Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT LANZAPELOTAS DE TENIS

Autor: Juan Francisco Moreno Aniceto

Tutor: Manuel Gil Ortega Linares

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

Sevilla, 2023

Trabajo Fin de Grado
Ingeniería de las Tecnologías Industriales

DISEÑO Y CONSTRUCCIÓN DE UN ROBOT LANZAPELOTAS DE TENIS

Autor:
Juan Francisco Moreno Aniceto

Tutor:
Manuel Gil Ortega Linares
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: DISEÑO Y CONSTRUCCIÓN DE UN ROBOT LANZAPELOTAS DE TENIS

Autor: Juan Francisco Moreno Aniceto

Tutor: Manuel Gil Ortega Linares

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

El objetivo de este proyecto ha sido realizar el diseño y construcción de un robot lanzapelotas de tenis capaz de lanzar pelotas en diferentes ángulos y con diferentes efectos dependiendo de la velocidad a la que gira cada motor.

El sistema de lanzamiento está basado en dos ruedas de fricción que giran en sentidos opuestos propulsadas por dos motores DC y comprimen la pelota con una fuerza determinada. Dicho sistema, es capaz de ajustar el ángulo vertical y horizontal de lanzamiento mediante dos mecanismos de engranajes análogos formados por una rueda conductora y una conducida.

Para lograr esto, se ha desarrollado el diseño 3D del robot mediante CAD (Computer Assisted Design), junto con la elección de componentes electrónicos que lo forman y su correspondiente integración.

De esta manera, se pueden comprobar los conocimientos obtenidos a lo largo del Grado en Ingeniería en Tecnologías Industriales, abarcando las ramas de mecánica, electrónica y automática.

Abstract

The objective of this project has been to design and build a tennis ball launcher robot capable of throwing balls at different angles and with different effects depending on the speed at which each motor rotates.

The launch system is based on two friction wheels that rotate in opposite directions propelled by two DC motors and compress the ball with a certain force. This system is capable of adjusting the vertical and horizontal angle of launch by means of two analogous gear mechanisms formed by a driving wheel and a driven wheel.

To achieve this, the 3D design of the robot using CAD (Computer Assisted Design) has been developed, together with the choice of electronic components that form it and their corresponding integration.

In this way, you can check the knowledge obtained throughout the Degree in Engineering in Industrial Technologies, covering the branches of mechanics, electronics and automation.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

AC	Corriente alterna
DC	Corrriente continua
RPM	Revoluciones Por Minuto
PWM	Modulación por ancho de pulsos
K_p	Constante Proporcional
K_i	Constante Integral
θ_1	Ángulo de lanzamiento horizontal
θ_2	Ángulo de lanzamiento vertical
m	Módulo
R	Radio de axoide
Z	Número dedientes
d	Distancia entre centros
η	Relación de transmisión
ω_1	Velocidad angular rueda inferior
ω_2	Velocidad angular rueda superior
D	Ciclo de trabajo
T	Período
ω_n	Frecuencia natual
ζ	Factor de amortiguamiento

1 INTRODUCCIÓN

Durante este proyecto se ha llevado a cabo el diseño y su posterior construcción de un robot lanzapelotas de tenis con el propósito de que pueda ser empleado en deportes de raqueta. El robot emplea un sistema de dos ruedas de fricción que giran en sentidos opuestos y comprimen la pelota para lanzarla. La velocidad de salida de la pelota será proporcional a la velocidad de giro de las ruedas y, además, podrá lanzarse la pelota con efecto topspin si la velocidad de giro de la rueda superior es mayor a la rueda inferior, o backspin en el caso contrario.

Figura 1. Diseño propuesto

Fuente: Elaboración Propia

Figura 2. Robot Real

Fuente: Elaboración Propia

El presente proyecto se ha centrado principalmente en el diseño de la parte mecánica, electrónica y control del robot, eligiéndose los componentes adecuados para su construcción, junto con la correspondiente integración y control del robot, el cuál se ha implementado mediante un microcontrolador Arduino y establecimiento de las referencias de posición y velocidad mediante comunicación Bluetooth entre el Arduino y una aplicación móvil que también ha sido desarrollada.

Para el control del robot, se ha estudiado el modelo de los motores DC en bucle abierto, obteniendo su correspondiente función de transferencia haciendo uso de Matlab y Simulink. Posteriormente se ha implementado el sistema en bucle cerrado, mediante un control PI cuyas variables proporcional e integral han sido obtenidas también mediante el empleo de Matlab a partir de la función de transferencia obtenida, logrando estabilizar la velocidad de los motores a la velocidad de referencia recibida por Bluetooth.

En una primera fase de diseño se ha desarrollado el prototipo hardware del robot haciendo uso de herramientas CAD (SolidWorks en este caso) tál y como se muestra en la figura 1. Este prototiopo se ha dividido en cuatro subsistemas con el fin de abordar cada parte por separado:

- Estructura

- Sistema de lanzamiento de pelotas

- Sistema de alimentación de pelotas

- Electrónica

En una segunda fase, se ha construido el robot (figura 2) y se han realizado pruebas experimentales para la obtención del modelo matemático de los motores DC y la obtención de las variables para sintonizar el control de velocidad. Posteriormente se analizan los resultados obtenidos para distintas velocidades de cada rueda, y se realizan pruebas de lanzamiento para dichas velocidades y distintos ángulos de lanzamiento.

Para el establecimiento de las referencias de posición y velocidad, se ha desarrollado una aplicación para conectarnos con el microcontrolador Arduino y comunicarnos con él por vía Bluetooth. Para ello, se ha empleado la herramienta MIT App Inventor, la cual nos permite desarrollar aplicaciones de una manera sencilla, compilarlas y descargarlas en nuestro dispositivo móvil en formato apk.

Finalmente, una vez logrado el control del robot se han llevado a cabo pruebas experimentales para determinar la velocidad necesaria para lanzar las pelotas a una distancia determinada.

2 DISEÑO MECÁNICO

Como se ha comentado anteriormente, el robot se ha dividido en cuatro subsitemas, correspondientes a la estructura, sistema de lanzamiento, sistema de alimentación de pelotas y a la electrónica. En este apartado describiremos en detalle cada uno de estos subsistemas junto a los componentes elegidos.

2.1. Estructura

La estructura es el soporte rígido de nuestro robot y sirve de base donde se colocarán el resto de los componentes. Con el fin de obtener una estructura lo más estable posible, se ha optado por usar perfiles cuadrados de aluminio con ranuras atornillables. De este modo se puede obtener una estructura sólida sin necesidad de soldar los perfiles, y en la que resulta sencillo instalar el resto de sitemas. Además, consta de una base de madera atornillada a la base inferior donde se colocará la electrónica, y un perfil de madera donde se apoya el sistema de lanzamiento con el fin de mitigar la fuerza de reacción ejercida por el mismo al lanzar la pelota.

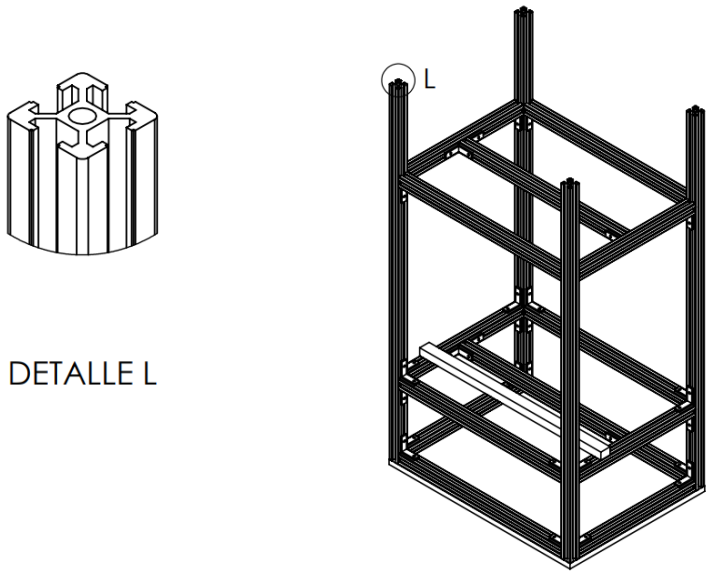


Figura 3. Estructura del robot

Fuente: Elaboración Propia

Se pueden ver más detalles en el Anexo 2, plano 1.3.

2.1.1 Lista de componentes de la estructura

Los componentes empleados para la estructura son los siguientes:

Nº	Componente	Dimensiones (cm)	Cantidad (uds)
1	Base de madera	39x29x1	1
2	Perfil de aluminio ranurado 2020	2x2x35	8

3	Perfil de aluminio ranurado 2020	2x2x25	6
4	Perfil de aluminio ranurado 2020	2x2x70	4
5	Perfil de madera	2x1.5x39	1
7	Soportes en forma de L + Tornillos		44

Tabla 1. Tabla de componentes de la estructura

Fuente: Elaboración Propia

Una vez establecida cómo será la estructura del robot, estamos en la disposición de seguir analizando el resto de los sistemas que van montados sobre esta.

2.2. Sistema de lanzamiento de pelotas

Este es el sistema principal de nuestro robot, el cual consta del mecanismo capaz de lanzar las pelotas a una velocidad de salida determinada, con o sin efecto y con un ángulo de lanzamiento determinado. Este sistema se muestra completo en la siguiente figura:

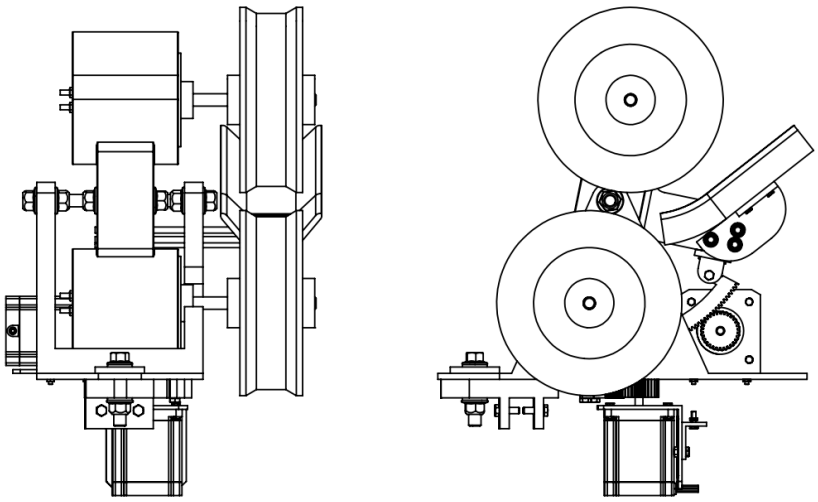


Figura 4. Sistema de lanzamiento de pelotas

Fuente: Elaboración Propia

Para una mejor comprensión, se ha dividido el sistema en dos partes las cuales se describen detalladamente en las siguientes secciones.

Cada parte la estudiamos como un mecanismo independiente, cada uno con un grado de libertad, compuesto por un par de engranajes con su correspondiente motor paso a paso que los acciona y nos permite posicionar los ángulos de lanzamiento en la posición deseada.

Empezaremos explicando de forma general el mecanismo de transmisión de movimiento empleado para posicionar los ángulos de lanzamiento. Posteriormente, analizaremos el mecanismo correspondiente al ángulo de lanzamiento vertical (θ_1) y seguidamente el mecanismo para el ángulo de lanzamiento horizontal (θ_2)

Además, estudiaremos el sistema de ruedas de fricción, el cuál consta de dos motores DC que giran en sentido

opuesto y a una velocidad determinada y que son los encargados de propulsar las ruedas encargadas de comprimir y lanzar las pelotas de la manera correcta.

Antes de empezar a estudiar los mecanismos que forman parte del sistema de lanzamiento, daremos una visión general de los elementos que lo forman. Para ello, se ha divido el ensamblaje en dos partes: la base, que es la que gira el conjunto completo y va apoyada sobre la estructura mediante un soporte como se detalla en la figura 5, y la barra sobre la cuál se montan los motores con sus respectivas ruedas, tal y como se muestra en la figura 6.

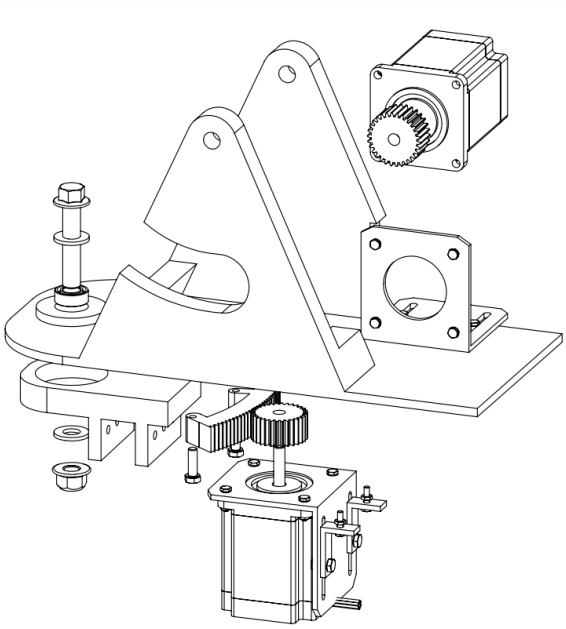


Figura 5. Sistema lanzapelotas – 1

Fuente: Elaboración Propia

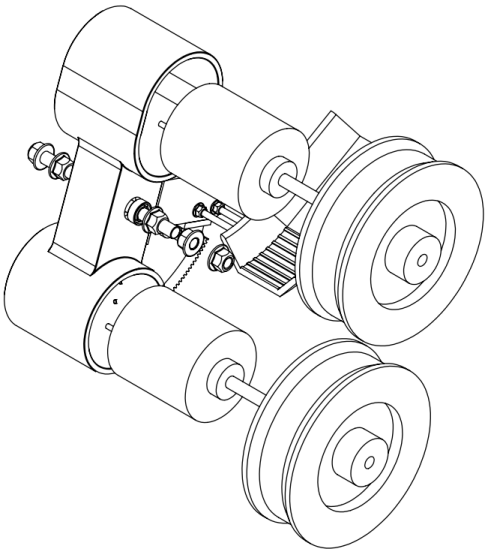


Figura 6. Sistema lanzapelotas – 2

Fuente: Elaboración Propia

Como se puede observar, cada eje de giro se compone de una varilla de acero (M8), tuercas y arandelas que fijan la posición entre la base y el soporte que lo une a la estructura, y entre la base y la barra sobre la que van posicionados los motores DC. A su vez, existen rodamientos de bolas que habilitan el giro entre la varilla y el elemento giratorio.

2.1.2 Mecanismos de transmisión del movimiento.

Para conseguir posicionar nuestro robot en los ángulos de lanzamiento deseados, haremos uso de dos mecanismos de transmisión del movimiento (engranajes), uno para el ángulo de lanzamiento vertical y otro para el horizontal.

Estos mecanismos consisten cada uno en un par de engranajes rectos con perfil de evolvente.

En un engranaje hay ruedas conductoras y ruedas conducidas. A veces una misma rueda puede jugar ambos papeles. En un engranaje que conste de dos ruedas, como el mostrado en la Figura 7 (conductora la 2 y conducida la 3) en la rueda conductora el par motor, Mmot, tiene el mismo signo que la velocidad angular (potencia positiva, 'entra' en el sistema). En la rueda conducida el par resistente, Mres, tiene signo contrario que la velocidad angular (potencia negativa, 'sale' del sistema). Otras fuerzas y pares que aparecen en la figura son las fuerzas de contacto Fc2 y Fc3 aplicadas en las ruedas 2 y 3, respectivamente, y los pares de dichas fuerzas respecto a los centros de las ruedas, MFc2 y MFc3.

En virtud del equilibrio dinámico de pares (se supone que las ruedas giran a velocidad angular constante, por lo que no hay pares de las fuerzas de inercia) el par de las fuerzas de contacto debe ser contrario a la velocidad angular en la rueda conductora y con el mismo signo en la conducida, para compensar los pares externos.

En nuestro caso, la rueda conductora será la propulsada por el motor paso a paso y la conducida será el arco de engranaje conectado a la barra donde van situados los motores DC de nuestro robot.

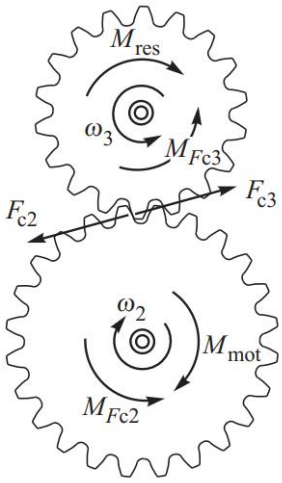


Figura 7. Ruedas conductora y conducida en un engranaje

Fuente: Referencia [1]

Por otro lado, los dientes de nuestros engranajes tienen la forma por ambos flancos de la función matemática evolvente de círculo.

La función evolvente es la trayectoria que describe el extremo de un hilo cuando se desenrolla de un carrete manteniendo el hilo tenso, como se ilustra en la Figura 8.

Es importante destacar que la única magnitud paramétrica que caracteriza a la evolvente es el radio del 'carrete' que se denomina circunferencia base, con radio RB.

La Figura 9 muestra un punto A cualquiera perteneciente a la curva evolvente con coordenadas polares r y ϕ.

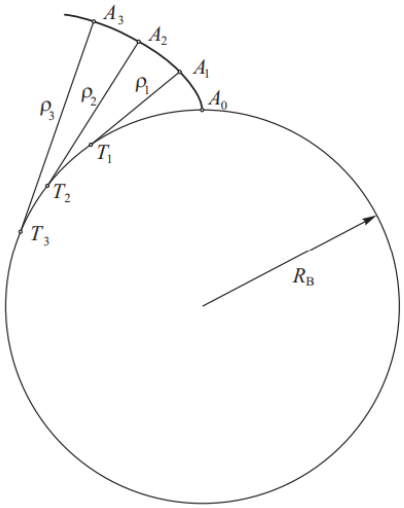


Figura 8. Perfil de evolvente

Fuente: Referencia [1]

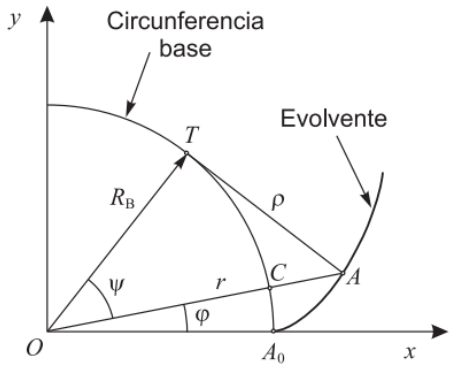


Figura 9. Descripción paramétrica del perfil de evolvente

Fuente: Referencia [1]

Analizando dicha figura se obtiene la función paramétrica del perfil de evolvente:

$$\begin{cases} r = \frac{R_B}{cos\psi} \\ \varphi = tg\psi - \psi \end{cases} \quad , \qquad \psi \in \left[\ 0 \qquad \pi/2\ \right).$$

Fuente: Referencia [1]

Dando distintos valores al parámetro ψ se obtienen las coordenadas de distintos puntos de la evolvente. El punto A0, el único perteneciente a la circunferencia base, se obtiene para ψ = 0. La expresión matemática de ϕ en función de ψ dada en la ecuaciónes una función trigonométrica, como lo son el seno o el coseno, que se denomina evolvente del ángulo y se expresa como Ev(ψ) = tgψ-ψ. Por tanto, la relación entre ϕ y ψ para la curva evolvente se puede expresar simplemente como ϕ = Ev(ψ).

Por otro lado, el módulo de una rueda dentada es una medida de la dimensión de sus dientes. Para que dos ruedas dentadas puedan engranar deben tener el mismo tamaño de dientes, es decir, deben tener el mismo

módulo. Dicho esto, las características, o ventajas, de los engranajes con perfil de evolventes son:

-El ángulo de empuje es constante durante el proceso de engrane. Esto implica menos ruido y vibración en la transmisión.

- Cualquier dentadura de perfil de evolvente puede engranar con otra del mismo módulo, independientemente de la distancia entre centros. Esto es conveniente porque no es necesaria mucha precisión en el posicionamiento de los ejes de las ruedas en la máquina. Esta distancia sí tiene influencia en la holgura entre dientes.

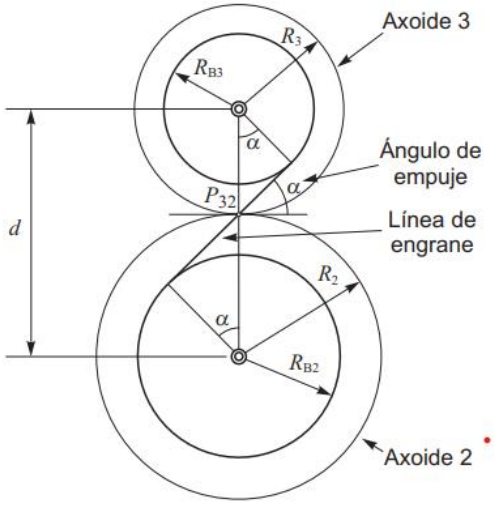


Figura 10. Ángulo de empueje y distancia entre centros

Fuente: Referencia [1]

2.1.3 Ángulo de lanzamiento vertical

En función del ángulo de lanzamiento vertical (θ_1) la pelota saldrá más inclinada o menos. Si el ángulo de lanzamietno es elevado y la velocidad de salida de la pelota es la apropiada, obtendremos un disparo tipo 'globo', mientras que si es bajo el disparo será más raso.

