Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Diseño, implementación y control de un sistema "ball and beam"

Autor: Alfonso Delgado Díaz
Tutor: Ignacio Alvarado Aldea

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Diseño, implementación y control de un sistema "ball and beam"

Autor:
Alfonso Delgado Díaz

Tutor:
Ignacio Alvarado Aldea
Profesor Titular de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Diseño, implementación y control de un sistema "ball and beam"

Autor: Alfonso Delgado Díaz

Tutor: Ignacio Alvarado Aldea

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de

Sevilla, 2020

El Secretario del Tribunal

A mi padre, Cefe, que siempre me mostró el camino a seguir.

A mi madre, Pepi, que le acompañó en esa dura tarea y la continúa hoy en día.

A mi hermano, Francisco Javier, que nunca dejará de estar a mi lado en el devenir de la vida.

Agradecimientos

En primer lugar, quiero mostrar en estas líneas mi agradecimiento al Profesor Titular de Universidad Ignacio Alvarado Aldea, que ha tutorizado este Trabajo en unas condiciones más que adversas y desconocidas para todos, estando siempre disponible para cualquier consulta, rectificación o propuesta y facilitándome todos los medios a su alcance como si fueran míos propios. Sin su ayuda, este Trabajo no sería como es hoy.

Por otro lado, es de justicia agradecer el apoyo mostrado por mi familia y amigos. De los primeros, especialmente por los que han convivido conmigo durante la mayor parte del tiempo en que se ha realizado este Trabajo, con la dificultad añadida de no tener la posibilidad ni siquiera de salir a tomar el aire cuando las cosas no salían como uno esperaba. Ellos han sido mi bocanada de aire fresco. *Gracias Mamá*, *gracias Nene*, *gracias Titi*. De los segundos, ellos saben que si les considero como tal, es porque no hacen falta demasiadas palabras. Solo me queda darles las gracias por escucharme aún sin tener ni idea de Ingeniería, por consolarme en los momentos duros y por sacarme una sonrisa incluso a través de una pantalla de ordenador.

Por último, un gracias que va dirigido un poco más lejos, pero que estoy seguro que llegará. A quién me apoya cuando ni las palabras, ni los besos ni los abrazos de aquí abajo solucionan nada. Es entoneces cuando, desde donde quiera que esté, su apoyo me reconforta y me hace seguir hacia delante. *Gracias Papá*.

Alfonso Delgado Díaz

Sevilla, Junio de 2020

Resumen

El presente trabajo consiste en el diseño, la implementación y el control de un sistema "ball and beam". En la primera parte del mismo, se desarrolla el estudio realizado para la elección del hardware y del software, utilizando, finalmente, una estructura de biela y excéntrica formada por dos barras huecas de aluminio que forman un carril sobre el que rueda sin deslizar una bola. Todas las piezas de plástico utilizadas son diseñadas e impresas con una impresora 3D. La posición lineal de la bola, que es la salida del sistema, se mide con un sensor láser infrarrojo VL53L1X colocado en un extremo de las barras. Estas se mueven gracias a un servomotor MG996R, que hace girar la excéntrica que, con una biela, consigue hacer que el sistema se mueva a conveniencia. La inclinación real de las barras, que es la entrada del sistema, se mide con una unidad de medida inercial (IMU). Todo lo anterior será controlado con una placa Arduino UNO, usando software libre y de realización propia. Se destaca también el uso un mando a distancia con un receptor IR para la modificación de las referencias por parte del usuario.

En segundo lugar, se analiza teóricamente el sistema para obtener una función de transferencia que lo modele correctamente. Tras esto, se estudia la estabilidad del mismo y se concluye que el sistema se debe controlar, como mínimo, con un controlador PD. Para comprobarlo, se hacen simulaciones de controladores lineales discretos y continuos observando resultados correctos. A continuación, se implementa un simulador para controladores predictivos usando un algoritmo GPC y se muestran las mejoras observadas frente al controlador lineal. La sección de simulaciones acaba explicando que, en principio, se requiere de un control en cascada para poder controlar el ángulo del servomotor, ya que de no ser así, se estaría dejando en bucle abierto al actuar el control primario sobre el ángulo de las barras y no el del servomotor.

Finalmente, se implementan los controladores que se han probado en simulación, ajustándolos debidamente al prototipo real y se muestran los resultados gráficos obtenidos mediante la comunicación implementada entre Arduino y MATLAB, tanto en tiempo real como a posteriori. Se concluye que el sistema se puede controlar de forma aparentemente aceptable con un control PD si se caracteriza de forma lineal la relación entre el ángulo de las barras y el del servomotor; y con un control PID en cascada si se resuelve en tiempo real los posibles errores que se detecten entre los ángulos. Además, se indica que el sistema se presta a la implementación de estrategias de control moderno más complejas y avanzadas en futuras ampliaciones.

Abstract

The current Proyect consits of the desing, implementation and control of a "ball and beam" system. In its first part, it takes place the study carried out in order to choose all hardware and software; using, finally, a connecting rod and eccentric wheel structure with two hollow Aluminium bars which make a lane through a ball can roll without sliding. All the plastic pieces are design and printed in a 3D printer. The linear position of the ball, which is the system output, is measured with a VL53L1X infrared laser sensor which has been put at one of the ends of the bars. These are moved thanks to a MG996R servomotor which makes the eccentric wheel to roll and, with the help of the connecting rod, gets the system move at your convenience. The real angle of the bars (which is the system input) is measured, in relation to the horizontal line, with a Inertial Measure Unit (IMU). All explained above is controled by an Arduino UNO board, using free and self-made software. The user will be able to change the set-point with a remote control based in a infrared detector.

In the second place, the system is theoretically analized in order to obtain a transfer function that illustrate it properly. After this, the system stability is studied, concluding that it should be controlled, at least, with a PD controller. To check it, some simulations of linear discrete and continuous controllers are done, with good results. Coming up next, a Model Predictive Controllers simulator is implemented using a GPC algorithm and the improvements to the others are shown. This section dedicated to modelation and simulations finishes with an explanation of the need of using a cascade controller to control the servomotor angle, because if it is not used, we would be keeping it in open loop using just the bars' angle as control action.

Finally, some of the controllers that have been tested in simulations, are implemented in the real "ball and beam" system, fitting properly all its parameters. The graffic results are shown thanks to the comunication between Arduino and MATLAB that has been turned on, both at real time and after the experiment. The conclusions are that the system could be acceptably controlled by a PD controller if it is characterized the relation between the bars' angle and the servomotor angle; and by a PID cascade controller if it is solved in real time the differences detected among these angles. Apart from that, indicate that the real system is opened to the implementation of modern and more complex control strategies in future extensions to this Proyect.

Índice

Índice de Tablas

Índice de Figuras

1 INTRODUCCIÓN

La inspiración existe, pero tiene que encontrarse trabajando.

Pablo Ruiz Picasso

HISTÓRICAMENTE, siempre ha sido un deseo del ser humano el control de los sistemas que este se ha ido encontrando en la naturaleza, y más especialmente, como se desprende de [1], a partir de la Revolución Industrial del siglo XIX, cuando, gracias a la máquina de vapor, las actividades de carácter automático pudieron empezarse a plantear; si bien no al nivel que se ven actualmente.

En este contexto, se presenta este Trabajo como la recopilación de todos los pasos que componen el diseño, la implementación y el control de un sistema real conocido como "ball and beam". A saber: el diseño de sus piezas en tres dimensiones a través de un programa específico para ello, la elección de los componentes que serán usados como sensores y actuadores del sistema, su implementación y construcción, su modelado teórico, la simulación de distintos controladores y, finalmente, el control del sistema real.

1.1 Motivación y objetivos

Todos los capítulos que se relacionan en este Trabajo van encaminados a cumplir como objetivo principal el entender de forma completa el sistema bajo estudio y el uso posterior de este como equipo de prácticas para asignaturas de Control Automático de distintos estudios de Grado en Ingeniería impartidos en la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla.

Efectivamente, es común que los alumnos de los primeros cursos de los Grados en Ingeniería no aprecien la importancia del Control Automático en la Industria y en la Ingeniería en general. Con la utilización futura de este Trabajo, se pretende modificar esta visión de los alumnos haciéndoles ver con un sistema real, diseñado y construido desde cero, cómo puede llegarse a controlar en torno al punto de referencia que se quiera.

Por este motivo se ha elegido un sistema, que como se verá más adelante es inestable en bucle abierto, es decir, que incrementos acotados en la entrada, pueden producir incrementos sin límites en la salida, ya que aun siendo, por lo general, más difícil de controlar, el resultado es mucho más impactante. De igual forma, con el objetivo de poder buscar la repetibilidad del sistema, se pretende buscar hardware de bajo coste y fácil obtención; así como software libre, que esté al alcance de todo aquel que quiera acceder a él a través de Internet.

Por otro lado, cabe destacar que se busca lograr, aunque solo en el capítulo de simulaciones, implementar un control más avanzado que el control lineal PID, que será la primera opción usada. El objetivo de estas dos

opciones de control es mostrar la importancia de las nuevas estrategias de control moderno, que están en perenne evolución y cómo estas pueden llegar a dar resultados mucho mejores que los controladores clásicos.

Finalmente, quiere hacerse mención a que el orden en el que se desarrollan los capítulos de este Trabajo no es necesariamente un orden cronológico, sino más bien un orden lógico en el que establecer todas las ideas, experimentos, comprobaciones y resultados que se han ido elaborando a lo largo de todo el cuatrimestre que ha comprendido esta labor, usando siempre, siguiendo una analogía con el control, la estrategia de realimentación negativa, para corregir los errores que se iban observando entre los objetivos a los que se quería llegar y los resultados reales que se obtenían.

1.2 Estado del arte

Es amplia la bibliografía que se puede encontrar referente al sistema "ball and beam" por su extensa utilización como equipo de laboratorio para la prueba de distintas estrategias de control que se han ido desarrollando, sobre todo en el ámbito académico, a lo largo de los años. Sin embargo, igual de amplia es la diversidad que se presenta tanto desde el punto de vista estructural como de hardware y software utilizado en cada caso.

En cuanto a la estructura usada para la implementación es común la presencia de una barra y una bola que rueda sobre ella en la dirección longitudinal de la misma, es decir, en el único grado de libertad que se deja libre. Sin embargo, aún siguiendo este denominador común hay dos grandes grupos bien diferenciados.

Por un lado, se encuentran los que utilizan un punto de apoyo fijo en uno de los extremos de la barra, que únicamente permite el giro solidario de esta y en el otro extremo se encuentra un mecanismo de biela que mueve solidario a una rueda excéntrica que tiene conectado el actuador en su centro. Así se consigue modificar la altura de la barra con el giro del servomotor. Estas estructuras son las que se presentan en los artículos científicos [2] y [3], obteniendo buenos resultados experimentales. Gráficamente puede verse en la siguiente imagen que utiliza [4] en su desarrollo.

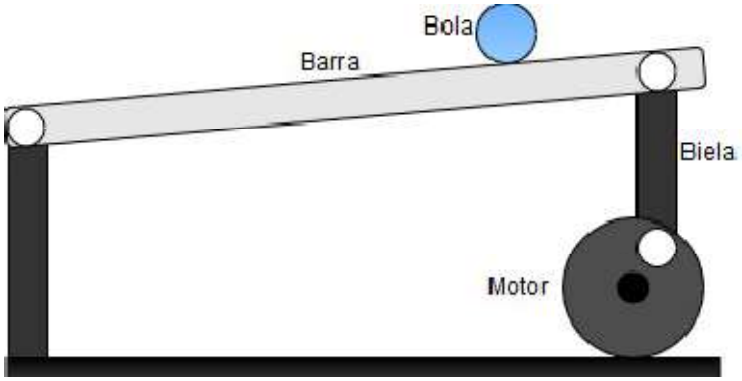

Ilustración 1 Estructura con biela y excéntrica

Esta opción presenta grandes ventajas como la unión de toda la estructura más complicada (actuadores y mecanismo de transmisión) en un extremo del prototipo o el carácter más armónico de los movimientos de la barra, al ser los ángulos de esta más pequeños y parecidos entre sí. Sin embargo, también se observa que el amplio recorrido que permite el servomotor se ve reducido en gran medida por el mecanismo de biela, ya que el ángulo máximo y mínimo que puede alcanzar la inclinación de la barra con respecto a la horizontal es más reducido.

El otro gran grupo encontrado es aquel que coloca el motor en el centro de las dos barras, consiguiendo por tanto, que el ángulo del servomotor sea directamente el ángulo que forman las barras con la horizontal. Es la estructura que se usa en los artículos [5] y [6]. Con ella, los resultados también son favorables usando una adecuada estrategia de control, por lo que es perfectamente válida. El problema es que se pierde precisión, o al menos esta se supedita a la que tenga intrínsecamente el propio actuador, ya que al no haber mecanismo de transmisión entre este y las barras, si el motor solo se puede mover en incrementos de, por ejemplo, un grado sexagesimal, esta será la precisión del ángulo de las barras.

Para que la comparación con la anterior sea más sencilla y de un simple vistazo puedan verse las similitudes y diferencias, se utiliza de nuevo la ilustración usada en [4] para mostrar la estructura explicada.

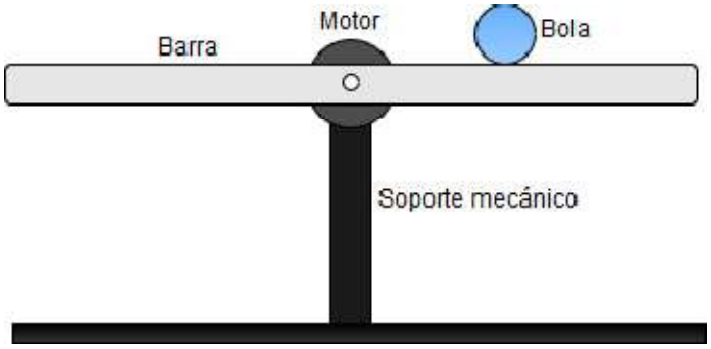

Ilustración 2 Estructura con motor en el centro de la barra

Además de estas dos estructuras principales, pueden llegarse a encontrar en la bibliografía otras muchas que suelen ser combinaciones o ampliaciones de las anteriores, por lo que no se considera relevante incluirlas aquí.

Variadas son también las opciones elegidas como actuador del sistema (motores paso a paso, servomotores, etc) o como sensores para medir la posición de la bola (infrarrojos, ultrasónicos, etc). Sin embargo, se deja la redacción de las distintas propuestas tenidas en cuenta, sus ventajas e inconvenientes y las elecciones finales para el capítulo relativo al hardware del sistema.

1.3 Conclusiones preliminares

Tal y como se desarollará en los siguientes capítulos, finalmente, el sistema "ball and beam" bajo estudio sigue la estructura de la Ilustración 1, usando dos baras de aluminio que forman un caril sobre el que se mueve una bola, cuya posición lineal se mide con un sensor láser infrarrojo colocado en un extremo en el soporte que une las barras. Las barras se mueven gracias a un servomotor que gira una excéntrica que, con una biela, llega hasta el soporte que une las dos barras en el extremo contrario al sensor. La inclinación real de las barras se mide con una unidad de medida inercial (IMU). Todas las piezas son diseñadas e impresas con una impresora 3D. El sistema será controlado con un Arduino UNO, usando software libre y de realización propia para ello. Las referencias serán modificadas usando un mando a distancia que funciona con infrarrojos.

Se concluye teóricamente que el sistema se puede controlar con un controlador PD y se hacen simulaciones de controladores lineales discretos y continuos observando buenos resultados. Se implementa un simulador para controladores predictivos usando un algoritmo GPC y se muestran las mejoras observadas frente al controlador lineal. Se explica que, en principio, se requiere de un control en cascada para poder controlar el ángulo del servomotor de modo y forma que se alcance el ángulo de las barras que haga que la posición lineal de la bola coincida con la referencia solicitada.

Finalmente, se implementan los controles que se han probado en simulación, ajustándolos debidamente al prototipo real y se muestran los correctos resultados obtenidos, concluyendo que el sistema se puede controlar de forma aparentemente aceptable con un control PD si se caracteriza de forma lineal la relación entre el ángulo de las barras y el del servomotor; y con un control PD en cascada con un PID si se resuelve en tiempo real los posibles errores que se detecten en los ángulos. Además, se indica que el sistema se presta a la implementación de estrategias de control moderno más complejas y avanzadas.

2 DISEÑO DEL PROTOTIPO

El proceso de diseño más óptimo integra las aspiraciones del arte, la ciencia y la cultura.

Jeff Smith

EL objetivo principal de este capítulo es desarollar de forma clara y concisa todos y cada uno de los elementos que componen la estructura del sistema "ball and beam" y, especialmente, explicar el diseño de las piezas que serán impresas en tres dimensiones, para así poder facilitar posibles reproducciones del prototipo o futuras mejoras del mismo.

2.1 Características generales

El sistema "ball and beam" debe estar compuesto por dos elementos fundamentales: una bola y una barra o conjunto de ellas para que, de alguna forma, pueda rodar la bola sobre ellas. Para este Trabajo, debido a circustancias que se detallarán en capítulos posteriores, relacionadas con la forma de medir la posición de la bola en tiempo real, se han elegido dos barras dispuestas la una paralela a la otra, de modo y forma que formen un carril por el cual pueda desplazarse la bola en el único grado de libertad que se le deja libre, esto es, en la dirección longitudinal de las barras.

Tras varias pruebas con distintas barras, finalmente se escogen como las más óptimas dos redondos huecos de aluminio bruto de un centímetro de diámetro y un espesor de un milímetro.

Ilustración 3 Barra de aluminio bruto hueca

Con el objetivo de conseguir un sistema lo más visual posible y que permita trabajar con el máximo rango de referencias, se usarán las barras tal y como se adquieren, aprovechando así toda su longitud. Se considera que para una bola de tamaño razonable, una buena separación entre centros es 25 mm, por lo que los extremos de las barras estarán separados 15 mm, logrando así dejar suficiente espacio como para poder utilizar bolas de diferentes materiales y diámetros hasta encontrar la que mejor resultados proporcione.

Cabe destacar también, que es indispensable algún elemento plano, como por ejemplo una plancha de madera, que tenga al menos 20 cm más que las propias barras, y que pueda ser utilizado como base de todo el sistema. Se tendrá en cuenta que sea lo suficientemente resistente como para que soporte el peso del sistema a la vez que permita que se le abran agujeros para poder fijar las piezas impresas en tres dimensiones a él.

Una vez están claras las partes del sistema que deben ser adquiridas directamente en un comercio, se pasa a relatar el diseño propiamente dicho. Tal y como se indicó en el capítulo de Introducción, de las dos opciones estructurales posibles, se ha escogido la que se ve en la Ilustración 1, es decir, aquella en la que se utiliza una biela y una excéntrica para trasladar los movimientos del motor a las barras.

Es por ello que será fundamental el diseño de las piezas necesarias para estas dos partes del sistema en un programa de edición de piezas en tres dimensiones para poder ser posteriormente impresas. En la parte del sistema que se denotará como fija, por ser la que no tiene el motor, es necesario colocar una pieza que una las dos barras y que a su vez, pueda bascular de forma solidaria con estas sobre dos pies que estarán fijados a la base. En este extremo debe tenerse también en cuenta el lugar donde irá el sensor.

En la parte denominada como móvil, por ser en la que se colocará el motor, se debe diseñar una pieza parecida a la del otro extremo, que una ambas piezas, una excéntrica que ruede solidaria al motor, una barra intermedia que una la pieza de las barras con la excéntrica y una pieza para poder sujetar el motor y anclarlo debidamente a la base.

2.2 Diseño en 3D

Cuando ya se tiene seguridad de las piezas que será necesario imprimir, es el momento de diseñarlas. Para ello, tras hacer un estudio de los distintos programas disponibles en el mercado, se ha optado por utilizar FreeCAD en su versión 0.18. Este es un programa de software libre, por lo que puede ser gratuitamente descargado por cualquiera que lo desee desde su página web oficial tanto para Windows como para Mac o Linux, [7], por lo que cumple la premisa de este proyecto de utilizar hardware y software que estén al alcance de todos, para así poder ser utilizado, retocado o mejorado en un futuro por aquel que lo desee o necesite.

Ilustración 4 Logo de FreeCAD

Además cabe destacar que es la mejor opción para empezar a diseñar si no se han utilizado antes otras aplicacaciones, por su gran facilidad de manejo y la amplia gama de ejemplos y tutoriales que se pueden encontrar en la red. Todo ello, sin dejar atrás ninguna característica necesaria para la labor que se quiere llevar a cabo presente en otros programas. En el caso de que la persona que vaya a utilizarlo lo vaya a hacer por primera vez, desde aquí se recomienda la visualización y realización del curso gratuito de FreeCAD impartido por "Obijuan Academy" [8].

Se pasa ahora a detallar el diseño de cada una de las piezas, separándolas en las dos partes en que se divide el prototipo, pues se considera que es la forma más adecuada para poder explicar todo lo necesario para caracterizarlas de forma unívoca.

2.2.1 Parte fija

La pieza que une las barras en el extremo sin motor debe tener dos agujeros pasantes separados sus centros 25 mm entre sí y simétrico el uno del otro respecto al eje vertical de la pieza. Cada uno de los cuales tiene un diámetro de 11 mm, lo que supone 1 mm más del diámetro de las barras, ya que es necesario dejar holguras en todas las piezas en las que deban encajar otras para que los desajustes propios de las impresoras 3D no provoquen que haya que lijar a posteriori.

Por otro lado, se le deberán dejar dos salientes a cada lado para introducir en ellos la parte interna de los rodamientos que estarán colocados por su parte externa en los pies que van unidos a la base. Como se ve en la siguiente imagen, se ha colocado un saliente a unos 4 cm del extremo trasero, donde se colocará, en un pricipio, el sensor láser para medir la distancia de la bola.

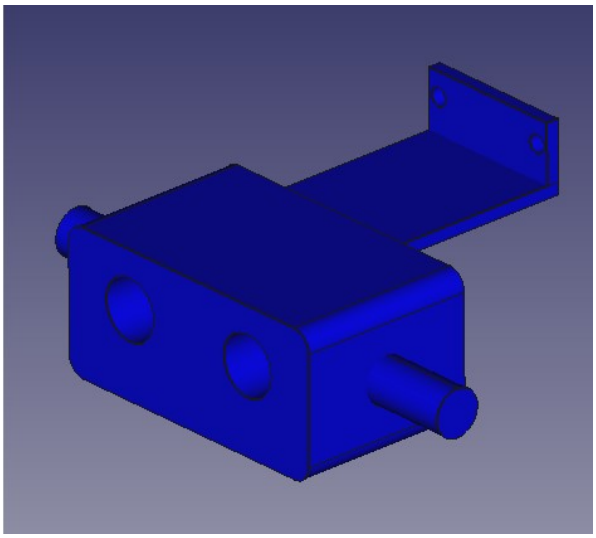

Ilustración 5 Pieza que une las barras en el extremo fijo

Como ya se ha dejado entrever en las líneas anteriores, para lograr los movimientos relativos de unas piezas frente a otras, tratando de evitar al máximo el rozamiento entre ellas, se ha optado por usar rodamientos. Concretamente, rodamientos como el que se detalla en el plano de la ilustración siguiente, que son típicamente usados para monopatines y pueden ser fácilmente encontrados.

Las dimensiones de los mismos son 22 mm de diámetro exterior (*D*), 8 mm de diámetro interior (*d*) y 7 mm de espesor total (*B*). La precisión de estas medidas no es infinita, por lo que se le han dado las holguras adecuadas cuando ha sido necesario.

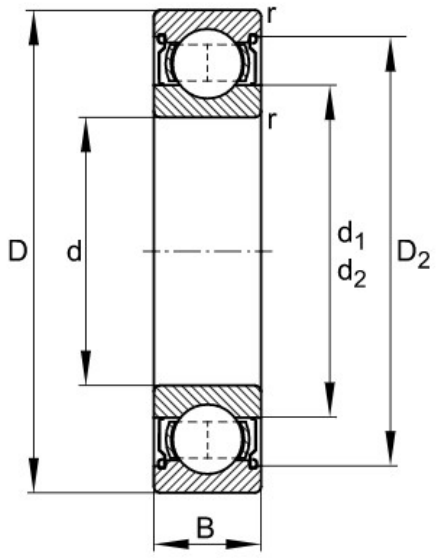

Ilustración 6 Plano del rodamiento

La pieza anterior debe moverse solidaria a las barras según vaya ordenando la parte móvil, por lo que se hace necesaria la inclusión de dos piezas fijas que vayan ancladas a la base por su parte baja y que en el extremo superior tengan el hueco para que encaje un rodamiento en el que se introducirán los salientes de la pieza anterior. Como es lógico, ambos pies son iguales, por lo que basta con diseñar uno e imprimirlo dos veces.

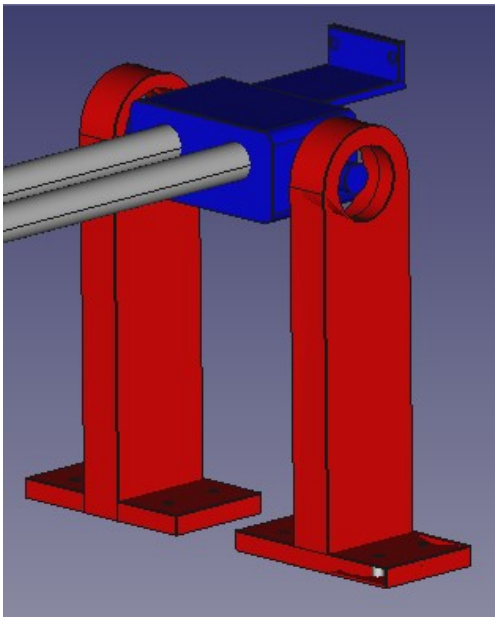

Ilustración 7 Diseño de la parte fija

Se muestra en la Ilustración anterior el montaje en 3D de la parte que no lleva el motor al completo, a excepción de la inclusión de los rodamientos y el sensor láser. Como puede observarse, cualquier movimiento en el extremo de las barras que no se ve en la figura, se puede realizar libremente sin oponer resistencia alguna por esta parte.

2.2.2 Parte móvil

Para la parte denominada móvil, lo primero que hay que diseñar es una pieza parecida a la primera que se mostró en la parte anterior, para poder unir las barras en este extremo. Sin embargo, en este caso únicamente se debe incluir un extremo para rodamiento en lugar de dos, ya que el motor está colocado en uno de los lados y todo el peso recae sobre él.

Con esto ya se tendrían las barras unidas y esperando a ser movidas por el servomotor. Este debe tener un apoyo que esté atornillado a la base para evitar que en un momento dado pueda moverse. Tal y como se ve en la imagen siguiente, se aprovecharán los agujeros que trae el propio servomotor, cuyo modelo ha sido obtenido de [9], para poder fijarlo a la pieza. Además, se ha dejado una abertura que atraviesa la pieza de lado a lado, con el objetivo de que si no fuera suficiente la sujeción, poder aumentarla con algún elemento externo, como por ejemplo unas bridas.

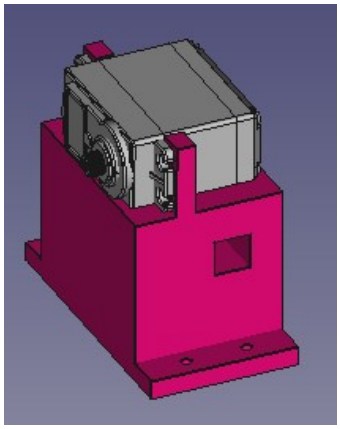

Ilustración 8 Base del servomotor

En el extremo de este servomotor, que lleva un aspa en forma de estrella de cuatro puntas, se debe colocar un disco que tenga el hueco para dicha estrella por un lado y por el otro, tenga un saliente que pueda ser introducido en un rodamiento. Para intentar que la amplitud de los ángulos sea la máxima posible, este saliente, se coloca en un extremo del disco, que tendrá 10 cm de diámetro.

Finalmente, los dos salientes de los que se ha hablado (el de la pieza que une las barras y el del disco con la excéntrica) deben ser unidos por una barra que tenga en ambos extremos un rodamiento. Esta pieza intermedia hará las veces de biela y permitirá el movimiento hacia arriba y hacia abajo de la pieza que une las barras, mientras el motor realice giros sobre su propio eje, teniendo así el mecanismo terminado, tal y como se puede ver en la siguiente Ilustración. Se hacen dos modelos de esta pieza, de diferentes longitudes, para poder probar ambos y utilizar finalmente el que mejores amplitudes de ángulo dé en la realidad. El más corto será de 7 cm de largo, con una distancia entre centros de los rodamientos de 4,5 cm, y el más largo medirá unos 11 cm y la distancia entre los centros de los rodamientos será de 8,5 cm.

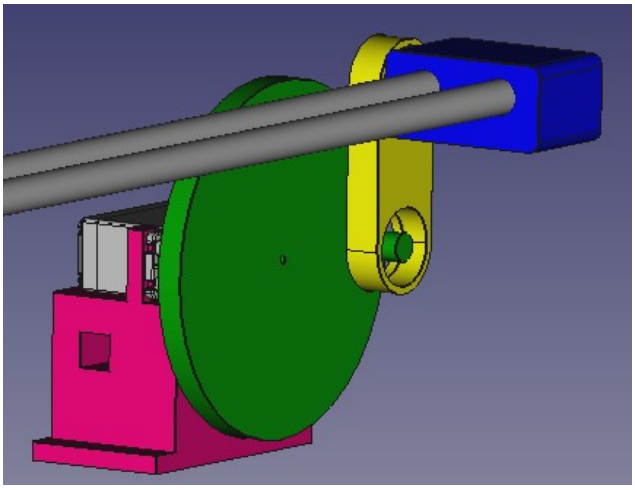

Ilustración 9 Parte móvil

2.2.3 Sistema completo

Para concluir este apartado de diseño de piezas en tres dimensiones, se considera interesante añadir una imagen del sistema "ball and beam" al completo en 3D, en el que se ha incluido, además de las piezas que se van a imprimir, las barras, el servomotor y una bola, para hacerlo lo más visual posible y que se parezca al máximo al sistema real.

Puede observarse como la entrada del sistema será el ángulo de las barras, que será modificado por el ángulo que en cada momento tenga el motor, y esto provocará cambios en la salida, que es la posición lineal de la bola medida desde el extremo en el que está colocado el sensor (que no se ve en la ilustración).

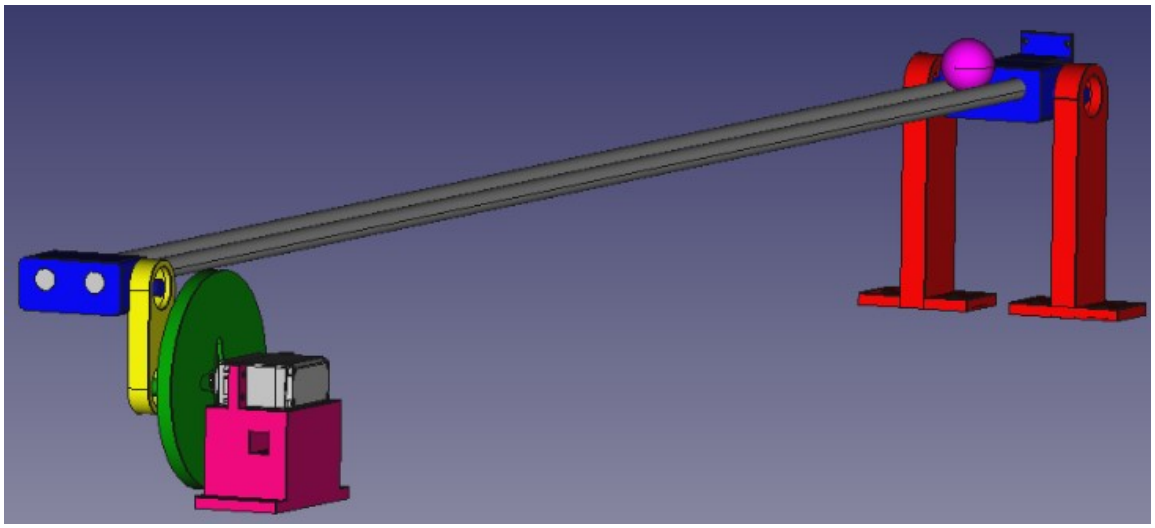

Ilustración 10 Modelado en 3D del sistema "ball and beam"

2.3 Impresión 3D

Una vez que se tienen todas las piezas diseñadas en FreeCAD, se deben exportar una a una a formato STL, que es el más común en el ámbito de la impresión 3D. Tras esto, estos archivos se pasan a un programa de laminado que directamente se introduce en la impresora 3D.

La impresora utilizada ha sido una de la marca PRUSA RESEARCH, concretamente el modelo i3, al ser esta la disponible por el tutor de este Trabajo en el Departamento de Ingeniería de Sistemas y Automática, que tal y como aparece en [10],utiliza filamento de plástico de 1,75 mm.

Ilustración 11 Impresora 3D PRUSA RESEARCH i3

En general, no se han presenteado grandes problemas a la hora de la impresión al tratarse de piezas no demasiado esbeltas en ninguna de sus direcciones y por tanto, no excesivamente grandes. Sin embargo, a la hora de imprimir el disco con la excéntrica que va unido al servomotor, en la primera capa apareció el efecto conocido como "warping", que en su traducción literal al español significa pandeo, y tal y como se refleja en [11], es un efecto que puede provocar problemas de adherencia entre capas y suele ser debido a un gradiente de temperaturas elevado entre la del plástico en el extrusor y la del panel sobre el que se va depositando el mismo.

Este problema fue rápidamente resuelto con un aumento de la temperatura de la plancha para acercarla más a la de extrusión y con una reducción en la velocidad. Con esto, a partir de la segunda capa ya no se observaron más imperfecciones y la pieza se pudo imprimir de forma correcta al igual que todas las demás.

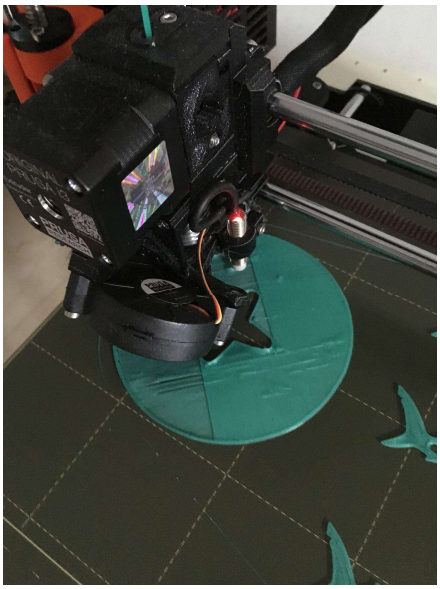

Ilustración 12 Detalle de la impresión de una de las piezas

3 HARDWARE

El ordenador nació para resolver problemas
que antes no existían.

Bill Gates

UNA vez que se tienen las piezas del sistema para poder construirlo de forma completa, se procede al montaje de las mismas tal y como se ha indicado en los capítulos anteriores, usando para ello todos los elementos que ya han sido descritos además de las piezas impresas en 3D. Tras esto, deben decidirse los dispositivos electrónicos que van a ser usados tanto para controlar el sistema como los sensores y actuadores del mismo. La enumeración de todas las posibilidades estudiadas y la argumentación de las elecciones finales son los objetivos de este capítulo.

3.1 Controlador

En cuanto a placas electrónicas que se utilicen para controlar sistemas del estilo del que nos ocupa en este Trabajo, el mercado está abarcado, casi en su totalidad por el archiconocido Arduino. Esta compañía de desarollo de hardware y software libre tiene como producto base la placa Arduino UNO, la cual está basada en el microcontrolador ATmega328P y según [12], consta de 14 pines de entrada/salida digitales, 6 entradas analógicas y conexión USB con el exterior, entre otras funcionalidades.

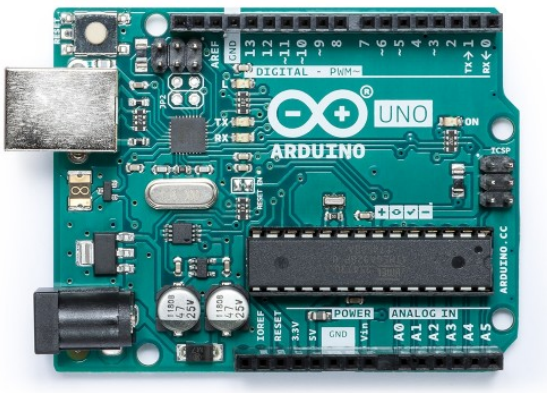

Ilustración 13 Placa Arduino UNO

El porqué del amplio uso de Arduino podría resumirse en una buena base que inicia la propia compañía, creando placas de relativo bajo coste, reutilizables, versátiles y con software libre; que a su vez se realimenta de forma positiva con la gran comunidad de usuarios que a nivel mundial aportan su conocimiento para que se amplíe el de los demás extendiendo el software original, creando nuevas librerías, resolviedo problemas en foros públicos, etc. Por todo ello, para la implementación del sistema "ball and beam", será lo que se utilice.

3.2 Actuadores

Como ya se ha indicado sucintamente en alguno de los capítulos anteriores el sistema cuenta con un solo actuador que debe ser capaz de producir un giro en las barras para que el ángulo que estas forman con la horizontal se vea modificado. Como era de esperar, el dispositivo electrónico que cumple esa funición es un motor. Sin embargo, existen distintos motores de diferentes tipos, que a primera vista, podrían cumplir las funcionalidades requeridas.

En primer lugar, se presenta la posibilidad de usar un motor paso a paso. Según [13], los motores paso a paso son dispositivos electromagneticos, rotativos e incrementales que convierten pulsos digitales en rotación mecanica.

Ilustración 14 Motor paso a paso

Efectivamente, podría decirse que hace las veces de convertidor digital-analógico, ya que con entradas de impulsos dados por sistemas digitales, es capaz de dar una salida analógica que lo posicione en un ángulo concreto.

Si bien tiene una serie de beneficios como su gran precisión, la alta confiabilidad o la falta de mantenimiento al no tener escobillas, también presenta algunas restricciones que hacen pensar en otras posibilidades de implementación. Entre otras, cabe destacar el elevado coste en relación con otras opciones y la complejidad relativa de su uso, requiriendo el apoyo de drivers para el control de su posición.

La segunda de las posibilidades que se estudia, y que ya se adelanta que será la escogida para ser usada en el sistema real es un servomotor. Estos dispositivos electrónicos están formados por tres partes fundamentales:

- Un transductor electromecánico, normalmente un motor de corriente continua con escobillas y de imanes permanentes.
- Un sensor de posición, normalmente un potenciómetro, con el objetivo de tener una realimentación de lo que ocurre en realidad.
- Una electrónica de control para aplicar un lazo de realimentación negativa que use los dos puntos anteriores.

