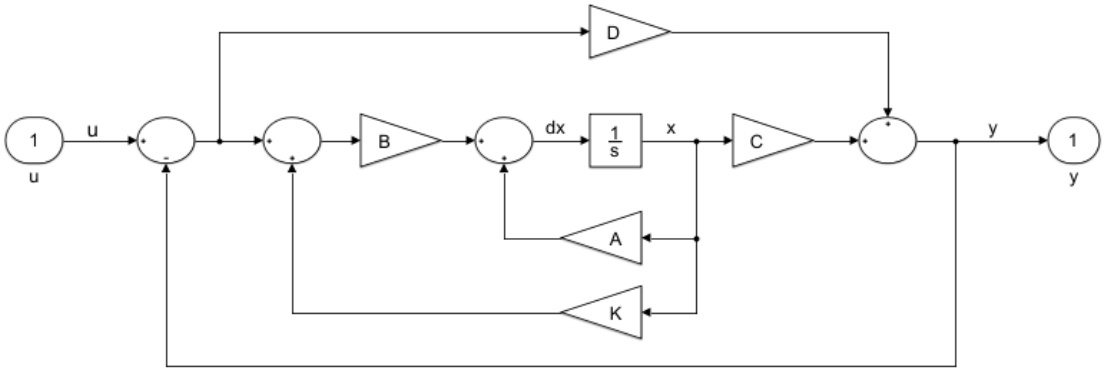


Figura 12. Diagrama de bloques de un sistema realimentado en espacio de estados

Por tanto, las ecuaciones que teníamos en espacio de estados:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned} \tag{14}$$

Se convierten en:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)Ky(t) \\ y(t) &= C(t)x(t) + D(t)Ky(t) \end{aligned} \tag{15}$$

Se puede resolver la ecuación de salida para y(t) y sustituir en la ecuación, resultando:

$$\begin{aligned} \dot{x}(t) &= (A + BK(I - DK)^{-1}C)x(t) \\ y(t) &= (I - DK)^{-1}Cx(t) \end{aligned} \tag{16}$$

Esto ofrece la ventaja de que los valores propios de A pueden ser controlados eligiendo K apropiadamente mediante la descomposición en sus valores propios de $(A + BK(I - DK)^{-1}C)x(t)$. Esto asume que el lazo abierto es controlable o que los

valores propios inestables de A pueden estabilizarse mediante la elección apropiada de K.

Típicamente, se suele simplificar eliminando D y eligiendo C como matriz unidad, lo que reduce las ecuaciones a lo siguiente:

$$\begin{aligned} \dot{x}(t) &= (A + BK)x(t) \\ y(t) &= x(t) \end{aligned} \qquad (17)$$

Finalmente, la descomposición de los autovalores queda reducida a $A + BK$.

Capítulo 4
Modelado de la planta

Modelo matemático

No podemos trabajar con un sistema y analizarlo si no conocemos su modelo matemático, ese conjunto de ecuaciones diferenciales que representan la dinámica del sistema. Un modelo no tiene por qué ser único para cada sistema, ya que puede haber varios modelos para el mismo sistema, según las necesidades que se planteen en el problema de control.

Vamos a tratar de expresar el modelo de nuestro sistema de dos formas diferentes, tanto en función de transferencia como la representación en el espacio de estados, y posteriormente vamos a trabajar con Matlab y Simulink para analizar los resultados y obtener algunas conclusiones.

A la hora de analizar y construir el modelo matemático es imprescindible declarar el número de grados de libertad del sistema. En este proyecto nos encontramos con dos, que son la posición de la bola y el ángulo de giro de la viga, que expresaremos en masa (m) y radianes (rad) respectivamente.

Por tanto, estas dos medidas, la posición de la bola y el ángulo de giro, son las variables de nuestro sistema de ecuaciones que definirán al sistema y nuestras coordenadas generalizadas, q_1 y q_2.

A la hora de realizar las ecuaciones matemáticas del modelo vamos a partir de una simplificación, como es que el eje de giro del motor y la línea por la que se desplaza el centro de la bola, además de ser perpendiculares entre sí, se cortan. Luego con esto se evita la aparición de términos cuadráticos en algunas de las ecuaciones.

Además, lo cual mencionaremos más adelante, vamos a suponer que la posición del motor es tan próxima al centro de masas de la viga que esta distancia es casi nula respecto a las demás componentes del sistema y, así de esta forma, la energía potencial del sistema sea independiente del ángulo de giro.

Para la realización del modelo en función de transferencia y el modelo en el espacio de estados se va a partir de la formulación lagrangiana y no de las ecuaciones del movimiento de Newton. La lagrangiana de un sistema es la diferencia entre las energías cinética y potencia (L=T-V), siendo por tanto un escalar.

De esta forma, mientras que con el procedimiento de Newton se coloca el énfasis sobre el agente exterior que actúa sobre el cuerpo (la fuerza), con el de Lagrange se manejan magnitudes asociadas al cuerpo (energías cinética y potencial). Este hecho es especialmente importante ya que hace que la lagrangiana de un sistema sea invariante ante los cambios de coordenadas. Esto permite pasar del espacio ordinario (en el que las ecuaciones de movimiento pueden ser muy complicadas) a un espacio de configuraciones elegido de tal forma que de una simplificación máxima.

Además, no se requiere establecer de forma exacta las fuerzas que actúan sobre el cuerpo, lo cual en algunas ocasiones ni siquiera es posible.

Se parte inicialmente de la conocida ecuación de Lagrange que se representa a continuación:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial F}{\partial \dot{q}_i} = Q_i \qquad (18)$$

Siendo:

L=T-V; Lagrangiana, donde T es la energía cinética y V la potencial.
Q_i=fuerza generalizada
q_i= cada una de las coordenadas generalizadas.
F=fuerza de disipación Rayleigh

Nuestras coordenadas generalizadas son:

q_1 la posición de la bola a lo largo de la viga.
q_2 el ángulo de giro de la viga respecto a la horizontal.

Vamos a empezar calculando la energía cinética total del sistema que será la que aporte la viga más la que aporte la esfera, que son las dos piezas en movimiento.

$$T = T_{viga} + T_{bola} \qquad (19)$$

$$T_{viga} = T_{rot} = \frac{1}{2} I_v w_v^2 = \frac{1}{2} I_v \dot{q}_2^2 \qquad (20)$$

$$T_{bola} = T_{tras} + T_{rot} = \frac{1}{2} m_b v_b^2 + \frac{1}{2} I_b w_b^2 \qquad (21)$$

A continuación, en la figura 13, se ilustran las variables q_1 y q_2 sobre el sistema de forma esquematizada para entender el significado y alcance de cada una en la trayectoria de la bola sobre la viga.

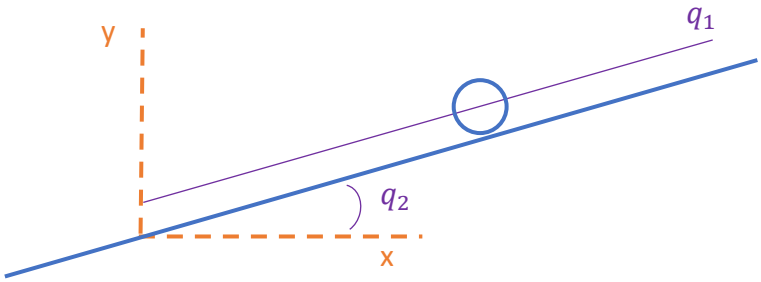


Figura 13. Representación de la trayectoria de la bola sobre la viga y sus coordenadas generalizadas q1 y q2

Puede observarse en la figura que la bola se desplaza en el eje vertical y horizontal debido a su propio movimiento y la inclinación de la viga. Este movimiento queda definido dentro de un triángulo rectángulo y, por lo tanto:

$$v_b^2 = \dot{x}^2 + \dot{y}^2 \qquad (22)$$

Donde, individualmente:

$$x = q_1 cosq_2 \qquad \dot{x} = \dot{q}_1\, cosq_2 - q_1\dot{q}_2 senq_2 \qquad (23)$$

$$y = q_1 senq_2 \qquad \dot{y} = \dot{q}_1\, senq_2 + q_1\dot{q}_2 cosq_2 \qquad (24)$$

Sustituyendo:

$$v_b^2 = \dot{q}_1^2 cos^2 q_2 + q_1^2\dot{q}_2^2 sen^2 q_2 - 2\dot{q}_1 q_1\dot{q}_2 senq_2 cosq_2 + \dot{q}_1^2 sen^2 q_2 + q_1^2\dot{q}_2^2 cos^2 q_2 + 2\dot{q}_1 q_1\dot{q}_2 senq_2 cosq_2 = \dot{q}_1^2 + q_1^2\dot{q}_2^2 \qquad (25)$$

La velocidad angular de la bola se calcula a través de la condición de rodadura sin deslizar, resultando:

$$\omega_b = \frac{\dot{q}_1}{R_b} \qquad (26)$$

Sustituyendo estas expresiones de velocidad en la ecuación de la energía cinética total del sistema se obtiene:

$$T = \frac{1}{2} I_v \dot{q}_2^2 + \frac{1}{2} m_b (\dot{q}_1^2 + q_1^2\dot{q}_2^2) + \frac{1}{2} I_b \left(\frac{\dot{q}_1}{R_b}\right)^2 \qquad (27)$$

O lo que es lo mismo:

$$T = \frac{1}{2} I_v \dot{q}_2^2 + \frac{1}{2} m_b \dot{q}_1^2 + \frac{1}{2} q_1^2 \dot{q}_2^2 + \frac{1}{2} I_b \left(\frac{\dot{q}_1}{R_b}\right)^2 \quad (28)$$

Siguiendo el mismo procedimiento, se puede calcular la energía potencial de la viga de forma análoga, tomando como punto de referencia la posición horizontal de la misma.

$$V = V_{viga} + V_{bola} \quad (29)$$

$$V_{viga} = -m_v g r cos q_2 \qquad V_{bola} = -m_b g q_1 sen q_2 \quad (30)$$

Donde g es la aceleración gravitacional y se considera negativa (-9.8 m/s^2) para evitar signos en la ecuación y r es la distancia desde la línea que pasa por el centro de la bola hasta el eje del motor, la cual se va a simplificar y considerar muy pequeña en relación a las demás medidas y, por tanto, se puede despreciar la energía potencial de la viga, $V_{viga} \approx 0$.

$$V = -m_b g q_1 sen q_2 \quad (31)$$

Por último, para aplicar la función lagrangiana, falta por calcular la fuerza de disipación de Rayleigh, que en este caso se considera nula por no contener ninguna fuerza disipativa.

Como se tienen dos coordenadas generalizadas, se obtendrán dos ecuaciones que rigen el comportamiento del sistema, donde la fuerza generalizada asociada a la coordenada posición de la bola es nula y la fuerza generalizada asociada a la coordenada de la viga es el par del motor, τ.

$$\left(m_b + \frac{I_b}{R_b^2}\right) \ddot{q}_1 + m_b g sen q_2 - m_b q_1 \dot{q}_2^2 = 0 \quad (32)$$

$$(I_v + m_b q_1^2) \ddot{q}_2 + 2 m_b q_1 \dot{q}_1 \dot{q}_2 - m_b g q_1 cos q_2 = \tau \quad (33)$$

Así quedaría definido el sistema dinámicamente, donde aparecen tres constantes, la masa de la bola (m_b) y los momentos de inercia de la viga (I_v) y la bola (I_b).

Sin embargo, se ha despreciado la segunda ecuación, ya que no se va a cerrar un lazo de control sobre el ángulo de la viga y resulta intrascendente en el modelo en función de transferencia.

Modelo en función de transferencia

Para obtener el modelo en función de transferencia se parte de la ecuación obtenida en el apartado anterior para la coordenada generalizada de posición de la bola (29) y posteriormente aplicaremos la transformada de Laplace.

A continuación, en la figura 14, se representa nuestro modelo particular y todas las variables reales que influyen en el sistema y en la función de transferencia.

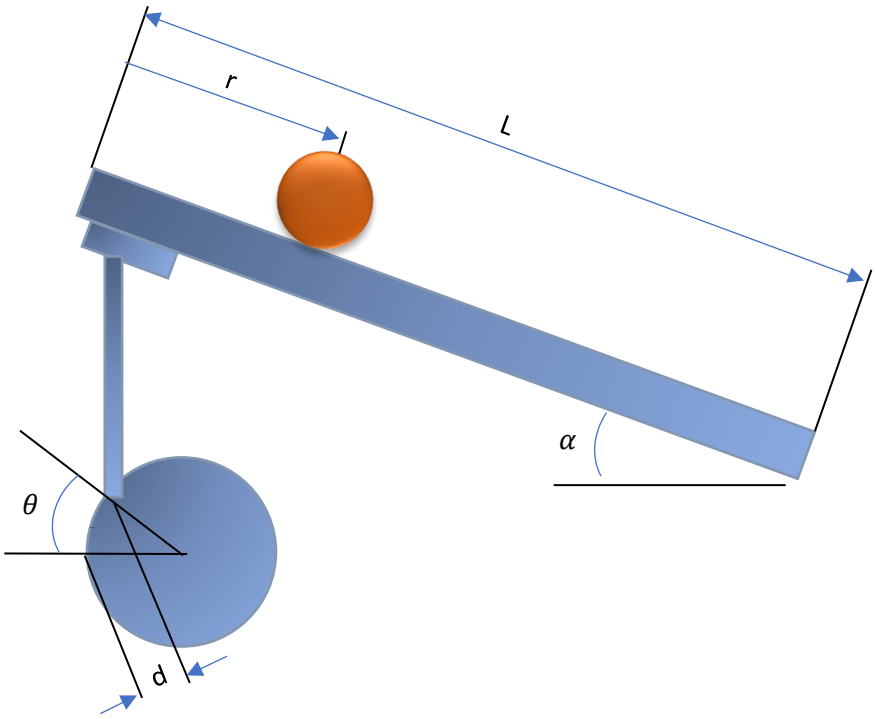


Figura 14. Sistema real con todas las constantes y variables que influyen en la planta para modelar la función de transferencia

Donde los valores y su nomenclatura se pueden ver en la tabla 3:

m	*Masa de la bola*
R	*Radio de la bola*
d	*Desplazamiento del brazo (offset)*
G	*Fuerza gravitacional*
L	*Longitud de la viga*
J	*Momento de inercia de la bola*
r	*Coordenada de la posición de la bola*
$\boldsymbol{\alpha}$	*Ángulo de la viga respecto a la horizontal*
$\boldsymbol{\theta}$	*Ángulo del servomotor*

Tabla 3. Símbolo y magnitud de las constantes y variables que influyen en la planta

Para obtener la función de transferencia, como se ha dicho anteriormente, se parte de la siguiente ecuación donde las coordenadas generalizadas se han sustituido por su valor real.

$$\left(\frac{J_b}{R_b^2} + m_b\right)\ddot{r} + m_b g sen\alpha - m_b r\dot{\alpha}^2 = 0 \qquad (34)$$

$\boldsymbol{q_1}$	r
$\boldsymbol{q_2}$	α

Tabla 4. Relación de coordenadas generalizadas con su valor real

Donde r es la posición de la bola medida desde el extremo de la viga y α es el ángulo de giro de esta, como se observa en la figura 14.

Para obtener la función de transferencia a partir de una ecuación diferencial esta debe ser necesariamente lineal y, como en este caso no lo es, hay que linealizarla en torno a un punto de equilibrio estable, que en este caso el único posible es α=0, que es la posición horizontal de la viga. Cabe recordar que cuando se linealiza una ecuación no lineal, el modelo que se obtiene solo se ajusta bien para valores cercanos al punto de equilibrio.

La ecuación linealizada queda de la siguiente forma, asumiendo que cuando $\alpha \approx 0$, el sen α se puede paroximar por el ángulo α(sen $\alpha \approx \alpha$)

$$\left(\frac{J_b}{R_b^2} + m_b\right)\ddot{r} = -m_b g\alpha \qquad (35)$$

Debemos relacionar el ángulo de la viga con el ángulo del servomotor, esta ecuación se puede aproximar como lineal mediante la siguiente igualdad.

$$\alpha = \frac{d}{L}\theta \qquad (36)$$

Sustituyendo en la ecuación 32:

$$\left(\frac{J_b}{R_b^2} + m_b\right)\ddot{r} = -m_b g \frac{d}{L}\theta \qquad (37)$$

Aplicando, ahora que es lineal, la transformada de Laplace a la ecuación para pasar de ecuaciones diferenciales al espacio polinómico definido por 's', donde existe el concepto de función de transferencia.

$$\left(\frac{J_b}{R_b^2} + m_b\right)R(s)s^2 = -m_b g \frac{d}{L}\Theta(s) \qquad (38)$$

Y tenemos definida la función de transferencia como el cociente entre la salida R(s) y la entrada $\Theta(s)$ al sistema, quedando representada por:

$$\mathrm{G}(s) = \frac{R(s)}{\Theta(s)} = -\frac{m_b g d}{L\left(\frac{J_b}{R_b^2} + m_b\right)}\frac{1}{s^2}\left[\frac{m}{rad}\right] \qquad (39)$$

Se puede observar que la función de transferencia tiene dos polos iguales en el origen, por lo que ya se puede prever que su comportamiento será inestable en lazo abierto, es decir, que será imprescindible cerrar el lazo de control para lograr la estabilidad.

Vamos a implementar el modelo en Matlab y para ello se necesitan los parámetros físicos del sistema, los cuales se van a elegir, estudiando nuestra planta real, los indicados en la tabla 5:

m	*0.160 kg*
R	*0.0285 m*
d	*0.03 m*
g	*-9.8 m/s²*
L	*0.43 m*
J	$(2/5)m_b R_b^2 = 5.1984 \cdot 10^{-5}$

Tabla 5. Parámetros físicos y su valor real en el modelo

Ahora introducimos estos valores con los siguientes comandos en el workspace de Matlab:

```
>> s=tf('s')
>> G=-(m*g*d)/(L*((J/R^2)+m)*s^2)
```

Resultando nuestra función de transferencia, que se describe a continuación.

$$G(s) = \frac{0.04704}{0.09632\ s^2} = \frac{0.4884}{s^2} \qquad (40)$$

Una vez tenemos la función de transferencia podemos estudiar y analizar su comportamiento ante una determinada entrada y diseñar un controlador que logre la estabilidad, lo cual estudiaremos más adelante.

Modelo en espacio de estados

Para obtener el modelo en representado en el espacio de estados también se parte de la primera ecuación (29) de las dos ecuaciones de Lagrange que se obtuvieron en el apartado general de modelado.

Como se ha visto en los aspectos teóricos, hay que elegir las variables de estado como aquellas que, conociendo su información en el estado inicial, y a través de conocer las entradas en cualquier instante, se conoce la salida para todo instante de tiempo. Se toma, por tanto, como variables de estado la posición de la bola (x) y su velocidad ($\dot{x}$) y como variable de entrada el ángulo de giro del motor (θ).

También se toma en este caso la simplificación del sistema linealizado y que ya estudiamos teóricamente, con lo que nos quedaría la representación habitual para un modelo en el espacio de estados:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad (41)$$

El cual hay que identificar términos a partir de la ecuación diferencial linealizada que se ha visto anteriormente:

$$\left(\frac{J_b}{R_b^2} + m_b\right)\ddot{r} = m_b g \frac{d}{L}\,\theta \qquad (42)$$

Para obtener finalmente el modelo en variables de estado para el sistema bola y viga, donde se ha elegido la matriz como [1 0] y la matriz D nula:

$$\begin{bmatrix}\dot{r}\\ \ddot{r}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}r\\ \dot{r}\end{bmatrix} + \begin{bmatrix}0\\ -\dfrac{m_b g d}{L\left(\frac{I_b}{R_b^2} + m_b\right)}\end{bmatrix}\theta \qquad (43)$$

Para implementar dicho modelo en Matlab utilizaremos los siguientes comandos, sustituyendo los valores de los parámetros físicos:

```
>> A = [0 1 ; 0 0]
>> B = [0 ; 0.4884]
>> C = [1 0]
>> D = [0]
>> BallandBeam = ss(A,B,C,D)
```

Con el modelo en el espacio de estados implementado en Matlab se puede ver el comportamiento ante una entrada determinada. Pero primero es conveniente realizar un estudio de la controlabilidad y la observabilidad del sistema.

Se recuerda que se decía de un sistema lineal y continuo era controlable si y solo si el rango de su matriz de controlabilidad era igual al orden del sistema. En este caso el orden del sistema es dos y la matriz de controlabilidad R queda como sigue:

$$R = (B\ AB) = \begin{pmatrix}0 & 0.4884\\ 0.4884 & 0\end{pmatrix} \qquad (44)$$

Introduciendo la matriz en Matlab y utilizando el comando rank(R) se obtiene que el rango de dicha matriz es dos y, por tanto, se puede decir que el sistema será controlable.

En el caso de la observabilidad la matriz a tener en cuenta, O, queda de la siguiente manera:

$$0 = \begin{pmatrix}C\\ CA\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad (45)$$

Se decía entonces que el sistema lineal era observable si y solo si el rango de la matriz O era igual al orden del sistema y su determinante no es nulo, cosa que también se comprueba que es cierta de forma análoga a la controlabilidad.

Por lo tanto, será posible controlar dicho sistema mediante la observación de la salida exclusivamente. Esto se estudiará más adelante cuando se tomen como variables de estado la salida y su derivada con respecto al tiempo.

Capítulo 5
Control teórico del sistema

Como se ha mencionado anteriormente, ya se prevé el sistema inestable en lazo abierto por la posición de los polos en el plano. Independientemente, se va a verificar esta condición sometiendo al modelo en función de transferencia a una entrada y analizar su respuesta.

Recordamos que la función de transferencia obtenida es la siguiente:

$$G(s) = \frac{0.4884}{s^2} \qquad (46)$$

Un método eficiente y clásico para analizar la estabilidad a partir del modelo es sometiéndolo a una entrada tipo salto y observar la salida, analizando si esta alcanza un valor estable en régimen permanente, es decir, para una entrada acotada la salida sea acotada

La manera óptima para hacer esto es introducir lo siguiente en la línea de comandos:

>> step(G(s))

Mostrando la gráfica que se ilustra a continuación.

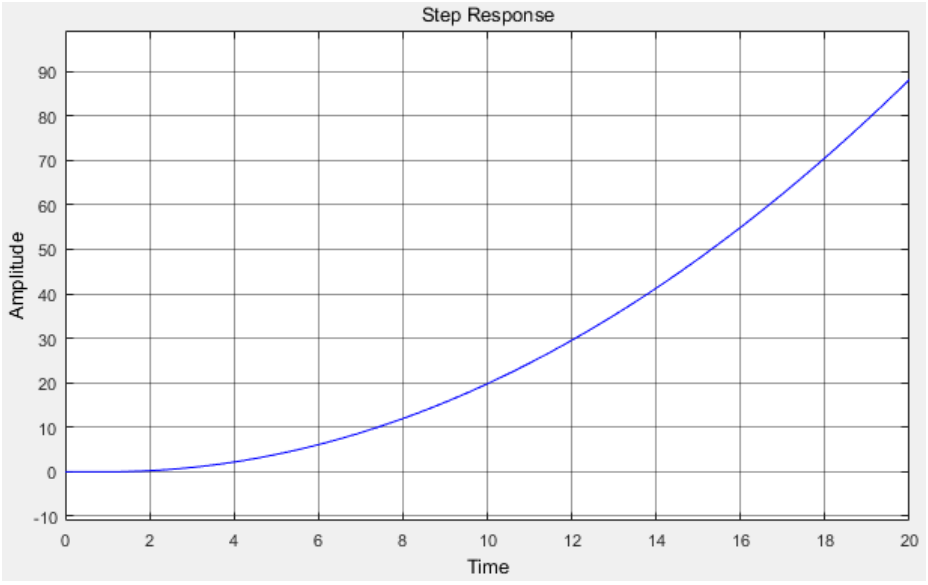


Figura 15. Respuesta temporal del sistema en lazo abierto ante una entrada escalón

Como se puede observar la salida tiende a infinito, con lo que se comprueba que el sistema es inestable en lazo cerrado. Esta respuesta se había augurado observando la posición de los polos en el plano y también, si se piensa en el sistema físico como tal, ya que conociendo que la entrada es la tensión del motor e introducimos un valor fijo, la viga se inclinará hacia un lado y la esfera rodará hacia un extremo sin control y sin posibilidad de alcanzar su punto de equilibrio

Por tanto, como ya se ha adelantado anteriormente, es imprescindible incorporar un controlador que nos estabilice el sistema.

En los siguientes apartados vamos a desarrollar cuatro formas de controlar el sistema ayudándonos de la ventana de comandos de Matlab. El código que apoya estos apartados se encuentra recopilado en el anexo.

Controlador PID

Para el control mediante PID se utiliza el modelo en función de transferencia y se cierra un lazo de realimentación respecto a la salida. Como ya se ha visto teóricamente la función de transferencia de un regulador PID es la siguiente:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \qquad (47)$$

A continuación, se representa el sistema general en simulink para ver la estructura que tiene el sistema en lazo cerrado con un controlador PID.

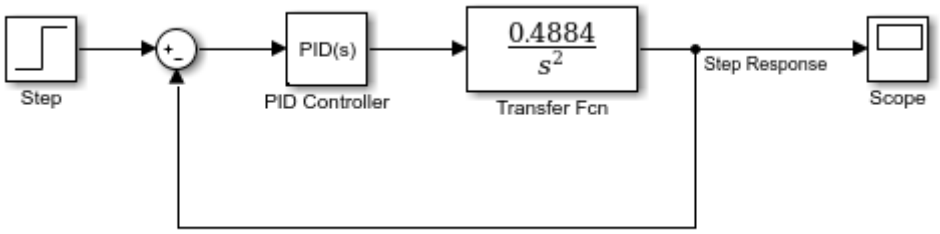


Figura 16. Estructura de un sistema realimentado con un controlador PID

Vamos a tratar de mejorar la respuesta modificando los parámetros del PID paso a paso, a la vez que se muestran las correspondientes funciones de transferencia y su respuesta temporal gráficamente.

Normalmente, a no ser que sea muy evidente, hay que comenzar probando con la acción proporcional simplemente, ya que muchas veces es suficiente. Así que se comprueba con los siguientes comandos de Matlab.

```
>> Kp=1;
>> C=pid(Kp);
>> sys_cl=feedback(C*G,1)
```

$$sys\_cl(s) = \frac{0.4884}{s^2 + 0.4884} \quad (48)$$

Con lo que se obtiene la función de transferencia del sistema en lazo cerrado y se observa que los polos han cambiado.

```
>>step(sys_cl)
>>axis([0 20 0 2])
```

Se puede mostrar gráficamente la respuesta del sistema en lazo cerrado ante entrada escalón con el comando Step como hicimos anteriormente.

