universidad de león

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA MECÁNICA

Trabajo de Fin de Grado

Diseño y construcción de una cola mecánica para robots cruadrúpedos

Design and construction of a mechanical tail for quadrupedal robots

Autor: Zilong Huang
Tutor: Juan Felipe Garcia Sierra

(Julio, 2023)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA MECÁNICA
Trabajo de Fin de Grado

ALUMNO: Zilong Huang

TUTOR: Juan Felipe Garcia Sierra

TÍTULO: Diseño y construcción de una cola mecánica para robots cruadrúpedos

TITLE: Design and construction of a mechanical tail for quadrupedal robots

CONVOCATORIA: Julio, 2023

RESUMEN:

Este proyecto surge como un intento en el campo de la biomimética de robots cuadrúpedos, con el objetivo de lograr una interacción emocional con perros mascotas mediante la instalación de una cola mecánica en el robot Unitree A1.El proyecto es un sistema integrado que consta de dos módulos principales: diseño mecánico y control.

El módulo de diseño mecánico consiste en la cola física y su mecanismo de transmisión.

El cuerpo principal de la cola está compuesto por múltiples articulaciones universales conectadas en serie, impulsadas por dos pares de cables de acero: un par se conecta a una polea colocada horizontalmente, lo que permite que la cola oscile lateralmente; mientras que el otro par se conecta a una polea colocada verticalmente, permitiendo que la cola oscile verticalmente. Ambas poleas están conectadas a sus respectivos motores.

El módulo de control se encarga de controlar la velocidad, dirección y número de ciclos de los dos motores, lo que permite que la cola oscile de diversas formas. Este módulo está compuesto por una caja de suministro de energía, una placa de desarrollo Arduino Nano con la placa de expansión CNC V4 y el entorno de desarrollo integrado Arduino IDE. El Arduino IDE se instala en una computadora y se utiliza para escribir el código de control, el cual se compila en un archivo ejecutable binario. Posteriormente, este archivo se envía al Arduino Nano a través de una interfaz USB y un cable. El microprocesador en el Arduino Nano ejecuta el código binario para controlar el funcionamiento de los motores.

Este trabajo ha recibido el primer premio en el concurso de prototipos de FGULEM-Universidad de León 2022-2023.

ABSTRACT:

This project arises as an attempt in the field of biomimetics of quadruped robots, aiming to achieve emotional interaction with pet dogs by installing a mechanical tail on the Unitree A1 robot. The project is an integrated system consisting of two main modules: mechanical design and control.

The mechanical design module consists of the physical tail and its transmission mechanism. The main body of the tail is composed of multiple universal joints connected in series, driven by two pairs of steel cables: one pair is connected to a horizontally placed pulley, allowing the tail to swing laterally; while the other pair is connected to a vertically placed pulley, allowing the tail to swing vertically. Both pulleys are connected to their respective motors.

The control module is responsible for controlling the speed, direction, and number of cycles of the two motors, allowing the tail to swing in various ways. This module consists of a power supply box, an Arduino Nano development board with the CNC V4 expansion board, and the Arduino IDE integrated development environment. The Arduino IDE is installed on a computer and is used to write the control code, which is then compiled into a binary executable file. This file is then sent to the Arduino Nano via a USB interface and cable. The microprocessor on the Arduino Nano executes the binary code to control the operation of the motors.

This work has received first prize in the FGULEM-University of León 2022-2023 prototype competition.

Palabras clave: Universal joints in series , Arduino , Stiffness against oscillation.

Firma del alumno:	**VºBº Tutor/es:**

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

En este trabajo, diseñamos y construimos una cola mecánica bioinspirada basada en el robot cuadrúpedo Unitree A1, con el objetivo de simular las emociones básicas de un perro mediante oscilaciones direccionales y frecuencias específicas.

La estructura principal de la cola mecánica consta de una serie de juntas universales especiales en serie, impulsadas por dos motores y cuatro cables de acero. Los motores son controlados por una placa Arduino, y la estructura en su conjunto presenta una gran flexibilidad y maniobrabilidad. A continuación se muestra en figura 1.1 un boceto del principio de funcionamiento[1].

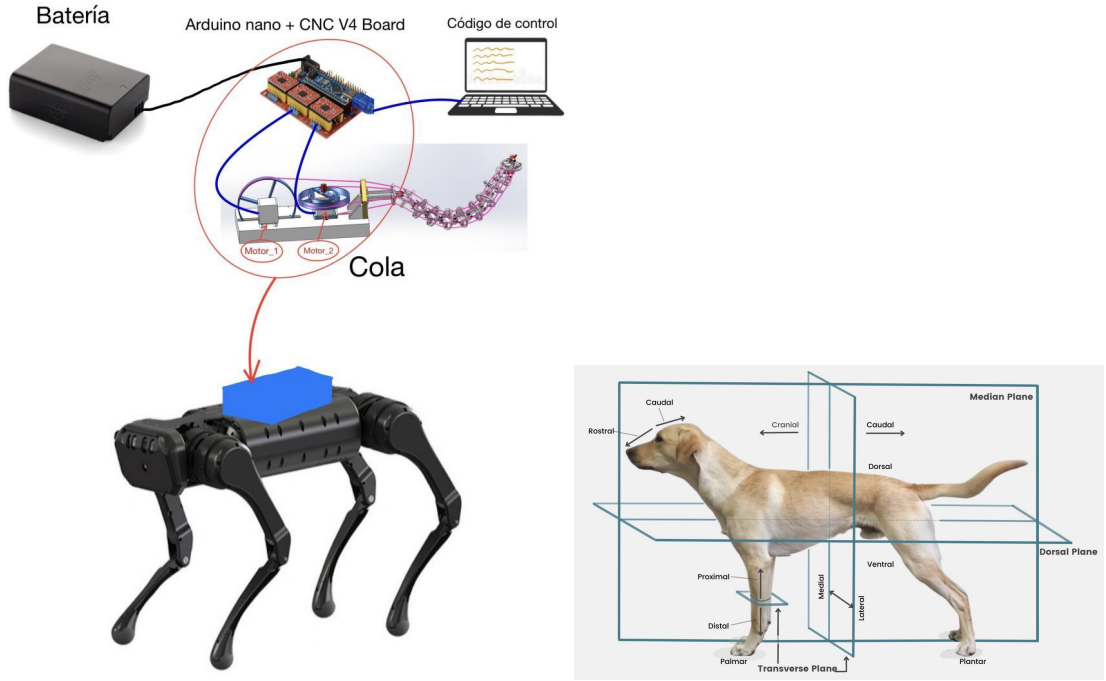


Figura 1.1. Boceto del principio de funcionamiento del trabajo (Funte: elaboración propia)

Este trabajo consiste en dos partes esenciales：
a)Diseñar y construir la cola (la cola física)
b)Hacer la parte informática (placa Arduino + software)
Las dos partes son independientes siempre y cuando nos aseguremos que los motores (servomotores o steppers) que usemos sean compatibles con Arduino.

Alcance del trabajo: En esta trabajo, la parte mecánica se diseñó utilizando un enfoque de diseño ligero y se utilizaron piezas de plástico impresas en 3D de PLA. Se implementó un control de lazo abierto para lograr movimientos aproximados, sin utilizar retroalimentación para ajustar los movimientos mediante control de lazo cerrado. No se incluye el diseño del circuito de control ni la instalación de sensores, y

tampoco se lleva a cabo la simulación virtual.
Además, es importante mencionar que el robot cuadrúpedo utilizado en este trabajo fue desarrollado y fabricado por la empresa china Unitree. Si bien su precisión y estabilidad no son comparables a las del robot Spot de Boston Dynamics en Estados Unidos, el robot Unitree A1 destaca por su relación costo-beneficio y su capacidad para satisfacer las necesidades generales de investigación y aplicaciones prácticas.

1.1 Antecedentes e ideas iniciales del trabajo

La cola de un perro es una herramienta fundamental para expresar emociones como el miedo, la ira, la sumisión, la obediencia y la alegría. Actualmente, podemos utilizar robots cuadrúpedos móviles para simular el comportamiento de un perro, pero hasta ahora estos robots no están equipados con colas bioinspiradas.

Partiendo de este contexto, en este trabajo consideramos el diseño y la construcción de una cola mecánica para un robot cuadrúpedo. Esta cola estará compuesta por al menos dos motores paso a paso o servomotores y será controlada mediante componentes adicionales necesarios, como una placa electrónica Arduino.

La cola estará compuesta por una serie de segmentos articulados (juntas universales). Utilizaremos el software de modelado 3D SolidWorks para diseñar la estructura mecánica y utilizaremos una impresora 3D para imprimir todas las piezas. Al mismo tiempo, construiremos un módulo de control utilizando Arduino para programar los movimientos predefinidos de la cola que imitan las acciones de la cola de un perro en diferentes estados emocionales. El módulo de control constará de una parte de software y una parte de hardware. Para la parte de software, utilizaremos el entorno de desarrollo integrado Arduino IDE y programaremos en un lenguaje similar a C. Para la parte de hardware, utilizaremos una configuración compuesta por Arduino Nano (placa de desarrollo) y CNC-shield-v4 (placa de expansión). El controlador que facilitará la conexión entre el software y el hardware será CH340. Una vez que se complete el ensamblaje físico, realizaremos pruebas en el campo para identificar y solucionar cualquier problema relacionado con la estructura mecánica durante el funcionamiento, y optimizaremos los parámetros de funcionamiento de los motores en diferentes modos de movimiento. Una vez que hayamos logrado la estructura mecánica y el programa de control óptimos para la cola, se considerará que el trabajo está terminado.

El objetivo final de este trabajo es que el modelo de cola se ejecute en el robot cuadrúpedo Unitree A1 y se pueda interactuar con un perro real para validar si podemos transmitir las emociones básicas de los perros a través del movimiento de la cola.

Por lo tanto, los objetivos específicos de este trabajo son los siguientes:

a)Diseñar y construir una cola para el robot cuadrúpedo con dos grados de libertad.

b)Programar previamente un conjunto de movimientos básicos que imiten el movimiento real de la cola de un perro.

c) Permitir que un operador humano controle remotamente el movimiento de la cola.

d)Instalar y probar la cola en el robot Unitree A1 y llevar a cabo interacciones con un perro real.

1.2 Trabajo relacionado

De hecho, el nombre de este trabajo es "Mechanical Tail" (Cola Mecánica), pero en esencia pertenece a la categoría de los brazos robóticos flexibles.

La mayoría de los brazos robóticos flexibles existentes están compuestos por múltiples juntas (articulaciones universales) en serie, impulsados por cables flexibles (generalmente cables de acero), y comparten el mismo principio de movimiento que este trabajo. Sin embargo, los brazos robóticos flexibles comerciales tienen un mejor rendimiento en cuanto a grados de libertad, estabilidad, precisión, potencia de salida y capacidad de operación. Por ejemplo, la cola mecánica de este trabajo tiene solo dos grados de libertad (dos pares de cables de acero que permiten que la cola se mueva en el plano horizontal y vertical), mientras que el brazo robótico flexible mostrado en la siguiente figura 1.2 está equipado con múltiples pares de cables de tracción y puede moverse en múltiples direcciones[2]:

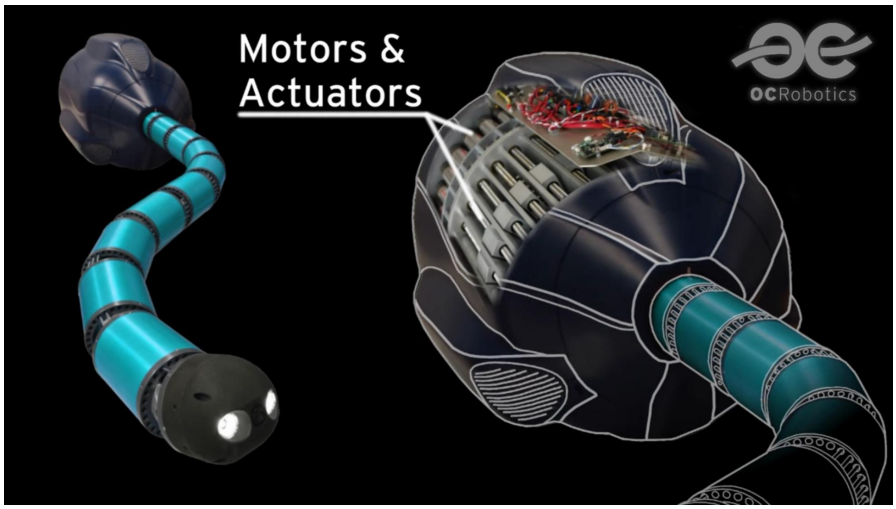


Figura 1.2. OC Robotics - Snake arm 101 (Funte:[2])

Comparado con los brazos robóticos rígidos tradicionales, los brazos robóticos flexibles tienen algunas limitaciones en cuanto a la salida de fuerza, especialmente en lo que respecta a la complejidad de los algoritmos de control. Sin embargo, tienen una mayor flexibilidad y capacidad de deformación, lo que les permite adaptarse a entornos y tareas complejas, así como proporcionar una mejor seguridad y capacidad de colaboración humano-robot.

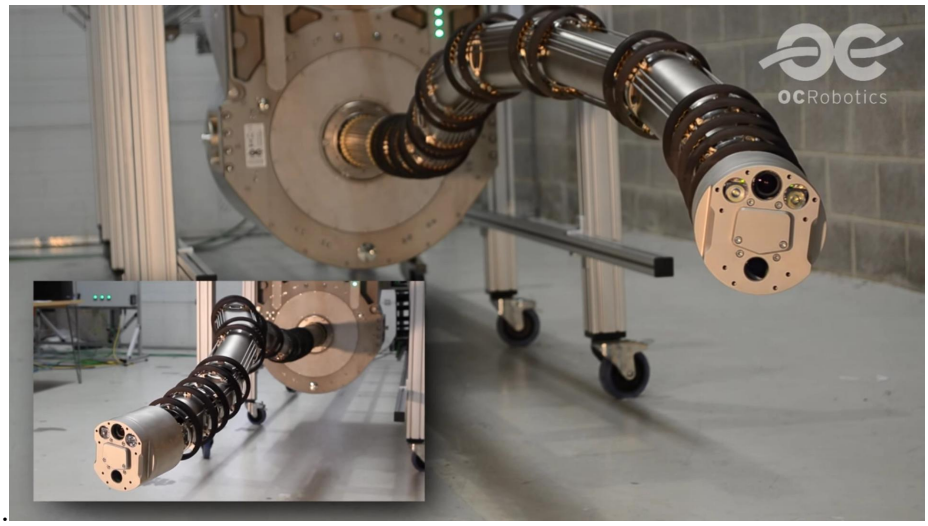
Los brazos robóticos flexibles se utilizan principalmente para reemplazar los brazos robóticos rígidos en espacios estrechos (como tuberías o escombros de terremotos). A continuación, se presentan algunos ejemplos de escenarios de aplicación comunes:

a)Entornos no estructurados: Los brazos robóticos flexibles pueden adaptarse a entornos no estructurados y complejos, como en operaciones de rescate en desastres, exploración marina, espacio exterior, entre otros.

b)Procesamiento flexible: Los brazos robóticos flexibles pueden realizar operaciones de procesamiento y ensamblaje flexibles, lo que los hace adecuados para procesos de producción con formas de piezas complejas y requisitos de flexibilidad.

En el ámbito de los brazos robóticos flexibles, hemos prestado especial atención a la serie de robots en forma de serpiente desarrollados por la empresa Ocrobotics.

Tomemos como ejemplo el robot en forma de serpiente II X125 desarrollado por esta empresa. Este brazo robótico tiene un alto grado de libertad y precisión de movimiento, especialmente una fuerte capacidad para sortear obstáculos. Este tipo de robot se utiliza para inspeccionar y medir entornos tridimensionales cerrados y desordenados[3], como mostrado a continuación en figura 1.3:



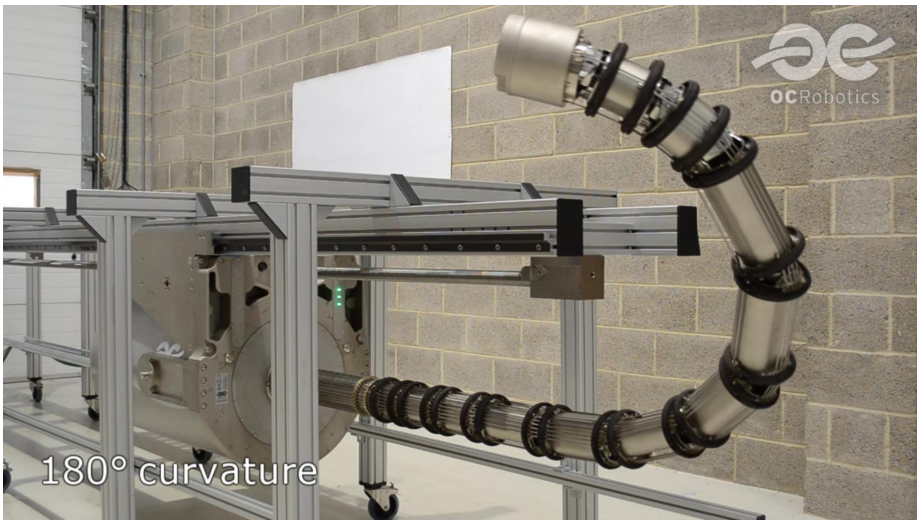


Figura 1.3. II X125 snake-arm robot (Funte: [3])

En conclusión, con el continuo avance de los materiales flexibles y la tecnología robótica, los brazos robóticos flexibles tienen un amplio campo de aplicación en áreas como la automatización industrial y la interacción humano-robot.

