



UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE MECATRÓNICA

**DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO ANIMATRÓNICO DE CABEZA Y
CUELLO PARA LA PROMOCIÓN Y DIFUSIÓN DE LA CARRERA DE MECATRÓNICA**

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTOR: MARCELO ANDRÉS BRAVARI PAZMIÑO

TUTORA: CARMEN JOHANNA CELI SÁNCHEZ

Quito – Ecuador

2026

CERTIFICADO DE RESPONSABILIDAD Y ADITORÍA DEL TRABAJO DEL TITULACIÓN

Yo, Marcelo Andrés Bravari Pazmiño con documento de identificación N° 0605144492; manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana puede usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 09 de Marzo de 2026

Atentamente,

A handwritten signature in blue ink that reads "Marcelo Bravari P". The signature is stylized with a large, looped 'P' at the end and a horizontal line underlining the name.

Marcelo Andrés Bravari Pazmiño

0605144492

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DEL TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Marcelo Andrés Bravari Pazmiño con documento de identificación N° 0605144492, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy el autor del Dispositivo Tecnológico: “Diseño y construcción de un prototipo animatrónico de cabeza y cuello para la promoción y difusión de la Carrera de Mecatrónica”, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 09 de Marzo de 2026

Atentamente,

A handwritten signature in blue ink that reads "Marcelo Bravari Pazmiño". The signature is stylized with a large, looped 'P' at the end and a horizontal line underlining the name.

Marcelo Andrés Bravari Pazmiño

0605144492

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DEL TITULACIÓN

Yo, Carmen Johanna Celi Sánchez con documento de identificación N° 1717437808, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO ANIMATRÓNICO DE CABEZA Y CUELLO PARA LA PROMOCIÓN Y DIFUSIÓN DE LA CARRERA DE MECATRÓNICA, realizado por Marcelo Andrés Bravari Pazmiño con documento de identificación N° 0605144492, obteniendo como resultado final el trabajo de titulación bajo la opción Dispositivo Tecnológico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 09 de Marzo de 2026

Atentamente,

A handwritten signature in blue ink, appearing to read 'Carmen Johanna Celi Sánchez', with a horizontal line underneath.

Ing. Carmen Johanna Celi Sánchez, Mgtr

1717437808

ÍNDICE

CERTIFICADO DE RESPONSABILIDAD Y ADITORÍA DEL TRABAJO DEL TITULACIÓN.....	ii
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DEL TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA.....	iii
CERTIFICADO DE DIRECCIÓN DEL TRABAJO DEL TITULACIÓN.....	iv
ÍNDICE DE FIGURAS	ix
ÍNDICE DE TABLAS.....	xii
RESUMEN.....	xiii
ABSTRACT	xiv
PLANTEAMIENTO DEL PROBLEMA	1
JUSTIFICACIÓN.....	2
OBJETIVOS	3
Capítulo I.....	4
Marco Teórico.....	4
1.1 Fundamentos teóricos de la robótica y la inteligencia artificial.....	4
 1.1.1. Robótica como disciplina de la ingeniería moderna.	4
1.2 El campo de la Inteligencia Artificial	5
1.3 Animatrónica como disciplina integrada.....	6
 1.3.1. El salto cognitivo en la síntesis mecatrónica.	6
Capítulo II	8
Diseño CAD.....	8
2.1 Diseño en CAD, procesamiento y fabricación de la carcasa animatrónica basada en personaje	8
2.2 Modelado paramétrico como base del diseño.....	8
2.3 Autodesk Fusion 360 como plataforma integrada CAD/CAE/CAM	9

2.4 Diferencia con modelado directo y justificación de la elección.....	9
2.5 Diseño Basado en Características y relación diseño–manufactura.....	10
2.6 Procesamiento de geometría y edición de mallas para integración estética– funcional.....	10
2.7 Diseño morfológico basado en personaje y criterios de escala	11
<i>2.7.1 Diseño de ojos, estructura.</i>	11
<i>2.7.2 Diseño estructura interna.....</i>	14
<i>2.7.3 Diseño del mecanismo tipo gimbal (cardán) para el cuello.</i>	17
<i>2.7.6. Actuación del yaw: servomotor MG90S y ventaja geométrica por accionamiento en el filo.....</i>	24
<i>2.7.7. Diseño y dimensionamiento del mecanismo de cuello.</i>	25
<i>2.7.8. Arquitectura real implementada: transmisión servo–horn–biela.</i>	26
<i>2.7.9. Momento gravitacional del conjunto y obtención del centro de masa.</i>	28
<i>2.7.10. Compensación elástica mediante resortes.</i>	31
2.7 Edición orgánica y reparación de mallas con Meshmixer	32
2.8 Reintegración en Fusion 360 y diseño de interfaces mecánicas para desmontaje	35
2.9 Estrategia de montaje desmontable y ensamblaje por acople + adhesión	37
2.10 Preparación de impresión 3D: corte, orientación y configuración en Creality Print.....	38
<i>2.10.1. Creality Print 6.3: el entorno de laminado (slicer) como interfaz CAD– manufactura.</i>	41
<i>2.10.2 Selección del material de impresión: PLA.</i>	44
2.11 Desafíos encontrados y aprendizajes en el diseño	45
2.12 Iteraciones del diseño y resultado final de la carcasa animatrónica	47
2.13 Validación y confirmación del diseño final	49
Capítulo III	52
Arquitectura e implementación del sistema electrónico del animatrónico.....	52

Arquitectura e implementación del sistema electrónico del animatrónico	52
3.1 Servomotores en sistemas animatrónicos.....	52
3.2 Principio de control por PWM y lazo de posición.....	53
3.3 Parámetros de selección relevantes para animatrónica.....	53
3.4 Servomotores empleados en el prototipo y justificación funcional.	54
3.5 Control multi servo con PCA9685 y calibración por ancho de pulso.....	55
3.6 Implicaciones eléctricas: picos de corriente, caída de tensión y robustez del sistema.....	56
3.7 Arquitectura de alimentación y distribución de potencia del prototipo.	56
<i>3.7.1. Separación de rieles: cómputo independiente y potencia externa para servos. .</i>	<i>57</i>
<i>3.7.2. Fuente conmutada para bus de actuadores ($V^+ = 5.15 V$).</i>	<i>57</i>
<i>3.7.3. Comparación técnica con alternativa portátil: baterías + LM2596.....</i>	<i>60</i>
<i>3.7.4 Regímenes de consumo según simultaneidad real del software de control.</i>	<i>62</i>
<i>3.7.5. Implicaciones de diseño: margen, estabilidad y confiabilidad del bus.</i>	<i>63</i>
Capítulo IV	66
Implementación de un Sistema Animatrónico Controlado por Voz mediante IA en Raspberry Pi 4B.....	66
Implementación de un Sistema Animatrónico Controlado por Voz mediante IA en Raspberry Pi 4B	66
4.1 Elección del Hardware: Raspberry Pi 4B	66
4.2 Implementación Inicial: DeepSeek 1.5B en CPU y Pipeline de Voz Integrado	67
<i>4.2.1 Captura de voz (Adquisición de audio).....</i>	<i>67</i>
<i>4.2.2 Transcripción (ASR / STT) con Whisper.</i>	<i>69</i>
<i>4.2.3 Inferencia del modelo de lenguaje (NLU/LLM) mediante Ollama.</i>	<i>70</i>
<i>4.2.4 Síntesis de voz (TTS) con Piper y reproducción con ALSA.</i>	<i>71</i>
4.3 Optimización del Modelo LLM: Cambio de DeepSeek 1.5B a LLaMA 3.2:1B... 72	72
<i>4.3.1 Desafíos Técnicos en la Transición de Modelos LLM.</i>	<i>73</i>

4.3.2 Descripción Detallada del Script <i>hablarr_ia.py</i>	74
4.4 Extensión para Sincronización de la Mandíbula con el Habla	76
4.4.1 Control del servomotor (mandíbula)	77
4.4.2 Algoritmo de sincronización mediante sílabas.....	77
Capítulo V.....	79
Pruebas y resultados	79
5.1 Resultado final y funcionamiento del sistema	79
5.2 Metodología de pruebas e integración incremental	80
5.3 Pruebas eléctricas y de actuadores.....	80
5.4 Pruebas mecánicas del cuello y selección del mecanismo funcional.....	82
5.5 Migración de control a Raspberry Pi y ejecución continua de rutinas	82
5.6 Pruebas del sistema de interacción por voz y selección del LLM	82
5.7 Integración del movimiento de mandíbula durante la locución.	84
5.8 Resultados económicos y estimación de costos.	85
Conclusiones	89
Recomendaciones.....	89
Referencias.....	91

ÍNDICE DE FIGURAS

Figura 1. <i>Vista lateral del ensamblaje ocular</i>	12
Figura 2. <i>Renderizado isométrico del módulo ocular completo con servomotores integrados.</i>	12
Figura 3. <i>Simulación de movimiento de párpados de un ojo.</i>	12
Figura 4. <i>Simulación de movimiento de párpados de dos ojos.</i>	12
Figura 5. <i>Ensamble del sistema ocular dual, montado sobre la base estructural principal.</i> 13	
Figura 6. <i>Todas las piezas del sistema ocular.</i>	13
Figura 7. <i>Ensamble del sistema ocular, visión superior.</i>	13
Figura 8. <i>Ensamble del sistema ocular, visión frontal.</i>	13
Figura 9. <i>Ensamblaje ojos.</i>	14
Figura 10. <i>Vista en corte de la estructura interna preliminar con alojamiento para mandíbula.</i>	14
Figura 11. <i>Detalle del anclaje para el servomotor de la mandíbula (MG996R) dentro del chasis.</i>	14
Figura 12. <i>Renderizado de la estructura interna optimizada para flujo de aire y paso de cables.</i>	15
Figura 13. <i>Análisis de la sección transversal de la carcasa mostrando donde va el servomotor MG996R.</i>	15
Figura 14. <i>Ensamble digital de la estructura interna con Mandíbula y hocico.</i>	16
Figura 15. <i>Vista izquierda del ensamblaje de estructura interna + sistema ocular + mandíbula.</i>	16
Figura 16. <i>Vista superior del ensamblaje de estructura interna + sistema ocular + mandíbula.</i>	17
Figura 17. <i>Articulación universal del Gimbal.</i>	17
Figura 18. <i>Cuerpos impresos usados en la realización del Gimbal.</i>	18
Figura 19. <i>Cuerpos usados en la construcción del Gimbal.</i>	19
Figura 20. <i>Diseño conceptual del mecanismo Gimbal para el cuello con 2 grados de libertad.</i>	20
Figura 21. <i>Base fija Fusion 360.</i>	21
Figura 22. <i>Disco principal rotante Fusion 360.</i>	21
Figura 23. <i>Diseño del caño de acero a implementar.</i>	22

Figura 24. <i>Plataforma superior del cuello Fusion 360.</i>	22
Figura 25. <i>Disco estabilizador Fusion360.</i>	23
Figura 26. <i>Base fija + Disco estabilizador + HDD.</i>	23
Figura 27. <i>Vista superior del disco principal rotante + MG90S+ HDD.</i>	24
Figura 28. <i>Diseño del conjunto completo Fusion 360.</i>	24
Figura 29. <i>Diseño adoptado para el cuello de la cabeza animatrónica.</i>	25
Figura 30. <i>Distancia eje-orificio del brazo de palanca efectivo del servo.</i>	27
Figura 31. <i>Ángulo entre el horn y la dirección de la biela.</i>	27
Figura 32. <i>Diagrama “brazo de palanca”.</i>	27
Figura 33. <i>Centro de masa en Fusion360Diagrama</i>	29
Figura 34. <i>Distancia desde el eje/pivote de inclinación hasta el centro de masa.</i>	30
Figura 35. <i>Diagrama “Momento Gravitacional”.</i>	31
Figura 36. <i>Vista de la estructura interna, evidenciando el espacio reservado para el servomotor MG996R.</i>	33
Figura 37. <i>Vista del corte hecho a la estructura.</i>	33
Figura 38. <i>Edición orgánica de la carcasa.</i>	34
Figura 39. <i>Múltiple división de la carcasa para posterior impresión.</i>	35
Figura 40. <i>Validación de diseños finales de carcasa superior</i>	36
Figura 41. <i>Validación de diseños finales de carcasa inferior</i>	36
Figura 42. <i>Soportes solidarios a la carcasa.</i>	37
Figura 43. <i>Acople a la carcasa adherido a la estructura interna.</i>	38
Figura 44. <i>Vista de laminado en Creality Print, carcasa superior.</i>	38
Figura 45. <i>Vista de laminado en Creality Print, carcasa inferior</i>	39
Figura 46. <i>Vista de (AI camera) CrealityK1C --Paso a CrealityPrint 6.3.</i>	40
Figura 47. <i>Vista Creality Print de los soportes y acoples.</i>	40
Figura 48. <i>Laminado y velocidades de impresión.--Vista de un cuerpo en Fusion360 previo a ser impreso.</i>	41
Figura 49. <i>Paso a Creality Print.</i>	42
Figura 50. <i>Diseño tren de engranajes.</i>	46
Figura 51. <i>Vista derecha, diseño impreso tren de engranajes.</i>	46
Figura 52. <i>Diseño inicial del modelo actual.</i>	47
Figura 53. <i>Apertura de mandíbula animatrónico (real).</i>	48
Figura 54. <i>Diseño completo en Fusion360 del animatrónico.</i>	50
Figura 55. <i>Animatrónico impreso.</i>	50

Figura 56. <i>Dimensiones del MG90S según su DataSheet.</i>	54
Figura 57. <i>Dimensiones del MG996R según su DataSheet.</i>	55
Figura 58. <i>Raspberry Pi conectada con I2C.</i>	56
Figura 59. <i>Vista dercha S-50-5.</i>	59
Figura 60. <i>Montaje inicial: Li-ion + LM2596 + Arduino Mega + servomotor.</i>	62
Figura 61. <i>Primeras pruebas Sistema ocular.</i>	81
Figura 62. <i>Comportamiento no deseado del LLM en pruebas iniciales: inclusión de razonamiento (“Thinking...”) y contenido no estrictamente alineado con la salida solicitada.</i>	83

ÍNDICE DE TABLAS

Tabla 1. <i>Especificaciones de 50W Single Output Switching Power Supply.</i>	58
Tabla 2. <i>Corriente total estimada por regímenes.</i>	63
Tabla 3. <i>Latencias de respuesta del animatrónico en pruebas por texto y por voz.</i>	84
Tabla 4 <i>Lista de componentes con sus precios para el Animatronic.</i>	85

RESUMEN

En el presente trabajo se diseñó y construyó un prototipo animatrónico de cabeza y cuello basado en un personaje de las Tortugas Ninja. El objetivo principal fue desarrollar un muñeco interactivo para la promoción de la carrera de Mecatrónica, integrando mecánica, electrónica y sistemas de control por voz con inteligencia artificial. Para el diseño mecánico se emplearon herramientas de Diseño Asistido por Computadora (CAD), destacando Autodesk-Fusion 360 para diseñar y modelar las estructuras internas, complementado con software de edición orgánica de mallas como Meshmixer para optimizar las partes externas del personaje. Asimismo, se utilizaron técnicas de fabricación mediante impresión 3D para materializar los diseños hechos en CAD, apoyándose en la aplicación de la impresora Creality K1C y del programa Creality Print para la edición de dichas piezas, su orientación óptima, su densidad, algunas configuraciones en los parámetros de impresión.

El sistema animatrónico resultante está controlado por una Raspberry Pi 4B de 4GB de RAM, la cual gestiona un asistente de voz impulsado por un modelo de lenguaje ligero y algoritmos de reconocimiento y síntesis de voz. El prototipo animatrónico es capaz de interactuar con usuarios mediante comandos de voz, simula hablar en tiempo real, moviendo la cabeza y abriendo o cerrando la mandíbula en forma sincronizada con la respuesta hablada por el asistente. En este documento se detallan los procesos de diseño en CAD de las estructuras y la carcasa del personaje, la integración de los servomotores para brindar movimiento al cuello y la mandíbula, así como los desafíos técnicos superados durante el desarrollo (por ejemplo, la revisión de mecanismos por limitaciones de torque de ciertos servomotores). Los resultados demuestran la viabilidad de un animatrónico de bajo costo y refuerzan la importancia de la iteración en ingeniería: a través de sucesivos rediseños y pruebas, se logró un modelo funcional que combina un aspecto fiel al personaje con un funcionamiento mecatrónico confiable y seguro.

Palabras clave: animatrónico, reconocimiento de voz, asistente de voz, Raspberry Pi, inteligencia artificial, servomotor, Mecatrónica.

ABSTRACT

In this project, an animatronic head and neck prototype was designed and built based on a character from the Teenage Mutant Ninja Turtles. The main objective was to develop an interactive doll to promote the Mechatronics degree programme, integrating mechanics, electronics and voice control systems with artificial intelligence. Computer-aided design (CAD) tools were used for the mechanical design, notably Autodesk-Fusion 360 to design and model the internal structures, complemented by organic mesh editing software such as Meshmixer to optimise the external parts of the character. Likewise, 3D printing manufacturing techniques were used to materialise the CAD designs, relying on the Creality K1C printer and the Creality Print programme to edit these parts, optimise their orientation and density, and configure some of the printing parameters.

The resulting animatronic system is controlled by a Raspberry Pi 4B with 4GB of RAM, which manages a voice assistant powered by a lightweight language model and voice recognition and synthesis algorithms. The animatronic prototype is capable of interacting with users through voice commands, simulating speech in real time, moving its head and opening or closing its jaw in synchronisation with the assistant's spoken response. This document details the CAD design processes for the character's structures and casing, the integration of servomotors to provide movement to the neck and jaw, as well as the technical challenges overcome during development (e.g., the revision of mechanisms due to torque limitations of certain servomotors). The results demonstrate the viability of a low-cost animatronic and reinforce the importance of iteration in engineering: through successive redesigns and tests, a functional model was achieved that combines a faithful appearance to the character with reliable and safe mechatronic operation.

Keywords: animatronic, speech recognition, voice assistant, Raspberry Pi, artificial intelligence, servomotor, Mechatronics

PLANTEAMIENTO DEL PROBLEMA

¿Cuáles son los desafíos actuales para captar la atención e interés del público hacia la ingeniería Mecatrónica, y de qué manera la implementación de un animatrónico interactivo de cabeza y cuello podría contribuir a difundir y promocionar la carrera, mostrando sus aplicaciones tecnológicas de forma atractiva y dinámica? La formación en Mecatrónica abarca campos de robótica, electrónica, control e inteligencia artificial, cuyo interés se vio potenciado con el cine comercial de Hollywood, y a menudo no es plenamente conocido por estudiantes preuniversitarios o por la comunidad en general que esta es la carrera que cumple esa visión de un científico. Los métodos tradicionales de difusión académica (charlas, trípticos, exposiciones estáticas) pueden resultar muy poco llamativas en comparación con demostraciones tecnológicas interactivas. Por ello, se plantea el desarrollo de un dispositivo animatrónico capaz de “dar vida” a una muestra tecnológica, interactuando con las personas mediante el habla y movimientos, como una estrategia innovadora para la promoción de la carrera, como una publicidad que se adapta a tiempos actuales.

En la actualidad, atraer la atención de jóvenes hacia las ingenierías requiere experiencias más inmersivas e innovadoras que un tríptico. La ausencia de herramientas didácticas altamente interactivas puede limitar el alcance de las campañas de difusión de la carrera de Mecatrónica. Un animatrónico conversacional presenta la oportunidad de abordar este problema al combinar elementos lúdicos y tecnológicos en una sola plataforma, permitiendo que la publicidad hacia la carrera sea visible y sonoro [1]. No obstante, el desarrollo de tal dispositivo implica retos técnicos significativos: integración de hardware mecánico con sistemas de control electrónico, implementación de algoritmos de inteligencia artificial en hardware de recursos limitados, y sincronización fluida entre voz y movimiento en tiempo real. Este trabajo busca resolver cómo enfrentar estos retos para construir un animatrónico funcional y atractivo que responda en tiempo real para poder así promocionar la carrera de Mecatrónica ante un público contemporáneo exigente en interacción y novedad.

JUSTIFICACIÓN

La creciente demanda de métodos innovadores para difundir las carreras STEM (Ciencia, Tecnología, Ingeniería y Matemáticas) motiva la adopción de nuevas herramientas tecnológicas, que sobre todo sean llamativas en concordancia con tiempos modernos donde la atención es muy superflua. En particular, la carrera de Mecatrónica, por su naturaleza multidisciplinaria, se beneficia de demostraciones prácticas que muestren la convergencia de sus campos. Un prototipo animatrónico interactivo constituye una vitrina ideal para ello, al integrar mecánica, electrónica e inteligencia artificial en un solo sistema visible y accesible al público.

La implementación de este proyecto tiene varias justificaciones clave:

- Un animatrónico que habla y se mueve tiene un alto poder de atracción sobre las personas, especialmente jóvenes. Su presencia en ferias educativas o eventos de promoción capta la atención de la audiencia de manera más efectiva que carteles o expositores tradicionales, creando una experiencia memorable que puede traducirse en un mayor impacto e interés por la Mecatrónica [1].
- El prototipo combina hardware (sensores, actuadores, microcomputadora) y software (algoritmos de voz e IA). Esto permite demostrar en tiempo real las aplicaciones de la Mecatrónica desde el control de servomotores hasta el reconocimiento de voz evidenciando el carácter práctico y moderno de la carrera. Los espectadores pueden así apreciar de forma tangible la relación entre teoría y práctica en esta carrera [2].
- Al desarrollar y presentar un dispositivo de estas características, la universidad se posiciona a la vanguardia en métodos de difusión académica. Además, el animatrónico puede servir como herramienta didáctica en futuras actividades, inspirando a estudiantes actuales a profundizar en proyectos de robótica e inteligencia artificial, y motivando a futuros postulantes al mostrarles de antemano lo que podrían crear al estudiar Mecatrónica [2].
- A diferencia de materiales impresos o presentaciones estáticas que pueden volverse obsoletas, un sistema animatrónico es reprogramable y escalable, y su plataforma podría reutilizarse en distintos contextos de divulgación, como en distintas ferias al ser transportable, haciendo de esta inversión un recurso sostenible para la promoción continua de la carrera.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar un prototipo animatrónico de cabeza y cuello para la promoción de la carrera Mecatrónica

OBJETIVOS ESPECÍFICOS

- Diseñar un prototipo animatrónico enfocado en la cabeza y el rostro, capaz de emular movimientos coordinados del rostro y la cabeza mediante software de ingeniería para su implementación.
- Construir el animatrónico utilizando tecnologías avanzadas como la impresión 3D para componentes personalizados y técnicas de mecanizado empleando métodos de manufactura adecuados para la integración óptima de las piezas mecánicas y electrónicas.
- Realizar pruebas de funcionamiento concentrándose en los movimientos de los ojos, cabeza y cuello garantizando su utilidad en la Carrera de Mecatrónica.

Capítulo I

Marco teórico

Este capítulo reúne los conceptos, antecedentes y definiciones necesarios para enmarcar el trabajo dentro de las disciplinas que lo intersectan. Su propósito es establecer un lenguaje técnico común y un contexto conceptual que permita la interpretación, en los siguientes capítulos, del desarrollo del prototipo y su comportamiento como sistema. En consecuencia, se abordan de manera ordenada los fundamentos de la robótica, la inteligencia artificial y la animatrónica, destacando los elementos teóricos que son relevantes para comprender la naturaleza interdisciplinaria del proyecto.

1.1 Fundamentos teóricos de la robótica y la inteligencia artificial

Este capítulo establece las bases conceptuales que sustentan el desarrollo del animatrónico, entendiendo el sistema como una integración Mecatrónica orientada a reproducir movimientos expresivos (mirada, postura y gestos) mediante varios actuadores, transmisión mecánica, electrónica de control y procesamiento computacional. En este marco, la robótica aporta los principios de diseño y control de sistemas capaces de ejecutar acciones físicas con distintos niveles de autonomía, los cuales en este proyecto solo serán los del habla, siendo los movimientos de gesticulación completamente dependientes. Mientras que la inteligencia artificial extiende la interacción al plano perceptivo y comunicativo, habilitando respuestas y comportamientos más naturales cuando se incorporan módulos de reconocimiento y generación de lenguaje.

1.1.1. Robótica como disciplina de la ingeniería moderna

La robótica se consolida en la Ingeniería Mecatrónica como un campo orientado al diseño, construcción y operación de máquinas capaces de ejecutar tareas que históricamente han requerido esfuerzo humano, con distintos niveles de autonomía y capacidad de interacción con su entorno. En términos contemporáneos, la robótica no es solo ejecutar tareas que han requerido del arduo esfuerzo humano, en tiempos contemporáneos la robótica ha trascendido ese puesto, un robot puede entenderse como una máquina operada automáticamente o con interacción directa con el ser humano, y la robótica como la disciplina de ingeniería asociada al desarrollo y uso de dichas máquinas. [3]

Aunque la robótica moderna es reciente, su raíz conceptual se encuentra en la tradición de los “Autómatas”, artefactos mecánicos diseñados para producir movimiento aparentemente “espontáneo” una vez iniciados, frecuentemente con fines artísticos o de exhibición técnica, como es en este caso. Esta línea histórica atraviesa desde mecanismos antiguos y medievales hasta los autómatas de precisión del periodo preindustrial, que funcionan como antecedentes directos de la Mecatrónica: mecanismos que integran transmisión de movimiento, regulación y “programación” mediante levas, engranajes y secuencias mecánicas predefinidas. [4]

Jacques de Vaucanson exhibió figuras que imitaban funciones humanas y animales (p. ej., el flautista y el pato mecánico), evidenciando el salto desde mecanismos ornamentales hacia sistemas que reproducen patrones coordinados de movimiento con intención mimética [5]. Posteriormente, el autómata de Henri Maillardet (dibujante-escritor) ejemplifica una forma avanzada para su época de “memoria mecánica” mediante levas, capaz de reproducir varios dibujos y textos: antecesor directo del concepto moderno de “programa”, en tanto secuencia almacenada que gobierna la salida del sistema. [6]

Sin embargo, la Robótica Industrial como práctica sistemática se consolida después de 1950, cuando convergen electrónica, control y manufactura automatizada. Un hito histórico y documentado es la instalación del primer robot industrial “Unimate” en una línea de ensamblaje (1961), lo cual marca el tránsito desde la automatización rígida hacia manipuladores reprogramables orientados a productividad y seguridad en entornos industriales, predecesores de la vida actual moderna [7].

1.2 El campo de la Inteligencia Artificial

La Inteligencia Artificial (IA) se comprende actualmente, con enfoque técnico-operativo, como el desarrollo de sistemas basados en máquinas que infieren desde la entrada que reciben cómo producir salidas tales como predicciones, recomendaciones, decisiones o contenido, con capacidad de influir en entornos físicos o virtuales, básicamente como Inteligencias Artificiales Generativas con diferentes grados de autonomía y adaptabilidad tras su despliegue[8].

Dentro de la IA, el Aprendizaje Automático (Machine Learning) se considera un eje central por su capacidad de construir modelos a partir de datos y experiencia. En formulación clásica, un programa “aprende” cuando mejora su desempeño en una tarea a partir de experiencia, aquí se usan “Parámetros de entrenamiento”, esta medida será un criterio de rendimiento, por los

parámetros de entrenamiento que puede o no soportar nuestro Hardware, estos parámetros formalizan el aprendizaje como optimización basada en evidencia empírica. [9]

Un caso representativo aparece en Ingeniería de Software, particularmente en trazabilidad de requisitos (requirements traceability), donde la literatura especializada ha descrito el problema como la necesidad de relacionar y mantener vínculos consistentes entre requisitos, diseño, implementación y verificación a lo largo del ciclo de vida[10].

1.3 Animatrónica como disciplina integrada

La animatrónica puede definirse como un campo interdisciplinario que integra principios de animación, diseño mecánico, electrónica y control, con el objetivo de construir figuras capaces de reproducir movimientos orgánicos y expresivos con la intención de un alto realismo perceptual. En contraste con robots industriales optimizados para exactitud, repetitividad y eficiencia, la robótica industrial se aleja del realismo perceptual, mientras la animatrónica se orienta a la verosimilitud, buscando convencer al observador de que la criatura posee intención, emoción y “vida” mediante micro-movimientos, ritmos y sincronía audiovisual.

Desde el punto de vista mecatrónico, el animatrónico se materializa como un sistema de actuadores (servomotores, neumática o hidráulica), transmisiones mecánicas (bielas, palancas, engranajes, cables) y una arquitectura de control que puede operar por secuencias preprogramadas o teleoperación. La popularización contemporánea del término y su estándar industrial se asocian históricamente a la tecnología Audio-Animatronics® desarrollada por Disney, documentada como introducida en atracciones de parques temáticos a inicios de la década de 1960 (p. ej., *Enchanted Tiki Room*, 1963), y posteriormente escalada a figuras de mayor complejidad narrativa, como las que conocemos ahora. [11]

Bajo este enfoque, el diseño del cuello y la cabeza en una figura animatrónica constituye una interfaz expresiva de alto impacto: pequeños cambios en inclinación, rotación y postura alteran de manera definitiva la lectura emocional del personaje. Por ello, mecanismos como (yaw-pitch) y variantes con roll o inclinación lateral suelen considerarse fundamentales en la “gramática corporal” del animatrónico.

1.3.1. El salto cognitivo en la síntesis mecatrónica

La transición desde autómatas clásicos limitados a “programación” mecánica determinista hacia animatrónica controlada por plataformas embebidas introduce una diferencia esencial: la capacidad de computación general y, más recientemente, la posibilidad de incorporar modelos

de IA generativa que producen salidas no estrictamente predefinidas. En particular, al integrar un Modelo de Lenguaje Grande (LLM) en la arquitectura de control, el animatrónico deja de estar restringido a repertorios cerrados de movimientos y diálogos, y adquiere capacidad de procesar lenguaje natural y generar contenido (respuestas) a partir de entradas como voz o texto. Esto cambia la concepción de animatrónica como la conocemos, siendo movimientos y diálogos repetitivos, la tortuga ninja por realizarse va mucho más allá, haciendo que este salto de en el clavo para lo que se le necesita, que es interactuar con el público[12].

Capítulo II

Diseño CAD

El animatrónico fue diseñado en modelos 3D tanto para propósitos de modelado con la adición de mecanismos internos, estructuras de montaje, restricciones de ensamblaje y fabricación aditiva para crear el diseño del modelo. Dado que el prototipo está destinado a la difusión pública y promoción académica de la Carrera de Ingeniería Mecatrónica, el diseño consideró ciertos requisitos en la construcción y su postproducción (rigidez, tolerancias, mantenibilidad, seguridad) y algunos otros requisitos comunicacionales (reconocibilidad del personaje, impacto visual y operación confiable en demostraciones públicas). En este método, el desarrollo se realizó como una iteración iterativa: cada iteración del modelo se modificó para evidenciar el ensamblaje en tienda, interferencias detectadas, rendimiento del actuador y limitaciones de fabricación.

2.1 Diseño en CAD, procesamiento y fabricación de la carcasa animatrónica basada en personaje

Desde el CAD y el ensamblaje hasta las pruebas y la trazabilidad de efectos físicos en sistemas mecatrónicos está representado por un hilo digital. La trazabilidad es un tema crítico aquí ya que los errores no solo se expresan como errores geométricos, sino como pérdidas financieras, siendo impresos y no cumpliendo sus funciones en su elaboración, por ejemplo, debido a posibilidades de pérdidas de movimiento, resultó muy rígido, causando fricción parasitaria, estrés térmico adicional en servomotores, vibración o puntos de atrapamiento durante la operación [13]. Por lo tanto, el diseño y la impresión se consideraron como partes funcionales: el control del diseño del actuador, el enrutamiento del cableado y el acceso al servicio, así como las condiciones para un ensamblaje repetible. En cuanto a la metodología, la teoría en el capítulo está dirigida a elaborar cómo la fabricabilidad es un elemento conectado del diseño digital en la fabricación, y cómo los diseños no deben divorciarse de las consideraciones funcionales de la fabricación, con la estética como el último objetivo de separación entre diseño y función.

2.2 Modelado paramétrico como base del diseño

El modelado paramétrico es una práctica para la representación de geometría mediante parámetros, relaciones y restricciones, lo cual es crucial cuando el componente de un sistema

consiste solo en partes interdependientes. En una carcasa que debe funcionar junto con otros mecanismos internos (ojos, mandíbula, cuello, soportes y estructura), el valor del uso del enfoque paramétrico está en la capacidad para cambios controlados: por ejemplo, cambiando espesores, ángulos, posiciones de montaje, etc., que pueden interferir en el ensamblaje, confirmando así el efecto de este proceso en el mundo real. Esto minimiza las inconsistencias dimensionales y ayuda a diseñar para un ciclo de repetibilidad más rápido y alto. El desarrollo de la carcasa no se limita al modelado de componentes separados, sino que también involucra la gestión del ensamblaje. A este nivel, el principio es diseñar de acuerdo con las restricciones funcionales: planos de referencia, ejes de rotación, superficies de soporte y holguras de movimiento. Una vez que el ensamblaje se retira del modelo de "unión de partes", se convierte en un sistema de interferencias validadas, rangos de movilidad y limitaciones físicas. En animatrónica, esta es una disciplina indispensable; pequeñas colisiones internas a menudo pueden terminar obstruyendo el camino, por lo que los servomotores, al realizar lo que están programados para hacer, necesitan consumir mucha corriente.

2.3 Autodesk Fusion 360 como plataforma integrada CAD/CAE/CAM

El diseño se desarrolló en la plataforma Autodesk Fusion 360 con las herramientas para modelado 3D, diseño industrial, CAM y análisis mecánico, etc. Esta fusión es especialmente beneficiosa en mecatrónica, ya que nos ayuda a mantener la consistencia de las geometrías antes del ensamblaje final y reducir los fragmentos de datos que comúnmente se encuentran cuando el desarrollo de software consiste en 3 aplicaciones diferentes y no se realiza un acceso trazable. La gestión de archivos y la sincronización también juegan roles importantes en permitir la continuidad digital en prototipos de alta iteración: eso es lo que se refieren como un modelo CAD, que se convierte en el núcleo del sistema, y que el diseño cambia basado en eso, de modo que las variaciones estén listas para ser fabricadas. En términos de un animatrónico destinado a la producción académica, esta gestión también tiene valor documental a través de un registro ordenado del proceso del prototipo que puede demostrar el proceso de ingeniería, ya que estos programas también tienen su propia historia.

2.4 Diferencia con modelado directo y justificación de la elección

El diseño paramétrico es una clase diferente en comparación con el modelado directo, debido a la naturaleza del árbol de historial. La geometría en el modelado directo se altera mediante la manipulación de vértices, bordes y caras a través de la manipulación de la geometría sin necesidad de parámetros y restricciones. Aunque las técnicas de modelado directo son útiles

para cambios instantáneos, para un sistema compuesto por bloques de construcción interdependientes como una carcasa, que necesita sincronizarse con mecanismos internos, alojamientos de actuadores, topes de movimiento, rutas de cables, puntos de fijación, etc., su uso exclusivo crea algunos problemas de coherencia global.

Por lo tanto, el diseño de la solución fue paramétrico, de modo que se preservó la resistencia geométrica y funcional del conjunto. Mantener la correlación explícita entre las partes mantiene la geometría cohesiva, incluso con los hallazgos de pruebas físicas (por ejemplo, correcciones para interferencias en movimientos de ojos/mandíbula, reequilibrio del soporte para minimizar la fricción o ajustes de grosor para hacerlo más rígido sin afectar la estética). La solución sirve para asegurar la continuidad del flujo digital, lo que también protege contra las etapas futuras (impresión, ensamblaje y validación) de heredar inconsistencias.

2.5 Diseño Basado en Características y relación diseño–manufactura

Como complemento al marco conceptual, el prototipo es consistente con el Diseño Basado en Características (FBD), que generalmente se considera una teoría para vincular la intención de diseño con la fabricabilidad. De esta manera, una característica puede considerarse información geométrica relevante para un proceso o propósito funcional: alojamientos para servomotores, agujeros para tornillos, superficies de soporte, guías, ranuras, chaflanes de inserción, topes finales y zonas de refuerzo, entre otras características. Estas no solo son características definidas como forma, sino como elecciones de ingeniería que definen cómo se ensambla, cuida, desmonta un componente y cómo se permite la tolerancia. Esto se aplica a un animatrónico utilizado en exhibición y publicidad académica por dos razones esenciales: el prototipo debe ser mantenible o resistente al uso repetido, y debe facilitar el desmontaje parcial y el reemplazo de piezas en caso de fallos, lo que significa que los tornillos, soportes y guías deben ser partes funcionales explícitas, no soluciones improvisadas. Esto significa que la carcasa no se considera una "cáscara estética", sino como un dispositivo con requisitos de ensamblaje, seguridad y operación.

2.6 Procesamiento de geometría y edición de mallas para integración estética–funcional

Las técnicas de procesamiento de geometría y edición de mallas poligonales se integraron en el flujo de trabajo, ya que la carcasa del personaje y algunos elementos externos dependen de geometrías trianguladas. Esta parte de los datos se encuentra entre las matemáticas aplicadas, la informática y la ingeniería y explora los algoritmos para obtener, reconstruir, analizar y manipular representaciones de modelos 3D complejos. Las mallas trianguladas son críticas en

la fabricación aditiva (principalmente en formato STL/OBJ) pero pueden ser difíciles de manejar si deben convertirse en componentes listos para ensamblar con elementos mecánicos incorporados [14]. En este trabajo, el procesamiento de mallas proporcionó una solución a dos requisitos contradictorios: la reconocibilidad estética del personaje y la coherencia mecánica con el sistema animatrónico (ojos, mandíbula, soporte estructural y cableado). Los procesos comunes son el suavizado de superficies, la reparación de mallas (por ejemplo, correcciones normales, eliminación de auto-intersecciones, cierre de agujeros), deformaciones controladas y ajustes locales para permitir un ensamblaje realista (grosos mínimos imprimibles, tolerancias de ajuste, pasos de tornillos, ventanas de mantenimiento y espacios para rutas de cables). La geometría estética resultante permite desarrollar una carcasa fabricable con respecto a los aspectos de ingeniería, pero no a expensas de la presentación de la forma.

2.7 Diseño morfológico basado en personaje y criterios de escala

Con el diseño morfológico, la selección de un personaje y el análisis de características predominantes son los primeros pasos para ayudar a asegurar que podamos reconocer ese personaje. La elección de la Tortuga Ninja está justificada tanto por la identidad cultural como por las características faciales que tienen problemas reales de integración: un hocico voluminoso, ojos grandes y una boca ancha que deben resolverse, por lo que se requiere una gran cantidad de espacio interno para que los mecanismos se implementen sin deformar la estética. En CAD, el concepto comprende primero establecer los volúmenes principales y las proporciones globales, luego la definición de superficies, pero teniendo en cuenta ser consistente con la escala y el espacio para los actuadores y sus componentes estructurales y de cableado. Como modelo para la difusión académica del personaje en un prototipo, este personaje también cumple un papel pedagógico: el público en general puede interpretar inmediatamente el objetivo del proyecto y puede ilustrar visual y atractivamente los principios de mecatrónica, control, diseño mecánico y conceptos de fabricación aditiva de actuación y accesorios [15].

2.7.1 Diseño de ojos, estructura

Inicialmente, se dibujó la estructura orbital y los ojos del personaje. Ahora se definieron las cavidades para los ojos y se dio el contorno básico de los párpados y el contorno ocular. Esta precisión era importante porque había que ser muy preciso en esta etapa: los globos oculares y sus soportes necesitarían encajar correctamente más tarde en el ensamblaje. Se seleccionó un diseño modular para separar los ojos como piezas propias para permitir su propia impresión y

luego ensamblaje en las cavidades. Se realizó una estructura que por sí sola pueda tener todos los servomotores requeridos en el movimiento ocular. La configuración de 4 servomotores MG90S para los párpados y dos para los movimientos de los ojos arriba/abajo e izquierda/derecha requirió la construcción del marco de acuerdo con la conceptualización de esos servomotores y su tamaño.

El sistema en los párpados es un mecanismo de enlace de cuatro barras (o paralelogramo articulado), mostrado en las Figuras 1 y 2. Funciona por párpado, con cada servomotor sirviendo como el conductor que guía una serie de enlaces que convierten una rotación angular en un desplazamiento circular en el lado opuesto, en el párpado. Al estar conectados en paralelo, fuerzan al párpado a seguir una trayectoria circular exacta, al igual que un ojo real, para hacer que el parpadeo sea rápido y fluido.

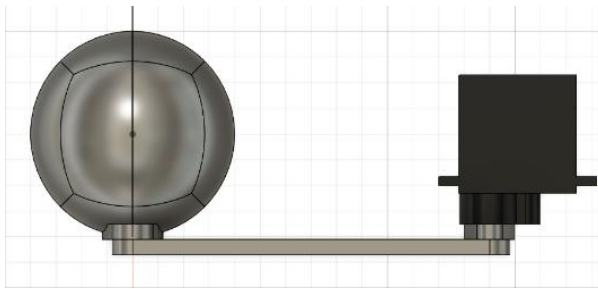


Figura 1. Vista lateral del ensamblaje ocular.