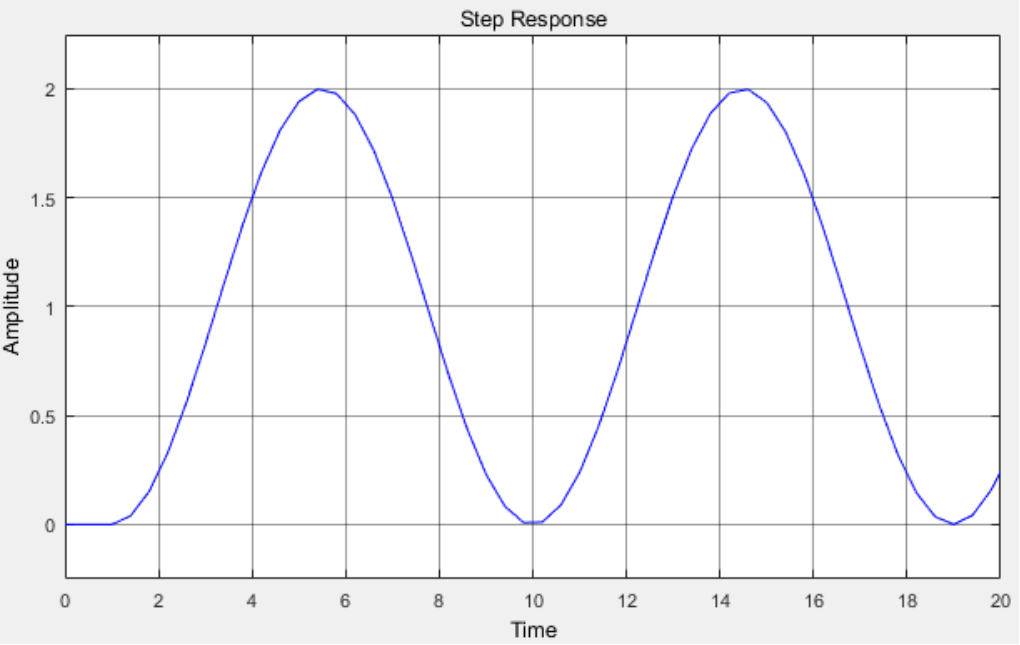


Figura 17. Respuesta temporal del sistema en lazo cerrado ante una entrada salto con acción proporcional unitaria

Como se puede observar el sistema es críticamente estable, por lo que vamos a añadir una acción derivativa para mejorar la situación mediante los siguientes comandos de Matlab.

```
>> Kp=1;
>> Kd=10;
>> C=pid(Kp,0,Kd);
>> sys_cl=feedback(C*G,1)
```

$$sys\_cl(s) = \frac{4.884s + 0.4884}{s^2 + 4.884s + 0.4884} \quad (49)$$

```
>>step(sys_cl)
>>axis([0 20 0 1.2])
```

Se obtiene ahora gráficamente la respuesta del sistema en lazo cerrado ante entrada escalón.

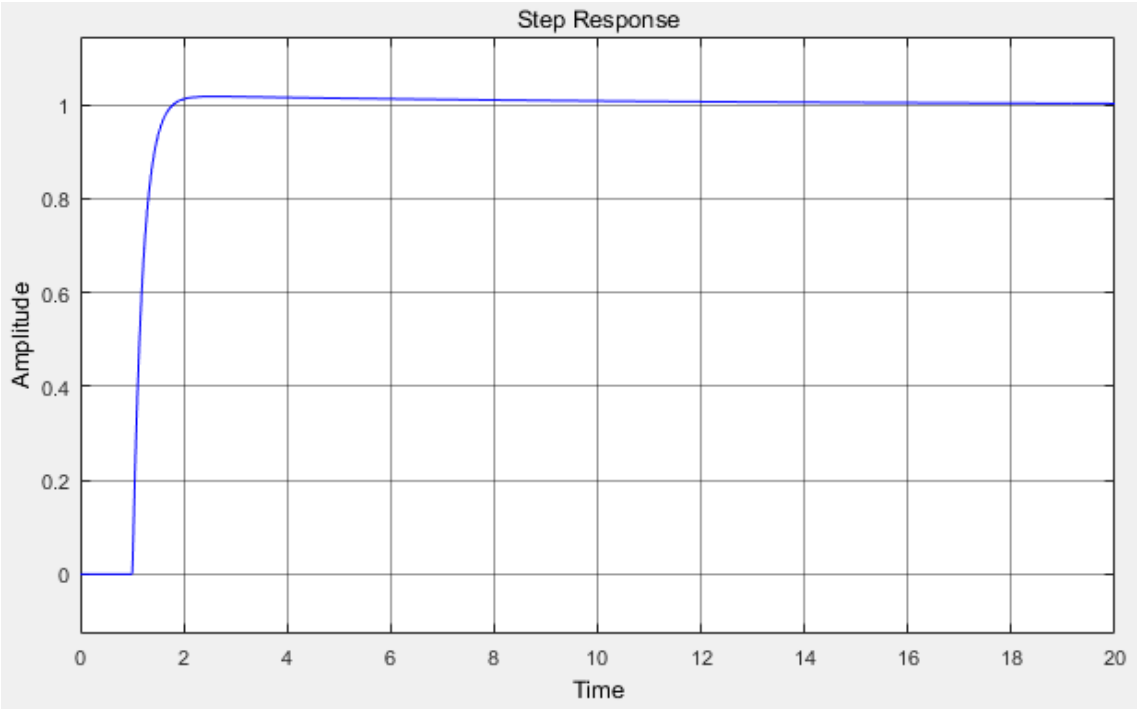


Figura 18. Respuesta temporal del sistema en lazo cerrado ante una entrada salto con acción proporcional-derivativa

Se ha conseguido estabilizar y mejorar el sistema gracias a la acción derivativa, consiguiendo un tiempo de establecimiento en régimen permanente más reducido, de 3.5 segundos, y una sobreoscilación del 5% aproximadamente. Como se están haciendo pruebas estos valores no son muy significativos, pero lo normal es diseñar un controlador que cumpla con unas especificaciones mínimas.

Teóricamente, precisamos de la acción integral para asegurarnos un error de posición nulo, por lo que a continuación se añade este término.

```
>> Kp=1
>> Ki=2
>> Kd=60
>> C=pid(Kp, Ki, Kd);
>> sys_cl=feedback(C*G,1)
```

$$sys\_cl(s) = \frac{29.3s^2 + 0.4884s + 0.9768}{s^3 + 29.3s^2 + 0.4884s + 0.9768} \quad (50)$$

```
>>step(sys_cl)
>>axis([0 5 0 1.2])
```

Se comprueba ahora la respuesta del sistema en lazo cerrado ante entrada salto.

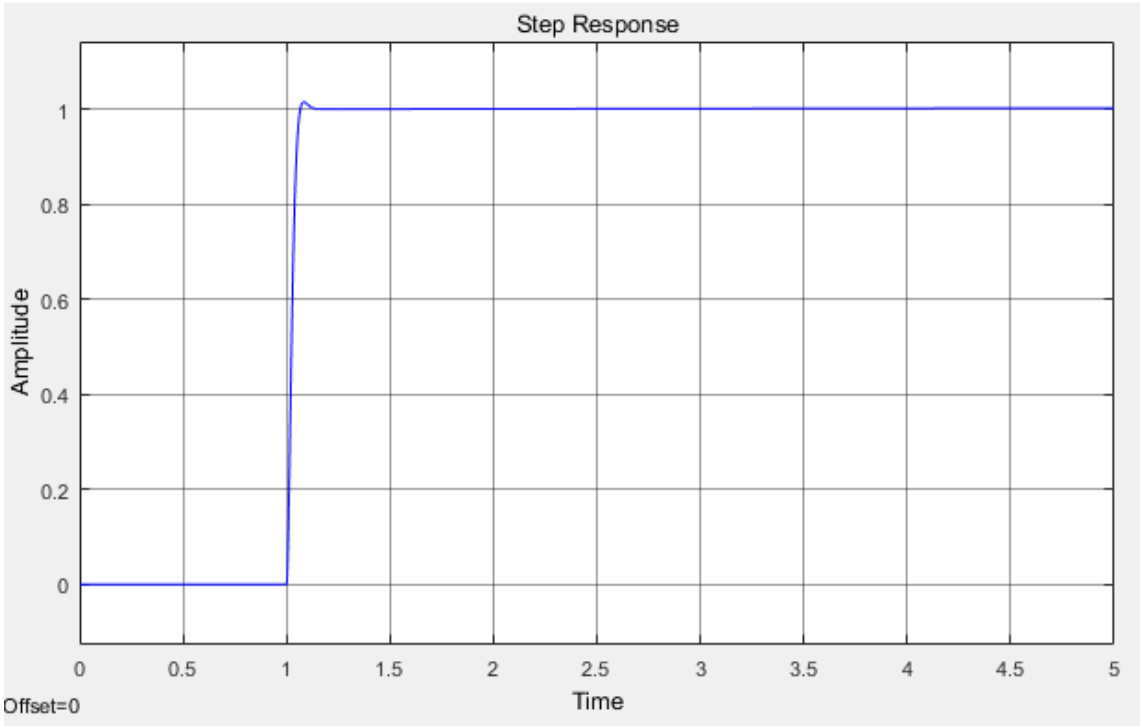


Figura 19. Respuesta temporal del sistema en lazo cerrado ante una entrada tipo salto con acción proporcional, derivativa e integral

Como puede observarse, la respuesta ha mejorado ligeramente ya que no tenemos sobreoscilación y el tiempo de establecimiento es de apenas 0.2 segundos, pero en cuestiones prácticas podemos asegurar que esta respuesta solo es posible con el modelo teórico y que no es físicamente realizable en la maqueta. Es muy probable que el motor no tenga la capacidad de realizar la respuesta que genera el regulador.

Lugar de las raíces

La principal idea del trabajo en el lugar de las raíces es estimar la respuesta del sistema en lazo cerrado a partir de la gráfica del lugar de las raíces del lazo abierto. Al agregar ceros y/o polos al sistema original (agregando un compensador), se modificarán el lugar de las raíces y, por tanto, la respuesta del sistema en lazo cerrado.

Añadir un polo	Añadir un cero
Desplaza el lugar de las raíces a la derecha	Desplaza el lugar de las raíces a la izquierda
Disminuye la estabilidad relativa	Aumenta la estabilidad relativa
Aumenta el tiempo de establecimiento	Disminuye el tiempo de establecimiento

Tabla 6. Efecto de añadir un polo o un cero en el sistema

Primero, veamos el lugar de las raíces de la planta en lazo abierto. Introducimos el siguiente comando para trazar el lugar de las raíces:

```
>> rlocus(G)
```

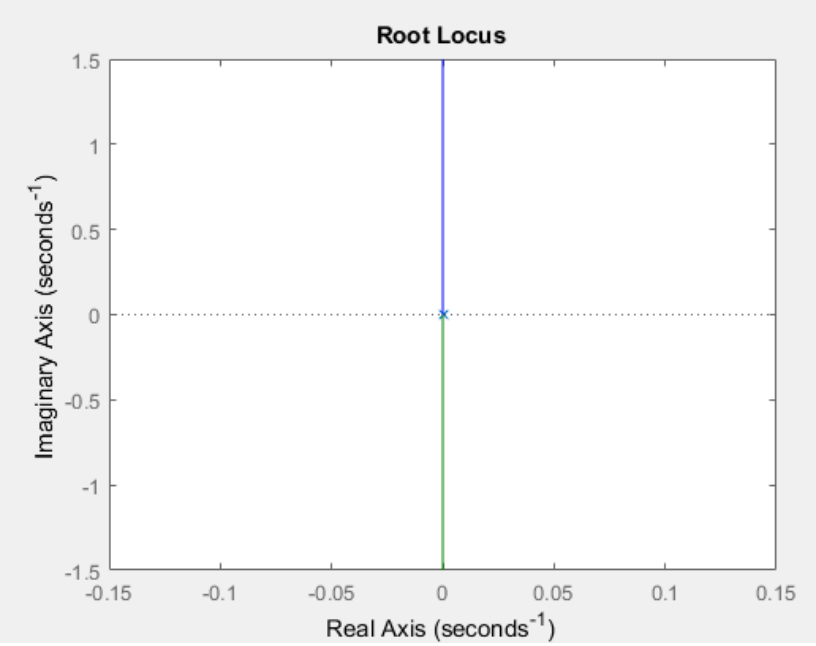


Figura 20. Lugar de las raíces del sistema en lazo abierto

Como vemos, el sistema en lazo abierto tiene dos polos en el origen, por lo que en lazo cerrado tendrá dos polos complejos conjugados que se extienden hasta el infinito a lo largo de los ejes imaginarios en función del valor de K. Es, por tanto, imposible estabilizar el sistema con un controlador proporcional.

Los criterios de diseño también se pueden trazar y analizar en el lugar de las raíces usando el comando sgrid.

Este comando genera una cuadrícula de relación de amortiguamiento constante y frecuencia natural. La relación de amortiguamiento y la frecuencia natural se encuentra utilizando la siguiente ecuación, que los relaciona con nuestros requisitos de exceso de porcentaje (%OS) y tiempo de establecimiento (Ts).

$$\%OS = 100e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \quad (51)$$

$$T_s = \frac{4}{\zeta\omega_n} \quad (52)$$

Hay que tener en cuenta que la ecuación con Ts existe al asumir que el sistema se ha resuelto cuando la respuesta permanece dentro del 2% de su valor final. A partir de estas ecuaciones, la relación de amortiguamiento y frecuencia natural se establece para ser 0.7 y 1.9 respectivamente.

```
>> sgrid(0.7, 1.9)
>> axis([-5 5 -2 2])
```

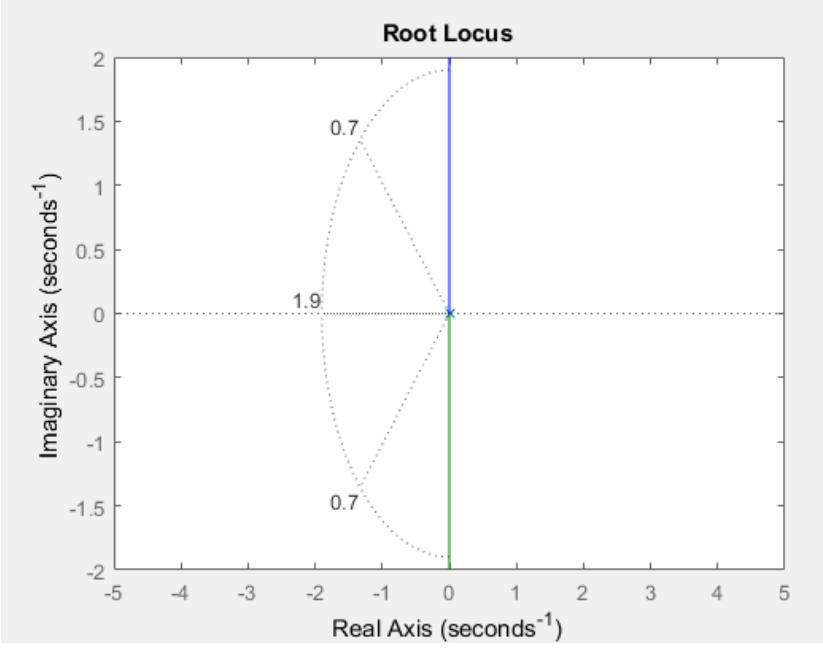


Figura 21. Lugar de las raíces del sistema con los requisitos de diseño

El área entre las dos líneas diagonales a la izquierda de la línea de puntos representa ubicaciones en las que el porcentaje de rebasamiento es inferior al 5%. El área fuera de la línea curva representa situaciones donde el tiempo de establecimiento es

inferior a 3 segundos. Hay que tener en cuenta que ninguna región se encuentra dentro de los criterios de diseño que se han impuesto como ejemplo. Para solucionar esto y llevar el lugar de las raíces al plano de la izquierda para la estabilidad, vamos a tratar de agregar un compensador de fase (Lead-compensator).

Un compensador de fase de primer orden tiende a desplazar el lugar de las raíces al plano de la izquierda. Su fórmula genérica se muestra a continuación:

$$C(s) = k_c \frac{(s - z_0)}{(s + p_0)} \qquad (53)$$

Donde la magnitud z_0 debe ser menor que p_0.

Ahora, vamos a agregar el controlador a la planta y veamos el lugar de las raíces. Posicionaremos el cero cerca del origen para cancelar uno de los polos. Por otro lado, el polo de nuestro compensador se colocará a la izquierda del origen para desplazar el lugar de las raíces más hacia el plano de la izquierda. Agregaremos los siguientes comandos para ello:

```
>> zo=0.01;
>> po=2.7;
>> C=tf([1 zo], [1 po]);
>> rlocus(C*G);
>> sgrid(0.70, 1.9);
>> axis([-3 0 -2 2]);
```

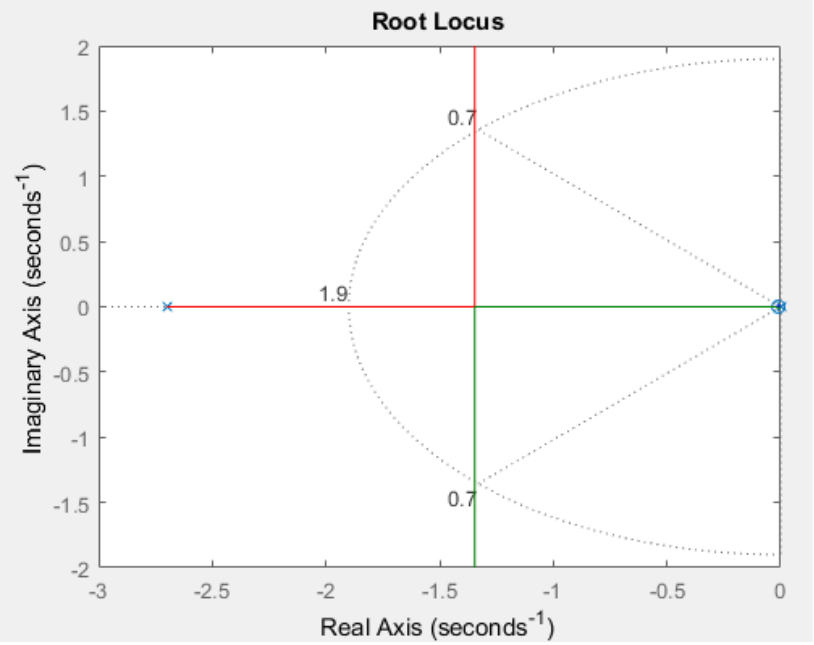


Figura 22. Lugar de las raíces con los polos desplazados al lado izquierdo

Ahora, las ramas del lugar de las raíces están dentro de unos criterios de diseño aceptables.

Una vez que hemos movido el lugar de las raíces al plano de la izquierda, podemos seleccionar una ganancia que satisfaga nuestros requisitos de diseño. Podemos usar el comando rlocfind para facilitarnos esto. Escribiremos en la línea de comandos:

```
>> [k, polos]=rlocfind(C*G)
```

Entonces vamos a la gráfica y seleccionamos un punto cerca de los indicados por las marcas de cruz en la gráfica.

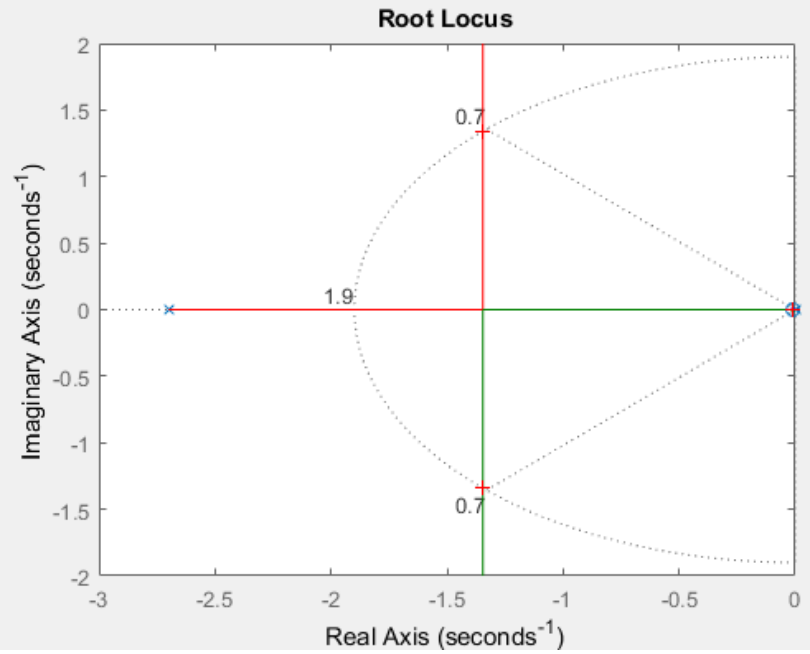


Figura 23. Lugar de las raíces seleccionando la ganancia

Después de hacer esto, vemos lo siguiente en la venta de comandos de Matlab.

```
selected_point =
        -1.3408+13373i

k=
        7.4207

polos =
        -1.3450+1.3373i
        -1.3450-1.3373i
        -0.0101
```

Ahora con la ganancia k ya tenemos el conjunto del controlador que estabilice nuestro sistema.

Este valor de Kc se puede colocar en el sistema y se puede obtener la respuesta temporal en lazo cerrado. A continuación, se muestra este sistema sobre nuestra planta con el nuevo controlador en simulink.

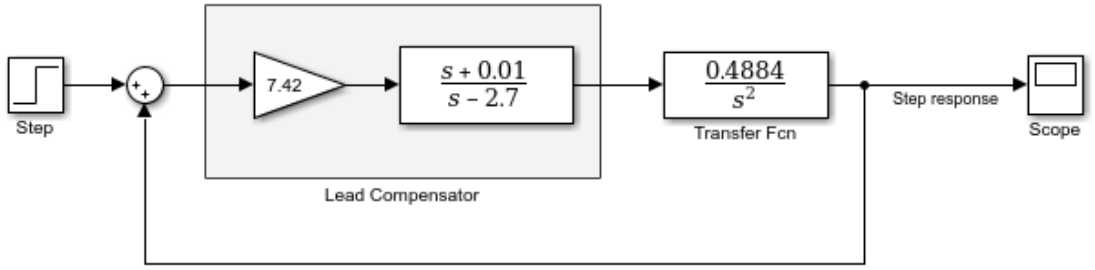


Figura 24. Diagrama de bloques del sistema con el Lead Compensator actuando sobre la función de transferencia

Podemos entonces con Matlab obtener la salida del lazo cerrado con este controlador como hemos hecho hasta ahora.

```
>> sys_cl=feedback(k*C*G,1);
>> step(sys_cl)
```

Y su respuesta temporal ante una entrada salto es la siguiente.

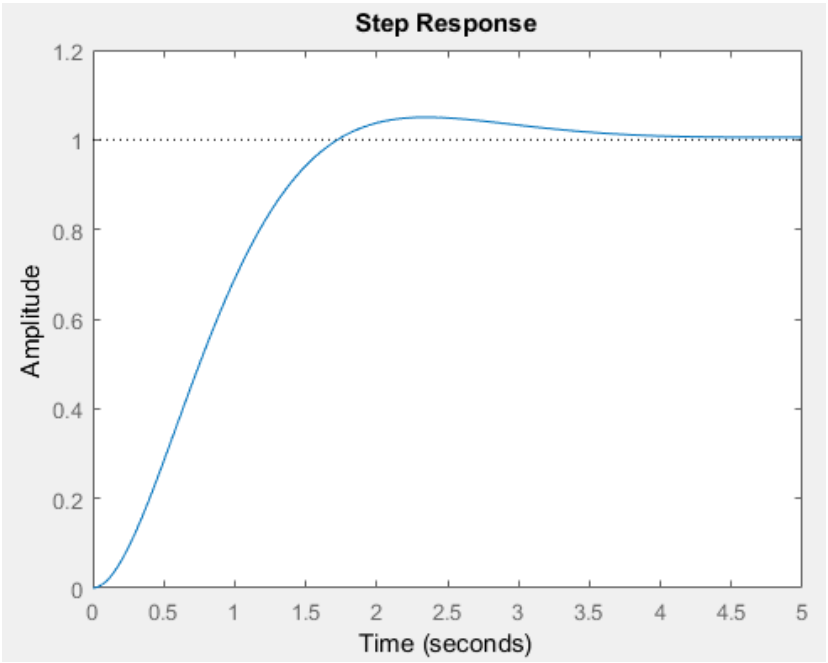


Figura 25. Respuesta temporal del sistema en lazo cerrado ante una entrada salto con un compensador de fase

PID digital

En nuestro modelo representado en simulink vamos a trabajar con un controlador PID discreto, por lo que es adecuado estudiar también el PID digital.

Como hemos visto en el apartado del control mediante PID, la función de transferencia del controlador para el sistema continuo es la siguiente:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \quad (54)$$

Debemos fijarnos en que la función de transferencia está expresada en función de 's', pero ahora debemos realizar la transformación para expresar la función de transferencia en función de 'z'. La ecuación es la siguiente:

$$C(z) = K_p + K_i \frac{z}{z-1} + K_d \frac{z-1}{z} = \frac{(K_p + K_i + K_d)z^2 - (K_p + 2K_d)z + K_d}{z^2 - z} \quad (55)$$

El primer paso, como se presupone lógico, es transformar la función de transferencia de un sistema continuo a la función de transferencia equivalente en un sistema discreto. Esto en Matlab lo podemos hacer con el comando c2d (continuous to discrete).

'c2d' es un comando al que le tenemos que especificar tres argumentos: el sistema o ecuación del modelo, el método y el tiempo de muestreo (Ts). Es recomendable que este tiempo de muestreo sea menor que $1/30\omega$, siendo ω la frecuencia de ancho de banda en lazo cerrado. El método que vamos a implementar se denomina zero-order hold, mantenedor de orden zero, 'zoh'.

Procedemos a introducir en Matlab estos comandos, recordando que nuestra función de transferencia 'G' del sistema es, como se vio en la ecuación 43:

$$G(s) = \frac{0.04704}{0.09632\ s^2} = \frac{0.4884}{s^2} \quad (56)$$

```
>> Ts=1/50;
>> Gd=c2d(G,Ts,'zoh')
```

Obteniendo la siguiente función de transferencia en tiempo discreto:

$$Gd(z) = \frac{9.76810^{-05}z + 9.76810^{-05}}{z^2 - 2z + 1} \quad (57)$$

Ahora estamos en disposición de observar la respuesta en lazo abierto ante una entrada salto. Vamos a definir una magnitud de 0.1 y un tiempo de 5 segundos. Para ello, introducimos los siguientes comandos.

```
>> [x,t]=step(0.1*Gd,5);
>> stairs(t,x)
```

Cuya respuesta podemos ver en la figura 26.