2. Diseño de la máquina

El diseño mecánico (machine design) consiste en concebir, analizar y calcular los principios de funcionamiento, estructuras, modos de movimiento, formas de transmisión de fuerza y energía, materiales y dimensiones de las diferentes piezas, así como los métodos de lubricación de una máquina, de acuerdo con las necesidades. Este proceso se lleva a cabo para convertirlos en modelos descriptivos concretos que sirvan como base para la fabricación. En figura 2.1 tenemos un diagrama de flujo del diseño de las máquinas:

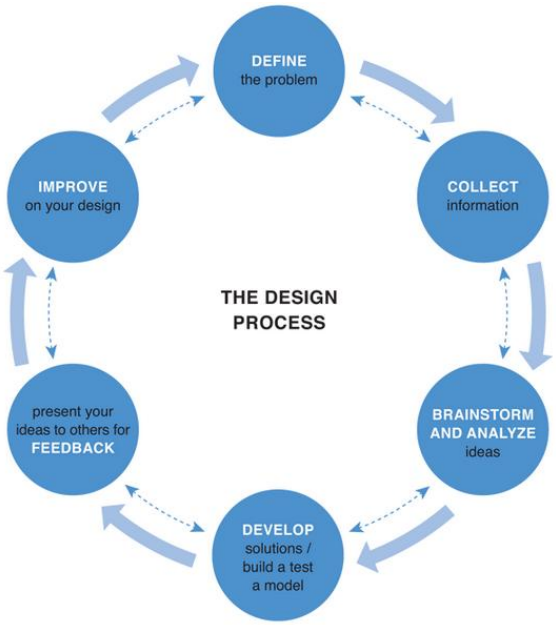


Figura 2.1. Mechine design (Funte: [12])

En este trabajo, el enfoque principal se centra en el diseño y optimización de la estructura mecánica, y todas las piezas son componentes no estándar diseñados por nosotros mismos.

El diseño mecánico se divide en tres categorías: diseño de nuevos modelos, diseño heredado y diseño de variantes.

a) El diseño de nuevos modelos implica la aplicación de tecnologías científicas maduras o tecnologías nuevas que han sido demostradas como viables mediante experimentos. Se trata de diseñar máquinas completamente nuevas que no han existido anteriormente.
b) El diseño heredado implica actualizar el diseño de máquinas existentes en función de la experiencia de uso y el desarrollo tecnológico, con el objetivo de mejorar su rendimiento, reducir los costos de fabricación o disminuir los gastos de uso.
c) El diseño de variantes implica realizar modificaciones o adiciones a máquinas

existentes para adaptarlas a nuevas necesidades, desarrollando productos variantes que difieren de los modelos estándar.

En este trabajo, se trata de un diseño de variantes. Con el objetivo de llenar ciertos vacíos en el campo de la biomecánica de los robots cuadrúpedos, se interactuará con un perro. Sin embargo, la idea básica de utilizar una articulación universal en serie y una transmisión por cables de acero no es original.

El objetivo principal del diseño mecánico es diseñar la mejor máquina posible bajo diversas restricciones, como materiales, capacidad de procesamiento, conocimientos teóricos y herramientas de cálculo. Esto implica realizar un diseño optimizado.

En el pasado, la optimización del diseño dependía principalmente del conocimiento, la experiencia y la visión del diseñador. Sin embargo, con el desarrollo de disciplinas como la teoría básica de la ingeniería mecánica, la ingeniería del valor y el análisis de sistemas, así como la acumulación de datos técnicos y económicos relacionados con la fabricación y el uso, y la amplia aplicación de la informática, la optimización se ha alejado gradualmente del juicio subjetivo y se ha basado en el cálculo científico.

En el caso de este trabajo, no se requiere una precisión y rendimiento del sistema muy alta, y los costos de fabricación y verificación son relativamente bajos. Por lo tanto, el diseño y optimización de la parte mecánica del trabajo se basarán principalmente en la intuición personal y la validación mediante prototipos, sin involucrar simulaciones virtuales ni cálculos complejos.

2.1 Tamaño

Para el diseño físico, es importante realizar una estimación de las dimensiones del objeto. Para ello, se deben tener en cuenta tanto la capacidad de carga máxima del robot A1 utilizado como base (5-7 kg), como las dimensiones del mismo (30 cm de ancho, 62 cm de largo, 6 kg de peso).

Por lo tanto, se estima que la longitud aproximada de la cola sea de alrededor de 30 cm, y el soporte de la base tenga dimensiones aproximadas de 25 cm x 20 cm x 10 cm. En figura 2.2 se muestra un esquema indicador:

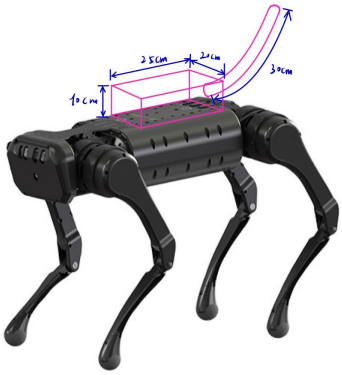


Figura 2.2. Tamaño estimado (Funte: elaboración propia)

2.2 Diseño del mecanismo de movimiento

En figura 2.3 se muestra el esqueleto del perro

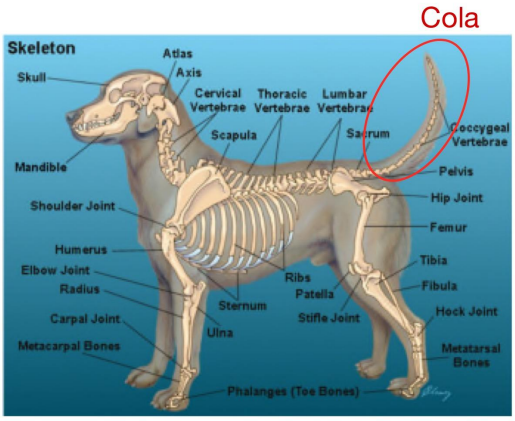
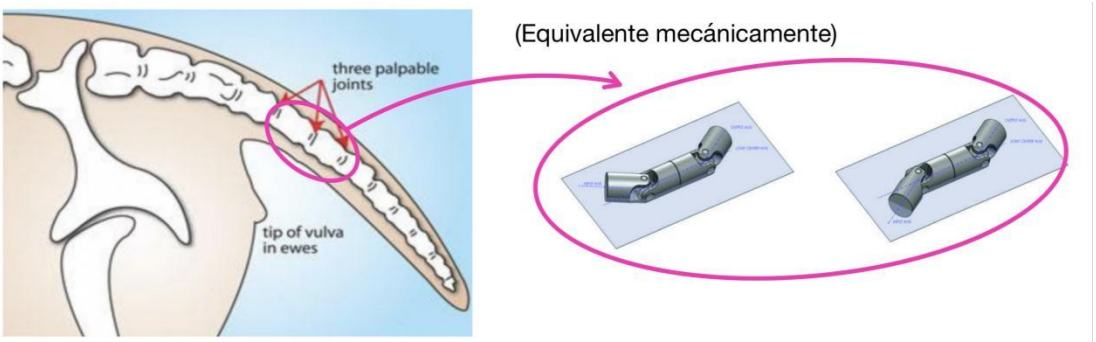


Figura 2.3. el esqueleto del perro (Funte:[4])

De hecho, la cola de los animales se puede simplificar equivalente a un mecanismo en serie de juntas cardánicas con múltiples grados de libertad y con tracción flexible entre las articulaciones (fibras musculares sirve como cables de tracción elásticos con deformación axial significativa), mientras que en el campo de la ingeniería generalmente se utiliza tracción rígida (alambres o cables de acero que pueden doblarse radialmente, pero la deformación axial se puede ignorar), como se muestra en figura 2.4:



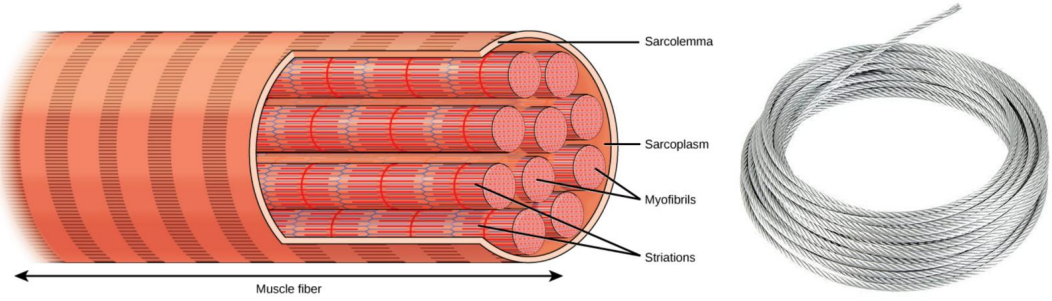


Figura 2.4. Analisis del mecanismo de movimiento de la cola (Funte: elaboración propia)

-----Tracción rígida entre las juntas universales de este trabajo, mediante cable metálico.
En resumen, basándonos en nuestra comprensión de la estructura de las articulaciones de la cola de los animales y las fibras de accionamiento, en este trabajo se utilizará un mecanismo en serie de juntas cardánicas impulsado por cuatro cables de acero. La estructura general se representa aproximadamente en figura 2.5:

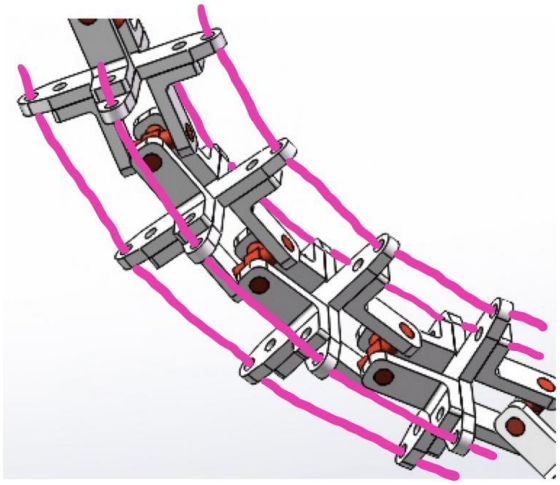


Figura 2.5. un mecanismo en serie de juntas cardánicas impulsado por cuatro cables de acero (Funte: elaboración propia)

2.3 Diseño y modelización de estructuras mecánicas

El cuerpo principal de la cola mecánica está compuesto por una serie de juntas universales especialmente diseñadas, impulsadas por dos motores y cuatro cables de acero. Dos cables de acero en el plano horizontal están conectados a un disco giratorio horizontal, mientras que los otros dos cables de acero en el plano vertical están conectados a un disco giratorio vertical. Cada motor proporciona torque a su

respectivo disco, lo que impulsa los cables de acero.

2.3.1 Herramientas y procedimiento de diseño

Este trabajo utiliza SolidWorks para realizar el modelado en 3D, siguiendo el proceso de diseño de "conceptualización__esbozo__modelado__optimización dimensional".

En figura 2.6 se presenta un ejemplo sobre cómo desarrorar las ideas de diseno:

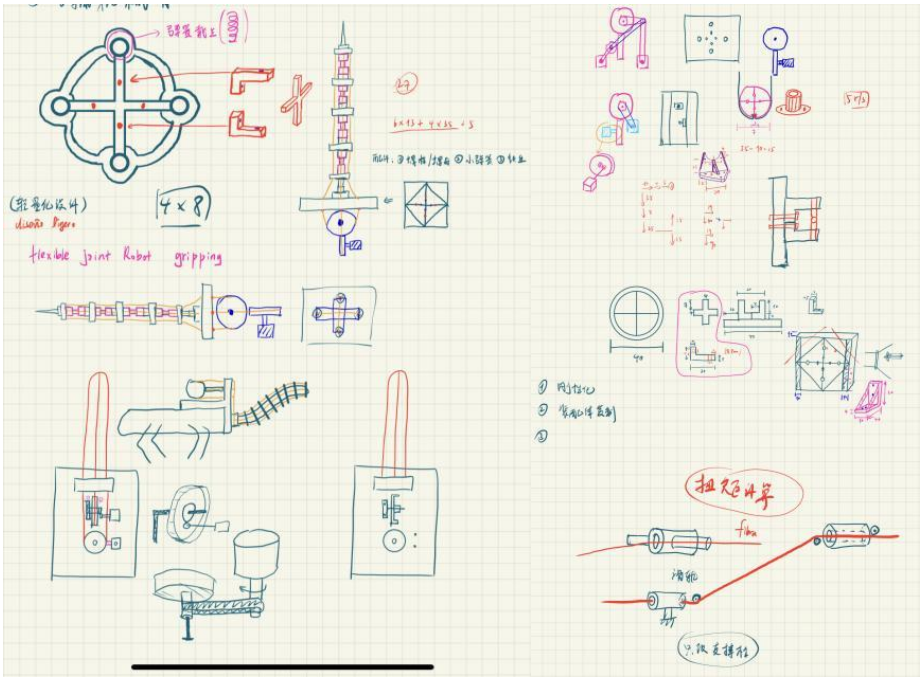


Figura 2.6. ejemjos de esbozos (Funte: elaboración propia)

Después de desarrorar las ideas se pueden convertir en modelo 3D mediante modelado , a continuación en figura 2.7 se muestra unos ejemplos:

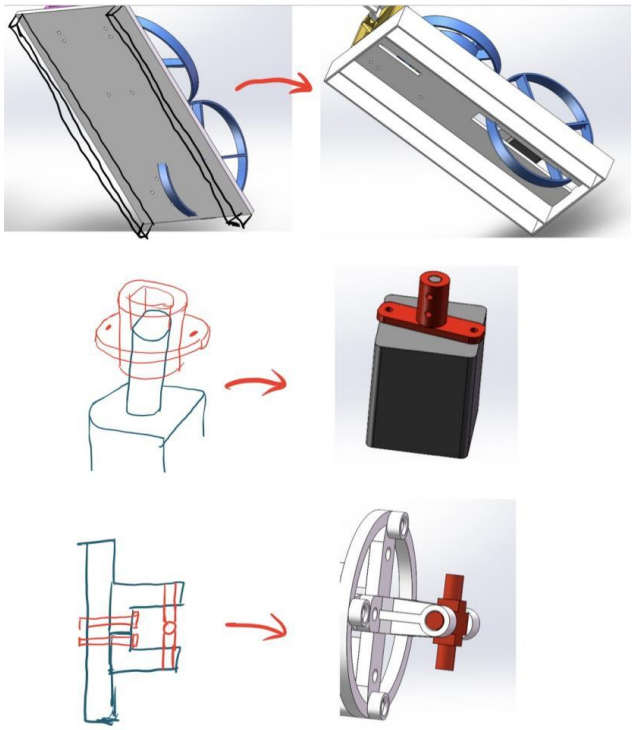


Figura 2.7. Modelado desde esbozos (Funte: elaboración propia)

Enfoque holístico:
Al ensamblar las piezas en el software de modelado, se pueden analizar y optimizar claramente las distancias, dimensiones, formas y relaciones geométricas de la estructura. En figura 2.8 hay unos ejemplos en precticar este idea:

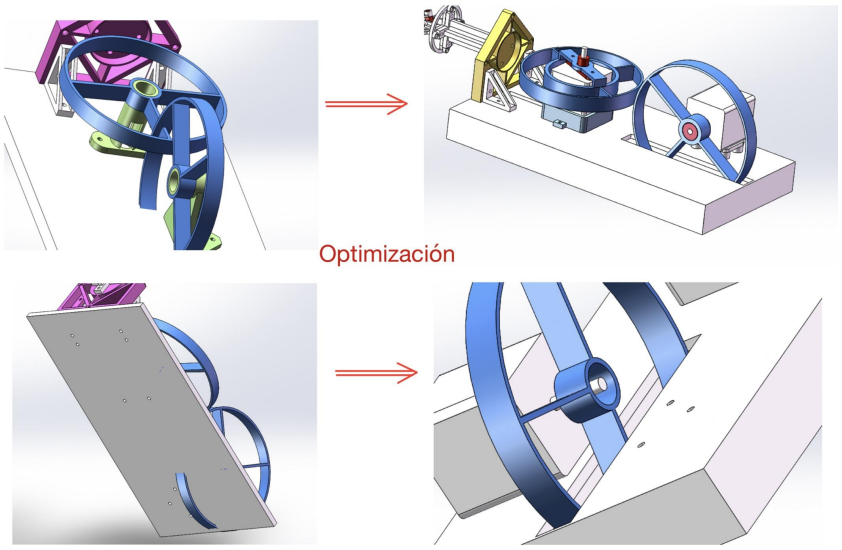


Figura 2.8. Enfoque holístico (Funte: elaboración propia)

Basándose en el enfoque holístico descrito, el diseño de la estructura puede iterarse rápidamente. La figura 2.9 enseña las tres iteraciones principales de la parte mecánica del trabajo.