Por ello, juega un papel importante en función del tipo de lanzamiento que queramos entrenar.

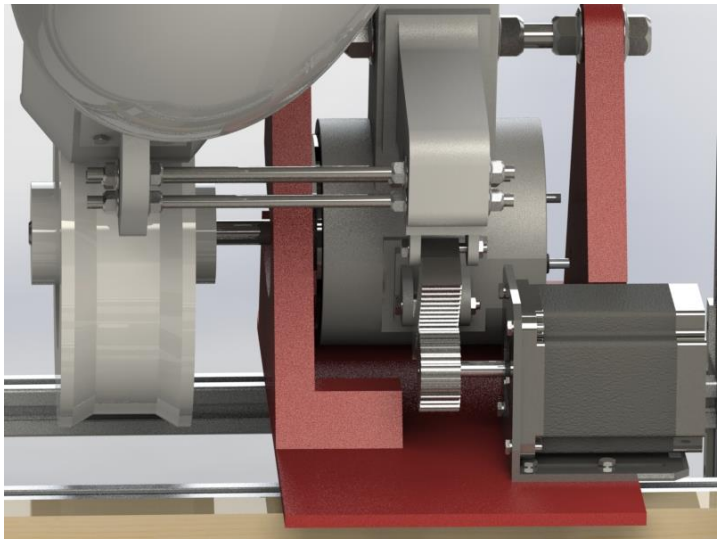


Figura 11. Mecanismo para Ángulo de lanzamiento vertical

Fuente: Elaboración Propia

Para nuestro robot, el rango de movimiento de este ángulo será aproximadamente entre 0 y 20º.

Centrándonos en el mecanismo que emplearemos para posicionar dicho ángulo, podemos apreciar en la figura 12 el diseño propuesto. Como ya se ha explicado en la sección anterior, el mecanismo consiste en un par de engranajes rectos con perfil de evolvente. En nuestro caso, el engranaje correspondiente a la rueda conducida es solo un arco del engranaje total, pero las características que lo definen son las mismas.

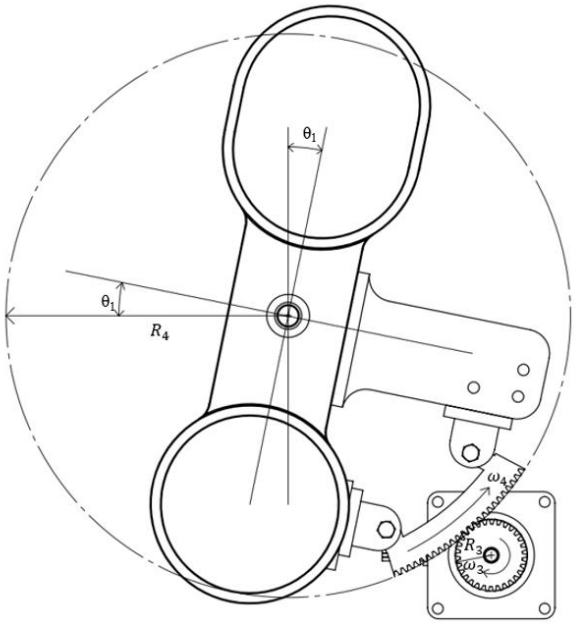


Figura 12. Mecanismo para Ángulo de lanzamiento vertical – Croquis

Fuente: Elaboración Propia

Para el diseño de los engranajes, partimos de una distancia entre centros de 138.6 mm.

Se ha definido el engranaje correspondiente a la rueda conductora con módulo normalizado de valor 1 y 30 dientes.

Con estos datos podemos calcular el radio de axoide:

$$m = \frac{2R_3}{Z_3} \qquad R_3 = \frac{1\ x\ 30}{2} = 15\ mm$$

Y conociendo la distancia entre ejes y adoptando la condición de engrane (mismo módulo), podemos obtener el radio y los dientes de la rueda conducida:

$$d = R_3 + R_4 \quad R_4 = d - \ R_3 = 138.6 - 15 = 123.6\ mm$$

$$m = \frac{2R_4}{Z_4} \qquad Z_4 = 2\ x\ 123.6 = 247$$

Sin embargo, para la rueda conductora solo nos quedamos con un arco de 24 dientes que va fijado a la barra en la que se instalan los motores DC de nuestro robot.

Una vez obtenidos estos datos, la relación de transmisión que se obtiene es la siguiente:

$$\eta = \frac{\omega_4}{\omega_3} = \frac{R_3}{R_4} = \frac{Z_3}{Z_4} = \frac{30}{247} = 0.12$$

Esto quiere decir que, por cada vuelta completa de la rueda conductora la rueda conducida girará 43.2º.

Suponiendo una posición inicial del mecanismo con $\theta_1 = 0$ grados, podemos establecer los grados que ha de girar la rueda conductora para que la rueda conducida llegue a 40 grados:

Rueda Conductora (º)	θ_1 (º)
0	0
41.6	5
83.3	10
125	15
166.6	20

Tabla 2. Comparación entre rueda conductora y θ_1

Fuente: Elaboración Propia

Como se puede ver en la tabla, para conseguir posicionar este ángulo a 20º necesitamos un poco menos de media vuelta de la rueda conductora.

2.1.4 Ángulo de lanzamiento horizontal

El mecanismo empleado para el posicionamiento del ángulo de leanzamiento horizontal (θ_2) es el mismo que para el ángulo de lanzamiento vertical, habiéndose empleado engranajes iguales ($m = 1, R_3 = R_5, R_6 = R_4$) colocados de forma que la distancia entre ejes sea la misma.

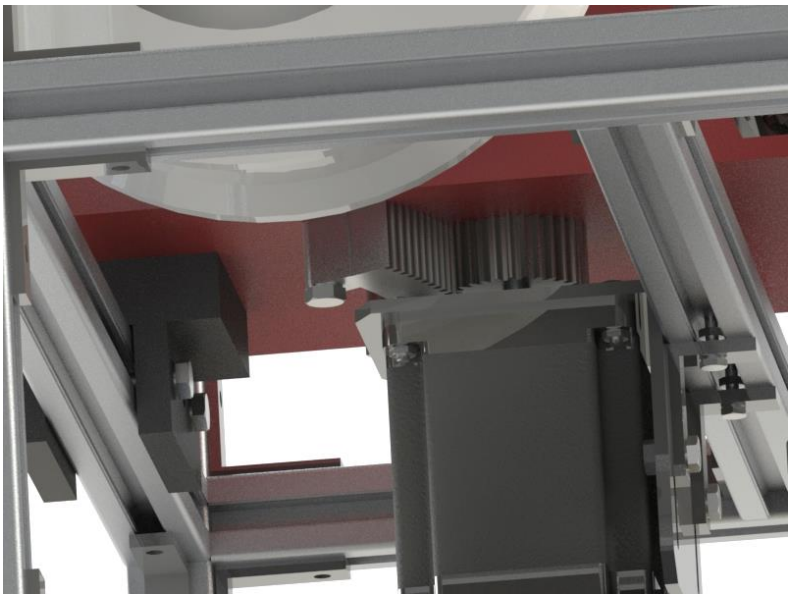


Figura 13. Mecanismo para Ángulo de lanzamiento horizontal

Fuente: Elaboración Propia

Por tanto, los engranajes comparten relación de transmisión. La diferencia será que en este caso el rango de funcionamiento es distinto, entre -15º y 15º, siendo el valor positivo el mostrado en la figura 2.2.3.

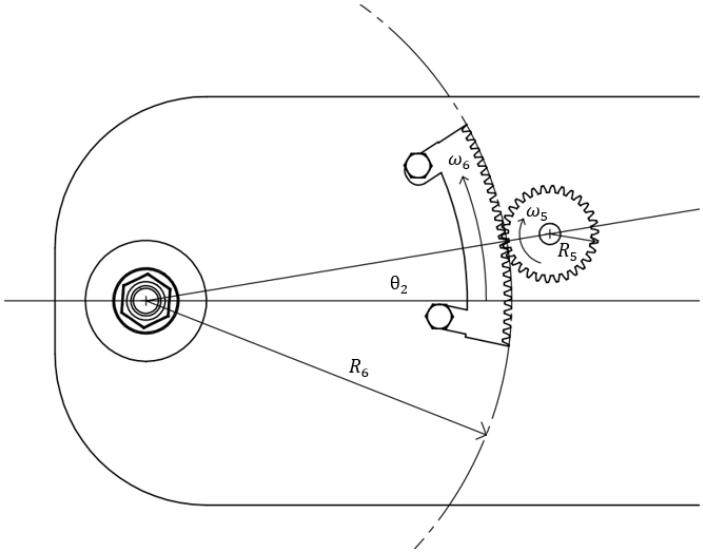


Figura 14. Mecanismo para Ángulo de lanzamiento horizontal – Croquis

Fuente: Elaboración Propia

Suponiendo una posición inicial del mecanismo con θ_2 = 0 grados, tal y como se muestra en la figura anterior, podemos establecer los grados que ha de girar la rueda conductora para que la rueda conducida llegue a -15º y 15º, como se muestra en la tabla siguiente:

Rueda Conductora (º)	θ_2 (º)
-125	-15
-83.3	-10
-41.6	-5
0	0
41.6	5
83.3	10
125	15

Tabla 3. Comparación entre rueda conductora y θ_2

Fuente: Elaboración Propia

El tipo de motor empleado para accionar las ruedas conductoras de ambos mecanismos es el motor paso a paso.

El motor paso a paso (Stepper) conocido también como motor de pasos es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es

capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control. El motor paso a paso se comporta de la misma manera que un conversor digital-analógico (D/A) y puede ser gobernado por impulsos procedentes de sistemas digitales. Este motor presenta las ventajas de tener precisión y repetitividad en cuanto al posicionamiento.

Dichos impulsos digitales junto con la alimentación de cada motor serán proporcionados por dos drivers de control de motores paso a paso, los cuales a su vez reciben la referencia desde el Arduino. El modelo elegido para los drivers es el DM542, como se verá en el apartado 4.1.4. Asimismo, el estudio entre la relación de impulsos (pasos) y los ángulos girados por cada motor para alcanzar las posiciones deseadas, se verá en el apartado 4.1.3.

En este caso concreto, los motores paso a paso elegidos son los NEMA 23, cuyas características son las siguientes:

Figura 15. Motor NEMA 23

Fuente: [2]

Especificaciones eléctricas		Especificaciones físicas	
Número de pieza del fabricante	23HS30-2804S	Tamaño del bastidor	57 x 57 mm
Tipo de motor	Bipolar paso a paso	Longitud del cuerpo	76 mm
Ángulo de paso	1,8	Diámetro del eje	6,35 mm
Par de mantenimiento	1.9 nm (7,62 kg)	Longitud del eje	21 mm
Corriente nominal/fase	2.8 A	Número de conectores	4
Fase de resistencia	1.13 ohms	Longitud del conector	500 mm
Inductancia	5,4 mH +/-20% (1 kHz)	Peso	1,2 kg

Tabla 4. Especificaciones del motor para a paso NEMA 23

Fuente: [2]

Una vez analizados los engranajes que nos permitirán posicionar los ángulos de lanzamiento en la posición deseada, procedemos a estudiar ahora el sistema de ruedas de fricción.

2.1.5 Sistema de ruedas de fricción

Este sistema consiste en dos ruedas situadas una encima de la otra a una distancia determinada. Ambas ruedas giran en sentido opuesto a una velocidad determinada, que no tiene por qué ser la misma.

La fuerza empleada en la pelota se divide de dos formas: una parte de ella pasa en la tranlación o desplazamiento de la pelota hacia adelante y la otra parte pasa en la rotación de la bola sobre sí misma. La cantidad de efecto que lleva la pelota, la determina la velocidad del giro. La pelota plana no lleva rotación, con lo cuál es el golpe más fuerte o rápido de todos.

En nuestro caso, la velocidad de salida de la pelota será proporcional a la velocidad de giro de las ruedas. Si ambas ruedas giran a la misma velocidad, entonces la pelota no tendrá rotación. A este tipo de lanzamiento se denomina lanzamiento plano. En cambio, podrá lanzarse la pelota con efecto topspin (girando hacia delante en el sentido de desplazamiento) si la velocidad de giro de la rueda superior es mayor a la rueda inferior. A este tipo de lanzamiento se denomina lanzamiento Liftado. Si la velocidad de giro de la rueda inferior es mayor a la de la rueda inferior, entonces el efecto resultante será el backspin (girando hacia atrás en el sentido de desplazamiento), tal y como se ilustra en la figura 16. Este último tipo de lanzamiento, se denomina lanzamiento Cortado.

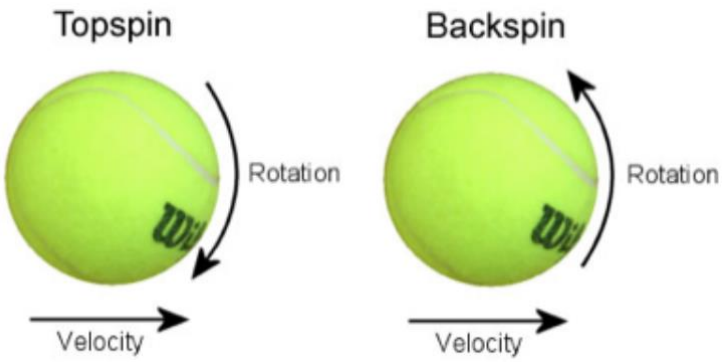


Figura 16. Topspin y Backspin

Fuente: [3]

Si el efecto de topspin (liftado) es muy pronunciado, provocará que cuando la pelota alcance el suelo y rebote hacia adelante con un ángulo menor al que impacta. Por el contrario, el backspin (cordado) hace que el ángulo con el que rebota la pelota sea mayor e incluso pueda detererse en seco. Esto se ilustra en la siguiente figura:

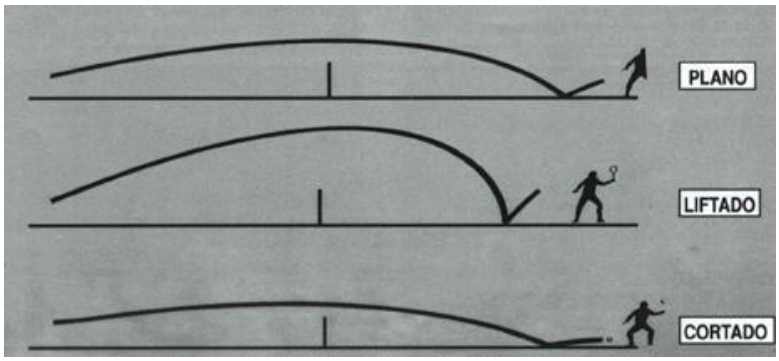


Figura 17. Tipos de lanzamiento

Fuente: [4]

Continuando con el sistema de ruedas de fricción, es de gran importancia imponer una distancia adecuada del

hueco que queda entre las ruedas de fricción con el fin de que la presión ejercida sobre la pelota sea la correcta. Una distancia demasiado grande hará que la presión ejercida sobre la pelota sea insuficiente y hará que el robot lance la pelota con una velocidad de salida inadecuada, mientras que una distancia demasiado pequeña podría impedir que el robot lance la pelota o la comprima demasiado. Esta distancia podremos modificarla subiendo o bajando la rueda superior, mientras que la rueda inferior permanece fija.

En la figura 18 podemos observar el sentido en el que gira cada una de las ruedas junto con los parámetros relativos a las ruedas y la distancia entre ejes con más detalle.

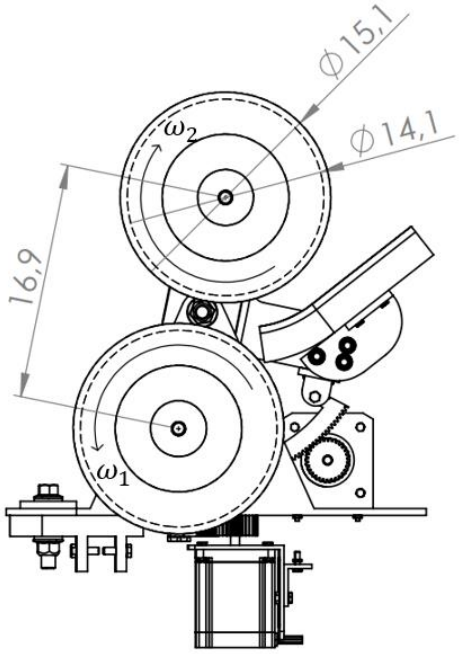


Figura 18. Sistema de ruedas de fricción

Fuente: Elaboración Propia

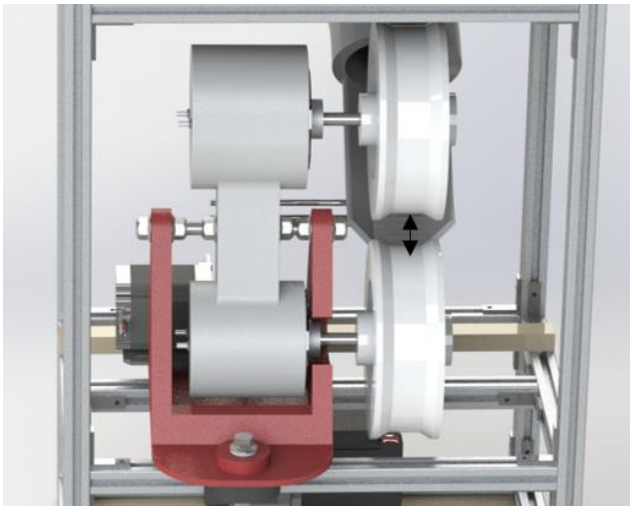


Figura 19. Distancia entre ruedas

Fuente: Elaboración Propia

Tras diversas pruebas, se ha comprobado que esta distancia debe ser de entre 2,5 y 5 centímetros.

Además de la distancia entre ruedas, el parámetro que determinará que lancemos la pelota con una velocidad de salida superior o inferior y con un efecto determinado, será la velocidad a la que giran los motores.

Los motores empleados para hacer girar las ruedas son motores de corriente continua de imán permanente.

Figura 20. Motor DC

Fuente: [5]

El principio de funcionamiento de los motores eléctricos de corriente directa o continua se basa en la repulsión que ejercen los polos magnéticos de un imán permanente cuando, de acuerdo con la Ley de Lorentz, interactúan con los polos magnéticos de un electroimán que se encuentra montado en un eje. Este electroimán se denomina "rotor" y su eje le permite girar libremente entre los polos magnéticos norte y sur del imán permanente situado dentro de la carcasa o cuerpo del motor.

Cuando la corriente eléctrica circula por la bobina de este electroimán giratorio, el campo electromagnético que se genera interactúa con el campo magnético del imán permanente. Si los polos del imán permanente y del electroimán giratorio coinciden, se produce un rechazo y un torque magnético o par de fuerza que provoca que el rotor rompa la inercia y comience a girar sobre su eje en el mismo sentido de las manecillas del reloj en unos casos, o en sentido contrario, de acuerdo con la forma que en que se encuentre conectado a la alimentación. Al cambiar los polos, cambia el sentido de giro.

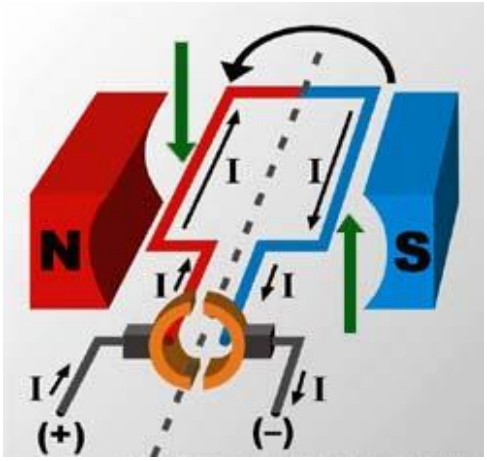


Figura 21. Funcionamiento de motor DC

Fuente: [6]

En esta otra figura se muestra, de forma esquemática y simplificada, un motor común de corriente directa (C.D) con un rotor formado por una simple bobina de una sola espira de color rojo y azul, para diferenciar cada mitad. Si seguimos el recorrido de la corriente eléctrica (I) asumiendo que fluye en el sentido convencional (del polo positivo "+" al polo negativo "–" de la batería, según indican las flechas negras), cuando en la mitad izquierda de la espira de color rojo se forma el polo norte "N" coincidiendo con la misma polaridad del campo magnético del imán permanente fijo al cuerpo del motor, se produce una fuerza de rechazo entre ambos polos iguales. Si aplicamos la "Regla de la mano izquierda" se puede determinar que esa mitad de la espira se moverá hacia abajo (flecha verde izquierda). Por otra parte, en la mitad derecha (de color azul) ocurrirá lo mismo, pero a la inversa, por lo que aplicando la propia regla comprobaremos que se moverá hacia arriba (flecha verde derecha).
La combinación de esas dos fuerzas o vectores actuando de forma opuesta y al unísono (de acuerdo con la Fuerza de Lorentz), provocará que el electroimán del rotor, formado aquí por esa simple espira, comience a girar en torno a su eje imaginario (representado por una línea de puntos en la figura) en dirección contraria a las manecillas de reloj en este ejemplo. Ese movimiento de rotación se encuentra señalado por la flecha negra en forma de semicírculo, que se encuentra dibujada al fondo de la espira.