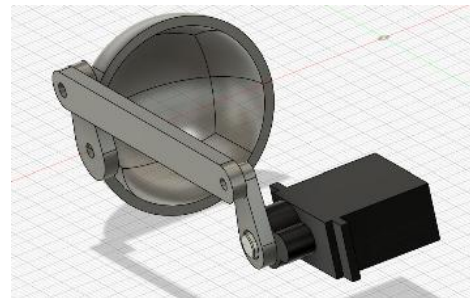


Figura 2. Renderizado isométrico del módulo ocular completo con servomotores integrados

En esta parte del documento, y a lo largo de la finalización del proyecto, se utilizó la aplicación "Symmetry" en Fusion 360 como material de apoyo para gestionar el tiempo de modelado. Su uso facilitó la minimización de iteraciones de diseño mediante la reproducción de geometrías similares para obtener apariencias gráficas consistentes y componentes uniformes. Este resultado se puede ver en las Figuras 3 y 4.

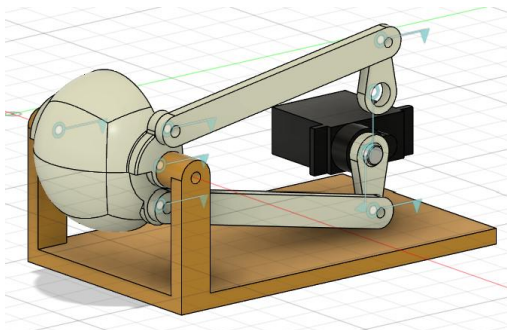


Figura 3. Simulación de movimiento de párpados de un ojo.

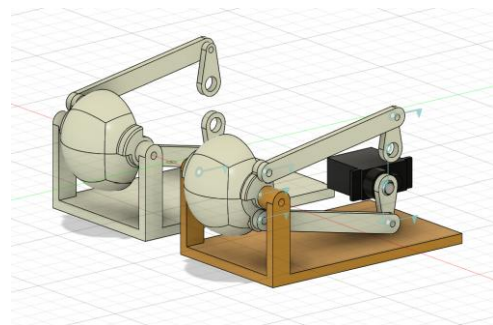


Figura 4. Simulación de movimiento de párpados de dos ojos.

La Figura 5 proporciona el modelado de los servomotores MG90S que se utilizaron para los párpados del proyecto. Con estos modelos dentro del marco de CAD, se anticipó el límite de espacio, las áreas de montaje y las interferencias antes de la producción de las piezas (Figura 6), y luego se ensamblaron todas las piezas necesarias. Las áreas de montaje fueron importantes para el diseño de la estructura en las etapas posteriores.

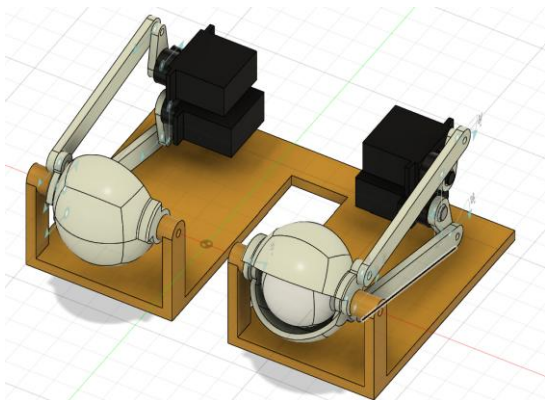


Figura 5. *Ensamble del sistema ocular dual montado sobre la base estructural principal.*



Figura 6. *Todas las piezas del sistema ocular.*

En las Figuras 7, 8 y 9 se evidencia la correspondencia entre el modelo CAD y el ensamblaje físico, observándose que las dimensiones y geometrías definidas en el diseño se ajustaron de manera adecuada en la fabricación. El acople entre piezas calzó correctamente, lo que confirma la coherencia dimensional del modelado y valida la precisión del diseño para su implementación en condiciones reales.

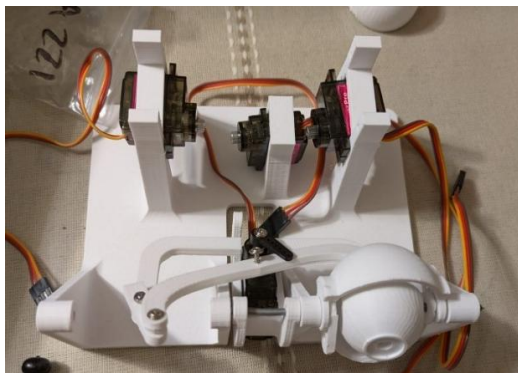


Figura 7. *Ensamblaje del sistema ocular, visión superior*

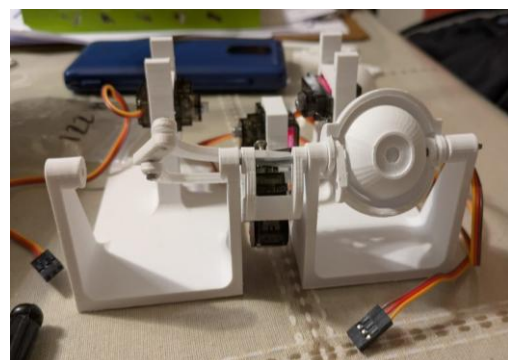


Figura 8. *Ensamblaje del sistema ocular, visión frontal*

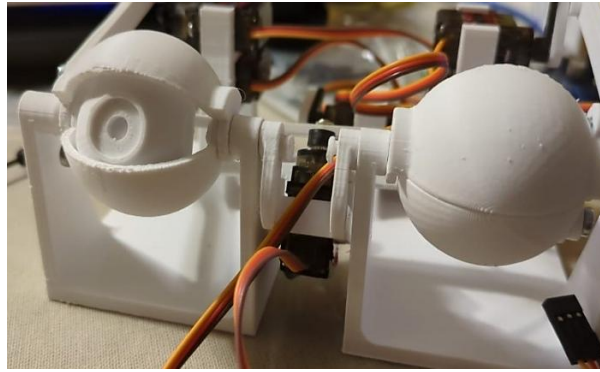


Figura 9. *Ensamblaje ojos.*

2.7.2 Diseño estructura interna

Se definieron características fundamentales de acuerdo con el sistema de soporte interno, con respecto a la Figura 10. Los componentes fueron construidos para conectarse en la cabeza para sostener y soportar la mandíbula y albergar los servomotores necesarios. El diseño del modelo se organizó en piezas en componentes funcionales y de fabricación. La Figura 11 ilustra la mandíbula inferior como una pieza móvil separada y con un tornillo a cada lado para sostener dicha mandíbula, que es capaz de articularse para asemejar el habla.

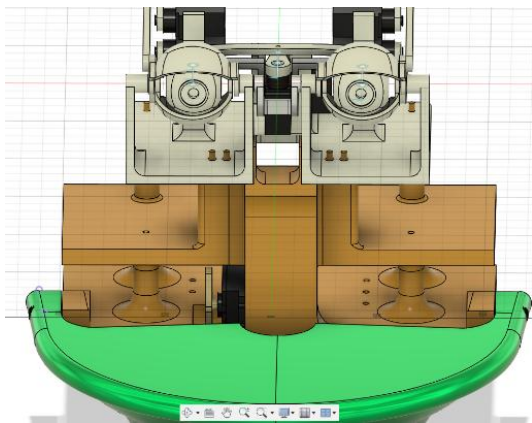


Figura 10. *Vista en corte de la estructura interna preliminar con alojamiento para mandíbula.*

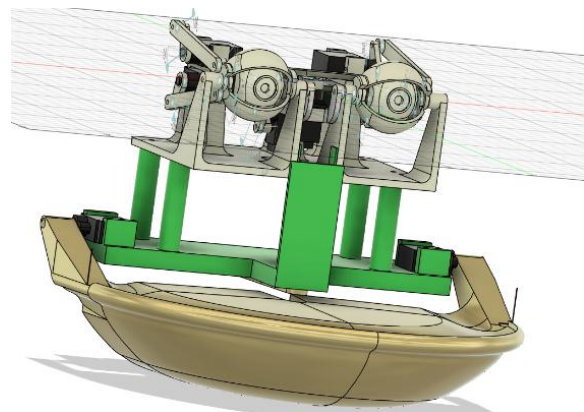


Figura 11. *Detalle del anclaje para el servomotor de la mandíbula (MG996R) dentro del chasis.*

Las Figuras 12 y 13 ilustran el diseño y trabajo de ingeniería del servomotor MG996R y el seccionamiento del espacio para el dispositivo. La estructura completa del modelo fue alterada ya que se aplicó un servomotor de alto torque detrás de la boca en acción para mover la mandíbula. El prototipo interno: ese marco central fue originalmente destinado a alojar un

microcontrolador o una microcomputadora, luego fue diseñado para que pasen cables por ahí, sin interferir al MG996R, y un par de tornillos para la mandíbula.

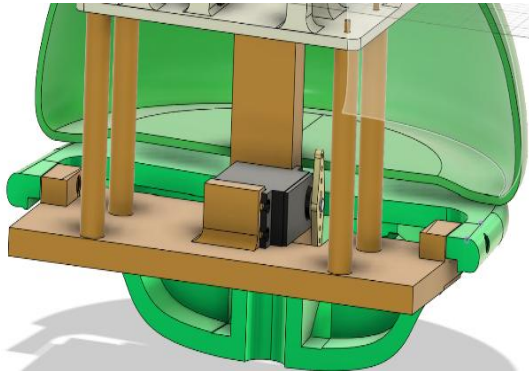


Figura 12. *Renderizado de la estructura interna optimizada para flujo de aire y paso de cables.*

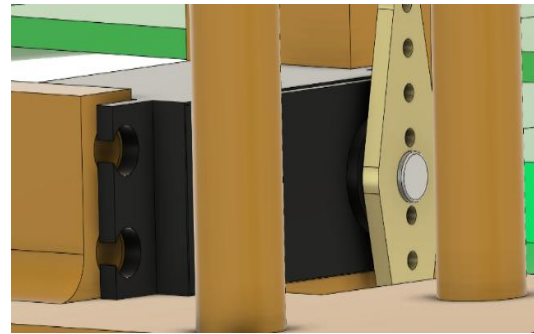


Figura 13. *Análisis de la sección transversal de la carcasa mostrando donde va el servomotor MG996R.*

La Estructura interna, como se muestra en la Figura 14, diseñada en Fusion 360, proporciona soportes para la disposición basada en los ojos. El elemento clave (que la imagen enfatiza) fue diseñado inicialmente para albergar los controles electrónicos (es decir, inicialmente un microcontrolador como Arduino). Durante el diseño, se identificó la necesidad de asegurar la estructura de la carcasa por razones de rigidez y estética. Se observó que, al considerar un Arduino, el tamaño de la Raspberry Pi con la carcasa y el ventilador no se tuvo en cuenta, y no calzaba, lo que resultó en un tamaño final demasiado pequeño por parte de la estructura. Tampoco se incluyeron espacios en el interior para acceder a los cables USB conectados a la Raspberry Pi, que se utilizan para conectar el micrófono, altavoces, teclado y mouse. Esta configuración no permitió que la integración electrónica se llevara a cabo completamente; sin embargo, hay suficiente espacio en la cabeza y en el marco para otras partes, como los cables de los seis servomotores MG90S de los párpados y los ojos, que se mueven a través de la estructura circular sin obstruir el servomotor, ya que está ubicado en el "primer piso" de la estructura, mientras que los cables atraviesan el segundo nivel. La principal conclusión fue planificar la disposición general de la electrónica en la parte mecánica, asegurando suficiente espacio para acomodar cambios. La decisión de colocar los controladores más grandes en la parte inferior del sistema desde el principio resultó ser clave, permitiendo la inclusión de

elementos ligeros o que no alberguen mucho espacio como dichos cables de los seis servomotores, más el servomotor pesado MG996R en la configuración.

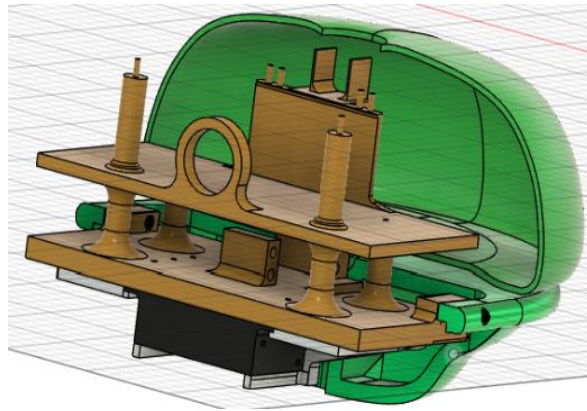


Figura 14. *Ensamble digital de la estructura interna con Mandíbula y hocico.*

Morfológicamente, el objetivo fue mantener la carcasa externa estéticamente cercana al personaje seleccionado, con una representación estilística del mismo en mente. Sin embargo, las formas internas del cuerpo debían minimizarse para facilitar la impresión 3D y el ensamblaje de componentes. Las piezas impresas se dividieron en un diseño modular, procesándose la cabeza en partes o módulos (por ejemplo, cráneo, hocico y mandíbula). Se consideró la mejor posición para la impresión de piezas individuales y si había un requisito de acceso al interior del espacio en etapas posteriores. Esta decisión no solo respondió a las limitaciones de volumen de la impresora 3D, sino que también se diseñó un plan para restringir los soportes durante la impresión e integrar partes internas para facilitar el ensamblaje. El seccionamiento se ejemplifica en la variada paleta de filamentos presentada en las Figuras 15 y 16, mediante los diferentes colores de filamento con los cuales se imprimieron las piezas.

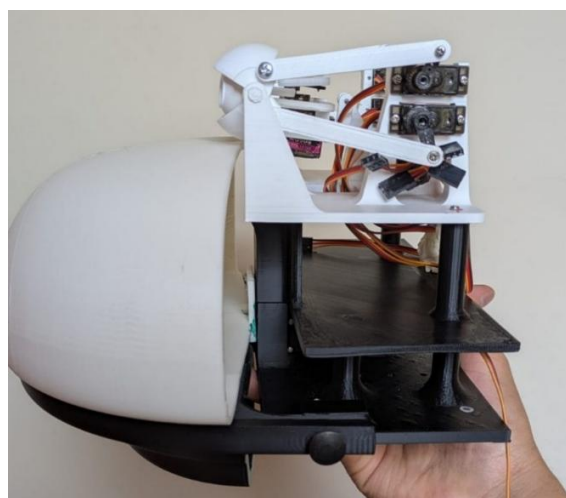


Figura 15. *Vista izquierda del ensamblaje de estructura interna + sistema ocular + mandíbula.*

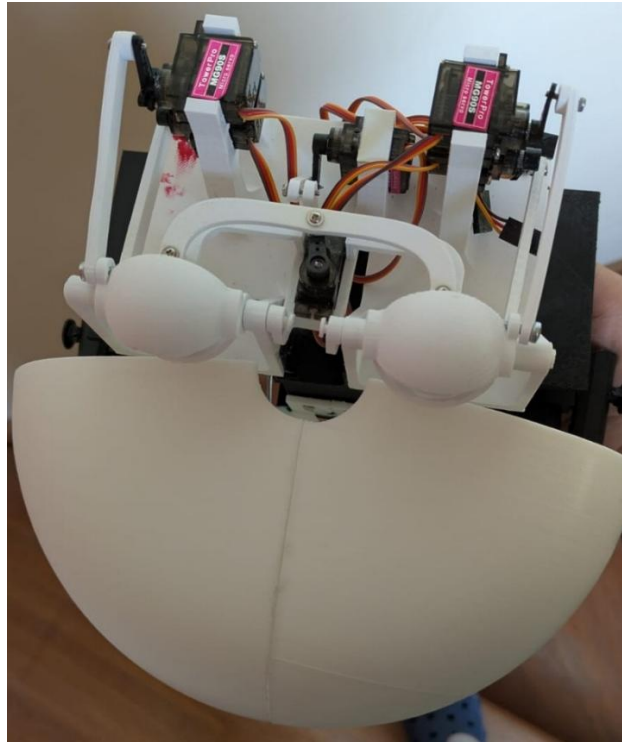


Figura 16. *Vista superior del ensamblaje de estructura interna + sistema ocular +mandibula.*

2.7.3 Diseño del mecanismo tipo gimbal (cardán) para el cuello

El diseño del cuello requería que este permitiera tanto la orientación suave de la cabeza como que la estructura mantuviera integridad estructural para soportarla. Para este propósito, y considerando los dos ejes ortogonales que son aproximadamente 90° y están definidos por tornillos, se aplican dos grados de libertad de rotación: cabeceo y balanceo (se utiliza un arreglo de cardán o junta universal). Las Figuras 17 y 18 indican un retorno a lo que se deseaba desde el principio (o sus ejes ortogonales a través de tornillos).

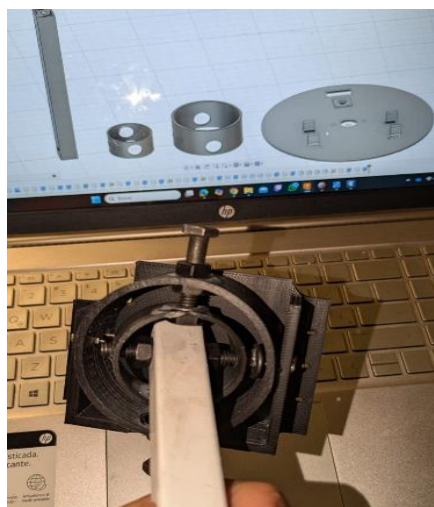


Figura 17. *Articulación universal del Gimbal.*



Figura 18. *Cuerpos impresos usados en la realización del Gimbal.*

En términos cinemáticos, la orientación de la plataforma interna puede describirse como una composición de rotaciones alrededor de dos ejes, por ejemplo:

$$\mathbf{R} = \mathbf{R}_{x(\varphi)} \cdot \mathbf{R}_{y(\theta)} \quad (1)$$

Donde φ y θ indican los ángulos de balanceo y cabeceo, respectivamente [18]. El cardán a su vez fue incluido en nuestro prototipo como un componente de desacoplamiento mecánico y el componente de rotación horizontal (yaw) se resolvió a través del mecanismo actuador del cuello (servomotores asignados para el movimiento lateral y elevación), evitando aplicar cargas excéntricas directas en un solo punto de soporte. Para dimensionar el actuador del cuello e investigar arreglos de transmisión, se tuvo en cuenta el momento estático relacionado con el peso de la cabeza con respecto al eje de rotación:

$$\tau_{req} \approx m \cdot g \cdot d \quad (2)$$

donde m es la masa equivalente del conjunto móvil, g la aceleración gravitacional y d la distancia desde el eje de rotación hasta el centro de masa. El criterio de selección se expresa como:

$$\tau_{servo} \geq S \cdot \tau_{req} \quad (3)$$

Donde S es el factor de seguridad. En uno de los esquemas propuestos, se asume que se construye un tren de engranajes acoplado a dos servomotores MG90S a lo largo de los extremos de la estructura para aumentar el par disponible en los dos lados. Sin embargo, más tarde se descartó debido a las limitaciones significativas encontradas por el sistema de prueba en condiciones de carga real, discutidas en la Sección 2.11.

Dado que el mecanismo gimbal o cardán aporta dos grados de libertad, se diseñó de forma separada la estructura que sostiene el módulo ocular y la mandíbula, principalmente por motivos de impresión, ensamblaje y validación por etapas. En la Figura 19 y 20 se observan los cuerpos y el diseño conceptual usado en la realización del mismo. Se realizaron aproximadamente tres iteraciones del diseño del cuello, manteniendo el gimbal como base cinemática y variando la estrategia de actuadores y transmisión.

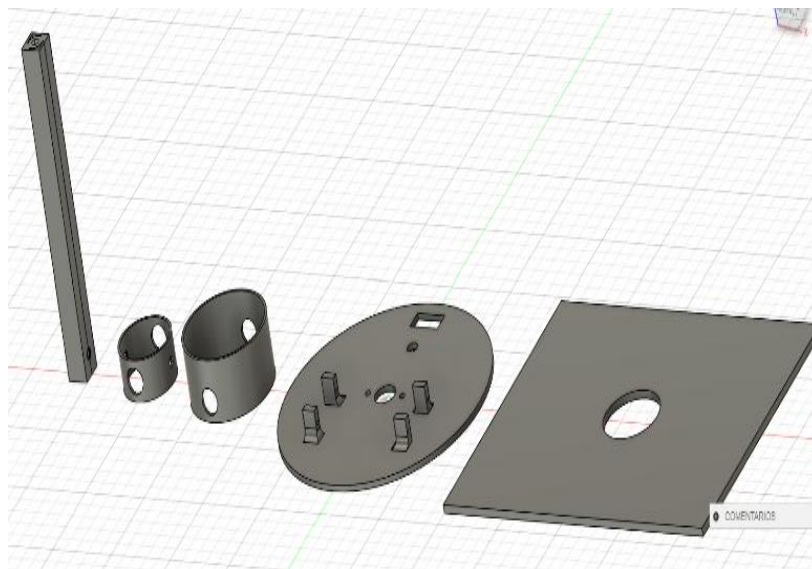


Figura 19. *Cuerpos usados en la construcción del Gimbal.*

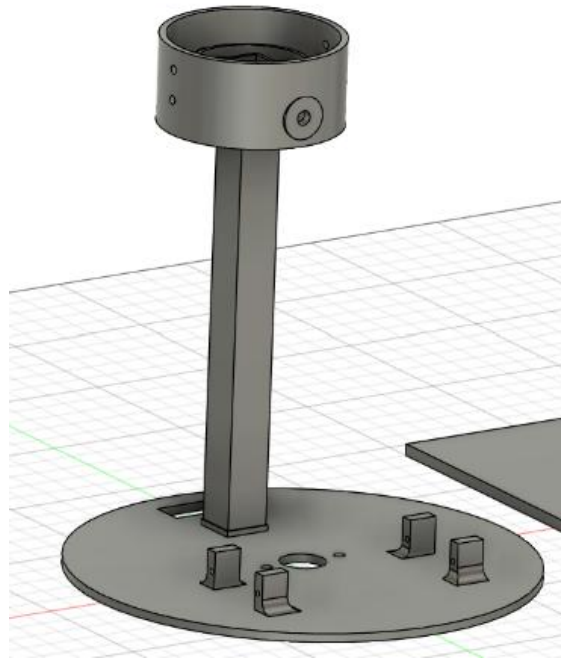


Figura 20. *Diseño conceptual del mecanismo Gimbal para el cuello con 2 grados de libertad.*

2.7.4. Eje adicional (yaw) mediante unión rotacional tipo "turntable" y accionamiento servo-perimetral

Se presenta un eje adicional (yaw) a través de una junta rotacional tipo "plato giratorio" y un accionamiento servo-periférico. Dado que el prototipo tiene como objetivo mejorar la expresividad del personaje sin imponer una arquitectura humanoide de múltiples ejes demasiado complicada, se integra un cierto grado de libertad de (yaw/pan), lo que permite que todo el conjunto de la cabeza se oriente con respecto a la base. Este grado de libertad fue concebido como un subsistema independiente del mecanismo de inclinación lateral del cuello; ambos movimientos pueden visualizarse durante una condición de trabajo (yaw + inclinación), aunque representan cadenas mecánicas diferentes con funciones distintas.

Aquí es cuando el motor del disco duro HDD por fin actúa como un actuador mecánico, al estar dividido por partes, se quedará la base del motor del disco duro pegada a la base fija, tiene un alojamiento redondo en su diseño, hecho para eso específicamente, del otro lado, al final de su estructura se pegará al disco principal, esto hará que opere sin control electrónico, poseyendo ya la capacidad de girar. Como componente mecánico, enlaza y rota la base fija al disco superior, proporcionando concentricidad al sistema y movilidad angular al sistema de yaw.

2.7.5. Estructura apilada y ruta de carga (base–disco actuado–plataforma superior)

La estructura se presenta de manera apilada y sigue un camino de carga (base fija – HDD – Disco principal – Disco estabilizador –plataforma superior). El conjunto del yaw se resuelve mediante una estructura apilada, en la que el componente clave es el disco principal, el cual integra simultáneamente: un enlace al HDD, un alojamiento para el MG90S (agujero rectangular), y una fijación al tubo de acero mediante tornillos (agujero circular). Funcionalmente, la secuencia se presenta como sigue:

1. **Base fija:** pieza inferior diseñada para estabilidad y para alojar el HDD como componente central del conjunto.

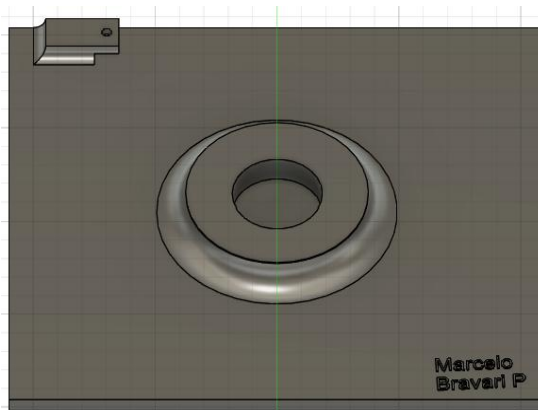


Figura 21. Base fija Fusion360

2. **Disco principal rotante (disco de acople + disco actuado):** pieza circular acoplada al HDD, que rota respecto a la base. En este disco se ubicó el MG90S y el punto de fijación del tornillo que asegura el caño de acero.

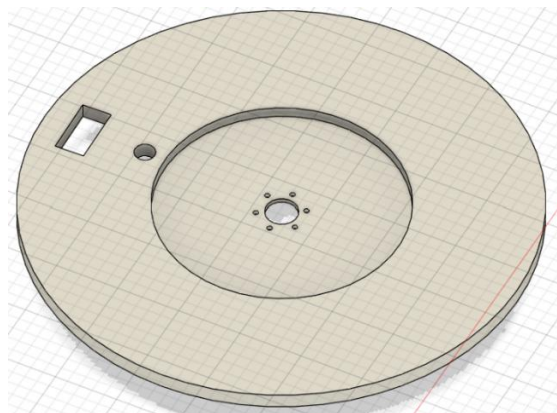


Figura 22. Disco principal rotante Fusion 360.

3. **Caño de acero:** elemento estructural vertical fijado mediante tornillo al disco principal; por diseño, el caño queda solidario al disco rotante, es decir, gira con el yaw y aporta rigidez global al conjunto.

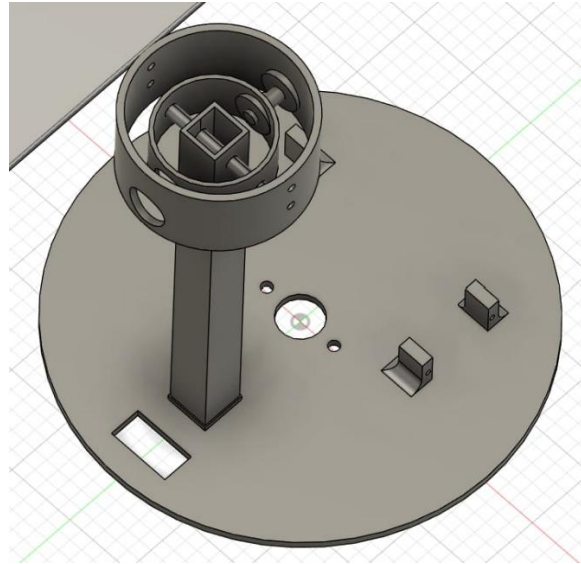


Figura 23. *Diseño del caño de acero a implementar.*

4. **Plataforma superior del cuello:** montada sobre el disco principal, constituye la interfaz para instalar el mecanismo de inclinación lateral previamente desarrollado.

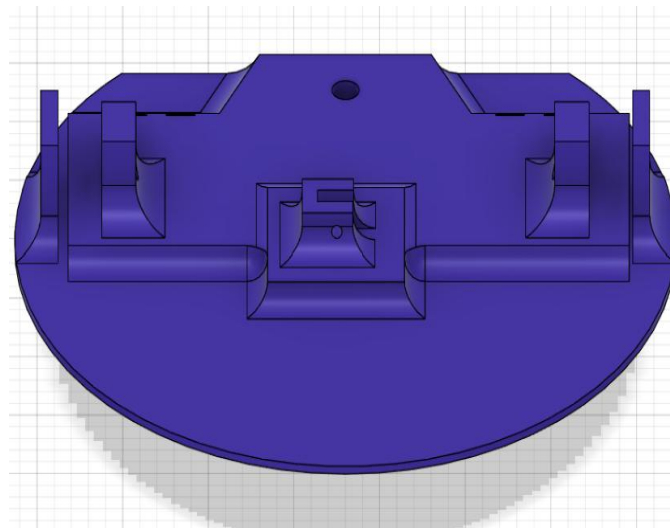


Figura 24. *Plataforma superior del cuello Fusion 360.*

5. **Disco estabilizador adicional (anillo verde):** incorporado como elemento de estabilidad, aumentando el área de apoyo y reduciendo la tendencia a vuelco cuando el conjunto opera con carga excéntrica o con movimientos repetitivos.

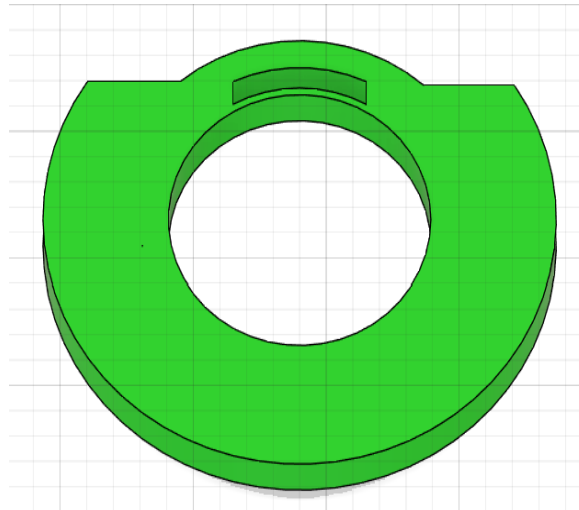


Figura 25. Disco estabilizador Fusion360.

Con esta disposición, el yaw se convierte en la “base cinemática” del sistema: la plataforma que sostiene el cuello y la cabeza reorienta su dirección global mediante yaw, y sobre ella se superpone la inclinación lateral generada por el mecanismo del cuello.

En la Figura 28 se muestra la implementación de estos cuerpos unidos en un diseño completo en Fusion 360, mientras que en las Figuras 26 y 27 se mira la incorporación de 3 cuerpos en uno, en cada uno, para llegar al final al ensamblaje completo que se muestra en la Figura 28.



Figura 26. Base fija + Disco estabilizador + HDD



Figura 27. Vista superior del disco principal rotante + MG90S+ HDD.

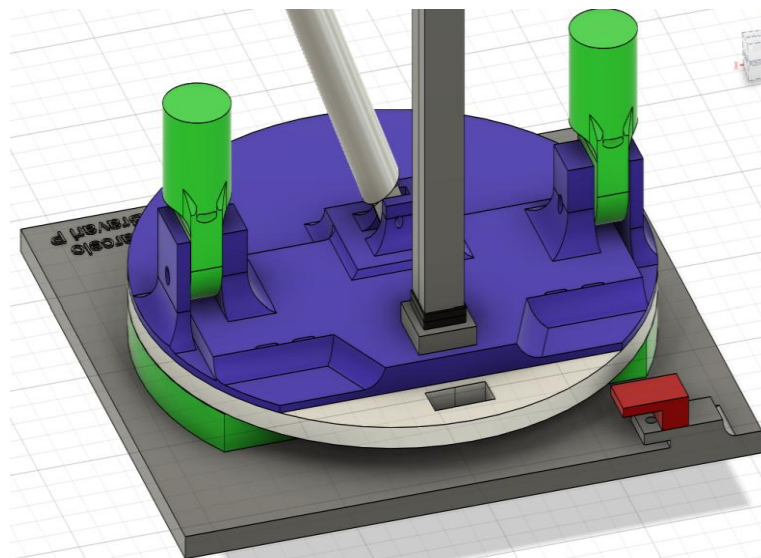


Figura 28. Diseño del conjunto completo Fusion 360.

2.7.6. Actuación del yaw: servomotor MG90S y ventaja geométrica por accionamiento en el filo

El movimiento yaw no lo genera el HDD, el motor del disco duro se puede decir que actúa más como un eje, pero el que genera el yaw es un servomotor MG90S montado en el disco principal (agujero rectangular). La decisión de ubicar este servo en el filo/perímetro del disco responde a un criterio de ventaja mecánica geométrica: al aplicar una fuerza tangencial lejos del centro, el par alrededor del eje de yaw aumenta proporcionalmente con el radio efectivo. En otras

palabras, aun tratándose de un servomotor de torque moderado, su ubicación periférica favorece la capacidad de rotación del conjunto al aprovechar un brazo de palanca mayor.

En cuanto a la integración con el cuello, se establece que el yaw fue diseñado como un subsistema independiente del cuello, sin alterar la cinemática interna del mecanismo de inclinación lateral. No obstante, promueve el uso de una combinación de movimientos que permite realizar acciones más expresivas, como la rotación lateral y la inclinación. Esta combinación mejora la comunicabilidad del personaje sin introducir múltiples ejes adicionales en el cuello, manteniendo así un equilibrio entre la expresividad del prototipo, la complejidad constructiva y el mantenimiento.

2.7.7. Diseño y dimensionamiento del mecanismo de cuello

El diseño del cuello para la cabeza animatrónica no se inspira en la robótica industrial en general, sino que sigue una lógica de posición orientacional expresiva. En lugar de crear grandes volúmenes de trabajo o trayectorias complejas, el propósito del mecanismo es producir un movimiento realista con un sistema más pequeño, imprimible y robusto, lo que permite un mayor grado de repetibilidad e integración estética en el cuerpo del personaje. La inclinación lateral (roll/inclinación) se ilustra en la Fig. 29, seguida de la rotación de yaw/pan del cuello, que resulta suficiente para comunicar señales no verbales como curiosidad, énfasis y duda, sin incrementar de manera exponencial la complejidad del sistema.

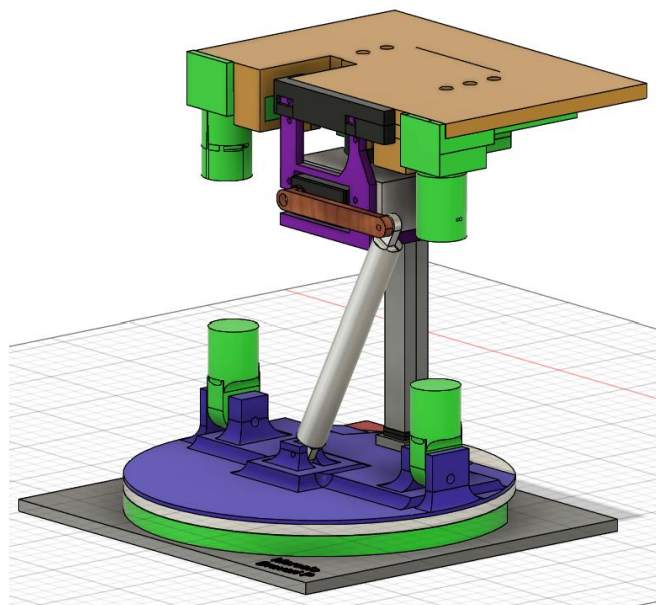


Figura 29. *Diseño adoptado para el cuello de la cabeza animatrónica.*

2.7.8. Arquitectura real implementada: transmisión servo–horn–biela

En el prototipo desarrollado, el “eslabón” no corresponde a un actuador lineal dedicado (husillo o cilindro), sino a una transmisión servo–horn–biela, donde el servomotor MG995 entrega un par τ_s a un brazo (horn), y este par se transforma en una fuerza F aplicada sobre la biela mediante una unión articulada. La biela transmite dicha fuerza al puntal central, generando un momento respecto al eje de inclinación del cuello y produciendo la inclinación lateral (roll/tilt) observada.

Bajo un modelo estático simplificado, la conversión de par a fuerza se expresa mediante la relación de momento:

$$\tau_s = r_{\perp} F \Rightarrow F = \frac{\tau_s}{r_{\perp}} \quad (7)$$

donde r_{\perp} es el brazo efectivo perpendicular entre el eje del servo y la línea de acción de la biela.

Considerando la longitud del horn medida en CAD como $l = 73.009 \text{ mm} = 0.073009 \text{ m}$, (esto se puede ver en la Figura 30) el brazo efectivo se modela como:

$$r_{\perp} = l \sin(\varphi) \quad (8)$$

siendo φ el ángulo entre el horn y la dirección de la biela.

En este proyecto se verificó experimentalmente que, durante el modo de inclinación lateral, el ángulo φ se mantiene aproximadamente constante alrededor de $\varphi \approx 61.9^\circ$ (con variaciones pequeñas), como se aprecia en las Figuras 31 y 32. Debido a que la cinemática del conjunto no busca “subir/bajar” la cabeza (pitch) sino inclinarla lateralmente (roll), mientras que el movimiento de giro izquierda–derecha corresponde a un servomotor MG90S adicional (yaw) en la base, que rota la estructura sin alterar significativamente la geometría relativa horn–biela [20]. Con $\varphi = 61.9^\circ$, se obtiene:

$$r_{\perp} = 0.073009 \text{ m} * \sin(61.9^\circ) \approx 0.06444 \text{ m} \quad (9)$$

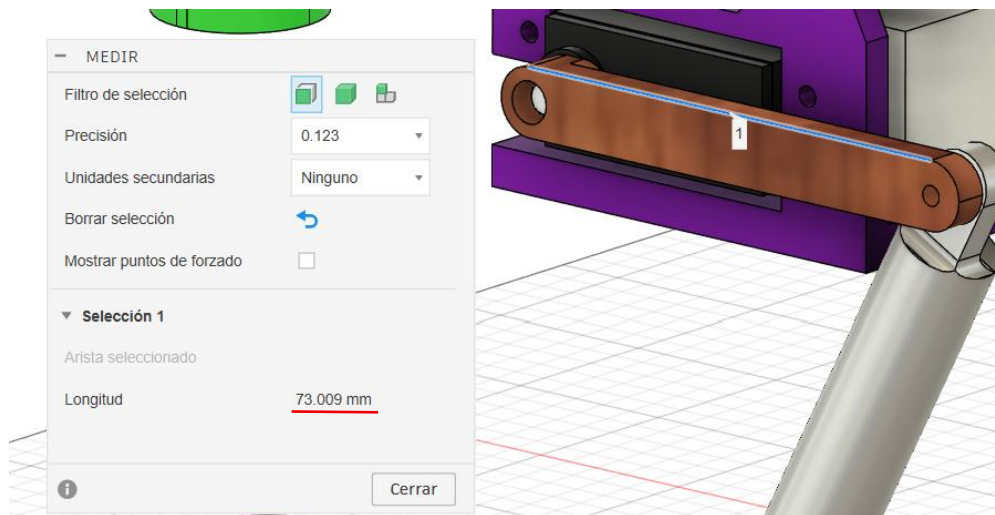


Figura 30. Distancia eje-orificio del brazo de palanca efectivo del servo.

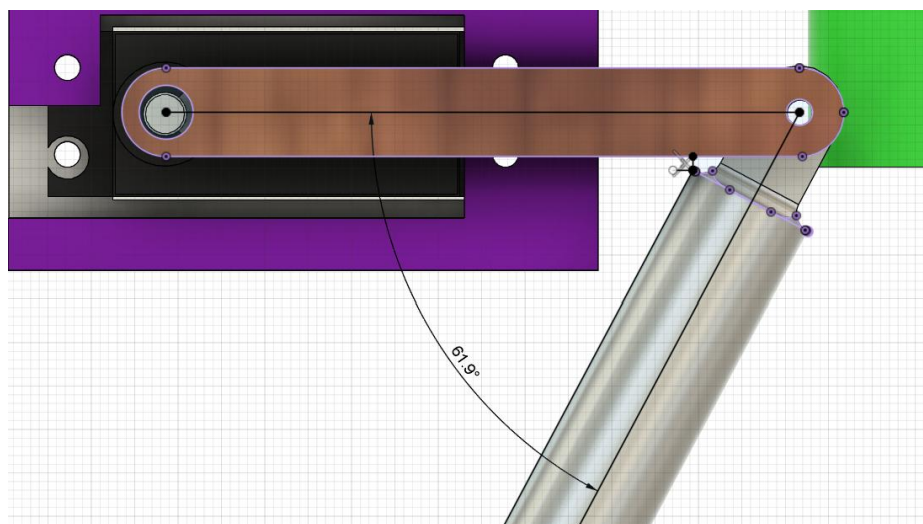


Figura 31. Ángulo entre el horn y la dirección de la biela.