Como se ve, los servomotores utilizan control para afinar al máximo la posición que se le pide en cada momento, por lo que alcanza de manera bastante exacta el ángulo que se le está pidiendo en cada instante de muestreo. Además, estos dispositivos trabajan bien a altas velocidades y aceleraciones, cuando se les solicita que hagan cambios bruscos de posición entre un punto extremo y el contrario, como se prevee que se necesite en el sistema "ball and beam". Por otro lado, cabe destacar como grandes ventajas de los servomotores su bajo coste gracias a la gran oferta que existe en el mercado actual de los componentes electrónicos y la facilidad para su uso en sistemas como al que se refiere este Trabajo.

Por todo lo expresado anteriormente, además de por el amplio conocimiento que sobre estos dispositivos hay disponible de forma libre, incluido software para su utlilización, será la tecnología utilizada para la actuación del sistema.

Existen en el mercado gran cantidad de servomotores con diferentes indicaciones y, consecuentemente, distintos precios. Se enumeran a continuación los tres modelos que se han estudiado incluyendo sus características principales, ordenados de menor a mayor calidad.

- **Microservo SG90**

Es uno de los servomotor más comunes para trabajar con Arduino y por ello suele estar incluido en la mayoría de los kits comerciales. Es de muy bajo coste, por lo que las prestaciones que ofrece no son muy buenas. Los engranajes de su interior son de plástico, por lo que aumenta la probabilidad de que se rompan ante un movimiento brusco o en caso de forzarlo por error.

Ilustración 15 Servomotor SG90

Como características principales se destaca que soporta un torque de hasta 1,8 kg·cm, pesa 9 g y trabaja entre 4,8 y 6 V.

- **Servo SM-S2309S**

El segundo servomotor estudiado es de características parecidas al anterior, presentando de igual manera el problema de tener engranajes de plástico. El peso es levemente superior, mientras que el torque máximo que es capaz de dar es tan solo de 1,3 kg·cm.

En general, las características de los dos servomotores anteriores no cumplen los requisitos necesarios para ser implementados en el sistema. Principalmente, la reducción de la durabilidad que provoca el uso de engranajes no metálicos hace que haya que descartarlos, ya que al estar el peso de las barras y la bola soportados por un solo lado de las mismas y directamente sobre el servomotor, cualquier movimiento brusco podría estropear un engranaje, dejando en consecuencia inutilizado el prototipo al completo.

- **Servo MG996R**

En consecuencia, se busca en el mercado un servomotor que tenga engranajes metálicos y por tanto, alta robustez. Uno comúnmente utilizado con esta característica es el servomotor MG996R, que tiene un torque máximo de entre 9,4 y 11 kg·cm trabajando entre 4,8 y 7,2 V. Como puede verse, supone un aumento de hasta diez veces en las prestaciones de los anteriores, evitando así que la falta de torque sea una restricción de diseño en el sistema. Sin embargo, la contrapartida es que su peso es de 55 g, aspecto que debería tenerse en cuenta si fuera a estar colocado en partes móviles del prototipo, pero no afectando en este caso al estar el servomotor fijado a la base del sistema.

Ilustración 16 Servomotor MG996R

Sabiendo que se va a utilizar este servomotor, se consulta consecuentementesu hoja de características [14], para estudiar el cableado que necesita y, como puede verse en la imagen, consta de tres cables distintos. Dos de ellos son, respectivamente, para la alimentación (que se conectará en principio al pin de 5 V que proporciona la placa Arduino UNO) y para la tierra. El tercero de los cables es para el control del servo, que aunque se explicará de forma más detallada en el capítulo de Software, será mediante PWM.

Ilustración 17 Pines del servomotor

3.3 Sensores

Para la construcción del sistema "ball and beam" hace falta tener medidas de dos variables distintas, por lo que harán falta dos sensores diferentes. Por un lado, la salida principal del sistema es la posición lineal de la bola, medida desde un extremo de las barras y en la dirección longitudinal de las mismas, la cual se adelanta ya como la mayor dificultad del sistema: saber dónde está la bola en cada mometo. Por otro lado, es también conveniente medir el ángulo de inclinación de las barras con respecto a la horizontal, con el objetivo de tener en un futuro realimentación de si la posición angular que el servomotor está proporcionando, se corresponde con la que realmente es necesaria para posicionar la bola en la referencia solicitada. Para mayor claridad, se estudiarán ambas necesidades por separado. Por último, se añadirán los dispositivos necesarios para poder realizar cambios en la referencia de forma interactiva.

3.3.1 Medida del ángulo de las barras

Como ya se ha comentado, se pretende medir la inclinación de las barras con repecto a la horizontal. El dispositivo electrónico que mejor se adapta a lo que se requiere es, sin duda, una Unidad de Medida Inercial, conocida por sus siglas en inglés como IMU. Este chip cuenta en su interior con sensores internos que miden aceleraciones y velocidades en cada uno de los ejes perpendiculares entre sí, concretamente un acelerómetro y un giroscopio, respectivamente, para cada uno de los tres ejes principales. Estas medidas pueden ser transformadas mediante cálculos que se realizan en el código del programa a posiciones angulares.

Con este sencillo dispositivo, se puede tener en tiempo real el valor de la posición de cada uno de los seis grados de libertad que se tienen en el espacio. Si se mantiene fijada a otro elemento que no se mueve linealmente, sino que únicamente rota o gira respecto a otro, las medidas se reducen a los tres ángulos de Euler, conocidos como Roll, Pitch y Yaw.

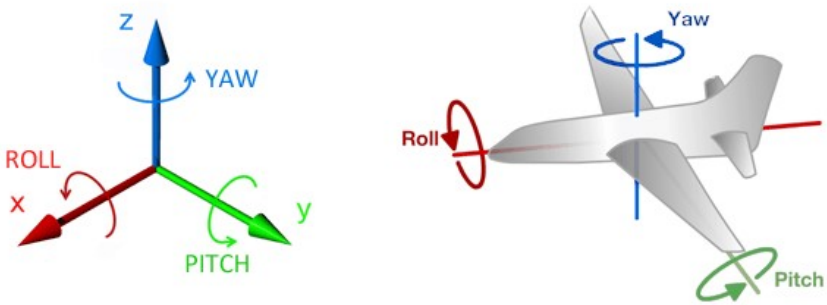

Ilustración 18 Ángulos de Euler

En el caso del sistema "ball and beam", el uso de la IMU se reduce aún más, ya que se usa solamente el ángulo Roll, que es el que describe la inclinación de las barras sobre la línea horizontal que formarían si los dos extremos de las mismas estuviesen exactamente a la misma altura.

De las distintas IMUs disponibles en el mercado, se ha optado por adquirir la que responde al modelo conocido como MPU 6050, que suele ser la más utilizada en proyectos de parecido alcance y especificaciones que el presente, por tener una óptima relación entre calidad y precio.

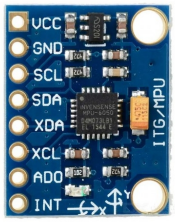

Ilustración 19 Unidad de medida inercial (IMU)

De los pines que tiene, únicamente serán usados los primeros cuatro, que corresponden a tensión de alimentación (3,3 V), tierra, SCL y SDA, respectivamente. Estos dos últimos son usados por el protocolo de comunicación I2C, tal y como se detallará en el siguiente capítulo.

3.3.2 Medida de la posición de la bola

Es capital para poder ni siquiera plantearse utilizar un controlador, conocer la posición de la bola en cada momento. Este punto es el más crítico de todo el sistema, ya que no existe una solución completa, uniforme y extendida que funcione a la perfección y, por tanto, que no presente problemática alguna. En consecuencia, se han estudiado tres posibles opciones que se van a presentar a continuación, siendo la última de ellas la finalmente elegida.

3.3.2.1 Sensor de ultrasonidos

Una de las opciones usadas por los desarrolladores de sistemas y que se encuentra comúnmente en la bibliografía, sobre todo de Trabajos de Fin de Grado sobre el sistema "ball and beam" es el uso de un sensor con tecnología de ultrasonidos, concretamente, el módulo HC-SR04.

Ilustración 20 Módulo HC-SR04

Este sensor suele dar buenos resultados cuando se quiere detectar la presencia de un objeto que se mueve en un espacio libre de obstáculos, ya que según [15], tiene un cono de propagación de las ondas de sonido de unos 30º, por lo que si hubiera algún objeto distinto al que se quisiera medir en medio, no sabría distinguir a cual de ellos estaría detectando. Esto ocurriría con las barras del sistema "ball and beam" que se ha montado, ya que las ondas de ultrasonido rebotarían en los primeros centímetros de las barras, haciendo casi imposible la detección segura y eficaz de la posición de la bola. Por todo ello, se decide descartar este dispositivo.

3.3.2.2 Medida de la resistencia interna del sistema

Con el objetivo de utilizar un método alternativo, que del que no se ha encontrado utilización en este tipo de sistemas en la bibliografía moderna, se presenta la posibilidad de medir la posición de la bola usando la resistencia eléctrica del circuito que esta cierra con las barras, motivo por el cual estas son metálicas, como se adelantó en el capítulo anterior. Para ello, como es lógico, debe garantizarse que la bola sea de igual forma metálica, para que se pueda cerrarse así el circuito.

Si se coloca el instrumento de medida de la resistencia en un extremo de las barras, por ejemplo, un polímetro, cuando la bola este en ese punto inicial, la resistencia será la mínima; mientras que cuando la bola esté en el extremo contrario de las barras, al aumentar la longitud del "circuito", la resistencia será mayor.

Además, desde un punto de vista teórico, la progresión del valor de la resistencia con la distancia de la bola al extremo inicial será lineal, ya que:

$$R = \frac{L}{\sigma A} = \frac{L}{cte}$$

Al ser la conductividad del material y la sección de las barras sendos valores constantes.

El primer problema que se presenta es que para conseguir que la medida sea correcta, el contacto de la bola con las barras debe ser suficientemente bueno. Se realizan pruebas con distintos materiales para las barras como acero al carbono o acero inoxidable. Sin embargo, al tener un recubrimiento para evitar la corrosión, este impide que se cierre el circuito con la bola, dando las mediciones con el polímetro únicamente valores espúreos. Es por ello que se opta por usar un tercer material: aluminio bruto, que cumple la especificaciones de ser buen conductor y no tener recubrimiento, obteniendo ya medidas constantes.

En cuanto a la tipología de las barras, se trabaja tanto con macizas como con huecas para comprobar cual de ellas tiene mejor respuesta. Se observa que cuando las barras son macizas, al tener que colocar los extremos del polímetro en el exterior de las mismas, se produce ruido provocando que no sea válida la medida. Por otro lado, cuando se usan barras huecas y se colocan los extremos del instrumento por dentro de las mismas, se observan medidas constantes que tienen sentido en la realidad. No se ha podido demostrar teóricamente el porqué de este fenómeno, si bien se cree que podría ser debido a interferencias electromagnéticas que se acentúan al exponerse el extremo del cable por el exterior de las barras o a un cierto recubrimiento que en realidad sí tienen las mismas por el exterior y que sea pequeño como para no interrumpir un buen contacto, pero suficiente como para que las medidas no sean correctas. Por todo lo anterior, unido al hecho de la reducción de peso que suponen, se usan las barras huecas de aluminio bruto como la única opción viable.

En cuanto a la bola, al ser imprescindible el hecho de que sea metálica, se usa en primera instancia una bola de acero obtenida como parte de un rodamiento industrial, garantizando así que su superficie sea completamente lisa y facilitando por tanto, la falta de rozamiento con las barras.

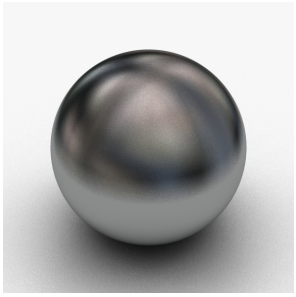

Ilustración 21 Bola de acero

Sin embargo, al intentar medir la resistencia con el polímetro, se ve que tampoco da valores concretos, por lo que se entiende que la bola debe tener también algún recubrimiento antioxidante para que las prestaciones en el objeto al que pertenecía sean las óptimas. Para solucionarlo, se le va a realizar un tratamiento invasivo añadiéndole un compuesto ácido para que reaccione con ella y elimine el recubrimiento. Efectivamente, se introduce la bola en un recipiente que se rellena con HCl, conocido como ácido clorhídrico o agua fuerte, hasta que quede completamente cubierta.

Sin entrar demasiado en aspectos químicos, el acero inoxidable tiene una capa de óxido de cromo que es la que reacciona con el ácido clorhídrico respondiendo a la siguiente reacción química ajustada:

$$Cr_2O_3 + 6HCl_{(aq)} \rightarrow 2CrCl_3 + 3H_2O$$

Como se ve, los productos de la reacción son agua y Cloruro de Cromo (III), que es un polvo de color morado que se deposita en el fondo y prácticamente no se ve a simple vista. Con este tratamiento, se consigue que se quede el acero en bruto y la conducción sea lo suficientemente buena.

Con todos los materiales adecuados, se procede a medir la resistencia con el instrumento de medida que se pretendería usar, es decir, el propio Arduino, que a través de sus entradas analógicas, puede medir la tensión que hay entre dos puntos que se le conecten. A partir de este dato, existen dos tipologías de circuito para obtener la resistencia en la que cae dicha tensión, los cuales se detallan a continuación.

- **Divisor resisitivo**

En esta sencilla circuitería, conocida por todos, se puede calcular una resistencia variable conociendo la tensión de alimentación, Vcc, y la resitencia fija de calibración, Rc, a partir de la tensión variable, Vo, siguiendo la siguiente ecuación.

$$Vo = \frac{Rc}{R + Rc} Vcc \Rightarrow \frac{Vcc}{Vo} = \frac{R + Rc}{Rc} \Rightarrow R = Rc\left(\frac{Vcc}{Vo} - 1\right)$$

Debe tenerse en cuenta que el valor de Vo vendrá dado por los valores analógicos de Arduino, esto es, entre 0 y 1023, por lo que habrá que pasarlo a valores de voltios entre 0 y Vcc de la forma: $Vo = \frac{Vcc}{1023} lectura$

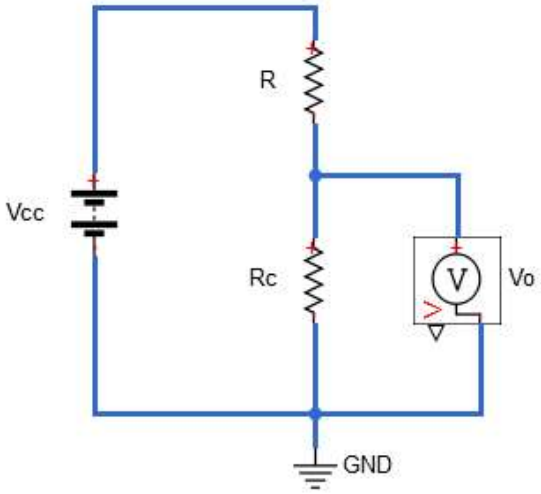

Ilustración 22 Divisor resistivo

- **Puente de Wheatsone**

La segunda de las circuiterías posibles para medir la resistencia del circuito que forman las barras y la bola es un puente de Wheatstone como el que aparece en la siguiente figura. En esta tipología, los valores de las resistencias R_1, R_2, R_3 y de la tensión de alimentación Vcc son conocidos y mediante la medida de la tensión diferencial $Vo = Vb - Va$, puede hallarse el valor de la resistencia Rx.

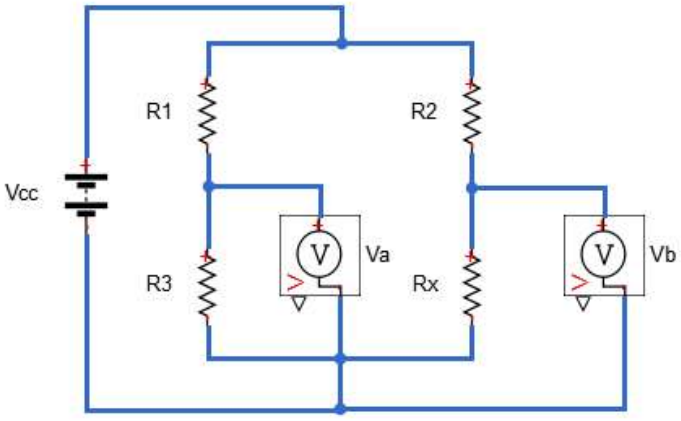

Ilustración 23 Puente de Wheatstone

Usando para ello la siguiente expresión:

$$Rx = R_2 \frac{R_3 Vcc - (R_1 + R_3)Vo}{R_1 Vcc + (R_1 + R_3)Vo}$$

Teniendo en cuenta que la tensión de alimentación del Arduino es $Vcc = 5V$, solo deben elegirse los valores de las resistencias fijas, que deben ser aproximadamente del orden de la resistencia que se pretende medir, con el objetivo de que las primeras no hagan que la segunda se convierta en despreciable. Sin embargo, esta selección no debe hacerse a la ligera, ya que las especificaciones técnicas del Arduino UNO, detalladas en [12], establecen que la intensidad máxima de corriente continua que soportan los pines de entrada/salida es de 20 mA. Esto provoca que la resistencia de calibración del divisor resistivo sea de, al menos, 330 Ω y las del puente de Wheatstone de aproximadamente el mismo valor.

Todo esto hace que, al ser la resistencia que se quiere medir de muy bajo valor (inferior a 1Ω), al implementar cualquiera de los dos circuitos en la realidad (labor que se ha realizado durante la realización de este Trabajo), el resultado que daba Arduino era siempre prácticamente el mismo independientemente del punto en el que se encontrara la bola. En consecuencia, se concluye que el uso de la resistencia interna del sistema como método para medir la posición de la bola debe ser descartado y se deja par futuras ampliaciones del presente Trabajo el estudio de la posibilidad de aumentar la resistencia de las barras y la bola de modo y forma que alguno de los circuitos anteriores pueda medirla de forma correcta.

3.3.2.3 Sensor láser infrarrojo

Tras haber descartado las dos propuestas anteriores, solo queda la posibilidad de usar un sensor láser infrarrojo para medir la posición de la bola. Según [16], muchas han sido las tecnologías estudiadas para medir distancia con sensores láser; sin embargo, en la actualidad, los que están consiguiendo mejores prestaciones, en gran medida por haber avanzado mucho en la reducción de tamaño y coste, son los que utilizan la tecnología de tiempo de vuelo, conocido por sus siglas en inglés como ToF. Estos sensores miden la distancia que hay entre el emisor y el objeto del que se quiere conocer la posición al cronometrar el tiempo que tardan los fotones en el trayecto de ida y vuelta entre ambos puntos del espacio.

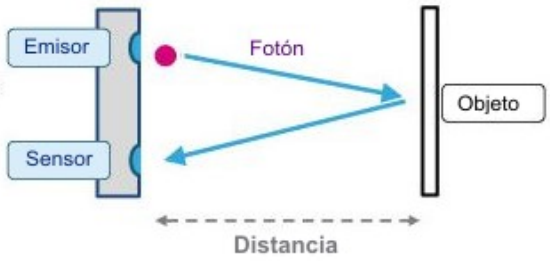

Ilustración 24 Tecnología ToF

Efectivamente, siguiendo el esquema de la ilustración, la distancia puede ser fácilmente calculada usando la siguiente ecuación:

$$distancia = velocidad_{luz} * \frac{tiempo_{fotón}}{2}$$

De los dispositivos que, utilizando esta tecnología, están disponibles en el mercado al por menor, se considera que el más adecuado para el uso que se pretende dar es el que responde al nombre de VL53L1X, diseñado por el fabricante ST y que viene a ser una versión mejorada de su antecesor el VL53L0X.

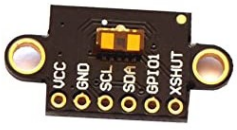

Ilustración 25 Sensor láser VL53L1X

Este sensor, según puede verse en [17], tiene una gran fiabilidad en sus medidas y puede alcanzar amplios rangos de distancia, desde los 4 cm hasta los 4 m. Además, permite adaptarlo al uso específico que se le quiera dar, ajustando distintos modos en función del rango de longitud que se quiera medir: corto, medio o largo, según puede verse en la siguiente tabla.

Tabla 1 Modos de distancia disponibles en el sensor

Modo de distancia	Máxima distancia en oscuridad	Máxima distancia con fuerte luz ambiental
Corto	136 cm	135 cm
Medio	290 cm	76 cm
Largo	360 cm	73 cm

Como se ve en la Ilustración 25, el sensor cuenta con seis pines, de ellos, en principio, solo se usarán los cuatro primeros, que corresponden con la alimentación de 5 V, la tierra, SCL y SDA, respectivamente. Estos dos últimos son usados por el protocolo de comunicación I2C, tal y como se detallará en el siguiente capítulo.

3.3.3 Medida de los cambios en la referencia

Ya ha sido detalladamente analizado y explicado el hardware estrictamente necesario para la implementación del sistema real "ball and beam" bajo estudio. Sin embargo, se considera oportuno referir aquí la explicación de como se pretenden indicar cambios en la referencia del sistema. Una opción sencilla podría ser indicar los cambios directamente desde el Monitor Serie que ofrece el programa de Arduino, pero con eso se estaría dejando inabilitado el puerto serie para otra utilidad, que ya se adelanta, será la representación gráfica de las variables del sistema a través de MATLAB.

En consecuencia, se plantean distintas posibilidades para cambiar la referencia de forma interactiva, es decir, con un dispositivo previamente programado que el usuario pueda tocar y utilizar, como por ejemplo un joystick o un mando a distancia. De las dos opciones, se cree que la más interasante es la segunda de ellas, principalmente porque siempre llama la atención la falta de cableado físico para trasmitir información y por mantener el uso de la tecnología infrarroja en el sistema.

Para el uso de un mando a distancia en Arduino, hacen falta dos dispositivos. Por un lado el mando propiamente dicho, y por el otro el receptor infrarrojo que se colocará en la base del prototipo.

Ilustración 26 Mando y receptor IR

Desde el punto de vista del cableado, el receptor IR tiene únicamente 3 pines, que corresponden a la alimentación de 5V, la tierra y la señal de control, que se realiza a través de PWM.

3.4 Esquema del cableado

Aunque en la explicación de cada uno de los dispositivos utilizados se han indicado sucintamente los pines de cada uno de ellos y la utilización de los mismos en el sistema "ball and beam", se considera oportuno para cerrar este capítulo la presentación de un esquema completo del cableado y las conexiones de todos ellos con la placa Arduino UNO. Como puede verse en la Ilustración de la página siguiente, aparece como actuador el servomotor y como sensores la IMU, el receptor IR para el mando a distancia y el sensor láser (que se ha representado como una placa de color verde oscuro, que no coincide con su formato real de la Ilustración 25); además de la propia placa de Arduino. Para su realización, se ha usado el programa Fritzing, cuya web oficial se referencia en [18].

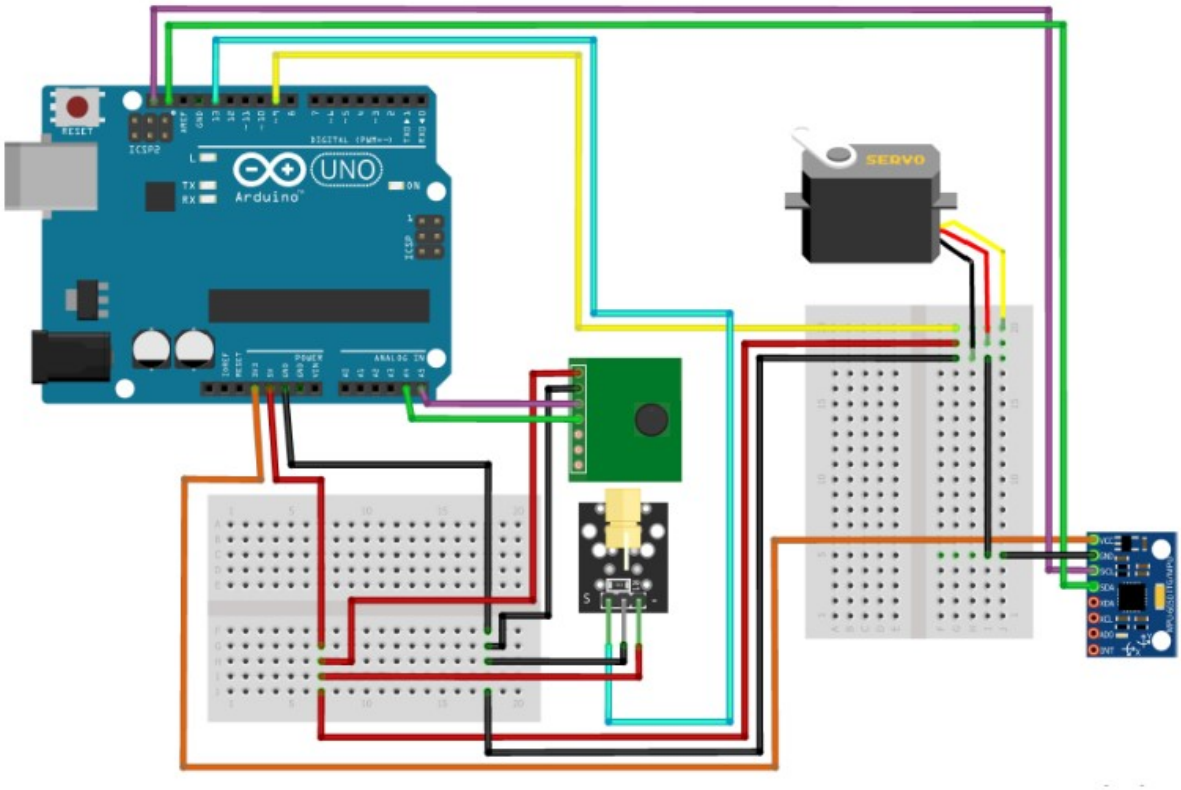

Ilustración 27 Esquema del cableado y las conexiones

Para facilitar la comprensión del mismo, y a modo de leyenda, se indica a continuación el significado y la utilización de cada uno de los distintos colores que han sido usados para diferenciar unos cables de otros.

- Rojo: tensión de alimentación de 5V, utilizado por el servomotor, el sensor láser y el receptor IR.
- Naranja: tensión de alimentación de 3,3 V, utilizado por la IMU.
- Negro: tierra, masa o tensión de referencia.
- Morado: conexión SCL, que coincide con la entrada A5 y es usado para la comunicación I2C por el sensor láser y la IMU.
- Verde: conexión SDA, que coincide con la entrada A4 y es usado para la comunicación I2C por el sensor láser y la IMU.
- Amarillo: conexión PWM en la entrada digital 9 y es usado por el servomotor para su control.
- Cian: conexión PWM en la entrada digital 11 y es usado por el receptor IR para su control.

Finalmente, destacar que debido a la alta demanda de corriente por parte del servomotor elegido, no es suficiente con conectar su alimentación al pin de los 5 V que da Arduino, por lo que se ha fabricado una fuente de alimentación externa con un cargador de telefonía que se conecta directamente a la red y da 5 V pero con 1 A de intensidad. Únicamente hay que tener en cuenta que el cable de tierra de este, debe conectarse al del servomotor y a su vez al pin GND de Arduino, con el fin de que haya una tierra común y que las referencias de tensión sean las mismas en todos los dispositivos.

En este punto se da por terminado el presente capítulo y con él la parte que podría ser incluida en el apartado de diseño del prototipo. Con un sistema ya montado y con los dispositivos necesarios para hacerlo funcionar, se está en condiciones de estudiar el software necesario para poderlo poner en funcionamiento, pasando así al siguiente capítulo.

4 SOFTWARE Y COMUNICACIÓN

Emplea tu tiempo cultivándote a través de los escritos de otros, así ganarás fácilmente lo que para ellos ha sido una dura tarea.

Sócrates

PARA cada uno de los dispositivos electrónicos que se han desarrollado en el capítulo anterior, se necesitan una serie de programas para poderlos conectar entre sí y hacerlos funcionar. Al usar una placa Arduino como controlador del sistema, será el entorno de desarrollo integrado de esta compañía, conocido como Arduino IDE [19], el utilizado para ello. En una segunda parte del capítulo se analizará la comunicación del sistema "ball and beam" con el exterior, y concretamente, el uso de MATLAB para la representación de los resultados obtenidos.

4.1 Bibliotecas utilizadas

Una de las formas más comunes de trabajar con dispositivos conectados a Arduino es usando las bibliotecas o librerias disponibles; bien las que ofrece la propia compañía desde la sección correspondiente de su página web oficial [20], bien las que implementan miembros de la amplia comunidad de Arduino en todo el mundo para facilitar trabajos futuros. En general, las bibliotecas, también llamadas librerias por una mala traducción del término inglés "library", son conjuntos de programas no ejecutables utilizados para desarrollar otros que normalmente son más complejos, ya que contienen todo lo necesario para la creación de funciones que puedan ser llamadas en cualquier momento. Así, se consigue una mayor legilibilidad de los programas y una considerable reducción del tiempo necesario para elaborarlos.

4.1.1 "Servo.h"

Para el posicionamiento del servomotor en el ángulo requerido, se hará uso de la biblioteca "Servo.h", una de las que está incluida en el propio Arduino IDE. Esta permite crear un objeto de tipo "Servo" al que se le ha dado el nombre de "myservo" en el espacio reservado para las variables globales para trabajar con él a través de las llamadas a las funciones que están predefinidas. Aunque las funcionalidades permitidas son bastante amplias, con el objetivo de usar las bibliotecas únicamente en lo estrictamente necesario y usar lo máximo posible programas de "cosecha particular", solo se usarán dos llamadas de esta biblioteca.

```
myservo.attach(9);
myservo.write(pos);
```

En la primera de ellas, que debe ser llamada en el bucle "setup", se asigna el pin digital por el que saldrá la señal de control hasta el servomotor, en este caso el pin 9; mientras que cuando se llama a la segunda, normalmente en el bucle "loop", se manda al servomotor la variable "pos" que ha sido previamente definida como un entero o un flotante y a la que se le ha asignado el valor concreto del ángulo que se pretenda alcanzar.

Como se indicó en el capítulo anterior, la placa Arduino UNO no cuenta con salidas analógicas, por lo que para poder enviar al servomotor la señal de un ángulo concreto, debe usarse la técnica conocida como PWM, que son las siglas de Pulse Width Modulation, traducido al español como Modulación por Ancho de Pulso.

Esta técnica permite generar salidas analógicas desde pines digitales gracias a la modificación del ciclo de trabajo de una señal periódica. En consecuencia, puede conseguise, por ejemplo, enviar mediante pulsos digitales que están a cero un tiempo y a uno el resto, un valor analógico de una tensión, como puede extraerse de la siguiente figura.

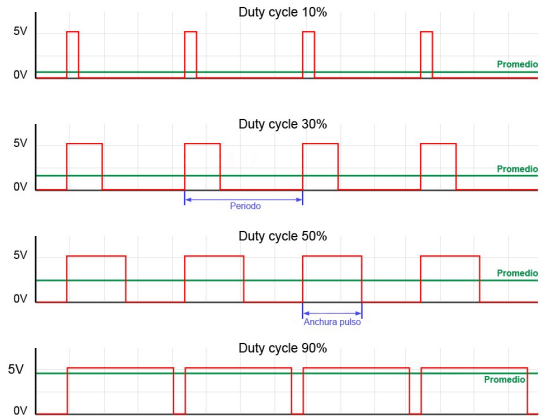

Ilustración 28 PWM

Efectivamente, el valor de salida únicamente depende de lo que se conoce como duty cycle o ciclo de trabajo, ya que con distintos anchos de pulso, se conseguirán diferentes ángulos. Este se define como el tiempo que la señal está en nivel alto respecto del total del periodo de la onda, como se observa en la siguiente expresión.

$$Duty_{cycle} = \frac{T_{ON}}{T}$$

En realidad gracias a la biblioteca "Servo.h" todo este trasfondo teórico no debe ser tenido en cuenta pues está ya previsto, pero es necesario conocerlo para poder trabajar de forma eficiente.

4.1.2 "Wire.h"

Esta biblioteca es la ofrecida por Arduino para el uso del protocolo de comunicación I2C, que será el elegido para comunicar entre sí y con el controlador tanto el sensor láser como la IMU. Antes de explicar la utilización de la biblioteca, se comentará sucintamente de forma teórica el funcionamiento del I2C.

El protocolo de comunicación "Inter-Integrated Circuit" es hoy en día, junto al SPI, un estándar de mercado en lo que a comunicación entre dispositivos se refiere. Entre sus principales ventajas, se destaca la sencillez de su cableado ya que el bus requiere únicamente de cuatro cables para su funcionamiento: uno para el envío de los datos y la información (SDA), otro para la señal de reloj (SCL) y los usados para alimentación y tierra.

Sigue una arquitectura de maestro-esclavo de tipo síncrona, en la que el maestro envía la señal de reloj a todos los esclavos para evitar así que cada dispositivo tenga la necesidad de disponer de un reloj interno. Para diferenciar unos esclavos de otros, es obligatorio que cada uno de ellos tenga asignada una dirección distinta, forzando así que no se envíen datos desde el maestro o hacia él de forma errónea. Cabe destacar que si se usa más de un dispositivo del mismo modelo, por ejemplo, dos sensores láser, habrá que modificar la dirección de uno de ellos para que no coincida con la que traen de serie.

Por último, reseñar que, como puede verse en la Ilustración de la página siguiente, en las líneas de alimentación se colocan unas resistencias llamadas de "pull-up" para elevar la tensión cuando está en reposo. Sin embargo, es un aspecto que no debe tenerse en cuenta en la práctica ya que la biblioteca "Wire.h" activa

las resistencias internas del Arduino.

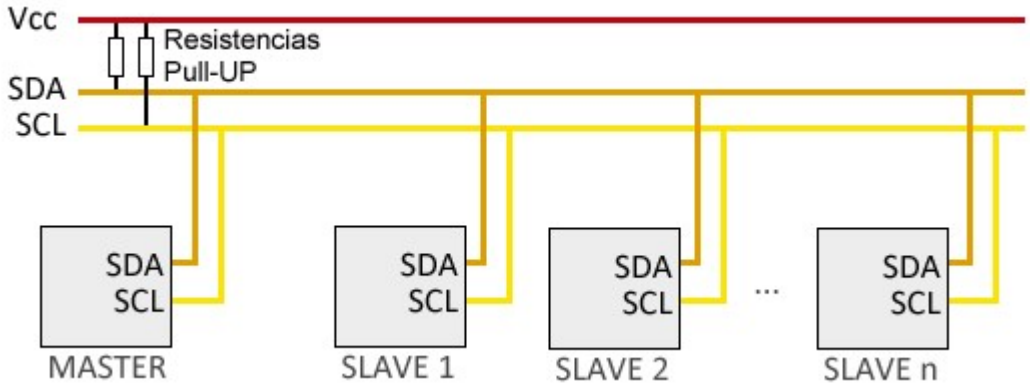

Ilustración 29 Protocolo de comunicación I2C

Teniendo ya claro el funcionamiento teórico de cómo se van a comunicar tanto la IMU como el sensor láser con el Arduino, se pasa a analizar el uso que se le ha dado a la biblioteca "Wire.h" en los programas de control del sistema "ball and beam". Serán las tres funciones siguientes las que se usarán de esta biblioteca en el "setup" del programa.

```
Wire.begin();
Wire.beginTransmission(DIR);
Wire.endTransmission(true);
```

La primera de ellas solo se llama una vez y se utiliza para inicializar la biblioteca y conectar el Arduino al bus I2C como maestro. Tras esto, siempre que se inicie una transmisión de datos, se entenderá que la dirección que se está indicando es de un dispositivo esclavo, ya que el puesto de maestro es, en principio, definitivo. Para ello, se utiliza la segunda de las funciones anteriormente relacionadas, antes de hacer cualquier actividad con alguno de los dispositivos esclavos, se debe reservar el uso del bus indicando la dirección correspondiente.

Tras haber hecho uso del sensor en cuestión, ya con la información en el maestro a su disposición, se debe cerrar la transmisión de datos mediante la tercera de las funciones anteriores. El argumento que se le pasa es un booleano que solo puede ser cierto o falso, en función de que se quiera liberar el bus después de la transmisión o mantener la conexión activa, respectivamente.

Entre estas dos últimas llamadas a función, se deberá trabajar con los dispositivos usando funciones de esta misma biblioteca o de cualquier otra que se incluya. Como ya se ha indicado, para usar de forma correcta el protocolo I2C con los dos dispositivos electrónicos del sistema "ball and beam" (sensor láser e IMU), lo primero que debe hacerse es identificar las direcciones de cada uno de ellos y definirlas al inicio del programa para poder usarlas posteriormente, tal y como sigue:

```
#define MPU 0x68
#define SENSOR 0x29
```

El uso de los comandos anteriores y de otras funciones de esta biblioteca, serán expuestos en la función programada para el uso de la IMU que se adjunta a continuación y será usada en cualquiera de los programas de control que se quieran implementar. Esta función ha sido programada por el autor, si bien está basada en distintos trozos de código disponibles en Internet y realizados por distintos miembros de la comunidad de Arduino, como por ejemplo [21]. Como se ve, la función no tiene ningún argumento de salida, pues en el propio puntero a flotante que se le entrega como argumento de entrada, se recibirá el resultado del ángulo Roll cuando se llame a la función.

Se accede al bus I2C para solicitar la lectura de cada uno de los dos sensores que tiene la IMU (acelerómetro y giroscopio). Nótese como se deja activa la comunicación al acabar las lecturas para que no pueda usarse el bus por otro dispositivo mientras la función se está ejecutando.

Se destaca también el uso de dos filtros para lograr que la medida sea lo más exacta posible. Por un lado, uno que aplica combinación sensorial con el fin de usar las medidas que han dado los dos sensores y, por el otro, un filtro que calcula la salida como una media ponderada de la medida actual y de la anterior, evitando así valores espúreos indeseados.

```
void LeeIMU(float *Roll)
{
  //Se leen los valores del Acelerómetro.
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Se pide el registro 0x3B, que corresponde al AcX.
  Wire.endTransmission(false); //Se deja activa la comunicación.
  Wire.requestFrom(MPU, 6, true); //A partir del 0x3B, se piden 6 registros.
  AcX = Wire.read() << 8 | Wire.read(); //Cada valor ocupa 2 registros.
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();

  //A partir de los valores del acelerómetro, se calcula el ángulo con la
//fórmula de la tangente.
  Acc = atan((AcY/A_R)/sqrt(pow((AcX / A_R),2) + pow((AcZ/A_R),2)))*RAD_TO_DEG;

  //Se leen los valores del Giroscopio.
  Wire.beginTransmission(MPU);
  Wire.write(0x43); //Se pide el registro 0x43, que corresponde al GyX.
  Wire.endTransmission(false); //Se deja activa la comunicación.
  Wire.requestFrom(MPU, 6, true); //A partir del 0x43, se piden 6 registros.
  GyX = Wire.read() << 8 | Wire.read(); //Cada valor ocupa 2 registros.
  GyY = Wire.read() << 8 | Wire.read();
  GyZ = Wire.read() << 8 | Wire.read();

  //A partir de los valores del giroscopio, se calcula el ángulo.
  Gy[0] = GyX / G_R;

  //Se calcula en tiempo real el incremento de tiempo que se va a utilizar.
  dt = (millis() - tiempo_prev) / 1000.0;
  tiempo_prev = millis();

  //Se aplica el filtro complementario para unir las medidas de los sensores.
  Angle = 0.98 * (Angle + Gy * dt) + 0.02 * Acc[0];

  //Se aplica el segundo filtro para mejorar las medidas.
  angle_filtrada = alpha_IMU * Angle + (1 - alpha_IMU) * angle_filtrada_ant;
  angle_filtrada_ant = angle_filtrada;

  //Se devuelve el valor del ángulo.
  *Roll = angle_filtrada;
}
```

4.1.3 "VL53L1X.h"

Al igual que la IMU, el sensor láser que va a ser usado para medir la posición lineal de la bola, también sigue el protocolo de comunicación I2C, por lo que deberán seguirse cuidadosamente las pautas marcadas en el subapartado anterior, relativo a la utilización de la biblioteca "Wire.h". Es importante reseñar en este punto, un detalle de la Ilustración 27, relativa al cableado de los dispositivos, en la puede verse que los cables correspondientes al SCL y SDA del sensor, están colocados en pines distintos a los de la IMU. Este hecho es debido a que en la placa Arduino UNO, los pines de entradas analógicas A4 y A5, corresponden, respectivamente, a SDA y SCL, por lo que pueden ser usados indistintamente.

