



## TRABAJO FIN DE GRADO

# ROBOT MODULAR DE ARTICULACIONES OBЛИCUAS

Jorge Jiménez López

05 TRABAJO FIN DE GRADO

INDUSTRIALES

TRABAJO FIN DE GRADO PARA  
LA OBTENCIÓN DEL TÍTULO DE  
GRADUADO EN INGENIERÍA EN  
TECNOLOGÍAS INDUSTRIALES

SEPTIEMBRE 2023

**Jorge Jiménez López**

DIRECTOR DEL TRABAJO FIN DE GRADO:  
**Antonio Barrientos Cruz**

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES  
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid  
Tel.: 91 336 3060  
info.industriales@upm.es

[www.industriales.upm.es](http://www.industriales.upm.es)

*“Eppur si muove”*

Galileo Galilei



## Agradecimientos

A mi familia, en especial a mis padres y mi hermano, por su apoyo incondicional. Por haberme aguantado en esta etapa y haber estado siempre ahí cuando les he necesitado. También a mi sobrino por haber iluminado mi vida. Espero poder ser una referencia en tu futuro.

A Ale, mi compañera de viaje. Porque sin ella me habría rendido. Ha confiado en mí más que yo mismo y ha cargado conmigo cuando yo no podía.

A los *makers* de ASROB, por su inestimable ayuda y consejo y por todo el desarrollo creativo llevado a cabo en los campamentos de robótica.

A mis compañeros de Strip Saints por ser mi oasis de paz en la tormenta.

A una gran parte del profesorado de esta escuela, por su calidad docente y humana, por su dedicación y consejo. En especial a los profesores Ignacio López Paniagua, Nieves Jimeno Aguilar, Eduardo de la Torre Arranz y por supuesto a mi tutor Antonio Barrientos Cruz.

Y por último a la sanidad pública. Porque sin su existencia yo no estaría aquí escribiendo estos agradecimientos y varias de las personas a quienes van dedicados tampoco estarían para leerlos.



## **Resumen**

En este proyecto se ha diseñado, fabricado y construido un robot manipulador modular de articulaciones oblicuas. Es un robot de tipo serpiente que consta de 6 grados de libertad. La configuración oblicua de las articulaciones, sumado al número de estas, hacen que el robot sea muy versátil y realice movimientos complejos de manera sencilla.

El control significativo de este tipo de robots es muy complejo, siendo un tema objeto de estudio en la actualidad. Con este brazo robótico se pretende ilustrar la versatilidad de sus movimientos con un control de posición en bucle cerrado sencillo.

El diseño del robot ha sido llevado a cabo íntegramente por el autor. Consta de 7 segmentos (6 segmentos móviles y una base fija). Con la acción de 6 servomotores describe movimientos de rotación simple. Esta rotación se describe en un ángulo determinado. El eje de rotación cambia entre los 30° y los -30° con respecto a la vertical.

Los segmentos son totalmente modulares por lo que es muy sencillo modificar el número de articulaciones y adaptar el manipulador a las necesidades requeridas.

Además, se ha realizado una amplia documentación para tratar de explicar en qué punto del desarrollo se encuentra esta tecnología.

## **Abstract**

In this project, it has been designed, made and built a modular handling robot with swivel oblique joints. It is a snake-like robot with 6 degrees of freedom. The oblique configuration in addition with the number of degrees of freedom make the robot very versatile and able to make complex movements in a simple way.

The meaningful control of this kind of robots is very complex. It is a subject under research nowadays. This robot arm is intended to show the versatility of its movements with a simple position closed loop control.

The robot has been designed in its entirety by the author. It consists of 7 segments (6 mobile segments fixed to a base). With the action of 6 servomotors it can describe simple swivel movements. This rotation is made in a fixed angle. The rotation axis angle changes between 30 degrees and minus 30 degrees to the vertical.

The segments are completely modular, so it is very simple to modify the number of joints and adapt the handling robot to its requirements.

In addition, it has been made an extensive documentation in order to explain in which development point is this kind of technology.



## Resumen ejecutivo

**Palabras clave:** Robot manipulador, articulación oblicua, diseño, impresión 3D, encoder.

En la actualidad existe una gran variedad de tipos de robots, en este proyecto se va a estudiar el caso de un robot manipulador modular no redundante con articulaciones oblicuas.

Los objetivos de este trabajo se van a agrupar en dos grandes categorías.

En el ámbito teórico se plantea realizar un estudio cinemático de este tipo de manipuladores a tres niveles; cinemática directa, inversa y modelo diferencial para tratar de plasmar las ventajas y desventajas que presenta el uso de un robot manipulador de estas características. Para la cinemática inversa no va a ser posible desarrollar un modelo, por lo que se va a documentar en qué estado de desarrollo se encuentra su obtención en la actualidad.

En el apartado práctico se va a llevar a cabo el diseño y fabricación tanto de un robot modular como de su interfaz de control. También se va a desarrollar una guía de montaje que permita que cualquier persona sea capaz de ensamblar el robot acorde a las especificaciones.

Es importante empezar realizando una breve clasificación del robot para justificar por qué se trata de un robot manipulador modular no redundante con articulaciones oblicuas.

Un robot se define como manipulador cuando es multifuncional, reprogramable para la realización de tareas diversas y capaz de mover piezas o herramientas mediante trayectorias variables.

Se han diseñado las parejas de barra-articulación de tal manera que los segmentos sean iguales entre sí y que las uniones entre segmentos sean idénticas. Esto permite la concatenación de tantos módulos como se quiera.

Las articulaciones oblicuas son aquellas que describen un giro en torno a un eje con una inclinación determinada y distinta de  $0^\circ$  y  $90^\circ$ . Es un tipo de articulaciones muy versátil que permiten la realización de movimientos complejos de forma simple, pero que, debido a la alta no linealidad que presentan, hacen que su control sea extremadamente complejo cuando el número de articulaciones de este tipo es elevado.

Un robot redundante es aquél que posee más grados de libertad que los que son estrictamente necesarios para el correcto posicionamiento y orientación del efecto del robot. Para robots planos, que realizan movimientos bidimensionales, el número de grados de libertad mínimo es 3, dos que proporcionan la posición y un tercero para la orientación.

En cambio, en el espacio se necesitan 6 grados de libertad. Tres para obtener las coordenadas cartesianas del punto ( $x$ ,  $y$ ,  $z$ ) y otros tres para obtener su orientación. Para eso se suelen considerar los ángulos de Euler ( $\Phi$ ,  $\theta$ ,  $\Psi$ ) que son el de precesión, nutación y rotación propia.

El robot manipulador objeto de estudio posee 6 grados de libertad, por lo que se trata de un robot no redundante. Sin embargo, la naturaleza modular de su estructura permite concatenar de manera muy sencilla tantos segmentos como se quiera dentro de unas limitaciones físicas, y, por tanto, modificar el grado de redundancia. Toda la documentación y desarrollo se va a llevar a cabo teniendo en cuenta que el robot se puede convertir en redundante simplemente añadiendo más módulos.

Para llevar a cabo el control de posición ha sido necesario realizar primero un estudio teórico de la cinemática del robot. Para ello se ha comenzado obteniendo los parámetros de Denavit-Hartenberg que describen únicamente cómo se enlaza un eslabón de la cadena cinemática con el siguiente. Una vez obtenidos estos parámetros se pueden obtener las matrices de

cambio de base que relacionan el sistema de referencia de un eslabón con el sistema de referencia del siguiente. Para obtener la relación del extremo del manipulador con respecto a la base, basta con concatenar todas las matrices de cambio de base y multiplicarlas. Lo que da como resultado una matriz formada por una submatriz de tipo traslación, que proporciona las coordenadas cartesianas y otra submatriz de tipo rotación de la que se obtienen los ángulos de Euler.

Lamentablemente el tipo de articulación hace que en las matrices de cambio de base aparezcan un elevado número de funciones trigonométricas. Al multiplicar 6 matrices, la no linealidad se dispara. La dificultad en los cálculos crece exponencialmente con el número de articulaciones.

El problema es complicado para la obtención de un modelo cinemático directo, pero en la actualidad no existe un método de resolución del modelo cinemático inverso. Entre otras cosas esto se debe a que no se puede aplicar el criterio de simplificación de Pieper, puesto que los últimos 3 grados de libertad no se cortan en un punto. Por tanto, se ha tratado de documentar en qué estado se encuentra la tecnología actualmente. Los últimos estudios utilizan inteligencia artificial y algoritmos genéticos para intentar dar solución al modelo.

Se ha planteado en qué consisten los modelos diferenciales de los robots, haciendo distinción entre el estudio de la matriz jacobiana analítica y la matriz jacobiana geométrica. Sin embargo, tampoco se ha dado una solución analítica a este problema porque habría que hacer el cálculo de las derivadas parciales con el elevado número de funciones trigonométricas antes mencionado.

También se ha desarrollado el método que ha permitido el diseño y programación de un robot modular con articulaciones oblicuas. Desde la concepción a partir de pocas especificaciones, hasta la construcción y puesta en marcha.

Para ello se han utilizado distintas herramientas de diseño y se ha llevado a cabo un proceso de prototipado rápido. Los diseños se fabricaban con ayuda de una impresora 3D y se analizaba su viabilidad de manera práctica. Esto ha permitido agilizar el proceso, si bien ha supuesto un gasto extra de recursos. Las piezas finales se han impreso con la misma impresora 3D pero con un termoplástico con mejores propiedades mecánicas.

El control del robot se realiza a través de una interfaz gráfica de usuario desarrollada en el app designer de Matlab. Ésta se comunica con una placa de desarrollo Arduino Mega 2560 que es la encargada de controlar el manipulador.

Se han dimensionado y empleado servomotores de rotación continua y se ha dado solución a la problemática del control de posición. Dicho control se ha hecho en un bucle cerrado con la ayuda de varios controladores PID. A los controladores les llegan las medidas tomadas por varios encoders de efecto Hall mediante el protocolo de comunicación I2C. Ha sido necesario diseñar y adaptar nuevas piezas para sujetar la placa y el imán del encoder, y así obtener un control de posición efectivo, con error nulo y sin deriva.

Los encoders empleados tienen el inconveniente de que poseen una dirección I2C fija, por lo que ha sido necesario disponer de un componente extra. Un multiplexor de direcciones I2C.

También ha sido necesario desarrollar distintos programas para realizar diversas pruebas y ensayos. Un código para hacer un barrido de direcciones I2C y verificar que la placa de Arduino reconoce el multiplexor, mostrando su dirección. Otro programa para realizar un barrido en los puertos I2C del multiplexor, y verificar que se detectan los encoders. Un tercer programa que indica la intensidad del campo detectado por el encoder para saber si la

distancia al imán es adecuada. Diversos programas para test de rotación, posición y comunicación con Matlab y por último el programa final que permite el control del robot.

Se ha desarrollado una guía de montaje paso a paso donde se ilustra el método de fabricación y cómo han de ensamblarse las distintas piezas, así como el método de calibración utilizado en los imanes.

Por último, se ha desarrollado un estudio de la viabilidad económica del proyecto, así como de las repercusiones a distintos niveles que pueda tener, como son los aspectos sociales o medioambientales.

**Códigos UNESCO:**

3304.91 Robótica

3310.88 Impresión 3D



## Índice

1. Introducción.....	11
1.1 Motivación.....	11
1.2 Marco.....	11
1.3 Objetivos.....	12
1.4 Estructura del documento .....	14
2. Antecedentes y estado del arte .....	15
2.1 Antecedentes .....	15
2.2 Estado del Arte .....	16
3. Estudio cinemático de un brazo con articulaciones oblicuas.....	23
3.1. Parámetros Denavit-Hartenberg .....	23
3.2. Modelo cinemático directo .....	26
3.3. Modelo cinemático inverso.....	26
3.4. Modelo diferencial.....	28
3.5. Ventajas y desventajas .....	29
4. Implementación.....	35
4.1 Diseño .....	35
4.1.1 Primera etapa, planteamiento del robot: .....	35
4.1.2 Segunda etapa, diseño del rodamiento: .....	36
4.1.3 Tercera etapa, planteamiento de los segmentos:.....	39
4.1.4 Cuarta etapa, diseño de los segmentos:.....	41
4.1.5 Quinta etapa, diseño del control de posición: .....	46
4.1.6 Sexta etapa, diseño de los soportes del encoder: .....	48
4.1.7 Séptima etapa, programación de un grado de libertad: .....	51
4.1.8 Octava etapa, diseño de la base, GUI y puesta en marcha: .....	52
4.2. Diseño 3D y fabricación .....	55
4.3. Dimensionamiento de los servomotores. ....	58
4.4 Electrónica y conexiones. .....	61
5. Guía de montaje.....	65
6. Resultados y discusión.....	75
6.1 Experimentos.....	76
6.2 Impactos .....	77
6.2.1 Aspectos legales: .....	77
6.2.2 Aspectos medioambientales: .....	77
6.2.3 Aspectos económicos: .....	77
6.2.4 Aspectos éticos: .....	77

6.2.5 Aspectos de seguridad laboral:	77
6.2.6 Objetivos de Desarrollo Sostenible	77
7. Conclusiones	79
7.1 Líneas Futuras	79
8. Bibliografía	81
Anexo I. Estructura de descomposición del proyecto (EDP)	85
Anexo II. Planificación temporal	86
Anexo III. Estudio Económico	87
Anexo IV. Índice de figuras	90
Anexo V. Índice de tablas	93
Anexo VI. Glosario y abreviaturas	94
Anexo VII: Dimensionamiento de servomotores	95
	95
Anexo VIII. Código	96

## 1. Introducción

### 1.1 Motivación

Este proyecto nace del deseo de estudiar el uso de articulaciones oblicuas en robots manipuladores.

Para ello se va a documentar y desarrollar todo el marco teórico necesario para comprender el funcionamiento de un robot manipulador con articulaciones oblicuas. Tratando de esta manera de dar respuesta a las ventajas e inconvenientes que tiene su uso en la industria.

Para ilustrar las conclusiones a las que se llegue en el análisis teórico se va a diseñar, fabricar y programar un robot manipulador no redundante con 6 grados de libertad. El diseño de este robot va a ser de carácter modular para facilitar la modificación de manera simple del número de grados de libertad, y, por tanto, la redundancia.

Se va a desarrollar una guía de montaje que ilustre todo el proceso de construcción del robot.

Por último, se va a llevar a cabo un breve estudio de las repercusiones a distintos niveles, ético, medioambiental, etc. así como un estudio que analice la viabilidad económica del proyecto.

### 1.2 Marco

Para entender por qué el robot de este proyecto es manipulador, modular, no redundante y de articulaciones oblicuas se va a desarrollar una breve explicación de cada apartado.

La clasificación de **manipulador** en robótica industrial hace referencia a aquellos robots que son multifuncionales, reprogramables y orientados a mover piezas o herramientas según trayectorias variables.

El diseño se va a hacer de tal manera que los segmentos sean iguales y las uniones entre ellos idénticas, lo que le va a dar el carácter **modular**. De esta manera es posible concatenar tantas articulaciones como se quiera.

Para analizar la redundancia se va a hacer una clasificación en cuanto a grados de libertad:

Para que un robot alcance todos los puntos dentro de su espacio de trabajo con la orientación deseada se necesita un número de grados de libertad que va a depender de si el movimiento del robot puede considerarse plano o si por el contrario es un movimiento tridimensional.

Para robots que trabajan en un espacio plano, son necesarios 3 grados de libertad. Dos para fijar la posición y un tercero para fijar la orientación.

En cambio, en un espacio con tres dimensiones son necesarios 6 grados de libertad. Tres para la posición y otros tres para la orientación.

Por tanto, podemos clasificar los robots en base a este criterio como:

**Deficientes:** son aquellos robots que poseen menos grados de libertad de los necesarios para situarse en cualquier punto dentro de su espacio de trabajo con el ángulo elegido. La cinemática tridimensional de este tipo de robots cumple la siguiente desigualdad [1]:

$$\dim(\bar{q}) = n < \dim(\bar{X}_{ee}) = 6 \quad (1.1)$$

Siendo  $\vec{q}$  el vector de los valores de las articulaciones,  $n$  el número de articulaciones,  $\vec{x}_{ee}$  el vector de las posiciones y orientaciones del extremo del robot.

**No redundantes:** son aquellos que tienen el número justo de grados de libertad y por tanto pueden realizar cualquier tarea dentro de su espacio de trabajo. Por tanto, en este tipo de robots se cumple [1]:

$$\dim(\vec{q}) = n = \dim(\vec{x}_{ee}) = 6 \quad (1.2)$$

**Redundantes:** aquellos donde el número de grados de libertad es mayor que lo estrictamente necesario, por lo que no solo son capaces de realizar cualquier tarea programada dentro de su espacio de trabajo, sino que además son capaces de esquivar obstáculos. Estos robots cumplen [1]:

$$\dim(\vec{q}) = n > \dim(\vec{x}_{ee}) = 6 \quad (1.3)$$

Si el número de grados de libertad es muy superior al mínimo necesario, se le denomina robot hiper-redundante.

En base a este criterio queda claro que el robot manipulador objeto de estudio en este trabajo es **no redundante**. Sin embargo, la característica modular del diseño permite añadir de manera simple tantos segmentos y grados de libertad como se quiera, dentro de unas limitaciones físicas. El grado de redundancia, por tanto, se puede modificar simplemente añadiendo más segmentos. Toda la documentación y desarrollo se va a efectuar teniendo en cuenta esta consideración.

Por último, las **articulaciones oblicuas** son aquellas que describen un giro en torno a un eje con una inclinación determinada y distinta de  $0^\circ$  y  $90^\circ$ . Es un tipo de articulaciones muy versátil que permiten la realización de movimientos complejos de forma simple. En cambio, presentan una elevada complejidad a la hora de llevar a cabo el control puesto que este tipo de articulaciones añaden una elevada no linealidad a la cinemática.

### 1.3 Objetivos

Con este trabajo se busca diseñar y construir un robot manipulador modular de articulaciones oblicuas y su interfaz gráfica de usuario. Así como realizar el estudio necesario para su control, y de esta manera poder utilizar el robot para ilustrar el movimiento característico de este tipo de articulaciones.

Se va a realizar un estudio del estado del arte sobre el uso y evolución de las articulaciones oblicuas en robots manipuladores para comprender su aplicación en la industria y analizar ventajas e inconvenientes frente a otros robots convencionales.

Los objetivos de este trabajo por tanto se pueden dividir en dos grandes grupos. Uno teórico y otro práctico.

Dentro de los objetivos teóricos se encuentra:

**Calcular los parámetros de Denavit-Hartenberg:** Para llevar a cabo un estudio de la cinemática directa se va a emplear el método de Denavit-Hartenberg, por lo que va a ser necesario obtener los parámetros característicos de este método, los cuales van a permitir la obtención de matrices de cambio de base entre los distintos sistemas de coordenadas de los segmentos.

**Calcular un modelo cinemático directo mediante dichos parámetros:** Se van a concatenar las matrices de cambio de base obtenidas para desarrollar un modelo cinemático directo.

**Documentar la problemática en la obtención de un modelo cinemático inverso:** Este tipo de robots presenta una elevadísima no linealidad, por lo que el cálculo de un modelo cinemático inverso es una problemática que actualmente está siendo objeto de estudio. Se pretende documentar el estado de desarrollo, así como plantear un método generalista para el cálculo de este modelo, explicando por qué no es aplicable en este caso particular.

**Explicar en qué consiste el modelo diferencial con la matriz Jacobiana:** Debido a la altísima no linealidad antes mencionada, la obtención analítica de un modelo diferencial escapa al alcance de este trabajo, no obstante, se va a plantear en qué consiste y como se obtiene dicho modelo.

**Evaluar ventajas y desventajas** de un robot manipulador de articulaciones oblicuas frente a uno convencional gracias a los modelos calculados y justificar su uso en la industria.

Por otro lado, entre los objetivos prácticos se encuentra:

**Diseñar un prototipo** que permita ilustrar los movimientos característicos de este tipo de articulaciones, planteando qué ángulos se puede dar a los segmentos para que el robot sea práctico y garantizando la modularidad del mismo.

**Diseñar en 3D las piezas que lo componen:** Se van a aprovechar los conocimientos de distintas herramientas de diseño 3D para modelar las piezas necesarias para construir el robot.

**Diseñar un método para el control de posición:** Se van a plantear distintos métodos para el control de posición del manipulador. Se va a optar por un control de posición en bucle cerrado y se van a explicar las ventajas y desventajas de distintos modos de implementación.

**Dimensionar los servomotores que van a mover las articulaciones:** Con las piezas ya diseñadas se van a dimensionar los servomotores que van a permitir el movimiento del robot. Éste va a ser un proceso iterativo, ya que, con los servomotores elegidos, será necesario realizar un ajuste en las piezas para añadir las sujetaciones necesarias y éstas van a ver modificada su peso, por lo que será necesario verificar el dimensionamiento.

**Fabricar las piezas por métodos aditivos:** Se va a emplear la impresión 3D de termoplástico. El proceso será el de prototipado rápido, por lo que se imprimirán varias versiones de las piezas y se analizará físicamente su viabilidad. En caso de ser necesario, se volverá a la etapa de diseño y se harán las modificaciones pertinentes hasta llegar a la obtención de unas piezas que cumplan los requisitos establecidos.

**Programar el control de posición en cadena cerrada del brazo:** Se van a realizar los ensayos necesarios para desarrollar la programación que permita llevar a cabo el control de posición.

**Montar todos los componentes y desarrollar una guía de montaje:** se va a llevar a cabo el montaje del robot, ilustrando y documentando paso a paso el proceso para que éste sea repetible por cualquier persona.

**Programar una interfaz:** Para facilitar el manejo del manipulador se va a desarrollar una aplicación que permita controlar cada articulación por separado.

**Calibrar el control de posición:** Con la interfaz desarrollada, será más sencillo calibrar el control de posición para llegar a un funcionamiento óptimo.

Además, se va a realizar un estudio de los costes tanto directos como indirectos que implican el desarrollo del proyecto, así como los impactos que puede suponer a distintos niveles.

## 1.4 Estructura del documento

Este documento está organizado por capítulos y consta de un índice y de varios anexos.

- Capítulo 1. Introducción: Justifica la motivación que ha llevado al desarrollo de este trabajo, da un marco a la clasificación del robot y plantea los objetivos del proyecto.
- Capítulo 2. Antecedentes y estado del arte: Propone una evolución general del nacimiento de la robótica y desarrolla un estudio del estado del arte sobre el uso de articulaciones oblicuas en robots manipuladores
- Capítulo 3: Estudio cinemático de un brazo con articulaciones oblicuas: Lleva a cabo la obtención de los parámetros de Denavit-Hartenberg, la obtención de un modelo cinemático directo y la documentación y un breve análisis del modelo cinemático inverso y diferencial.
- Capítulo 4: Implementación: Desarrolla todo el proceso de diseño desde la concepción hasta la fabricación, programación y puesta en marcha de un manipulador modular con articulaciones oblicuas.
- Capítulo 5: Guía de montaje: Ilustra todo el proceso de ensamblado de las piezas del robot.
- Capítulo 6: Resultados y discusión: Muestra los resultados obtenidos y realiza un breve estudio sobre los impactos a distintos niveles.
- Capítulo 7: Conclusiones: Plantea posibles líneas de desarrollo.
- Capítulo 8: Bibliografía.
- Anexo I: Estructura de descomposición del proyecto.
- Anexo II: Planificación temporal del proyecto.
- Anexo III: Estudio económico.
- Anexo IV: Índice de figuras.
- Anexo V: Índice de tablas.
- Anexo VI: Glosario y abreviaturas.
- Anexo VII: Dimensionamiento de servomotores.
- Anexo VIII: Código.

## 2. Antecedentes y estado del arte

### 2.1 Antecedentes

Hoy en día se está muy familiarizado con el término robot, y hay tantos tipos que es complicado encontrar una definición que se ajuste adecuadamente a todos.

El origen de la palabra robot nació gracias al escritor y dramaturgo checo Karel Čapek. Proviene de una obra de teatro de ciencia ficción con el título de *Rossumovi univerzální roboti* que se traduce como “Robots Universales Rossum”. Una obra escrita en 1920 y estrenada en 1921 que alcanzó tal éxito que popularizó la palabra robota, que significa “trabajo forzado” en checo.

No obstante, se tienen registros de mecanismos autómatas con cientos de años de antigüedad. En el antiguo Egipto se desarrolló la clepsidra, un sofisticado reloj de agua que permitía el control del tiempo especialmente durante la noche, cuando los relojes solares perdían su utilidad.

Arquímedes estudió y perfeccionó el uso de la palanca y la polea, que en conjunción permitían la creación de articulaciones.

Herón de Alejandría desarrolló varios tratados de neumática e hidráulica y creó la primera máquina de vapor.

En la edad Media aparecieron los primeros autómatas, como el gallo de Estrasburgo.

En la edad Moderna el ingeniero Jacques Vaucanson perfeccionó la creación de autómatas con la fabricación de mecanismos autónomos como “El flautista” y “El pato con aparato digestivo”.

Leonardo Torres Quevedo construyó su autómata “El ajedrecista” a principios del S.XX Una máquina que, gracias a unos electroimanes situados debajo de un tablero de ajedrez era capaz de jugar de manera autónoma un final de rey y torre contra un rey controlado por un humano y ganar en cada una de las ocasiones.

Sin embargo, es a mediados del S.XX donde se encuentra un punto de inflexión en la robótica. Desde el desarrollo en 1961 del robot de transferencia programable Unimate por parte de la compañía estadounidense Unimation, el crecimiento del uso de robot en la industria ha sido exponencial.

En la actualidad, la Real Academia Española (RAE) define robot como: “*Máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones*”.

“*Máquina que imita la figura y los movimientos de un ser animado*”.

Mientras que la Real Academia de Ingeniería (RAING) lo define como: “*Manipulador multifuncional reprogramable diseñado para mover materiales, componentes, herramientas o dispositivos especializados mediante una serie de movimientos variables programados para la realización de distintas tareas*”.

Por último, la Organización Internacional de Estándares (ISO, International Organization for Standardization) define robot en la norma ISO 8373:2012 como: “*Mecanismo accionado programable en dos o más ejes con un grado de autonomía, que se mueve dentro de su entorno para realizar las tareas previstas*”.

“*Un robot incluye el sistema de control y la interfaz de sistema de control*”.

## 2.2 Estado del Arte

Los robots modulares con la forma de movimiento característica de una serpiente fueron introducidos y desarrollados por el Profesor Shigeo Hirose en el Instituto Tecnológico de Tokio. Hirose estudió la mecánica del movimiento sobre el plano de estas articulaciones [2] y construyó un gran número de robots de este tipo [4].

En 1975 creó el ACM III (Active Cord Mechanism). Consistía en 20 enlaces y su movimiento era únicamente sobre el plano:

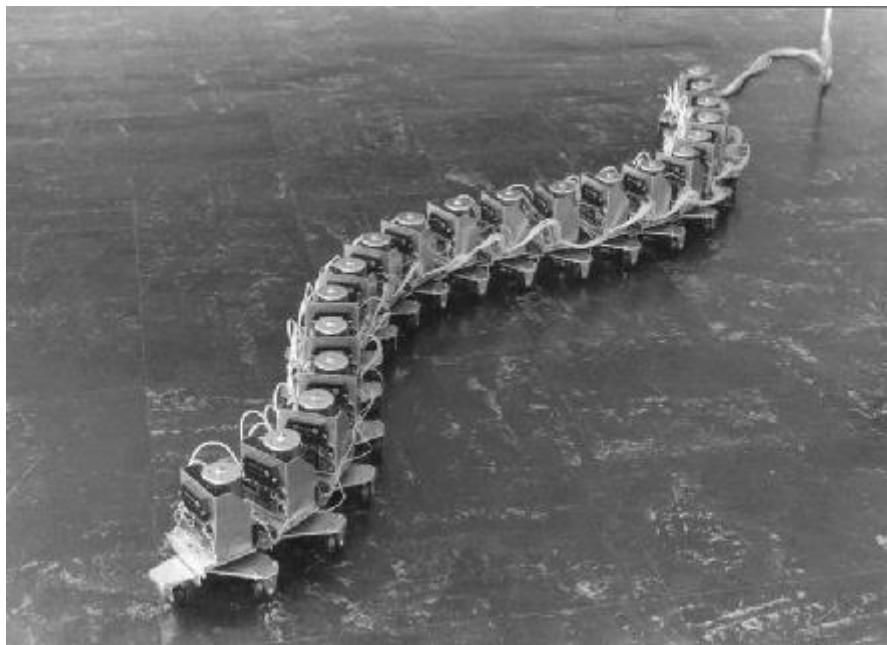


Ilustración 1.1: The Active Cord Mechanism ACM III por Hirose [3].

Más adelante construyó la versión ACM-R2:



Ilustración 1.2: The Active Cord Mechanism ACM-R2 por Hirose [3].

Con el diseño del ACM-R3, convirtió el movimiento plano en movimiento tridimensional:



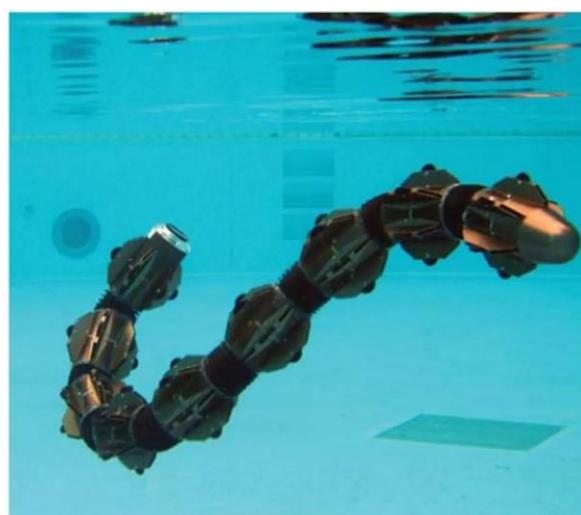
*Ilustración 1.3: The Active Cord Mechanism ACM-R3 por Hirose [3].*

Tras el éxito del ACM-R3 se mejoró la amplitud de movimiento y se añadió la posibilidad de equipar ruedas en los mismos ejes de las articulaciones. Llegando a la versión ACM-R4:



*Ilustración 1.4: The Active Cord Mechanism ACM-R4 por Hirose [3].*

Más adelante desarrolló una versión anfibia, el ACM-R5:



*Ilustración 1.5: The Active Cord Mechanism ACM-R5 por Hirose [3].*

En 1987 Takanashi [5] introdujo las articulaciones con juntas oblicuas. El diseño consiste en dos ejes angulares y dos segmentos oblicuos con ángulos suplementarios. Los segmentos están unidos mediante un engranaje. Esto permite que, al rotar un eje con respecto a otro, se produzca un cono de revolución. El giro del otro eje orienta dicho cono. Coordinando ambas rotaciones la articulación realiza dos movimientos puros, flexión plana y orientación.

Este tipo de unión conecta mecánicamente una sección con la siguiente de dos formas. La primera es a través de un cardán que une los dos ejes y mantiene los enlaces unidos. La segunda es mediante los segmentos oblicuos [6].

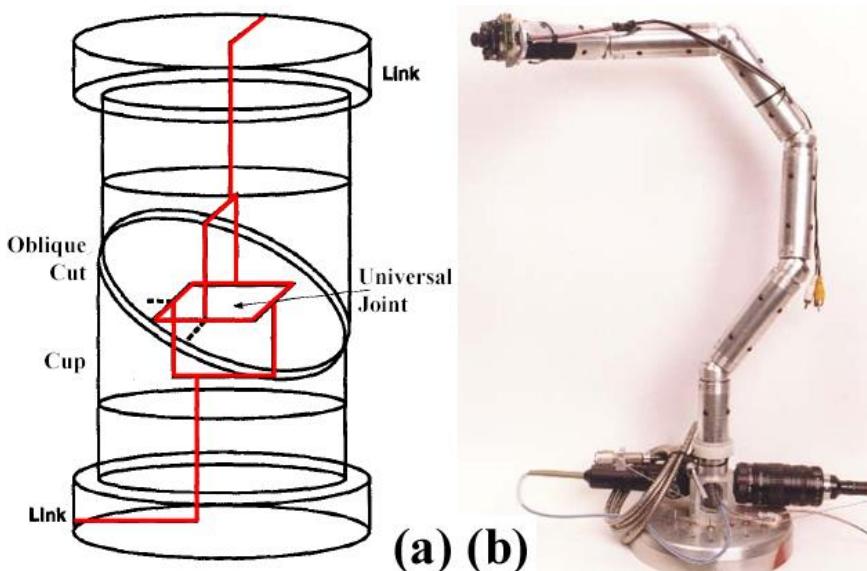


Ilustración 1.6: (a) Diseño de articulación oblicua giratoria. (b) Robot Serpiente de JPL [6].

Esto dio como resultado que en 1995 el *Jet Propulsion Laboratory*, presentase el prototipo de robot manipulador *Serpentine Robot*. Este robot cuenta con 12 grados de libertad. Consta de seis segmentos unidos mediante 5 articulaciones de 2 grados de libertad y en el interior porta un boroscopio que añade los 2 últimos grados. Este diseño supone una mejora en la libertad de movimiento con respecto a los robots convencionales de 6 grados de libertad.

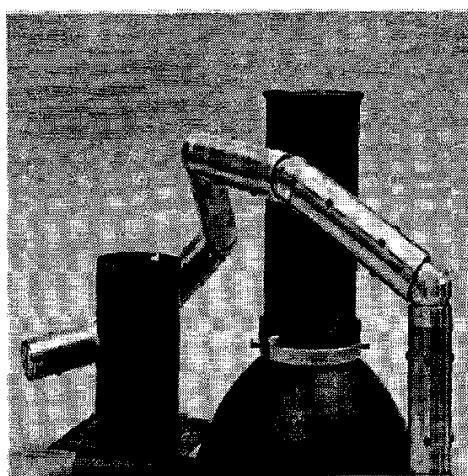


Ilustración 1.7: Robot Serpiente de JPL esquivando un obstáculo [6].

Este tipo de articulaciones sigue en desarrollo, prueba de ello es que en los últimos años han aparecido proyectos muy interesantes como el que se aprecia en el vídeo *LEGO Oblique swivel joint mechanisms* en el canal de youtube *Akiyuki Brick Channel* [7]. Este diseño cuenta

con 4 grados de libertad controlados por servomotores medianos de Lego Mindstorms EV3 [8] que controlan el giro de cada sección.