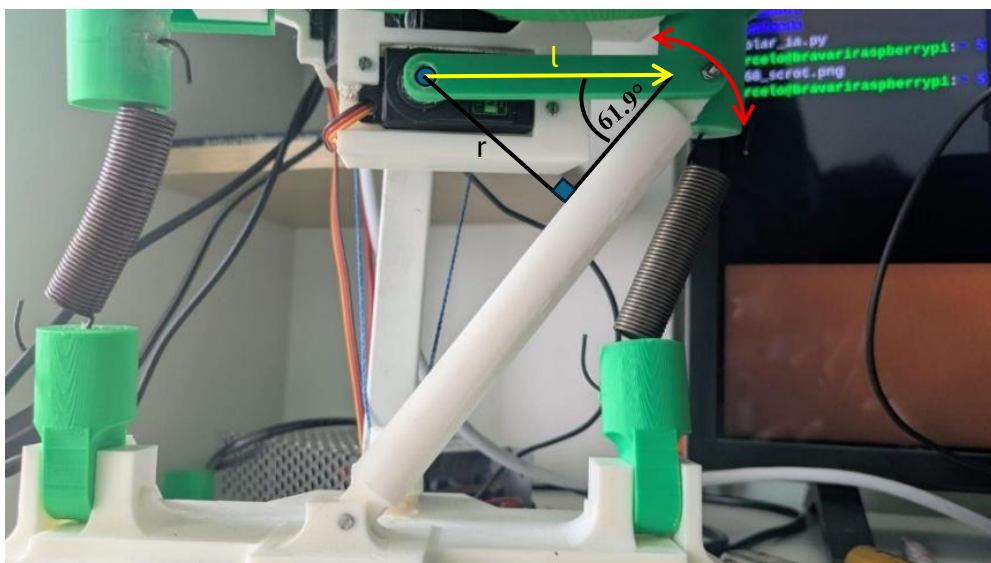


Figura 32. Diagrama "brazo de palanca"

El par nominal del MG995 se expresa frecuentemente en kgf\cm, por lo que se convierte a:

$$\tau_s[\text{N}\cdot\text{m}] = \tau_s[\text{kgf}\cdot\text{cm}] (9.80665) (0.01)$$

Para dos condiciones de operación (asociadas a niveles de alimentación), colocando sus máximos y mínimos de tensión, se obtiene:

$$\tau_{s(4.8V)}(9.4 \text{ kgf}\cdot\text{cm}) \approx 0.922 \text{ N}\cdot\text{m} \qquad \tau_{s(5.5V)}(13 \text{ kgf}\cdot\text{cm}) \approx 1.275 \text{ N}\cdot\text{m}$$

Y, por tanto, la fuerza transmitida por la biela queda estimada como:

$$F \approx \frac{0.922 \text{ N}\cdot\text{m}}{0.06444 \text{ m}} \approx 14.30 \text{ N} \qquad F \approx \frac{1.275 \text{ N}\cdot\text{m}}{0.06444 \text{ m}} \approx 19.78 \text{ N}$$

Esta formulación permite justificar el diseño desde el dimensionamiento: la capacidad efectiva del sistema depende no solo del servo, sino también de la ventaja mecánica introducida por r_{\perp} y por el brazo equivalente desde el punto de aplicación hasta el eje de inclinación del cuello r_p . [21] Que transforma fuerza en momento útil sobre el eje:

$$M_{\text{cuello}} \approx F r_p \tag{10}$$

2.7.9. Momento gravitacional del conjunto y obtención del centro de masa.

El principal esfuerzo resistivo durante la inclinación lateral proviene del momento gravitacional del conjunto móvil (cabeza y mecanismos solidarios). Dicho momento se modela como:

$$M_g(\beta) = m g d \sin(\beta) \tag{11}$$

donde m es la masa del conjunto móvil, g es la gravedad, d es la distancia desde el eje/pivote de inclinación hasta el centro de masa, y β es el ángulo real de inclinación lateral. Usando la herramienta de centro de masa y la función de medición geométrica, el parámetro d fue evaluado en Autodesk Fusion 360. El d fue definido como el brazo de palanca geométrico del

eje real de rotación del grado de libertad "roll" (la inclinación lateral) y el centro de masa (CM) en el conjunto de inclinación. [22] La elección corresponde a la formulación clásica del torque por peso, el momento con respecto a un eje se obtiene de la distancia mínima (perpendicular) entre el eje y la línea de acción del peso aplicado al CM. Como resultado, d no se midió desde el servomotor actuador ni desde los elementos de transmisión (cuerno/varilla) ya que ambos componentes no forman el eje cinemático alrededor del cual el conjunto rota durante la inclinación lateral. En el mecanismo implementado, el eje de rotación está definido por el tornillo-pasador del cardán, que es una bisagra (unión rotativa) y permite el movimiento de rotación alrededor de una línea fija [23].

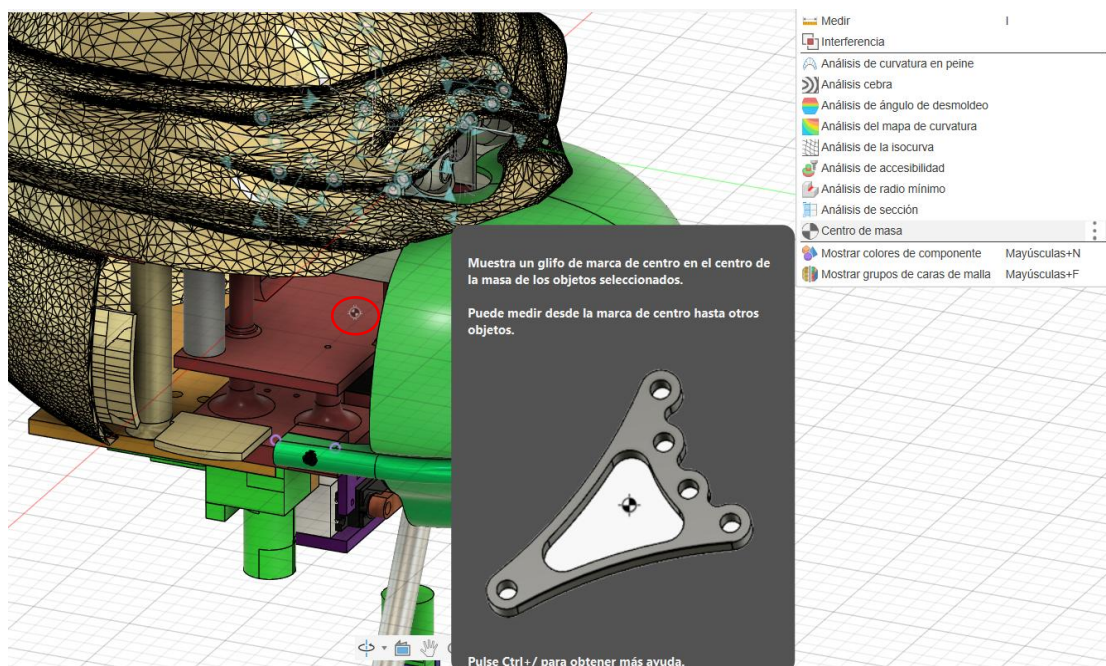


Figura 33. Centro de masa en Fusion360.

Para materializar este criterio en el entorno CAD (Autodesk Fusion), se procedió de la siguiente manera: Se calculó el centro de masa del subensamble móvil mediante la herramienta *Inspect* → *Center of Mass*, como se ve en la Figura 33, obteniendo el marcador del CM del conjunto; se identificó el centro geométrico del eje del tornillo del gimbal como referencia del pivote real; y con el fin de evitar ambigüedades al seleccionar aristas externas del cilindro (que pueden devolver distancias a bordes y no al eje), se incorporó en el modelo un orificio guía de 0.2 mm de diámetro ubicado en el centro del alojamiento del tornillo. Este orificio no tiene función estructural, sino que opera como marcador interno del eje, permitiendo seleccionar de forma inequívoca el punto central asociado al pivote.

Finalmente, se aplicó la herramienta *Inspect* → *Measure* para obtener la distancia mínima entre el marcador del CM y el punto representativo del eje del gimbal (ubicado por el orificio guía), adoptando dicho valor como d . Con esta metodología, la distancia empleada en los cálculos queda explícitamente vinculada al eje real de rotación del roll, garantizando coherencia física entre la geometría del mecanismo y el modelado estático utilizado para estimar el esfuerzo requerido al actuador, obteniéndose:

$$d = 82.279 \text{ mm} = 0.082279 \text{ m}$$



Figura 34. Distancia desde el eje/pivote de inclinación hasta el centro de masa.

Considerando una masa aproximada del conjunto móvil $m = 3 \text{ kg}$ y una inclinación lateral máxima de $\beta = 26^\circ$, esto del resultado de la configuración de referencia de la Fig.34 ($\beta = 0^\circ$). Siendo que la cabeza está dispuesta en forma horizontal. Entonces la designación de 26° viene de la variación de consigna del servomotor entre los ángulos programados $\alpha_{\text{prog}} = 89^\circ$, y $\alpha_{\text{prog}} = 115^\circ$ corresponde a $\Delta\alpha_{\text{prog}} = 26^\circ$, y siendo que el ángulo entre el horn y la dirección de la biela no varía, la cabeza es la que se inclina dichos 26° . Entonces el momento gravitacional resulta:

$$M_g(26^\circ) = (3)(9.81)(0.082279)\sin(26^\circ) \approx 1.061 \text{ N}\cdot\text{m}$$

El signo del momento depende de la convención del sentido positivo o negativo de giro; para fines de dimensionamiento se utiliza la magnitud $|M_g| \approx 1.061 \text{ N}\cdot\text{m}$, como referencia del esfuerzo que el sistema debe compensar para mantener y controlar la inclinación lateral.

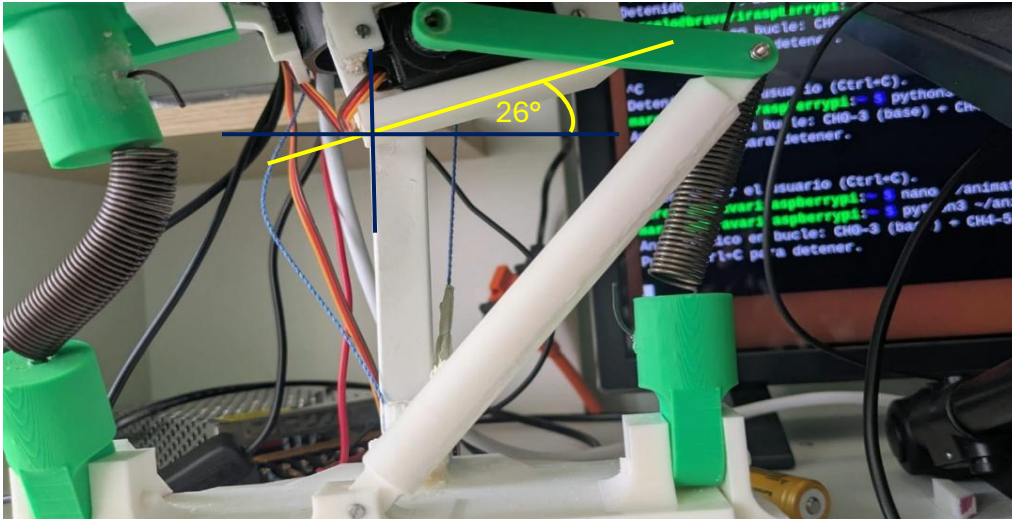


Figura 35. Diagrama "Momento Gravitacional"

2.7.10. Compensación elástica mediante resortes

El sistema involucra un componente elástico (resortes) como un sistema de soporte mecánico para amortiguar el peso del conjunto móvil y evitar que la carga directa recaiga sobre el brazo principal del actuador principal durante la inclinación lateral del cuello. Esta decisión no se tomó de manera aislada con respecto a un componente "adicional", sino como un recurso de diseño que llevaría a optimizar el rendimiento del mecanismo bajo condiciones de operación reales donde la masa bruta de la cabeza, la fricción resultante de las articulaciones, la inercia y las pequeñas holguras características de las piezas impresas pueden aumentar la intensidad de las vibraciones y crear tensiones excesivas en el servomotor. Funcionalmente, los resortes cumplen una función de soporte parcial: añaden energía elástica cuando el mecanismo cambia de posición y son responsables de devolver parte de esa energía para mantener una posición estable en el motor sin la necesidad de los torques constantes del servomotor. Esto significa que el actuador ya no "soporta" todo el peso por sí solo, y ahora tiene que controlar principalmente la orientación (conduce el gesto) y los resortes proporcionan carga estática y amortiguación de carga dinámica (lo que participa en la distribución de la carga) [24].

Esta incorporación ayuda a suavizar las transiciones, mientras que los cambios de inclinación tienden a ser menos bruscos, porque hay menos fuerza: el sistema elástico ejerce el efecto de amortiguación mecánica (lo que implica atenuar sacudidas o temblores visibles). En segundo lugar, facilita menos picos de demanda para el actuador: al compartir la carga, el esfuerzo del servomotor se reduce en posiciones vitales y se logra menos calentamiento, y el proceso de acción es más repetible y menos propenso a "temblar" o micro-oscilaciones cuando el control intenta mantenerse en la postura adecuada. Por último, los resortes proporcionan más tolerancia

a las perturbaciones, por ejemplo, pequeños golpes, vibraciones o cambios repentinos de ángulo, lo que resulta en que el cuello regrese mejor a la posición de equilibrio. [25].

En última instancia, los resortes sirven como un mecanismo de compensación pasiva que ofrece un ligero alivio al cuello y una mejor calidad de movimiento al atenuar ligeramente las fuerzas del conjunto móvil y reducir la carga efectiva del servomotor durante la inclinación lateral de la cabeza. Un aspecto que a menudo se pasa por alto es la seguridad: eliminar bordes peligrosos y no permitir que los mecanismos en movimiento comprometan el riesgo de atrapamiento, donde el sistema también puede cerrarse rápidamente o desmontarse. En la animatrónica, el mantenimiento es una condición más operativa que un "extra"; el prototipo necesitará permitir el acceso a fijaciones, servos y cableado en el área de prueba sin destruir la carcasa para que la demostración pública sea duradera con el tiempo. Por eso, la carcasa superior del cráneo era desmontable mediante la combinación de los programas Meshmixer y Fusion 360-----servomotor.

2.7 Edición orgánica y reparación de mallas con Meshmixer

Mientras Fusion 360 fue un buen comienzo en la preparación paramétrica para el proceso de generación de la parte mecánica, Autodesk Meshmixer intervino para afinar los detalles orgánicos de la carcasa para la imprimibilidad del modelo. Meshmixer es una herramienta diseñada específicamente para manejar mallas trianguladas (STL/OBJ), que tiene capacidades de escultura digital, suavizado de superficies y reparación de geometría que pueden considerarse como parte de un aspecto complementario al CAD convencional. Las carcasas basadas en personajes suelen tener geometría orgánica típicamente expresada a través de mallas STL trianguladas. Estas mallas son suficientes para la impresión, pero es difícil integrar componentes mecánicos internos y se requiere referencias de ensamblaje. Aquí, entonces, el procesamiento de geometría permite las operaciones necesarias: reparación de mallas (normales, agujeros, auto-intersecciones), suavizado, deformación controlada y segmentación para impresión. El valor aquí está en la realización de que una malla "estética" que casi nunca se ensambla directamente: demanda ser convertida en una carcasa utilizable con ciertos espesores, aberturas de servicio, aberturas para tornillos y un volumen de movimiento apropiado de los mecanismos asociados con la malla. La diferencia en la transición de malla a CAD proporciona una limitación adicional: aunque el CAD paramétrico puede manejar sólidos y superficies definidas, la malla es un material discreto. Como resultado, el flujo de trabajo a menudo resulta en un híbrido, manteniendo la malla donde da forma estética, pero produciendo

sólidos paramétricos en áreas donde se necesita precisión física realista (interfaces, soportes, carcasas y uniones). Los métodos iniciales para el flujo digital se adaptaron a la estructura interna del prototipo, con un énfasis en criterios de fabricación aditiva en lugar de modificaciones morfológicas.

La Fase uno reveló la necesidad de fabricar el cuerpo interno en un solo bloque, lo cual fue costoso en términos de gran consumo de material y un aumento sustancial en soportes (especialmente en el 'segundo piso' o nivel superior), lo que llevó a un aumento en el tiempo de fabricación y el desperdicio asociado. Así que la estructura interna se dividió en dos piezas para minimizar el uso de soportes y simplificar el proceso de impresión. Sin embargo, como esta estructura no necesitaría ser reorganizada, generalmente se trató como una forma sólida típica en capturas sucesivas de Fusion 360, otras partes se trataron como una forma orgánica y como una malla triangulada y no una textura poligonal. Esto se puede observar en detalle en las Figuras 36 y 37.

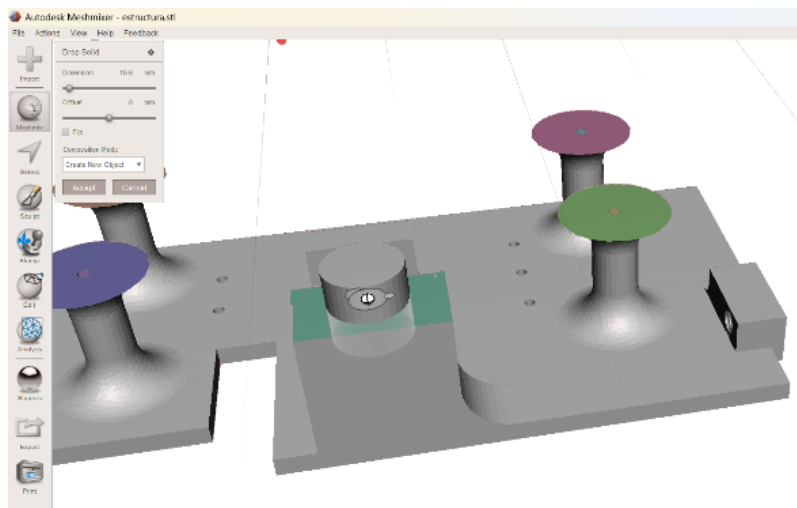


Figura 36. Vista de la estructura interna, evidenciando el espacio reservado para el servomotor MG996R.

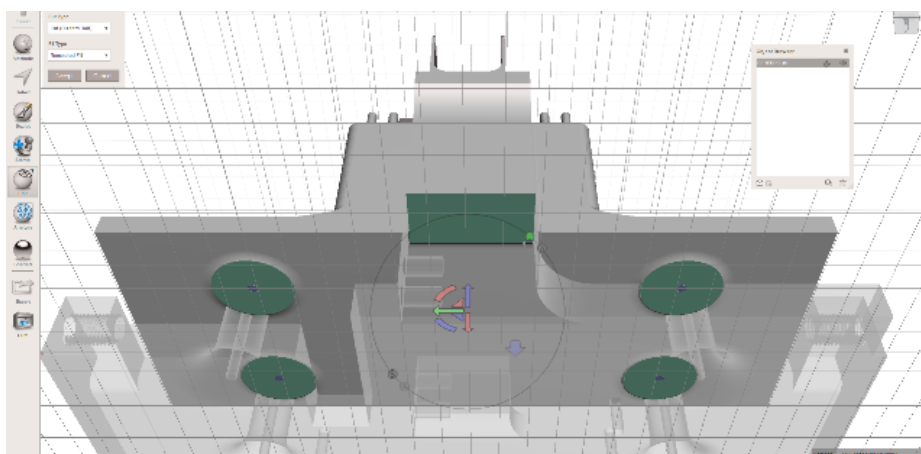


Figura 17. Vista del corte hecho a la estructura.

En el proyecto, Meshmixer cumplió dos funciones complementarias: Edición orgánica de la carcasa visible del personaje y preparación geométrica para impresión de componentes que, por su escala o complejidad, no podían resolverse de forma eficiente únicamente con operaciones paramétricas.

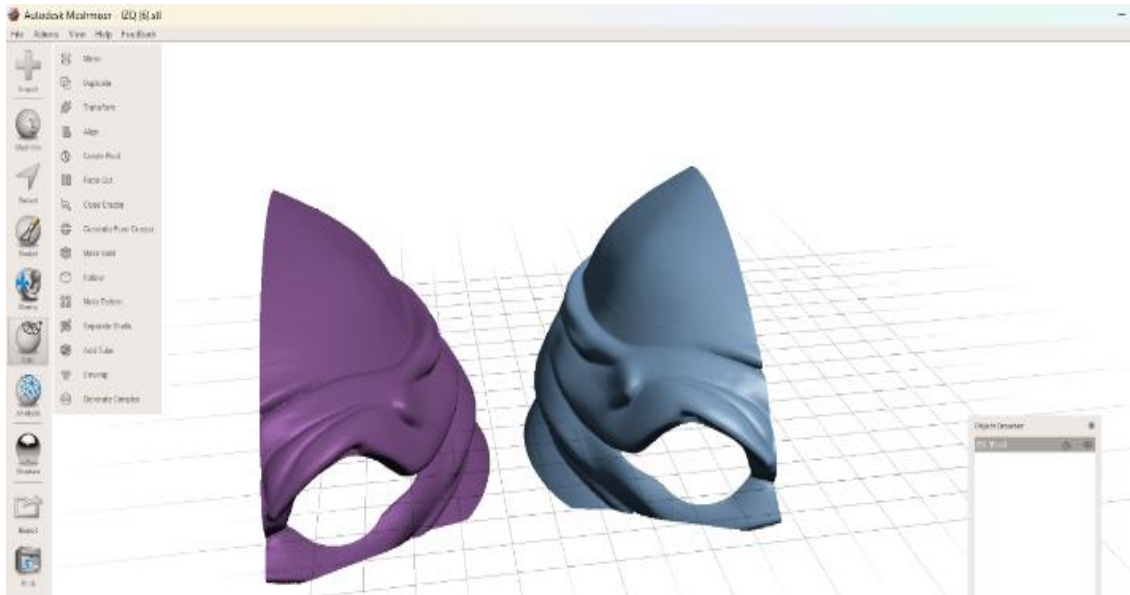


Figura 38. Edición orgánica de la carcasa.

La Figura 38 muestra la modificación morfológica de la carcasa y, especialmente, del cráneo superior, utilizando Meshmixer, donde la geometría fue ajustada usando herramientas más adaptables, en contraste con el CAD paramétrico clásico. Tales intervenciones incluyeron cambios de forma para preservar el reconocimiento estético del personaje, y elementos de Fusion 360 que eran desafiantes sobre la base sólido-paramétrica (por ejemplo, crear aberturas para los ojos, cortes locales y modificaciones de superficie que requerían continuidad visual y control de contornos orgánicos).

En segundo lugar, Meshmixer podría usarse para realizar tareas directamente relacionadas con la fabricabilidad: segmentación de malla para establecer cortes controlados, y reparación subsiguiente para preservar cada segmento como un sólido imprimible (hermético), libre de los fallos típicos de malla (agujeros, normales inconsistentes o elementos no múltiples) que pueden ocurrir en la etapa de laminación o causar defectos durante la impresión. La Figura 39 muestra la generación de cortes controlados.

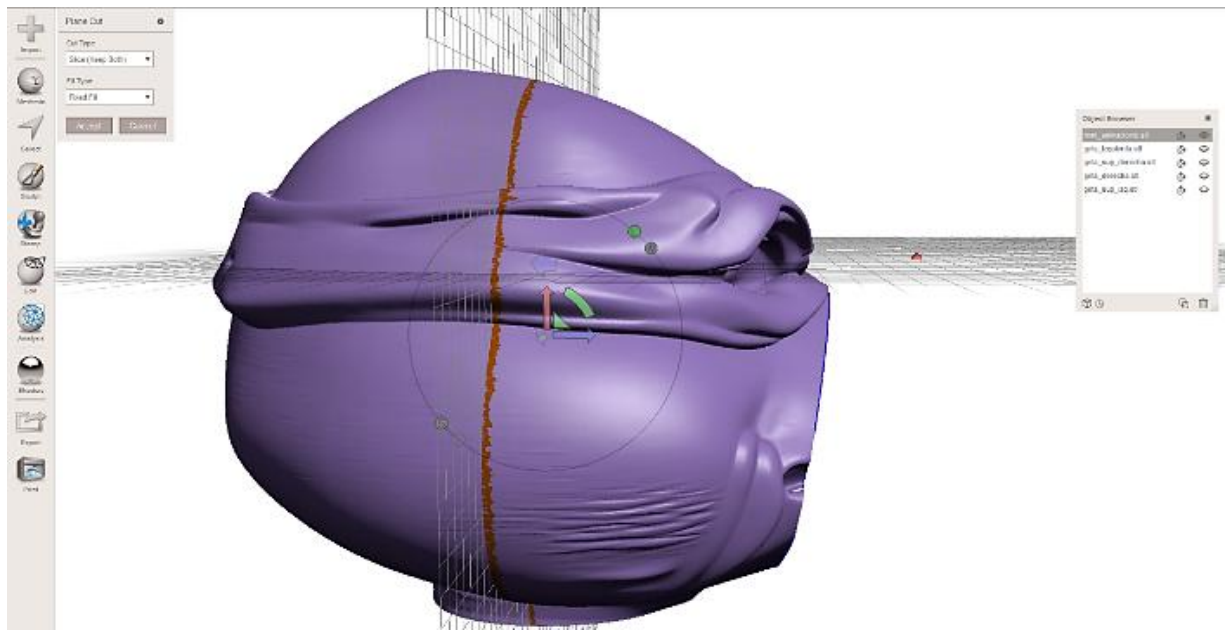


Figura 39. *Múltiple división de la carcasa para posterior impresión.*

2.8 Reintegración en Fusion 360 y diseño de interfaces mecánicas para desmontaje

Después de modificar la carcasa y el cráneo en Meshmixer, estos fueron reintegrados en Fusion 360, donde se dibujaron y modelaron directamente las interfaces mecánicas (soportes, alojamientos y superficies de acople) en el ensamblaje interno. Pero las geometrías provenían de un flujo de trabajo de escultura y reparación, por lo que se importaron como mallas trianguladas. Esto se refleja en las Figuras 40 y 41, la carcasa se muestra con la malla visible. Esto no sugiere su uso como referencia geométrica en lugar de un sólido paramétrico nativo que ya había sido construido [26].



Figura 40. Validación de diseños finales de carcasa superior.

Este paso fue crucial por una razón práctica: las partes de la carcasa ya habían sido impresas antes de tiempo, luego de darse cuenta de que debe ser desmontable se implementó una estrategia de acople en dos subconjuntos. Por tanto, para asegurar compatibilidad dimensional con la carcasa “post Meshmixer”, se diseñaron en Fusion 360 nuevos soportes adaptados a la geometría actual, verificando interferencias y holguras con la carcasa triangulada como referencia.

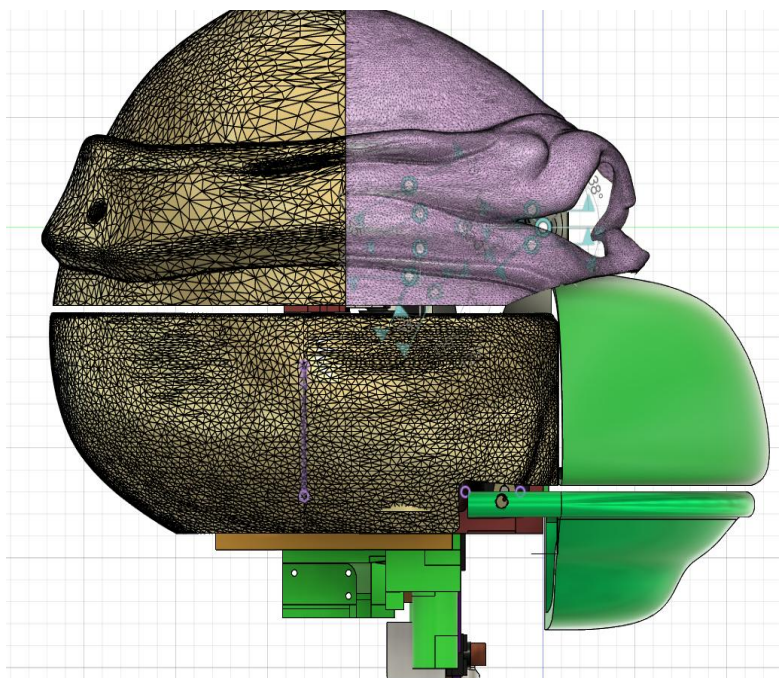


Figura 41. Validación de diseños finales de carcasa inferior.

2.9 Estrategia de montaje desmontable y ensamblaje por acople + adhesión

Con el fin de que la carcasa fuese desmontable (requisito asociado a mantenimiento, transporte y demostraciones públicas), se implementó una nueva estrategia:

- Un conjunto de soportes solidarios a la carcasa, concebidos para fijarse a ella, en la práctica, debido a que ya estaban impresas las partes de la carcasa y se requería una unión rápida, se empleó adhesivo para su integración.
- Un conjunto complementario de soportes solidarios a la estructura interna, diseñado para recibir el primer conjunto mediante un encaje geométrico.

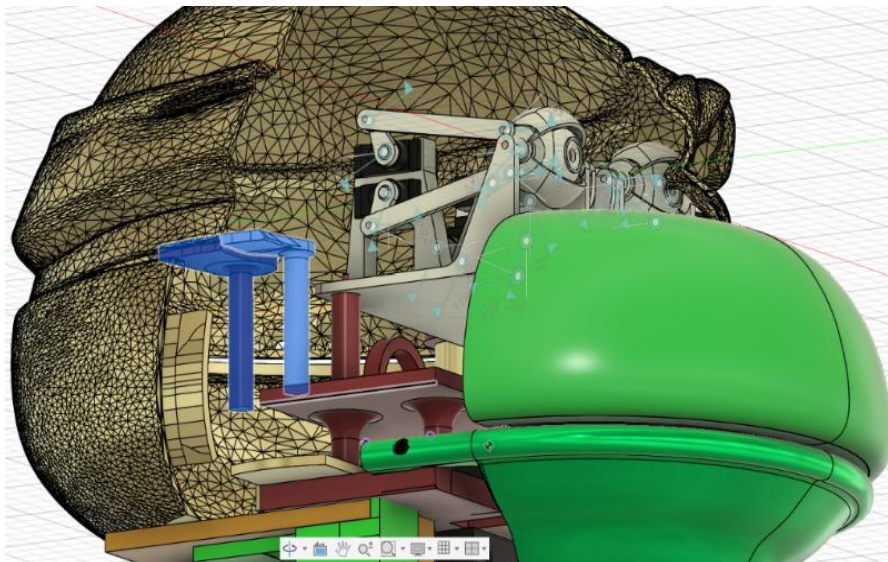


Figura 42. Soportes solidarios a la carcasa.

El sistema, funcionalmente, encajaría ya que los soportes montados en la carcasa del estuche se colocaron para "entrar" dentro de las estructuras de soporte, lo que significa que la carcasa puede ser removible sin que el resto del mecanismo interno se desmonte. Estos soportes se muestran en azul en las Figuras 42 y 43. Además del beneficio de una mayor mantenibilidad, dicho diseño también disminuye el riesgo de daño a la integridad del cableado y los actuadores cuando se inspecciona o ajusta el ensamblaje, y para el reemplazo y reimpresión de módulos externos (sin comprometer la integridad del ensamblaje mecatrónico) [27].

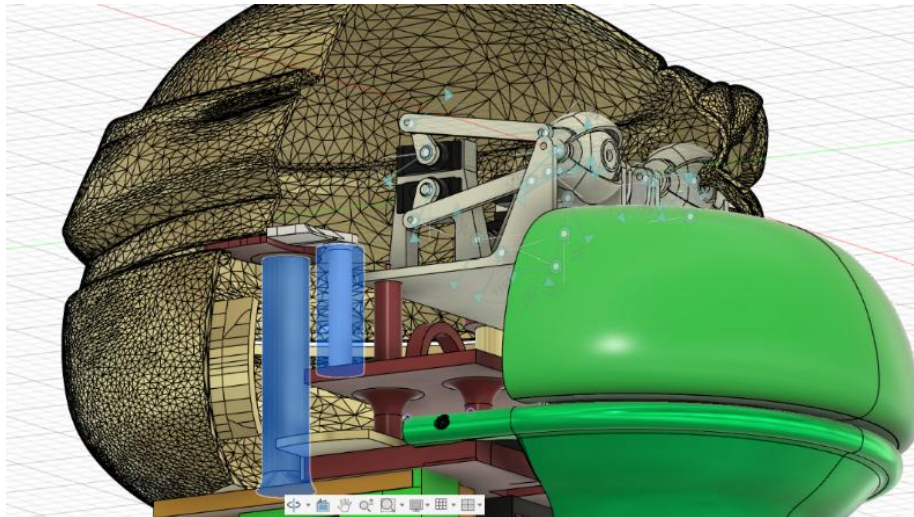


Figura 43. Acople a la carcasa adherido a la estructura interna.

2.10 Preparación de impresión 3D: corte, orientación y configuración en Creality Print

La fabricación por deposición de material impone limitaciones anisotrópicas para las capas debido a la anisotropía mecánica, tolerancias de diseño dependientes del proceso, la necesidad de soportes, deformaciones por contracción, y la resistencia del proceso de impresión depende de la orientación de la impresión. Estos fenómenos gobiernan directamente la corrección estructural y la estabilidad dimensional del ensamblaje en una carcasa animatrónica.

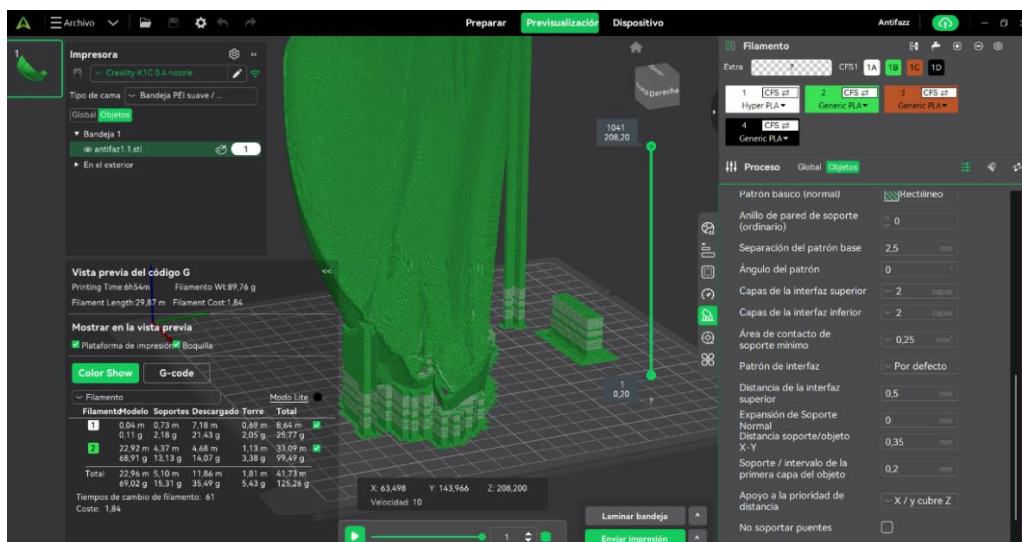


Figura 44. Vista de laminado en Creality Print, carcasa superior.

El corte, ilustrado en la Fig. 44, es la interfaz mecánica entre la geometría y el material. En este proyecto, se ha utilizado Creality Print 6.3 como el entorno de corte para K1C, donde se han definido parámetros que afectan directamente su resistencia, acabado superficial, tiempos de fabricación y riesgos de fallos: orientación de la pieza, generación de soportes, densidad de

relleno, espesores de pared, alturas de capa y estrategias de adhesión. Teóricamente, esta etapa se considera un proceso de conversión de necesidades mecánicas a parámetros de proceso, es decir, perímetros aumentados incrementan la rigidez de partes de borde delgado, la orientación de las partes incrementa la resistencia en ciertas direcciones críticas, y la planificación de soportes disminuye defectos en partes funcionales. La Fig. 45 muestra estos soportes y la orientación del cuerpo para minimizar los tiempos de impresión.

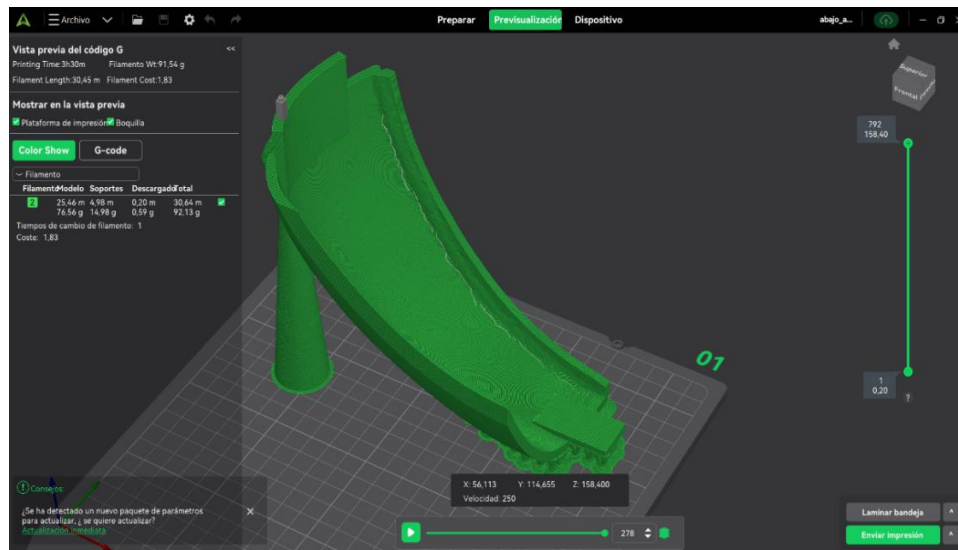


Figura 45. Vista de laminado en Creaform Print, carcasa inferior.

Como un elemento de aseguramiento de calidad en forma de un componente de monitoreo mediante una cámara AI puede detectar fallos mientras se imprimen las piezas y si se utilizan bajo un sistema cargado con estrés y ciclos repetidos. La K1C, tiene como elemento de calidad una cámara que graba en tiempo real, con funcionalidades de detección asistida (cámara AI) para monitorear el proceso. Los detalles oficiales del producto dicen que la cámara incluye observación en vivo pero también alertas para errores de impresión, video/time-lapse. En el contexto del proyecto, esta capacidad se ve como una herramienta de control de calidad cuando hay un evento inesperado como un gran grado de deformación al salir de la boquilla, una primera capa anormal o un fallo en la calidad de impresión, el sistema detiene la impresión, ahorrando el desperdicio de material y previniendo que componentes estructurales diseñados para soportar golpes se impriman con defectos que están fuera de vista durante el uso, con la posibilidad de arreglar el problema y continuar con la impresión. La documentación de soporte de Creaform explica correcciones de sensibilidad y comportamientos de funciones AI, así como

códigos/alertas alrededor de la detección de problemas (donde hubo una pausa por calidad o detección de deformaciones).



Figura 46. Vista de (AI camera) CrealityK1C.

Con el diseño final de cada componente de la carcasa definido y las mallas listas, se procedió a la etapa de fabricación mediante impresión 3D. Creality Print 6.3 actúa como un slicer o laminador: permite importar los modelos 3D (en formato STL u OBJ), posicionarlos virtualmente en la cama de impresión, configurar los parámetros de impresión (calidad, relleno, soportes, etc.) y finalmente generar el código G necesario para que la impresora fabrique las piezas. La Creality K1C es una herramienta esencial para la rápida iteración y producción de componentes del animatrónico. Su diseño se enfoca en la velocidad y la capacidad de procesar materiales de ingeniería. La K1C implementa la estructura de movimiento CoreXY, que ofrece una plataforma estable y ligera para el cabezal de impresión. Este diseño cinemático permite que la impresora alcance velocidades máximas de 600 mm/s y aceleraciones de 20000 mm/s². La rapidez en la producción de prototipos es vital para el ciclo de desarrollo de sistemas mecatrónicos.

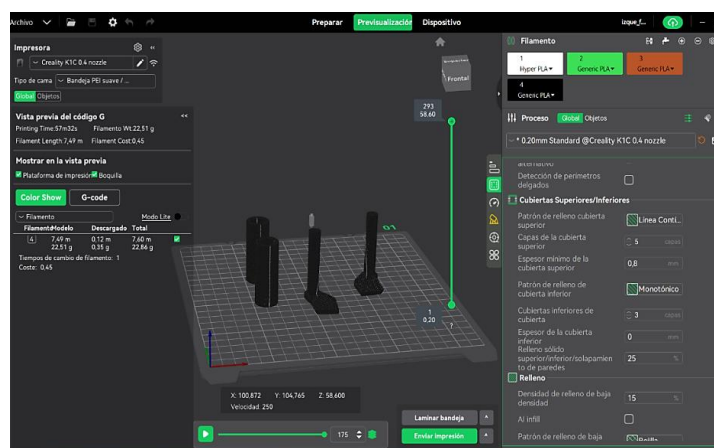


Figura 47. Vista Creality Print de los soportes y acoples.

2.10.1. Creality Print 6.3: el entorno de laminado (slicer) como interfaz CAD–manufactura

En términos funcionales, el laminador cumple el rol de “compilador” de manufactura: transforma geometrías trianguladas o sólidas exportadas (p. ej., STL/3MF) en trayectorias de deposición y en un conjunto de instrucciones G-code que gobiernan el movimiento, la extrusión y las condiciones térmicas durante el proceso. Esta etapa no es meramente operativa, sino determinante para la calidad final, ya que condiciona parámetros críticos como se aprecia en la Figuras 48 y 49, donde se miran algunos de estos parámetros. Los parámetros que se pueden modificar en esta opción son: la altura de capa, número de perímetros, densidad y patrón de relleno, generación de soportes, velocidad/aceleración, temperatura, retracción y estrategia de costura (seam); variables que inciden directamente sobre la resistencia mecánica, la estabilidad dimensional y el acabado superficial de las piezas.