La dificultad añadida de este dispositivo es que requiere de una biblioteca específica, cuya descarga está disponible en la red [22], para la obtenición de la medida. Para la utilización de esta biblioteca, además de su inclusión al principio del programa, se debe definir un objeto como variable global para trabajar con el sensor

(al igual que ocurría con el servomotor). Además, en el bucle de inicializaciones "setup", una vez que se ha reservado el bus I2C con la dirección predefinida del dispositivo, se debe continuar fijando el tiempo de espera tras el que el sensor debe dejar de enviar si no se recibe respuesta alguna, conocido como "timeout" en 100 ms.

Tras esto, como puede verse en el código adjunto, debe activarse el sensor con la función `sensor.init()`. Es conveniente poner un punto de seguridad aquí para que no se siguiese ejecutándo el programa si fallase la activación por cualquier motivo. Con el sensor ya inicializado, se concreta el modo de distancia que vaya a usarse siguiendo la Tabla 1, expuesta en el capítulo anterior (en este caso será el de distancia corta por ser menor a un metro todo lo que se va a medir) y se marca el tiempo de lectura del sensor en 35 ms, teniendo en cuenta que, según el datasheet del mismo, en el modo de distancia corta, este debe ser al menos de 33 ms. Finalmente, se cierra la transmisión y se libera el bus I2C para que pueda ser usado por otros dispositivos o por este mismo en otro momento.

```
void setup(){
  Wire.beginTransmission(SENSOR); //Inicia la comunicación I2C.
  sensor.setTimeout(100);
  sensor.init();
  delay(10);
  if (!sensor.init())
  {
    Serial.println("¡Fallo al detectar e inicializar el sensor!");
    while (1);
  }
  delay(10);
  sensor.setDistanceMode(VL53L1X::Short); //Modo de distancia corta.
  sensor.setMeasurementTimingBudget(35000);
  Wire.endTransmission(true); //Finaliza la transmisión y deja libre el bus.
}
```

De igual forma a como se hizo con la IMU, se considera oportuno programar una función que pueda ser llamada en todo momento desde el bucle "loop" y que devuelva en el mismo puntero a entero que se le entrega, la distancia real de la bola desde el sensor en ese tiempo de muestreo. Esta función, llamada "LeeSensor" y cuyo código se adjunta a continuación, debe comenzar solicitando el bus para la transmisión de los datos para después usar únicamente cuatro funciones de la biblioteca "VL53L1X.h". Primero, se inicia una emisión continua los datos de la distancia, con un tiempo de muestreo de, igualmente, 35 ms; tras lo que se toma una muestra y se acaba la emisión de datos. Una vez que se ha cerrado y liberado la comunicación por el bus I2C, se adquiere la distancia en milímetros medida y se pasa a centímetros para trabajar más fácilmente.

Finalmente, con el objetivo de evitar valores espúreos, propios de sensores de relativo bajo coste como el que se está utilizando aquí, se aplica un filtro similar al segundo de los usados en la IMU, en el que la salida de la función será una media ponderada del valor de la distancia actual y el anterior.

```
void LeeSensor(int *pos){

  int distancia;
  //Se inicia la lectura, y se guarda el resultado.
  Wire.beginTransmission(SENSOR);
  sensor.startContinuous(35);
  sensor.read();
  sensor.stopContinuous();
  Wire.endTransmission(true);
  distancia = sensor.ranging_data.range_mm;
  //Se pasa de mm a cm.
  distancia = distancia/10;
  //Se aplica un filtrado para mejorar las medidas.
  distancia_filt=alpha_Sensor*distancia+(1-alpha_Sensor)*distancia_filtrada_ant;
  distancia_filtrada_ant = distancia_filt;
  //Se devuelve la posición.
  *pos = distancia_filtrada;
}
```

4.1.4 "IRremote.h"

La última de las bibliotecas que debe ser añadida en el programa y, por consiguiente, que debe ser explicada en este capítulo es la referente al sensor infrarrojo usado para recibir los cambios de referencia a través del mando a distancia. Su descarga es igual de accesible que las anteriormente mencionadas, ya que su uso está más que extendido [23]. Para su utilización, es necesaria la definición de dos objetos como variables globales: uno para el receptor en sí mismo y en el que se indica a que pin PWM está conectado y otro para la decodificación de los resultados.

```
IRrecv ReceptIR(PIN_ReceptorIR); // Objeto del receptor IR.
decode_results LecturaReceptIR; // Objeto de decodificación de resultados.
```

Son, por tanto, dos las partes que hay que entender para poder usarlo con facilidad y corrección. Por un lado, se debe comprobar cada ciertos instantes de muestreo si se ha recibido alguna señal. Cabe destacar que debido a que en la tecnología infrarroja la señal no se corta y a que los tiempos de muestreo son muy pequeños, normalmente es suficiente con realizar esta comprobación uno de cada diez tiempos de muestreo.

En el caso de que se haya recibido algo, entra en juego la segunda de las pesquisas y es que, cuando se pulsa una de las teclas del mando a distancia, este envía una señal al receptor, que está codificada en hexadecimal, por lo que debe ser traducida de modo y forma que se identifique con la cadena de caracteres correspondiente. Por ejemplo, si le llega 0xFFA25D, entonces se le asignará a una variable de tipo "String" llamada "Boton" los caracteres "POWER". Este proceso se realiza en una función llamada `TraduceIR` y que debe ser añadida en el programa, como puede verse en el Anexo A, pero que no se adjunta aquí por no caer en repetición.

Una vez que se tiene la información necesaria para poder conocer qué tecla es la que ha sido pulsada, se indica que el receptor siga leyendo hasta que vuelva a recibir un nuevo dato. En la variable Boton se conoce que pueden estar guardados los caracteres de cualquiera de las teclas del mando a distancia, pero solo algunas de ellas tendrán algún tipo de utilidad en la práctica.

En este caso, se usará "POWER" para modificar el valor de una bandera que hará que el sistema se empiece a controlar en caso de no estar haciéndolo o que pare de hacerlo en caso contrario. Para modificar la referencia, que es el objetivo del uso de este dispositivo, se utilizará la tecla superior derecha del mando, "FUNC/STOP", que habilitará la recepción de una nueva referencia, la cual será redefinida como diez veces el valor de la tecla numérica que se pulse, tal y como se aprecia en el siguiente trozo de código.

```
if (ReceptIR.decode(&LecturaReceptIR))  // Si se ha recibido algo...
{
  Boton = TraduceIR(); //Se lee la tecla pulsada.
  ReceptIR.resume();  //Se continúa leyendo.
}
/* Se define qué hacer con dicha tecla */
if (Boton == "POWER" && encendido == 0)
{
  encendido = 1; //Se activan los controladores.
}
else if (Boton == "POWER" && encendido == 1)
{
  encendido = 0; //Se desactivan los controladores.
}
else if (Boton == "FUNC/STOP" && lee_ref == 0)
{
  lee_ref = 1; //Se prepara para leer una nueva referencia.
}
else if (lee_ref == 1 && (Boton == "1" || Boton == "2" || Boton == "3" || Boton
== "4" || Boton == "5" || Boton == "6" || Boton == "7" || Boton == "8" || Boton
== "9") )
{
  lee_ref = 0; //Deja de leer una nueva referencia.
  ref = 10*Boton.toInt(); //Se actualiza el valor de la referencia.
}
```

4.2 Representación de los datos

Es conocido que el propio entorno de programación de Arduino incluye dos herramientas para la representación de los datos que se vayan obteniendo en los sucesivos tiempos de muestreo con la ejecución del programa. Estos son el "Serial Monitor" para la muestra de datos numéricos y para la la recepción de los mismos a través de teclado y el "Serial Plotter" para la representación gráfica de los mismos datos. Sin embargo, este método no es el más recomendable, ya que la calidad de las gráficas es reducida y, lo que es aún más importante, no se guardan en memoria, por lo que no pueden ser utilizadas a no ser que se haga una captura de la pantalla del ordenador, algo nada aconsejable.

En consecuencia, se presenta aquí otra forma de representar los resultados que se vayan obteniendo en los sucesivos experimentos prácticos que se realicen con el sistema "ball and beam". Se utilizará para ello el programa MATLAB, al cual se le enviarán los datos desde el Arduino por el puerto serie y podrá tratarlos como si estuviesen en su "Workspace".

Es importante en este punto hacer un breve paréntesis teórico sobre la tecnología del puerto serie. Este no es más que una interfaz de comunicaciones de datos digitales, en la cual, en lugar de envíar paquetes de información, se envían los datos bit a bit. Como puede esperarse, sus ventajas fundamentales son su enome sencillez y la dificultad de equivocación o recepción errónea de la información y su principal desventaja es la lentitud que conlleva el hecho de que cada vez solo se envíe un bit. Sin embargo, en los últimos tiempos este problema ha sido satisfactoriamente resuelto dando lugar al archiconocido "Universal Serial Bus", normalmente nombrado por sus siglas como USB.

Ilustración 30 Conectores típicos de puerto serie

Efectivamente, a través del cable USB que le da alimentación al Arduino desde el ordenador personal, el primero puede devolverle al segundo los datos más importantes del experimento en ese tiempo de muestreo. Para evitar errores en cualquiera de los dos extremos, debe fijarse un orden concreto, como por ejemplo:

- Posición real de la bola
- Referencia
- Ángulo de inclinación de las barras medido por la IMU
- Ángulo de inclinación de las barras deseado por el controlador
- Ángulo de giro del servomotor

Estos datos serán enviados en el programa de Arduino en una sola línea al final del bucle "loop", que se ejecuta cada tiempo de muestreo gracias a la creación de una variable llamada "valores" del tipo "String".

```
valores =
String(pos)+","+String(ref)+","+String(Rol)+","+String(theta)+","+String(alpha);
Serial.println(valores);
```

Para poder recibir todos estos datos, lo primero que debe hacerse en MATLAB es establecer la conexión con Arduino a través del puerto serie. Por seguridad, se borran las comunicaciones que hubiera podido haber en el pasado con el puerto al que está conectado el Arduino, el cual debe obtenerse previamente y que en este caso es COM4. Con el comando "serial" se crea un objeto puerto serie que deberá tener las mismas características que se hayan establecido en el "setup" del programa de Arduino y se abre con el comando "fopen".

```
%% RECEPCIÓN DE LOS DATOS DEL SISTEMA BALL AND BEAM %%
clear all; close all; clc;

%Se borran las comunicaciones que hubiera con el puerto COM4 (al que está
%conectado el Arduino).
delete(instrfind({'Port'},{'COM4'}));
%Se crea un nuevo objeto de puerto serie con las características adecuadas.
s = serial('COM4','BaudRate',9600,'Terminator','CR/LF');
%Se abre el puerto creado.
fopen(s);
```

De todos los datos, como es lógico, lo que más interés genera es la posición lineal de la bola frente a la referencia que en cada momento se le esté dando, pues el control de esta variable es el objetivo final de este Trabajo. Es por eso que se programa que en tiempo real, es decir, mientras se está realizando el experimento, se muestre en pantalla esta gráfica, que se va actualizando constantemente, para así poder ver los posibles errores o fallos en el control de manera rápida, sencilla y directa. Para ello debe prepararse la figura que se va a ir viendo e inicializar todos los vectores en los que se van a guardar los datos. Nótese que la duración del experimento puede ser sencillamente modificada en cualquier momento si se quiere realizar uno más corto o más largo.

```
%Se establecen los parámetros de la medida.
tmax = 150; %Tiempo de captura.
rate = 30; %Número de tomas por iteración.

%Se prepara la figura que se va a ir viendo en tiempo real.
f = figure('Name','Posición de la bola y referencia en tiempo real');
b = axes('XLim',[0 tmax],'YLim',[0 80]);
l1 = line(nan,nan,'Color','r','LineWidth',2);
l2 = line(nan,nan,'Color','b','LineWidth',2);
xlabel('Tiempo','FontSize', 14)
ylabel('Posición(cm)','FontSize', 14)
title('Sistema Ball and Beam','FontSize', 16)
grid on
hold on

%Se inicializan las variables que se van a guardar.
rho = zeros(1,tmax*rate);
ref = zeros(1,tmax*rate);
roll = zeros(1,tmax*rate);
theta = zeros(1,tmax*rate);
alpha = zeros(1,tmax*rate);
```

Una vez se tienen todo lo necesario, se pasa a ejecutar un bucle en tiempo real mientras que dure el experimento en el que en cada pasada se lee el puerto serie y se guarda una nueva componente de los vectores con las variables del sistema, tras lo que se actualiza la gráfica que se está dibujando en tiempo real. Para ello, como puede verse en el siguiente trozo de código, se utiliza el comando "drawnow", en lugar del típico "plot".

```
%Se ejecuta el bucle cronometrado, previa comprobación del tiempo actual.
i = 1; t = 0;
tic
while t<tmax
    %Se mide el tiempo real al inicio de cada iteración.
    t = toc;
    %Se lee el puerto serie y se guarda como un vector que luego se
    %descompone en sus variables.
    a = fscanf(s,'%d,%d,%f,%f,%f')';
    rho(i) = a(1);
    ref(i) = a(2);
    roll(i) = a(3);
    theta(i) = a(4);
    alpha(i) = a(5);
```

```
    %Se dibuja la gráfica con la posición y la referencia.
    x = linspace(0,i/rate,i);
    set(l1,'YData',rho(1:i),'XData',x);
    set(l2,'YData',ref(1:i),'XData',x);
    %Se actualiza la figura y se incrementa el índice para indicar que se
    %ha tomado una nueva muestra.
    drawnow
    i = i+1;
end
```

Al acabar el bucle y con él, la recogida de los datos, debe procederse a cerrar el puerto serie y, posteriormente, a borrar el objeto que se creó, evitando asi posibles problemas cuando se vuelva a conectar el Arduino. Por otro lado, cuando la experiencia acabe, se crearán automáticamente una serie de gráficas con el resto de las variables, debidamente nombradas y con las leyendas correspondientes, como puede verse a continuación.

```
%Al acabar la recogida de datos, se cierra y borra el puerto serie.
fclose(s);
delete(s);
clear s;
disp('Puerto serie cerrado y borrado');

%Se representan gráficamente los datos obtenidos.
figure(2)
subplot(2,1,1);
plot(ref(2:2000),'r','LineWidth',2);
hold on
plot(rho(2:2000),'LineWidth',2);
grid
legend('Referencia \omega','Salida \rho');
xlabel('Tiempo','FontSize', 14),ylabel('Posición y referencia','FontSize',
14);
title('Control primario','FontSize', 16);
xlim auto
ylim auto;

subplot(2,1,2)
plot(theta(2:2000),'LineWidth',2);
grid
legend('Ángulo de las barras \theta');
xlabel('Tiempo','FontSize', 14),ylabel('Ángulo de las barras
\theta','FontSize', 14);
title('Acción de control','FontSize', 16)
xlim auto;
ylim auto;

%Ídem para la otra gráfica, no se adjunta aquí el código por repetitivo.
```

Para poner en práctica todo lo expresado en este apartado, basta con cargar el código del programa de control en Arduino en el que, por supuesto, se incluyan las dos líneas de código que se indicaron anteriormente para poder enviar los datos por el puerto serie desde Arduino hasta MATLAB. Posteriormente, se debe ejecutar en un mismo Script el código de MATLAB que se ha mostrado anteriormente separado en trozos para facilitar así su comprensión al lector.

Tras esto, solo será necesario activar el control con el mando a distancia y se verá como, conforme la posición de la bola va cambiando en la realidad, también lo hace en la pantalla del ordenador su representación gráfica. De igual manera, si se decide modificar la referencia usando el mando, tal y como se ha venido explicando, se observará su incremento o decremento de la misma en la gráfica, facilitando así la comprensión de lo que está ocurriendo en la realidad.

5 MODELADO TEÓRICO

Un modelo resulta siempre parcial, pero ofrece recursos para prograsar en el conocimiento.

Jean-Pierre Changeux

EL análisis de todo sistema dinámico que en un futuro se pretenda controlar, debe comenzar con el estudio mecánico del mismo que concluya con la obtención de una ecuación diferencial que describa unívocamente el comportamiento del sistema, siempre que esto sea posible

Esto es debido a que la mayoría de los controladores que hasta el momento se han desarrollado, según se obtiene de la redacción de Ogata en [24], se basan en una función de transferecia del sistema, que como es bien conocido, se define como la relación lineal entre la salida y la entrada de un sistema, por lo que será necesario linealizar la citada ecuación diferencial, aplicarle la Transformada de Laplace para trasladarla al dominio de la frecuencia, y finalmente, obtener la función de transferencia correspondiente.

5.1 Modelado de sistemas

Para ello, históricamente, los sistemas se han podido analizar desde dos puntos de vista: la mecánica vectorial y la mecánica analítica, tal y cómo describe el profesor Toscano en [25]. Con el objetivo de obtener unos resultados lo más concluyentes posibles, se estudiarán ambos caminos y se comprobará como los resultados son equivalentes.

En cualquier caso, el sistema bajo estudio está compuesto por un primer sólido formado por dos barras huecas de logitud L y masa total de ambas, M, situadas una paralela a la otra, que se mueven de forma solidaria y están separadas entre sí una distancia d, cada una de las cuales es de radio r_{ext}, con espesor e; y por un segundo sólido que es una esfera maciza de radio R y masa m, que rodará sin deslizar a través del carril que forman las dos barras anteriormente mencionadas.

Las variables que describen el movimiento de ambos sólidos son, por un lado, ρ, que se define como la distancia a la que se encuentra el centro de la esfera medida desde el extremo de las barras y por el otro, θ, la cual se entiende como el ángulo que forman las barras con la horizontal.

Es por tanto la variable de salida de nuestro sistema la posición lineal de la bola, ρ, y la variable de entrada, es decir, sobre la que se tiene capacidad de control, el ángulo θ.

De forma clara puede verse todo lo anteriormente indicado en la siguiente figura.

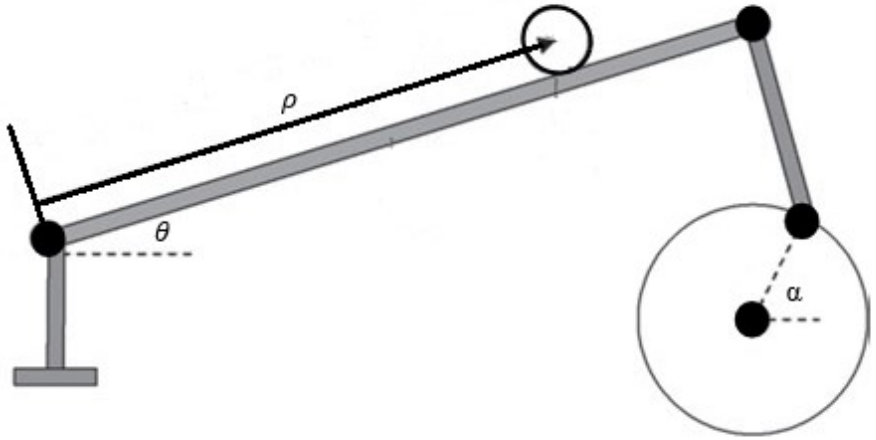

Ilustración 31 Sistema "ball and beam"

Se deja para más adelante la relación del ángulo del servomotor, definido como α con el ángulo de las barras, θ, que como puede observarse en la figura no son el mismo, pero puede ser referido el primero a este último mediante el uso, únicamente, de la geometría del sistema.

5.2 Modelado según la mecánica vectorial

En primera instancia, se hará uso de los desarrollos de la Dínamica Vectorial del Sólido Vinculado para obtener la ecuación diferencial que relaciona la variable de entrada y la de salida. Siguiendo los desarrollos de [25], se debe comenzar aplicando el conocido como Principio de Liberación, que textualmente dice:

"Todo sistema material sometido a vínculos puede ser tratado como libre de vínculos con tal de añadir a las fuerzas activas las llamadas fuerzas vinculares en las direcciones adecuadas a los vínculos que representan".

En consecuencia, se estudian las fuerzas que se ejercen sobre la bola usando para ello la siguiente figura.

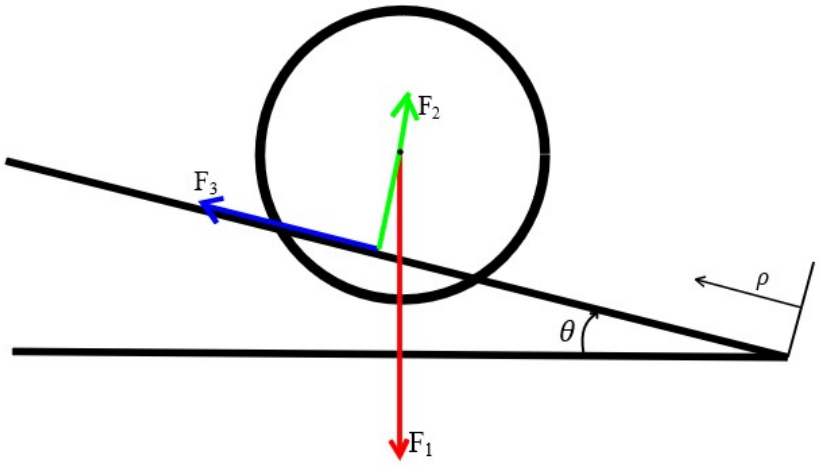

Ilustración 32 Fuerzas aplicadas a la bola

Como se ve en la imagen, a la bola se le están aplicando un total de tres fuerzas, que se describen como:

- F_1 es es peso de la bola: $\overrightarrow{F_1} = m\vec{g}$
- F_2 es la resultante de las reacciones debida al apoyo de la bola sobre las dos barras, y solo actúa en la dirección vertical de los ejes fijos, normalmente se denota como fuerza normal: $\overrightarrow{F_2} = \vec{N}$
- F_3 es la resultante de la fuerza de rozamiento que, en principio, se opone al movimiento lógico de la bola. Es la suma de dos fuerzas de rozamiento iguales que afectan puntualmente en cada uno de los apoyos de la bola con las barras: $\overrightarrow{F_3} = \overrightarrow{F_r}$. Donde, a su vez, se debe cumplir, que para que sea aceptada la premisa de que la bola rueda sin deslizar sobre las barras, $\left|\overrightarrow{F_r}\right| \leq \mu\left|\vec{N}\right|$, donde μ es una magnitud adimensional conocida como coeficiente de rozamiento estático que depende de los dos materiales que están en contacto.

Como se ha indicado, la fuerza de rozamiento no es conocida, simplemente esta acotada, por lo que para calcular esta fuerza F_3, siguiendo [26], se aplica la ecuación del movimiento de rotación que ocurre alrededor de un eje que pasa por el centro de masas de la esfera, y es producido por un torque, τ, que afecta a la bola por el simple hecho de estar rotando y tener, por tanto, una aceleración angular no nula.

Una vez que se tenga la expresión del torque anteriormente mencionado, bastará dividirlo por el brazo correspondiente para obtener la fuerza que se busca, aplicando que *"Momento es igual a fuerza por brazo"*.

$$F_3 = \frac{\tau}{r}$$

Donde se ha definido *r* como el radio de giro de la bola, es decir, la distancia entre el centro de la esfera y el punto de apoyo con las barras, el cual, como es lógico, es menor que el radio geométrico de la bola, *R*, ya que si no fuese así, la bola se caería entre las dos barras.

Por otra parte, del Teorema del Momento Cinético, obtenido como consecuencia de la segunda Ley de Newton, se tiene que *"La derivada del momento ciéntico de una partícula es igual al momento resultante sobre ella"*. Y extrapolando este enunciado al sólido rígido objeto de estudio, es decir, la esfera, se obtiene que:

$$\tau = I * \ddot{\varphi}$$

Donde *I* es la inercia de una esfera maciza de radio *R* y masa *m*, que se calcula mediante la aplicación de geometría de masas, resultando ser:

$$I = \frac{2}{5}mR^2$$

Por otro lado, se debe caer en la cuenta de que la bola rueda sin deslizar sobre las barras, es decir, no patina, por lo que la velocidad relativa entre las barras y la bola en el punto de contacto, es nula.

En consecuencia se cumple que:

$$\dot{\rho} = r * \dot{\varphi}$$

Donde se han definido que φ es el ángulo de giro de la bola sobre un eje que atraviesa su centro de masas y es perpendicular al plano en el que se está moviendo esta longitudinalmente.

Expresión que puede ser derivada con respecto al tiempo, llegando a:

$$\ddot{\rho} = r * \ddot{\varphi} \Rightarrow \ddot{\varphi} = \frac{\ddot{\rho}}{r}$$

Sustituyendo las tres últimas ecuaciones en la que las precede, el resultado es:

$$\tau = \frac{2}{5}m\frac{R^2}{r}\ddot{\rho}$$

Expresión que podemos sustituir en la que se tenía anteriormente de la fuerza F_3 para definitívamente obtener la ecuación que la define:

$$F_3 = \frac{1}{r}\left(\frac{2}{5}m\frac{R^2}{r}\ddot{\rho}\right) \Rightarrow F_3 = \frac{2}{5}m\frac{R^2}{r^2}\ddot{\rho}$$

Una vez se tiene definidas convenientemente todas las fuerzas que dominan el sistema, se está en condiciones de aplicar la Segunda Ley de Newton en cada una de las direcciones.

- Dirección longitudinal a las barras:

$$mg\sin\theta - \frac{2}{5}m\frac{R^2}{r^2}\ddot{\rho} = m\left(\ddot{\rho} - \rho\dot{\theta}^2\right)$$

- Dirección perpendicular a las barras:

$$mg\cos\theta - \left|\vec{N}\right| = 0 \Rightarrow \left|\vec{N}\right| = mg\cos\theta$$

Para el cálculo de la aceleración usada en la primera ecuación, tal y como se expresa en [27], se hace uso de las coordenadas polares, que son en realidad en las que está referido el problema.

$$\vec{x} = \rho\vec{u}_\rho + \theta\vec{u}_\theta$$

Derivando para obtener la velocidad, se obtiene:

$$\vec{v} = \frac{dx}{dt} = \frac{d\rho}{dt}\vec{u}_\rho + \rho\frac{d\vec{u}_\rho}{dt} = \dot{\rho}\vec{u}_\rho + \rho\dot{\vec{u}}_\rho$$

Y por otra parte, se tiene que:

$$\dot{\vec{u}}_\rho = \frac{d\vec{u}_\rho}{d\theta}\frac{d\theta}{dt} = \dot{\theta}\,\vec{u}_\theta$$

Obteniendo la ecuación de la velocidad de una partícula en coordenadas polares.

$$\vec{v} = \dot{\rho}\vec{u}_\rho + \rho\dot{\theta}\,\vec{u}_\theta$$

De nuevo, se deriva la velocidad con respecto del tiempo para obtener la aceleración.

$$\vec{a} = \frac{dv}{dt} = \frac{d\rho}{dt}\vec{u}_\rho + \rho\frac{d\vec{u}_\rho}{dt} + \frac{d(\rho\dot{\theta})}{dt}\vec{u}_\theta + \rho\dot{\theta}\,\frac{d\vec{u}_\theta}{dt} = \dot{\rho}\vec{u}_\rho + \rho\dot{\theta}\,\vec{u}_\theta + \left(\dot{\rho}\dot{\theta} + \rho\ddot{\theta}\right)\vec{u}_\rho + \rho\dot{\theta}\,\dot{\vec{u}}_\theta$$

Y por otra parte, se tiene que:

$$\dot{\vec{u}}_\theta = \frac{d\vec{u}_\theta}{d\theta}\frac{d\theta}{dt} = -\dot{\theta}\,\vec{u}_\rho$$

Obteniendo la ecuación de la aceleración de una partícula en coordenadas polares.

$$\vec{a} = \left(\ddot{\rho} - \rho\dot{\theta}^2\right)\vec{u}_\rho + \left(2\dot{\rho}\dot{\theta} + \rho\ddot{\theta}\right)\vec{u}_\theta$$

Y usando únicamente la parte correspondiente a la dirección de ρ, se llega a lo que se usó en la aplicación de la Ley de Newton.

Tras este paréntesis, y partiendo de la primera ecuación, se dividen ambos miembros por la masa *m* y se reordenan los términos para llegar a lo siguiente:

$$g\sin\theta + \rho\dot{\theta}^2 = \left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\ddot{\rho}$$

Y, finalmente, despejando la aceleración lineal de la bola en función del ángulo de la barra y de parámetros estrictamente geométricos, se alcanza la ecuación diferecial que describe el movimiento del sistema "ball and beam".

$$\ddot{\rho} = \frac{g\sin\theta + \rho\dot{\theta}^2}{\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)}$$

En este punto, con el objetivo de tener únicamente en la ecuación parámetos que han sido tomados como de

partida en el problema, se va a hallar la relación entre el radio de giro *r* y el radio geométrico de la bola *R*, usando la distancia de separación entre barras, tal y como se ve en la siguiente figura.

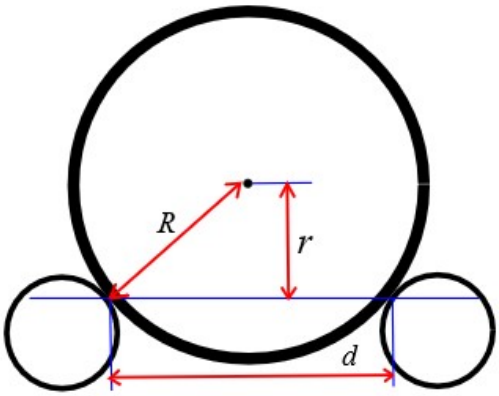

Ilustración 33 Relación entre radio de giro y radio geométrico

En aplicación directa del Teorema de Pitágoras sobre el triángulo rectángulo que se forma, se obtiene que el radio de giro de la bola al cuadrado es igual a la diferencia entre los cuadrados del radio geométrico de la esfera y la mitad de la distancia entre las barras, esto es:

$$r^2 = R^2 - \frac{d^2}{4}$$

Resultado que puede ser sustituido en la ecuación anterior, llegando a:

$$\ddot{\rho} = \frac{g \sin\theta + \rho\dot{\theta}^2}{\left(1 + \frac{2}{5}\frac{R^2}{R^2 - \frac{d^2}{4}}\right)}$$

Para concluir y dejar una expresión lo más elegante posible, se opera algebraicamente obteniendo la relación definitiva entre aceleración lineal de la bola y ángulo de las barras:

$$\boxed{\ddot{\rho} = \frac{g \sin\theta + \rho\dot{\theta}^2}{\left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)}}$$

5.3 Modelado según la mecánica analítica

A continuación, se va a hacer uso de la Mecánica Analítica, que es una disciplina paralela a la Mecánica Vectorial y estudia el movimiento de los sistemas apoyándose en magnitudes escalares en lugar de en vectores. Para ello se definen una serie de cooredenadas generalizadas en función de las cuales se puede describir de forma unívoca el sistema bajo estudio. En el caso del sistema "ball and beam", que tiene un solo grado de libertad, ya que dado un valor de θ, solo hay un posible valor de ρ, se define una coordenada generalizada. A saber:

$$q_1 = \rho$$

Tras esto, es el momento de aplicar la Ecuación de Lagrange para la Mecánica Analítica para la coordenada generalizada, que dice:

"La diferencia entre la derivada con respecto al tiempo de la derivada parcial de la Lagrangiana con respecto a $\dot{q}_k$ y la derivada parcial de la Lagrangiana con respecto a dicha coordenada generalizada, es igual a las fuerzas generalizadas no conservativas en la dirección de esa coordenada"

Lo que es probable que se entienda bastante mejor expresado matemáticamente como sigue.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q_1}} - \frac{\partial \mathcal{L}}{\partial q_1} = Q_1^{NC}$$

Donde, siguiendo la notación de [25], se define:

- $\mathcal{L}$ como la Lagrangiana, que cumple la ecuación: $\mathcal{L} = T - V$, donde T y V son, respectivamente, las energías cinética y potencial del sistema.
- q_l como la coordenada generalizada del problema.
- Q_1^{NC} como la resultante de las fuerzas no conservativas que se aplican al sistema en la dirección de la coordenada q_l, en este caso, no hay ninguna presente que se tenga en cuenta, es decir, $Q_1^{NC} = 0$.

Como se ha indicado, es necesario obtener las energías cinética y potencial del sistema para poder sustituir sus valores en la Ecuación de Lagrange. Se empezará por la energía cinética, que se calcula como la suma de las energías cinéticas de cada uno de los cuerpos que forman parte del sistema, es decir, la bola y las barras.

$$T = T_{bola} + T_{barras}$$

La energía cinética, como es conocido, es la que concierne a la velocidad que tenga un cuerpo, es por eso que la de un sólido rígido tiene dos términos, por un lado, el referido a la velocidad lineal y por otro el que se refiere a la velocidad de rotación del mismo. En el caso de la bola, que es una esfera maciza que se mueve rodando en la dirección de ρ, con velocidad angular $\dot{\varphi}$, según lo expresado en el apartado anterior, se llega a la siguiente expresión.

$$T_{bola} = \frac{1}{2}mv^2 + \frac{1}{2}I\dot{\varphi}^2$$

De esta ecuación el término de la inercia es conocido del análisis del apartado anterior, al igual que la forma de expresar la velocidad de rotación de la bola en función de la derivada de ρ. Además, del cálculo de la aceleración en coordenadas polares, como paso intermedio se llegó a la velocidad, necesaria en este punto.

Si se calcula el cuadrado del módulo de la velocidad como la suma de sus componentes al cuadrado y se sustituye en la ecuación que se tenía de la energía cinética de la bola, junto con los demás resultados anteriores, se llega a:

$$T_{bola} = \frac{1}{2}m\left(\dot{\rho}^2 + \rho^2\dot{\theta}^2\right) + \frac{1}{5}m\frac{R^2}{r^2}\dot{\rho}^2$$

En segundo lugar, se calcula la energía cinética de las barras, que al no tener libertad de movimiento de traslación, reduce su expresión a, únicamente, el término debido a la velocidad de rotación.

$$T_{barras} = \frac{1}{2}I_{barras}\dot{\theta}^2$$

Sumando ambos términos se llega a la expresión de la energía cinética del sistema:

$$T = \frac{1}{2}m\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\dot{\rho}^2 + \frac{1}{2}m\rho^2\dot{\theta}^2 + \frac{1}{2}I_{barras}\dot{\theta}^2$$

Es el momento ahora de calcular la energía potencial del sistema, la cual es bastante más sencilla que la de la cinética. De igual forma, se calcula como la suma de las energías potenciales de cada uno de los dos sólidos, que, teniendo en cuenta que el eje de las equis se ha tomado positivo y hacia la derecha, es:

$$V = -mg\rho\sin\theta - Mg\frac{L}{2}\sin\theta$$

Llegados a este punto, se está en condiciones de calcular la Lagrangiana del sistema, siguiendo su definición expuesta más arriba.

$$\mathcal{L} = \frac{1}{2}m\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\dot{\rho}^2 + \frac{1}{2}m\rho^2\dot{\theta}^2 + \frac{1}{2}I_{barras}\dot{\theta}^2 + mg\rho\sin\theta + Mg\frac{L}{2}\sin\theta$$

Una vez se tiene una expresión analítica de la Lagrangiana del sistema, se pasa aplicar la Ecuación de Lagrange a la única coordenada generalizada del sistema, esto es, a ρ. Para lograr la mayor claridad posible se realizarán todas las operaciones necesarias paso a paso.

$$\frac{\partial \mathcal{L}}{\partial \dot{\rho}} = m\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\dot{\rho}$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\rho}} = m\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\ddot{\rho}$$

$$\frac{\partial \mathcal{L}}{\partial \rho} = m\rho\dot{\theta}^2 + mg\sin\theta$$

Finalmente, se unen tal y como establece la Ecuación de Lagrange los resultados anteriores, además de la definición de fuerzas no conservativas que se hizo al principio de este apartado.

$$m\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)\ddot{\rho} - m\rho\dot{\theta}^2 - mg\sin\theta = 0$$

Que reordenando, puede escribirse como:

$$\ddot{\rho} = \frac{g\sin\theta + \rho\dot{\theta}^2}{\left(1 + \frac{2}{5}\frac{R^2}{r^2}\right)}$$

Lógicamente, a esta ecuación podría aplicarse la relación entre el radio de giro y el geométrico, de la misma forma a como se hizo en el apartado anterior, llegando a:

$$\boxed{\ddot{\rho} = \frac{g\sin\theta + \rho\dot{\theta}^2}{\left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)}}$$

Como puede observarse, la ecuación diferencial es exactamente la misma que se obtuvo tras el análisis mediante la Mecánica Vectorial. Cabe destacar que difiere en parte de las ecuaciones que se encuentran en la bibliografía por haberse usado en este desarrollo coordenadas polares en lugar de cooredenas cartesianas.

5.4 Linealización del modelo

Una vez que se tiene la ecuación diferencial que describe el sistema bajo estudio, se debe cerciorar del hecho de que se trate de una ecuación lineal y en caso de que no lo sea, linealizarla debidamente, tal y como se desprende de [28], ya que, como se comentó en la introducción a este capítulo, la mayoría de los controladores solo son de aplicación ante modelos lineales de los sistemas dinámicos.