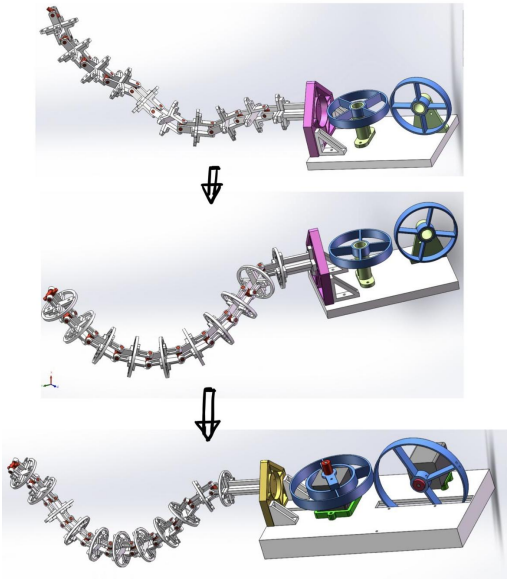


Figura 2.9. las tres iteraciones principales (Funte: elaboración propia)

2.3.2 Desmontaje de estructuras mecánicas y modelado de piezas

Después de un serie de diseño y mejoras nos sale el modelo entero del mecanismo mecánico del trabajo , que se presenta a continuación en figura 2.10:

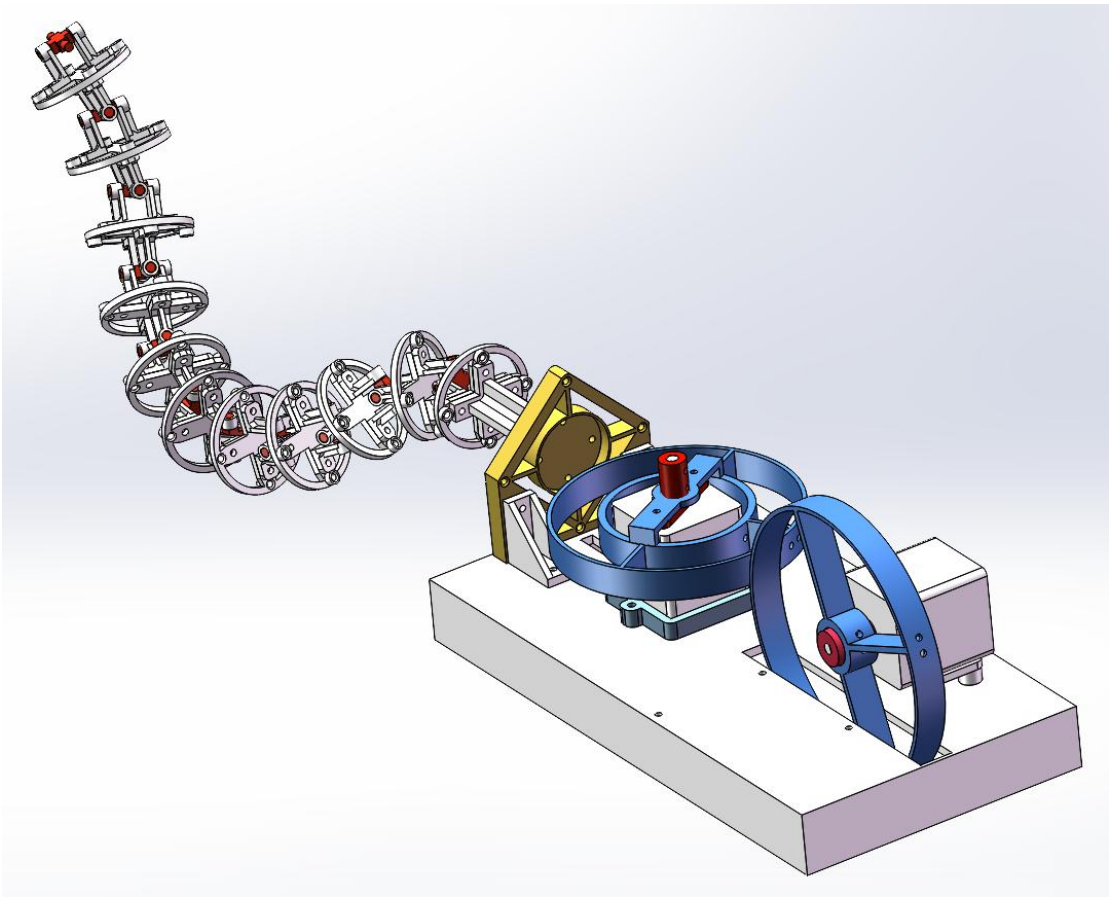


Figura 2.10. Estructura mecanica global (Funte: elaboración propia)

Cadena de juntas universales:
Esta cadena de juntas universales en serie es la estructura principal de la cola, que se mustra en figura 2.11:

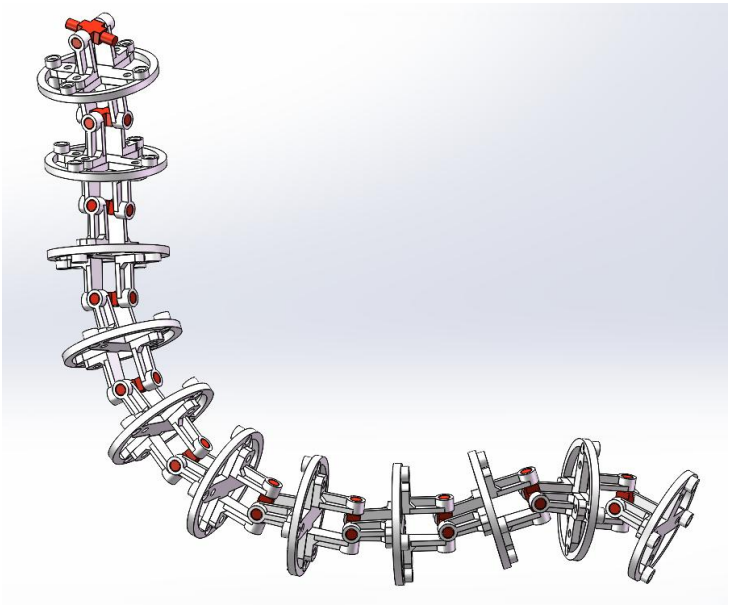


Figura 2.11. la estructura principal de la cola (Funte: elaboración propia)

La unidad básica ("articulaciones") se mustra en figura 2.12:

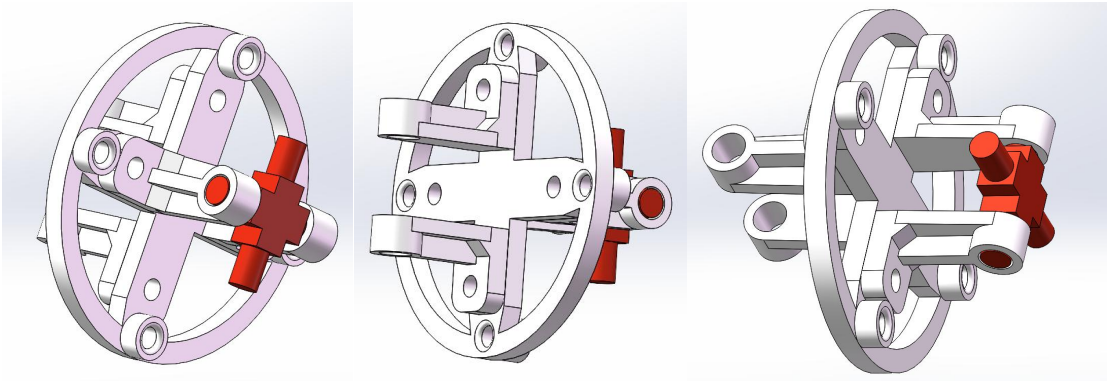


Figura 2.12. Articulación (Funte: elaboración propia)

Toda la cadena consta de las tres tipos de piezas siguientes mostrados en figura 2.13:

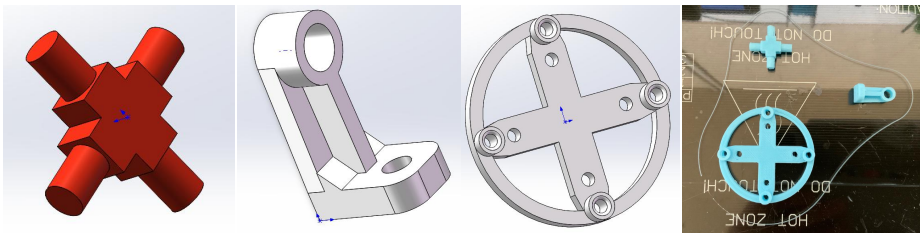


Figura 2.13. Componentes de la articulación (Funte: elaboración propia)

Mecanismo de accionamiento (acoplamiento polea-motor):

El mecanismo de accionamiento de la cola consta de dos poleas con ejes de rotación ortogonales. La estructura de sí se enseña en figura 2.14

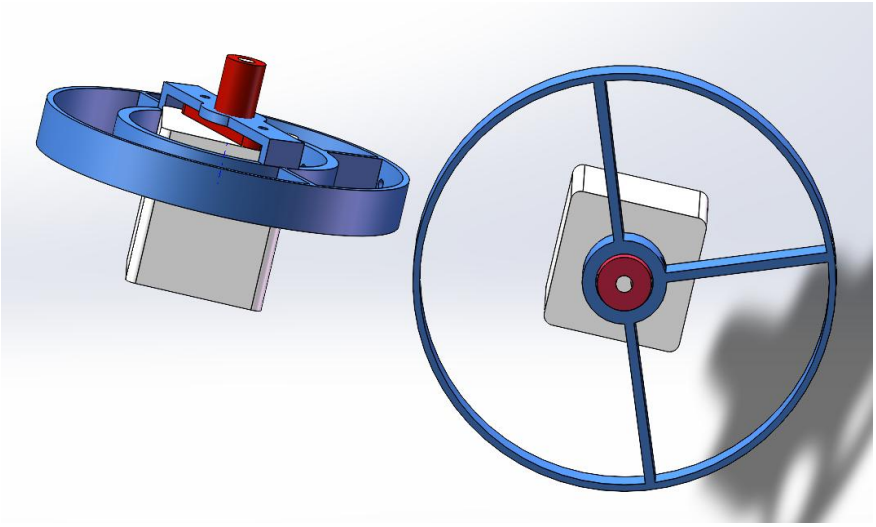


Figura 2.14. mecanismo de accionamiento (Funte: elaboración propia)

Sus componentes se presentan en figura 2.15:

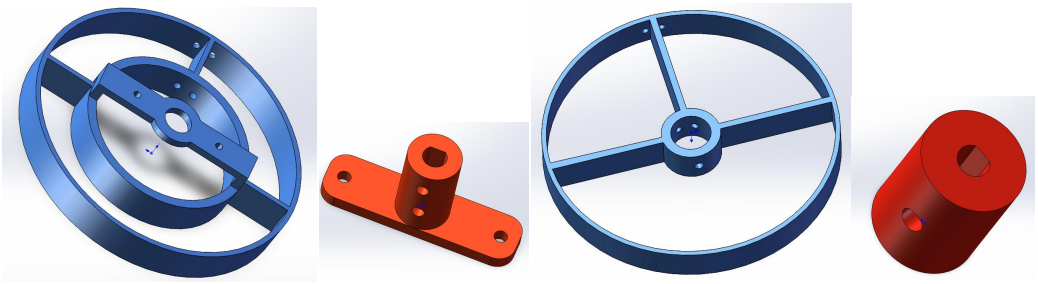


Figura 2.15. componentes (Funte: elaboración propia)

Soporte:
La parte principal de la cola y el mecanismo de accionamiento se apoyan en la estructura de soporte , que de muestra en figura 2.16:

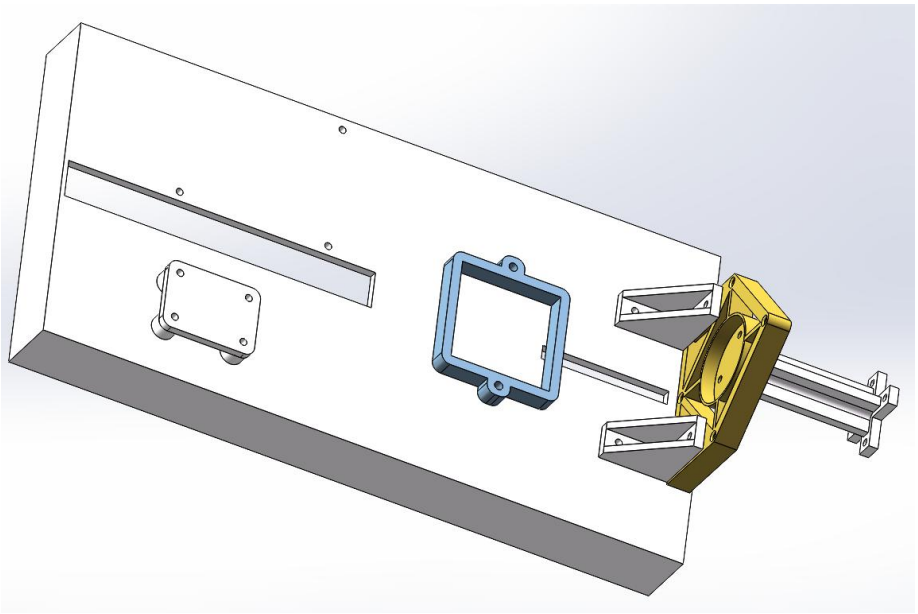


Figura 2.16. Soporte (Funte: elaboración propia)

Sus componentes se enseñan en figura 2.17:

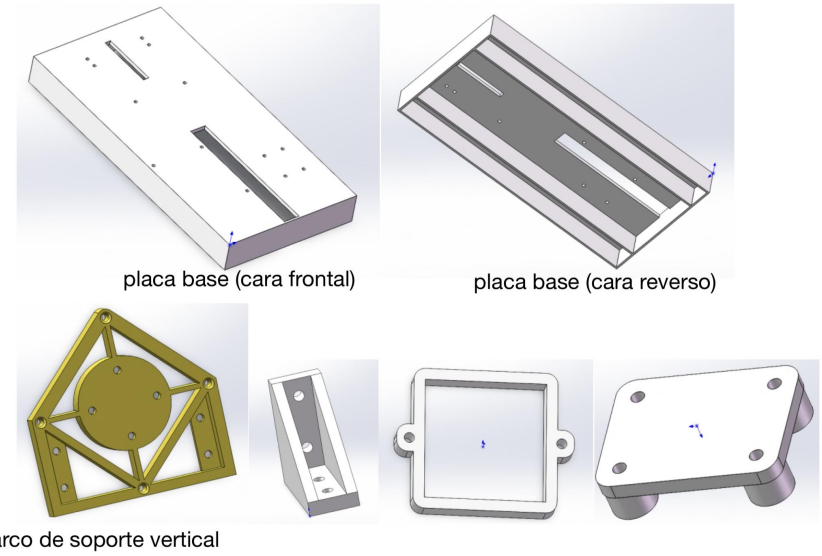


Figura 2.17. Componentes del soporte (Funte: elaboración propia)

2.4 Claves del diseño

En el diseño de la estructura mecánica hay que tener en cuenta muchos detalles, además del mecanismo de movimiento del cuerpo principal. Cuatro de los factores más importantes que influirán en el resultado final del trabajo son: la holgura, la rigidez de la cola a la oscilación , la modularidad y la fricción.

