Proyecto Fin de Carrera
Ingeniería Industrial

Ball and Beam. Equipo de Prácticas de bajo Coste Basado en Arduino

Autor: Jon Alexander Ortiz Díaz

Tutor: Ignacio Alvarado Aldea

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

Sevilla, 2017

Proyecto Fin de Carrera
Ingeniería Industrial

Ball and Beam. Equipo de Prácticas de bajo Coste Basado en Arduino

Autor:

Jon Alexander Ortiz Díaz

Tutor:

Ignacio Alvarado Aldea

Profesor Contratado Doctor

Dep. de Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera: Ball and Beam. Equipo de Prácticas de bajo Coste Basado en Arduino

Autor: Jon Alexander Ortiz Díaz
Tutor: Ignacio Alvarado Aldea

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

Agradecimientos

A mi tutor Ignacio por ser un ejemplo como docente. Por ser exigente desde la cercanía y por todo lo que he aprendido estos últimos meses gracias a ti.

A mis padres por vuestro apoyo constante y paciencia infinita. Por animarme a seguir en los peores momentos y sobre todo, por creer en mí siempre, incluso cuando no era fácil hacerlo.

A Yanire por ejercer de hermana mayor aunque tengas menos edad que yo. Por tenderme una mano siempre que la he necesitado. Por ser mi socia, confidente y amiga.

A Paloma por estar a mi lado cada día. Por todo lo que hemos vivido juntos y lo que nos queda, pero sobre todo por hacerme feliz.

Jon Alexander Ortiz Díaz
Sevilla, 2017

Resumen

El objeto del proyecto es el de crear un equipo de prácticas de control, de bajo coste, basado en la creación de un sistema “Ball and Beam” gobernado gracias al uso de un microcontrolador Arduino.

La elección de este sistema se debe a que se trata de un importante modelo de laboratorio para enseñar ingeniería de control y sistemas. Es muy popular al tratarse de un sistema simple y fácil de entender que puede ser utilizado para estudiar muchos de los métodos clásicos y modernos de diseño de controladores.

Índice

Índice de Figuras

1. Objeto del proyecto

El objeto del proyecto es el de crear un equipo de prácticas de control basado en la creación de un sistema "Ball and Beam" gobernado mediante un control PID.

La elección de este sistema se debe a que se trata de un importante modelo de laboratorio para enseñar ingeniería de control y sistemas. Es muy popular al tratarse de un sistema simple y fácil de entender que puede ser utilizado para estudiar muchos de los métodos clásicos y modernos de diseño de controladores.

Figura 1-1. Ball and beam genérico

El objetivo de control es regular automáticamente la posición de la bola en la barra modificando el ángulo de la misma.

En términos de ingeniería de control, el sistema es inestable en lazo abierto porque la salida del sistema (la posición de la bola) puede incrementarse sin límite como respuesta a una entrada constante (el ángulo de inclinación de la barra).

Por tanto, un controlador tiene que ser empleado para mantener la bola en una posición deseada en la barra.

1.1 Objetivos

En el proceso de diseño y construcción de este equipo de prácticas se va a intentar cumplir con una serie de objetivos:

- **Facilidad de reproducción**: Se procurará que las piezas que forman la estructura sean sencillas de imprimir y que los materiales y componentes utilizados se puedan encontrar sin dificultad.
- **Bajo coste:** Se intentará incurrir en el menor coste posible sin llegar a comprometer la fiabilidad, precisión y calidad del sistema.
- **Hacer uso de software y hardware libre:** Entendiéndose como tal aquel que da a los usuarios la libertar de poder ejecutarlo, copiarlo y distribuirlo, estudiarlo, cambiarlo y mejorarlo sin tener que pedir permisos al desarrollador original ni a ninguna otra entidad específica. No es sinónimo de que sea gratis ya que se puede o no cobrar por las copias y modificaciones. Esta decisión se basa en la creencia personal de que el conocimiento en la medida de lo posible debe ser patrimonio de la humanidad.

2. Planta real

2.1 Descripción de la planta

El sistema consiste en una bola que puede rodar libremente sobre un par de barras paralelas, las cuales pueden ser inclinadas mediante un mecanismo de biela-manivela. El movimiento de la manivela es accionado por un servomotor mientras que la posición de la bola en la barra es registrada mediante dos sensores ultrasónicos.

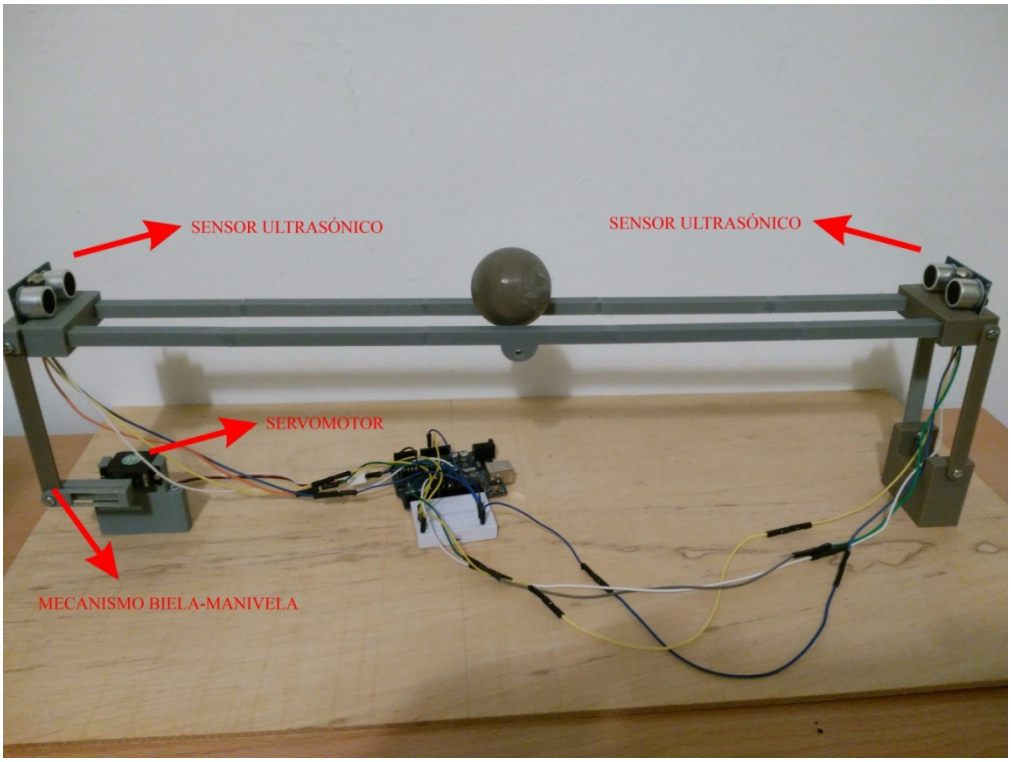


Figura 2-1. Prototipo ball and beam

Para realizar una descripción pormenorizada de la planta se debe hablar de los siguientes aspectos:

- Componentes electrónicos utilizados y programación de los mismos
- Estructura del Ball and Beam
- Guía de montaje

2.2 Componentes electrónicos

Para el correcto funcionamiento del equipo de prácticas se necesita al menos:

- Un controlador que genere la señal que gobierna al actuador.
- Un sensor que determine el estado del sistema.
- Un actuador, que modifique al sistema de manera controlada.

2.2.1 Controlador

Para la realización de este proyecto se ha elegido la placa Arduino Uno Rev3, el cual tiene como microcontrolador el modelo ATmega328P de la marca Atmel y que tiene una arquitectura de tipo AVR. La placa posee 13 entradas/salidas digitales (6 de ellas pueden utilizarse como salidas de tipo PWM), 6 entradas analógicas, conector USB, clavija hembra tipo Jack, conector ICSP y botón de reset.

2.2.1.1 Características placa Arduino UNO

Voltaje de funcionamiento: 5 V

Voltaje de entrada recomendado: 7 – 12 V. Este voltaje de entrada ofrecido por la fuente externa siempre es rebajado a los 5 V de trabajo mediante un circuito regulador de tensión que viene incorporado en la placa.

Voltaje de entrada mínimo – máximo: 6 – 20 V. Aunque el circuito regulador de tensión debe rebajar a los 5V, todo el voltaje sobrante se convertirá en calor lo que además de ineficiente podría llegar a dañar la placa. Por debajo de los 6V, podría resultar insuficiente para la alimentación.

Corriente continúa por pin E/S: 40 mA. Hay que tener en cuenta que la placa agrupa los pines digitales de tal forma que tan solo pueden aportar 100 mA a la vez al conjunto de los pines nº 0,1,2,3 y 4 y 100 mA más el resto de los pines (del 5 al 13). Es decir la placa Arduino puede proporcionar como mucho 200 mA en un instante.

Memoria Flash (ATmega328P): memoria persistente donde se almacena permanentemente el programa que ejecuta el microcontrolador. En el caso del ATmega328P tiene una capacidad de **32 KB**, de los cuales 512 bytes están ya ocupados por el "bootloader" o "gestor de arranque".

Memoria SRAM (ATmega238P) : memoria volátil donde se alojan los datos que en ese instante el programa necesita crear o manipular para su correcto funcionamiento. Estos datos suelen tener un contenido variable y cada uno es de un tipo concreto. Independientemente del tipo de dato, su valor siempre será eliminado cuando se deje de alimentar eléctricamente al microcontrolador. Esta memoria tiene una capacidad de **2KB.**

Memoria EEPROM (ATmega328P): memoria persistente donde se almacenan los datos que se desea que permanezcan grabados una vez apagado el microcontrolador para poderlos usar posteriormente en siguientes reinicios. En el caso del ATmega328P tiene una capacidad de **1KB.**

Registros del microcontrolador (ATmega328P): Espacios de memoria existentes dentro de la propia CPU de un microcontrolador. Sirven para albergar los datos necesarios para la ejecución de las instrucciones previstas próximamente; para almacenar temporalmente los resultados de las instrucciones recientemente ejecutadas y sirven además para alojar las propias instrucciones que en ese mismo momento están ejecutándose. En este caso tiene una capacidad de **8 bits.**

Chip ATmega16U2: Se trata de un elemento "traductor" que facilita al chip ATmega328P la manipulación de la información transferida por USB sin que este tenga que conocer los entresijos del protocolo USB.

Reloj: Para controlar la frecuencia de trabajo del microcontrolador, la placa Arduino posee un reloj, el cual funciona a una frecuencia de **16MHz**. Este reloj que posee la placa es un resonador cerámico con una precisión del **0,5%.**

2.2.1.2 Pines de Arduino y funciones asociadas

Figura 2-2. Arduino UNO

"Vin": Se puede utilizar para dos cosas diferentes: si la placa está conectada mediante la clavija a alguna fuente externa, podemos conectar a este pin cualquier componente electrónico para alimentarlo directamente con el nivel de voltaje que esté aportando la fuente en ese momento sin pasar por el regulador de tensión de la placa. Si está alimentada mediante USB entonces este pin aportará 5 V. En cualquier caso la intensidad máxima de corriente es de 40 mA. También puede usarse para alimentar la placa directamente desde alguna fuente de alimentación externa sin utilizar ni la clavija ni el cable USB.

"5V": Tanto si la placa está alimentada mediante cable USB como si está alimentada por una fuente externa, podemos conectar a este pin cualquier componente para que reciba 5V. También puede usarse para alimentar la propia placa desde una fuente de alimentación sin utilizar ni la clavija ni el cable USB.

"3.3V": Este pin ofrece un voltaje de 3.3 voltios. Este voltaje se obtiene a partir del recibido indistintamente a través del cable USB o de la clavija tipo Jack y está regulado por un circuito específico incorporado a la placa: el LP2985. La corriente máxima generada es de 50 mA. En este caso no podemos conectar ninguna fuente externa aquí porque el voltaje es demasiado limitado para poder alimentar la placa.

"GND": Pines conectados a tierra. Es muy importante que todos los componentes de nuestros circuitos compartan una tierra común como referencia.

"IOREF": Este pin es una duplicación regulada del pin "Vin". Su función es indicar a las placas supletorias conectadas a nuestra placa Arduino el voltaje al que trabajan los pines de entrada/salida de esta, para que las placas supletorias se adapten automáticamente a este voltaje de trabajo.

"AREF": Ofrece un voltaje de referencia para poder aumentar la precisión de las entradas analógicas.

"RESET": Se emplea para reiniciar o detener el microcontrolador.

Entradas y salidas digitales: La placa Arduino dispone de 14 pines de entrada o salida (según convenga) digitales, numeradas desde la 0 hasta la 13. Es aquí donde se conectarán sensores y actuadores y cualquier otro componente que necesite comunicarse con la placa de alguna manera.

Todos estos pines funcionan a 5V, pueden proveer o recibir un máximo de 40 mA y disponen una resistencia "pull-up" interna de entre 20 KΩ y 50 KΩ que inicialmente está desconectada.

Salidas analógicas (PWM): Varios de los pines digitales tienen la posibilidad de generar señales de tipo PWM. Estos pines son el 3, 5, 6, 9,10 y 11 que pueden "simular" un comportamiento analógico.

Las siglas PWM vienen de "Pulse Width Modulation" (Modulación de Ancho de Pulso). Lo que hace este tipo de señal es emitir una señal cuadrada formada por pulsos de frecuencia constante. Al variar la duración de estos pulsos, estaremos variando proporcionalmente la tensión promedio resultante.

Cada uno de los pines tiene una resolución de 8 bits. Por lo tanto, obtendremos un máximo de 2^8 (256) valores diferentes posibles.

Los pines PWM son controlados por tres temporizadores diferentes que mantienen la frecuencia constante de los pulsos emitidos.

Entradas analógicas: La placa Arduino posee 6 entradas analógicas que pueden recibir voltajes dentro de un rango de valores continuos entre 0 y 5V. No obstante, la

electrónica de la placa tan solo puede trabajar con valores digitales, por lo que es necesaria una conversión previa del valor analógico recibido a un valor digital lo más aproximado posible. Esta se realiza mediante un circuito conversor analógico/digital incorporado en la placa.

Cada uno de los pines dispone de un canal de 10 bits ("bits de resolución") para guardar el valor de voltaje convertido digitalmente. La cantidad de bits de resolución marcan el grado de precisión conseguida en la conversión de la señal analógica a digital. En nuestro caso vemos que hay un máximo de 2^{10} (1024) valores diferentes. La placa Arduino puede distinguir para el voltaje digital desde el valor 0 hasta el valor 1023.

Estos pines tienen también toda la funcionalidad de los pines de entrada-salida digitales.

. Otros usos de los pines de la placa:

Pin 0 (RX) y pin 1 (TX): permiten que el microcontrolador ATmega328P pueda recibir directamente datos en serie (por el pin RX) o transmitirlos (por el pin TX) sin pasar por la conversión USB-serie que realiza el chip ATmega16U2.

Pines 2 y 3: se pueden usar, con la ayuda de programación software para gestionar interrupciones.

Pines 10 (SS), 11 (MOSI), 12 (MISO) y 13 (SCK): se pueden usar para conectar algún dispositivo con el que se quiera llevar a cabo comunicaciones mediante el protocolo SPI basándose en la librería SPI.

Pines A4 (SDA) y A5 (SCL): se pueden usar para conectar algún dispositivo con el que se quiera llevar a cabo comunicaciones mediante el protocolo I2C/TWI y haciendo uso de la librería Wire.

Pin 13: Este pin está conectado directamente a un LED incrustado en la placa de forma que si el valor del voltaje de este pin es HIGH se encenderá, y si dicho valor es LOW se apagará.

A continuación se muestra la relación de pines entre el microprocesador ATmega328P y la placa Arduino UNO Rev3.

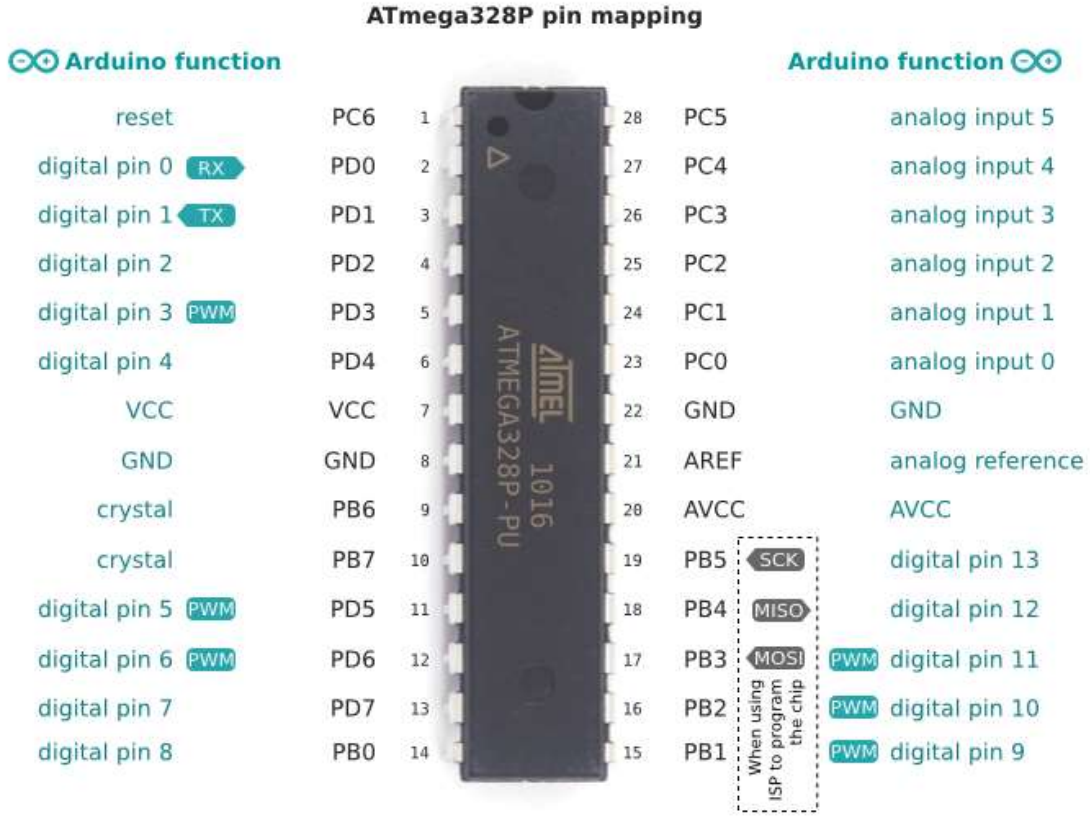


Figura 2-3. Relación de pines entre el microprocesador ATmega328P y la placa Arduino UNO

2.2.1.3 Ventajas Arduino

- **Es libre y extensible**: esto quiere decir que cualquiera que desee ampliar y mejorar tanto el diseño hardware de las placas como el entorno de desarrollo software y el propio lenguaje de programación, puede hacerlo sin problemas. Esto permite que exista un rico "ecosistema" de extensiones, tanto de variantes de placas no oficiales como de librerías software de terceros, que pueden adaptarse mejor a nuestras necesidades concretas.
- **Posee una gran comunidad**: muchas personas lo utilizan, enriquecen la documentación y comparten continuamente sus ideas.
- **Su entorno de programación es multiplataforma**: se puede instalar y ejecutar en sistemas Windows, Mac Os X y Linux. Esto no ocurre con el software de muchas otras placas.