Como puede observarse a simple vista, la ecuación obtenida no es una ecuación lineal, ya que apararece el seno del ángulo de inclinación de las barras y el cuadrado de la velocidad del mismo. Para linealizar una ecuación se reordena como una igualdad a cero y se establece un punto de equilibrio en torno al cual se linealizará. Este debe ser tal que las derivadas primeras y segundas de las variables sea cero.

$$f(\rho, \theta, \dot{\theta}, \ddot{\rho}) = \left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)\ddot{\rho} - g\sin\theta - \rho\dot{\theta}^2 = 0$$

$$f(\rho_0, \theta_0, 0, 0) = 0$$

A continuación, se toman variables incrementales con respecto a ese punto de equilibrio y se calculan las derivadas parciales de la función con respecto a cada variable de la que depende, particularizadas para ese punto de equilibrio.

$$\left|\frac{df}{d\ddot{\rho}}\right|_{eq} = \left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)$$

$$\left|\frac{df}{d\rho}\right|_{eq} = -\dot{\theta}_{eq}^2 = 0$$

$$\left|\frac{df}{d\theta}\right|_{eq} = -g\cos\theta_{eq} = -g$$

$$\left|\frac{df}{d\dot{\theta}}\right|_{eq} = -2\rho\dot{\theta}_{eq} = 0$$

Finalmente, se suman los productos de esas derivadas parciales por las variables incrementales y se obtiene la ecuación linealizada:

$$\left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)\ddot{\rho} - g\theta = 0$$

Que reordenándola para dejar una variable en función de la otra queda:

$$\boxed{\ddot{\rho} = \frac{g\theta}{\left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)}}$$

Al mismo resultado se podría haber llegado haciendo uso de dos suposiciones. Por un lado, tomando como cierto que los ángulos de las barras van a ser, generalmente, pequeños. Por lo que en aplicación del desarrollo en Serie de Taylor de las funciones trigonométricas, se tiene que:

$$\sin\theta \sim \theta$$

Y en segundo lugar, aceptar que la velocidad del ángulo de la barra iba a ser pequeña, ya que de no ser así, podrían entrar en juego fuerzas centrípetas que provocarían que al amplificarse mucho el valor de ρ, aumentara consecuentemente de manera muy abrupta el valor de θ, es decir, la velocidad $\dot{\theta}$ se haría muy grande, provocando que la fuerza centrípeta tendiera a expulsar la bola hacia fuera en lugar de recuperar su posición deseada.

$$\dot{\theta} \sim 0$$

Obteniendo, de nuevo, como ecuación lineal que describe el movimiento del sistema "ball and beam" la siguiente:

$$\boxed{\ddot{\rho} = \frac{g\theta}{\left(1 + \frac{8R^2}{20R^2 - 5d^2}\right)}}$$

Podría pensarse que por la aplicación de la primera de las suposiciones se estaría cometiendo un error demasiado grande. Para desmentir esto, se calculará dicho error. Lo único que es conocido es el ángulo máximo y mínimo del servomotor, nombrado al principio del capítulo como α.

El servomotor tiene la posibilidad de moverse como máximo hasta 180º, por lo que al colocarlo de forma que el ángulo 0º corresponda con el ángulo 0º de inclinación de las barras, sus valores límites son:

$$-90° \leq \alpha \leq 90°$$

Por otro lado, se conoce la relación geométrica entre el ángulo del motor y el de las barras, ya que apoyándose en la siguiente figura, tal y como describe [3], las distancia de los arcos es la misma en ambos círculos.

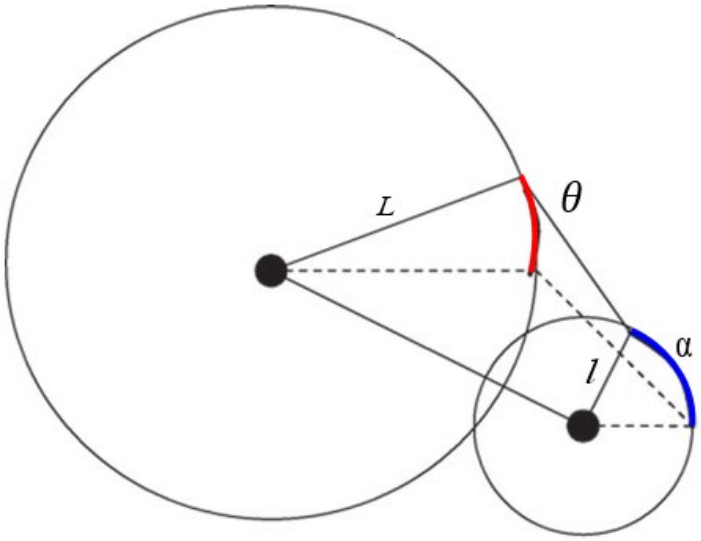

Ilustración 34 Relación entre el ángulo de las barras y el del motor

Como es sabido, los arcos son iguales al producto del radio por el ángulo, por lo que es sencillo extraer la relación simplemente igualando ambos arcos.

$$l \sin \alpha = L \sin \theta$$

Se calcula ahora el máximo ángulo que alcanzará θ, que coincidirá con el máximo que alcance α. De forma análoga se calcula el mínimo.

$$\theta_{máx} = \arcsin\left(\frac{l}{L} sin\, \alpha_{máx}\right) = \arcsin\left(\frac{4.5}{100} * sin\frac{\pi}{2}\right) = \ 0.045016\, rad = 2.5792°$$

$$\theta_{mín} = \arcsin\left(\frac{l}{L} sin\, \alpha_{mín}\right) = \arcsin\left(\frac{4.5}{100} * sin -\frac{\pi}{2}\right) = \ -0.045016\, rad = -2.5792°$$

Si se aplica la aproximación anteriormente expresada en el seno, en el peor de los casos el error cometido será:

$$e_1(\%) = \frac{\theta_{máx} - \sin \theta_{máx}}{\sin \theta_{máx}} 100 = 0.03378\%$$

Que como puede observarse son unos errores más que asumibles para la primera aproximación que se está haciendo del modelo.

5.5 Obtención de la función de transferencia

El último de los pasos del modelado teórico de cualquier sistema dinámico que pretenda controlarse, es la obtención de la función de transferencia que lo describe. Para ello, siguiendo el desarrollo de [24], se aplica la Transformada de Laplace suponiendo condiciones iniciales nulas.

Se recuerda que al transformar, la derivada primera se corresponde con un término de orden uno de la variable de la frecuencia, *s*, y que la derivada segunda se transforma en un término de segundo orden.

Con todo esto, se obtiene lo siguiente:

$$s^2\left(1+\frac{8R^2}{20R^2-5d^2}\right)P(s)=g\Theta(s)$$

Es el momento ahora de reordenar los términos de modo y forma que se tenga en el lado izquierdo de la ecuación el cociente entre la Transformada de Laplace de la variable de salida y la de la entrada, y en el lado derecho de la misma, la expresión que las relaciona, por supuesto, en función de la variable *s*.

$$G(s)=\frac{P(s)}{\theta(s)}=\frac{g}{\left(1+\frac{8R^2}{20R^2-5d^2}\right)s^2}$$

Particularizando para los valores reales iniciales del sistema, que son:

- $g=9.81\frac{m}{s^2}$
- $d=0.015\,m$
- $R=0.012\,m$

Se obtiene, finalmente, la función de transferencia real que describe el sistema ball and beam bajo estudio en este Trabajo.

$$G(s)=\frac{5.9224}{s^2}$$

A partir de ella, en capítulos posteriores se identificará el sistema, se analizarán sus carácterísticas en bucle abierto y bucle cerrado y se intentará diseñar unos controladores que sean capaces de lograr el control del sistema.

6 ANÁLISIS DE ESTABILIDAD

La vida es como montar en bicicleta. Para mantener la estabilidad, hay que seguir pedaleando.

Albert Einstein

UNA vez que se tiene la función de transferencia que, teóricamente, define el sistema "ball and beam", es el momento de pasar a estudiar la estabilidad del mismo, esto es, discernir su estabilidad en bucle abierto y tratar de corregirla en caso de ser negativa, en bucle cerrado. Para ello, siguiendo la teoría clásica del Control Automático y con el apoyo de [29], se van a analizar los polinomios del numerador y del denominador de la función de transferencia.

En Control Automático, es muy importante ver las raíces de ambos polinomios, es decir, los puntos que hacen que estos se anulen. Se definen como ceros de un sistema a las raíces del numerador y como polos de un sistema a las raíces del denominador. En principio, la estabilidad del sistema viene determinada por la posición de sus polos dentro del plano complejo, entendiéndose este como un plano de dos ejes perpendiculares, donde se representa la parte real del polo,$\mathfrak{R}$, en el eje horizontal, y la parte imaginaria,$\mathfrak{T}$, en el eje vertical.

Utilizando como recurso gráfico la siguiente figura de [30], se entiende perfectamente lo anteriormente expresado.

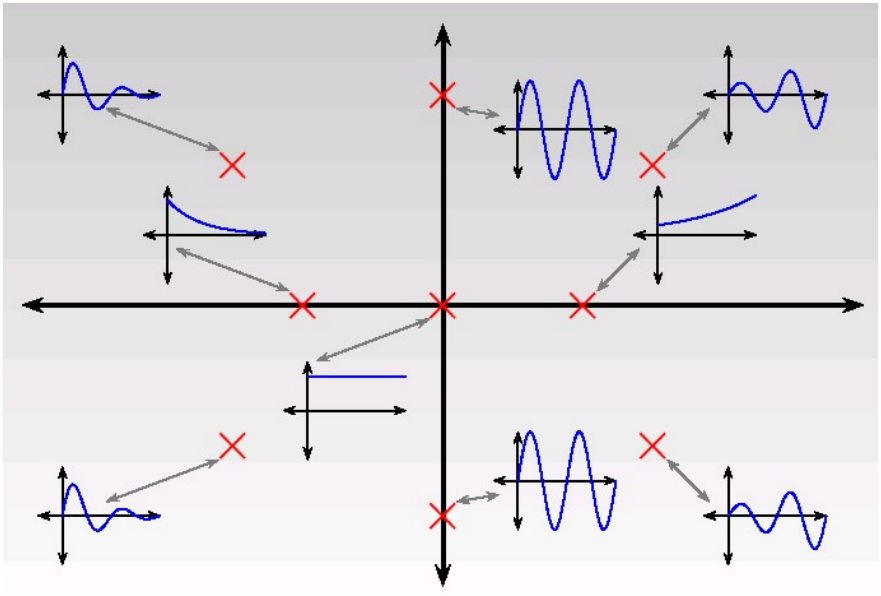

Ilustración 35 Estabilidad de un sistema en función de la posición de los polos

6.1 Estabilidad en bucle abierto

Si se recuerda del capítulo anterior, la función de transferencia obtenida es una ganacia estática en el numerador y el cuadrado de s en el denominador. Por consiguiente, se confirma que no tiene ningún cero y que tiene dos integradores, que es como se conocen a los polos en el origen, es decir, cuando el valor de la variable de la frecuencia que hace nulo el denominador es $s = 0$.

En aplicación del Teorema de Estabilidad, que dice *"Un sistema lineal e invariante en el tiempo descrito por una función de transferencia G(s) es estable si y solo si todos sus polos están situados en el semiplano izquierdo abierto del plano complejo"*, se puede afirmar que si un sistema tiene algún polo con parte real positiva o nula, entonces será inestable. Dado que el sistema bajo estudio tiene dos polos con parte real nula, se confirma que es inestable en bucle abierto.

Para comprobar esto desde un punto de vista de simulaciones, se crea en MATLAB la función de transferencia del sistema, se le aplica en la entrada un escalón unitario y se grafica el resultado, obteniendo:

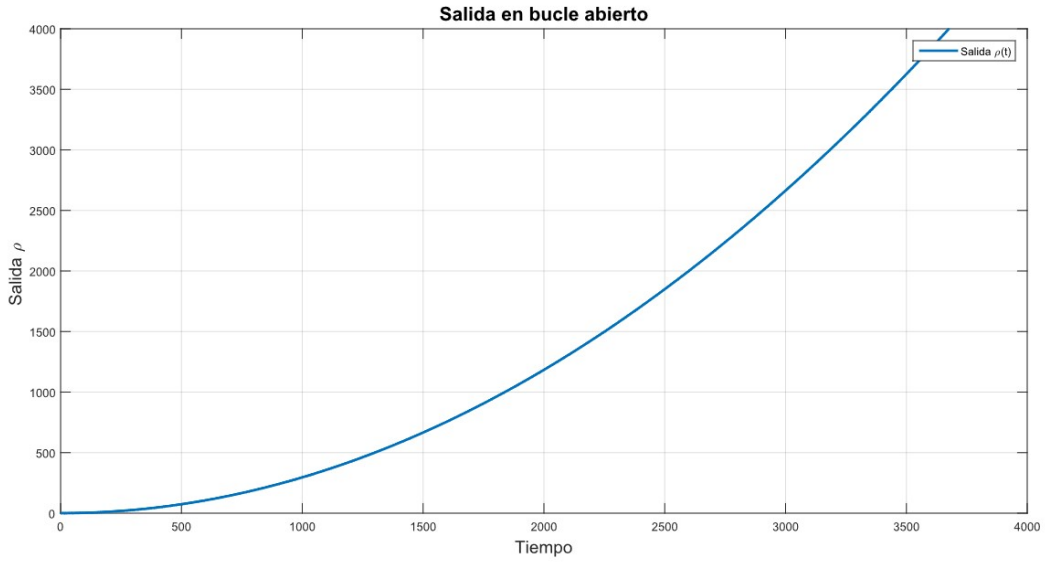

Ilustración 36 Salida en bucle abierto del sistema

Como puede observarse en la gráfica, se confirma lo que desde un punto de vista teórico, ya había sido justificado, es decir, que cuando se le aplica una entrada acotada al sistema, en este caso, un escalón unitario, la salida del sistema no es acotada, sino que crece indefinidamente, por lo que se dice que el sistema es inestable en bucle abierto y cumple:

$$Si\ |u(t)| < K_u\ \ \forall t, entonces\ |y(t)| \to \infty$$

6.2 Estabilidad en bucle cerrado

En consecuencia de la conclusión del apartado anterior, se hace necesario el uso de un controlador para poder estabilizar la posición de la bola en el "set-point" que se desee, usando como entrada el ángulo de inclinación de las barras. Este control se hará en bucle cerrado y utilizando el concepto de realimentación negativa, que implica que la salida del sistema vuelva a la entrada y se calcule la diferencia de esta con la referencia que se quiere alcanzar, dando lugar a un error que será lo que le entre al bloque del controlador y, la salida de este, será la entrada al modelo del sistema, tal y como se ve en la figura.

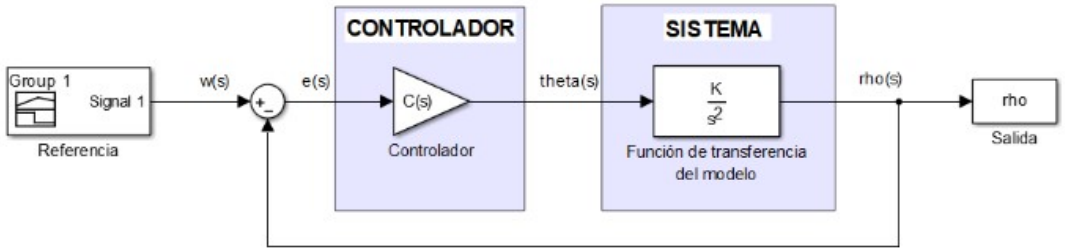

Ilustración 37 Bucle de control con realimentación negativa

Para poder caracterizar de una vez el bucle cerrado con realimentación negativa, será necesaria la obtención de la función de transferencia en bucle cerrado del controlador más el modelo, que se puede sacar directamente del esquema de control.

$$G_{bc} = \frac{\rho(s)}{R(s)} = \frac{G(s)\theta(s)}{E(s) + Y(s)} = \frac{G(s)C(s)E(s)}{E(s) + G(s)\theta(s)} = \frac{G(s)C(s)}{1 + G(s)C(s)}$$

6.3 Control lineal

El primer tipo de control que se va a estudiar como posible es un control PID, es decir, Proporcional Integral y Derivativo. Es conocido así, por ser la acción de control proporcional al error, a la derivada del error y a la integral de este, y matemáticamente se define como:

$$\Delta u(t) = K_p \left(e(t) + T_d \frac{de(t)}{dt} + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau \right)$$

Tal y como se ve en la figura de más arriba, el controlador debe tener una función de transferencia que relacione la entrada del modelo con el error, que se denota como *C(s)*, y es de la forma:

$$C(s) = \frac{\theta(s)}{E(s)} = K_p \left(1 + T_d s + \frac{1}{T_i s} \right)$$

Como se entiende por sentido común, no es obligatorio el uso de cada uno de sus términos, ya que si no es estrictamente necesario, podría usarse solo la parte proporcional (controlador P), o la proporcional y la derivativa, dando lugar a un control PD, o únicamente la parte proporcional y la integral, teniendo entonces un control PI. Por lo que se va a ir introduciendo cada uno de los controladores de forma escalonada para poder ir analizando la respuesta que se observa.

Para estudiar la respuesta del sistema en bucle cerrado ante entradas acotadas representadas como escalones de distintas amplitudes, se va a hacer un estudio en tiempo continuo mediante la herramienta Simulink de MATLAB usando los bloques de función de transferencia y de controlador PID y se van a representar, en cada uno de los casos, los resultados obtenidos.

6.3.1 Control proporcional: P

El control proporcional tiene como función de transferencia únicamente una constante K_p, que irá multiplicando al error.

$$C(s) = K_p$$

Para verlo de una forma más gráfica se muestra el esquema de Simulink en el que se ve el diagrama de bloques para este controlador.

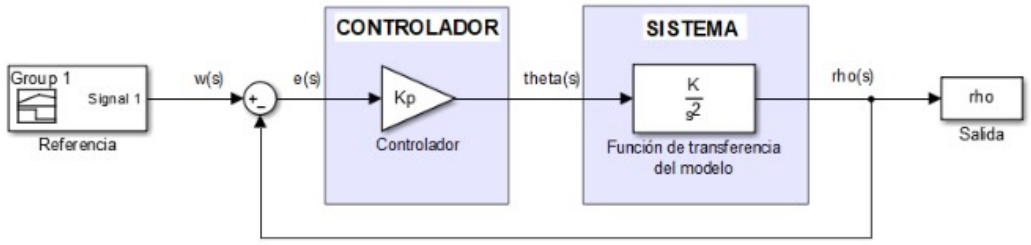

Ilustración 38 Bucle de control con controlador P

Por lo que su función de transferencia en bucle cerrado será la siguiente:

$$G_{bc}(s) = \frac{G(s)C(s)}{1 + G(s)C(s)} = \frac{\frac{K_p K}{s^2}}{1 + \frac{K_p K}{s^2}} = \frac{K_p K}{s^2 + K_p K}$$

Como puede observarse, se reconocen dos polos, por ser un polinomio de orden dos el denominador, su valor se obtiene igualándolo a cero:

$$s_{1,2} = \pm\sqrt{-K_p K}$$

Se sabe del capítulo de modelado teórico que $K > 0$, y como la constante proporcional K_p, también se elige positiva, los polos de la función de transferencia en bucle cerrado serán dos polos complejos puros conjugados, es decir, con parte real nula, por lo que estarán sobre el eje imaginario vertical del plano complejo. Basándose en la Ilustración 35, puede verse como es el caso en el que el sistema es críticamente estable, es decir, que para entradas de tipo escalón, no produce salidas infinitas pero tampoco se estabiliza en torno a un punto, sino que se mantiene oscilante eternamente con una amplitud constante, por lo que no es una solución adecuada.

Utilizando una ganancia para el controlador moderadamente pequeña, de $K_p = 0.1$, para evitar que tenga demasiadas oscilaciones y se empeore la visibilidad de la gráfica y un tiempo de simulación de 300, se obtiene la siguiente gráfica de salida y referencia en función del tiempo, en simulación continua con Simulink.

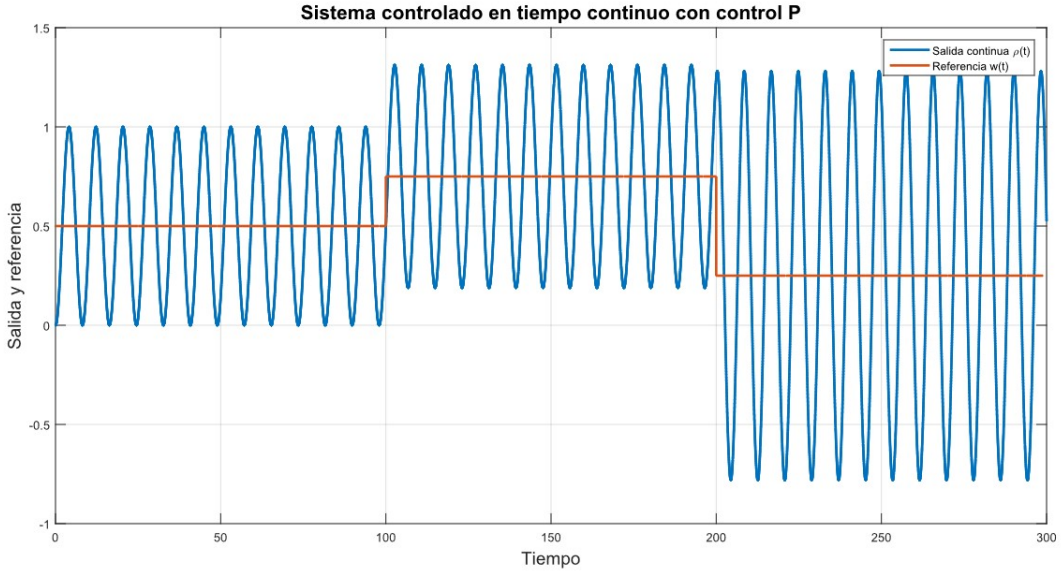

Ilustración 39 Salida en bucle cerrado con control P en simulación continua

Como se observa, viene a confirmar lo que se esperaba, que con el controlador P no se puede alcanzar un control adecuado.

6.3.2 Control proporcional-derivativo: PD

La conclusión del subapartado anterior nos lleva a tener que introducir otro de los términos en la función de transferencia del controlador. Se tienen dos opciones posibles, el derivativo o el integral. El primero de ellos tiene como carácterística principal mejorar el régimen transitorio y el segundo, eliminar el error en régimen permanente. Debido a que, en realidad no se alcanza régimen permanente en el apartado anterior, es más lógico aplicar un control PD para lograr estabilizar el transitorio con la introducción del mismo.

La función de transferencia del controlador es la del producto de una ganacia K_p y un cero en T_d, tal y como aparece en la ecuación:

$$C(s) = K_p(1 + T_d s)$$

Con esto, el bucle de control del sistema bajo estudio queda como aparece en la figura siguiente, donde se han separado las dos partes del controlador para mayor claridad visual.

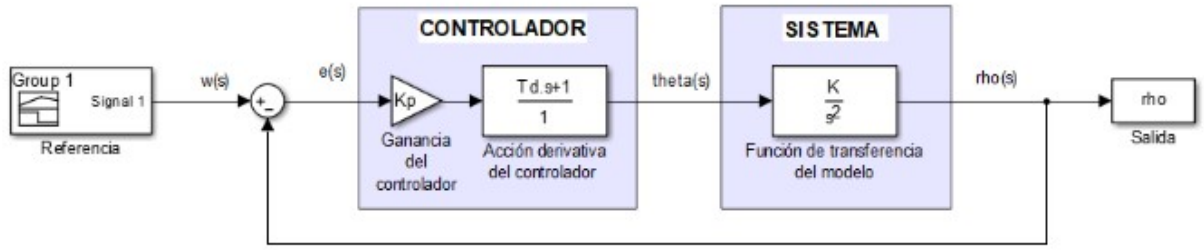

Ilustración 40 Bucle de control con controlador PD

Por lo que la función de transferencia del sistema en bucle cerrado queda de la siguiente forma:

$$G_{bc}(s) = \frac{G(s)C(s)}{1 + G(s)C(s)} = \frac{\frac{K_pK}{s^2}(1 + T_d s)}{1 + \frac{K_pK}{s^2}(1 + T_d s)} = \frac{K_pKT_d s + K_pK}{s^2 + K_pKT_d s + K_pK}$$

Tal y como se ve, se obtiene un sistema de segundo orden con un cero, ya que el numerador es de orden uno. La posición del cero queda determinada por la raiz del numerador:

$$K_pKT_d s + K_pK = 0 \Rightarrow s = \frac{-K_pK}{K_pKT_d} = \frac{-1}{T_d}$$

Al ser T_d una constante positiva, el cero estará colocado siempre en el semiplano izquierdo del plano complejo, por lo que se denominará cero de fase mínima y no dará problemas de estabilidad ni picos iniciales.

Por otro lado, las raíces del denominador descrubrirán los polos del sistema, que serán concluyentes para poder saber si el sistema, finalmente, se estabiliza.

$$s^2 + K_pKT_d s + K_pK = 0 \Rightarrow s_{1,2} = \frac{-K_pKT_d \pm \sqrt{(K_pKT_d)^2 - 4K_pK}}{2}$$

Como se ve, en función del valor que se le dé a los parámetros, el resultado puede ser distinto. En realidad, uno de los valores ya está fijado por el modelo del sistema: $K = 5.9224$, por lo que se tiene libertad de diseño para los otros dos. Si se toma el mismo valor que en el subapartado anterior, de $K_p = 0.1$ y un tiempo derivativo $T_d = 1$, entonces los polos estarán colocados en:

$$s_{1,2} = -0.2961 \pm 0.7103\, j$$

Es decir, se tendrían dos polos complejos conjugados con parte real negativa, por lo que serían completamente estables y el sistema sería controlable con este controlador. Por supuesto, lo que aquí se ha planteado es una situación ideal en tiempo continuo, que no es realizable por ningún microcontrolador, que como es bien sabido, trabajan en tiempo discreto como todas las computadoras reales.

Sin embargo resulta interesante y por eso se redacta en este Trabajo, plantear los controladores en continuo, pues si ni siquiera en tiempo continuo funciona el controlador, casi imposible será que este funcione en tiempo discreto, cuando los errores de discretización y otros factores entran en juego.

Para comprobar los resultados teóricos desde un punto de vista práctico, se simula en la herramienta Simulink de MATLAB el bucle de control de la figura de más arriba con los parámetros expresados en los párrafos anteriores y se grafica el resultado de la salida frente al conjunto de escalones de distinta amplitud dados como referencia.

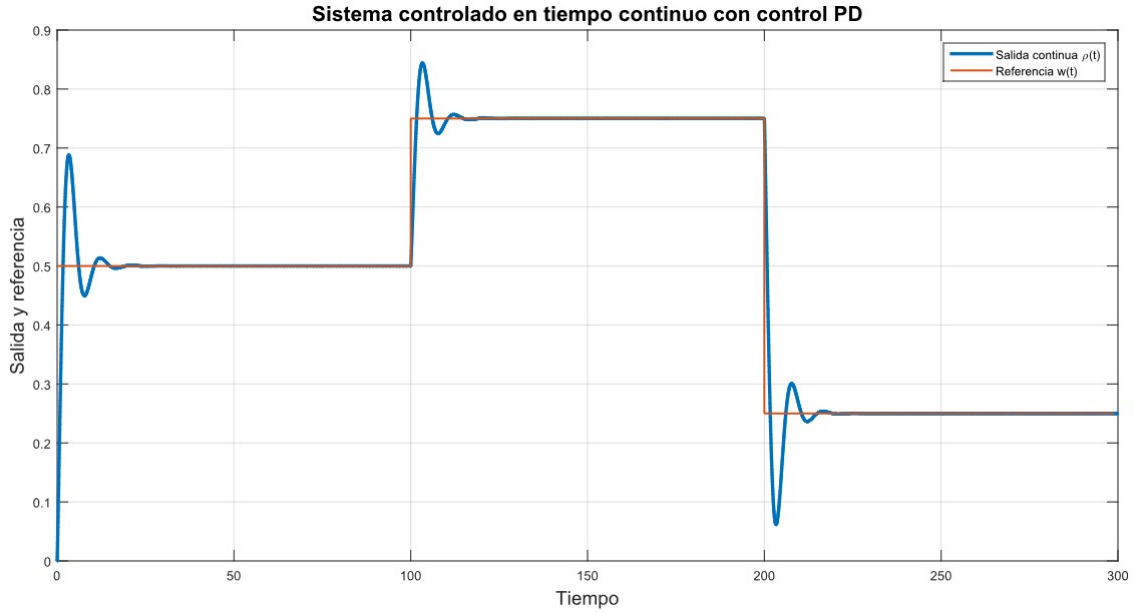

Ilustración 41 Salida en bucle cerrado con control PD en simulación continua con $T_d = 1$

Efectivamente, se logra un buen control del sistema, pero en el que se observan una primera sobreoscilación importante y leves oscilaciones posteriores en los cambios de referencia, pero no se producen oscilaciones mantenidas ni el indeseable efecto de "kick back" o vuelta atrás, que consiste en que la salida baje antes de alcanzar la referencia, con la pérdida de energía que ello conlleva. Para intentar disminuir las sobreoscilaciones y mejorar ese transitorio inicial, se hace una segunda prueba con $T_d = 2$, obteniendo la siguiente gráfica.

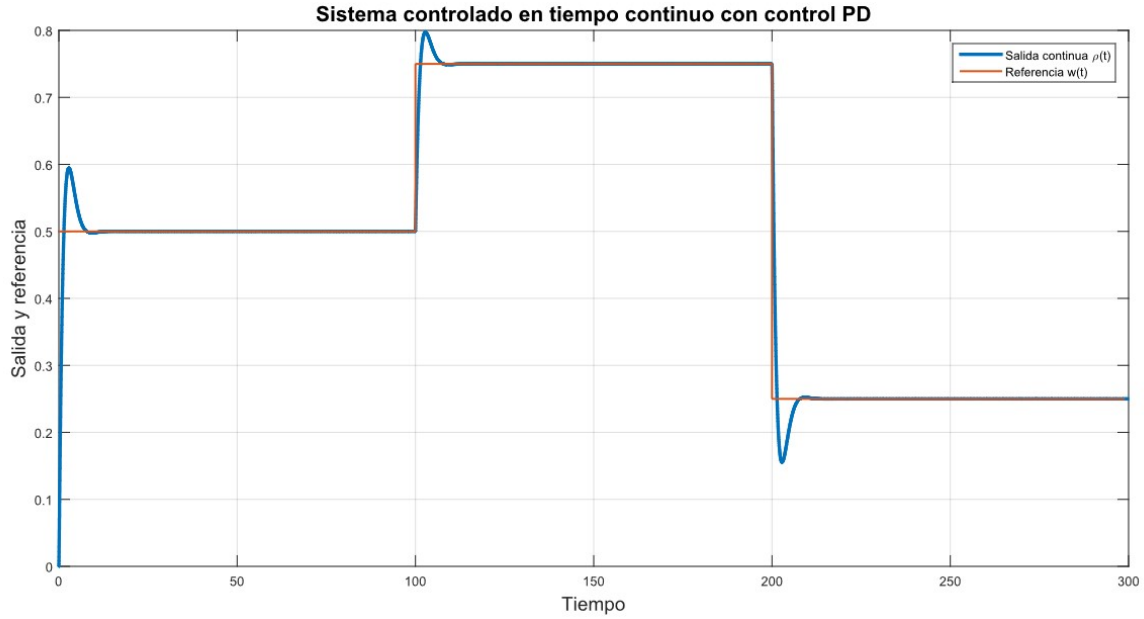

Ilustración 42 Salida en bucle cerrado con control PD en simulación continua con $T_d = 2$

En este caso, la posición de los polos habría cambiado hasta colocarse en:

$$s_{1,2} = 0.5922 \pm 0.6950j$$

Por lo que se mantiene la casuística de ser dos polos complejos conjugados con parte real negativa, y por tanto, estables, aunque cambien, como es lógico, los valores concretos de las amplitudes tanto reales como imaginarias.

Por otro lado, puede verse como el error en régimen permanente es nulo en cualquiera de los dos casos, por lo que se hace del todo innecesaria la introdución de un término integral para reducirlo. En conclusión, si se usa un controlador de tipo lineal para controlar el sistema "ball and beam", el más óptimo para ello es un Proporcional Derivativo, es decir, un PD.

7 SIMULACIONES EN TIEMPO DISCRETO

Quien controla el presente, controla el pasado y quien controla el pasado, controlará el futuro.

George Orwell

COMO se adelantó en el capítulo anterior, los resultados obtenidos en tiempo continuo no se pueden dar por definitivos, ya que al pretendese implementar un sistema real que funciona en tiempo discreto, debe ser este el dominio que se use para el diseño de los controladores. En primer lugar, para lograr comprobar la idoneidad del control PD descrito anteriormente, se implementa en lenguaje de Scripts de MATLAB un simulador para controlar cualquier sistema dado por su función de transferencia en tiempo continuo. Tras diseñar los parámetros del controlador que den una mejor respuesta ante referencias dadas como entradas, se presentarán de forma comparada ambas respuestas, la continua y la discreta.

Por otra parte, también es objetivo de este capítulo la presentación del Control Predictivo como uno de los tipos de control más importantes en la actualidad por su uso en investigación y en la industria. Además de un breve desarrollo teórico que sustente las aplicaciones prácticas, se presentará un simulador en lenguaje de Scripts de MATLAB, que al igual que el ya nombrado para controladores lineales, permite obtener la respuesta de un sistema usando un control de tipo GPC. Para finalizar, se estudiarán los resultados obtenidos y se comparán con los del controlador lineal.

7.1 Discretización de un controlador PD

En primer lugar, se debe aclarar que cuando se trabaja con sistemas que van a ser controlados con microcontroladores, como por ejemplo Arduino, como es este caso, y con sensores y actuadores, el cálculo de la acción de control se realiza en ciertos instantes concretos de tiempo que son múltiplos de un tiempo de muestreo prefijado, por lo que se conocen como sistemas muestreados. En efecto, el sistema de control recibe una serie finita de valores de la entrada y de la salida del sistema, y ofrece una serie, también finita, de acciones de control.

Como trabajar con secuencias de números no es algo cómodo, tal y como se expresa en [31], se utiliza la conocida como Transformada Z, para pasar del dominio del tiempo continuo al del tiempo discreto. Si se parte de una señal continua denominada *x(t)*, se puede muestrear con un tiempo de nuestreo *T*, para obtener una secuencia del tipo:

$$\{x_k\} = x(0), x(T), x(2T), \dots, x(kT)$$

Lo cual puede ser reescrito usando sumatorios y la función Delta de Dirac hasta quedar de la forma:

$$x^*(t) = \sum_{k=0}^{\infty} x(kT)\delta(t - kT)$$

A esta función se le puede implementar la Transformada de Laplace para pasarla al dominio de la frecuencia en tiempo continuo y, tras esto, utizar el cambio de variables $z = e^{Ts}$ para obtener definitivamente su Transformada Z y tenerla ya en tiempo discreto:

$$\mathcal{Z}\{x_k\} = \sum_{k=0}^{\infty} x_k z^{-k}$$

Sin embargo, este proceso es bastante arduo de hacer a mano, por lo que, aunque este sea su fundamento teórico, en la práctica, se utiliza la función de MATLAB `'c2d'` para poder pasar un sistema desde el dominio continuo al discreto. Para ello, primero se debe definir la función de transferencia en continuo con los parámetros del modelo y fijar un tiempo de muestreo, que por simplicidad y sin pérdida de generalidad se tomará aquí como $T_s = 1$, tal y como se hace en el código siguiente:

```
clear all; close all; clc;

%Función de transferencia en tiempo continuo.
g = 9.81;
R = 0.012;
d = 0.015;
K = g/(1+(8*R^2/(20*R^2-5*d^2)));
G = tf( K, [1 0 0]);

%Paso de continuo a discreto usando el método de discretización que incluye
%un mantenedor de orden cero (zero-order hold).
Ts = 1;
Gd = c2d(G,Ts,'zoh');
```

Al ejecutarlo, se obtiene la función de transferencia del sistema en tiempo discreto que es:

$$G(z) = \frac{2.961z + 2.961}{z^2 - 2z + 1}$$

Como puede observarse, la función de transferencia en discreto viene dada en potencias positivas, es decir, referida al futuro, por lo que, en principio, habría que pasarlo a potencias negativas multiplicando tanto el numerador como el denominador por z^{-2}. Sin embargo, realmente no hace falta hacer esto, porque solo son relevantes los coeficientes del numerador y del denominador de la función de transferencia, por lo que se puede trabajar con ellos como si fueran vectores. Para ello, se usa el hecho de que una función de transferencia en MATLAB, es una estructura con dos celdas: numerador y denominador, y usando el comando `'cell2mat'`, se pueden extraer dichas componentes.

```
%Vectores con las componentes de los polinomios.
denom = cell2mat(Gd.den);
numer = cell2mat(Gd.num);
numer = numer(find(numer,1,'first'):end);

%Se obtienen sus dimensiones.
dim_denom = length (denom);
dim_numer = length (numer);
dim_numer = length(numer)-(find(numer,1,'first')-1);
```

Cabe destacar en este punto que ha sido necesario redefinir el numerador y su dimensión para que no tome como componente dentro del vector, al cero que iría multiplicando al término de orden dos, es decir, a z^2.
Una vez que se tiene correctamente definido el sistema por las componentes de sus polinomios de la función de transferencia en tiempo discreto, se va a diseñar los parámetros del controlador. En primer lugar, se escoge una K_p que haga que se tengan unas sobreoscilaciones moderadas con un T_d cualquiera, en este caso, se escoge

una constante proporcional diez veces menor a la que se eligió en el capítulo anterior, para lograr que el sistema no se inestabilice. Para elegir el tiempo derivativo, se va a forzar que los polos en bucle cerrado sean reales negativos puros, lo que se consigue anulando el radicando de la ecuación genérica que se obtenía entonces.

$$\left(K_p K T_d\right)^2 - 4K_p K = 0 \Rightarrow K_p K T_d^2 = 4 \Rightarrow T_d = \sqrt{\frac{4}{K K_p}} = \sqrt{\frac{4}{5.9224 * 0.01}} = 8.2183$$

En este punto se tendría todo lo necesario para implementar el controlador al sistema y obtener la acción de control. Sin embargo, se cae en la cuenta de que la ecuación que define la acción de control para un controlador PID, lleva consigo el cálculo de la derivada y la integral del error, entendiéndose este como una función continua. Al ser el error ahora una secuencia discreta, se debe buscar una forma de aproximar estas operaciones.

En el caso de la integral al calcularse como una aproximación del área que se encierra bajo la curva, existen tres métodos posibles: Euler hacia delante, Euler hacia detrás y Tustin. De todos, el que normalmente se utiliza es este último, no solo por dar mejores aproximaciones, sino porque garantiza que el controlador no se inestabiliza al discretizarse.