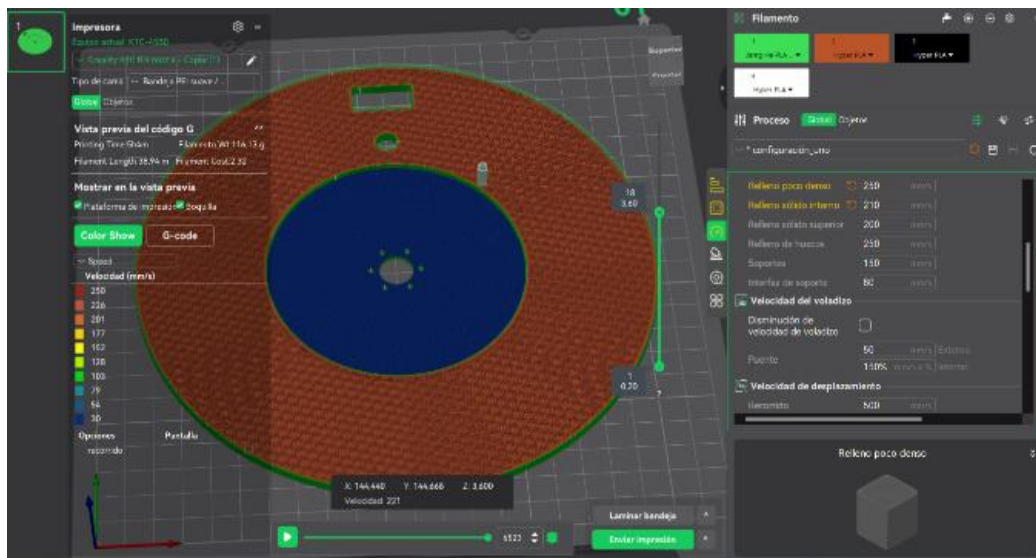


Figura 48. Laminado y velocidades de impresión.

Desde el punto de vista de recursos computacionales, Creality Print requiere un sistema de 64 bits, con un mínimo de 4 GB de RAM y compatibilidad gráfica OpenGL (siendo OpenGL 4.1 recomendable para visualización avanzada de capas en 3D), además de requisitos de procesador y almacenamiento coherentes con aplicaciones de modelado y visualización. En documentación oficial se especifica explícitamente la condición de 64 bits y la RAM mínima, así como un perfil recomendado superior para un desempeño fluido en tareas de previsualización y análisis de capas. En el contexto del proyecto, esto adquiere relevancia práctica porque la verificación previa de trayectorias, soportes y transiciones de material (cuando aplica) reduce iteraciones físicas, desperdicio y tiempo de prototipado.

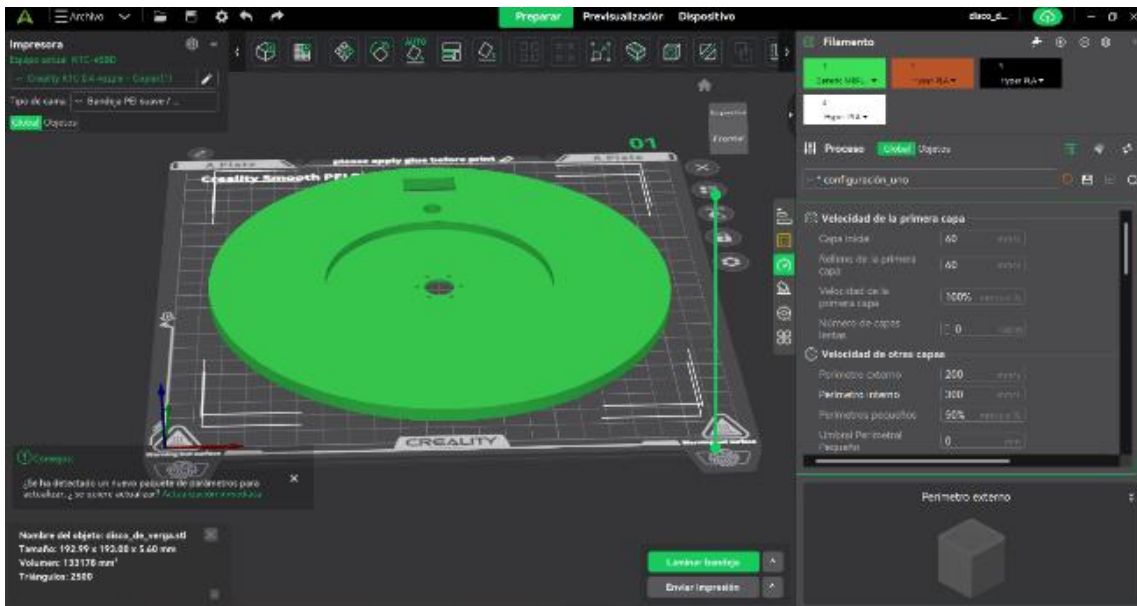


Figura 49. Paso a CrealityPrint 6.3

Para el caso particular de un animatrónico donde las piezas impresas como soportes, carcasas, bases, uniones, alojamientos de servos y componentes estructurales, serán sometidas a cargas cíclicas, vibración y esfuerzos concentrados en puntos de fijación el laminado deja de ser una configuración “estándar” y se convierte en una decisión de ingeniería. Por ejemplo, orientar una pieza para alinear perímetros con la dirección principal del esfuerzo, incrementar perímetros externos en zonas de fijación, o ajustar densidad de relleno local mediante modificadores, impacta directamente en la resistencia a delaminación y en la vida útil frente a fatiga. En la práctica del prototipado del sistema, estas decisiones se vuelven especialmente críticas en componentes vinculados a servomotores (p. ej., soportes de MG90S/MG995, uniones y bases), donde una falla de impresión no solo afecta estética: compromete la geometría funcional, la repetibilidad del movimiento y la seguridad del conjunto.

No obstante, desde un enfoque académico riguroso, la cámara AI debe entenderse como una capa de supervisión complementaria, no como sustituto del control de proceso. La efectividad real depende de condiciones de iluminación, configuración de sensibilidad, tipo de material, geometría y del propio firmware; por ello, en este trabajo se la considera un apoyo para reducir fallos evidentes y mejorar trazabilidad del prototipado, mientras que la validación final de integridad estructural se sustenta principalmente en: Revisión de vista de capas y soportes en Creality Print, la selección de orientación y parámetros de adhesión, el control térmico y de velocidad para evitar sub-extrusión o deformación, y las pruebas funcionales de montaje y operación bajo carga en el ensamblaje animatrónico.

En síntesis, Creality Print 6.3 no se limita a “laminar”; constituye el entorno donde se formalizan las condiciones de manufactura que harán viable el sistema mecatrónico, articulando la intención del diseño (CAD) con la realidad de fabricación (FFF). En un animatrónico, donde la impresión 3D no es decorativa sino estructural y funcional, este software se vuelve un componente metodológico del proyecto: define la calidad de las piezas y, por extensión, la confiabilidad del movimiento, la repetibilidad de la actuación y la durabilidad del prototipo en operación.

En este proyecto, Creality Print fue esencial para determinar la orientación óptima de cada pieza y para realizar ajustes finales en el corte de estas. Inicialmente, se cargaron los componentes segmentados de la cabeza (cráneo, hocico, mandíbula, soportes internos, etc.) en la interfaz del slicer. Se evaluaron distintas orientaciones y rotaciones para cada pieza, con el objetivo de minimizar la cantidad de soportes de impresión y asegurar una buena calidad superficial en las zonas exteriores visibles. Por ejemplo, el *antifaz* (sección frontal) se orientó de manera que la parte convexa quedara hacia arriba, evitando que la amplia curva de la nariz necesitara soportes por debajo; de esta forma se logró una superficie lisa en el exterior del antifaz y cuerpo al no requerir material de soporte que pudiera dejar marcas. Del mismo modo, la mandíbula inferior se colocó con la abertura de la boca hacia abajo, permitiendo que la mayoría de los voladizos se apoyaran por sí mismos y limitando los soportes solo a la parte interna de la cavidad bucal.

En la fase de pre-fabricación, Creality Print se utilizó como herramienta de *slicing* para transformar la geometría final (proveniente de CAD y de malla editada) en trayectorias de deposición imprimibles, incorporando además estimaciones de tiempo, consumo de material y validación visual de zonas críticas antes de la fabricación. En términos metodológicos, esta etapa se asumió como el “último filtro” del hilo digital: aquí se verificó que las decisiones geométricas (cortes, espesores, ventanas y alojamientos) mantuvieran coherencia con restricciones reales de impresión (voladizos, puentes, estabilidad en cama y removibilidad de soportes).

La división, realizada previamente en Meshmixer, permitió que cada porción se imprimiera con menos soportes y menor altura, reduciendo significativamente el tiempo de fabricación. Creality Print fue utilizado para simular estas condiciones de impresión y calcular estimaciones de duración, material consumido y posibles áreas conflictivas. Adicionalmente, el *slicer* permitió ajustar parámetros globales de fabricación según un balance entre rigidez, tiempo de

impresión y resolución superficial. Con base en simulaciones y pruebas, se empleó un infill moderado (~20%) para mantener resistencia estructural suficiente en la carcasa sin penalizar de forma desproporcionada el tiempo total de impresión, y se adoptó una altura de capa fina ($\approx 0.16\text{--}0.20$ mm) para favorecer la continuidad visual de superficies curvas y rasgos faciales. Complementariamente, cuando existió riesgo de despegue o deformación por área de contacto reducida, se incorporaron brims/faldones de adherencia como medida de estabilidad durante el proceso, reduciendo la probabilidad de fallas por pérdida de adhesión en cama.

Otro aspecto importante configurado en Creality Print fue la generación de estructuras de soporte personalizadas. Si bien se buscó minimizar su uso mediante la correcta orientación, algunas piezas complejas como ciertas partes de la máscara o los rebordes de la cabeza requirieron soportes removibles para imprimirse correctamente. El slicer permitió ajustar parámetros como el patrón y densidad de estos soportes, de modo que fueran fáciles de retirar y dejaran la menor marca posible. Adicionalmente, se verificó la adhesión a la base de cada pieza; en casos de piezas con poca superficie de contacto se añadieron *brims* o faldones de adherencia para prevenir deformaciones (*warp*) durante la impresión. Gracias a esta cuidadosa configuración en Creality Print, cada componente de la carcasa se fabricó exitosamente en impresora 3D, obteniendo piezas precisas que encajan entre sí y presentan un buen acabado, aptas para el montaje del animatrónico.

Finalmente, en piezas con voladizos inevitables (por ejemplo, rebordes o elementos internos con geometría negativa), se configuraron soportes removibles con densidad y patrones orientados a facilitar su extracción, reduciendo marcas en la superficie y evitando daños en bordes funcionales. De esta manera, la preparación en Creality Print no se limitó a “laminar” modelos, sino que operó como una etapa de ingeniería de fabricación: se controlaron variables de proceso para asegurar que los componentes fabricados encajen entre sí, con acabado adecuado y repetibilidad suficiente para el montaje del animatrónico.

2.10.2 Selección del material de impresión: PLA

Para la fabricación aditiva de la carcasa y múltiples elementos auxiliares se seleccionó PLA (Polylactic Acid) como material principal, debido a que el propósito del prototipo es demostrativo y de divulgación académica, donde se prioriza: calidad visual, facilidad de impresión y baja tasa de fallas, y costos de fabricación controlados para iteraciones sucesivas. En este contexto, el PLA ofrece ventajas prácticas ampliamente reportadas: es un material fácil de imprimir, con baja tendencia al *warping* y adecuado para piezas de distintos tamaños, lo

cual favorece ciclos iterativos rápidos sin introducir complejidad térmica excesiva en el proceso. [28]

Desde la perspectiva termo-mecánica, el PLA presenta limitaciones conocidas (p. ej., menor resistencia a temperatura y comportamiento relativamente frágil frente a otros termoplásticos), por lo que su selección no se justificó como “material estructural extremo”, sino como un material óptimo para prototipado funcional con alta calidad geométrica y buen acabado superficial. En el proyecto, los requisitos de carga elevada se resolvieron mediante elementos metálicos (tornillería, ejes, uniones y puntos de fijación), delegando en el polímero principalmente la función de carcasa, soporte geométrico y encapsulado estético-funcional, con refuerzos locales cuando fue necesario. Esta decisión es coherente con un prototipo orientado a exhibición: el desempeño global se logra por la combinación de un material imprimible y estable (PLA) con herrajes metálicos en las zonas donde la confiabilidad mecánica es crítica.

Adicionalmente, el PLA facilita la economía de iteración: al mantener un costo por pieza relativamente bajo frente a alternativas como PETG/ABS en servicios universitarios y flujos de prototipado, se habilitan más versiones y correcciones sin elevar significativamente el costo total del desarrollo, lo cual es clave en un proceso iterativo con ajustes por interferencias, tolerancias reales y mantenibilidad. [29]

2.11 Desafíos encontrados y aprendizajes en el diseño

A lo largo del proceso de diseño y construcción del animatrónico se presentaron diversos desafíos técnicos, cuyos análisis y soluciones contribuyeron al aprendizaje del equipo. Uno de los retos más significativos fue el diseño del mecanismo de movimiento para el cuello del personaje. Inicialmente, se concibió un sistema con engranajes y múltiples servomotores para lograr movimientos en varios grados de libertad (inclinación y giro de la cabeza). En concreto, se incorporaron dos pequeños servomotores modelo MG90S a los lados del cuello, acoplados mediante brazos y engranajes a la estructura de la cabeza, con la idea de no usar servos de mayor torque (modelo MG995 o MG996R) en los movimientos del cuello y así prescindir del uso excesivo de corriente en el uso de modelos de mayor torque. Esta configuración se ve presente en las Figuras 50 y 51, donde se pretendía repartir el esfuerzo entre varios motores de un consumo bajo de corriente; no obstante, en la práctica presentó problemas de desempeño.

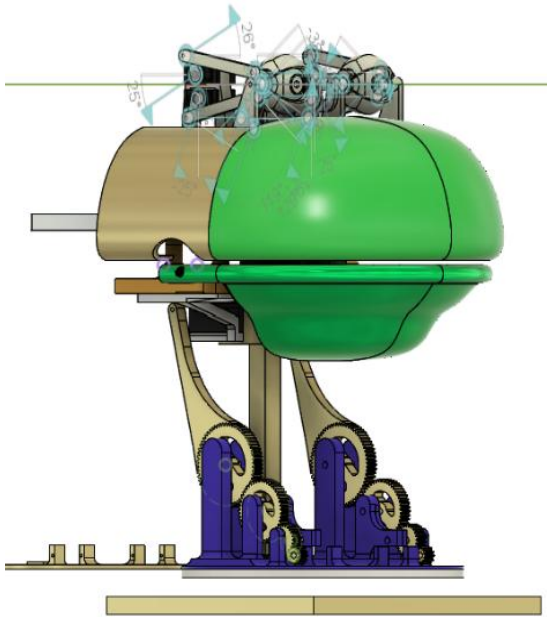


Figura 50. *Diseño tren de engranajes.*



Figura 51. *Vista derecha, diseño impreso tren de engranajes.*

Prototipo inicial del mecanismo de cuello con servomotores laterales (diseño luego descartado). En esta versión, se fijaron dos micro-servos MG90S (uno a cada lado) conectados mediante pequeños engranajes al eje del cuello. Durante las pruebas mecánicas, se evidenció falta de torque efectivo: los servomotores pequeños no tenían la fuerza suficiente para mover la cabeza con fluidez, especialmente considerando el peso de la carcasa impresa. Como resultado, el movimiento de la cabeza era en el mejor de los casos limitado e insuficiente para los requerimientos del animatrónico, añadiendo que solo se obtuvo ruido y calentamiento de los servomotores al no alcanzar sus ángulos programados por falta de torque. Este error de diseño llevó a descartar por completo la configuración con múltiples servos y engranajes en el cuello.

Lecciones valiosas surgieron de este fallo. En primer lugar, se reafirmó la importancia de realizar un análisis de torque y carga antes de implementar un sistema de actuadores múltiples: agregar más servos pequeños no necesariamente incrementa la fuerza si cada uno individualmente no puede con la carga y, peor aún, pueden interferir entre sí mecánicamente. En segundo lugar, se aprendió que a veces una solución más simple resulta más efectiva; tras eliminar los servos laterales, se reorganizó el diseño del cuello alrededor de un único servomotor de alto torque (el MG995 central) que, con la palanca adecuada, fue capaz de brindar el movimiento deseado de la cabeza, este diseño con el MG995 central se muestra en la Figura 52. De hecho, al simplificar el mecanismo se redujo la fricción y holguras, y una

programación más sencilla al necesitar solo un servomotor para esta tarea, mejorando la confiabilidad.

Otros desafíos enfrentados incluyen la resistencia mecánica de piezas impresas y la precisión del ensamblaje. Por ejemplo, cuando se hizo otro diseño, muy parecido al diseño actual, la primera versión del brazo o palanca que transmite el movimiento del servomotor al cuello resultó ser estructuralmente débil en puntos críticos, lo que provocó pequeñas grietas tras varios ciclos de apertura y cierre de la boca. Identificado este problema, se rediseñó dicha pieza reforzándola con mayor espesor y fillets (curvas) en las uniones para distribuir mejor el esfuerzo. La versión modificada del brazo del cuello resistió adecuadamente, evitando futuras fracturas. Asimismo, se realizaron ajustes finos en la tolerancia de las uniones entre piezas impresas: se lijaron o calibraron aquellas superficies que tenían contacto deslizante para lograr un movimiento suave, pero sin holgura excesiva. Este proceso de prueba y error fue fundamental para optimizar el desempeño del animatrónico, evidenciando cómo cada iteración del diseño aportó mejoras incrementales. La estabilidad entonces se reemplazó por el sistema de resortes final, ya que era de nuevo, la mejor y más sencilla solución.

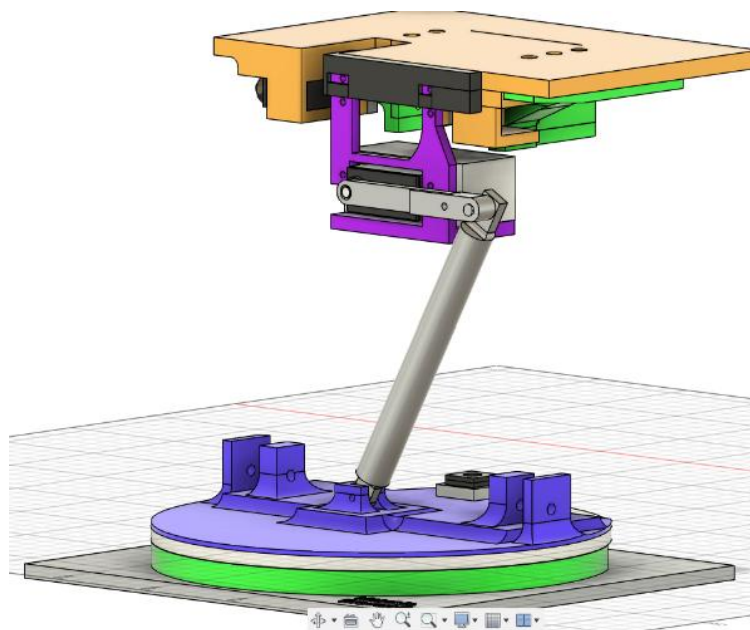


Figura 52. *Diseño inicial del modelo actual.*

2.12 Iteraciones del diseño y resultado final de la carcasa animatrónica

Después del proceso de diseño y prueba, producimos la versión final de la carcasa animatrónica, integrando las lecciones de versiones anteriores. Este capítulo presenta los

resultados parciales más significativos y también explica por qué decidimos alterarlos para alcanzar el diseño final.



Figura 53. *Apertura de mandíbula animatrónico (real).*

En la Figura 53 se muestra la versión final de la cabeza animatrónica propuesta, no cambia mucho y conserva casi todos los elementos del diseño original, pero con leves modificaciones clave en su ejecución. En cuanto al mecanismo de la mandíbula, la solución definitiva utiliza un motor servo MG996R dedicado a la mandíbula a través de una palanca reforzada, un MG995 en el cuello conectado a un brazo que levanta la cabeza y la gira ligeramente hacia la izquierda.

Debido a que esta configuración bastante sencilla funcionó bien, la primera alternativa con un mecanismo de engranajes era más complicado y fue descartado a tiempo. La mandíbula puede abrirse y cerrarse en un ángulo apropiado para simular un movimiento de habla, y el servomotor de alto torque puede trabajar con fuerza y rapidez, dando realismo al movimiento sin temor a que el motor se atasque. La versión exacta del cuello es una base rotacional básica donde hay un MG90S, que puede girar ligeramente la cabeza de derecha a izquierda, combinado con un MG995 en el cuello, que levanta la cabeza y la desplaza ligeramente hacia la izquierda, dando el movimiento de tipo inclinación lateral del cuello. En el último prototipo, se priorizó la necesidad de un control fuerte y preciso de un movimiento.

En cuanto a la parte exterior de la carcasa, el primer modelo de hocico impreso demostró funcionar desde el principio y la versión final lo mantuvo ya que tenía el perfil requerido y se integraba adecuadamente con el resto de la cabeza. Pero otros elementos diseños fueron revisados iterativamente: la carcasa del cráneo superior, por ejemplo, fue reimpressa después de agregar más puntos de anclaje atados al interior del cráneo, de modo que, aunque funcionaba, la carcasa no saltara alrededor del hueso interno. De manera similar, el ajuste entre las secciones del hocico y el cráneo fue rediseñado para minimizar las líneas de separación claras y para que la carcasa tuviera un aspecto "suave". Las juntas visibles fueron cubiertas con masilla y lijadas como tratamiento final con la esperanza de algún día pintar la cabeza con algunos de los colores del personaje (máscara de color, tono de piel verde, etc.). Carcasa terminada, junto con las partes internas. Aquí puedes ver la cabeza de la Tortuga Ninja con todas las partes impresas en la cabeza (mandíbula articulada, hocico y cráneo) que juntas forman el aspecto exterior, y dentro, ves el servo de la mandíbula y el servo del cuello en el centro. La configuración consistía en asegurar los cables del motor servo y alinearlos en las paredes internas para que sus partes no chocaran con nada más en el sistema. Junto con todo esto, todas las cubiertas también fueron aseguradas y mantenidas con adhesivo, para asegurar que la carcasa, cuando esté en movimiento, funcione como una sola pieza sólida.

El desafío final es encontrar el equilibrio entre el detalle estético y la simplicidad mecánica. La cabeza de la Tortuga Ninja necesitaba ser atractiva, sí, pero también funcional por dentro. Algunos detalles minuciosos en el modelo 3D tuvieron que ser sacrificados para poder imprimir o mover (las cercas se doblaron para que las texturas menos finas no fueran necesarias en la impresión, y las paredes de la carcasa eran lo suficientemente gruesas para que los tornillos, soportes y ejes se sujetaran a una superficie). Todas estas fueron decisiones que requirieron un poco de juicio ingenieril, con la integridad estructural y la función siempre anteponiéndose al aspecto perfecto. Sin embargo, algunos aspectos estéticos que se habían perdido pudieron ser reintroducidos de manera inteligente utilizando herramientas CAD de manera iterativa con la mecánica adecuada, y satisfizo algunos de los objetivos técnicos y artísticos del proyecto.

2.13 Validación y confirmación del diseño final

El diseño fue validado de manera iterativa (digital y física) para confirmar el diseño final. Primero, se verificó todo el ensamblaje con CAD basado en la detección de interferencias, análisis de secciones y verificación de la trayectoria de movimiento de las partes móviles

(párpados, transmisión ocular y apertura de la mandíbula) para verificar que los ángulos de trabajo previstos fueran alcanzables sin colisiones. En segundo lugar, se imprimieron prototipos parciales (específicamente del módulo ocular y elementos de transmisión) para comprobar tolerancias, holguras y facilidad de ensamblaje. Por último, los criterios de aceptación para el diseño se basaron en: (i) la posibilidad de movimiento repetido sin atascos, (ii) suficiente rigidez en los puntos de anclaje, (iii) acceso para fines de mantenimiento (reemplazo de servos y guía de cables), y (iv) la consistencia estética para la demostración pública; un acabado visual consistente con el personaje y adecuado para el objetivo de difusión

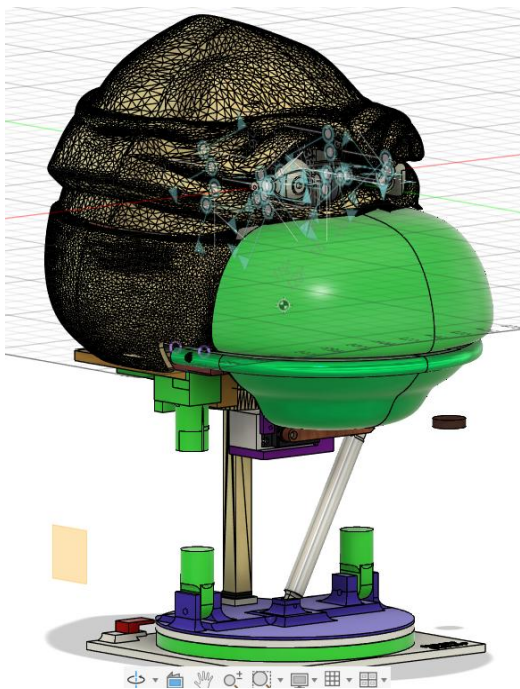


Figura 54. *Diseño completo en Fusion360 del animatrónico.*



Figura 55. *Animatrónico impreso.*

En síntesis, la versión definitiva de la carcasa animatrónica cumple con los objetivos de diseño planteados: representa fielmente al personaje en sus dimensiones y estética, Figuras 54 y 55, a la vez que alberga eficientemente los componentes mecatrónicos que le dan vida (servomotores, cables, iluminación). Las decisiones de rediseño desde descartar mecanismos complicados por otros más simples, hasta reforzar piezas críticas fueron tomadas para optimizar la funcionalidad y confiabilidad del sistema. De este modo, el Capítulo 2 ha documentado el ciclo completo de diseño, procesamiento e impresión de la carcasa animatrónica, evidenciando cómo a través de la iteración y la integración de diversas

herramientas (CAD, Meshmixer, slicer) se logró materializar con éxito la visión conceptual de un personaje animado dotado de movimiento físico. En capítulos posteriores se detallará la arquitectura e implementación de la electrónica del proyecto, también se verá la incorporación del sistema de control por voz e Inteligencia Artificial que complementa a esta estructura, completando así el desarrollo integral del prototipo animatrónico.

Capítulo III

Arquitectura e implementación del sistema electrónico del animatrónico

Este capítulo proporciona una arquitectura detallada del sistema electrónico del prototipo desde un enfoque integral, incluyendo la revisión de los principios técnicos necesarios para interpretar su funcionamiento, y la descripción de su implementación en el animatrónico. El objetivo es establecer una base clara que explique cómo se articulan prácticamente los subsistemas de control y potencia, sin reducir el análisis a una simple lista de componentes. Como consecuencia, la lógica organizativa del sistema puede mantener un rendimiento estable bajo condiciones de operación reales que implican demandas transitorias, movimientos simultáneos y sensibilidad a perturbaciones eléctricas, asegurando así la fiabilidad general en pruebas y demostraciones.

3.1 Servomotores en sistemas animatrónicos

En el sistema animatrónico orientado a la demostración y difusión pública, la parte principal de actuación para producir gestos controlados con repetibilidad son los servomotores. A diferencia de un actuador puramente "encendido/apagado", el servomotor de uso recreativo utiliza un motor de corriente continua, un tren de engranajes, un sensor de posición (generalmente un potenciómetro) y electrónica de control en bucle cerrado para controlar el actuador hacia un punto angular establecido (esta es la arquitectura de un solo servomotor). Este marco facilita movimientos expresivos como la apertura/cierre de párpados, el movimiento ocular, la rotación del mecanismo, la apertura adecuada de la mandíbula, y todo con un compromiso adecuado entre costo, compacidad e integración sencilla.

En cuanto al diseño, un servomotor no se considera simplemente como "un motor angular", sino como una estructura electromecánica cuya verdadera capacidad depende directamente de su par disponible, velocidad, rigidez de transmisión (engranajes), la calidad de la regulación interna y, sobre todo, el suministro eléctrico bajo carga [30]. En la animatrónica, con movimientos rápidos (parpadeos), movimientos cíclicos (barridos oculares) y movimientos de alta demanda mecánica (cuello), la elección de los servos debe justificarse por la función, el régimen de trabajo y el margen de seguridad contra picos de corriente (condiciones cercanas al bloqueo), también cabe recalcar que los servomotores de engranaje metálico son lo más adecuado para estos proyectos.

3.2 Principio de control por PWM y lazo de posición

El motor servo RC generalmente se controla utilizando una señal PWM interpretada en comandos de posición. En los servos analógicos clásicos, la referencia se codifica como un pulso dentro de un período cercano a 20 ms (≈ 50 Hz), el ancho del pulso define la posición deseada. Aunque los parámetros típicos del sistema son generalmente 'estándar' (aproximadamente ~ 1.0 – 2.0 ms), en realidad el rango útil está determinado por el servo y el grado de calibración para evitar topes mecánicos o sobreesfuerzo. Por lo tanto, es técnicamente válido aplicar un valor conservador para el rango de pulsos y los límites angulares cuando los servos trabajan acoplados a mecanismos impresos en 3D que tienen varias tolerancias y rigidez. El motor servo funciona internamente creando un bucle cerrado: compara la posición medida con el punto de ajuste obtenido del PWM para corregirlo en uso con el motor y el engranaje. Por lo tanto, en caso de una falla mecánica o una carga externa, el servo se activa para trabajar más y mantener su posición, lo que se manifiesta como un aumento en la corriente. La relación directa entre carga-corrección-corriente es la razón por la cual, en el dimensionamiento eléctrico del animatrónico, los valores críticos no se limitan a los consumos "sin carga", sino también a los consumos causados por aceleraciones, bloqueos parciales, alta fricción o condiciones de par máximo cercano.

3.3 Parámetros de selección relevantes para animatrónica

Para un diseño animatrónico funcional y mantenible, los parámetros más relevantes de un servomotor son el par nominal (y par de bloqueo), la velocidad, el tipo de engranajes (plástico vs metálico), el rango angular disponible y la robustez térmica ante uso repetitivo. En particular, el par especificado por el fabricante suele expresarse en kg·cm y debe convertirse conceptualmente a N·m al momento de justificar esfuerzos y márgenes; sin embargo, en animatrónica el par no se interpreta aislado, ya que el mecanismo puede incorporar ventaja mecánica (brazos, bielas, manivelas) o elementos de compensación (resortes) que reducen el par neto exigido al servo.

Adicionalmente, la selección debe considerar el régimen de operación: servos de pestañeo trabajan con ciclos breves y picos cortos, mientras que servos de barrido ocular o rotación continua trabajan con demanda repetitiva sostenida. Esta distinción es importante porque, aun con el mismo modelo de servo, el calentamiento y la probabilidad de inestabilidad eléctrica dependen del *duty cycle* real. Por lo tanto, en un animatrónico multi servo, la ingeniería del

sistema debe contemplar simultaneidad de movimientos, picos de corriente, desacoples mecánicos y una alimentación con margen.

3.4 Servomotores empleados en el prototipo y justificación funcional

En el prototipo se emplean servomotores de la familia “micro” y “estándar” de engranajes metálicos para cubrir funciones con exigencias diferenciadas. Para movimientos de bajo a medio esfuerzo, donde se prioriza compacidad y respuesta rápida, se utilizan servomotores tipo micro como el MG90S, ampliamente usados por su tamaño reducido y tren de engranajes metálicos, adecuados para ciclos repetitivos como pestañeos y movimientos secundarios. Sus especificaciones típicas reportan par del orden de unos pocos kg·cm, suficiente para mecanismos livianos cuando la geometría del eslabonamiento reduce la carga efectiva sobre el eje del servo.

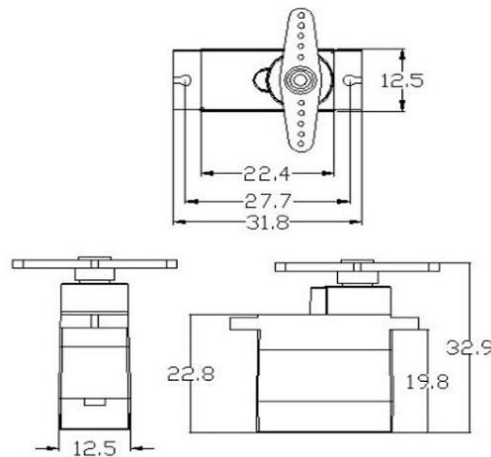


Figura 56. Dimensiones del MG90S según su DataSheet.

Para funciones de mayor exigencia mecánica se incorporan servos de mayor tamaño y par, como MG995/MG996R, apropiados cuando la carga incluye inercia y esfuerzos más altos (por ejemplo, accionamiento de mandíbula o mecanismos con mayores brazos de palanca). En estos casos, además del par, resulta crítico considerar corrientes elevadas bajo carga y cercanas al bloqueo; por ejemplo, para MG996R se reportan valores de corriente del orden de amperios en condiciones exigentes, lo cual refuerza la necesidad de una fuente dedicada y cableado con caída de tensión controlada. [32]

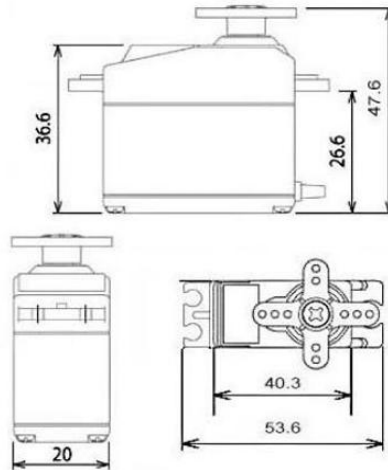


Figura 57. Dimensiones del MG996R según su DataSheet.

En síntesis, la elección de servomotores en el prototipo no responde únicamente a disponibilidad, sino a una asignación por función: servos micro para gestos rápidos y repetitivos de baja carga, y servos de alto par para movimientos donde el esfuerzo mecánico es más significativo o donde se requiere mayor holgura operacional frente a fricción, desalineaciones o variaciones de montaje.

3.5 Control multi servo con PCA9685 y calibración por ancho de pulso

Cuando se requiere controlar múltiples servomotores desde una microcomputadora, resulta conveniente emplear un controlador PWM dedicado, debido a que la temporización precisa de múltiples señales puede verse afectada si se intenta generar PWM por software en paralelo con tareas de procesamiento (p. ej., síntesis de voz, ejecución de modelos ligeros, o comunicación I²C). En este proyecto se utiliza el controlador PCA9685, un driver PWM de 16 canales con resolución de 12 bits, controlado por bus I²C, que permite generar señales PWM estables y descargadas del CPU principal.

A nivel de integración, es importante separar conceptualmente la alimentación de lógica (VCC) de la alimentación de potencia para servos (V+). El PCA9685 está diseñado para que la lógica opere a bajo voltaje, mientras que la etapa de potencia que alimenta los servos se inyecta externamente en la bornera V+ y GND, compartiendo referencia de tierra con la microcomputadora para mantener coherencia eléctrica del PWM. Esta separación es relevante porque evita que las corrientes pulsantes de los servomotores circulen por el riel de 5 V de la microcomputadora, reduciendo el riesgo de reinicios o inestabilidad por caídas transitorias de voltaje [33].

En cuanto a calibración, bibliotecas como ServoKit (CircuitPython) permiten definir el rango efectivo de pulsos para mapear ángulos con seguridad mecánica. Ajustar explícitamente el rango de microsegundos por canal evita que el servo intente sobrepasar límites físicos del mecanismo, lo cual disminuiría vida útil, aumentaría consumo y elevaría probabilidad de calentamiento.

3.6 Implicaciones eléctricas: picos de corriente, caída de tensión y robustez del sistema

En los servos, el consumo eléctrico dependiente de la carga es muy diferente. El movimiento sin carga generalmente permite una corriente moderada; con un esfuerzo creciente, se requieren corrientes altas por parte del servo para mantener el par y compensar las perturbaciones. El escenario más importante es alrededor del bloqueo, el punto de picos de corriente alta. En servos de alto par, como el MG996R, se reportan corrientes de alrededor de 2.5 A a 6 V en configuraciones muy exigentes, lo que justifica el uso de una fuente de alimentación externa, cableado de baja resistencia y una distribución en "estrella" para reducir la caída de voltaje y la interacción de ruido entre canales.

El microordenador (Raspberry Pi 4 Modelo B) también se mantiene alimentado por una fuente independiente, consistente con las políticas de suministro normales para una operación estable, manteniendo la fuente de alimentación y el bus del servo diferentes. Esto disminuye el acoplamiento entre los máximos de corriente del actuador y la electrónica de control, aumentando la fiabilidad general del animatrónico en demostraciones públicas [34].

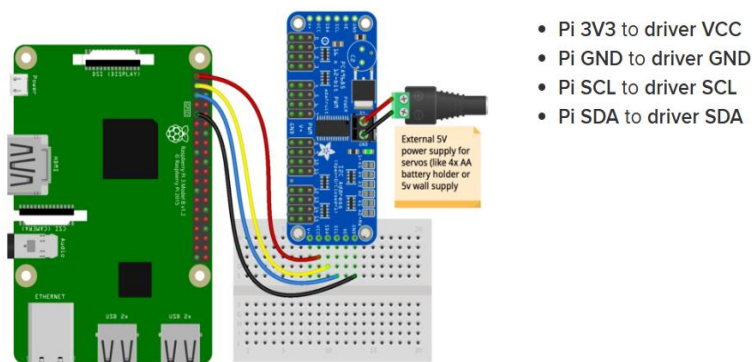


Figura 58. Raspberry Pi conectada con I2C

3.7 Arquitectura de alimentación y distribución de potencia del prototipo

El sistema electrónico del animatrónico integra a una microcomputadora (Raspberry Pi 4B), un controlador PWM por bus I²C (PCA9685) y nueve actuadores rotacionales (servomotores). Desde la perspectiva de ingeniería eléctrica, el desafío principal no es únicamente “entregar 5 V”, sino sostener un bus de potencia estable frente a una carga altamente dinámica: los servos

demandan corrientes transitorias elevadas durante arranques, cambios bruscos de consigna, inversión de sentido y eventos de fricción mecánica. En este tipo de arquitectura, una caída momentánea de tensión puede traducirse en pérdida de torque, vibración, jitter, e incluso reinicios del sistema de control si el riel de cómputo queda acoplado al riel de potencia. Por lo anterior, se adoptó una arquitectura de alimentación desacoplada por función, separando el riel de cómputo del riel de actuadores, y concentrando el análisis en el segundo, dado que domina la demanda instantánea de corriente.

3.7.1. Separación de rieles: cómputo independiente y potencia externa para servos

La Raspberry Pi 4B se alimenta mediante su fuente USB-C dedicada, conforme a la recomendación de 5 V con capacidad mínima de 3 A [35]. En paralelo, el conjunto de servomotores se alimenta externamente a través del bus V+ del módulo PCA9685, manteniendo referencia común de tierra (GND) para garantizar integridad de señal. Esta configuración se fundamenta en la topología del propio módulo: el pin VCC alimenta únicamente la lógica del controlador, mientras que el bus V+ debe ser alimentado aparte cuando se conectan servos. Este enfoque reduce el riesgo de que los picos de corriente de los actuadores induzcan caída de tensión en el riel de cómputo, incrementando la confiabilidad del sistema en operación repetitiva y durante demostraciones públicas.

3.7.2. Fuente conmutada para bus de actuadores ($V+ = 5.15 V$)

Para el bus de actuadores se empleó una Fuente conmutada industrial de 5 V / 10 A, ajustada a 5.15 V para compensar caídas resistivas en el caso que se den. Este tipo de fuente ofrece una reserva de corriente adecuada para cargas con picos y un comportamiento más estable frente a demanda simultánea de múltiples servos, característica crítica en prototipos animatrónicos donde varios ejes pueden moverse de forma concurrente [36].

Este margen controlado es útil en buses de servo, donde la resistencia de conductores, borneras y pistas puede generar pérdidas en escenarios de carga simultánea. Las fuentes industriales de 50 W para 5 V / 10 A reportan rizado bajo (del orden de decenas de mV) y especificaciones de regulación adecuadas para cargas pulsantes. Además, las Fuentes conmutadas son una mejor opción en cuanto a las fuentes lineales que tienen una gran pérdida de energía, una parte que se pierde por el transformador y otra por los reguladores de tensión, esto no pasa con las fuentes conmutadas. En términos de seguridad y robustez, la elección de una fuente de clase industrial facilita operar con margen ante sobrecorrientes transitorias y reduce el riesgo de colapso del bus durante acciones combinadas (p. ej., inclinación de cuello + pestañeo + habla).

En pruebas de laboratorio, se verificó que la fuente S-50-5 mantiene 5.1 V en bornes bajo cargas cercanas a 4 A (emulando movimientos simultáneos de los servos con la Pi en procesamiento intenso), sin caídas significativas ni sobrecalentamiento apreciable. De esta manera, el sistema de alimentación dimensionado adecuadamente constituye un pilar fundamental para el correcto funcionamiento y la confiabilidad a largo plazo del animatrónico, protegiendo la inversión en los componentes y ofreciendo un margen de operación holgado que contribuye al éxito global del proyecto, la fuente de poder se puede observar en la Figura 59.

Tabla 1. Especificaciones de 50W Single Output Switching Power Supply.