En el motor de corriente directa el colector o conmutador sirve para conmutar o cambiar constantemente el sentido de circulación de la corriente eléctrica a través del enrollado de la bobina del rotor cada vez que completa media vuelta. De esa forma el polo norte del electroimán coincidirá siempre con el polo norte del imán permanente y el polo sur con el polo sur del propio imán. Al coincidir siempre dos polos magnéticos, que en todo momento van a ser iguales, se produce un rechazo constante entre ambos, lo que permite al rotor mantenerse girando ininterrumpidamente sobre su eje durante todo el tiempo que se encuentre conectado a la corriente eléctrica.

En nuestro caso, las características de cada motor DC son las mostradas en la tabla:

Especificaciones eléctricas		Especifcaciones físicas	
Potencia	120 W	Diámetro del cuerpo	76 mm
Voltaje	12 V	Longitud del cuerpo	83 mm
Corriente continua máxima	12 A	Diámetro del eje	10 mm
Velocidad sin carga	5100 rpm	Longitud del eje	70 mm

Tabla 5. Especificaiones de los motores DC

Fuente: [5]

Además, como se verá en la sección correspondiente al control del robot, la velocidad de los motores será controlada en bucle cerrado mediante un control PI en el que se mide la velocidad de cada motor mediante sensores y se realiza la acción de control en cada periodo de muestreo.

2.3. Sistema de alimentación de pelotas

Este sistema simplemente es el engargado de hacer llegar las pelotas desde la caja de alimentación hasta las ruedas de fricción que lanzan la pelota, además de controlar la frecuencia con la que se lanzan y de asegurar que las pelotas no se quedan bloqueadas.

La salida de la caja de alimentación está conectada con las ruedas de fricción mediante un tubo flexible que se adapta al ángulo de lanzamiento vertical y horizontal en cada caso.

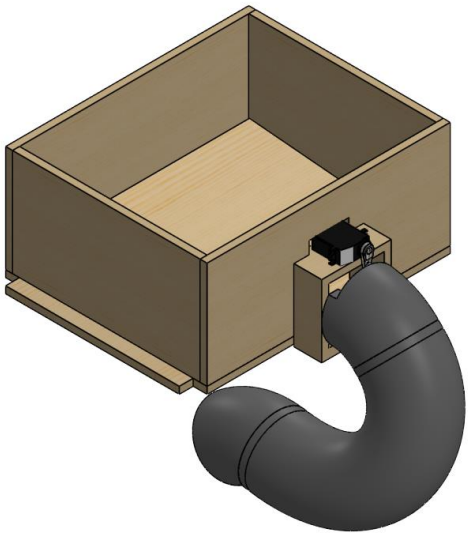


Figura 22. Sistema de alimentación de pelotas

Fuente: Elaboración Propia

Como se puede observar en el Anexo 2, Plano 3, el interior de la caja está inclinado hacia la salida de las pelotas con el objetivo de que estas puedan fluir hacia el tubo y posteriormente hacia el sistema de lanzamiento.

La frecuencia con la que se lanzan las pelotas está controlada mediante un servomotor que permite el paso o no de pelotas.

Figura 23. Servomotor

Fuente: [7]

Un servomotor es un actuador rotativo o motor que permite un control preciso en términos de posición angular, aceleración y velocidad, capacidades que un motor normal no tiene. En definitiva, utiliza un motor normal y lo combina con un sensor para la retroalimentación de posición.

Pero, los servomotores no son en realidad una clase específica de motor, sino una combinación de piezas específicas, que incluyen un motor de corriente continua o alterna, y son adecuados para su uso en un sistema de control de bucle cerrado.

Los servomotores se controlan enviando un pulso eléctrico de ancho variable, o modulación de ancho de pulso (PWM), a través del cable de control. Hay un pulso mínimo, un pulso máximo y una frecuencia de repetición.

Por lo general, un servomotor sólo puede girar 90° en cualquier dirección para un movimiento total de 180°. La posición neutra del motor se define como la posición en la que el servo tiene la misma cantidad de rotación potencial tanto en el sentido de las agujas del reloj como en el sentido contrario.

El PWM enviado al motor determina la posición del eje, y se basa en la duración del pulso enviado a través del cable de control; el rotor girará a la posición deseada.

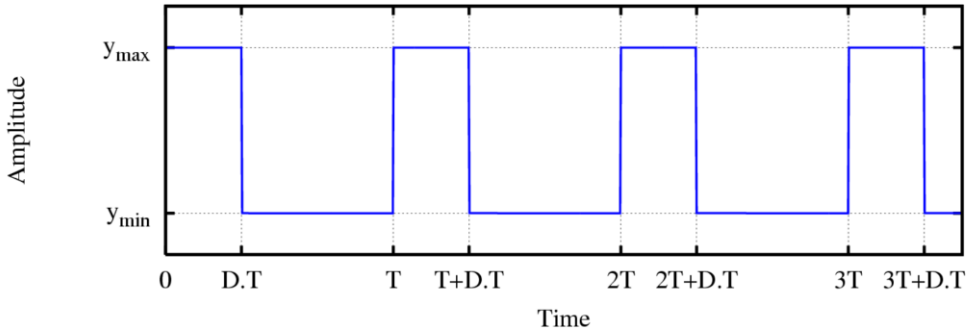


Figura 24. Señal PWM

Fuente: [8]

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

Donde:

D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra (portadora) y el ciclo de trabajo está en función de la moduladora.

Este tipo de señales son muy utilizadas en electrónica y son también las que emplearemos para controlar los motores DC.

Una vez explicada la fase de diseño, pasamos a explicar cómo ha sido la construcción y puesta en marcha del robot.

3 CONSTRUCCIÓN

En esta sección se explicará como ha sido el proceso de construcción de construcción del robot, detallando los inconvenientes y modificaciones que han tenido lugar con respecto al diseño propuesto.

En un primer momento, se desarrolló la estructura junto con la caja de alimentación del robot, con los componentes detallados en el apartado 2.1.1 y con el montaje del servomotor, comprobando además los ángulos y el tiempo que necesario para que suelte una sola pelota.

Figura 25. Estructura construida

Fuente: Elaboración Propia

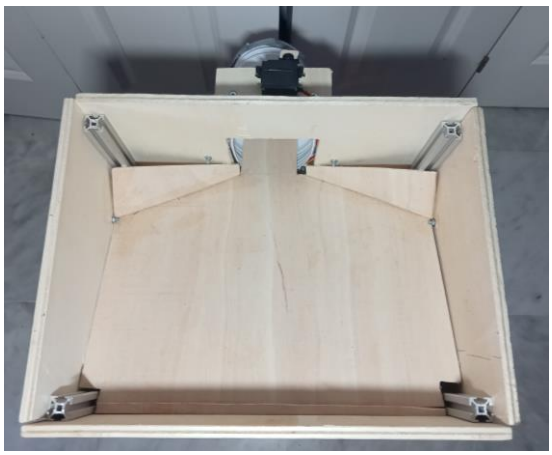


Figura 26 Caja de alimentación

Fuente: Elaboración Propia

Los parámetros del servomotor resultantes, que son implementados en Arduino, son los siguientes:

Posición cerrada	80 grados
Posición abierta	115 grados
Tiempo abierto	0,3 segundos

Tabla 6. Parámetros del servomotor

Fuente: Elaboración Propia

En paralelo, los distintos modelos 3D de cada parte del sistema de lanzamiento, son convertidos a formato STL e imprimidos en 3D por la empresa Innova 3D, en material PLA (ácido pliláctico), el cuál posee las siguientes propiedades físicas y mecánicas:

Densidad	1,24 g/cm3

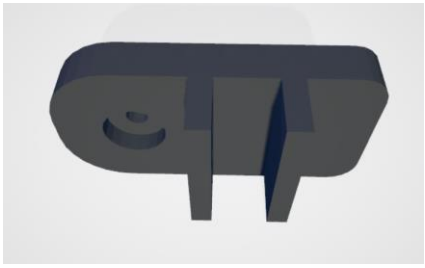
Resistenci a la tracción	3309 MPa
Límite elástico	55MPa
Resistencia a la compresion	66 MPa
Resistencia a la flexión	485 MPa
Temperatura de deformación	55 ºC
Muy baja resistencia a la humedad	-
Elongación	3%

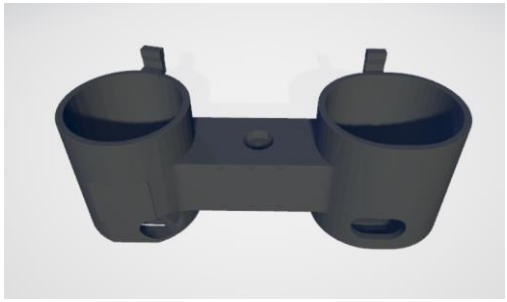
Tabla 7. Propiedades físicas y mecánicas del PLA

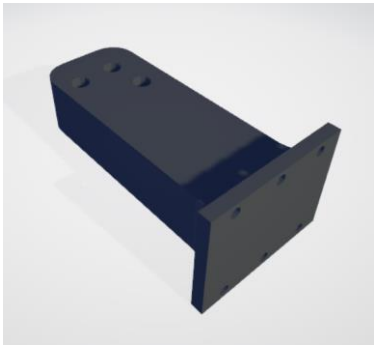
Fuente: [9]

Además, las propiedades de cada pieza dependen de la cantidad de relleno. En nuestro caso, el relleno elegido ha sido del 20%.

A continuación, se muestra cada modelo que ha sido impreso. Hay que tener en cuenta que se imprimen dos unidades de las piezas que forman los mecanismos de engranajes, uno para el ángulo vertical y otro para el ángulo horizontal.







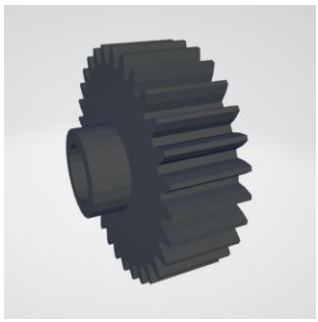
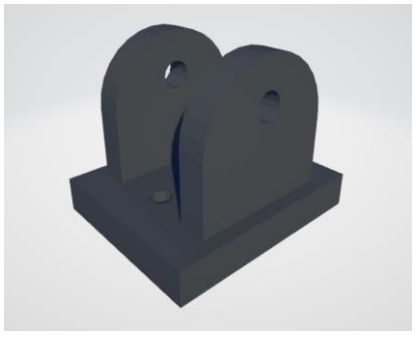
	
	
	

Tabla 8. Modelos Impresos en 3D

Fuente: Elaboración Propia

Una vez que se ha dispuesto de todas las piezas, juntos con los motores DC, rodillos, motores paso a paso, drivers, sensores, batería, y el resto de los componentes electrónicos, ser realiza el montaje del sistema de lanzamiento de pelotas, conectándolo con la electrónica y el tubo flexible por donde nos llegarán las pelotas. El resultado obtenido es el mostrado a continuación:

Figura 27. Sistema lanzapelotas construido

Fuente: Elaboración Propia

Figura 28. Rodillos y entrada del tubo de alimentación

Fuente: Elaboración Propia

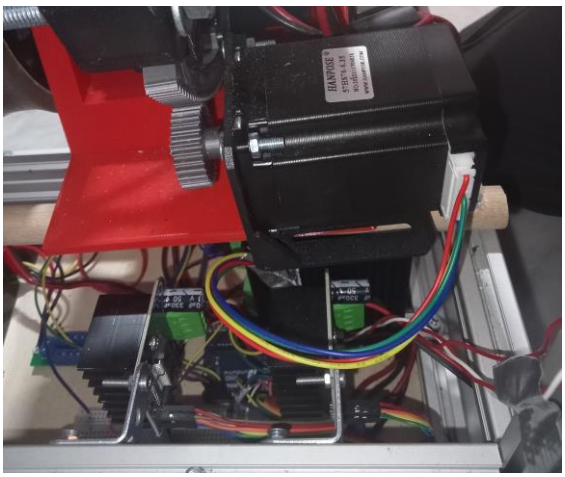


Figura 29. Mecanismo de Ángulo de lanzamiento Vertical

Fuente: Elaboración Propia

Figura 30. Mecanismo de Ángulo de lanzamiento Horizontal

Fuente: Elaboración Propia

Tras la construcción del robot, se ha comprobado que sin montar los motores DC ni rodillos, ni conectar el tubo de alimentación de pelotas, los mecanismos para el ángulo de lanzamiento Vertical y Horizontal funcionan correctamente. Sin embargo, al montar estos componentes, el mecanismo para el ángulo de lanzamiento Horizontal no logra funcionar correctamente, debido a que sufre dificultades debido a la fuerza hacia abajo que ejerce el tubo lanzapelotas, y al peso del propio sistema. Asimismo, el mecanismo para el ángulo de lanzamiento vertical también sufre ciertas dificultades. Para empezar, antes de conectar el robot a la batería es necesario colocar el ángulo vertical en la posición para cero grados, y aunque logra mantenerla, pierde algunos dientes. Además, no el ángulo máximo que se alcanza es de aproximadamente 20 grados.

A pesar de dichas dificultades, es posible modificar en cierta medida tanto el ángulo de lanzamiento vertical como horizontal.

Tras esto, se ha comprobado que tras mandar referencias de voltaje a los motores DC en bucle abierto, el sistema de ruedas de fricción funciona correctamente y logra lanzar pelotas de una forma adecuada.

Como se verá posteriormente, para obtener las velocidades medidas se han empleado unos módulos seguidores de línea que detectan el cambio entre el color negro de los rodillos, a una línea blanca, como se ilustra en la figura 31, de manera que se pueden contar las vueltas que da cada rodillo y obtener las revoluciones por minuto (RPM). En el apartado 4.1.6 se explica detalladamente como consigue esto dicho sensor.

Figura 31. Módulo seguidor de línea

Fuente: Elaboración Propia

Para que dichos sensores funcionen correctamente y nos den una lectura adecuada, es necesario que estén colocados a una distancia de entre 0.2mm y 15mm, lo cual significa que debe aproximarse bastante al rodillo y es necesario ajustarlo bien para que no se suelte o roce con el rodillo y se puedan producir daños en el sensor.

Además, es necesario calibrar el potenciómetro de manera que nos permita obtener los cambios de color correctamente.

Con esto, se ha conseguido obtener una lectura relativamente buena, con el correspondiente filtrado, de las RPM obtenidas en cada rueda, de manera que nos permita realizar el posterior control PI en bucle cerrado.

4 ELECTRÓNICA Y CONTROL DEL ROBOT

En esta sección se mostrará una descripción de los componentes electrónicos empleados, junto con una explicación del código y la obtención de los parámetros necesarios para el control PI de los motores DC. Además, se muestran los resultados obtenidos tanto en bucle abierto como en bucle cerrado.

4.1. Componentes electrónicos

El control de nuestro robot estará centrado en un modelo similara al Arduino Uno R3, el cual se apoya de un módulo bluetooth HC-05 para la conexión con un dispositivo móvil desde el que controlaremos el robot por medio de una App.

Asimismo, se emplearán dos drivers DM542 para el control de los motores paso a paso NEMA 23, dos drivers BTS7960B para los motores DC, dos módulos de seguimiento de línea para la medición de la velocidad angular de cada rueda yel servomotor que accionará el lanzamiento de pelota.

4.1.1 Alimentación del Robot

La alimentación es de corriente continua, suministrada por una batería de plomo de 12V y 9AH como la mostada en la figura siguiente:

Figura 32. Batería de plomo

Fuente: [10]

Dicha batería va conectada mediante un cable con pinzas con el que encenderemos el robot, a un panel de distribución de corriente de 2 tomas como el que se muestra en la figura 33, que nos permite extender la salida a 13 tomas de corriente TC a 12V.

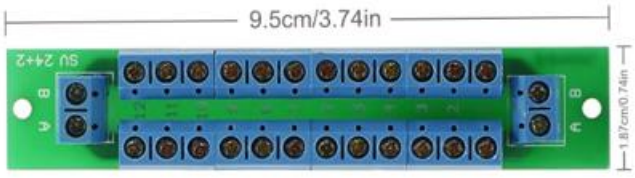


Figura 33. Panel de distribución

Fuente: [11]

De dicho panel, sacaremos las tomas de 12V que necesitamos para alimentar los distintos componentes electrónicos del robot.

Un caso especial se ha dado para los drivers DM542, que controlan los motores paso a paso NEMA23. Dichos

drivers, como se verá en su correspondiente apartado, funcionan a una tensión de entre 20V y 50 V y no podrían funcionar de manera óptima con 12V.

Para alcanzar la tensión necesaria de 20V, se ha hecho empleo de un convertidor elevador DC-DC, como el mostrado en la figura 34, que convierte 12V suministrados desde una de las tomas del panel de distribución nombrado, y los convierte a 24V y 5A, que son suficientes para alimentar nuestros drivers.

Figura 34. Convertidor DC-DC

Fuente: [12]

El inconveniente de usar este convertidor es que consume 120W, lo cual reduce considerablemente la duración de la batería.

Para entender mejor como funciona un convertidor elevador DC-DC, se da una breve explicación de su funcionamiento.

Cuando necesitamos obtener un voltaje o intensidad distinta a la suministrada por la alimentación de nuestro proyecto, o bien necesitamos pasar de corriente continua a alterna o viceversa, existen distintos dispositivos en la electrónica de potencia que nos permiten realizar estas funciones. Dependiendo del caso, se puede necesitar un Inversor (Convertidor DC/AC), Rectificador (Convertidor AC/DC), Regulador (AC a frecuencia constante) Convertidor DC/DC, tal y como se muestra en la siguiente figura:

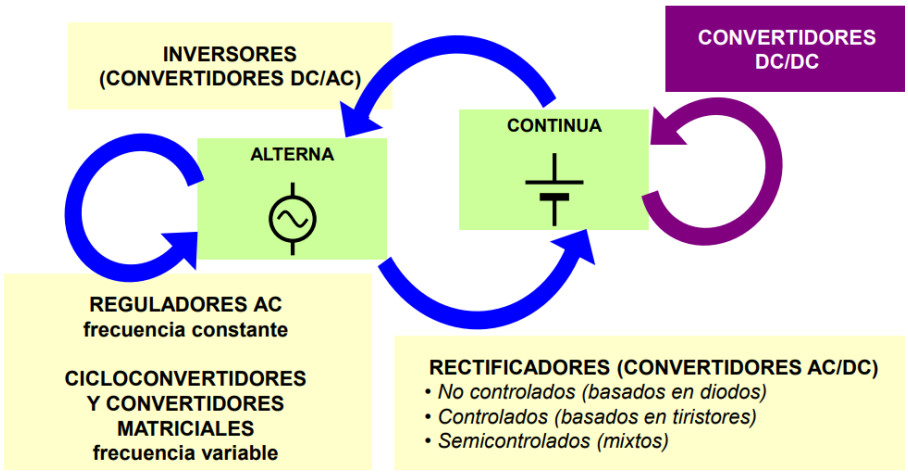


Figura 35. Tipologías de convertidores

Fuente: [13]

A su vez, en el ámbito de convertidores DC-DC, podemos encontrar distintos tipos dependiendo de lo que necesitemos:

- Convertidor DC/DC Elevador (boost): Modo de conducción continua y discontinua
- Convertidor DC/DC reductor
- Convertidor DC/DC reductor-elevador
- Convertidor Cúk

En nuestro caso, necesitamos un convertidor DC/DC elevador.

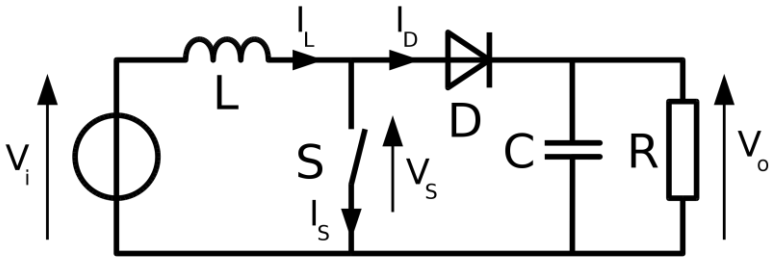


Figura 36. Esquema convertidor DC/DC elevador

Fuente: [13]

Un convertidor elevador (boost) DC-DC es un circuito electrónico que se utiliza para aumentar el voltaje de una fuente de alimentación DC. Es un tipo de fuente de alimentación conmutada que contiene al menos dos interruptores semiconductores (diodo (D) y transistor (S)), y al menos un elemento para almacenar energía (condensador (C), bobina(L) o combinación de ambos). Frecuentemente se añaden filtros construidos con inductores y condensadores para mejorar el rendimiento.

Cuando el interruptor (S) está cerrado, la corriente fluye a través del inductor y se almacena energía en el inductor.