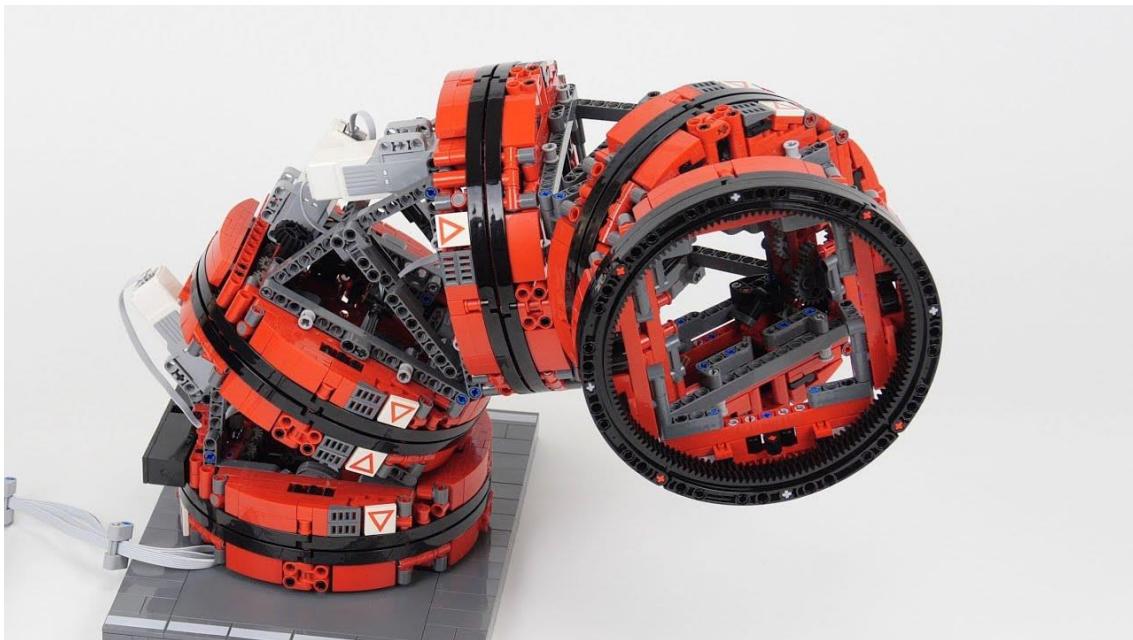


Ilustración 1.8: LEGO oblique swivel joint mechanisms por Akiyuki Brick Channel [8].

Basándose en este diseño y continuando un proyecto de Allen Pan llamado *Giving snakes their legs back* [9], James Bruton ha fabricado un prototipo de exoesqueleto artrópodo para su proyecto *Giving LEGS their SNAKES back* [10]. El robot consta de 6 patas con 3 grados de libertad cada una y nace de la idea de dar un movimiento más orgánico al robot propuesto por Allen Pan.



Ilustración 1.9: *Giving LEGS their SNAKES back* por James Bruton [11].

El movimiento de este tipo de uniones es muy atractivo por su simplicidad, pero su dificultad para un control preciso hace que aún no esté extendido su uso en robots manipuladores. El control requiere de la aplicación de métodos de inteligencia artificial, y se cree que las realidades virtuales inmersivas y el lenguaje natural pueden acabar desempeñando un papel importante en el control de robots redundantes e hiper-redundantes. No obstante, una industria donde si está fuertemente arraigado el uso de este tipo de mecanismos es en la aeronáutica [12].

En 1960 Jack Britt [13] propuso un diseño de tobera rotativa, dividida en segmentos y con motores montados en la pared exterior.

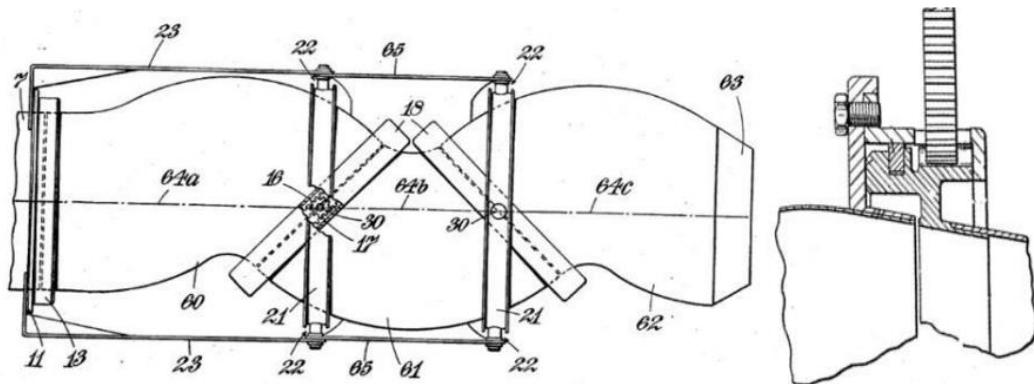


Ilustración 1.10: Estructura de tobera rotativa en tres etapas por Jack Britt [13].

Este diseño fue evolucionando a través de varias etapas con mejoras patentadas por distintos autores como Carroll E. Johnson [14] en 1966 (ilustración 1.11), Gerhard Kopp [15] en 1969 (ilustración 1.12) y Dudley O. Nash [16] en 1972 (ilustración 1.13) hasta que finalmente llegó la primera aplicación práctica con el motor del avión Yakovlev 141 (Yak-141) un caza de combate tipo VTOL (Vertical Take Off and Landing) (ilustración 1.14) [17].

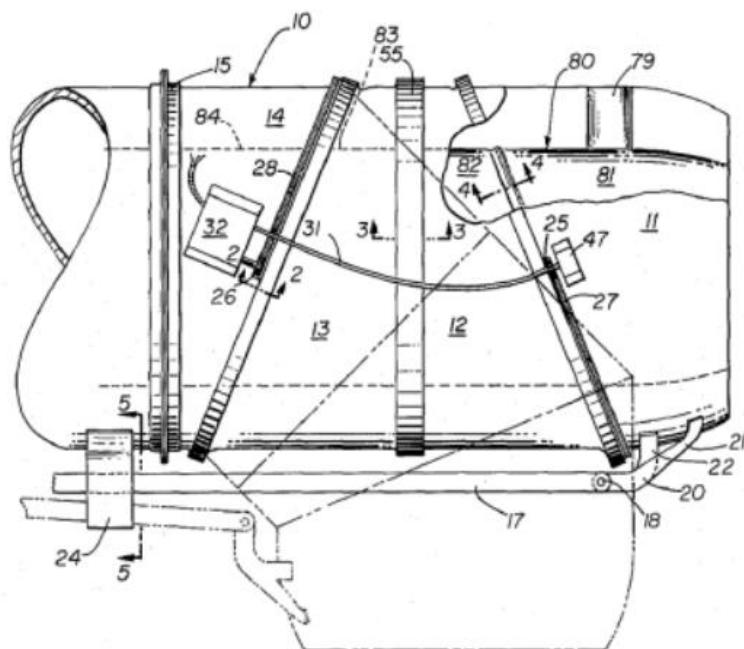


Ilustración 1.11: Estructura de tobera rotativa multietapa por Carroll E. Johnson [14].

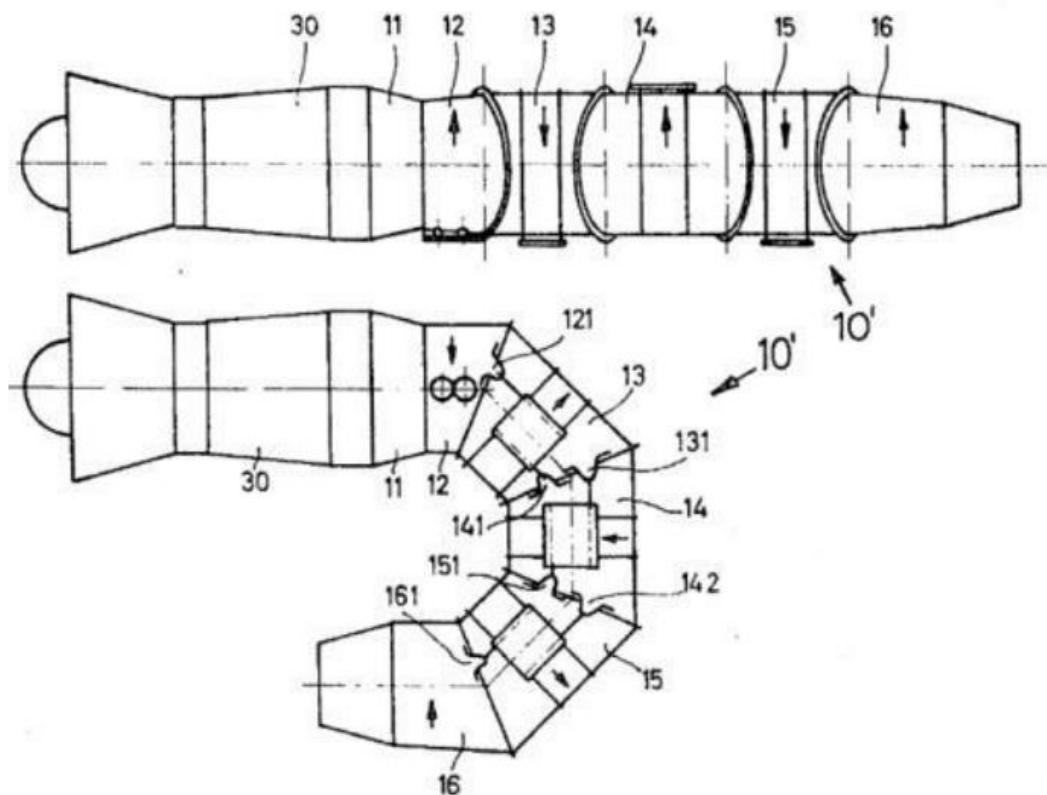


Ilustración 1.12: Estructura de tobera rotativa de cuatro etapas por Gerhard Kopp [15].

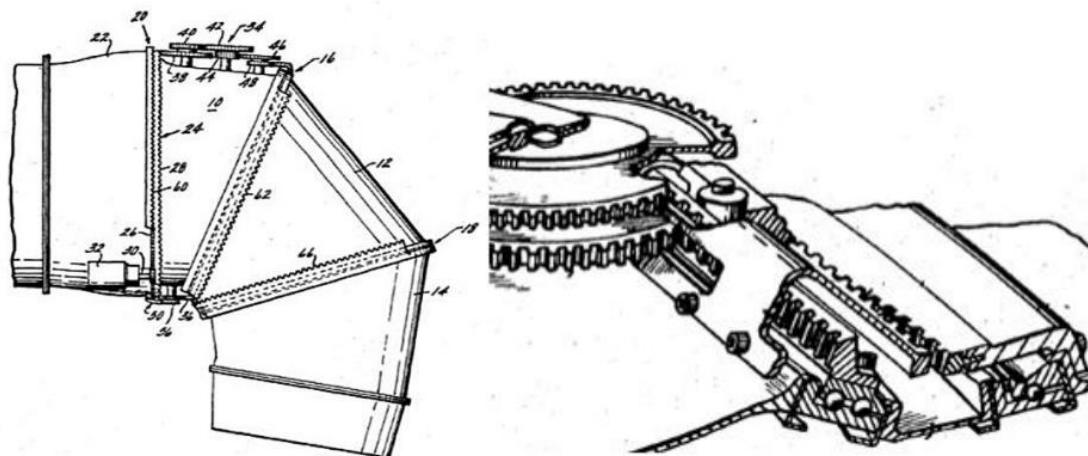


Ilustración 1.13: Sistema de accionamiento de tobera multiengranaje 3BSD (3 Bearing Swivel Duct) por Dudley O. Nash [16].

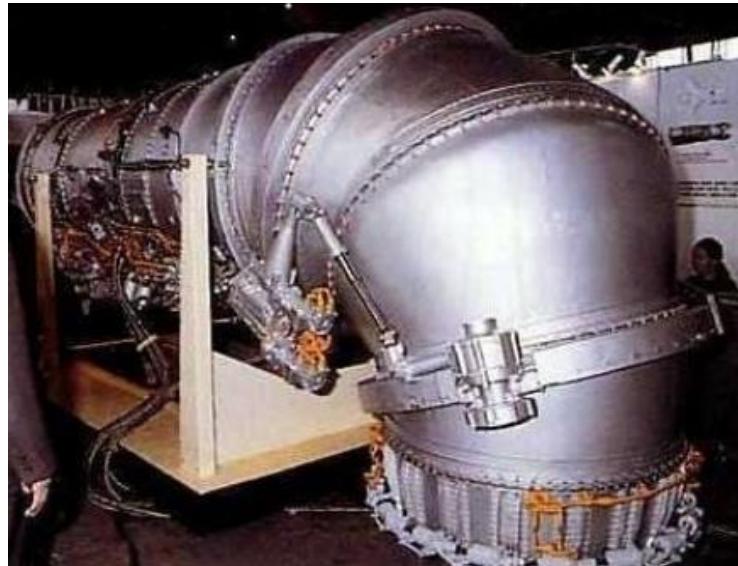


Ilustración 1.14: Motor principal R-97V del avión militar Yakovlev-141[17].

El proceso de desarrollo culminó con la creación del Rolls-Royce LiftSystem® [18] que junto con el motor del caza F135 recibió en 2001 el trofeo Collier por ser el mayor logro aeronáutico y astronáutico del año en los Estados Unidos.



Ilustración 1.15: Módulo rotativo de tres rodamientos (3BSM) del Rolls-Royce LiftSystem [12].

Hoy en día es un diseño muy extendido utilizado por las grandes potencias para sus cazas VTOL o STOVL (Short Take Off and Vertical Landing) [12].

### 3. Estudio cinemático de un brazo con articulaciones oblicuas

A continuación, se va a llevar a cabo un estudio de la cinemática del robot que va a permitir definirlo teóricamente y controlarlo en la práctica.

La cinemática es el estudio del movimiento de un robot con respecto a un sistema de referencia sin tener en cuenta las fuerzas que intervienen en el proceso. Se va a realizar a tres niveles. En primer lugar, se va a tratar el modelo cinemático directo, que establece una relación entre la localización del extremo del robot y los valores de las articulaciones ( $q_1, q_2, q_3, q_4, q_5$  y  $q_6$ ). Luego se va a estudiar la cinemática inversa, que consiste en determinar qué valores articulares ( $q_1, q_2, q_3, q_4, q_5$  y  $q_6$ ) debe adoptar el robot para conseguir una posición y una orientación concreta del extremo dentro del espacio de trabajo. Por último, se va a realizar un estudio del modelo diferencial a través de la matriz Jacobiana. Este modelo establece una relación entre las velocidades de las articulaciones y las del extremo.

Para llevar a cabo este desarrollo es necesario empezar determinando los valores de los parámetros de Denavit-Hartenberg.

#### 3.1. Parámetros Denavit-Hartenberg

El estudio mediante los parámetros de Denavit-Hartenberg es un proceso sistemático que permite describir matricialmente la estructura de una cadena cinemática cuyas articulaciones solo presentan un único grado de libertad. Permite obtener un modelo cinemático directo (MCD) a partir de la realización de cambios de base mediante matrices de transformación homogénea (MTH). El movimiento relativo de dos segmentos puede estar relacionado por 4 transformaciones básicas [19].

- Rotación de un **ángulo**  $\theta_i$  alrededor del eje  $Z_{i-1}$ .
- Traslación de una **distancia**  $d_i$  a lo largo del eje  $Z_i$ . Lo que supone aplicar el vector  $(0, 0, d_i)$ .
- Traslación de una **distancia**  $a_i$  a lo largo del eje  $X_i$ . Lo que supone aplicar el vector  $(a_i, 0, 0)$ .
- Rotación de un **ángulo**  $\alpha_i$  alrededor del eje  $X_i$ .

Estas transformaciones se definen sobre el sistema móvil, por tanto, el método matemático que permite calcular la matriz D-H es la postmultiplicación de matrices de rotación o traslación según el caso. Lo que resulta en una matriz de cambio de base del tipo  ${}^{i-1}A_i$ . La matriz de D-H queda por tanto definida de la siguiente manera [19]:

$${}^{i-1}A_i = \text{Rotz}(\theta_i) T(0, 0, d_i) T(a_i, 0, 0) \text{Rotx}(\alpha_i) \quad (3.1)$$

Siendo **Rotz** la matriz de rotación con respecto al eje Z, **T** la matriz de traslación y **Rotx** la matriz de rotación con respecto al eje X.

Por lo que [19]:

$$\begin{aligned} {}^{i-1}A_i &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} \cos \theta_i & -\cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2) \end{aligned}$$

Para la aplicación de este método se ha procedido a establecer un sistema de coordenadas ortonormal y cartesiano para cada segmento del robot, así como para el sistema fijo.

$$\{S_i\} = (x_i, y_i, z_i) \text{ con } i = 0, 2, \dots, n \text{ (siendo } n \text{ el número de gdl. En este caso } n = 6)$$

Con este sistema se ha desarrollado un gráfico (ilustración 3.1) para facilitar la comprensión de la cadena cinemática del robot, así como la obtención de los parámetros D-H de cada una de las articulaciones. Los triángulos representan rotaciones en torno a la perpendicular a las bases que están juntas (eje  $Z_i$  con  $i = 1, 2, \dots, 6$ ). Nótese que para los giros oblicuos se ha modificado ligeramente la simbología para representar que el robot está recto en estado de reposo.

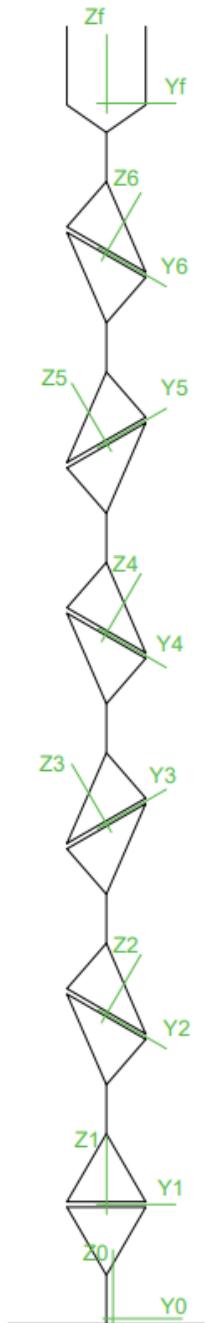


Ilustración 3.1: Cadena cinemática del robot con los sistemas de coordenadas de cada elemento.

Del análisis de la figura, se puede observar que:

- Todos los ejes  $X_i$  son paralelos y salientes del plano del papel, por lo que todos los valores de  $\theta_i$  van a coincidir con el giro de cada una de las articulaciones  $q_i$ .
- Los valores  $d_i$  son constantes que vienen determinadas por la geometría de los segmentos.
- Los valores de  $a_i$  son todos nulos puesto que los centros de todos los sistemas de coordenadas están alineados.
- El ángulo de los ejes  $Z_i$  va oscilando entre  $+30^\circ$  y  $-30^\circ$  con respecto al eje  $Z_0$ .

Con todo esto se puede confeccionar la siguiente tabla:

Articulación	$\theta$	$d$	$a$	$\alpha$
1	$q_1$	53 mm	0	0
2	$q_2$	45mm	0	$-30^\circ$
3	$q_3$	70mm	0	$60^\circ$
4	$q_4$	70mm	0	$-60^\circ$
5	$q_5$	70mm	0	$60^\circ$
6	$q_6$	70mm	0	$-60^\circ$

Tabla 3.1: Parámetros de D-H del robot.

Aplicando estos valores a la matriz D-H se obtienen las siguientes matrices de cambio de base:

$${}^0A_1 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & 53 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

$${}^1A_2 = \begin{bmatrix} \cos q_2 & \frac{-\sqrt{3}}{2} \sin q_2 & \frac{-1}{2} \sin q_2 & 0 \\ \sin q_2 & \frac{\sqrt{3}}{2} \cos q_2 & \frac{1}{2} \cos q_2 & 0 \\ 0 & \frac{-1}{2} & \frac{\sqrt{3}}{2} & 45 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

$${}^2A_3 = \begin{bmatrix} \cos q_3 & \frac{-1}{2} \sin q_3 & \frac{\sqrt{3}}{2} \sin q_3 & 0 \\ \sin q_3 & \frac{1}{2} \cos q_3 & \frac{-\sqrt{3}}{2} \cos q_3 & 0 \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

$${}^3A_4 = \begin{bmatrix} \cos q_4 & \frac{-1}{2} \sin q_4 & \frac{-\sqrt{3}}{2} \sin q_4 & 0 \\ \sin q_4 & \frac{1}{2} \cos q_4 & \frac{\sqrt{3}}{2} \cos q_4 & 0 \\ 0 & \frac{-\sqrt{3}}{2} & \frac{1}{2} & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

$${}^4A_5 = \begin{bmatrix} \cos q_5 & \frac{-1}{2} \sin q_5 & \frac{\sqrt{3}}{2} \sin q_5 & 0 \\ \sin q_5 & \frac{1}{2} \cos q_5 & \frac{-\sqrt{3}}{2} \cos q_5 & 0 \\ 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

$${}^5A_6 = \begin{bmatrix} \cos q_6 & \frac{-1}{2} \sin q_6 & \frac{-\sqrt{3}}{2} \sin q_6 & 0 \\ \sin q_6 & \frac{1}{2} \cos q_6 & \frac{\sqrt{3}}{2} \cos q_6 & 0 \\ 0 & \frac{-\sqrt{3}}{2} & \frac{1}{2} & 70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

### 3.2. Modelo cinemático directo

El modelo cinemático directo resuelve un problema estático y geométrico para calcular la posición y orientación del extremo manipulador con respecto a la base. Conocidos los valores de las articulaciones ( $q_1, q_2, q_3, q_4, q_5$  y  $q_6$ ) permite obtener la posición y orientación del extremo ( $x, y, z, \Phi, \theta, \Psi$ ). Siendo  $\Phi, \theta, \Psi$  los ángulos de Euler de precesión, nutación y rotación propia o intrínseca respectivamente.

Para obtener el modelo cinemático directo según el método de Denavit-Hartenberg basta con multiplicar en orden todas las matrices de cambio de base obtenidas con los parámetros de D-H según la siguiente fórmula [19]:

$$\mathbf{T}_n = {}^0\mathbf{A}_1 {}^1\mathbf{A}_2 \dots {}^{n-1}\mathbf{A}_n \quad (3.9)$$

Para el caso particular de 6 articulaciones:

$$\mathbf{T}_6 = {}^0\mathbf{A}_1 {}^1\mathbf{A}_2 {}^2\mathbf{A}_3 {}^3\mathbf{A}_4 {}^4\mathbf{A}_5 {}^5\mathbf{A}_6 \quad (3.10)$$

La matriz  $\mathbf{T}_6$  proporciona por tanto la orientación mediante la submatriz de rotación como la posición mediante la submatriz de traslación del extremo referido a la base en función de las 6 coordenadas articulares ( $q_1, q_2, q_3, q_4, q_5$  y  $q_6$ ). Tiene la forma [19]:

$$\mathbf{T}_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)$$

Por razones evidentes no se va a plantear el valor final de la matriz  $\mathbf{T}_6$  de manera simbólica y se va a realizar el cálculo en Matlab [20] mediante la herramienta Robotics System Toolbox [21] cuando sea necesario para el control del robot.

### 3.3. Modelo cinemático inverso

El estudio cinemático inverso trata de dar valores articulares ( $q_1, q_2, q_3, q_4, q_5$  y  $q_6$ ) conocida la posición y la orientación del extremo del robot ( $x, y, z, \Phi, \theta, \Psi$ ).

Este modelo presenta un problema, y es que la solución generalmente no es unívoca, sobre todo en el caso de los robots redundantes. Además, la solución tampoco tiene por qué ser cerrada. Una solución cerrada es aquella que se puede expresar como [19]:

$$q_i = f_i(x, y, z, \Phi, \theta, \Psi) \quad (3.12)$$

Las ventajas de las soluciones cerradas es que garantizan que se alcance la solución en un tiempo definido. Además, permiten seleccionar la opción más adecuada cuando existen múltiples soluciones.

Un método para buscar soluciones cerradas a robots lineales fue propuesto en 1991 por M. Galicki [22]. No obstante, no aplica restricciones al movimiento de las articulaciones debido a la dificultad añadida del problema, lo que limita su aplicación práctica. En 1989 W. Jacak desarrolló un trabajo pionero en el modelado matemático de un manipulador discreto y redundante plano en espacio cartesiano [23]. Ese mismo año presentó también un modelo de la cinemática de un robot manipulador redundante rotativo mediante una máquina finita de estados [24].

En 2002 se presentó *An improved inverse kinematic and velocity solution for spatial hyper-redundant robots* [25], que utiliza conceptos de curvas Backbone y una aproximación modal para resolver la redundancia de este tipo de robots. En 2012 *A multi-objective approach for the motion planning of redundant manipulators* [26] combina el método pseudo-inverso en bucle cerrado con un algoritmo genético multiobjetivo para controlar las posiciones de las articulaciones [12].

Por último, en 2018, Isiah Zaplana y Luis Basanez, de la Universidad Politécnica de Barcelona propusieron en su trabajo [27] una solución al problema de la cinemática inversa en robots manipuladores redundantes. Hacen un análisis del espacio de trabajo, seleccionan una serie de articulaciones que se denominan *articulaciones redundantes* y parametrizan sus variables articulares, transformando el problema redundante en no redundante.

Aún no existe un método general que resuelva la cinemática inversa de un robot lineal con un elevado número de grados de libertad y sigue siendo un campo de estudio activo, por tanto, se va a realizar una breve exposición de cómo se puede plantear, de manera simple y general, la búsqueda, en caso de existir, de estas soluciones cerradas.

Se puede optar bien por métodos geométricos o bien por manipular las ecuaciones del MCD. La dificultad de ambas opciones crece exponencialmente con el número de grados de libertad.

Los métodos geométricos consisten en buscar relaciones geométricas o trigonométricas entre las coordenadas en el espacio de trabajo y las coordenadas articulares  $q_i$ . Es un método difícilmente sistematizable y depende de la habilidad de quien lo resuelve. Además, la dificultad de las relaciones como ya se ha dicho crece de manera exponencial con el incremento de los gdl. Haciendo este método aplicable tan solo en casos de robots con pocos grados de libertad.

Para llevar a cabo el método de manipulación de las ecuaciones del MCD habría que despejar las 6 incógnitas  $q_1, \dots, q_6$  en el sistema de 12 ecuaciones no lineales  $n_x, \dots, p_x$  de la ecuación (3.11). Este método es completamente inutilizable para robots con muchos grados de libertad.

En algunos casos particulares se puede desacoplar la cinemática del robot y dividir el problema cinemático de 6 gdl en dos más sencillos de 3 gdl cada uno. Un primer problema que resuelve la posición donde debe estar la muñeca del robot y obtiene  $q_1, q_2$  y  $q_3$  y un segundo que, conociendo los valores ya calculados orienta la muñeca y obtiene  $q_4, q_5$  y  $q_6$ .

Para ello se tiene que cumplir la llamada *Condición de Pieper* [28]. Este desacople cinemático es aplicable cuando los últimos 3 grados de libertad se cortan en un punto [19]. Lamentablemente este no es el caso del robot objeto de estudio, por lo que la simplificación no se puede utilizar.

Existen no obstante otros métodos de carácter iterativo que tienen la ventaja de que son generalizables para cualquier robot. En cambio, presentan problemas en la convergencia de las soluciones y los mínimos locales.

Estos métodos parten del conocimiento del MCD. Se propone una solución y se establece una función error a minimizar. A continuación, se itera la propuesta de dicha solución intentando mejorarla minimizando el error. Este método también es muy laborioso para robots de un elevado número de gdl.

Es por esto por lo que, nuevamente se va a recurrir al Robotics System Toolbox [21] de Matlab [20] para el control del robot y no se va a profundizar más en la resolución teórica del MCI.

### 3.4. Modelo diferencial

El modelo diferencial trata de establecer una relación entre las velocidades y aceleraciones del extremo del robot con las velocidades de las variables articulares  $q_i$ . Esta relación normalmente la utiliza el sistema de control para establecer con qué velocidades deben trabajar cada una de las articulaciones para conseguir que el extremo del robot describa una trayectoria definida a lo largo del tiempo.

Para llevar a cabo el estudio diferencial se utiliza la matriz Jacobiana. En general una matriz Jacobiana es aquella cuyos términos son las derivadas parciales del sistema de ecuaciones multivariable objeto de estudio.

Tienen por tanto la forma:

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1(x) \\ \vdots \\ \nabla f_n(x) \end{bmatrix} \quad (3.12)$$

Siendo  $\nabla f_i(x)$  el gradiente de la i-ésima componente escalar.

En robótica en cambio se pueden tomar dos matrices Jacobianas distintas, según el interés del estudio.

Si se toman las ecuaciones que definen el modelo cinemático se pueden relacionar mediante la Jacobiana las variables articulares ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5$  y  $\dot{q}_6$ ) con las variables de posición y orientación del extremo ( $x, y, z, \dot{\phi}, \dot{\theta}$  y  $\dot{\psi}$ ). Dando lugar a la Jacobiana analítica.

No obstante, puede resultar ventajoso conocer las velocidades instantáneas de traslación y rotación del extremo ( $v_x, v_y, v_z, \omega_x, \omega_y, \omega_z$ ) y relacionarlas con las variables articulares ( $\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5$  y  $\dot{q}_6$ ) obteniendo de esta manera la Jacobiana geométrica.

Para obtener la Jacobiana analítica se parte del modelo cinemático directo [19]:

$$x=f_x(q_1, q_2, q_3, q_4, q_5, q_6)$$

$$y=f_y(q_1, q_2, q_3, q_4, q_5, q_6)$$

$$z=f_z(q_1, q_2, q_3, q_4, q_5, q_6)$$

$$\Phi = f_\Phi(q_1, q_2, q_3, q_4, q_5, q_6) \quad (3.13)$$

$$\Theta = f_\Theta(q_1, q_2, q_3, q_4, q_5, q_6)$$

$$\Psi = f_\Psi(q_1, q_2, q_3, q_4, q_5, q_6)$$

Aplicando derivadas parciales [19]:

$$\begin{aligned}\dot{x} &= \sum_{i=1}^6 \frac{\partial f_x}{\partial q_i} \dot{q}_i & \dot{y} &= \sum_{i=1}^6 \frac{\partial f_y}{\partial q_i} \dot{q}_i \\ \dot{z} &= \sum_{i=1}^6 \frac{\partial f_z}{\partial q_i} \dot{q}_i & \dot{\Phi} &= \sum_{i=1}^6 \frac{\partial f_\Phi}{\partial q_i} \dot{q}_i \\ \dot{\theta} &= \sum_{i=1}^6 \frac{\partial f_\theta}{\partial q_i} \dot{q}_i & \dot{\Psi} &= \sum_{i=1}^6 \frac{\partial f_\Psi}{\partial q_i} \dot{q}_i\end{aligned}\quad (3.14)$$

Expresado en forma matricial [19]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\Phi} \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix} = J_a \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \text{ siendo } J_a = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \dots & \frac{\partial f_x}{\partial q_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\Psi}{\partial q_1} & \dots & \frac{\partial f_\Psi}{\partial q_6} \end{bmatrix} \quad J_a \equiv \text{Jacobiana analítica} \quad (3.15)$$

Debido a su complejidad y extensión, no se va a proponer una solución de la matriz Jacobiana analítica.

### 3.5. Ventajas y desventajas

Como se ha podido comprobar, el estudio de robots con este tipo de articulaciones posee una extensa complejidad debido principalmente a que son sistemas altamente no lineales, lo que limita actualmente su uso en la industria una vez superan los 3 grados de libertad.

Sin embargo, se va a aprovechar el trabajo desarrollado en *Simulation of simple movements of Arm-Z oblique swivel joint chain manipulator* [12] para plasmar las ventajas de utilizar articulaciones oblicuas mediante una serie simulaciones. Éstas se han hecho con la concatenación de distintos segmentos como los que se muestran a continuación:

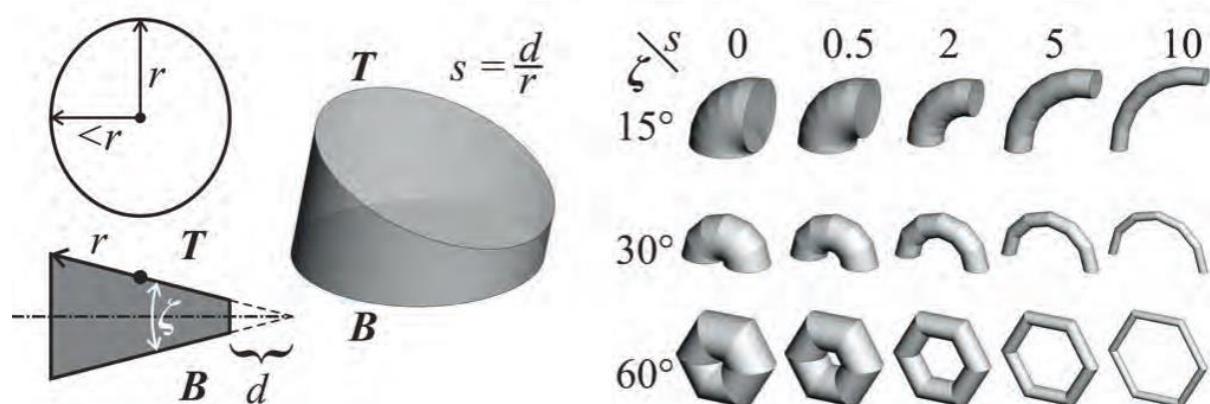


Ilustración 3.2: Izquierda: visualización del segmento definido por tres parámetros  $r$ ,  $d$  y  $\xi$ . Derecha: distintas configuraciones variando  $\xi$  y  $s=d/r$  [12].

Mediante la observación del movimiento dentro del espacio de trabajo se pueden apreciar ventajas como la simplicidad y suavidad del movimiento. Hay que destacar que el robot puede cambiar simétricamente de orientación y realizar trayectorias sin salirse del plano.

En el primer experimento se muestran 10 pasos de la simulación en la que el brazo pasa de estar completamente doblado hacia un lado, pasa por la posición vertical y se dobla completamente hacia el otro:

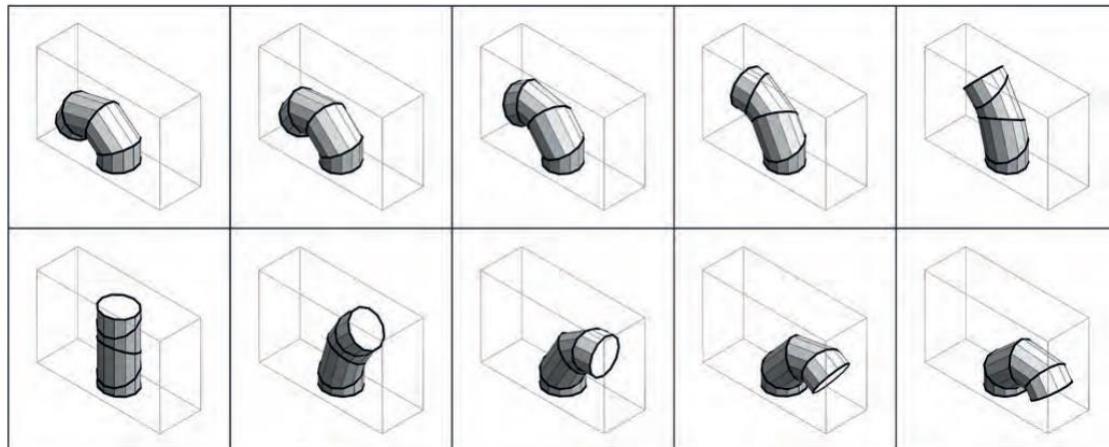


Ilustración 3.3: De arriba a la izquierda abajo a la derecha: El brazo pasa de posición toroidal a posición vertical en cinco pasos y se dobla simétricamente hacia el otro lado [12].

