



## GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela de Ingeniería de Fuenlabrada

Curso académico 2022-2023

### Trabajo Fin de Grado

Brazo robótico de bajo coste para la docencia universitaria

**Tutor:** Julio Vega Pérez

**Autor:** Vidal Pérez Bohoyo



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-NC-SA International License (Creative Commons AttributionNonCommercial-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato; y *(b) adaptar*: remezclar, transformar y crear a partir del material. El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante.
- *No comercial.* Usted no puede hacer uso del material con propósitos comerciales.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

# Agradecimientos

---

En primer lugar, quiero agradecer a mi profesor guía, Julio Vega, por su apoyo y orientación a lo largo de este trabajo. Sus conocimientos y sugerencias fueron de gran ayuda para enfocarlo adecuadamente y alcanzar resultados significativos.

A mi familia, gracias por su apoyo incondicional y por creer en mí en cada paso de mi formación académica. En especial a mi primo Pablo, que desde pequeño me ha introducido en el mundo de la ingeniería y siempre ha estado ahí cuando más lo he necesitado.

A mi novia, por compartir este viaje conmigo y haberme levantado tras cada caída. Su compañía y paciencia me ha hecho lograr lo que en un momento pensé que era imposible. Gracias por ser y haber sido un pilar fundamental en mi vida.

Agradezco también a todas las personas que participaron en la investigación, brindando su tiempo y conocimientos. Como ha sido el caso de Julio Salvador Lora.

Por último, quiero agradecer a todas las personas que, de alguna manera, me han apoyado durante este proceso, incluso si no están mencionadas específicamente. Sus palabras de apoyo y sus consejos me han motivado aún más en este proyecto.

Este logro es el resultado de un esfuerzo conjunto, y agradezco sinceramente a cada persona que ha sido parte de él. Espero que este trabajo contribuya al desarrollo de la robótica y que pueda inspirar futuras investigaciones.

*A mi familia y a mi novia.*

Madrid, 11 de Octubre de 2023

*Vidal Pérez Bohoyo*

# Resumen

---

La robótica —disciplina enfocada en el diseño, construcción y programación de robots— se compone de diversas ramas entre las cuales interesa destacar la robótica educativa, caracterizada por ofrecer robots útiles para el aprendizaje, y la robótica de bajo coste, campo de estudio que busca superar las barreras económicas que presentan los robots tradicionales para garantizar que todo el mundo pueda acceder a esta tecnología, ambas ramas estrechamente relacionadas con este proyecto.

Los robots usados comúnmente en las universidades son realmente costosos, por lo que es complicado adquirir una gran cantidad de ellos, limitando su disponibilidad. Es por esto que en este trabajo se ha desarrollado una solución para este problema. Concretamente, se ha desarrollado un brazo robótico industrial de bajo coste que puede ser fabricado mediante cualquier impresora 3D convencional.

El modelo 3D se ha desarrollado con la herramienta de diseño 3D FreeCAD, por ser de código abierto. Por otro lado, el hardware empleado es fácilmente adquirible a través de internet, lo que ha permitido desarrollar el robot de manera accesible y económica.

El robot final ha sido integrado en ROS 2 Humble, un *middleware* de código abierto ampliamente utilizado en aplicaciones robóticas. Esta integración ha proporcionado una interfaz estandarizada y eficiente para que cualquier aplicación desarrollada en este ecosistema pueda hacer uso de él. Además, ha sido configurado para utilizarse en el *framework* MoveIt 2, lo que facilita aún más su uso.

Con la combinación de estas herramientas y tecnologías, se ha logrado desarrollar un robot altamente funcional y resistente, preparado para utilizarse en un entorno académico formando a nuevas generaciones. El resultado obtenido representa un esfuerzo de investigación y desarrollo significativo, que abre la puerta a futuras mejoras y avances en el campo de la robótica.