- **Su entorno y el lenguaje de programación son simples y claros**: son muy fáciles de aprender y de utilizar, a la vez que flexibles y completos para que los usuarios avanzados puedan aprovechar y exprimir todas las posibilidades del hardware.
- **Las placas Arduino son baratas**.
- **Las placas Arduino son reutilizables y versátiles**: reutilizables porque se puede aprovechar la misma placa para varios proyectos, y versátiles porque las placas Arduino proveen varios tipos diferentes de entradas y salidas de datos.

2.2.2 Sensor

Con el objetivo de medir la distancia a la que se encuentra la bola se han utilizado dos sensores ultrasónicos HC-SR04. Esto lo consigue enviando un ultrasonido a través de un transductor (uno de los cilindros que funciona como un altavoz) y espera a que este ultrasonido rebote sobre un objeto y vuelva, retorno que es detectado por el otro transductor (funcionamiento similar a un micrófono).

El sensor devuelve el tiempo transcurrido entre el envío y la posterior recepción del ultrasonido. Como la velocidad de propagación de cualquier sonido en un medio como el aire es de valor conocido ese tiempo transcurrido lo podremos utilizar para determinar la distancia entre el sensor y el objeto que ha provocado su rebote.

2.2.2.1 Características

- Corriente de reposo: < 2mA
- Corriente de trabajo: 15 mA
- Ángulo de medición: 30º
- Ángulo de medición efectivo: < 15º
- El rango de distancia que puede detectar está comprendido entre 2 cm a 400 cm.
- Resolución: La precisión es de 0.3 cm
- Dimensiones : 45 mm x 20 mm x 15 mm
- Frecuencia de trabajo: 40 KHz.

- Se recomienda que el objeto a detectar disponga de una superficie plana y de al menos 0.5 m^2.

2.2.2.2 Funcionamiento

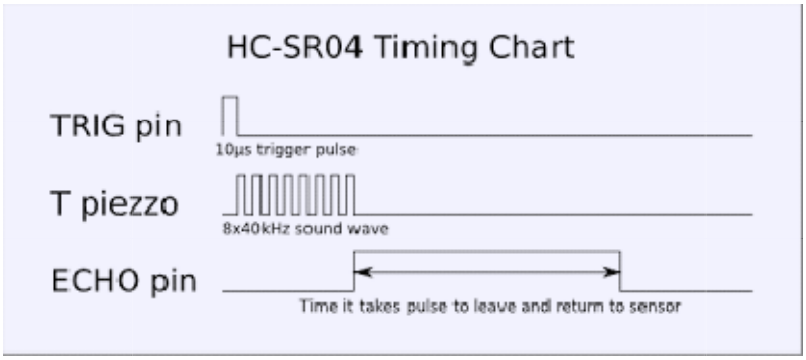


Figura 2-4. Funcionamiento sensor HC-SR04

Enviar un pulso HIGH de al menos 10 microsegundos por el pin Trig.

El sensor enviará 8 pulsos de 40KHz (ultrasonido) y coloca el pin Echo en HIGH. En este momento se detecta este evento y se inicia un conteo de tiempo.

El pin Echo se mantendrá en HIGH hasta recibir el eco reflejado por el obstáculo, momento en el cual el pin Echo cambiará a LOW, es decir, se terminará de contar el tiempo.

La distancia es proporcional a la duración del pulso y se puede calcular (utilizando la velocidad del sonido) mediante la siguiente fórmula:

distancia = velocidad del sonido x tiempo /2

Hay que tener en cuenta que la distancia que recorre la onda es el doble de la distancia que hay al obstáculo, por lo que es necesario dividir entre dos para conseguir la distancia real.

La velocidad del sonido es de 343 m/s a una temperatura de 20 ºC, por cada grado que aumenta la temperatura se produce un incremento de la velocidad del sonido de 0.6 m/s.

2.2.2.3 Disposición de los pines

Mirando la figura siguiente podemos identificar cada uno de los pines que aparecen en la placa del sensor.

Figura 2-5. Disposición pines del sensor HC-SR04

Vcc: Alimentación del sensor. Se debe de conectar a una fuente de +5V (4.5V min - 5.5V max)

Trig: Es el responsable de enviar el puso ultrasónico a través de un transductor. Deberá ser conectado a un pin de salida digital (Output) de la placa Arduino.

Echo: Es el responsable de recibir el eco de ese pulso; por tanto, se deberá conectar a un pin de entrada digital (Input) de la placa Arduino.

GND: Se debe conectar a tierra.

2.2.2.4 Programación de los sensores

El sensor proporciona una señal analógica que nos da la posición de la bola.

Se utilizarán dos sensores HC-SR04 que medirán la posición actual de la bola en el raíl. Cada uno de estos sensores serán llamados por una función de nombre sonar a la que se le pasa como parámetros los pines a los que está conectado en la placa Arduino y que devuelve la distancia a la que se encuentra la pelota. El código de esta función es el siguiente:

```
float sonar(int Echo, int Trig){
  digitalWrite(Trig, LOW); // Estabilizamos el sensor antes de enviar el pulso de activacion
  delayMicroseconds(4);
  digitalWrite(Trig, HIGH);    //Envío del pulso de activación
```

```
  delayMicroseconds(10);      // La duración de este pulso dura 10 microsegundos
  digitalWrite(Trig,LOW);
  duracion = pulseIn(Echo,HIGH); //Cuenta el tiempo transcurrido hasta recibir el rebote ultrasonico
  return(duracion*0.017);     //Devuelve la distancia
 }
```

De esta manera tendríamos dos valores (uno por sensor) de la distancia a la que se encuentra la pelota. Sin embargo estos valores no pueden utilizarse tal y como son recibidos de la función debido a que los sensores tienen las siguientes particularidades:

- El rango de funcionamiento de cada uno de los sensores en la práctica está comprendido entre los 2 cm y los 23 cm.
- El sensor en ocasiones da mediciones erróneas y alejadas de la distancia real, siendo este fenómeno más acusado en el sensor situado cercano al servomotor. Estas medidas erróneas son causadas por la inclinación de la barra respecto a la pelota, ya que como se mencionó anteriormente el sensor está diseñado para medir la distancia a superficies planas.
- Se necesita un tiempo entre la medición de un sensor y el siguiente. Esto es debido a que el pulso ultrasónico enviado por el primer sensor en activarse podría ser recibido por el otro sino se le da el tiempo suficiente para que dicho pulso rebote sobre la superficie de la bola.

Para estas particularidades se van a llevar a cabo una serie de medidas con las que conseguir una medición más fiable:

- Para atenuar las mediciones erróneas se hará uso de un filtro paso bajo

$$y_k = \alpha\, y_k + (1-\alpha) y_{k-1} \;\; con\; 0 < \alpha < 1$$

  De esta manera cada medida obtenida tendrá un peso de la medición actual y de la anterior.

- Una vez aplicado el filtro restringiremos esta medida para que este comprendida entre 2 cm y 23 cm.
- Entre la medición de un sensor y otro aplicaremos un *delay()* que sea suficiente para que no se acoplen los pulsos ultrasónicos entre los sensores.
- Como las mediciones del sensor más alejado del servomotor son más fiables se utilizará siempre el dato proporcionado por este siempre que se encuentre en el rango de 2 a 23 cm.
- Por último, para que la medida este comprendida entre 2 y 44cm se establece como “2” el punto más cercano al sensor más alejado del servo y como “44” el

punto más cercano a dicho sensor. Para ello y dado que el sensor solo da medidas entre 2 y 23 cm se tendrá que realizar un cambio de variable en la medida de dicho sensor como se verá en el código.

Este sería el código:

```
/* YI es la medición del sensor más alejado del servomotor */
   YIanterior = YI;       // Asignamos la última medida del sensor a YIanterior
   YI = sonar(EchoI, TrigI);  // Realizamos una medición con ese sensor
   YI = alfa * YI + (1 - alfa) * YIanterior;  // Le aplicamos el filtro
   if (YI > rangoMaximo) {YI = rangoMaximo;}
   if (YI < rangoMinimo) {YI = rangoMinimo}
  delay(5);  // Dejamos pasar 5 milisegundos para que no exista acoplamiento entre sensores
  YDanterior = YD;        // Se repite el mismo proceso para el otro sensor
  YD = sonar(EchoD, TrigD);
  YD = alfa * YD + (1 - alfa) * YDanterior;
  if (YD > rangoMaximo) {YD = rangoMaximo;}
  if (YD < rangoMinimo) {YD = rangoMinimo;}
 Yreal = 46 –YD;  //Cambio de variable para que la medida este comprendía entre 2 y 44 centímetros.
 if (YI < Maximo) {Y = YI;}  // Si la medición de este sensor está dentro del rango de funcionamiento, la asignaremos como señal de salida del sistema.
else {Y = Yreal;} // Sino asignaremos la del otro sensor.
```

2.2.3 Actuador

El actuador que se ha empleado para el movimiento de la barra es un servomotor SM-S2309S.

Figura 2-6. Servomotor SM-S2309S

Los servomotores son motores "gearhead", por tanto, son motores de corriente continua con engranajes los cuales limitan la velocidad de giro pero aumentan el par. Incorporan además un potenciómetro y cierta circuitería de control para poder establecer la posición del eje del motor de forma precisa. Es decir, su eje no gira libremente sino que rota un determinado ángulo, indicado a través de una señal de control. Lo que hace especial a un servo, es por tanto, que podemos ordenarle que gire una cantidad de grados concreta, cantidad que dependerá de la señal de control enviada por el microcontrolador previamente programado.

Hay que tener en cuenta que los servomotores están limitados en su giro en un rango aproximado entre los 0º y 180º.

2.2.3.1 Características

- Torque: 4.8V---1 Kg x cm
  6 V ---1.2 Kg x cm

- Velocidad: 4.8V---- 0.12 s/60º
  6 V ---- 0.10 s/60º

- Peso: 9g
- Dimensiones: 22.2 mm x 11.6 mm x 21.5 mm
- Rango de rotación: 180º

2.2.3.2 Funcionamiento

El desplazamiento angular del eje lo controlamos mediante pulsos de duración variable.

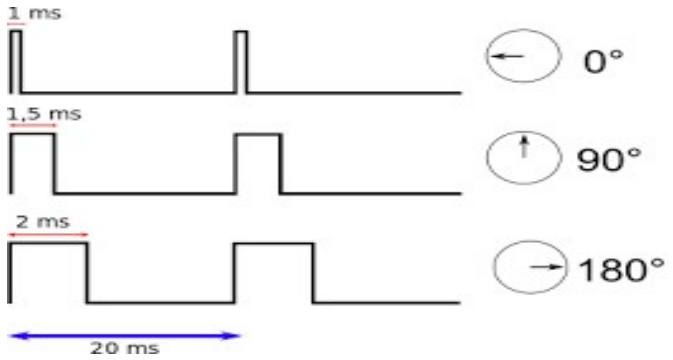


Figura 2-7. Funcionamiento del servomotor

Concretamente y según la figura anterior:

- Si el valor HIGH (5 V) del pulso se mantiene 1 ms, el eje del servo se ubicará en 0º
- Si el valor HIGH (5 V) del pulso se mantiene 1.5 ms, el eje del servo se ubicará en su posición central.
- Si el valor HIGH (5 V) del pulso se mantiene 2 ms, el eje del servo se ubicará en 180º.

Para obtener un valor intermedio bastará con interpolar para conseguir el ángulo deseado.

Sin embargo, estos valores son aproximados ya que por ejemplo en el servo utilizado para este proyecto, la posición central se obtiene con un valor HIGH mantenido durante 1.4ms.

2.2.3.3 Disposición de los pines

El servomotor utilizado consta de tres cables que podemos distinguir por sus colores:

- **Rojo**: para recibir la alimentación eléctrica. El cable de alimentación ha de conectarse a una fuente que pueda proporcionar 5V y al menos 1A.
- **Negro**: para conectarse a tierra.
- **Blanco**: sirve para transmitir al servo, de parte del microcontrolador, los pulsos eléctricos de una frecuencia fija de 50 Hz, que ordenarán el giro concreto de su eje.

2.2.3.4 Programación

En este caso se va a hacer uso de la librería oficial "Servo". Esta librería utiliza el Timer1 de Arduino por lo que dicho Timer no podremos utilizarlo para ningún tipo de interrupción.

Lo primero que hay que realizar es declarar un objeto de tipo Servo en la zona de declaraciones de variables globales. Suponiendo que lo llamamos "miservo" esto se haría con la instrucción:

Servo.miservo;

La primera función imprescindible que debemos escribir (dentro del setup()) es:

miservo.attach(): vincula el objeto "miservo" con el pin digital de la placa Arduino donde está conectado físicamente el cable de control del servomotor. Su parámetro precisamente es el numero de ese pin. Esta función no tiene valor de retorno.

A partir de aquí, únicamente haremos uso de la siguiente función de la librería para controlar el servomotor:

miservo.writeMicroseconds(): controla el eje del servomotor. Su parámetro de tipo "int" es la duración del pulso de la señal de control. Esta función no tiene valor de retorno.

En el código:

```
miservo.writeMicroseconds(reposo +  V); // siendo V la acción de control escalada que puede ser positiva o negativa.
```

La variable reposo indica la posición central del servo que es aquella donde el raíl forma un ángulo de 0º con respecto a la horizontal a la que se le suman las acciones de control correspondientes.

Con esta instrucción el servomotor girará hasta la posición indicada por el controlador.

2.2.4 Fuente de alimentación

La utilización de una fuente de alimentación externa se hace imprescindible para llevar a cabo este proyecto. El servomotor exige más corriente de la que puede aportar el cable USB (aproximadamente 0,5 A) por lo que si se prescinde de una fuente de alimentación externa podemos obtener movimientos erráticos del eje del servo e incluso reseteos de la placa Arduino.

En nuestro caso se utilizará un transformador AC/DC que en su salida entrega un voltaje de 12 V y una intensidad de 1 A.

Figura 2-8. Fuente alimentación externa

La intensidad ya será suficiente para el correcto funcionamiento del servomotor y el voltaje está dentro del rango de voltaje adecuado para el funcionamiento de la placa Arduino (7-12V).

2.3 Estructura

Cada una de las piezas que componen la estructura se ha diseñado mediante software de modelado 3D. La fabricación de las mismas se ha realizado mediante el uso de tecnología 3D, más concretamente por el proceso de Fabricación por Filamento Fundido (FFF) siendo el material utilizado ABS.

En el Anexo A se describe con detalle los siguientes aspectos relacionados con el diseño y fabricación de la estructura:

- Tecnologías y materiales existentes de impresión 3D.
- Argumentación de la elección de tecnología FFF y de material ABS.
- Proceso de impresión 3D. Donde se describen todos los pasos que hay que acometer para la fabricación de las piezas.

2.4 Guía de montaje

Una vez descritos todos los componentes que conforman nuestro proyecto, se pasa a indicar los pasos necesarios para el montaje del Ball & Beam.

Las piezas impresas en 3D que necesitamos para el montaje de la estructura son las siguientes:

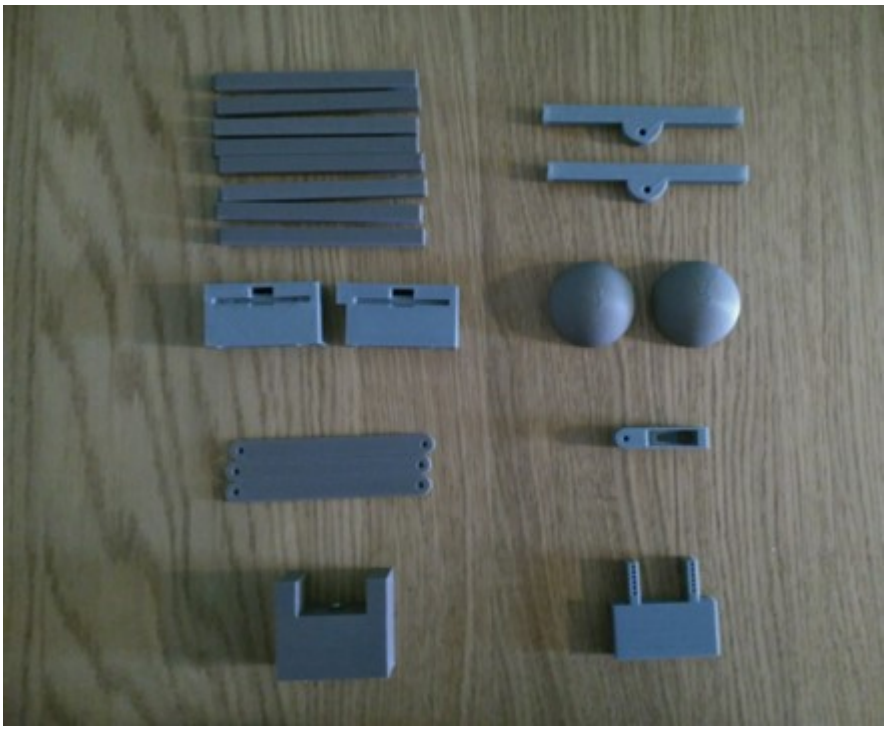


Figura 2-9. Piezas impresas para montaje del ball and beam

De izquierda a derecha y de arriba abajo:

(A) 8 Segmento raíl
(B) 2 Segmento central raíl
(C) 2 Soporte sensor
(D) 2 Semiesfera
(E) 3 Biela
(F) 1 Manivela
(G) 1 Soporte voladizo
(H) 1 Soporte servomotor

Los planos de estas piezas se encuentran en el Anexo B.