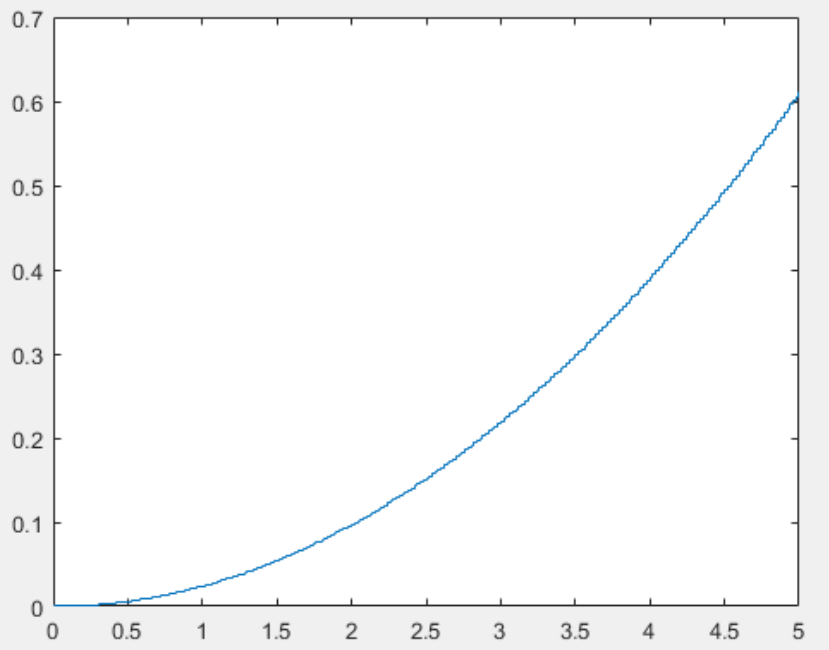


Figura 26. Respuesta temporal del sistema discreto en lazo abierto

Se observa una pequeña forma de escalera debido a las características de trabajar en tiempo discreto y el 'zoh'. Como es de esperar y ya hemos estudiado, la respuesta es inestable.

También podemos ver el lugar de las raíces del sistema discreto haciendo:

```
>> rlocus(Gd);
```

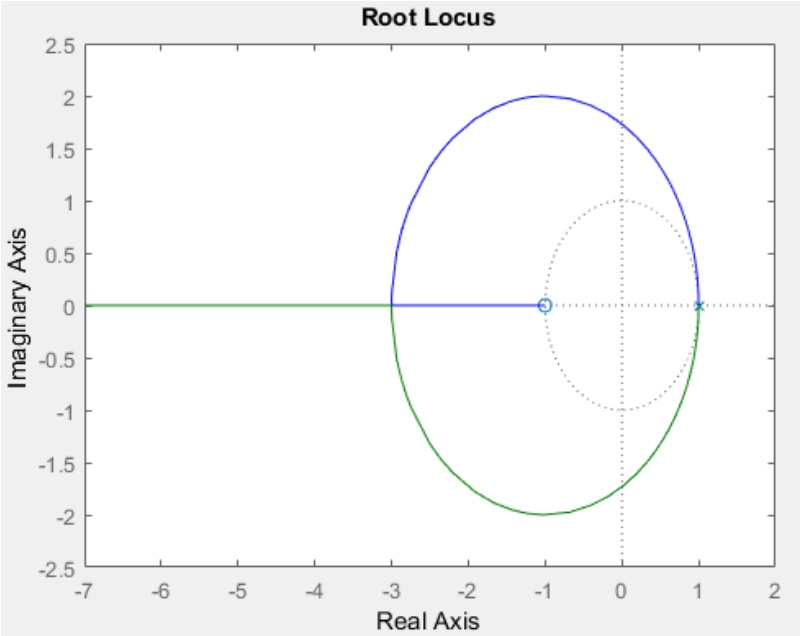


Figura 27. Lugar de las raíces del sistema discreto

Observamos como el sistema en lazo abierto tiene dos polos en z=1 y en lazo cerrado los polos van a estar fuera del círculo unidad sea cual sea el valor de K. El sistema nunca se va a hacer estable solo con un controlador con constante Kp independientemente de su valor, como ya vimos anteriormente para el caso continuo, y necesitaremos, al menos, la parte derivativa para construir un controlador PD y estabilizar el sistema.

De todos modos, vamos a cerrar el lazo y añadir la constante proporcional Kp para analizar los resultados. Implementaremos un valor Kp=100.

```
>> Kp=100;
>> sys_cl=feedback(Kp*Gd,1);

>> [x,t]=step(0.1*sys_cl,5);
>> stairs(t,x)
```

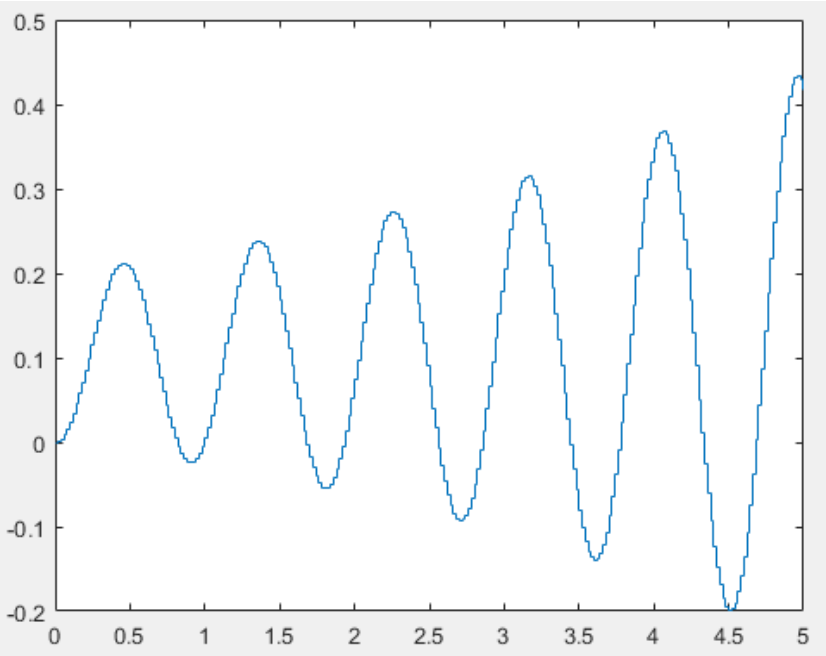


Figura 28. Respuesta en lazo cerrado del sistema discreto con Kp=100

Observamos, como era de suponer, que el sistema tiende a ser inestable. Al analizar esta tendencia, podemos deducir que el sistema nunca se va a hacer estable solo con un controlador con constante Kp independientemente de su valor y que necesitaremos, al menos, de la parte derivativa para construir con un controlador PD y estabilizar el sistema, como hemos deducido del lugar de las raíces, figura 27.

Vamos a continuar añadiendo esta constante derivativa Kd manteniendo la constante proporcional Kp=100. Elegimos un valor Kd=20 y escribimos en Matlab:

```
>> z=tf('z',Ts);

>> Kp=100;
>> Kd=20;

>> C=Kp+Kd*(z-1/z);

>> sys_cl=feedback(C*Gd,1);

>> [x,t]=step(0.1*sys_cl,5);
>> stairs(t,x)
```

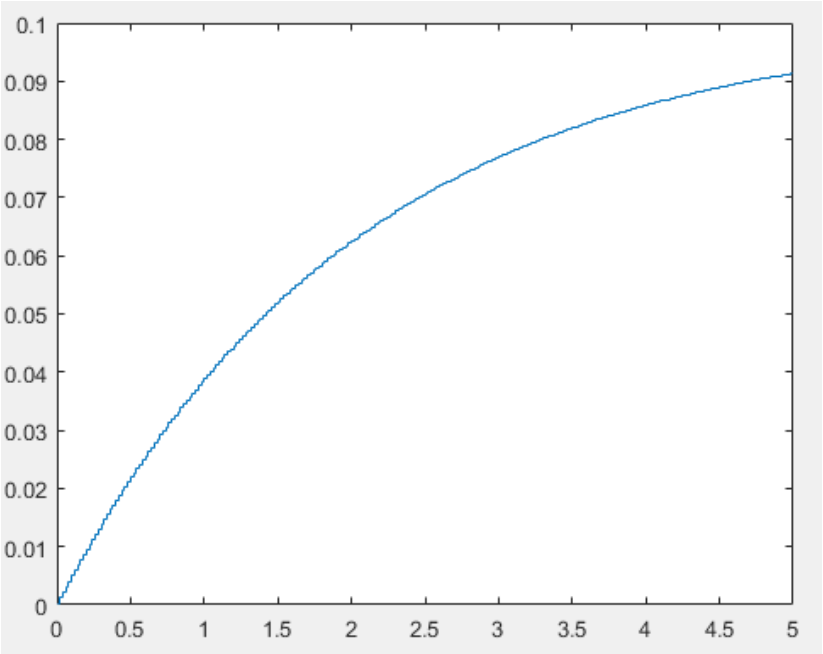


Figura 29. Respuesta en lazo cerrado del sistema discreto con Kp=100 y Kd=20

Se aprecia que el sistema es estable pero el tiempo de establecimiento es demasiado alto. Estudiando la teoría del PID, sabemos que al incrementar Kp decrementamos este tiempo de establecimiento. Por tanto, vamos a probar con un Kp mayor y volver a estudiar el caso del PD como acabamos de hacer.

```
>> z=tf('z',Ts);

>> Kp=2000;
>> Kd=20;

>> C=Kp+Kd*(z-1/z);

>> sys_cl=feedback(C*Gd,1);

>> [x,t]=step(0.1*sys_cl,5);
>> stairs(t,x)
```

Obteniendo la siguiente gráfica:

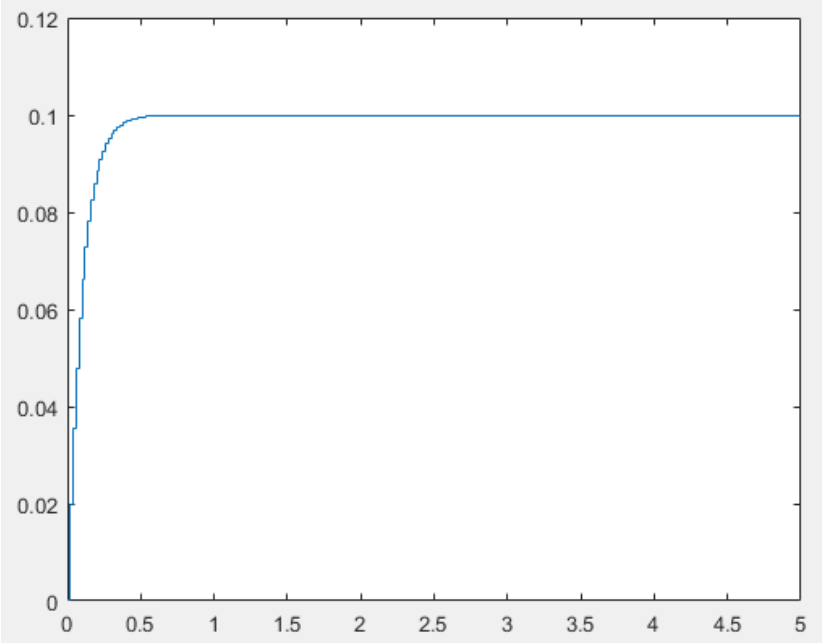


Figura 30. Respuesta en lazo cerrado del sistema discreto con Kp=2000 y Kd=20

Observamos claramente como la respuesta es mejor ya que alcanza el estacionario en menor tiempo. Realmente en este modelo teórico a mayor constante proporcional vamos a obtener una mejor respuesta, pero en nuestra planta real una mayor constante Kp va a producir que la bola alcance más velocidad y sea incontrolable.

En este problema ideal no es necesario la implementación de la constante integral, pero como vamos a ver en el caso real, esto no será así. Los modelos difieren bastante en su caso teórico y su implementación práctica, pero todo ello lo veremos más adelante en el capítulo del control real del sistema.

Control mediante realimentación de estados

Anteriormente se mostró el sistema en espacio de estados y se analizó la controlabilidad y observabilidad concluyendo que sí se puede realizar un control mediante la realimentación de estados. Ahora se van a proponer unas herramientas y procedimientos para estudiar el sistema y controlarlo eficazmente mediante esta metodología, principalmente desde Matlab.

Lo primero es crear el modelo en Simulink que representa nuestro sistema, en el que tenemos el ángulo como variable de entrada y la posición de la bola como salida:

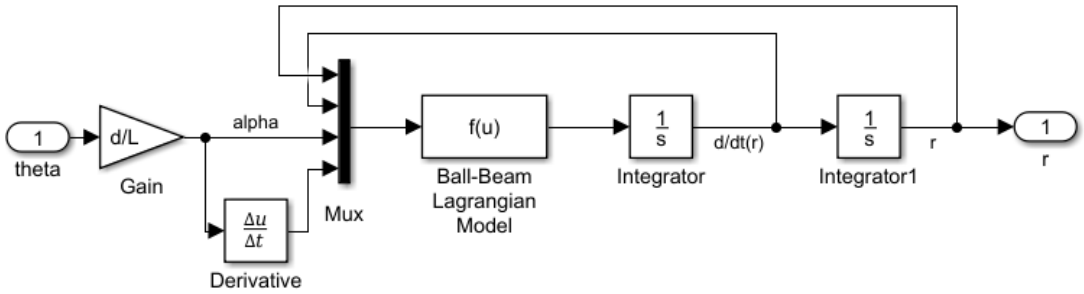


Figura 31. Sistema bola balancín. Ángulo-posición

Donde la función 'f' representa la Lagrangiana del modelo, en función de 'u', que es el ángulo de la viga.

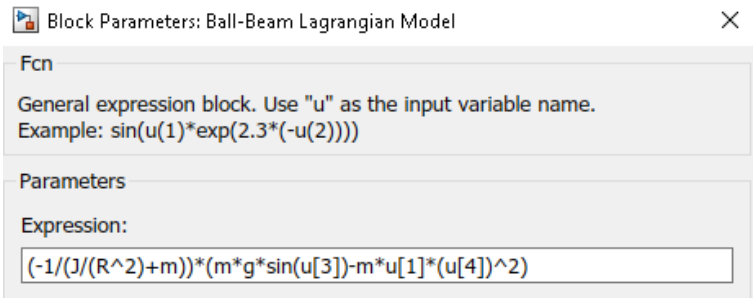


Figura 32. Función Lagrangiana del sistema

Introducimos en Matlab unos ajustes previos para las simulaciones posteriores.

```
>> nPts= 1e3;
>> t= linspace(0, 40, nPts)';
>> U= ones(nPts,1);
```

Se procede a abrir el sistema de simulink y simula para obtener la respuesta.

```
>> open_system('balancin_sistema');
>> [tsal, ~, sal]= sim('balancin_sistema', t, [], [t,U]);
>> plot(tsal,sal)
```

Desplegándose la siguiente figura de la salida del sistema, cuya respuesta es inestable como cabía esperar.

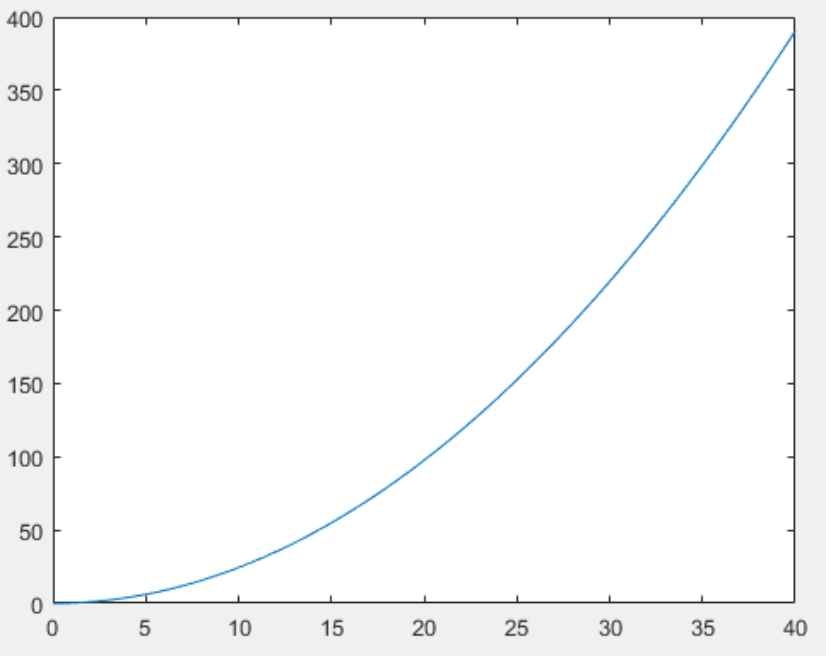


Figura 33. Respuesta del sistema en espacio de estados

Pedimos que nos genere los valores de las matrices del espacio de estados a partir del modelo, las cuales resultan las mismas que introducimos manualmente cuando generamos el modelo en espacio de estados en el anterior apartado.

También, solicitamos que nos pase de espacio de estados a función de transferencia a raíz de las matrices.

```
>> [A,B,C,D] = linmod('balancin_sistema')
>> [num,den] = ss2tf(A,B,C,D);
>> ball_ss = ss(A,B,C,D);
>> figure
>> lsim(ball_ss,U,t)
```

Ahora vamos a intentar controlar el sistema. Proponemos dos polos para llevar nuestro sistema a una zona estable y creamos con el comando place() una matriz de ganancia de realimentación de estado K, de estructura 1x2.

```
>> p1 = -2;
>> p2 = -2.5;

>> K = place(A,B,[p1,p2])
>> t = 0:0.01:5;
>> u = 0.25*ones(size(t));
>> sys_cl = ss(A-B*K,B,C,D);
>> [y,t,x] = lsim(sys_cl,u,t);
>> plot(t,y)
```

Donde la matriz K resultante se puede ver en la ventana de comandos:

K =

10.2381 9.2143

La simulación nos resulta lo siguiente:

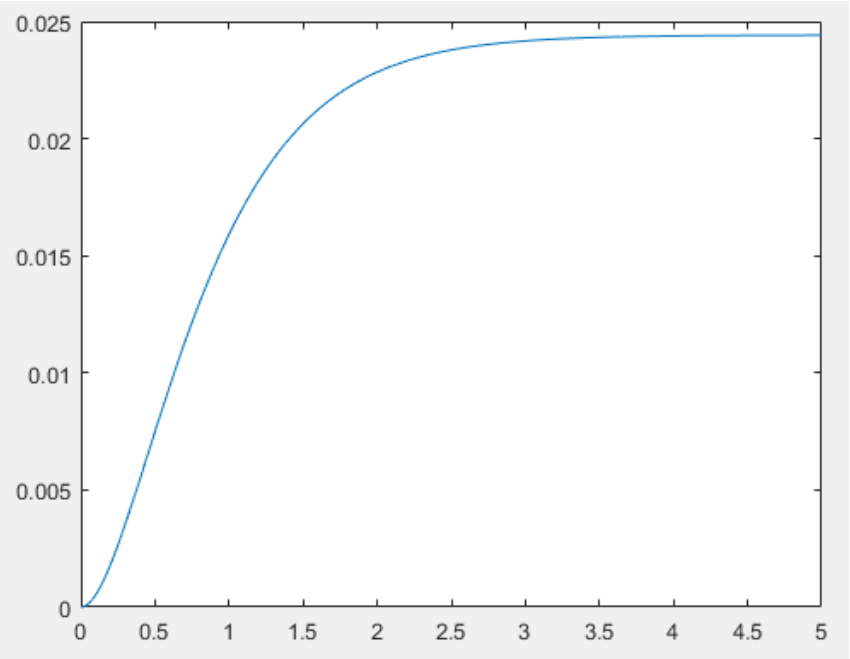


Figura 34. Respuesta del sistema con realimentación de estados.

Vemos que la salida tiende a ser estable, aunque la respuesta puede ser mejorable. Ahora queremos eliminar el error estacionario para cualquier magnitud de entrada que impongamos.

A diferencia de un PID, donde se realimenta la salida y se resta a la señal que queremos como referencia para calcular el error, en un controlador por espacio de estados realimentamos ambos estados.

Es imprescindible calcular cual debería ser el valor del estado, multiplicarlo por la ganancia elegida K y usar un nuevo valor como referencia para calcular la entrada. Esto se puede realizar añadiendo una ganancia constante seguidamente de la referencia. Lo mostraremos a continuación.

```
>> u=ones(size(t));
>> Nbar=rscale(ball_ss,K)
>> [y,t,x] = lsim(Nbar*sys_cl,u,t);
>> plot(t,y)
```

Resultando la ganancia en la referencia:

```
Nbar =

  10.2381
```

Esta ganancia Nbar se genera a partir de la función rscale(), que eliminar el error estacionario a partir del factor de escala, la proporción de lo que se aleja la salida respecto al valor deseado

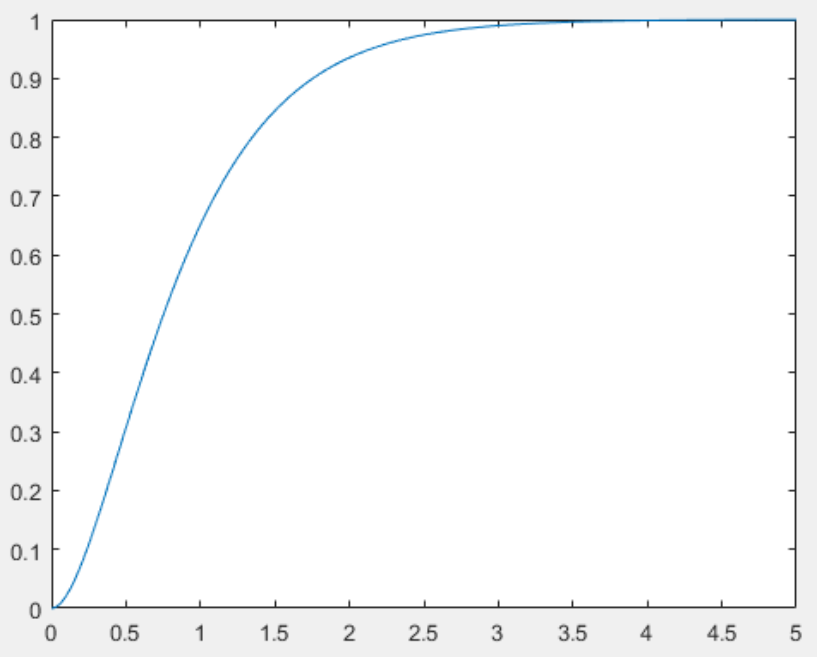


Figura 35. Respuesta del sistema con realimentación de estados y ganancia en la entrada

El error estacionario ha sido eliminado y obtenemos la salida en un tiempo razonable y sin sobreoscilaciones. Hemos logrado controlar el sistema mediante la realimentación de estados de una forma teórica con las herramientas de Matlab.

Cabe destacar que un problema de diseño no tiene necesariamente una única respuesta. Usando este método o cualquier otro, podríamos obtener múltiples resultados diferentes igual de válidos.

Capítulo 6
Descripción del equipo

Arduino

La comunicación entre la planta y el equipo se tiene que realizar mediante una herramienta que nos permita controlar los parámetros y observar los resultados en tiempo real. Arduino cumple estos requisitos, es una plataforma abierta que facilita la programación de un microcontrolador y una de las más utilizadas en el área de desarrollo de prototipos para la robótica.

Para la realización de este proyecto se ha empleado un Arduino Mega 2560 Rev3. Esta placa tiene un potente procesador AVR ATMEGA2560 con un suficiente espacio de memoria como son 256 kB de Memoria Flash y 8 kB de Memoria RAM.

Arduino es ideal para pequeños y medianos proyectos de robótica y una de las razones es por el gran número de conexiones de entrada y salida que soporta. Este dispositivo tiene 54 entradas y salidas digitales, de las que 15 pueden ser usadas como PWM. Además, tiene 16 entradas analógicas, 4 UARTs (puerto serie de hardware), un oscilador de cristal de 16 MHz y una conexión USB. La conexión con el ordenador se realiza mediante puerto serie y la propia placa dispone de un conversor serie-USB para que se pueda conectar fácilmente al ordenador.

Figura 36. Arduino Mega 2560 rev3

Esta plataforma tiene como objetivo acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinares y se ha expandido rápidamente gracias a su plataforma hardware con diseño abierto, su entorno desarrollado de código abierto, su lenguaje fácil de aprender basado en librerías, un entorno de desarrollo integrado (IDE) también con código abierto y que está disponible y se puede trabajar desde Windows, Mac y Linux.

Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de estar conectados a un ordenador, ya que una vez cargado el código con sus respectivas librerías en la placa ya solo será necesario conectar el dispositivo a la placa.

Pulse-Width Modulation. PWM

La modulación por ancho de pulsos o PWM (pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicación o para controlar la cantidad de energía que se envía a una carga.

Podemos modificar el tiempo de una señal, o ciclo de trabajo, para simular una señal analógica El ciclo de trabajo es la relación del tiempo que permanece la señal en estado activo y el periodo completo de la señal. Cabe destacar que el periodo completo de la señal siempre permanece constante.

El PWM tiene numerosas aplicaciones, pero la que nos interesa en este caso es la capacidad para controlar el ángulo de giro de un servomotor, aplicando más o menos potencia. También, nos va a ser útil para controlar el nivel de tensión que introducimos a un Pin para establecer un nivel de contraste en un Display, lo que tradicionalmente se realiza con un potenciómetro y una entrada de 5 voltios pero que, para ahorrarnos un componente físico, se puede realizar mediante una señal PWM, que veremos más adelante.

En la siguiente imagen se ilustra como se puede variar el ángulo de giro de un servomotor cambiando el ciclo de trabajo según el porcentaje de la señal en nivel alto.