Sin embargo, debido a que el control que se va a implementar es un PD, solo será necesario aproximar la derivada, para lo que se toman incrementos del valor del error actual y el anterior, atendiendo a la forma:

$$\frac{de(t)}{dt} \approx \frac{e_k - e_{k-1}}{T_s}$$

Con lo que la expresión del PD en tiempo discreto queda:

$$u_k = K_p \left(e_k + \frac{T_d}{T_s}(e_k - e_{k-1}) \right)$$

Retrasado esta expresión en un tiempo de muestreo y restándola a la anterior se obtiene la siguiente expresión en la que se basa la acción de control actual en la pasada y en los errores presentes y pasados.

$$u_k = u_{k-1} + q_0 e_k + q_1 e_{k-1} + q_2 e_{k-2}$$

Donde se han definido tres constantes que solo dependen de los parámetros del controlador, por tanto, no deben ser calculadas en cada tiempo de muestreo:

$$q_0 = K_p \left(1 + \frac{T_d}{T_s}\right)$$

$$q_1 = K_p \left(-1 - 2\frac{T_d}{T_s}\right)$$

$$q_2 = K_p \left(\frac{T_d}{T_s}\right)$$

Lo que queda expresado en el programa que se está haciendo como sigue:

```
%Parámetros del controlador.
Kp = 0.01;
Td = 8.2183;

%Constantes para la discretización.
q0 = Kp*(1 + Td/Ts);
q1 = Kp*(-1 - 2*(Td/Ts));
q2 = Kp*(Td/Ts);
```

Una vez que se tiene todo esto, puede pasarse a la simulación del sistema, lo cual se realiza en un bucle `'for'` en el que en cada tiempo de muestreo desde uno inicial hasta un tiempo de simulación prefijado se calcula la salida del sistema, el error frente a la referencia y la acción de control. Para el cálculo de la salida se utiliza la función de transferencia en discreto de la siguiente forma:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{den(z)}{num(z)}$$

$$1 * y_k + den(2) * y_{k-1} + den(3) * y_{k-2} + \cdots + den(dimden) * y_{k-dimden-} \\ = num(1) * u_{k-1} + num(2) * u_{k-2} + \cdots + num(dimnum) * u_{k-dimnum}$$

A la hora de la implementación, se cae en la cuenta de que los primeros valores del error, la salida y la acción de control deben estar inicializados para que el simulador pueda acceder a ellos y calcular los posteriores basándose en estos primeros. Es por esto que la simulación se traslada en el tiempo `'ini'` tiempos de muestreo.

A continuación, se muestra todo el código que implementa lo anteriormente expresado y que seguiría secuencialmente a los otros tres trozos que ya se han presentado en este capítulo.

```
%Tiempo de simulación y valor de inicio de la misma.
T_simulacion = 300;
ini = max(dim_denom, dim_numer);

%Referencia.
r=[zeros(1,ini),0.5*ones(1,T_simulacion/3),0.75*ones(1,T_simulacion/3), ...
    0.25*ones(1,(T_simulacion/3))]';

%Inicialización de vectores para la simulación.
t=(0:Ts:T_simulacion-1)';
y=zeros(1,ini)';
e=zeros(1,ini)';
u=zeros(1,ini)';

%Bucle de simulación.
for k=ini+1:Ts:T_simulacion+ini
    y(k) = -denom(2:dim_denom)*y((k-1):-1:(k-(dim_denom-1))) ...
           + numer(1:dim_numer)*u((k-1):-1:(k-1-(dim_numer-1)));
    e(k) = r(k) - y(k);
    u(k) = u(k-1) + q0*e(k) + q1*e(k-1) +  q2*e(k-2);
end
```

Tras la ejecución del programa, es el momento de la representación gráfica de los resultados obtenidos. Para ello, se recortan los vectores para que no se representen las primeras componentes que se habían inicializado a cero y solo servían para poder ser usadas en el simulador.

En primer lugar, se muestra la salida del sistema controlado frente a la referencia, para comprobar como se intenta ajustar la primera a la segunda. De igual forma, en una segunda gráfica paralela a esta se presenta la acción de control, que se observa nula cuando la salida y la referencia se igualan, pero que aumenta de forma brusca, ya sea positiva o negativamente, cuando la amplitud del escalón que representa la referencia, se ve modificada, debido al consecuente aumento del error.

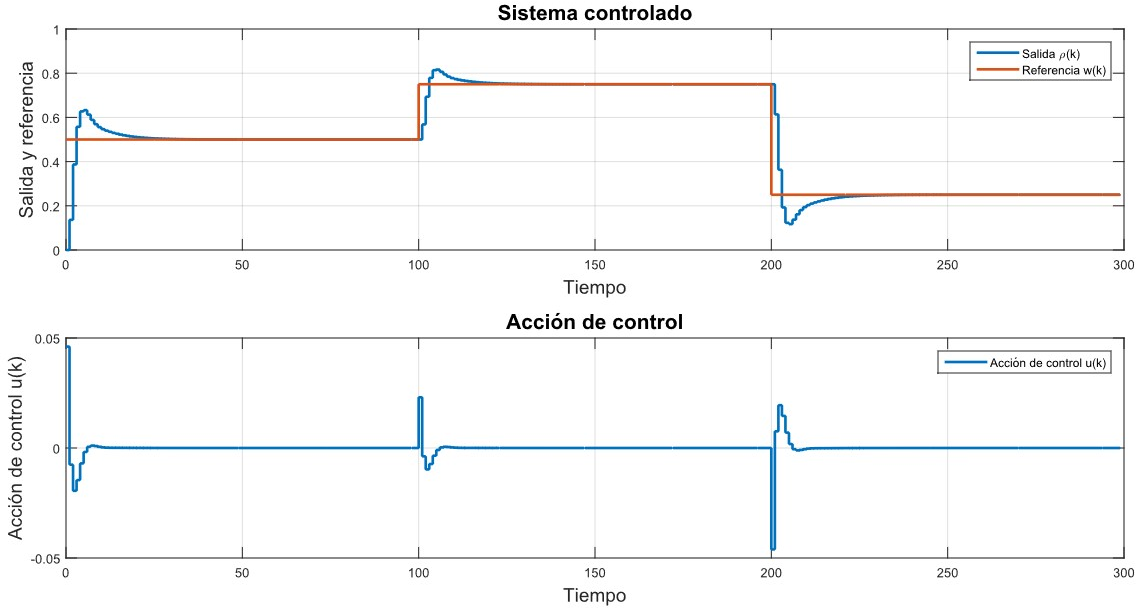

Ilustración 43 Control PD discretizado: salida y acción de control

Se observa un control correcto, en el que hay una primera sobreoscilación debida al cambio de referencia, pero que gracias a haberse forzado que los polos del sistema en bucle cerrado sean reales puros, no presenta oscilaciones posteriores a esta primera. Además, el error en régimen permante se vislumbra nulo, por lo que se confirma de de nuevo que no es necesario incluir un término integral al control.

Por otra parte, se va a representar de nuevo la salida en discreto con la referencia, pero, a su vez, la salida en continuo que se obtuvo en el capítulo anterior con la simulación en Simulink.

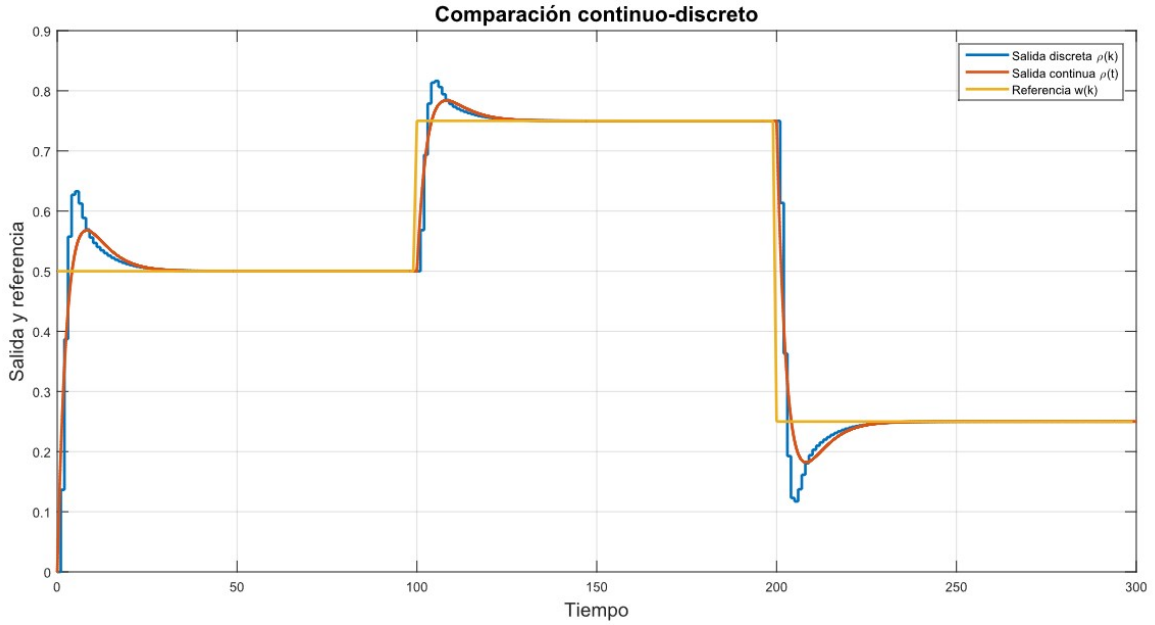

Ilustración 44 Comparación del control PD en tiempo continuo y discreto

Efectivamente, se comprueba como en esencia la respuesta es la misma en ambos dominios, con una sola sobreoscilación inicial y error en régimen permanente nulo. Sin embargo, es reseñable como dicha sobreoscilación es de mayor amplitud en la salida obtenida con la simulación discreta, debido al error de discretización por la aproximación de la derivada del error.

A pesar de esto, puede tomarse como válida la aproximación, ya que el control es completamente aceptable y, lo más importante, implementable en un microcontrolador real para poder, finalmente, controlar el sistema "ball and beam", objetivo final de este Trabajo.

7.2 Control Predictivo basado en Modelo

Tras haber comprobado el correcto funcionamiento de los controladores lineales, se plantea la posibilidad de encontrar un mejor control que sea capaz de dar una respuesta más rápida y, en definitiva, más cercana a la referencia en el menor tiempo posible para el sistema bajo estudio. En esta búsqueda, se elige uno de los controles más importantes en la actualidad desde el punto de vista de la investigación, ya que anulamente son numerosos los artículos científicos, tesis doctorales y trabajos académicos en general que se publican sobre este tema. Se está hablando, efectivamente, del Control Predictivo basado en Modelo.

7.2.1 Generalidades del MPC

Siguiendo como referencia los desarrollos de [32], el "Model Predictive Control", conocido por sus siglas en inglés como MPC, es un conjunto de técnicas que se organizan en torno a una serie de ideas comunes que resultan en estructuras de control similares, por lo que no puede esperarse una ecuación única y genérica que solucione todos los problemas, como ocurre en el caso del PID. Lo que sí es común, como se ha comentado, son las tres ideas en las que se basa:

- Uso de un modelo, que debe haber sido obtenido con anterioridad y de cuya exactitud depende en gran medida la bondad del control final, el cual consiste en la predicción de las variables controladas (salidas y estados) del proceso a lo largo de un intervalo de tiempos de muestreo futuros que se conoce como horizonte de predicción.
- Resolución de la minimización de una función de coste, la cual puede ser de diversas formas y es lo que suele distinguir a un algoritmo de otro. Es típico el uso de una función cuadrática.
- Aplicación de una estrategia de horizonte deslizante, que consiste en que en cada instante de tiempo, el horizonte se desplaza un tiempo de muestreo hacia el futuro. Así, se repite la optimización de todo el problema en cada tiempo de muestreo, pero solo se utiliza la primera componente del vector de acciones de control obtenido, ya que en el siguiente tiempo de muestreo, se conocerá el valor de la salida en el instante actual y podrá usarse para que el control sea más exacto.

Siguiendo estas premisas, de forma esquemática, podría verse la estructura básica del MPC siguiendo la siguiente figura.

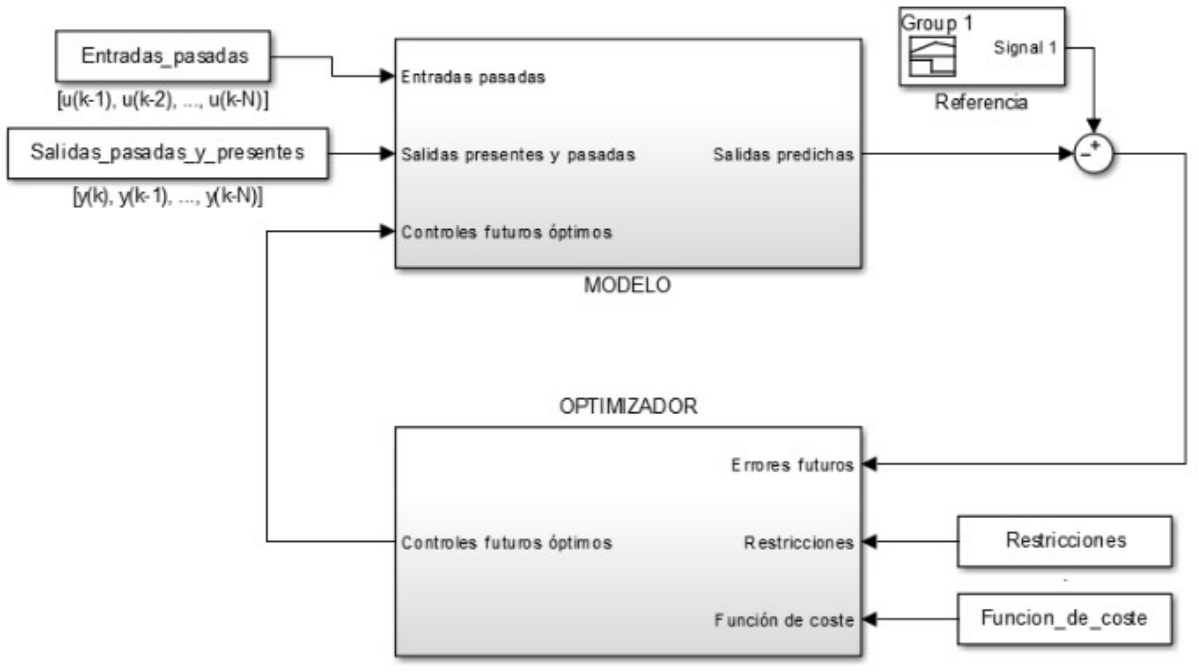

Ilustración 45 Esquema de la estructura del MPC

Del esquema anterior, deben elegirse dos aspectos, que son de gran importancia, de entre distintas posibilidades que existen hoy en día. Por un lado, el modelo de predicción debe escogerse en función de la información de la que se disponga. En este caso, dado que se tiene la función de transferencia en discreto del sistema, se utilizará este modelo. Como es sabido, los polinomios de la función de transferencia, relacionan la salida y la entrada del sistema de la forma:

$$A(z^{-1})y(t) = B(z^{-1})u(t)$$

Por lo que las predicciones de la salida en el instante *t+k* referido al instante *t*, puede expresarse como:

$$\hat{y}(t+k|t) = \frac{B(z^{-1})}{A(z^{-1})}u(t+k|t)$$

El segundo tema que se debe tratar es la función de coste, que como se ha referido anteriormente, es donde reside la mayor variedad de algoritmos que resuelven el MPC. Para lograr obtener el mínimo de esta como una función explícita de la referencia y de las entradas y salidas pasadas, se va a optar por tomar la función de coste como una función cuadrática, que en la práctica será una simplificación de la genérica, que es:

$$J(N_1, N_2, N_u) = \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2$$

Donde se han definido:

- N_1 y N_2 como el inicio y el final del horizonte de predicción, respectivamente.
- N_u como el horizonte de control.
- $\delta(j)$ y $\lambda(j)$ como secuencias de ponderación para dar más o menos peso al error entre la salida predicha y la referencia o al esfuerzo de control, respectivamente.

7.2.2 Un algoritmo: el GPC

Una vez que se tiene una idea general sobre lo que consiste el Control Predictivo basado en Modelo, es el momento de buscar un algoritmo que sea implementable a un sistema real. Es inmensa la cantidad de algoritmos que se encuentran en la bibliografía y mayor aún la que no se encuentra, por ser software privado, protegido por grandes compañías y cuyas licencias solo están al alcance de industrias de gran tamaño.

Teniendo en cuenta el carácter inestable del sistema "ball and beam" en bucle abierto, y por tanto la imposibilidad de obtener una secuencia finita con las salidas ante una entrada en escalón que se acabe estabilizando, se descarta la opción del conocido como "Dynamic Matrix Control" (DMC) y se decide que la mejor opción de las disponibles es el Control Predictivo Generalizado normalmente nombrado por sus siglas en inglés, GPC.

Este algoritmo usa el modelo de función de transferencia tipo CARIMA (Controlled AutoRegresive Integrated Moving Average), que sigue la forma:

$$A(z^{-1})y(t) = B(z^{-1})z^{-d}u(t-1) + C(z^{-1})\frac{e(t)}{1-z^{-1}}$$

Que como puede observarse es idéntico al modelo de función de transferencia, pero teniendo en cuenta el posible retraso del sistema, así como las perturbaciones del mismo. Si se considera que las perturbaciones del sistema se pueden aproximar por un ruido blanco, entonces se cumple que:

$$C(z^{-1}) = 1$$

Por lo que el modelo pasa a ser:

$$A(z^{-1})y(t) = B(z^{-1})z^{-d}u(t-1) + \frac{e(t)}{\Delta}, con\ \Delta = 1 - z^{-1}$$

En cuanto a la función de coste, la que aquí se usa es tal cual la que se describió en el subapartado anterior, es decir, una función cuadrática, pero tomando $\delta(t) = 1$ y $\lambda(t) = \lambda = cte$.

Para la obtención de la predicción óptima de $y(t+j)$ se aplica un desarrollo que parte de esta ecuación del modelo y que, desde un punto de vista teórico puede encontrarse en [32]. En lo que resta de capítulo se va a explicar como se ha pasado de dicho desarollo a un simulador en lenguaje de Scripts de MATLAB, que se apoya en [33] y permite obtener un control GPC del sistema "ball and beam".

En primer lugar, se debe tener la función de transferencia en continuo, pasarla a discreto y obtener los componentes de los polinomios que definen el numerador $\left(B(z^{-1})\right)$ y el denominador $\left(A(z^{-1})\right)$ de esta, tal y como se hizo en el apartado anterior de controladores lineales, por lo que ese paso no se repite aquí por repetitivo. Una de las ventajas que ofrece este algoritmo es que permite tener en cuenta un posible retraso, que de forma implícita tenga el sistema. Este no se conoce a priori, hasta que no se realicen pruebas del control del sistema físico, y puede que sea nulo, pero para lograr mayor generalidad en esta simulación, se supondrá un retraso de un tiempo de muestreo: $d = 1$.

Tras esto, se definen los horizontes de predición y de control y el parámetro de ponderación λ. Por simplicidad, se tomarán los horizontes por defecto y no se ponderará con λ, es decir, $N_1 = d + 1 = 2$; $N_u = N = N_2 - N_1 + 1$; $\lambda = 1$. Como horizonte de control se toma $N_2 = d + 3 = 4$. Puede verse a continuación:

```
%Retraso del sistema.
d = 1;

%Parámetros del control: horizontes y ponderación.
N1 = d+1;
N2 = d+3;
N = N2-N1+1;
Nu = N;
lambda = 1;

%Grados de los polinomios A y B que definen el sistema.
na = length(A)-1;
nb = length(B)-1;

%Polinomio producto del polinomio A por el operador 'delta' = (1-z^-1).
Avir = conv([1 -1],A);
```

A continuación se obtienen los polinomios $F_j(z^{-1})$ y $E_j(z^{-1})$, que se definen, respectivamente, como el resto dividido por z^{-j} y el cociente de la división polinómica entre el polinomio 1 y el polinomio $\tilde{A}(z^{-1}) = \Delta * A(z^{-1})$, por lo que se cumple la conocida como ecuación diofántica:

$$1 = E_j(z^{-1})\tilde{A}(z^{-1}) + z^{-j}F_j(z^{-1})$$

```
%Inicialización de la matriz de polinomios F.
F = zeros(N2,na+1);
poluno = [1 zeros(1, na+1)];
F1aux = poluno-Avir;
F(1,:) = F1aux(1,2:na+2);

%De forma recurrente, se calculan los sucesivos polinomios de F.
for j=1:(N2-1)
    for i=0:na-1
        F(j+1,i+1) = F(j,i+2) - F(j,1)*Avir(1,i+2);
    end
    F(j+1,na+1) = -F(j,1)* Avir(1,na+2);
end
```

```
%Partiendo de que E_1 = 1 y usando los polinomios F, se calculan los E.
E=zeros(N2,N2);
E(1,1) = 1;
for j=1:(N2-1)
    E(j+1,:) = E(j,:);
    E(j+1,j+1) = F(j,1);
end

%Tras haberse usado para calcular los polinomios Ej, se recorta el vector
%polinomial F para dejarle solo los polinomios a partir de d+1.
F = F(d+1:end,:);
```

Como se ha comprobado, es obligatorio el cálculo primero de los polinomios F para que después sean estos usados en el cálculo de los polinomios E, ya que el polinomio E_{j+1} es igual al E_j pero añadiéndole el primer término del polinomio F_j correspondiente.

A continuación, se calcula un nuevo polinomio denominado G_j como el producto del polinomio E_j y el polinomio del denominador, B. Con lo que ya se tendría la mejor predicción para las salidas del sistema en base a elementos ya definidos:

$$\hat{y}(t+j|t) = G_j(z^{-1})\Delta u(t+j-1) + F_j(z^{-1})y(t)$$

Con el objetivo de encontrar una estructura matricial que describa lo anterior, se divide el vector de predicciones en (1) una respuesta forzada, es decir, provocada por los cambios en la acción de control y que depende de la matriz G y (2) una respuesta libre, que es la que tendría el sistema si no se actuara sobre él, es decir, si los incrementos en la acción de control fuesen nulos. Esta solo depende de valores pasados, más concretamente del vector F y de una matriz G'.

Con esto, se puede reescribir lo anterior como:

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{f}$$

donde

$$\boldsymbol{f} = F(z^{-1})y(t) + G'(z^{-1})\Delta u(t-1)$$

Por lo que con la obtención de estos tres términos: vector F y matrices G y G', se tendría todo lo necesario para poder comenzar la optimización. Cabe destacar que con el objetivo de reescribir el término de respuesta libre usando una analogía a lo que sería un vector de estados, suele ser común en la bibliografía encontrarlo de la forma:

$$\boldsymbol{f} = F_x(z^{-1})\,\boldsymbol{x}$$

```
%Cálculo de los polinomios Gj.
Gpols= zeros(N2,N2+nb);
for j=1:N2
    Gpols(j,:) = conv(E(j,:),B);
end

%Montaje de la matriz G a partir de los polinomios contenidos en Gpols.
G = zeros(N2-d,N2-d);
for j=1:N2-d
    k=1;
    for i=j:-1:1
        G(j,k) = Gpols(j+d,i); %Se usan los polinomios desde d+1.
        k=k+1;
    end
end
```

```
%Cálculo de la matriz G', que se denota como Gprima.
Gprima=[];
if ((nb+d)>0) %Si el grado del polinomio B es mayor de cero o existe
    %retraso, entonces Gprima tendrá valor distinto de cero.
    Gprima = zeros(N2-d,nb+d);
    for j=1:N2-d
        Gprima(j,:) = Gpols(d+j,(1+j):(j+nb+d));
    end
end
%Se reescribe el término de la respuesta libre, f, como f=Fx*x, donde
Fx = [F, Gprima];
```

En este punto se trata el único aspecto que todavía no ha aparecido en el programa: la ley de control. Como se comentó con anterioridad, esta ley de control responde a una función cuadrática simplificada de la general, que escrita en forma matricial usando la nomenclatura de este capítulo es:

$$J = (Gu + f - w)^T(Gu + f - w) + \lambda u^T u$$

Desde un punto de vista matemático, se puede demostrar que, en el caso de no haber restricciones, la optimización de este tipo de funciones da lugar a una expresión explícita, que puede ser implementable en un programa sin la necesidad de hacer uso de funciones internas de cálculo numérico. Para ello, es necesario transformar la función de coste hasta que quede de una forma similar a lo que se conoce como "Quadratic Programming" o QP.

$$J = \frac{1}{2}u^T Hu + bu + f_0$$

Donde se ha definido:

$$H = 2(G^T G + \lambda I)\ ;\ b = 2(f - w)^T G\ ;\ f_0 = (f - w)^T(f - w)$$

Efectivamente, siendo el objetivo del control obtener el argumento que minimiza en *u* la función de coste *J*, se tiene lo siguiente:

$$u^* = \arg\min_u J(u) = -H^{-1} b^T$$

Como es lógico, la optimización debe hacerse dentro del bucle de simulación y en cada tiempo de muestreo, ya que el vector *b* está cambiando constantemente. Sin embargo, para reducir el coste computacional del algoritmo, al no variar la matriz *H*, ni por tanto su inversa, estas se calculan únicamente una vez antes de entrar al bucle.

Finalmente, es necesario definir una serie de parámetros para la simulación. Tanto el tiempo de simualción como la referencia, se fijan en los mismos valores que en la simulación de controladores lineales, para que los resultados sean más fácilmente comparables con los del control PD. Por un aspecto práctico, es imprescindible que la simulación empiece, al menos dos tiempos de muestreo después del valor del orden del denominador, ya que hasta ese punto se deberá acceder, y no tendrían sentido índices negativos. De igual manera se inicializan los vectores en los que se guardarán la salida, la acción de control y sus incrementos.

```
%Matriz invariante para la minimización.
H = 2*(G'*G + lambda*eye(Nu)); Hinv = inv(H);

%Tiempo de simulación y valor de inicio de la misma.
TMAX = 300; ini = max(na,nb+d+1)+1;

%Referencia e inicialización de vectores para la simulación.
w = [0.5*ones(TMAX/3,1);0.75*ones(TMAX/3,1);0.25*ones(TMAX/3,1)];
y = zeros(TMAX,1); u = zeros(TMAX,1); deltau = zeros(TMAX,1);
```

En este momento ya se tiene todo lo necesario para comenzar el bucle de simulación. En él, en primer lugar, se calcula la salida del sistema usando los polinomios de la función de transferencia en discreto como se hizo en el algoritmo para simular el control PD, pero ahora teniendo en cuenta el posible retraso que tenga. En segundo lugar, el definido como "vector de estados", que sin serlo, contiene los valores presentes y pasados de la salida y los valores pasados de la acción de control y, a continuación, se calcula el vector *b*, siguiendo la definición de más arriba. Es en este punto en el que se calcula la optimización del sistema propiamente dicha, que da la secuencia óptima de acciones de control.

Siguiendo la estrategia de horizontes deslizantes, será únicamente la primera componente de esta secuencia la que se usa como incremento y, por tanto, se suma a la que se tenía en el tiempo de muestreo anterior para obtener, finalmente, la acción de control en el instante actual.

```
%Bucle de simulación.
for k = ini:TMAX-N2
    y(k) = -A(2:na+1)*y((k-1):-1:(k-na)) ...
        + B(1:nb+1)*u((k-d-1):-1:(k-1-nb-d)); %Salida del sistema.

    x=[y(k:-1:k-na);deltau(k-1:-1:k-nb-d)]; %"Vector de estados".
    b = 2*(Fx*x-w(k+N1:k+N2))'*G; %Vector b, que usa la parte útil de w.
    duopt = -Hinv*b'; %Óptimización.
    deltau(k) = duopt(1); %Uso de la primera de las componentes.
    u(k) = u(k-1) + deltau(k); %Obtención de la acción de control actual.
end
```

Una vez que se ha simulado, se representa gráficamente la salida frente a la referencia y, por otro lado la acción de control, al igual que se hizo en el control PD y se muestran los resultados.

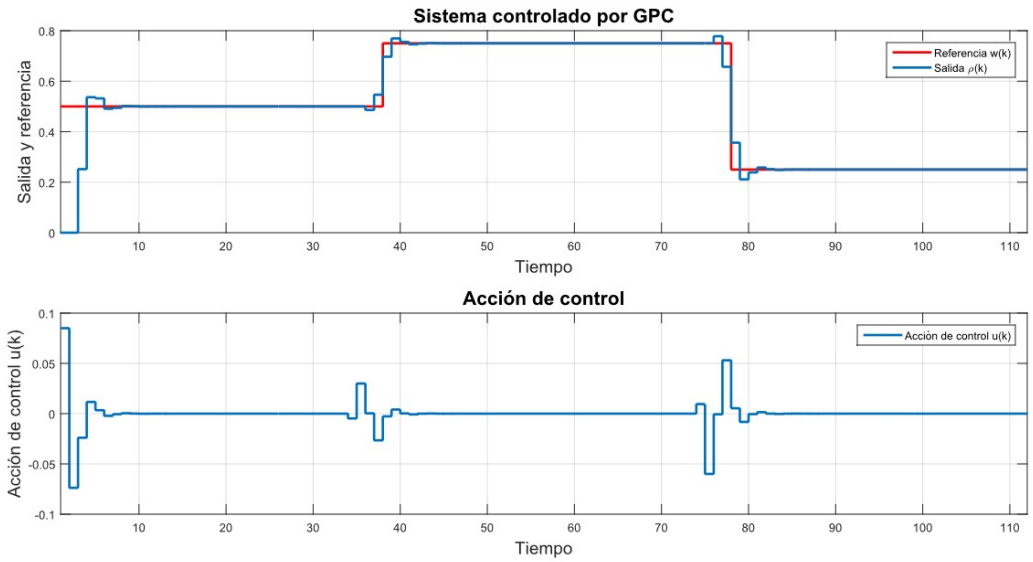

Ilustración 46 Sistema controlado por GPC: salida y acción de control

En la ilustración anterior, puede verse como, efectivamente, el controlador predice las salidas en lugar de actuar únicamente sobre aspectos pasados, que ya no pueden ser solucionados. Con esto, se consigue que se anticipe a los cambios en la referencia, y por tanto, que la salida empiece a subir (o a bajar) antes de que lo haga la señal en escalón que representa a la referencia.

Por otro lado, el control es muy rápido y preciso. En ningún momento se aprecian sobreoscilaciones elevadas, oscilaciones mantenidas, y mucho menos errores en régimen permanente.

Además, se observa que el hecho de que se haya implementado un cierto retraso en el sistema, no es un problema para el control GPC, ya que lo solventa como propia parte del sistema, sin mayor problema. Radica aquí una de las mayores ventajas del control GPC frente al control PID. Se muestra a continuación dos nuevas gráficas para comprobar lo anteriormente dicho. La primera es la aplicación del algoritmo explicado en este apartado, pero suponiendo que el sistema no tiene retraso, esto es, $d = 0$ y la segunda, es con un retraso tres veces mayor al anteriormente representado, es decir, $d = 3$.

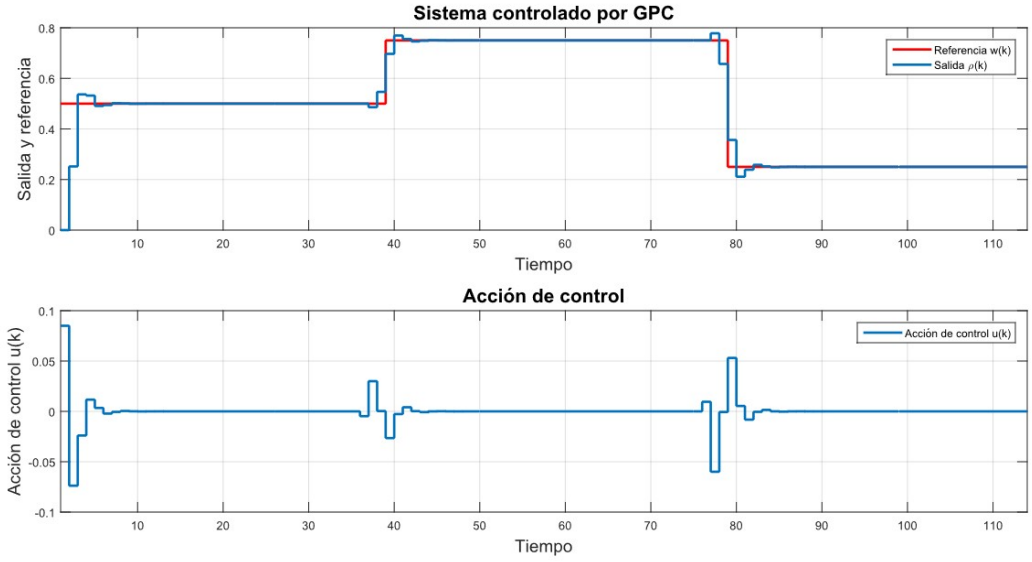

Ilustración 47 Sistema controlado por GPC: salida y acción de control sin retraso

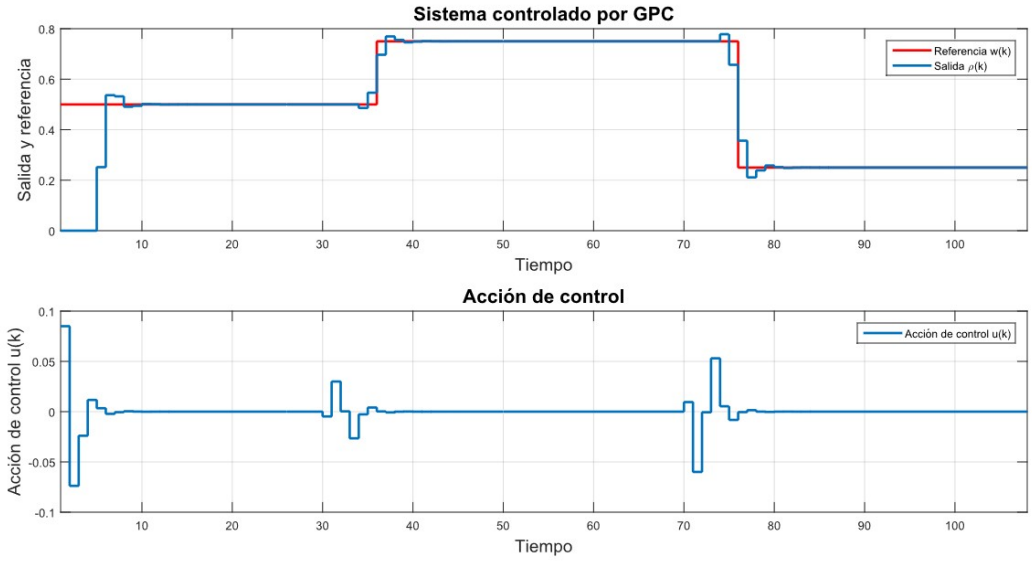

Ilustración 48 Sistema controlado por GPC: salida y acción de control con retraso d=3

Efectivamente, los únicos cambios relevantes que se aprecian son los referentes a los primeros instantes de simulación, en los que no se ve cambio en la salida aunque la referencia ya sea distinta de cero. Sin embargo, en los siguientes cambios de referencia, este retraso no es observable. Esto es debido a que en el primer cambio, al ser las acciones de control pasadas nulas, el algoritmo no tiene información suficiente para actuar a tiempo, apreciándose entonces el retraso del sistema; pero en los siguientes saltos, las acciones de control se adelantan a los cambios de referencia, actuando antes de que estos se produzcan, por lo que la salida cambia también y el retraso que hubiera en el sistema queda solventado de manera brillante.

Sin embargo, es preciso indicar que para que esto funcione de manera adecuada y sus beneficios sean suficientemente buenos, es necesario que sean conocidas las referencias futuras que se pretende que el sistema alcance, ya sea como un vector de valores, o de cualquier otra forma. Es decir, que estas no deben cambiar en tiempo real mientras que el control se está llevando a cabo y las acciones de control se están calculando. En ese caso, los resultados, si bien serán muy satisfactorios y, en general, mejores que los controladores lineales, no podrán mostrar cambios en la salida antes de que cambien las referencias.

Con esto, se puede confirmar la bondad del Control Predictivo basado en Modelo en general y del algoritmo GPC en particular y como, si bien requiere de mayores conocimietos técnicos, matemáticos e ingenieriles para su implementación, al menos en simulación, los resultados son sorprendentemente mejores que los que pudiera dar un controlador lineal tipo PID, por muy buena y exacta que sea su sintonización.

Finalmente, cabe destacar que con unas simples modificaciones en el código, podrían introducirse restricciones al sistema, ya sea en la acción de control, en sus incrementos o en la salida. Si bien en ese caso, ya no se podría obtener una ecuación explícita que resuelva el problema de optimización y se tendría que trabajar con métodos numéricos para su resolución. La inclusión de restricciones es del todo impensable en otro tipo de controladores y es donde radica la mayor parte del mercado actual en torno al Control Predictivo. Si una compañía necesita un control complicado que deba atenerse a una serie de restricciones, lo más probable es que si espera resultados medianamente aceptables, se decante por este tipo de algoritmos.

8 SIMULACIONES EN TIEMPO CONTINUO

La vida no es sino una continua sucesión de oportunidades para sobrevivir.

Gabriel García Márquez

EN capítulos anteriores se ha estudiado de manera detallada el sistema "ball and beam" objeto de este Trabajo tomando como entrada del mismo el ángulo que forman las barras con la horizontal y como salida la posición lineal de la bola respecto al extremo de las barras. En este capítulo, tratando de dar el último paso antes de controlar el sistema real que se ha diseñado y construido, se plantea el esquema de control más parecido a la realidad, en el que la acción de control sobre la que se puede actuar no es el ángulo de las barras, sino el ángulo del servomotor. Como novedad, en este capítulo no se va a utilizar la función de transferencia del sistema, que como es sabido parte de la linealización en torno a un punto de equilibrio de las ecuaciones difenciales que dominan el sistema y por tanto, puede discrepar de la realidad. En su lugar, se van a integrar directamente estas ecuaciones no lineales para ver si la respuesta es muy diferente a lo que se había observado hasta ahora.

8.1 Introducción

Efectivamente, no se puede perder de vista que el único actuador del sistema es el servomotor, el cual trabaja usando como variable un ángulo medido sobre su eje, que puede oscilar entre 0º y 180º, por lo que el control que se implemente deberá dar finalmente un valor de este ángulo, denominado α. La primera de las opciones que se plantean es que el valor del ángulo θ que se obtenga del control PD, sea el que se use para introducirlo en el servomotor, debidamente corregido geométricamente con la aproximación que ya se presentó en el capítulo de Modelado Dinámico, y que no es más que una relación lineal entre ambos.

$$\alpha = \frac{L}{b}\theta$$

Siguiendo este razonamiento, el esquema de control que se obtiene es el que se representa en la siguiente Ilustración. Como puede observarse, existe una única realimentación negativa en la que se resta la salida de la posición lineal de la bola con la referencia que se le da como un tren de escalones de distinta amplitud y que, tras introducise en el bloque de control PD, da el ángulo θ. Aplicando directamente la ecuación del movimiento, se le realiza el seno a ese ángulo y se multiplica por la ganancia estática del sistema, para después integrar dos veces el resultado hasta llegar a la posición que se necesita.