2.4.1 Eliminación de rebabas y soportes

En algunas de las piezas, debido a su geometría, es inevitable el tener que hacer uso de soportes para su impresión 3D. En otras, pueden aparecer pequeñas imperfecciones o rebabas. Por tanto, antes de comenzar el montaje de la estructura será necesario realizar algún tratamiento superficial sobre dichas piezas.

- Para la eliminación de los soportes se usará una pequeña cuchilla o cutter que habrá que utilizar con cuidado para no dañar el resto de la pieza.
- Para quitar rebabas indeseables y suavizar superficies rugosas se podrá hacer uso de una lija.

Figura 2-10. Piezas con soportes

2.4.2 Montaje del raíl

En la realización de este paso usaremos las 8 piezas de segmento de rail (A) y las 2 de segmento central de rail (B).

Figura 2-11. Montaje del raíl

Como se explicó con anterioridad, las barras que forman el rail no se pueden imprimir en una sola pieza debido a su gran longitud (superior al tamaño de la plataforma de impresión). Por ello, necesitamos fusionar los diferente segmentos hasta formar dos barras identicas de cinco segmentos cada una. La secuencia de piezas de cada una de las barras será : (A)-(A)-(B)-(A)-(A).

Para este proceso se hará uso de un disolvente líquido como la acetona que aplicada de forma correcta logrará la unión química de las distintas partes. Más concretamente se utilizará un método conocido en el argot como “jugo de ABS”. Este producto se consigue dejando que un trozo de filamento ABS se disuelva completamente en acetona, de forma que al final se consigue una disolución líquida uniforme de aspecto viscoso.

Se aplicará esta solución en los encajes de los segmentos con ayuda de un pincel y se unirán fisicamente las piezas. La acetona actuará sobre la superficie de las piezas

para después evaporarse progresivamente, de forma que el ABS que estaba predisuelto en la misma ayuda a fortalecer y sellar la zona de unión.

Hay que asegurarse de que cuando realizamos la unión fisica de las distintas partes, estas queden completamente rectas sin ningún escalón entre segmentos. Si observamos que no es así, se deberá corregir manualmente antes de que la acetona se haya evaporado.

Por último pero no menos importante habrá que extremar los cuidados en la manipulación directa con la acetona que podría ocasionar problemas como: creación de nubes tóxicas, inflamables y/o explosivas o el provocar problemas de quemaduras o irritación en caso de contacto con la piel o mucosas.

2.4.3 Colocación de sensores

Se deben colocar los dos soportes de sensor (C), en los extremos de las barras que acabamos de conformar.

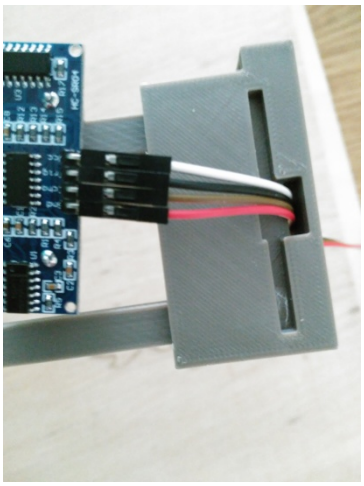


Figura 2-12. Colocación sensor en pieza impresa

Estos soportes han sido diseñados de tal manera que una vez colocados los sensores, haya una distancia entre estos y el principio del raíl (lugar hasta el que puede llegar la bola) de dos centímetros. Esta distancia no es casual, se trata de la distancia mínima que requiere el sensor para que su medición sea fiable.

Ambas barras deben entrar hasta el fondo de los orificios de esta pieza. Es posible que a pesar de que se eliminaron los soportes en estos orificios pueda quedar algún resto de material dentro imposibilitando esto. En ese caso se deberá volver a hacer uso de una lija o incluso aplicar acetona líquida con un pincel hasta que consigamos que las barras se introduzcan hasta el final.

Una vez conseguido, se colocará el sensor en la cavidad destinada para ello como se puede ver en la imagen.

Ya tenemos el raíl en el que se desplazará la bola completamente montado.

2.4.4 Fijación en tablero

En este paso se utilizará el soporte voladizo (G) y el soporte del servomotor (H) además de un tablero de contrachapado.

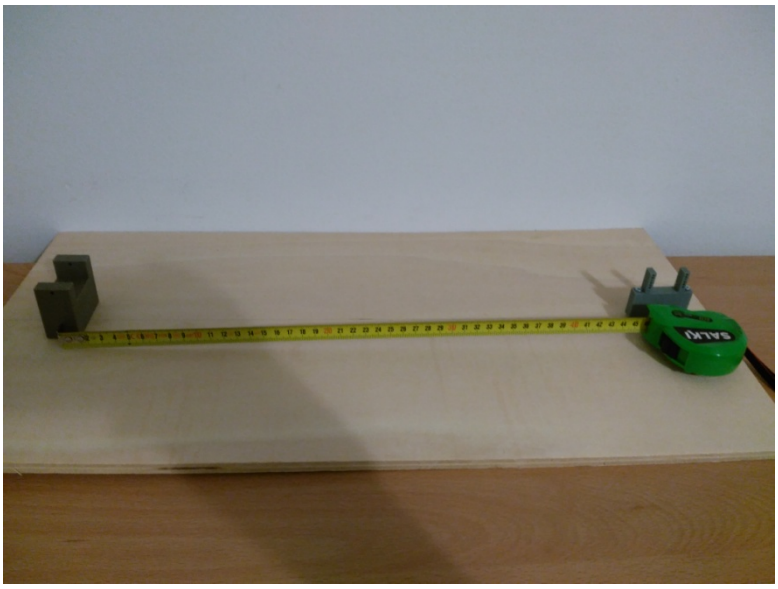


Figura 2-13. Fijación en tablero de piezas

En el tablero de contrachapado, que debe ser de 5 mm de grosor, dibujaremos una línea recta para tener una referencia de donde hay que colocar las piezas. Una vez hecho, se introducirán unos tornillos tirafondos de métrica 3 y 35 mm de longitud en los agujeros de las piezas dispuestos para ello.

En este punto se procederá a atornillar una de las piezas al tablero. Para ello es de vital importancia que los tornillos se posicionen sobre la línea previamente dibujada.

La distancia entre los tornillos más próximos de ambas piezas debe distar 46 cm, para lo cual se hará una marca sobre la línea para finalmente anclar la pieza restante.

2.4.5 Colocación del servomotor y del mecanismo biela – manivela

En este último paso del montaje de la estructura debe cumplirse (como se comentó en el capítulo dedicado al modelado matemático) que el ángulo formado entre la manivela y la biela sea de 90º. Para ello realizaremos las siguientes acciones:

1º. Con el servomotor sin fijar a su soporte y mediante un sketch de Arduino buscamos la posición de 90º de giro. Esta posición es su posición central (el giro del servo está comprendido entre 0 y 180º), por lo que nos quedará el mismo recorrido del eje del servo en ambos sentidos.

2º. Encaje a presión de la pieza biela (E) en el servomotor y anclaje del mismo en su soporte utilizando los pequeños tornillos que vienen incluidos en la caja del servomotor. Este soporte dispone de agujeros a distintas alturas para asegurar la posibilidad de formar un ángulo de 90º entre la manivela y la biela. En principio, su colocación correcta será en el tercer agujero empezando a contar desde arriba.

3º. Se encaja a presión la pieza plástica ("horns") en la cavidad de la biela dispuesta para ello y se atornilla la misma al servomotor con un tornillo pequeño como el utilizado anteriormente.

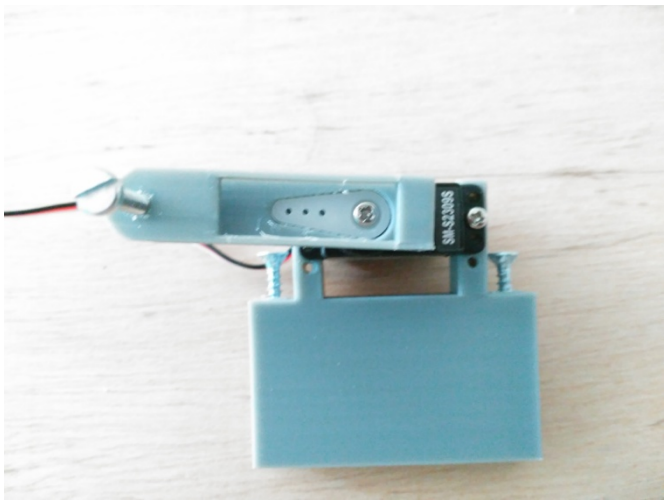


Figura 2-14. Colocación del servomotor en pieza impresa

En la imagen se puede apreciar cómo debe colocarse el conjunto servomotor-soporte-biela.

Es fundamental que el encaje entre el horns y la biela sea perfecto para que el movimiento del eje del servo y de la biela sea solidario y no exista un problema de holguras..

4º La biela se atornillará por un lado al raíl a través del agujero existente en la pieza de soporte del sensor (C) y por el otro extremo a la manivela (F). Se utilizará en ambos un tornillo de métrica 3 de 16 mm de longitud. Estas uniones deben permitir el libre giro de la biela-manivela sin que exista un rozamiento excesivo.

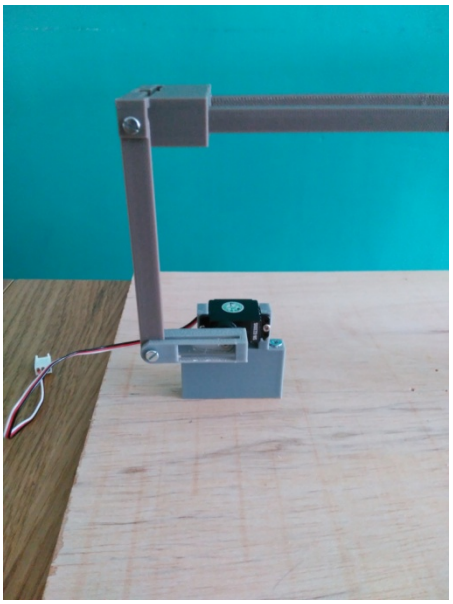


Figura 2-15. Montaje conjunto biela - manivela

5º En el otro extremo de la estructura, utilizaremos dos piezas (E) que a pesar de haberlas denominado como biela, se utilizarán como barras fijas. Dichas piezas se atornillarán por uno de sus lados a la pieza (G) de manera que no se permita el giro entre ambas. Por el otro lado se atornillará al soporte del sensor (C) de manera que permita el giro entre ellas.

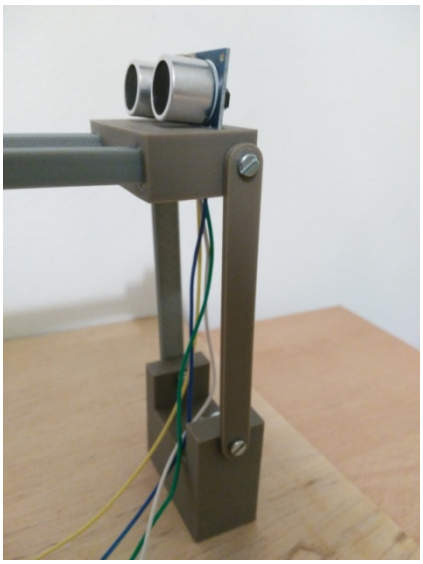


Figura 2-16. Montaje barras fijas

2.4.6 Montaje de la bola

Por último, nos queda disponer de la bola que recorrerá el raíl y de la que debemos conocer en todo momento su posición. Esta pelota se ha impreso en dos mitades por la imposibilidad de fabricarla en un sola pieza debido a las limitaciones de la impresión 3D. Por ello, es necesario pegar ambas partes con "jugo ABS" (como se comento anteriormente). Una vez pegada y secada esta unión, se aplicará con un pincel acetona de manera uniforme en su superficie para eliminar rugosidades y pulir la bola. Con ello conseguimos que el rozamiento futuro de la pelota y la viga sea menor.

Figura 2-17. Bola

2.4.7 Conexiones con Arduino

Para completar el montaje del "Ball and Beam" solo queda mostrar cómo se debe conectar los sensores y actuadores a Arduino. Para ello se seguirá, el siguiente diagrama de conexiones.

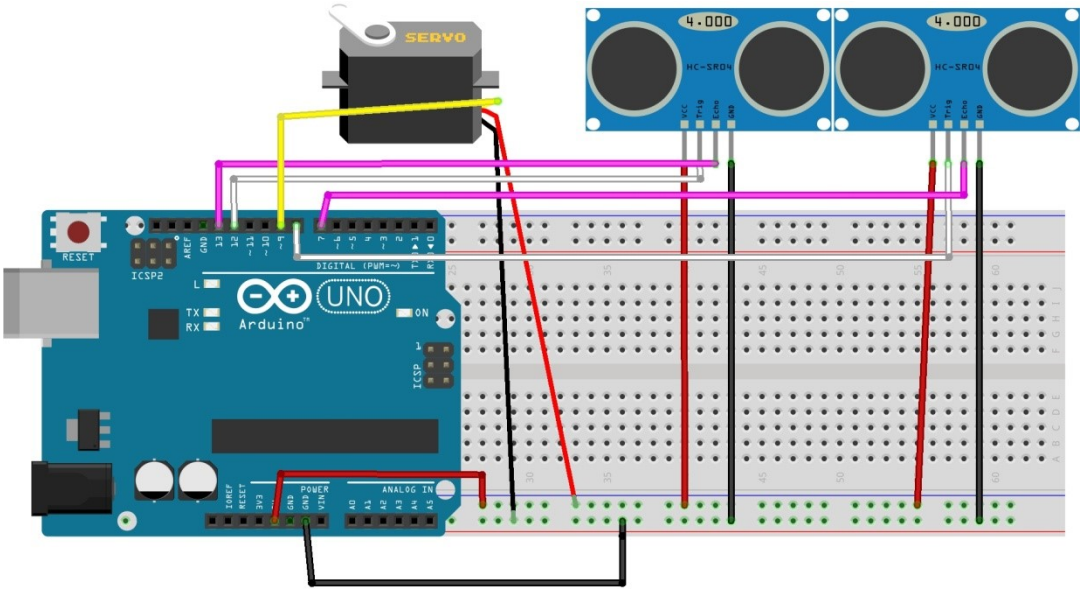


Figura 2-18. Diagrama de conexiones en Arduino

Correspondencia de pines en Arduino:

- Pin 9: Pin de señal del servomotor.
- Pin 12 y 11: Pin Trig y Echo del sensor ultrasónico situado encima del servo.
- Pin 8 y 7: Pin Trig y Echo del otro sensor ultrasónico.

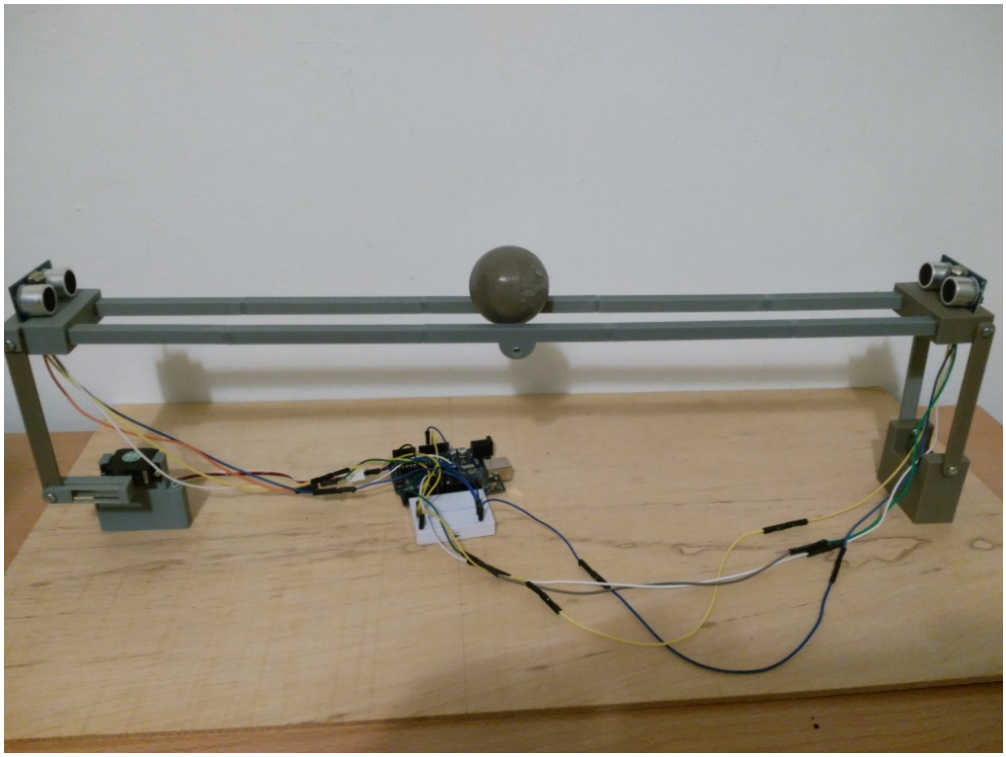


Figura 2-19. Estructura montada

3. Modelo matemático

Estudiamos el movimiento de una esfera a lo largo de un carril inclinado.

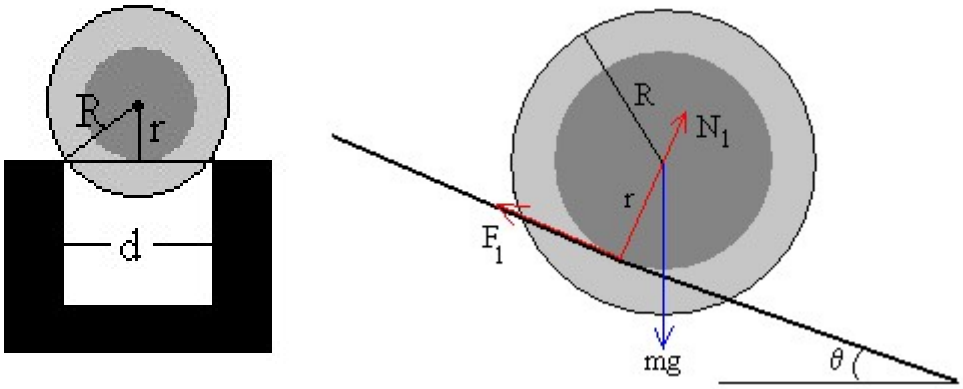


Figura 3-1. Fuerzas sistema ball and beam

Se trata de una esfera de radio R que baja rodando sin deslizar a lo largo de un carril cuyos raíles, dispuestos en paralelo, están separados una distancia d < 2R.