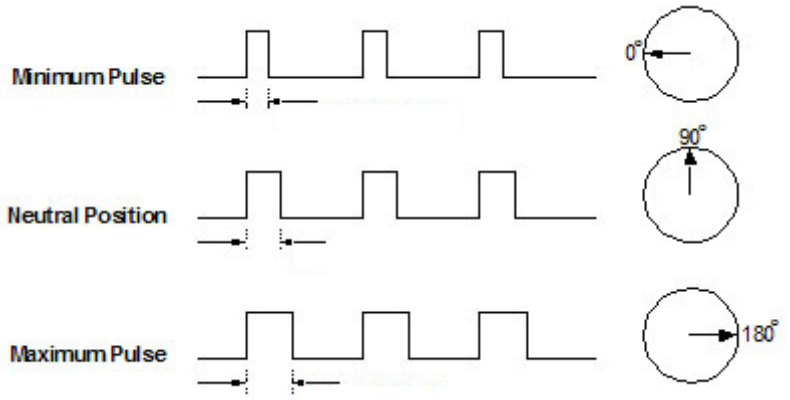


Figura 37. Diferentes ciclos de trabajo de una señal PWM

Esta herramienta es de gran utilidad ya que estamos alimentando la señal con una tensión fija de 0 o 5 V y podemos controlar una variable con valores intermedios de su máximo rango 0-255.

Sensor

Para captar la posición de la bola se requiere de un sensor, definiéndose este, como un dispositivo diseñado para recibir información de una magnitud del exterior y transformarla en otra magnitud, normalmente eléctrica, y que posteriormente seamos capaces de cuantificar y manipular.

Para medir la posición en cada instante de la bola tenemos una gran variedad de componentes en el mercado. Hay diferentes modos de captar la posición de la bola, pero principalmente estos métodos se diferencian en la precisión, la linealidad y el precio. Las opciones varían entre sensores de ultrasonidos u ópticos, donde vamos a separar estos últimos entre los que miden la distancia por TOF o mediante triangulación.

Sensores ultrasonidos

Es un tipo de sensor que utiliza las propiedades de propagación del sonido para medir longitudes. En concreto utiliza los ultrasonidos, un tipo de ondas sonoras que se encuentra por encima del espectro audible por los seres humanos.

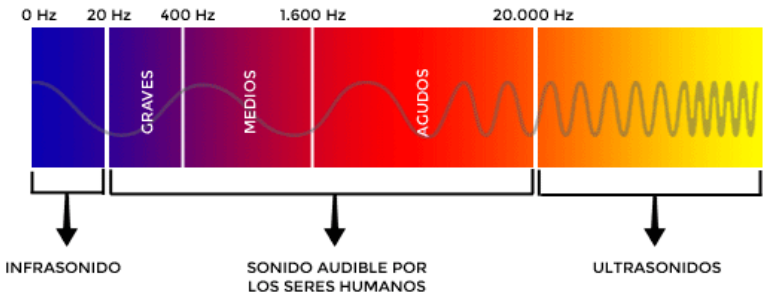


Figura 38. Rango de frecuencias de diferentes tipos de sonidos

Su funcionamiento consiste en enviar una onda ultrasónica a través del disparador o trigger, esta rebota contra el objeto y el receptor o echo detecta la onda. Midiendo el tiempo que ha tardado en viajar dicha onda, podemos conocer la distancia.

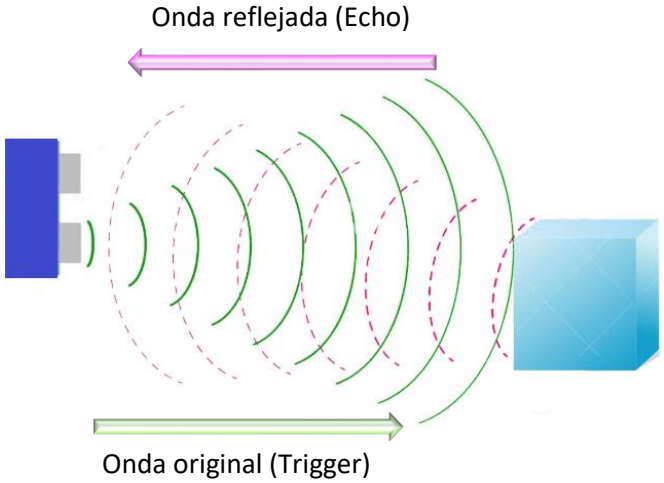


Figura 39. Funcionamiento de la medición de distancia de un sensor de ultrasonidos

Son sensores baratos y sencillos de incorporar. Sin embargo, también son los menos precisos. Puede haber casos en los que la superficie refleje la onda falseando la medición, o que en un entorno con numerosos objetos se produzcan rebotes provocando ecos. Además, tampoco es adecuado si se usa más de uno porque se pueden producir interferencias entre ellos, aunque incluso no se crucen sus haces de acción.

Sensores infrarrojos

Medición mediante TOF

Este tipo de elementos tienen un sensor infrarrojo láser. Pueden operar con elevada luz ambiental infrarroja y suelen incorporar un sistema de compensación de la medición que lo permite hacer funcionar incluso detrás de un cristal protector.

Los ecos, la reflectancia de los objetos y las condiciones ambientales no afectan de forma significativa a su funcionamiento.

Por otro lado, el ángulo de acción es más estrecho, lo cual puede resultar una ventaja si el objeto a medir está justo en frente del sensor, aunque una inconveniente en otros casos, como en la detección de obstáculos.

Sin embargo, la mayor diferencia con los sensores de distancia de ultrasonidos tradicionales consiste en la forma en tomar la medición. Estos sensores TOF (Time

of flight) envían un pulso de luz infrarroja y miden el tiempo que tarda el haz en rebotar y volver al emisor.

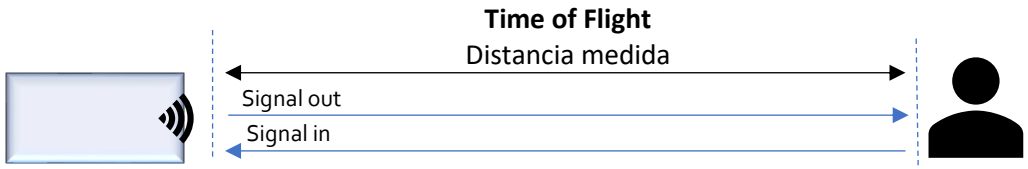


Figura 40. Funcionamiento de la medición de la distancia mediante TOF

El rango de medición depende de las condiciones del entorno, de las características del objetivo y del modo de funcionamiento.

La comunicación se realiza mediante el bus I2C, con una señal de reloj (SCL) y otra para el envío de datos (SDA).

Medición mediante triangulación

Este tipo de sensores están compuestos por un LED infrarrojo junto al dispositivo detector de posición (PSD) y un procesador integrado encargado de realizar el cálculo de la distancia.

El sensor barre el entorno y escanea en tiempo real los objetos situados en frente suyo y proporciona una salida en forma de un rango de tensión analógica.

El emisor LED produce un haz de luz infrarroja con una determinada longitud de onda. Esta luz es pulsada para aminorar la influencia de las características ambientales y cromáticas del objeto en cuestión.

El detector de posición PSD (posittion sensitive detection) es en realidad un sensor de pequeñas dimensiones CCD lineal que recibe la luz reflejada en cualquier obstáculo en dirección de la trayectoria del rayo. El detector emplea triangulación para calcular la longitud del sensor al objeto situado en frente de él.

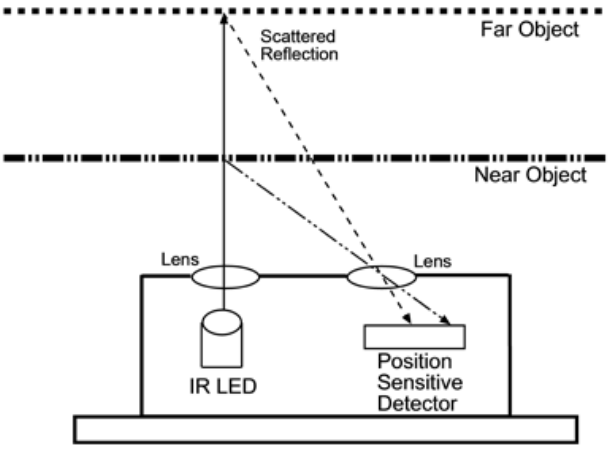


Figura 41. Funcionamiento de la medición de distancia de un sensor de infrarrojos mediante triangulación

Sharp GP2Y0A02YK0F

En nuestro caso se ha elegido el sensor de infrarrojos del fabricante Sharp, que obtendrá la salida de forma analógica mediante triangulación. Se ha seleccionado este componente por la cohesión entre su precisión, precio, disponibilidad en el laboratorio y sencillez de implementación.

Este sensor debe ser alimentado entre 4.5 y 5.5 voltios lo más estables posibles, y mide con precisión entre 10 a 80 cm, obteniendo en la salida diferentes valores de tensión entre 0.4 y 3.2 V para determinar la distancia del objeto. En la figura 42 se muestra esta relación no lineal entre salida y distancia.

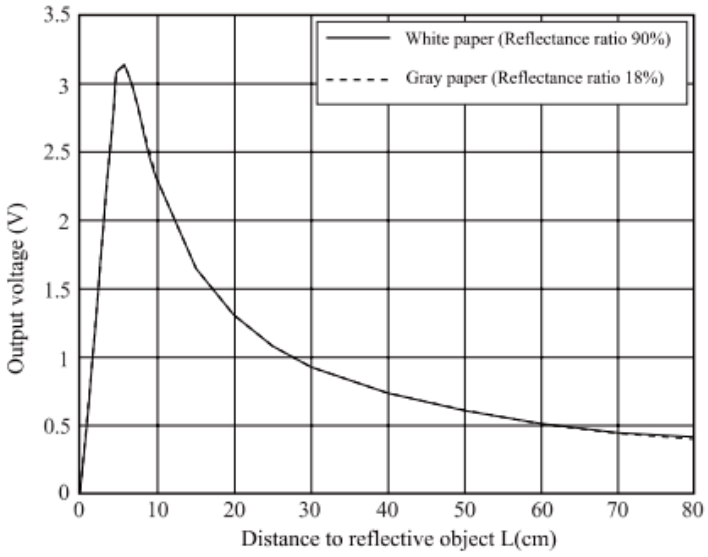


Figura 42. Gráfica de la relación entre distancia del objeto (cm) y voltaje de salida (V)

A pesar de que el datasheet muestra esta relación, Figura 10, entre la tensión y la distancia del objeto, en realidad, únicamente nos sirve como referencia. En la planta real, esta curva va a variar en función de más factores, como la luminosidad ambiente, la capacidad del objeto de absorber la luz, la humedad o la temperatura. Por lo que tenemos que obtener nuestra característica del sensor personalizada para nuestro sistema.

Filtro del sensor

El sensor analógico no presenta por defecto una gran precisión y es muy sensible a ruidos. Una forma de mitigar esto es mediante el diseño de un filtro para conseguir una salida más regular y que el ruido no nos condicione la señal que obtenemos.

A la hora de realizar un filtrado de la señal, para tener un valor regular y uniforme en la salida, tenemos varias opciones.

Una de ellas consiste en incorporar un condensador a nivel de hardware entre las patillas de la señal y tierra del propio sensor. Con un condensador de 47 µF conseguimos reducir sensiblemente el ruido electrónico a la hora de tomar el valor analógico equivalente a la medida. A continuación, se muestra la señal obtenida con/sin el condensador.

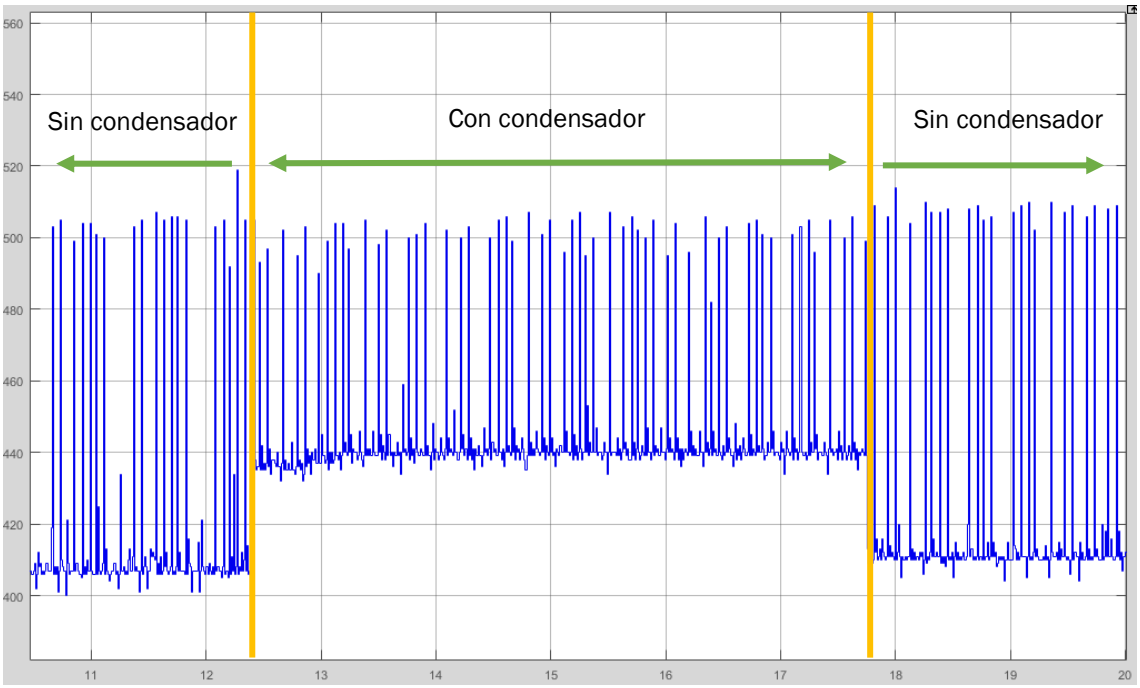


Figura 43. Señal obtenida con condensador de 47 µF y sin él

La señal también puede ser filtrada mediante software, realizando una función que, a partir de los valores analógicos obtenidos, consiga una serie de medidas más regulares con menor ruido para trabajar en las siguientes etapas de forma más adecuada. A continuación, se han planteado dos alternativas que se distinguen en la complejidad y, por tanto, la cantidad de recursos y el tiempo de procesamiento.

La primera opción es tomar 4 medidas consecutivas del sensor almacenándolas en un vector y realizar una media aritmética de todas ellas, para reducir la incidencia del ruido y obtener un valor promedio con el que trabajar. Figura 46, señal naranja

```
function sensorfilter = filter(sensor)

sensorfilter=sum(sensor);
sensorfilter=sensorfilter/4;

end
```

Figura 44. Código que hace la media aritmética de los últimos valores medidos

Otra opción es obtener 4 medidas consecutivas del sensor, almacenarlos en un vector, ordenar los valores, eliminar ahora el mayor y menor, y con los 2 centrales, hacer la media aritmética, que será el valor con el que trabajaremos. De esta forma, se eliminan los valores situados en los extremos y que se pueden considerar anómalos, obteniendo un valor de la señal más fiable y preciso. Figura 46. señal azul.

```
function sensorfilter = filter(sensor)
sensorfilter=0.0;
sortedValues=[0,0,0,0];
for a=1:4 %4 medidas
   if(sensor(a)<sortedValues(1) || a==1)
      j=1; %%inserta en la primera posicion
   else
     for j=2:a
        if(sortedValues(j-1)<=sensor(a)&& sortedValues(j)>=sensor(a))
        %%j
        break;
        end
     end
   end
   for k=a:-1:j+1 %movemos los valores una posición
      sortedValues(k)=sortedValues(k-1);
   end
   sortedValues(j)=sensor(a); %se inserta lectura actual
end
for a=2:3 %hacemos media de los intermedios
   sensorfilter=sensorfilter+sortedValues(a);
end
sensorfilter=sensorfilter/2;
end
```

Figura 45. Código que genera un vector con las últimas medidas, elimina los valores extremos y hace la media de los intermedios

En la siguiente figura se comparan los dos tipos de alternativas de las figuras 44 y 45 mediante filtrado por software.

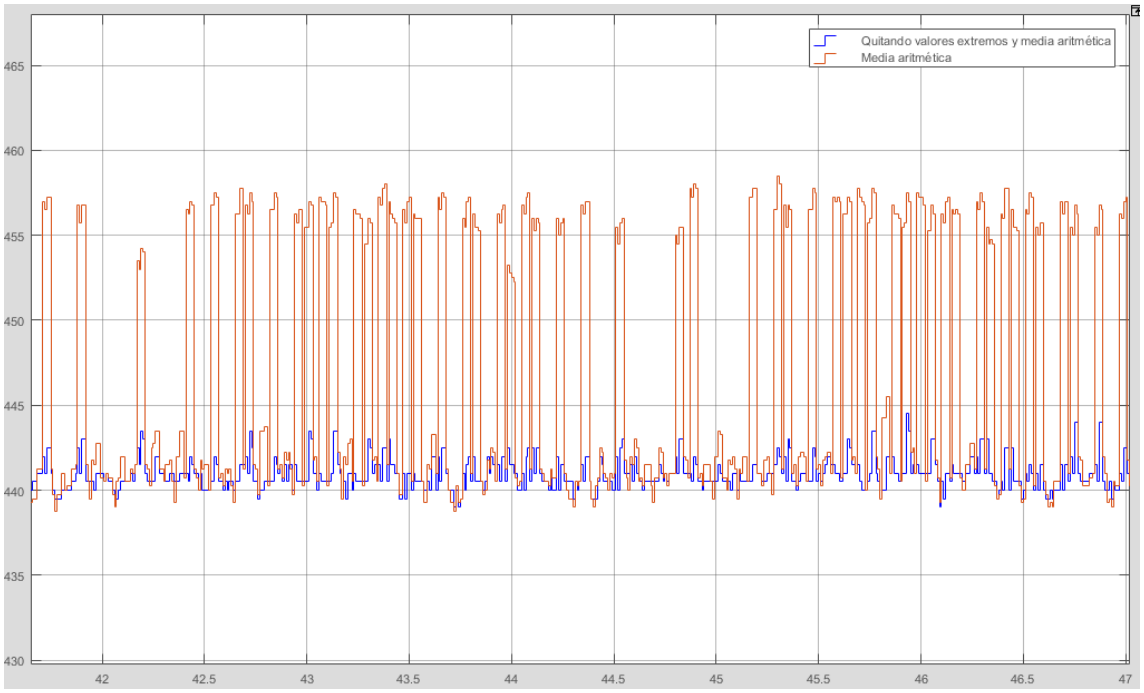


Figura 46. Comparativa de la señal con la media aritmética (naranja) y la media aritmética quitando los valores extremos (azul)

Debemos tener en cuenta que este tipo de filtrado por software se realiza tomando medidas consecutivas del sensor, lo que tiene como gran inconveniente el hecho de generar un retraso acumulativo en la ejecución del programa mientras la planta se está moviendo, es decir, podemos estar midiendo la posición de la bola, pero cuando la procesamos, la posición ha cambiado.

Lograr el equilibrio entre el número de muestras que tomamos del sensor y la correcta ejecución del programa será un aspecto fundamental y que se ajustará experimentalmente.

Caracterización del sensor

Una vez que tenemos la capacidad de leer los valores del sensor con el menor ruido posible llega el momento de obtener nuestra ecuación característica del sensor, entendiendo esta como la ecuación que relaciona la entrada con la salida.

Lo primero que vamos a hacer es tomar una serie de medidas del sensor en distintas posiciones. Colocamos la bola manualmente en posiciones arbitrarias de la viga y anotamos en Excel la señal obtenida. Tabla 7.

Medida (cm)	Salida sensor
49,3	263,5
40,3	315
31,6	398,3
21	519

Tabla 7. Tabla de medidas y su distancia (cm) obtenidas del sensor experimentalmente

Seleccionando estos datos generamos el gráfico de dispersión de la siguiente manera: Insertar>Gráfico>Dispersión. A continuación, seleccionamos un punto de la gráfica, click derecho>agregar línea de tendencia.

Entre las opciones posibles, elegimos la que mejor se adapte en nuestro rango de medidas y obtenga menor error cuadrático medio R^2, en nuestro caso, línea logarítmica. El resultado se muestra a continuación.

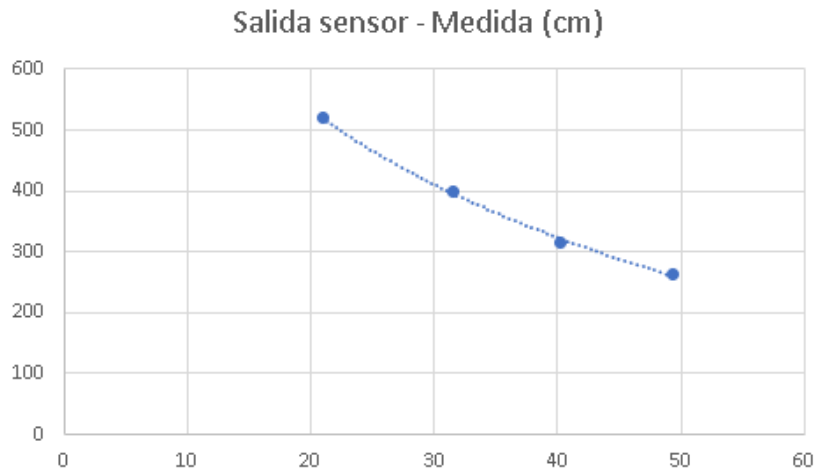


Figura 47. Línea característica del sensor con los puntos y sus medidas tomadas experimentalmente

A su vez, también marcamos la casilla 'Presentar ecuación en el gráfico', para obtener directamente la ecuación que relaciona la tensión con la distancia.

$$Señal = -303.5 \ln(Distancia) + 1443.1 \quad (58)$$

En nuestro caso nos interesa obtener la distancia en función de la tensión, ya que vamos a trabajar directamente con la distancia y será nuestro parámetro de control, por lo que vamos a expresar la ecuación de manera inversa.

$$Distancia = -41.09 \ln(Señal) + 277.61 \quad (59)$$

Con esta ecuación ya podemos obtener la distancia de la bola en tiempo real según la medida que leamos del sensor.

Liquid Crystal Display (LCD)

Se ha optado por añadir un módulo en forma de pantalla LCD en el sistema con el objetivo de poder ver en tiempo real la medida de una forma más visual y estética.

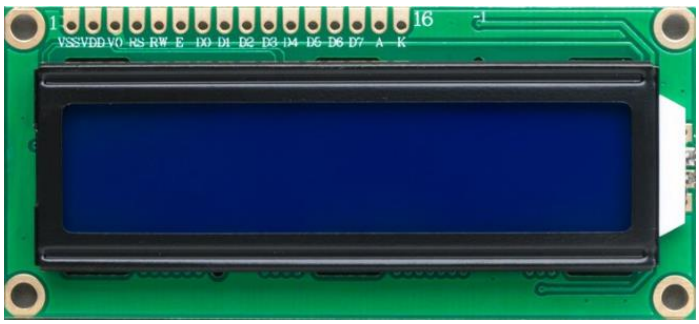


Figura 48. LCD 16x2 compatible con Arduino

Este display es 16x2, es decir, tiene 16 caracteres en 2 filas. Contiene una pantalla de cristal líquido (LCD: liquid crystal display) donde sus principales características y ventajas son el bajo precio, su bajo consumo y la capacidad para mostrar caracteres de texto y números.

A continuación, se va a mostrar la tabla de pines del módulo LCD y su conexión correspondiente con Arduino.

PIN LCD	Conexión Arduino
1. VSS	GND (Tierra)
2. VDD	5 Voltios
3. Vo - Control de contraste	Pin PWM. Pin 7
4. RS- Selector entre comandos y datos	Pin 12
5. RW – Escritura y lectura de comandos y datos	GND (Tierra)
6. E -Sincronización de lectura de datos	Pin 11
7-10 D0-D3 – Pines de datos de 8-bit	Sin conexión
11-14 D4-D7 - Pines de datos de 8-bit	Pines 5-2
15. A – Alimentación luz de fondo	5 Voltios
16. K – Luz de fondo	GND (Tierra)

Tabla 8. Correspondencia de cada pin de la LCD con su conexión de Arduino

Cabe destacar que el Pin 3 del LCD se puede conectar a un potenciómetro que regule el contraste variando la resistencia o, en su defecto, conectándolo a una señal PWM que regula la tensión que determina el contraste, con idéntico resultado. En nuestro caso se ha optado por esta segunda opción, regulando el contraste mediante un bloque Simulink y la patilla 7 PWM del Arduino, poniendo un valor de 65 sobre 255.

La mayor complejidad viene en el hecho de trabajar con el Display en Simulink, ya que no encontramos una librería íntegramente dedicada a trabajar con este componente para esta plataforma, por lo que se ha optado por generar un nuevo bloque mediante la utilidad S-Function, que se alimenta de los archivos .h y .cpp de la librería ya existente para el sketch de Arduino, con sus pertinentes modificaciones. A continuación, se explica su creación.

Bloque S-Function

Este bloque sirve de interfaz entre Simulink y Matlab, permitiendo utilizar funciones programadas en nuestro modelo.

Una S-function puede estar programada en C o en Matlab, lo que nos permitirá incorporar la librería de Arduino y adaptarla para Simulink. Para ejecutarla, tiene que ser compilada como MEX-file, que se puede hacer mediante la orden ‘mex archivo.c’.

Las S-Function escritas en C, como es nuestro caso, se dividen en tres partes, una para inicializar, otra que se ejecutará de manera periódica durante la simulación y una que se ejecutará para finalizar correctamente la simulación.

En la inicialización se declara el funcionamiento del comienzo, número de entradas y salidas y el tiempo de muestreo del bloque. En nuestro caso solamente tenemos una salida, de tipo uint16, que es la señal que mandamos a la lcd para que sea escrita en el display.

A continuación, se van a explicar de forma resumida los pasos seguidos para crear esta función.

Lo primero es copiar los dos archivos de la librería de Arduino 'LiquidCrystal.cpp' y 'LiquidCrystal.h' y pegarlas en la carpeta donde vamos a trabajar. Posteriormente creamos el bloque S-Function en nuestro modelo, que vamos a guardar como 'lcd', y lo configuramos como viene detallado en el Anexo, utilizando una estructura similar al sketch de Arduino, pero con modificaciones relevantes.

Una vez realizado esto, construimos el bloque haciendo click en 'Build'. Vamos a Matlab y abrimos el archivo que se ha creado llamado 'lcd_wrapper.c'. En este archivo nos dirigimos a las funciones void y escribimos delante de ellas: ' extern "C" ', para que la función pueda referirse a ellas y a los archivos de la librería. Por último, cambiamos el nombre del archivo 'lcd_wrapper.c' por 'lcd_wrapper.cpp'. Nuestra función ya está lista.

Actuador

Se define un actuador como un dispositivo con la capacidad de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control. Son los elementos que influyen directamente en la señal de salida, modificando su magnitud según las instrucciones que reciben de la unidad de control.