La siguiente imagen muestra el mismo experimento, pero con un mayor número de segmentos, 7 en lugar de 4:

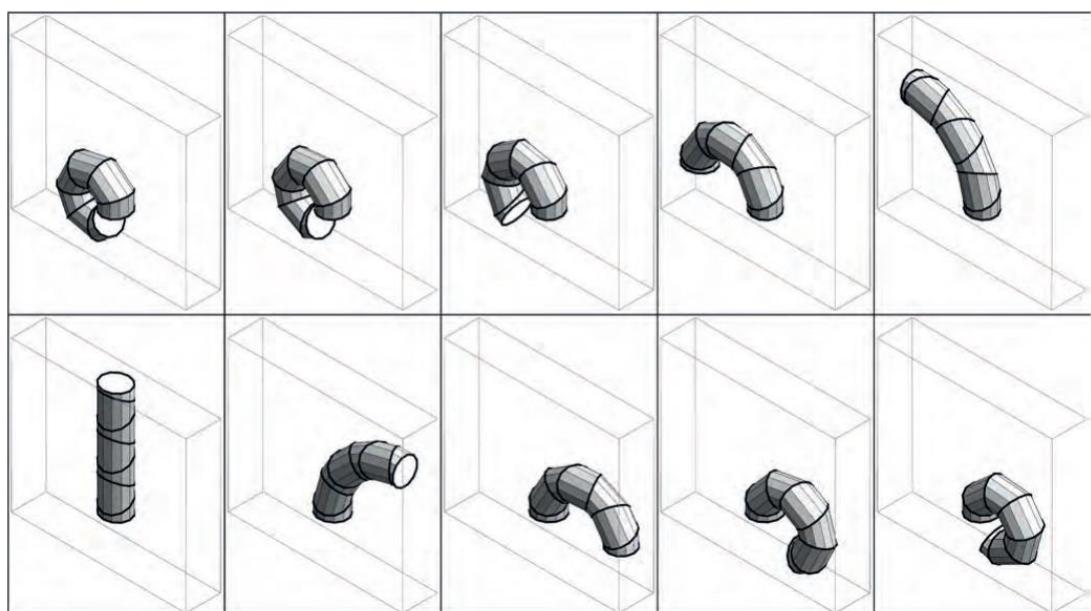
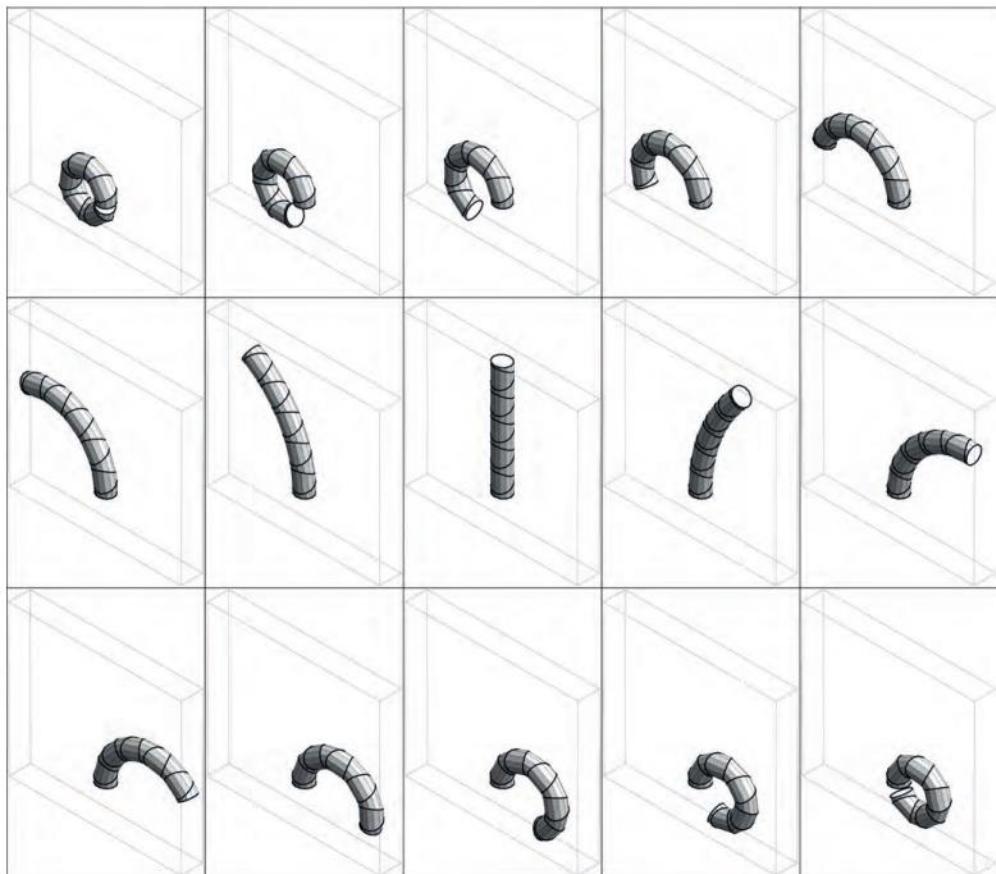


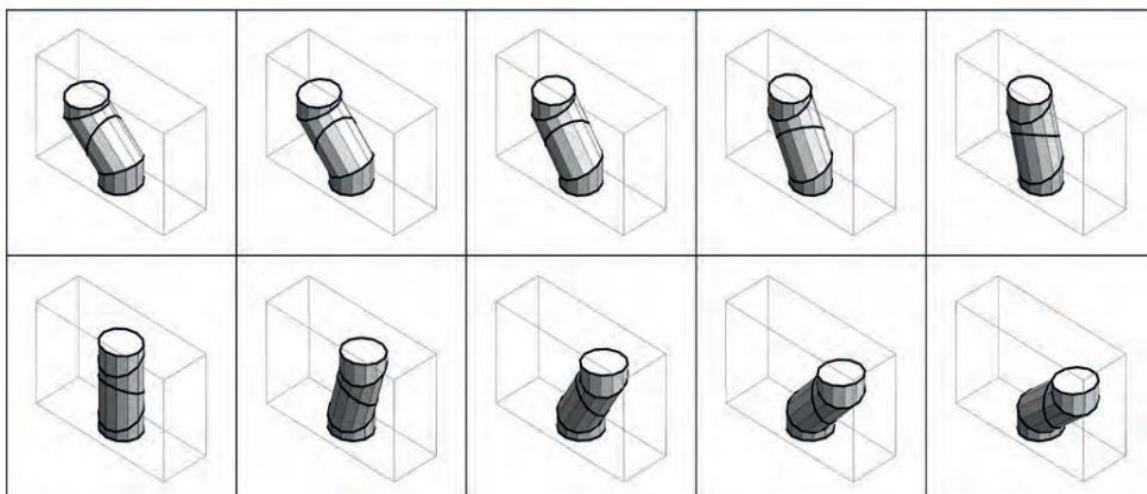
Ilustración 3.4: Mismo movimiento que en la ilustración 3.2 pero con 7 segmentos en lugar de 4 [12].

Para la siguiente simulación, el número de segmentos es 11, lo suficientemente grande como para intersecar consigo mismo. Por tanto, se ha partido de una posición ligeramente desenrollada:



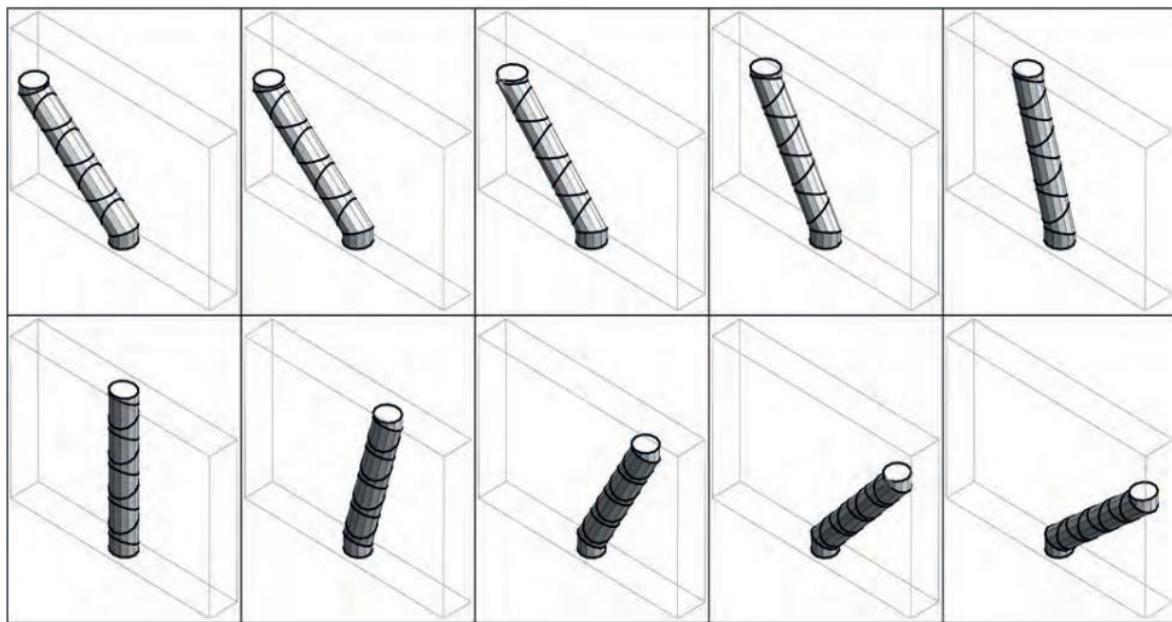
*Ilustración 3.5: Movimiento desde espiral ligeramente desenrollada a recta y de nuevo a espiral ligeramente desenrollada en el lado opuesto. El tambaleo es imperceptible como se puede apreciar con la estrechez de la caja [12].*

La siguiente simulación muestra cómo el robot puede describir una trayectoria dejando en todo momento el extremo orientado de manera horizontal y paralela a la base:



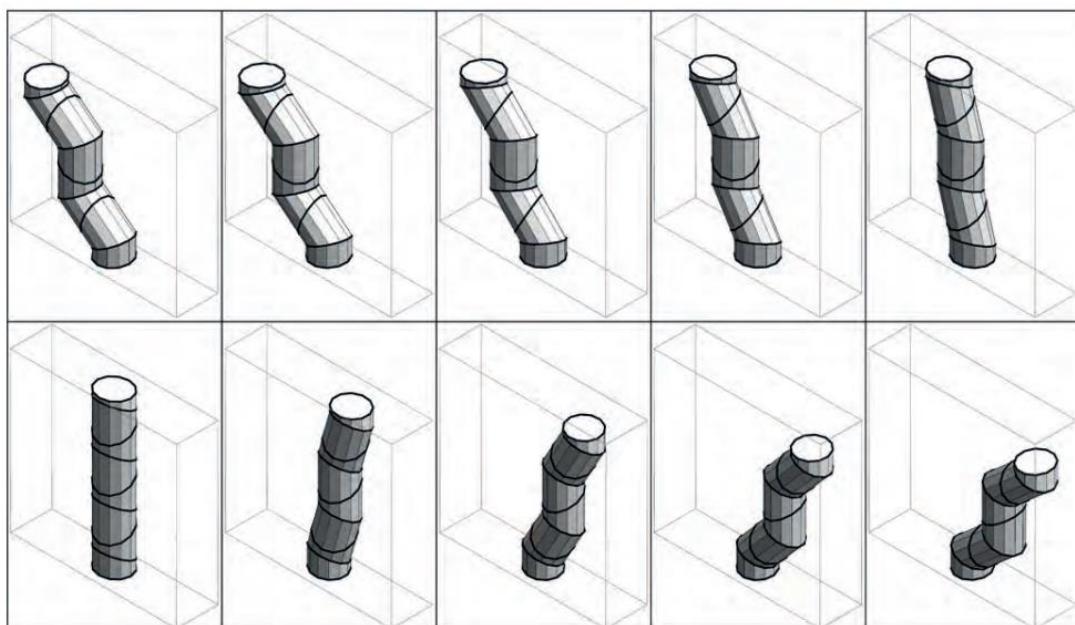
*Ilustración 3.6: De arriba a la izquierda abajo a la derecha: El brazo se estira manteniendo el extremo nivelado y se vuelve a doblar simétricamente en el otro lado. El movimiento es perfectamente suave y no hay tambaleo [12].*

El mismo experimento, pero para un número mayor de grados de libertad, 10 en lugar de 4:



*Ilustración 3.7: Mismo movimiento que en la ilustración 3.5 pero con 10 segmentos en lugar de 4 [12].*

En la siguiente figura se muestra el mismo experimento con un menor rango de movimiento:



*Ilustración 3.8: Experimento similar con menos segmentos y otro patrón de doblado distinto [12].*

Por último, la ilustración 3.8 muestra un experimento en el que el extremo describe una elipse sobre el plano vertical. El movimiento no es perfecto pues se aprecia cómo el extremo no siempre permanece paralelo a dicho plano:

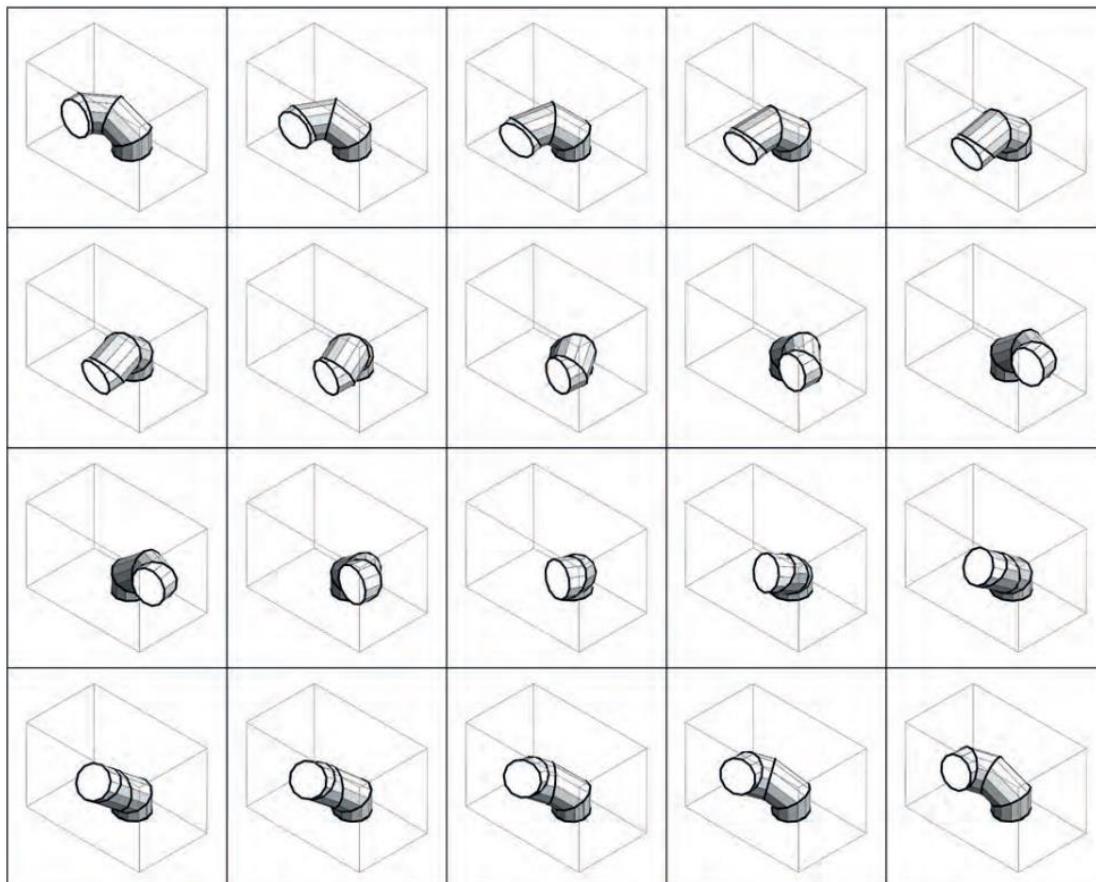


Ilustración 3.9: De arriba a la izquierda abajo a la derecha: 20 pasos en el dibujo de una elipse con el extremo del manipulador. La primera y la segunda parte del movimiento dibujan la parte superior y la inferior respectivamente [12].



## 4. Implementación

En esta segunda parte del trabajo se va a desarrollar punto por punto el proceso que se ha seguido desde las primeras etapas de concepción y diseño del robot hasta su puesta en marcha y pruebas de funcionamiento.

### 4.1 Diseño

El diseño empezó a partir de unas simples premisas en cuanto a forma y dimensiones. La idea original era hacer un brazo manipulador de 6cm de diámetro y 6 grados de libertad con articulaciones a 45 grados. Los segmentos tendrían que ser modulares para poder concatenar tantos como se quisiese. El control se haría mediante una placa de desarrollo Arduino Mega 2560 o equivalente [29] y el manejo a través de una aplicación de Matlab [20]. El proceso llevado a cabo fue el de prototipado rápido, aplicando pequeñas modificaciones e imprimiendo prototipos para analizar físicamente las ventajas y desventajas de los diseños.

#### 4.1.1 Primera etapa, planteamiento del robot:

Para empezar a plantear el diseño de las piezas, se realizó una maqueta muy simple en Tinkercad [30] (Ilustración 4.1). No se llegó a ningún diseño satisfactorio en el que las secciones fuesen concéntricas, por lo que se puede apreciar cómo, en el primer prototipo, las secciones están desplazadas. Una vez el brazo estuvo ensamblado en el programa, se comenzó una pequeña simulación moviendo cada uno de los grados de libertad para hacer un análisis preliminar del espacio de trabajo (Ilustración 4.2). Se observó que los ángulos de 45º no parecían muy versátiles y se empezó a valorar descartar estas uniones.

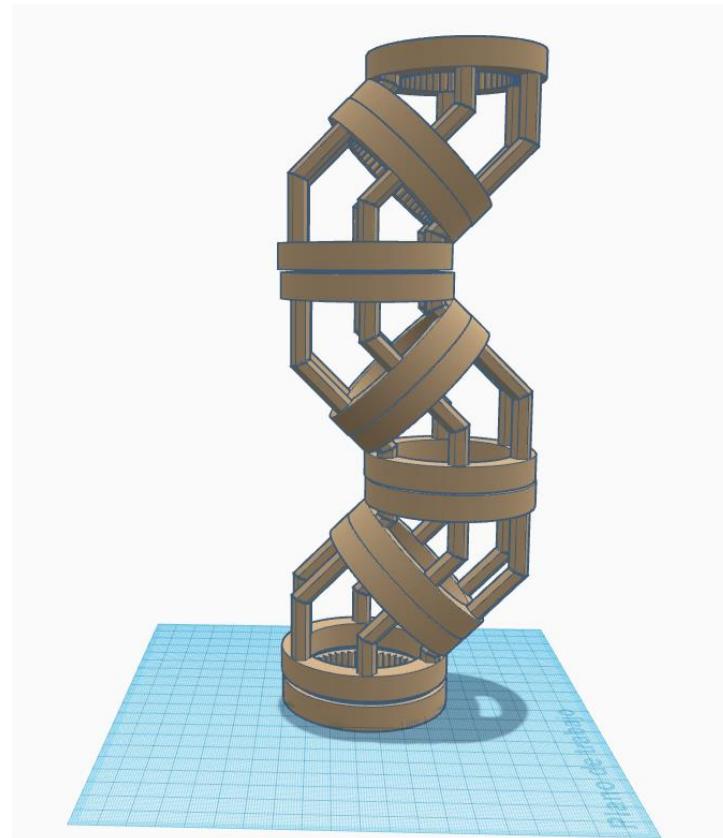


Ilustración 4.1: Primer prototipo.

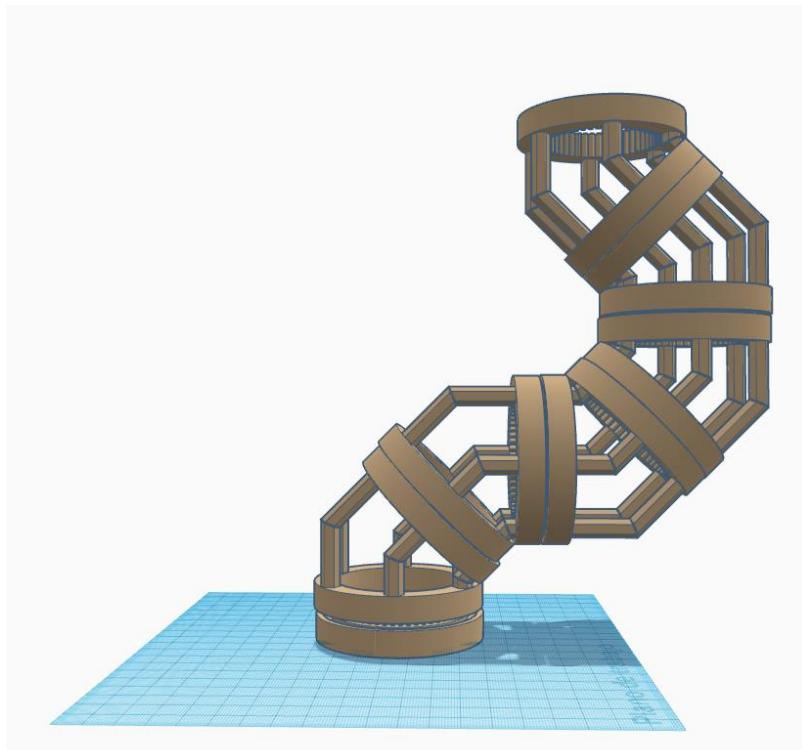


Ilustración 4.2: Las secciones no son versátiles.

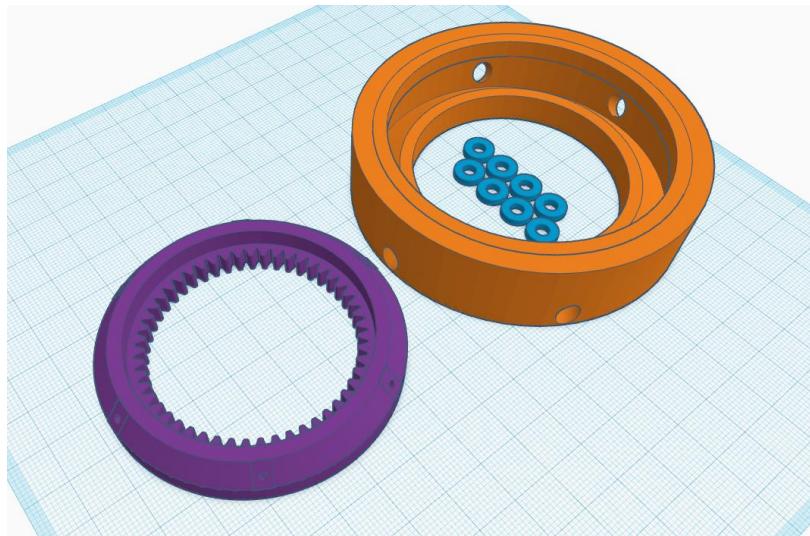
En esta primera etapa también se planteaba gestionar el cableado por el interior del robot, y los ángulos de las articulaciones impedían totalmente la gestión de cables de manera cómoda y simple. Con estos dos problemas se decidió descartar definitivamente las secciones a 45 grados.

#### 4.1.2 Segunda etapa, diseño del rodamiento:

Se tomó la decisión de usar un primer segmento a 30 grados, seguido de cuatro segmentos de 60 grados y acabando con un último de 30 grados. De tal manera que el vector director del plano de la unión cambiase entre los -30° y los 30° con respecto a la vertical estando en la posición de origen. Esta idea surgió de la patente de D. O. Nash [16]. Con esta configuración definida se procedió a tratar el siguiente problema. El diseño de un rodamiento que permitiese el giro entre dos secciones y que soportase esfuerzos en cualquier dirección sin desmontarse.

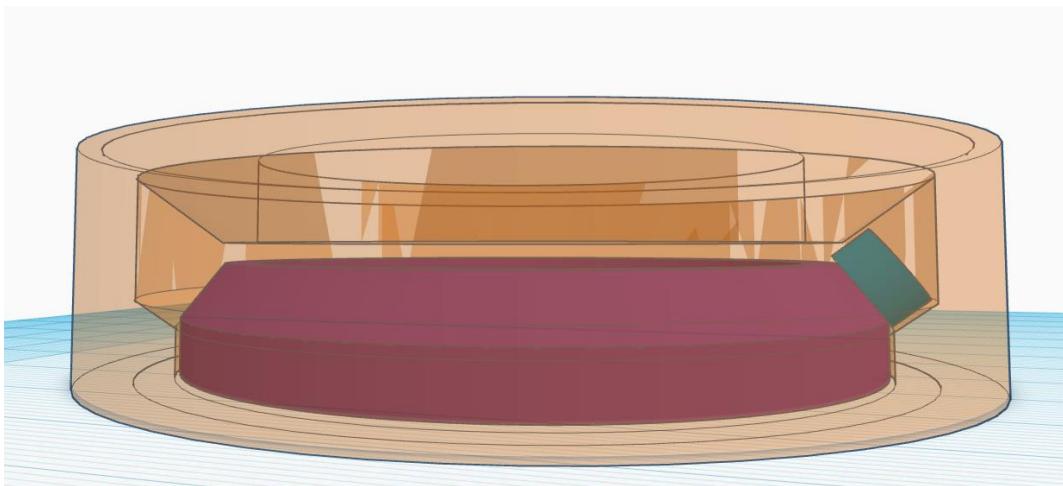
En un primer momento se barajó la opción de hacer una hendidura en las piezas donde alojar bolas de rodamiento y afianzar de manera externa la unión. No se llegó a ningún diseño satisfactorio y fiable así que se procedió a un diseño de rodamiento de rodillos oblicuo. Se diseñaron e imprimieron dos prototipos.

El primer modelo (Ilustración 4.3) constaba de seis agujeros a través de los cuales se atornillarían 6 rodillos que serían los que mantendrían en posición el rodamiento. En un principio se valoró utilizar rodillos del mismo plástico impreso, pero la prueba de rodadura no fue satisfactoria. En carga se atascaban y no giraban lo suficientemente bien.



*Ilustración 4.3: Primer prototipo de rodamiento de rodillos oblicuo.*

El segundo modelo (Ilustración 4.4) se hizo con más espacio para los rodillos y con un único agujero a través del cual se tendrían que ir introduciendo y atornillando a la pieza. Esta idea fue rápidamente descartada porque el plástico, aun con más superficie, seguía siendo inefficiente y el precio de los rodillos de bronce era demasiado elevado. Se planteó también modificarlo para poder llenar de bolas de acero el interior, pero no impedían tan bien el desmontaje del rodamiento bajo esfuerzo.



*Ilustración 4.4: Vista de la rodadura del segundo prototipo de rodamiento de rodillos oblicuo.*

Todos estos problemas llevaron a la conclusión de que montar un rodamiento que fuese eficiente y no se desmontase era complicado y consumía mucho tiempo. Por tanto, se decidió aprovechar la potencia del método de fabricación y diseñar un rodamiento que no tuviese que ser montado, pero que tampoco pudiese desmontarse.

Surgió entonces la idea de hacer una corona exterior y otra interior a modo de planeta y añadir entre ambas una serie de ruedas dentadas a modo de satélites. Los dientes de estas ruedas tendrían que ser oblicuos en dos direcciones para impedir que se desmontase. A la corona interior se le añadieron dientes rectos de módulo 1 para engranar con el piñón del servomotor, y una pequeña elevación con respecto a la corona exterior para separar levemente los segmentos y que no rozasen entre ellos (Ilustración 4.5).

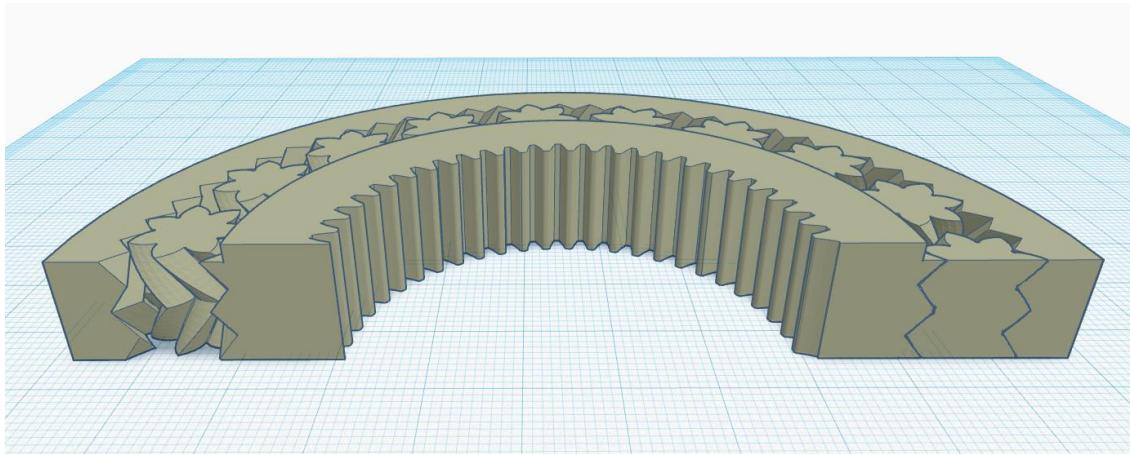


Ilustración 4.5: Sección de prototipo de rodamiento.

Se llegó a un primer diseño que fue rápidamente descartado tras la primera impresión debido a que la pared de la corona exterior era demasiado delgada y flexible y permitía desarmar el rodamiento muy fácilmente. Las siguientes versiones engrosaron la pared y añadieron agujeros para los tornillos, tanto en dirección axial en la cara superior de la corona interior como en dirección radial en la superficie de la corona exterior, llegando al diseño definitivo (Ilustración 4.6). Este diseño era considerablemente más ancho que los 6cm que se barajaron en un principio porque necesitaba albergar todas las modificaciones.

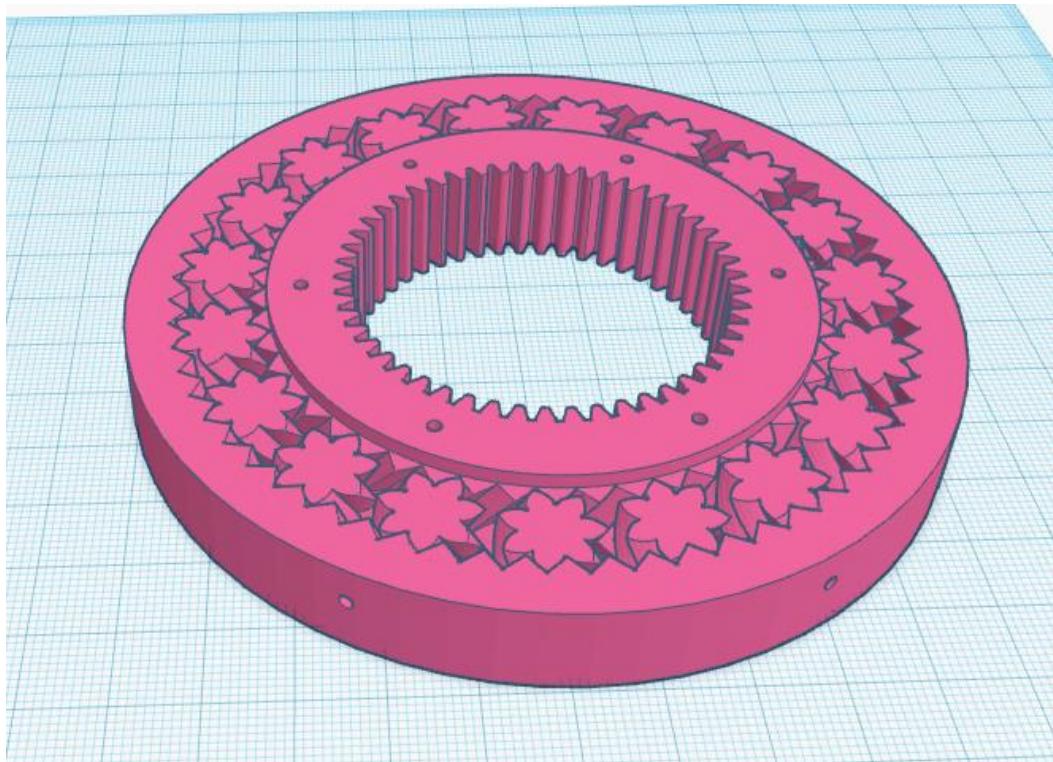


Ilustración 4.6: Rodamiento definitivo.

#### 4.1.3 Tercera etapa, planteamiento de los segmentos:

Con el rodamiento planteado, se procedió al diseño de los segmentos, que igualmente tendrían que ser más anchos. El diseño se hizo lo más simple posible, pero sin olvidar las limitaciones del método de fabricación y teniendo en mente en todo momento que debían ser iguales y modulares. Para evitar la necesidad de imprimir con soporte, los voladizos no podían ser muy pronunciados y en la laminación no podía aparecer una capa que se imprimiese sobre el vacío. Teniendo estas restricciones en cuenta, se diseñó un alojamiento para fijar a la corona exterior (Ilustración 4.7), una cara plana para atornillarla a la corona interior (Ilustración 4.8) y un cilindro hueco conectando ambas. Las piezas obtenidas fueron los que se muestran en las ilustraciones 4.9 y 4.10. La ilustración 4.11 muestra la unión del primer prototipo de segmento al rodamiento. Los pesos de estas piezas, proporcionados por el software Ultimaker Cura [31] son los que se utilizarían para una primera aproximación en el dimensionamiento de los servomotores.

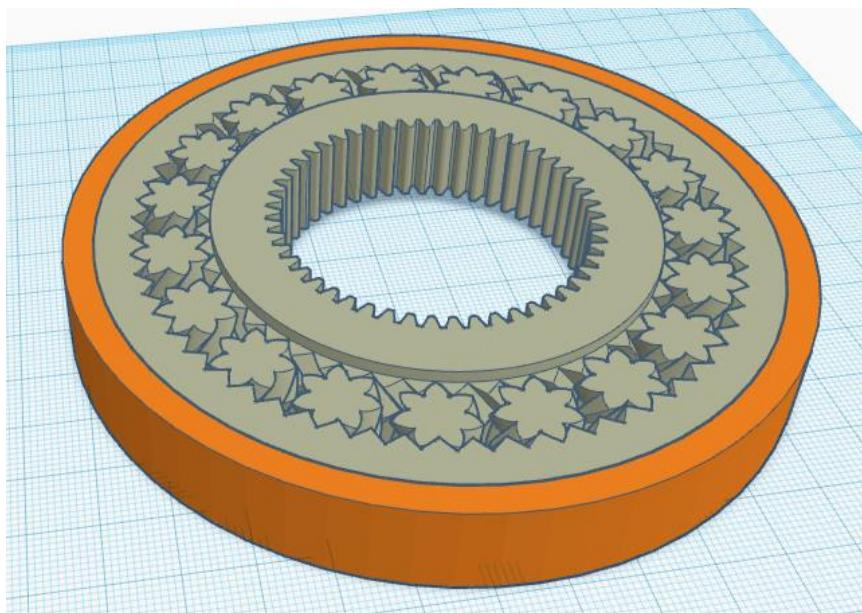


Ilustración 4.7: Unión de la corona exterior del rodamiento.