La esfera tiene dos puntos de contacto con el carril, como podemos apreciar en la figura. Hay, por tanto, dos fuerzas de rozamiento F_r y dos reacciones N del carril. La fuerza de rozamiento F_r tiene la dirección del carril, la reacción N es perpendicular al carril y pasa por el centro de la esfera.

- La resultante de las fuerzas de rozamiento es $F_1 = 2F_r$
- La resultante de las dos reacciones iguales y simétricas $N_1 = 2N_r$/R

Siendo r la distancia entre el plano que contiene las dos vías del carril y el centro de la esfera $r^2 = R^2 - d^2/4$.

Las fuerzas que actúan sobre la esfera son:

- El peso mg
- La resultante de las reacciones N_1
- La resultante de las fuerzas de rozamiento F_1

El sistema de referencia utilizado en las siguientes ecuaciones es:

x: Esta coordenada corresponde a la dirección que marcan las dos barras paralelas y sobre la que rodará la pelota.

y: Dirección normal a las barras paralelas.

Las ecuaciones del movimiento de la esfera son:

Movimiento de traslación del centro de masas:

$$m\ g\ \sin\theta - F_1 = m\ a = m\ \frac{\partial^2 x(t)}{\partial t^2} \qquad (\text{I})$$

Equilibrio en la dirección perpendicular al plano inclinado:

$$m\ g\ \cos\theta = N_1 \qquad (\text{II})$$

Suponiendo que la bola rueda sin deslizar, tenemos la siguiente ecuación:

$$a = \frac{\partial^2 x(t)}{\partial t^2} = r\alpha = \text{ r } \frac{\partial^2 \theta(t)}{\partial t^2}$$

Despejando se obtiene la **relación entre x y θ**: $\frac{\partial^2 \theta(t)}{\partial t^2} = \frac{1}{r}\frac{\partial^2 x(t)}{\partial t^2}$ (III)

Nos queda tratar la **ecuación de rotación de la esfera**:

$$F_1\ r = I_{esfera}\ \frac{\partial^2 \theta(t)}{\partial t^2} \qquad \text{siendo } I_{esfera} = \frac{2}{5}\ m\ R^2$$

Despejando de (III) y (IV):

$$F_1 = \frac{2}{5}\ m\ \frac{R^2}{r^2}\ \frac{\partial^2 x(t)}{\partial t^2}$$

Sustituyendo la F_1 en la ecuación (I):

$$m\,g\ sin\,\theta - \frac{2}{5}\,m\,\frac{R^2}{r^2}\,\frac{\partial^2 x(t)}{\partial t^2} = m\frac{\partial^2 x(t)}{\partial t^2}$$

Operando:

$$m\,g\ \sin\theta = (\,1 + \frac{2}{5}(R/r)^2)\ m\frac{\partial^2 x(t)}{\partial t^2}$$

$$\frac{\partial^2 x(t)}{\partial t^2} = \frac{g}{1 + \frac{2}{5}\,(R/r)^2}\sin\theta$$

Se trata de una ecuación no lineal, por lo que vamos a ver si es posible su linealización aplicando la aproximación para pequeños ángulos $\sin\theta \approx \theta$.

Linealización

Para ver el ángulo máximo de inclinación de la barra, es necesario saber cuál es la relación entre el ángulo de inclinación de las barras (θ) y el ángulo del brazo del servomotor (β).

Debido a la geometría de la estructura podemos relacionar ambos ángulos mediante la siguiente ecuación:

$$b * \sin\,\beta = L * \sin\theta$$

Siendo: Longitud brazo del servo (b) = 0.045 m y Longitud de las barras (L) = 0.46 m.

Hay que recordar que el brazo del servo en su posición de equilibrio se encuentra situado en la mitad de su recorrido. El recorrido del servo es de 180º por lo que en dicha posición central podrá girar 90º en sentido horario y otros 90º en sentido anti horario. Por tanto $\beta_{max} = 90º$.

Aplicando la ecuación anterior con los datos que disponemos, se obtendrá el valor de θ_{max}:

$$\sin\theta_{\max} = \frac{b * \sin\beta_{max}}{L} = \frac{0.045 * 1}{0.46} = 0.0978\,rad \;\mapsto\; \theta_{\max} = 0.098\,rad$$
$$= 5.614º$$

El error máximo que se puede producir al realizar la aproximación de pequeños ángulos $\sin\theta \approx \theta$ es:

$$\frac{\theta_{\max} - \sin\theta_{\max}}{\sin\theta_{\max}} * 100 = \frac{0.098 - 0.0978}{0.0978} * 100 = 0.20\%$$

Es un error completamente asumible en nuestro sistema, por lo que se puede realizar la linealización propuesta, quedando:

$$\frac{\partial^2 x(t)}{\partial t^2} = \frac{g}{1 + \frac{2}{5}\,(R/r)^2}\theta(t)$$

Función de transferencia

Para hallar la función de transferencia suponemos las condiciones iníciales iguales a cero:

$$S^2\,X(s) = \frac{g}{1 + \frac{2}{5}\,(R/r)^2}\theta(\mathrm{s})$$

Reorganizamos los términos:

$$\frac{X(s)}{\theta(s)} = \frac{g}{1 + \frac{2}{5}\,(R/r)^2}\,\frac{1}{S^2}$$

Datos:

g = 9.8 m/s^2 R = 0.022 m r = 0.016 m

Introduciendo la función de transferencia con estos datos se obtiene la respuesta del sistema en bucle abierto ante una entrada en escalón unitaria.

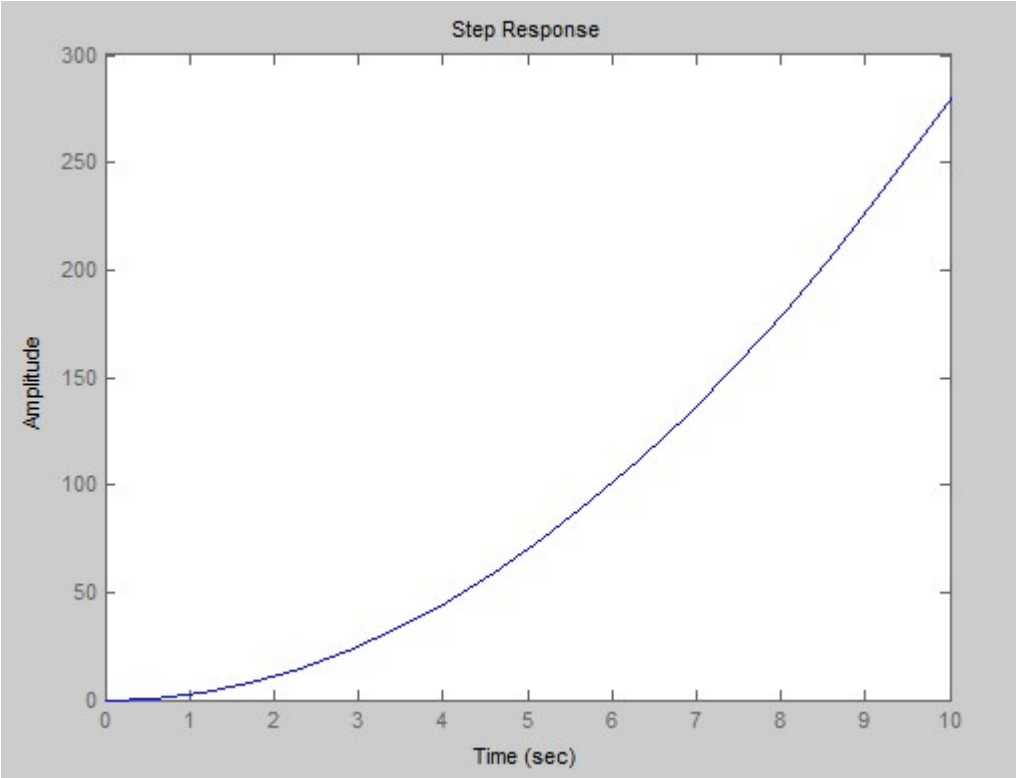


Figura 3-2. Respuesta del sistema en bucle abierto ante un escalón unitario

Como se puede apreciar en la figura anterior el sistema es inestable en bucle abierto por lo que será necesario hacer uso de algún método de control.

4. Identificación del sistema

En este capítulo se llevará a cabo la identificación del sistema comparando el modelo teórico con la planta real. Ante la misma entrada (de tipo escalón) se debe asegurar que las salidas de ambos modelos sean similares.

Para ello haremos uso de la función de transferencia obtenida en el capítulo anterior:

$$\frac{X(s)}{\theta(s)} = \frac{g}{1 + \frac{2}{5}(R/r)^2} \frac{1}{S^2}$$

Datos: g = 9.8 m/s^2 R = 0.022 m r = 0.016 m

Sustituyendo los datos podemos escribir la función de transferencia de la siguiente manera:

$$\frac{X(s)}{\theta(s)} = \frac{K}{S^2} \quad siendo\ K = 5.58$$

4.1 Modelo teórico

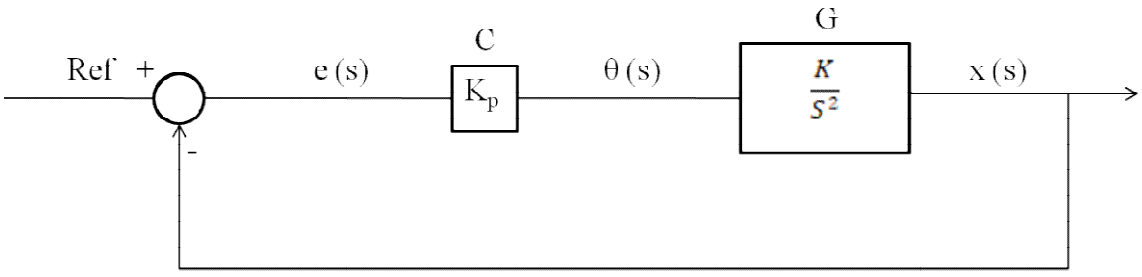


Figura 4-1. Diagrama de bloques con acción proporcional

Del diagrama de bloques sacamos:

$$C * G = \frac{K_p * K}{S^2}$$

Y la función de transferencia en bucle cerrado:

$$G_{bc} = \frac{C * G}{1 + C * G} = \frac{\frac{K_p * K}{S^2}}{1 + \frac{K_p * K}{S^2}} = \frac{K_p * K}{S^2 + K_p * K}$$

Se hallan los polos del sistema: $S^2 + K_p * K = 0 \Longmapsto S = \pm\sqrt{-K_p * K}$

Al ser $K_p > 0$, ambos polos van a ser complejos puros por lo que el sistema será críticamente estable.

Se va a comprobar esta afirmación con la ayuda de Matlab y otorgando a K_p un valor de 1.

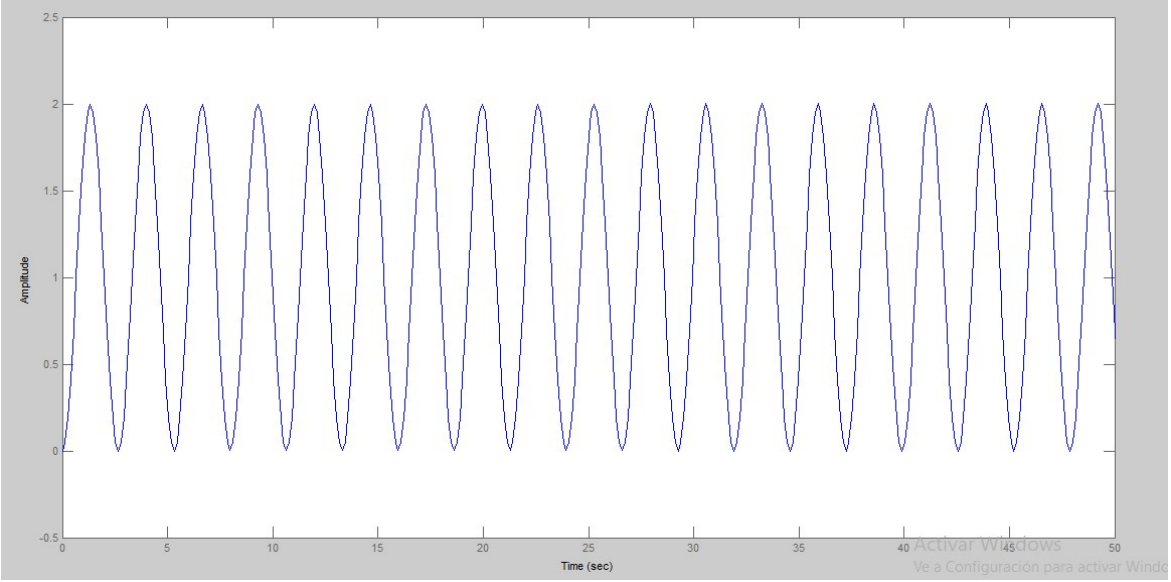


Figura 4-2. Respuesta del sistema en bucle cerrado con acción proporcional.

Como se predijo el sistema es críticamente estable por lo que se hace necesario incorporar un término derivativo.

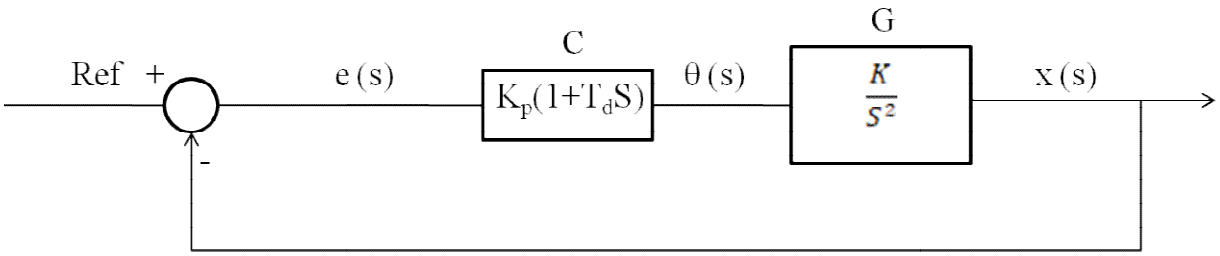


Figura 4-3.Diagrama de bloques con acción proporcional y derivativa.

Del diagrama de bloques sacamos:

$$C * G = \frac{K_p * K * (1 + T_d S)}{S^2}$$

Y la función de transferencia en bucle cerrado:

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

Se va a suponer que $K_p * K = 1$, quedando la función de transferencia como:

$$G_{bc} = \frac{(1 + T_d S)}{S^2 + (1 + T_d S)}$$

Se hallan los polos del sistema: $S^2 + T_d S + 1 = 0 \Longmapsto S = \frac{-T_d \pm \sqrt{{T_d}^2 - 4}}{2}$

Si se da un valor $T_d = 2$, los polos serán:

$$S = \frac{-2 \pm \sqrt{4 - 4}}{2} = -1\ (Doble)$$

En este caso, tenemos un polo doble negativo.

Esta es la gráfica de la respuesta del sistema ante una entrada escalón unitario:

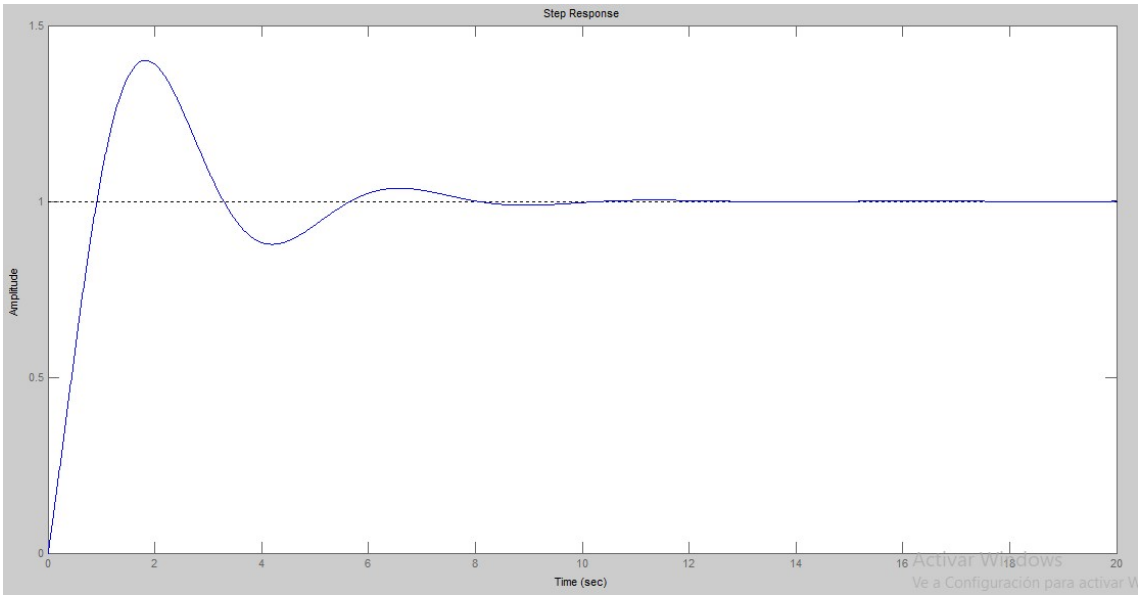


Figura 4-4. Respuesta del sistema en bucle cerrado con acción proporcional y derivativa.

Al ser los polos negativos, se consigue que el sistema sea estable.

4.2 Planta real

Los valores obtenidos de la constante proporcional (Kp) y derivativa (Kd) en el modelo teórico serán utilizados en nuestro sketch en Arduino. De esta forma se obtendrá una gráfica de la respuesta de la planta ante un escalón unitario. Esta grafica deberá ser semejante a la obtenida en el modelo teórico.

Sin embargo, al realizar esta tarea comprobamos que la pelota no llega a moverse. El motivo de que la posición de la bola no varíe, y por tanto la prueba no sea válida, se debe a la existencia de una zona muerta que los valores de Kp y Kd utilizados no son capaces de vencer. La zona muerta se define como aquella donde la sensibilidad del instrumento es nula lo que hace que no cambie su indicación y señal de salida.

En nuestra planta, la zona muerta llega a abarcar un 60% del rango de valores que puede tomar la señal de entrada. El caso más extremo lo encontramos cuando la pelota está parada ya que el rozamiento estático es bastante elevado.