En nuestro proyecto, para variar la inclinación de la viga que a su vez hace rodar la bola, usaremos un servomotor.

En el mercado hay una gran variedad de componentes, que se diferencian en las dimensiones y peso de este, la alimentación, la velocidad y como característica más determinante, el torque.

El toque es una variable que hace referencia a la fuerza necesaria que se ha de aplicar a un objeto para hacer que este rote. Como se puede ver en la figura 49, un objeto que gira sobre un eje 'O', necesita una fuerza 'F' a una distancia 'r' desde el punto de giro al lugar donde se aplica la fuerza para efectuar la rotación. La relación entre estos parámetros es el torque.

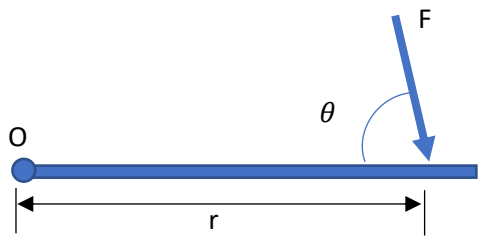


Figura 49. Torque de una viga sobre un eje de giro

Cuya fórmula se define a continuación:

$$Torque = r \cdot F \cdot sen(\theta) \quad (60)$$

En nuestro proyecto y en el área de los servomotores, el torque se explica cómo el momento de fuerza que ejerce el motor sobre el eje de transmisión de potencia o, la tendencia de una fuerza para girar un objeto alrededor de un eje, punto de apoyo, o pivote.

Por tanto, el torque de nuestro servomotor nos indica el peso que puede mover en función de la distancia o longitud de la viga.

Tower Pro MG995

Este modelo de la marca Tower Pro ha sido el elegido para la realización del proyecto por su relación entre características técnicas, económicas y disponibilidad en el laboratorio. Este servomotor puede girar 180 grados (90 a cada lado) y su pequeño tamaño hace que sea fácilmente acoplable a nuestra maqueta.

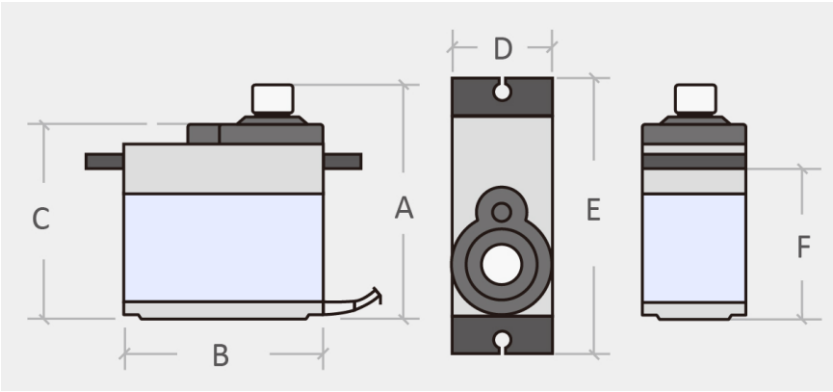


Figura 50. Medidas del Servomotor MG995

A continuación, la tabla con las características principales.

Peso (g)	55
Torque (kg) (4.8V)	8.5
Torque (kg) (6V)	12
Speed (sec/60deg)	0.2
A(mm)	42.7
B(mm)	40.9

C(mm)	37
D(mm)	20
E(mm)	54
F(mm)	26.8

Tabla 9. Características principales del servo y su valor

Este componente tiene tres pines, que corresponden a alimentación, masa y la señal. En el datasheet podemos ver el esquema que identifica cada conexión, figura 51, donde el color de cada cable nos facilita la tarea de saber a qué corresponde cada pin.

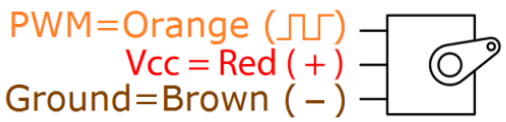


Figura 51. Conexiones de un servomotor

Fuente de alimentación

Una cuestión a la que nos enfrentamos en la realización del sistema es la correcta alimentación de los componentes que actúan en la planta real, principalmente del actuador, que requiere de más corriente para poder mover la estructura.

Durante varios ensayos se ha utilizado el cable USB de Arduino que transmite 5 voltios para alimentar el servomotor, pero en ocasiones se nota la carencia de potencia. Para solucionar este problema se ha incorporado una fuente de alimentación externa que se conectará directamente a la red eléctrica a 220 voltios de corriente alterna y podremos trabajar a su salida con un valor constante de 5 voltios y 3 amperios, suficiente para poder trabajar con el servomotor y el resto de los componentes que necesitemos conectar a ella.

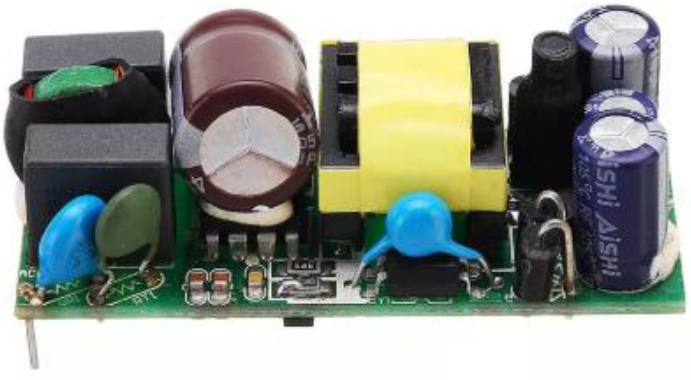


Figura 52. Fuente de alimentación con salida a 5 Voltios y 3 amperios

Interruptor eléctrico

Se ha añadido en la maqueta un interruptor de encendido/apagado junto a un led identificativo. Este elemento permite visualizar el estado de la alimentación del motor, en caso de estar en marcha o estar desconectado.

Añadir este componente resulta muy útil para conocer si estamos suministrando tensión al motor y poder apagarlo con seguridad sin tener que enchufar y desenchufar el cable a la red cada vez que queramos hacer pruebas y trabajar con él.

El led se ha elegido de color azul por la disponibilidad en el laboratorio y una resistencia limitadora de 560 ohmios, que conseguirá una vida útil del led muy larga ya que la corriente que pasa por él es realmente baja, pero suficiente para que luzca.

Resumen de componentes y conexiones

El material utilizado con todos los elementos que se han empleado en este proyecto adaptados a la estructura del sistema viga-bola se enumera a continuación.

- Placa Arduino Mega 2560 rev3.
- Cable USB tipo A/B para conectar la placa Arduino con el ordenador.
- Sensor Sharp 2Y0A21 F 06.
- Servomotor TowerPro MG995.
- Fuente de alimentación de 5 Voltios.
- Pantalla LCD 16x2
- Condensador de 47 µF.
- Interruptor eléctrico.
- Led azul.
- Resistencia limitadora de 560 Ω
- Protoboard de 7x3.5 cm. como elemento auxiliar para conectar todos los componentes además de los cables y pines necesarios.
- Fundas termoretráctiles para proteger los cables tras las soldaduras.
- Caja Retex serie 101 para colocar todos los componentes y sus conexiones. Dimensiones 190x115x75 cm.

Para facilitar la tarea visual de comprender las conexiones de todos los componentes se va a optar por poner cada esquema de conexiones de forma independiente, aunque en la práctica todos los circuitos estén conectados entre sí. De esta manera la amalgama de cables no interfiere en la capacidad de entender dónde va cada elemento y su conexión.

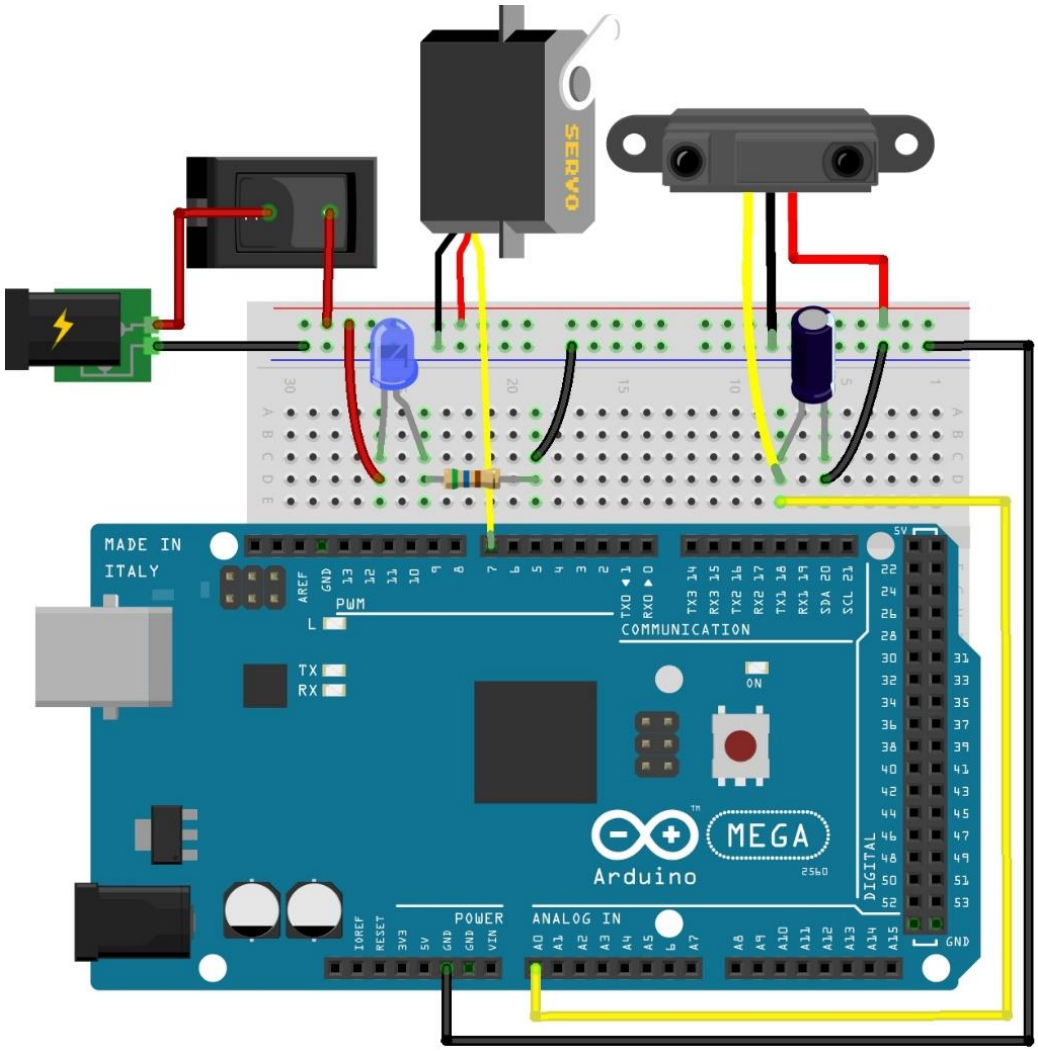


Figura 53. Esquema de conexiones del Arduino Mega 2560 con el sensor Sharp, el servomotor TowerPro, la fuente de alimentación, el interruptor y el led identificativo

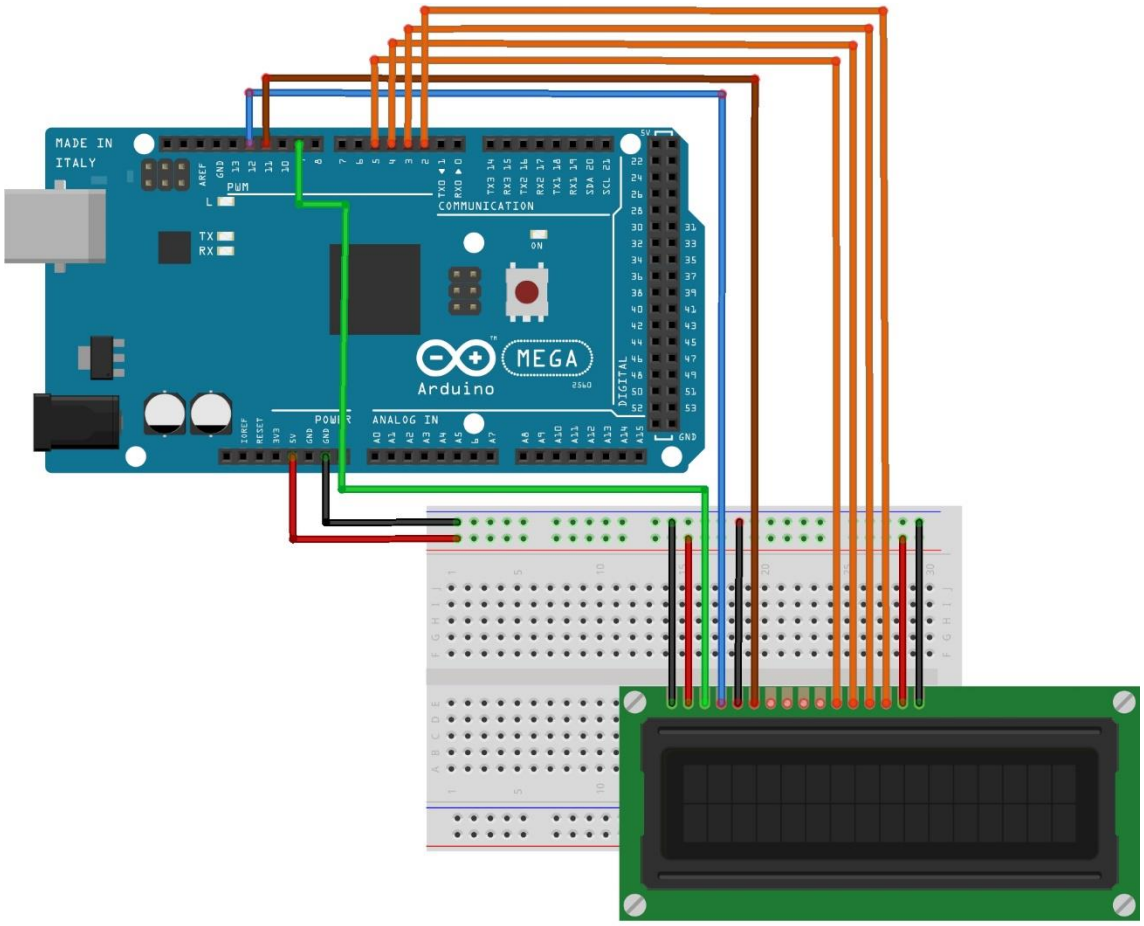


Figura 54. Esquema de conexiones de Arduino Mega 2560 con el Display 16x2

Cabe destacar, aunque no se ilustre en los esquemas de conexiones individuales, que las tomas a tierra de los diferentes componentes que se muestran en las imágenes están cortocircuitadas a la toma de tierra de Arduino, siendo esta la misma en todos los elementos.

En la figura 55 se muestra el interior de la caja con el conjunto de elementos y conexiones realizadas.

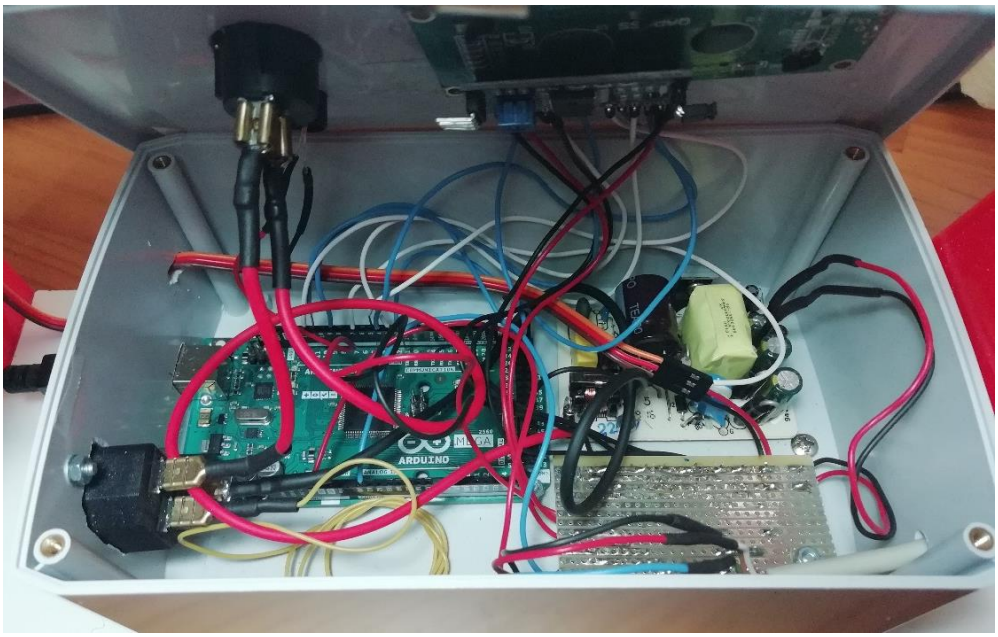


Figura 55. Conjunto de conexiones dentro de la caja contenedora

Debido a las circunstancias del proyecto, el orden y estética interior de la caja resulta mejorable, lo cual se comenta en el capítulo '8. Conclusiones y líneas futuras' al final de esta memoria.

Las soldaduras entre cables se han protegido con fundas termorretráctiles que aíslan y hacen más duraderas las conexiones.

Mecanizado de la caja

El servomotor y el sensor están comunicados con Arduino y este, con todos los elementos auxiliares que se citaron en el apartado anterior "Resumen de componentes y conexiones". Todo ello queda resguardado en una caja contenedora de medidas 190x115x75 cm. Esta caja ha tenido que ser mecanizada para hacer hueco al interruptor, el led, el espacio para que la LCD sea visible por el usuario y los cables correspondientes a la alimentación, USB, servomotor y sensor de distancia. Todo ello de la forma más eficiente y estética posible.

A continuación, se muestran los planos de mecanizado. No se ilustran de la forma normalizada según UNE ya que el objetivo es que el usuario se haga una idea del procedimiento llevado a cabo y no tanto de las dimensiones reales y su escala precisa en este documento. Las acotaciones están tomadas en milímetros.

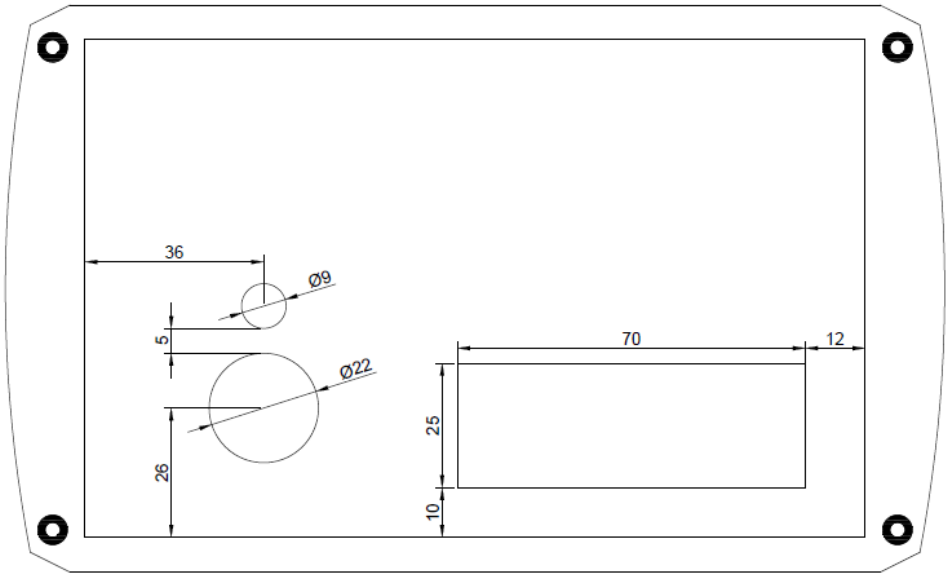


Figura 56. Planta de la tapadera de la caja. Con huecos para el led, interruptor y la LCD

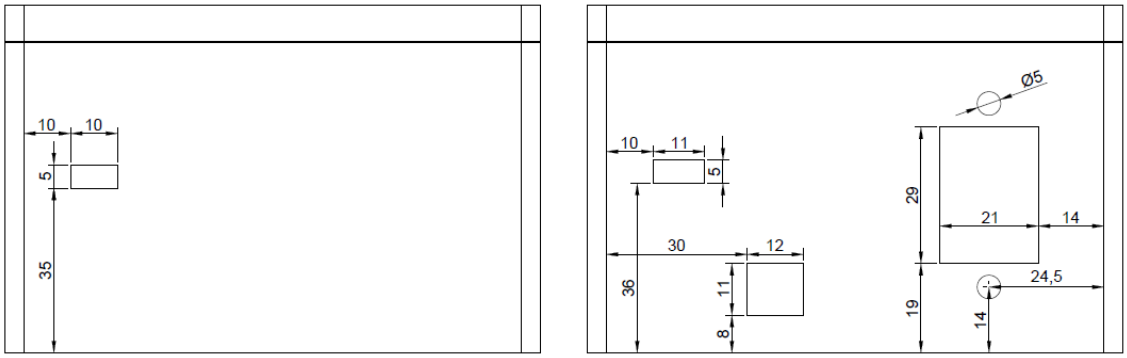


Figura 57. Perfiles izquierdo y derecho de la caja. Huecos para la manguera de cables del sensor, cables del actuador, el USB y la fuente de alimentación

Este mecanizado se ha realizado mediante taladros de diferentes diámetros y una lija para mejorar el acabado.

Capítulo 7

Implementación real del controlador mediante Simulink

Programa en Simulink

A continuación, se va a mostrar el esquema Simulink desarrollado y posteriormente, se va a explicar de forma detallada cada etapa independiente. Como se ve en la figura 58, se ha pretendido separar lo máximo posible cada proceso para que se entienda de forma clara y, en un simple vistazo, mediante colores y sombreados, se identifique cada mecanismo del programa.

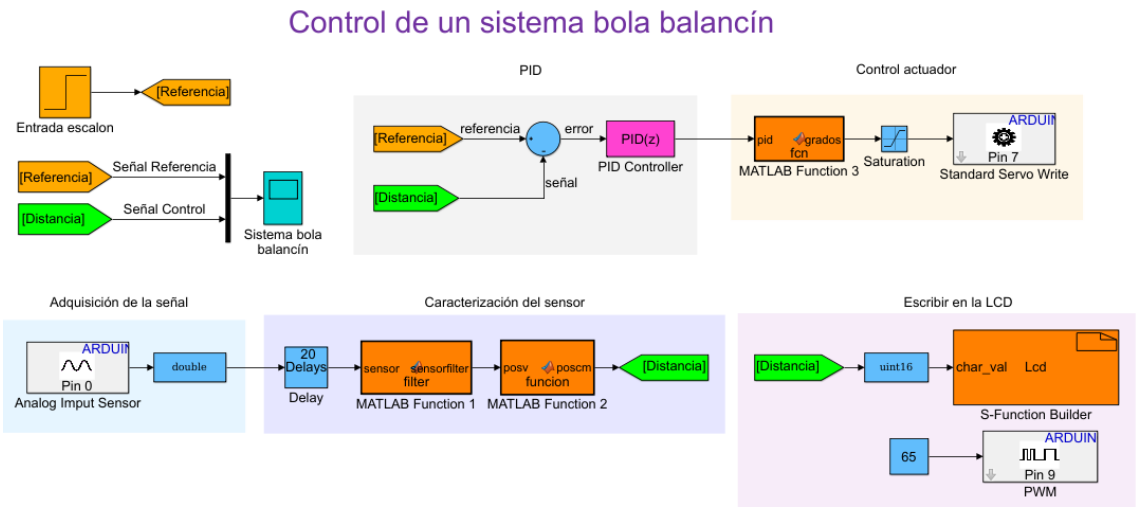


Figura 58. Diagrama de bloques del programa desarrollado en Simulink

El primer paso es incorporar un bloque correspondiente a la referencia que será introducida por el usuario. Podrá introducir un valor entre los límites de la viga que determina la posición en la que la bola permanecerá en estado estacionario. Este valor queda almacenado en la etiqueta Referencia.

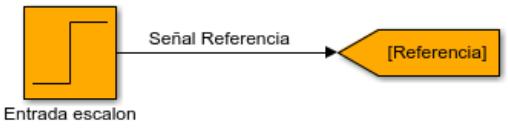


Figura 59. Diagrama de bloques de la señal de referencia mediante una entrada escalón

Para realizar el control del sistema necesitamos obtener la señal del sensor, para ello incorporamos el bloque 'Analog Input', que lee el Pin 0 de nuestro Arduino representando su señal analógica como un valor digital entre 0 y 1023. Posteriormente lo transforma a un dato double para poder trabajar con ello en los siguientes pasos y evitar posibles problemas con otros bloques.

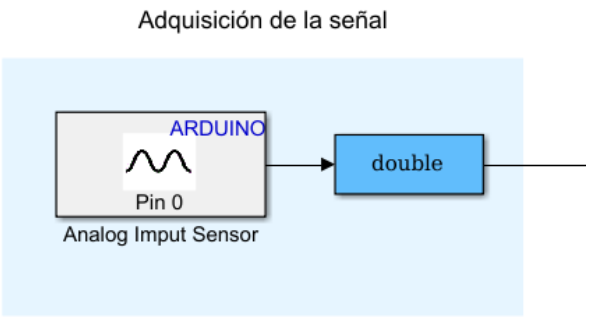


Figura 60. Diagrama de bloques de la adquisición de la señal

Una vez tenemos el valor del sensor como un nivel de tensión debemos transformarlo en distancia (en cm) para operar con él. Esta es una parte crítica del programa, ya que de la calidad de la señal que obtengamos, dependerá la precisión de nuestros resultados.

Primero, vamos a realizar un filtrado por software. Introducimos el bloque 'Delay', que toma medidas consecutivas del sensor y las guarda en un vector para ahora, realizar un filtrado con el bloque 'MATLAB Function 1' ya explicado anteriormente. Después, el nivel de tensión ya filtrado lo pasamos a centímetros caracterizando el sensor como se ha explicado mediante el bloque 'Matlab Function 2'. El valor de la medida lo guardamos en la etiqueta 'Distancia'. El código de las funciones de Matlab se refleja en las figuras 62 y 63.