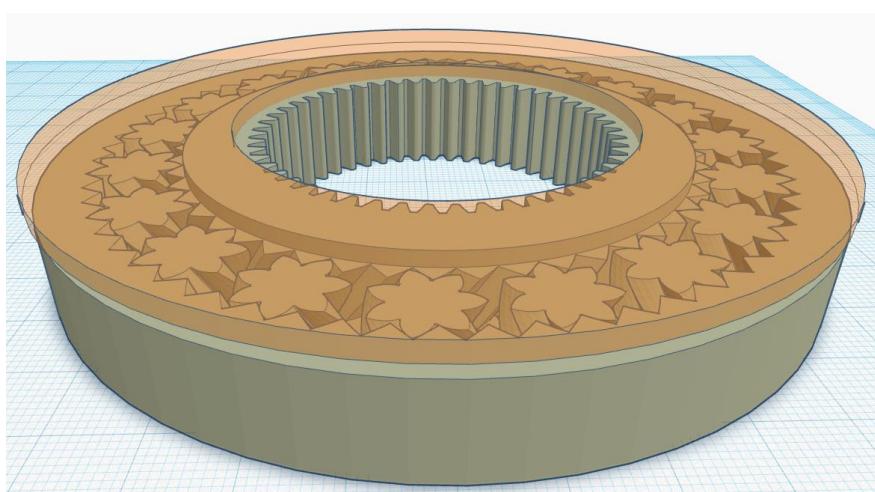


Ilustración 4.8: Unión de la corona interior del rodamiento.

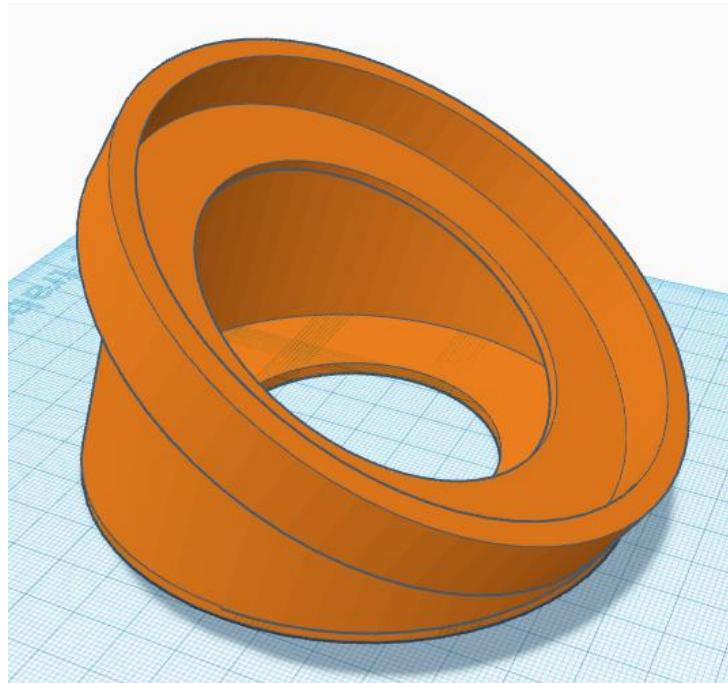


Ilustración 4.9: Prototipo de segmento de 30 grados.

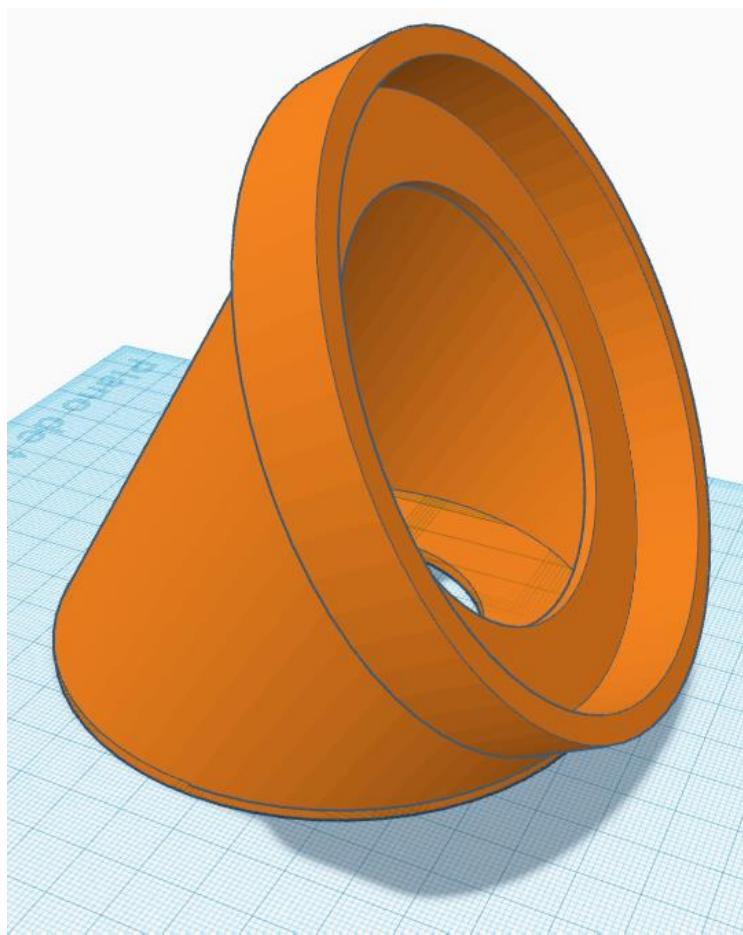


Ilustración 4.10: Prototipo de segmento de 60 grados.

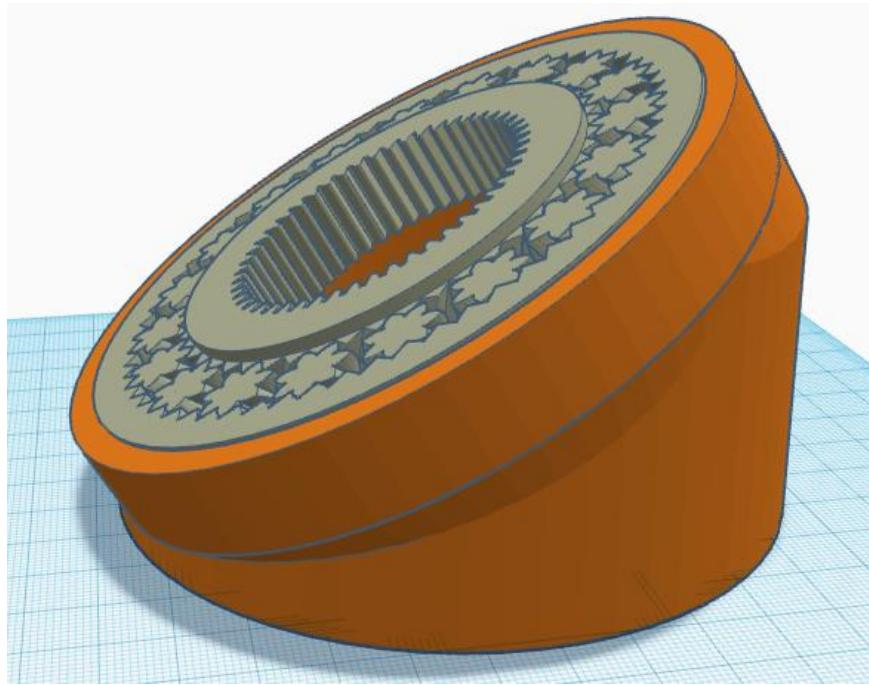


Ilustración 4.11: Unión entre rodamiento y primer prototipo de segmento de 30 grados.

#### 4.1.4 Cuarta etapa, diseño de los segmentos:

Tras una primera iteración en el dimensionamiento de los servos, se procedió a diseñar el alojamiento y la fijación de los servomotores. Se hizo un agujero rectangular en la pared del cilindro para poder introducir el servo. Éste se asienta sobre una estructura que es distinta para el segmento de 30 grados y para el segmento de 60 por motivos de simplicidad de impresión.

En este punto se decidió que la canalización de cables se iba a realizar por el exterior para facilitar la gestión y la prueba individual de cada servomotor de cara a la utilización a nivel docente del robot.

También se añadieron unas pequeñas aletas en los laterales del agujero que sirven de soporte a la hora de imprimir y añaden resistencia extra a la estructura. Al segmento de 60 grados se le añadieron unos agujeros para facilitar el atornillado y tanto al segmento de 60 como al de 30 se añadieron unos orificios a las estructuras de soporte de los servomotores por los que pasar los cables. Se creó, también, un primer diseño de piñones con dientes de módulo 1 para engranar con el rodamiento, con estrías para mejor fijación y un agujero pasante para poder atornillarlo al servomotor. (Ilustración 4.17)

Para verificar la validez del diseño se acudió a la comunidad de Tinkercad [30] y se utilizó el modelo *Continuous Servo* de Randy Sarafan [32]. Se hizo un montaje para analizar la alineación de los agujeros y la posibilidad de introducción del servo en el alojamiento (Ilustraciones 4.15, 4.16 y 4.18).

Se imprimieron unas primeras versiones a las que hubo que realizar pequeñas modificaciones dimensionales para que el servomotor entrase con un ‘clic’ pero sin tener que forzar en exceso la pieza. El resultado fue el siguiente:

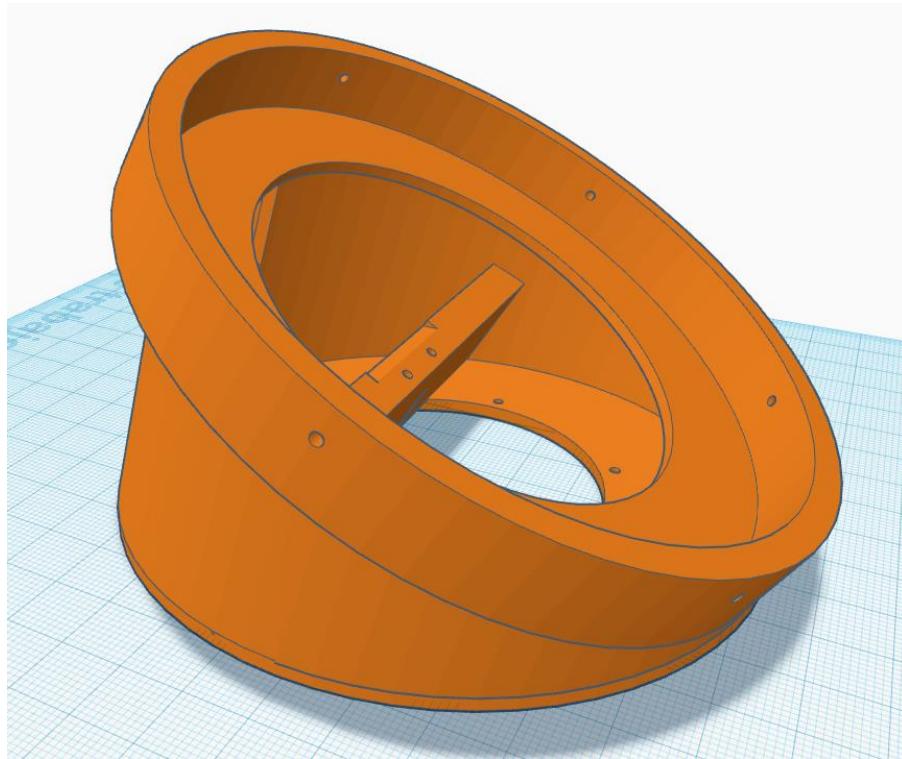


Ilustración 4.12: Segmento de 30 grados definitivo.

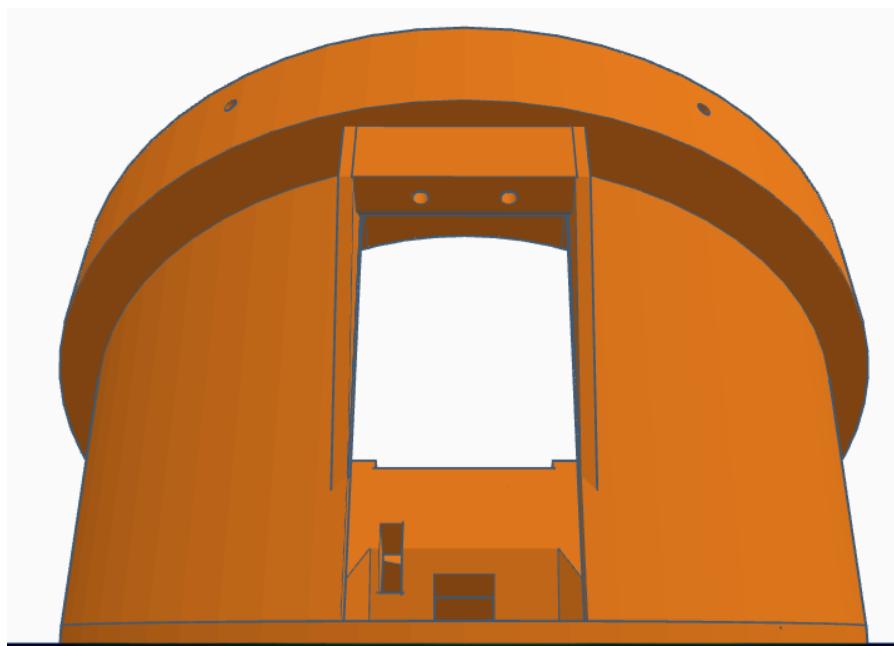


Ilustración 4.13: Detalle de las aletas, el rebaje y los orificios en la estructura del alojamiento del servomotor en el segmento de 30 grados definitivo.

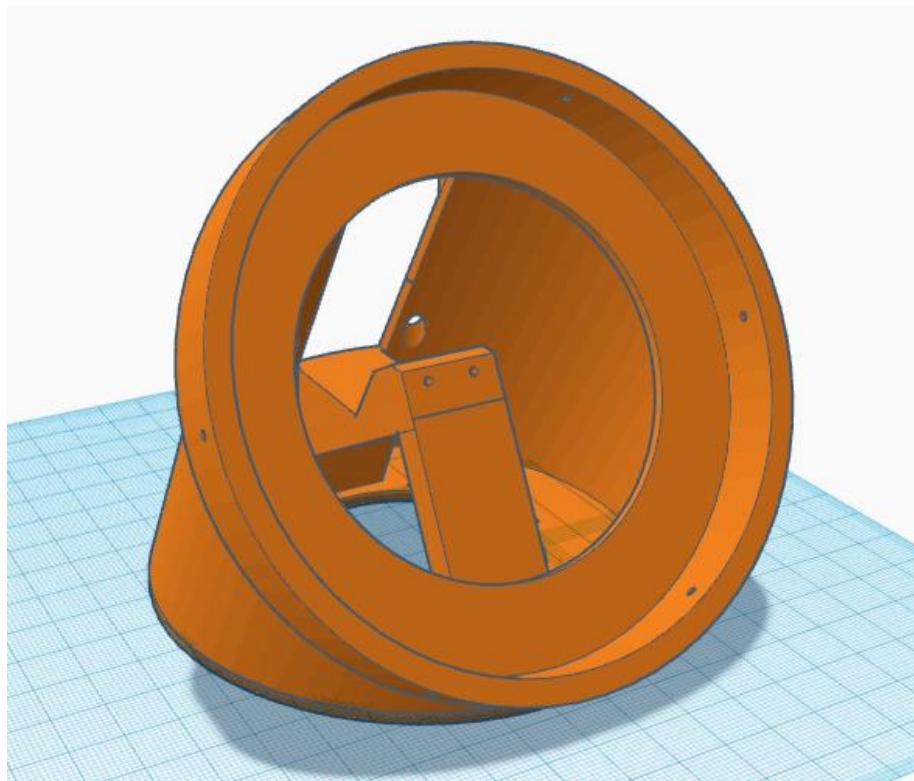


Ilustración 4.14: Segmento de 60 grados definitivo.

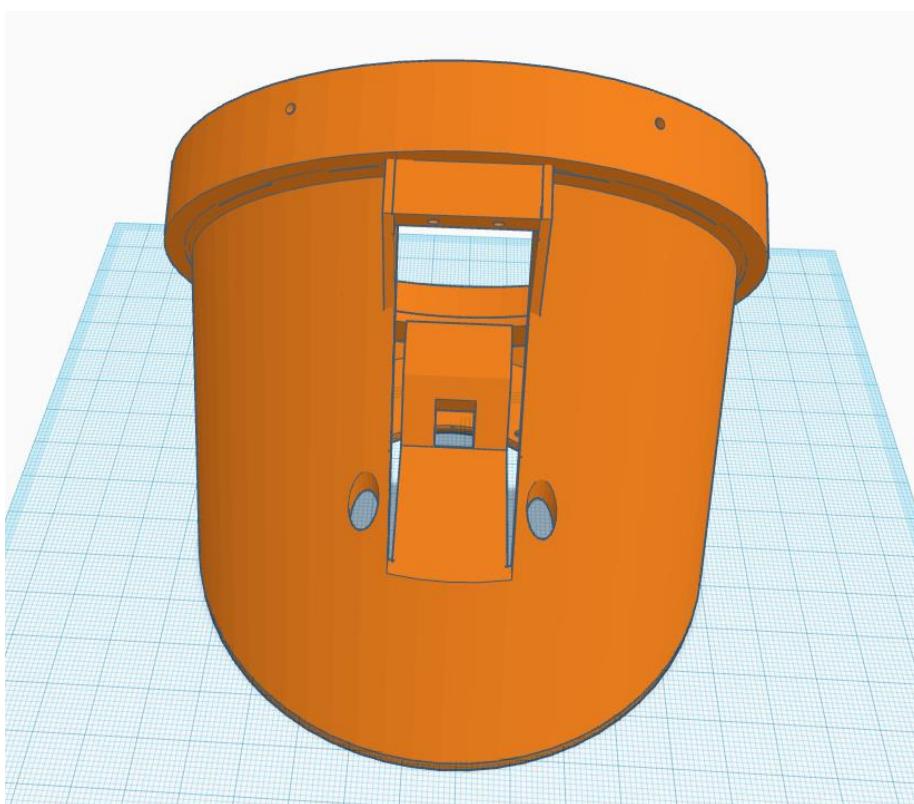


Ilustración 4.15: Detalle de las aletas, el rebaje y los orificios en la estructura del alojamiento del servomotor en el segmento de 60 grados definitivo.

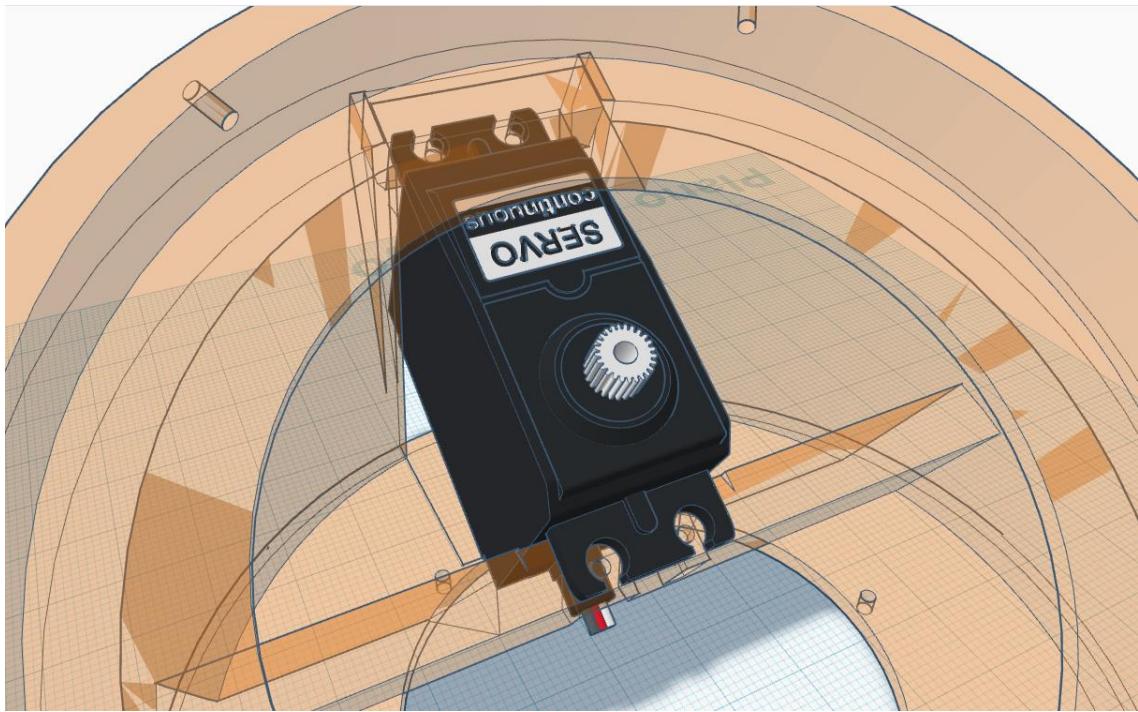


Ilustración 4.16: Detalle de la unión del servomotor con el segmento definitivo de 30 grados.

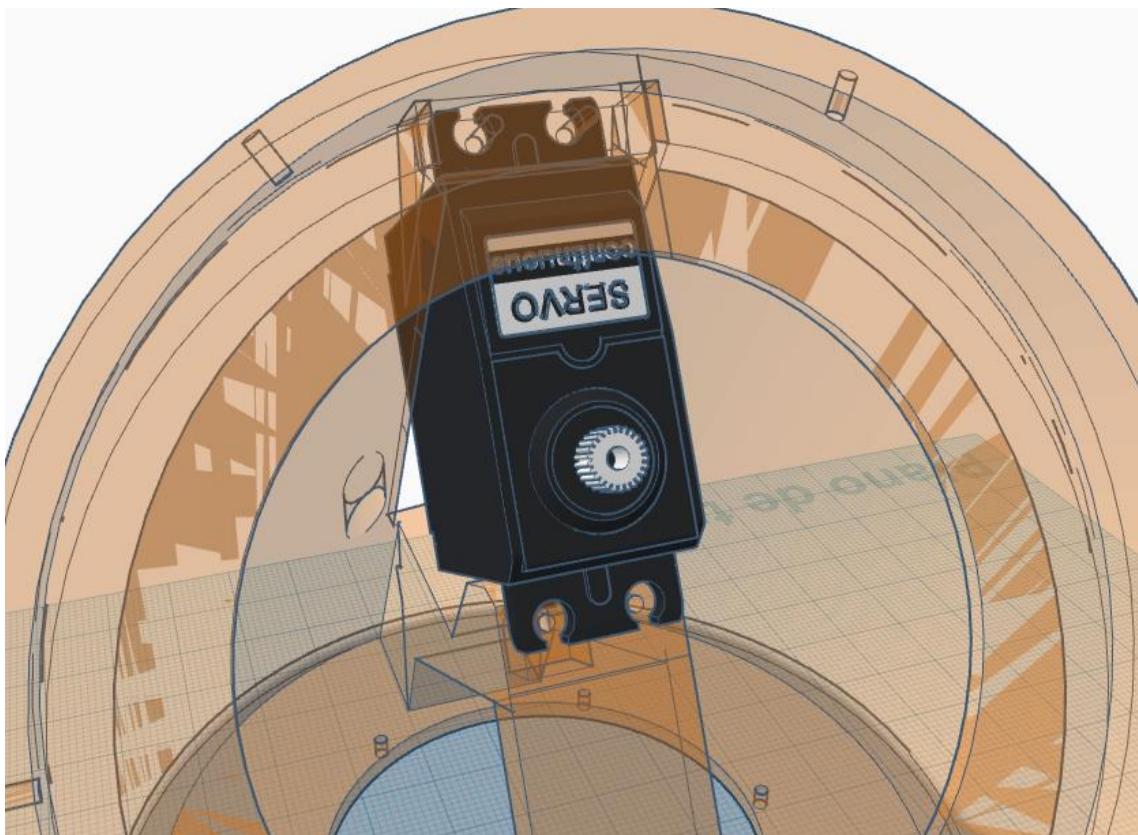


Ilustración 4.17: Detalle de la unión del servomotor con el segmento definitivo de 60 grados.

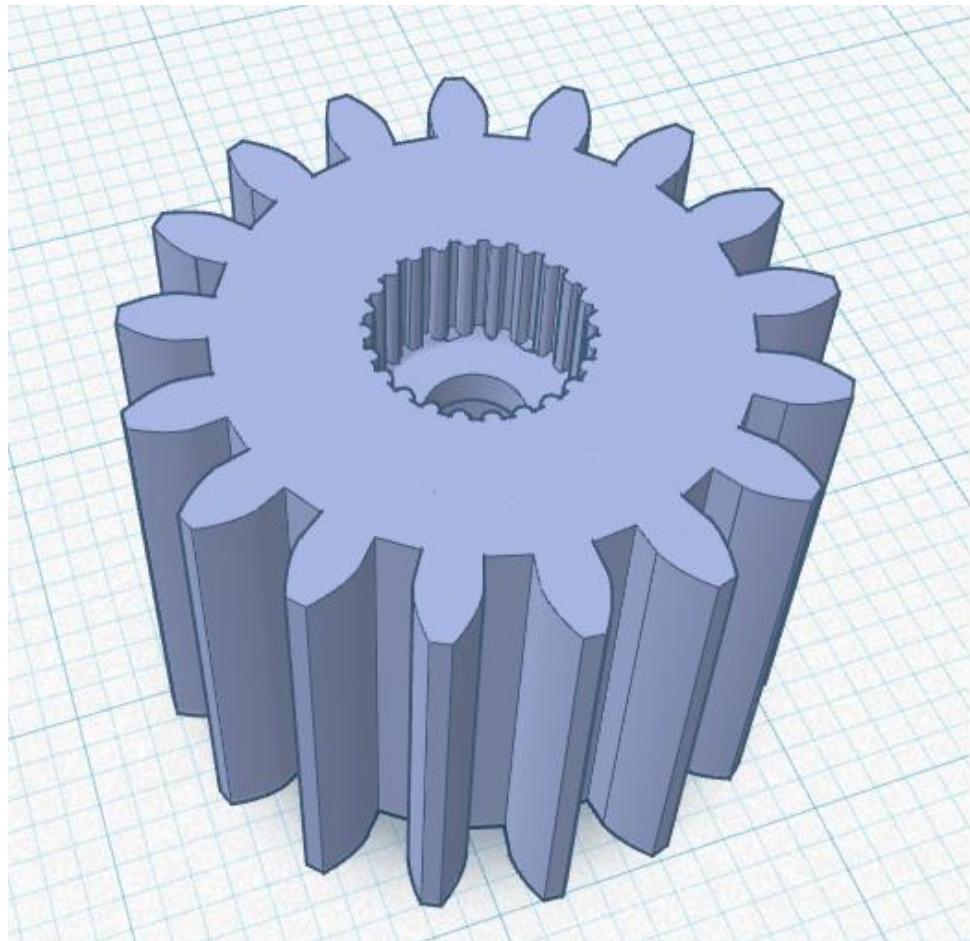


Ilustración 4.18: Detalle del estriado del piñón.

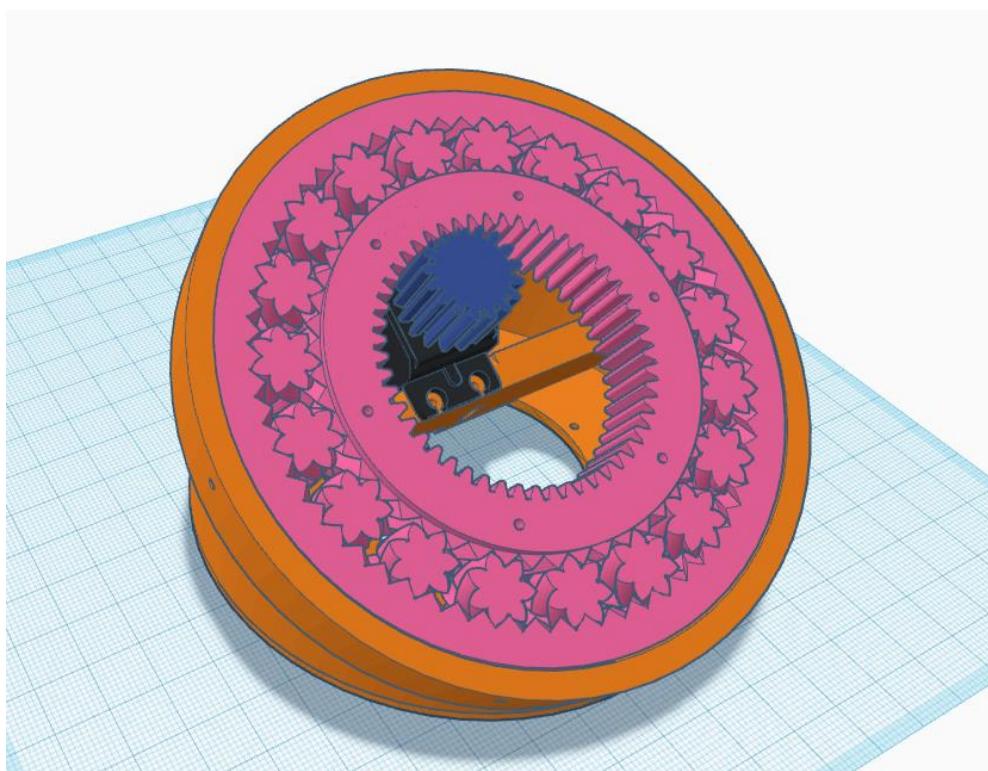


Ilustración 4.19: Detalle de montaje con segmento de 30 grados, servomotor, rodamiento y piñón.

#### 4.1.5 Quinta etapa, diseño del control de posición:

Llegados a este punto, se volvió a iterar el dimensionamiento de los servomotores para verificar que, a pesar del aumento de peso al añadir los soportes, seguían estando dentro del rango incluso en las condiciones más desfavorables. Una vez verificado, se llevó a cabo la primera prueba de funcionamiento. Se ensambló un primer grado de libertad oblicuo y se escribió un breve programa para que el servomotor girase 90 grados a cada lado de manera continua.

Surgieron varios fallos. En primer lugar, el tipo de control que tiene un servomotor de funcionamiento continuo es de velocidad en bucle abierto. Mientras que los servomotores de 180 grados tienen control de posición en bucle cerrado a costa de girar solo media vuelta. El robot necesita lo mejor de los dos mundos.

Por un lado, un giro continuo permite aprovechar el par de la reducción del piñón-rodamiento, que es de 1:3. También evita sufrir una disminución del ángulo girado por la articulación, que sería de 60 grados si se usase un servomotor de 180 grados. Por otro lado, se necesita también un control de posición en bucle cerrado, no solo para garantizar el error de posición nulo, sino para mantener energizados los motores una vez el robot ha llegado a la posición de mando.

A pesar de eso, se decidió utilizar el control en bucle abierto de velocidad mediante un control del tiempo de encendido. Se consideró asumible el error de posición y una esperable deriva con el paso del tiempo de funcionamiento. También se asumió que la posición de 0 se le daría a mano.

Con esto en mente se montó una primera articulación y se hizo una tanda de pruebas. La primera consistió en cronometrar en tres tandas el tiempo que tardaba el servo en girar diez vueltas completas la articulación, al máximo de potencia, en cada sentido y con la carga de un único segmento y en vacío. Llegando a un total de 12 mediciones. (tabla 4.1). Se observó una ligera diferencia en los tiempos de giro horario y antihorario.

Tiempo (s) para 10 vueltas, giro antihorario sin carga.	Tiempo (s) para 10 vueltas, giro horario sin carga.	Tiempo (s) para 10 vueltas, giro antihorario con carga.	Tiempo (s) para 10 vueltas, giro horario con carga.
30.23	30.78	30.27	31.02
30.42	30.83	29.84	30.97
30.55	30.90	29.95	30.88

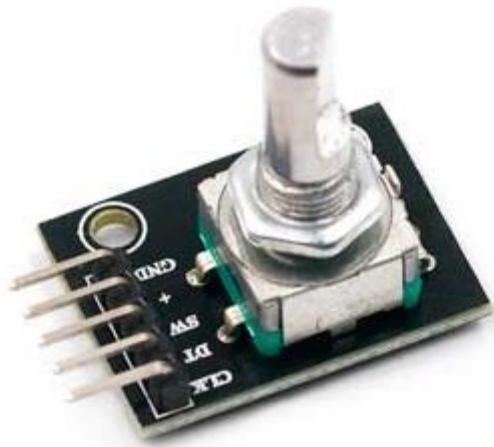
Tabla 4.1: Tiempos medidos en ensayo de giro de 10 vueltas en ambos sentidos con y sin carga.

Con esos resultados se programó un pequeño test que consistía en girar 90 grados horarios y antihorarios alternativamente para ver cómo funcionaba el control. Lo primero que se observó es que los tiempos calculados no eran fieles a la hora de hacer un giro preciso, puesto que una vez se dejaba de suministrar la señal PWM a los servomotores, éstos seguían teniendo cierta inercia. Se intentó corregir alimentando en sentido contrario durante unos milisegundos para forzar el frenado. Surgió un nuevo problema, y es que el ancho de la señal PWM en sentido horario está entre 0,5ms y 1ms mientras que el ancho de la señal para el sentido antihorario está entre 1,5ms y los 2ms. El período en ambos casos es de 20ms. Esto hace que el tiempo de frenado sea distinto dependiendo del sentido, puesto que solo llega un pulso que puede ser entre 2 y 3 veces más corto en un sentido que en otro.

A pesar de eso, con unos pequeños ajustes en los tiempos, se consiguió llegar a la posición de consigna con un levísimo error. Durante varios ciclos el montaje estuvo funcionando como estaba previsto, pero después de un cierto tiempo de test se produjo un fallo súbito y el servomotor empezó a pararse antes de lo que estaba consignado. Tras varias pruebas con el

mismo error se llegó a la conclusión de que iba a ser imposible hacer un control de posición suficientemente satisfactorio y se tomó la decisión de implementar encoders.

Se barajó la posibilidad de utilizar encoders rotativos [33] de 20 pulsos por vuelta. Unidos a la corona interior y suponiendo una relación de transformación idéntica a la del servo, darían 60 pulsos por giro completo de la articulación. Esto supondría un error de posición de 6 grados. El error se considera asumible, además se puede engranar con una relación aún más pequeña para aumentar la precisión. En cualquier caso, para controlar de manera adecuada los 6 encoders, se iban a necesitar 12 interrupciones, una por cada canal de cada encoder. La placa de Arduino Mega 2560 [29] tan solo tiene 6 interrupciones.



*Ilustración 4.20: Encoder rotativo.*

En este punto se barajó la opción de utilizar bien una placa Arduino Due [34] ya que tiene una interrupción en cada pin o bien utilizar otro tipo de placa de desarrollo que tuviese un microcontrolador STM32 [35] puesto que implementan un control de interrupciones exclusivamente hardware. El principal inconveniente de Arduino Due [33] es el precio, y el de la placa STM32 [35] es la conectividad con Matlab [20] de cara a hacer la GUI (Graphical User Interface) de control.

Tras esto se decidió utilizar encoders de efecto Hall. En concreto los AS5600 [36] ya que no necesitan contacto. Con este tipo de encoders se puede colocar por separado el sensor y el imán. Además, tienen una resolución de 12 bits, por lo que proporcionan 4096 pulsos por vuelta y son muy económicos.

El siguiente paso fue observar en qué punto se podrían colocar los imanes. Por un lado, si se colocan en el eje del servomotor se puede llegar a tener una precisión elevadísima con 12.288 pulsos por vuelta de la articulación. Eso supone un error de menos de 1,76 minutos de arco, pero la grandísima cantidad de pulsos puede dar problemas en el microcontrolador, sobre todo teniendo en cuenta que se necesitan 6 encoders. Además, tampoco existe una solución simple para anclar las piezas en esa posición. Por tanto, se decidió que la mejor opción sería colocar el imán en el centro de la unión entre segmentos. El error teórico de esta configuración es menor a 5,28 minutos de grado, que aun siendo mayor que en el otro caso, es considerablemente menor que con los encoders rotativos [33].