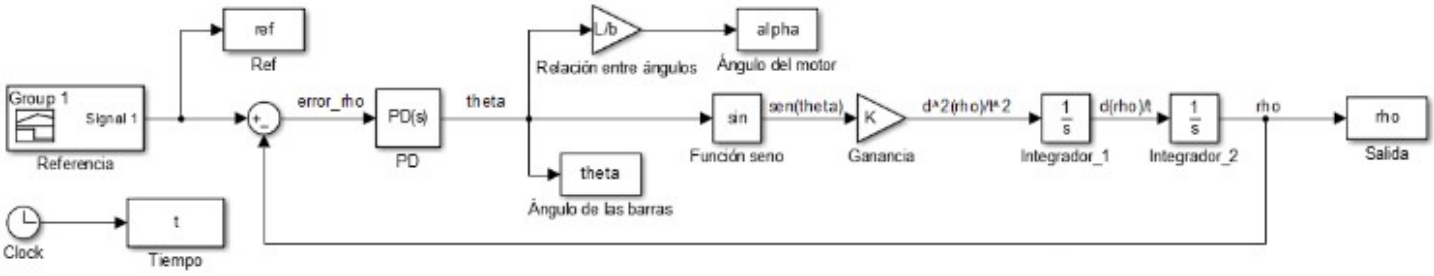

Ilustración 49 Esquema de control no lineal

Sin embargo, rápidamente se ve que se estaría implementando un control en bucle abierto del ángulo α del servomotor, ya que en ningún momento se comprueba el valor real del ángulo de las barras para compararlo con el que se debe alcanzar según el bloque de control PD, pudiendo producir errores que pueden llegar a ser considerables en la práctica y que es posible que no sean ni siquiera detectables.

Es por eso que se hace uso de una unidad de medida inercial (IMU) para tener en tiempo real el dato del ángulo real que forman las barras con la horizontal, pero para poder usar este dato debe completarse el esquema anterior incluyendo una segunda realimentación que permita calcular el error del ángulo θ, dando lugar a lo que se conoce como control en cascada.

8.2 Control en cascada

En Control de Sistemas, es típico encontrar situaciones en las que se puedan medir las perturbaciones que puedan llegar a afectar a la salida de los actuadores. Si no se hace nada para evitarlo, estas perturbaciones inciden sobre el sistema durante un tiempo sin que se realice una acción de control en su contra, es por eso que la idea es evitar que este efecto se acumule y acabe afectando de manera negativa a la salida.

Para ello, se utiliza un lazo interno de realimentación para controlar que el nivel de actuación aplicado coincida con el deseado. Este segundo lazo, que es interno al sistema de control principal se conoce como lazo de control secundario o esclavo; en contraposición a el primario o maestro, que es el lazo externo. Cabe destacar que es de vital importancia para el correcto funcionamiento del sistema que el lazo secundario sea más rápido que el primario. La estructura que se está explicando puede verse en la siguiente figura.

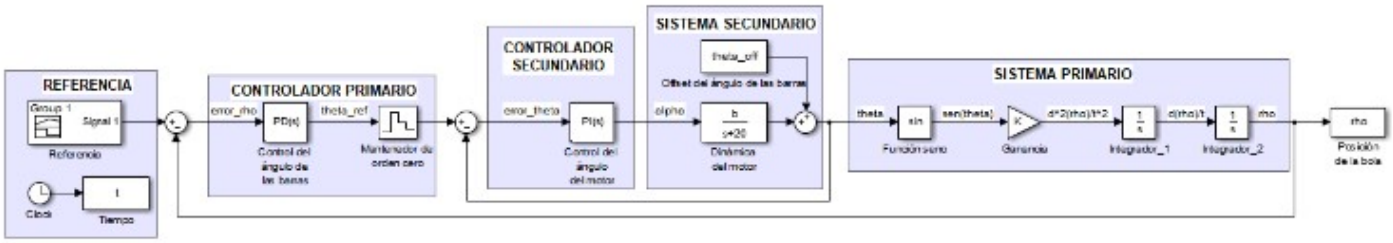

Ilustración 50 Control en cascada

Efectivamente, al sistema se le da una referencia para la posición lineal de la bola, ρ_{ref}, que se compara con la salida del sistema (dada por el sensor láser), ρ, y se obtiene un error en ρ que entra en el primer controlador. Este controlador da como resultado un valor del ángulo de las barras, θ_{ref}, que es el valor de referencia que se querría alcanzar,y que se alcanzaría instantáneamente en un sistema ideal. Este valor de referencia se compara con la salida del sistema secundario, gracias a la IMU, que da el ángulo real de las barras y se obtiene el error en θ que finalmente se introduce en el controlador secundario.

Este controlador secundario da el ángulo α al que debe moverse el servomotor, y que es en realidad, la única acción de control externa que tiene el sistema, por ser este de un solo actuador. La dinámica del motor da como salida el ángulo θ que se mueven las barras respecto de la horizontal. Finalmente, con este ángulo θ, se entra en el sistema primario que gracias a la doble integración de la aceleración, logra obtener la posición lineal de la bola.

Se destacan algunos aspectos relevantes del esquema anterior. Por un lado, el control del sistema primario sigue siendo un PD, tal y como se ha explicado detalladamente en capítulos anteriores. Tras este se introduce un mantenedor de orden cero para que los valores del ángulo θ_{ref} sean constantes en los instantes de muestreo en los que no se está calculando su valor.

Por otro lado, se ha supuesto que la dinámica del motor responde a un sistema de primer orden, con una ganancia que coincide con el radio de la excéntrica (cambiado de signo para lograr que se obtengan ángulos positivos) y un polo bastante rápido, con el objetivo de que no sea dominante con respecto al sistema primario, ya que al elegir un servomotor de gama media en lugar de uno gama baja, su respuesta temporal es lo suficientemente rápida y precisa como para aceptar dicha consideración en primera aproximación. Como es sabido, la mejor opción para controlar un sistema de primer orden es un controlador PI, que será diseñado por el método analítico o de cancelación de dinámicas. A la salida del sistema secundario se le incluye un posible offset en el ángulo θ para incluirle más realismo a la simulación.

Tal y como se ve reflejado en [34], la práctica común para la sintonización de controladores en cascada en empezar por el lazo secundario para después controlar el lazo primario, una vez que el ángulo θ se ajuste lo mejor posible a su referencia, θ_{ref}. El método de cancelación de dinámicas consiste en hacer que el cero del controlador PI se cancele con el polo de la función de transferencia de la dinámica del motor, para ello, se estudia el producto de ambos bloques.

$$C_2G_2 = K_p\frac{T_is+1}{T_is} * \frac{-b}{s+20} = K_p\frac{T_is+1}{T_is} * \frac{-0.05*b}{0.05s+1}$$

Por lo que se obtiene que el tiempo integral debe ser igual a la constante en bucle abierto: $T_i = 0.05$. Para el cálculo de la ganancia K_p, se impondrá que la constante de tiempo en bucle cerrado debe ser un quinto de la de bucle abierto, es decir, cinco veces más rápido, con lo que se llega a:

$$K_p = \frac{\tau_{ba}}{\tau_{bc}*b} = \frac{-0.05}{0.01*0.0425} = -235.2941$$

Una vez que se tiene el controlador secundario controlado, pasa a controlar el primario, para lo cual se busca una ganancia que no inestabilice el sistema y una constante derivativa que mejore el transitorio. El esquema que se utiliza para las simulaciones es el siguiente.

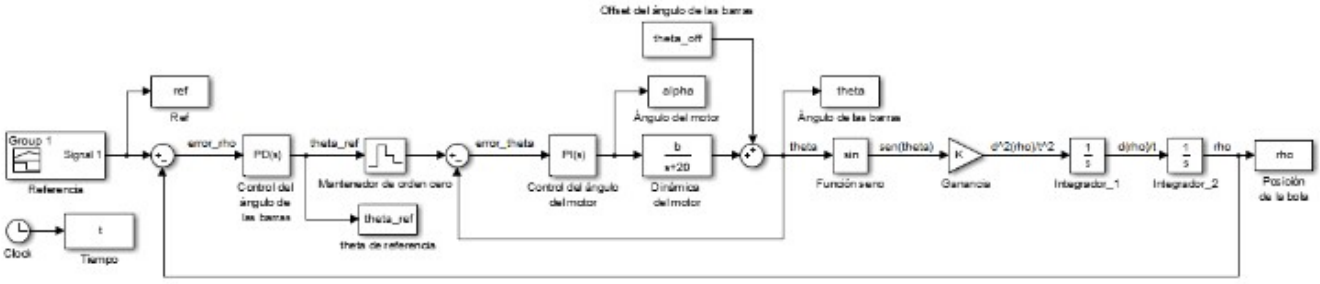

Ilustración 51 Esquema de simulación en tiempo continuo

Como puede verse, es el mismo esquema en base al que se ha explicado el control en cascada pero usando una serie de bloques para mandar al Workspace de MATLAB las variables que son necesarias mostrar en la representación gráfica.

Finalmente, se llega a la conclusión de que los parámetros más óptimos para el controlador primario en la simulación que se está realizando son:

$$K_p = 0.01\,;T_d = 6$$

De manera arbitraria, se elige un offset para el ángulo de las barras de cinco grados sexagesimales, debidamente pasados a radianes para introducirlos en el esquema antes del sistema primario. Con todo lo detallado en el capítulo se obtienen los siguientes resultados que se muestran de forma gráfica.

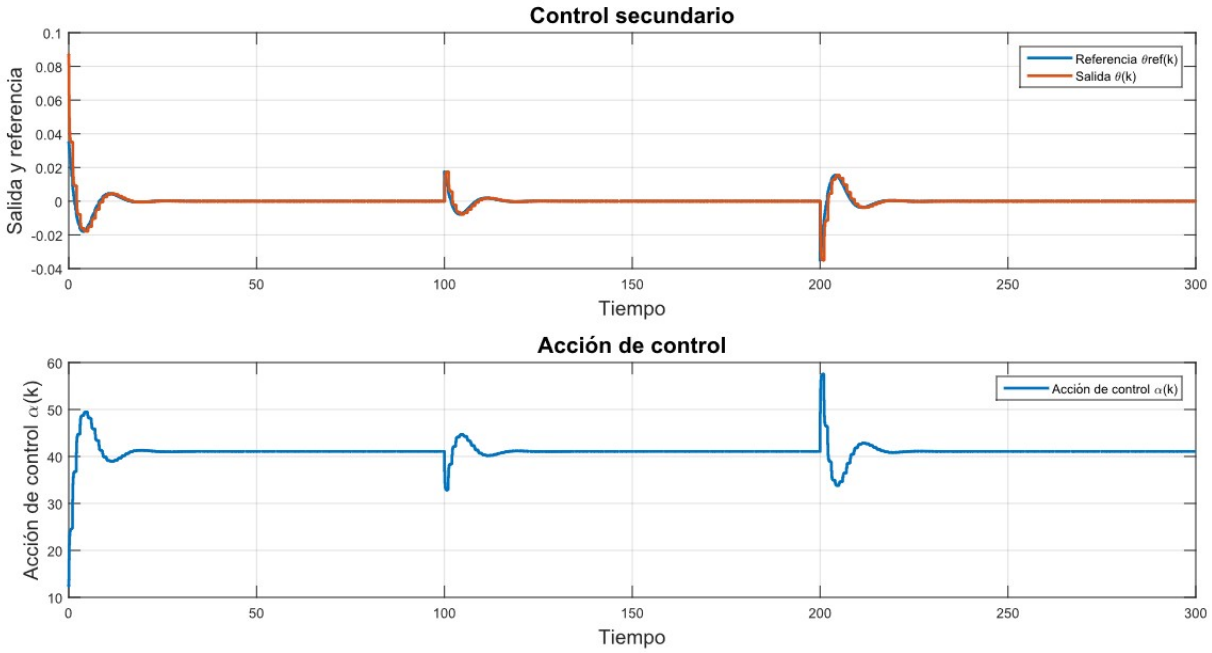

Ilustración 52 Control secundario

En este primer conjunto de gráficas se muestra el control secundario, es decir, el ángulo de inclinación de las barras frente a la referencia que se obtiene del control primario. La acción de control que se representa en la segunda de las gráficas es el ángulo del motor, en grados sexagesimales. Se observa como el control es más que correcto, por lo que se considera bien controlado el lazo interno.

Es lógico que cuando se implemente en la realidad el ajuste no será tan óptimo y preciso, debido al ruido en los sensores y a las holguras del motor y de las piezas impresas por la impresora 3D, además de por el hecho de que la dinámica del motor pueda que no responda exactamente a un sistema de primer orden sino a otra dinámica más complicada y difícil de averiguar.

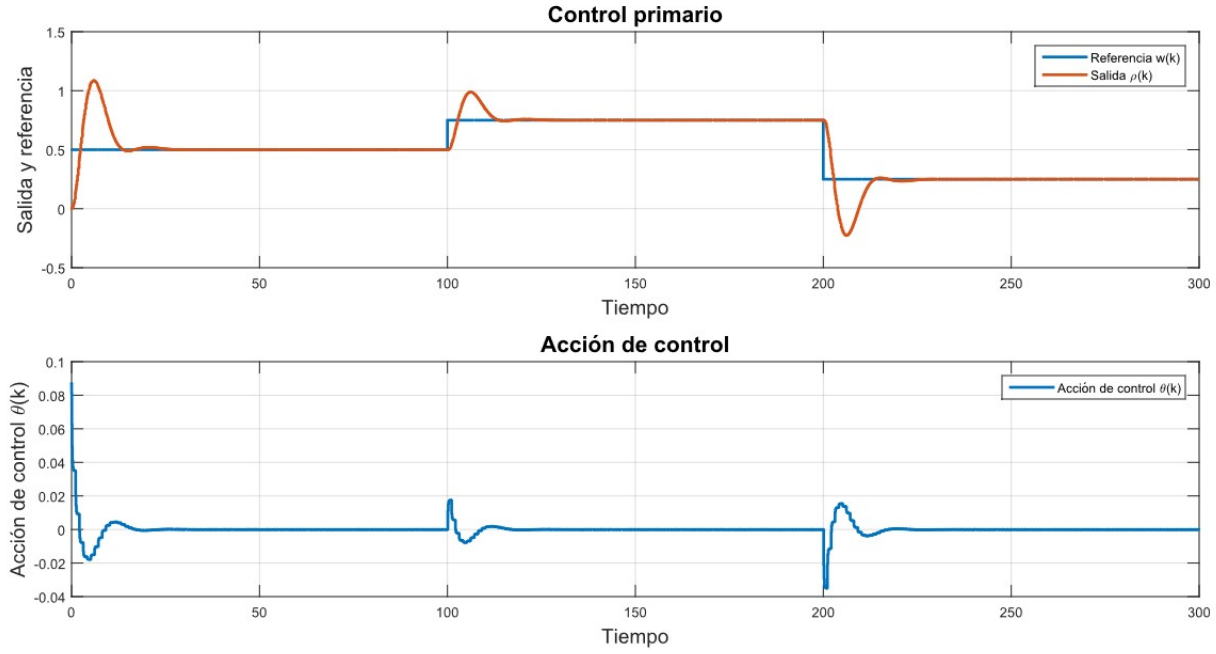

Ilustración 53 Control primario

En este segundo par de gráficas se ve el sistema primario o externo, es decir el control de la posición de la bola frente a la referencia que se le ha dado como un tren de escalones de distinta amplitud. El control es bastante bueno, observando una única sobreoscilación de bastante amplitud en los cambios de referencia, pero solventándola rápidamente y consiguiendo que se estabilice de la forma esperada.

En cuanto a la acción de control, en este caso se representa el ángulo de inclinación de las barras, que aunque no se puede actuar directa y físicamente sobre él, hace las veces de acción de control para este sistema primario.

De igual forma a como se ha señalado con respecto a los resultados del control del sistema secundario, lo que aquí se presentan son simulaciones, y como tal se deben tomar. Aún habiéndose estudiado de manera pormenorizada los sistemas y aún habiéndose ajustado todos los términos intervinientes de la manera más exacta posible, cuando se pruebe en el sistema real construido, los resultados no serán tan óptimos.

Aún así, se quiere hacer notar que las simulaciones son de gran importancia en Ingeniería, ya que dan una primera idea del comportamiento de los sistemas y ayudan en gran medida a ahorrar recursos temporales, económicos y materiales en el siguiente paso de la cadena: el control del sistema real.

9 CONTROL LINEAL DE LA PLANTA REAL

Los humanos no pueden soportar mucha realidad.

T. S. Eliot

TODA vez que se tiene el sistema perfectamente caracterizado y se han realizado las preceptivas simulaciones del mismo, es el momento de pasar a controlar la planta real del prototipo "ball and beam". Como ya se ha explicado pormenorizadamente con anterioridad, el sistema puede ser controlado con un controlador PD, que será la primera de las propuestas que se hagan. A continuación, se introducirán las medidas de la IMU para implementar un control en cascada de dos controladores lineales: un PD y un PID. Los resultados gráficos de todos los experimentos se mostrarán en este capítulo usando la comunicación con MATLAB explicada en el Capítulo 4.

Sin embargo, previo a esto, debe hacerse una caracterización del comportamiento real tanto del sensor láser como de la relación entre el ángulo de las barras y el del servomotor, con el objetivo trabajar con medidas lo más cercanas posibles a la realidad.

9.1 Caracterizaciones

9.1.1 Relación entre los ángulos

Efectivamente, se pretende en primera instancia, observar la relación entre el ángulo del servomotor y el que las barras forman con la horizontal en cada momento, ya que el primero de ellos es la acción de control externa que en realidad se puede modificar y el segundo es la salida de un futuro controlador lineal que se diseñe. Para ello, se implementa un programa sencillo en Arduino que vaya dando incrementos de un grado sexagesimal al ángulo del servomotor y en cada tiempo de muestreo guarde el ángulo que mide el "Roll" de la IMU. Tanto el Hardware como el Software necesario para la utilización de ambos dispositivos, ha sido detallado en los Capítulos 3 y 4, respectivamente.

Cabe destacar que, tras la realización de distintas pruebas y con el objetivo de que no vascule y pueda llegar a dar la vuelta la pieza que une las barras con la excéntrica del servomotor, los ángulos máximos y mínimos que se le podrán pedir al servomotor serán de 180º y 45º, respectivamente. Lógicamente, el primero de ellos se corresponderá con el valor mínimo del ángulo de las barras y el segundo, al máximo. Se toman del "Monitor Serie" los valores de θ y se copian tal cual al principio del código que se adjunta a continuación. Como puede verse, además se define el vector de ángulos del servomotor y, tras esto, se calcula la recta que mejor ajusta los datos introducidos en sendos vectores, usando para ello la función "polyfit" de MATLAB con un polinomio de primer orden.

Para observar el resultado de la forma más visual posible, se programa la representación en una única gráfica del conjunto discreto de datos obtenidos en el experimento y la recta de mejor ajuste que se ha calculado.

```
%Ángulo que forman las barras con la horizontal.
theta = [ ... ];
%Ángulo del motor, desde 180° hasta 45° y vuelta hacia atrás.
alpha = [180, 180:-1:45, 45:180, 180]';

%Se busca la recta que más se ajuste y se forma dicha recta.
p = polyfit(alpha,theta,1);
y = alpha*p(1)+p(2);

%Se representa gráficamente.
plot(alpha, y, 'LineWidth',2);
hold on
plot(alpha,theta,'LineWidth',2);
grid
legend('Recta de mejor ajuste','Datos reales');
xlabel('Ángulo del motor \alpha','FontSize', 14),ylabel('Ángulo de las
barras \theta','FontSize', 14);
title('Relación entre los ángulos \theta y \alpha','FontSize', 16)
```

La ecuación de la recta que se ha obtendido es la siguiente:

$$\theta = -0.0371\alpha + 2.9356$$

Que como puede observarse en la gráfica generada, representa casi a la perfección la realidad del comportamiento característico de la relación de ambos ángulos. Nótese como se ha dado un recorrido tanto de ida como de vuelta al motor para que los resultados sean más exactos.

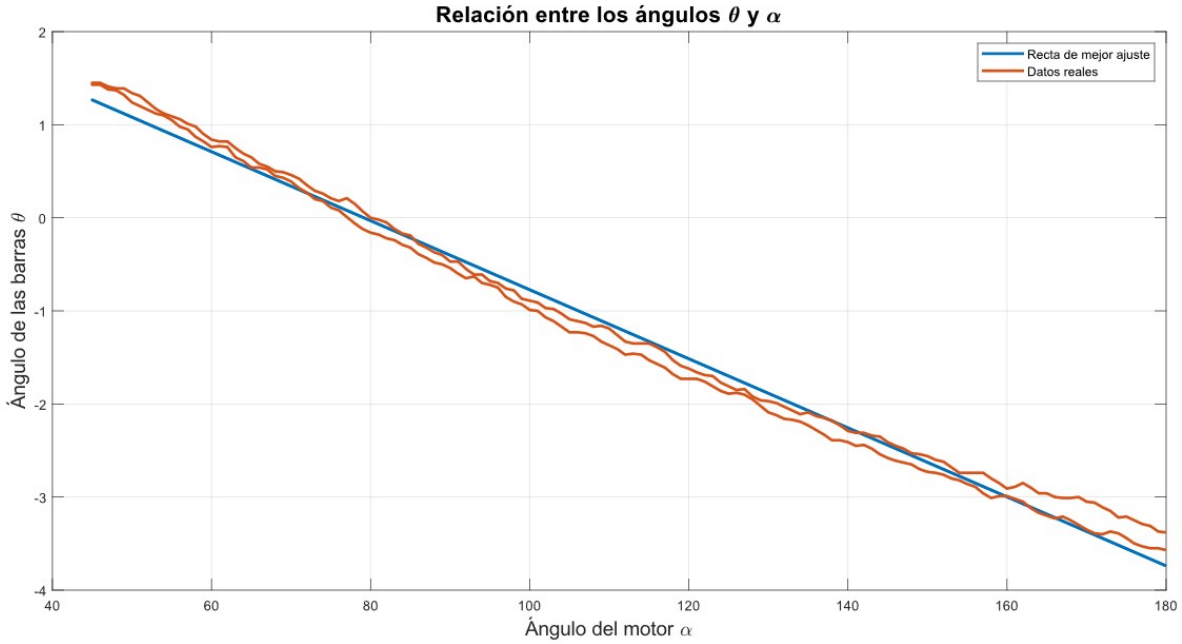

Ilustración 54 Caracterización de la relación entre los ángulos α y θ

Por consiguiente, cuando el controlador lineal dé un valor del ángulo θ que deba alcanzarse en cada instante de muestreo para que la bola se posicione en el "set-point" marcado, basta despejar de la anterior ecuación para saber el ángulo α que debe introducirse al servomotor como acción de control.

$$\alpha = -\frac{(\theta - 2.9356)}{0.0371}$$

9.1.2 Distancia medida por el sensor

Siguiendo el mismo objetivo que en el subapartado anterior, se van a caracterizar las medidas del sensor láser. Previo a esto, es importante reseñar en este punto las dificultades que se han presentado durante la elaboración de este Trabajo debido a las medidas del sensor láser. Como se indicó en el Capítulo correspondiente, uno de los mayores problemas de este sistema es la obtención de la posición de la bola en cada instante de muestreo y lo cierto es que el sensor láser no lo soluciona totalmente, por lo que es necesario dar una serie de indicaciones.

Por un lado, los sensores de tipo ToF, como el VL35L1X, están pensados para medir posiciones de objetos voluminosos y, sobre todo, de superficie plana, de modo y forma que las ondas del mismo puedan rebotar en el objeto y llegar de nuevo al sensor. La dificultad de su uso en el sistema "ball and beam" recae la propia esfericidad de la bola, ya que el punto que se está buscando medir es uno muy concreto (el centro de la misma) y son muy pocas las ondas que rebotan en ese punto exacto y son capaces de volver al sensor.

En consecuencia, tras la realización de una serie de pruebas experimentales con la bola metálica a la que se ha venido haciendo mención en los Capítulos de Hardware, Modelado y Simulaciones, se considera necesario el uso de una bola de mayor diámetro. Tras probar primero con una bola de billar blanca de unos 2 cm de radio, se obtienen unos resultados prácticamente igual de pobres, por lo que definitivamente se decide utilizar una bola de 4 cm de radio impresa en 3D con las mismas condiciones que el resto de piezas del sistema. Efectivamente, al aumentar el diámetro de la bola, aumenta también la superficie que está expuesta a una posible recepción de las ondas infrarrojas del sensor láser y con ello, mejoran notablemente las medidas.

Además de esto, se aprecia que uno de los principales problemas de la medición del sensor es que, debido a la forma cónica en que se emiten las ondas infrarrojas, muchas de estas se expanden hacia fuera del sistema, sin que haya posibilidad alguna de que lleguen a la bola. Para solucionar este hecho, se considera la posibilidad de incluir unas paredes laterales de unos 30 cm de alto y 60 cm de largo que estén colocadas de forma perpendicular a la base del sistema en su parte más alejada al sensor, esto es, la más cercana al servomotor. Con esto, se consigue que las ondas infrarrojas que tendían a "escaparse" del sistema, choquen en las paredes interiores y, finalmente, lleguen a rebotar en la bola para que el sensor pueda medir su posición de la forma más concreta posible. La estructura final puede verse en la Ilustración 55. En la misma, también se observa como se ha utilizado una pieza extra (de color gris), para elevar la altura a la que realmente está el sensor, para que este apunte lo más exactamente posible al centro de la bola.

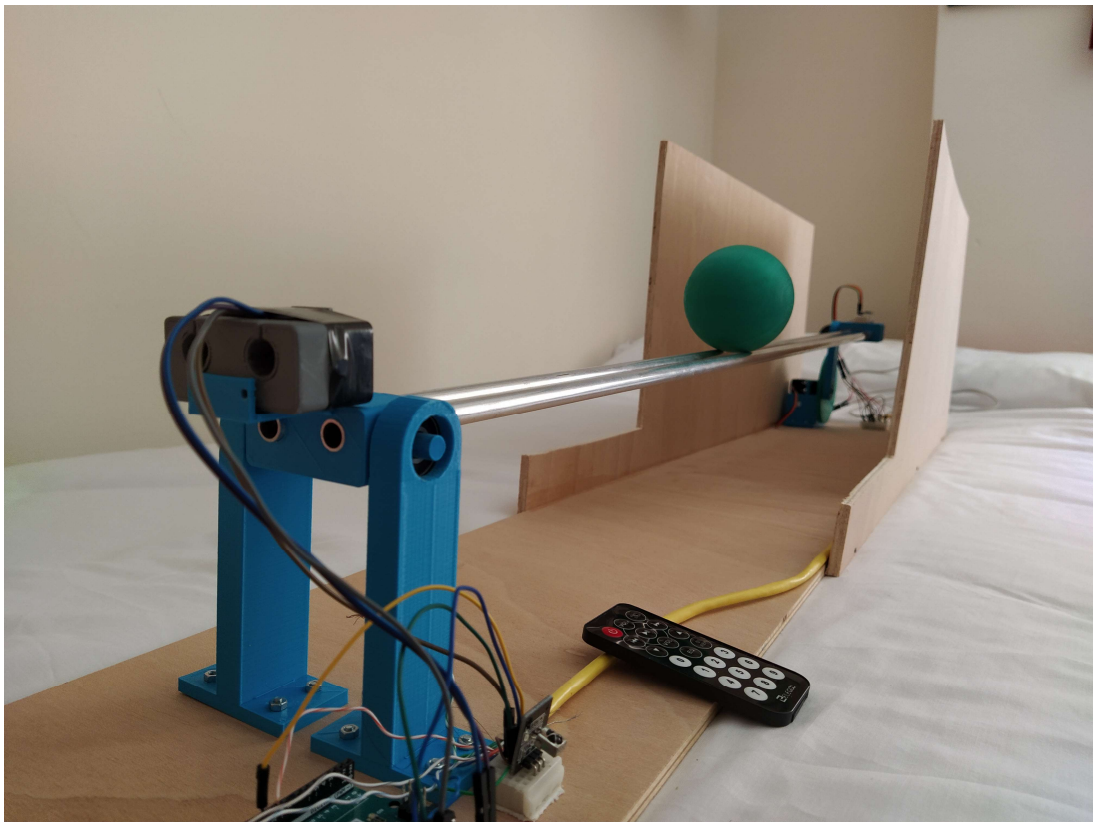

Ilustración 55 Sistema real

Una vez que se han indicado las pesquisas que han marcado la definitiva estructura del sistema "ball and beam" y se ha presentado una fotografía con su visualización, es el momento de pasar a la caracterización propiamente dicha del sensor láser. Para ello, se ha colocado una cinta métrica en la base del sistema y se ha ido moviendo la bola de centímetro en centímetro a lo largo de las barras, mientras se iba anotando en una tabla, el resultado que en ese momento daba, a través del "Monitor Serie" de Arduino, el sensor láser. Para ello se ha ejecutado un sencillo código programado en base a la función LeeSensor, que se explicó en el Capítulo de Software. El resultado de la experiencia, que se ha repetido en varias ocasiones con resultados similares, se resume en la siguiente tabla, colocando únicamente los valores múltiplos de cinco para evitar que su extensión fuera innecesariamente larga.

Tabla 2 Caracterización del sensor láser

Distancia real medida con la cinta métrica (cm)	Distancia medida con el sensor láser (cm)
5	5
10	10
15	15
20	20
25	26
30	33
35	42
40	51
45	62
50	68
55	75
60	80
65	82
70	85
75	87
80	89

Como puede observarse a simple vista, la relación no es perfectamente lineal, ya que a medida que las distancias empiezan a aumentar, estas tienden a pecar por exceso. Sin embargo, se entiende que podría caracterizarse por una recta que fuese más acertada que simplemente la toma directa de los datos medidos por el sensor. Es por ello que se aplica a los datos recogidos un código muy similar al utilizado para la caracterización de la relación entre los ángulos y que se ha expuesto anteriormente, obteniendo que la recta que mejor ajusta los resultados es la siguiente:

$$pos_{medida} = 1.2705\, pos_{real} - 1.8526$$

En consecuencia, si se quiere aplicar en los programas de control la medida de la posición que mejor se ajuste linealmente a la realidad, bastará con aplicar la siguiente ecuación al final de la función que calcula la posición.

$$pos_{real} = \frac{pos_{medida} + 1.8526}{1.2705}$$

Sin embargo, tal y como se puede ver en la Ilustración 56, en la que se muestra el conjunto de medidas tomadas junto con la recta de mejor ajuste, la aproximación de las medidas del sensor láser por una recta no es perfecta. Es por esto que se propone un segundo método alternativo en el que la aproximación sea un poco más fina. Se trata de lo que se conoce como "lookuptable" y consiste en guardar en el programa de Arduino una matriz con las componentes de la caracterización y, tras realizar la lectura, ver entre qué dos valores se encuentra para dar como medida real el resultado de la interpolación lineal entre ambos, tal y como se detalla en el trozo de código que se adjunta a continuación.

```
int distan, i;
int A[2][76] = {{...},{...}};
for(i=0;i<75;i++){
  if(distan>=A[1][i] && distan<=A[1][i+1]){
   distan=A[0][i]+((A[0][i+1]-A[0][i])/(A[1][i+1]-A[1][i]))*(distan-A[1][i]);
    }
  }
```

Con ambos métodos se consiguen resultados más que aceptables tras la aplicación de todas las indicaciones ya explicadas para lograr optimizar las medidas, como el aumento del tamaño de la bola o la colocación de paredes verticales. Si bien el resultado no es completamente exacto con la aproximación lineal, por lo que en los controles que se realicen y que serán mostrados a continuación será usado, en principio, el método recién explicado relativo a la "lookuptable" y se comprobará que, en el rango de referencias que se van a utilizar, no se van a producir errores apreciables.

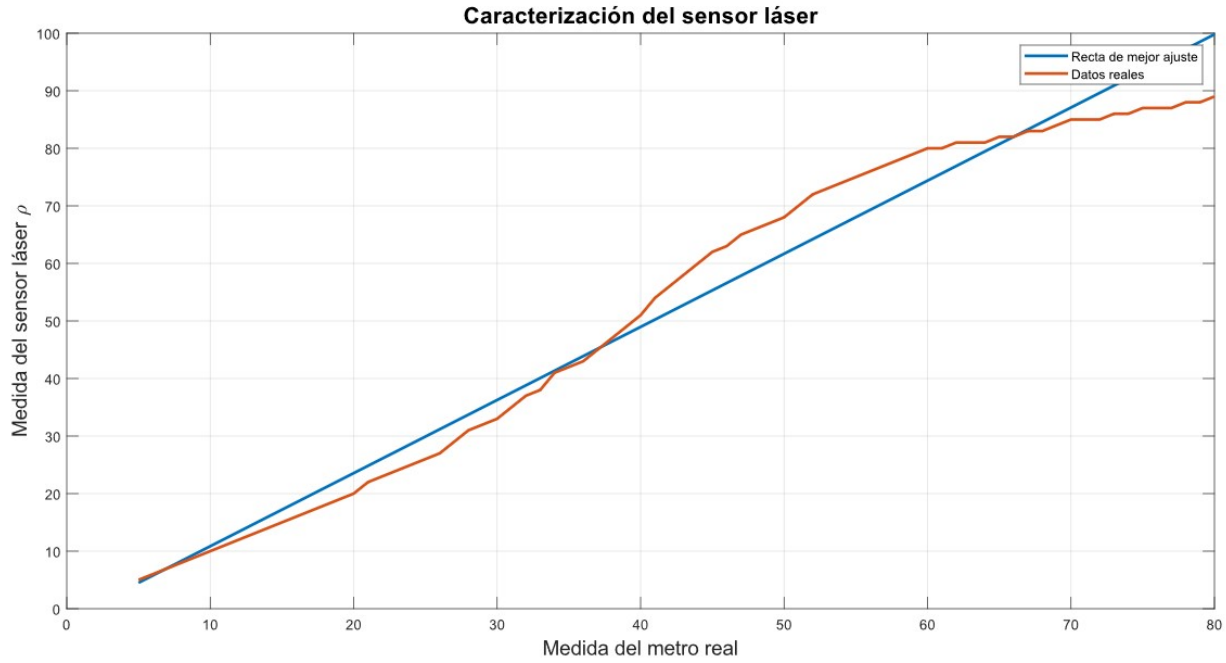

Ilustración 56 Caracterización de las medidas del sensor láser

9.2 Control PD de la planta real

Como se indicó en los Capítulos relativos a las Simulaciones, el sistema "ball and beam" no puede ser controlado simplemente con un controlador proporcional, ya que, debido a su doble integrador, la salida en bucle cerrado sería la de un sistema críticamente estable. En consecuencia, la primera propuesta que se hace para controlar la planta real es un controlador Proporcional Derivativo. Este primer controlador no hace uso de la IMU, ya que utiliza la relación obtenida en el apartado anterior para relacionar el ángulo de las barras (que es la salida del controlador) con el del servomotor (que es la actuación real que se tiene sobre el sistema).

Si bien uno de los códigos se adjuntará íntegramente al final de este Trabajo en el Anexo A, se indica aquí que para el reparto de tiempos se ha utilizado lo que se conoce como ejecutivo cíclico, que es una estrategia sencilla para la implementación de sistemas de tiempo real en los que no hay demasiadas tareas involucradas y por tanto, el programador puede repartir el tiempo a cada una de ellas. Además, dada su sencillez no es necesario tener grandes conocimientos de programación en Arduino para poder comprender el código rápidamente, por lo que hace que este sea fácilmente legible y es de aplicación en todos los modelos de placas de todas las marcas del mercado. Para su implementación, se deben seguir los siguientes pasos:

- Se mide el instante de inicio de cada iteración.
- Se mide el instante de finalización de cada iteración.

- Se calcula la diferencia entre ambos, obteniendo el tiempo que ha tardado en realizarse una iteración.
- Se lleva a cabo, justo antes del comienzo de la siguiente iteración, una espera correspondiente a la diferencia entre el tiempo de muestreo predefinido, en este caso 35 ms y lo que se ha tardado en realizar las tareas.
- Se actualizará el tiempo que tarda cada iteración únicamente cada vez que se hagan todas las tareas, con el fin de evitar demasiados cálculos innecesarios.

Siguiendo lo anterior, en este programa serán dos las tareas que hay que realizar. Por un lado, el cálculo de la acción de control y su paso al servomotor para que actúe consecuentemente y por el otro, la lectura de la referencia. Debido a que la segunda tarea se realiza mediante la lectura de un receptor infrarrojo al que le llegan los cambios a través de un mando a distancia, se considera suficiente comprobar si se ha realizado una modificación en la referencia una de cada cinco veces que se calcula la acción de control, comprobando en la práctica que no se observan problemas al respecto.

En términos generales, el bucle que se repite continuamente en el Arduino tiene la estructura que sigue a continuación.

```
void loop() {
  /* Inicialización de variables del control */
  /* ... */
  delay(T); // Espera para cumplir tiempo de muestreo.
  antes = millis();
  cont ++;

  /* Prueba del sensor */
  //Solo se hace una vez para ver que el sensor no se ha quedado colgado.
  if(flag == 0){
    LeeSensor(&pos);
    delay(10);
    flag = 1;
  }

  /* Primera tarea: Control */
  //Se realiza cada instante de muestreo.
  if ((encendido == 1) && (cont != 5)) {
    LeeSensor(&pos); // Lectura de la entrada.
    pos = pos + 4; // Ajuste de offset.

    /* Saturación */
    if (pos > 90) pos = 90;
    if (pos < 5) pos = 5;

    /* Cálculo de la acción de control */
    ek = pos-ref;
    theta = Kp*(ek + (Td/Tm)*(ek - ek_1));
    ek_1 = ek;

    /* Paso del ángulo de las barras al del servomotor */
    alpha = -(theta - 2.9356)/0.0371;
    /* Saturación */
    if (alpha > alpha_motor_max) alpha = alpha_motor_max;
    if (alpha < alpha_motor_min) alpha = alpha_motor_min;

    /* Escribimos salida */
    servo.write(alpha);
  }
```

```
  /* Segunda tarea: Receptor IR */
  if (cont == 5)
  { /* Según lo explicado en el capítulo de Software */
    /* Se actualiza el tiempo de espera */
    despues = millis();
    cont = 0; //Se reinicia el contador.
    T = Tm - (despues - antes);
  }
}
```

Como es lógico, este trozo de código solo da unas ideas generales de como se debe implementar el ejecutivo cíclico y se aprovecha para introducir la tarea del control PD al completo. Nótese como se utiliza la función LeeSensor que, al igual que el funcionamiento del receptor IR, se explicó en el Capítulo 4, por lo que no se ha incluido aquí para no caer en repeticiones. Además, es de reseñar el uso de saturaciones tanto en la salida como en la actuación ya que se ha comprobado experimentalmente que esos valores máximos y mínimos no deben superarse nunca por la propia física del sistema. Finalmente, cabe destacar sobre el código anterior la implementación de la ecuación que relaciona el ángulo del servomotor con el de las barras.

A continuación, una vez se tiene el programa que implementa el control, se puede pasar al diseño de los parámetros del controlador PD. Para ello, si bien se utilizarán como partida los resultados obtenidos en las simulaciones de los Capítulos anteriores, se diseñarán siguiendo un método heurístico en el que se elegirá en primer lugar una ganancia del controlador que sea lo suficientemente grande para activar el sistema pero sin que provoque sobreoscilaciones demasiado elevadas y después se modificará el término derivativo para que el transitorio mejore, y en este caso, haga que el sistema pase de ser críticamente estable a simplemente estable.