2.4.1 Redundancia de holgura

Para aumentar la flexibilidad de las juntas, debe dejarse cierta holgura entre la cruz de la junta universal y las demás piezas, y los tornillos de fijación de cada junta no deben apretarse del todo. A continuación en Figura complementaria_1 y figura 2.18 tenemos dos ejemplos:

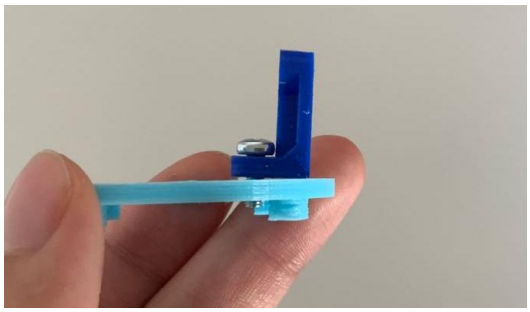


Figura complementaria_1. El tornillo de fijación (Funte: elaboración propia)

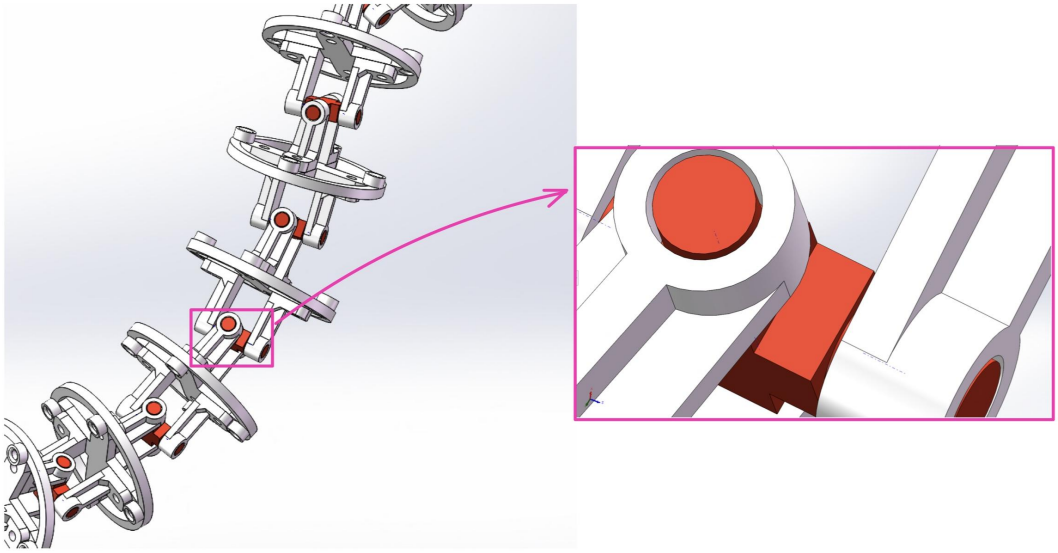


Figura 2.18. Redundancia de holguras (Funte: elaboración propia)

El diseño debe tener en cuenta las tolerancias (tolerancias), especialmente para los ajustes del agujero y el eje, y debe permitir una redundancia de holgura adecuada. La cantidad de redundancia depende del tipo de material, el método de fabricación, etc.

Los siguientes son ejemplos de redundancias de holgura preestablecidas que son excesivas o insuficientes debido a la dilatación y contracción de la pieza y a errores de fabricación

En figura 2.19 se muestra un ejemplo del exceso de redundancia de holguras:

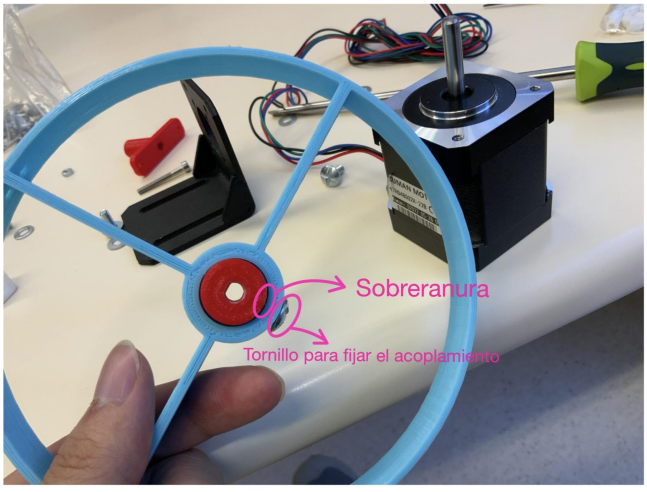


Figura 2.19. Exceso de redundancia de holguras (Funte: elaboración propia)

Insuficiente redundancia de holgura, lo que provoca que el eje no se introduzca en el orificio, que se indica en figura 2.20:

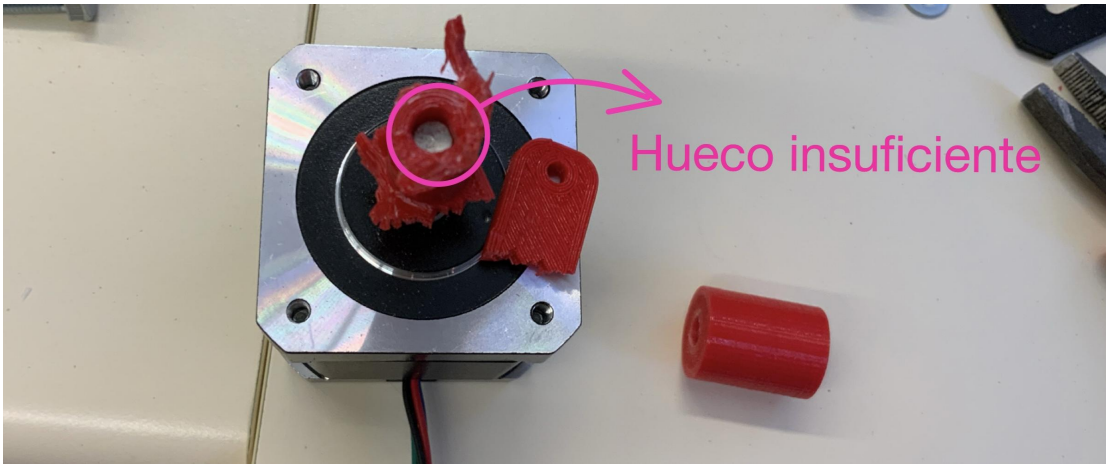


Figura 2.20. Insuficiente redundancia de holgura (Funte: elaboración propia)

El método de encontrar la redundancia óptima a bajo coste, debido a la diversidad de la situación real, el coste aritmético de calcular la redundancia de holgura por métodos teóricos es demasiado alto, para este trabajo fue posible probar el ajuste eje-agujero imprimiendo una pieza auxiliar (parte de la pieza original) para obtener una redundancia de holgura más adecuada mediante una prueba de bajo coste. A continuación en figura 2.21 se enseña un ejemplo de este idea:

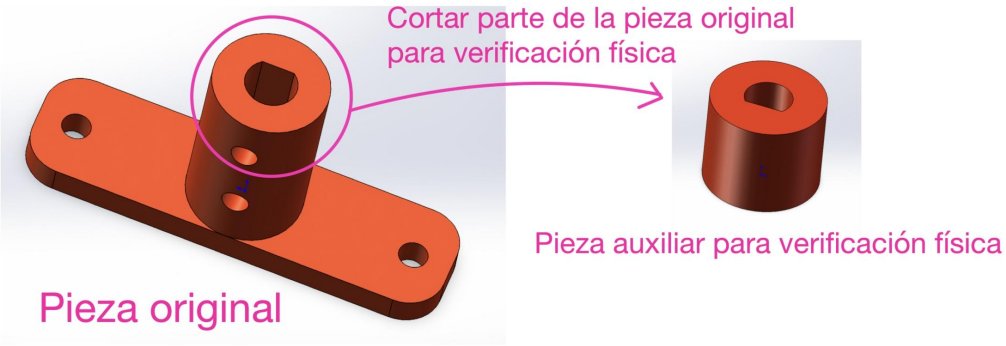


Figura 2.21. Pieza auxiliar (Funte: elaboración propia)

En la práctica, si el eje y el agujero siguen sin ajustarse, se utiliza una lima para rectificarlos , como en figura 2.22:

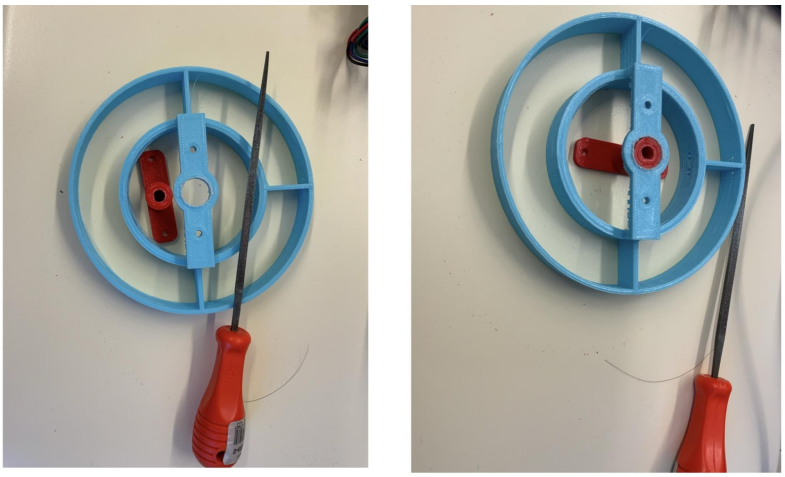


Figura 2.22. Rectificar con lima (Funte: elaboración propia)

Ajuste final del eje y el agujero en figura 2.23:

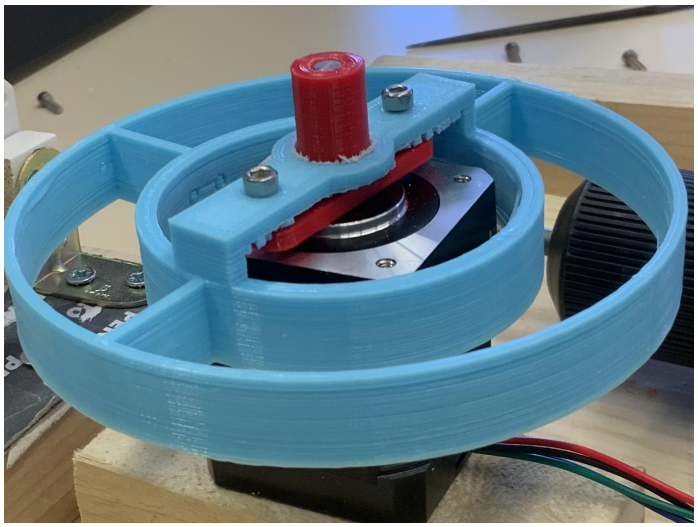


Figura 2.23. Ajuste final del eje y el agujero (Funte: elaboración propia)

2.4.2 Ajuste de la rigidez de la cola contra el oscilación (muelle insertado entre las articulaciones)

Aquí se introduce un concepto clave del trabajo: la rigidez de la cola para resistir la oscilación.

Las articulaciones se gira entre sí con pequeños ángulos, y se combinan en serie para formar el movimiento oscilante de toda la cola. En condiciones ideales, se busca que los ángulos de giro entre las articulaciones adyacentes sean uniformes para lograr un movimiento suave de la cola.

La rigidez a la compresión de los resortes entre las articulaciones se combina en serie para proporcionar la rigidez contra el balanceo de toda la cola. Si no se incluyen los resortes, se producirán diferencias significativas en los ángulos de giro entre las articulaciones adyacentes, lo que puede causar bloqueos entre los cables de acero y las articulaciones, impidiendo un movimiento suave de la cola. El esquema de principio de la regidez de la cola contra la oscilación se elabora a continuación en figura 2.24:

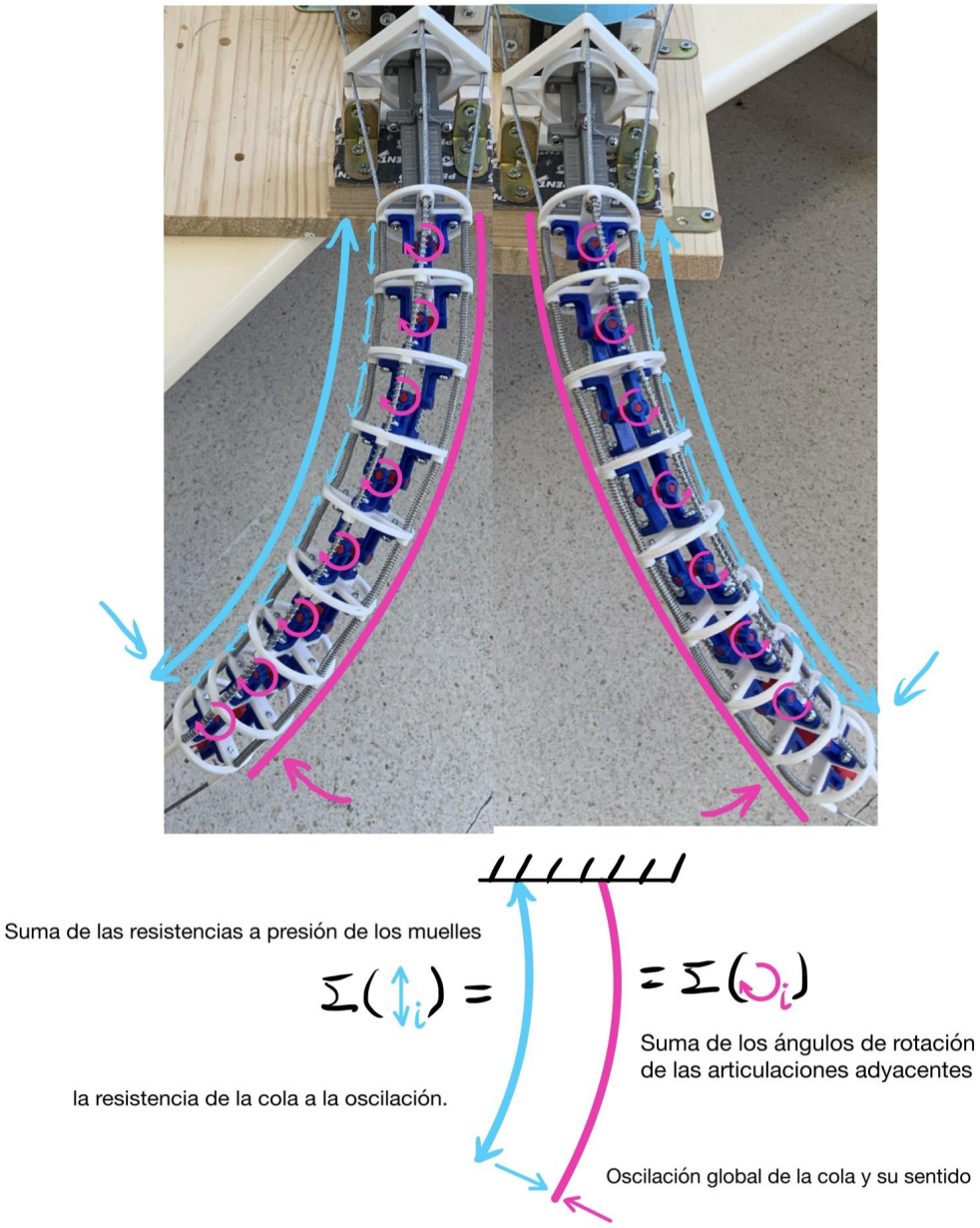


Figura 2.24. Esquema para demonstrar la rigidez de la cola para resistir la oscilación. (Funte: elaboración propia)

Los resortes son componentes clave para lograr un giro uniforme entre las

articulaciones. Hemos realizado pruebas sin incluir los resortes y en esas circunstancias el control de la cola es extremadamente deficiente. Debido a la gran flexibilidad de las articulaciones de la cola, si no se utilizan los resortes, la influencia de la gravedad, la fricción y la inercia provocará una distribución desigual de los ángulos de giro entre las articulaciones, así como bloqueos frecuentes, como se indica en figura 2.25:

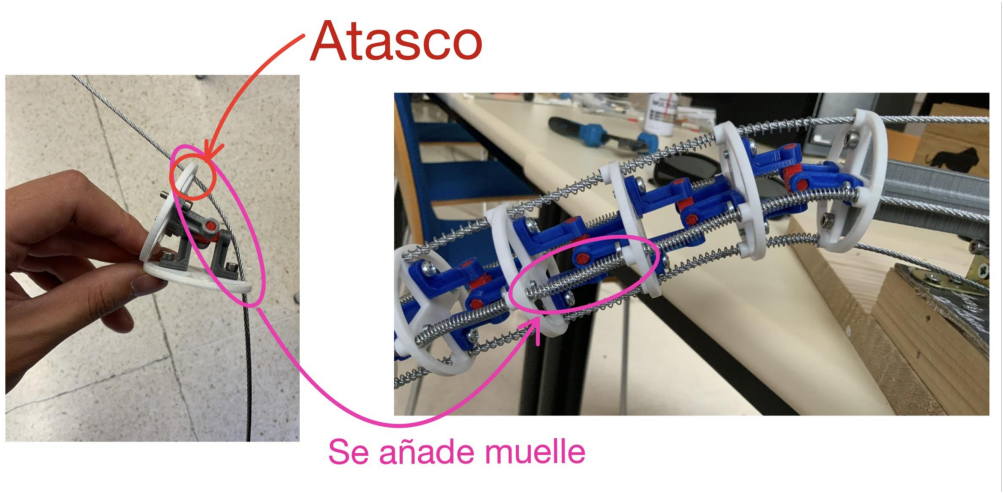


Figura 2.25. El cable metálico se atasca con el borde del agujero (Funte: elaboración propia)

Modificar las longitudes de las muelles insertados para ajustar la rigidez a la oscilación de la cola:
Con el fin de lograr la variación de ángulos entre cada par de articulaciones de acuerdo a la tendencia deseada, se puede ajustar la longitud de los resortes entre las articulaciones. En el estado de suspensión libre, la distancia entre cada par de articulaciones es igual, mientras que los resortes se comprimen alrededor de los cables de acero insertados en los espacios entre las articulaciones. Por lo tanto, cuanto más largo sea el resorte en una posición específica entre las articulaciones, mayor será su rigidez para resistir la rotación en esa dirección. En general, la parte oscilante cercana al extremo distal de la cola experimenta una menor influencia de la gravedad y la inercia, por lo que requiere una menor rigidez a la oscilación, lo que corresponde a resortes más cortos. Por otro lado, la parte oscilante cerca del extremo proximal de la cola experimenta una mayor influencia de la gravedad y la inercia, por lo que requiere una mayor rigidez a la oscilación, lo que corresponde a resortes más largos.

longitud : par de muelles_2 < par de muelles_1 < par de muelles_3

A continuación se muetra en Figura complementaria_2 un ejemplo de mejorar la oscilación de la cola mediante modificar las longitudes de los muelles insertsdos entre diferentes pares de articulaciones.