Modelo	S-50-5
SALIDA (OUTPUT)	
Voltaje DC	5 V
Corriente nominal	10 A
Rango de corriente	0 – 10 A
Potencia nominal	50 W
Rizado y ruido (máx.)	75 mVp-p
Rango de ajuste de voltaje	4,75 – 5,5 V
Tolerancia de voltaje	± 2,0 %
Regulación de línea	± 0,5 %
Regulación de carga	± 1,0 %
Tiempo de arranque / subida / mantenimiento	200 ms / 100 ms / 20 ms (a carga completa)
ENTRADA (INPUT)	
Rango de voltaje	85 – 132 VAC / 170 – 264 VAC (seleccionable por switch) o 240 – 370 VDC
Rango de frecuencia	47 – 63 Hz
Eficiencia (típica)	71 %
Corriente AC	1,3 A / 115 VAC; 0,65 A / 230 VAC
Corriente de irrupción (máx.)	Arranque en frío: 18 A / 115 VAC; 36 A / 230 VAC

Corriente de fuga < 1 mA / 240 VAC

PROTECCIÓN (PROTECTION)

Sobrecarga	105 – 150 % de la potencia nominal de salida
Tipo de protección	Limitación de corriente tipo <i>foldback</i> ; recuperación automática al eliminar la condición de falla

Desde el punto de vista de conversión de energía, la S-50-5 pertenece a la familia de fuentes conmutadas (SMPS). A diferencia de una fuente lineal (transformador 50/60 Hz + rectificación + regulación disipativa), una SMPS rectifica primero la red y luego conmuta la energía a alta frecuencia para transferirla a través de un transformador compacto. Esta estrategia mejora la eficiencia y reduce volumen y masa, lo cual es especialmente valioso en un animatrónico donde se prioriza la compacidad y la integración mecánica [37].



Figura 59. Vista dercha S-50-5.

En una implementación típica de esta clase de fuentes, el frente de entrada incorpora un bloque de protección y compatibilidad electromagnética (EMC/EMI): fusible, limitación de corriente de arranque (NTC) y un filtro EMI formado por inductancias (p. ej., choque de modo común) y capacitores X/Y. Su función es doble: es atenuar el ruido conducido que la propia conmutación genera hacia la red y establecer caminos controlados de retorno para corrientes de modo común, reduciendo la probabilidad de interferencia sobre otros equipos y sobre la propia electrónica del sistema. El rol de los capacitores tipo Y en el retorno de corrientes de modo común y el diseño de filtros de entrada en convertidores conmutados se discute explícitamente en guías de baja EMI para reguladores/conversores DC/DC. [38]

Luego del filtro de entrada, la energía se procesa mediante un puente rectificador (rectificación de onda completa), obteniendo un bus DC de alto voltaje, el cual se estabiliza con un condensador electrolítico primario (bulk capacitor). Este capacitor cumple una función crítica: almacenar energía para cubrir los intervalos en los que la red instantáneamente entrega menos

potencia (por la forma senoidal) y, además, reducir el rizado del bus DC antes de la etapa de conmutación.

La conversión principal se realiza mediante un interruptor de alta frecuencia, típicamente un MOSFET controlado por un modulador PWM. Al conmutar a decenas de kHz (o más), el transformador puede operar con un flujo magnético alternante mucho más rápido que en 50/60 Hz. Esto explica por qué el transformador de una SMPS es significativamente más pequeño que el de una fuente lineal de potencia equivalente: para una tensión dada, el área efectiva del núcleo y el número de espiras requeridas disminuyen cuando aumenta la frecuencia de excitación (base de los criterios de diseño magnético usados en fuentes conmutadas). Este principio está tratado en material clásico de diseño de magnéticos para convertidores conmutados. [39]

En el secundario del transformador, la energía se reconvierte a DC mediante diodos rectificadores rápidos (frecuentemente Schottky en salidas de 5 V por su baja caída directa y velocidad), y se filtra con una red de salida (capacitores y, según diseño, elementos inductivos) para reducir rizado y mejorar la respuesta a transitorios. El lazo de regulación se cierra a través de realimentación aislada (por ejemplo, combinación TL431 + optoacoplador), que permite mantener el aislamiento galvánico entre primario y secundario mientras se regula con precisión la tensión de salida frente a variaciones de carga; este esquema es ampliamente documentado en notas técnicas de diseño de realimentación aislada. [40]

Finalmente, una característica relevante para uso mecatrónico es la presencia de protecciones internas típicas de fuentes industriales: sobrecorriente/sobrecarga, cortocircuito y, dependiendo del diseño, sobretensión y sobretensión. Estas protecciones son especialmente pertinentes en animatrónica porque un atasco mecánico o un golpe puede llevar un servo a condiciones cercanas a “stall”, elevando la corriente de manera abrupta. Al dimensionar la alimentación se busca, por tanto, no solo la potencia nominal, sino también un comportamiento seguro ante fallos y transitorios.

3.7.3. Comparación técnica con alternativa portátil: baterías + LM2596

Como alternativa de portabilidad, una solución basada en baterías Li-ion (3.7 V) con conversión mediante LM2596 es factible en prototipos ligeros; sin embargo, el LM2596 es un convertidor buck especificado para 3 A nominales, por lo que su aplicación en un bus de servos requiere un análisis más estricto de margen térmico, filtrado, y simultaneidad de cargas [41].

En el contexto del presente prototipo orientado a demostraciones públicas continuas se priorizó estabilidad del bus y tolerancia a picos por encima de la portabilidad, justificando la migración hacia una fuente de 10 A.

En una etapa inicial del proyecto se implementó una solución de alimentación basada en dos celdas Li-ion de 3.7 V conectadas en serie, con una capacidad nominal de 12 000 mAh, cuyo voltaje fue regulado mediante un convertidor reductor (buck) LM2596. La salida regulada del LM2596 se distribuyó hacia una protoboard utilizada como nodo de interconexión, desde la cual se derivaron las líneas de alimentación positiva y negativa hacia el servomotor en prueba. En paralelo, la señal de control PWM se generó desde el Arduino Mega, conectándose directamente al conductor de señal del servomotor, tal como se observa en la configuración experimental mostrada en la Figura correspondiente. Este arreglo permitió validar de manera rápida el funcionamiento básico del actuador y su respuesta ante comandos de posicionamiento, con un montaje flexible y reconfigurable propio de etapas de prototipado temprano.

La elección de esta arquitectura se justificó porque, en el momento de su implementación, el sistema aún se encontraba en fase exploratoria y no se preveía el uso simultáneo de múltiples servomotores. Conforme el prototipo evolucionó hacia un animatrónico con mayor número de grados de libertad, se evidenció la necesidad de incrementar la cantidad de servomotores y, en consecuencia, elevar los requerimientos del subsistema eléctrico. En particular, se identificó que el crecimiento en número de actuadores y la ocurrencia de demandas transitorias (arranques, cambios bruscos de posición y cargas mecánicas variables) podían inducir caídas de tensión y pérdidas de desempeño si la alimentación no se dimensionaba adecuadamente. Por ello, el proyecto transitó desde esta solución inicial basada en baterías y protoboard (Figura 60), hacia una arquitectura de alimentación más robusta, orientada a mejorar la estabilidad del bus de potencia de servos, reducir el riesgo de subtensión y sostener la operación simultánea de varios actuadores con mayor confiabilidad.

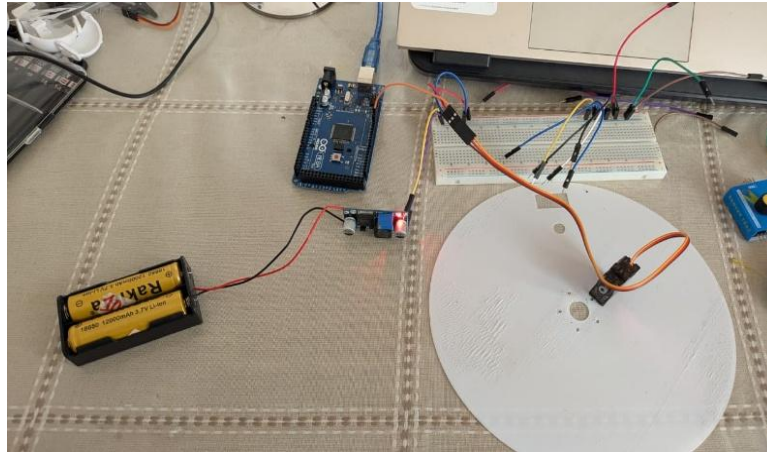


Figura 60. Montaje inicial: Li-ion + LM2596 + Arduino Mega + servomotor.

3.7.4 Regímenes de consumo según simultaneidad real del software de control

El perfil de corriente no depende únicamente de la cantidad total de servos instalados, sino de cuántos se desplazan simultáneamente bajo el programa de control. En `animatronic_loopp_all.py` se definen dos comportamientos superpuestos: un movimiento continuo por pasos en CH4, CH5, CH8 y CH15, y un pestañeo periódico de CH0–CH3, con un tiempo de permanencia mayor en ojos abiertos que en transición de cierre/apertura. En consecuencia, el consumo puede agruparse en tres regímenes operativos:

- **Bajo consumo (ojos abiertos):** se mueven simultáneamente 4 servos (CH4, CH5, CH8, CH15) mientras CH0–CH3 permanecen en hold.
- **Continuo (evento de pestañeo dentro del loop):** durante el intervalo de pestañeo, se mueven simultáneamente 8 servos (CH0–CH5, CH8, CH15).
- **Intermitente (habla):** además del régimen anterior, se activa CH9 (MG996R) únicamente cuando el sistema genera voz y sincroniza mandíbula, por lo que su “duty” es bajo en comparación con el resto.

Datos de referencia de catálogo (picos): MG90S ~0.75 A de stall; MG996R ~2.5 A de stall y ~0.5 A en operación; MG995 reporta corrientes en el orden de 0.5–0.9 A en operación y stall ~2.5 A (según fichas comerciales [30], [32]) Cabe recalcar que nunca se ha llegado a “I total_pico”. El criterio usado para “típico” fue emplear valores de operación moderada acordes a tu montaje (MG995 es el más cargado pero asistido por resortes; MG996R trabaja ~1 kg y solo durante habla). Los “picos” se mantienen como envoltente conservadora.

El sistema opera la mayor parte del tiempo entre bajo consumo y continuo, y entra a intermitente únicamente cuando hay habla (CH9). Por ello, aun cuando el pico teórico es alto, el consumo promedio real se mantiene dentro del margen de la fuente seleccionada.

Tabla 2. Corriente total estimada por regímenes.

Régimen	Servos que se mueven a la vez (según <code>animatronic_loop_all.py</code>)	Servos en hold	I _{total} mín (A)	I _{total} típ (A)	I _{total} pico (A)	P _{total} típ @ 5.15 V (W)
Bajo consumo (ojos abiertos, sin pestañeo)	CH4, CH5, CH15 (MG90S) + CH9 (MG995) → 4 moviéndose	CH0–CH3 (MG90S) en POS_FIN	0.88	2.07	5.55	10.66
Continuo (evento de pestañeo dentro del loop)	CH0–CH3 (MG90S) + CH4, CH5, CH15 (MG90S) + CH9 (MG995) → 8 moviéndose	—	1.24	2.75	7.75	14.16
Intermitente (habla: mandíbula activa)	Lo anterior + CH8 (MG996R) → 9 moviéndose	—	1.54	3.25	10.00	18.03

3.7.5. Implicaciones de diseño: margen, estabilidad y confiabilidad del bus

Con base en la Tabla 2, el dimensionamiento con una fuente de 10 A para V+ proporciona un margen suficiente para cubrir: operación típica sostenida sin caída apreciable de tensión; transitorios simultáneos durante pestañeo + inclinación; y eventos intermitentes de habla con mandíbula. Este margen es especialmente relevante en servomotores, donde corrientes instantáneas elevadas pueden ocurrir incluso sin llegar al bloqueo total, por efectos combinados de inercia, fricción y esfuerzos en los extremos del recorrido.

Adicionalmente, el ajuste del bus a 5.15 V permite amortiguar pérdidas resistivas en el cableado y distribución, manteniendo el riel dentro de rangos seguros para servos estándar (típicamente 4.8–6.0 V en MG90S; 4.8–7.2 V en MG995/MG996R). En síntesis, la arquitectura desacoplada (Pi independiente + V+ externo) y el sobredimensionamiento moderado del bus de servos constituyen un pilar de confiabilidad para el prototipo.

Desde una perspectiva de integración mecatrónica, esta separación de alimentación reduce el riesgo de fallas cruzadas: el bus de servos es una carga altamente dinámica (conmutación interna de los servos, variaciones bruscas de corriente y ruido conducido), mientras que la Raspberry Pi es sensible a subtensiones y a perturbaciones transitorias. Al mantener la Pi fuera del bus V+, se evita que caídas momentáneas o ruido de alta frecuencia inducidos por los actuadores provoquen reinicios, degradación de comunicación o inestabilidad del sistema de control. No obstante, para conservar referencia común de señales, se estableció una masa compartida (GND común) entre el PCA9685 y la Raspberry Pi, condición necesaria para la integridad del enlace I²C y de la lógica de control, manteniendo al mismo tiempo la potencia de actuadores fuera del dominio de alimentación de la SBC.

En términos de estabilidad del bus, el margen de corriente disponible se complementa con decisiones de cableado y distribución orientadas a minimizar pérdidas y acoplamientos no deseados: recorridos de potencia cortos hacia la bornera V+/GND del PCA9685, reducción de resistencias de contacto, y una distribución que evita “cadenas” largas donde la caída de tensión se acumule servo tras servo. Este criterio es especialmente importante en regímenes con varios canales en movimiento, donde la corriente efectiva se incrementa y se vuelve más probable observar síntomas típicos de un bus subdimensionado (jitter en servos, pérdida de posición bajo carga o fallas esporádicas por caída local de tensión). En consecuencia, el diseño del bus se trató como un subsistema crítico y no como un detalle accesorio del montaje.

Finalmente, se concluye que la arquitectura eléctrica implementada prioriza margen, robustez y seguridad operativa para demostraciones públicas: el sistema mantiene comportamiento estable en la operación continua de los servos de orientación, absorbe los transitorios propios de movimientos rápidos (pestañeo) y admite eventos intermitentes de mayor demanda (mandíbula durante habla) sin comprometer la electrónica de control. Este enfoque, junto con la modularidad del cableado y el uso de un controlador dedicado (PCA9685), establece una base sólida para las siguientes etapas del proyecto: expansión controlada del número de actuadores, incorporación de sensores y, de ser necesario, mejoras de filtrado/ordenamiento del arnés para reducir aún más susceptibilidad a interferencias y facilitar mantenimiento. En este capítulo se presentó la implementación electrónica del animatrónico y, de manera central, la arquitectura de alimentación adoptada para garantizar operación confiable en un sistema con múltiples actuadores. Se justificó el uso de un bus dedicado de potencia para servos

(V+ externo a la bornera del PCA9685) y la alimentación independiente de la Raspberry Pi, estableciendo un desacoplamiento funcional que protege la unidad de control frente a perturbaciones típicas de cargas electromecánicas. Asimismo, el análisis de regímenes de operación permitió dimensionar el suministro con holgura razonable, sustentando la decisión de emplear una fuente de alta capacidad como base del sistema. Como resultado, la plataforma electrónica queda preparada para soportar operación repetitiva en contextos de exhibición, manteniendo estabilidad eléctrica y reduciendo la probabilidad de fallas por subtensión o transitorios, lo cual es coherente con el objetivo del prototipo: demostraciones robustas, seguras y mantenibles en escenarios de divulgación académica.

Capítulo IV

Implementación de un Sistema Animatrónico Controlado por Voz mediante IA en Raspberry Pi 4B

Este capítulo examina los fundamentos teóricos y tecnológicos en los cuales se apoya el desarrollo del prototipo animatrónico. Se describen inicialmente los criterios de selección del hardware principal, específicamente la plataforma de procesamiento Raspberry Pi 4B, considerando sus capacidades de cómputo y ventajas para la aplicación propuesta. A continuación, se detalla la implementación del sistema de voz embebido, que incluye el reconocimiento automático del habla, la generación de respuestas mediante modelos de lenguaje (LLM) y la síntesis de voz, integrados en la Raspberry Pi. Se discuten las optimizaciones realizadas en el modelo de lenguaje empleado (cambio de DeepSeek 1.5B a LLaMA 3.2:1B) para mejorar el rendimiento dentro de las limitaciones de hardware, así como los desafíos técnicos afrontados durante este proceso. Posteriormente, se describe el funcionamiento del script principal desarrollado (*hablarr_ia.py*), módulo por módulo, ilustrando el flujo completo de la interacción por voz. Finalmente, se presenta una extensión añadida al sistema: la sincronización del movimiento mandibular del animatrónico con el habla sintetizada, y se resume el resultado final del prototipo operando de forma autónoma.

4.1 Elección del Hardware: Raspberry Pi 4B

Para el hardware base se seleccionó una Raspberry Pi 4 Modelo B con 4 GB de RAM, debido a su equilibrio entre capacidad de cómputo, consumo energético moderado y disponibilidad de interfaces para integrar audio e I²C en un sistema embebido. Este modelo integra un procesador Broadcom BCM2711 de cuatro núcleos Cortex-A72 a 1.8 GHz (64 bits), acompañado de 4 GB LPDDR4 [42], lo cual permite ejecutar un *pipeline* de voz completo (STT→LLM→TTS) con inferencia local y tiempos de respuesta aceptables para interacción humano-máquina.

Esta elección se justifica en el enfoque de Edge AI (IA en el borde): el procesamiento se realiza en el dispositivo, reduciendo dependencia de conectividad, aumentando privacidad y disminuyendo latencia asociada a redes. En sistemas animatrónicos, donde la interacción debe percibirse fluida y “en vivo”, la latencia total de conversación (captura de voz + transcripción + generación + síntesis + reproducción) es un parámetro crítico. En ese sentido, la Pi 4B ofrece suficiente potencia para tareas de cómputo intensivo moderado sin requerir una computadora

de escritorio, manteniendo un consumo máximo del orden de ~ 7.6 W (1.5 A a 5 V) [43], adecuado para una plataforma portátil o integrada en un montaje de animatrónica.

Asimismo, la conectividad de la Raspberry Pi 4B fue determinante. Los puertos USB permiten integrar micrófonos o tarjetas de sonido externas (si se requiere mayor calidad), y el encabezado GPIO de 40 pines incorpora buses I²C, indispensables para la comunicación con el controlador PCA9685. A nivel teórico, el PCA9685 actúa como un generador PWM multicanal independiente del microprocesador, reduciendo carga de tiempo real en la CPU y estandarizando el control de servomotores con frecuencia típica de 50 Hz. Esta separación de funciones (cómputo de IA por CPU y control PWM por periférico dedicado vía I²C) es coherente con buenas prácticas de ingeniería en sistemas embebidos: minimizar *jitter* y desacoplar tareas de control físico de procesos de alto cómputo (ASR/LLM/TTS), que pueden introducir variabilidad temporal.

4.2 Implementación Inicial: DeepSeek 1.5B en CPU y *Pipeline* de Voz Integrado

En la etapa inicial se implementó el sistema utilizando un modelo de lenguaje grande (LLM) DeepSeek-R1 de 1.5 B parámetros, ejecutado localmente en la Raspberry Pi mediante Ollama. Este modelo, con un tamaño aproximado de 1.1 GB en disco [44], se consideró viable dentro de las restricciones de memoria y CPU de la plataforma, permitiendo generar respuestas conversacionales sin depender de servicios externos. Teóricamente, un LLM funciona como un modelo autoregresivo que estima distribuciones de probabilidad de secuencias de tokens, produciendo texto por muestreo/decodificación a partir de un *prompt*. En un sistema embebido, la viabilidad práctica del LLM depende de la memoria necesaria para cargar pesos (incluyendo cuantización), el costo computacional por token y la latencia tolerable en turnos conversacionales.

A partir de este núcleo, se integró un *pipeline* completo de voz (STT→LLM→TTS) para lograr interacción natural con el usuario, en donde cada módulo se diseñó como una etapa claramente separada y verificable.

4.2.1 Captura de voz (*Adquisición de audio*)

La captura de voz constituye la primera etapa del sistema y, en términos de teoría de procesamiento digital de señales (DSP), corresponde al proceso de adquisición de una señal acústica analógica (presión sonora) y su conversión a una representación digital adecuada para etapas posteriores de reconocimiento automático del habla. En un sistema embebido como la

Raspberry Pi 4B, la adquisición debe equilibrar calidad, robustez y costo computacional. Por ello, la captura se implementó mediante el subsistema ALSA (Advanced Linux Sound Architecture) utilizando la utilidad *arecord*, la cual permite controlar parámetros de muestreo y formato del archivo de salida. La señal se almacenó en formato WAV PCM lineal, lo cual resulta ventajoso para *pipelines* de voz debido a que evita pérdidas por compresión y preserva la forma de onda necesaria para la extracción de características por el módulo ASR.

Un aspecto clave es la elección de parámetros de muestreo. Desde teoría de muestreo, la frecuencia seleccionada debe satisfacer el criterio de Nyquist respecto al contenido espectral del habla. En la práctica, el habla humana concentra la mayor parte de la información lingüística por debajo de ~ 8 kHz, por lo que trabajar a 16 kHz (Nyquist = 8 kHz) es una convención extendida en ASR: reduce tamaño de archivo y costo de procesamiento sin afectar significativamente la inteligibilidad para reconocimiento. Asimismo, se empleó cuantización típica de 16 bits, que ofrece un compromiso adecuado entre rango dinámico y tamaño de datos; valores menores elevan el ruido de cuantización y pueden degradar el reconocimiento, mientras que valores mayores aportan beneficios marginales a costa de mayor carga de E/S. La captura con *arecord* se estructuró como una ventana temporal fija (por ejemplo 6 s), lo cual simplifica el flujo de control y permite segmentación de turnos conversacionales; sin embargo, también introduce un costo: si el usuario habla menos, se procesa silencio adicional. En esta etapa, se priorizó simplicidad y confiabilidad del *pipeline* sobre optimizaciones avanzadas como detección automática de voz (VAD).

Desde el punto de vista de ingeniería, la captura también se diseñó considerando interferencias típicas en ambientes reales: ruido de fondo, reverberación, y variaciones en distancia al micrófono. Aunque el sistema no implementa aún cancelación de ruido o beamforming, la decisión de capturar audio en formato PCM lineal y con parámetros estándar permite incorporar posteriormente módulos de preprocesamiento (p. ej. filtrado, normalización RMS o VAD) sin reestructurar la arquitectura. En suma, la etapa de captura se implementó de forma deliberadamente robusta y estándar, asegurando una entrada de audio consistente y compatible con Whisper, que es sensible a degradaciones severas de señal, pero tolerante a variaciones moderadas cuando se mantienen parámetros de muestreo adecuados.

4.2.2 Transcripción (ASR / STT) con Whisper

La etapa de ASR (Automatic Speech Recognition) o STT (Speech-to-Text) se implementó con Whisper, ejecutado en la Raspberry Pi mediante su interfaz de línea de comandos. Whisper, en particular, adopta un enfoque de aprendizaje profundo donde el audio se transforma primero a una representación de espectrograma (p. ej. mel-espectrograma) y luego se decodifica a tokens de texto. Este paradigma tiene dos ventajas cruciales para el proyecto: alta robustez a variaciones de acento e incluso ruido moderado, y soporte multilingüe sin necesidad de modelos separados por idioma. Dado que el sistema corre en una Raspberry Pi 4B (CPU), se seleccionó el modelo Whisper Tiny, que representa el extremo liviano de la familia. Esta decisión se sustenta en una restricción central de la plataforma: la transcripción por ASR es una tarea intensiva en cómputo, y en CPU el tiempo de inferencia crece significativamente con modelos más grandes. Whisper Tiny reduce el costo computacional y mantiene la transcripción en escalas de tiempo compatibles con interacción conversacional, aceptando el compromiso de menor precisión en condiciones adversas [45]. En la práctica del sistema implementado, la transcripción se lanzó con parámetros explícitos: idioma español (language es), formato de salida texto (output_format txt) y un modo no verboso para simplificar el post-procesado. Esto es importante en ingeniería de sistemas: controlar el formato de salida de Whisper evita contaminar la entrada del LLM con logs, timestamps o metadatos.

Un aspecto técnico relevante de la implementación fue la gestión del entorno Python en sistemas Debian/Ubuntu modernos. Al intentar instalar Whisper con pip3 se presentó el error “externally-managed-environment” (PEP 668), el cual bloquea instalaciones directas de paquetes en el Python del sistema. Para mantener integridad del entorno y evitar “romper” dependencias del sistema operativo, se adoptó un enfoque aislado mediante herramientas tipo pipx, lo cual instala el paquete en un entorno virtual controlado y expone el binario whisper en el PATH del usuario. Esta decisión, aunque inicialmente fue un inconveniente, se alinea con buenas prácticas de ingeniería: reproducibilidad, menor riesgo de corrupción del sistema, y facilidad para desinstalar o actualizar componentes de IA sin afectar el entorno global.

Finalmente, el texto transcrito se utilizó como entrada directa al módulo de lenguaje. En este punto, la transcripción debe evaluarse como un *front-end* imperfecto: errores de STT se propagan a la conversación (fenómeno de *error cascading*). Por ello, el diseño del prompt y la estrategia de respuesta del LLM se orientaron a tolerar cierta ambigüedad: frases cortas, tono estable y evitación de respuestas excesivamente elaboradas cuando la entrada pudiera estar

truncada o ruidosa. Esta consideración, aunque práctica, es fundamental a nivel teórico en sistemas encadenados: maximizar calidad global requiere no solo optimizar cada módulo, sino diseñar la interacción entre módulos para ser robusta a fallos parciales.

4.2.3 Inferencia del modelo de lenguaje (NLU/LLM) mediante Ollama

La etapa de NLU/LLM constituye el núcleo cognitivo del sistema: recibe texto de entrada y produce una respuesta textual. En la implementación realizada, el LLM se ejecutó localmente vía Ollama, lo cual ofrece una capa de abstracción y despliegue: descarga y administra modelos, corre un servicio local y permite ejecutar inferencias desde scripts sin necesidad de compilar manualmente dependencias o gestionar motores como llama.cpp de forma explícita. En el sistema, la interacción con el LLM se implementó desde Python mediante `subprocess.run` llamando a `ollama run <modelo>`, y enviando el prompt por entrada estándar. Esta decisión técnica simplificó la integración: evita depender de librerías HTTP o SDKs y se apoya en una interfaz estable de terminal. El prompt se construyó con `textwrap.dedent`, lo cual permite formatearlo claramente, mantener reglas explícitas y evitar sangrías accidentales. A nivel metodológico, este prompt no solo incluye la pregunta del usuario, sino también restricciones operativas para controlar el comportamiento del modelo: responder en español, en una frase corta, sin razonamiento visible (“Thinking”), sin metatexto y sin inglés. Estas restricciones no son meramente estéticas; desde una perspectiva de ingeniería, reducen la entropía de la salida y mejoran la confiabilidad del TTS, ya que Piper sintetiza mejor textos cortos y limpios que párrafos largos con símbolos, listas o marcas extrañas.

Un detalle implementado de forma práctica fue el post-procesado de la salida del LLM. Se observó que algunos modelos podían emitir líneas adicionales, eco parcial del prompt o saltos de línea redundantes. Para estabilizar la salida, se implementó la función `ultima_linea_no_vacia()`, que recorre la salida del modelo desde el final y toma la última línea significativa como “frase final”. Este criterio es simple, pero efectivo en un sistema embebido: reduce el riesgo de que el TTS pronuncie fragmentos no deseados y sirve como mecanismo de saneamiento sintáctico. Además, se imprimió la salida completa del modelo en modo *debug* durante pruebas, lo cual permitió diagnosticar inconsistencias, latencias, o mensajes inesperados del motor. En términos de rendimiento, la inferencia en CPU en Raspberry Pi impone una restricción fuerte: el tiempo por token y el tamaño del modelo determinan la latencia. En la etapa inicial se empleó DeepSeek 1.5B por ser lo suficientemente pequeño para cargar en memoria, aunque con latencia perceptible. Posteriormente, el proyecto migró al

modelo final Llama 3.2:1b, pero ese cambio pertenece al bloque siguiente (4.3). Lo importante aquí es que la arquitectura del código se diseñó desde el inicio para ser modular: cambiar el modelo era un cambio de una sola variable (`MODELO_OLLAMA`), sin reescribir el pipeline [46]. Eso es, en sí mismo, una decisión de diseño de software orientada a escalabilidad: la lógica STT/LLM/TTS permanece idéntica, aunque el modelo interno cambie.

4.2.4 Síntesis de voz (TTS) con Piper y reproducción con ALSA

La etapa de Text-to-Speech (TTS) convierte el texto generado por el LLM en una señal de audio audible. Teóricamente, los sistemas TTS modernos suelen dividirse en dos componentes, un modelo lingüístico/acústico que convierte texto en características acústicas (como espectrogramas) y un vocoder que convierte esas características en forma de onda. Piper implementa un enfoque moderno y eficiente, diseñado específicamente para correr en CPU con modelos relativamente livianos, razón por la cual resulta apropiado para Raspberry Pi. El modelo de voz utilizado es `_ES-davefx-medium.onnx`, se seleccionó por ser un modelo en español con calidad media y tamaño manejable, lo que permite síntesis rápida sin saturar la memoria disponible.

En la implementación real, Piper se integró como un comando ejecutable `piper --model ... --output_file ...` recibiendo el texto por `stdin`. Este patrón es especialmente útil porque permite encadenar herramientas de terminal y mantener el pipeline simple y trazable: el script produce texto y lo entrega directamente a Piper; Piper genera un WAV; y finalmente se reproduce con `aplay`. El uso de WAV como formato intermedio tiene ventajas técnicas: es un contenedor estándar PCM sin compresión, reconocido por ALSA y fácil de inspeccionar, reproducir y medir (por ejemplo, para obtener duración del audio). Esto fue importante cuando posteriormente se sincronizó el movimiento mandibular, ya que se necesitó conocer la duración del WAV para coordinar actuadores con el tiempo de reproducción. [47]

Otro aspecto esencial fue la gestión de rutas y ejecutables en Linux. Parte del trabajo consistió en asegurar que Piper estuviera correctamente instalado y accesible en el entorno del usuario, resolviendo cuestiones asociadas a rutas tipo `~/local/bin` y garantizando que el shell encuentre el ejecutable sin tener que escribir rutas completas. Este ajuste, aunque operativo, es crítico en un sistema embebido: un pipeline puede fallar no por la teoría del modelo, sino porque el binario no se encuentra en el `PATH`. Las verificaciones con comandos como `which piper` y la

confirmación de funcionamiento mediante generación y reproducción de WAV fueron pasos de validación indispensables antes de conectar el TTS a la salida del LLM.

Finalmente, la reproducción con aplay se eligió por su simplicidad y compatibilidad directa con ALSA. Desde la ingeniería de sistemas, usar herramientas nativas reduce dependencias, minimiza riesgos de incompatibilidad y facilita depuración: si hay audio distorsionado, se puede aislar si proviene del WAV, de Piper o del dispositivo de salida. Con esto, el sistema logró cerrar el ciclo conversacional completo: texto generado por IA → voz sintética reproducible localmente [48]. Esta elección técnica es consistente con el enfoque general del proyecto: máxima autonomía local, componentes ligeros y controlables, y un pipeline verificable por etapas.

4.3 Optimización del Modelo LLM: Cambio de DeepSeek 1.5B a LLaMA 3.2:1B

Durante el desarrollo se identificó la oportunidad de mejorar el rendimiento del sistema sustituyendo el modelo de lenguaje. Finalmente se reemplazó DeepSeek 1.5B por LLaMA 3.2:1B (un modelo de ~1 mil millón de parámetros de la familia LLaMA 3.2 de Meta), logrando así menor uso de memoria, mayor velocidad de inferencia y mejores respuestas para nuestro caso de uso.

LLaMA 3.2:1B, pese a tener un conteo de parámetros ligeramente inferior (\approx 1 mil millón vs 1.5 mil millones), está optimizado en eficiencia y precisión, al provenir de una poda y destilación del modelo LLaMA original de 8B. En otras palabras, “no hay que dejarse engañar por su menor tamaño: LLaMA 3.2 de 1B está diseñada para ser eficiente y precisa” [4.6]. Este modelo logra respuestas de calidad comparable o superior a DeepSeek, pero con menor latencia y exigencias de hardware reducidas. De hecho, LLaMA 3.2:1B ofrece tiempos de inferencia más rápidos (debido a su menor tamaño, cargando y procesando más ágilmente los *prompts*) y puede ejecutarse cómodamente en CPU de la Raspberry Pi sin requerir GPU ni grandes cantidades de RAM. En pruebas realizadas, se observó que el modelo LLaMA respondía con mayor rapidez a preguntas generales y tendía a generar respuestas más concisas y relevantes, mejorando la experiencia conversacional.

Vale destacar que el modelo LLaMA 3.2:1B quedó como la solución final implementada en el sistema, ejecutándose localmente vía Ollama en la Raspberry Pi. No se utiliza ningún servicio externo ni modelos propietarios, asegurando que toda la inferencia ocurre en el dispositivo local. Esto refuerza la privacidad (no se envían datos de voz ni texto fuera) y la autonomía del

sistema, a costa únicamente de trabajar con un modelo de parámetros moderados. En síntesis, la elección de LLaMA 3.2:1B permitió mejorar el rendimiento (reduciendo tiempos de respuesta) y la eficiencia en memoria del asistente animatrónico, sin sacrificar la calidad de las respuestas generadas.

4.3.1 Desafíos Técnicos en la Transición de Modelos LLM

La sustitución del modelo de lenguaje no estuvo exenta de dificultades técnicas, las cuales se documentaron y resolvieron durante el proceso de implementación:

1. **Limitaciones de memoria con modelos mayores:** Al explorar alternativas, se intentó ejecutar modelos de mayor tamaño como *Phi-3 “mini”* (un modelo de ~3.8B parámetros de Microsoft) o LLaMA 3.2:3B. Sin embargo, la Raspberry Pi 4 de 4 GB de RAM resultó insuficiente para cargar estos modelos voluminosos. En las pruebas, la carga de Phi-3 mini (3.8B) mediante Ollama descargó aproximadamente 2.2 GB de datos del modelo, y al intentar inicializarlo se encontraron errores de memoria insuficiente o *swapping* extremo que hacía inviable su uso. De manera general, con solo 4 GB, la Pi 4 no puede alojar cómodamente modelos de varios billones de parámetros; estudios independientes confirman que incluso en una Pi 5 con 8 GB no es realista ejecutar modelos por encima de ~7B parámetros, por lo que en una Pi 4 (4 GB) el límite práctico es aún menor (alrededor de 3B). Esto explica por qué los modelos candidatos más grandes fueron descartados tras los primeros intentos fallidos, orientando la decisión hacia modelos más pequeños y eficientes como LLaMA 1B.
2. **Incompatibilidades de parámetros en Ollama:** Durante la transición se encontraron inconvenientes relacionados con la herramienta Ollama en distintas versiones. En particular, ciertas banderas de configuración ya no eran reconocidas o causaban comportamientos anómalos. Por ejemplo, la opción `quantize` (para especificar el nivel de cuantización al cargar un modelo) no fue aceptada en una versión más reciente de Ollama, dado que la cuantización apropiada ya venía predefinida con el modelo descargado. Del mismo modo, el parámetro `num_gpu_layers` (usado para controlar la carga de capas en GPU en *llama.cpp*) carecía de efecto en la Raspberry Pi al no disponer esta de acelerador GPU compatible, e incluso su presencia en comandos podía provocar advertencias. Estas diferencias de versión requirieron ajustar la forma de invocar los modelos: se eliminó el uso de *flags* obsoletas, dejando que Ollama gestione automáticamente la cuantización

óptima, y asegurando que la ejecución fuera totalmente en CPU (por defecto, Ollama detecta la ausencia de GPU y opera en modo CPU).

3. **Actualización del identificador de modelo en el código:** Tras descargar y probar exitosamente *llama3.2:1b* en Ollama, fue necesario modificar el script principal (*hablar_ia.py*) para que utilizara el nuevo modelo. En concreto, se cambió el parámetro o constante que define el modelo activo (por ejemplo, la variable `MODELO_OLLAMA`) para que apunte al nombre exacto "llama3.2:1b", en sustitución del identificador del modelo anterior DeepSeek. Este ajuste sencillo fue imprescindible para redirigir correctamente las peticiones de generación de texto al modelo LLaMA a través de la API local de Ollama. A partir de entonces, el sistema comenzó a usar LLaMA 1B por defecto en cada interacción.

Superados estos obstáculos, la transición de DeepSeek a LLaMA se completó con éxito. Se concluyó que LLaMA 3.2:1B era la opción más adecuada dados los recursos limitados de la Raspberry Pi y la necesidad de respuestas coherentes en tiempo razonable, mientras que los modelos más grandes no podían operar fiablemente en este entorno.

4.3.2 Descripción Detallada del Script `hablarr_ia.py`

El núcleo de la solución reside en el script Python *hablarr_ia.py*, desarrollado para orquestar todas las etapas del flujo de interacción por voz con el animatrónico. A continuación, se detalla su funcionamiento técnico por módulos, siguiendo el orden de ejecución:

1. **Inicialización y configuraciones:** El script comienza importando las librerías necesarias (por ejemplo, `subprocess` para ejecutar comandos del sistema, posiblemente `requests` para invocar la API de Ollama, y bibliotecas para controlar el PCA9685). Se definen parámetros globales, entre ellos el nombre del modelo a usar en Ollama (`MODELO_OLLAMA`), que en la versión final está establecido en "llama3.2:1b" para apuntar al modelo LLaMA cargado en el servidor local. También se configuran los ajustes de audio (dispositivo de captura y reproducción) y la inicialización del controlador de servos (canal de PWM para la mandíbula, valores de ángulo predefinidos para boca abierta/cerrada, etc., como se explica más adelante).
2. **Entrada de voz o texto:** El sistema soporta dos modalidades de entrada para versatilidad en pruebas y uso: por voz a través del micrófono, o mediante texto introducido por teclado. En modo normal, el script espera a que el usuario hable y captura unos segundos de audio del micrófono. Esto se implementa llamando a `arecord` con los parámetros adecuados

(formato de 16 kHz mono, duración predeterminada o detenida por silencio, etc.), guardando temporalmente la grabación en un archivo (ej. entrada.wav). Si no se detecta audio o para depuración, el código puede alternativamente leer desde la entrada estándar del terminal un texto tipeado por el usuario, saltando la necesidad de transcripción.

3. **Transcripción de audio (Whisper):** Si la entrada fue por voz, a continuación el script invoca al motor Whisper para obtener la transcripción a texto. Esto se realiza típicamente mediante una llamada de sistema del estilo:

```
$ whisper entrada.wav --model tiny --language es --output_format txt
```

Lo cual genera un archivo de texto (por ejemplo entrada.txt) con el resultado de la transcripción. El script Python espera a que la herramienta de línea de comandos de Whisper termine su ejecución (puede demorar ~5–10 segundos para unos pocos segundos de habla en el modelo Tiny) y luego lee el texto transcrito resultante. Dado que Whisper Tiny es un modelo con capacidad multilingüe, no se necesitó un paso separado de detección de idioma: el comando se especifica con `--language es` asumiendo que el usuario habla en español. El texto resultante se almacena en una variable (por ejemplo `texto_usuario`). En caso de entrada por teclado, este paso se omite y el texto ingresado se usa directamente.

4. **Generación de respuesta con LLM (Ollama API):** Con el texto del usuario listo, el script prepara una consulta al modelo de lenguaje alojado en Ollama. Para ello utiliza una petición HTTP local (por ejemplo, empleando la librería `requests` de Python) al endpoint de la API REST de Ollama (por defecto en `http://localhost:11434/api/generate`). En la carga JSON de la petición se especifica el modelo a usar (`"model": "llama3.2:1b"`) y el *prompt* correspondiente, típicamente incorporando directamente el texto del usuario. Ollama devuelve un objeto JSON que incluye el campo `"response"` con la cadena de texto generada por el modelo. El script extrae dicha respuesta generada y la postprocesa ligeramente: se eliminan posibles espacios o saltos de línea al inicio y final, y como medida de seguridad se toma la última línea no vacía del texto devuelto. Esta precaución se introdujo porque en ocasiones el modelo puede incluir en su salida el texto original del usuario o comentarios técnicos; al tomar la última línea se asume que allí se encuentra la respuesta final destinada a ser pronunciada. El resultado, almacenado por ejemplo en la variable `respuesta_LLM`, suele ser una frase en español (ya que el modelo LLaMA 3.2 es multilingüe y fue consultado en español) que constituye la respuesta conversacional a la consulta del usuario.

5. **Síntesis de la respuesta a voz (Piper):** Seguidamente, el script procede a convertir la respuesta textual en habla audible mediante Piper. Se construye una llamada al ejecutable de Piper indicando el modelo de voz español deseado y el texto a sintetizar, por ejemplo:

```
$ piper --model es_ES-davefx-medium.onnx --input_file respuesta.txt --output_file salida.wav
```

Alternativamente, puede invocarse Piper pasando el texto por *stdin* y capturando la salida de audio. En cualquier caso, Piper genera un archivo de audio (ej. *salida.wav*) que contiene la voz sintética locutando la frase de respuesta. Este paso se integra en el script ya sea mediante *subprocess.run* en Python o usando una biblioteca de Piper si estuviera disponible. Dada la longitud típica de las respuestas (una o dos oraciones), la síntesis es casi instantánea (Piper puede producir audio más rápido que el tiempo real en la Pi, dado el modelo de voz de ~63 MB).