Siguiendo esta estrategia se llega a la conclusión de que la mejor combinación de parámetros obtenida ha sido $K_p = 0.13$ y $T_d = 450$. Para poder observar los resultados de este controlador, se debe ejecutar el programa de Arduino para que se cargue en la placa y, posteriormente, correr el programa de MATLAB que se explicó en el Capítulo 4 y con el que se van a obtener las representaciones gráficas. Efectivamente, como ya se comentó en su momento, se genera automáticamente la siguiente figura:

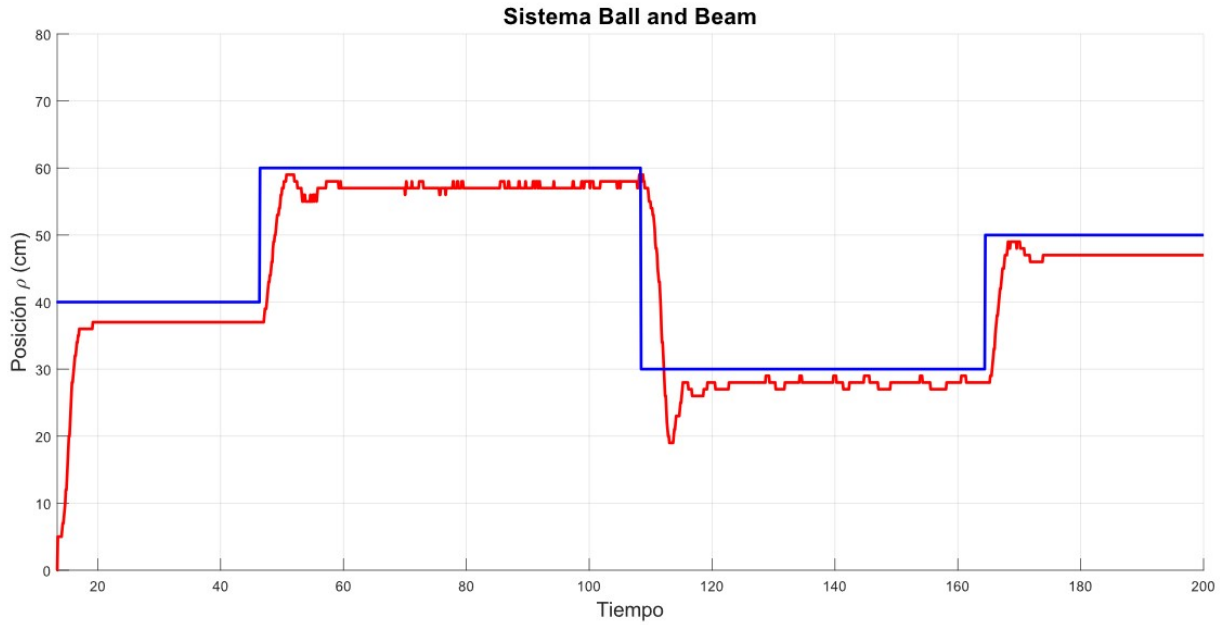

Ilustración 57 Gráfica en tiempo real de los resultados del control PD

Esta gráfica aparece en primer a instancia vacía y en ella se ve la posición a cero y la referencia en el valor en el que está inicializado en el programa, en este caso, a 40 cm. Conforme van pasando los instantes de muestreo, el programa de Arduino va pasando la información por el puerto serie a MATLAB y este lo va representando hasta obtener finalmente la gráfica anterior.

Al terminar el experimento, se generan también de forma automática dos nuevas gráficas con dos ejes en paralelo en cada una de ellas. La primera tiene en la parte alta lo mismo que se ha mostrado en tiempo real, es decir, la posición instantánea de la bola y la referencia que en cada momento se está recibiendo, pero de forma más ajustada y con su leyenda correspondiente. Justo debajo se verá la acción de control, esto es, el ángulo que forman las barras con la horizontal y que se calcula por el propio controlador.

La segunda de las gráficas que aparecen (Ilustración 59) vuelve a mostrar en su parte de arriba el ángulo de las barras, para poder compararlo con el ángulo que se le dice al servomotor que debe alcanzar, que se representa consecuentemente en la parte de abajo de la gráfica. En este apartado no dará demasiada información, pero cobrará más importancia en el siguiente, cuando se incluya la IMU. Para facilitar la visualización y de los resultados, se han usado únicamente dos colores y se ha mostrado en todos ellos leyenda y nombre de los ejes.

Finalmente, antes de mostrar los resultados, se debe destacar que se han recortado de todas las gráficas los primeros instantes de muestreo en los que la referencia tenía un valor no nulo (por estar definida así en el programa), mientras que la posición se mantenía a cero, debido a que aún no se había pulsado el botón "POWER" que activa el control; entendiéndose que de no hacerlo, podría confundirse este lapso de tiempo con un retraso del sistema, nada más lejos de la realidad.

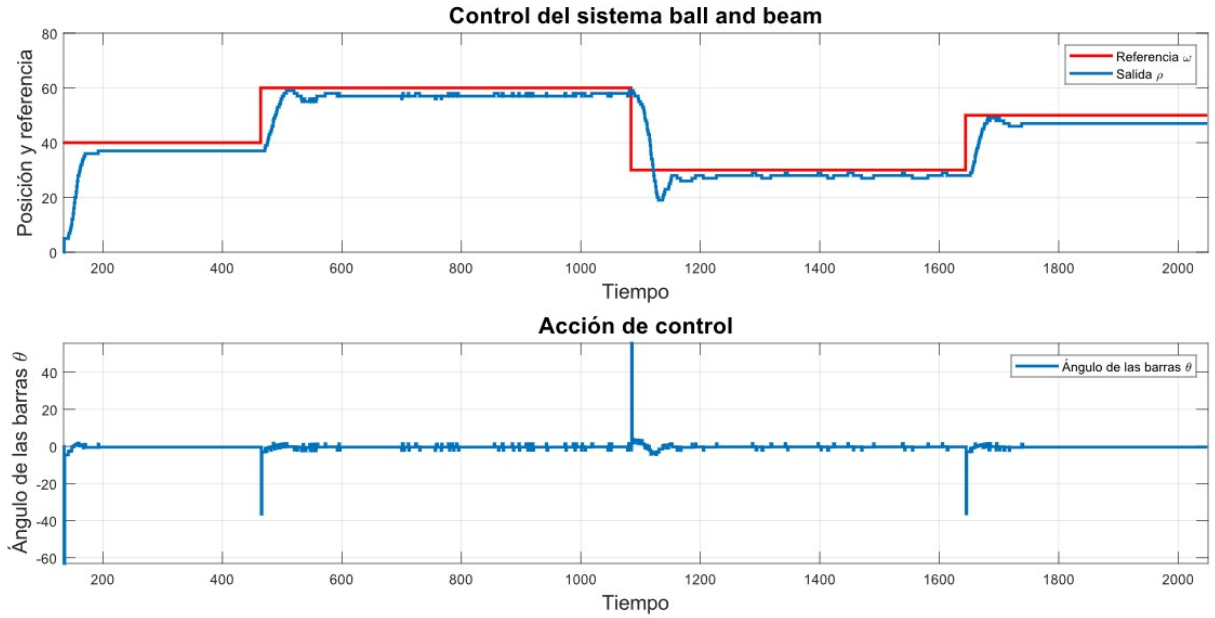

Ilustración 58 Posición lineal con controlador PD

En esta primera gráfica, puede verse como, inicialmente, el sistema tiende a alcanzar la referencia que se le ha marcado (en este caso de 40 cm), sin demasiada dificultad, en un corto espacio de tiempo y sin presentar sobreoscilaciones ni oscilaciones mantenidas. En los siguientes cambios de referencia en sentido positivo (de 40 a 60 cm y de 30 a 50 cm), se observa una leve subida de la posición de la bola por encima del punto que finalmente alcanza en régimen permanente. No ocurre lo mismo en el tercer cambio de referencia del experimento, el único en sentido negativo (de 60 a 30 cm), en el cual sí se observa una clara sobreoscilación al llegar hasta los 20 cm. Sin embargo, rápidamente es capaz de recuperar la posicón y alcanzar el régimen permanente sin mayores dificultades.

Por otro lado, es necesario destacar que, sobre todo en las referencias más extremas, el sistema oscila levemente debido al ruido del sensor, entre otros factores, que aún estando en una posición fija, puede dar un valor levemente superior o inferior que haga que el sistema tenga que recuperar la posición más cercana posible al "set-point"

Por último, cabe reseñar como se aprecia en todas las referencias dadas un cierto error en régimen permanente, pecando siempre por defecto y aproximadamente de una amplitud similar en todos ellos (unos dos centímetros), que aunque no es algo excesivamente crítico, se tratará de solventar con la inclusión de una compensación de zona muerta en el siguiente experimento, donde se explicará el porqué de este error.

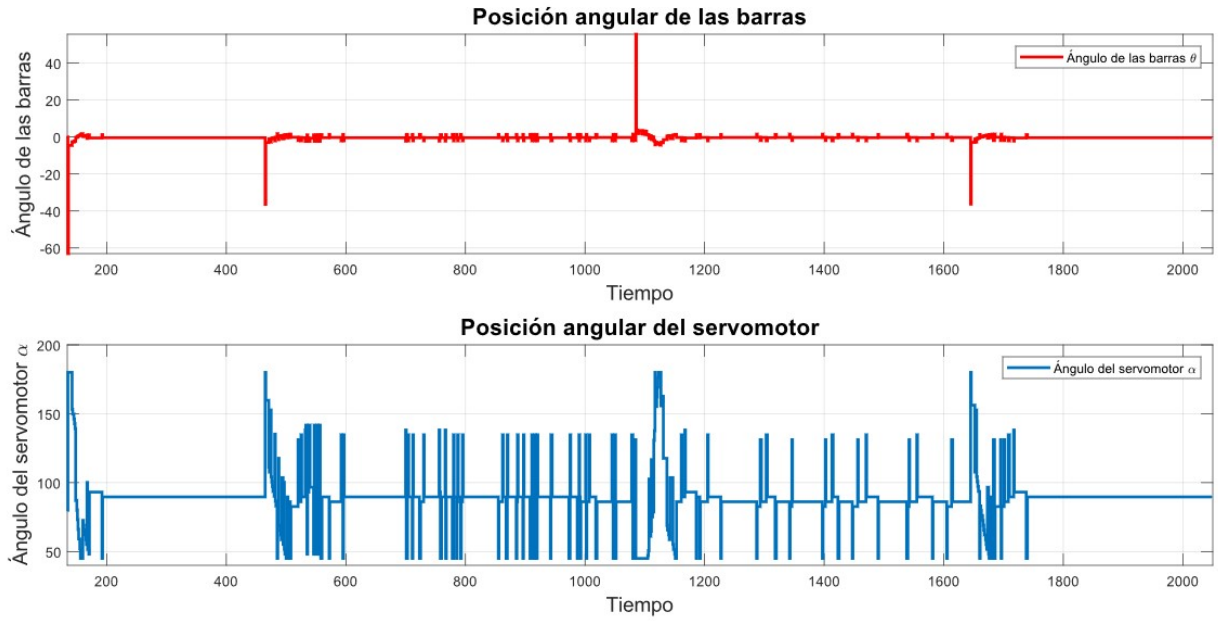

Ilustración 59 Posiciones angulares con controlador PD

En cuanto a la segunda de las gráficas, se observa como la acción de control sube (o baja) de manera muy abrupta cuando se realizan los cambios en la referencia, ya que es en ese momento cuando el error crece más y, sobre todo, cuando más difiere el error en el instante de muestreo actual con respecto al anterior, haciendo que, en consecuencia, tanto la acción proporcional como la derivativa se vean fuertemente aumentadas. Sin embargo, en la gráfica del ángulo del servomotor, se ve como oscila constantemente entre valores altos y bajos, excepto cuando la acción de control, es decir, el ángulo θ,es nulo. Esto es debido a que se está haciendo en realidad un control en bucle abierto del ángulo del servomotor y, por tanto, los posibles ruidos que haya en la medida, se ven amplificados por el término proporcional y/o derivativo y posteriormente atenuados por la relación entre los ángulos dada por la caracterización.

9.2.1 Compensación de zona muerta

Como ya se ha indicado en el primer experimento, se observa un cierto error en régimen permanente en la posición de la bola con respecto a la referencia solicitada. Esto es debido a la existencia de lo que se conoce como zona muerta que no es más que unos ciertos valores de la acción de control, que por ser muy próximos a cero, no tienen efecto sobre el sistema, es decir, si por ejemplo se da un ángulo de 0.05º, puede que no sean suficientes para que la bola se mueva. Para abordar la compensación de esta zona muerta, lo más común según [35] es la modificación de la señal de control de modo y forma que cuando sea inferior a cero se le reste este valor y cuando sea superior a cero se le sume. De este modo, la señal de entrada al sistema siempre va a estar fuera del valor que no provoca cambios en el sistema. De forma matemática puede verse comos sigue:

$$u_k = \begin{cases} u_k + ZM \; si \; u_k > 0 \\ u_k - ZM \; si \; u_k < 0 \end{cases}$$

En este caso, con el objetivo de evitar que el servomotor siga oscilando cuando ya esté en régimen permanente por ruidos en la medida, se va a incluir una tolerancia, $\epsilon = 0.01°$, en el cálculo del ángulo de las barras con compensación de zona muerta. Tras algunas pruebas se determina que $ZM = 0.1°$, por lo que basta con añadir las siguientes líneas de código tras el cálculo del valor del ángulo θ de las barras para aplicar la compensación.

```
if (theta>=epsilon) theta = theta+ZM;
else if (theta<=-epsilon) theta = theta-ZM;
else if(theta>-epsilon && theta<epsilon) theta = 0;
```

Tras realizar algunas pruebas, se considera oportuno reajustar levemente los valores de los parámetros del controlador hasta dejarlos en $K_p = 0.14$ y $T_d = 447$, obteniendo los resultados que se muestran a continuación.

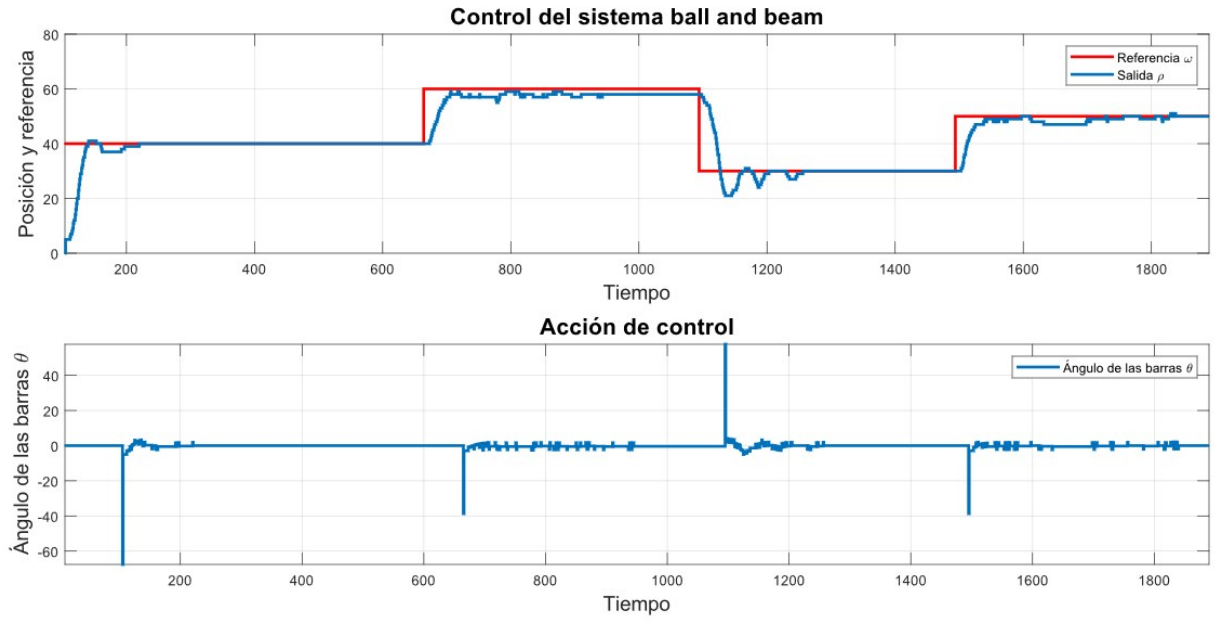

Ilustración 60 Posición lineal con controlador PD y compensacion de zona muerta

En este segundo experimento se observa como, al introducir la compensación de zona muerta, el error en régimen permamente se ve drásticamente reducido hasta prácticamente desaparecer en alguna de las amplitudes de los escalones dados como referencia y solo en uno de ellos se mantiene levemente. Además, no se observan prácticamente sobreoscilaciones ni oscilaciones mantenidas excepto en el cambio de referencia en sentido negativo, como ocurría en el experimento anterior; mientras que el tiempo de subida en todos los cambios de referencia es relativamente corto, concluyendo, por tanto, que se ha obtenido un controlador más que correcto.

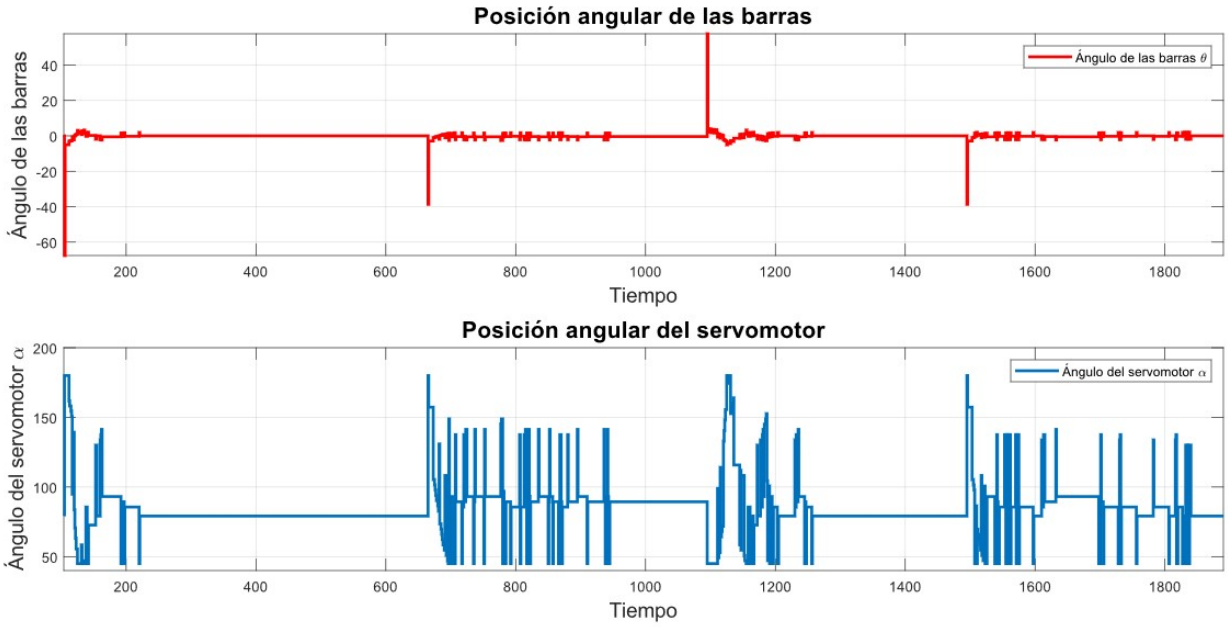

Ilustración 61 Posiciones angulares con control PD y compensación de zona muerta

En relación con la segunda gráfica, sin embargo, no se observan grandes diferencias con respecto al experimento sin compensación de zona muerta, algo que por otro lado era de esperar, ya que los grandes cambios en la acción de control volverán a producirse cuando se introducen las modificaciones en el valor de la referencia, los ruidos en las medidas se mantienen y, el control de la posición angular del servomotor sigue siendo en bucle abierto, por lo que se producirán grandes picos que suben y bajan con un solo instante de

muestreo de diferencia y que no son nada buenos para el control. Esta última circunstancia solo podrá ser resuelta con un control en bucle cerrado de la posición angular y, por tanto, con la utilización de la IMU, como se detallará en el siguiente apartado.

9.3 Control PD en cascada con PID de la planta real

Con el objetivo de poder tener una realimentación del valor real del ángulo que forman las barras con la horizontal en cada instante de muestreo, se hará uso de la ya explicada Unidad de Medida Inercial. Con ella, tal y como se detallaba en el Capítulo 8, se formará, junto con el control PD implementado en el apartado anterior, lo que se conoce como un control en cascada.

En cuanto a la implementación en código de este controlador, la estructura a seguir es muy parecida a la que se usó para el control PD. La diferencia recae en el hecho de que ahora son tres las tareas a llevar a cabo por lo que el reparto del tiempo en el ejecutivo cíclico se complica. En este caso, la tarea más rápida y que, por tanto, deberá realizarse en cada instante de muestreo es el control secundario, es decir, la lectura de la IMU, su comparación con el valor del ángulo θ que se desea alcanzar en ese momento, la obtención del ángulo α como acción de control y su envío al servomotor.

Como puede verse en el trozo de código adjunto, solo se calculará la acción de control si se ha pulsado la tecla "POWER" del mando a distancia. En caso contrario (que no se haya pulsado o que se haya apagado), se resetean los errores y se colocan las barras en la posición inicial en la que están aproximadamente rectas. Nótese además cómo el cálculo de esta acción de control cambia con respecto a la del apartado anterior (cuando únicamente se usaba la relación de los ángulos en bucle abierto), ya que aquí se utiliza un controlador PID, por lo que debe actualizarse en cada iteración el valor de la integral del error, I_k, como la suma de todos los errores anteriores, además de actualizar los errores como ya se hacía en el control PD.

```
/* Primera tarea: Control secundario*/
//Se realiza cada instante de muestreo, debe ser lo más rápido posible.
if (encendido == 1) {
  /* Lectura de la entrada */
  LeeIMU(&Roll); //Se utiliza la función para leer la IMU.

  /* Cálculo de la acción de control */
  ek2 = Roll - theta; //Se calcula el error.
  Ik = Ik_1 + ek2; //Se calcula la integral del error.
  alpha = Kp2 * (ek2 + Tm*Ik/Ti2 + (Td2/Tm)*(ek2-ek2_1));
  Ik_1 = Ik; //Se actualizan los errores.
  ek2_1 = ek2;

  /* Saturación */
  if (alpha > 180) alpha = 180;
  if (alpha < 45)  alpha = 45;

  /* Escribir la salida */
  servo.write(alpha);
 }
  /* Si está apagado el control */
   else {
    Ik = 0; //Se resetean los errores.
    Ik_1 = 0;
    servo.write(80); //Se lleva a la posición inicial.
  }
```

Tras esto, se deberá realizar el control primario, que consiste en calcular el valor del ángulo de las barras deseado tal y como se hizo en el apartado anterior. Esta tarea se realizará cinco veces más lenta que el control secundario, es decir, cada vez que el contador sea múltiplo de cinco. La última tarea, la lectura de un posible cambio de la referencia, deberá hacerse más lenta aún, concretamente, cada vez que se realizan dos medidas de la posición, es decir, cada vez que el contador sea mayor de diez. Al acabar esta tarea, además, el contador se deberá resetear para volver a valer cero.

Se pasa a continuación al diseño de los parámetros de ambos controladores. Como se sabe que el controlador primario ya ha funcionado en el apartado anterior, se parte de estos valores para calcular los valores óptimos del controlador secundario, que deben ser tal que se alcancen las posiciones angulares requeridas por el controlador primario lo suficientemente rápido, pero sin que llegue a sobreoscilar o a oscilar de forma mantenida en el tiempo. Cabe destacar que el término derivativo se utiliza para mejorar el régimen transitorio, pero que al ir multiplicado por la derivada del error (en realidad por una aproximación de esta), puede provocar que se vea en gran medida aumentado el ruido de las lecturas de la IMU, por lo que su valor deberá ser bajo. El término integral se usará, definitivamente, para conseguir un error en régimen permanente nulo ante entradas en escalón. Finalmente, con los valores del control secundario ya fijados, se deberán retocar los del control primario para intentar llegar a la mejor respuesta posible. Con todo, los parámetros elegidos se resumen en la siguiente tabla.

Tabla 3 Parámetros del control PD en cascada con PID

Parámetro	Control primario	Control secundario
K_p	0.14	7.75
T_d	600	1.5
T_i	-	1000

Con los parámetros de los controladores ya fijados y el código del programa elaborado, se está en condiciones de mostrar los resultados obtenidos en forma de representación gráfica. De igual forma a como se hizo anteriormente, se vuelve a utilizar el software de comunicación con MATLAB para esta labor.

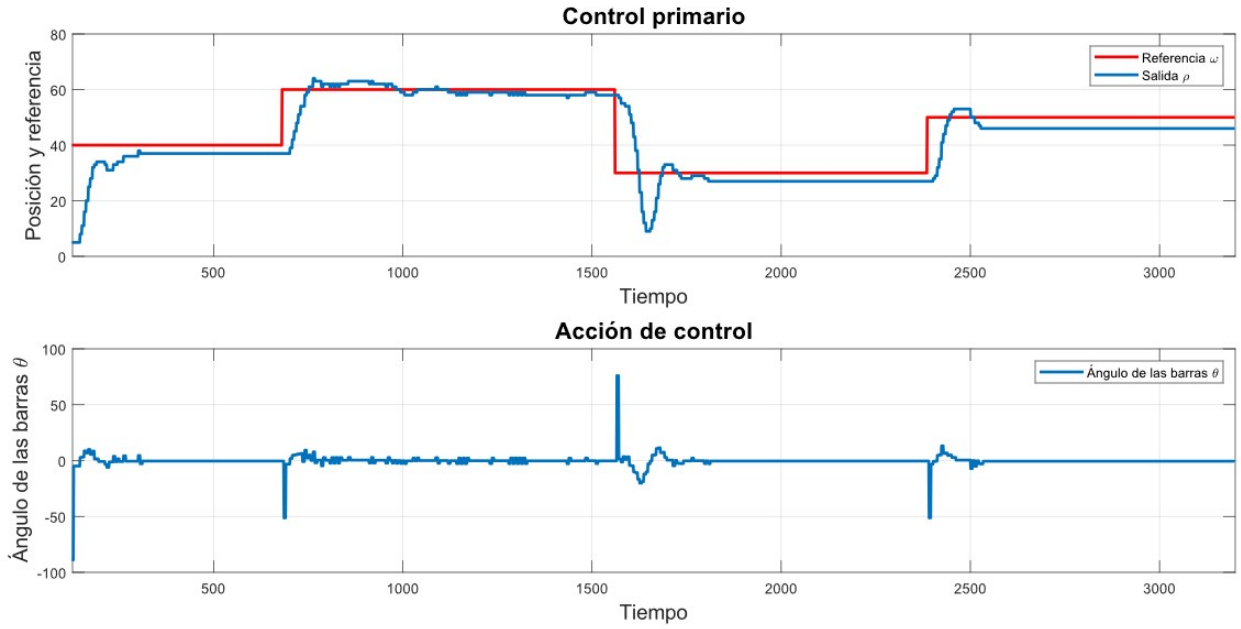

Ilustración 62 Control primario del controlador PD en cascada con PID

En el caso del control primario, en el que se pretende que la posición de la bola alzance la referencia solicitada a través del mando a distancia, se observa en primer lugar que el sistema responde de manera bastante rápida a los cambios de referencia, siendo los tiempos de subida relativamente bajos. En cuanto a los regímenes transitorios, se observan ciertas sobreoscilaciones de no demasiada amplitud en el caso de los cambios de referencia positivos. Sin embargo, en el cambio de referencia en sentido negativo, en el que se pasa de 60 a 30 cm, la sobreoscilación sí que es considerablemente alta, debido, probablemente, a que el margen de actuación en este sentido es mayor.

Por otro lado, se observa que, en relación con los regímenes permanentes, el sistema presenta un cierto error (de hasta 4 cm en el caso de la última referencia) nada despreciable que hace pensar que la compensación de la zona muerta que se hizo en el apartado anterior, será de igual forma necesaria ahora, por lo que se aplicará en la siguiente experiencia.

Finalmente, cabe destacar de nuevo cómo se hace notar el ruido del sensor láser en los valores cercanos a 60 cm, que provoca que el sistema esté constantemente moviéndose del punto de referencia al recibir un valor distinto a este y, consecuentemente, enviar una acción de control que hace que se mueva la bola.

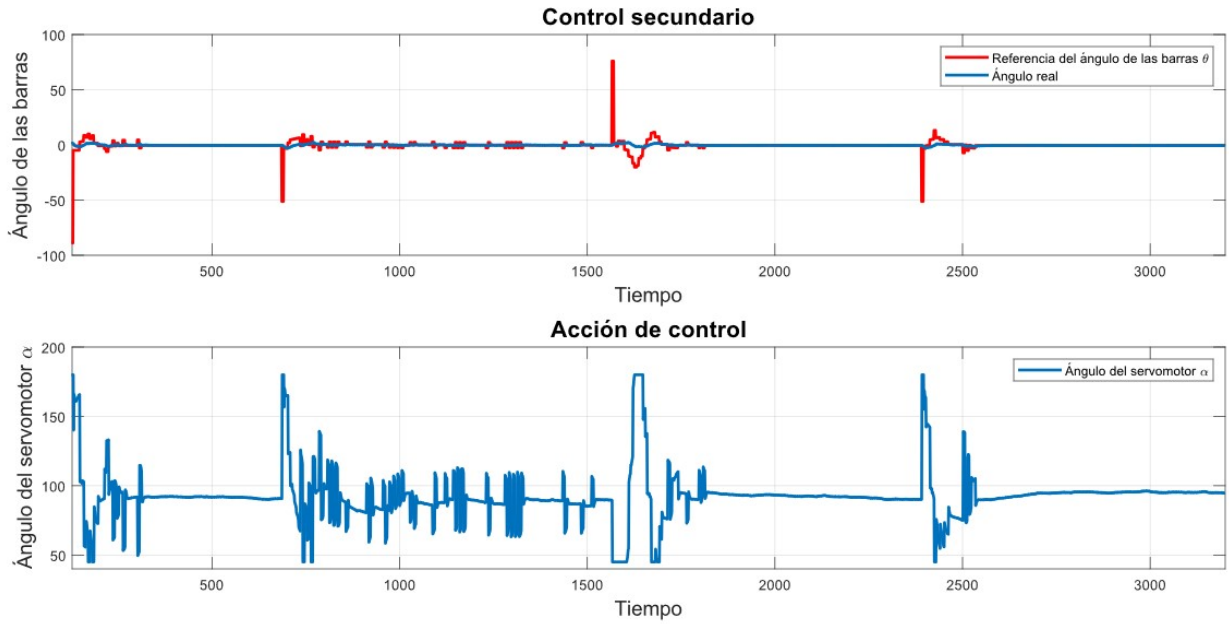

Ilustración 63 Control secundario del controlador PD en cascada con PID

Pasando a la interpretación de la segunda gráfica, es aquí donde se encuentran las diferencias más reseñables con respecto al controlador en el que únicamente se usaba un control PD, ya que, efectivamente, se observa como ahora sí que se produce un control en bucle cerrado de la posición angular de las barras. En la parte superior, se puede ver en colo rojo el valor del ángulo θ que según el control primario se debería alcanzar y que hace aquí las veces de referencia, su curva es muy parecida a la del apartado anterior, presentando picos elevados en los cambios de referencia. En color azul se ve el valor medido por la IMU en cada instante de muestreo y cómo tiende en todo momento a alcanzar su referencia. Como es lógico, no es capaz de llegar a los valores altísimos y a la vez instantáneos de los picos ni a valores del ángulo que por la física del propio sistema "ball and beam" estén fuera de su rango, pero no es este un problema que merezca mayor preocupación.

En la parte inferior, se ve como ahora los valores del ángulo del servomotor no son simples picos hacia arriba y hacia abajo como parecía en el apartado anterior, sino que su amplitud está modelada por el error que presenta el ángulo real con respecto a su referencia, como era de esperar al haber usado para su obtención un control con realimentación negativa. Además, se observa como el efecto integral del control secundario hace que el error en régimen permanente de este sea nulo.

9.3.1 Compensación de zona muerta

Exactamente igual a como se hizo en el apartado anterior, se va a incluir una compensación de zona muerta para intentar reducir los errores en régimen permanente que se han observado de forma clara en el subapartado que precede a este. Tal y como se hizo entonces, basta con sumarle o restarle al ángulo de las barras θ, el valor de la zona muerta fijado en 0.1º, usando la tolerancia ϵ de 0.01º para evitar que oscile demasiado como ya se ha comentado con anterioridad.

En este caso, se han dejado los parámetros del controlador exactamente igual al experimento sin compensación de zona muerta, por considerarse que los resultados son correctos. Antes de mostrar las gráficas, se vuelve a insitir en el hecho de que estas no comienzan en el instante cero al haber sido recortadas, con el fin de evitar que el tramo en el que la acción de control se mantiene a cero y la referencia en un valor no nulo por no haberse pulsado todavía el botón "POWER" que activa el control, se confunda con un retraso del sistema.

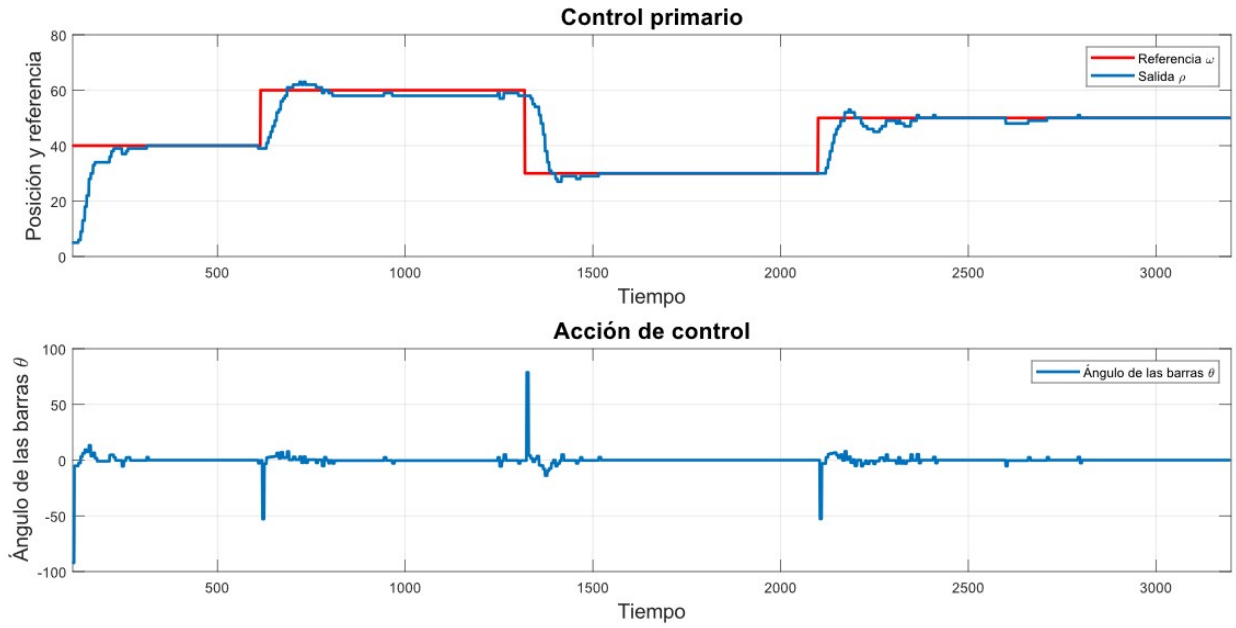

Ilustración 64 Control primario del controlador PD en cascada con PID y compensación de ZM

Efectivamente, se comprueba como el error en régimen permamente se reduce en gran medida, llegándose a anular en varias amplitudes de la referencia y reduciéndose considerablemente en la restante. Vuelve a observarse de nuevo como la respuesta transitoria es diferente en función del salto que se le esté dando en la referencia, siendo en este caso las oscilaciones más mantenidas en la última de las referencias marcadas.

Por otro lado, cabe reseñar como algunas perturbaciones debidas al ruido de la medida, que se ven sobre todo en los valores más altos de las referencias, pueden ser rápidamente recuperados por el control para volverlo a llevar cerca de la misma, como puede verse en el tramo final.

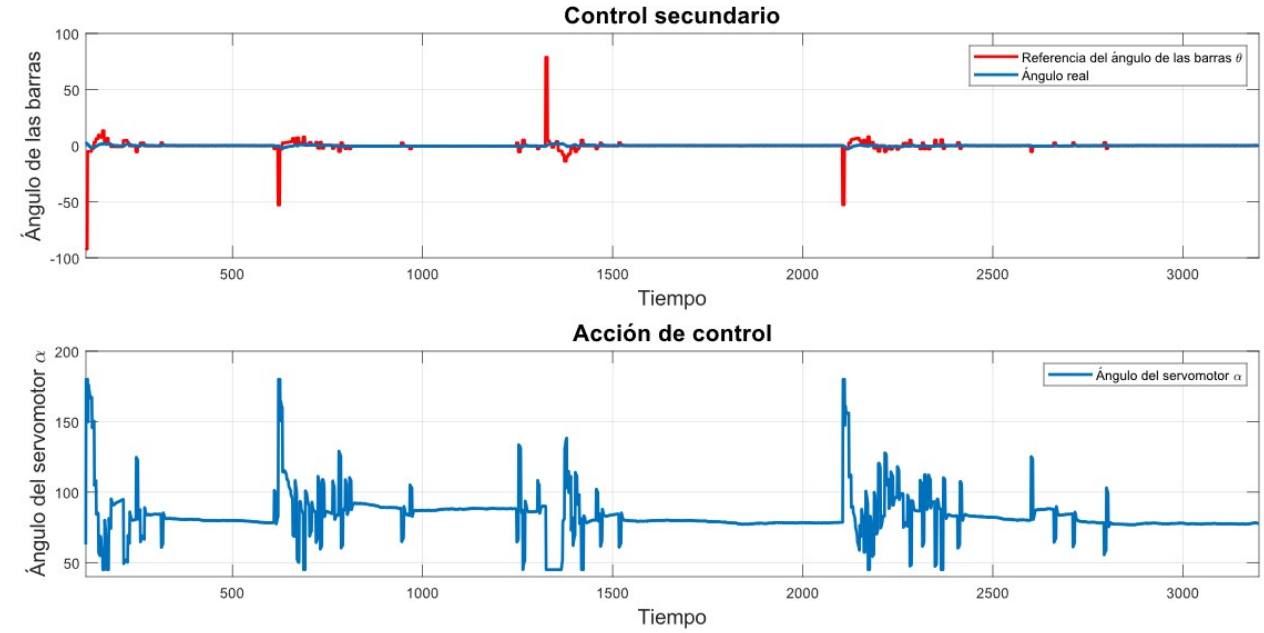

Ilustración 65 Control secundario del controlador PD en cascada con PID y compensación de ZM

En relación con el control secundario, poco o nada reseñable puede decirse, ya que para este la compensación de zona muerta es una modificación de su entrada que no le afecta en su funcionamiento, ya que trabaja con el ángulo θ que le pasen, por lo que la representación gráfica obtenida presenta las mismas características que en el caso sin compensación de zona muerta, esto eso, la visualización del control en bucle cerrado del mismo.