Los encoders AS5600 [36] tienen tres modos de control. El primero de ellos es mediante I2C. Esto supone dos problemas, el primero es que el protocolo I2C no está pensado para la comunicación a larga distancia. No obstante, la longitud del cable no va a exceder en ningún caso el metro, por lo que no debería haber problema colocando una resistencia pull-up

adecuada. El otro inconveniente de este modo es que solo tienen una única dirección I2C (0x36) y no es programable. Con este tipo de control se necesitaría un multiplexor I2C, como puede ser el TCA9548A [37], para ir seleccionando qué encoder controlar. Es un multiplexor con 8 puertos I2C y se necesitan solo 6 para controlar los encoders.

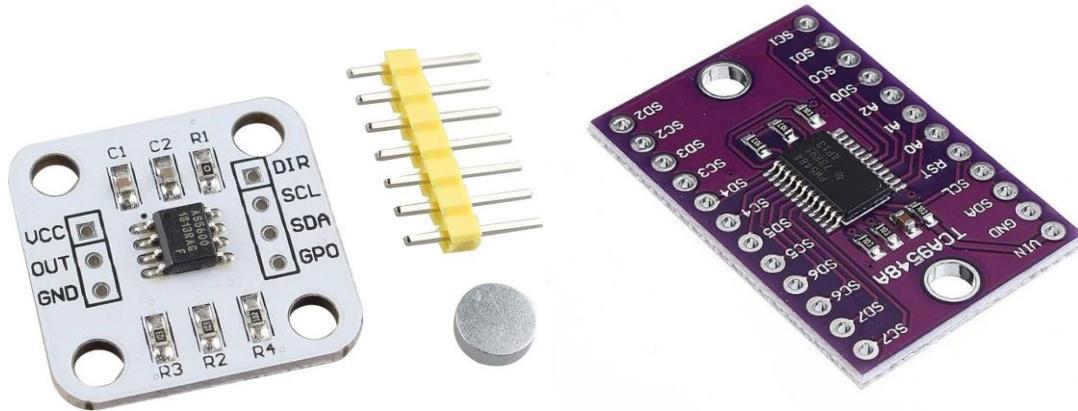


Ilustración 4.21: Izquierda: Encoder AS5600 [36]. Derecha: Multiplexor I2C TCA9548A [37].

El segundo modo de control es mediante PWM. Es un método más complejo a la hora de programar y que además puede suponer un problema a la hora de capturar todos los pulsos PWM de cada uno de los encoders.

Por último, existe un tercer modo de control analógico mediante nivel de tensión. No obstante, tiene el inconveniente de que la resolución de este modo es de 12 bits, en cambio el conversor Analógico-Digital (ADC) de Arduino Mega 2560 [29] es tan solo de 10 bits, por lo que los 4096 pulsos por vuelta se convertirían en 1024.

Una última característica es que se les puede programar una vez el rango de funcionamiento. Pero no son reprogramables y cualquier fallo se vuelve irreversible.

Con todo esto en mente, se tomó la decisión de utilizar los encoders de efecto Hall AS5600 [35] frente a los rotativos [33] por la reducción en el error de posición. Se eligió también el modo de control mediante el protocolo I2C para no perder resolución y por simplicidad a la hora de la programación, a pesar de necesitar el multiplexor TCA9548A [37] para poder controlar los 6 encoders, que comparten dirección.

#### 4.1.6 Sexta etapa, diseño de los soportes del encoder:

Con los encoders elegidos, y puesto que una parte de los segmentos ya estaban impresos, se procedió a diseñar una pieza que se atornillase junto a la base de los segmentos para posicionar el imán en el centro de la unión. También se diseñó una nueva pieza que se atornillase junto a los servomotores y sostuviese la placa con el sensor en su posición. Además, fue necesario ajustar las dimensiones de la rueda dentada por motivos de espacio. De esta manera no sería necesario reimprimir todas las piezas de nuevo y el montaje, aun con más piezas, sería más sencillo y ajustable.

La pieza que sostiene el imán es un pequeño listón con agujeros en los extremos para atornillar en un diámetro de la base del segmento. En el centro tiene un alojamiento cilíndrico para encajar por presión el imán. Además, consta de un agujero en la base para poder extraerlo introduciendo una lezna o, en este caso, la punta de un cable DuPont.

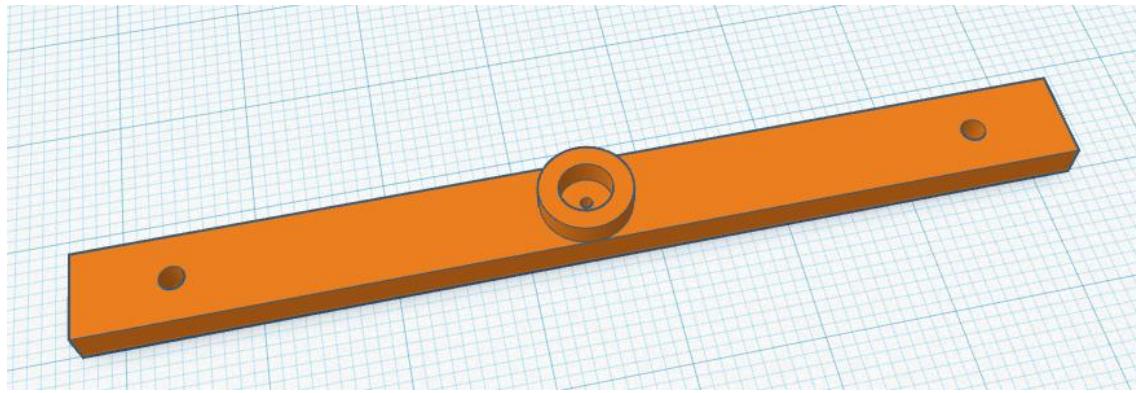


Ilustración 4.22: Pieza que sujeta el imán en su posición.

Para la pieza que sujeta la placa del encoder se partió de un perfil recto en U. En un lateral se añadieron agujeros para atornillar junto al servomotor y una hendidura para alojar el nervio que posee el servo. En el otro lado fue necesario hacer un agujero rectangular en el centro para permitir el atornillado. Posteriormente fue reforzado con un nervio central que además facilita la impresión. También tiene un vaciado para no chocar con la rueda dentada. Por último, consta de 4 agujeros en las esquinas para fijar la placa.

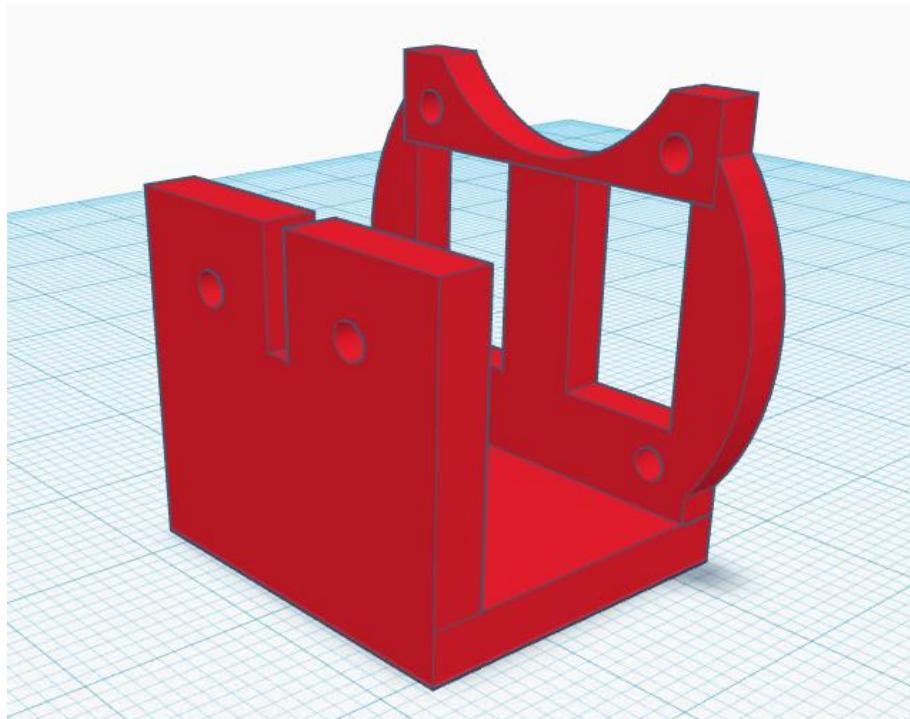


Ilustración 4.23: Pieza que sujeta la placa del encoder.

A continuación, se volvió a verificar el diseño haciendo un montaje en Tinkercad [30] para el cual fue necesario hacer un pequeño modelo de la placa del encoder.

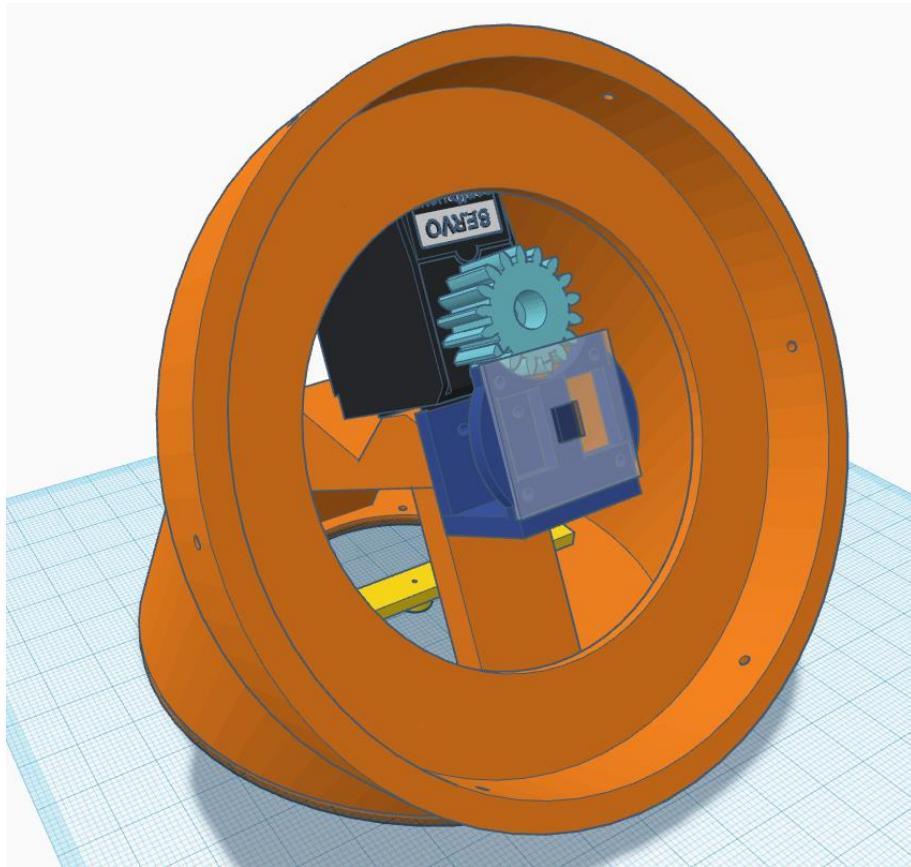


Ilustración 4.24: Detalle de montaje del encoder en un segmento de 60 grados.

Nótese la diferencia de altura del piñón definitivo frente al prototipo anterior para hacer hueco a la placa:

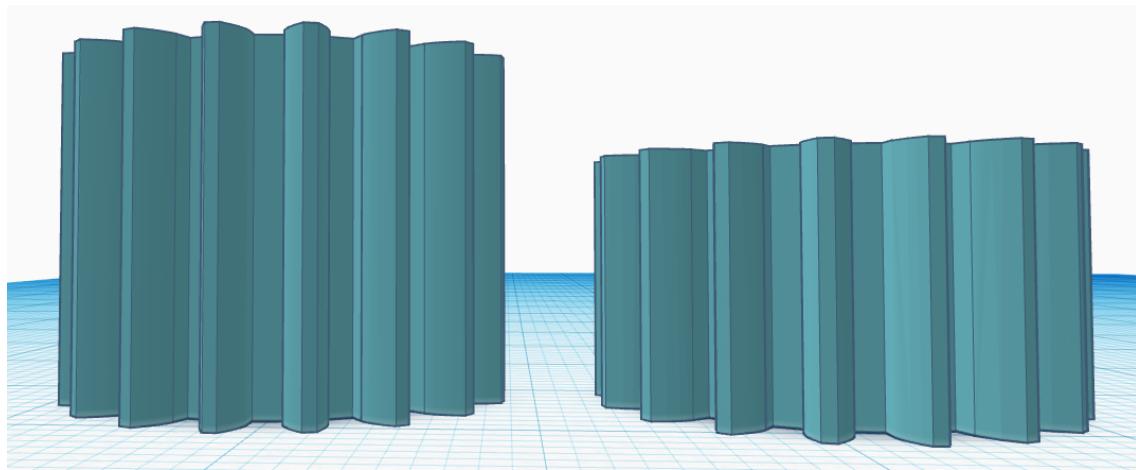


Ilustración 4.25: Altura del piñón definitivo (derecha) frente al prototipo anterior (izquierda).

#### 4.1.7 Séptima etapa, programación de un grado de libertad:

El siguiente paso fue realizar nuevas pruebas para verificar que todo funcionaba. Se montó una unión con encoder y se utilizaron varios códigos.

El primer código [38] hace un barrido de direcciones I2C para encontrar los dispositivos conectados. Desde la dirección 0x09 hasta la 0x77.

El multiplexor TCA9548A [37] tiene 3 bits para programar la dirección I2C, entre 0x70 y 0x77. Poniendo los 3 bits de configuración a GND, el programa tendría que detectar la dirección 0x70 o 112 en decimal.

```
Escaneando bus I2C...addr: 8
addr: 9      addr: 10      addr: 11      addr: 12
addr: 13     addr: 14      addr: 15      addr: 16
addr: 17     addr: 18      addr: 19      addr: 20
addr: 21     addr: 22      addr: 23      addr: 24
addr: 25     addr: 26      addr: 27      addr: 28
addr: 29     addr: 30      addr: 31      addr: 32
addr: 33     addr: 34      addr: 35      addr: 36
addr: 37     addr: 38      addr: 39      addr: 40
addr: 41     addr: 42      addr: 43      addr: 44
addr: 45     addr: 46      addr: 47      addr: 48
addr: 49     addr: 50      addr: 51      addr: 52
addr: 53     addr: 54      addr: 55      addr: 56
addr: 57     addr: 58      addr: 59      addr: 60
addr: 61     addr: 62      addr: 63      addr: 64
addr: 65     addr: 66      addr: 67      addr: 68
addr: 69     addr: 70      addr: 71      addr: 72
addr: 73     addr: 74      addr: 75      addr: 76
addr: 77     addr: 78      addr: 79      addr: 80
addr: 81     addr: 82      addr: 83      addr: 84
addr: 85     addr: 86      addr: 87      addr: 88
addr: 89     addr: 90      addr: 91      addr: 92
addr: 93     addr: 94      addr: 95      addr: 96
addr: 97     addr: 98      addr: 99      addr: 100
addr: 101    addr: 102     addr: 103     addr: 104
addr: 105    addr: 106     addr: 107     addr: 108
addr: 109    addr: 110     addr: 111     addr: 112 Encontrado!
addr: 113    addr: 114     addr: 115     addr: 116
addr: 117    addr: 118     addr: 119
Terminado
```

Ilustración 4.26: Barrido de direcciones I2C.

El segundo código [38] hace un barrido dentro de la dirección del multiplexor I2C TCA9548A [37] para detectar los dispositivos conectados en sus 8 terminales (numerados del 0 al 7), además muestra su dirección I2C nativa. En esta prueba sólo había conectado un único encoder en el terminal 7. Obsérvese cómo la dirección fija del encoder es, en efecto 0x36.

```
TCA escaner listo
  Escaneando salida 0
  Escaneando salida 1
  Escaneando salida 2
  Escaneando salida 3
  Escaneando salida 4
  Escaneando salida 5
  Escaneando salida 6
  Escaneando salida 7
  - Encontrado I2C 0x36
Finalizado
```

Ilustración 4.27: Barrido de salidas del multiplexor TCA9548A [37].

Se adaptó un programa [39] para verificar que la distancia del imán era correcta, éste saca un código binario por el puerto serie y se detiene cuando se llega a una posición adecuada.

0b1100111 si detecta el imán muy fuerte; está demasiado cerca.

0b1010111 si detecta el imán muy débil; está demasiado lejos.

0b1110111 si el campo detectado es adecuado.

En este caso, el imán estaba demasiado cerca y hubo que hacer una corrección dimensional en la altura del soporte del encoder.

```
Magnet status: 1010111  
Magnet status: 1110111  
Magnet status: 1100111
```

Ilustración 4.28: Posibles salidas en la verificación de la lectura del imán.

Una vez se hizo la comprobación de que todos los elementos funcionaban adecuadamente y con las piezas definitivas impresas se hizo un test para el control de posición en bucle cerrado. Para ello se utilizó la librería PID\_v1 [40] y se escribió un código en el que se le daban órdenes de mando al servomotor por consulta periódica. Se observó que la lectura obtenida por el encoder pasaba de 4096 a 0 y esto hacía que el PID funcionase mal en ese rango.

Se barajaron varias ideas para solucionar el problema. Una era no llevar la cuenta absoluta, sino hacer el control de manera incremental. Otra opción era implementar vía software una solución para el salto brusco entre 0 y 4096 y que el PID detectase que se había pasado y retrocediese en lugar de seguir girando. Finalmente se optó por mantener las ventajas de tener la cuenta absoluta y simplemente reducir 20 grados la acción de cada unión; 10 grados por cada lado. De esta manera, con un PID sobreamortiguado no se llega nunca al punto crítico de funcionamiento con cierto margen de seguridad.

Con estos ajustes, el control de posición en bucle cerrado estaba terminado. Utilizando la cuenta absoluta del encoder, éste funciona esencialmente como un potenciómetro de 12 bits. Esto significa que el servomotor de funcionamiento continuo se ha convertido en uno de 340 grados con la salvedad de que no existe un tope mecánico, aunque se puede valorar su implementación más adelante.

#### 4.1.8 Octava etapa, diseño de la base, GUI y puesta en marcha:

Ya se había conseguido el control de posición en bucle cerrado y empezaba a ser el momento de construir definitivamente el robot. Para ello era necesario diseñar una pieza que se había dejado para el final, la base del robot con la primera articulación en horizontal.

Para ello se aprovechó la experiencia obtenida en el diseño de los segmentos y se planteó una solución similar. Por tanto, la pieza consta de un alojamiento para el rodamiento, un cuerpo cilíndrico hueco que acaba en una corona para atornillar a un sistema fijo y una estructura para atornillar el servomotor en su posición. Cuenta también con unos nervios para reforzar la estructura. En este caso fue imposible impedir la utilización de soportes a la hora de imprimir puesto que los voladizos están en ángulo recto con respecto a la horizontal. Fue necesario hacer dos iteraciones para llegar a una solución en la que el servomotor entrase con un ‘clic’ sin forzar la pieza.

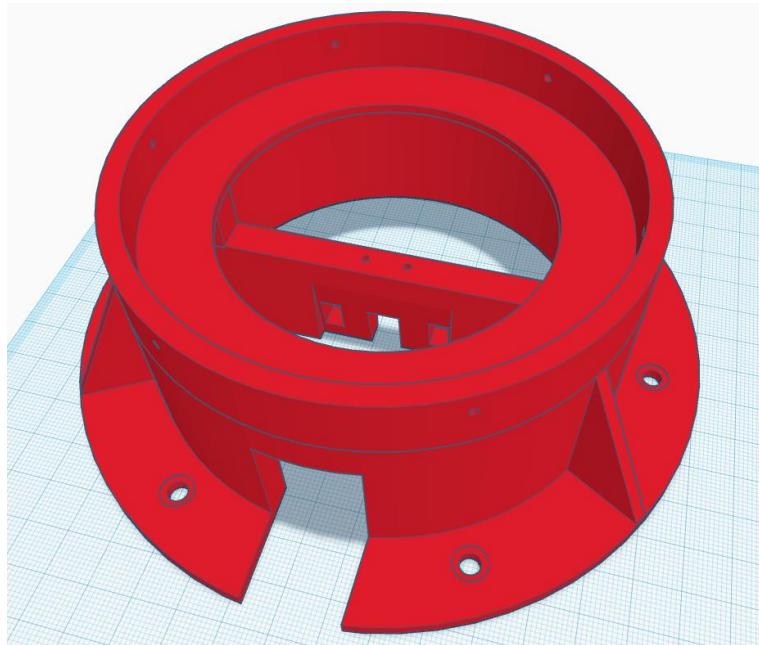


Ilustración 4.29: Diseño definitivo de la base del robot.

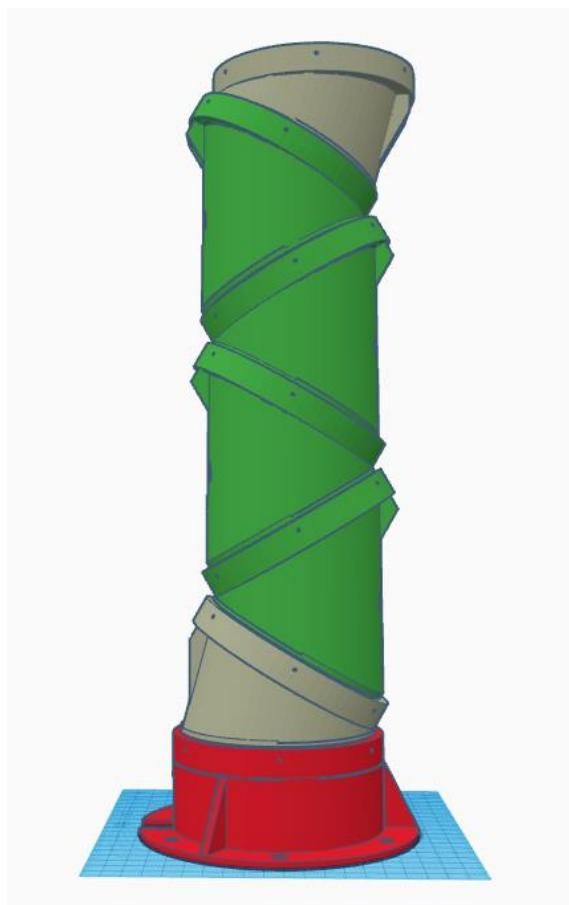


Ilustración 4.30: Vista del robot.

Tras esto se procedió realizar las últimas pruebas para verificar que el método utilizado para el control de posición en bucle cerrado funciona con varios grados de libertad, por lo que se montaron dos articulaciones (desacopladas para prevenir problemas en caso de fallo) y se adaptó el código de pruebas para admitir otro grado de libertad.

Utilizando un único PID se observó que el control no era suave, ya que el error acumulado se trasladaba de una articulación a la otra. Es necesario utilizar un PID distinto para cada articulación, aunque todos comparten constantes  $k_p$ ,  $k_i$  y  $k_d$ .

Con el segundo PID el control era adecuado. Si bien es cierto que la respuesta era ligeramente distinta entre ambos servomotores, la diferencia no es significativa. Por tanto, ya solo restaba el diseño de la GUI. Para ello se adaptó nuevamente el código de las pruebas haciendo que la placa de Arduino Mega 2560 [29] lea cada vez que estuviese disponible el puerto serie. La información se le manda como una cadena de caracteres con los valores articulares separados por coma, “q1,q2,q3,q4,q5,q6”. Arduino lee las primeras 5 veces hasta el separador coma “,” y la última hasta el carácter final de línea “\n” y guarda los valores en 6 variables tipo string. Luego las convierte a entero y le añade un desplazamiento de valor  $180^\circ$ . Los encoders tienen un rango entre  $0^\circ$  y  $360^\circ$  (están centrados en  $180^\circ$ ), en cambio la GUI da un rango entre  $-170^\circ$  y  $170^\circ$  (está centrada en  $0^\circ$ ).

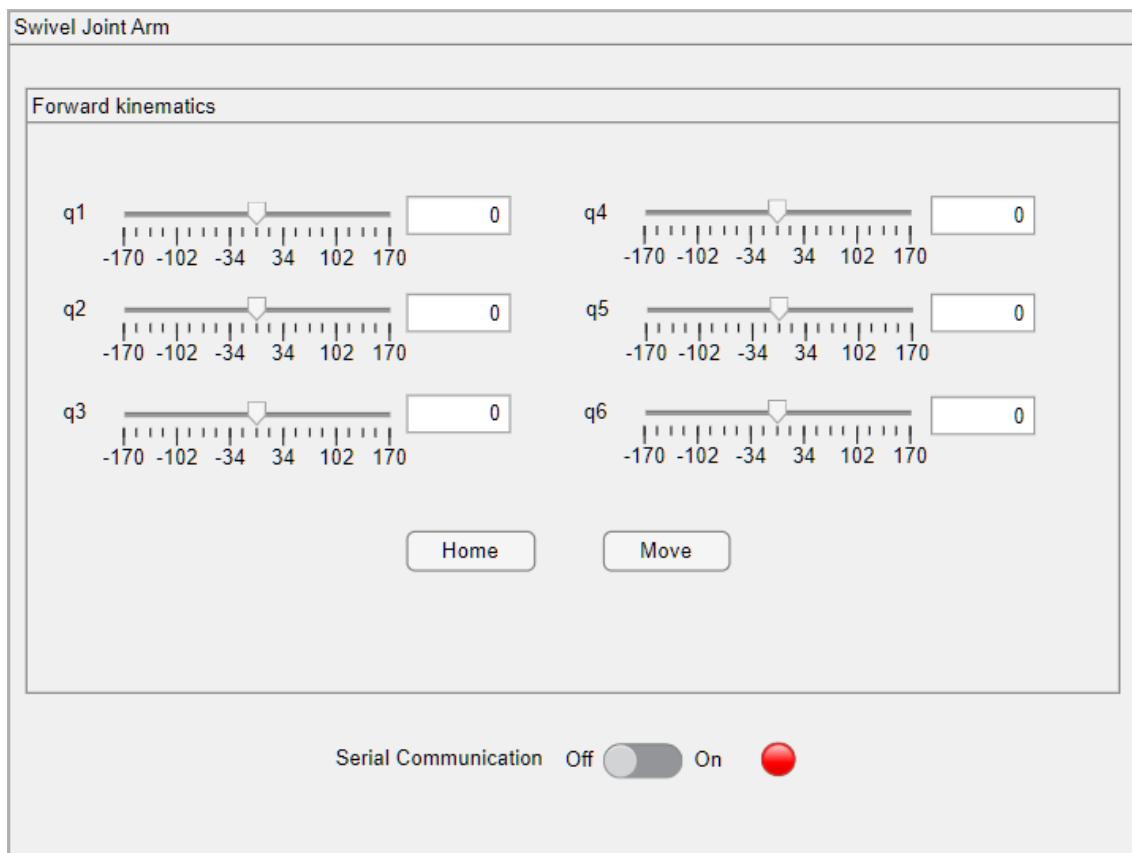


Ilustración 4.31: Graphical User Interface desarrollada con Matlab App Designer.

Como se puede apreciar, la GUI tiene 6 sliders conectados a 6 cuadros numéricos donde introducir, bien deslizando o bien escribiendo directamente el valor articular deseado. El botón *Move* envía por el puerto serie los valores elegidos para llevar a cabo el movimiento. El botón *Home* envía un 0 a cada articulación para que el robot vuelva a la posición de origen. Por último, hay un botón en la parte inferior para conectar y desconectar la comunicación por el puerto serie. El indicador de la derecha se enciende en color rojo si está deshabilitada y en verde si está habilitada.

La interfaz tiene además dos funcionalidades extra. La primera es que, si se desconecta la comunicación serie, antes de cerrarse envía un 0 a cada articulación. La segunda es que, si se cierra la aplicación sin apagar la comunicación serial, ésta se va a deshabilitar igualmente, no sin antes enviar un 0 de nuevo a cada articulación. De esta manera se garantiza que la

comunicación por el puerto serie esté cerrada si no está en uso y que el robot siempre esté en su posición inicial cuando se haya acabado de trabajar con él.

Con la interfaz ya terminada se procedió a la calibración de los PID de manera experimental. Finalmente, los parámetros para los 5 primeros grados de libertad fueron:

$$K_p=2 \quad K_i=0,5 \quad K_d=0,005$$

Se observó que el último grado de libertad tenía una sobreoscilación, por lo que se le redujo el  $K_p$  a 0,5.

## 4.2. Diseño 3D y fabricación

Se han empleado diferentes herramientas de diseño 3D para la creación de las piezas con la idea de sacar las mayores ventajas a cada una de ellas. Para los dientes oblicuos en dos direcciones del rodamiento se utilizó AutoCad 2023 [41] de Autodesk. Para pequeñas y rápidas modificaciones dimensionales, visualización 3D y obtención de capturas para la documentación, se utilizó Tinkercad [30] también de Autodesk. Para todo lo demás se utilizó el software libre FreeCad [42].

Lo primero que hay que destacar sobre el diseño de las piezas es que se han hecho sin tolerancias porque de esta manera el prototipado es más rápido. Se ha aprovechado la funcionalidad llamada “expansión horizontal” que proporciona el software Ultimaker Cura [31] para modificar la anchura de las capas en los ejes X e Y.

Esto ha sido determinante en la impresión del rodamiento. Con una expansión horizontal de -0.10mm las ruedas quedaban totalmente adheridas e inamovibles. Con una expansión horizontal de -0.20mm las piezas se imprimían separadas, pero tras un breve rodaje, las holguras empezaban a ser considerables. Fijando la expansión horizontal en -0.15mm se conseguía que las piezas quedasen solo levemente adheridas. Todo este ajuste fino habría sido inviable si se hubiese tenido que rehacer la pieza para cada prueba. Este es el motivo principal de transferir las tolerancias del diseño a la fabricación.

Para romper los puntos de fusión, se procedió a atornillar un listón de madera a dos de los tornillos diamatrales de la corona interior. A continuación, el listón se fijó a un tornillo de banco, y con la ayuda de una goma unida a un mango, se aplicó un par de rotación con la mano. Tras la ruptura de las soldaduras, era necesario hacer un rodaje extensivo de la pieza para pulir las superficies de fricción y minimizar el par resistente a la rodadura cuando estuviese en funcionamiento.

Cada una de las piezas se ha impreso con unos parámetros de densidad y expansión horizontal distintos. A continuación, se muestra una breve tabla con los principales parámetros con los que se han impreso todas las piezas (tabla 4.2) y una tabla donde se indican las variaciones de parámetros entre las piezas (tabla 4.3).

Altura de capa	Líneas de pared	Capas inferiores	Capas superiores	Patrón de relleno
0,2mm	3	5	5	Giroide
Temperatura de impresión	Velocidad de impresión	Velocidad de desplazamiento	Refrigeración de capa	Estructura de adherencia
200°C	40mm/s	120mm/s	Si	Ninguna

Tabla 4.2: Parámetros generales de impresión comunes a todas las piezas.

Pieza	Expansión Horizontal (mm)	Densidad de relleno (%)	Soportes
Base	-0,10	20	Si
Piñón	-0,10	20	No
Rodamiento	-0,15	20	No
Segmento 30°	-0,10	15	No
Segmento 60°	-0,10	15	No
Soporte encoder	0	15	No
Soporte imán	0	15	No

Tabla 4.3: Parámetros específicos de cada pieza.

También hay que destacar otras características de las impresiones como son el peso de cada pieza, la longitud de filamento utilizado y el tiempo de impresión tanto individual como total.

Pieza	Peso/ud (con soportes)(g)	Peso total(g)	Longitud/ud (m)	Longitud total (m)	Tiempo /ud	Tiempo total
Base	123 (150)	150	50,46	50,46	15h47'	15h47'
Piñón	2	18	0,58	3,48	13'	1h18'
Rodamiento	54	324	18,03	108,18	5h2'	30h12'
Segmento 30°	78	156	25,14	52,28	9h17'	18h34'
Segmento 60°	101	404	33,86	135,44	11h34'	46h16'
Soporte encoder	3	18	1,12	6,72	28'	2h48'
Soporte imán	1	6	0,50	3	12'	1h12'

Tabla 4.4: Pesos, longitudes de filamento y tiempos de impresión por pieza.

Peso total	Longitud total	Tiempo total
1kg 76g	359,56 m	4 días 20h 7'

Tabla 4.5: Peso, longitud de filamento y tiempo de impresión total del proyecto.

Todas las piezas se han impreso con un clon de una impresora 3D Prusa i3 Hephestos mk2 con marco de aluminio [43] (ilustración 4.27) fabricada, montada, modificada y calibrada por el autor. Se barajó la opción de utilizar una impresora Ender 3 [44] pero en pos de la homogeneidad se decidió utilizar una única máquina. Además de esta manera no era necesario duplicar el número de bobinas de plástico a utilizar a costa de alargar el tiempo de impresión.

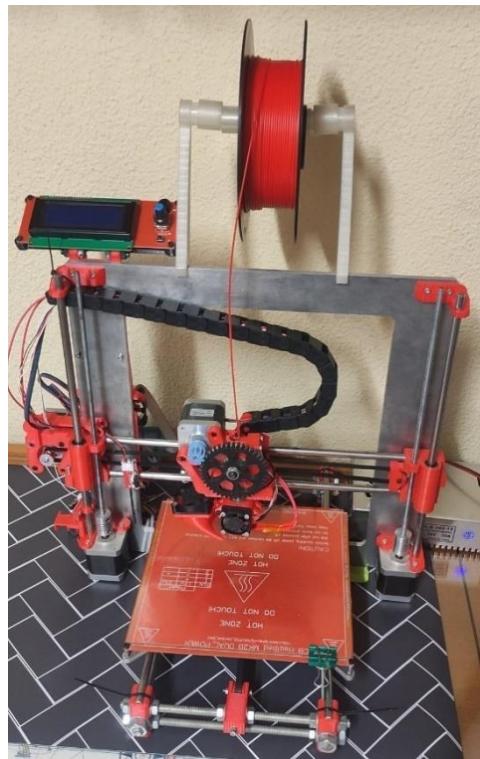


Ilustración 4.32: Impresora empleada en el proyecto.

El termoplástico utilizado para los prototipos ha sido el ácido poliláctico (PLA) de la marca Smart Materials [45] por los siguientes motivos:

- Precio. Al ser un plástico muy extendido, es muy económico.
- Facilidad de impresión. Tiene un amplio margen de error.
- Estabilidad dimensional en el enfriamiento. Apenas sufre warping aunque hay que tener cuidado en la impresión de voladizos sin soporte porque puede generar errores, sobre todo sin el uso de un ventilador de capa.
- Relativa facilidad de mecanizado. Si bien es cierto que ciertas operaciones acaban calentando y ablandando el plástico, en general es fácil de lijar y taladrar.
- No necesita el uso de cama caliente.
- No desprende olores ni gases tóxicos.

Para las piezas definitivas se buscaba un aumento en las propiedades mecánicas, por tanto, el termoplástico utilizado ha sido PLA+ de la marca iBoss [46]. Este plástico es una mejora del PLA convencional, al que se le añaden aditivos de distinta formulación según la marca, para mejorar de manera considerable las propiedades tanto térmicas como mecánicas del PLA. Con mayor módulo de Young y llegando casi a igualar la resistencia ante impacto del Acrilonitrilo Butadieno Estireno (ABS) pero a su vez evitando sus desventajas y manteniendo las ventajas del PLA sin aditivos (ilustración 4.28) [47].