Cuando el interruptor (S) se abre, la energía almacenada en el inductor se transfiere a la carga a través del diodo. Esto resulta en una tensión de salida mayor que la tensión de entrada.

La relación entre la tensión de entrada y la tensión de salida está determinada por el ciclo de trabajo del interruptor, que se controla mediante un circuito de control.

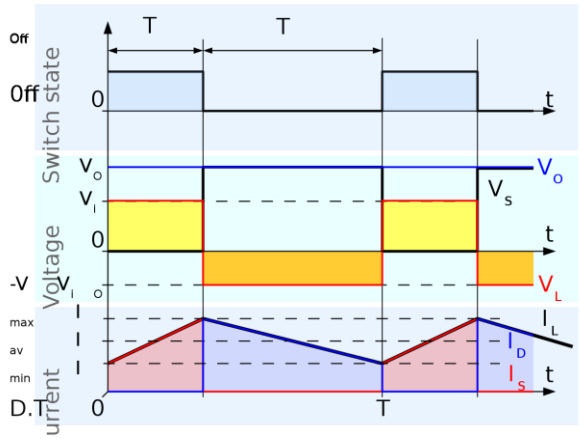


Figura 37. Formas de onda de corriente y voltaje en un convertidor Boost operando en modo continuo.

Fuente: [13]

Una vez explicada la alimentación de cada componente del robot, pasamos a detallar brevemente cada componente.

4.1.2 Microcontrolador (Arduino Uno R3)

Esencialmente la placa Arduino Uno es una placa electrónica basada en el chip de Atmel ATmega328. Tiene 14 pines digitales de entrada / salida, es el Arduino Pinout de los cuales 6 los puede utilizar como salidas PWM, 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reset.

El software de la placa incluye un controlador USB que puede simular un ratón, un teclado y el puerto serie.

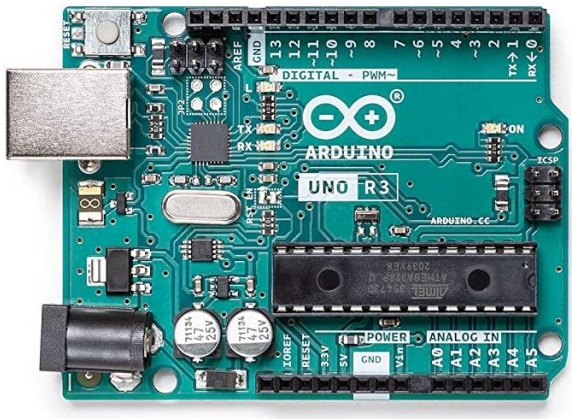


Figura 38. Arduino Uno R3

Fuente: [14]

Esta placa tiene todo lo necesario para apoyar el microcontrolador, basta con conectarlo a un ordenador con un cable USB o con un adaptador. El Arduino Uno se diferencia de todas las placas anteriores en que no utiliza el FTDI USB a serie driver chip. En lugar de ello, cuenta con el Atmega8U2 programado como convertidor de USB a serie.

Sus características son:

- Microcontrolador: ATmega328
- Voltaje de operación: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límites): 6-20V
- Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6
- Corriente DC por pin de E/S: 40 mA
- Corriente DC para 3.3V Pin: 50 mA
- Memoria Flash: 32 KB de los cuales 0,5 KB utilizados por el bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad de reloj: 16 MHz

Este microcontrolador será el encargado de controlar el sistema, realizando en bucle las operaciones detalladas en el Anexo 1, Código. Es importante dejar libres los pines 0 y 1, ya que serán utilizados de manera interna para la comunicación serial con nuestro portátil, en el que se recibirán los datos leídos por los sensores.

4.1.3 Módulo Bluetooth HC – 05

El módulo Bluetooth HC-05 consta de 6 pines, puede actuar como maestro o como esclavo y acepta un mayor número de órdenes de configuración. Esta configurado de fábrica para trabajar como Esclavo, es decir, preparado para recibir peticiones de conexión, pero podemos configurarlo para trabajar con Maestro utilizando comandos AT, en este estado puede conectarse con otros módulos Bluetooth.

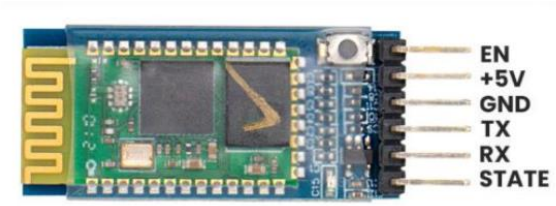


Figura 39. Módulo HC-05

Fuente: [15]

En nuestro caso será utilizado como esclavo, de manera que será conectado con nuestro dispositivo móvil mediante una App, la cuál enviará las referencias de posición, velocidad, y la orden de lanzar pelota. Este chip recibe los datos y se los transmite al Arduino, por tanto, es necesario conectar los pines de TX y RX con el Arduino. En nuestro caso serán conectados con los pines A0 y A1.

Las características técnicas del módulo HC-05 son las siguientes:

- Voltaje de operación: 3.6V - 6V DC
- Consumo corriente: 50mA
- Bluetooth: V2.0+EDR
- Frecuencia: Banda ISM 2.4GHz
- Modulación: GFSK(Gaussian Frequency Shift Keying)
- Potencia de transmisión: 4dBm, Class 2
- Sensibilidad: -84dBm a 0.1% BER
- Alcance 10 metros
- Interfaz de comunicación: Serial TTL
- Velocidad de transmisión: 1200bps hasta 1.3Mbps
- Baudrate por defecto: 38400,8,1,n.
- Seguridad: Autenticación y encriptación
- Temperatura de trabajo: -20C a +75C
- Compatible con Android
- Dimensiones: 37*16 mm
- Peso: 3.6 gramos

Como ya se ha comentado, para enviar las órdenes a nuestro microcontrolador nos conectamos al módulo Bluetooth, mediante nuestro dispositivo móvil.

La aplicación, realizada en MIT App inventor, tiene la siguiente interfaz:

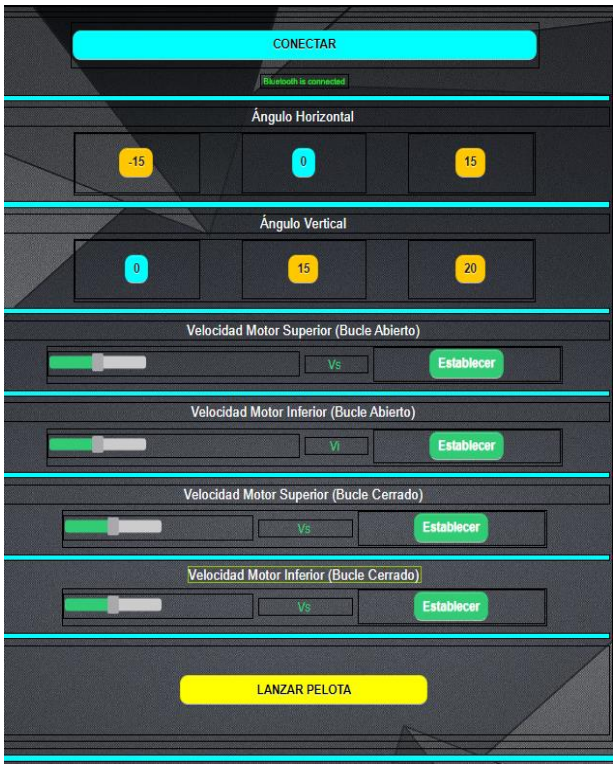


Figura 40. Interfaz App desarrollada con MIT App Inventor

Fuente: Elaboración Propia

El detalle de cómo ha sido creada esta app, consta de una parte gráfica en la que se agregan y personalizan los elementos de los que consta, en este caso, botones, sliders (barras con valor ajustable) y cuadros de texto que nos muestran los valores de los sliders.

Por otro lado, tiene la parte de programación de la funcionalidad de cada elemento, la cuál se realiza de una forma sencialla mediante bloques. Al iniciar la aplicación, nos aparece un botón con el que nos conectamos al módulo HC-05 (estando previamente conectado a la batería).

Una vez conectados, se nos despliegan una serie de secciones, cada una dedicada a implementar una funcionalidad.

En las dos primeras secciones, encontramos las opciones de ángulos horizontal y vertical en las que puede posicionarse el robot. Al pulsar sobre uno de los botones, los motores paso a paso girarán el número de grados que deben girar, previamente calculado en los apartados 2.1.3 y 2.1.4. Por simplicidad, solo se han configurado tres posiciones para cada ángulo, podiendo extenderse a cualquier posición dentro del rango de funcionamiento.

Cuando el microcontrolador recibe la posición deseada, ejecuta el comando para que los drivers DM542 muevan los motores paso a paso el número de pasos calculado para cada posición mediante la siguiente fórmula:

$$nº\,pasos = \frac{grados}{360} \times R$$

Siendo R la resolución configurada en nuestro controlador, que en nuestro caso es de 400 pasos por vuelta.

En la siguiente tabla se muestran los comandos que debe recibir el Arduino por Bluetooth para moverse a cierta posición, junto con los grados que esto resulta en la rueda conductor y el numero de pasos que tiene que moverse:

Comando	Posición	Ángulos a girar Rueda Conductora	Número de pasos
Ángulo Horizontal			
b	-15º	-125º	-139
c	0º	0º	0
d	15º	125º	139
Ángulo Vertical			
e	0º	0º	0
f	15º	125º	139
g	20º	167	186

Tabla 9. Comandos y número de pasos para cada posición

Fuente: Elaboración Propia

Cuando se marcan 0 pasos, es la posición por defecto a la que se inicia el robot, y a la que deberá volver si se envía el comando correspondiente.

A continuación, encontramos las secciones para activar los motores DC en bucle abierto. Mediante los sliders, podemos establecer el valor a enviar, el cuál será un número de 0 a 255 (1 byte). La velocidad con la que girarán los motores será proporcional al valor enviado. En este caso, el Arduino al recibir el comando enviará una señal PWM a los drivers BTS7960B, que transformará dicha señal un voltaje de entre 0 y 12V que llegan al motor DC.

Las dos secciones siguientes, son las correspondientes para establecer la velocidad de cada motor, pero en este caso en bucle cerrado haciendo uso de un control PI, dando en lugar de un número de 0 a 255, la referencia de velocidad en RPM a la que queremos que gire nuestro motor. Esto se explicará en el apartado 4.2 .

Al pulsar cada uno de los botones para establecer la velocidad, se enviarán al microcontrolador los comandos ‘h’, ‘i’, ‘j’ y ‘k’ junto el número introducido en cada slider, respectivamente.

Por último, tenemos el botón “Lanzar pelota”, que envía el comando “a” y activa el servomotor dejando salir una sola pelota, abriéndolo por un instante abierto y volviéndolo a cerrar. Los parámetros que se envían al servomotor se han determinado en el apartado 2.3.

En el Anexo 1, Código, se puede encontrar como funciona la lectura de datos bluetooth en detalle.

4.1.4 Driver controlador de motores paso a paso DM542

Es un controlador de motores paso a paso de alta tensión (20 a 50V), que funciona a través de tecnología DSP lo cual lo convierte en una solución rápida y versátil para el manejo de motores a pasos, este controlador trae integrada tecnología anti-resonancia y control de corriente hasta 4.2A.

Figura 41. Driver DM542

Fuente: [2]

Las características técnicas de este controlador son las siguientes:

- Voltaje de operación: 20V – 50V DC
- Corriente pico de salida: 1 ~ 4.2A
- Corriente de señal lógica: 7 ~ 10 mA (típico 10 mA)
- Frecuencia de entrada de pulso: 0~200kHz
- 15 resoluciones de micro-pasos seleccionables de 400-25600
- Protección contra sobretensión y sobrecorriente
- Resistencia de aislamiento: 500MΩ
- Adecuado para motores de 2 fases
- 8 ajustes de corriente de salida seleccionables de 1A a 4.2A a través de interruptores DIP
- Dimensiones: 118 x 76 x 34 mm

Normalmente, los motores paso a paso necesitan un número determinado de pulsos para completar una vuelta completa. El motor paso a paso NEMA 23 tiene un ángulo de paso de 1.8°, por lo que necesita 200 pulsos para completar una vuelta completa (pulsos * 1.8° = 360°).

Por otro lado, el driver DM542 puede ser configurado para operar en diferentes modos de microstepping, lo que aumenta la precisión de la posición del motor. Por ejemplo, si se configura el driver en modo 1/2 de microstepping, se dividiría cada paso en 2 subpasos, lo que aumentaría la resolución a 400 pasos por revolución. Entonces, para girar un ángulo de 1 grado con el driver en modo 1/2 de microstepping, se necesitarían 400/360 = 1.11 pasos.

Para nuestro robot, necesitaríamos dos drivers DM542, uno para cada motor NEMA 23. El esquema de conexiones lo podemos ver en el plano 4 del Anexo 2, planos.

Para controlar cada driver, necesitamos incluir en nuestro código la librería <AccelStepper.h>, la cual nos incluye las funciones necesarias para configurar y enviar los pulsos. Como se puede apreciar en el código, el funcionamiento para moverse a una posición determinada es el siguiente:

```
// Movemos el motor horizontal a la posición guardada para -15°
motor_h.moveTo(pos1);
// Avanzamos el motor hasta llegar a la posición deseada
while (motor_h.distanceToGo() != 0) {
  motor_h.run();
}
// Establecemos la posición actual del motor como la posición guardada
motor_h.setCurrentPosition(pos1);
delay(100);
```

La conexión con el Arduino, la alimentación y los motores paso a paso puede consultarse en el plano 4 del Anexo 2, planos.

4.1.5 Driver controlador de motores motores DC BTS7960B

Los drivers basados en BTS7960 son drivers para controlar motores de corriente continua de alta potencia, capaces de proporcionar hasta 43A de corriente a una tensión de alimentación de entre 6 a 27V.

La lógica del driver funciona a con voltajes de 3.3V a 5V por lo cual es compatible con la mayoría de los microprocesadores. Además, admiten realizar el control de la velocidad del motor mediante PWM con una frecuencia máxima de 25 kHz.

El BTS7960 incorpora mecanismos de protección contra cortocircuito, sobre/infra voltaje, y sobre temperatura. También dispone de dos pines que permiten medir la corriente entregada por el driver.

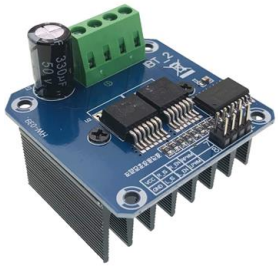


Figura 42 Driver BTS7960B

Fuente: [16]

Figura 43. Circuito integrado BTS7960B

Fuente: [17]

Estos drivers emplean dos circuitos integrados BTS7960 como el que se muestra en la figura 43, un medio puente H de la familia NovalithIC. Sumando ambos integrados se configura un puente H completo.

El BTS7960 incorpora un transistor MOSFET Canal-p para la parte alta del medio puente, y un MOSFET Canal-N para la parte baja.

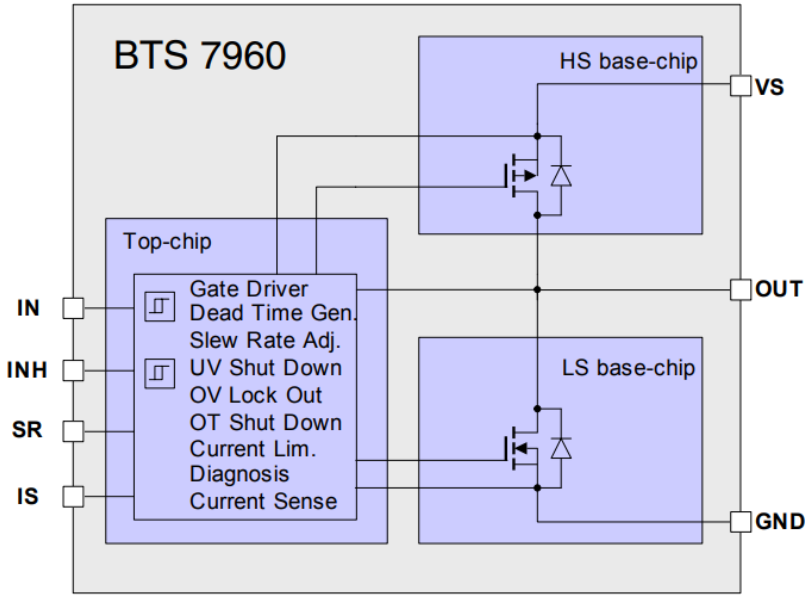


Figura 44. Esquema Circuito integrado BTS7960B

Fuente: [17]

Las conexiones con los motores DC, alimentación y microcontrolador, y la programación de estos drivers, se realiza de tal forma que cada motor DC gire en sentido opuesto. El motor superior girará en sentido horario y el inferior antihorario.

Análogamente al resto de componentes, las conexiones pueden consultarse en el anexo 2, Planos.

4.1.6 Módulo seguidor de líneas TCRT5000

El TCRT5000 es un sensor de reflexión que incluyen un led emisor de infrarrojos y un fototransistor en el mismo encapsulado. El led emite un haz de luz infrarroja invisible para el ojo humano que se refleja en una superficie clara y es capturado por el fototransistor que tiene una película de filtrado de luz natural que solo permite el paso de los infrarrojos emitidos por el led. Por lo tanto, de acuerdo con el color de la superficie, el fototransistor recibe un valor mayor o menor de reflexión.

Sus características técnicas son:

- Sensor: TCRT5000
- Alimentación: 3,3 a 5 Vdc
- Longitud de onda: 950 nm
- Distancia de medida: 0.2 mm a 15 mm
- Salida: Doble, analógica y digital.
- Dimensiones: 32mm x 14mm x 12mm

Además, su sensibilidad es ajustable por potenciómetro, lo cual es necesario si queremos obtener una medida fiable.

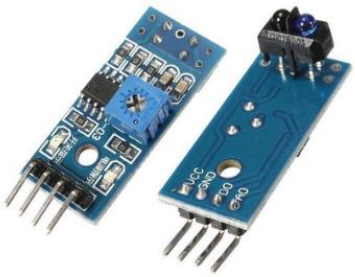


Figura 45. Módulo seguidor de líneas TCRT5000

Fuente: [18]

Este sensor, es colocado tal y como se muestra en la figura 31, y nos permitirá obtener la lectura de RPM de las ruedas.

Para ello, distingue el cambio de color negro a blanco. Consta de una salida analógica y otra digital, con la lectura del sensor. En este caso se hace uso de la salida digital. Esta nos da un valor de uno cuando detecta negro, y cero al detectar la línea blanca.

Una vez que tenemos una correcta lectura, hacemos uso de las interrupciones de las que consta el microcontrolador en los pines 2 y 3. Las interrupciones pueden ser detectadas en flancos de subida, bajadao cuando se detecta un cambio. En este caso, las hemos configurados para flancos de subida, haciendo uso de la siguiente función:

```
attachInterrupt(digitalPinToInterrupt(sensor_vel_sup), interrupcion0, RISING);
```

De esta manera, al detectarse un flanco de subida, entra en la función (interrupción0) a la que llama, y dentro de ella un contador se incrementa en una unidad. Cada 100 milisegundos, procesamos el valor del contador por medio de la función medirRPM_sup(), multiplicándolo por 600 para obtener las revoluciones por minuto.

Posteriormente, es necesario filtrar la señal de RPM obtenida para obtener un valor más limpio y poder realizar el control. Para ello se hace uso de un filtro EMA (Exponential Moving Average):

$$A_n = \alpha * \mathrm{M} + (1 - \alpha) * A_{n-1}$$

Dónde:

A_n : valor filtrado

A_{n-1} : valor filtrado anterior

M: valor muestreado de la señal para filtrar

α: factor que va de 0 a 1

Este filtro aporta un suavizado que depende directamente del valor, tiene ventajas en el cálculo ya que utiliza una única instrucción y almacena en memoria solo el valor filtrado anterior. En esta ocasión utilizaremos el filtro exponencial como filtro pasa bajos para eliminar ruidos de altas frecuencias en la señal que permita obtener un valor medio de las RPM calculadas.

Una disminución del valor Alpha aumenta la suavidad de la señal, haciéndola más estable y menos susceptible a cualquier ruido, pero esto también provoca un aumento del tiempo de respuesta del sistema, de alguna forma retrasa la señal respecto de la original.

4.2. Control Automático

Una vez somos capaces de actuar sobre los motores DC y medir las RPM de cada rueda, estamos en disposición de realizar el control automático de dichos motores.

La idea es implementar un control PI (Proporcional – Integral) para controlar las velocidades de cada rueda, estableciendo la velocidad de referencia y comprobar que cada motor se mantiene en torno a esa referencia

El control PI, es un tipo de controlador de retroalimentación utilizado para controlar un proceso en función de la diferencia entre una variable medida y una variable de referencia. Se llama así porque combina una acción proporcional e integral para lograr un control preciso del proceso.

La acción proporcional proporciona una respuesta inmediata y proporcional a la diferencia entre la variable medida y la variable de referencia. La acción integral integra esta diferencia a lo largo del tiempo para proporcionar un control a largo plazo y reducir el error en estado estable.