Debido a todo lo mencionado no podemos conseguir que ambas gráficas sean iguales por lo que será necesario ajustar los parámetros del PID a mano.

5. Control PID en planta real

Antes de realizar el ajuste de los parámetros del controlador PID se va a ver como incorporar a nuestro sketch de Arduino un tiempo de muestreo y las acciones de control.

5.1 Configuración tiempo de muestreo

Para el diseño de cualquier control PID es fundamental que los ciclos de medida y reacción siempre tengan la misma duración.

Se ha determinado que el tiempo de muestreo será de 50 ms a pesar de que con el microprocesador utilizado podríamos conseguir tiempos de muestreo menores. Sin embargo con ciclos de medida menores tendríamos los siguientes problemas:

- Acoplamiento entre los pulsos ultrasónicos de ambos sensores, perdiendo fiabilidad en las medidas.
- Tiempo insuficiente para que el servomotor gire entre dos ángulos diferentes.
- La medida de velocidad de la bola pierde precisión porque las diferencias en la posición de la bola entre ciclos son ínfimas y el cálculo de la velocidad de la bola se tomará como la diferencia en su posición en dos ciclos consecutivos del programa.

Veamos la manera de mantener el tiempo de muestreo (intervalo de tiempo en que se ejecuta la ley de control).

5.1.1 Interrupciones temporales en Arduino

Los microprocesadores incorporan el concepto de interrupción que es un mecanismo que permite asociar una función a la ocurrencia de un determinado evento. Cuando ocurre el evento el procesador "sale" inmediatamente del flujo normal del programa y ejecuta la función de interrupción asociada ignorando por completo cualquier otra tarea. Al finalizar la función el procesador vuelve al flujo principal, en el mismo punto donde había sido interrumpido.

Arduino dispone de una serie de eventos en los que definir interrupciones. Por un lado tenemos las interrupciones de los timers (temporizadores). Por otro lado, tenemos las interrupciones de hardware, que responden a eventos (normalmente cambios en la tensión) ocurridos en ciertos pines físicos. En el caso que nos ocupa las interrupciones para mantener el tiempo de muestro harán uso de las interrupciones de los timers.

En la placa Arduino Uno disponemos de 3 timers distintos que presentan las siguientes particularidades:

- **Timer 0**: es un temporizador de 8 bits, es decir, puede tomar $2^8 = 256$ valores o puede contar de 0 a 255. Este temporizador es usado en las funciones delay(), millis() y micros (). Si se usan estas funciones en nuestro programa no podremos usar este temporizador.

- **Timer 1**: es un temporizador de 16 bits, es decir, puede tomar $2^{16} = 65536$ valores o puede contar de 0 a 65535. Este temporizador se usa en la librería servo().

- **Timer 2** : es un temporizador de 8 bits. Es similar al timer 0 con la diferencia de que este es el que se usa para la función tone().

También las salidas PWM usan los temporizadores. En el caso de Arduino Uno:

- El timer 0 controla las salidas PWM 5 y 6.
- El timer 1 controla las salidas PWM 9 y 10.
- El timer 2 controla las salidas PWM 3 y 11.

En el caso que nos ocupa no se podrá hacer uso del timer 0 debido a que es necesaria la utilización de la función delay() entre las mediciones de los dos sensores ni del timer 1 debido a que se hará uso de la librería servo().

Por ello, el timer elegido para llevar a cabo la interrupción es el timer 2.

5.1.2 Modo de temporización output compare

El temporizador se puede configurar para funcionar de distintas maneras, una de ellas es el modo output compare, donde la interrupción se dispara al alcanzar cierto valor de conteo. Para ello en el caso del timer 2 hay que modificar los registros: TCCR2A y TCCR2B. La descripción de dichos registros se puede ver en los siguientes cuadros.

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 5-1. Registros TCCR2A y TCCR2B

Para configurar el temporizador en modo output compare, modo CTC hay que poner WGM21 = 1 manteniendo el resto a cero como se puede entender en el siguiente cuadro.

Mode	WGM22	WGM21	WGM20	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Figura 5-2. Modos de operación del timer 2

5.1.3 Ajuste del preescalador

El preescalador no es más que un divisor de frecuencia. El reloj de la placa Arduino oscila a 16MHz, así que el periodo es :

$$T = \frac{1}{16.000.000} = 62{,}5\ ns$$

Al estar utilizando el timer 2, podríamos contar hasta 256 x 0,0000000625 = 0,000016 s = 0.016 ms

Como necesitamos un tiempo mayor (50 ms) en nuestra interrupción temporal, se tendrá que hacer uso del preescalador, que lo que hace es incrementar el timer en vez de en cada flanco, cada un número fijo de flancos (es pues un divisor de frecuencia con la que se incrementa el timer).

Para hacer uso del preescalador se hará uso del siguiente cuadro:

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Figura 5-3. Preescaladores timer 2

5.1.4 Valor de disparo

Se va a ajustar el preescalador de forma que el tiempo máximo que se pueda contar sea mayor que el tiempo al que queremos que salte la interrupción.

Se parte del caso más favorable para que sea posible realizar este conteo (CS22=1, CS21=1, CS20 =1) que corresponde a un preescalador de 1024. En este caso el periodo al que se incrementa el valor del temporizador es:

$$T = \frac{1}{\frac{16000000}{1024}} = 0.000064\ s = 0.064\ ms$$

Por lo tanto, el máximo valor que podemos contabilizar es:

$$256 * T = 0.01638\ s \approx 16.38\ ms$$

Se acaba de comprobar que no es posible utilizar ninguno de los tres timers de la placa para fijar el tiempo de muestreo. Se utilizará como alternativa el uso de la función millis().

5.1.5 Uso de la función millis() como temporizador

En este proyecto se puede hacer uso de la función millis() como temporizador debido a que en nuestro código no existen multitareas sino que todo el código se emplea en medir, calcular y actuar. Por ello para asegurar que se mantiene el tiempo de muestreo se puede proceder:

```
tiempo = millis();   // Se llama a la función millis() y se asigna a la variable tiempo
if (tiempo - tiempoAnterior >= tiempoMuestreo ){
```

Si el tiempo que se acaba de asignar menos el tiempo de la anterior vez que se accedió al *if* es mayor o igual al tiempo de muestreo se accederá nuevamente al *if*. Si no se cumple esta condición se seguirá volviendo a ella hasta que esta sea cierta.

En caso de entrar y tras ejecutar el código pertinente se deberá terminar ejecutando la siguiente sentencia:

```
tiempoAnterior = tiempo;
```

De esta manera actualizamos el tiempo anterior para futuras comprobaciones de entrada en el *if*

Con ello se ha conseguido que se entre en el bucle de medición-actuación cada 50 ms.

5.2 Programación del PID

El controlador recibe una señal externa que representa el valor que se desea alcanzar. Esta señal recibe el nombre de punto de referencia (o set point), la cual es de la misma naturaleza y tiene el mismo rango de valores que la señal que proporciona el sensor. El controlador resta la señal de punto actual a la señal de punto de referencia, obteniendo así la señal de error, que determina en cada instante la diferencia que hay entre el valor deseado y el valor medido.

La señal de error es utilizada por cada uno de los tres componentes del controlador PID.

5.2.1 Control proporcional

La acción de control proporcional (P), da una salida del controlador que es proporcional al error, es decir:

$$u(t) = Kp\, e(t)$$

donde K_p es una ganancia proporcional ajustable. Un controlador proporcional puede controlar cualquier planta estable, pero posee desempeño limitado y error en régimen permanente (off-set).

En nuestro código se resume en la siguiente línea:

```
P = Kp * error;
```

5.2.2 Control derivativo

La acción de control derivativa (D)**,** da una salida del controlador que es proporcional a la diferencia del error entre el ciclo actual y el anterior.

$$u(t) = Kd\, \frac{de(t)}{dt}$$

Esta acción tiene carácter de previsión, lo que hace más rápida la acción de control, aunque tiene la desventaja importante que amplifica las señales de ruido y puede provocar saturación en el actuador.

La derivada de la acción de control derivativa no es más que la velocidad que lleva la bola en un momento determinado:

$$\frac{de(t)}{dt} = \frac{error - errorAnterior}{tiempoMuestreo}$$

Debido a la amplificación de las señales de ruido y a que las mediciones proporcionadas por nuestro sensor no son 100% fiables se hace imprescindible usar algún tipo de filtro. En este caso se hará uso de un filtro promedio, este tomará las últimas diez últimas velocidades registradas y calculará la media de ellas. Este cálculo se realizará por medio de una función de nombre *promedioVel* que nos devolverá la velocidad media.

```
float promedioVel(){
  for(int i=0; i<numVelocidad-1; i++){
        vectorVelocidad[i]=vectorVelocidad[i+1];  //Desplaza los valores una posición hacia la izquierda del vector
    }
  vectorVelocidad[numVelocidad - 1]= (error - errorAnterior)/tiempoMuestreo; //En la última posición del vector asignamos la ultima velocidad grabada
  sumaVelocidad=0;  //Inicializamos el sumatorio de las velocidades
  for(int i=0;i<numVelocidad;i++){   //Se recorre el vector realizando un sumatorio de todos sus valores
        sumaVelocidad=sumaVelocidad+vectorVelocidad[i];
  }
  return sumaVelocidad/numVelocidad;} // Se devuelve la velocidad media
```

Una vez obtenida la velocidad filtrada, solo tenemos que aplicar la siguiente línea de código para obtener la acción derivativa.

```
        D = Kd * velocidadMedia;
```

5.2.3 Control integral

La acción de control integral (I), da una salida del controlador que es proporcional al error acumulado, lo que implica que es un modo de controlar lento.

$$u(t) = Ki \int_0^t e(\tau)\, d\tau$$

La señal de control $u(t)$ tiene un valor diferente de cero cuando la señal de error $e(t)$ es cero. Por lo que se concluye que dada una referencia constante el error en régimen permanente es nulo.

Esta acción no ha sido posible incluirla en nuestro sistema debido a la existencia de la zona muerta por lo que los resultados conseguidos presentarán error en régimen permanente.

5.3 Ajuste de los parámetros del controlador

Como se comentó en el anterior capítulo, los parámetros del controlador se van a ajustar manualmente. En cada uno de los ensayos se sigue el siguiente procedimiento:

- Se establece como referencia una distancia de 13 centímetros durante los primeros 25 segundos.
- Posteriormente, se establece como referencia una distancia de 39 centímetros durante los siguientes 25 segundos.
- Por último se vuelve a la referencia de 13 centímetros hasta que la señal se estabilice.

En los ensayos que se van a realizar se van a visualizar tres señales:

- La referencia **"ref"** dibujada de color rojo.
- La señal de salida del sistema **"y"** (ángulo girado por el brazo del servo) dibujada de color azul.
- La señal de entrada del sistema **"u"** (posición de la bola)

Ensayo 1: Kp = 20 y Kd = 6000

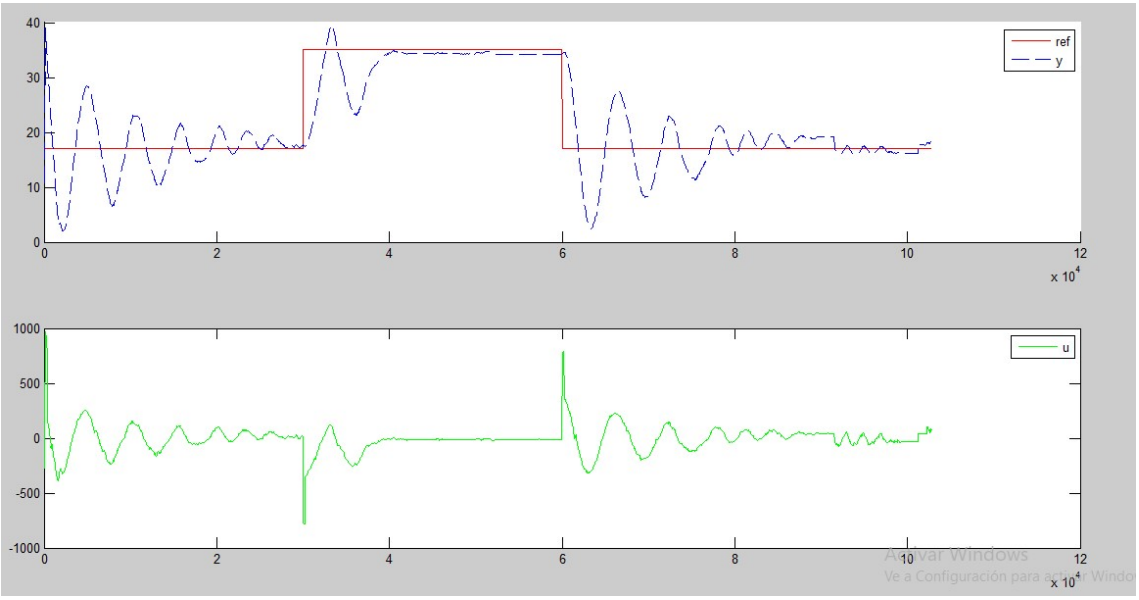


Figura 5-4. Gráficas ensayo 1.

Se puede observar en la gráfica:

- Tiempo de establecimiento muy elevado. En los 25 segundos que dura el primer tramo del ensayo ni siquiera llega a estabilizarse.
- Oscilaciones continuas y de gran amplitud.

En el próximo ensayo se va a incrementar notablemente la constante derivativa con el fin de evitar que los errores se hagan tan grandes como en este ensayo. También se modificará el término proporcional pero en menor medida.

Ensayo 2: Kp = 30 y Kd = 20000

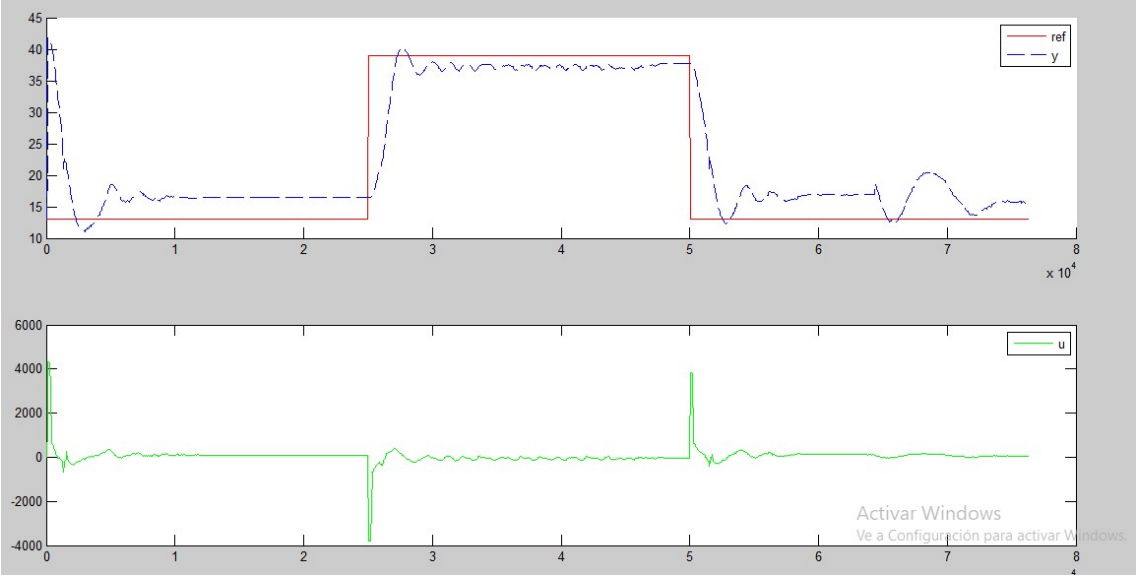


Figura 5-5. Gráficas ensayo 2.

En este caso:

- Las oscilaciones que se producen son cercanas al punto de referencia y por tanto de pequeña amplitud.
- El tiempo de establecimiento es mucho menor.
- Sin embargo, el utilizar una Kd tan elevada podría provocar inestabilidad en el sistema y el control se antoja muy agresivo.

En el último ensayo vamos a utilizar un compensador de zona muerta con el objetivo de disminuir Kp y Kd haciendo un control menos agresivo.

Ensayo 3: Kp = 15 y Kd = 10000 con compensador de zona muerta

Antes de ver los resultados del ensayo se va a explicar que es un compensador de zona muerta y porque vamos a utilizarlo.

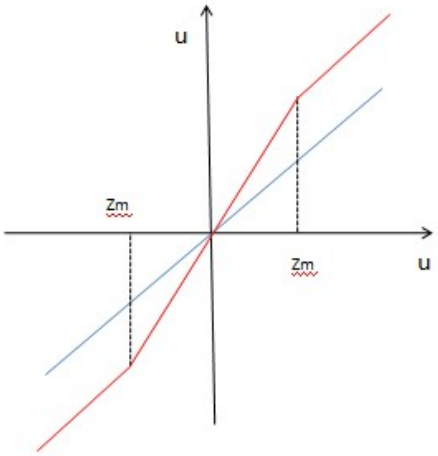


Figura 5-6. Compensador zona muerta.

En azul tenemos una entrada de ganancia unitaria y en rojo la entrada una vez aplicada el compensador de zona muerta siendo zm el valor de la zona muerta. En la práctica la forma de conseguir esta entrada es:

- Si $u > zm \quad \Longmapsto u = u + zm$
- Si $u < -zm \Longmapsto u = u - zm$
- Si– $zm < u < zm \Longmapsto Toma\ valores\ entre\ u - zm\ y\ u + zm$ de forma lineal.

Con esta entrada conseguimos un control más agresivo en los extremos y un control más suave en la zona muerta con lo que gracias a este compensador podemos reducir Kp.