Las principales desventajas del ABS son:

- Emisión de gases tóxicos que además tienen un fuerte olor a “plástico quemado”.
- Sufren mucho warping debido a la gran variación dimensional en el enfriamiento, lo que hace obligatorio el uso de cama caliente.
- Adhesión entre capas muy pobre debido también a esa fuerte dependencia volumétrica con la temperatura.

A todas las ventajas presentadas por el PLA+ hay que añadir que su precio es solo levemente superior al del PLA convencional y al del ABS, lo que lo hace idóneo para este proyecto. Por todo esto se descartaron otros plásticos y se optó por el PLA+.

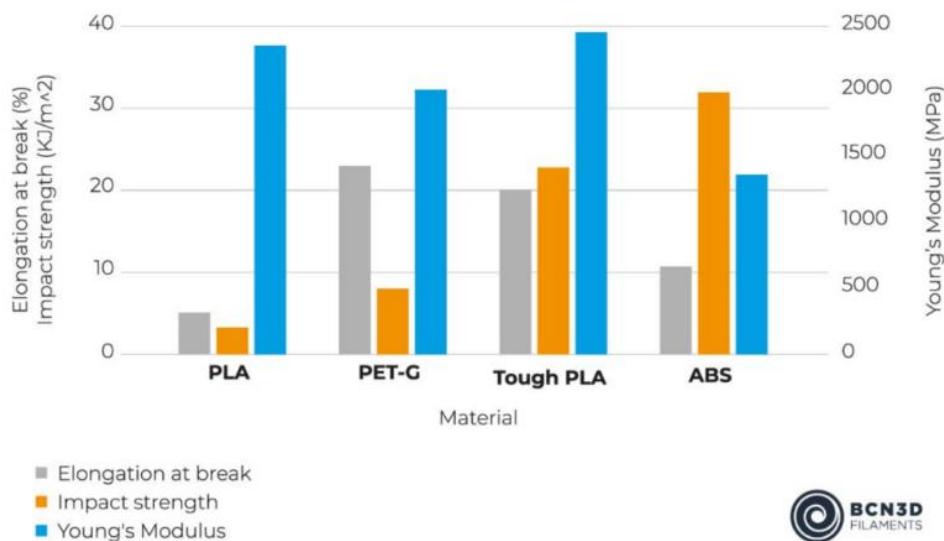


Ilustración 4.33: Comparativa de las propiedades mecánicas de los principales termoplásticos para impresión 3D [45].

#### 4.3. Dimensionamiento de los servomotores.

Para el dimensionamiento de los servos se realizó una pequeña hoja de cálculo en Excel con los pesos de las piezas y las distancias en la posición más desfavorable al servo que las iba a tener que mover.

Los pesos de las piezas son los obtenidos mediante el software Ultimaker Cura [31] y son los referidos en la tabla 4.4, mientras que los pesos de los servomotores empleados son los indicados en las hojas de especificaciones de los mismos. Para las distancias se utilizó el esquema de la ilustración 3.1 sabiendo que la distancia entre los centros de las caras es de 4,6 cm para el segmento de 30 grados y 7cm para el de 60.

Se llevaron a cabo varias simplificaciones, la primera y más importante es que se utilizó un modelo de masas puntuales. Los pesos de los segmentos se situaron a la mitad de la distancia entre caras, el peso tanto de los rodamientos como de los servomotores se situó sobre la articulación. El peso tanto de los piñones como de las piezas que sostienen el imán y el encoder, como el propio encoder, cables y tornillería se consideró despreciable. Esta simplificación es más desfavorable que la situación real, porque a pesar de despreciar varios pesos, la acción de la masa de los servomotores se ha situado considerablemente más lejos de la articulación. Por último destacar que en el primer grado de libertad oblicuo, en la situación más desfavorable actúa la fuerza con una proyección sobre un plano de  $30^\circ$ , por lo que tiene un factor de reducción de  $\cos(30^\circ)$ . Obsérvese que la primera articulación, al ser horizontal, no tiene que superar un par resistente al giro, por lo que los cálculos no se han realizado para ese punto.

A continuación, se indica una breve tabla con los valores obtenidos con la tabla Excel para el dimensionamiento de los servomotores. Para la selección de servomotor se ha omitido la acción de la reducción 1:3 para tener ese factor de seguridad, y por tanto se han buscado servos que, sin acción de la reductora, pudiesen superar esos momentos.

Articulación	2	3	4	5	6
Par resistente máximo (kg · cm)	13,067	9,178	4,722	1,723	0,179

Tabla 4.6: Pares resistentes máximos de las articulaciones para el dimensionamiento de los servomotores.

Para los 5 primeros segmentos se ha utilizado el servomotor TowerPro MG995 [48] que proporciona entre 9,4 kg·cm y 13 kg·cm trabajando a 4,8v y entre 11 kg·cm y 15 kg·cm trabajando a 6v (según distintas fuentes). Es un servomotor de funcionamiento continuo con la reductora interna metálica, lo que le permite soportar grandes cargas. Es cierto que para el primer grado de libertad en la estimación más desfavorable no se cumple que supere el par resistente máximo sin necesidad de la acción de la reductora. En este caso se tendría un par de 33kg·cm frente a los 13,067kg·cm necesarios. Para ese caso simplemente se tiene menos factor de seguridad y se ha considerado que un factor de 2,5 en lugar de 3 es asumible. Para el resto de las articulaciones, el servomotor seleccionado cumple las especificaciones sobradamente.



Ilustración 4.34: Servomotor MG995 [48].

Dado el mínimo par resistente a superar por el último grado de libertad, se planteó la opción de utilizar un microservomotor SG90 o equivalente [49] modificado para el funcionamiento continuo. Proporcionan un par según especificaciones de 2,5 kg·cm con un peso reducido (9 gramos) y podrían utilizarse incluso para el gdl 5. No obstante, al tener un factor de forma distinto, implicaría rediseñar los segmentos y la sujeción del encoder solo para esa unión, así que se optó por un servomotor con el mismo factor de forma que el resto. El servomotor utilizado en la última articulación es un Turnigy TGY-AN3 [50] porque tiene el mismo rango de funcionamiento (4,8v-6v) y proporciona un par de entre 3,2 kg·cm y 3,6 kg·cm, que es más que suficiente. Además, al tener la reductora interna de plástico, supone una reducción de peso de 17g frente al TowerPro MG995 [48].



Ilustración 4.35: Microservomotor TowerPro SG90 [49].



Ilustración 4.36: Servomotor Turnigy TGY-AN3 [50].

Va a ser necesario añadir una extensión a los cables de los servomotores, por lo que también se va a dimensionar el cable de dicha extensión. Para ello se parte de los datos tomados de la fuente de alimentación con un servomotor en funcionamiento continuo.



Ilustración 4.37: Valores de corriente de un servomotor.

Se puede observar que salvo unos picos instantáneos, la corriente demandada ronda los 250mA, por lo que se va a emplear cable AWG 22. A continuación se muestra una tabla con las principales características del estándar de clasificación AWG.

AWG	Dia mm	SWG	Dia mm	Max Amps	Ohms / 100 m
11	2.30	13	2.34	12	0.47
12	2.05	14	2.03	9.3	0.67
13	1.83	15	1.83	7.4	0.85
14	1.63	16	1.63	5.9	1.07
15	1.45	17	1.42	4.7	1.35
16	1.29	18	1.219	3.7	1.48
18	1.024	19	1.016	2.3	2.04
19	0.912	20	0.914	1.8	2.6
20	0.812	21	0.813	1.5	3.5
21	0.723	22	0.711	1.2	4.3
22	0.644	23	0.610	0.92	5.6
23	0.573	24	0.559	0.729	7.0
24	0.511	25	0.508	0.577	8.7
25	0.455	26	0.457	0.457	10.5
26	0.405	27	0.417	0.361	13.0
27	0.361	28	0.376	0.288	15.5
28	0.321	30	0.315	0.226	22.1
29	0.286	32	0.274	0.182	29.2
30	0.255	33	0.254	0.142	34.7
31	0.226	34	0.234	0.113	40.2
32	0.203	36	0.193	0.091	58.9
33	0.180	37	0.173	0.072	76.7
34	0.160	38	0.152	0.056	94.5
35	0.142	39	0.132	0.044	121.2

Ilustración 4.38: Estándar AWG [51].

#### 4.4 Electrónica y conexiones.

Se va a realizar una breve exposición de la electrónica utilizada, así como de los montajes eléctricos necesarios.

La placa de desarrollo con la que se va a controlar el proyecto va a ser la de Arduino Mega 2560 o equivalente [29] por su mayor capacidad de procesamiento frente a otras opciones como, por ejemplo, Arduino Uno R3 [52]. La placa recibirá por el puerto serie los datos proporcionados por Matlab [21], y se encargará tanto de ejecutar el control de los PID como de controlar el puerto I2C como maestro para mandar constantemente instrucciones al multiplexor TCA9548A [37], y controlar así los servomotores. También suministrará alimentación a los encoders mediante el puerto de 3,3V. La alimentación a 5v de los servomotores, el puerto I2C y la alimentación de la lógica del multiplexor la suministrará una fuente de tensión externa.

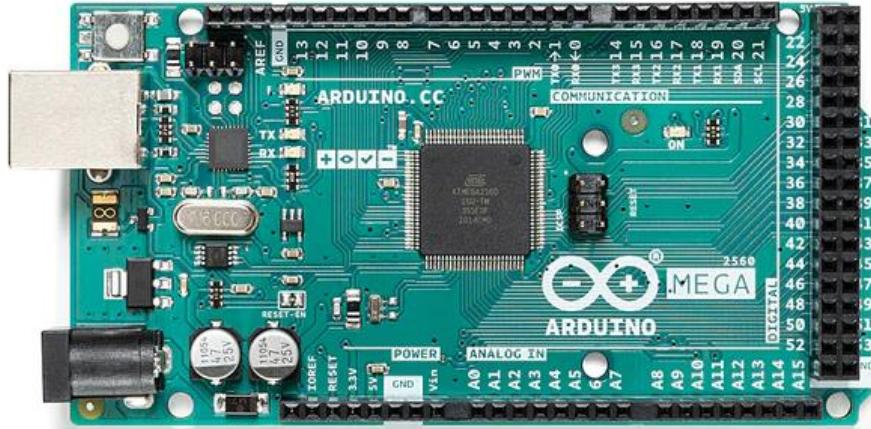


Ilustración 4.39: Placa de desarrollo Arduino Mega 2560 [29].

Todas las conexiones se harán sobre una protoboard o placa de pruebas por simplicidad, en concreto se va a usar la MB-102 por su tamaño y número de conexiones.

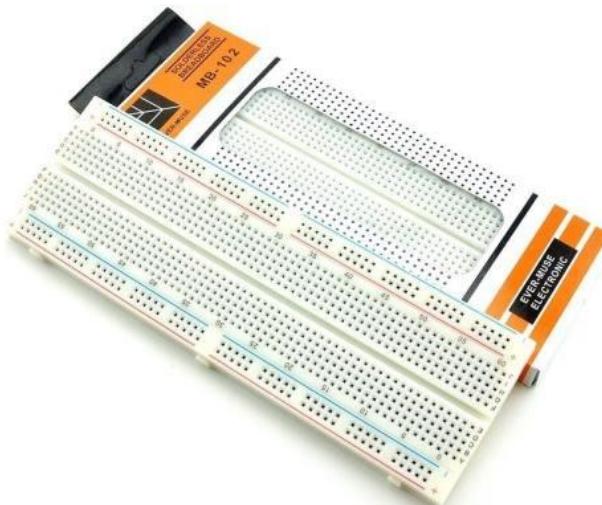


Ilustración 4.40: Protoboard MB-102.

El multiplexor funcionará como esclavo y estará conectado en al puerto I<sub>2</sub>C mediante dos resistencias de pull-up de  $4,7\text{k}\Omega$ , una para cada cable (SDA y SCL), según el esquema siguiente [53].

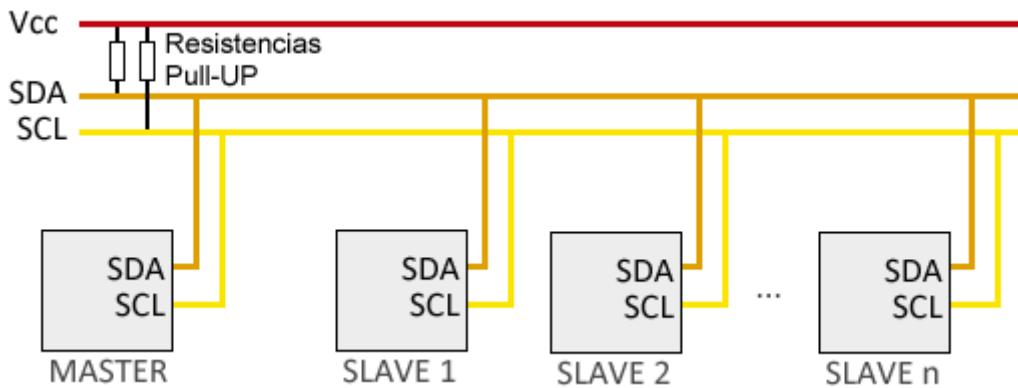


Ilustración 4.41: Esquema de conexión I2C [53].

Como ya se ha indicado anteriormente, el multiplexor TCA9548A [37] tiene 3 pines para programar la dirección I2C a utilizar, el A0, A1 y A2. Para ello basta con conectar los pines a tensión o masa según se indique en la tabla 4.7 para seleccionar la dirección deseada. En este caso, se han conectado los 3 cables a GND y por tanto la dirección con la que se trabaja es la 0x70.

Por último, hay 8 parejas de pines SDA<sub>i</sub> y SCL<sub>i</sub> que irán conectados a los pines SDA y SCL de cada uno de los 6 encoders, quedando por tanto 2 parejas libres.

Destacar que ambas tiras de pines vienen por separado, y es necesario realizar la soldadura de los mismos a la placa.

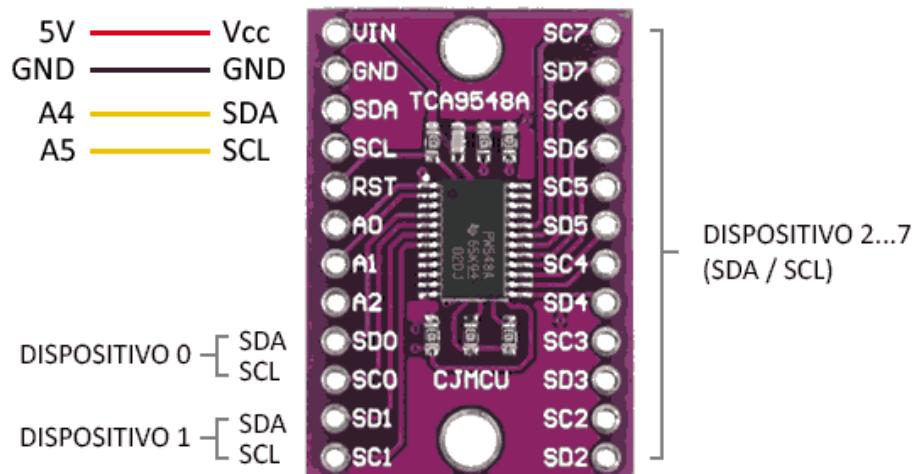


Ilustración 4.42: Pin-out de multiplexor I2C AS5600 [38] [39].

A2	A1	A0	Dirección I2C
L	L	L	0x70
L	L	H	0x71
L	H	L	0x72
L	H	H	0x73
H	L	L	0x74
H	L	H	0x75
H	H	L	0x76
H	H	H	0x77

Tabla 4.7: Programación de las direcciones I2C del multiplexor AS5600 [39]. L=Low, H=High.

Los encoders a utilizar son los AS5600 [39], como ya se ha mencionado anteriormente, que tienen la siguiente disposición de pines o pin-out.

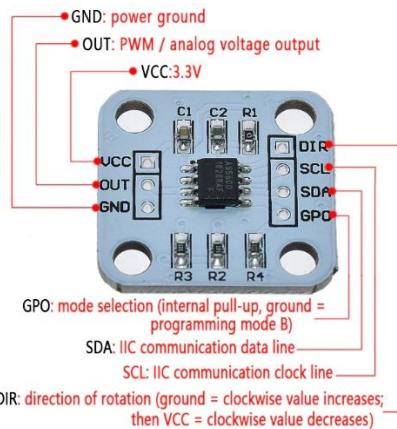


Ilustración 4.43: Pin-out de encoder AS5600 [39].

Como se puede apreciar, consta de 7 pines, pero para reducir la cantidad de cable necesario se han tomado las siguientes medidas.

No es necesario el pin GPO ya que no va a programarse el encoder debido a la imposibilidad de la reprogramación. A la hora del montaje, se va a llevar a cabo una cuidadosa colocación del imán y el ajuste fino de la posición 0 se hará mediante software en caso de ser necesario.

Lo mismo ocurre con el pin OUT, la comunicación se hará mediante I2C por lo que la salida analógica y de PWM se dejará al aire.

Para evitar un posible ruido al dejar el pin DIR al aire, se va a puentejar en la propia placa mediante la soldadura de un cable hasta el pin GND y de esta manera, ahorrar llevar un cable extra a la placa de pruebas.

Los pines de la alimentación, así como los de la comunicación I2C son necesarios por motivos evidentes.

La alimentación se llevará a cabo mediante la placa de Arduino Mega 2560 [29], conectando el pin de 3,3V. Destacar que esta conexión se hará, como indican las especificaciones, con un condensador electrolítico de desacople de  $1\mu F$  para cada encoder. De esta manera evitamos el posible ruido que pueda llegar a las mediciones.

Nuevamente, indicar que la tira de pines va por separado y es necesario hacer la soldadura para cada encoder.

A continuación, se expone un breve resumen de cada una de las conexiones:

**Servomotores:** Alimentación conectada a 5v proporcionados por una fuente de suministro y a través de un condensador de desacople de  $470\ \mu F$ . Datos conectados a los pines 2-7 de Arduino Mega 2560 [29].

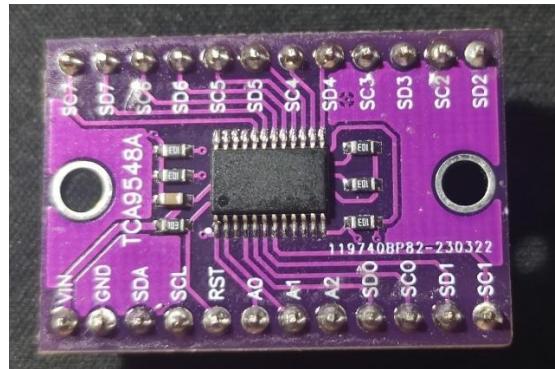
**Encoders:** Alimentación conectada al pin de 3,3v de Arduino Mega 2560 [29] a través de un condensador de desacople de  $1\mu F$  para cada encoder. Los pines SCL y SDA conectados a los pines SCx y SDx del encoder multiplexor TCA9548A [37].

**Multiplexor:** Alimentación conectada a 5v proporcionados por una fuente de suministro. Pines A0, A1 y A2 conectados a GND de Arduino Mega 2560 [29]. Pines SCL y SDA conectados a los pines SCL y SDA de Arduino mediante una resistencia de pull-up de  $4,7k\Omega$ .

## 5. Guía de montaje

En este punto se va a desarrollar una guía paso a paso del montaje del robot.

Lo primero es llevar a cabo la soldadura de los pines tanto del multiplexor como de los encoders. Soldar también un cable de unos 3,5cm que puentee el pin de dirección *DIR* con el pin *GND* en la placa del encoder.

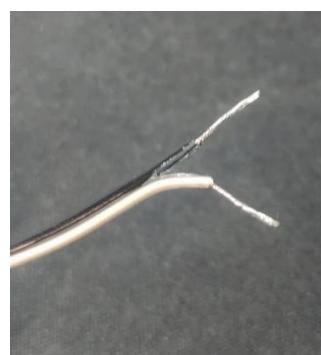


*Ilustración 5.1: Soldadura de los pines del multiplexor I2C.*



*Ilustración 5.2: Soldadura y puenteado de los pines del encoder.*

Es necesario fabricar los cables tanto de alimentación como de comunicación I2C de los encoders. Para ello se parte de una bobina de cable AWG 28 multicolor y se separan por un lado los colores amarillo y verde para la comunicación I2C y los colores blanco y negro para la alimentación. Es necesario separar ligeramente los cables y pelar aproximadamente 7mm en cada extremo. Los hilos se enrollan sobre si mismos y se cortan los últimos 2mm.



*Ilustración 5.3: Extremos preparados de los cables de alimentación del encoder.*

Con los extremos preparados se procede a crimpar los cables. En un extremo se pondrán pines hembra y en el otro, pines macho. Destacar la importancia de conservar la orientación a la hora de crimpar para evitar tener que retorcer el cable más adelante, cuando se introduzcan los cables en el conector.



Ilustración 5.4: Crimpadora y cables de alimentación del encoder con pines hembra.

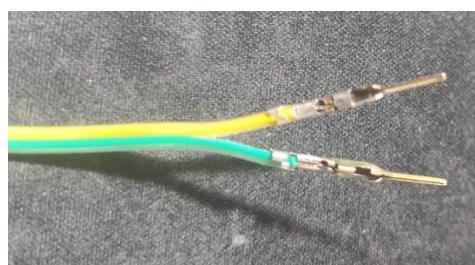


Ilustración 5.5: Cables de comunicación I2C del encoder con pines macho.

Se procede a colocar los conectores a los pines hembra. Los cables de comunicación I2C utilizan conectores JST de 2 pistas, mientras que la alimentación, debido a que el pinout del encoder tiene el pin *OUT* entre los pines *VCC* y *GND*, utilizan una clavija DuPont de 3 pistas. Destacar la importancia de dejar la pista del medio vacía.



Ilustración 5.6: Cables de alimentación del encoder con conector DuPont de 3 pistas.



Ilustración 5.7: Cables de comunicación I2C del encoder con conector JST de 2 pistas.

Para los extremos macho de los cables se va a utilizar un tubo de plástico termorretráctil para enfundar los pines además de identificarlos. Para los cables de alimentación se sigue el código estándar de colores y se utiliza el negro para el cable negro, ya que irá conectado a GND y el rojo para el cable blanco, ya que irá conectado a tensión. Por indisponibilidad de colores se ha usado en los cables de comunicación I2C un tubo azul para el cable verde y un tubo negro para el cable amarillo.

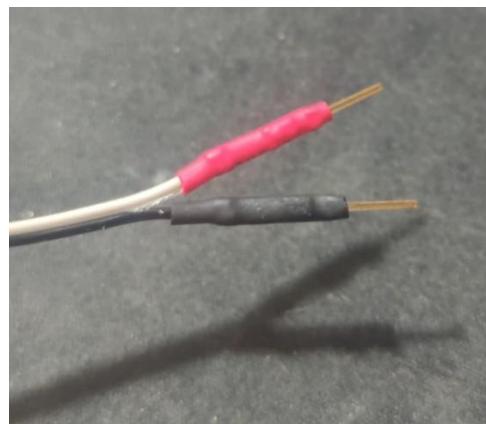


Ilustración 5.8: Tubo termorretráctil en los pines macho de los cables de alimentación del encoder.

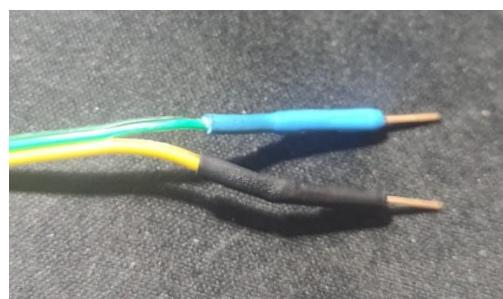


Ilustración 5.9: Tubo termorretráctil en los pines macho de los cables de comunicación I2C del encoder.

Se conectan los cables a los pines del encoder. Para la alimentación se utiliza el código de colores estándar y el cable negro se conecta a GND y el blanco con tubo rojo a VCC. Para los cables de comunicación I2C, el amarillo se conecta al reloj, pin *SCL*, y el verde al pin de datos *SDA*.

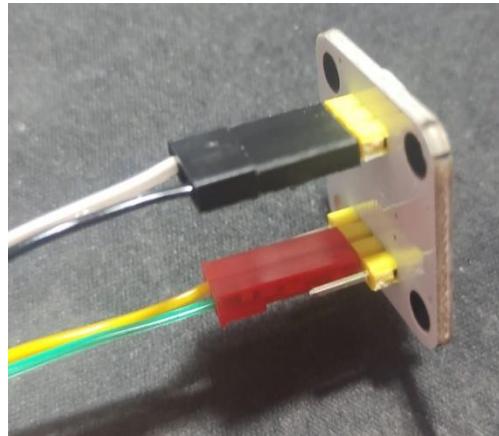


Ilustración 5.10: Conexión de los cables al encoder.

A pesar de que los servomotores vienen con un cable, este no es lo suficientemente largo, por lo que es necesario fabricar unos extensores. Para ello se va a utilizar cable AWG 22. Los cables tienen que ser macho-macho puesto que en un extremo van conectados a la clavija hembra de los servomotores y por el otro van conectados a los pines de la protoboard y de Arduino. Se les va a añadir una clavija DuPont de 3 pistas a un extremo. El otro irá con tubo termorretráctil del mismo color que el cable, salvo en el caso del cable blanco que se utilizará tubo azul. Para evitar posibles desconexiones, se le va a añadir unos clips de seguridad a la unión con el servo.

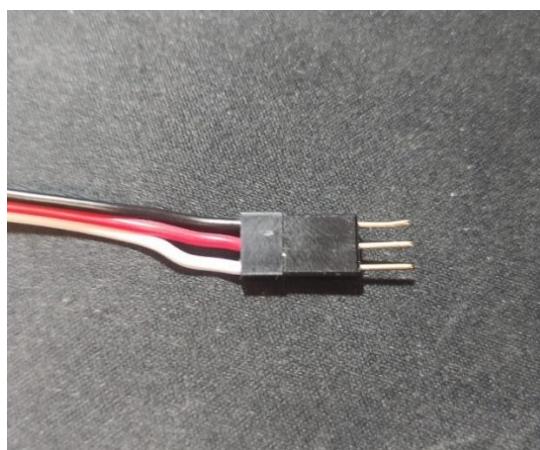


Ilustración 5.11: Cables de extensión del servomotor.



Ilustración 5.12: Tubo termorretráctil en los pines del cable de extensión del servomotor.

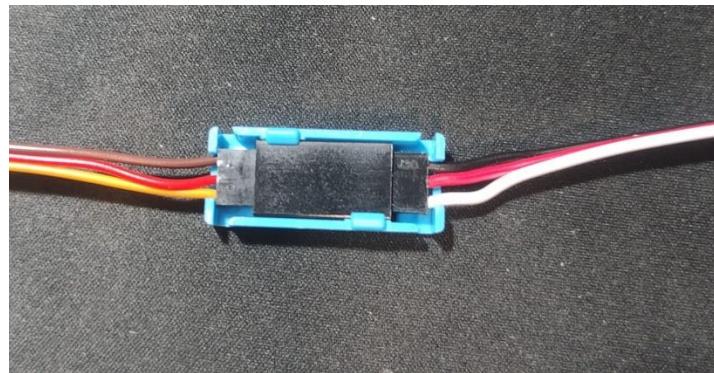


Ilustración 5.13: Clip de seguridad en la extensión del cable del servomotor.

En este punto se va a proceder al montaje de la primera articulación. Se atornilla el servomotor en su posición utilizando la tornillería proporcionada por el fabricante. Recordar que el soporte del encoder se atornilla junto con el servomotor.

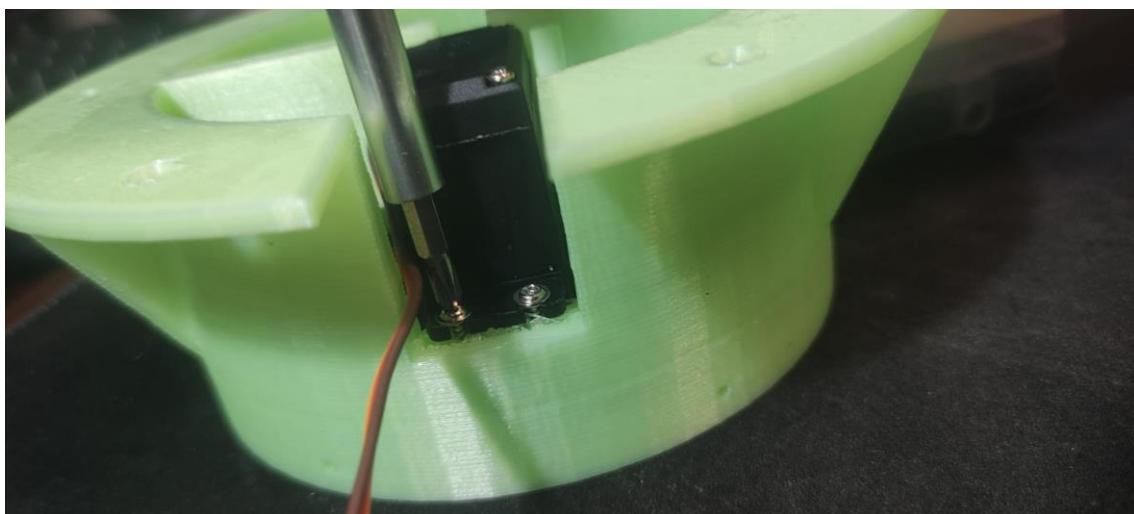


Ilustración 5.14: Atornillado del servomotor de la primera articulación.



Ilustración 5.15: Atornillado del soporte del encoder y el servomotor de la primera articulación.

Con el encoder fijado en su sitio, se pasa el cable por el alojamiento previsto para ello. Y se procede a atornillar el piñón al eje del servomotor. El fabricante nuevamente suministra un tornillo para esta función. Es importante atornillarlo hasta el fondo para que el piñón no sobresalga por encima del soporte del encoder, puesto que chocaría con la placa.



Ilustración 5.16: Atornillado del piñón.

Se atornilla el encoder en su posición. Para ello se han utilizado 4 tornillos autorroscantes DIN7981 M3\*5. Los cables se pasan por los laterales del soporte del encoder y posteriormente por los alojamientos previstos a tal efecto. Obsérvese que la distancia entre caras del soporte del encoder es tal que permite sujetar en posición los pines, previniendo una posible desconexión en caso de tirón.

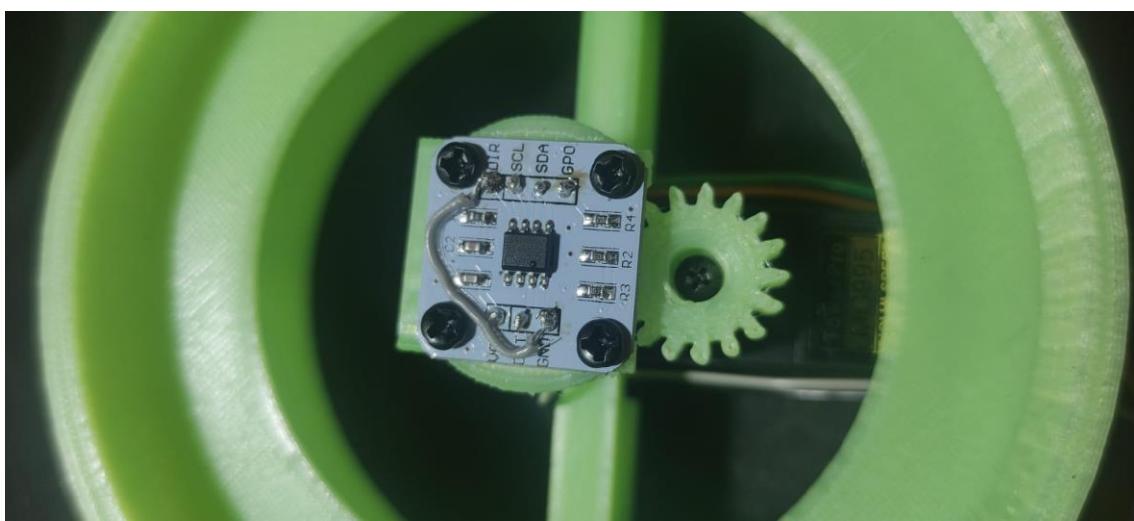


Ilustración 5.17: Atornillado del encoder.



Ilustración 5.18: Gestión de los cables.

Se coloca en su posición el rodamiento y se atornilla la corona exterior a la base por los laterales mediante tornillos autorroscantes DIN7981 M2.3\*8. Se aconseja hacer la rosca a los agujeros del rodamiento antes de colocarlo, introduciendo el mismo tornillo que se va a usar en el orificio. Si bien no es necesario, va a facilitar el montaje.

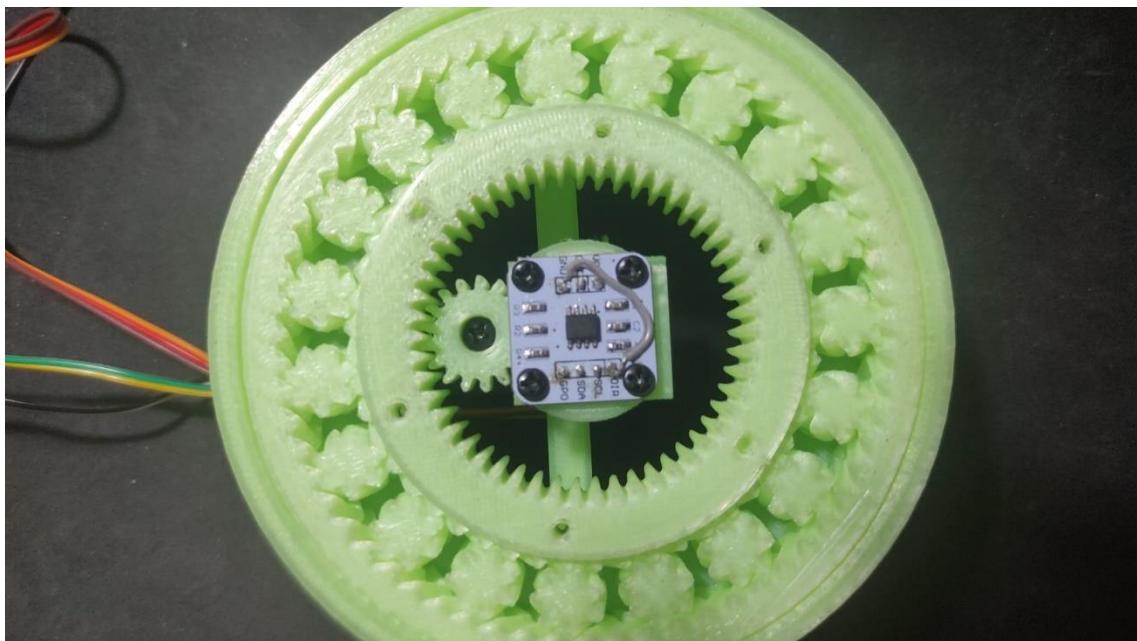


Ilustración 5.19: Colocación del rodamiento.