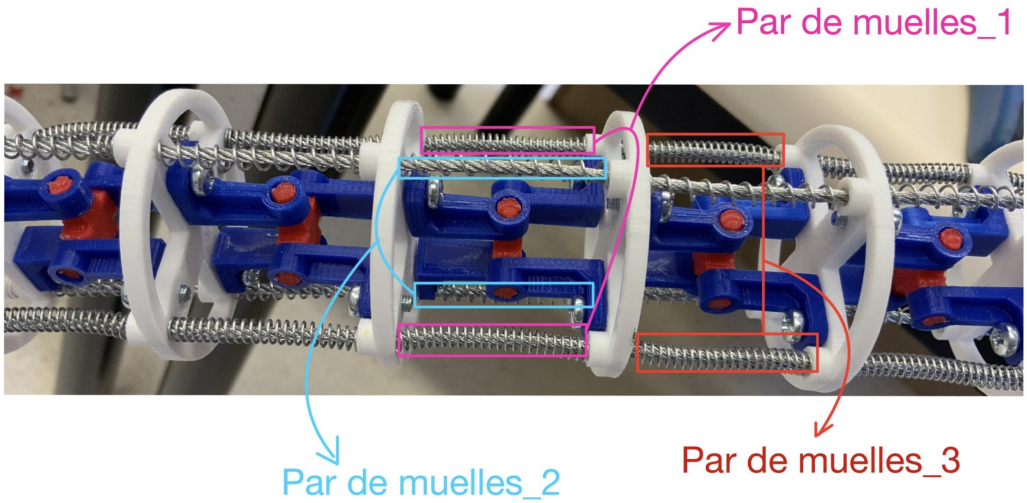


Figura complementaria_2. Modificar las longitudes de los muelles (Funte: elaboración propia)

La longitud del muelle puede modificarse con unas tijeras, como en Figura complementaria_3:

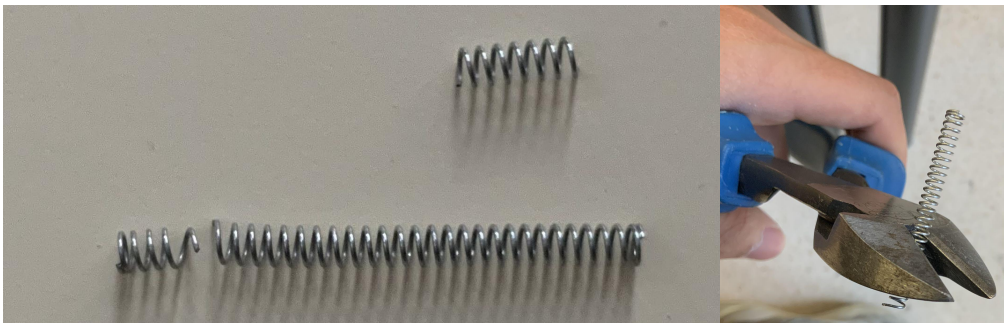


Figura complementaria_3. Modificar manualmente con tijera (Funte: elaboración propia)

2.4.3 Diseño modular

Para un mecanismo de movimiento específico, se debe intentar diseñarlo con múltiples componentes conectados mediante pernos. Esto tiene el propósito de facilitar las mejoras y optimizaciones en etapas posteriores. Si se diseña un componente individual de manera demasiado compleja o tiene dimensiones grandes, puede llevar a un desperdicio de materiales y tiempo debido a la falta de confiabilidad de la impresora 3D durante el proceso de impresión. En tales casos, los esfuerzos invertidos en diseñar el componente complejo o grande pueden ser en vano.

2.4.4 Reducir fricción

Agregar esquinas redondeadas en el borde de los agujeros durante el proceso de diseño, como en figura 2.26.

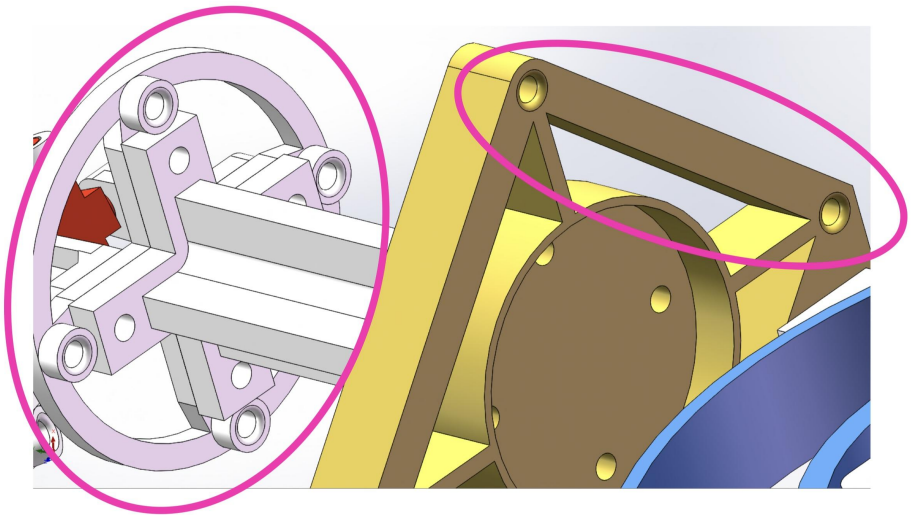


Figura 2.26. Esquinas redondeadas. (Funte: elaboración propia)

El redondeo de esquinas mediante procesamiento manual, como en figura 2.27:

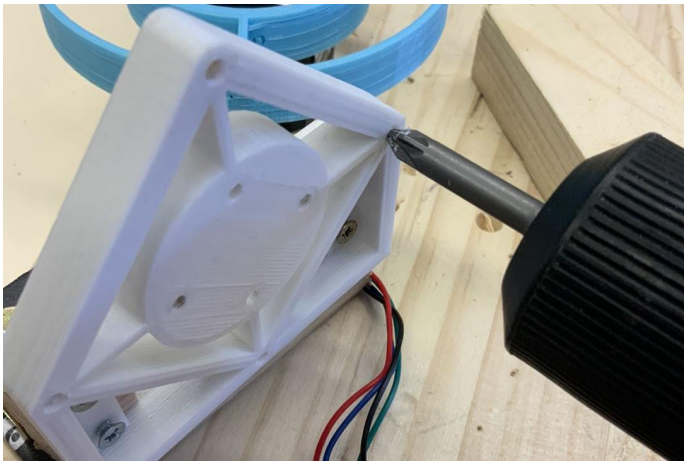
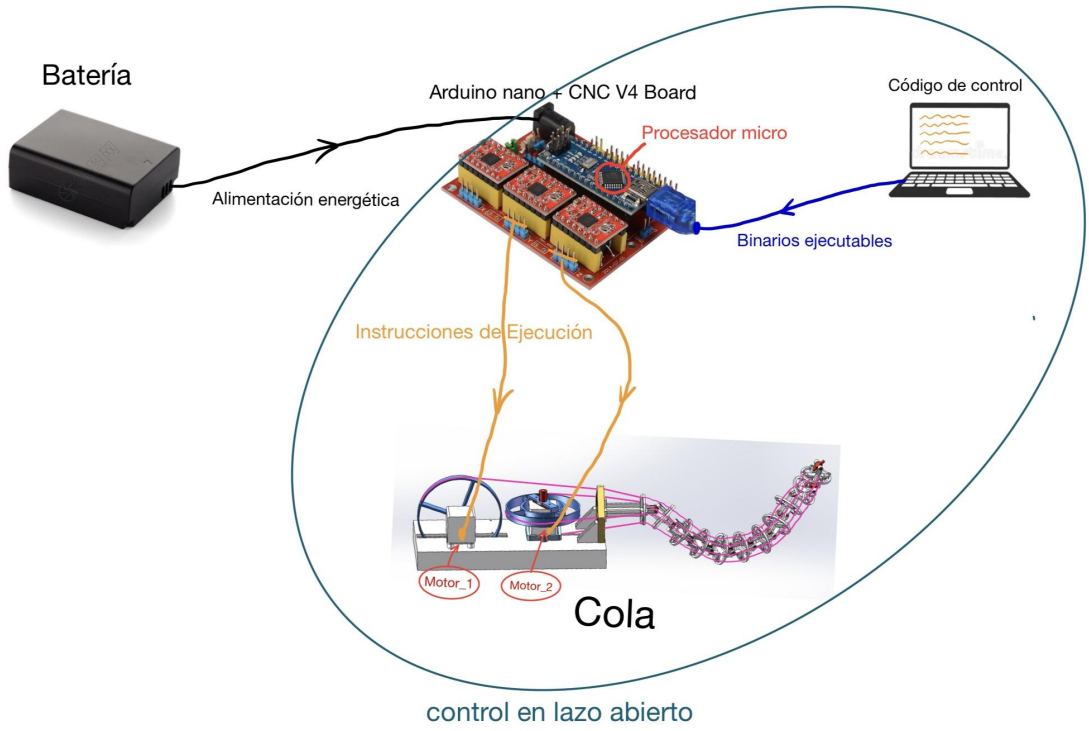


Figura 2.27. Procesamiento manual. (Funte: elaboración propia)

3. Electrónica y Control

Descripción de los principios de control:
Esta sección está compuesta por una caja de alimentación, una placa de desarrollo Arduino Nano junto con una placa de expansión CNC V4 y el entorno de desarrollo integrado Arduino IDE. El Arduino IDE se encuentra en un dispositivo de computadora y se utiliza para escribir el código de control, compilarlo en un archivo ejecutable binario y luego enviar el archivo ejecutable al Arduino Nano a través de una interfaz USB y un cable. El microprocesador en el Arduino Nano ejecuta el código binario y envía instrucciones a través del cable para controlar el funcionamiento de los motores. La placa de expansión CNC V4 actúa como un centro de transmisión, encargada de recibir y enviar información de control.

A continuación en figura 3.1 se muestra un diagrama esquemático:



Figua 3.1. Esquema del principio del bloque de control. (Funte: elaboración propia)

La serie de placas de desarrollo Arduino se basa en un enfoque de diseño fácil de usar y de código abierto. Tienen ventajas como flexibilidad, conveniencia, bajo umbral de entrada, una amplia variedad de interfaces, funcionalidad múltiple y facilidad de expansión. Por estas razones, se utilizan ampliamente en el campo del diseño electrónico. En el mercado actual, hay una amplia variedad de placas de desarrollo Arduino disponibles, como Arduino UNO/UNO R3, Arduino 101/Intel Curie, Arduino Micro, Arduino Ethernet, Intel Galileo, entre otros. Estas placas son fáciles de

aprender y utilizar, tienen una interfaz sencilla y son asequibles en términos de precio[5].

Las ventajas únicas de Arduino son las siguientes:

a)Apertura: Arduino es un trabajo de hardware de código abierto que se inició temprano, y tanto su circuito de hardware como su entorno de desarrollo de software son completamente abiertos.

b)Facilidad de uso: Arduino es simple y fácil de usar, no requiere la instalación de controladores adicionales. Utiliza un lenguaje de programación similar a C, con el que solo se involucran dos módulos principales en la función principal: setup (configuración) y loop (bucle).

c)Facilidad de comunicación: Arduino ha establecido un marco relativamente unificado. Utiliza métodos de inicialización comunes para algunas operaciones de nivel inferior y ha calibrado sus puertos para señales digitales y analógicas. Esto hace que sea muy conveniente para los principiantes comunicarse sobre circuitos o programas.

3.1 Arduino Nano y CNC-shield-V4

En este trabajo, estamos utilizando la placa de desarrollo Arduino Nano y el módulo de expansión CNC Shield V4.

3.1.1 Arduino Nano[6]

Arduino Nano es una placa de desarrollo inteligente basada en el microcontrolador ATmega328P de 8 bits de Microchip. Tiene un rendimiento potente y es similar a Arduino UNO, pero con un tamaño más compacto. Arduino Nano tiene pines de doble fila que se pueden apilar fácilmente en una placa de ensayo y se puede conectar de forma flexible a otros módulos mediante cables Dupont. También se puede conectar a una computadora a través de un puerto Mini-B USB.

Las placas de desarrollo de ARDUINO no requieren un conocimiento profundo de electrónica para desarrollar circuitos de control. Además de controlar los motores paso a paso en este trabajo, también se pueden agregar muchos módulos funcionales, como sensores ultrasónicos, infrarrojos, Bluetooth, inalámbricos, motores de corriente continua, servomotores, entre otros. En términos sencillos, se puede considerar a las placas de desarrollo ARDUINO como un tipo de Lego electrónico.

Arduino Nano	Parametros
controlador	ATmega328P
Flash	32KB
Sram	2KB
EEPROM	1KB
Pines de entrada analógica	8 ud
I/O digital	22 ud
PWM	6 ud
Frecuencia de reloj	16MHz

Tabla 3.1. Arduino Nano tabla de datos (Funte: [6])

A continuación en figura 3.2 mostramos un Arduino Nano y su esquema de pins:

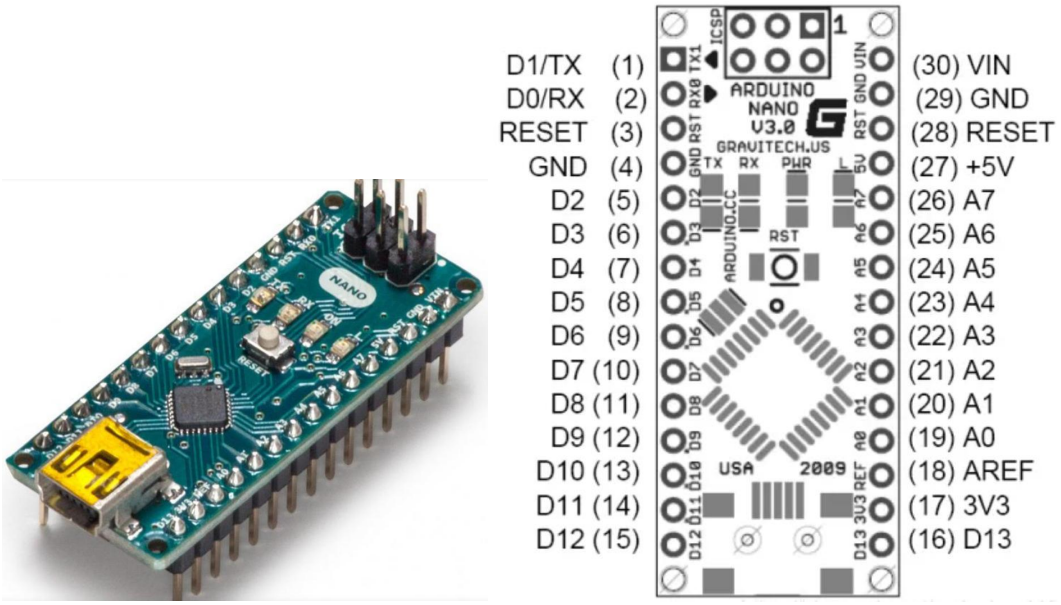


Figura 3.2. Arduino Nano y esquema de sus pins. (Funte: [6])

El ATmega328P tiene un total de 32 pines y tres portes funcionales, PortB, PortC y PortD. El microcontrolador tiene un número limitado de pines, todos los cuales se multiplexan entre sí para lograr una función específica mediante la configuración del software.