En nuestro caso, el controlador PI se utiliza para ajustar la señal de control (PWM) que actúa sobre los drivers BTS7960B, en función de la diferencia entre la velocidad medida de las ruedas y la velocidad de referencia deseada. La acción proporcional ajusta la señal de control de manera proporcional a la diferencia de velocidad, mientras que la acción integral ajusta la señal de control para eliminar el error en estado estable y garantizar que el motor alcance la velocidad de referencia.

El esquema representado en álgebra de bloques sería el mostrado en la siguiente figura:

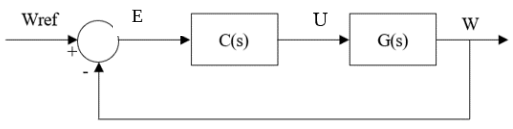


Figura 46. Control en bucle cerrado

Fuente: Elaboración Propia

Donde Wref es la velocidad angular de referencia enviada al robot y W es la velocidad angular leída de las ruedas (en RPM), y U es la acción de control que proporciona el controlador (señal PWM en nuestro caso).

C(s) es nuestro controlador, que en este caso es un PI y está definido de la siguiente forma:

$$C(s) = K_p + \frac{K_i}{s}$$

E es el error que tendremos en cada periodo de muestreo entre la velocidad de referencia y la velocidad medida.

Por último, G(s) es la función de transferencia que representa el modelo matemático del motor DC, el cuál necesitamos para sintonizar nuestro controlador (obtener constantes K_p y K_i).

Este modelo tiene la forma de un sistema de primer orden:

$$G(s) = \frac{K}{\tau s + 1}$$

Siendo $K = \frac{\Delta y}{\Delta u}$, es decir, la relación de ganancia entre la entrada y la salida del proces.

Y τ es la constante de tiempo, la cual nos indicará el tiempo en el cual el sistema tiene un 63,21% del valor en estado estacionario.

4.1.1 Obtención de modelo en bucle abierto

Para obtener K y τ, necesitamos estudiar el sistema en bucle abierto, sin aplicación de control.

Para ello, aplicamos una entrada en escalón (señal PWM con valor 120). Haciendo uso de la herramienta serial plotter de Arduino, podemos ver cómo es la respuesta del sistema ante esta entrada. Sin embargo, para que resulte más fácil analizar los datos, se han enviado a Matlab mediante comunicación serial. Una vez los tenemos en Matlab, hacemos uso de la herramienta systemIdentification:

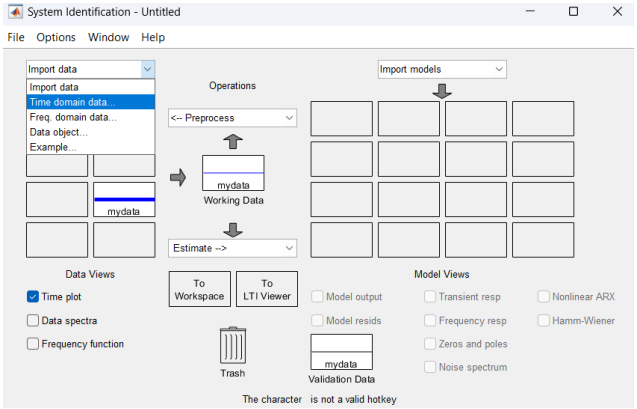


Figura 47. System Identification – Matlab

Fuente: Elaboración Propia

Elegimos la opción de datos en dominio del tiempo, y agregamos nuestros vectores de entrada y salida, establecemos el tiempo de muestreo de 0.01 segundos, pinchamos en “Time plot” y obtenemos la siguiente representación del sistema en bucle abierto:

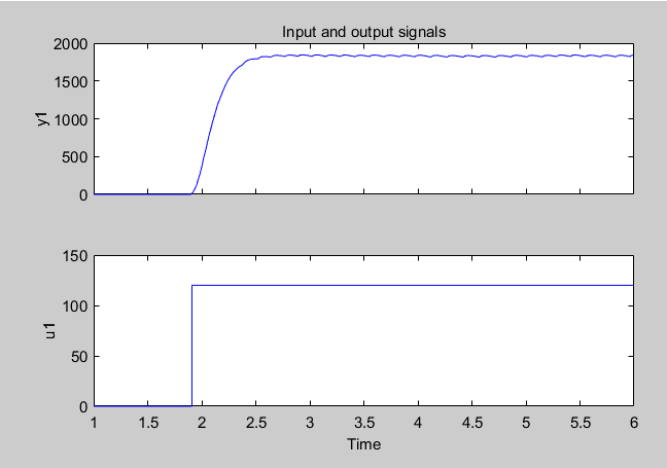


Figura 48. Sistema en bucle abierto – Matlab

Fuente: Elaboración Propia

Como puede observarse, con una entrada en escalón de 120 se obtienen aproximadamente 1830 RPM. Por seguridad, se ha establecido este valor como máximo para el funcionamiento de este robot. A partir de esta gráfica, podríamos calcular nuestros parámetros a mano. Sin embargo, Matlab puede obtener una aproximación del sistema más precisa. Para ello, clicamos en "Process Models", "Estimate" y representamos la aproximación:

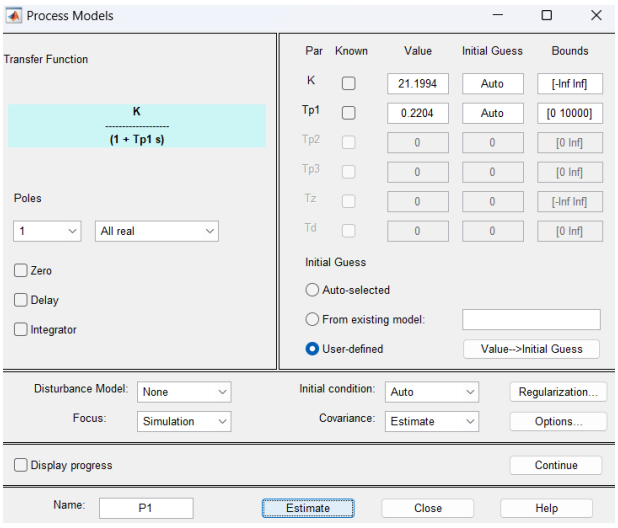


Figura 49. Process Models – Matlab

Fuente: Elaboración Propia

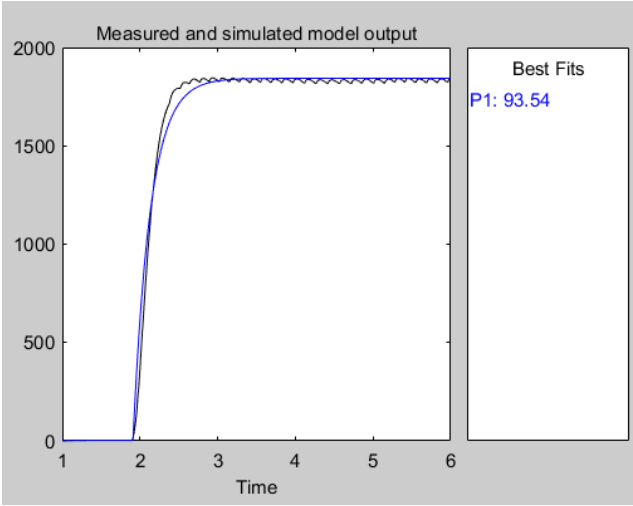


Figura 50. Modelo aproximado – Matlab

Fuente: Elaboración Propia

Por tanto, obtenemos la siguiente función de transferencia resultante, con una aproximación del 93.54% es:

$$G(s) = \frac{15.361}{0.2224s + 1}$$

Una vez que disponemos de la función de transferencia, la exportamos al Workspace de Matlab, y obtenemos la función de tranferencia mediante el comando G=tf(P1).

4.1.2 Obtención de parámetros del control PI

Al combinar la función de transferencia del sistema de primer orden, con el control PI, obtenemos la siguiente función de transferencia:

$$G(s) = \frac{K(K_p s + K_i)}{\tau s^2 + s}$$

Fuente: [19]

Ahora, la función de transferencia en bucle cerrado con realimentación negativa unitaria se obtiene de la siguiente manera:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)}$$

Al operar esto nos da:

$$\frac{C(s)}{R(s)} = \frac{K(K_p s + K_i)}{s^2 + \left(\frac{KK_p + 1}{\tau}\right)s + \frac{KK_i}{\tau}}$$

Fuente: [19]

La forma del denominador es equivalente al de la ecuación general de un sistema de segundo orten la cual viene dada por:

$$s^2 + 2\zeta\omega_n s + \omega_n^2$$

donde ζ es el factor de amortiguamiento y ω_n es la frecuencia natural.

Comparando ambos denominadores, podemos obtener los valores de K_p y K_i:

$$K_p = \frac{2\,\zeta\omega_n\tau - 1}{K}$$

$$K_i = \frac{\omega_n^2\tau}{K}$$

Fuente: [19]

Dependiendo del valor que tome ζ el sistema tendrá diversos comportamientos, los cuales vamos a tratar a continuación:

- ζ=0 Sistema Oscilatorio
- 0< ζ <1 Sistema Subamortiguado
- ζ=1 Sistema Criticamente Amortiguado
- ζ>1 Sistema Sobre Amortiguado

Estos sistemas adoptan las siguientes respuestas en función de dichos factores:

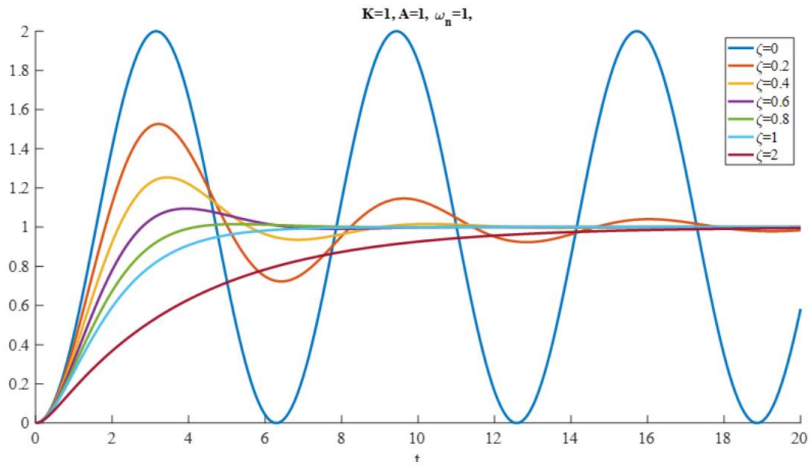


Figura 51. Variación del factor de amortiguamiento

Fuente: [20]

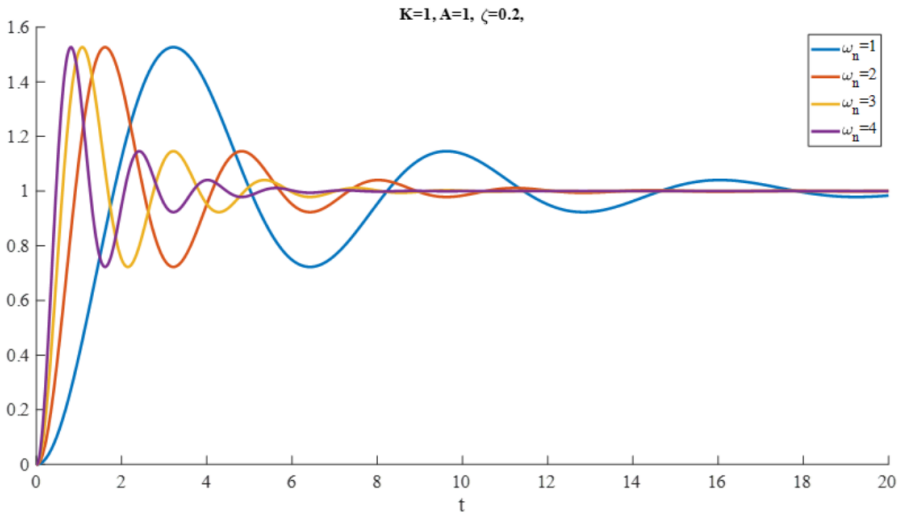


Figura 52. Variación de la frecuencia natural no amortiguada

Fuente: [20]

Los valores de ζ y ω_n los podemos obtener a partir de las especificaciones de tiempo de subida (T_s) y sobreoscilación (OS%):

$$\zeta = \frac{-\ln(OS\%)}{\sqrt{\pi^2 + \ln(OS\%)^2}}$$

$$\omega_n = \frac{4}{\zeta T_s}$$

Fuente: [19]

Una vez explicado como obtener dichos parámetros, los vamos a obtener por medio de Matlab, haciendo uso del comando "pidtool", tal y como se ilustra en la figura 53.

Introducimos la función de transferencia obtenida, seleccionamos controlador PI, tiempo de muestreo y las especificaciones deseadas, y obtenemos los siguientes parámetros:

$$K_p = 0.04825$$

$$K_i = 0.3199$$

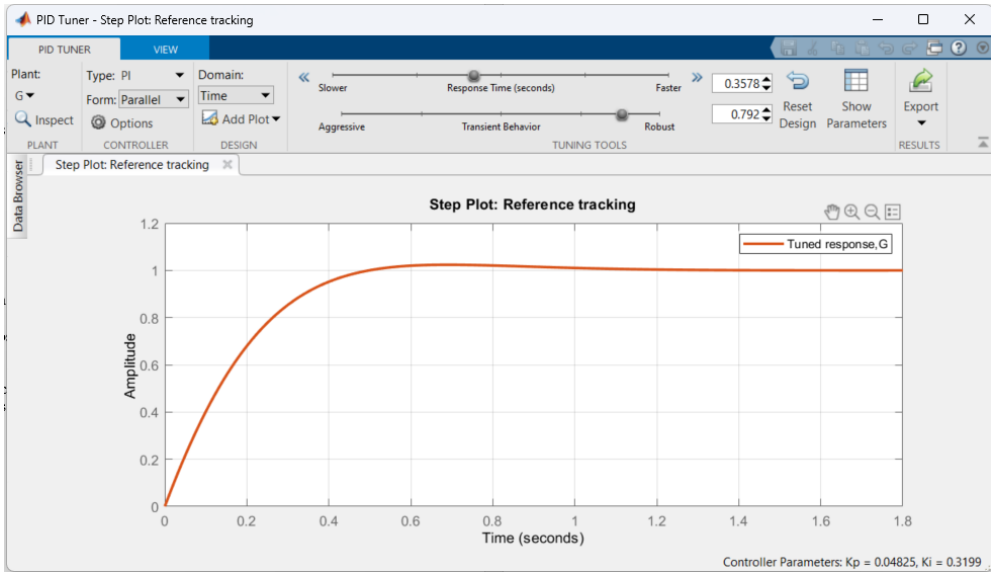


Figura 53. Sintonización de controlador PI mediante PID Tunner – Matlab

Fuente: Elaboración Propia

Una vez obtenidos dichos parámetros, procedemos a introducirlos en nuestro código, que ejecuta el bucle de control para cada motor con un tiempo de muestreo de 0.01 segundos. Dicho bucle de control es el siguiente:

```
void controlPI_sup() {

  rpm_s = medirRPM_sup();
  rpm_s_filt = (int)(alpha * rpm_s) + ((1 - alpha) * rpm_s_filt);
  rpm_s_filt_filt = (int)(alpha2 * rpm_s_filt) + ((1 - alpha2) * rpm_s_filt_filt);
  ekS = VEL_REF_SUP_W1 - rpm_s_filt_filt;
  eintegralS += ekS * Ts;
  pwm_valueS = kp * ekS + ki * eintegralS;

  if (pwm_valueS > 255)pwm_valueS = 255;
  if (pwm_valueS < 0)pwm_valueS = 0;

  Serial.print(VEL_REF_SUP_W1);
  Serial.print(",");
  Serial.print(rpm_s_filt_filt);
  Serial.print(",");
  Serial.print(ekS);
  Serial.print(",");
  Serial.print(pwm_valueS);
  Serial.print(",");
  controlador_mot_sup.TurnRight(pwm_valueS);
}
```

Al representar los valores obtenidos para la velocidad de referencia (azul), velocidad medida (rojo), error (verde) y señal de control pwm (naranja), obtenemos los siguientes resultados para la rueda superior:

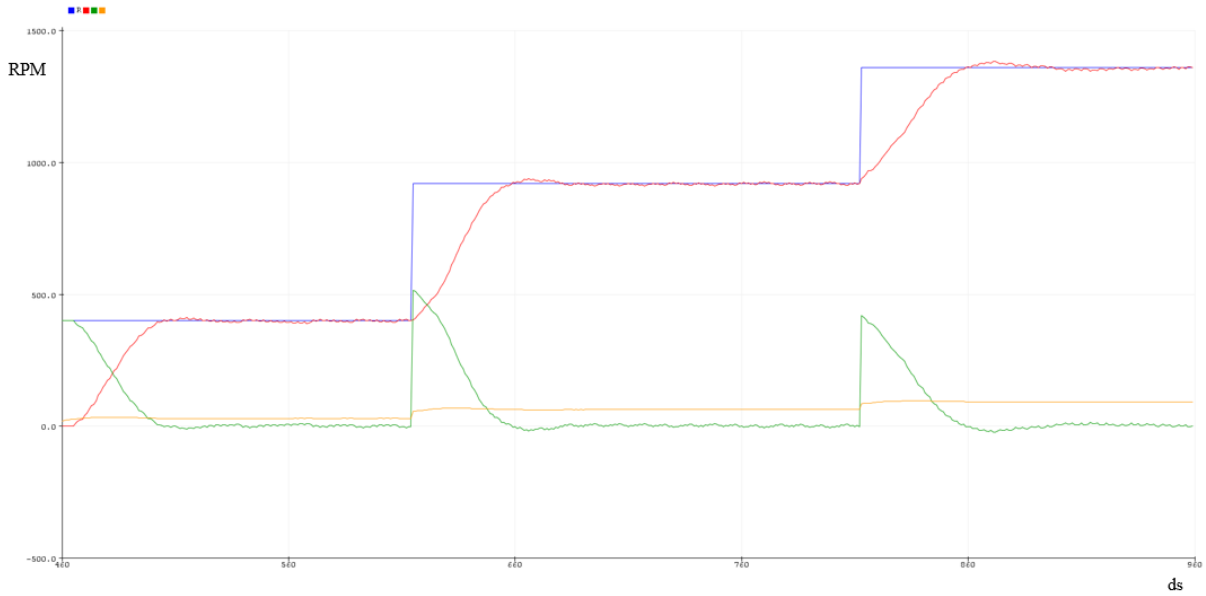


Figura 54. Resultados del control PI para la rueda superior

Fuente: Elaboración Propia

En la figura, observamos cómo al establecer una velocidad de referencia, el error empieza siendo dicha velocidad al estar la rueda parada, y el microcontrolador va ajustando la señal PWM hasta que el error se hace prácticamente nulo y la velocidad de la rueda leída por el sensor de línea está entorno a la referencia fijada. Al establecer otras velocidades de referencia, el microcontrolador vuelve a ajustar la velocidad.

Análogamente, podríamos obtener unos resultados similares para la rueda inferior.

Llegado a este punto, se procede a realizar unas pruebas experimentales para determinar la velocidad necesaria para lanzar las bolas a una distancia determinada.

5 PRUEBAS EXPERIMENTALES

En esta última sección, se llevan a cabo distintos lanzamientos de pelotas varianto el ángulo de lanzamiento vertical y las RPM de cada rueda, haciendo llegar la pelota a una distancia determinada.

Para empezar, en el siguiente enlace se puede observar un breve vídeo del robot en funcionamiento:

https://drive.google.com/file/d/1_gMLcO1Kv6345kdXauMp2hxTL5rzF_zy/view?usp=sharing

La distancia de lanzamiento varía dependiendo del ángulo de lanzamiento vertical (θ_1), la velocidad de salida de la pelota, que depende de las RPM a las que gire cada rueda, y además de la fuerza de compresión que se ejerce sobre la pelota. Dicha fuerza, no se ha calculado en este proyecto, empleando únicamente el resto de las variables. Sin embargo, a mayor sea dicha fuerza, más lejos será capaz de lanzar la pelota, hasta cierto punto en el que la compresión sea excesiva. Si la fuerza es insuficiente, igualmente el robot no será capaz de lanzar la pelota de manera óptima.

En la siguiente tabla se recogen los lanzamientos reazliados y los resultados obtenidos:

θ_1(º)	ω_2 (RPM)	ω_1 (RPM)	Distancia (metros)
15	1500	1500	10
15	1500	2000	12
20	2000	2000	15
20	2000	2500	17
20	2500	2500	20

Tabla 10. Resultados de lanzamiento

Fuente: Elaboración Propia

Estas distancias son aproximadas. Para obtenerlas, se ha tomado como referencia el ancho de la pista de fútbol sala de la ETSI (15 metros). Como se ha mencionado, la distancia varía dependiendo de la fuerza de compresión que se ejerce sobre la pelota, la cuál no ha sido calculada. Como puede observarse, para lanzar la pelota a 20 metros es necesario establecer el ángulo de lanamiento vertical en 20 grados y una velocidad de 2500 RPM en cada rueda.