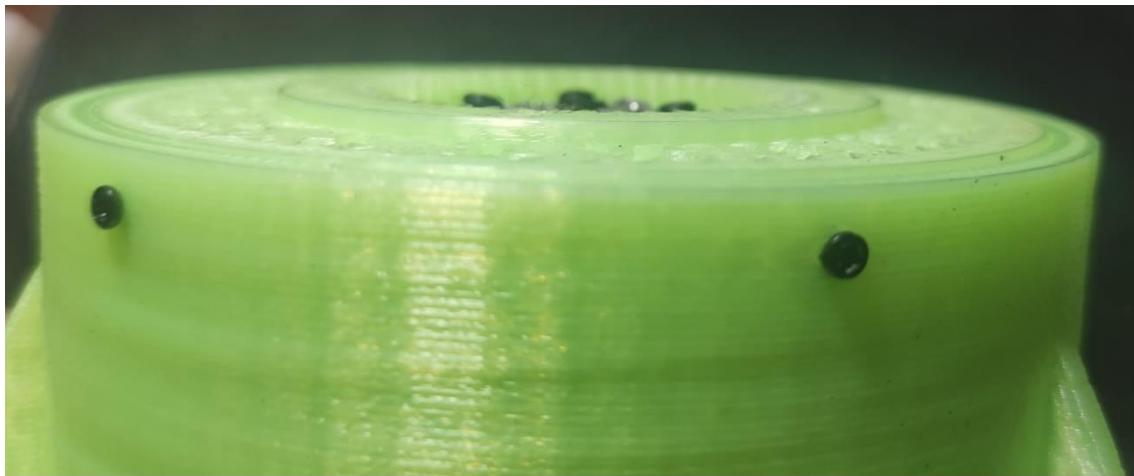


Ilustración 5.20: Atornillado de la corona exterior del rodamiento.

En este punto hay que calibrar la colocación de los imanes. Para eso se necesita utilizar el código que proporciona las lecturas del encoder. El proceso consiste en introducir el imán y medir el valor del encoder, ajustando levemente la posición en cada iteración. Para sacar el imán de la pieza se puede usar un pin macho de un cable DuPont o una lezna. Sirve de ayuda hacer una marca en la base del soporte del imán. Cuando la marca queda alineada con el servomotor y el sensor detecta  $180^\circ$ , el imán está en su posición. Se recuerda que se hace esto para evitar programar el encoder puesto que éstos no son reprogramables. No es necesario atornillar la pieza a la corona interior del rodamiento, solo presentarla encima. Se aconseja utilizar como referencia el punto en el que el imán se pega al cable DuPont. En ese punto se encuentra la unión de los polos norte y sur del imán, lo que significa que está el valor de  $0^\circ$  o el de  $180^\circ$ . Basta con identificar cuál es y colocarlo en su posición.

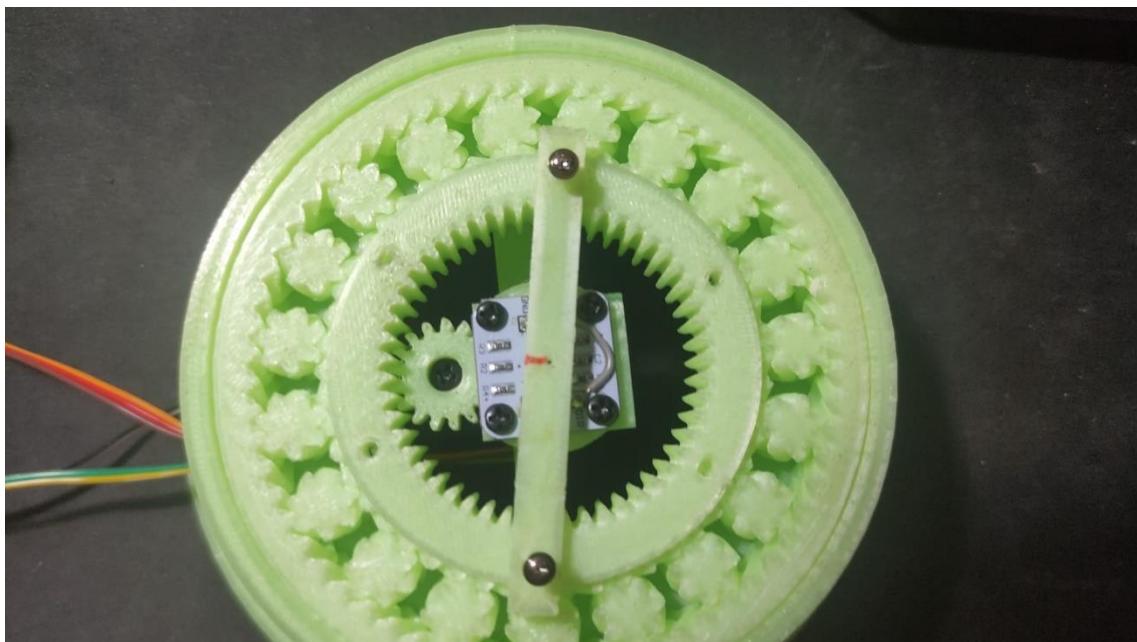


Ilustración 5.21: Calibración de la colocación de los imanes. Se observa una leve desviación.

Una vez se ha calibrado el imán, se procede a atornillar el segmento del siguiente grado de libertad. En este caso la tornillería utilizada es autorroscante DIN7981 M2.3\*10. El soporte de imán se atornilla junto con el segmento en el diámetro perpendicular al eje de simetría. Nuevamente se aconseja hacer la rosca previamente a los agujeros del rodamiento. A partir de aquí sólo queda repetir el mismo proceso para cada grado de libertad. Se aconseja atornillar el servomotor del siguiente segmento antes que el segmento en sí mismo para facilitar el montaje. El diseño modular del robot facilita añadir tantos grados de libertad como se busque (teniendo en cuenta las limitaciones mecánicas). Se recuerda que el orden de los segmentos es 30°, 60°, 60°, 60°, 60° y 30°.

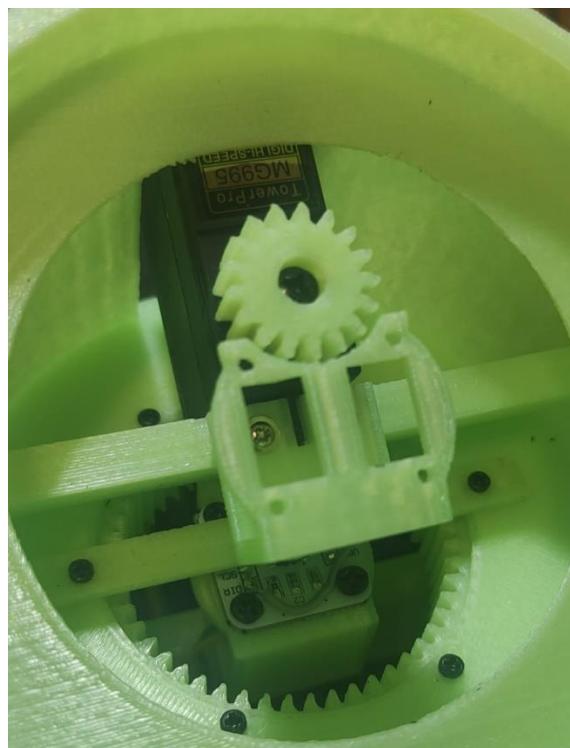


Ilustración 5.22: Atornillado del segmento de 30° a la base.

A continuación, se muestra una imagen del robot montado en el estado de reposo. Si se han calibrado bien los imanes, las articulaciones tienen que quedar rectas.



Ilustración 5.23: Robot montado.

## 6. Resultados y discusión

Inicialmente se ha llevado a cabo un estudio del estado del arte sobre el uso de las articulaciones oblicuas en robots manipuladores y cómo ha evolucionado en las últimas décadas, así como una clasificación de los tipos de robots para justificar por qué el robot de este proyecto es un manipulador modular no redundante.

Se han obtenido de manera satisfactoria los parámetros de Denavit-Hartenberg para este robot, lo que ha permitido calcular las matrices de cambio de base entre los sistemas de referencia de los distintos segmentos.

Mediante estas matrices se ha llegado a la obtención de un modelo cinemático directo, si bien es cierto que la altísima no linealidad y el número de grados de libertad ha impedido que se plasme el resultado de manera analítica.

Se ha conseguido documentar la problemática del cálculo de un modelo cinemático inverso para este tipo de robots, lo que ha permitido justificar el escaso uso en la industria de estos robots con más de 3 grados de libertad.

Se ha mostrado una metodología para el cálculo del modelo diferencial mediante la obtención de la matriz jacobiana, aunque, de nuevo, la elevadísima no linealidad y el número de grados de libertad han impedido que se plasme de manera analítica.

Es un campo objeto de estudio en la actualidad, por lo que la documentación tan actualizada (julio de 2023) ha permitido la ilustración de las ventajas que proporcionan este tipo de robots mediante el análisis por simulación del espacio de trabajo.

Se ha conseguido un diseño satisfactorio de robot con unos ángulos en las articulaciones que lo hacen muy versátil.

El proceso de prototipado rápido ha proporcionado buenos resultados, llegando a la obtención de segmentos completamente modulares. Las piezas cumplen todos los requisitos y se han ido adaptando a los contratiempos que se han ido encontrando.

El método de fabricación ha permitido la obtención de piezas que de otra manera habría sido imposible fabricar, muestra de ello es el diseño del rodamiento. Los dientes oblicuos en dos direcciones impiden que se desmonte. De no fabricarse directamente en su posición, habría sido imposible armarlo.

El control de posición en bucle cerrado es completamente satisfactorio, el uso de los encoders de efecto Hall ha sido un acierto que ha permitido anular el error de posición. Su implementación mediante I2C ha facilitado mucho la programación al funcionar esencialmente como un potenciómetro de 12 bits de resolución.

Se han conseguido aprovechar las ventajas en la libertad de rotación al tener un motor continuo, pero añadiendo las ventajas de tener un servomotor con control de posición en bucle cerrado gracias al modo de funcionamiento I2C de los encoders.

La calibración de los imanes ha sido sencilla gracias a que el punto de unión entre los polos norte y sur del imán marcan los puntos 0º y 180º del funcionamiento del encoder.

La interfaz gráfica de usuario es completamente funcional, permite ingresar valores tanto con un slider como escribiendo directamente el valor deseado. Además, tiene dos funcionalidades que aseguran que el robot siempre que no está en uso, se encuentra en posición vertical.

## 6.1 Experimentos

Ya solo queda verificar la validez de lo estudiado en el capítulo 3.5 mediante experimentación con el robot.

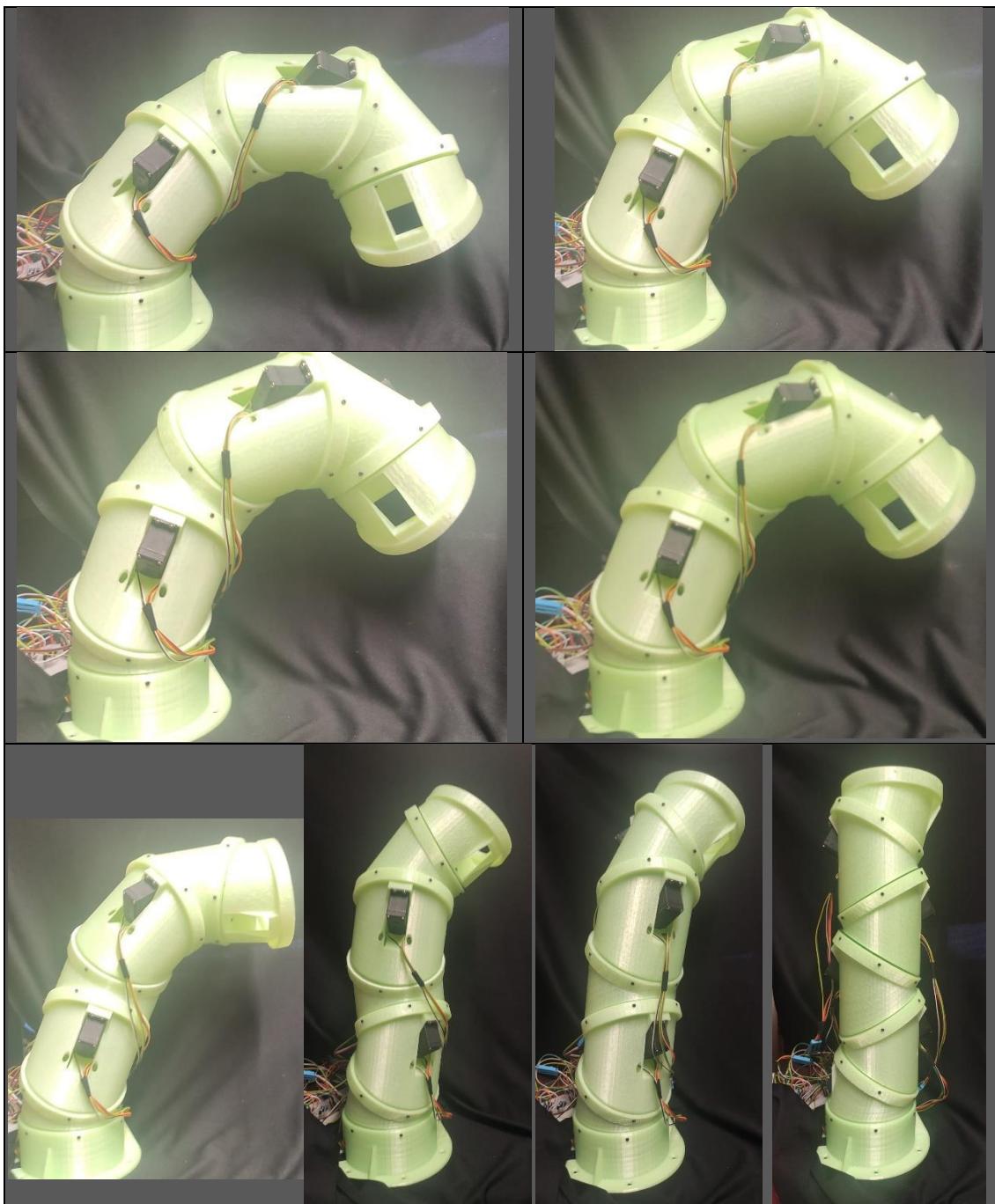


Ilustración 6.1: Experimento para validar el capítulo 3.5.

Como se puede apreciar, efectivamente es posible realizar el movimiento de parcialmente dobrado a estirado sin salirse del plano.

## 6.2 Impactos

A continuación, se van a analizar los impactos del desarrollo de este proyecto en los distintos aspectos que puedan verse afectados.

### 6.2.1 Aspectos legales:

Este proyecto está sujeto al Real Decreto 842/2002, del 2 de agosto, por el que se aprueba el Reglamento electrotécnico para baja tensión.

### 6.2.2 Aspectos medioambientales:

Debido al proceso de prototipado rápido, se han fabricado bastantes más piezas y prototipos de los necesarios. Además, el método de fabricación no es infalible y ha sido necesario volver a fabricar piezas que han salido con defectos.

Esto ha supuesto un consumo extra de energía y un gasto adicional de plástico.

Gracias a que, tanto el plástico utilizado para los prototipos, como el utilizado para las piezas finales son termoplásticos (PLA y PLA+ respectivamente), todas las piezas defectuosas y los prototipos desechados se pueden triturar y volver a fundir para crear nuevas bobinas de filamento perfectamente válido.

La marca Smart Materials [45] que se ha usado en este proyecto cuenta con una línea sostenible en la que tanto el plástico como el empaquetado se producen empleando material reciclado y consumo de proximidad. Además, tiene activa una campaña de recogida y reciclaje de los residuos de impresión.

### 6.2.3 Aspectos económicos:

Como se ha podido comprobar, este tipo de robots, a pesar de ofrecer grandes ventajas tienen el gran inconveniente de la dificultad en el control, lo que hace que no se usen a nivel industrial robots con más de 3 grados de libertad.

Conseguir desarrollar un método para el cálculo de la cinemática inversa y facilitar de esta manera el control del robot podría suponer la apertura de un nuevo nicho de mercado en distintos sectores.

### 6.2.4 Aspectos éticos:

Gracias al estudio del arte se ha podido apreciar que uno de los sectores pioneros en el desarrollo de este tipo de articulaciones es el militar y armamentístico.

La historia ha demostrado que el desarrollo de cualquier tecnología sea cual sea su objetivo original puede terminar con su uso a nivel bélico y militar.

### 6.2.5 Aspectos de seguridad laboral:

Se ha desarrollado una funcionalidad en la interfaz gráfica de usuario gracias a la cual, frente a cualquier desconexión, el robot vuelve a su posición inicial, reduciendo de esta manera el riesgo de aplastamiento del operario que lo utilice.

### 6.2.6 Objetivos de Desarrollo Sostenible

Este proyecto está alineado con los objetivos de desarrollo sostenible. Especialmente con los puntos:

9.5 Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo.

12.5 De aquí a 2030, reducir considerablemente la generación de desechos mediante actividades de prevención, reducción, reciclado y reutilización.

## 7. Conclusiones

Los robots manipuladores modulares con articulaciones oblicuas son muy atractivos por la simplicidad y versatilidad de sus movimientos. Al ser modulares permiten llegar al nivel de redundancia deseado de manera trivial.

Lamentablemente, la dificultad para desarrollar un modelo cinemático inverso impide el control significativo de este tipo de robots, lo que en la actualidad reduce fuertemente su aplicación a nivel industrial.

Este trabajo ha intentado desarrollar la teoría necesaria para llevar a cabo el control mínimo del robot y plasmar y documentar las dificultades de seguir profundizando en dicho control.

Además, el prototipo construido permite ilustrar perfectamente la simplicidad y versatilidad de la que se ha hablado con un control muy sencillo gracias a su interfaz de usuario.

### 7.1 Líneas Futuras

Se podrían llevar a cabo varias mejoras, principalmente en la interfaz gráfica de usuario.

Algunas funcionalidades que se plantean son el control de la cinemática inversa (implementando métodos numéricos) y la programación por puntos. Son líneas que se planteó introducir en un principio pero que se vieron fuera del alcance del proyecto por su extensión.

Se muestra a continuación una propuesta de dicha interfaz:

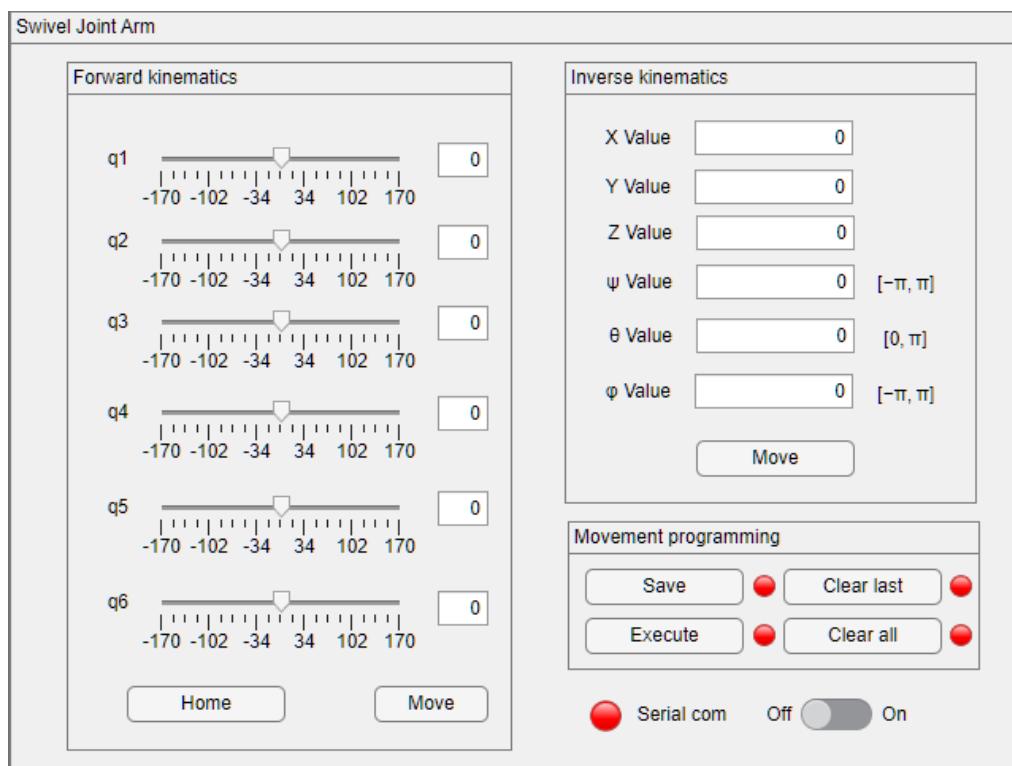


Ilustración 7.1: Propuesta de interfaz avanzada.

También se planteó añadir a la interfaz una simulación del robot para que se visualizase el movimiento antes de realizarlo. Para ello se desarrolló un modelo de simulink al que se

añadieron las articulaciones del robot y un sistema de referencia en el extremo. Esta herramienta permite la visualización de simulaciones, pero, de nuevo, excedía la extensión del trabajo y no se terminó de implementar.

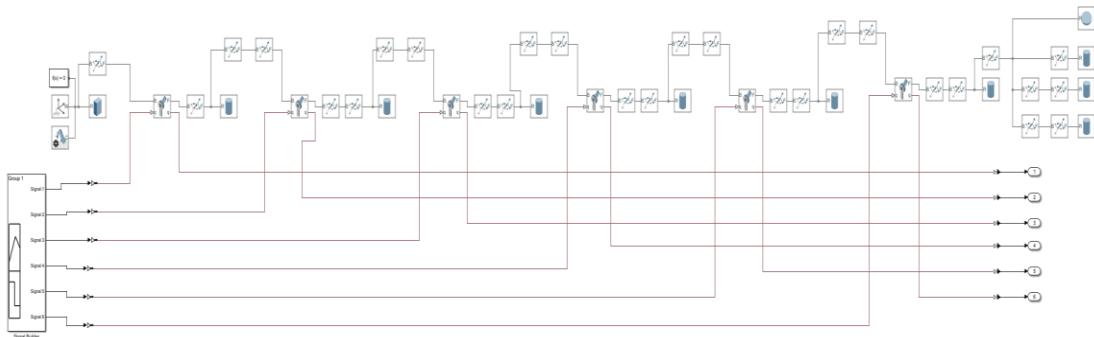


Ilustración 7.2: Modelo de Simulink.

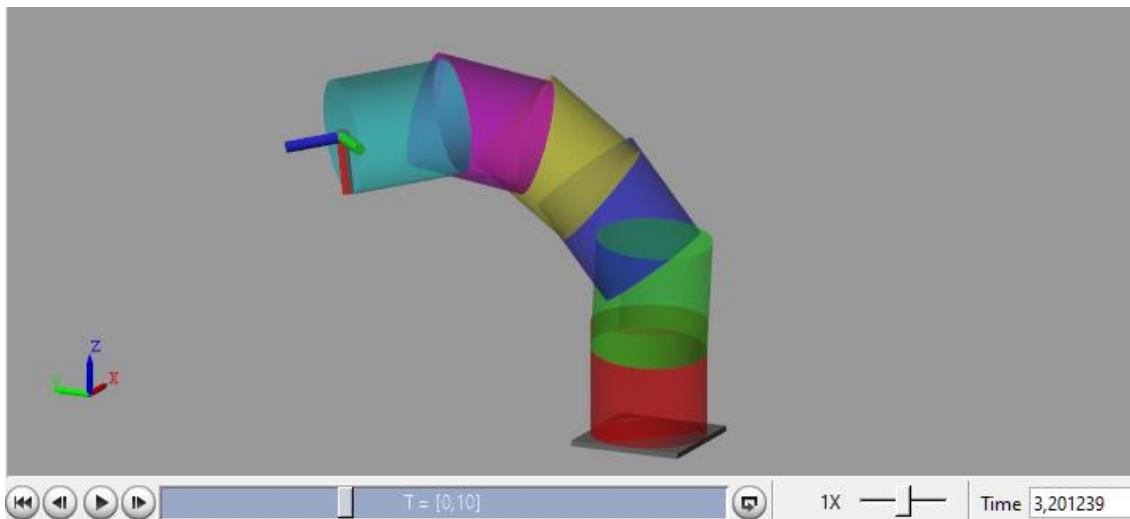


Ilustración 7.3: Simulación del modelo en Simulink.

Otra propuesta de interés es la de conectar el robot a Arduino Cloud y tener un Dashboard de control en la nube.

En cuanto a las mejoras a nivel hardware del proyecto, lo primero que se debería mejorar es la gestión de cables. Lamentablemente no se llegó a ninguna forma satisfactoria de gestión por el interior.

Gestionar el cable por el exterior genera tensiones y tirones que deberían ser evitados. Además, al usar un protocolo de comunicación que no está pensado para largas distancias, como el I2C, genera una enorme restricción en la longitud del cable.

Sería interesante cambiar el método de comunicación con los encoders para no depender de la longitud del cable o bien instalar un Arduino Nano en cada servomotor y comunicarse con el control vía WiFi.

## 8. Bibliografía

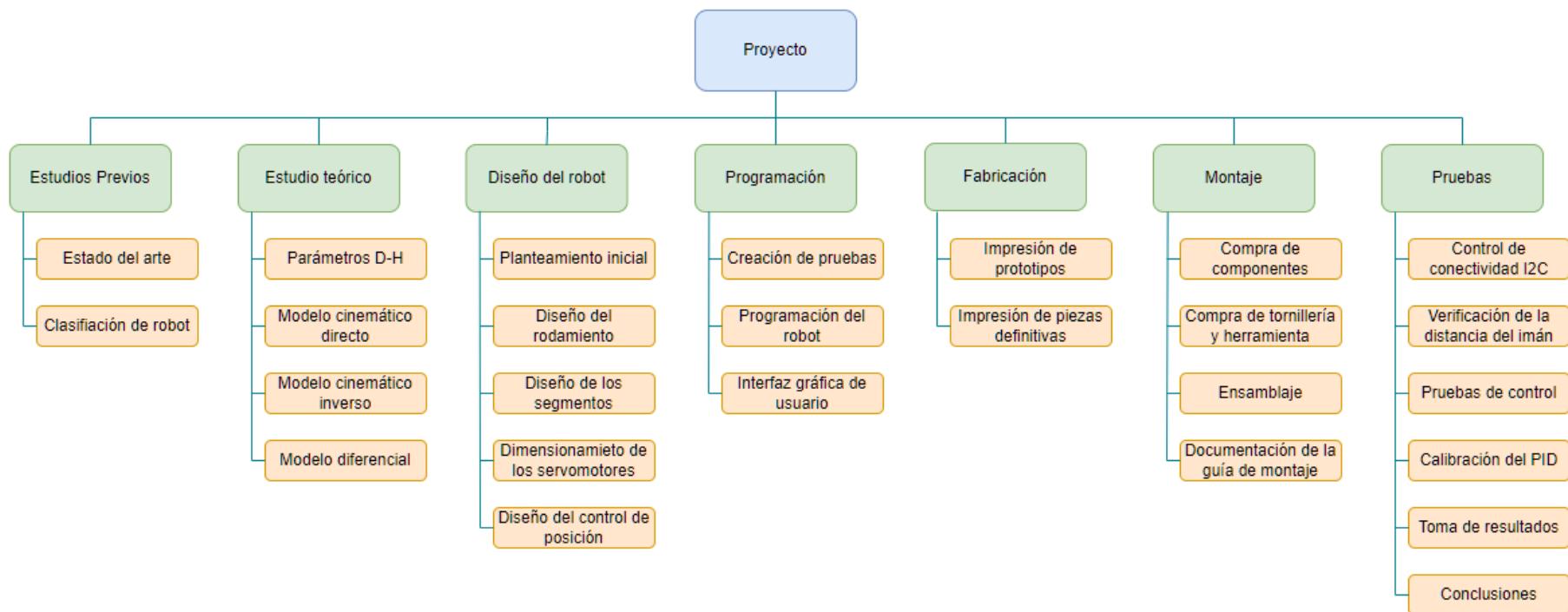
- [1] G. S. Chirikjian, "Theory and applications of hyper-redundant robotic manipulators", thesis. California Institute of Technology, 1992.
- [2] S. Hirose *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators*. Oxford: Oxford University Press, 1993.
- [3] J. Liu, Y. Wang, B. Li y S. Ma, "Current research, key performances and future development of search and rescue robots", *Frontiers Mech. Eng. China*, vol. 2, n.º 4, p. 404–416, octubre de 2007.
- [4] J. K. Hopkins, B. W. Spranklin y S. K. Gupta, "A survey of snake-inspired robot designs", *Bioinspiration & Biomimetics*, vol. 4, n.º 2, pp. 021001, enero de 2009.
- [5] H. Ikeda, N. Takanashi. Joint Assembly Movable Like A Human Arm[P]. US, 4 683 406, julio de 1987
- [6] E. Shammas, A. Wolf, H. B. Brown y H. Choset, "New joint design for three-dimensional hyper redundant robots", en 2003 IEEE/RSJ Int. Conf. Intell. Robots Syst., Las Vegas, NV, USA.
- [7] Akiyuki Brick Channel. LEGO Oblique swivel joint mechanisms. (20 de agosto de 2022). Disponible: <https://www.youtube.com/watch?v=fSMC4tfOqHM>
- [8] "MINDSTORMS EV3 Support | Everything You Need | LEGO® Education". LEGO® Education. <https://education.lego.com/en-gb/product-resources/mindstorms-ev3/downloads/user-guide/>
- [9] Allen Pan. giving snakes there legs back. (13 de agosto de 2022). Disponible: <https://www.youtube.com/watch?v=1SgGfMlbCoM>
- [10] James Bruton. Giving LEGS their SNAKES Back. (4 de octubre de 2022). Disponible: <https://www.youtube.com/watch?v=USBnAr8Bz6c>
- [11] "James Bruton is giving legs their snakes back | Arduino Blog". Arduino Blog. <https://blog.arduino.cc/2022/10/07/james-bruton-is-giving-legs-their-snakes-back/>
- [12] E. Zawidzka y M. Zawidzki, "Simulation of simple movements of Arm-Z oblique swivel joint chain manipulator", *Pomiary Automatyka Robotyka*, vol. 27, n.º 2, p. 59–67, junio de 2023.
- [13] J. Britt. Jet Pipe Arrangements For Jet Propulsion Engines[P]. US, 2933891, 1960
- [14] C. E. Johnson. Variable Vectoring Nozzle for Jet Propulsion Engines[P]. US, 3260049, 1966
- [15] G. Kopp, G. O. Madelung. Swivelable Jet Nozzle, Intended Especially for Vertical Take-Off and Short Take-Off Planes[P]. US, 3443758, 1969
- [16] D. O. Nash. Swivelable Jet Nozzle[P]. US, 3687374, 1972
- [17] H. Qi, X. Yang, Z. Wang y H. Zhu, "Development Research and Crucial Technology Analysis of Scaled 3-Bearing Swivel Duct Nozzle Rotary Drive System", IOP Conf. Series: Mater. Sci. Eng., vol. 816, pp. 012015, junio de 2020.
- [18] "Rolls-Royce LiftSystem®". Rolls-Royce: Delivering complex power solutions | Rolls-Royce. Disponible: <https://www.rolls-royce.com/products-and-services/defence/aerospace/combat-jets/rolls-royce-liftsystem.aspx>

- [19] A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, Fundamentos de Robótica, McGraw-Hill, 2007.
- [20] "MATLAB - El lenguaje del cálculo técnico". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. <https://es.mathworks.com/products/matlab.html>
- [21] "Robotics System Toolbox". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink.
- [22] M. Galicki, "A closed solution to the inverse kinematics of redundant manipulators", Mechanism Mach. Theory, vol. 26, n.º 2, pp. 221–226, enero de 1991.
- [23] W. Jacak, "A discrete kinematic model of robots in the Cartesian space", IEEE Trans. Robot. Automat., vol. 5, n.º 4, pp. 435–443, 1989.
- [24] W. Jacak, "Strategies of searching for collision-free manipulator motions: automata theory approach", Robotica, vol. 7, n.º 2, pp. 129–138, abril de 1989.
- [25] F. Fahimi, H. Ashrafiuon y C. Nataraj, "An improved inverse kinematic and velocity solution for spatial hyper-redundant robots", IEEE Trans. Robot. Automat., vol. 18, n.º 1, pp. 103–107, 2002.
- [26] M. d. G. Marcos, J. A. Tenreiro Machado y T. P. Azevedo-Perdicoúlis, "A multi-objective approach for the motion planning of redundant manipulators", Appl. Soft Comput., vol. 12, n.º 2, pp. 589–599, febrero de 2012
- [27] I. Zaplana y L. Basanez, "A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis", Mechanism Mach. Theory, vol. 121, pp. 829–843, marzo de 2018.
- [28] D. L. Pieper, The kinematics of manipulators under computer control. Stanford, Calif: Stanford University, 1968.
- [29] "Mega 2560 Rev3 | Arduino Documentation". Arduino Docs | Arduino Documentation. <https://docs.arduino.cc/hardware/mega-2560>
- [30] "Tinkercad | From mind to design in minutes". Tinkercad. <https://www.tinkercad.com/>
- [31] "UltiMaker Cura". UltiMaker. <https://ultimaker.com/software/ultimaker-cura/>
- [32] "Randy Sarafan | Tinkercad". Tinkercad. <https://www.tinkercad.com/users/5ulWoilA7JV>
- [33] HandsOn Tech – Open Source Electronics Platform. <http://handsontec.com/dataspecs/module/Rotary%20Encoder.pdf>
- [34] "Due | Arduino Documentation". Arduino Docs | Arduino Documentation. <https://docs.arduino.cc/hardware/due>
- [35] "STM32 Nucleo-64 boards". Mouser Electronics. <https://www.mouser.es/pdfdocs/enDM00105823.pdf>
- [36] Sensor Solutions | ams. [https://ams.com/documents/20143/36005/AS5600\\_DS000365\\_5-00.pdf](https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf)
- [37] Analog | Embedded processing | Semiconductor company | TI.com. <https://www.ti.com/lit/ds/symlink/tca9548a.pdf>