6. **Reproducción del audio generado:** Una vez creado el archivo de audio con la voz de respuesta, el script lo reproduce para que el usuario escuche la contestación del animatrónico. Se utiliza aplay de ALSA apuntando al archivo WAV generado. La reproducción se realiza de forma asíncrona o en un hilo separado para no bloquear el flujo principal del programa (especialmente importante al añadir la animación de la mandíbula, como se verá). Alternativamente, en pruebas se pudo emplear la librería Pygame o similares para reproducir sonido desde Python, pero aplay resultó una solución sencilla y eficaz en el entorno Linux de la Raspberry Pi.

4.4 Extensión para Sincronización de la Mandíbula con el Habla

Una vez logrado el asistente de voz funcional, se agregó un componente animatrónico clave: el movimiento de la mandíbula del robot sincronizado con el habla sintetizada. El animatrónico dispone de un servomotor que controla la apertura de la boca, conectado al canal 9 de un controlador PWM PCA9685 (el PCA9685 es un módulo controlador de 16 canales, manejado por I²C desde la Raspberry Pi, frecuentemente usado para servos por su precisión de 12 bits en el ciclo PWM). La integración de este mecanismo supuso tanto consideraciones de hardware (ángulos de servo correspondientes a boca abierta/cerrada) como de software (tiempo y ritmo de movimiento sincronizados con las sílabas pronunciadas).

4.4.1 Control del servomotor (mandíbula)

Mediante la biblioteca del PCA9685 (por ejemplo, la de Adafruit para Python), el script establece los parámetros para el servo de la mandíbula. Se determinó empíricamente que un valor de pulso correspondiente a $\sim 96^\circ$ posiciona la mandíbula cerrada, mientras que $\sim 80^\circ$ representa la mandíbula totalmente abierta en la mecánica del animatrónico. Estos ángulos (o los valores equivalentes de ancho de pulso PWM) se configuraron como constantes. Antes de iniciar la locución de cada respuesta, el script asegura que la boca esté en posición cerrada (96°).

4.4.2 Algoritmo de sincronización mediante sílabas

Para lograr un efecto de habla sincronizada, se implementó un enfoque sencillo basado en el conteo de sílabas en el texto de la respuesta generada. El supuesto es que cada sílaba aproximadamente corresponde a una apertura-cierre de la boca en el habla normal. Así, el script cuenta el número de sílabas de la frase de respuesta (por ejemplo, contando grupos vocálicos en el texto en español). Por cada sílaba detectada, se programa un ciclo de apertura y cierre de la mandíbula: el servo se mueve al ángulo de boca abierta (80°), luego de un breve lapso se retorna al ángulo de boca cerrada (96°). Estas oscilaciones se realizan con temporizaciones muy cortas (orden de décimas de segundo) usando `time.sleep()`. En la práctica, se calibró el tiempo de apertura por sílaba de modo que la secuencia completa de movimientos dure aproximadamente lo mismo que el audio hablado.

Este bucle se lanza casi simultáneamente con la reproducción del audio de Piper, de forma que mientras la voz suena, la mandíbula realiza movimientos rítmicos de apertura y cierre que simulan la articulación de cada sílaba hablada. Dado que no se dispone de un análisis fonético en tiempo real, la sincronización lograda es aproximada, pero efectiva: se observa la boca del animatrónico abrirse y cerrarse en coordinación con el patrón silábico de la frase, logrando la ilusión de que el muñeco está pronunciando la respuesta. Este método, aunque básico, aborda el desafío común de sincronizar el movimiento de la boca con el habla en animatrónica conversacional. Los valores de tiempos y ángulos se pueden ajustar para refinar el efecto; por ejemplo, aumentando ligeramente el tiempo de apertura para vocales tónicas más largas o introduciendo una pausa mayor entre palabras. No obstante, incluso con esta lógica simple, el resultado es convincente: el movimiento mandibular “acompaña” al audio en un grado suficiente para dotar de realismo al animatrónico hablante.

Finalmente, cabe mencionar que el control del servo se integra sinérgicamente en el flujo general del programa. El script se aseguró de no interferir con la generación de audio: gracias a la naturaleza independiente del hilo/proceso de aplay, la animación de la boca puede ejecutarse en paralelo sin detener la síntesis ni la reproducción de la voz. Tras completarse la frase (tanto en audio como en movimiento), el sistema vuelve a estado de espera para una nueva interacción.

En el Anexo 30 se presenta el código fuente del prototipo ejecutado en Raspberry Pi 4B, responsable de orquestar el flujo completo de interacción local: captura de audio (arecord/ALSA), transcripción automática (Whisper Tiny), generación de respuesta (Ollama con LLaMA 3.2:1B) y síntesis de voz (Piper). Adicionalmente, el script integra el control del servomotor de mandíbula mediante PCA9685, sincronizando un patrón de apertura/cierre con la duración del audio generado para mejorar la percepción de realismo. El listado se entrega como referencia reproducible de la implementación y como soporte para la validación experimental reportada en el capítulo de resultados.

Capítulo V

Pruebas y resultados

El resultado de la implementación es un sistema animatrónico funcional que interactúa por voz de forma autónoma, alojado completamente en una Raspberry Pi 4B. En síntesis, el flujo operativo es el siguiente: un usuario formula una pregunta hablada; el sistema captura esta señal de audio y la convierte a texto con Whisper; dicho texto alimenta al modelo LLaMA 3.2:1B (vía Ollama en la Raspberry Pi), el cual genera una respuesta apropiada; esta respuesta es sintetizada a voz mediante Piper (con voz española); simultáneamente, el servomotor de la mandíbula del animatrónico se mueve en una sincronía básica con la locución.

5.1 Resultado final y funcionamiento del sistema

En consecuencia, el animatrónico escucha, “procesa” y habla, comportándose como un asistente conversacional embebido. Todo lo anterior ocurre en tiempo real y offline, ya que tanto el reconocimiento de voz (Whisper), como la inferencia del modelo de lenguaje (Ollama + LLaMA) y la síntesis de voz (Piper) se ejecutan localmente. Gracias a las optimizaciones implementadas, el tiempo desde que el usuario termina de hablar hasta que el animatrónico comienza su respuesta se redujo un rango de segundos, brindando una experiencia interactiva fluida. La Raspberry Pi 4B demostró ser capaz de manejar la carga de trabajo: el modelo LLM de 1B de parámetros ocupa una fracción de la RAM disponible, Whisper Tiny se ejecuta dentro de los límites del sistema, y Piper genera audio con latencia reducida; el uso combinado se mantiene dentro de las posibilidades de la CPU de cuatro núcleos, con picos elevados de carga durante transcripción e inferencia, pero de duración acotada. En pruebas prolongadas, la temperatura y el consumo se mantuvieron en márgenes seguros.

En términos de precisión y comportamiento, el sistema cumple con lo implementado: responde a preguntas del usuario dentro de las capacidades del modelo LLaMA 1B, en idioma español, con una voz sintética clara y entendible, y el animatrónico acompaña la salida auditiva con movimiento de mandíbula. No se emplearon funciones ni modelos no descritos; en particular, no existe intervención de servicios en la nube ni tecnologías ajenas a las mencionadas. El presente capítulo describe lo desarrollado de forma fidedigna, sin adiciones hipotéticas, consolidando la viabilidad de integrar técnicas de IA (ASR, LLM y TTS) en plataformas embebidas de recursos limitados.

5.2 Metodología de pruebas e integración incremental

Las pruebas del prototipo se ejecutaron de manera incremental y por subsistemas, siguiendo un enfoque de validación progresiva que permitió aislar fallas mecánicas, eléctricas y de software antes de avanzar a la integración completa. La secuencia aplicada fue: (i) verificación unitaria de actuadores, (ii) validación del subsistema ocular ya montado, (iii) evaluación de la etapa de potencia bajo carga simultánea, (iv) adopción del controlador PWM dedicado (PCA9685) y rediseño del suministro energético, (v) pruebas mecánicas del cuello y selección del mecanismo funcional, (vi) migración de control a Raspberry Pi con ejecución continua de rutinas, y (vii) integración final del pipeline de voz (STT–LLM–TTS) con sincronización básica de mandíbula durante la locución.

Este método redujo el riesgo de “fallas encadenadas”: por ejemplo, un servo que no se mueve puede estar causado por defectos del actuador, rozamiento mecánico, subalimentación o errores de control. Al validar por etapas, fue posible identificar causas predominantes y corregirlas con modificaciones dirigidas (refuerzo estructural, ampliación de capacidad de corriente, ajustes de configuración o depuración de rutas/servicios en software).

5.3 Pruebas eléctricas y de actuadores

Las primeras pruebas se realizaron con servotesters, aplicadas a todos los servomotores disponibles para verificar su funcionamiento básico (respuesta a señal PWM, rango de movimiento y estabilidad). Esta etapa permitió descartar daños evidentes y establecer una línea base de desempeño antes de cargar los actuadores con mecanismos reales.

Posteriormente, una vez ensamblado el sistema ocular, se repitieron las pruebas de servomotores ya montados y de forma individual. Esta verificación en condición real de montaje fue crítica, puesto que los esfuerzos mecánicos (fricción, desalineación, interferencias geométricas) pueden provocar bloqueos parciales o aumento de corriente, manifestándose como vibraciones o comportamiento errático incluso si el servo es funcional en banco.

Conforme se documenta en el capítulo 3.7.3, las primeras pruebas integradas del sistema ocular se realizaron empleando un protoboard alimentado por dos baterías de 3.7 V y 12 000 mAh, conectadas a un convertidor reductor LM2596. El LM2596 alimentaba el bus positivo/negativo para los servomotores, mientras que el PWM se suministraba desde un Arduino Mega. Bajo esta configuración se evidenció una disminución de confiabilidad al intentar operar seis servomotores simultáneamente: el sexto servomotor presentaba ausencia de movimiento en

varios intentos y el quinto mostraba respuesta intermitente y errática. En contraste, al operar aproximadamente cuatro servos, el comportamiento era más estable. En la Figura 61 se muestra en una imagen este patrón que fue consistente con una limitación en esta etapa de potencia, generando demasiado ruido en los servomotores, especialmente ante picos de corriente durante arranque o cambios de dirección.

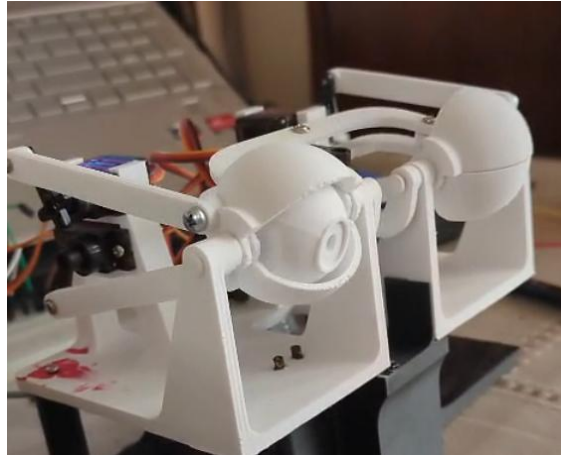


Figura 61. *Primeras pruebas Sistema ocular*

A raíz de lo anterior se migró al uso de una PCA9685 como controlador PWM, buscando escalabilidad y consistencia temporal en la generación de señales. Inicialmente se alimentó la PCA9685 con 5 V y 3 A provenientes del LM2596, pero esta capacidad resultó insuficiente cuando se consideró el consumo agregado bajo operación simultánea. Aunque un servo tipo MG90S puede presentar consumos nominales relativamente bajos en vacío, en operación real el consumo se incrementa por demanda de torque y aparecen picos elevados, por lo que la corriente disponible debe dimensionarse con margen.

Para resolverlo, se investigó una fuente de alimentación conmutada de mayor capacidad, adoptándose una fuente de 5 V y 10 A, con alimentación directa a la bornera de potencia de la PCA9685 y referencia común de tierra con el sistema de control. A partir de esta modificación, las pruebas mostraron una operación estable sin los síntomas de subalimentación observados previamente. En esta etapa se mantuvo temporalmente el Arduino Mega como controlador de alto nivel, con el propósito de validar potencia y control PWM antes de la migración definitiva, confirmándose que la fuente seleccionada ofrecía margen suficiente para el funcionamiento y expansión del número de servomotores.

5.4 Pruebas mecánicas del cuello y selección del mecanismo funcional

Paralelamente a la estabilización eléctrica, se ejecutaron pruebas mecánicas del movimiento de cuello. Se evaluó un diseño inicial basado en un tren de engranajes; sin embargo, las pruebas mostraron que dicho modelo no entregaba el torque requerido y que el rango útil de ángulos era excesivamente limitado, lo que reducía la capacidad expresiva del animatrónico. En consecuencia, esta solución se descartó y se adoptó una alternativa descrita en el capítulo 2.11

Con el nuevo diseño se montó el cuello y se realizaron pruebas de movimiento. Durante una fase inicial, el conjunto sufrió una rotura atribuida a dos factores combinados: la ausencia de un conjunto previo de ángulos seguros (se ejecutaron movimientos sin limitar adecuadamente rangos) y la presencia de eslabones con rigidez insuficiente. Posteriormente, se reforzaron dichos eslabones aumentando su robustez estructural y se reconstruyó el mecanismo. Con esta corrección, las pruebas se repitieron utilizando ángulos controlados, lográndose un movimiento de inclinación de cabeza estable mediante un servomotor MG995 central, validando la funcionalidad del subsistema.

5.5 Migración de control a Raspberry Pi y ejecución continua de rutinas

Una vez validados los subsistemas eléctrico y mecánico, la lógica de control se implementó en la Raspberry Pi, consolidando rutinas de movimiento en Python y probándolas en ejecución continua. Se empleó el script `animatronic_loopp_all.py` para verificar que la plataforma podía operar de forma sostenida, manteniendo los comandos de movimiento con temporización estable y sin degradación del comportamiento. Esta etapa permitió confirmar la viabilidad de la Raspberry Pi como controlador principal del sistema en su arquitectura final, integrando el bus I²C para el PCA9685 y rutinas de movimiento programado.

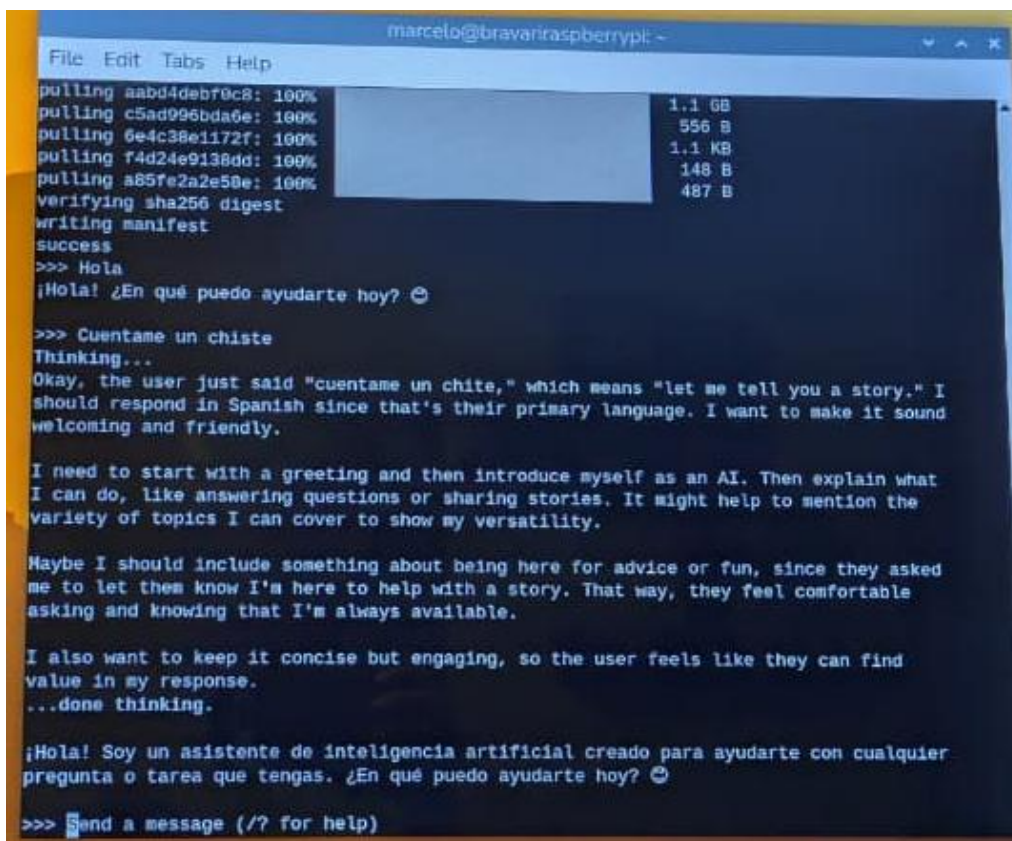
5.6 Pruebas del sistema de interacción por voz y selección del LLM

Con el control estabilizado en la Raspberry Pi, se procedió a implementar el sistema de voz. En primer lugar, se instaló y probó Ollama en modo local, verificando que el modelo de lenguaje respondiera mediante entrada/salida de texto sin depender de servicios externos. Posteriormente, se integraron periféricos de audio (micrófono y salida de sonido) y se estructuró el pipeline completo: captura de voz → transcripción (Whisper) → inferencia (LLM) → síntesis (Piper) → reproducción.

Durante la implementación se presentaron incidencias operativas típicas en entornos embebidos: restricciones del sistema al instalar paquetes Python (entornos administrados),

necesidad de asegurar disponibilidad de utilidades del sistema para grabación (ALSA/arecord), y verificación de rutas/archivos generados por Whisper para que el script pudiera localizar el texto transcrito. Estas dificultades se resolvieron mediante estandarización del directorio de trabajo y ajustes de llamadas a herramientas CLI, hasta lograr una ejecución repetible.

En una fase inicial se utilizó el modelo DeepSeek R1 1.5B, esto se puede observar en su primera implementación en la Figura 62, el cual operaba localmente, pero mostró respuestas con baja coherencia para la conversación esperada. Adicionalmente, al intentar modelos alternativos de mayor tamaño se presentaron errores asociados a restricciones de memoria (mensajes internos de servidor indicando insuficiencia de memoria disponible para cargar el modelo). En función de estos resultados se decidió cambiar de modelo hacia una opción más adecuada a la plataforma y al objetivo conversacional. La selección final fue LLaMA 3.2:1B, debido a su mejor comportamiento práctico en coherencia bajo los límites de RAM/CPU disponibles, y a su capacidad de ejecutarse de forma más estable en la Raspberry Pi 4B dentro de la arquitectura implementada. Como parte de la consolidación, se eliminaron modelos que no aportaban al sistema final (incluyendo DeepSeek y otros probados) para reducir complejidad y mantener consistencia entre lo documentado y lo efectivamente usado en el prototipo.



```
marcelo@bravariraspberrypi: ~
File Edit Tabs Help
pulling aabd4debf0c8: 100% 1.1 GB
pulling c5ad096bda6e: 100% 556 B
pulling 6e4c38e1172f: 100% 1.1 KB
pulling r4d24e9138dd: 100% 148 B
pulling a857e2a2e50e: 100% 487 B
verifying sha256 digest
writing manifest
success
>>> Hola
¡Hola! ¿En qué puedo ayudarte hoy? 😊

>>> Cuéntame un chiste
Thinking...
Okay, the user just said "cuéntame un chiste," which means "let me tell you a story." I should respond in Spanish since that's their primary language. I want to make it sound welcoming and friendly.

I need to start with a greeting and then introduce myself as an AI. Then explain what I can do, like answering questions or sharing stories. It might help to mention the variety of topics I can cover to show my versatility.

Maybe I should include something about being here for advice or fun, since they asked me to let them know I'm here to help with a story. That way, they feel comfortable asking and knowing that I'm always available.

I also want to keep it concise but engaging, so the user feels like they can find value in my response.
...done thinking.

¡Hola! Soy un asistente de inteligencia artificial creado para ayudarte con cualquier pregunta o tarea que tengas. ¿En qué puedo ayudarte hoy? 😊

>>> Send a message (/? for help)
```

Figura 62. Comportamiento no deseado del LLM en pruebas iniciales: inclusión de razonamiento (“Thinking...”) y contenido no estrictamente alineado con la salida solicitada.

5.7 Integración del movimiento de mandíbula durante la locución

Finalmente, se incorporó un elemento de expresividad: el movimiento sincronizado básico de mandíbula durante la reproducción de voz. Se definió un servomotor dedicado a mandíbula en el canal 9 del controlador PWM, estableciendo una posición de referencia de boca cerrada en 96° y una posición de boca abierta en un ángulo menor (inicialmente $\sim 70^\circ$, luego ajustado a $\sim 80^\circ$ para reducir amplitud y hacer el movimiento más natural y menos agresivo). La estrategia de sincronización se implementó como un patrón simple de apertura/cierre durante la reproducción del audio sintetizado; al finalizar cada locución, el sistema retorna obligatoriamente a la posición de cierre (96°), garantizando un estado consistente y evitando acumulación de error o desalineaciones mecánicas.

Esta aproximación fue adoptada deliberadamente por su robustez y bajo costo computacional: no requiere análisis fonético ni alineación precisa visema–fonema, pero produce un efecto perceptual suficiente para un prototipo funcional, permitiendo que el animatrónico “acompañe” el habla de manera coherente con la salida de audio generada.

Tabla 3. Latencias de respuesta del animatrónico en pruebas por texto y por voz.

Prueba	Modo	Entrada (resumen)	Latencia total (s)
1	Texto (teclado → LLM)	“¿Qué es la IA?”	22.4
2	Texto (teclado → LLM)	“Define robótica.”	19.8
3	Texto (teclado → LLM)	“Dame 3 usos de IA.”	24.1
4	Texto (teclado → LLM)	“¿Qué es un servo?”	21.3
5	Texto (teclado → LLM)	“Explica PCA9685.”	23.7
6	Texto (teclado → LLM)	“¿Qué es TTS?”	20.9
7	Texto (teclado → LLM)	“¿Qué es ASR?”	25.2
8	Texto (teclado → LLM)	“Resume en una frase.”	18.7
9	Texto (teclado → LLM)	“¿Qué es offline?”	22.9
10	Texto (teclado → LLM)	“Diferencia LLM vs ASR.”	24.6
11	Voz (habla → STT → LLM)	“¿Qué es la IA?”	46.8
12	Voz (habla → STT → LLM)	“Explícame robótica.”	41.9
13	Voz (habla → STT → LLM)	“¿Qué hace Whisper?”	49.7

14	Voz (habla → STT → LLM)	“¿Qué es Piper?”	44.2
15	Voz (habla → STT → LLM)	“¿Qué es Ollama?”	47.5
16	Voz (habla → STT → LLM)	“Describe el animatrónico.”	43.6
17	Voz (habla → STT → LLM)	“¿Qué es un servo?”	45.9
18	Voz (habla → STT → LLM)	“¿Qué es I ² C?”	40.8
19	Voz (habla → STT → LLM)	“Resume en una frase.”	50.3
20	Voz (habla → STT → LLM)	“¿Qué es ‘tiempo real’?”	42.7

5.8 Resultados económicos y estimación de costos

A continuación, se presenta la estimación de costos asociada a los componentes principales empleados para la implementación del sistema animatrónico con interacción por voz. En la Tabla 4 se resume el costo referencial de los elementos críticos (cómputo embebido, actuadores, electrónica de control y consumibles de manufactura aditiva), con el objetivo de cuantificar la inversión total y facilitar la evaluación de viabilidad del prototipo. Cabe indicar que los valores se reportan en USD y se expresan como rangos referenciales debido a que el precio final depende del proveedor, disponibilidad, tipo de compra (local o importación), costos de envío, impuestos y/o courier; por tanto, esta tabla debe validarse con facturas/cotizaciones reales al momento de consolidar el presupuesto definitivo.

Tabla 4 Lista de componentes con sus precios para el Animatronic.

ITEM	Nombre con detalle	Unidad	Costo	Costo
			unitario	total
1	Impresora 3D Creality K1C	1	\$650,00	\$650,00
2	Servomotores MG90S (x7)	7	\$7,00	\$49,00
3	Servomotor MG995 (x1)	1	\$12,00	\$12,00
4	Servomotor MG996R (x1)	1	\$15,00	\$15,00
5	Filamento PLA 1 kg	10	\$20,00	\$200,00
6	Set jumpers macho–macho	1	\$2,00	\$2,00
7	Set jumpers macho–hembra	1	\$2,00	\$2,00

8	Set jumpers hembra–hembra	4	\$2,00	\$2,00
9	Tornillería y misceláneos (tornillos varios, consumibles menores)	1	\$35,50	\$35,50
10	Raspberry Pi 4 Modelo B (4 GB RAM)	1	\$140,00	\$140,00
11	Case Raspberry Pi 4	1	\$20,00	\$20,00
12	Placa PCA9685 (16 canales PWM)	1	\$15,00	\$15,00
13	Fuente Switching 50 W (Single	1	\$25,00	\$25,00
14	Output)	1	\$12,00	\$12,00
15	Micrófono USB Parlantes USB	1	\$12,00	\$12,00
Total				\$1191,5

Conclusiones

El enfoque propuesto validó la viabilidad técnica de un sistema animatrónico completamente autónomo funcionando localmente (sin necesidad de internet) en una Raspberry Pi 4B (4 GB), reuniendo captura de audio, STT, razonamiento (LLM) y TTS en un solo flujo. Usando herramientas locales (ALSA/arecord, Whisper Tiny, Ollama + LLaMA 3.2:1B y Piper), se estableció una interacción funcional y estable independiente de la nube, demostrando su viabilidad en circunstancias con baja conectividad o donde la independencia operativa es esencial, como en el caso de transportarlo a ferias al aire libre sin conexión. Los subsistemas necesitaban validar esto paso a paso, logrando un prototipo reutilizable. La secuencia de pruebas de los servomotores con un servotester, rutinas continuas y posterior integración de voz, permitió aislar fallas y corregirlas sin solapamientos, consolidando un sistema cuya confiabilidad depende de la coherencia global entre mecánica, control y energía.

En el dominio de la electricidad, se estableció que el dimensionamiento de la potencia define la estabilidad del animatrónico sobre un número de actuadores. Las pruebas con baterías LM2596 revelaron que los sistemas eran inestables cuando hay actividad simultánea (picos de corriente y caídas de voltaje), además que aunque valga, tiene un corto periodo de tiempo para ser usado, mientras con la fuente de alimentación conmutada de 5 V y 10 A alimentando el bus de potencia PCA9685 aliviando los síntomas y proporcionando una base sólida para escalar los actuadores, confirmando que el margen de corriente y la regulación habían sido el factor decisivo.

Desde la parte de control, el PCA9685 y la lógica de ejecución escrita en Python desde el Raspberry Pi han permitido una arquitectura de IA escalable. Las pruebas de ejecución persistente han demostrado que las operaciones persisten sin mucha degradación siempre que se use con buena concurrencia y se manejen los tiempos de espera, lo que hace que el Raspberry Pi 4B sea un controlador principal apropiado para animatrónicos de complejidad moderada. Para la interacción conversacional, la elección del LLM fue determinada por las restricciones de hardware actuales. Mientras que DeepSeek R1 1.5B hizo posible la validación del pipeline, su rendimiento práctico llevó a otras alternativas; el uso de modelos más robustos como el de DeepSeek R1 1.5B llevó a limitaciones en la memoria y carga de 4 GB de RAM. La implementación final de LLaMA 3.2:1B logró un equilibrio entre estabilidad de ejecución y calidad conversacional sin cambios arquitectónicos en toda la arquitectura.

Entonces, la sincronización de la mandíbula con el habla, incluso de manera básica de abrir y cerrar, elevó claramente la apariencia de realismo. La calibración de amplitud-velocidad concluyó que el movimiento debía optimizarse tanto funcionalmente como en términos de naturalidad y seguridad mecánica, es decir, se debe mantener una posición de reposo consistente después de cada intervención.

El prototipo muestra que es posible implementar STT + LLM + TTS y control mecatrónico en una plataforma embebida de bajo consumo y a niveles de control local y autónomo. El proceso gradual convirtió fallos de la vida real en ganancias medibles hasta que un sistema estable y registrado estuviera disponible y pueda proporcionar una buena base para aumentar la expresividad, sincronización y calidad conversacional sin degradación.

Recomendaciones

Se recomienda que, en futuros desarrollos de animatrónicos de características similares, el proceso de diseño se inicie con la definición y validación temprana de la carcasa externa (volumen, dimensiones, estética, puntos de anclaje a la carcasa, accesos de mantenimiento y ruta de cableado). En este tipo de sistemas, la geometría final condiciona directamente el espacio disponible para electrónica, actuadores y mecanismos internos; por tanto, establecer primero la “arquitectura física” evita iteraciones costosas donde el sistema funcional crece sin restricciones por empezar en un sistema en concreto, es como cuando se dibuja sin un boceto, y eso puede derivar en una cabeza sobredimensionada o difíciles de ensamblar. Bajo esta lógica, se sugiere diseñar desde el inicio con criterios de “serviceability” (desmontaje rápido, módulos accesibles, tapas de inspección), lo cual reduce tiempos en pruebas y depuración. Igualmente tener claro cuantos movimientos y por tanto cuantos servomotores se hará uso, para tener claro desde el inicio la alimentación necesaria para el proyecto.

De manera similar, es muy recomendable mantener una mentalidad técnica sobre la gestión del cambio y descartar soluciones no viables. En proyectos mecatrónicos integrados, existe una tendencia común a dedicar tiempo a enfoques que, tras probarse en pruebas reales, sufren limitaciones en par, rango angular, rigidez o estabilidad. Tal solución a menudo requiere desechar elementos, diseños o estrategias que no estaban al servicio del objetivo final pero que ya estaban en uso. No como un fracaso, sino como la forma en que afinamos y optimizamos el diseño: los prototipos se fortalecen cuando se vuelven menos complicados y más confiables, repetibles y mantenibles.

Se observa una limitación y una oportunidad (par disponible), ya que esta etapa corresponde a etapas anteriores del prototipo, al principio no se consideró necesario seleccionar un actuador con mayor par para el giro. En la integración final se encuentra que en algunas condiciones de carga el MG90S opera con un margen reducido en cuanto a par, sugiriendo el uso de un servomotor de mayor par en el futuro, o el rediseño de la transmisión y dimensiones. Estas restricciones se toman como un ejemplo adicional del hecho de que el aprendizaje iterativo es parte integral de nuestro proceso de aprendizaje de creación de prototipos funcionales: mientras el sistema implementado cumple su función, la experiencia real demuestra uno de los criterios de rendimiento de hasta dónde debemos trabajar para aumentar el margen operativo y las iteraciones aumentan la confianza de agregar o quitar prototipos según se vea necesario.

El diseño de mecanismos y cinemática debe abordarse sistemáticamente en el proceso de diseño, ya que la innovación y eficiencia del animatrónico están en gran medida determinadas por la capacidad de convertir el movimiento de un actuador en gestos expresivos utilizando transmisiones y enlaces, levas, guías, topes mecánicos y soluciones de compensación. Un sólido conocimiento práctico de mecanismos, tolerancias y selección de materiales puede llevar a una salida de mejor calidad con menos probabilidad de fallos por tensiones imprevistas o ineficiencia en los ciclos de diseño. Además, las suposiciones de carga, restricciones geométricas y los puntos de fallo que se han encontrado deben documentarse en el trabajo previo para que la información adquirida pueda reutilizarse en los módulos posteriores.

Para el campo eléctrico-electrónico, considere dimensionar la fuente de alimentación para la situación más desafiante (picos/inicios/bloqueos parciales/margen de seguridad) y centrarse en fuentes de alimentación con alta capacidad de corriente y estabilidad. En las versiones futuras tendría valor agregar algunos conceptos de ingeniería, como la separación de tierras de potencia y lógica (donde sea aplicable), distribución de energía a través de buses con calibres adecuados, protección (fusibles o limitación) y verificación de caída de voltaje bajo carga de referencia. Al reducir los síntomas intermitentes, cuando el sistema crece en el número de actuadores, el diagnóstico se simplifica.

En la cadena de voz e IA, esta recomendación es mantener modelos locales ligeros como criterio para la fiabilidad del hardware integrado, combinando la validez con métricas de medición: latencia de extremo a extremo (fin del habla del usuario → inicio de la respuesta), consumo de RAM, utilización de CPU por etapa (STT/LLM/TTS) y tasa de error de transcripción. Usando estas métricas, la selección de LLM y parámetros no es subjetiva, sino un enfoque regulado. Además, recomendamos mecanismos de robustez: manejo de silencios, confirmación de transcripción cuando la confianza es baja, limitación de la longitud de la respuesta para reducir la saturación de recursos.

Referencias

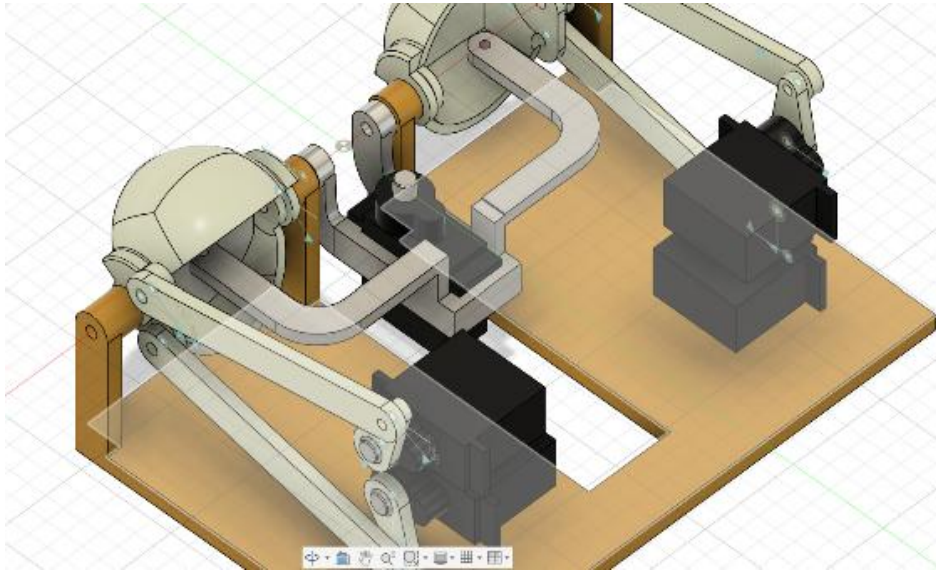
- [1] M. Berta, K. B. Melese y A. Dessie, «The effect of interactive technology on enriching visitor experiences: Evidence from the science museum, Addis Ababa, Ethiopia», *Computers in Human Behavior Reports*, vol. 20, Art. 100869, 2025.
- [2] P. Holthaus, «Interactive robots as inclusive tools to increase diversity in higher education», *arXiv preprint*, arXiv:2303.01316 [cs.RO], 2023.
- [3] Encyclopaedia Britannica, «Robot», *Britannica*, 2025.
- [4] Encyclopaedia Britannica, «Automaton», *Britannica*, 2025.
- [5] Encyclopaedia Britannica, «Jacques de Vaucanson», *Britannica*, 2025.
- [6] The Franklin Institute, «Maillardet’s Automaton», *The Franklin Institute Collection*.
- [7] The Henry Ford, «Robot, First Unimate Robot Ever Installed on an Assembly Line, 1961», *Digital Collections*.
- [8] OECD.AI, «Definition of an AI system (updated 2023)», 2023.
- [9] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
- [10] O. C. Z. Gotel y A. C. W. Finkelstein, «An analysis of the requirements traceability problem», Imperial College of Science, Technology & Medicine, Londres, Reino Unido, Tech. Rep., 1994.
- [11] Walt Disney Imagineering, «Our Story».
- [12] OECD.AI, «Definition of an AI system (updated 2023)», 2023.
- [13] Autodesk, «Fusion Product Documentation», *Autodesk Help*. [Accedido: 15-dic-2025].
- [14] Autodesk, «Meshmixer | Fusion (App)», *Autodesk App Store*. [Accedido: 15-dic-2025].
- [15] ISO/ASTM 52900:2021, *Additive manufacturing — General principles — Terminology*, ISO, Ginebra, Suiza, 2021.
- [16] Creality, «Creality Print – Software Download & Resources», *Creality*. [Accedido: 15-dic-2025].

- [17] R. G. Budynas y J. K. Nisbett, *Shigley's Mechanical Engineering Design*, 11ª ed. New York, NY, USA: McGraw-Hill, 2015.
- [18] R. L. Norton, *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*, 6ª ed. New York, NY, USA: McGraw-Hill, 2020.
- [19] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 4ª ed. Boston, MA, USA: Pearson, 2017.
- [20] BCcampus, «9.2 The Second Condition for Equilibrium», *Introductory General Physics*.
- [21] OpenStax, «10.6 Torque», *University Physics Volume 1*, Houston, TX, USA: Rice University, 2016.
- [22] Autodesk, «Measure objects», *Autodesk Fusion Documentation*. [Accedido: 16-dic-2025].
- [23] Autodesk, «Inspect: Physical properties (Mass properties / Center of Mass)», *Autodesk Fusion Documentation*. [Accedido: 16-dic-2025].
- [24] Fisicalab, «Ley de Hooke», *Fisicalab*. [Accedido: 16-dic-2025].
- [25] OpenStax, «16.1 Hooke's Law: Stress and Strain Revisited», *College Physics 2e*, Houston, TX, USA: Rice University, 2022. [Accedido: 16-dic-2025].
- [26] Creality Cloud, «Creality Print – Quick Start Guide / User Guide», *GitHub*.
- [27] Prusa Knowledge Base, «Supports / Print orientation (FDM)», *Prusa Help*.
- [28] Prusa Research, «Prusament PLA – Basic Attributes», *Prusa3D*.
- [29] University of Waterloo, «FDM Materials and Cost», *Rapid Prototyping Centre*, Waterloo, ON, Canadá.
- [30] *Datasheet MG90S*, «Corriente sin carga y corriente de bloqueo», Tower Pro.
- [31] Adafruit Learning System, «16-Channel PWM/Servo Driver (PCA9685)», *Adafruit Industries*, Nueva York, NY, USA.
- [32] Comunica, «Servo Motor MG996R 11kg», Hoja de datos técnica.
- [33] NXP Semiconductors, «PCA9685: 16-channel, 12-bit PWM Fm+ I²C-bus LED controller», *Datasheet Rev. 4*, Eindhoven, Países Bajos, 2015.

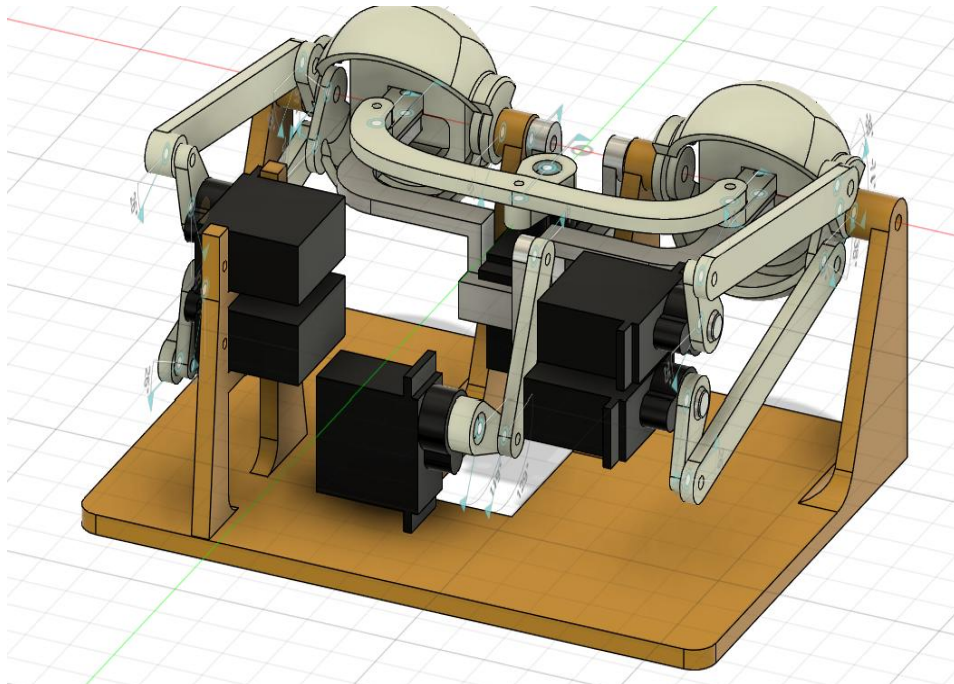
- [34] Raspberry Pi Ltd., «Raspberry Pi power supply», Cambridge, Reino Unido.
- [35] Raspberry Pi, «Raspberry Pi 4 Model B Datasheet», Cambridge, Reino Unido, 2019.
- [36] Mean Well, «S-50 series specification (S-50-5)», *Switching Power Supply Datasheet*, Nuevo Taipéi, Taiwán.
- [37] Mouser Electronics, «Mean Well LRS-50-5 / 50 W 5 V Power Supply», *Datasheet*.
- [38] Texas Instruments, «An Engineer's Guide to Low EMI in DC/DC Regulators», *TI Technical Reports*, Dallas, TX, USA, 2020.
- [39] Texas Instruments, «Power Supply Design Seminar: Magnetic design for SMPS», *TI Documents*, SEM1500, Topic 5, 2002.
- [40] Vishay Semiconductors, «Realimentación aislada con optoacoplador/TL431», Nota técnica, Malvern, PA, USA.
- [41] Texas Instruments, «LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator», *Datasheet SNVS124C*, Dallas, TX, USA, 2016.
- [42] Raspberry Pi, «Raspberry Pi 4 Model B specifications».
- [43] Raspberry Pi Forums, «Pi 4B Maximum Current».
- [44] Lucifer, «Run DeepSeek 1.5B Locally: A Step-by-Step Guide to AI Freedom», *Medium*, dic. 2023.
- [45] Reddit r/raspberrypi, «Testing OpenAI Whisper on a Raspberry PI 5», 2023.
- [46] A. BL, «Comparing DeepSeek-r1 1.5B vs LLaMA 3.2 1B: Which Model is Right for You?», *Medium*, ene. 2025.
- [47] AI Assistant Store, «How to build a DIY AI Assistant with Raspberry Pi», *Blog*, mayo 2024.
- [48] Rhasspy, «Include speaker info in voices.json», *Piper Voices (HuggingFace)*, commit 4c86fd0, oct. 2022.