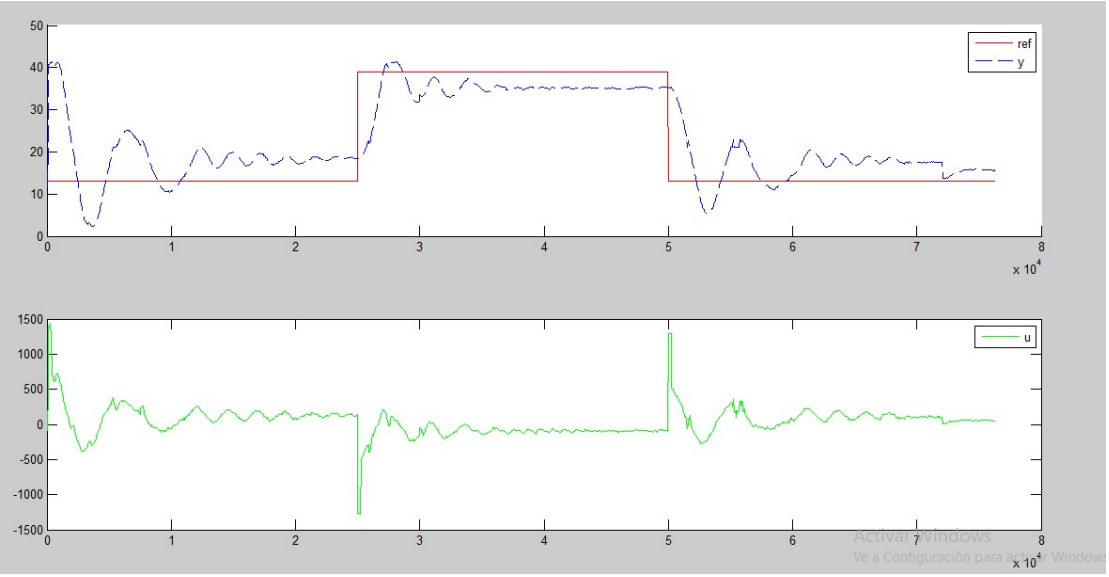


Figura 5-7. Gráficas ensayo 3.

En este caso tenemos un control mucho menos agresivo y la gráfica es aceptable. Sin embargo, presenta oscilaciones y tiempo de establecimiento algo mayores que en el anterior ensayo.

6. Conclusiones

6.1 Objetivos previos

En el objeto del proyecto se fijaron una serie de objetivos de los que se procede a hacer balance de ellos:

- **Facilidad de reproducción:** Se ha conseguido con tan solo ocho piezas diferentes y fácilmente imprimibles conformar la estructura de un Ball and Beam totalmente funcional.

- **Bajo coste:** El coste total del proyecto es de aproximadamente 60€, cifra muy inferior a los Ball and Beam que se encuentran en el mercado. El coste total se detalla en el Anexo C.

- **Hacer uso de software y hardware libre:** Se han utilizado en gran medida este tipo de software y hardware. Entre el software libre se ha hecho uso de Freecad, Cura y Fritzing. Como hardware libre se ha hecho uso de la placa Arduino Uno así como de los sensores y actuador utilizado.

6.2 Posibles mejoras y modificaciones

A pesar de haber cumplido los objetivos preestablecidos este equipo de prácticas podría ser mejorado en algunos aspectos:

- **Fiabilidad:** En ocasiones se producen algunas mediciones erróneas de los que afectan a la fiabilidad del equipo. Podría solucionarse con la adquisición de sensores de mejor calidad.

- **Precisión:** La existencia de zona muerta ha impedido la utilización de una acción integral de control por lo que existe un error en régimen permanente.

Por otro lado se podrían incorporar tiras LED unidas a nuestro raíl para poder visualizar el set-point e incluso seguir el recorrido de la bola o disponer de un servomotor que emita menos ruido.

Estas mejoras, sin embargo, aumentarían el coste del proyecto.

Anexo A. Impresión 3D

En la actualidad existen distintas tecnologías de impresión 3D en el mercado. Todas ellas a pesar de ser diferentes, poseen la característica en común de utilizar un proceso aditivo (con aporte de material), en el que se crea un objeto mediante las capas sucesivas de material. Esta característica difiere de los métodos tradicionales de mecanizado (técnicas de procesos sustractivos), que se basan principalmente en la eliminación de material por métodos como fresado, torneado, corte, etc.

A.1 Tecnologías existentes

SLA (Estereolitograficas): Consiste en la aplicación de un haz de luz ultravioleta a una resina líquida (contenida en un cubo) sensible a la luz. La luz UV va solidificando la resina capa por capa. La base que soporta la estructura se desplaza hacia abajo para que la luz vuelva a ejercer su acción sobre el nuevo baño, así hasta que el objeto alcance la forma deseada.

Con este método se consiguen piezas de altísima calidad, aunque presentan el inconveniente de que se desperdicia cierta cantidad de material en función del soporte que sea necesario fabricar.

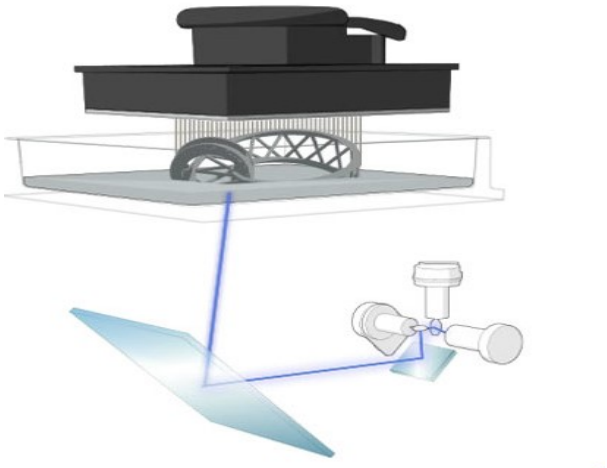


Figura I-1. Esquema funcionamiento impresión SLA

***Selective Laser Sintering* (SLS):** esta tecnología se nutre del láser para imprimir los objetos en 3D.

A pesar de tener ciertas similitudes con la tecnología SLA, ésta permite utilizar un gran número de materiales en polvo (cerámica, cristal, nylon, poliestireno, etc.). El láser impacta en el polvo, funde el material y se solidifica. Todo el material que no se utiliza se almacena en el mismo lugar donde inició la impresión por lo que, no se desperdicia nada.

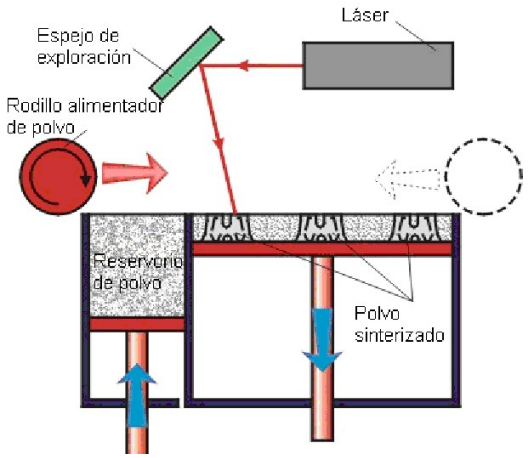


Figura I-2. Esquema funcionamiento impresión SLS

Inyección de tinta. La impresora crea el modelo de una capa a la vez mediante la difusión de una capa de polvo (yeso o resinas) y la impresión de un aglutinante en la sección transversal de la pieza, utilizando un proceso de inyección de tipo tinta. Esto se repite hasta que cada capa ha sido impresa. Esta tecnología permite la impresión de prototipos de varios colores, con salientes o voladizos y las piezas hechas de elastómeros. La fuerza de adhesión del polvo impreso se puede mejorar con impregnación de ceras o polímeros termoestables. Este es el sistema de impresión 3D más parecido a una impresora habitual (de tinta en folio), pero en lugar de inyectar gotas de tinta en el papel, inyectan capas de fotopolímero líquido que se pueden curar en la bandeja de construcción.

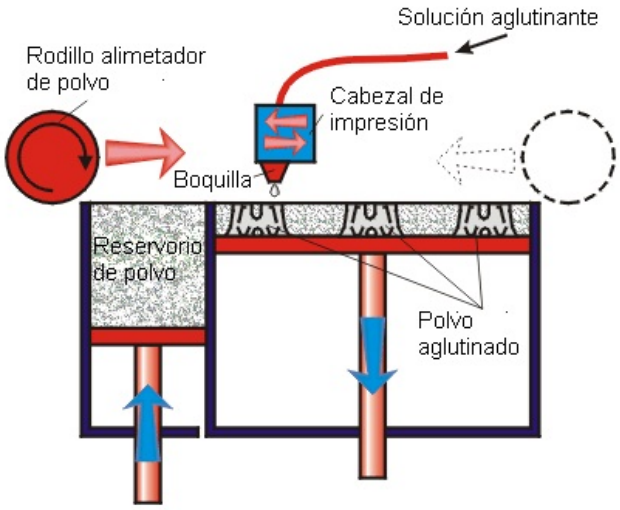


Figura I-3. Esquema funcionamiento impresión inyección de tinta

Deposición por extrusión (FDM o FFF): Se trata de la tecnología de impresión 3D más extendida entre el público general por su flexibilidad de uso y su relativa facilidad de construcción. La gran mayoría de impresoras que podemos encontrar en el mercado utilizan la tecnología de extrusión de material.

El proceso que usa se conoce como FDM (Fused Deposition Modeling) o Modelado por Deposición Fundida, pero como este término está registrado por la multinacional Stratasys Inc., se conoce también como FFF (Fused Filament Fabrication) o Fabricación por Filamento Fundido, para evitar problemas legales.

El proceso FFF utiliza un filamento de plástico que se enrolla en una bobina y se va desenrollado para suministrar material a una boquilla de extrusión que puede iniciar o detener el flujo de fundido. La boquilla se calienta para fundir el material y se puede mover en las tres dimensiones (x , y , z) mediante un mecanismo de control numérico que es controlado directamente mediante un software de fabricación asistido por ordenador (CAM). El modelo o pieza se produce por extrusión de pequeños aportes de material termoplástico para formar capas sobre una base o plataforma donde el material se endurece inmediatamente después de la extrusión desde la boquilla.

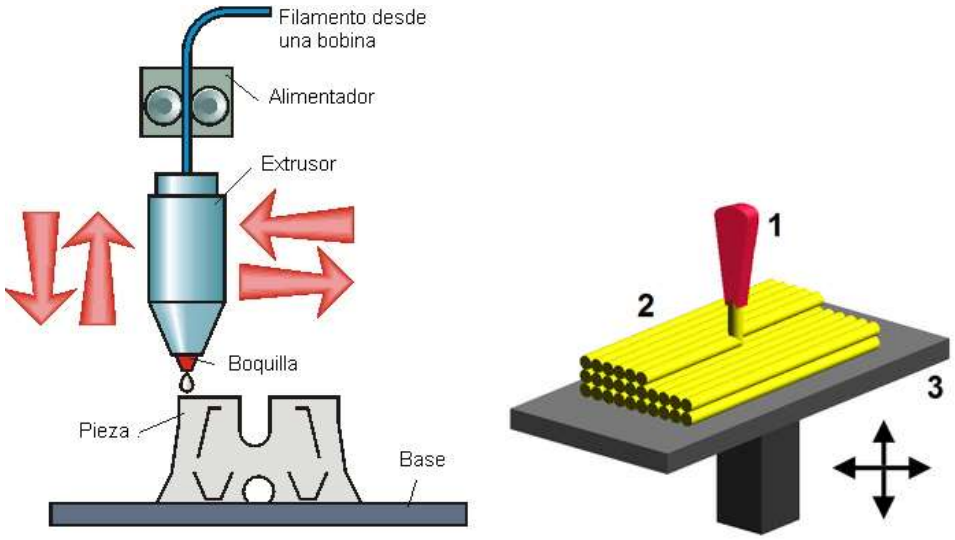


Figura I-4. Esquema funcionamiento impresión FFF

Esta tecnología, que será la utilizada para llevar a cabo este proyecto, es la ideal para la realización de prototipos ya presenta una ventaja considerable respecto a las otras tecnologías: su relación calidad-precio. Una impresora FFF tiene un rango de precios entre los 600€ (aquellas que vienen en un kit de montaje) hasta los cerca de 3000€ de una impresora ya montada y con más prestaciones. Las impresoras 3D que utilizan cualquiera de las otras tecnologías pueden costar entre los 5000€ (tecnología SLA) hasta más de 100.000€ (las de inyección de tinta).

A.2 Materiales

Como se ha comentado anteriormente el material utilizado por las impresoras 3D con tecnología FFF es un termoplástico. Dentro de los termoplásticos, encontramos distintos tipos que son aptos para la impresión 3D (ABS, HIPS, PLA, Nailon…) pero nos vamos a centrar en los dos más utilizados.

PLA: Este polímero (ácido poliláctico) es biodegradable ya que es fabricado a partir de materias primas renovables (almidón de maíz). Entre sus principales características destacan:

- Funde a 185 ºC aproximadamente.
- No presenta problemas de encogimiento (“warping”) al solidificar en la base, razón por la cual no necesita utilizar una plataforma calefactora.

- Gracias a su carácter no tóxico, es utilizado para la fabricación de objetos que entran en contacto con alimentos.
- La precisión de las piezas es mayor que utilizando material ABS.
- Es difícil de manipular una vez realizada la impresión dada su elevada velocidad de enfriamiento y solidificación.

ABS: este polímero (acrílico butadieno estireno), pertenece a la familia de los termoplásticos o plásticos térmicos, pero contienen una base de elastómeros a base de polibutadieno que los hace más flexible y resistente a los choques.

- Su temperatura de fusión está entre 200 y 250 ºC
- Se encoge en contacto con el aire, por lo que la plataforma de impresión se debe precalentar con el fin de evitar el despliegue de las piezas.
- Puede soportar temperaturas muy bajas (-20 ºC) y muy elevadas (80 ºC)
- Piezas con alta resistencia
- Permite fusionar piezas mediante procesos químicos (como por ejemplo, acetona)

En un principio, ambos materiales son válidos para la impresión del prototipo pero debido a la necesidad de fusionar algunas de las piezas impresas nos decantaremos por la utilización de ABS.

A.3 Proceso de impresión 3D

Figura I-5. Proceso impresión 3D

A.3.1 Diseño

El diseño de las piezas de la estructura del "Ball and Beam" se ha realizado con Freecad, programa de diseño asistido por ordenador (CAD). La elección de Freecad por delante de otros más conocidos (Solidworks, CATIA…) se debe a la naturaleza de software de código abierto que posee este programa. Otro factor positivo es que se trata de un programa gratuito con lo que cualquiera que desee hacer modificaciones o mejoras de las distintas piezas puede hacerlo.

Hay que destacar que no es lo mismo diseñar un objeto para ser visualizado en pantalla que para ser impreso. Debido a las limitaciones propias de la impresora 3D hay que tener presente una serie de recomendaciones:

- La pieza tiene que estar diseñada como un sólido cerrado.
- El tamaño de la pieza está condicionado por la zona de impresión. En nuestro caso, la pieza de mayor tamaño que se puede realizar es de 20 x 20 x 20 cm.
- El espesor mínimo de cualquier pared del objeto debe ser igual al diámetro de la boquilla del extrusor de la impresora 3D (en nuestro caso 0.4 mm). Aun así se recomienda que el espesor sea del doble de este diámetro.
- En el caso de que dos o más piezas deban encajar hay que hacer uso de holguras. Esto es debido a que aunque en la pantalla ambas midan lo mismo (lo que provocaría un encaje perfecto) en la realidad, debido a una pequeña dilatación del material al enfriarse, el encaje no será suficiente para que ambas piezas puedan unirse.
- En la medida de lo posible, realizar las piezas con una geometría sencilla y con el menor número de voladizos (ángulos con respecto a la horizontal mayores de 45º). En el caso de que el ángulo sea mayor del indicado y debido a que la impresión se realiza por capas, es necesario el uso de soportes. Estos soportes evitan que la deposición de material se realice "al aire" y tengo un lugar donde apoyarse. Como parte negativa, estos soportes pueden ser difíciles de eliminar posteriormente y conllevan un gasto adicional de material.

A.3.2 Generación archivo *.stl*

Una vez realizado el diseño de las distintas piezas se debe generar un archivo ".STL" de cada una de ellas. Se trata de un tipo de archivo informático de diseño asistido por computadora (CAD) que define la geometría de objetos 3D, excluyendo información como color, texturas o propiedades físicas.

Es el formato estándar para las tecnologías de fabricación aditiva. Utiliza una malla de triángulos cerrada para definir la forma de un objeto. Cuantos más pequeños son estos triángulos, mayor será la resolución del fichero final; el tamaño de los triángulos

está directamente proporcionado con el peso del fichero, por lo que habrá que llegar a una solución de compromiso entre la resolución y el peso del fichero.

Todos los archivos ".STL" que se han creado de las piezas de la estructura se incluyen en el CD adjunto al proyecto.

A.3.3 Software de laminado

Los archivos ".STL" deberán ser tratados en un software de laminado. Este programa realiza un corte micrométrico de la pieza, descomponiéndolo en cientos de capas cuidadosamente alineadas, para posteriormente y con la imagen generada de esta capa, generar el camino que el cabezal de impresión de nuestra impresora 3D realizará para depositar en el lugar correspondiente la cantidad de plástico necesaria para conformar el objeto.

Los actuales programas de laminado permiten diferentes opciones que influirán en el resultado final:

- Se puede configurar entre un rango de valores la altura de capa de la impresión. Se define como tal al paso existente entre una capa y otra. Cuanto menor sea este paso mayor será la resolución de la pieza. En nuestro caso se utilizará una altura de capa de (0.19 mm).
- Podemos colocar la pieza de la manera más adecuada. Se procurará colocar en aquella dirección en la que la pieza quede apoyada de forma estable sobre la superficie y que tenga que generar la menor cantidad de material de soporte. Además hay que tener en cuenta a la hora de colocar el objeto, que como se ha mencionado anteriormente, la resolución en el plano horizontal queda determinada por el diámetro de la boquilla (0.4mm) mientras que en el plano vertical será la altura de capa la que nos dará la resolución.
- Se puede elegir la cantidad de relleno que queremos que posea la pieza (densidad). Siendo posible desde piezas huecas a completamente macizas.

A.3.4 Generación archivo .*Gcode.*

Una vez laminada la pieza, obtendremos un fichero “.gcode”.

El G-code es el nombre que habitualmente recibe el lenguaje de programación más usado en control numérico, el cual posee múltiples implementaciones pero que se usa principalmente en automatización.