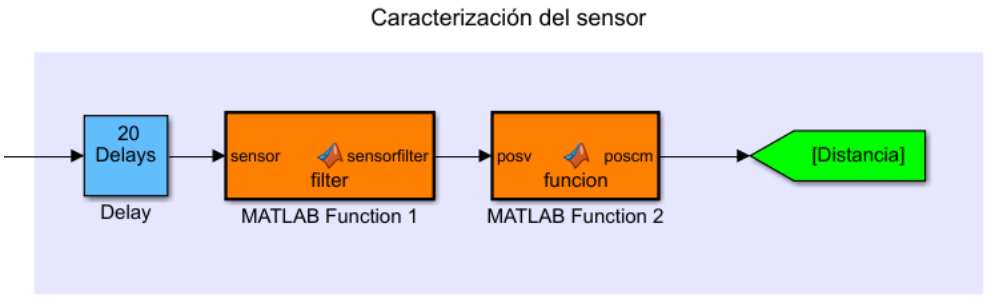


Figura 61. Diagrama de bloques de la caracterización del sensor

Matlab Function 1

```
function sensorfilter = filter(sensor)
sensorfilter=0.0;
sortedValues=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
for a=1:20 %20 medidas
    if(sensor(a)<sortedValues(1) || a==1)
        j=1; %inserta en la primera posicion
    else
      for j=2:a
          if(sortedValues(j-1)<=sensor(a)&&
sortedValues(j)>=sensor(a))
          break;
          end
      end
    end
    for k=a:-1:j+1 %movemos los valores una posición
        sortedValues(k)=sortedValues(k-1);
    end
    sortedValues(j)=sensor(a); %se inserta lectura actual
end
for a=8:12 %hacemos media de los intermedios
    sensorfilter=sensorfilter+sortedValues(a);
end
sensorfilter=sensorfilter/5;
end
```

Figura 62. Código bloque Matlab Function 1

Matlab Function 2

```
function poscm = funcion(posv)
poscm=-0.1138*posv+77.958;
end
```

Figura 63. Código bloque Matlab Function 2

Con la señal de referencia y la medida real de la planta ya estamos en disposición de trabajar con el controlador. Restamos a la señal de referencia la señal de control generando la diferencia que será el error debido al cerrar el lazo. Este valor es la entrada del controlador PID que, mediante las variables proporcional, derivativa e integral, obtendrá una señal que corregirá la posición de la viga para lograr el equilibrio estacionario deseado.

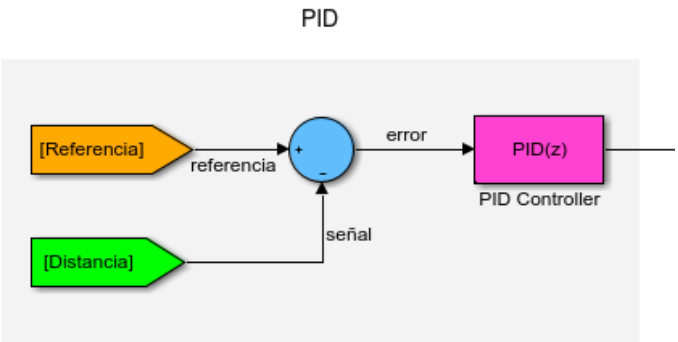


Figura 64. Diagrama de bloques del controlador PID

A continuación se muestra el interior del bloque PID, donde destaca que vamos a poner un Sample Time de 0.01. El controlador en forma Parallel , fuente de datos interna y vamos a eliminar la opción de usar un filtro derivativo.

Estamos en disposición en ajustar los parámetros P, I y D que haremos en el siguiente apartado.

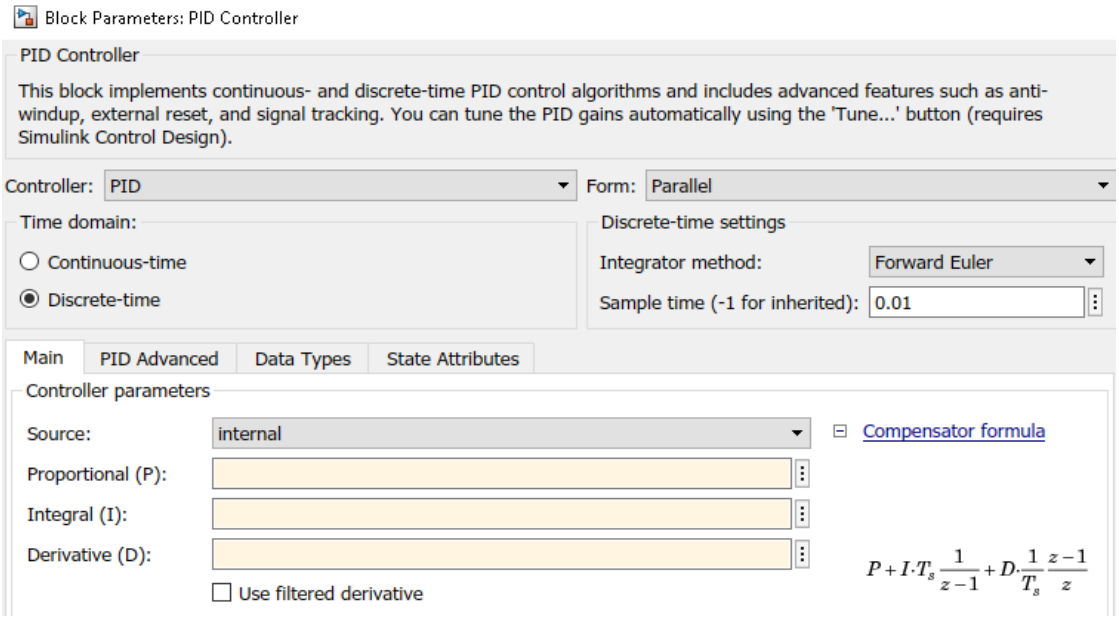


Figura 65. Bloque PID

El último paso es mandar esta señal PID al actuador, en nuestro caso el servomotor, para corregir el ángulo de la viga.

Primero, pasa por el bloque 'Matlab Function 3'. Cabe destacar que la señal PID oscila entre valores alrededor del 0 y el servomotor tiene un rango de giro entre 0 y 180 en nuestro proyecto, por tanto, parece más conveniente que la señal PID varíe

alrededor de ese valor, de 90 y estableciendo este como el punto de equilibrio. Este código se puede consultar en la figura 67.

Posteriormente esa señal se satura poniendo unos límites mínimo y máximo de 0 y 180, aunque no tendría por qué ser necesario, para asegurarnos no dañar el servomotor ni provocar un error en el programa mandando señales fuera de sus límites.

Con el bloque 'Standard Servo Write' escribimos en el Pin 7 del Arduino, donde está la señal de control del servo, el valor del ángulo en el que se posiciona el servomotor en tiempo real.

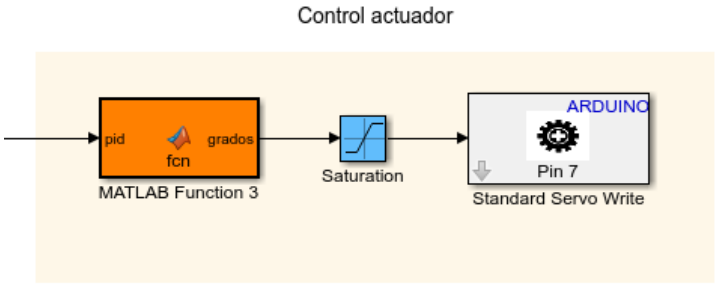


Figura 66. Diagrama de bloques del control del actuador

Matlab Function 3

```
function grados = fcn(pid)
grados = ((pid - (-180))*(180 - 0))/(180 - (-180)) + 0;
end
```

Figura 67. Código bloque Matlab Function 3

Independientemente del procedimiento para controlar el sistema, incorporamos una secuencia de bloques para escribir en la pantalla LCD y mostrar la información visualmente.

Obtenemos el valor de la distancia ya en centímetros que recogemos de la etiqueta y lo pasamos a uint16, para trabajar con un valor entero sin signo de 16 bits. Mediante el bloque 'S-Function Builder' desarrollamos una aplicación para poder comunicarnos con la LCD y visualizar el dato en tiempo real, como se ha explicado en su apartado correspondiente y cuyo código se puede consultar en el Anexo.

A su vez, mandamos un valor constante de 65 al Pin 9 PWM, que puede variar entre 0 y 255, de Arduino que corresponde al Pin Vo de contraste de la LCD. Esto supone una alternativa para controlar el contraste mediante software sin tener que incorporar un potenciómetro en la maqueta.

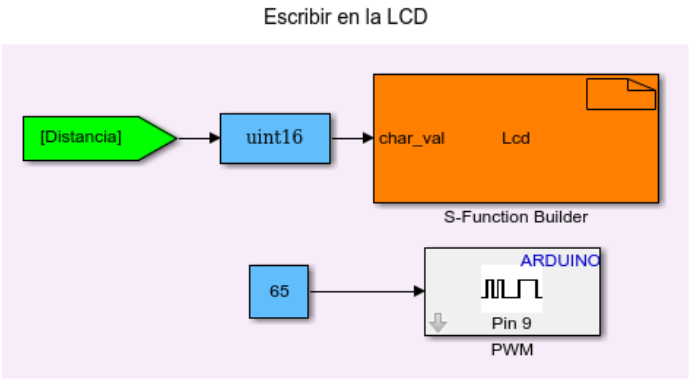


Figura 68. Diagrama de bloques del procedimiento para escribir en la LCD

En todo momento podemos visualizar la evolución de la Señal de control respecto a la Señal de referencia en una misma imagen en un Scope. El diagrama de bloques correspondiente es el siguiente.

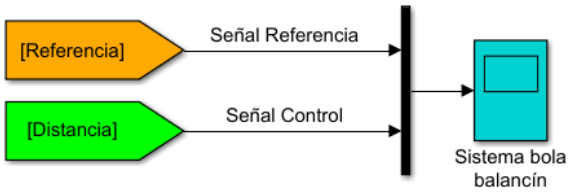


Figura 69. Diagrama de bloques de la gráfica Señal referencia - Señal control

Estamos recopilando la información de las etiquetas Referencia y Distancia para unirlas con un multiplexor de dos entradas y mostrarlas en el mismo Scope.

Sintonizado de los parámetros

Una vez tenemos todo el sistema definido, llega el momento de adaptar el controlador PID y ajustar los parámetros del mismo para lograr la estabilidad de la planta en óptimas condiciones. Lo vamos a realizar de una forma práctica y experimental, estudiando los efectos que tiene cada componente en nuestro sistema real y analizando los resultados.

Nuestro procedimiento va a ser ajustar primero la constante proporcional y posteriormente la derivativa. Y, por último, la constante integral para un ajuste fino en caso de encontrar error estacionario.

Lo primero es ajustar la **componente proporcional Kp**. Un buen valor es aquel que es capaz de mover la bola en la viga de manera que esta no alcance mucha velocidad, ya que luego sería más complicado detenerla. Este valor inclina la viga más cuanto más lejos está de la posición deseada.

Probamos con un valor de Kp=2.

Kp=2	Ki=0	Kd=0

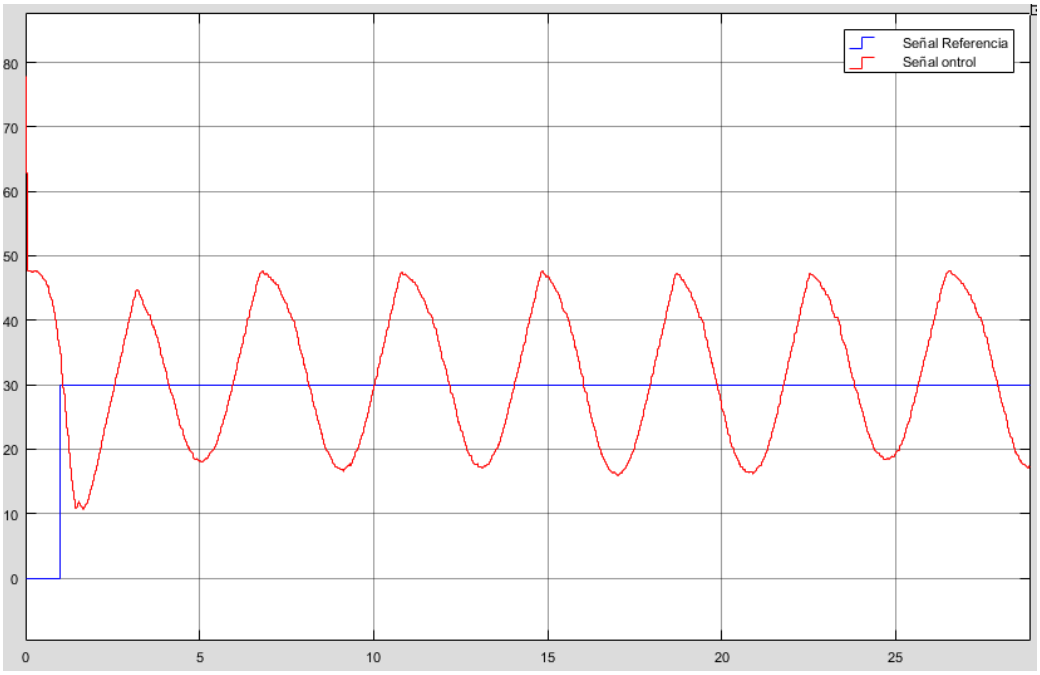


Figura 70. Sistema críticamente estable controlado solo con constante proporcional Kp

Solo con este parámetro no vamos a lograr la estabilidad del sistema. Así que procedemos ahora a **introducir la constante Kd**.

Este parámetro influye en el grado en que se diferencian las posiciones del ciclo actual y el anterior. Para conseguir un buen valor, quitamos el término proporcional para que no participe. Sabremos si un valor es insuficiente si el sistema no reacciona como para detener la bola, y si nos hemos excedido si reacciona demasiado y el sistema es inestable

Un valor de Kd=4 parece adecuado para nuestro sistema.

El control mediante un término Porporcional y Derivativo (PD) es suficiente para muchas aplicaciones. En otras, el sistema se puede mejorar con la acción integral.

Vamos a observar la respuesta del sistema mediante nuestro controlador PD.

Apreciamos, figura 71, que la bola logra pararse por completo, pero no en nuestra posición indicada. Lo que ocurre es que necesitamos del término integral para que la bola se mueva cuando apenas tiene velocidad, está cerca de la posición de referencia, pero no logra alcanzarla.

Kp=2	Ki=0	Kd=4

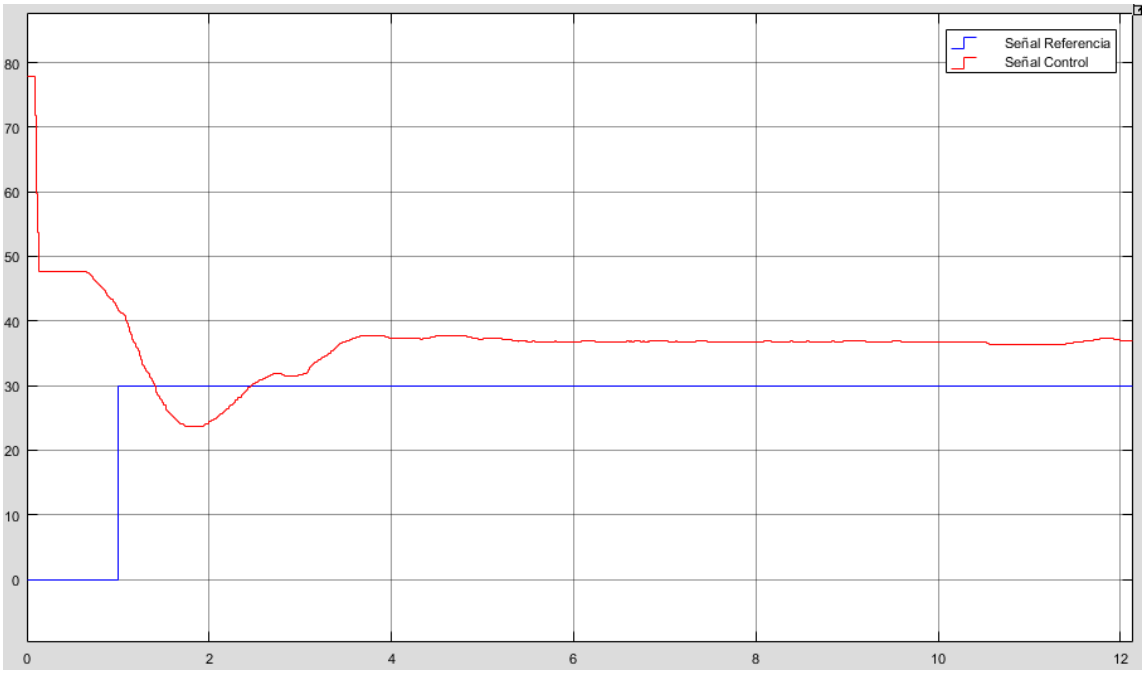


Figura 71. Sistema estable mediante un controlador PD que no alcanza correctamente la posición especificada

En este punto cabe recordar que en la planta ideal no era necesario introducir constante integral ya que la función de transferencia tiene dos polos en el origen

como vimos teóricamente y pudimos ilustrar en la figura 30, donde se consigue un buen resultado con un contralor únicamente PD.

Sin embargo, en la planta real no obtenemos la misma respuesta, como también se podría esperar, ya que las características de nuestra planta varían de las analizadas teóricamente porque influyen más constantes de las que podemos tener en cuenta. Por tanto, será necesario incluir la componente integral para obtener un buen resultado.

Introducimos a nuestro controlador la **constante integral Ki**.

Como hemos dicho, este parámetro es importante cuando la velocidad de la bola es tan baja que la viga no se inclina lo suficiente para moverla, pero en cambio, aún no está en la posición deseada. Como la velocidad es cero, el término diferencial no actúa.

Por lo que podemos definir este parámetro como de precisión, de ajuste fino. Considera la posición de la bola y cuánto tiempo lleva ahí. Hay que acotar debidamente su acción porque de lo contrario el sistema se volverá inestable.

Este es el parámetro más complejo de especificar. Un valor insuficiente no logará colocar la bola en la posición de equilibrio en un tiempo adecuado. Un valor excesivo puede crear inestabilidad en el sistema.

Probamos con:

Kp=2	Ki=0.1	Kd=4

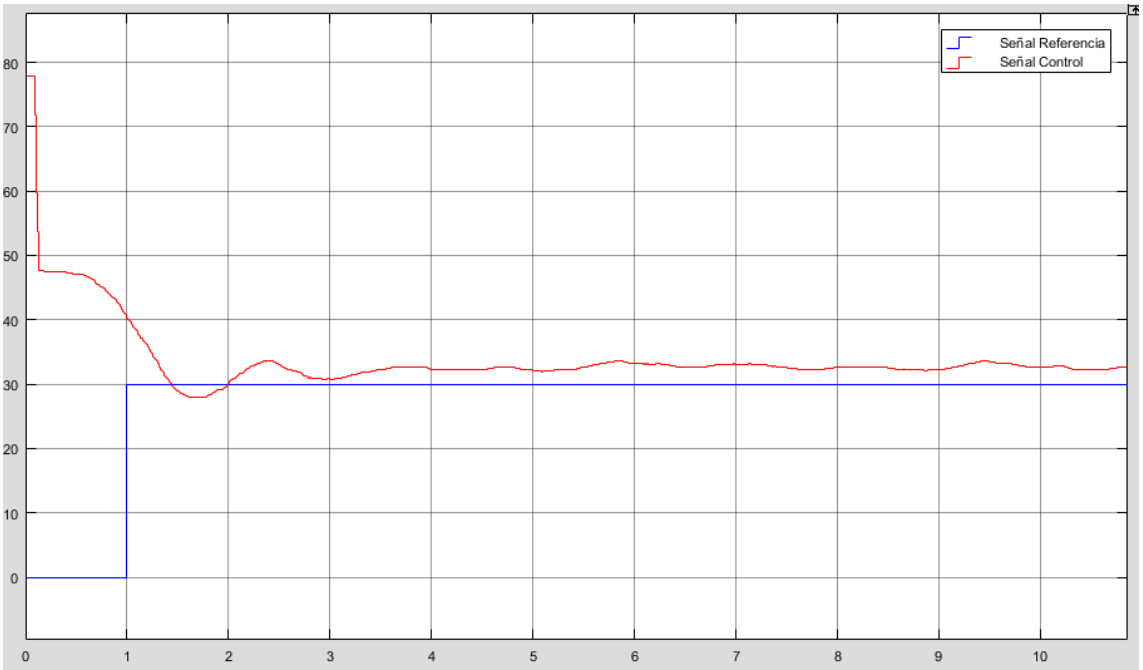


Figura 72. Sistema estable con un controlador PID que no logra alcanzar la posición especificada

Se aprecia como el valor se aproxima más al punto de equilibrio, pero aún no lo suficiente. Probamos a aumentar la constante integral Ki.

Kp=2	Ki=0.4	Kd=4

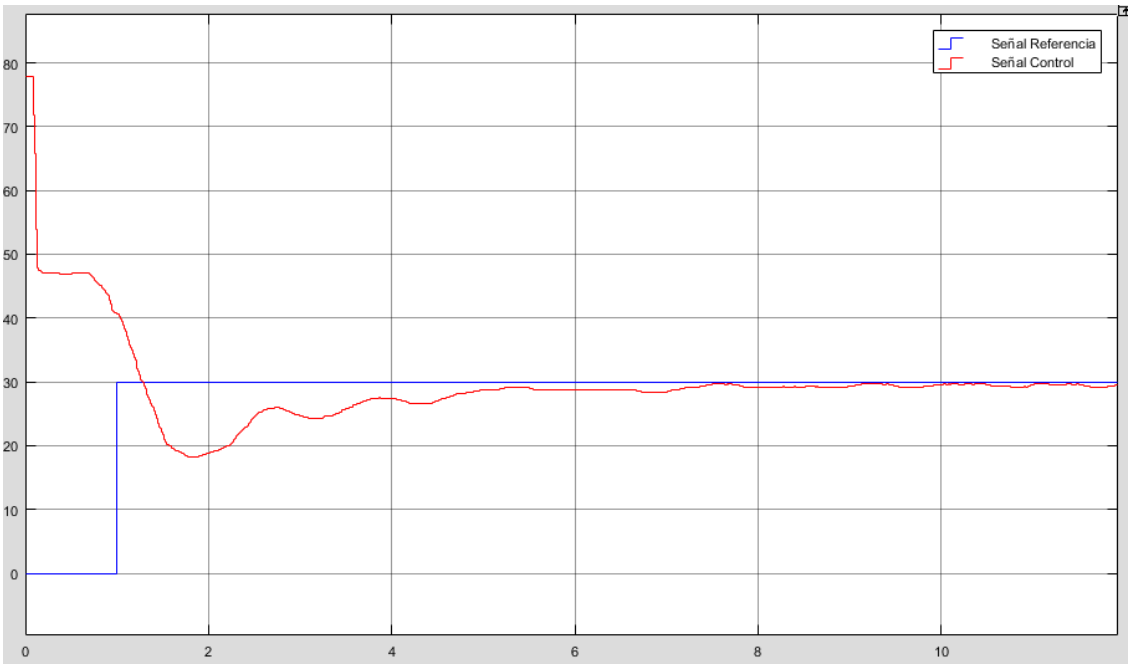


Figura 73. Sistema amortiguado con un controlador PID que alcanza la posición especificada

Logramos la posición deseada, pero tardamos demasiado en alcanzarla. Es posible que se puede mejorar la respuesta y alcanzar la posición de equilibrio más rápidamente, por lo que seguimos ajustando la constante integral por un valor intermedio.

Kp=2	Ki=0.25	Kd=4

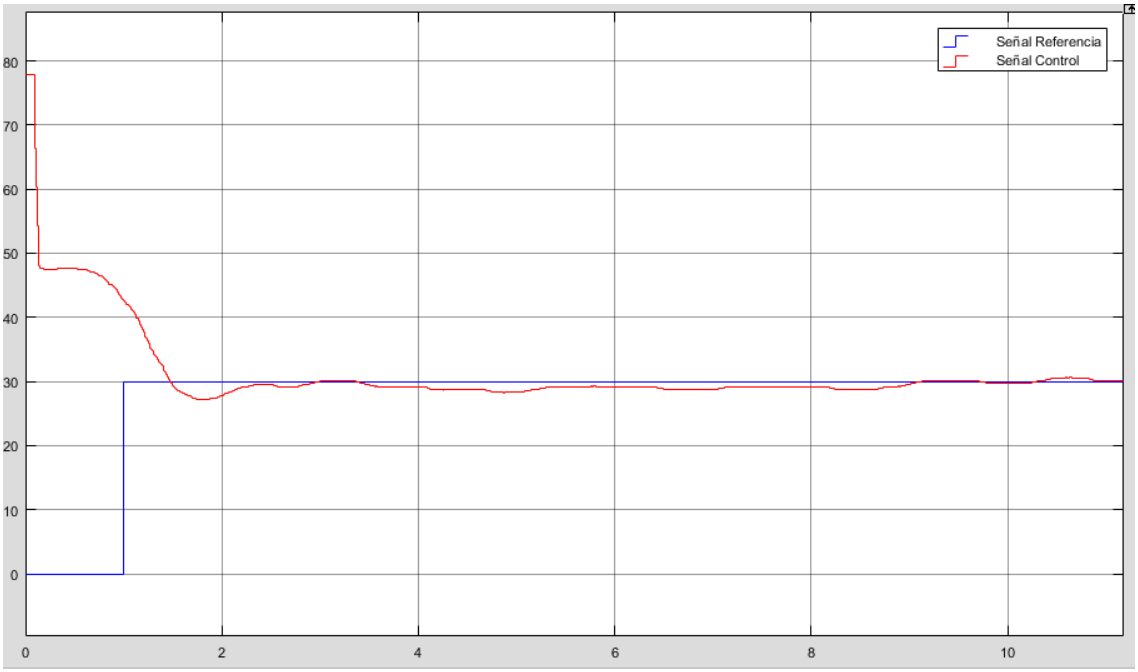


Figura 74. Sistema estable mediante un controlador PID que logra la posición especificada en un tiempo razonable

En la figura 74 se observa que tenemos un mejor resultado ya que esta vez sí conseguimos el estacionario en un tiempo razonable de 3 segundos aproximadamente y una ligera sobreoscilación.

Estos resultados son extrapolables a cambiar el valor de referencia, lo cual es imprescindible para verificar que los parámetros y el sistema funciona correctamente.

A continuación, se van a mostrar los resultados de estas simulaciones cambiando el valor de la referencia durante el tiempo de ejecución y probando con diferentes bolas de pesos y dimensiones para analizar la respuesta que se obtiene,

Primero, con la bola de billar opaca y mate con la que hemos venido trabajando y obtenido los parámetros anteriormente mostrados.