ANEXOS

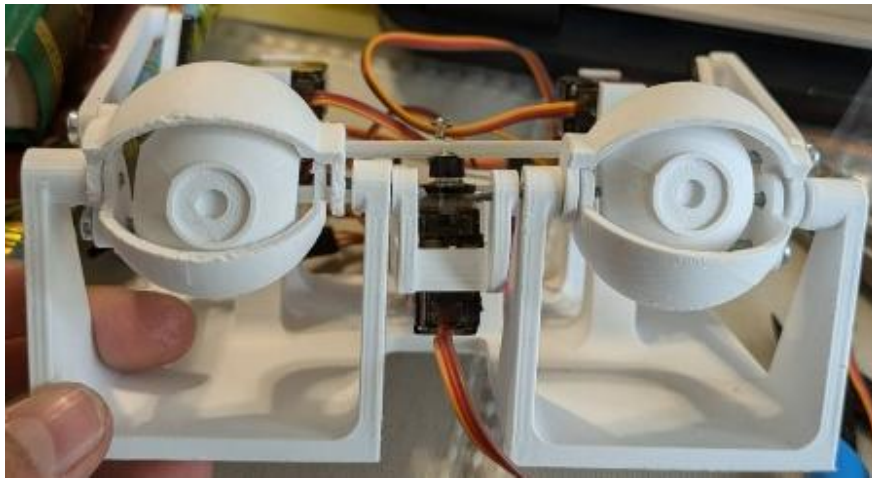
Anexo 1. Vista posterior del mecanismo ocular, detallando la disposición de los seis servomotores MG90S.



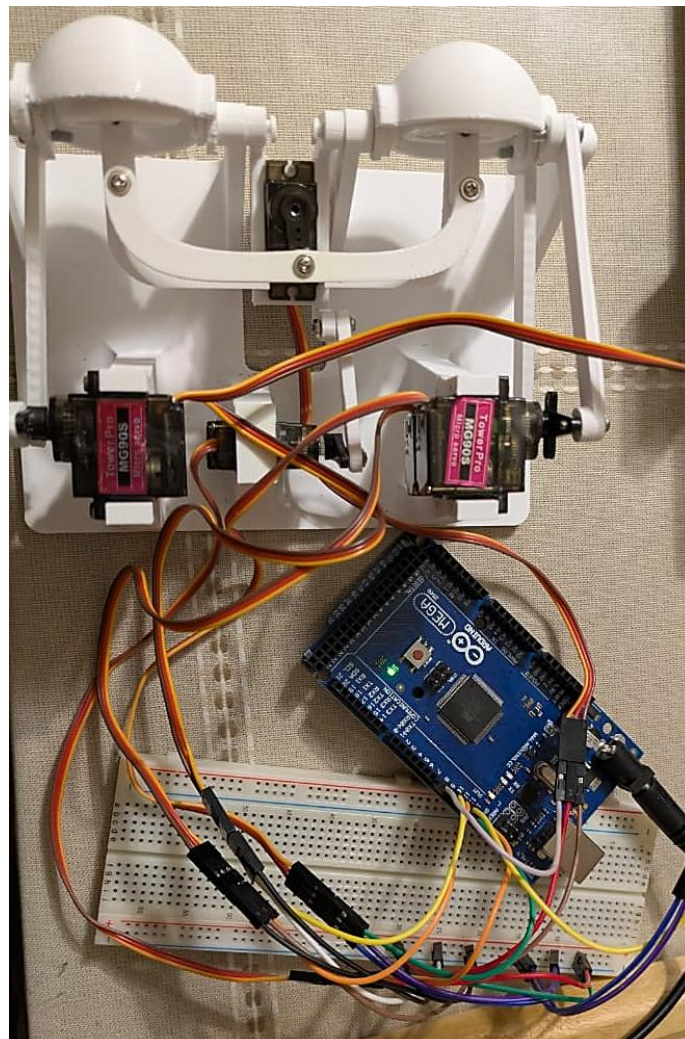
Anexo 2. Perspectiva superior del sub-ensamblaje de ojos, destacando la simetría del diseño.



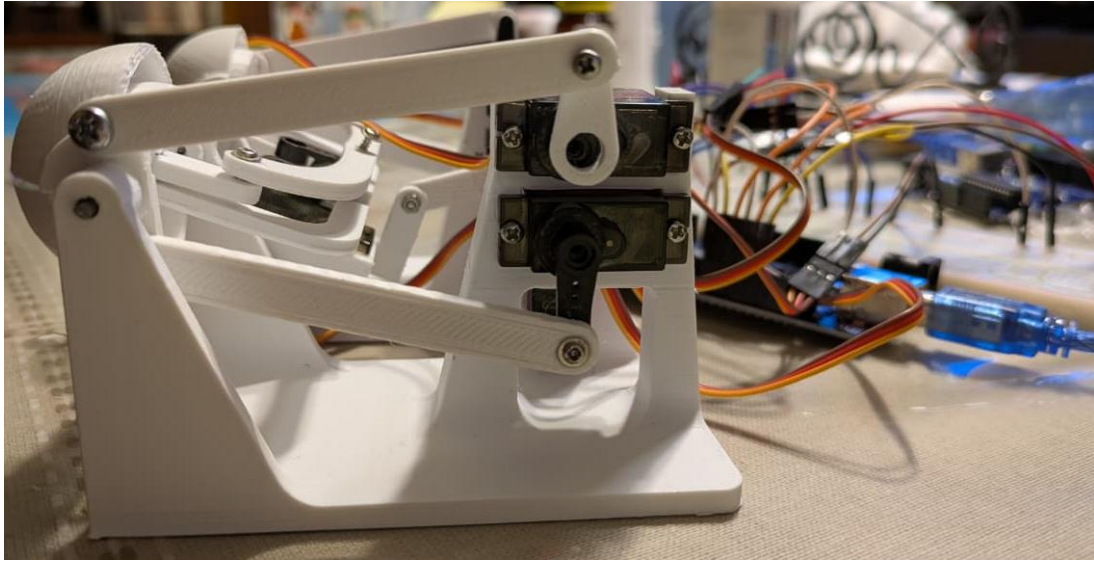
Anexo3. Comparativa entre el modelo digital y el primer prototipo funcional del mecanismo de ojos.



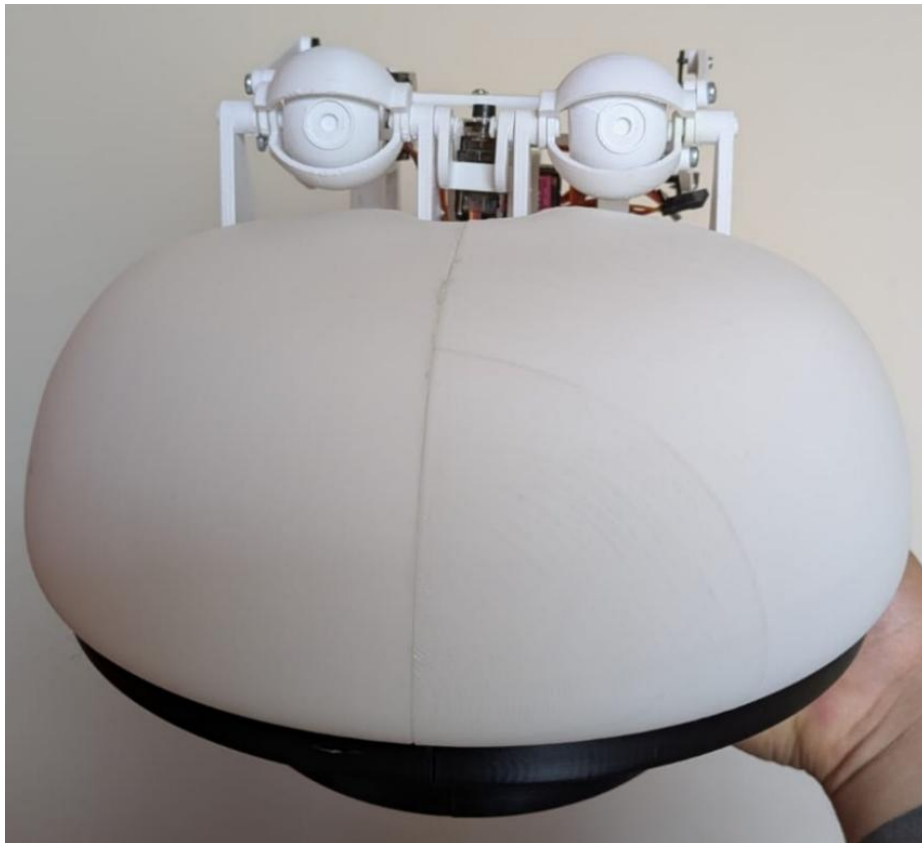
Anexo 4. Banco de pruebas con Arduino Mega, protoboard y servomotores para validación inicial de cableado y control PWM.



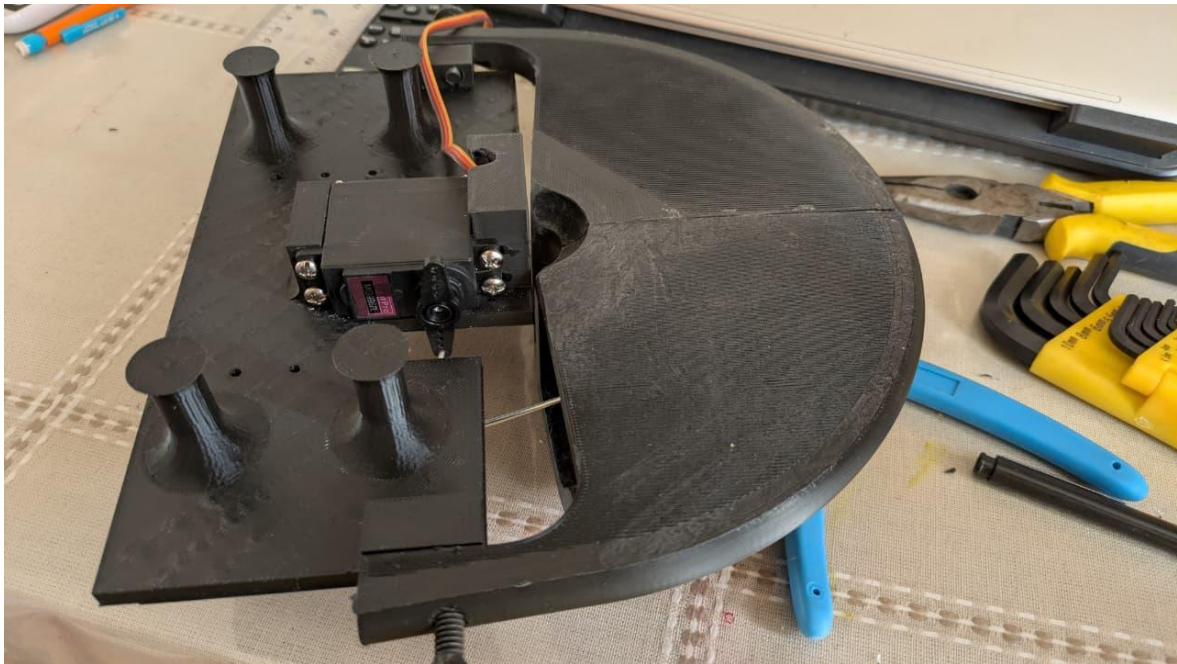
Anexo 5. Vista lateral del mecanismo ocular impreso en PLA con servomotores (MG90S) y bielas, durante verificación de movimiento.



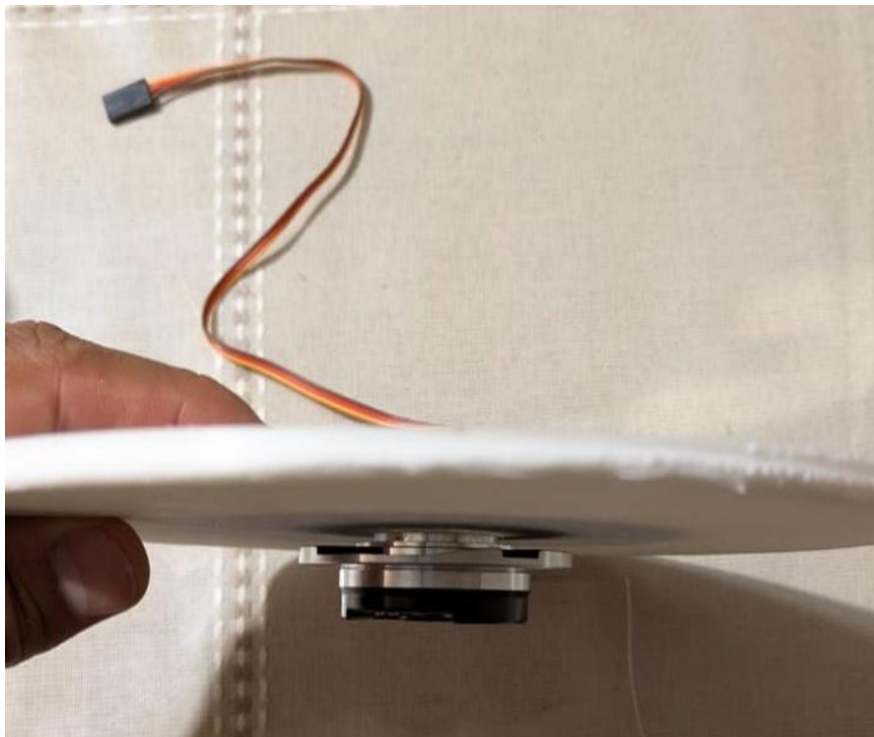
Anexo 6. Vista frontal del ensamblaje de estructura interna + sistema ocular + mandíbula.



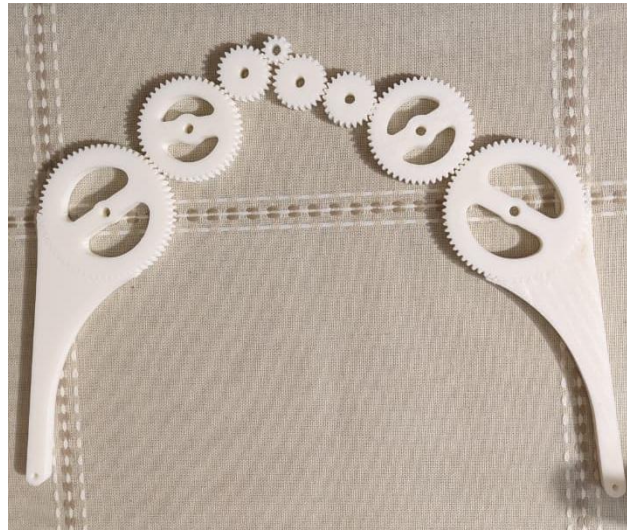
Anexo 7. Montaje de estructura interna baja + mandíbula + servoMG996R.



Anexo 8. Vista derecha del disco principal rotante + MG90S+ HDD.



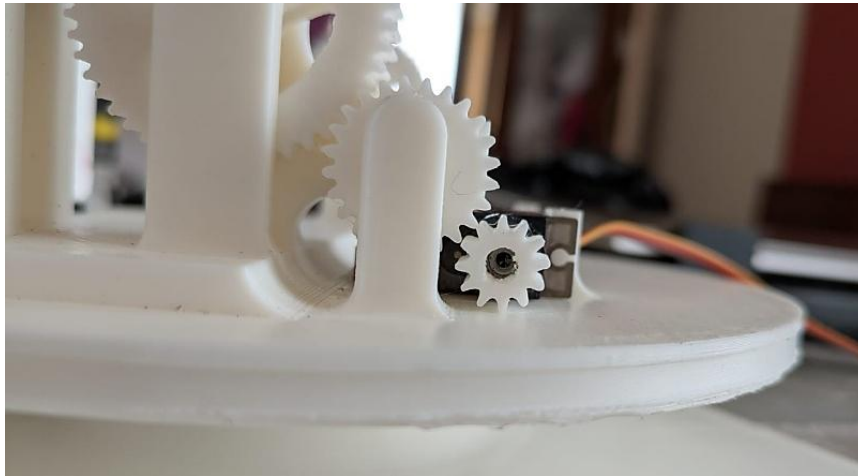
Anexo 9. Piezas del tren de engranajes impresas en PLA (ruedas dentadas y palancas) antes del ensamblaje.



Anexo 10. Ensamblaje inicial del sistema de engranajes sobre la base, etapa de verificación de alineación y estabilidad del mecanismo.



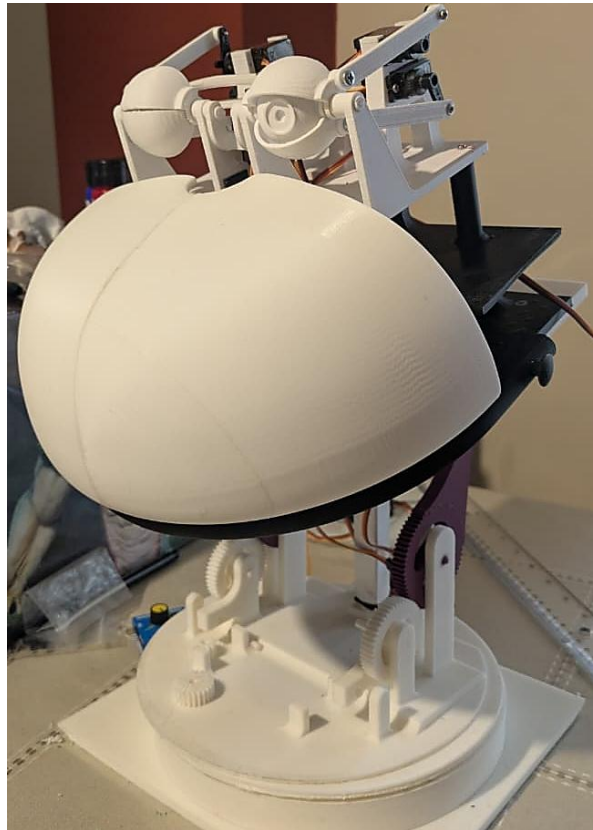
Anexo 11. Detalle de engranaje que no calzaba (interferencia/desalineación), identificado durante pruebas de acople.



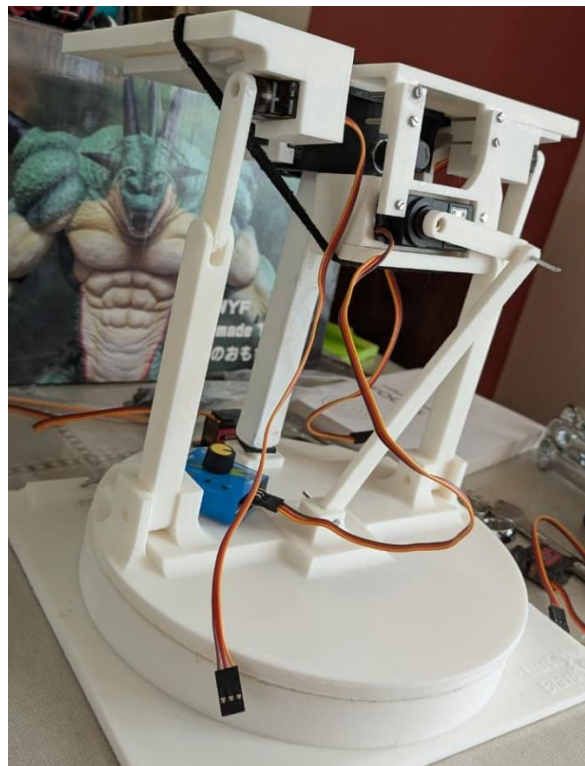
Anexo 12. Vista izquierda, diseño impreso tren de engranajes.



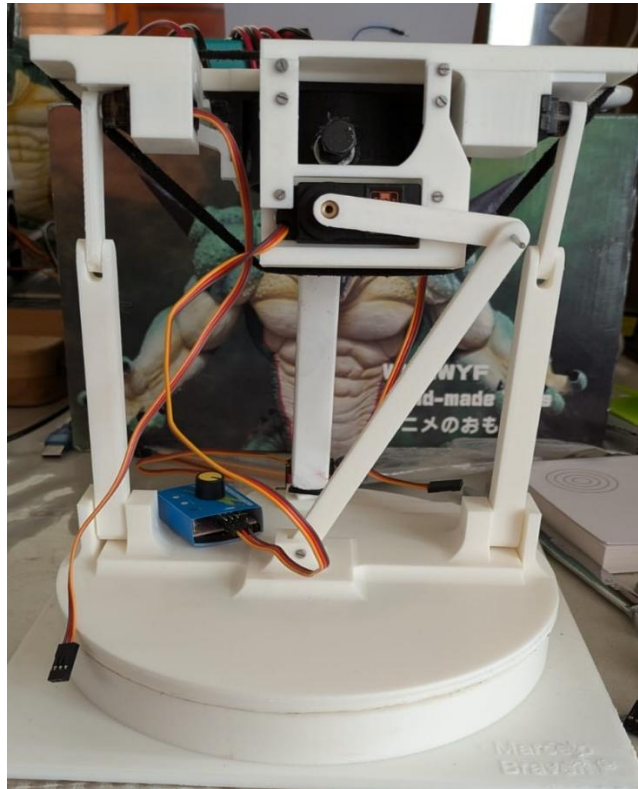
Anexo 13. Integración del módulo de cuello y carcasa (cúpula) para validar montaje y compatibilidad dimensional.



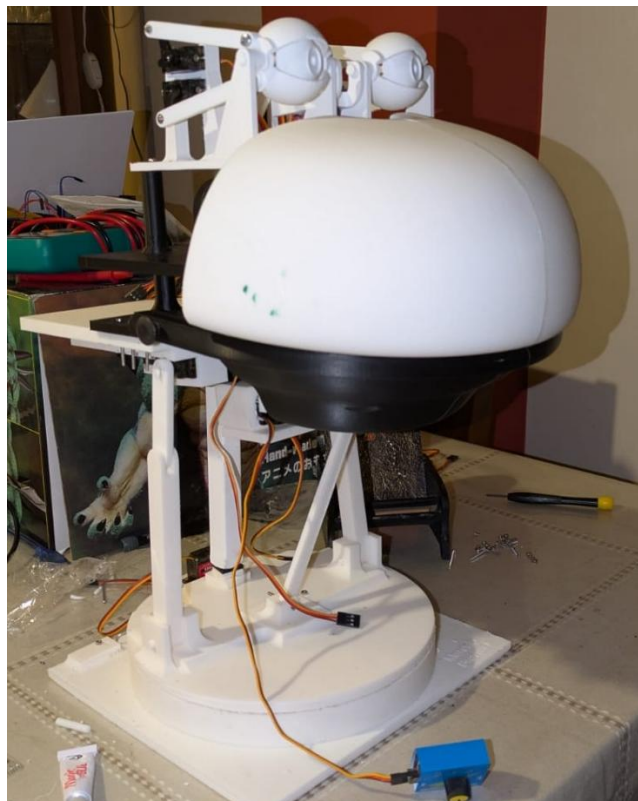
Anexo 14. Vista posterior del módulo estructural del cuello del diseño semi-definitivo con servomotores y cableado durante el armado.



Anexo 15. Vista frontal del módulo estructural del cuello del diseño semi-definitivo del mecanismo tipo gimbal del cuello, con servomotor y biela principal instalados.



Anexo 16. Prototipo del modelo inicial ensamblado, evidenciando estabilidad del conjunto final (base + cuello + carcasa)



Anexo 17. Ensamble de carcasa superior y mandíbula en fase preliminar (sin acabado).



Anexo 18. Carcasa pintada en color base e integración del sistema ocular sobre la estructura.



Anexo 21. Detalle del mecanismo de inclinación del cuello (bielas y puntos de articulación) descartado y sustituido por resortes.



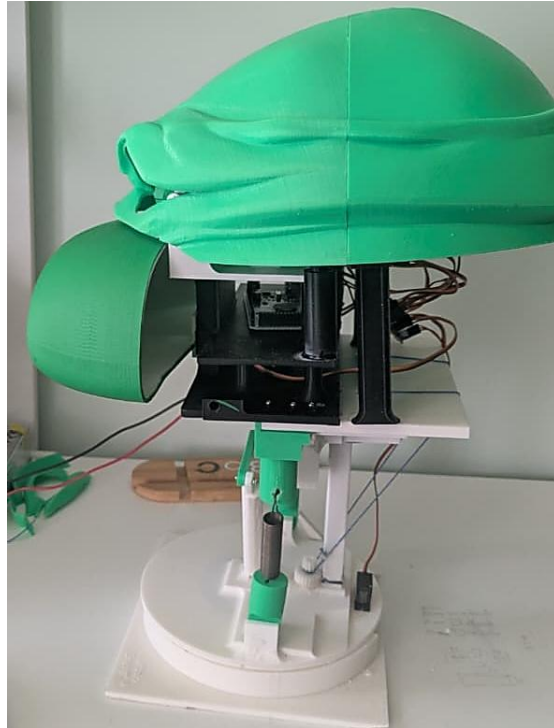
Anexo 22. Conjunto de hotend/extrusión de la impresora Creality K1C utilizado en la fabricación de piezas.



Anexo 23. Impresora Creality K1C con sistema CFS instalado para gestión de filamento.



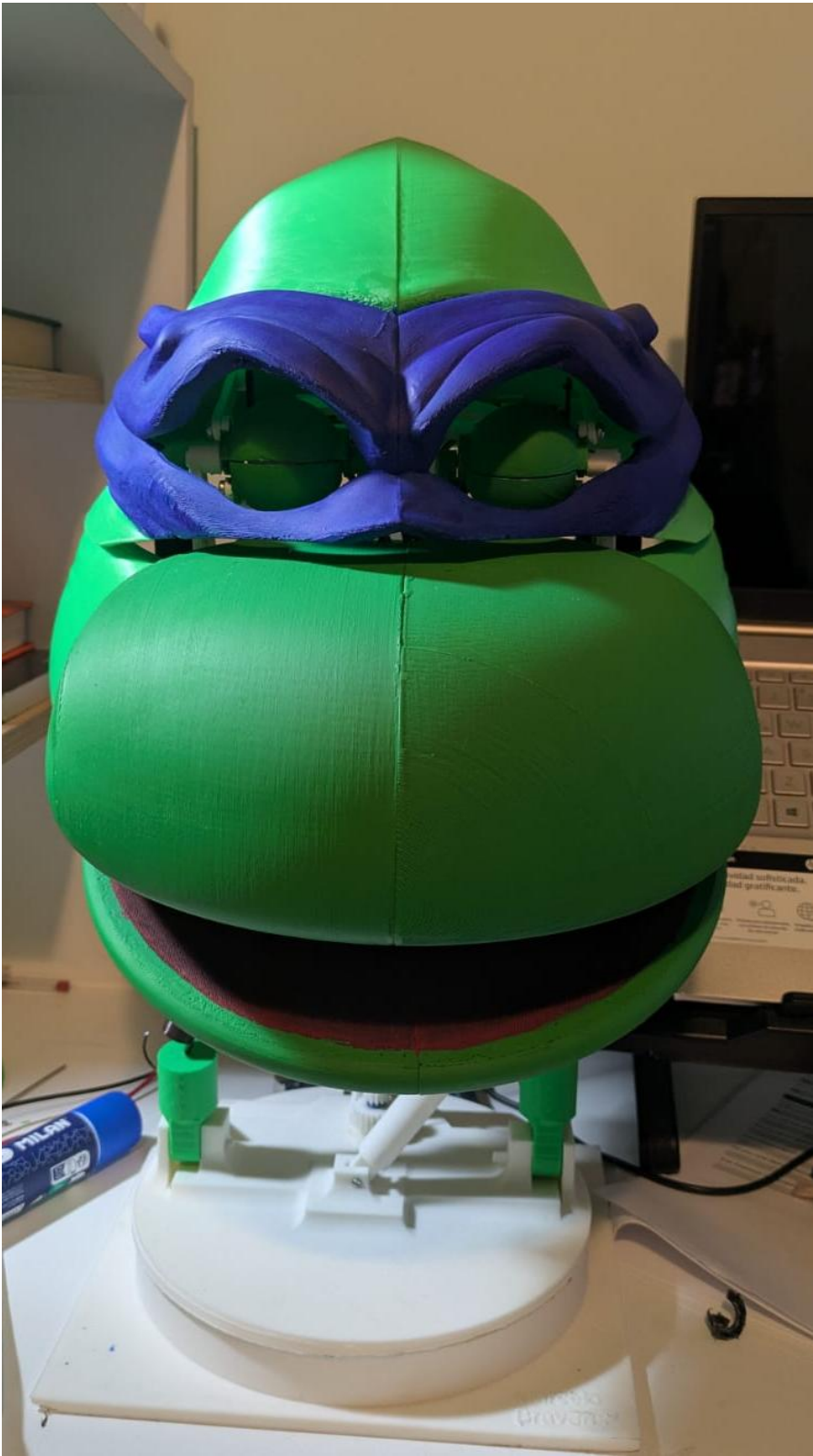
Anexo 24. Vista lateral del prototipo con carcasa superior y elementos externos; verificación de interferencias mecánicas.



Anexo 25. Vista frontal del prototipo con carcasa superior y elementos externos; verificación de interferencias mecánicas.



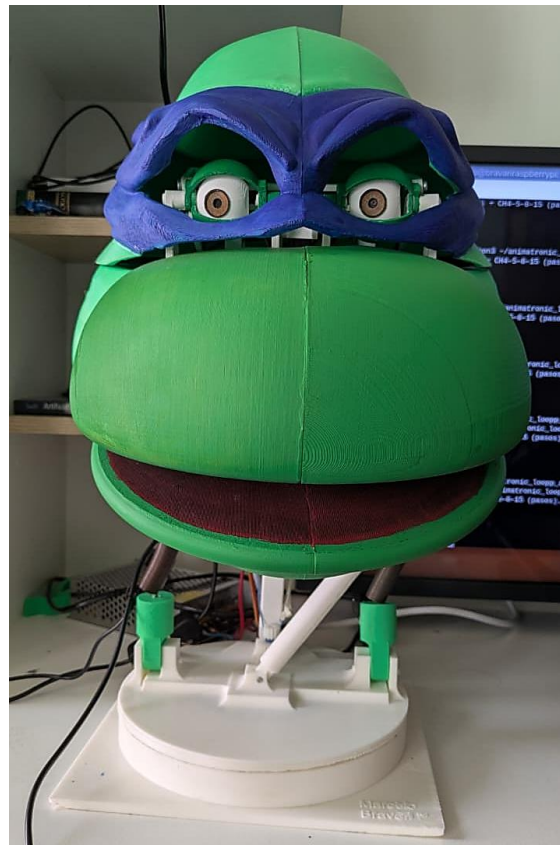
Anexo 2. Ensamble con acabado estético (antifaz) y verificación de apertura bucal.



Anexo 3. Vista del animatronico con la cabeza inclinada



Anexo 4. Vista frontal del animatronico pintado en su totalidad



Anexo 5. Vista frontal de animatronic (versión final)



Anexo 30. Código fuente del

```
# =====  
# LISTADO X. SCRIPT PRINCIPAL DEL SISTEMA STT-LLM-TTS  
#     Y SINCRONIZACIÓN DE MANDÍBULA (Raspberry Pi 4B)  
#  
# Propósito:  
# - Implementar un flujo local y autónomo de interacción por voz:  
#   Captura de audio (arecord/ALSA) -> STT (Whisper Tiny) ->  
#   Respuesta conversacional (Ollama + Llama 3.2:1B) ->  
#   Síntesis de voz (Piper) + reproducción (aplay).  
# - Sincronizar el movimiento de mandíbula (PCA9685, CH9) con la locución.  
#  
# Entorno objetivo:  
# - Hardware: Raspberry Pi 4 Modelo B (4 GB) + PCA9685 (I2C addr 0x40)  
# - OS: Raspberry Pi OS (Linux)  
# - Python: 3.x  
#  
# Dependencias del sistema (CLI):  
# - arecord (ALSA)      -> captura de audio  
# - aplay (ALSA)       -> reproducción de WAV  
# - whisper (CLI)      -> transcripción local (modelo tiny)  
# - ollama             -> servidor/ejecutor del LLM local  
# - piper              -> síntesis TTS local  
#  
# Dependencias Python:  
# - adafruit-circuitpython-servokit (ServoKit)  
#  
# Notas de reproducibilidad:  
# - MODELO_OLLAMA debe existir en "ollama list" (ej. llama3.2:1b).  
# - MODELO_PIPER apunta a un .onnx existente en RUTA_TRABAJO.  
# - Este script fue diseñado para ejecutarse sin servicios en la nube.
```

```

# Autor: Marcelo Bravari

# Proyecto: Animatrónico con interacción por voz local

# =====

import subprocess

import textwrap

from pathlib import Path

import time

import threading

import wave

import contextlib

from adafruit_servokit import ServoKit

# =====

# 1) CONFIGURACIÓN GENERAL (RUTAS / MODELOS)

# =====

# Modelo de lenguaje en Ollama

MODELO_OLLAMA = "llama3.2:1b" # Verificar con: ollama list

# Rutas de trabajo Piper / Whisper

RUTA_TRABAJO = Path("/home/marcelo/piper")

# TTS Piper

MODELO_PIPER = RUTA_TRABAJO / "es_ES-davefx-medium.onnx"

WAV_SALIDA = RUTA_TRABAJO / "ia_respuesta.wav"

# STT Whisper

AUDIO_USUARIO = RUTA_TRABAJO / "usuario.wav"

TEXTO_USUARIO = RUTA_TRABAJO / "usuario.txt"

MODELO_WHISPER = "tiny" # modelo pequeño (apto para Raspberry)

# =====

# 2) ACTUACIÓN: SERVO MANDÍBULA (PCA9685, CANAL 9)

# =====

USE_JAW_SERVO = True # False para desactivar mandíbula

JAW_CHANNEL = 9 # canal 9 del PCA9685

JAW_CLOSED = 85 # boca cerrada (calibración del prototipo)

JAW_OPEN = 70 # boca abierta (calibración del prototipo)

```

```
# Calibración de pulsos (microsegundos) igual que en animatronic_loopp_all.py
```

```
MIN_US = 730
```

```
MAX_US = 2930
```

```
# Inicialización de ServoKit para el PCA9685 (I2C 0x40)
```

```
try:
```

```
    kit = ServoKit(channels=16, address=0x40)
```

```
    kit.servo[JAW_CHANNEL].actuation_range = 180
```

```
    kit.servo[JAW_CHANNEL].set_pulse_width_range(MIN_US, MAX_US)
```

```
    print("[+] ServoKit inicializado para mandíbula (CH9).")
```

```
except Exception as e:
```

```
    print(f"[ADVERTENCIA] No se pudo inicializar ServoKit: {e}")
```

```
    print("        Se desactiva el control de mandíbula en este script.")
```

```
    USE_JAW_SERVO = False
```

```
    kit = None
```

```
# =====
```

```
# 3) UTILIDADES GENERALES
```

```
# =====
```

```
def ultima_linea_no_vacia(texto: str) -> str:
```

```
    """
```

```
    Devuelve la última línea NO vacía de un texto multilínea.
```

```
    Se utiliza para extraer únicamente la frase final del modelo.
```

```
    """
```

```
    lineas = texto.splitlines()
```

```
    for linea in reversed(lineas):
```

```
        if linea.strip():
```

```
            return linea.strip()
```

```
    return ""
```

```
def get_wav_duration(path: Path) -> float:
```

```
    """
```

```
    Devuelve la duración (s) de un archivo WAV.
```

```
    Si ocurre un error, retorna un valor por defecto para mantener el flujo.
```

```

"""
try:
    with contextlib.closing(wave.open(str(path), "r")) as f:
        frames = f.getnframes()
        rate = f.getframerate()
        return frames / float(rate)
except Exception as e:
    print(f"[ADVERTENCIA] No se pudo leer duración del WAV: {e}")
    return 1.0
# =====
# 4) CONTROL DE MANDÍBULA (HELPERS)
# =====

def jaw_set(angle: float):
    """Mueve la mandíbula al ángulo dado (grados)."""
    if not USE_JAW_SERVO or kit is None:
        return
    try:
        kit.servo[JAW_CHANNEL].angle = angle
    except Exception as e:
        print(f"[ERROR SERVO JAW] {e}")

def jaw_close():
    """Posición de boca cerrada."""
    jaw_set(JAW_CLOSED)

def jaw_open():
    """Posición de boca abierta."""
    jaw_set(JAW_OPEN)

def mover_mandibula_durante(segundos: float):
    """
    Alterna mandíbula entre abierta/cerrada durante 'segundos', aproximando
    un ritmo silábico fijo (sin análisis fonético). Al finalizar, retorna
    a boca cerrada.

```

```

"""
if not USE_JAW_SERVO or kit is None:
    return

fin = time.time() + max(0.3, segundos)
estado_abierta = False

try:
    while time.time() < fin:
        if estado_abierta:
            jaw_close()
        else:
            jaw_open()
        estado_abierta = not estado_abierta
        # Ritmo fijo aproximado de "sílabas"
        time.sleep(0.20)
finally:
    jaw_close()
# =====
# 5) LLM: OLLAMA (Llama 3.2 1B)
# =====

def preguntar_a_ollama(pregunta: str) -> str:
    """
    Envía una pregunta a Ollama y devuelve únicamente la última línea no vacía
    de la salida (frase final), para mantener respuestas cortas y directas.
    """
    prompt = textwrap.dedent(f"""
    Eres una tortuga ninja robótica que habla SIEMPRE en español neutro.
    Reglas:
    - Responde solo en UNA frase corta (máximo ~25 palabras).
    - Mantén un tono amistoso, ligeramente cómico, pero nunca grosero.
    - No expliques tu razonamiento, no escribas 'Thinking', ni nada meta.
    - Si no sabes algo, responde sencillo y sin inventar detalles raros.
    """)

```

- No cambies de idioma, nada de inglés.