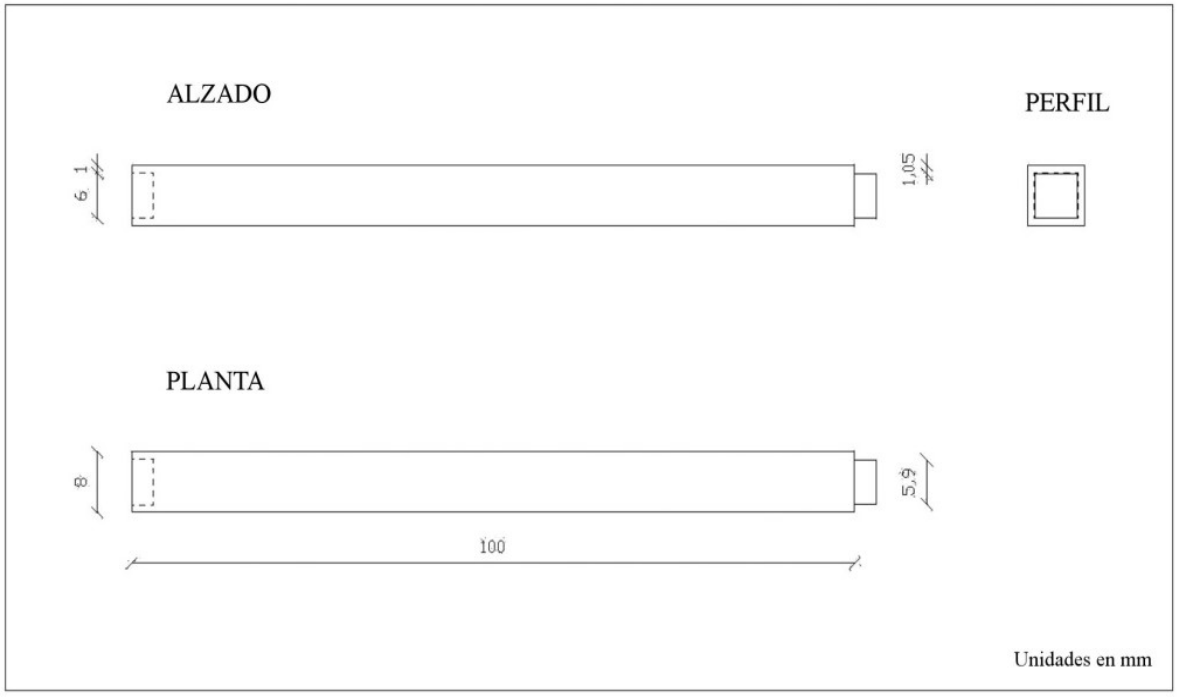
En términos generales, G-code es un lenguaje mediante el cual las personas pueden decir a máquinas herramientas controladas por computadora qué hacer y cómo hacerlo. Esos "qué" y "cómo" están definidos mayormente por instrucciones sobre adónde moverse, cuán rápido moverse y qué trayectoria seguir.

Por tanto, este será el fichero que tendremos que entregar a la impresora 3D para que comience a imprimir.