3.1.2 CNC V4 Expansion Board[7]

El tipo de placa de expansión tiene el papel de ser el centro de transmisión de datos en el módulo de control. En ella se instala una placa de desarrollo Arduino Nano con un microprocesador, que se utiliza para procesar y ejecutar el código de control. La placa de expansión proporciona interfaces y funciones adicionales que permiten conectar varios sensores, actuadores y otros dispositivos externos. A través de esta placa de expansión, el Arduino Nano puede comunicarse y controlar estos dispositivos externos, lo que permite lograr funciones e interacciones más complejas.

Aquí en figura 3.3 tenemos un foto física de CNC V4:

Figura 3.3. CNC V4 Expansion Board. (Funte: [7])

Conecciones:

Xdir --> D2 Ydir --> D3 Zdir --> D4

Xstep --> D5 Ystep --> D6 Zstep --> D7

En figura 3.4 se muestra Diagrama de la conexión de CNC V4 Expansion Board

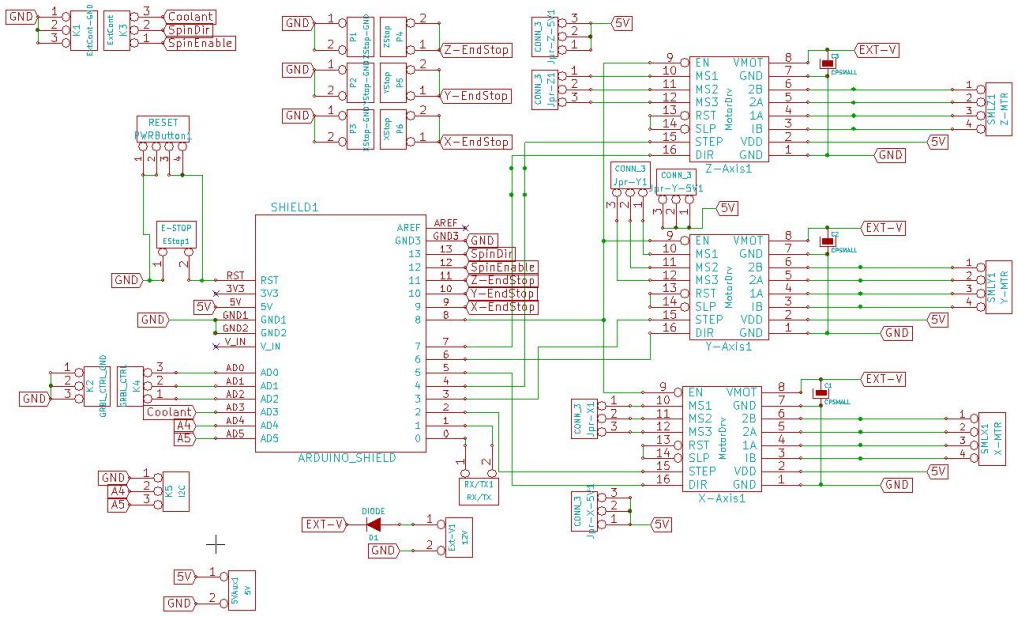


Figura 3.4. Diagrama de la conexión de CNC V4 Expansion Board. (Funte: [7])

3.1.3 Ensamblaje del módulo de control

Mediante la combinación del Arduino Nano y la placa de expansión CNC-shield-V4, los usuarios pueden construir de manera flexible diversos sistemas de control y trabajos de robots, y realizar la transmisión y la interacción de datos. A continuación en figura 3.5 presentamos el ensamblaje del módulo de control:

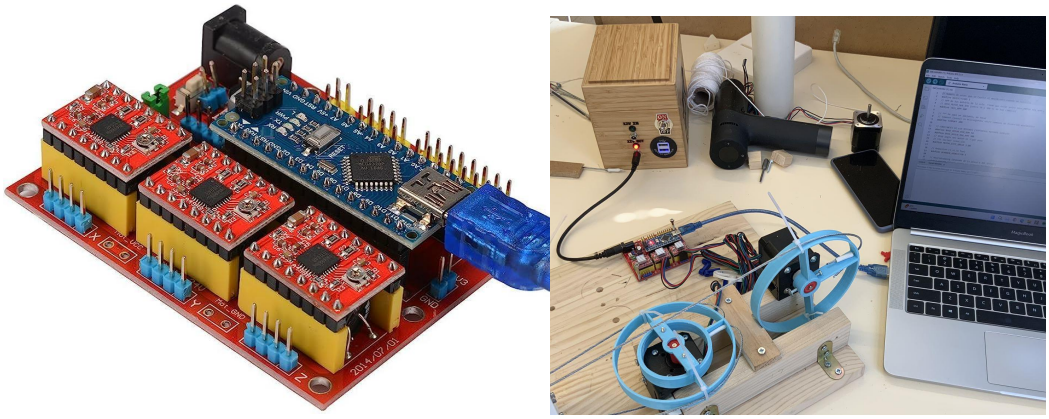


Figura 3.5. Ensamblaje del módulo de control. (Funte: elaboración propia)

3.2 Arduino IDE (entorno de desarrollo integrado)[9]

El Arduino IDE es un entorno de programación estándar lanzado por Arduino, utilizado para escribir y cargar código en placas de circuito. Puede ejecutarse en Windows, Mac OS X y Linux. Este entorno se puede utilizar con cualquier placa de Arduino.

En este trabajo, el programa de control se escribe y ejecuta en Arduino IDE utilizando un lenguaje similar a C. Durante el proceso de escritura del código, se puede realizar pruebas y modificaciones en tiempo real, y el flujo de control del programa es fácil de leer y a continuación en figura complementaria_4 se enseña el interfaz de Arduino IDE:

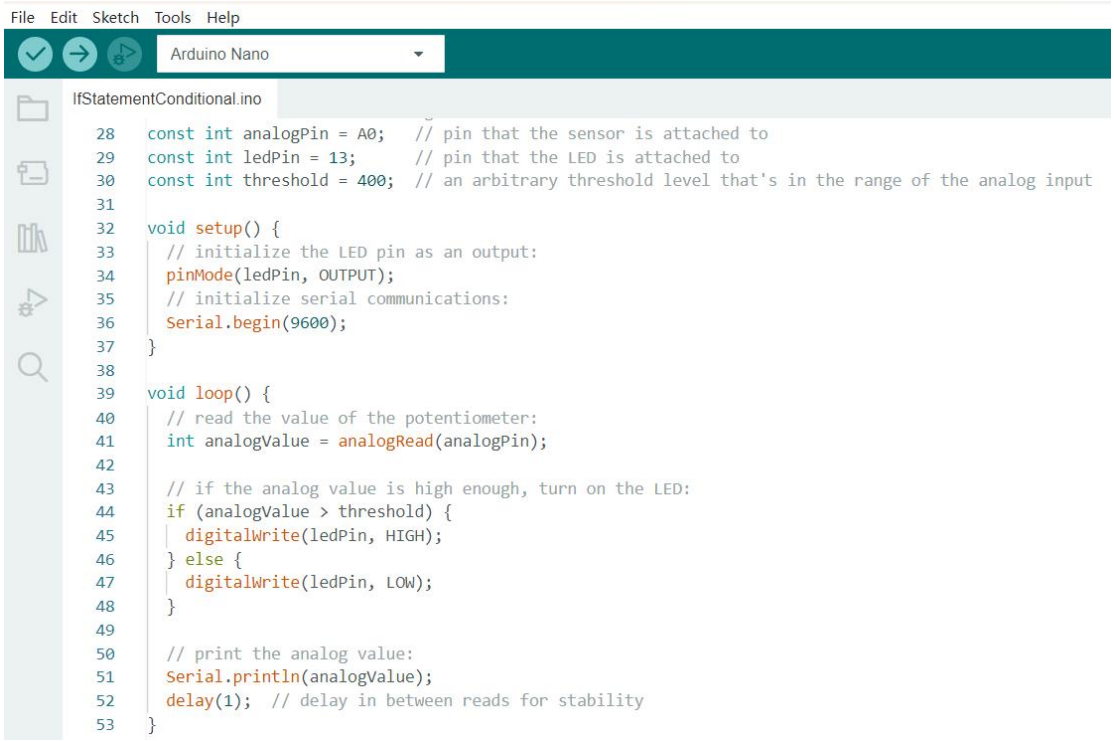


Figura complementaria_4. Interfaz de Arduino IDE. (Funte: [9])

3.3 Los motores y su control

En este trabajo se utiliza un motor paso a paso, que tiene excelentes características de respuesta de arranque, inversión y repetibilidad de movimiento. Es especialmente adecuado para escenarios en los que se requiere un cambio frecuente de dirección de rotación en el trabajo.

3.3.1 Principios básicos de los motores paso a paso

El motor paso a paso es un actuador que convierte pulsos eléctricos en desplazamiento angular. Cuando el controlador del motor paso a paso recibe una señal de pulso, impulsa al motor paso a paso a girar en una dirección determinada en un ángulo fijo (es decir, un paso).
Se controla la cantidad de pulsos para controlar el desplazamiento angular y lograr una posición precisa; Se controla la frecuencia de los pulsos para controlar la velocidad y aceleración de giro del motor, logrando así un control de velocidad. Como en figura 3.6 se presenta un motor paso a paso:

Figura 3.6. motor paso a paso. (Funte: elaboración propia)

Tipo y parámetros del motor utilizado en este trabajo

Motores paso a paso (stepper) Kuman:
Model No.:17HD48002H-22B
Step Angle: 1.8 deg.
Holding Torque: 59Ncm(84 oz.in)
Rated Current/phase: 1.7A
Frame Size: 41 x 41mm
Body Length: 47mm
Shaft Diameter: 5mm
Shaft Length: 22mm
Weight: 380g

A continuación se muestra el diagrama de conexión del circuito de control[10]. En la esquina superior derecha se encuentran las conexiones entre Arduino y el controlador. VCC, pulso, dirección y habilitación son los 4 pines proporcionados por el microcontrolador. VCC se conecta a los tres terminales "+" del controlador. En figura 3.7 se enseña el diagrama de conexión del circuito de control:

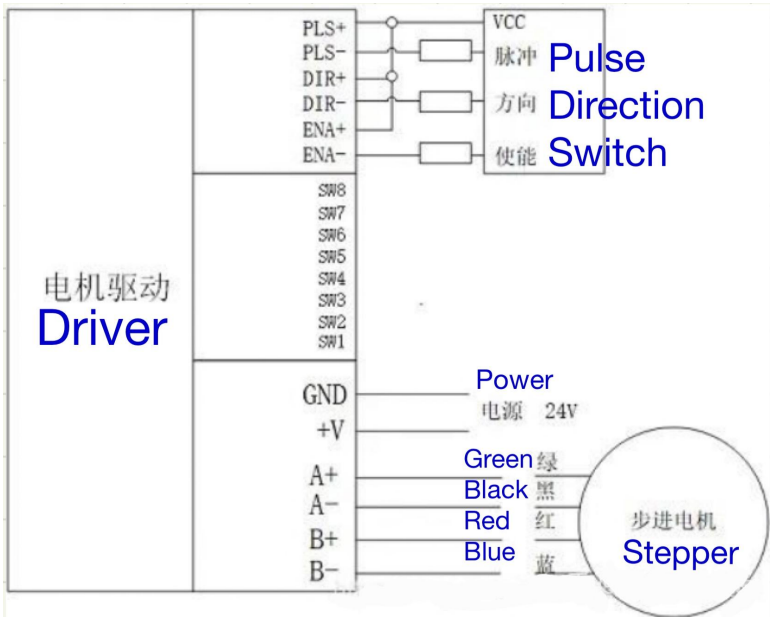


Figura 3.7. El diagrama de conexión del circuito de control. (Funte: [10])

3.3.2 Ejemplo de código para el control de los motores paso a paso

Aquí tenemos un ejemplo de programa de control. En este caso, se utilizan los pines SW2, SW3, SW4 y SW5 para la conexión con el controlador. Simplemente controlando la cantidad y la frecuencia de los pulsos, se puede controlar el ángulo y la velocidad de giro del motor. A continuación se muestra el código:

```
#define VCC 2
#define PLS 3
#define DIR 4
#define ENA 5
void setup() {
 pinMode(VCC, OUTPUT);
 pinMode(PLS, OUTPUT);
 pinMode(DIR, OUTPUT);
 pinMode(ENA, OUTPUT);
}
void loop() {
 digitalWrite(VCC, HIGH);
 digitalWrite(ENA, HIGH);
 digitalWrite(DIR, HIGH); //gira positiva
 //una revolucion positiva,  tarda 1,0 segundo
```

```
 //Aqui el driver define una revolucion como 1600 pasos
 for(int x=0; x<1600; x++){
   digitalWrite(PLS, HIGH);
   //Inicialmente, 1600 x 625 microsegundos = 1 segundo. Sin embargo, debido a que
       el intervalo de tiempo es muy corto y el ejecución del programa también
       requiere tiempo. Después de dividirlo por 2, el tiempo total es
       aproximadamente igual a 1 s.
   delayMicroseconds(625/2);
   digitalWrite(PLS, LOW);
   delayMicroseconds(625/2);
 }
 delay(1000); //una pausa de 1 s
 digitalWrite(DIR, LOW); //invertir la dirección de un motor
 //invertir una vuelta completa en 1 segundo
 for(int x=0; x<1600; x++){
   digitalWrite(PLS, HIGH);
   delayMicroseconds(625/2);
   digitalWrite(PLS, LOW);
   delayMicroseconds(625/2);
   }
 delay(1000); //una pausa de 1 s
}
```

El código anterior permite que el motor paso a paso gire hacia adelante 1 vuelta y luego se detenga durante 1 segundo, luego retroceda 1 vuelta y luego se detenga durante 1 segundo, el tiempo para girar 1 vuelta también es de 1 segundo, y así sucesivamente.
Si el circuito de control da un voltaje alto al pin ENA entonces es equivalente a dejar que el driver acepte la señal de control, si el ENA da un voltaje bajo, el motor no se moverá no importa como se dé la señal de pulso. El motor paso a paso puede controlarse generando una cierta frecuencia de pulsos en el pin PLS y controlando la temporización de los pulsos.

4. Códigos de control para este trabajo y su interpretación

En este apartado vamos a explicar el programa que usamos para llevar a cabo el control de varios movimientos de la cola.