CONCLUSIONES

A lo largo de este proyecto, se ha realizado el diseño de un robot lanzapelotas de tenis, el cual ha ido desarrollándose poco a poco hasta obtener la forma del robot construido monstrado. En dicho proceso, han ido surgiendo dificultades que se han afrontado para conseguir que el robot sea funcional.

Entre dichas dificultades destacan las siguientes:

-Sujección de los engranajes a los motores paso a paso. Al haberse diseñado manualmente los engranajes, no se ha realizado bien el sistema que sujeta los engranajes conductores con los motores paso a paso. Esto ha dado lugar a que dichos engranajes acaben "resbalando" debido a la fuerza a la que están sometidos. Esto supone que, si resbala el engranaje para el ángulo vertical, se pierde la posición de referencia deseada y la barra por donde entran las pelotas puede llegar a rozar con la rueda superior y dañarla.

-Los drivers para los motores paso a paso seleccionados funcionan correctamente, pero debido a que su tensión de alimentación es de al menos 20V, se ha tenido que aumentar la tensión mediante un convertidor DC-DC elevador, lo cuál no estaba previsto en un principio, y supone un consumo extra de energía para nuestro robot.

-Elección de los motores DC y su integración en el diseño. La elección de dichos motores ha resultado complicada, pues sus características no son tan fáciles de encontrar en internet, así como unas ruedas de frición que se adapten a ellos. Finalmente, se han aprovechado los motores extraídos de una antigua máquina lanzapelotas, de manera que la integración con las ruedas de fricción también extraídas de ella se ha resuelto de forma sencilla. Además, posteriormente, por errores de diseño en la barra que sujeta los motores, ha sido complicado poner estos en su lugar debido a los cables que salen de los motores están posicionados por laterales, de tal manera que para encajar los motores no ha sido posible hacerlo como se esperaba y se ha tenido que dañar la barra para colocarlos.

-Calibración de los sensores de línea. Dado que es necesario leer la velocidad de las ruedas para realizar su posterior control en bucle cerrado, se eligieron estos sensores para realizar dicha función. El inconveniente es que dichos sensores tienen una distancia de lectura bastante corta y para un funcionamiento óptimo es necesario que estén a menos de 1.5 centímetros de las ruedas, lo cual supone una distancia que no estaba prevista y que podría suponer que los sensores se dañen si se sueltan o rozan con las ruedas. Por tanto, tal vez hubiera sido mejor emplear otro tipo de sensor.

Sin embargo, el mayor inconveniente del robot construido es su dificultad para desplazarlo y su fragilidad.

A pesar de las dificultades mencionadas, se ha conseguido realizar un robot capaz de posicionar los ángulos de lanzamiento y las velocidades de las ruedas en la referencia deseada (en bucle cerrado), la cuál es establecida por vía bluetooth con la aplicación móvil desarrollada, tal y como se ha explicado en este proyecto.

Como posibles trabajos futuros, se plantea ampliar el robot mediante otro sistema, el cual consiste en un láser que sea capaz de seguir la posición en la que caerá la pelota, así como el correspondiente estudio y calibración del robot para que realice el comportamiento que describa el laser. Además, podría llevarse a cabo un estudio de la distancia de lanzamiento incluyendo una variación en la fuerza de compresión ejercida sobre la pelota

Por último, agradecer a mis profesores de la Escuela Técnica Superior de Ingenieros de Sevilla todos los conocimientos transmitidos.

Así mismo, agradecer a mi familia el apoyo mostrado durante estos años, sin el cuál no hubiera sido posible nada de esto. Os quiero mucho.

REFERENCIAS

[1] Departamento de Ingeniería Mecánica y Fabricación, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Teoría de Máquinas y Mecanísmos.

[2] «https://es.aliexpress.com/item/32452872946.html?spm=a2g0o.productlist.0.0.3e1b5869eSvsgb&algo_pvid=1ac709d2-0737-4833-b332-2a989bf2cc18&aem_p4p_detail=202209221013431972550918333817000058622 9&algo_exp_id=1ac709d2-0737-4833-b332-2a989bf2cc18-1&pdp_ext_f=%7,» [En línea].

[3] «https://allhealthalltennis.wordpress.com/2013/06/14/tennis-physics-topspin/,» [En línea].

[4] «https://www.lawebdeltenis.net/tenis-efectos/#:~:text=Efectos%20en%20el%20saque%3A&text=El%20cortado%20se%20pega%20de,derecha%20(para%20un%20diestro).,» [En línea].

[5] «https://spanish.alibaba.com/p-detail/Sports-60794736602.html,» [En línea].

[6] «https://motoresygeneradores.com/como-funciona-un-motor-de-corriente-continua/#:~:text=El%20principio%20de%20funcionamiento%20de,encuentra%20montado%20en%20un%20eje.,» [En línea].

[7] «https://tienda.bricogeek.com/motores/117-servomotor-estandar-3001hb.html,» [En línea].

[8] «https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos,» [En línea].

[9] «https://plasticos-brello.com/material/pla-acido-polilactico/,» [En línea].

[10] «https://shop.mleon.com/baterias-plomo/43829-bateria-plomo-12v-9ah-151x65x94.html?search_query=bateria+12v&results=604,» [En línea].

[11] «https://www.amazon.es/Evemodel-tablero-distribuci%C3%B3n-energ%C3%ADa-entradas/dp/B0814B5P5M/ref=asc_df_B0814B5P5M/?tag=googshopes-21&linkCode=df0&hvadid=529957738534&hvpos=&hvnetw=g&hvrand=6147517353758755526&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlo,» [En línea].

[12] «https://www.amazon.es/Convertidor-Transformador-Regulador-Suministrar-Adaptador/dp/B07PZBZW9H/ref=sr_1_17?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=33RJTLD9VVCIV&keywords=convertidor%2Bdc-dc%2Belevador&qid=1682191280&sprefix=convertidor%2Bdc-dc%2Bel,» [En línea].

[13] Departamento Ingeniería Electrónica, Transparecias de Electrónica de Potencia, 2019.

[14] «https://descubrearduino.com/arduino-uno/,» [En línea].

[15] «https://naylampmechatronics.com/inalambrico/43-modulo-bluetooth-hc05.html,» [En línea].

[16] «https://www.luisllamas.es/controla-motores-de-gran-potencia-con-arduino-y-bts7960/,» [En línea].

[17] «https://www.infineon.com/dgdl/BTS7960_Datasheet.pdf?folderId=db3a304412b407950112b408e8c90004&fileId=db3a304412b407950112b43945006d5d,» [En línea].

[18] «https://www.electrohobby.es/posicion/324-sigue-lineas-1ch.html,» [En línea].

[19] E. Kossove, PI Controller Research and Design, 2018.

[20] «https://controlautomaticoeducacion.com/control-realimentado/sistemas-de-segundo-orden/,» [En línea].

ANEXO 1: CÓDIGO

```
// Incluimos las librerías necesarias
#include <Servo.h>
#include <AccelStepper.h>
#include <SoftwareSerial.h>
#include "BTS7960.h"

// Creamos un objeto de la clase SoftwareSerial para la comunicación serial
SoftwareSerial BTserial(A0, A1);

// Definimos los pines del motor NEMA 23 Horizontal
#define DIR_PIN_h A2
#define STEP_PIN_h A3

// Definimos los pines del motor NEMA 23 Vertical
#define DIR_PIN_v 4
#define STEP_PIN_v 5

// Creamos dos objetos AccelStepper con el driver DM542 y los pines correspondientes
AccelStepper motor_h(AccelStepper::DRIVER, STEP_PIN_h, DIR_PIN_h);
AccelStepper motor_v(AccelStepper::DRIVER, STEP_PIN_v, DIR_PIN_v);

// Definir los parámetros del driver DM542
#define STEPS_PER_REV 400
#define MICROSTEPS 16

// Definimos las velocidades y aceleración máximas del motor NEMA 23
#define MAX_SPEED 1000.0
#define MAX_ACCEL 500.0

// Creamos una instancia de la clase Servo para controlar el servo
Servo servo;                    // Crear objeto servo
int servoPin = 8;               // Pin del servo
int delayTime = 300;           // Tiempo en milisegundos que el servo se mantiene abierto

// Definir variable para almacenar la posición actual de los motores
int current_position1 = 0;

// Definir posiciones guardadas para -15, 0 y 15 grados para el motor horizontal
const int pos1 = -139; // -15° en pasos
const int pos2 = 0;     // 0° en pasos
const int pos3 = 139;  // 15° en pasos

// Definir posiciones guardadas para 0, 15 y 22 grados para el motor vertical
const int pos4 = 0; // 0° en pasos
const int pos5 = 139;     // 15° en pasos
const int pos6 = 220;  // 22° en pasos

// Definimos los pines de entrada y salida del driver BTS7960B para el motor superior
const int EN_SUP = 9;
const int L_PWM_SUP = 10;
const int R_PWM_SUP = 11;

// Definimos los pines de entrada y salida del driver BTS7960B para el motor inferior
const int EN_INF = 12;
const int L_PWM_INF = 7;
const int R_PWM_INF = 6;

//Variables para los sensores de linea
const int sensor_vel_sup = 2;
const int sensor_vel_inf = 3;

//Variables para comunicación con Matlab
//byte buffer_tx[4];
//byte buffer_rx[3];

//Variables para filtro EMA
float alpha = 0.2;
float alpha2 = 0.1;
```

```
// Variables para almacenar la velocidad en rpm de cada rueda
float rpm_s = 0.0;
int rpm_s_filt = rpm_s;
int rpm_s_filt_filt = rpm_s_filt;
volatile int counter_s = 0;

float rpm_i = 0;
int rpm_i_filt = rpm_i;
int rpm_i_filt_filt = rpm_i_filt;
int counter_i = 0;


long prev_t = 0;
float eprev = 0;


//Creamos controlador del los motores superior e inferior
BTS7960 controlador_mot_sup(EN_SUP, L_PWM_SUP, R_PWM_SUP);
BTS7960 controlador_mot_inf(EN_INF, L_PWM_INF, R_PWM_INF);

//Variables para almacenar las velociadades
String com_vel;
int VEL_REF_SUP_W1;
int VEL_REF_INF_W2;
int VEL_MOT_SUP_W1;
int VEL_MOT_INF_W2;

//Variables para control en bucle cerrado
bool control = 0;
int pwm_valueS = 0;
int pwm_valueI = 0;

float ekS = 0;
float ekI = 0;

float eintegralS = 0;
float eintegralI = 0;


float Ts = 0.01;
float kp = 0.04825;
float ki = 0.3199;

// Funcion que se ejecuta durante cada interrupion del sensor superior
void interrupcion0()
{

  counter_s++;

}

// Funcion que se ejecuta durante cada interrupion del sensor inferior
void interrupcion1()
{

  counter_i++;

}

void setup() {
  // Iniciar la comunicación serial
  Serial.begin(57600);
  // Iniciar la comunicación bluetooth
  BTserial.begin(9600);

  //Configuramos pines para recepción y transmisión bluetooth
  pinMode(A0, INPUT);
  pinMode(A1, OUTPUT);

  //Configuramos pines de los sensores de línea
  pinMode(sensor_vel_sup, INPUT);
  pinMode(sensor_vel_inf, INPUT);
  attachInterrupt(digitalPinToInterrupt(sensor_vel_sup), interrupcion0, RISING); // Interrupcion
0 (pin2)
  attachInterrupt(digitalPinToInterrupt(sensor_vel_inf), interrupcion1, RISING); // Interrupcion
1 (pin3)
```

```
  // Configuramos el pin de dirección del motor NEMA 23 horizontal como salida
  pinMode(DIR_PIN_h, OUTPUT);
  // Configuramos el pin de paso del motor NEMA 23 horizontal como salida
  pinMode(STEP_PIN_h, OUTPUT);
  // Configuramos el pin de dirección del motor NEMA 23 vertical como salida
  pinMode(DIR_PIN_v, OUTPUT);
  // Configuramos el pin de paso del motor NEMA 23 vertical como salida
  pinMode(STEP_PIN_v, OUTPUT);
  // Establecemos la velocidad máxima del motor NEMA 23 horizontal
  motor_h.setMaxSpeed(MAX_SPEED);
  // Establecemos la aceleración máxima del motor NEMA 23 horizontal
  motor_h.setAcceleration(MAX_ACCEL);
  // Establecemos la velocidad máxima del motor NEMA 23 vertical
  motor_v.setMaxSpeed(MAX_SPEED);
  // Establecemos la aceleración máxima del motor NEMA 23 vertical
  motor_v.setAcceleration(MAX_ACCEL);
  // Establecemos la posición actual de los motores como la posición inicial guardada
  motor_h.setCurrentPosition(pos2);
  motor_v.setCurrentPosition(pos4);

  // Conectar el servo al pin correspondiente y cerrar inicialmente
  servo.attach(servoPin);
  servo.write(115);

  //Configurar pines de motores DC superior e inferior
  pinMode(EN_SUP, OUTPUT);
  pinMode(L_PWM_SUP, OUTPUT);
  pinMode(R_PWM_SUP, OUTPUT);

  pinMode(EN_INF, OUTPUT);
  pinMode(L_PWM_INF, OUTPUT);
  pinMode(R_PWM_INF, OUTPUT);

}



//Función para medir velocidad del motor superior
float medirRPM_sup() {
  float rpm = 0;
  rpm = 600 * counter_s;
  counter_s = 0;
  return rpm;
}

//Función para medir velocidad del motor inferior
float medirRPM_inf() {
  float rpm = 0;
  rpm = 600 * counter_i;
  counter_i = 0;
  return rpm;
}

//Función para realizar el control PI del motor superior
void controlPI_sup() {

  rpm_s = medirRPM_sup();
  rpm_s_filt = (int)(alpha * rpm_s) + ((1 - alpha) * rpm_s_filt);
  rpm_s_filt_filt = (int)(alpha2 * rpm_s_filt) + ((1 - alpha2) * rpm_s_filt_filt);
  ekS = VEL_REF_SUP_W1 - rpm_s_filt_filt;
  eintegralS += ekS * Ts;
  pwm_valueS = kp * ekS + ki * eintegralS;

  if (pwm_valueS > 255)pwm_valueS = 255;
  if (pwm_valueS < 0)pwm_valueS = 0;

  Serial.print(VEL_REF_SUP_W1);
  Serial.print(",");
  Serial.print(rpm_s_filt_filt);
  Serial.print(",");
  Serial.print(ekS);
  Serial.print(",");
  Serial.print(pwm_valueS);
  Serial.print(",");
```

```
  controlador_mot_sup.TurnRight(pwm_valueS);
}

//Función para realizar el control PI del motor inferior
void controlPI_inf() {

  rpm_i = medirRPM_inf();
  rpm_i_filt = (int)(alpha * rpm_i) + ((1 - alpha) * rpm_i_filt);
  rpm_i_filt_filt = (int)(alpha2 * rpm_i_filt) + ((1 - alpha2) * rpm_i_filt_filt);
  ekI = VEL_REF_INF_W2 - rpm_i_filt_filt;
  eintegralI += ekI * Ts;
  pwm_valueI = kp * ekI + ki * eintegralI;

  if (pwm_valueI > 255)pwm_valueI = 255;
  if (pwm_valueI < 0)pwm_valueI = 0;

  Serial.print(VEL_REF_INF_W2);
  Serial.print(",");
  Serial.print(rpm_i_filt_filt);
  Serial.print(",");
  Serial.print(ekI);
  Serial.print(",");
  Serial.println(pwm_valueI);
  controlador_mot_inf.TurnRight(pwm_valueI);
}

void loop() {

  //Activamos drivers de los motores DC
  controlador_mot_sup.Enable();
  controlador_mot_inf.Enable();

  if (control) {
    controlPI_sup();
    controlPI_inf();
  }

  if (!control) {
    rpm_s = medirRPM_sup();
    rpm_s_filt = (int)(alpha * rpm_s) + ((1 - alpha) * rpm_s_filt);
    rpm_s_filt_filt = (int)(alpha2 * rpm_s_filt) + ((1 - alpha2) * rpm_s_filt_filt);
    Serial.println(rpm_s_filt_filt);

    rpm_i = medirRPM_inf();
    rpm_i_filt = (int)(alpha * rpm_i) + ((1 - alpha) * rpm_i_filt);
    rpm_i_filt_filt = (int)(alpha2 * rpm_i_filt) + ((1 - alpha2) * rpm_i_filt_filt);
    Serial.println(rpm_i_filt_filt);
    //byte rpm_s_L=rpm_s_filt_filt & 0xff;
    //byte rpm_s_H=(rpm_s_filt_filt>>8) & 0xff;
    //   buffer_tx[0]='R';
    //   buffer_tx[1]=rpm_s_L;
    //   buffer_tx[2]=rpm_s_H;
    //   buffer_tx[3]='\n';
    //   Serial.write(buffer_tx,4);
  }

  delay(100);
  // Si se recibe algo por Bluetooth
  if (BTserial.available()) {
    // Leer el comando recibido por Bluetooth
    char command = BTserial.read();

    //Iniciamos casuistica
    switch (command) {
      case 'a':
        servo.write(80);              // Abrir el servo girándolo 90 grados
        delay(delayTime);             // Esperar un tiempo determinado
        servo.write(115);
        delay(100);// Cerrar el servo
        break;

      case 'b':
        // Movemos el motor horizontal a la posición guardada para -15°
        motor_h.moveTo(pos1);
        // Avanzamos el motor hasta llegar a la posición deseada
```

```
  while (motor_h.distanceToGo() != 0) {
    motor_h.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_h.setCurrentPosition(pos1);
  delay(100);
  break;

case 'c':
  // Movemos el motor horizontal a la posición guardada para 0°
  motor_h.moveTo(pos2);
  // Avanzamos el motor hasta llegar a la posición deseada
  while (motor_h.distanceToGo() != 0) {
    motor_h.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_h.setCurrentPosition(pos2);
  delay(100);
  break;

case 'd':
  // Movemos el motor horizontal a la posición guardada para 15°
  motor_h.moveTo(pos3);
  // Avanzamos el motor hasta llegar a la posición deseada
  while (motor_h.distanceToGo() != 0) {
    motor_h.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_h.setCurrentPosition(pos3);
  delay(100);
  break;

case 'e':
  // Movemos el motor vertical a la posición guardada para 0°
  motor_v.moveTo(pos4);
  // Avanzamos el motor hasta llegar a la posición deseada
  while (motor_v.distanceToGo() != 0) {
    motor_v.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_v.setCurrentPosition(pos4);
  delay(100);
  break;

case 'f':
  // Movemos el motor vertical a la posición guardada para 15°
  motor_v.moveTo(pos5);
  // Avanzamos el motor hasta llegar a la posición deseada
  while (motor_v.distanceToGo() != 0) {
    motor_v.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_v.setCurrentPosition(pos5);
  delay(100);
  break;

case 'g':
  // Movemos el motor a la posición guardada para 30°
  motor_v.moveTo(pos6);
  // Avanzamos el motor hasta llegar a la posición deseada
  while (motor_v.distanceToGo() != 0) {
    motor_v.run();
  }
  // Establecemos la posición actual del motor como la posición guardada
  motor_v.setCurrentPosition(pos6);
  delay(100);
  break;

case 'h':
  // Actualizar velocidad motor superior
  while (BTserial.available()) {
    if (BTserial.available() > 0) { // Si se recibe algo por Bluetooth
      char letra = BTserial.read();
      com_vel += letra;
    }
  }
```

```
      if (com_vel.length() > 0) {
        VEL_MOT_SUP_W1 = com_vel.toInt(); //Convertimos a valor numerico
        control = false;
        controlador_mot_sup.TurnRight(VEL_MOT_SUP_W1);
        delay(100);
      }
      com_vel = "";
      break;

    case 'i':
      // Actualizar velocidad motor inferior
      while (BTserial.available()) {
        if (BTserial.available() > 0) { // Si se recibe algo por Bluetooth
          char letra = BTserial.read();
          com_vel += letra;
        }
      }
      if (com_vel.length() > 0) {
        VEL_MOT_INF_W2 = com_vel.toInt(); //Convertimos a valor numerico
        control = false;
        controlador_mot_inf.TurnRight(VEL_MOT_INF_W2);
        delay(100);
      }
      com_vel = "";
      break;

    case 'j':
      // Actualizar referencia velocidad motor superior bucle cerrado
      while (BTserial.available()) {
        if (BTserial.available() > 0) { // Si se recibe algo por Bluetooth
          char letra = BTserial.read();
          com_vel += letra;
        }
      }
      if (com_vel.length() > 0) {
        VEL_REF_SUP_W1 = com_vel.toInt(); //Convertimos a valor numerico
        control = true;
      }
      com_vel = "";
      delay(100);
      break;
    case 'k':
      // Actualizar referencia velocidad motor inferior bucle cerrado
      while (BTserial.available()) {
        if (BTserial.available() > 0) { // Si se recibe algo por Bluetooth
          char letra = BTserial.read();
          com_vel += letra;
        }
      }
      if (com_vel.length() > 0) {
        VEL_REF_INF_W2 = com_vel.toFloat(); //Convertimos a valor numerico
        control = true;
      }
      com_vel = "";
      delay(100);
      break;
  }
 }

}
```