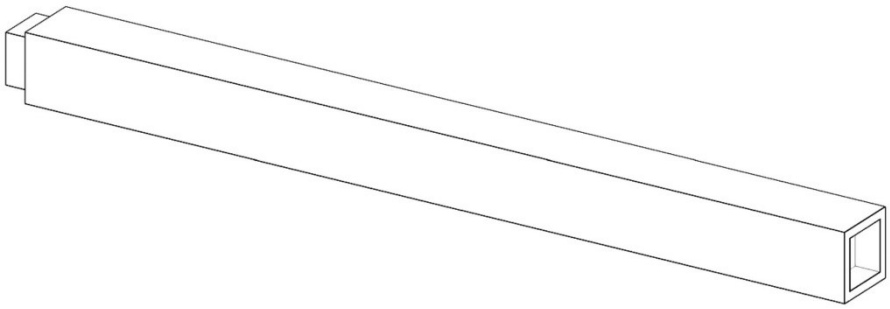
Anexo B. Planos

PIEZA A. SEGMENTO RAIL

PLANIMETRÍA. ESCALA 1:1

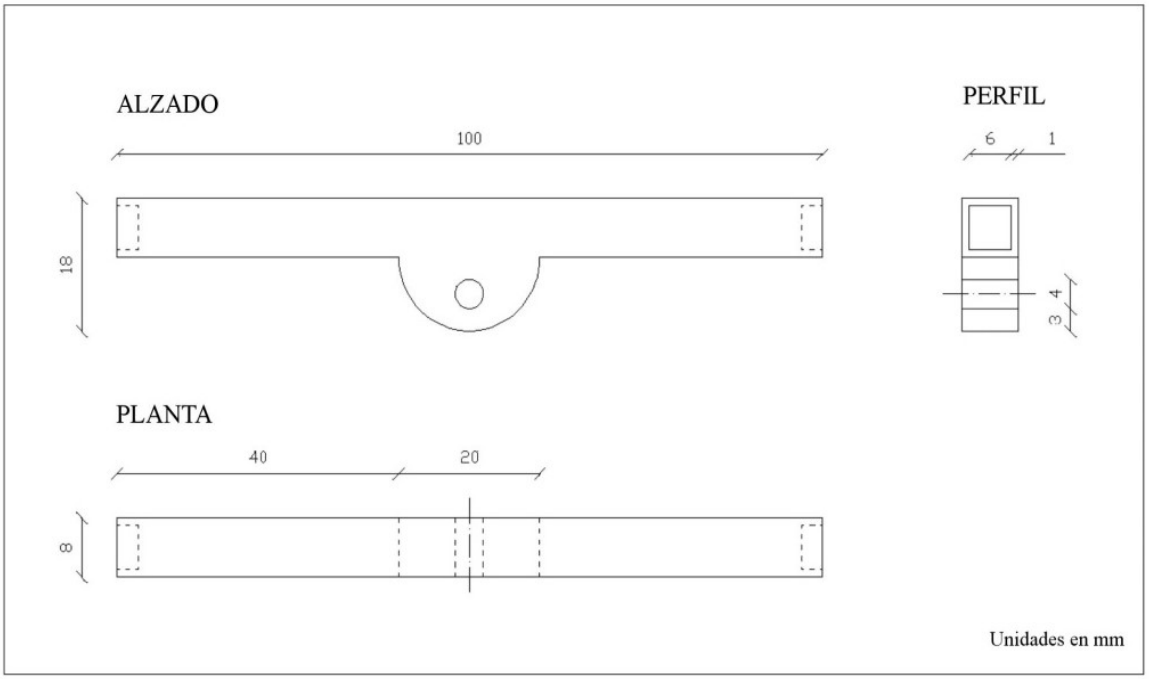


VOLUMETRÍA

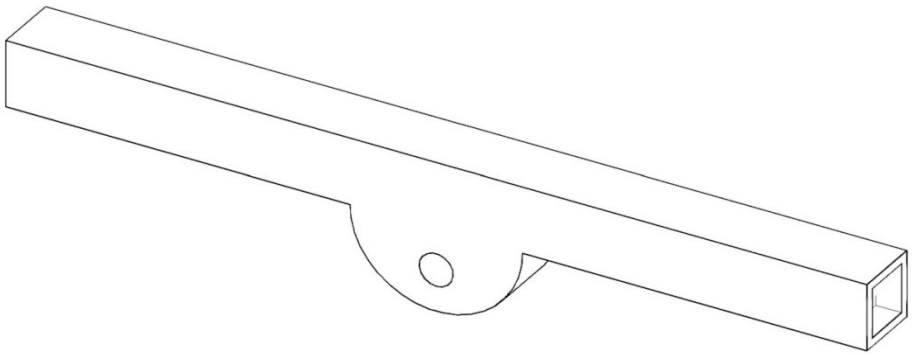


PIEZA B. SEGMENTO CENTRAL RAIL

PLANIMETRÍA. ESCALA 1:1

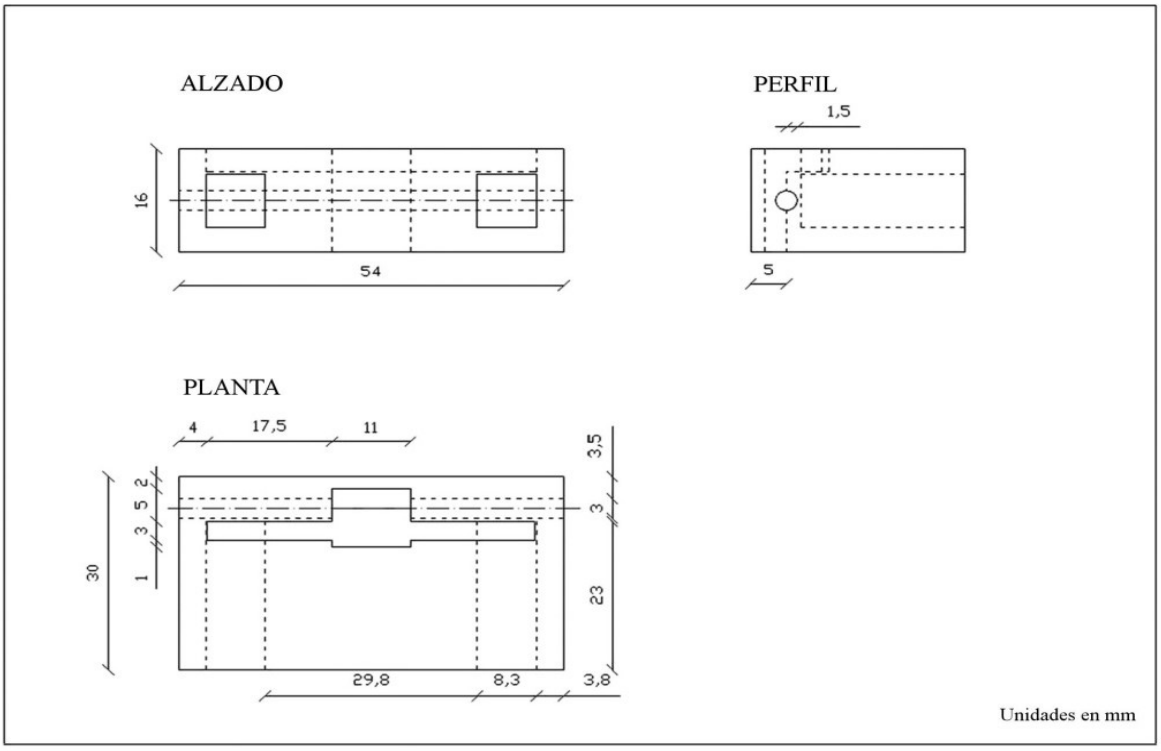


VOLUMETRÍA

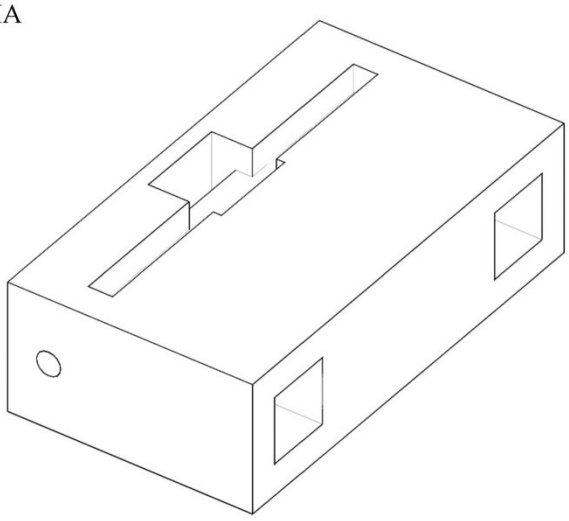


PIEZA C. SOPORTE SENSOR

PLANIMETRÍA. ESCALA 1:1

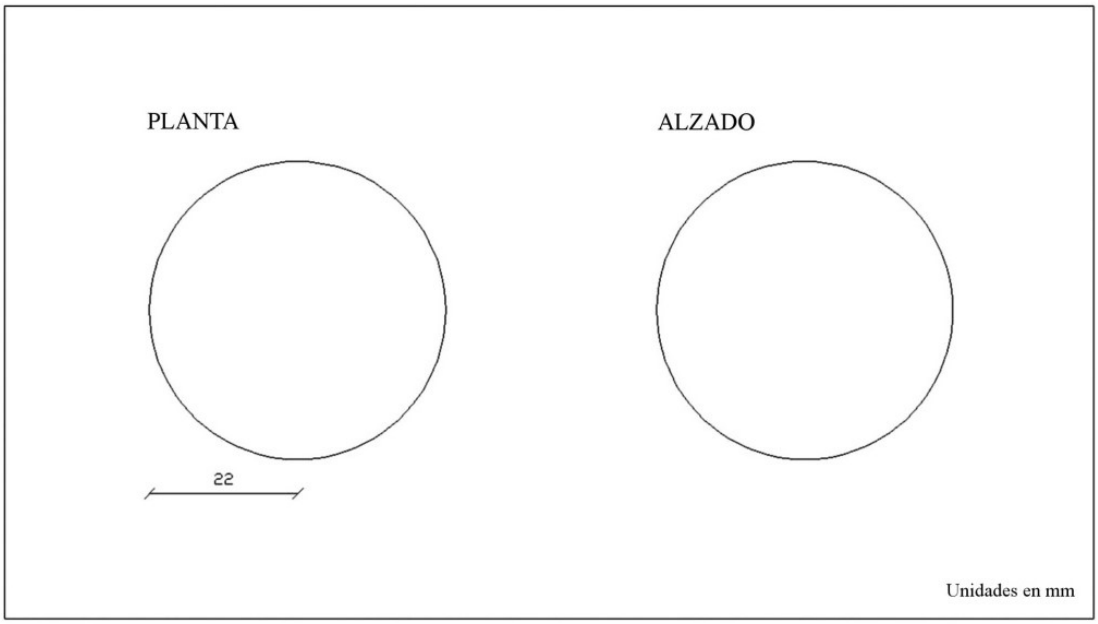


VOLUMETRÍA

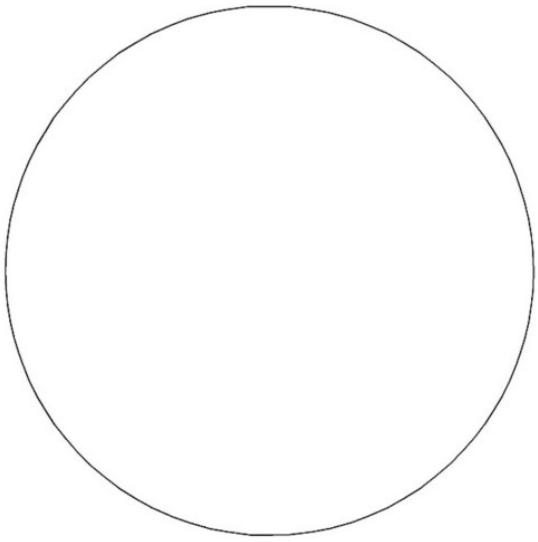


PIEZA D. ESFERA

PLANIMETRÍA. ESCALA 1:1

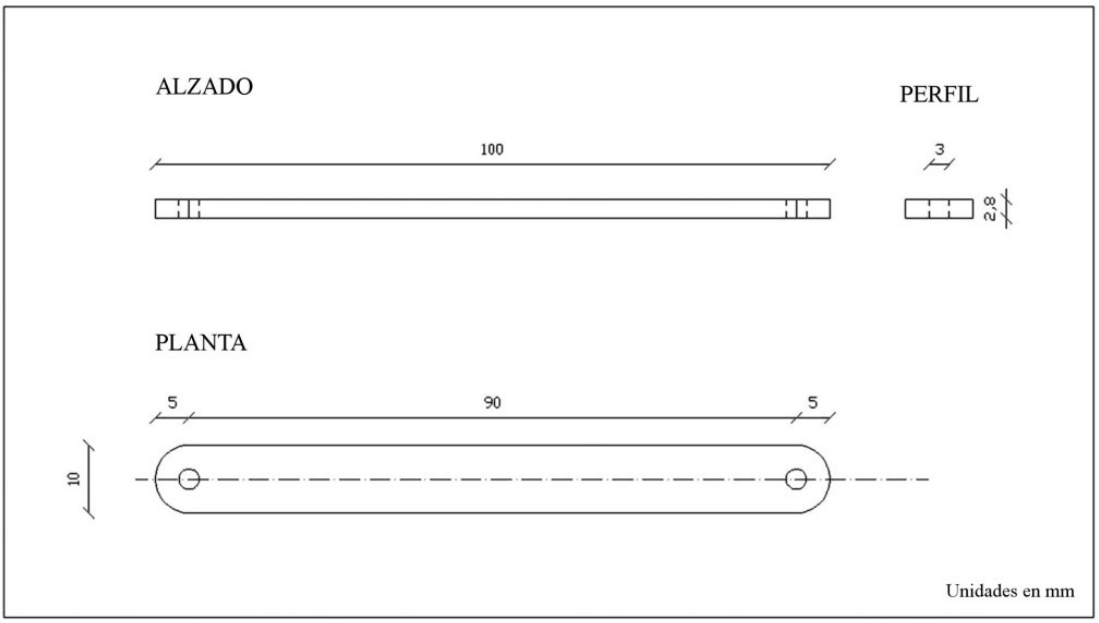


VOLUMETRÍA

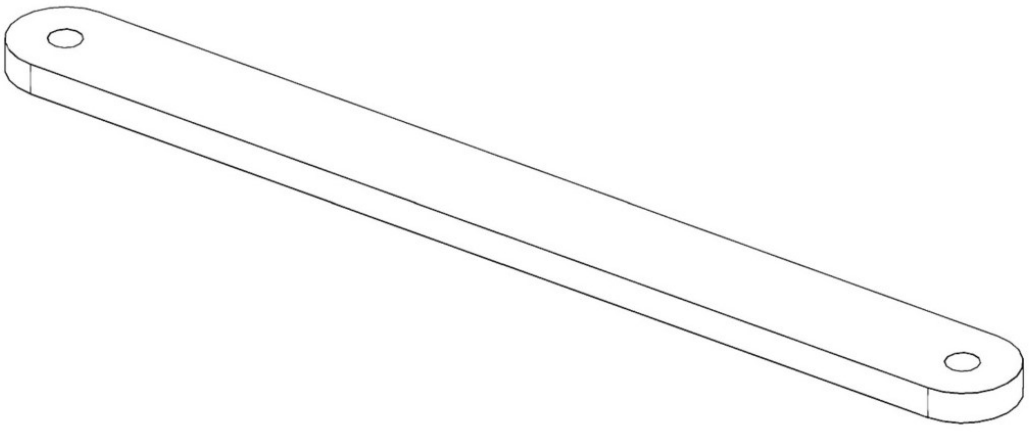


PIEZA E. BIELA

PLANIMETRÍA. ESCALA 1:1

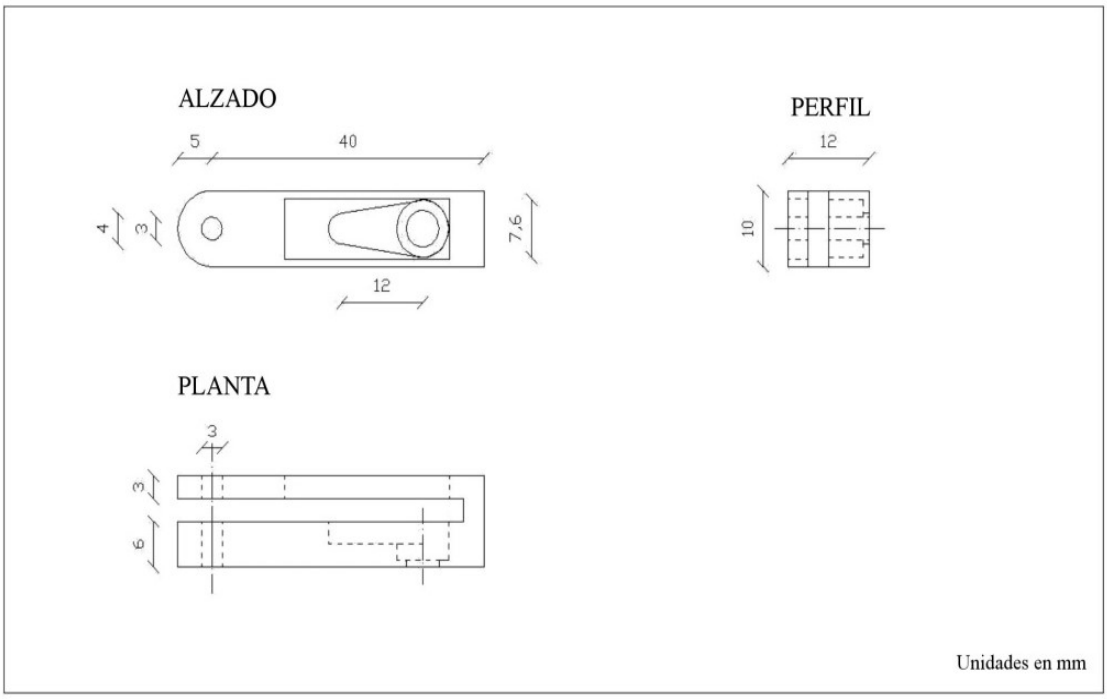


VOLUMETRÍA

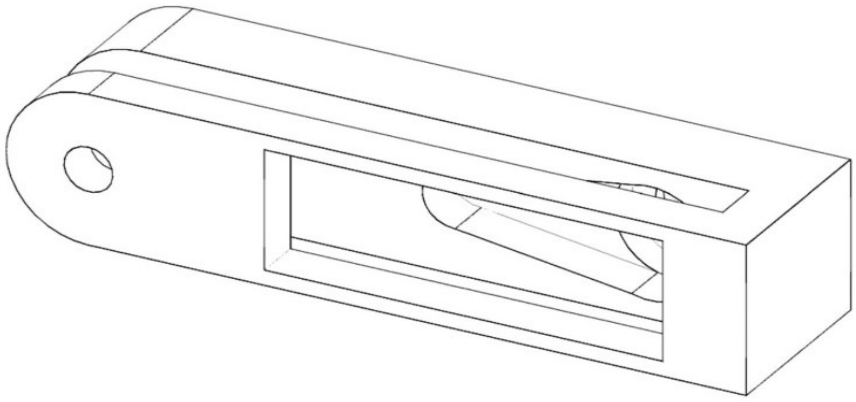


PIEZA F. MANIVELA

PLANIMETRÍA. ESCALA 1:1

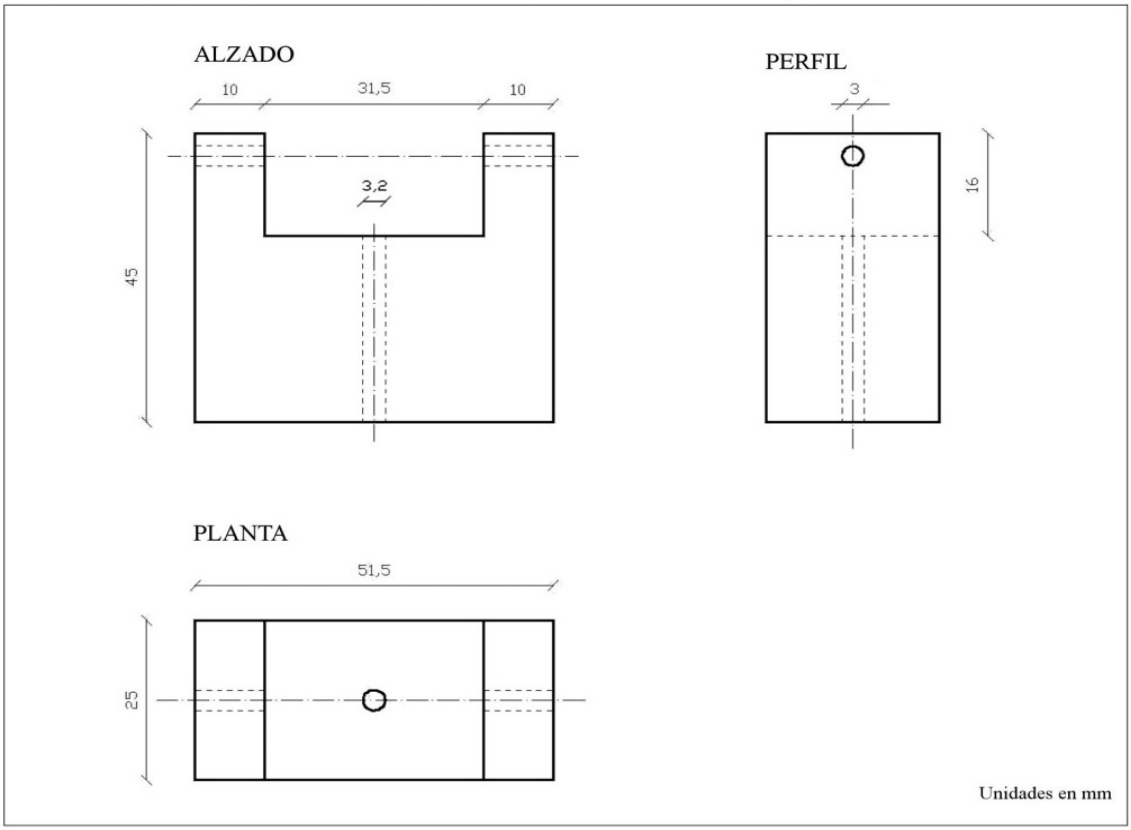


VOLUMETRÍA

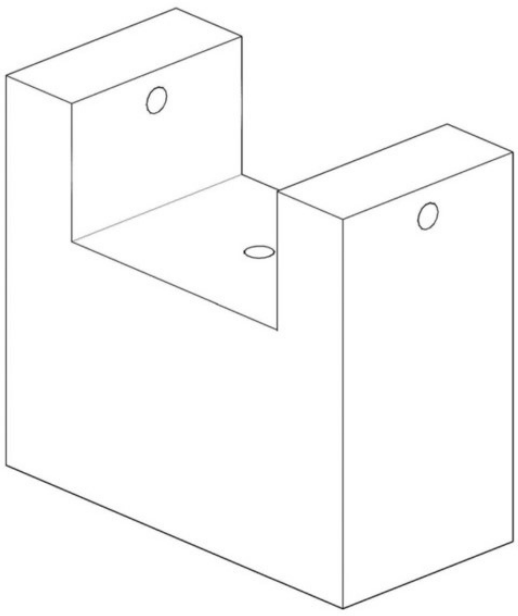


PIEZA G. SOPORTE CONTRAPESO

PLANIMETRÍA. ESCALA 1:1

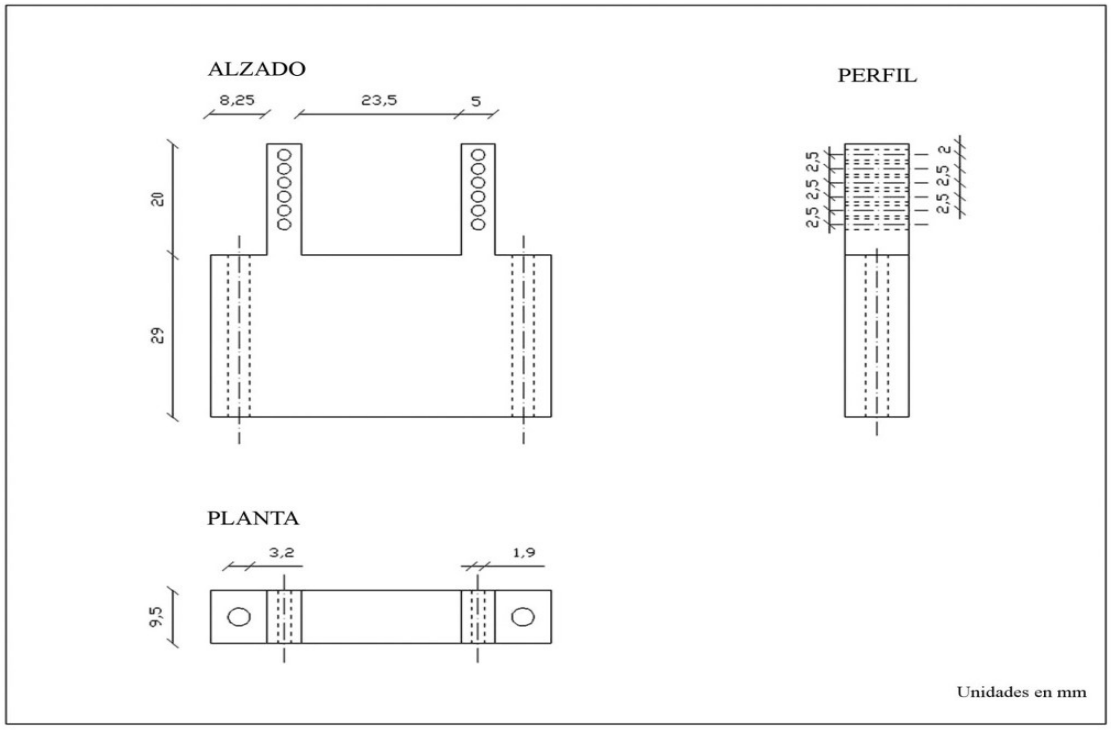


VOLUMETRÍA

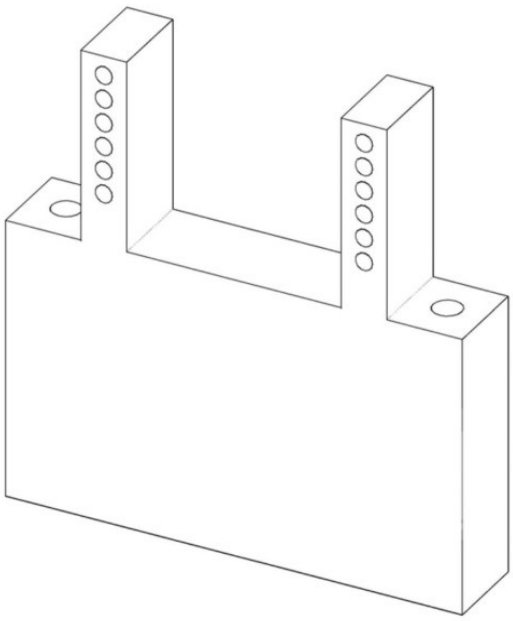


PIEZA H. SOPORTE SERVOMOTOR

PLANIMETRÍA. ESCALA 1:1



VOLUMETRÍA



Anexo C. Presupuesto

Uno de los objetivos de este proyecto es que el sistema fuese de bajo coste. Vamos a ver en detalle cual es el gasto en el que hay que incurrir para replicar este "Ball and Beam".

En primer lugar, se va a estudiar cual es el precio de las piezas impresas en 3D que forman la estructura del sistema. Teniendo en cuenta que una bobina de filamento ABS para la impresora utilizada (Zortrax M200) es de 800 gramos y su precio aproximado es 35€, podemos concluir que el coste por gramo de material es de 0.044 €/g.

Las piezas impresas y sus precios son los siguientes:

Piezas	**Gramos**	**Precio unitario**	**Cantidad**	**Precio**
Segmento raíl	6 g	0.26 €/ud	8 unidades	2.08 €
Segmento central raíl	7 g	0.31 €/ud	2 unidades	0.62 €
Soporte sensor	16 g	0.70 €/ud	2 unidades	1.40 €
Soporte voladizo	21 g	0.92 €/ud	1 unidad	0.92 €
Biela	5 g	0.22 €/ud	3 unidad	0.66 €
Manivela	4 g	0.18 €/ud	1 unidad	0.18 €
Soporte servomotor	11 g	0.49 €/ud	1 unidad	0.49 €
Semiesfera	10 g	0.44 €/ud	2 unidad	0.88 €
			Precio Total	**7.23 €**

Resto de materiales utilizados

Componentes	**Cantidad**	**Precio unitario**	**Precio total**
Arduino UNO rev3	1	19.95 €	19.95 €
Protoboard	1	2.50 €	2.50 €
Sensor ultrasónico HC-SR04	2	3.30 €	6.60 €
Servomotor SM-S2309S	1	12.00 €	12.00 €
Cables jumpers	15	0.20 €	3.00 €
Tablero contrachapado	1	3.95 €	3.95 €
Tornillos métrica 3	2	0.10 €	0.20 €
Tirafondos métrica 3	6	0.10 €	0.60 €
Fuente alimentación AC/DC	1	3.50 €	3.50 €
		Precio Total	**52.30 €**

Por tanto, el precio final sería de ***59,53€.***

Este precio incluso podría reducirse con los siguientes cambios:

- En caso de utilizar otro modelo impresora 3D (como la PrusaI3) , podemos encontrar que el kilogramo de material puede llegar a costar alrededor de 20€.

- Tanto Arduino como los sensores y actuadores utilizados podrían sustituirse por clones de los mismos. Esto es posible por la propia característica de "hardware libre" del universo Arduino, que permite que cualquier fabricante pueda disponer de toda la información necesaria para poder realizar una réplica de la placa como de sus componentes.

Sin embargo, estos cambios que pueden suponer un mínimo ahorro en la cuantía total, podrían ocasionar numerosos problemas desde el punto de vista del funcionamiento del conjunto.

Por último, indicar que en este presupuesto solo se tienen en cuenta los costes de los diferentes materiales ya que son en los que incurriría cualquiera que desee replicar este equipo de prácticas, siempre que se tenga el acceso a la utilización de una impresora 3D.

Anexo D. Código de programación

```
#include <Servo.h>

/* Parametros modificables del control */
float Kp = 15;//20
float Kd = 10000;//6000
float Ki = 0;
boolean filtroPID = 1; // Si filtroPID=1 activo, si filtroPID=0 inactivo.
float referencia = 13; // Para parar la pelota en el centro referencia =23.
float compensadorZM=400;

Servo miservo;
const int EchoI = 7;
const int TrigI = 8;
const int  EchoD = 11;
const int  TrigD = 12;
float Yreal;
float YI;
float YD;
float Y;
float U;
float P;
float I;
float D;
float rangoMaximo=23;
float rangoMinimo=2;
float YDanterior=0;
float YIanterior=0;
float alfa = 0.3;//0.25
float vectorVelocidad[]={0,0,0,0,0};
int numVelocidad=5;
float sumaVelocidad;
float velocidadMedia;
unsigned long tiempo;
unsigned long  tiempoAnterior = 0;
float tiempoMuestreo = 50;
float reposo = 1350;//1450
float error;
float errorAnterior=0;
float duracion;
float distancia;

void setup() {
  Serial.begin(115200);
  miservo.attach(9);
  pinMode (EchoD,INPUT);
  pinMode (TrigD,OUTPUT);
```

```
  pinMode (EchoI,INPUT);
  pinMode (TrigI,OUTPUT);
}

void loop() {
    static float errorAcumulado=0;
    static float errorAnteriorAcumulado=0;
    tiempo = millis();

   if (tiempo - tiempoAnterior >= tiempoMuestreo ){
        YIanterior = YI;
        YI = sonar(EchoI,TrigI);
        YI = alfa*YI + (1-alfa)*YIanterior;
        if (YI>rangoMaximo){YI=rangoMaximo;}
        if (YI<rangoMinimo){YI=rangoMinimo;}

        delay(5);

        YDanterior=YD;
        YD = sonar(EchoD,TrigD);
        YD= alfa*YD +(1-alfa)*YDanterior;
        if (YD>rangoMaximo){YD=rangoMaximo;}
        if (YD<rangoMinimo){YD=rangoMinimo;}
        Yreal = 44- YD;

        if (YI < rangoMaximo){Y=YI;}
        else { Y=Yreal; }
        error = Y - referencia;

        // control PID
       P = Kp * error;
       velocidadMedia = promedioVel();
       D = Kd * (velocidadMedia/tiempoMuestreo);
        U = P+D;

        if (filtroPID==1){
           if (U>400){U=U+400;}
           else{if (U<-400){U=U-400;}
            else{U=map(U,-400,400,U-400,U+400);}}

        miservo.writeMicroseconds(reposo+U);

        Serial.print(referencia);
        Serial.print(",");
        Serial.print(Y);
        Serial.print(",");
        Serial.print(U);
        Serial.print(",");
        Serial.println(tiempo);
```

```
      errorAnterior = error;
      tiempoAnterior = tiempo;
      errorAnteriorAcumulado=errorAcumulado;
    }
  }
float promedioVel(){
  for(int i=0; i<numVelocidad-1; i++){
    vectorVelocidad[i]=vectorVelocidad[i+1];
    }
  vectorVelocidad[numVelocidad - 1]= error - errorAnterior;
  sumaVelocidad=0;
  for(int i=0;i<numVelocidad;i++){
    sumaVelocidad=sumaVelocidad+vectorVelocidad[i];
  }
  return sumaVelocidad/numVelocidad;}

float sonar(int Echo, int Trig){
    digitalWrite(Trig, LOW);     /* Por cuestión de estabilización del sensor*/
    delayMicroseconds(4);
    digitalWrite(Trig, HIGH);    /* envío del pulso ultrasónico*/
    delayMicroseconds(10);
    digitalWrite(Trig,LOW);
    duracion = pulseIn(Echo,HIGH);
    return(duracion*0.017);
  }
```