4.1 Definiciones de macros

Para definir los parámetros importantes como la relación de reducción, el intervalo de tiempo, el paso (resolución angular) y el número de pasos, se pueden utilizar macros y asignar portes.

```
// Numero de veces que se repite el movimiento cuando se solicita
#define REPETITIONS 5
// RPM. Estar entre BASE_MOTOR_MIN_RPM y BASE_MOTOR_MAX_RPM (entre 50 y 300 si no hay cambios)
#define TAIL_RPM 100
// Timeout (pausa) en milisegundos entre cambios de d3ireccion movimiento
#define TIMEOUT 100

// CONSTANTES DE LOS MOTORES (MOTORES GEMELOS)
#define BASE_MOTOR_MIN_RPM 50
#define BASE_MOTOR_MAX_RPM 300
// Step angle del motor en º      por paso(Sin incluir reduccion)
#define MOTOR_STEP_ANLGE 1.80
// Reduccion (1 si no hay)
#define GEARBOX_REDUCTION 1
// Microstepping (depende de la placa o del driver)
// (El CNC V4 deberia tener 16, pero es mentira - porque esta mal soldada)
#define MICROSTEPPING_FACTOR 1

// Reduccion final (Gearbox + Microstepping)
#define REDUCTION (GEARBOX_REDUCTION*MICROSTEPPING_FACTOR)
// Pasos por revolucion teniendo en cuenta la reduccion
#define STEPS_PER_REVOLUTION (360.000 / MOTOR_STEP_ANLGE * REDUCTION)
// Precision del motor en arcosegundos
#define ARCSEC_RESOLUTION (360.0/STEPS_PER_REVOLUTION*60*60)

// Maxima velocidad (en RPM).
// Esta limitada por dos factores: el motor (incluyendo su reduccion mecanica) y el sw Arduino.
// - Respecto al motor: hay que probar ensayo / error.
// - Respecto al sw: Es una limitacin software del la librera Stepper.h, que define
```

```
// delay = 60000000 / steps / speed;
// por lo que si steps * speed es mayor de 60000 el retardo sera siempre el mismo
// (habremos alcanzado la velocidad maxima)
#define ARDUINO_MAX_RPM (60000000 / STEPS_PER_REVOLUTION)
#define MIN_USEC_DELAY_PER_STEP (1000000.0 / (BASE_MOTOR_MAX_RPM / 60.0)
/ STEPS_PER_REVOLUTION)

// Placa CNC V4:
#define X_DIR_PIN 2
#define Y_DIR_PIN 3
#define Z_DIR_PIN 4

#define X_STEP_PIN 5
#define Y_STEP_PIN 6
#define Z_STEP_PIN 7

#define X_ENABLE_PIN 8
#define Y_ENABLE_PIN 8
#define Z_ENABLE_PIN 8

long usecDelayPerStep = MIN_USEC_DELAY_PER_STEP;
```

4.2 Limitar la velocidad del motor al rango

```
void setTailRPM (long rpm) {
  long motorsRPM = rpm;

  if (rpm > BASE_MOTOR_MAX_RPM)
    motorsRPM = BASE_MOTOR_MAX_RPM;
  else if (rpm < BASE_MOTOR_MIN_RPM)
    motorsRPM = BASE_MOTOR_MIN_RPM;

  usecDelayPerStep = 1000000.0 / (motorsRPM / 60.0) / STEPS_PER_REVOLUTION;
}
double altCurrentAngle = 0.0;
double azCurrentAngle   = 0.0;
double altTargetAngle = 0.0;
double azTargetAngle   = 0.0;
```

4.3 Visualización de los parámetros y variables del

motor en la interfaz interactiva

```
void printMotorsSpecifications () {
  Serial.println("Motors specifications: ");
  Serial.print("   - Motor step angle: ");
  Serial.println(MOTOR_STEP_ANLGE);
  Serial.print("   - Steps per revolution: ");
  Serial.println(360.0/MOTOR_STEP_ANLGE);
  Serial.println("   + Reduction: ");
  Serial.print("      - Gearbox reduction: ");
  Serial.println(GEARBOX_REDUCTION);
  Serial.print("      - Board microstepping factor: ");
  Serial.println(MICROSTEPPING_FACTOR);
  Serial.print("      - Total reduction: ");
  Serial.println(REDUCTION);
  Serial.print("   - Steps per revolution [with reduction]: ");
  Serial.println(STEPS_PER_REVOLUTION);
  Serial.print("   - Resolution with gearbox and microstepping (arc sec): ");
  Serial.println(ARCSEC_RESOLUTION);
  Serial.println("   + RPM:");
  Serial.print("      - Motor minimum RPM: ");
  Serial.println(BASE_MOTOR_MIN_RPM);
  Serial.print("      - Motor maximum RPM: ");
  Serial.println(BASE_MOTOR_MAX_RPM);
  Serial.print("      - Arduino SW maximum RPM for this motor: ");
  Serial.println(ARDUINO_MAX_RPM);
  Serial.print("      - Minimum delay per step (in microsecs): ");
  Serial.println(MIN_USEC_DELAY_PER_STEP);

  Serial.println();
}
```

4.4 Realizar ajustes del motor

```
// PRE: Ya se ha fijado la direccion
void manualRotation (long altSteps, long azSteps, long altUsecStepDelay, long
azUsecStepDelay) {
  int y = 0;
  int z = 0;
  // Temporizadores para los pasos
```

```
  long currentMicros = 0;
  long lastAltMicros = 0;
  long lastAzMicros   = 0;

  // Pasos altura
  while (y < altSteps) {
    currentMicros =   micros();
    if (currentMicros - lastAltMicros >= altUsecStepDelay)
    {
      lastAltMicros = currentMicros;
      digitalWrite(Y_STEP_PIN,HIGH);
      y++;
      digitalWrite(Y_STEP_PIN,LOW);
    }
  }

  // Pasos azimut
  while (z < azSteps) {
    currentMicros =   micros();
    if (currentMicros - lastAzMicros >= azUsecStepDelay)
    {
      lastAzMicros = currentMicros;
      digitalWrite(Z_STEP_PIN,HIGH);
      z++;
      digitalWrite(Z_STEP_PIN,LOW);
    }
  }
}

long degreesToSteps (double degrees) {
  long steps = degrees / 360.0 * STEPS_PER_REVOLUTION;
  return steps;
}

void rotateSteppers (double altDegrees, double azDegrees) {
  long altSteps = degreesToSteps(altDegrees);
  long azSteps   = degreesToSteps(azDegrees);

  // Direccion altura
  if (altSteps < 0) {
    digitalWrite(Y_DIR_PIN,HIGH);
    altSteps = altSteps * -1;
  }
  else
```

```
    digitalWrite(Y_DIR_PIN,LOW);
  // Direccion azimut
  if (azSteps < 0) {
    digitalWrite(Z_DIR_PIN,HIGH);
    azSteps = azSteps * -1;
  }
  else
    digitalWrite(Z_DIR_PIN,LOW);

  // Se rotan en orden, primero altura (levantar/bajar cola), despues azimut (agitar
cola)
  manualRotation (altSteps, azSteps, usecDelayPerStep, usecDelayPerStep);
 // respecto al ultimo movimiento.
  delay (TIMEOUT);
}
```

4.5 Una serie de comandos preestablecidos

```
// Realiza movimientos en funcion de la "emocion" seleccionada.
// (Le indicamos la inicial de la emocion)
void moveTail(char emotion) {
  long altTargetSteps = 0.0;
  long azTargetSteps   = 0.0;
  int i = 0;

  switch (emotion) {
    // happy
    case 'h':
      // Cola para arriba y despues movimiento oscilante a izquierda y derecha
      rotateSteppers(-90.0, -90.0);
      for (i = 0; i < REPETITIONS; i++) {
        rotateSteppers(0.0, -180.0);
        rotateSteppers(0.0, 180.0);
      }

      // Vuelta a posicion de arranque
      rotateSteppers(90.0, 90.0);
      break;
    // sad
    case 's':
      rotateSteppers(90.0, 0.0);
```

```
        // En este caso, cada repeticion, se trata como un segundo
        delay(REPETITIONS * 1000);
        rotateSteppers(-90.0, 0.0);
        break;
      // angry
      case 'a':
        // TO DO
        break;
      default:
        break;
    }
}

void doAltAzRotationFromSerial(){
  char letra;
  //Serial.println("Esperando datos de puerto serie...");
  if (!Serial.available())
    return;
  //Serial.println("Leyendo datos de puerto serie...");
  letra = Serial.read();
  moveTail(letra);
}
```

4.6 Realizar la configuración de portes y pines

```
void setup(){
  pinMode(Y_DIR_PIN, OUTPUT);
  pinMode(Z_DIR_PIN, OUTPUT);

  pinMode(Y_STEP_PIN, OUTPUT);
  pinMode(Z_STEP_PIN, OUTPUT);

  // Velocidad de la cola en RPM
  // (tiene que estar entre BASE_MOTOR_MIN_RPM y BASE_MOTOR_MAX_RPM).
  // seria 50 min, 300 max
  setTailRPM(TAIL_RPM);
  Serial.begin(9600);
  //printMotorsSpecifications();
}
void loop(){
  doAltAzRotationFromSerial();
}
```

5.Problemas y soluciones durante el montaje

Debido a las limitaciones del equipo de fabricación de las piezas, durante el proceso de montaje del modelo se han presentado principalmente dos problemas. El primero es la dificultad para imprimir debido al tamaño excesivo de las piezas. El segundo problema es la frecuente aparición de fallos en la impresión, como muestra en figura 5.1 un ejemplo de fallo de impresión 3D.

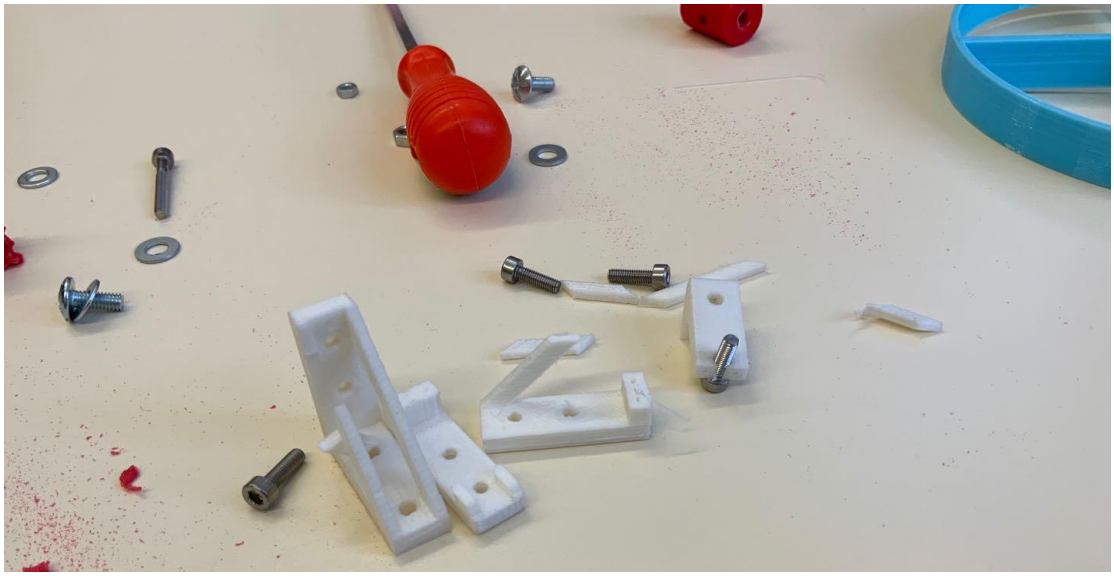


Figura 5.1. Fallo de piezas. (Funte: elaboración propia)

5.1 La placa base es demasiado grande para imprimir

Solución : Dividir las piezas en dos partes y volver a montarlas como mustra en figura 5.2:

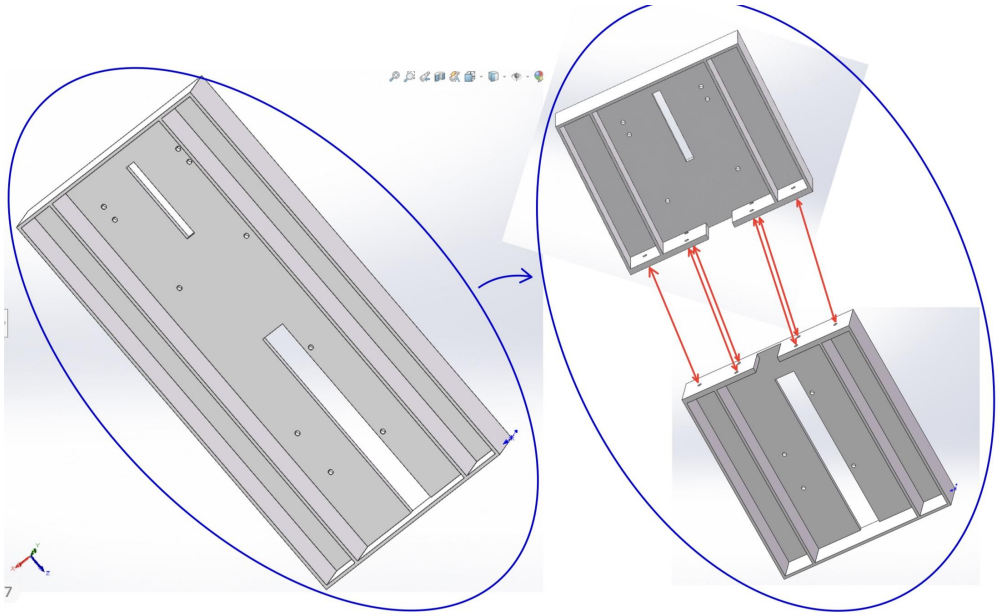
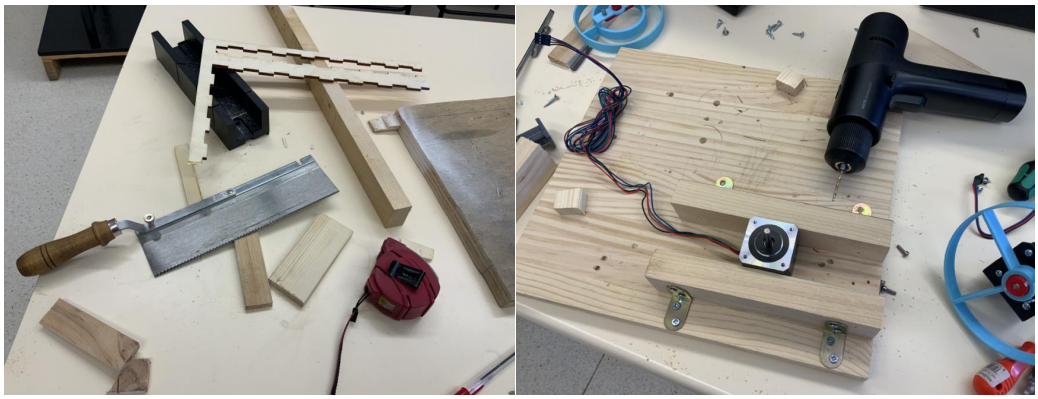


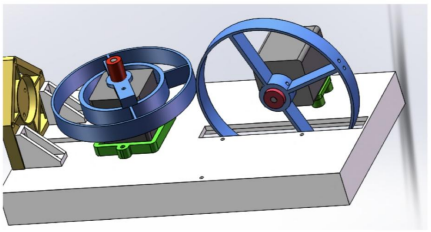
Figura 5.2. Dividir las piezas en dos partes y volver a montarlas. (Funte: elaboración propia)

5.2 Problemas de precisión y distorsión de piezas

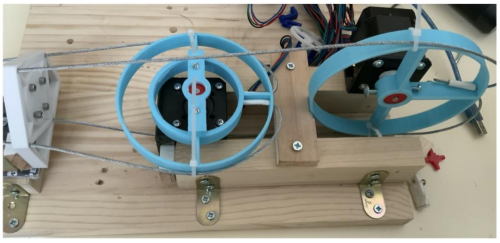
La impresión 3D tiene una precisión limitada y una eficiencia baja, lo que puede ocasionar deformaciones en las piezas, especialmente al imprimir piezas de gran tamaño (aproximadamente de 20 cm x 20 cm o más). Además, es común que ocurran pequeños fallos durante la impresión que hacen que la parte restante sea inutilizable.

Solución :
Con el fin de acelerar el proceso de prueba, mientras esperamos que se fabriquen las piezas en el laboratorio de impresión 3D, hemos decidido utilizar madera y fabricar manualmente las piezas de gran tamaño que no requieren una alta precisión. Manteniendo las relaciones geométricas y las formas de movimiento intactas, realizaremos un montaje equivalente para verificar la efectividad de la estructura mecánica, como se muestra en figura 5.3:





Prototipo final
(Todas las piezas imprimidas en 3D)



Prototipo temporal
(unas piezas hechas con mano)
(Sirve para comprobar el mecanismo)

Figura 5.3. un montaje equivalente. (Funte: elaboración propia)

6.Prueba

Después de completar el montaje del trabajo, realizamos pruebas en dos estados emocionales de las mascotas: excitación y calma.