ANEXO 2: PLANOS

Plano nº 1.1: Robot Lanzapelotas

Plano nº 1.2: Robot Lanzapelotas - 2

Plano nº 1.3: Robot Lanzapelotas - Estructura

Plano nº 2.1: Sistema de lanzamiento

Plano nº 2.2: Sistema de lanzamiento – 3 vistas estándar

Plano nº 2.3: Sistema de lanzamiento - 1

Plano nº 2.4: Engranaje para ángulo de lanzamiento horizontal

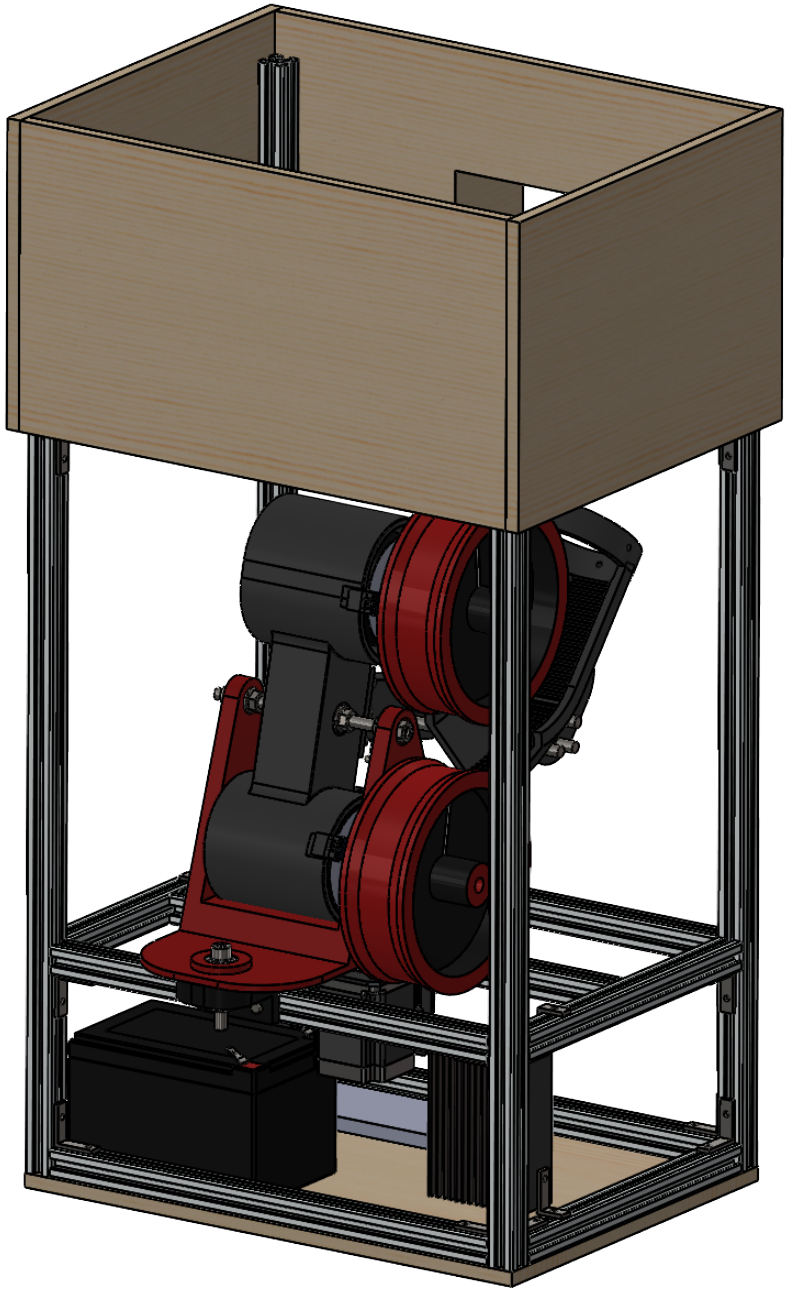
Plano nº 2.5: Sistema de lanzamiento - 2

Plano nº 2.6: Engranaje para ángulo de lanzamiento vertical

Plano nº 3.1: Sistema de alimentación

Plano nº 3.2: Caja de alimentación

Plano nº 4.1: Esquema de Conexiones



SOLIDWORKS

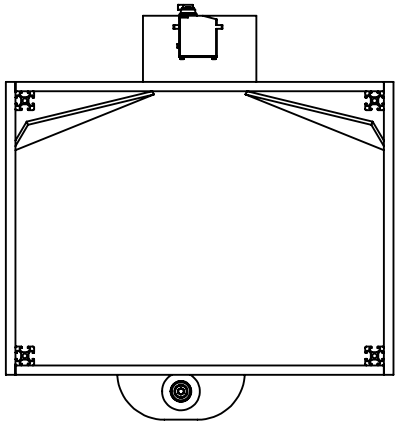
Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

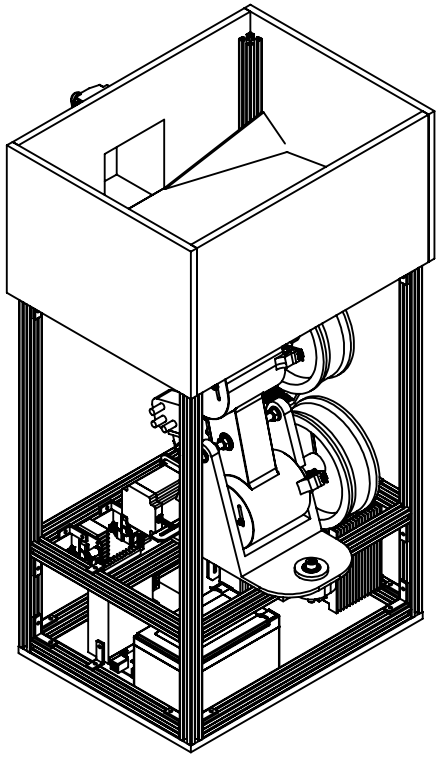
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

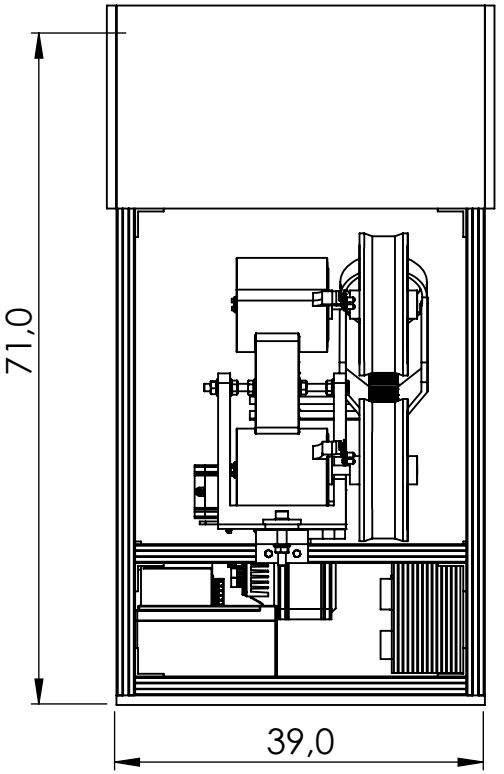
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

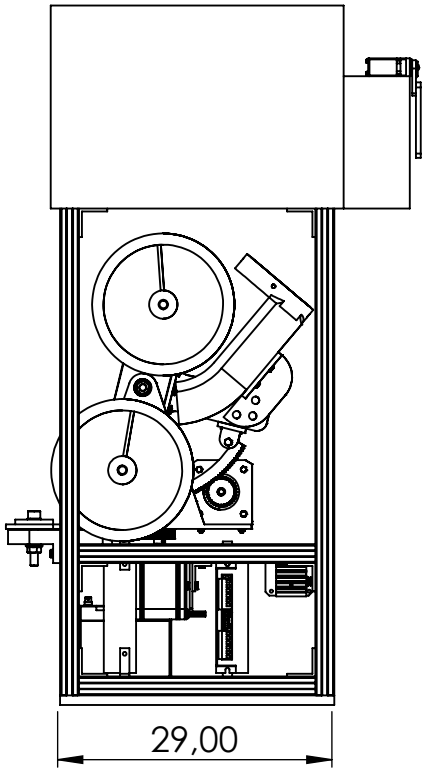
ROBOT LANZAPELOTAS

ESCALA 1:5	PLANO Nº
UNIDADES EN CM	1.1









SOLIDWORKS

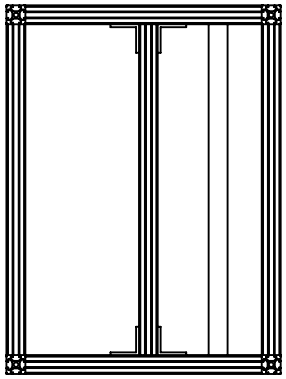
Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

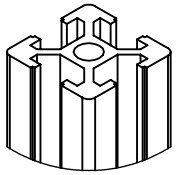
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

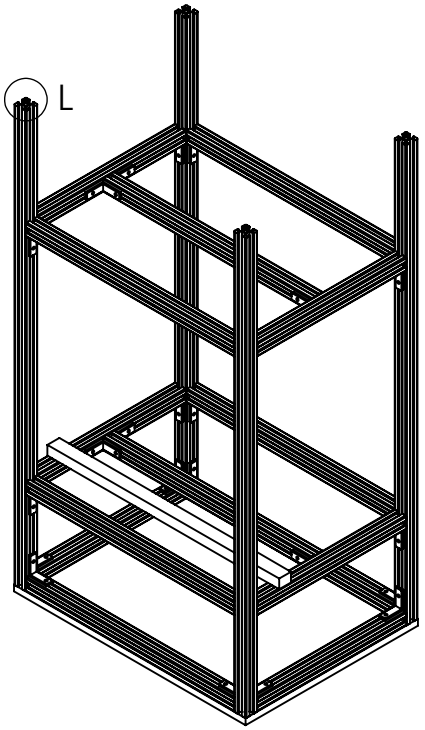
ROBOT LANZAPELOTAS - 2

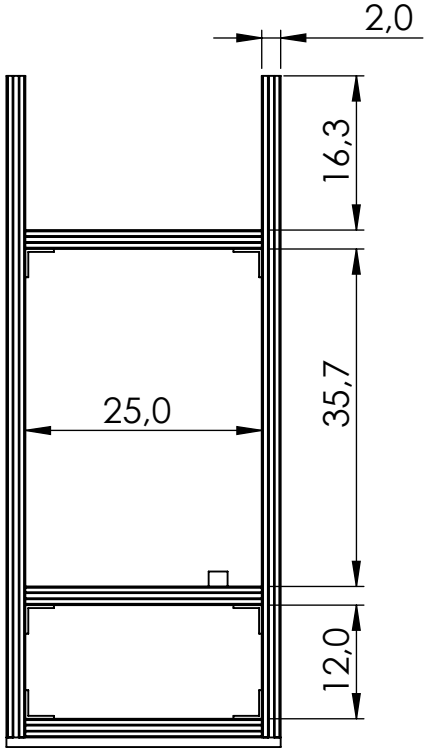
ESCALA 1:8	PLANO Nº
UNIDADES EN CM	1.2

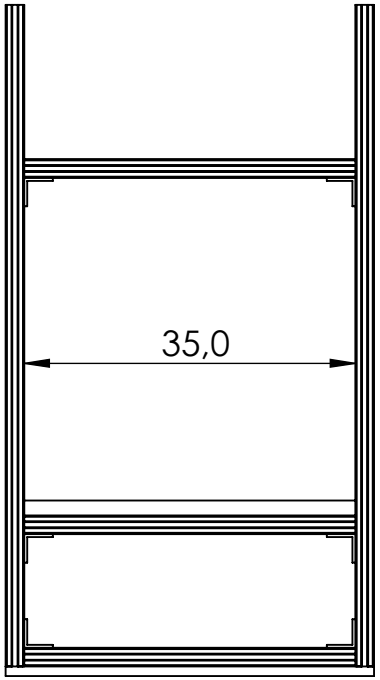




DETALLE L







Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

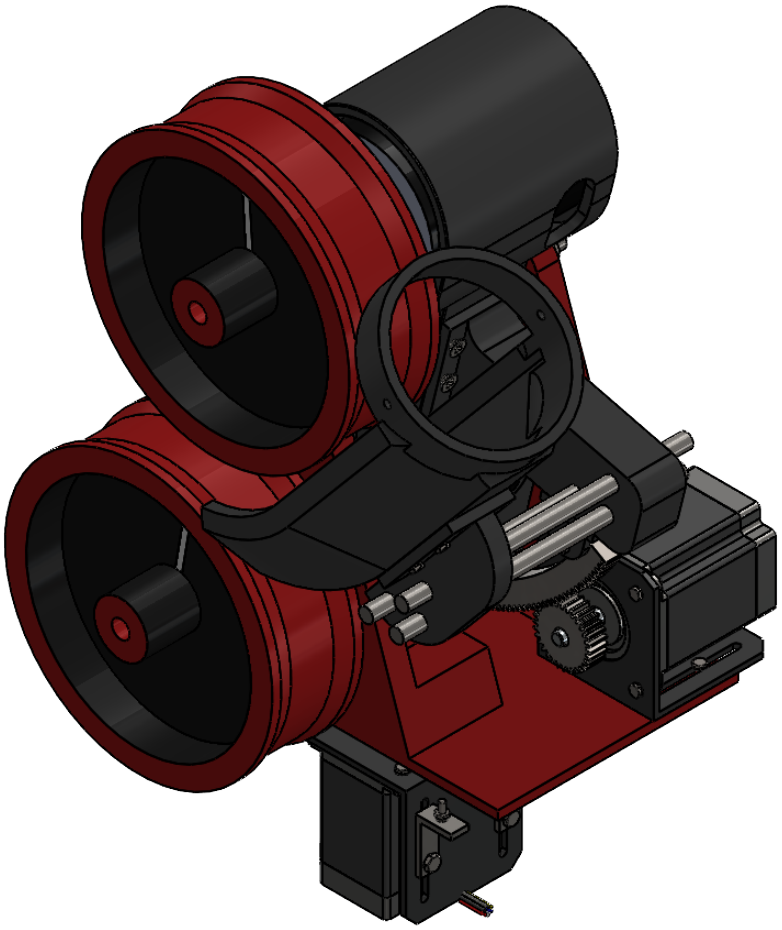
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

ROBOT LANZAPELOTAS -ESTRUCTURA

ESCALA 1:8	PLANO Nº
UNIDADES EN CM	1.3

SOLIDWORKS



Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

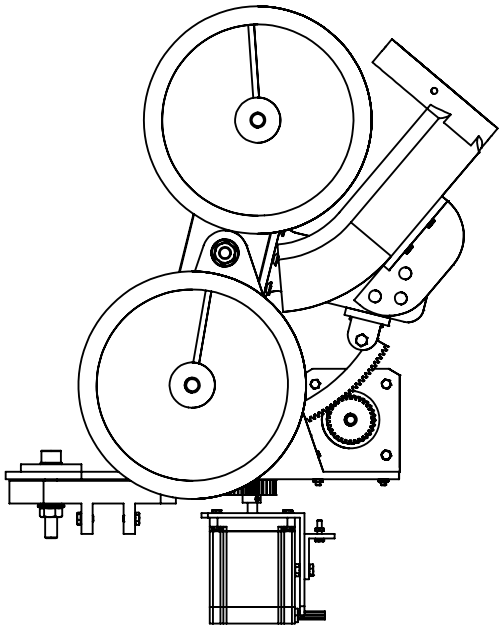
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

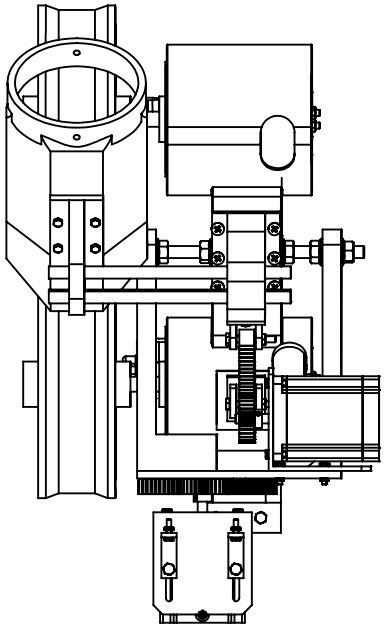
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

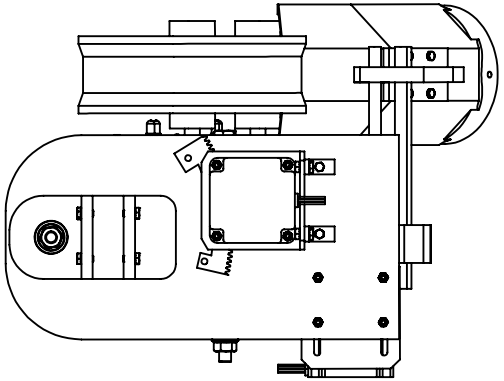
SISTEMA DE LANZAMIENTO

ESCALA 1:3	PLANO Nº
UNIDADES EN CM	2.1

SOLIDWORKS







Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

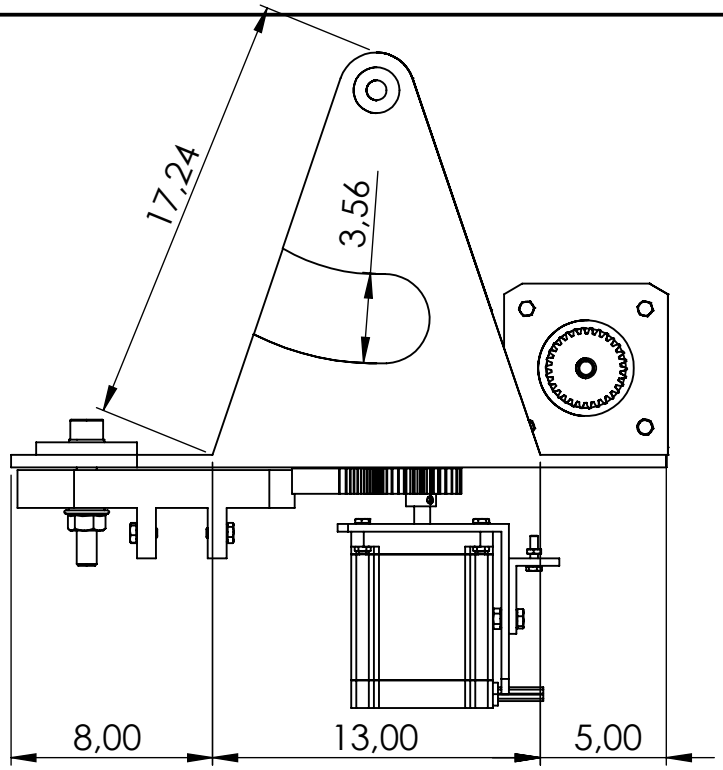
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

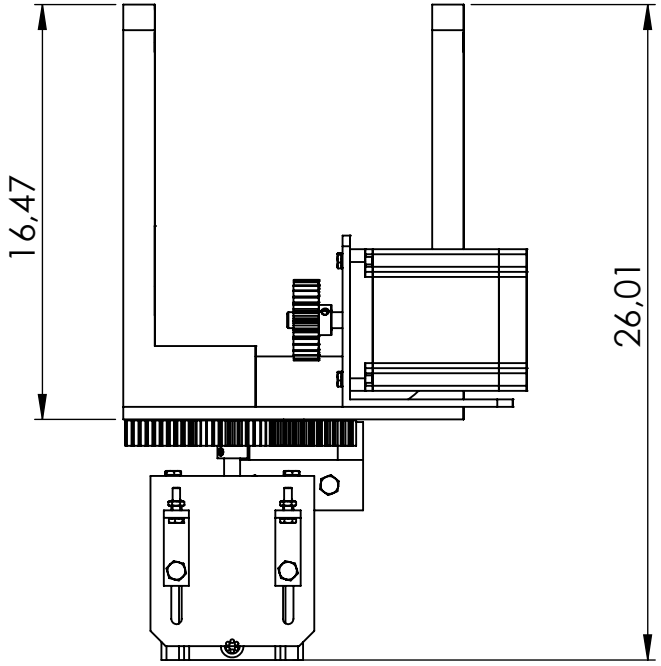
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

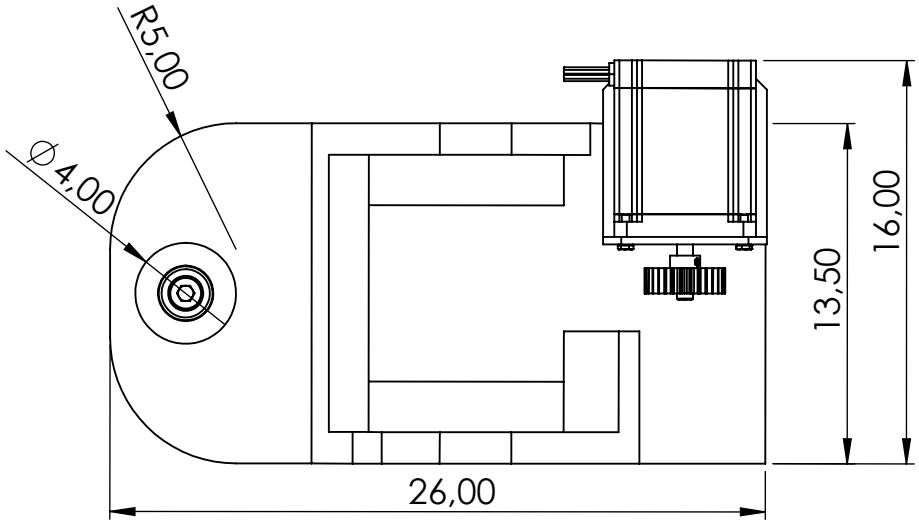
SISTEMA DE LANZAMIENTO -
3 VISTAS ESTÁNDAR