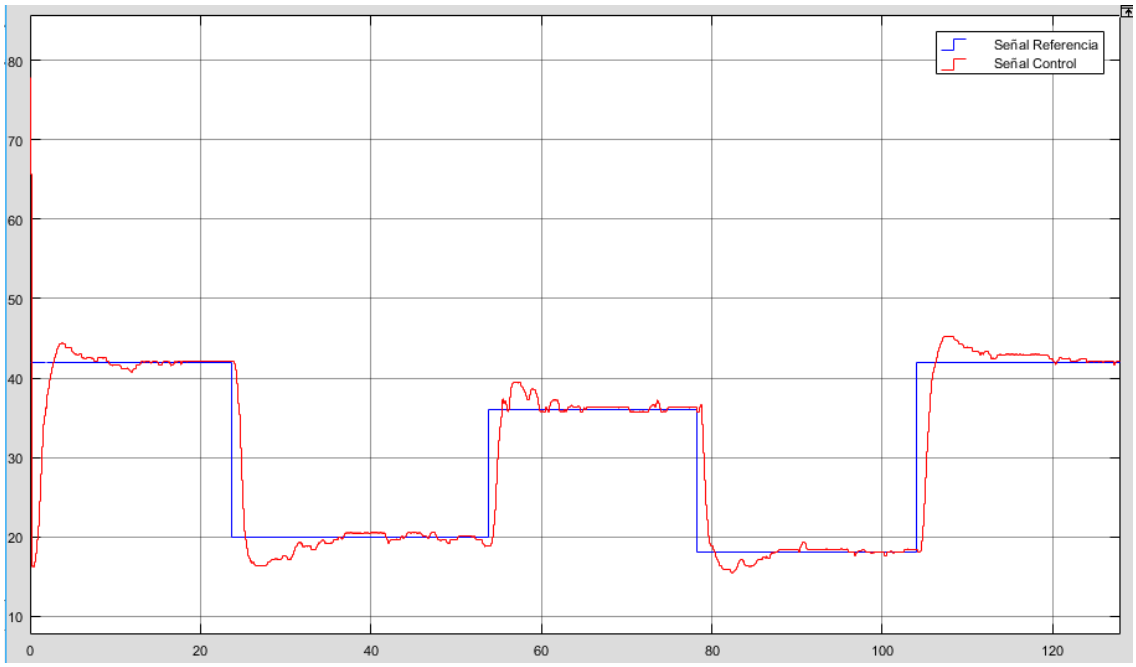


Figura 75. Respuesta del sistema cambiando la referencia durante la simulación

Nos puede surgir la idea de que la capacidad del objeto de absorber la luz puede influir en la adquisición de la señal y, por tanto, en el control del sistema. Vamos a cambiar la bola por otra de billar de iguales dimensiones, pero muy brillante.

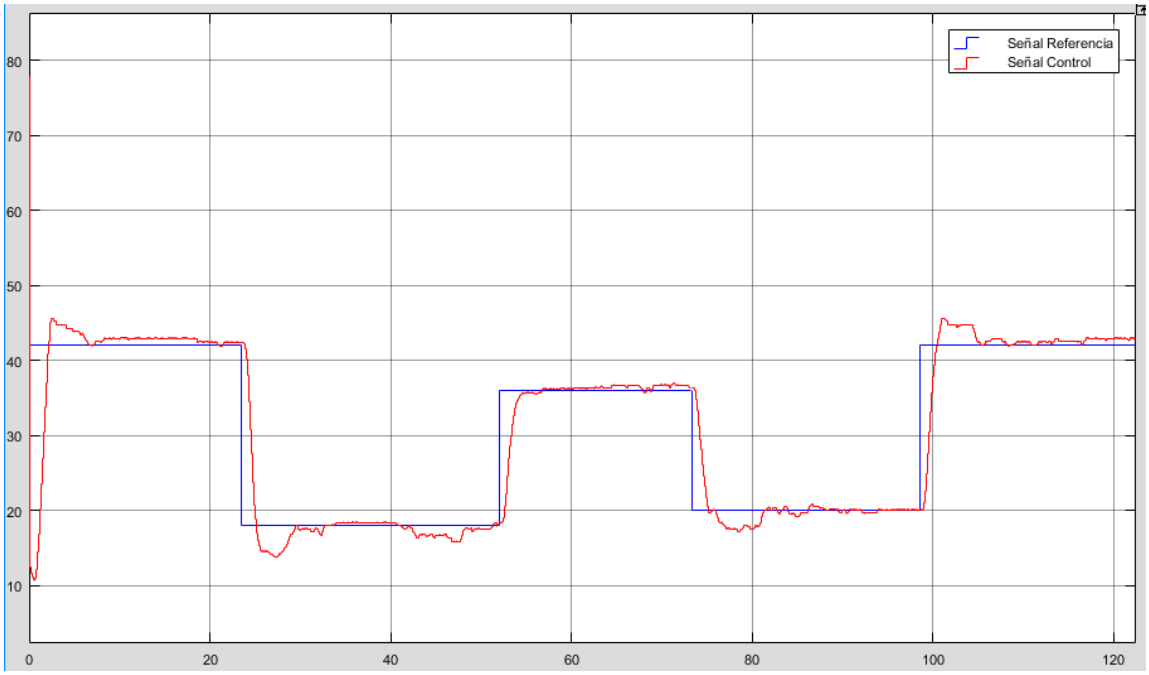


Figura 76. Respuesta del sistema cambiando la referencia durante la simulación en el caso de una bola más brillante

El resultado es satisfactorio ya que se demuestra que el sensor de distancia tiene una buena respuesta ante superficies más o menos brillantes ya que la respuesta del sistema es prácticamente igual en ambas situaciones.

Una vez probadas las dos bolas de billar de diferentes características ante la luz, vamos a introducir bolas de diferentes materiales y dimensiones.

En bolas más pequeñas nos encontramos con un problema. La sección de los carriles proporcionalmente ocupa más área en la propia bola que en una más grande y esto dificulta la rodadura de la misma, sobre todo cuando nos enfrentamos a bolas de goma o corcho.

En estos casos es común que la bola se detenga cerca de la posición deseada pero no logre moverse, por lo que tendremos que subir la constante integral. Lo que provoca esto también es que en ocasiones en las que la bola logra moverse, esta se pase de su posición y no se detenga, por lo que experimentalmente se ha concluido que la mejor decisión es subir la constante integral y derivativa para bolas pequeñas y de material que dificulta la rodadura por los carriles de aluminio instalados.

En la siguiente imagen se muestra la respuesta en tiempo real de una bola pequeña de corcho con los siguientes parámetros.

Kp=2	Ki=0.8	Kd=4

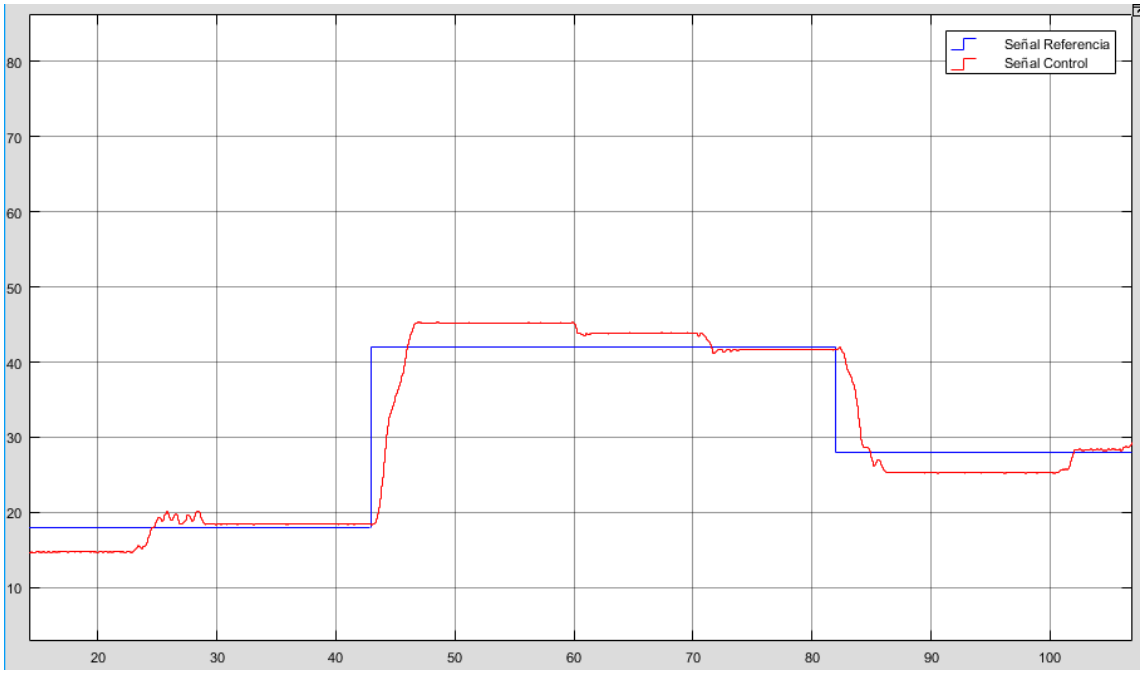


Figura 77. Respuesta temporal de una bola de corcho de pequeñas dimensiones

Se aprecia que la bola no se mueve de forma regular, sino a impulsos, debido a la dificultad de rodadura y que la viga se tiene que inclinar demasiado aun estando cerca de la posición de equilibrio para que la bola alcance esa posición.

También se ha probado con una bola de futbolín, más pequeña y de mayor densidad, donde los problemas de rodadura no se manifiestan. En este caso es más fácil establecer el equilibrio. En la figura 78 se ilustra esta respuesta con los siguientes parámetros.

Kp=2	Ki=0.7	Kd=2.5

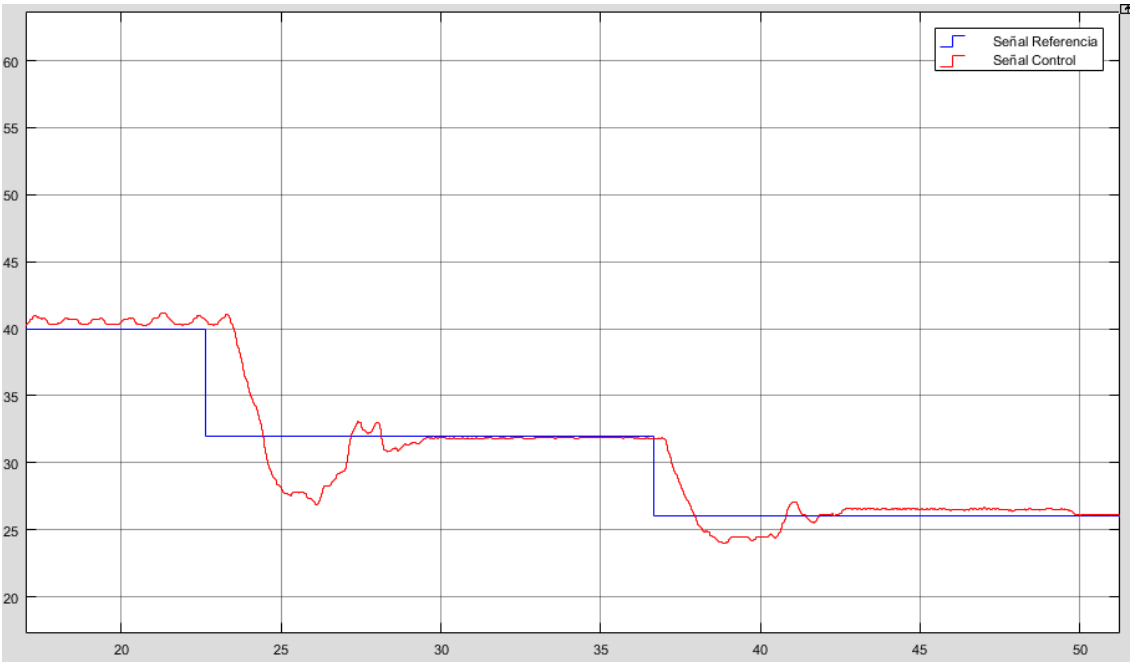


Figura 78. Respuesta temporal de una bola de futbolín

Cabe destacar que se ha probado realizar el ejercicio con los datos obtenidos teóricamente, pero los resultados no son nada satisfactorios debido a la diferencia entre el modelado del sistema real e ideal, las simplificaciones impuestas y las capacidades que tiene un servomotor real en la práctica.

El valor de la señal se puede analizar en tiempo real desplegando la gráfica del Scope donde se comparan ambas señales o, directamente, echando un vistazo a la maqueta y mirando el Display físico que se ha colocado.

Figura 79. Visualización de la posición de la bola en el sistema

Esto supone una forma más cómoda, didáctica y estética de comprobar en todo momento cual es la posición exacta de la bola en nuestro sistema.

Capítulo 8

Conclusiones y líneas futuras

Conclusiones

- Se ha estudiado el modelo y obtenido la ecuación matemática del sistema y su representación en función de transferencia y espacio de estados.
- Se ha estudiado el sistema y controlado teóricamente con diferentes métodos mediante Matlab.
- Se ha construido y montado un sistema real estudiando cada componente y adaptándolo a nuestras necesidades.
- Se ha diseñado un programa en Simulink para el sistema real y controlado mediante un PID, analizando la reacción que tiene cada variable en el sistema.
- Se ha logrado la posición estable en el estacionario en un tiempo y oscilación tolerables. Este equilibrio se ha conseguido independientemente del valor de referencia que impongamos.
- Se ha desarrollado una práctica visual con un alto potencial educativo para alumnos de la escuela de ingenierías industriales de Valladolid y se han propuesto unos apartados que pueden servir como guion para el trabajo que tengan que realizar dichos alumnos en el laboratorio de forma autónoma

.

Durante estos meses me he enfrentado a la realización de un proyecto físico y real con los problemas que ello plantea. Primero, he tenido que documentarme metódicamente para trabajar con el modelo teórico y comprender lo que estamos realizando específicamente para posteriormente realizar el proyecto real, donde los componentes electrónicos no siempre funcionan igual que indica el datasheet y hay que adaptarse en cada momento realizando las modificaciones que sean oportunas. Por otro lado, he comprobado que, en ocasiones, los modelos teóricos no siempre corresponden con las aplicaciones reales, y aunque nos sirvan de referencia, cada modelo es único porque no podemos tener en cuenta todas las variables que intervienen en un sistema.

Personalmente, me ha ilusionado construir un modelo real, enfrentarme a la dificultad de trabajar con hardware y observar los cambios que originaban las decisiones que iba tomando.

Como resumen, he puesto en práctica y ampliado mis conocimientos aprendidos durante el grado, en especial sobre automática, control y electrónica. También, he desarrollado otras capacidades no tan cuantificables como son la autogestión, la autoexigencia y el trabajo individual pero apoyado por mi tutor y el personal de laboratorio que me han ayudado en la resolución de mis dudas.

Líneas futuras

Una línea de mejora es la implementación de más modos de control en la planta real mediante Simulink. Como se ha visto en el control teórico de la planta podemos alcanzar la estabilidad trabajando en el lugar de las raíces o en realimentación de estados, aunque en el presente proyecto solo se realiza el control real mediante un PID en tiempo discreto.

Otra posible mejora sería optimizar el orden del interior de la caja envolvente, figura 55. En un principio no se tenía claro todo lo que se iba a incorporar, como la idea del Display que surgió en la mitad del desarrollo, además de la colocación de los componentes y plaquita de conexiones dentro de la caja, que se ha realizado sobre la marcha.

Por tanto, se podría haber planificado previamente el lugar de cada componente y sus conexiones para optimizar el uso de cables y el orden dentro de la caja, para facilitar la tarea de intercambiar un componente en caso de avería o comprobar una conexión en caso de un mal funcionamiento.

Cabe recordar que el proyecto se plantea como un prototipo, dejando un abanico de opciones a mejorar en la construcción de las siguientes maquetas.

Bibliografía

[1] FRIEDLAND, B. (2005). Control System Desing: An introduction to State-Space Methods. Dover Publications.

[2] PHILLIPS, C.L. (2014). Digital Control System Analysis & Design. Global Edition.

[3] HOSEVN, A. A.y SIMA, M. (2016). Ball & Beam: Controller Design. Babol Noshirvani University of Technology. https://www.academia.edu/19948851/Ball_And_Beam_Controller_Design

[4] ARACIL, J. y GÓMEZ-ESTERN, F. Sistemas realimentados de segundo orden. Apuntes de regulación automática. http://www.esi2.us.es/~fabio/cabie

[5] Ball & Beam. Control Tutorials. University of Michigan. Carnegie Mellon University. http://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SystemModeling

[6] Controlador PID. Wikipedia. https://es.wikipedia.org/wiki/Controlador_PID

[7] Controlador por espacio de estados. Wikipedia https://es.wikipedia.org/wiki/Espacio_de_estados

[8] Métodos Ziegler y Nichols. https://sites.google.com/site/picuino/ziegler-nichols

[9] Técnicas de compensación y diseño. Lugar de las raíces. http://www.eis.uva.es/~eduzal/icontrol/compen.pdf

[10] Sensor Sharp GP2Y0A02YKOF. https://www.pololu.com/file/0J156/gp2y0a02yk_e.pdf

[11] Sensor Sharp. ¿Qué es? ¿Cómo funciona? https://www.luisllamas.es/arduino-sharp-gp2y0a02yk0f/

[12] Servomotor TowerPro MG995. https://www.towerpro.com.tw/product/mg995/

[13] LCD Arduino Tutorial. https://www.arduino.cc/en/Tutorial/LiquidCrystalDisplay

[14] Caja universal para montajes industriales. Retex. http://www.retex.es/es/Cajas/Cajas-universales/Serie-101.axd

[15] Sensores ultrasonidos. https://programarfacil.com/blog/arduino-blog/sensor-ultrasonico-arduino-medir-distancia/

[16] Sensores TOF. https://www.luisllamas.es/arduino-sensor-distancia-vl53l0x/

[17] Arduino. ¿Qué es Arduino?. http://arduino.cl/que-es-arduino/

[18] Arduino Mega 2560 rev3. https://store.arduino.cc/mega-2560-r3

[19] Fritzing. http://fritzing.org/home/

[20] Fritzing. Bloque Sharp 2Y0A2X. http://fritzing.org/projects/sharp-2y0a21-2y0a02y-arduino

[21] Autocad. https://www.autodesk.es/

[22] Matlab. https://es.mathworks.com/products/matlab.html

[23] Matlab. Simulink. https://es.mathworks.com/products/simulink.html

[24] Matlab. S-Function builder. https://es.mathworks.com/help/simulink/s-function-builder.html

[25] Bugreport para instalar Support Hardware de Matlab. https://es.mathworks.com/support/bugreports/details/1741173

Apéndice

Puesta en marcha

Para la realización de este proyecto al igual que para que un futuro alumno trabaje con él tanto en casa como en la escuela, hay que realizar unos pasos previos de instalación de paquetes y configuración del equipo con Arduino que se detallarán a continuación.

Instalar Support Package

Para la realización del proyecto vamos a trabajar con dos paquetes de soporte que se pueden instalar desde la interfaz de Matlab.

Para ello hacemos click en el menú Matlab > Add-Ons > Get Hardware Support Packaged

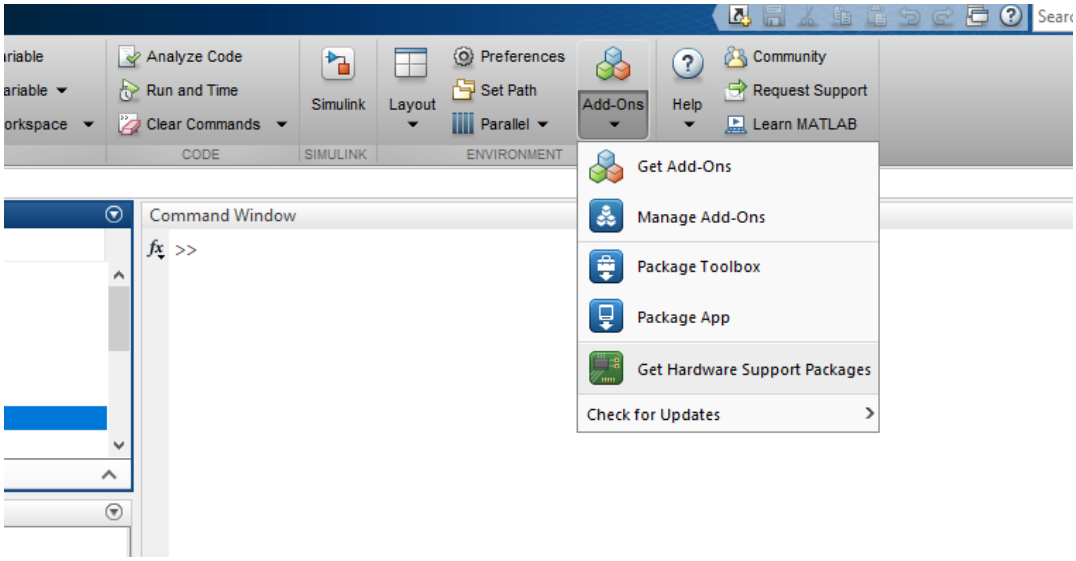


Figura 80. Menú Matlab > Add-Ons

Esperamos a que se abra una ventana con todas las opciones de paquetes disponibles.

En este caso nos interesan dos:

MATLAB Support Package for Arduino Hardware y Simulink Support Package for Arduino Hardware.

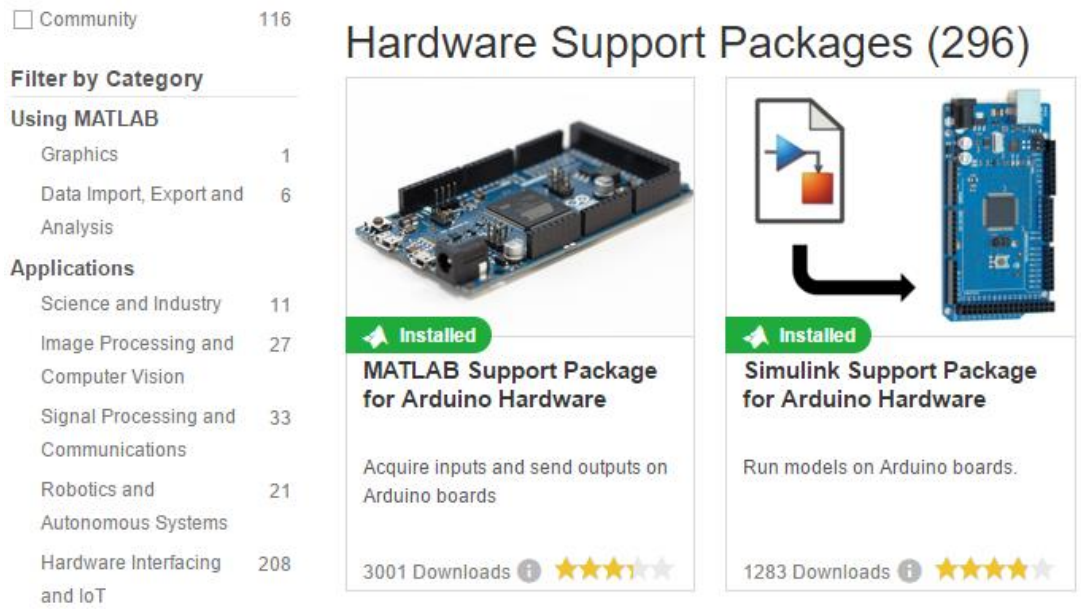


Figura 81. Paquetes de soporte para Matlab

MATLAB Support Package for Arduino Hardware habilita el uso de Matlab para comunicarse con una tarjeta Arduino, donde se podrá leer y escribir datos de sensores a través de Arduino e inmediatamente ver y analizar los resultados en Matlab sin que se precise un compilador.

Este paquete está disponible con versiones de Matlab R2014a en adelante.

Simulink Support Package for Arduino Hardware permite crear y ejecutar modelos en Simulink sobre tarjetas Arduino. Este paquete incluye una librería con bloques Arduino para configurar y acceder a sensores, actuadores y comunicación entre interfaces. Esto también permite interaccionar y programar algoritmos desarrollados en Simulink y que se ejecuten sobre la tarjeta Arduino

Este paquete está disponible con versiones de Matlab R2013a en adelante.

Básicamente hacemos click en cada uno y damos al botón de Install, donde se procederá a la instalación y únicamente habrá que seguir los pasos que se indican.

Sin embargo, este segundo paquete, 'Simulink Support Package for Arduino Hardware', actualmente presenta un bug en algunas versiones, pero el servicio ha publicado un documento (bufreport) que lo soluciona [25]. Lo que hay que realizar es seguir los pasos que se indican y que se van a explicar a continuación

El mensaje de error indica "*Download Error: There was a problem downloading the third-party software. To resolve this issue, contact Technical Support*". Y el archivo de registro envía una notificación que dice: "*Received fatal alert: protocol_Versio*n".

La solución que se toma es cambiar el protocolo HTTPS predeterminado de la versión Java utilizada por Matlab a TLSv1.2.

Para realizar esto, hay que descargar el .zip correspondiente a nuestra versión de sistema que viene alojado en la página de Matlab.

Figura 82. Ficheros para corregir el bug

Una vez descargado, nos encontramos con dos subcarpetas: ´bin´, con un archivo ´java.opts´ y la carpeta ´bugreport´. Tendremos que dirigirnos a la ubicación origen de Matlab, si no sabemos dónde se encuentra la ruta de instalación, escribimos en la línea de comandos de Matlab:

>> pwd

En la raíz Matlab > R2017a, pegamos la carpeta bugreport. A su vez, en Matlab > R2017 > bin > win64, pegamos el archivo java.opts.

Después de realizar esto, se podrá instalar este segundo paquete de soporte satisfactoriamente.

Comunicar Arduino con Simulink

Una vez que se hayan instalado los paquetes de soporte de Arduino para Matlab y Simulink, podremos comunicarnos con la tarjeta Arduino Mega 2560. El objetivo es ejecutar el programa en el ordenador simultáneamente con la planta para monitorizar y ajustar los parámetros del modelo en tiempo real, pero para ello habrá que seguir unas instrucciones de configuración básicas para preparar el equipo.

Debemos hacer click en *Tools* > *Run on target Hardware* > *Prepare to run*.