6.1 Estado de excitación del pero

En primer lugar comprobamos el estado de excitacón como muestra en figura 6.1:

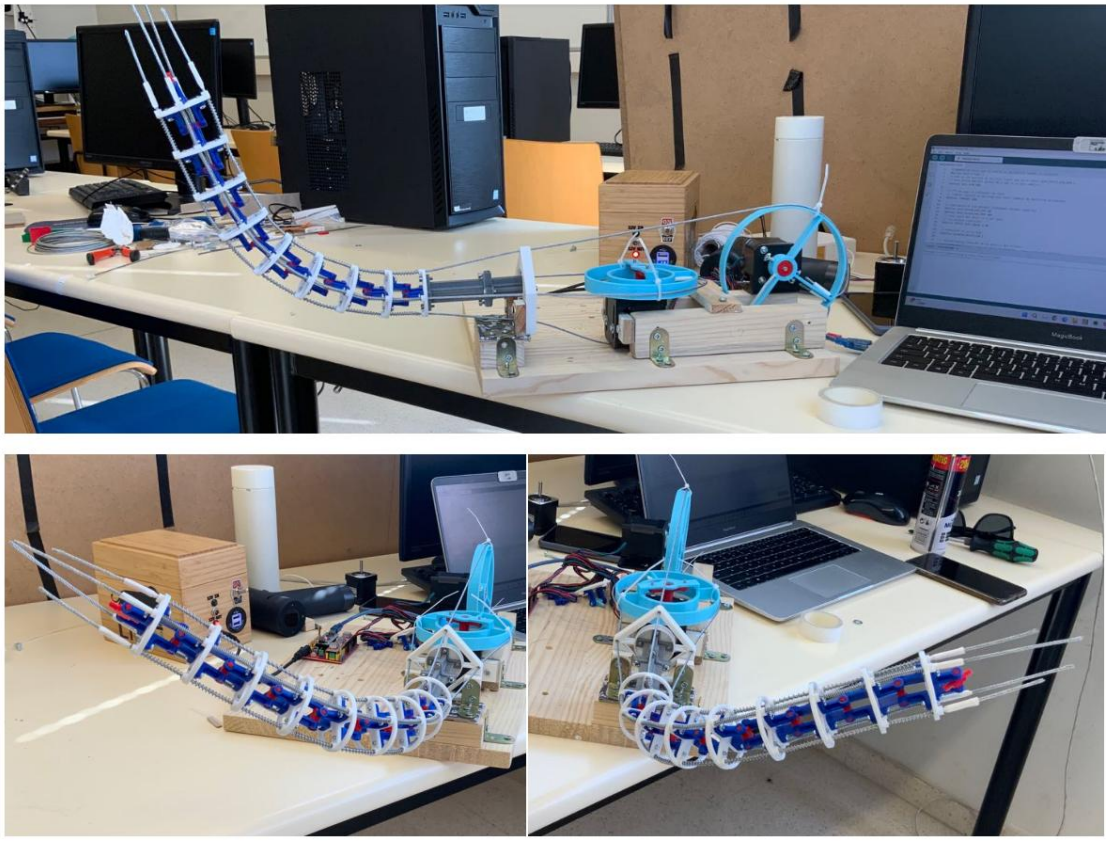


Figura 6.1. Estado de excitación del pero. (Funte: elaboración propia)

6.2 Estado de relajación

Luego comprobamos el estado de relajación como muestra en figura 6.2:

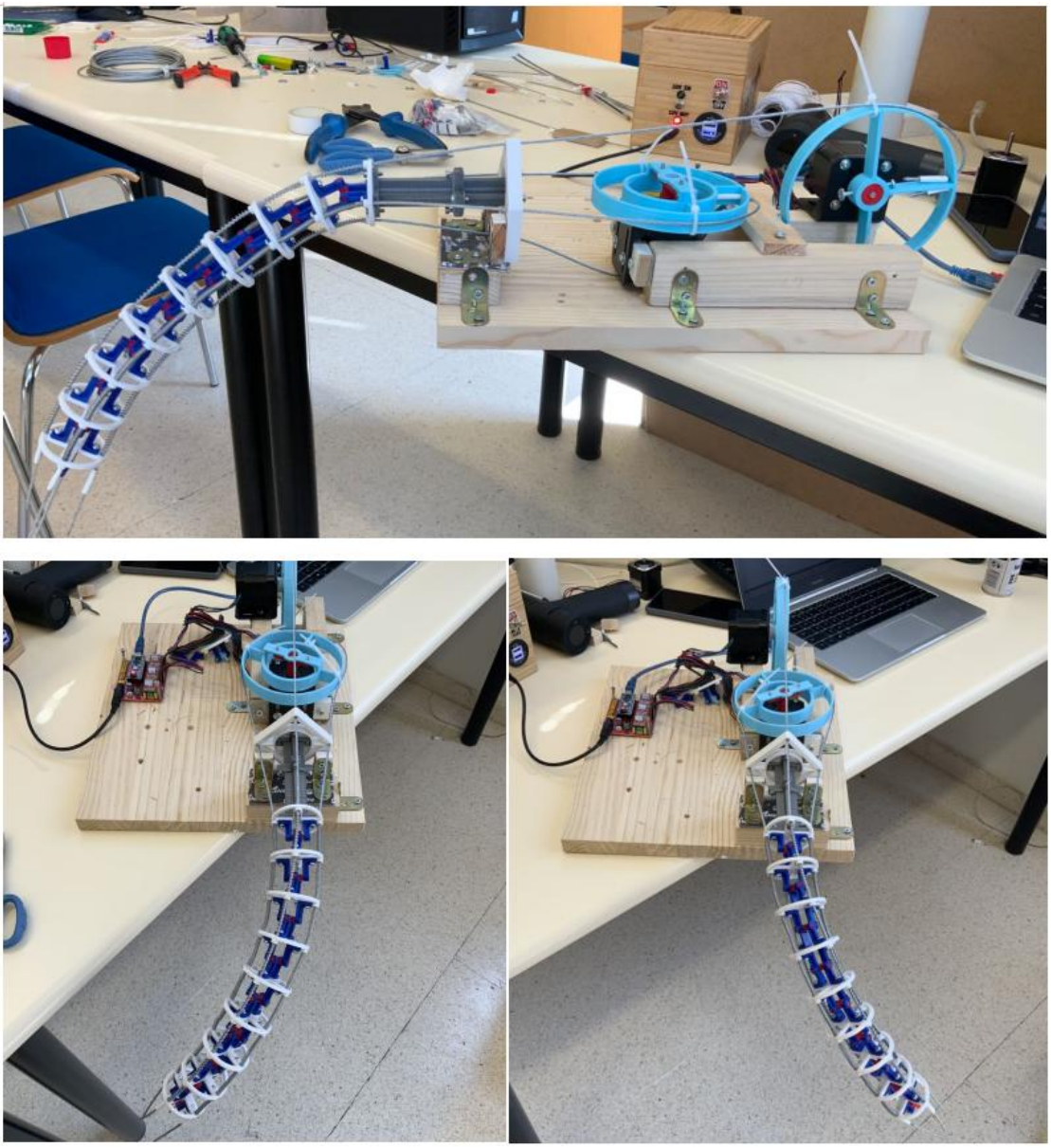


Figura 6.2. Estado de relajación. (Funte: elaboración propia)

Llegamos a la conclusión de que tanto la parte mecánica como la parte de control electrónico del trabajo son capaces de cumplir con los objetivos establecidos.

7. Presupuesto

Salvo el coste de los consumibles PLC para la impresión, que deberás consultar al personal del FABLAB de la Universidad de León, el resto de los costes del trabajo los puedes encontrar por tu cuenta.

En figura 7.1 se muestra el material consumible de PLC para impresión 3D:

Figura 7.1. Consumible de material PLC.(Funte: elaboración propia. Funte: Amazon)

Después de la consulta, utilizamos unos 200 gramos de consumibles PLC con un coste de unos 7 euros.

Aquí tenemos la tabla de prosupuesto:

Lista de componentes	Cantidad	Precios unitarios(€)	Precios totales(€)
Elementos de fijación	1 lote	31,55	31,55
Consumibles(material PLC)	200 g	7	7
Arduino nano	1 ud	2,99	2,99
Placa de expansión cnc shield V4	1ud	4,19	4,19
Caja de batería	1 ud	16,30	16,30

Motor paso a paso (Kuman)	2 ud	23,55	47,10
			Σ = 109,13 €

Tabla 7.1. Prosupuesto (Funte: elaboración propia)

8. Conclusiones

Este trabajo surge como un intento en el campo de la biomimética de robots cuadrúpedos, con el objetivo de lograr una interacción emocional con perros mascotas mediante la instalación de una cola mecánica en el robot Unitree A1.

El trabajo es un sistema integrado que consta de dos módulos principales: diseño mecánico y control.

El módulo de diseño mecánico consiste en la cola física y su mecanismo de transmisión. El cuerpo principal de la cola está compuesto por múltiples articulaciones universales conectadas en serie, impulsadas por dos pares de cables de acero: un par se conecta a una polea colocada horizontalmente, lo que permite que la cola oscile lateralmente; mientras que el otro par se conecta a una polea colocada verticalmente, permitiendo que la cola oscile verticalmente. Ambas poleas están conectadas a sus respectivos motores.

El módulo de control se encarga de controlar la velocidad, dirección y número de ciclos de los dos motores, lo que permite que la cola oscile de diversas formas. Este módulo está compuesto por una caja de suministro de energía, una placa de desarrollo Arduino Nano con la placa de expansión CNC V4 y el entorno de desarrollo integrado Arduino IDE. El Arduino IDE se instala en una computadora y se utiliza para escribir el código de control, el cual se compila en un archivo ejecutable binario. Posteriormente, este archivo se envía al Arduino Nano a través de una interfaz USB y un cable. El microprocesador en el Arduino Nano ejecuta el código binario para controlar el funcionamiento de los motores.

Este trabajo ha obtenido el primer premio en el concurso de prototipos de FGULEM-Universidad de León 2022-2023, actividad que se encuadra en el Plan de Transferencia de Conocimiento Universidad-Empresa 2021-2023 (Plan TCUE), aprobado mediante Acuerdo 134/2021, de 9 de diciembre de la Junta de Castilla y León (BOCYL No238 de 13 de diciembre de 2021), cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) y la Junta de Castilla y León.

9. Trabajo futuro

Expansión desde la perspectiva de la teoría del control:
Como se mencionó anteriormente, el control en este trabajo se realiza en un bucle abierto, donde la computadora envía directamente instrucciones de funcionamiento a los motores a través de la placa de desarrollo Arduino. En este enfoque, no se tiene en cuenta el efecto real del motor. Sin embargo, debido a la inercia de la cola, los cambios rápidos de dirección del motor pueden generar un golpe de par notable, lo que reduce la continuidad del movimiento de la cola e incluso puede dañar el motor.

Por lo tanto, se puede considerar agregar una retroalimentación utilizando un algoritmo de control de alimentación directa más un controlador PID, con el fin de construir un lazo de control cerrado. Esto permitiría que el proceso de accionamiento del motor sea más suave, proporcionando un amortiguamiento para frenados bruscos o cambios de dirección rápidos.

Dado que el mecanismo mecánico de este trabajo presenta una fricción significativa, es difícil realizar una modelización matemática mediante ecuaciones diferenciales. Por lo tanto, se propone realizar una identificación directa del sistema de la cola. Esto permitirá establecer rápidamente los parámetros adecuados de alimentación directa más PID para cada modo de oscilación.

En cuanto a la identificación del sistema, se considera que los dos torques de salida del motor son los parámetros clave en este trabajo. A través de la entrada y retroalimentación de estos dos torques, se pueden obtener las características de control del sistema. Al analizar la respuesta del sistema con diferentes estímulos, es posible obtener las características del sistema en el espacio de parámetros correspondiente. Como indicado en figura 9.1 tomamos dos pares como parámetros claves para la identificación del sistema.

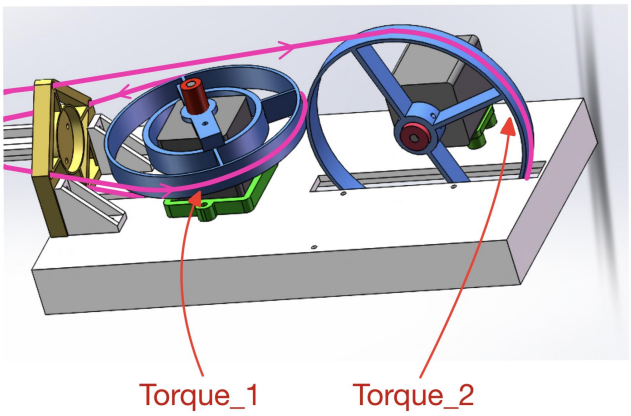


Figura 9.1. Dos parámetros claves para la identificación del sistema. (Funte: elaboración propia)

Basado en el algoritmo de alimentación anticipada (feedforward) y control proporcional integral derivativo (PID), consideramos el uso de MATLAB Simulink para diseñar un lazo de control de retroalimentación. A continuación se muestra un ejemplo de simulación de control de motor, que incluye el lazo de control y el código del algoritmo[11]. En figura 9.2 se musestra un ejemplo de flujo de control simulado en MATLAB Simulink.

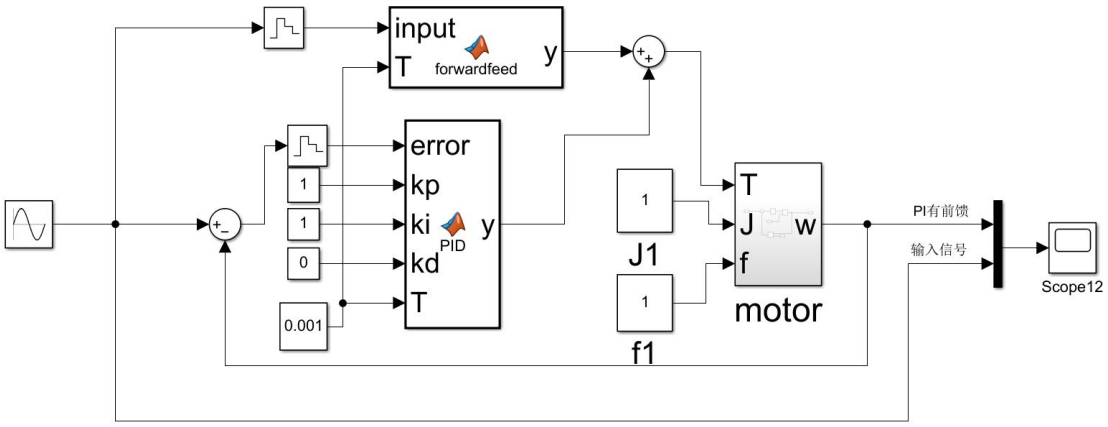
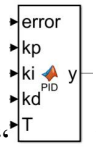


Figura 9.2. Diagrama de flujo de control. (Funte: [11])

Function “input T”  :

```
function y = forwardfeed(input,T)
persistent last_input;
if isempty(last_input)
        last_input = 0;
end
%Gf(s) = u_out(s)/u_in(s) = s+1
%Gf(z) = u_out(z)/u_in(z) = 2-z^-1
y = input+(input-last_input)/T;
last_input = input;
pause(T);
```

Function “  ”:

```
function y = PID(error,kp,ki,kd,T)
%inicialization
persistent i_out;
if isempty(i_out)
```

```
        i_out = 0;
end
persistent last_error;
if isempty(last_error)
        last_error = 0;
end
%pid culculation
p_out = error*kp;
i_out = i_out + error * ki * T;
d_out = (error - last_error) * kd / T;

%output the results
out = p_out + i_out + d_out;

last_error = error;

y = out;
pause(T);

%tout = out;
%iout = i_out;
```

10. Biografías

[1]Canine Conditioning Coach. Canine Anatomy:Glossary of terms
URL:https://canineconditioningcoach.com/canine-anatomy-terms/

[2]OC Robotics. OC Robotics - Snake arm 101
URL:https://www.youtube.com/watch?v=Ij8VX9YUT_Y

[3]OC Robotics. Series II, X125 snake-arm robot successfully delivered to University College London
URL:https://www.ocrobotics.com/news-en/series-ii-x125-snakearm-robot-successfully-delivered-to-university-college-london/

[4]ANIMAL LIBERATION. Docking a piglet's tail
URL:https://www.al.org.au/docking-a-piglets-tail

[5]huaweicloud. 小白入门 Arduino，一步一图搭建开发环境
URL:https://bbs.huaweicloud.com/blogs/303419

[6]CSDN. Arduino 极速入门教程——两篇文章让你会用 Arduino（上）
URL:https://blog.csdn.net/qq_44884716/article/details/113528255

[7]CSDN. Arduino Nano 引脚分配图及定义详解
URL:https://blog.csdn.net/malcolm_110/article/details/95320094

[8]Autodesk. How to Use the CNC V4 Board (despite Its "quirks")
URL:https://www.instructables.com/How-to-Use-the-CNC-V4-Board-despite-Its-quirks/

[9]CSDN.【Arduino】Arduino IDE 使用教程
URL:https://blog.csdn.net/as480133937/article/details/105331315

[10]zhihu. Arduino 单片机控制步进电机
URL:https://zhuanlan.zhihu.com/p/97117108

[11]gitee. forwardfeed_simulink
URL:https://gitee.com/hhrccc/forwardfeed_simulink.git

[12]Mr. Franklin's Science Lab. Mechanical design flow
https://franklinscience.weebly.com/design-process.html