- [38] "Cómo usar un extensor de bus I2C TCA9548A y Arduino". Luis Llamas. <https://www.luisllamas.es/como-usar-extensor-i2c-tca9548a-y-arduino/>
- [39] "AS5600 Magnetic encoder - A practical example — Curious Scientist". Curious Scientist. <https://curiousscientist.tech/blog/as5600-magnetic-encoder-a-practical-example>
- [40] GitHub: Let's build from here · GitHub. [https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID\\_v1.h](https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h)
- [41]"Autodesk AutoCAD 2024 | Obtener precios y suscribirse al software AutoCAD". Autodesk | Software de diseño, ingeniería y construcción en 3D. <https://www.autodesk.es/products/autocad/>
- [42] "FreeCAD: Tu propio modelador paramétrico 3D". FreeCAD: Your own 3D parametric modeler. [https://www.freecad.org/index.php?lang=es\\_ES](https://www.freecad.org/index.php?lang=es_ES)
- [43] "Prusa i3 Hephestos/es - RepRap". RepRap.org. [https://reprap.org/wiki/Prusa\\_i3\\_Hephestos/es](https://reprap.org/wiki/Prusa_i3_Hephestos/es)
- [44] "Ender-3 3D Printer". creality-es. [https://www.creality.com/es/products/ender-3-3d-printer?spm=..page\\_2372271.products\\_display\\_1.1&spm\\_prev=..index.header\\_1.1](https://www.creality.com/es/products/ender-3-3d-printer?spm=..page_2372271.products_display_1.1&spm_prev=..index.header_1.1)
- [45] "Filamentos impresora 3D Filamentos Pla ABS Hips". Smart Materials 3D. <https://www.smartmaterials3d.com/>
- [46] "清远爱宝仕智能科技有限公司". 清远爱宝仕智能科技有限公司. <https://www.iboss3d.com/>
- [47] "¿Qué harías con nuestros nuevos materiales, el Tough PLA y BVOH?" BCN3D Technologies. <https://www.bcn3d.com/es/tough-pla-y-bvoh/>
- [48] "MG995 – Tower Pro". Tower Pro – Specialized . Faithful . Earnest . Quality . Innovation . Service. <https://www.towerpro.com.tw/product/mg995/>
- [49] EE Home Redirect page. [http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/stores/sg90\\_datasheet.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)
- [50] "Turnigy TGY-AN3 Standard Analog Servo 25T 3.6kg / 0.13sec / 38g". Hobbyking. [https://hobbyking.com/en\\_us/turnigy-tgy-an3-standard-analog-servo-3-6kg-0-13sec-40g.html](https://hobbyking.com/en_us/turnigy-tgy-an3-standard-analog-servo-3-6kg-0-13sec-40g.html)
- [51] AWG ¿qué es? (s.f.). pepegreen. <https://pepegreen.com/awg-que-es/>
- [52] "Uno R3 | Arduino Documentation". Arduino Docs | Arduino Documentation. <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>
- [53] "El bus I2C en Arduino". Luis Llamas. <https://www.luisllamas.es/arduino-i2c/>

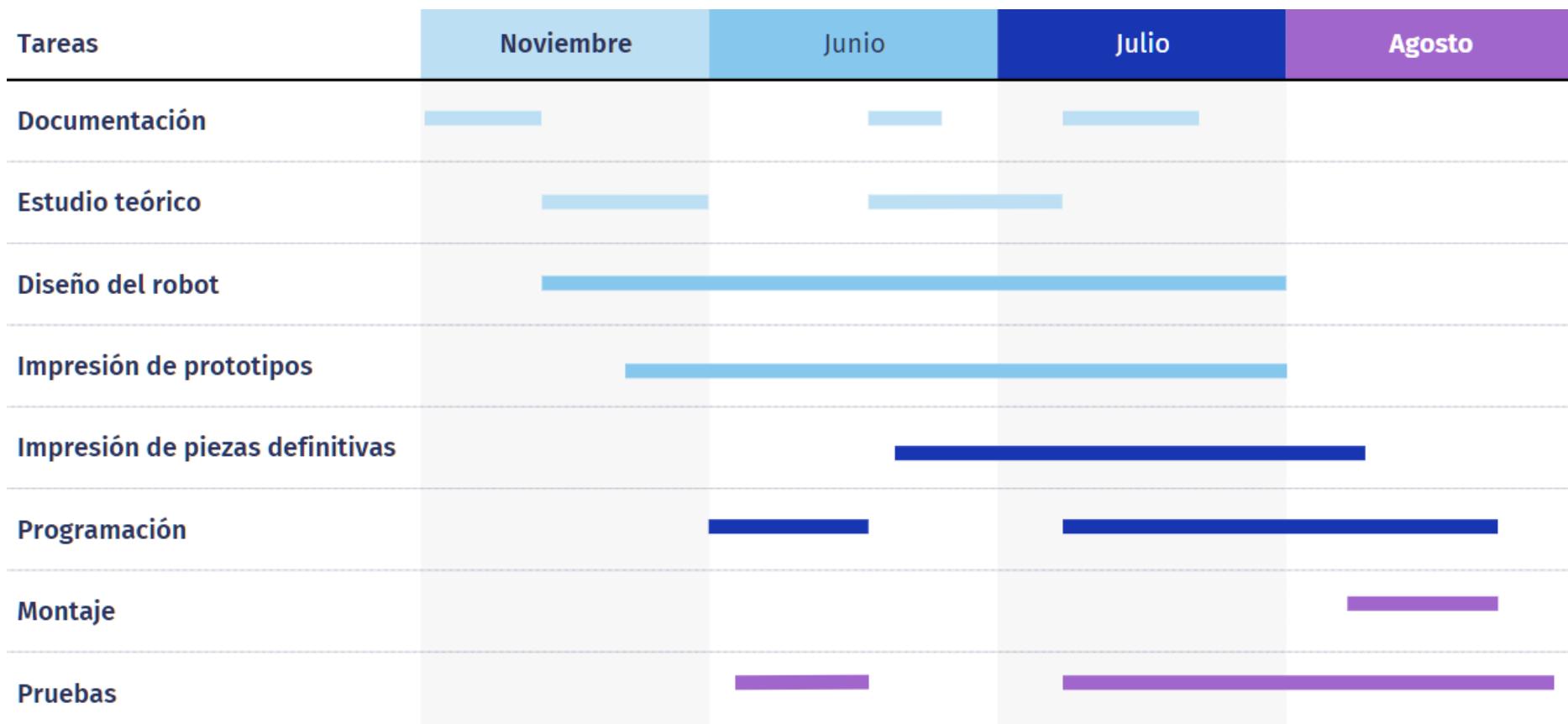


## Anexo I. Estructura de descomposición del proyecto (EDP)



## Anexo II. Planificación temporal

A continuación, se muestra un diagrama de Gantt con la planificación temporal del proyecto. Debido a las características del trabajo y al método de fabricación se han tenido que llevar varias tareas en paralelo. Fue necesaria una segunda etapa de documentación para tratar la problemática de la cinemática inversa y una tercera para tratar el control de posición en bucle cerrado.



## Anexo III. Estudio Económico

En este apartado se va a presentar el presupuesto del proyecto.

### Costes directos:

Para el cálculo del coste de personal se va a partir del salario bruto medio en España de un Ingeniero de Investigación y desarrollo que es de 31.986€ al año.

A continuación, se va a calcular las horas de trabajo anuales de un empleado.

Duración de un año	365 días
Fines de semana	-104 días
Vacaciones	-22 días
Festivos	-14 días
Total de días efectivos	225 días
Horas de trabajo diarias	8 horas
<b>TOTAL horas de trabajo anuales</b>	<b>1800 horas</b>

Tabla anexo III.1: Horas laborales al año.

Para el cálculo del coste efectivo por hora del empleado:

$$\frac{31.986\text{€}}{1800 \text{ horas}} = 17,77\text{€/hora}$$

Puesto que el TFG consta de 12 créditos que suponen 25 horas cada crédito:

$$12 \text{ créditos} \times 25 \frac{\text{horas}}{\text{crédito}} = 300 \text{ horas}$$

Por tanto, el sueldo del empleado sería:

$$300 \text{ horas} \times 17,77 \frac{\text{€}}{\text{hora}} = 5331\text{€}$$

Para el coste directo del software y equipo utilizado se va a desarrollar el cálculo de las amortizaciones, considerándolas directas. Para el software se considera una suscripción anual, mientras que para el equipo se estima el período de vida en 10 años. Para todos los elementos se va a estimar un aprovechamiento del 20%. En el caso de que el proyecto fuese desarrollado por una start-up sería:

Concepto	Coste (€)	Amortización anual (€)
AutoCad	2.342,00	468,40
Matlab	3.500,00	700,00
Office	140,40	28,08
Ordenador	1.500,00	30,00
Impresora 3D	600,00	12,00
Fuente de alimentación	99,00	1,98
<b>Coste total</b>	<b>8.181,40</b>	<b>1.240,46</b>

Tabla anexo III.2: Amortizaciones en el caso de start-up.

Puesto que es un proyecto creado para la universidad, el software anterior está bajo la licencia campus de la UPM, por lo que la tabla anterior quedaría:

<b>Concepto</b>	<b>Coste (€)</b>	<b>Amortización anual (€)</b>
<b>AutoCad</b>	0,00	0,00
<b>Matlab</b>	0,00	0,00
<b>Office</b>	0,00	0,00
<b>Ordenador</b>	1.500,00	30,00
<b>Impresora 3D</b>	600,00	12,00
<b>Fuente de alimentación</b>	99,00	1,98
<b>Coste total</b>	<b>2199,00</b>	<b>43,98</b>

Tabla anexo III.3: Amortizaciones en el caso de UPM.

A continuación, se va a desglosar el coste del material:

<b>Concepto</b>	<b>Coste unitario (€/ud)</b>	<b>Cantidad</b>	<b>Total (€)</b>
Servomotores TowerPro MG995	2,84	5	14,20
Servomotor Turnigy TGY-AN3	4,62	1	4,63
Encoder AS5600	0,64	6	3,84
Multiplexor TCA9548A	0,74	1	0,74
Arduino Mega 2560	34,71	1	34,71
Protoboard MB-102	2,14	1	2,14
Condensador electrolítico 1µF	0,15	6	0,90
Condensador electrolítico 470µF	0,35	1	0,35
Resistencia 4,7kΩ	0,04	2	0,08
Kit de crimpadora + pines	24,78	1	24,78
Cable AWG 28	4,52	1	4,52
Kit de tornillería	6,51	1	6,51
Bobina PLA Smart Materials 750g	16,49	1	16,49
Bobina PLA+ iBOSS 1Kg	19,79	2	39,58
<b>Total</b>	-	-	<b>153,47</b>
<b>Total (21% IVA)</b>	-	-	<b>185,70</b>

Tabla anexo III.4: Cuadro de precios de los materiales.

Por lo que los costes directos resultan:

<b>Concepto</b>	<b>Coste (€)</b>
Personal	5.331,00
Amortización	43,98
Material	185,70
<b>Total</b>	<b>5540,68</b>

Tabla anexo III.5: Costes directos.

### **Costes indirectos:**

Para calcular los consumos eléctricos se va a utilizar el tiempo de impresión de la tabla 4.5.

Concepto	Consumo (kWh)
Impresión de prototipos	58
Impresión de piezas finales	58
Iluminación	10
Climatización	120
Ordenador	240
Otros	4
<b>Total</b>	<b>490</b>

Tabla anexo III.6: Consumos eléctricos.

Tomando como referencia el precio medio de la electricidad en los meses de junio, julio y agosto de 2023, se obtiene un precio de 0,25€/kWh El precio del consumo eléctrico es, por tanto:

$$490\text{ kWh} \times 0,25 \frac{\text{€}}{\text{kWh}} = 122,5\text{€}$$

Los costes indirectos son:

Concepto	Coste (€)
Energía eléctrica	122,50
Teléfono e internet	508,00
Administrativos	20,00
Otros	15,00
<b>Total</b>	<b>665,50</b>

Tabla anexo III.7: Costes indirectos.

### **Coste total del proyecto:**

Sumando los costes directos e indirectos se obtiene el coste total del proyecto que es:

Concepto	Coste (€)
Costes directos	5540,68
Costes indirectos	665,50
<b>Total</b>	<b>6206,18</b>

Tabla anexo III.8: Coste total del proyecto.

El coste total del proyecto es de SEIS MIL DOSCIENTOS SEIS EUROS CON DIECIOCHO CÉNTIMOS.

## Anexo IV. Índice de figuras

Ilustración 1.1: The Active Cord Mechanism ACM III por Hirose [3].....	16
Ilustración 1.2: The Active Cord Mechanism ACM-R2 por Hirose [3].....	16
Ilustración 1.3: The Active Cord Mechanism ACM-R3 por Hirose [3].....	17
Ilustración 1.4: The Active Cord Mechanism ACM-R4 por Hirose [3].....	17
Ilustración 1.5: The Active Cord Mechanism ACM-R5 por Hirose [3].....	17
Ilustración 1.6: (a) Diseño de articulación oblicua giratoria. (b) Robot Serpiente de JPL [6]. .....	18
Ilustración 1.7: Robot Serpiente de JPL esquivando un obstáculo [6]. .....	18
Ilustración 1.8: LEGO oblique swivel joint mechanisms por Akiyuki Brick Channel [8].	19
Ilustración 1.9: Giving LEGS their SNAKES back por James Bruton [11]. .....	19
Ilustración 1.10: Estructura de tobera rotativa en tres etapas por Jack Britt [13]. .....	20
Ilustración 1.11: Estructura de tobera rotativa multietapa por Carroll E. Johnson [14].	20
Ilustración 1.12: Estructura de tobera rotativa de cuatro etapas por Gerhard Kopp [15].	21
Ilustración 1.13: Sistema de accionamiento de tobera multiengranaje 3BSD (3 Bearing Swivel Duct) por Dudley O. Nash [16]. .....	21
Ilustración 1.14: Motor principal R-97V del avión militar Yakovlev-141[17]. .....	22
Ilustración 1.15: Módulo rotativo de tres rodamientos (3BSM) del Rolls-Royce LiftSystem [12]. .....	22
Ilustración 3.1: Cadena cinemática del robot con los sistemas de coordenadas de cada elemento.....	24
Ilustración 3.2: Izquierda: visualización del segmento definido por tres parámetros $r$ , $d$ y $\xi$ . Derecha: distintas configuraciones variando $\xi$ y $s=d/r$ [12]. .....	29
Ilustración 3.3: De arriba a la izquierda abajo a la derecha: El brazo pasa de posición toroidal a posición vertical en cinco pasos y se dobla simétricamente hacia el otro lado [12]. .....	30
Ilustración 3.4: Mismo movimiento que en la ilustración 3.2 pero con 7 segmentos en lugar de 4 [12].....	30
Ilustración 3.5: Movimiento desde espiral ligeramente desenrollada a recta y de nuevo a espiral ligeramente desenrollada en el lado opuesto. El tambaleo es imperceptible como se puede apreciar con la estrechez de la caja [12]. .....	31
Ilustración 3.6: De arriba a la izquierda abajo a la derecha: El brazo se estira manteniendo el extremo nivelado y se vuelve a doblar simétricamente en el otro lado. El movimiento es perfectamente suave y no hay tambaleo [12]. .....	31
Ilustración 3.7: Mismo movimiento que en la ilustración 3.5 pero con 10 segmentos en lugar de 4 [12].....	32
Ilustración 3.8: Experimento similar con menos segmentos y otro patrón de doblado distinto [12]. .....	32
Ilustración 3.9: De arriba a la izquierda abajo a la derecha: 20 pasos en el dibujo de una elipse con el extremo del manipulador. La primera y la segunda parte del movimiento dibujan la parte superior y la inferior respectivamente [12].....	33
Ilustración 4.1: Primer prototipo. .....	35
Ilustración 4.2: Las secciones no son versátiles.....	36
Ilustración 4.3: Primer prototipo de rodamiento de rodillos oblicuo.....	37
Ilustración 4.4: Vista de la rodadura del segundo prototipo de rodamiento de rodillos oblicuo. .....	37
Ilustración 4.5: Sección de prototipo de rodamiento .....	38
Ilustración 4.6: Rodamiento definitivo. .....	38

Ilustración 4.7: Unión de la corona exterior del rodamiento.....	39
Ilustración 4.8: Unión de la corona interior del rodamiento.....	39
Ilustración 4.9: Prototipo de segmento de 30 grados. ....	40
Ilustración 4.10: Prototipo de segmento de 60 grados. ....	40
Ilustración 4.11: Unión entre rodamiento y primer prototipo de segmento de 30 grados. .....	41
Ilustración 4.12: Segmento de 30 grados definitivo. ....	42
Ilustración 4.13: Detalle de las aletas, el rebaje y los orificios en la estructura del alojamiento del servomotor en el segmento de 30 grados definitivo. ....	42
Ilustración 4.14: Segmento de 60 grados definitivo. ....	43
Ilustración 4.15: Detalle de las aletas, el rebaje y los orificios en la estructura del alojamiento del servomotor en el segmento de 60 grados definitivo. ....	43
Ilustración 4.16: Detalle de la unión del servomotor con el segmento definitivo de 30 grados. ....	44
Ilustración 4.17: Detalle de la unión del servomotor con el segmento definitivo de 60 grados. ....	44
Ilustración 4.18: Detalle del estriado del piñón. ....	45
Ilustración 4.19: Detalle de montaje con segmento de 30 grados, servomotor, rodamiento y piñón. ....	45
Ilustración 4.20: Encoder rotativo. ....	47
Ilustración 4.21: Izquierda: Encoder AS5600 [36]. Derecha: Multiplexor I2C TCA9548A [37]. ....	48
Ilustración 4.22: Pieza que sujeta el imán en su posición. ....	49
Ilustración 4.23: Pieza que sujeta la placa del encoder. ....	49
Ilustración 4.24: Detalle de montaje del encoder en un segmento de 60 grados....	50
Ilustración 4.25: Altura del piñón definitivo (derecha) frente al prototipo anterior (izquierda). ....	50
Ilustración 4.26: Barrido de direcciones I2C. ....	51
Ilustración 4.27: Barrido de salidas del multiplexor TCA9548A [37]....	51
Ilustración 4.28: Posibles salidas en la verificación de la lectura del imán. ....	52
Ilustración 4.29: Diseño definitivo de la base del robot. ....	53
Ilustración 4.30: Vista del robot. ....	53
Ilustración 4.31: Graphical User Interface desarrollada con Matlab App Designer.....	54
Ilustración 4.32: Impresora empleada en el proyecto. ....	57
Ilustración 4.33: Comparativa de las propiedades mecánicas de los principales termoplásticos para impresión 3D [45].....	58
Ilustración 4.34: Servomotor MG995 [48]. ....	59
Ilustración 4.35: Microservomotor TowerPro SG90 [49]. ....	60
Ilustración 4.36: Servomotor Turnigy TGY-AN3 [50]. ....	60
Ilustración 4.37: Valores de corriente de un servomotor. ....	60
Ilustración 4.38: Estándar AWG [51]. ....	61
Ilustración 4.39: Placa de desarrollo Arduino Mega 2560 [29]. ....	62
Ilustración 4.40: Protoboard MB-102. ....	62
Ilustración 4.41: Esquema de conexión I2C [53]. ....	63
Ilustración 4.42: Pin-out de multiplexor I2C AS5600 [38] [39]. ....	63
Ilustración 5.1: Soldadura de los pines del multiplexor I2C. ....	65
Ilustración 5.2: Soldadura y puenteado de los pines del encoder. ....	65
Ilustración 5.3: Extremos preparados de los cables de alimentación del encoder. ....	65
Ilustración 5.4: Crimpadora y cables de alimentación del encoder con pines hembra. 66	66
Ilustración 5.5: Cables de comunicación I2C del encoder con pines macho. ....	66

Ilustración 5.6: Cables de alimentación del encoder con conector DuPont de 3 pistas.	66
Ilustración 5.7: Cables de comunicación I2C del encoder con conector JST de 2 pistas.	67
Ilustración 5.8: Tubo termorretráctil en los pines macho de los cables de alimentación del encoder.	67
Ilustración 5.9: Tubo termorretráctil en los pines macho de los cables de comunicación I2C del encoder.	67
Ilustración 5.10: Conexión de los cables al encoder.	68
Ilustración 5.11: Cables de extensión del servomotor.	68
Ilustración 5.12: Tubo termorretráctil en los pines del cable de extensión del servomotor.	68
Ilustración 5.13: Clip de seguridad en la extensión del cable del servomotor.	69
Ilustración 5.14: Atornillado del servomotor de la primera articulación.	69
Ilustración 5.15: Atornillado del soporte del encoder y el servomotor de la primera articulación.	69
Ilustración 5.16: Atornillado del piñón.	70
Ilustración 5.17: Atornillado del encoder.	70
Ilustración 5.18: Gestión de los cables.	71
Ilustración 5.19: Colocación del rodamiento.	71
Ilustración 5.20: Atornillado de la corona exterior del rodamiento.	72
Ilustración 5.21: Calibración de la colocación de los imanes. Se observa una leve desviación.	72
Ilustración 5.22: Atornillado del segmento de 30° a la base.	73
Ilustración 5.23: Robot montado.	74
Ilustración 6.1: Experimento para validar el capítulo 3.5.	76
Ilustración 7.1: Propuesta de interfaz avanzada.	79
Ilustración 7.2: Modelo de Simulink.	80
Ilustración 7.3: Simulación del modelo en Simulink.	80

## **Anexo V. Índice de tablas**

Tabla 3.1: Parámetros de D-H del robot.....	25
Tabla 4.1: Tiempos medidos en ensayo de giro de 10 vueltas en ambos sentidos con y sin carga.....	46
Tabla 4.2: Parámetros generales de impresión comunes a todas las piezas.....	56
Tabla 4.3: Parámetros específicos de cada pieza.....	56
Tabla 4.4: Pesos, longitudes de filamento y tiempos de impresión por pieza.....	56
Tabla 4.5: Peso, longitud de filamento y tiempo de impresión total del proyecto.....	56
Tabla 4.6: Pares resistentes máximos de las articulaciones para el dimensionamiento de los servomotores.....	59
Tabla 4.7: Programación de las direcciones I2C del multiplexor AS5600 [39]. L=Low, H=High.....	63
Tabla anexo III.1: Horas laborales al año.	
Tabla anexo III.2: Amortizaciones en el caso de start-up.	
Tabla anexo III.3: Amortizaciones en el caso de UPM.	
Tabla anexo III.4: Cuadro de precios de los materiales.	
Tabla anexo III.5: Costes directos.	
Tabla anexo III.6: Consumos eléctricos.	
Tabla anexo III.7: Costes indirectos.	
Tabla anexo III.8: Coste total del proyecto.	

## **Anexo VI. Glosario y abreviaturas**

### **Términos:**

Crimpar: Método de unión de cables sin soldadura.

Crimpadora: Herramienta que permite crimpar. También llamada alicates de engaste.

Encoder: Codificador.

Pin: Terminal o patilla metálica de los contactos de un conector.

Protoboard: Placa de prototipos.

Pull-up: Resistencia empleada para asegurar un valor lógico concreto en un puerto colocada en el lado de tensión. Si se colocase en el lado de GND se llamaría pull-down.

Warping: Deformación producida en impresión 3D al despegarse de la cama la primera capa de impresión. Se debe a la contracción del plástico en el enfriamiento.

### **Siglas y abreviaturas:**

3BSD: 3 Bearing Swivel Duct.

ABS: Acrylonitrile butadiene styrene.

D-H: Denavit-Hartenberg.

gdl: grado de Libertad.

GND: Ground.

GUI: Graphical User Interface.

I2C: Inter-Integrated Circuit.

MCD: Modelo Cinematico Directo

MCI: Modelo Cinematico Inverso

MTH: Matriz de Trasnformación Homogénea.

PID: controlador con acción proporcional, integral y derivativa.

PLA: Polylactic acid.

PWM: Pulse-Width Modulation.

SCL: Serial Clock Line.

SDA: Serial Data Line.

STOVL: Short Take Off and Vertical Landing.

VTOL: Vertical Take Off and Landing.

## Anexo VII: Dimensionamiento de servomotores

A continuación, se muestra la tabla Excel creada para el dimensionamiento de los motores

	Servo 2	Segm. 2	Rodam. 2	Servo 3	Segm. 3	Rodam. 3	Servo 4	Segm. 4	Rodam. 4	Servo 5	Segm. 5	Rodam. 5	Servo 6	Segm. 6					
Peso (kg)		0,101	0,054	0,053	0,101	0,054	0,053	0,101	0,054	0,053	0,101	0,054	0,038	0,078					
Distancia J6 (cm)														2,3					
Par J6 (kg.cm)														0,1794					
Distancia J5													3,5	7	7	9,3			
Par J5 (kg.cm)													1,7229						
Distancia J4(cm)													3,5	7	7	10,5	14	14	16,3
Par J4(kg.cm)													4,7224						
Distancia J3(cm)													3,5	7	7	10,5	14	14	17,5
Par J3(kg.cm)													9,1779				21	21	23,3
Distancia J2(cm)													3,5	7	7	10,5	14	14	17,5
Par J2(kg.cm)													13,0678				21	21	24,5
																28	28	30,3	

## Anexo VIII. Código

Programa para barrido de direcciones I2C:

```
#include <Wire.h>
uint8_t twi_writeTo(uint8_t address, uint8_t* data, uint8_t length, uint8_t wait, uint8_t sendStop)
{
    if (!wait) return 4;
    Wire.beginTransmission(address);
    while (length) {
        Wire.write(*data++);
        length--;
    }
    return Wire.endTransmission(sendStop);
}
void scanI2CBus(byte from_addr, byte to_addr, void(*callback)(byte address, byte result) )
{
    byte rc;
    byte data = 0;
    for( byte addr = from_addr; addr <= to_addr; addr++ ) {
        rc = twi_writeTo(addr, &data, 0, 1, 0);
        callback( addr, rc );
    }
}

void scanFunc( byte addr, byte result ) {
    Serial.print("addr: ");
    Serial.print(addr,DEC);
    Serial.print( (result==0) ? " Encontrado!": "      ");
    Serial.print( (addr%4) ? "\t":"\n");
}

const byte start_address = 8;
const byte end_address = 119;

void setup()
{
    Wire.begin();
    Serial.begin(9600);
    Serial.print("Escaneando bus I2C... ");
    scanI2CBus( start_address, end_address, scanFunc );
    Serial.println("\nTerminado");
}

void loop()
{
    delay(1000);
}
```

## Programa para barrido de direcciones del multiplexor:

```
/*
 * TCA9548 I2CScanner.ino -- I2C bus scanner for Arduino
 *
 * Based on https://playground.arduino.cc/Main/I2cScanner/
 *
 */

#include "Wire.h"

#define TCAADDR 0x70

void tcaselect(uint8_t i) {
    if (i > 7) return;

    Wire.beginTransmission(TCAADDR);
    Wire.write(1 << i);
    Wire.endTransmission();
}

void setup()
{
    while (!Serial);
    delay(1000);

    Wire.begin();

    Serial.begin(115200);
    Serial.println("\nTCA escaner listo");

    for (uint8_t t=0; t<8; t++) {
        tcaselect(t);
        Serial.print(" Escaneando salida "); Serial.println(t);

        for (uint8_t addr = 0; addr<=127; addr++) {
            if (addr == TCAADDR) continue;

            Wire.beginTransmission(addr);
            if (!Wire.endTransmission()) {
                Serial.print(" - Encontrado I2C 0x"); Serial.println(addr,HEX);
            }
        }
        Serial.println("Finalizado");
    }
}

void loop()
{}
```

## Programa para comprobar la distancia del imán:

```
#include <Wire.h> //This is for i2C

//===== TCA9548 Multiplexer i2C
#define tcaAddress 0x70
enum muxUsersIndex {magEnc, muxFree1, muxFree2, muxFree3, muxFree4,
muxFree5, muxFree6, muxFree7};

//===== AS5600 magnetic encoder
#include <AS5600.h>
AS5600 magnEncoder;
const float magnEncStepsDegrees = 4096/360;

//-----
//Magnetic sensor things
int magnetStatus = 0; //value of the status register (MD, ML, MH)

void setup()
{
    Serial.begin(115200); //start serial - tip: don't use serial if you don't need it (speed
    considerations)
    Wire.begin(); //start i2C
    Wire.setClock(800000L); //fast clock

    checkMagnetPresence(magEnc); //check the magnet (blocks until magnet is found)
}

void loop()
{
}

void checkMagnetPresence(int magnEnIndex)
{
    //This function runs in the setup() and it locks the MCU until the magnet is not
    positioned properly

    while ((magnetStatus & 32) != 32) //while the magnet is not adjusted to the proper
    distance - 32: MD = 1
    {
        magnetStatus = 0; //reset reading

        tcaselect(magnEnIndex);
        Wire.beginTransmission(0x36); //connect to the sensor
        Wire.write(0x0B); //register map: Status: MD ML MH
        Wire.endTransmission(); //end transmission
        Wire.requestFrom(0x36, 1); //request from the sensor

        while (Wire.available() == 0); //wait until it becomes available
        magnetStatus = Wire.read(); //Reading the data after the request
    }
}
```

```

    Serial.print("Magnet status: ");
    Serial.println(magnetStatus, BIN); //print it in binary so you can compare it to the
table
}

//Status register output: 0 0 MD ML MH 0 0 0
//MH: Too strong magnet - 100111 - DEC: 39
//ML: Too weak magnet - 10111 - DEC: 23
//MD: OK magnet - 110111 - DEC: 55

//Serial.println("Magnet found!");
//delay(1000);
}

void tcaselect(uint8_t i){

if (i > 7) return;
Wire.beginTransmission(tcaAddress);
Wire.write(1 << i);
Wire.endTransmission();
delay(10);
}

//================================================================ Magnetic encoder
float magnEncGetPosition(int magnEncIndex)
{
    tcaselect(magnEncIndex);
    return magnEncoder.readAngle();
}

float magnEncGetPosDegrees(int magnEncIndex)
{
    tcaselect(magnEncIndex);
    return magnEncoder.readAngle()/magnEncStepsDegrees;
}

```

## Programa del robot:

```
===== Required libraries
#include <Wire.h>
#include <Servo.h>
#include <PID_v1.h>

===== TCA9548 Multiplexer i2C
#define tcaAddress 0x70
enum muxUsersIndex {magEnc1, magEnc2, magEnc3, magEnc4, magEnc5,
magEnc6, muxFree1, muxFree2};

===== AS5600 magnetic encoder
#include <AS5600.h>
AS5600 magnEncoder;
const float magnEncStepsDegrees = 4096/360;

===== Servo and PID
Servo myservo1, myservo2, myservo3, myservo4, myservo5, myservo6;
float pos0=180;                                //initial position
float pos1=180, pos2=180, pos3=180, pos4=180, pos5=180, pos6=180;

double kp = 2 , ki = 0.5 , kd = 0.005;           // modify for optimal performance

double input1 = 0, output1 = 0, setpoint1 = 0;
double input2 = 0, output2 = 0, setpoint2 = 0;
double input3 = 0, output3 = 0, setpoint3 = 0;
double input4 = 0, output4 = 0, setpoint4 = 0;
double input5 = 0, output5 = 0, setpoint5 = 0;
double input6 = 0, output6 = 0, setpoint6 = 0;

PID myPID1(&input1, &output1, &setpoint1, kp, ki, kd, DIRECT);
PID myPID2(&input2, &output2, &setpoint2, kp, ki, kd, DIRECT);
PID myPID3(&input3, &output3, &setpoint3, kp, ki, kd, DIRECT);
PID myPID4(&input4, &output4, &setpoint4, kp, ki, kd, DIRECT);
PID myPID5(&input5, &output5, &setpoint5, kp, ki, kd, DIRECT);
PID myPID6(&input6, &output6, &setpoint6, 0.5, ki, kd, DIRECT);

===== Serial communication
String StrQ1, StrQ2, StrQ3, StrQ4, StrQ5, StrQ6;

void setup()
{
    Wire.begin();
    Serial.begin(115200);                      // Define baud rate

    myservo1.attach(2);
    myservo2.attach(3);
    myservo3.attach(4);
    myservo4.attach(5);
    myservo5.attach(6);
```

```

myservo6.attach(7);

checkpos1(pos0, magEnc1);      //initial position
checkpos2(pos0, magEnc2);
checkpos1(pos0, magEnc1);
checkpos2(pos0, magEnc2);
checkpos1(pos0, magEnc1);
checkpos2(pos0, magEnc2);

myPID1.SetMode(AUTOMATIC);    //set PID in Auto mode
myPID1.SetSampleTime(1);      // refresh rate of PID controller
myPID1.SetOutputLimits(-90, 90); // this is the MAX PWM value to move motor, here
change in value reflect change in speed of motor.

myPID2.SetMode(AUTOMATIC);
myPID2.SetSampleTime(1);
myPID2.SetOutputLimits(-90, 90);

myPID3.SetMode(AUTOMATIC);
myPID3.SetSampleTime(1);
myPID3.SetOutputLimits(-90, 90);

myPID4.SetMode(AUTOMATIC);
myPID4.SetSampleTime(1);
myPID4.SetOutputLimits(-90, 90);

myPID5.SetMode(AUTOMATIC);
myPID5.SetSampleTime(1);
myPID5.SetOutputLimits(-90, 90);

myPID6.SetMode(AUTOMATIC);
myPID6.SetSampleTime(1
myPID6.SetOutputLimits(-90, 90);

delay(1500);
}

void loop()
{
//Lectura de datos por SERIAL
if (Serial.available())
{
StrQ1 = Serial.readStringUntil(',');
StrQ2 = Serial.readStringUntil(',');
StrQ3 = Serial.readStringUntil(',');
StrQ4 = Serial.readStringUntil(',');
StrQ5 = Serial.readStringUntil(',');
StrQ6 = Serial.readStringUntil('\n');

pos1 = StrQ1.toInt()+180;
}

```

```

pos2 = StrQ2.toInt()+180;
pos3 = StrQ3.toInt()+180;
pos4 = StrQ4.toInt()+180;
pos5 = StrQ5.toInt()+180;
pos6 = StrQ6.toInt()+180;
}

checkpos1(pos1, magEnc1);
checkpos2(pos2, magEnc2);
checkpos3(pos3, magEnc3);
checkpos4(pos4, magEnc4);
checkpos5(pos5, magEnc5);
checkpos6(pos6, magEnc6);

}

//================================================================= Closed loop position control

void checkpos1(int p, int magEnclIndex)
{
    tcaselect(magEnclIndex);
    setpoint1=p;
    input1 = (magnEncGetPosDegrees(magEnclIndex));
    myPID1.Compute();
    myservo1.write(map (output1, -90, 90, 180, 0));
    //delay(50);
}

void checkpos2(int p, int magEnclIndex)
{
    tcaselect(magEnclIndex);
    setpoint2=p;
    input2 = (magnEncGetPosDegrees(magEnclIndex));
    myPID2.Compute();
    myservo2.write(map (output2, -90, 90, 180, 0));
    //delay(50);
}

void checkpos3(int p, int magEnclIndex)
{
    tcaselect(magEnclIndex);
    setpoint3=p;
    input3 = (magnEncGetPosDegrees(magEnclIndex));
    myPID3.Compute();
    myservo3.write(map (output3, -90, 90, 180, 0));
    //delay(50);
}

void checkpos4(int p, int magEnclIndex)
{
    tcaselect(magEnclIndex);
}

```

```

setpoint4=p;
input4 = (magnEncGetPosDegrees(magEnclIndex));
myPID4.Compute();
myservo4.write(map (output4, -90, 90, 180, 0));
//delay(50);
}

void checkpos5(int p, int magEnclIndex)
{
tcaselect(magEnclIndex);
setpoint5=p;
input5 = (magnEncGetPosDegrees(magEnclIndex));
myPID5.Compute();
myservo5.write(map (output5, -90, 90, 180, 0));
//delay(50);
}

void checkpos6(int p, int magEnclIndex)
{
tcaselect(magEnclIndex);
setpoint6=p;
input6 = (magnEncGetPosDegrees(magEnclIndex));
myPID6.Compute();
myservo6.write(map (output6, -90, 90, 180, 0));
//delay(50);
}

//================================================================= Mux i2C channel selection
void tcaselect(uint8_t i)
{
if (i > 7) return;
Wire.beginTransmission(tcaAddress);
Wire.write(1 << i);
Wire.endTransmission();
//delay(1);
}

//================================================================= Magnetic encoder
int magnEncGetPosition(int magEnclIndex)
{
tcaselect(magEnclIndex);
return magnEncoder.readAngle();
}

float magnEncGetPosDegrees(int magnEnclIndex)
{
tcaselect(magnEnclIndex);
return magnEncoder.readAngle()/magnEncStepsDegrees;
}

```