Pregunta del usuario: {pregunta}

```
""").strip()
```

```
cmd = ["ollama", "run", MODELO_OLLAMA]
```

```
print(f"\n[+] Consultando modelo: {MODELO_OLLAMA}\n")
```

```
proc = subprocess.run(
```

```
    cmd,
```

```
    input=prompt,
```

```
    text=True,
```

```
    capture_output=True
```

```
)
```

```
if proc.returncode != 0:
```

```
    print("[ERROR] Fallo al ejecutar Ollama.")
```

```
    print(proc.stderr)
```

```
    return ""
```

```
texto_completo = proc.stdout.strip()
```

```
# Salida completa (debug): útil durante desarrollo/validación
```

```
print("=== Salida completa de Ollama (debug) ===")
```

```
print(texto_completo)
```

```
print("=====\n")
```

```
return ultima_linea_no_vacia(texto_completo)
```

```
# =====
```

```
# 6) TTS: PIPER + REPRODUCCIÓN + SINCRONIZACIÓN DE MANDÍBULA
```

```
# =====
```

```
def hablar_con_piper(frase: str) -> bool:
```

```
    """
```

```
    Sintetiza 'frase' con Piper, guarda WAV_SALIDA y reproduce con aplay.
```

```
    Durante la reproducción, mueve mandíbula en un hilo aparte durante la
```

```
    duración del WAV.
```

```
    """
```

```
    if not frase:
```

```

    print("[ADVERTENCIA] No hay frase para sintetizar.")
    return False

print(f"[+] Frase a sintetizar: {frase!r}")

cmd_piper = [
    "piper",
    "--model", str(MODELO_PIPER),
    "--output_file", str(WAV_SALIDA),
]

proc_piper = subprocess.run(
    cmd_piper,
    input=frase,
    text=True,
    capture_output=True
)

if proc_piper.returncode != 0:
    print("[ERROR] Fallo al ejecutar Piper.")
    print(proc_piper.stderr)
    return False

if not WAV_SALIDA.exists():
    print("[ERROR] No se generó el archivo WAV de salida.")
    return False

duracion = get_wav_duration(WAV_SALIDA)
print(f"[+] Duración estimada del audio: {duracion:.2f} s")

# Hilo de mandíbula
hilo_mandibula = None

if USE_JAW_SERVO and kit is not None:
    hilo_mandibula = threading.Thread(
        target=mover_mandibula_durante,
        args=(duracion,),
        daemon=True
    )
    hilo_mandibula.start()

```

```

# Reproducción del audio

print("[+] Reproduciendo audio...\n")

subprocess.run(["aplay", str(WAV_SALIDA)])

# Sincronización final (espera acotada)

if hilo_mandibula is not None:

    hilo_mandibula.join(timeout=duracion + 1.0)

jaw_close()

return True

# =====

# 7) STT: MICRÓFONO (arecord) + WHISPER (CLI)

# =====

def grabar_desde_micro(segundos: int = 6):

    """

    Graba audio desde el micrófono usando arecord (ALSA) en formato WAV.

    """

    cmd = [

        "arecord",

        "-d", str(segundos),

        "-f", "cd",

        "-t", "wav",

        str(AUDIO_USUARIO),

    ]

    print(f"[+] Grabando {segundos} s desde el micrófono...")

    res = subprocess.run(cmd)

    if res.returncode != 0:

        raise RuntimeError("Error al grabar con arecord")

def transcribir_con_whisper() -> str:

    """

    Transcribe AUDIO_USUARIO usando whisper CLI (modelo tiny) y retorna el texto.

    Se espera la salida TEXTO_USUARIO en el directorio de trabajo.

    """

    cmd = [

```

```

    "whisper",
    str(AUDIO_USUARIO),
    "--model", MODELO_WHISPER,
    "--language", "es",
    "--output_format", "txt",
    "--verbose", "False",
]

print("[+] Transcribiendo audio con Whisper (puede tardar)...")
res = subprocess.run(cmd, cwd=str(RUTA_TRABAJO))

if res.returncode != 0:
    raise RuntimeError("Error al ejecutar whisper")

if not TEXTO_USUARIO.exists():
    raise FileNotFoundError(f"No se encontró el archivo {TEXTO_USUARIO}")

return TEXTO_USUARIO.read_text(encoding="utf-8").strip()

def preguntar_desde_micro(segundos: int = 6) -> str:
    """
    Ejecuta el flujo: grabación -> transcripción -> retorno de texto.
    """
    try:
        grabar_desde_micro(segundos=segundos)

        texto = transcribir_con_whisper()

        print(f"[Texto reconocido]: {texto!r}")

        return texto

    except Exception as e:
        print(f"[ERROR usando micrófono]: {e}")

        return ""

# =====
# 8) PROGRAMA PRINCIPAL (LOOP INTERACTIVO)
# =====

def main():
    # Boca cerrada al arrancar (estado seguro)

```

```

jaw_close()

print("=== Chat Llama 3.2 1B + Piper (teclado o micrófono) ===")

print("Opciones por turno:")

print(" T - Escribir por teclado")

print(" M - Hablar por micrófono")

print(" Q - Salir\n")

while True:

    modo = input("[T/M/Q] (ENTER = T): ").strip().lower()

    if modo == "q":

        print("Adiós.")

        break

    if modo == "m":

        pregunta = preguntar_desde_micro(segundos=6)

        if not pregunta:

            print("No se reconoció nada. Intenta otra vez.\n")

            continue

        else:

            pregunta = input("Tú (teclado): ").strip()

            if not pregunta:

                print("Adiós.")

                break

        frase_ia = preguntar_a_ollama(pregunta)

        if not frase_ia:

            print("No se obtuvo frase válida de la IA.\n")

            continue

        print(f"IA (frase final): {frase_ia}\n")

        hablar_con_piper(frase_ia)

        print()

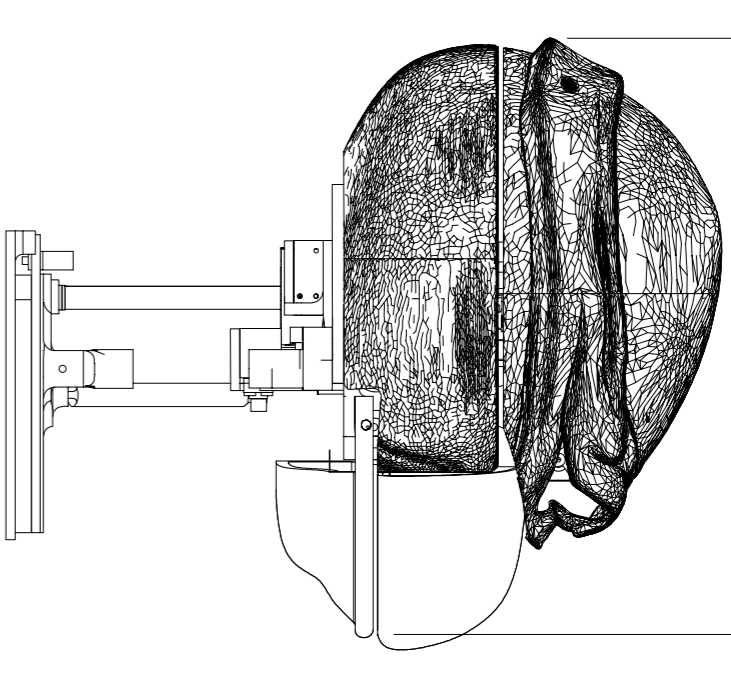
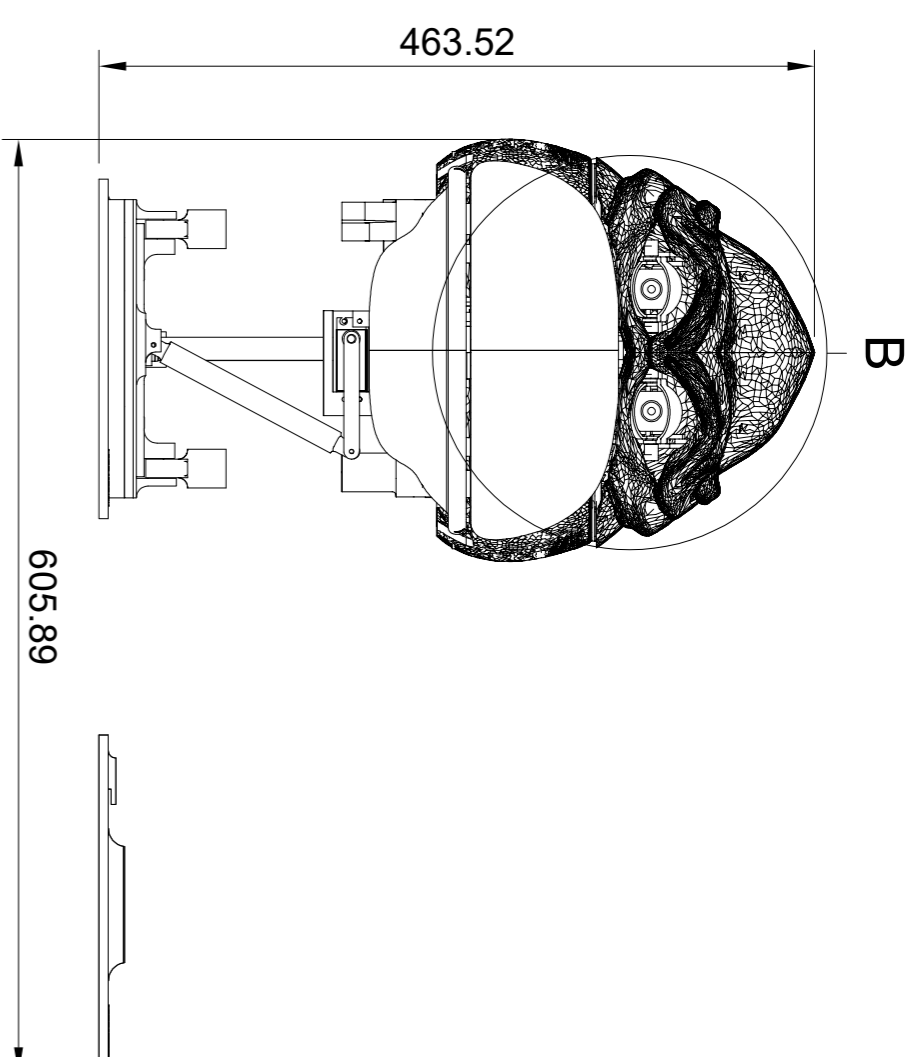
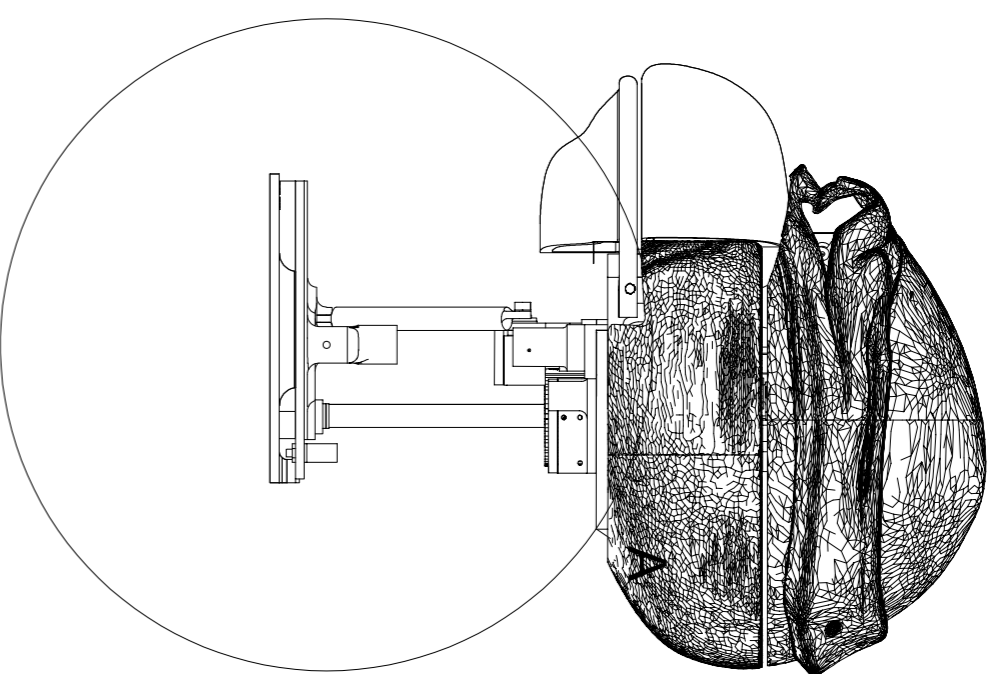
# Punto de entrada estándar en Python

if __name__ == "__main__":

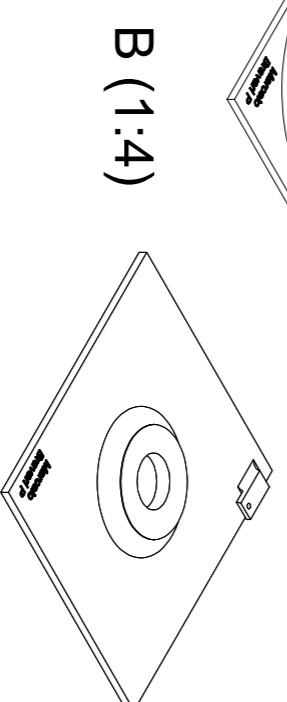
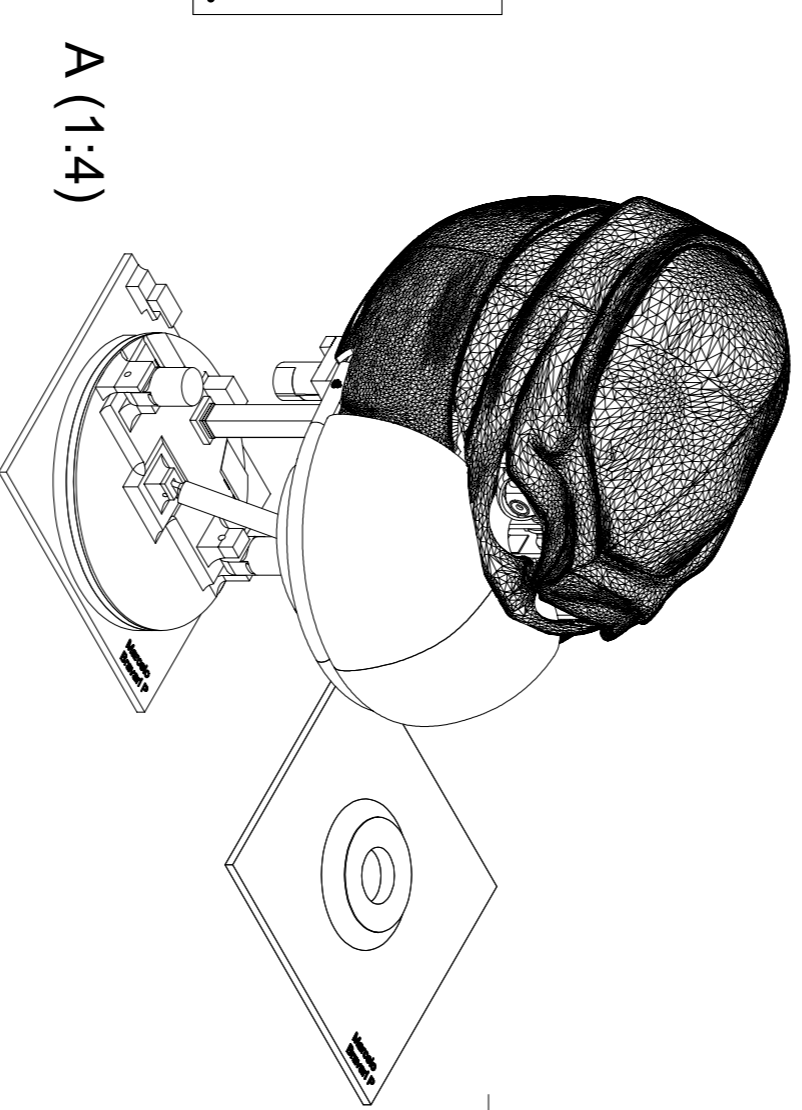
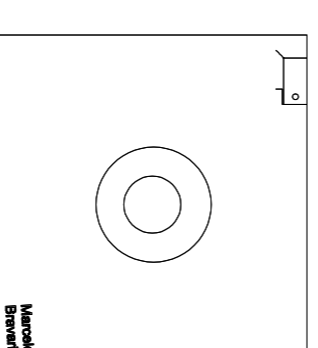
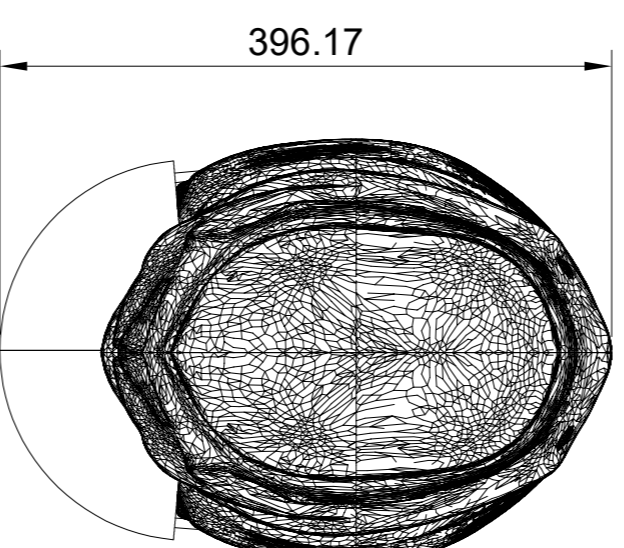
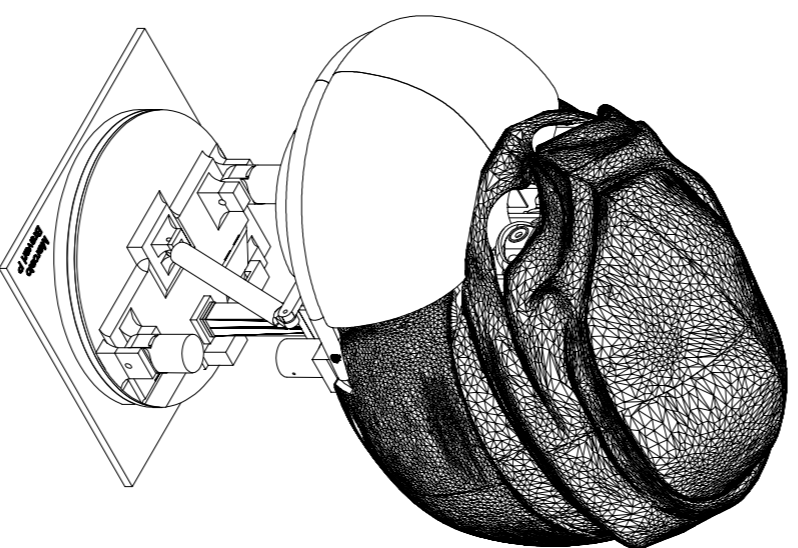
    main()

```

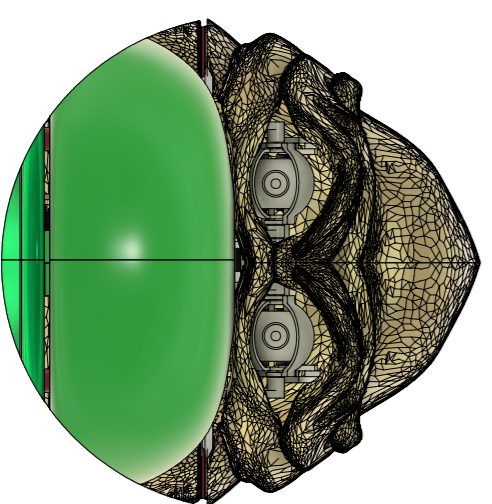
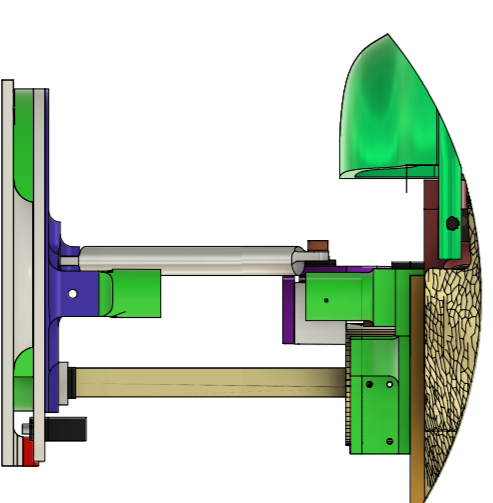
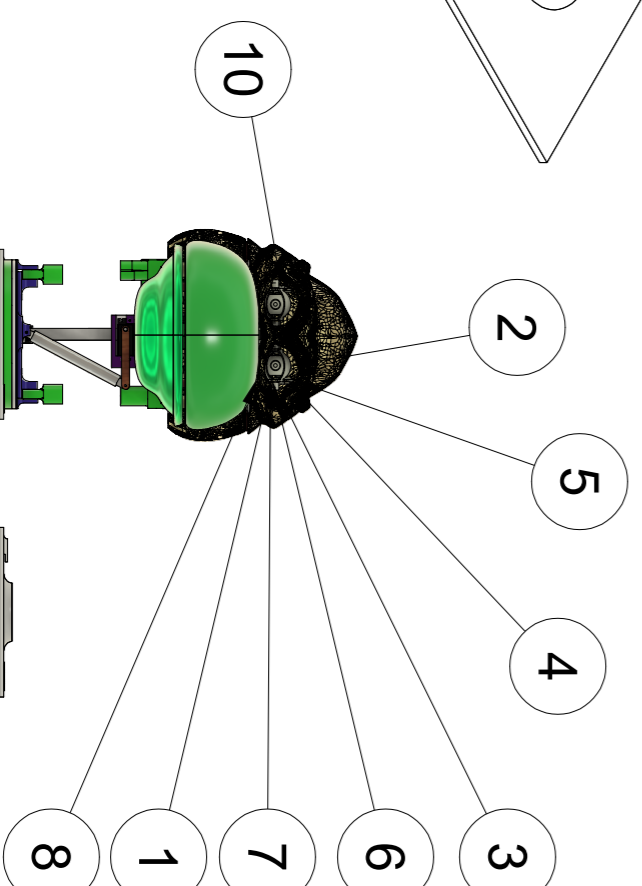
Listado de partes			
Elemento	Cantidad	Número de pieza	Nombre de pieza
1	1	MG90S v1	MG90S v1
2	1	Pa1	Pa1
3	1	Base	Base
4	1	AServo_1	AServo_1
5	1	L1	L1
6	1	Pa2	Pa2
7	1	L2	L2
8	1	AServo_2	AServo_2
9	1	Componente 14	Componente14
10	1	Pa3	Pa3
11	1	Component7 (Mirror)	Component7(Mirror)
12	1	AServo_3	AServo_3
13	1	L3	L3
14	1	Pa4	Pa4
15	1	L4	L4
16	1	AServo_4	AServo_4
17	1	MG90S v1(Mirror)	MG90S v1(Mirror)
18	1	Servo	Servo
19	1	MG90S v1 (1)	MG90S v1 (1)
20	1	Carcasa	Carcasa
22	1	AServo_5	AServo_5
23	1	L5	L5
24	1	L6a	L6a
25	1	L6b	L6b
26	1	Ojo_1	Ojo_1
27	1	MG90S v1 (2)	MG90S v1 (2)
29	1	AServo_6	AServo_6
30	1	L7	L7
31	1	Ojo_D	Ojo_D
32	1	MG996R	MG996R v2
33	1	Animatronic VYT VYT	Animatronic VYT v25
33.5	1	Componente 9	Componente9
33.6	1	Componente 10	Componente10
33.7	1	Componente 11	Componente11
33.10	1	MG996R	MG996R v2
34	1	MG90S v3	MG90S v3 v2



386.41



B (1:4)

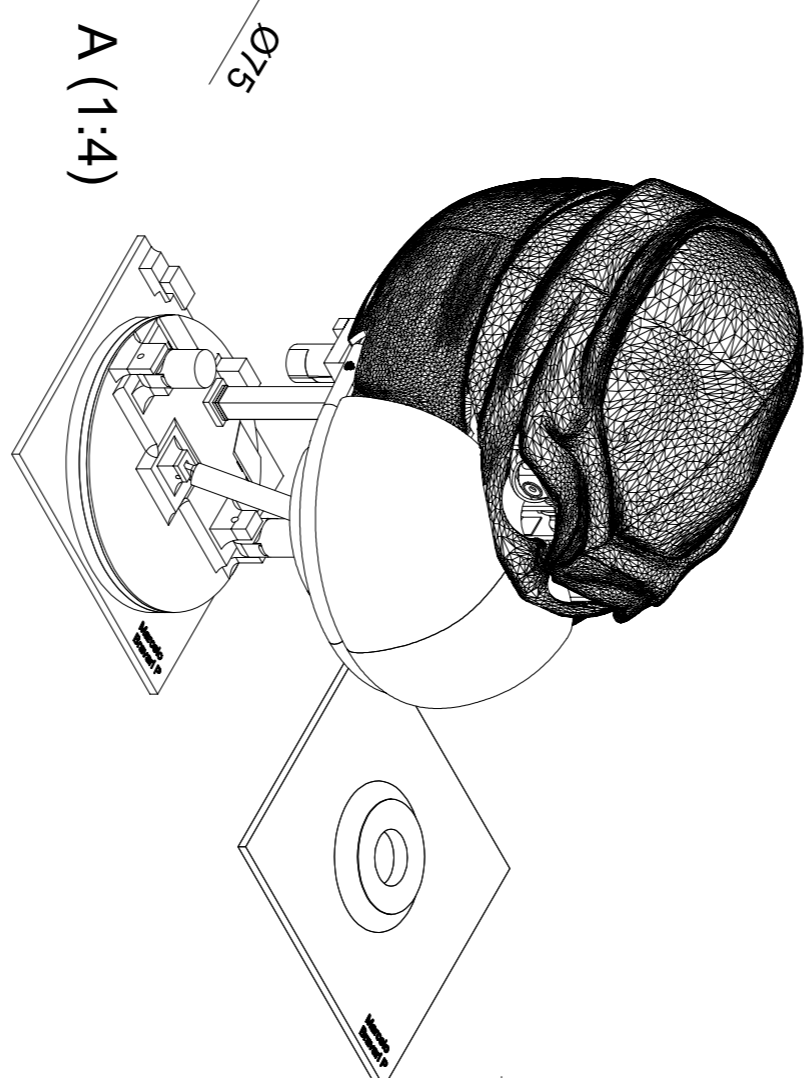
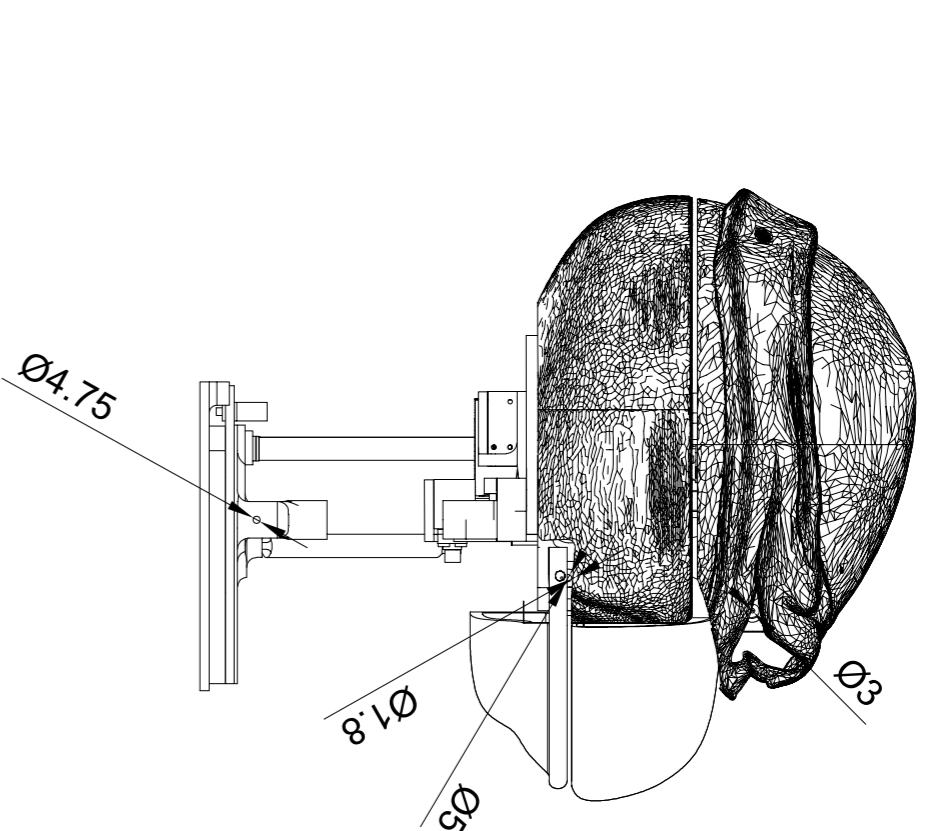
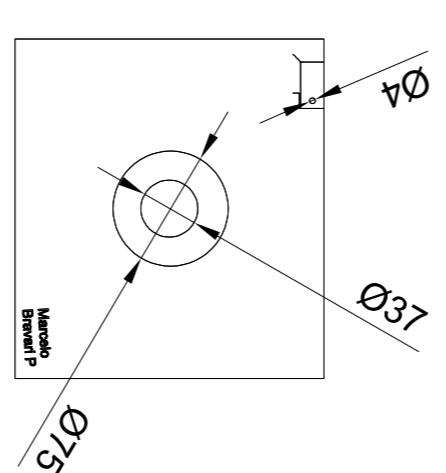
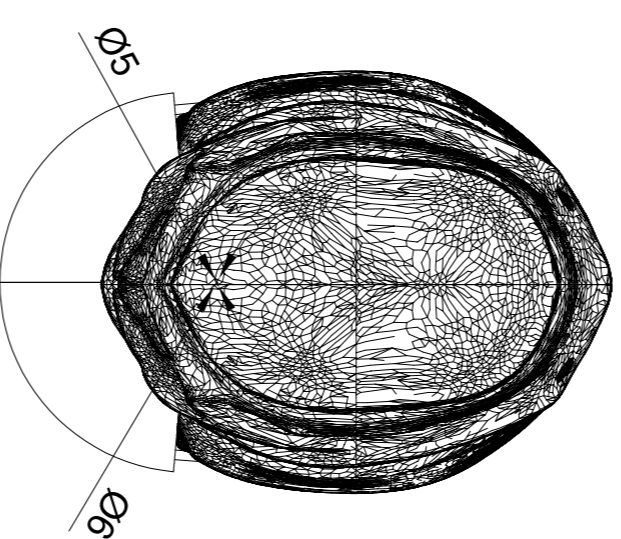
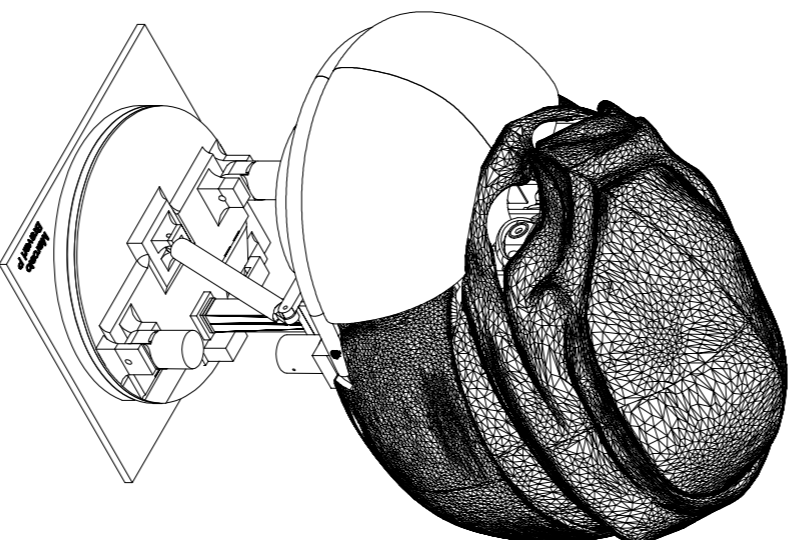
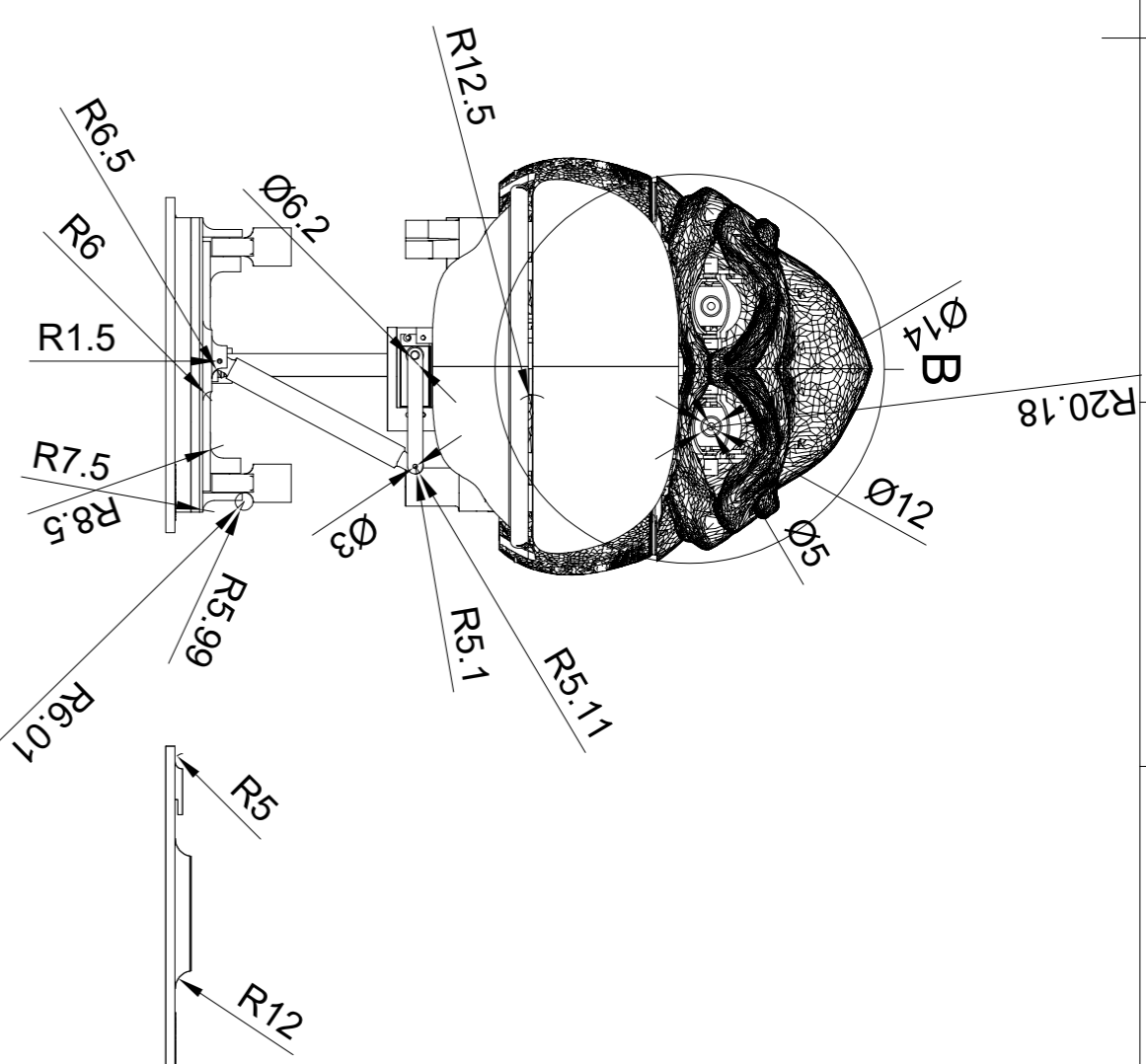
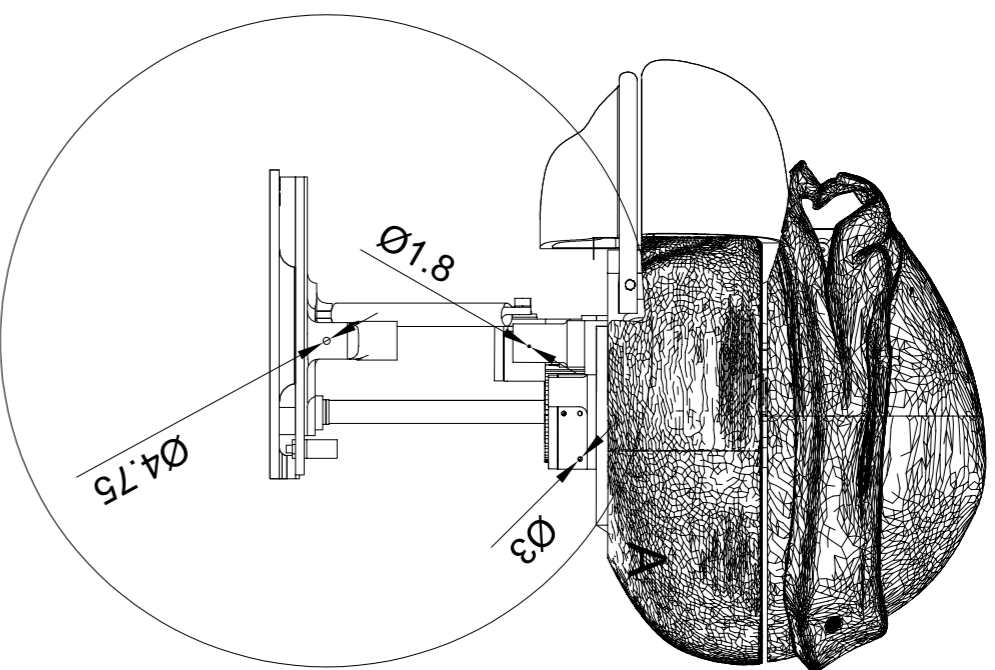


Dept.	Technical reference	Created by	Approved by
		marcelo bravani 21/01/2026	
		Document type	Document status

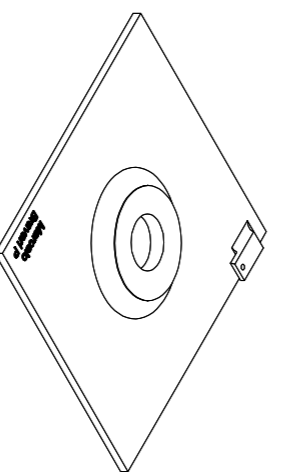
Animatronic geta

Rev.	Date of issue	DWG No.	Sheet
			1/1

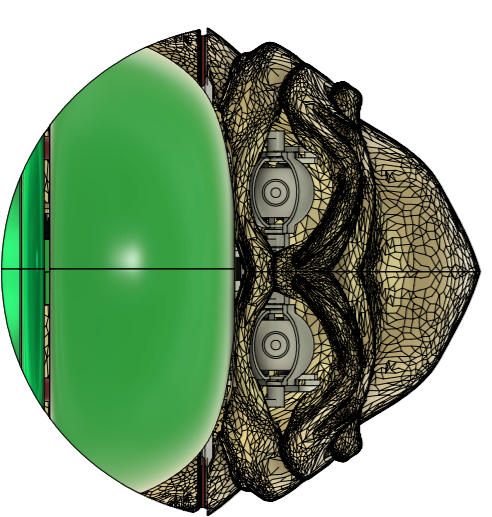
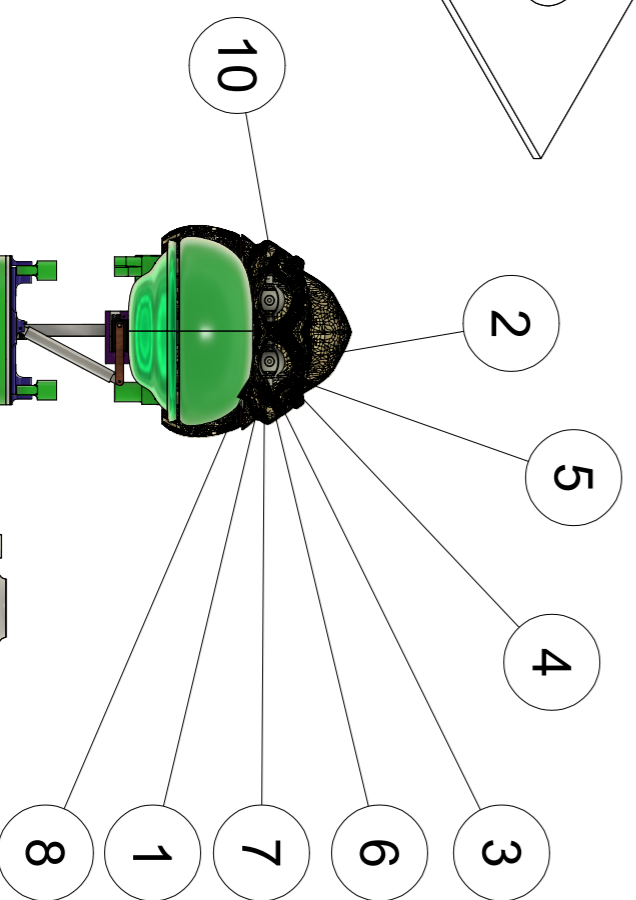
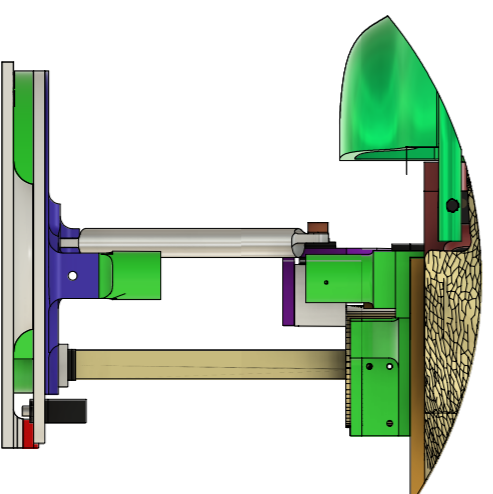
Listado de partes			
Elemento	Cantidad	Número de pieza	Nombre de pieza
1	1	MG90S v1	MG90S v1
2	1	Pa1	Pa1
3	1	Base	Base
4	1	AServo_1	AServo_1
5	1	L1	L1
6	1	Pa2	Pa2
7	1	L2	L2
8	1	AServo_2	AServo_2
9	1	Componente 14	Componente14
10	1	Pa3	Pa3
11	1	Component7 (Mirror)	Component7(Mirror)
12	1	AServo_3	AServo_3
13	1	L3	L3
14	1	Pa4	Pa4
15	1	L4	L4
16	1	AServo_4	AServo_4
17	1	MG90S v1(Mirror)	MG90S v1(Mirror)
18	1	Servo	Servo
19	1	MG90S v1 (1)	MG90S v1 (1)
20	1	Carcasa	Carcasa
22	1	AServo_5	AServo_5
23	1	L5	L5
24	1	L6a	L6a
25	1	L6b	L6b
26	1	Ojo_1	Ojo_1
27	1	MG90S v1 (2)	MG90S v1 (2)
29	1	AServo_6	AServo_6
30	1	L7	L7
31	1	Ojo_D	Ojo_D
32	1	MG996R	MG996R v2
33	1	Animatronic VYT VYT	Animatronic VYT v25
33.5	1	Componente 9	Componente9
33.6	1	Componente 10	Componente10
33.7	1	Componente 11	Componente11
33.10	1	MG996R	MG996R v2
34	1	MG90S v3	MG90S v3 v2



B (1:4)



A (1:4)



Dept.	Technical reference	Created by	Approved by
		marcelo bravani 21/01/2026	
		Document type	Document status
Title		DWG No.	
Animatronic geta			
Rev.	Date of issue		
		Sheet	1/1