9.4 Control PID en cascada con PID de la planta real

Si bien los controladores anteriores ya han servido para cumplir los objetivos de este Trabajo, se considera oportuno implementar un último controlador en cascada que consiga mejorar los resultados previos. Con el fin principal de eliminar definitivamente el error en régimen permanente, se añade un término integral al control primario de la posición lineal de la bola. Para ello, como es conocido, es necesario ir guardando el valor de la integral del error como la suma de todos los errores anteriores y este es el término que va ponderado por el cociente entre el tiempo de muestreo y el tiempo integral, que deberá ser diseñado y definido previamente.

Para su implementación en el código del programa, además de las preceptivas definiciones e inicializaciones de variables, es necesario modificar la parte en la que se calcula el valor del ángulo de las barras, θ, que se desea obtener, quedando como se muestra en el código adjunto. De nuevo, esta acción de control solo debe calcularse una vez cada cinco que se calcule la del control secundario, para permitirle a este que alcance la referencia en ángulo pedida antes de darle una nueva. Se destaca también el uso de la saturación en la lectura del sensor para evitar errores graves por posibles valores espúreos.

```
/* Segunda tarea: Control primario */
if ((cont % 5) == 0)
{
  if (encendido == 1) {

    LeeSensor(&pos); // Lectura de la entrada.
    pos = pos + 4; //Se ajusta el offset del sensor.

    /* Saturación de la lectura */
    if (pos > 90) pos = 90;
    if (pos < 5) pos = 5;
    /* Cálculo de la acción de control */
    ek = pos - ref;
    Ik1 = Ik1_1 + ek;
    theta = Kp1 * (ek + Tm*Ik1/Ti1 + (Td1/Tm)*(ek-ek_1));
    Ik1_1 = Ik1;
    ek_1 = ek;
  }
}
```

En cuanto a los valores de los parámetros de los controladores, se ha comprobado que los resultados son aceptables usando unos muy parecidos a los del control en cascada anterior, por lo que únicamente son levemente modificados para optimizar el control. Para el tiempo integral, se ha usado un valor bastante elevado para que su efecto no sea demasiado dominante, ya que solo se desea utilizarlo para eliminar el error en régimen permamente, tratando de que esto no provoque que, por los ruidos en las medidas del sensor láser ya comentados con anterioridad, se produzcan efectos indeseados en el control que hagan que el resultado sea peor. En consecuencia, los valores escogidos para el control PID en cascada con otro control PID se resumen en la siguiente tabla:

Tabla 4 Parámetros del control PID en cascada con PID

Parámetro	Control primario	Control secundario
K_p	0.13	7.5
T_d	600	1.5
T_i	1510	1000

Una vez se ha aplicado todo lo anteriormente relacionado al código del programa, que puede verse de forma completa en el Anexo A, se carga en la placa de Arduino el programa y se ejecuta el programa de comunicación de MATLAB para finalmente obtener los siguientes resultados:

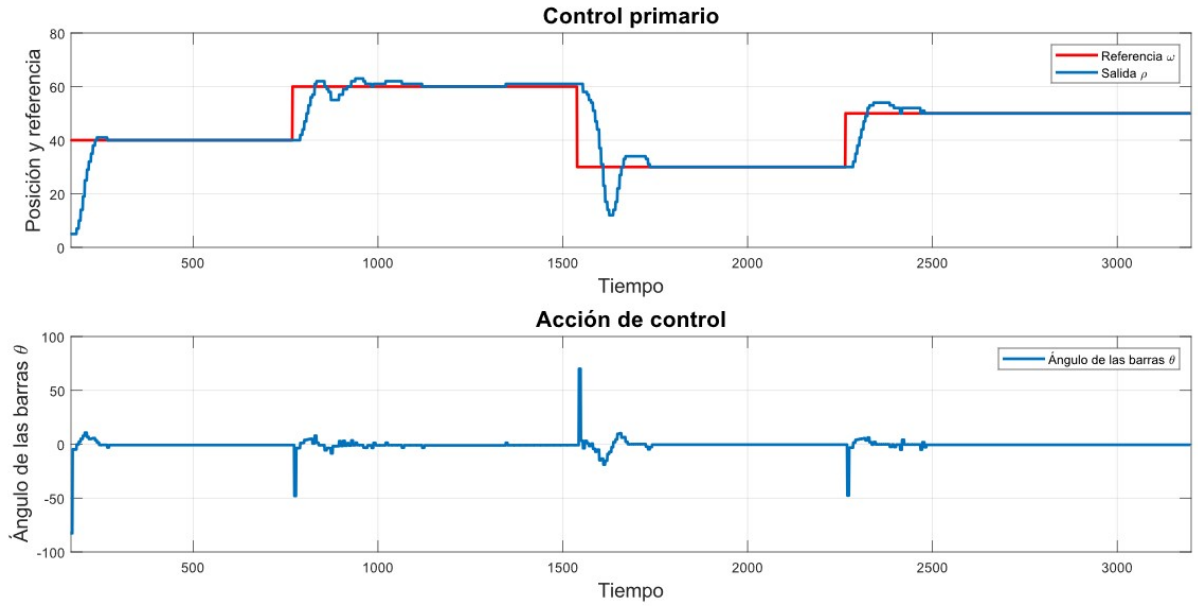

Ilustración 66 Control primario del control PID en cascada con PID

En este caso, se observa como claramente reduce el error en régimen permanente hasta desaparecer prácticamente en todas las referencias dadas. Además, se puede ver como la rapidez del sistema no se ve reducida en tanto y en cuanto el tiempo de subida sigue siendo basante corto.

Por otro lado, se aprecia como, sobre todo al utilizar la referencia de 60 cm, presenta unas leves oscilaciones mantenidas, pero que rápidamente son recuperadas de forma extraordinaria y en ningún caso hacen que tienda a la inestabilización. De nuevo vuelve a ser en el cambio de referencia en sentido negativo cuandose produce una sobreoscilación de mayor amplitud, pero que de igual forma se consigue recuperar hasta establecerse de forma permanente en el "set-point" de 30 cm.

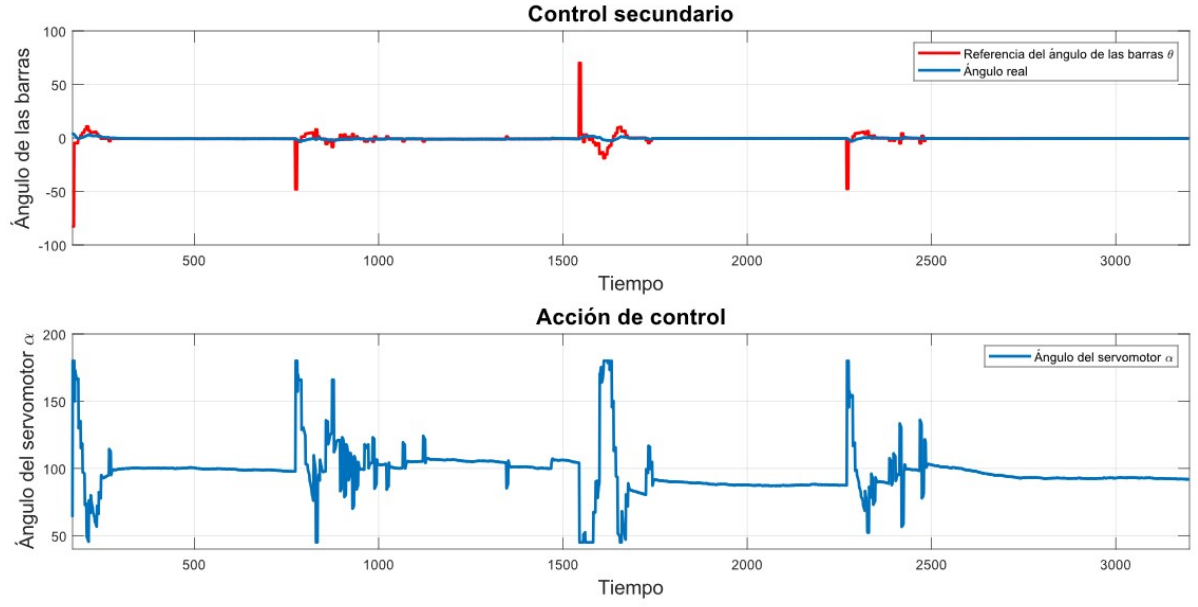

Ilustración 67 Control secundario del control PID en cascada con PID

Si bien el control secundario no se ha visto sustancialmente modificado, se destaca aquí la buena representación del efecto del término integral ya que se aprecia como, aún estando en régimen permanente, si existe algo de error, este se va integrando, hasta lograr modificar la acción de control para alcanzar la posición deseada.

Es bien conocido que en Control Automático el problema principal de controlar una salida de un sistema mediante el uso de una entrada que sea modificable, suele ser dividido en dos subproblemas. Por un lado, el seguimiento de referencia que consiste en alcanzar un valor (fijado por el usuario) de la variable de salida cuando esta no se encuentra en dicha referencia y en la facilidad que tenga el sistema de adaptarse a posibles cambios en la misma que se hagan en tiempo real. El otro subproblema es la regulación, que consiste en mantener el valor de la variable que se tiene (y que es el que se quiere tener por ser igual a la referencia), cuando se producen perturbaciones en el sistema que tiendan a sacarlo de ese equilibrio en el que se encuentra.

Tras haber mostrado el buen funcionamiento de los controladores implementados para resolver el problema de seguimiento de referencia, se muestra a continuación una última gráfica relativa al experimento realizado para comprobar la bondad del controlador PID en cascada con otro PID para solucionar el problema de regulación. Para ello, se ha dejado que se posicione en el valor de la referencia inicial de 30 cm y tras esto, se le ha dado manualmente una perturbación a la bola hacia el lado derecho de las barras, dejándola a continuación libre para que trate de recuperar su posición.

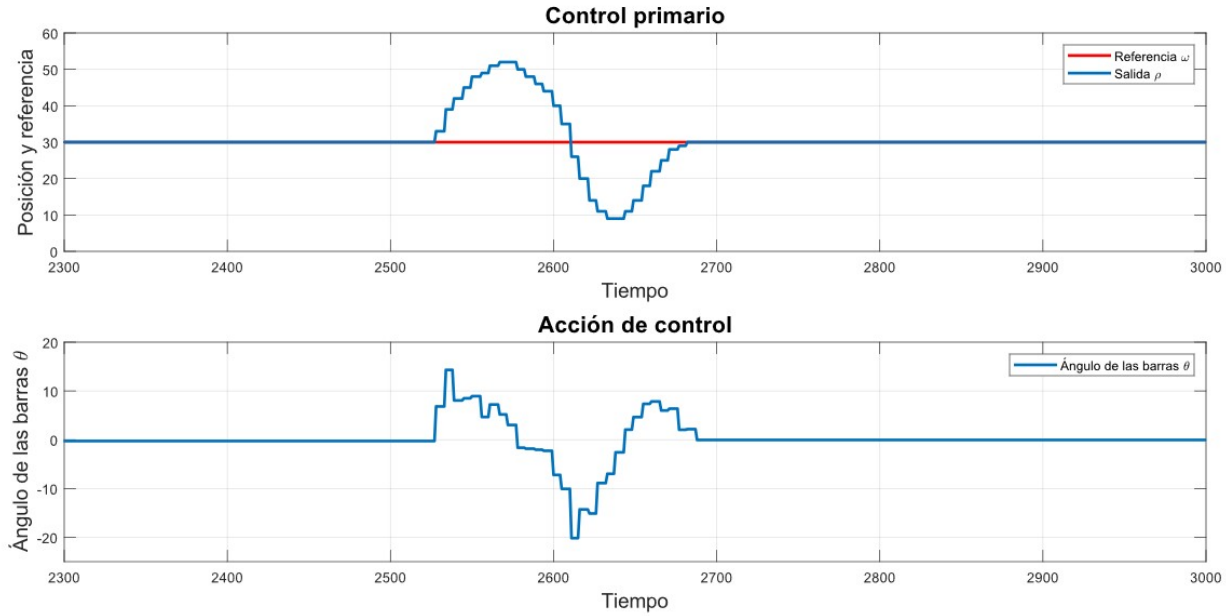

Ilustración 68 Control de la posición en regulación ante perturbaciones

Puede observarse como, efectivamente, la bola se ha movido hacia la derecha hasta pasar de los 50 cm, actuando el controlador en consecuencia para hacer que la acción de control la hiciera volver a su posición inicial. Tras una única sobreoscilación de aproximadamente la misma amplitud que la inicial, la bola se vuelve a posicionar en los 30 cm manteniéndose ahí indefinidamente sin mostrar error en régimen permanente alguno, concluyendo definitivamente que se han obtenido correctas soluciones para el control del sistema "ball and beam" tanto para seguimiento de referencias como para regulación.

10 CONCLUSIONES Y AMPLIACIONES FUTURAS

No pienso nunca en el futuro porque llega muy pronto.

Albert Einstein

TRAS haber realizado a través de todos los capítulos que componen el presente Trabajo, un recorrido que ha querido ser fiel al lógico transcurrir del estudio completo de un sistema real en el ámbito del Control Automático, en este último capítulo, a modo de conclusión, se pretenden presentar todos y cada uno de los pasos realizados con el fin de comprobar la bondad de la realización de los mismos y el cumplimiento de los objetivos propuestos. Además, se plantearán las posibles ampliaciones futuras que por parte del autor se considera que serían las más adecuadas para engrandecer este Trabajo.

10.1 Conclusiones

Como ya se ha podido ver con anterioridad, el Trabajo se divide distintos capítulos que pueden ser agrupados en tres partes bien diferenciadas, las cuales dan como resultado definitivo, el diseño y la implementación del sistema "ball and beam", el análisis del mismo y, finalmente, el control del sistema real que se puede ver en la Ilustración 68. Las tres partes a las que se ha hecho mención son, respectivamente:

- Estudio previo, diseño e implementación.
- Estudio teórico del sistema y simulaciones de controladores para el mismo.
- Control real de la planta.

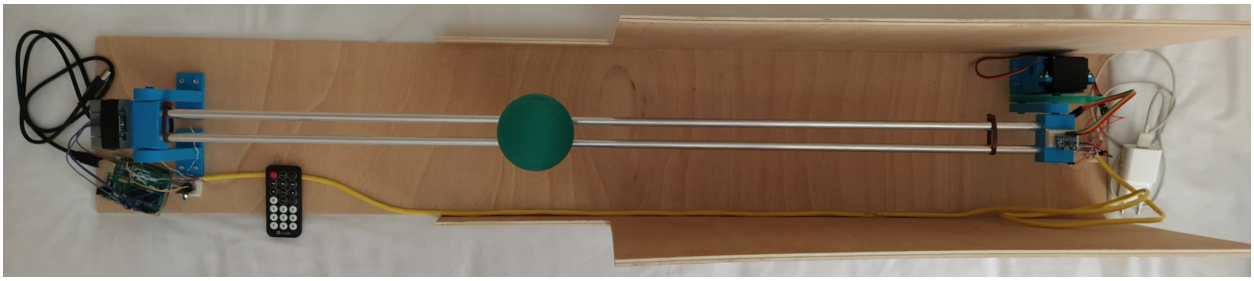

Ilustración 69 Vista en planta del sistema real

En la primera de ellas, se ha logrado entender las distintas opciones que en la bibliografía se encuentran sobre este sistema en concreto, eligiendo la estructura que se consideraba más oportuna. De igual forma, se han diseñado las piezas necesarias para el montaje, imprimiéndolas en 3D y se ha realizado un estudio pormenorizado de las distintas opciones de hardware disponibles en el mercado explicando el cableado que conforma el sistema. Por otra parte, se ha desarrollado el software necesario para la utilización de todos los dispositivos, así como la unión de todos ellos con Arduino y la comunicación con MATLAB para la representación de los resultados.

En la segunda parte, se ha realizado un modelado teórico del sistema mediante dos metodologías distintas de la Mecánica, obteniendo la función de transferencia que lo modela. Se ha hecho un análisis teórico de la estabilidad del mismo y se han simulado distintos controladores tanto en tiempo discreto como en continuo que teóricamente se ha visto que son suficientes para controlar el sistema "ball and beam", además se han probado otros esquemas más completos y modernos. A su vez, como apoyo a todo este análisis se ha ido detallando desde un punto de vista teórico los fundamentos que apuntalan estos conocimientos del ámbito del Control.

Por último, en la tercera parte, se ha aplicado lo anteriormente desarrollado para poder controlar la planta real con los distintos controladores implementados. Se han mostrado las ventajas e inconvenientes de cada uno de ellos mediante el apoyo en una serie de representaciones gráficas obtenidas con la comunicación Arduino-MATLAB ya mencionada.

En definitiva, los objetivos que se han ido marcando en cada uno de los capítulos, así como los generales que inicialmente se establecieron al comenzar este Trabajo pueden darse por completados de forma satisfactoria, dando, en consecuencia, por terminado el presente Trabajo de Fin de Grado.

10.2 Ampliaciones futuras

Como último apartado, se considera oportuno dejar constancia de las posibles ampliaciones futuras que a partir de este Trabajo se podrían hacer por parte de distintos alumnos en próximos proyectos, si bien, por el alcance limitado que tiene un Trabajo de Fin de Grado, no han sido incluidas aquí.

En primer lugar, en relación con la obtención de la posición de la bola mediante la medida de la resistencia interna que forman con la bola, podría profundizarse en su estudio y, basándose en lo que se ha explicado en el Capítulo 3, conseguir implementar este método; si bien implicaría usar una bola metálica de mayor diámetro a la presentada en los Capítulos teóricos si se quiere usar a la vez el sensor láser, sin ser tampoco válida la usada finalmente en el control real por ser de un material plástico.

Por otro lado, el prototipo del sistema "ball and beam" es perfectamente compatible con cualquier tipo de controlador que sea implementable en una placa de Arduino UNO. En consecuencia, la primera propuesta que se hace es la implantación del control predictivo GPC que se desarrolla en la parte de simulaciones, directamente sobre el sistema real, usando todos los resultados que en dicho Capítulo se muestran. Además de esto, podrían utilizarse cualquier otro tipo de controladores más modernos y avanzados que el alumno considerara oportuno. De igual forma, podría hacerse un estudio pormenorizado de distintos métodos de diseño de controladores lineales (PID) de los muchos que se encuentran en la bibliografía e implementarlos en el sistema real, mostrando las ventajas y desventajas de cada uno de ellos.

Finalmente, en caso de considerarse oportuno, podrían hacerse también una serie de enunciados de prácticas de laboratorio para distintos cursos de los Grados impartidos en la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla que tengan como objetivo el uso de este sistema "ball and beam" por parte de los alumnos para que estos interactúen con sistemas reales lo antes posible y puedan comprender que los conocimientos que adquieren en las distintas asignaturas relacionadas con el Control Automático en sus titulaciones tienen directas aplicaciones prácticas.

ANEXO A: CÓDIGOS DE ARDUINO

DE los distintos controladores implementados, se muestra a continuación el más completo de todos, esto es, el que se compone de dos PID en cascada. Cabe destacar que en el caso del PD en cascada con PID solo es necesario cambiar la línea del controlador primario y en el caso del PD basta con eliminar el control secundario y sustituirlo por la relación entre el ángulo de las barras y el del servomotor, tal y como se detalla en el Capítulo 9. De igual forma para la inclusión de la zona muerta en cualquiera de los controladores, solo hay que incluir la línea que se explicó en el capítulo anteriormente citado. Se considera, por tanto, más que suficiente la inclusión en este Anexo únicamente del código de este controlador, con el objetivo de evitar caer en repetición si se adjuntaran todos los demás, por estar ya explicados las diferencias que los distinguen.

```
/***** CONTROL DE SISTEMA BALL AND BEAM *****/

/* ----- Bibliotecas ----- */
#include "IRremote.h" //Control remoto IR.
#include "Servo.h"  //Control del servomotor.
#include "Wire.h" //Control de la IMU.
#include "VL53L1X.h" //Control del sensor láser.

/* ----- Parámetros ----- */
#define MPU 0x68 // Dirección I2C de la IMU.
#define A_R 16384.0 //Ratio de conversión para pasar de valores del Acelerómetro
a ángulos (32768/2).
#define G_R 131.0 //Ratio de conversión para pasar de valores del Giroscopio a
ángulos (32768/250).
#define RAD_A_DEG = 57.295779 //Conversión de radianes a grados (180/pi).
#define SENSOR 0x29 //Dirección I2C del sensor láser.

/* ----- Asignación de pines ----- */
const int PIN_ReceptorIR = 11;  //Pin al que se conecta el Receptor IR.
const int PIN_Servo = 9; //Pin al que se conecta la línea de datos del servo.

/* ----- Definición de objetos ----- */
IRrecv ReceptIR(PIN_ReceptorIR); //Objeto del receptor IR.
decode_results LecturaReceptIR; //Objeto de decodificación de resultados del
receptor IR.
Servo servo;  //Objeto del servomotor.
VL53L1X sensor; //Objeto del sensor láser.

/* ----- Variables globales ----- */
//Controlador primario, externo o para el proceso más lento (PID).
float Kp1 = 0.13;
float Td1 = 0.6*1000; //ms
float Ti1 = 1.51*1000; //ms

//Controlador secundario, interno o para el proceso más rápido (PID).
float Kp2 = 7.5;
float Td2 = 0.0015*1000; //ms
float Ti2 = 1*1000; //ms

//Tiempo de muestreo en ms.
const unsigned long Tm = 35;
```

```
//Temporización.
unsigned long antes;
unsigned long despues;

//Filtro para el sensor láser.
float distancia_filt = 0;
float distancia_filtrada_ant = 0;
float alpha Sensor = 0.75;

//Valores RAW de la IMU (MPU-6050 da los valores en enteros de 16 bits).
int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;

//Ángulos de la IMU (solo lo necesario para obtener el Roll).
float Acc;
float Gy;
float Angle;

//Filtro para la IMU.
float angle_filtrada = 0;
float angle_filtrada_ant = 0;
float alpha_IMU = 0.5;

//Cálculos internos de la IMU.
long tiempo_prev;
float dt;

/* ----- Prototipos de funciones ----- */
String TraduceIR(void); //Función del receptor IR que traduce la tecla recibida.
void LeeIMU(float *Roll); //Función para la lectura del ángulo de la IMU.
void LeeSensor(int *pos); //Función para la lectura de la posición del sensor.

/* ----- Bloque de inicialización ----- */
void setup() {

  Serial.begin(9600); //Establecer comunicación con el Arduino en la línea serie
a 9600 baudios.

  ReceptIR.enableIRIn();//Habilitar el receptor IR.

  servo.attach(PIN_Servo); //Para activar el servo, se conecta el al pin
predefinido.
  servo.write(80); //Se posiciona el servo en la posición  inicial, donde
aproximadamente las barras están rectas.
  delay(100); //Se espera para asegurar que llega.

  Wire.begin(); //Para activar el sensor láser, primero se inicia la librería
//Wire y se conecta el Arduino al bus I2C, que permite trabajar con los pines
//genéricos SDA (para datos) y SCL (para reloj).
  Wire.beginTransmission(SENSOR); //Se conecta el Arduino al bus como maestro
para la dirección del sensor láser.
  Wire.setClock(400000); //Se marca una frecuencia de 400 kHz para el bus I2C.
  sensor.setTimeout(100); //Se establece un sobretiempo de 100 ms.
  sensor.init(); //Se inicia el sensor, se esperan 10 ms y se comprueba si está
activado. En caso de no estarlo, se muestra un mensaje de error.
  delay(10);
  if (!sensor.init())
  {
    Serial.println("¡Fallo al detectar e inicializar el sensor!");
    while (1);
  }
  delay(10);
```

```
  sensor.setDistanceMode(VL53L1X::Short); //Se establece el modo de distancia
corta porque se trabajará con medidas menores a un metro.
  sensor.setMeasurementTimingBudget(35000); //Se marca el tiempo de muestreo.
  Wire.endTransmission(true); //Se finaliza la transmisión al dispositivo
esclavo (sensor láser) que fue iniciada por beginTransmission().

  Wire.beginTransmission(MPU); //Para activar la IMU, se conecta el Arduino al
bus I2C como maestro para la dirección de la IMU.
  Wire.write(0x6B); //Se escriben los datos en un dispositivo esclavo en
respuesta a una petición de un maestro.
  Wire.write(0); //Se despierta la IMU.
  Wire.endTransmission(true); //Se finaliza la transmisión al dispositivo
esclavo (IMU) que fue iniciada por beginTransmission().

    delay(10);
}

/* ----- Bucle principal (Ejecutivo cíclico) ----- */
void loop() {

  /* Definición de variables */
  String valores; //Para el envío de datos a MATLAB para su representación.

  static int encendido = 0;  //Bandera para que el control funcione o no.
  static int lee_ref = 0;  //Bandera para leer una nueva referencia.
  static int ref = 40; //Referencia para la posición lineal de la bola.
  String Boton; //Botón recibido por Receptor IR.

  static float theta = 0; //Acción de control para el controlador primario
//(ángulo que forman las barras con la horizontal).
  int pos; //Salida del controlador primario leida por el sensor láser (posición
//lineal de la bola).

  //Variables del controlador primario(PID_1).
  float Ik1;
  static float Ik1_1 = 0;
  static float ek_1 = 0;
  float ek = 0;

  static float alpha = 80; //Acción de control para el controlador secundario
//(ángulo del servomotor).
  float Roll; //Salida del controlador secundario leida por la IMU (ángulo real
//que forman las barras con la horizontal).

  //Variables del controlador secundario(PID_2).
  float Ik;
  static float Ik_1 = 0;
  float ek2 = 0;
  static float ek2_1 = 0;

  //Variables de control del ejecutivo cíclico.
  static int cont = 0;
  static unsigned long T = Tm;

  /* Control del tiempo de muestreo */
  //En cada ejecución del bucle, se espera hasta que se cumpla el tiempo de
//muestreo,se mide el tiempo real para actualizarlo al final y se aumenta un
//contador para saber que tarea corresponde hacer.
  delay(T);
  antes = millis();
  cont ++;
```

```
  /* Primera tarea: Control secundario*/
  //Esta tarea debe ser lo más rápida posible, por lo que se realiza en cada
//iteración del bucle, siempre y cuando se haya pulsado el botón "POWER" del
//mandoa distancia.
  if (encendido == 1) {
    LeeIMU(&Roll); //Para leer la posición angular real de las barras, se llama
a la función LeeIMU y se copia el valor del ángulo en la variable que se pasa.

    ek2 = Roll - theta; //Se calcula el error.
    Ik = Ik_1 + ek2; //Se recalcula el error acumulado.
    alpha = Kp2 * (ek2 + Tm*Ik/Ti2 + (Td2/Tm)*(ek2-ek2_1)); //Se calcula la
//acción de control usando la estructura de un controlador PID.
    Ik_1 = Ik; //Se actualizan las variables de los errores.
    ek2_1 = ek2;

    //Se satura la acción de control para que no use valores no permitidos o que
//afecten a la física del sistema.
    if (alpha > 180) alpha = 180;
    if (alpha < 45)  alpha = 45;

    servo.write(alpha); //Se manda la acción de control al servomotor.
  }
  else { //Si está apagado el control, se resetean los errores.
    Ik = 0;
    Ik_1 = 0;
    servo.write(80); //Se manda el valor para que las barras estén
aproximadamente rectas.
  }

/* Segunda tarea: Control primario */
//La segunda tarea debe realizarse al menos 5 veces más lenta que la primera,
//para que el control en cascada funcione correctamente, siempre que se haya
//pulsado el botón "POWER".
  if ((cont % 5) == 0)
  {
    if (encendido == 1) {
    LeeSensor(&pos); //Para leer la posición lineal de la bola, se llama a la
//función LeeSensor y se copia el valor del ángulo en la variable que se pasa.
      pos = pos + 4; //Se suma el valor del radio de la bola para posicionar su
//centro.
      if (pos > 90) pos = 90; //Se satura la lectura en los valores que se sabe
//que no sobrepasará.
      if (pos < 5) pos = 5;

      ek = pos - ref; //Se calcula el error.
      Ik1 = Ik1_1 + ek; //Se recalcula el error acumulado.
      theta = Kp1 * (ek + Tm*Ik1/Ti1 + (Td1/Tm)*(ek-ek_1)); //Se calcula la
//acción de control usando la estructura de un controlador PID.
      Ik1_1 = Ik1; //Se actualizan las variables de los errores.
      ek_1 = ek;

      //Cabe señalar que esta salida no se escribe en ningún actuador, sino que
//será usada en el siguiente tiempo de muestreo por el controlador secudario.
      }
  }

  /* Tercera tarea: Receptor IR */
  //Debido a la velocidad del Receptor IR, es suficiente con realizarla una de
//cada 10 iteraciones.
  if (cont > 10)
  {
    if (ReceptIR.decode(&LecturaReceptIR))  // Si se ha recibido algo...
```

```
    {
      Boton = TraduceIR(); //Se lee la tecla pulsada.
      ReceptIR.resume();  //Se continúa leyendo.
    }

    //Se define qué hacer en función de la tecla pulsada.
    if (Boton == "POWER" && encendido == 0)
    {
      encendido = 1; //Se activan los controladores.
    }
    else if (Boton == "POWER" && encendido == 1)
    {
      encendido = 0; //Se desactivan los controladores.
    }
    else if (Boton == "FUNC/STOP" && lee_ref == 0)
    {
      lee_ref = 1; //Se prepara para leer una nueva referencia.
    }
    else if (lee_ref == 1 && (Boton == "1" || Boton == "2" || Boton == "3" ||
Boton == "4" || Boton == "5" || Boton == "6" || Boton == "7" || Boton == "8" ||
Boton == "9") )
    {
      lee_ref = 0; //Deja de leer una nueva referencia.
      ref = 10*Boton.toInt(); //Se actualiza el valor de la referencia como diez
//veces el valor pulsado.
    }

    //Se actualiza el tiempo de espera y se reinicia el contador, porque ya han
//pasado diez iteraciones.
    despues = millis();
    cont = 0;
    T = Tm - (despues - antes);
  }

  //Se envían los resultados obtenidos por el puerto serie.
  valores = String(pos) + "," + String(ref) + "," + String(Roll) + "," +
String(theta) + "," + String(alpha);
  Serial.println(valores);
}

/* ----- Funciones ----- */
/* Traductor de Receptor IR */
// Esta función analiza el valor almacenado en LecturaRecepIR y lo sustituye por
su valor.
String TraduceIR(void)
{
  String Boton;

  switch (LecturaReceptIR.value)
  {
    case 0xFFA25D: Boton = "POWER"; break;
    case 0xFFE21D: Boton = "FUNC/STOP"; break;
    case 0xFF629D: Boton = "VOL+"; break;
    case 0xFF22DD: Boton = "FAST BACK";    break;
    case 0xFF02FD: Boton = "PAUSE";    break;
    case 0xFFC23D: Boton = "FAST FORWARD";   break;
    case 0xFFE01F: Boton = "DOWN";    break;
    case 0xFFA857: Boton = "VOL-";    break;
    case 0xFF906F: Boton = "UP";    break;
    case 0xFF9867: Boton = "EQ";    break;
    case 0xFFB04F: Boton = "ST/REPT";    break;
    case 0xFF6897: Boton = "0";    break;
```

```
    case 0xFF30CF: Boton = "1";    break;
    case 0xFF18E7: Boton = "2";    break;
    case 0xFF7A85: Boton = "3";    break;
    case 0xFF10EF: Boton = "4";    break;
    case 0xFF38C7: Boton = "5";    break;
    case 0xFF5AA5: Boton = "6";    break;
    case 0xFF42BD: Boton = "7";    break;
    case 0xFF4AB5: Boton = "8";    break;
    case 0xFF52AD: Boton = "9";    break;
    case 0xFFFFFFFF: Boton = " REPEAT"  ; break;

    default:
      Boton = " other button   ";
  }

  return Boton;
}

/* Lectura de ángulos de la IMU */
//Esta función permite obtener el ángulo Roll de la IMU.
void LeeIMU(float *Roll)
{
  //Se leen los valores del Acelerómetro.
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Se pide el registro 0x3B, que corresponde al AcX.
  Wire.endTransmission(false); //Se deja activa la comunicación.
  Wire.requestFrom(MPU, 6, true); //A partir del 0x3B, se piden 6 registros.
  AcX = Wire.read() << 8 | Wire.read(); //Cada valor ocupa 2 registros.
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();

  //A partir de los valores del acelerómetro, se calcula el ángulo con la
//fórmula de la tangente.
  Acc = atan((AcY/A_R)/sqrt(pow((AcX / A_R),2) + pow((AcZ/A_R),2)))*RAD_TO_DEG;

  //Se leen los valores del Giroscopio.
  Wire.beginTransmission(MPU);
  Wire.write(0x43); //Se pide el registro 0x43, que corresponde al GyX.
  Wire.endTransmission(false); //Se deja activa la comunicación.
  Wire.requestFrom(MPU, 6, true); //A partir del 0x43, se piden 6 registros.
  GyX = Wire.read() << 8 | Wire.read(); //Cada valor ocupa 2 registros.
  GyY = Wire.read() << 8 | Wire.read();
  GyZ = Wire.read() << 8 | Wire.read();

  //A partir de los valores del giroscopio, se calcula el ángulo.
  Gy[0] = GyX / G_R;

  //Se calcula en tiempo real el incremento de tiempo que se va a utilizar.
  dt = (millis() - tiempo_prev) / 1000.0;
  tiempo_prev = millis();

  //Se aplica el filtro complementario para unir las medidas de los sensores.
  Angle = 0.98 * (Angle + Gy * dt) + 0.02 * Acc[0];

  //Se aplica el segundo filtro para mejorar las medidas.
  angle_filtrada = alpha_IMU * Angle + (1 - alpha_IMU) * angle_filtrada_ant;
  angle_filtrada_ant = angle_filtrada;

  //Se devuelve el valor del ángulo.
  *Roll = angle_filtrada;
}
/* Lectura de posición del sensor */
```

```
//Esta función permite obtener la posición en cm que mide el sensor.
void LeeSensor(int *pos){
  //Se definen las variables y la caracterización del sensor.
  int distancia, i;
  int A[2][76] = {{ 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80},
                  { 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
22, 23, 24, 25, 26, 27, 29, 31, 32, 33, 35, 37, 38, 41, 42, 43, 45, 47, 49, 51,
54, 56, 58, 60, 62, 63, 65, 66, 67, 68, 70, 72, 73, 74, 75, 76, 77, 78, 79, 80,
80, 81, 81, 81, 82, 82, 83, 83, 84, 85, 85, 85, 86, 86, 87, 87, 87, 88, 88,89}};

  //Se inicia la lectura, y se guarda el resultado.
  Wire.beginTransmission(SENSOR);
  sensor.startContinuous(35);
  sensor.read();
  sensor.stopContinuous();
  Wire.endTransmission(true);
  distancia = sensor.ranging_data.range_mm;

  //Se pasa de mm a cm.
  distancia = distancia/10;

  //Se corrige la medida usando, o bien la recta de mejor ajuste a la
caracterización del sensor (descomentando la siguiente línea), o bien el método
de la "lookuptable" (tal y como está).
  //distancia = (1.8526 + distancia)/1.2705;
  for(i=0;i<75;i++){
    if(distancia>=A[1][i] && distancia<=A[1][i+1]){
      distancia=A[0][i]+((A[0][i+1]-A[0][i])/(A[1][i+1]-A[1][i]))*(distancia-
A[1][i]);
    }
  }

  //Se aplica un filtrado para mejorar las medidas.
  distancia_filt=alpha_Sensor*distancia+(1-alpha_Sensor)*distancia_filtrada_ant;
  distancia_filtrada_ant = distancia_filt;

  //Se devuelve la posición.
  *pos = distancia_filt;

}
```

REFERENCIAS

[1] F. Salas, «Breve historia del Control Automático», Escuela Técnica Superior de Ingeniería, Universidad de Sevilla.

[2] A. Taifour Ali et al, «Design and implementation of ball and beam system using PID controller», Sudan University of Science and Technology.

[3] W. Yu, «Nonlinear PD regulation for ball and beam system». International Journal of Electric Engineering Education, 46/1.

[4] L. Ruano Pérez, «Diseño e integración de un sistema de realimentación para una maqueta bola y barra e implementación de estrategias de control», TFG, Universidad Politécnica de Madrid, 2018.

[5] A. E. Puglesi et al, «Equipo educativo para la enseñanza de la mecatrónica, sistema de bola y barra», Revista Iberoamericana de Ingeniería Mecánica, 2011.

[6] J. Hauser et al, «Nonlinear control via approximate input-output linealization: the ball and beam example», IEEE Transactions on Automatic Control, 1992.

[7] https://www.freecadweb.org/downloads.php

[8] http://www.iearobotics.com/wiki/index.php?title=Dise%C3%B1o_de_piezas_con_Freecad

[9] https://grabcad.com/

[10] «Manual de impresión. Manual de usuario para impresora 3D PRUSA i3 MK3S».

[11] L. Esteve Ros, «Diseño de carcasa para impresora 3D modelo PRUSA i3», Universidad Politécnica de Valencia, 2016.

[12] «Pinout diagram of Arduino UNO», Arduino offcial store.

[13] S. Jennings, «Motores paso a paso», Informador Técnico 65, 47-58.

[14] «Datasheet of Servomotor MG996R».

[15] «Datasheet of ultrasonic sensor HC-SR04», 2012.

[16] S. Leibson, «Principios básicos sobre la medición a distancia y el reconocimiento de gestos al usar los sensores ToF», Digi-Key Electronics, 2018.

[17] «Datasheet of VL53L1X sensor».

[18] https://fritzing.org/home/

[19] https://www.arduino.cc/en/Main/Software

[20] https://www.arduino.cc/en/reference/libraries

[21] https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

[22] https://github.com/pololu/vl53l1x-arduino

[23] https://github.com/z3t0/Arduino-IRremote

[24] K. Ogata, «Ingeniería de Control Moderna». Ed. Pearson, 2010.

[25] M. Tosacano, «Mecánica. Ingenieros Industriales», 2017.

[26] S. Ortiz, «Sintonización de un controlador PID basado en un algoritmo heurístico para el control de un Ball and Beam», Universidad Autónoma de Querétaro, 2014.

[27] E. Drake, «Fundamentos Físicos de la Ingeniería. Mecánica», 2016.

[28] P. Bolzern et al, «Fundamentos del Control Automático». Ed. Mc Graw Hill, 2009.

[29] D. Muñoz de la Peña et al, «Fundamentos del Control Automático. Primera Parte», 2015.

[30] Diapositivas del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga.

[31] K.Ogata, «Sistemas de Control en Tiempo Discreto», Ed. Prentice Hall, 1996.

[32] E. Camacho et al, «Model Predictive Control», Ed. Springer, 2007.

[33] D. Rodríguez, «Diapositivas de la asignatura Complementos de Control. 4º. GITI», Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, 2019.

[34] D. Rodríguez y T. Álamo, «Diapositivas de la asignatura Ingeniería de Control. 3º. GITI», Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, 2018.

[35] R. Aranda, «Control de motores de corriente continua con compensación de zona muerta», TFG, Univerdad de Almería, 2018.