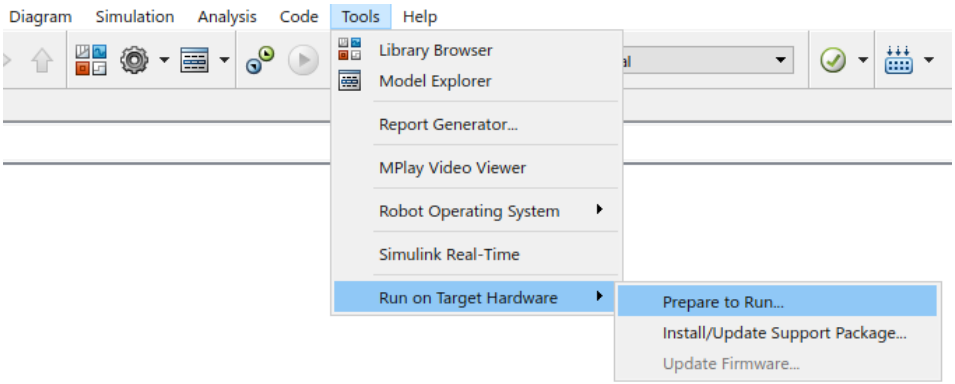


Figura 83. Menú Simulink para comunicarse con Hardware externo

En la nueva pestaña debemos indicar en el campo *Solver* > *Stop time* un valor de inf, para que la simulación dure hasta que nosotros la queramos finalizar manualmente.

En el apartado 'Hardware Implementation', debemos hacer click en el desplegable 'Hardware Board' y seleccionar nuestra tarjeta Arduino en cuestión, en nuestro caso, Arduino Mega 2560. En Host-Board connection podemos seleccionar el puerto con el que nos comunicamos que, si no sabemos y no realizamos más comunicaciones simultáneamente, podemos poner Automatically como se ve en la figura 84, donde será el software en buscar el puerto Arduino automáticamente.

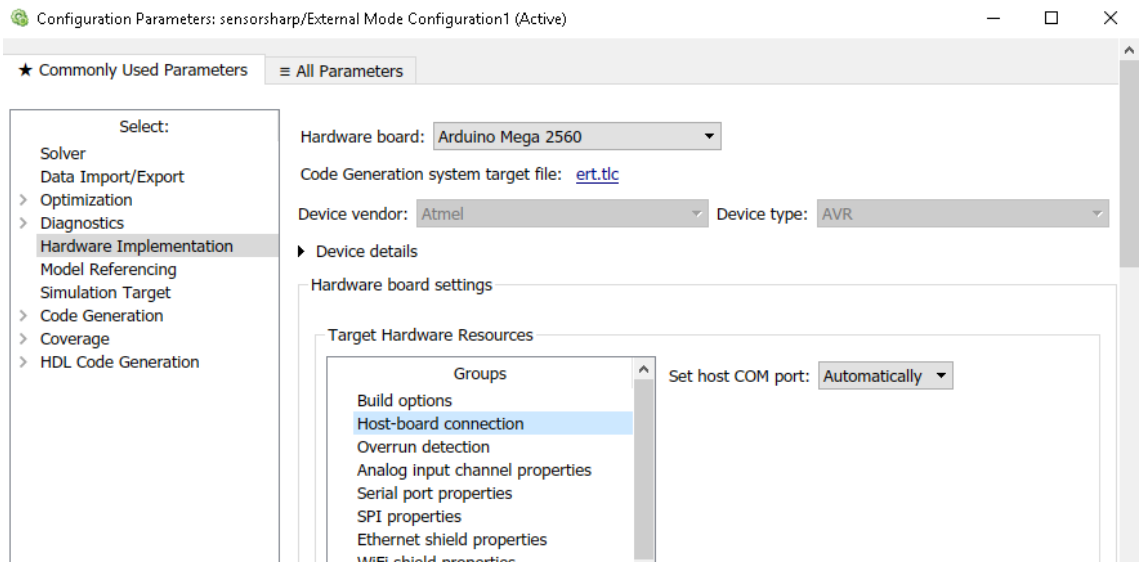


Figura 84. Comunicación con Arduino Mega 2560.

Por último, en la barra del menú principal de Simulink, donde pone 'Normal', debemos indicar 'External' para cambiar el modo de ejecución, y en el tiempo de ejecución poner infinito, 'inf', como se ilustra en la siguiente imagen. Esto nos permitirá, además, cambiar parámetros en tiempo real mediante se ejecuta la simulación.

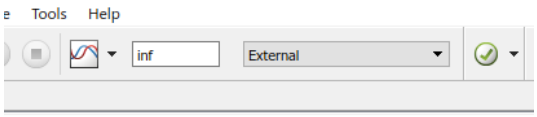


Figura 85. Configuración Simulink para ejecutar junto con Arduino

Para simular un modelo ahora únicamente tendremos que pulsar el botón 'Run'.

Modelo de práctica para un alumno

Ya hemos mencionado anteriormente que este proyecto tiene una finalidad didáctica para enseñar los fundamentos del control y la automática de una forma visual y práctica. Es adecuado, por lo tanto, poner una serie de pasos que se podrían proponer para que realizara el futuro alumno de forma autónoma como apartados de una práctica o experimento de laboratorio.

También es conveniente proponer que las maquetas que se diseñen y fabriquen para cada alumno en cada estación de trabajo tengan ligeras diferencias, ya no grandes rasgos, sino en la composición y componentes, como bien puede ser colocar el sensor a una distancia mayor o menor o la masa y características de la bola que rueda por la plataforma, variando entre una bola de Ping-pong, de golf, de goma…

En este proyecto se plantean tres prácticas o apartados que pueden servir de guía para el trabajo en el laboratorio de un futuro alumno.

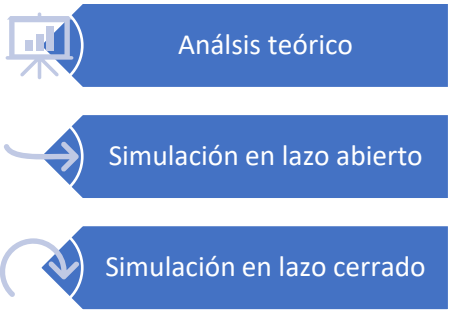


Figura 86. Esquema de prácticas a seguir por el alumno

Análisis teórico. Obtención de la función de transferencia mediante la ecuación del modelo y control teórico en Matlab.

Es decir, partimos de la siguiente expresión obtenida en el Capítulo 4:

$$\mathrm{G}(s) = \frac{R(s)}{\Theta(s)} = -\frac{m_b g d}{L\left(\frac{J_b}{R_b^2} + m_b\right)} \frac{1}{s^2} \left[\frac{m}{rad}\right] \qquad (61)$$

En este momento, se realizará el estudio teórico correspondiente únicamente con comandos de Matlab para que el alumno se familiarice con ellos.

Una vez se han introducido los parámetros del sistema que serán ligeramente únicos en cada estación de trabajo por la diferencia de bolas y creado la función de transferencia, se puede realizar una respuesta salto:

>> Step(G(s))

El alumno verá que la respuesta es inestable mediante la gráfica y observando los polos en el origen, concluyendo que será necesaria la presencia de un controlador para estabilizar el sistema.

Para ello podrá elegir valores Kp, Ki y Kd, cerrar el lazo y observar los resultados. Podrá empezar únicamente con la constante proporcional y posteriormente añadir la derivativa. En el caso teórico no tiene por qué ser necesaria la constante integral, lo cual puede deducir el estudiante.

Como ejemplo, el alumno puede llegar a los siguientes comandos en Matlab:

```
>> Kp=1; Ki=0; Kd=4;
>> C=pid(Kp,Ki,Kd);
>> Sys_cl=feedback(C*G,1)
```

Finalmente tendrá que lograr una gráfica como la que se muestra a continuación.

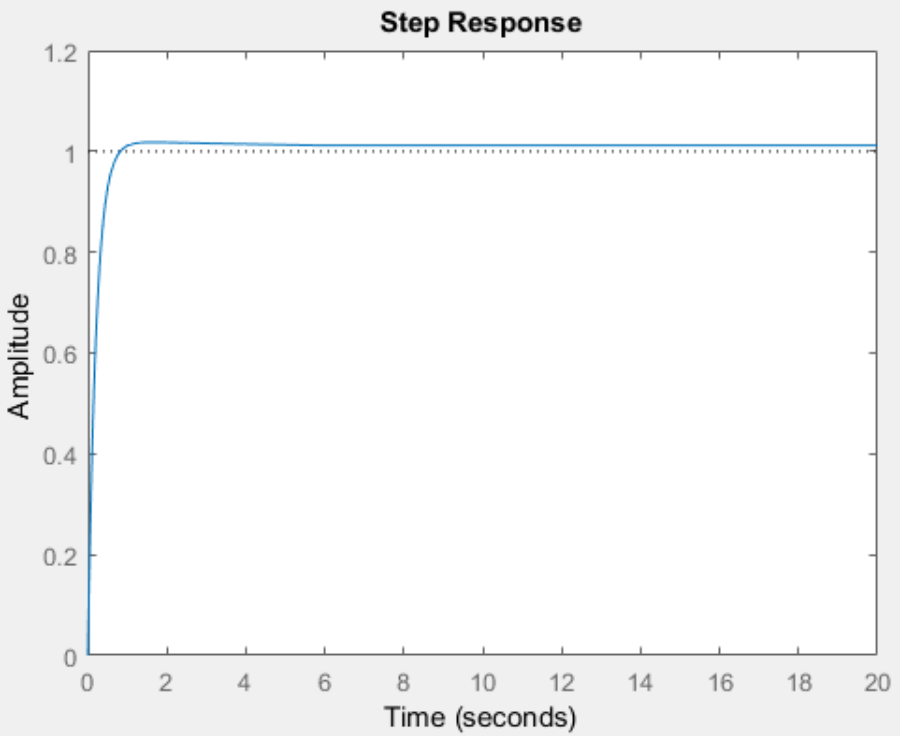


Figura 87. Respuesta estable del sistema con un controlador PD

También se puede plantear otros tipos de control como el PID discreto o la realimentación de estados. El objetivo de este primer punto es que el alumno trabaje en Matlab y estudie diferentes respuestas del sistema y funciones en la ventana de comandos.

Alternativamente, se puede introducir la teoría de Ziegler y Nichols y que se escojan unos valores PID acorde a algún método, aunque esto no sea transportable al análisis con la planta real, pero es una manera adecuada de que el alumno estudie de una forma más práctica, mediante Matlab, lo visto en las clases de teoría. A su vez, asimilará la diferencia y dificultad que surge cuando trasladamos un modelo teórico a la planta real ya que es imposible de emular todas las variables que entran en juego y construir un modelo idílico.

Simulación en lazo abierto. Caracterización del sensor.

Para una sección de la práctica se le puede facilitar al alumno parte del modelo obviando la función de Matlab que caracteriza el sensor y la parte del actuador. El objetivo es que el estudiante realice la simulación en lazo abierto y obtenga una salida en función a lo que queremos medir como referencia.

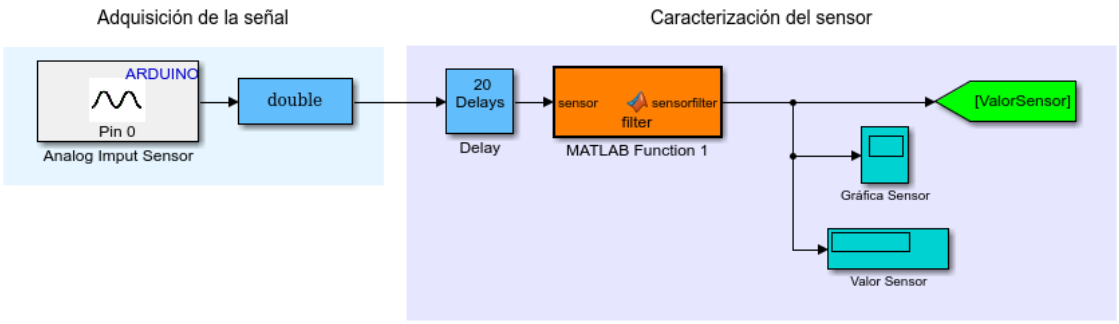


Figura 88. Esquema Simulink en lazo abierto

Del sensor conseguimos una medida de tensión y tenemos que convertirla en centímetros para trabajar con ella. El alumno tendrá que desarrollar esta relación teniendo en cuenta que el sensor no es lineal y, además, que cada uno va a tener una función ligeramente distinta y única, ya que los sensores se colocarán a diferentes distancias en cada maqueta y no todos obtienen la misma respuesta ante la misma entrada.

Podrá apuntar los valores que vaya obteniendo en una hoja de cálculo como Excell y obtener dicha relación 'Variable medida – Variable deseada' según el tipo de línea que más se aproxime en cada caso.

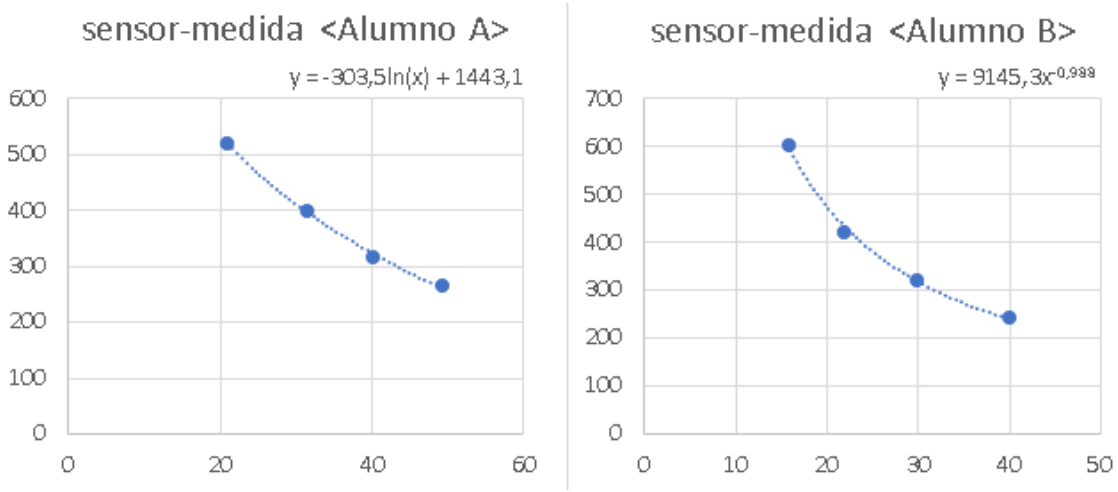


Figura 89. Línea de tendencia para dos alumnos con sensores que difieren entre sí

Con esta curva el alumno ya está en disposición de obtener la ecuación que caracteriza su sensor.

Simulación en lazo cerrado. Diseño del controlador

Como punto final, el alumno cerrará el lazo del sistema y desarrollará el sistema PID, una vez que ya se ha elegido la referencia y caracterizado el sensor.

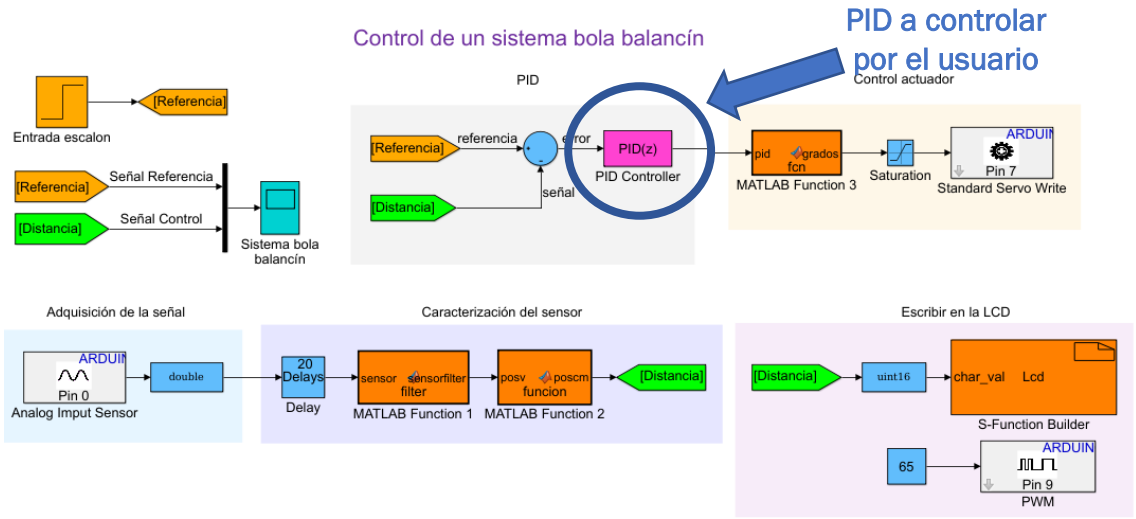


Figura 90. Esquema Simulink en lazo cerrado

Se puede proponer como objetivo que el estudiante obtenga una salida estable en un rango de +-2 cm en un tiempo determinado y sin bruscas oscilaciones. El

resultado se podrá acompañar con capturas del Scope comparando la referencia con la salida y con fotografías a la maqueta donde se visualice el valor en el Display físico.

Para ello tendrá que estudiar las componentes de un PID y sus consecuencias en la reacción de un sistema de segundo orden en una planta real. El estudiante deberá trabajar metódicamente y comprender lo que ocurre en su sistema en particular. Esto supone un punto final que significa la comprensión del controlador PID y uno de los fundamentos del control automático.

Anexo

Código

Control teórico del sistema mediante Matlab

Codigo_control.m

```
%% Control teórico del sistema
%% Parámetros y función de transferencia
m = 0.16;
R = 0.0285;
g = -9.8;
L = 0.43;
d = 0.03;
J = 5.1984e-5;

s=tf('s');

G=-(m*g*d)/(L*((J/R^2)+m)*s^2)

%% Controlador PID

%% P
Kp=1;
C=pid(Kp);
sys_cl=feedback(C*G,1)
step(sys_cl)
axis([0 20 0 2])

%% PD
Kp=1;
Kd=10;
C=pid(Kp,0,Kd);
sys_cl=feedback(C*G,1)
step(sys_cl)
axis([0 20 0 1.2])

%% PID
Kp=1
Ki=2
Kd=60
C=pid(Kp, Ki, Kd);
sys_cl=feedback(C*G,1)
step(sys_cl)
axis([0 5 0 1.2])
```

```
%% Lugar de las raices
rlocus(G)
sgrid(0.7, 1.9)
axis([-5 5 -2 2])
zo=0.01;
po=5;
C=tf([1 zo], [1 po]);
rlocus(C*G);
sgrid(0.70, 1.9);
[k, polos]=rlocfind(C*G)

sys_cl=feedback(k*C*G,1);
step(sys_cl)
axis([0 5 0 1.2])

%% PID discreto
Ts=1/50;
Gd=c2d(G,Ts,'zoh')

[x,t]=step(0.1*Gd,5);
stairs(t,x)


%% P
Kp=100;
sys_cl=feedback(Kp*Gd,1);

[x,t]=step(0.1*sys_cl,5);
stairs(t,x)

%% PD
z=tf('z',Ts);

Kp=100;
Kd=20;

C=((Kp+Kd)*z^2-(Kp+2*Kd)*z+Kd)/(z^2+z);

sys_cl=feedback(C*Gd,1);

[x,t]=step(0.1*sys_cl,5);
stairs(t,x)


%% PID
z=tf('z',Ts);

Kp=2000;
Kd=20;

C=((Kp+Kd)*z^2-(Kp+2*Kd)*z+Kd)/(z^2+z);
```

```
sys_cl=feedback(C*Gd,1);

[x,t]=step(0.1*sys_cl,5);
stairs(t,x)
```

Codigo_control_ss.m

Control teórico del sistema en Matlab mediante espacio de estados.

```
%% Parámetros y abrimos sistema simulink
m = 0.16;
R = 0.0285;
g = -9.8;
L = 0.43;
d = 0.03;
J = 5.1984e-5;

nPts= 1e3;
t= linspace(0, 40, nPts)';
U= ones(nPts,1);

open_system('balancin_sistema');
[tsal, ~, sal]= sim('balancin_sistema', t, [], [t,U]);
plot(tsal,sal)

%% Modelo en espacio de estados
[A,B,C,D] = linmod('balancin_sistema')
[num,den] = ss2tf(A,B,C,D);
ball_ss = ss(A,B,C,D);
figure
lsim(ball_ss,U,t)

%% Seleccionamos los polos
p1 = -1.5;
p2 = -1;

K = place(A,B,[p1,p2])
t = 0:0.01:5;
u = 0.25*ones(size(t));
sys_cl = ss(A-B*K,B,C,D);
[y,t,x] = lsim(sys_cl,u,t);
plot(t,y)

%% Incorporamos ganancia en la referencia
u=ones(size(t));
Nbar=rscale(ball_ss,K)
[y,t,x] = lsim(Nbar*sys_cl,u,t);
plot(t,y)
```

```
%% Filtro Kalman
nPts= 1e3;
t= linspace(0, 40, nPts)';
U= ones(nPts,1);

open_system('ball_lc_Kalman');
[tsal, ~, sal]= sim('ball_lc_Kalman', t, [], [t,U]);
plot(tsal,sal)
```

rscale.m

```
function[Nbar]=rscale(a,b,c,d,k)

% Given the single-input linear system:
% . % x = Ax + Bu
% y = Cx + Du
% and the feedback matrix K,
%
% the function rscale(sys,K) or rscale(A,B,C,D,K)
% finds the scale factor N which will
% eliminate the steady-state error to a step reference
% for a continuous-time, single-input system
% with full-state feedback using the schematic below: %
% /--------\ % R + u | . | % ---> N --->() ----> | X=Ax+Bu |--> y=Cx ---> y
% -| \--------/ % | | % |<---- K <----| %
% 8/21/96 Yanjie Sun of the University of Michigan % under the supervision of Prof.
D. Tilbury
% 6/12/98 John Yook, Dawn Tilbury revised

error(nargchk(2,5,nargin))

% --- Determine which syntax is being used ---
nargin1 = nargin;
if (nargin1==2) , % System form
   [A,B,C,D] = ssdata(a);
   K=b;
elseif (nargin1 == 5), % A,B,C,D matrices
   A=a; B=b; C=c; D=d; K=k;
else
   error('Input must be of the form (sys,K) or (A,B,C,D,K)')
end;

% compute Nbar
s = size(A,1);
Z = [zeros([1,s]) 1];
N = inv([A,B;C,D])*Z';
```

```
Nx = N(1:s); Nu = N(1+s);
Nbar=Nu + K*Nx;
```

Funciones utilizadas en Simulink

Matlab Function 1

Filtro de la señal del sensor.

```
function sensorfilter = filter(sensor)
sensorfilter=0.0;
sortedValues=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
for a=1:20 %20 medidas
   if(sensor(a)<sortedValues(1) || a==1)
      j=1; %%inserta en la primera posicion
   else
    for j=2:a
       if(sortedValues(j-1)<=sensor(a)&& sortedValues(j)>=sensor(a))
       break;
       end
    end
   end
   for k=a:-1:j+1 %movemos los valores una posición
      sortedValues(k)=sortedValues(k-1);
   end
   sortedValues(j)=sensor(a); %se inserta lectura actual
end
for a=8:12 %hacemos media de los intermedios
   sensorfilter=sensorfilter+sortedValues(a);
end
sensorfilter=sensorfilter/5;
end
```

Matlab Function 2

Caracterización del sensor.

```
function poscm = funcion(posv)
poscm=-0.1138*posv+77.958;
end
```

Matlab Function 3

Mapeado de la señal PID para controlar el actuador

```
function grados = fcn(pid)
grados = ((pid - (-180))*(180 - 0))/(180 - (-180)) + 0;
end
```

S-Function

Dentro del bloque S-Function Builder.

Libraries > Include:

```
#include <math.h>
#ifndef MATLAB_MEX_FILE
#include "LiquidCrystal.h"
#include "LiquidCrystal.cpp"

// initialize the library by associating any needed LCD interface pin
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#endif
```

Libraries > Outputs

```
/* This sample sets the output equal to the input
     y0[0] = u0[0];
 For complex signals use: y0[0].re = u0[0].re;
     y0[0].im = u0[0].im;
     y1[0].re = u1[0].re;
     y1[0].im = u1[0].im;
*/

if(xD[0] == 1)
{
   #ifndef MATLAB_MEX_FILE
   lcd.setCursor(0, 1);
   lcd.print(char_val[0]);
   lcd.setCursor(3, 1);
   lcd.print("cm");
   #endif
}
```

Libraries > Discrete Update

```
/*
 * Code example
 *   xD[0] = u0[0];
*/

if(xD[0] != 1){
   # ifndef MATLAB_MEX_FILE
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.setCursor(1, 0);
    lcd.print("-TFG BALANCIN-");
    #endif
    //done with initialization
    xD[0] = 1;
}
```

Lcd_wrapper.cpp

```
/*
 * Include Files
 *
 */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif



/* %%%-SFUNWIZ_wrapper_includes_Changes_BEGIN --- EDIT HERE TO _END */
#include <math.h>
#ifndef MATLAB_MEX_FILE
#include "LiquidCrystal.h"
#include "LiquidCrystal.cpp"

// initialize the library by associating any needed LCD interface pin
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#endif
/* %%%-SFUNWIZ_wrapper_includes_Changes_END --- EDIT HERE TO _BEGIN */
#define u_width 1
/*
 * Create external references here.
 *
 */
/* %%%-SFUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
/* extern double func(double a); */
/* %%%-SFUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */

/*
 * Output functions
 *
 */
extern "C" void Lcd_Outputs_wrapper(const uint16_T *char_val,
                        const real_T *xD)
{
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
/* This sample sets the output equal to the input
     y0[0] = u0[0];
 For complex signals use: y0[0].re = u0[0].re;
     y0[0].im = u0[0].im;
     y1[0].re = u1[0].re;
     y1[0].im = u1[0].im;
*/
```

```
if(xD[0] == 1)
{
   #ifndef MATLAB_MEX_FILE
   lcd.setCursor(0, 1);
   lcd.print(char_val[0]);
   lcd.setCursor(3, 1);
   lcd.print("cm");
   #endif
}
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
}

/*
 * Updates function
 *
 */
extern "C" void Lcd_Update_wrapper(const uint16_T *char_val,
                                   real_T *xD)
{
/* %%%-SFUNWIZ_wrapper_Update_Changes_BEGIN --- EDIT HERE TO _END */
/*
 * Code example
 *   xD[0] = u0[0];
*/

if(xD[0] != 1){
   # ifndef MATLAB_MEX_FILE
     lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.setCursor(1, 0);
    lcd.print("-TFG BALANCIN-");
    #endif
    //done with initialization
    xD[0] = 1;
}
/* %%%-SFUNWIZ_wrapper_Update_Changes_END --- EDIT HERE TO _BEGIN */
}
```

Hojas de características

Las hojas de características o datasheets de los componentes utilizados en el desarrollo de este proyecto se encuentran alojados en la capeta adjunta llamada “TFG_01382_anejos”.