ESCALA 1:05	PLANO Nº
UNIDADES EN CM	2.2

SOLIDWORKS







SOLIDWORKS

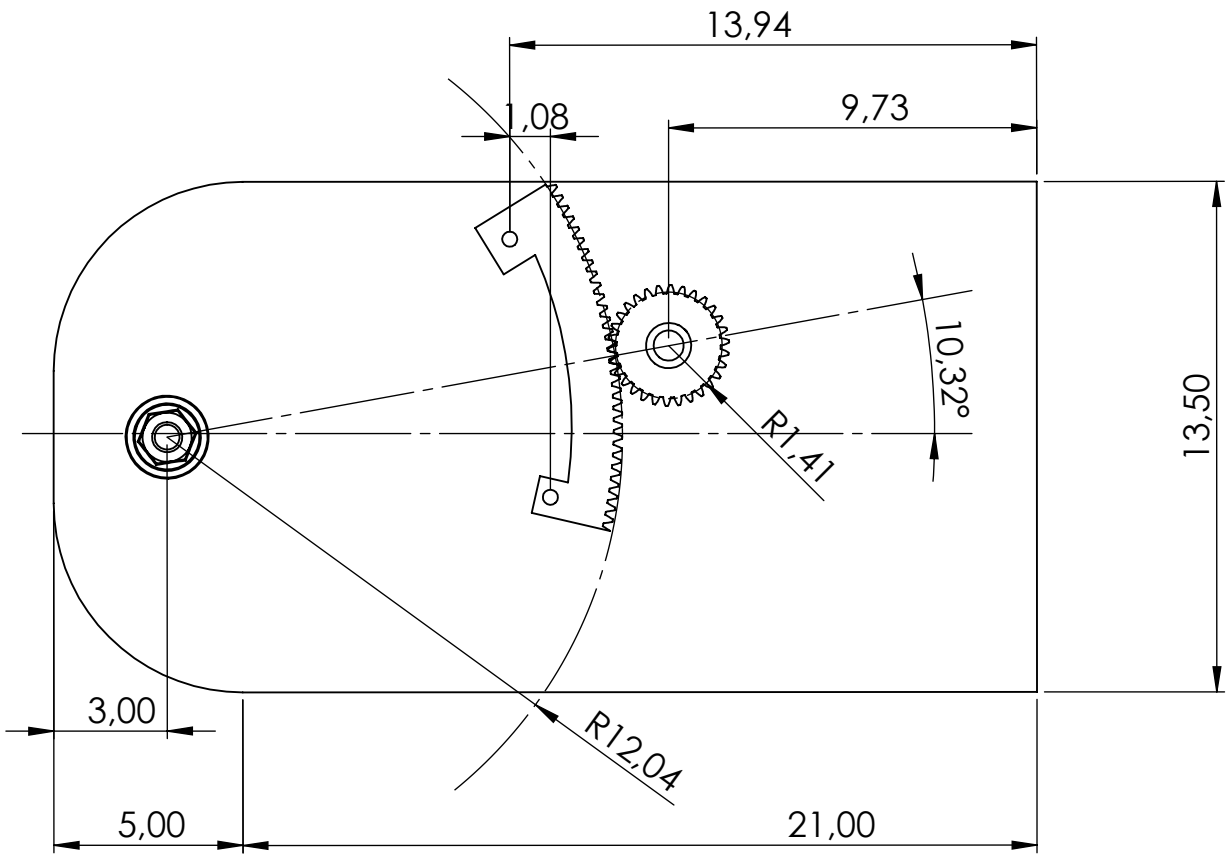
Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

SISTEMA DE LANZAMIENTO - 1

ESCALA 1:3	PLANO Nº
UNIDADES EN CM	2.3



SOLIDWORKS

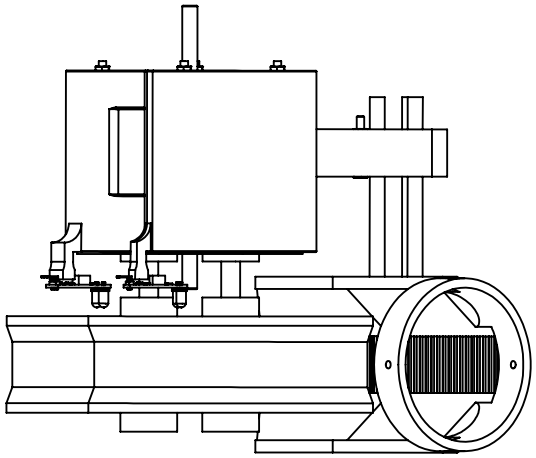
Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

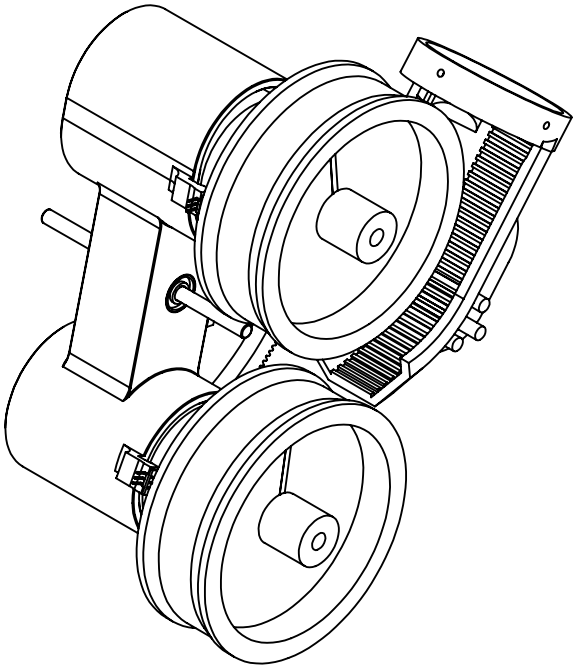
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

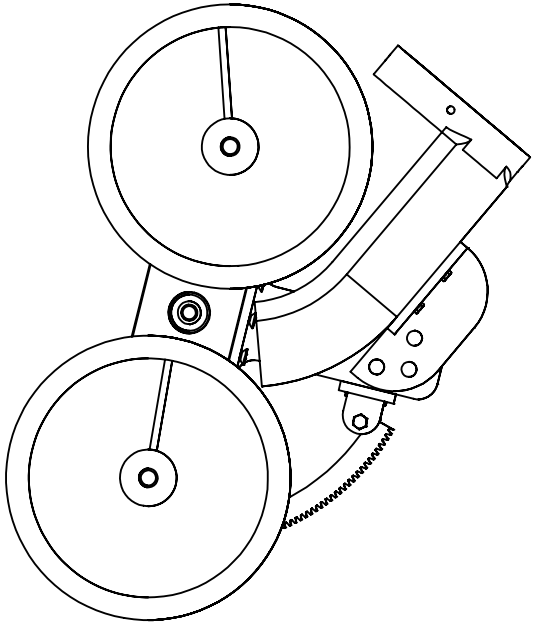
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

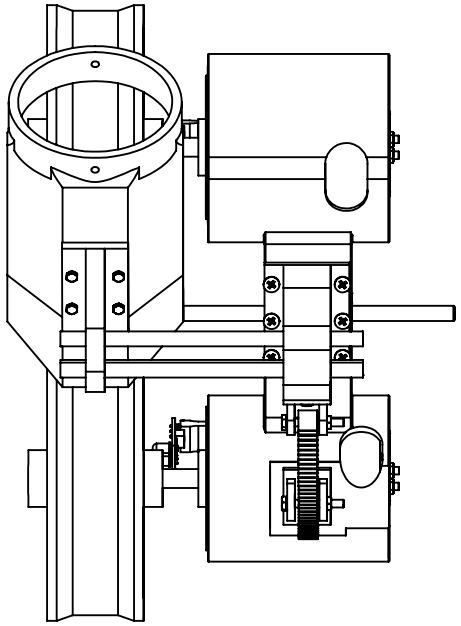
ENGRANAJE PARA ÁNGULO DE
LANZAMIENTO HORIZONTAL

ESCALA 1:2	PLANO Nº
UNIDADES EN CM	2.4









Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

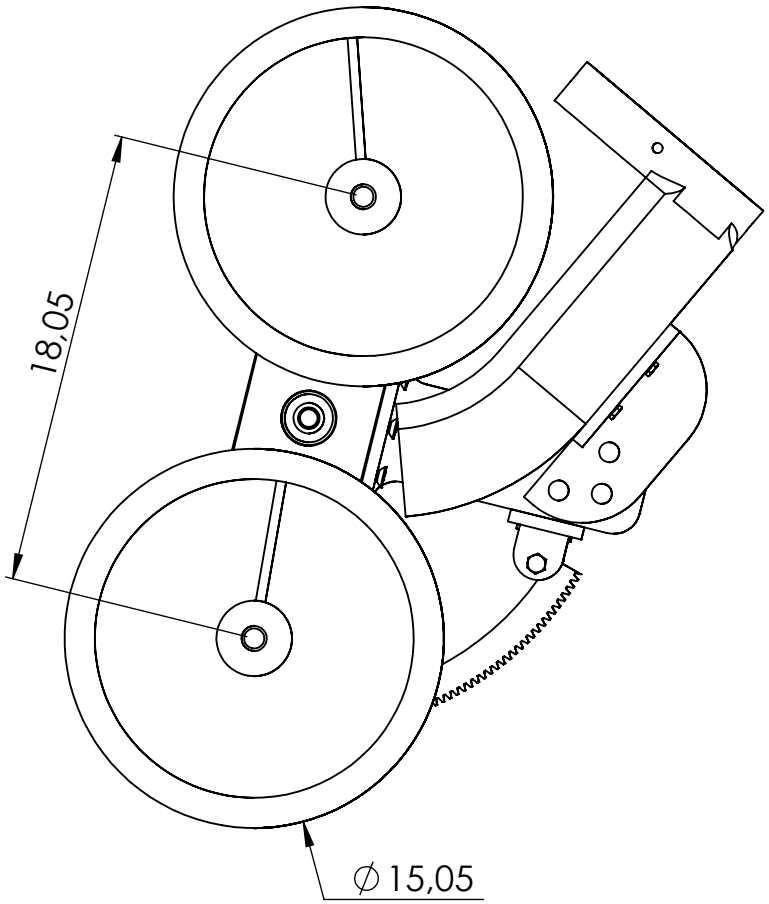
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

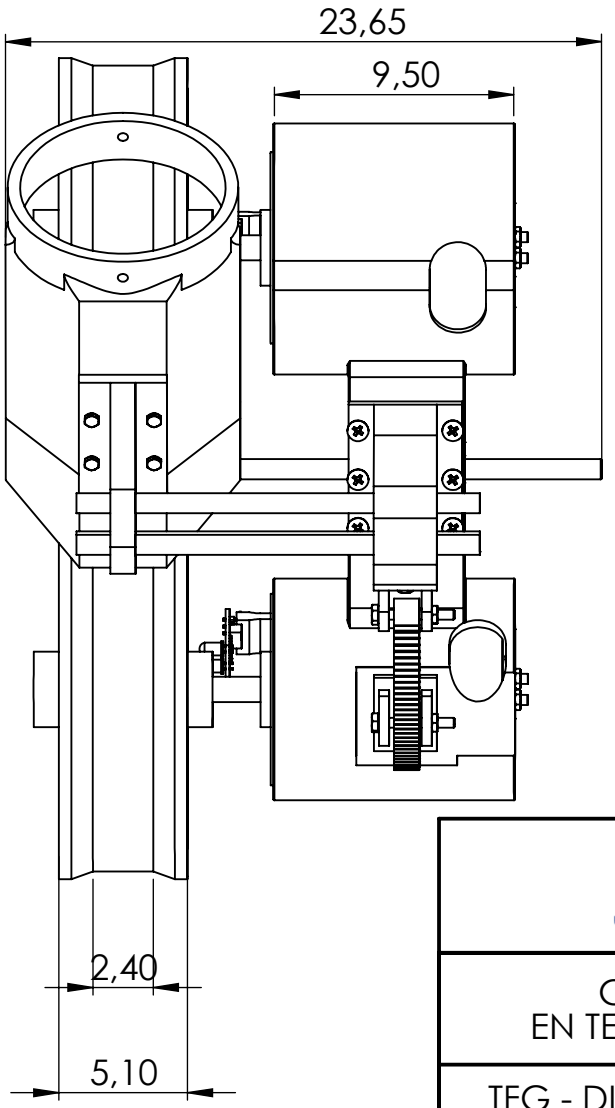
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

SISTEMA DE LANZAMIENTO - 2

ESCALA 1:4	PLANO Nº
UNIDADES EN CM	2.5

SOLIDWORKS





Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

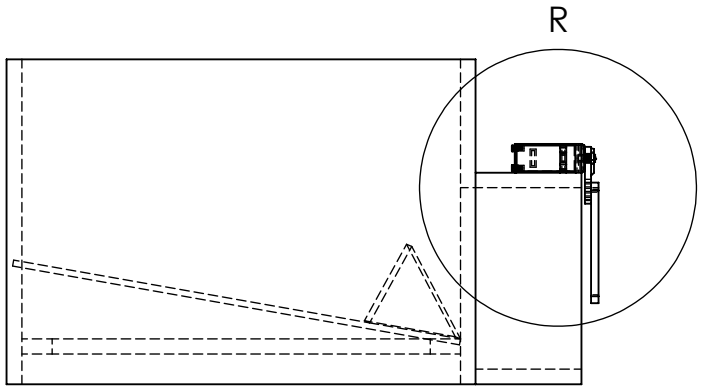
GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

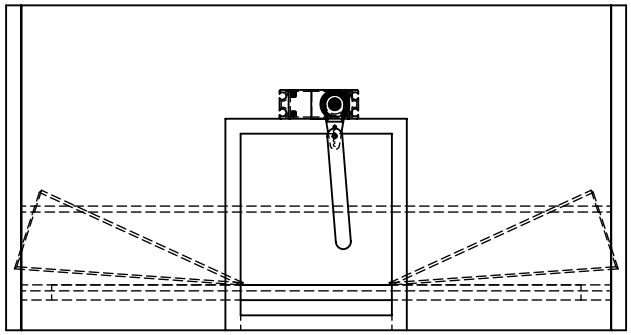
TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

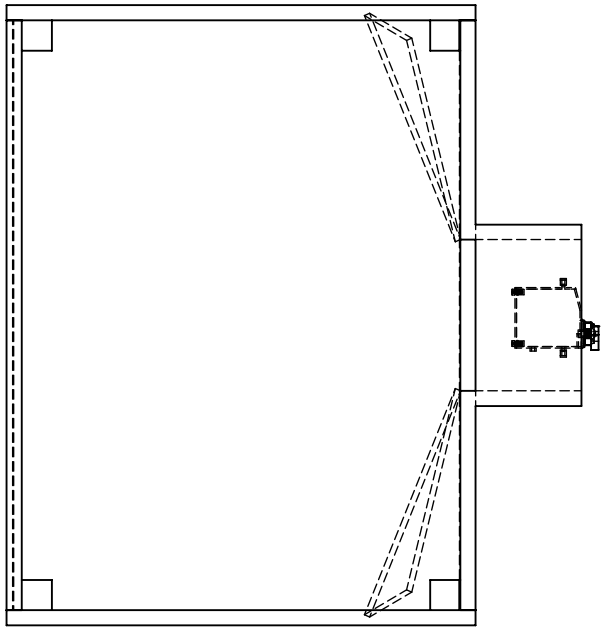
SISTEMA DE LANZAMIENTO - 2

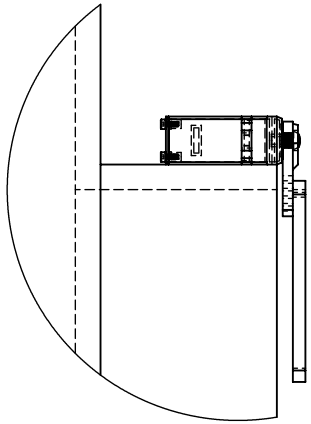
ESCALA 1:3	PLANO Nº
UNIDADES EN CM	2.6

SOLIDWORKS









DETALLE R
ESCALA 1 : 3

SOLIDWORKS

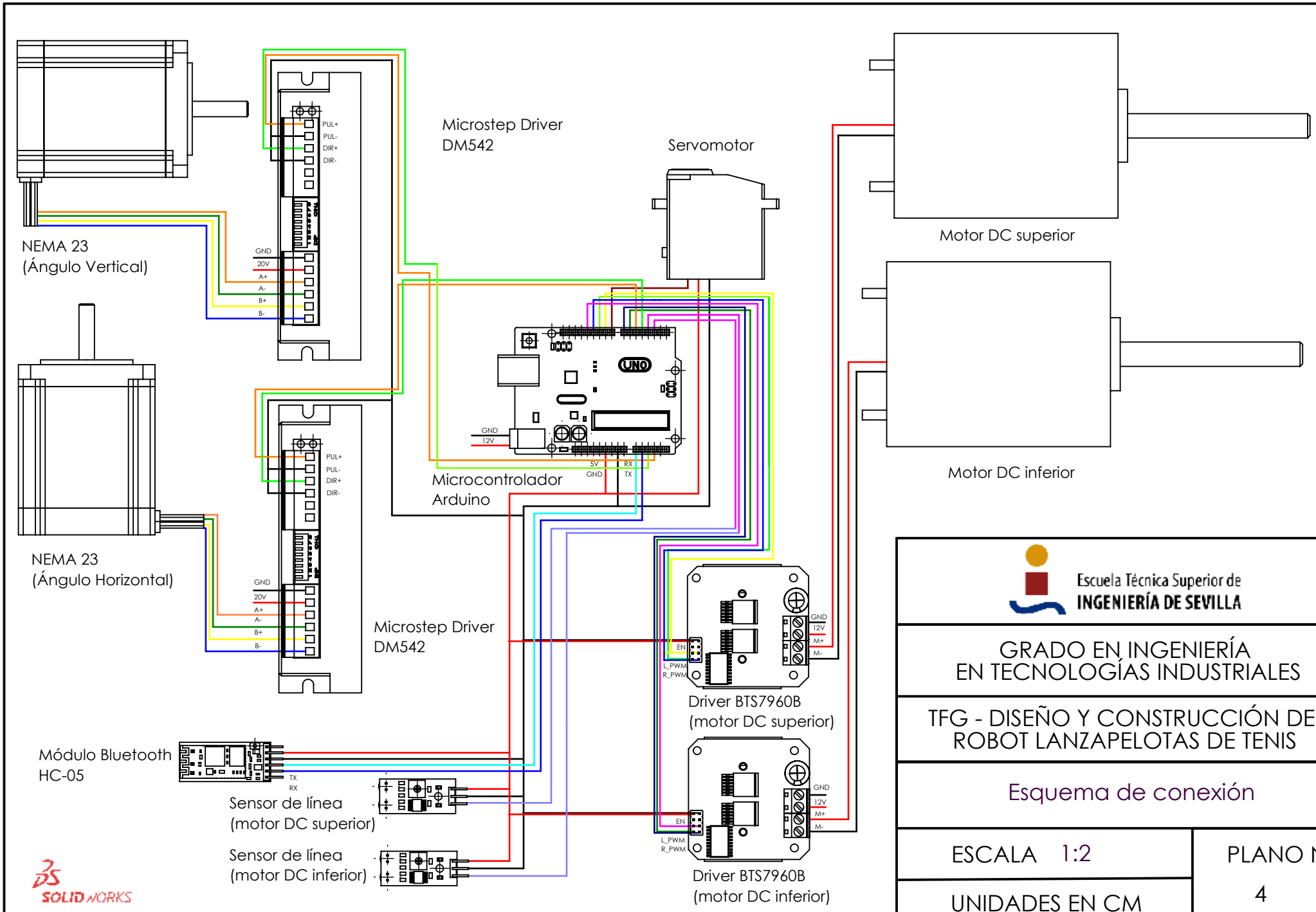
Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

SISTEMA DE ALIMENTACIÓN

ESCALA 1:05	PLANO Nº 3
UNIDADES EN CM	



Escuela Técnica Superior de
INGENIERÍA DE SEVILLA

GRADO EN INGENIERÍA
EN TECNOLOGÍAS INDUSTRIALES

TFG - DISEÑO Y CONSTRUCCIÓN DE
ROBOT LANZAPELOTAS DE TENIS

Esquema de conexión

ESCALA 1:2

UNIDADES EN CM

PLANO Nº
4