Finalmente, se han realizado numerosas pruebas que evalúan los distintos aspectos técnicos que definen el rendimiento y características del robot final.

# Abstract

---

Robotics —discipline focused on the design, construction and programming of robots— is made up of various branches among which it is interesting to highlight educational robotics, characterized by offer useful robots for learning, and low-cost robotics, a field of study that seeks to overcome barriers economics presented by traditional robots to ensure that everyone can access this technology, both branches closely related to this project.

The robots commonly used in universities are really expensive, so It is difficult to acquire a large quantity of them, limiting their availability. That is why In this work, a solution for this problem has been developed. Specifically, it has developed a low-cost industrial robotic arm that can be manufactured using any conventional 3D printer.

The 3D model has been developed with the FreeCAD 3D design tool, as it is open source. By On the other hand, the hardware used is easily available over the Internet, which has allowed develop the robot in an accessible and economical way.

The final robot has been integrated into ROS 2 Humble, a widely open source *middleware* used in robotic applications. This integration has provided an interface standardized and efficient so that any application developed in this ecosystem can make use of it. Besides, has been configured to be used in the *framework* MoveIt 2, making it even easier to use.

With the combination of these tools and technologies, it has been possible to develop a highly efficient robot. functional and resistant, prepared to be used in an academic environment training new generations. He The result obtained represents a significant research and development effort, which opens the door to future improvements and advances in the field of robotics.

Finally, numerous tests have been carried out that evaluate the different technical aspects that They define the performance and characteristics of the final robot.

# Acrónimos

---

**DOF** *Degrees Of Freedom*

**STEM** *Science, Technology, Engineering and Mathematics*

**DIY** *Do It by Yourself*

**CAD** *Computer-Aided Design*

**FDM** *Fused Deposition Modeling*

**ROS** *Robot Operating System*

**CNC** *Control Numérico por Computadora*

**PWM** *Pulse Width Modulation*

**DDS** *Data Distribution Service*

**SCARA** *Selective Compilant Assembly Robot Arm*

**URDF** *Unified Robot Description Format*

**XML** *eXtensible Markup Language*

**UART** *Universal Asynchronous Receiver/Transmitter*

**STL** *Standard Tessellation Language*

# Índice general

---

<b>1. Introducción</b>	<b>1</b>
1.1. Robótica . . . . .	1
1.2. Robótica de servicio . . . . .	3
1.3. Robótica industrial . . . . .	6
1.4. Robótica educativa . . . . .	9
1.4.1. Robótica en la educación secundaria . . . . .	9
1.4.2. Robótica en la universidad . . . . .	9
1.5. Robótica de bajo coste . . . . .	10
<b>2. Estado del arte</b>	<b>12</b>
<b>3. Objetivos</b>	<b>19</b>
3.1. Descripción del problema . . . . .	19
3.2. Requisitos . . . . .	20
3.3. Metodología . . . . .	21
3.4. Plan de trabajo . . . . .	21
<b>4. Plataforma de desarrollo</b>	<b>23</b>
4.1. Software . . . . .	23
4.1.1. Python . . . . .	23
4.1.2. Grbl . . . . .	24
4.1.3. G-code . . . . .	24
4.1.4. FreeCAD . . . . .	25
4.1.5. ROS 2 . . . . .	26
4.1.6. MoveIt! . . . . .	27
4.2. Hardware . . . . .	28
4.2.1. MKS DLC32 . . . . .	28
4.2.2. Motores Nema 17 . . . . .	29
4.2.3. Controlador TMC2209 . . . . .	30

4.2.4. Fuente de alimentación genérica . . . . .	31
<b>5. Desarrollo hardware del manipulador</b>	<b>32</b>
5.1. Geometría del manipulador . . . . .	32
5.2. Modelo alámbrico del manipulador . . . . .	34
5.3. Bocetos . . . . .	37
5.4. Elección de componentes electromecánicos . . . . .	39
5.4.1. Motores . . . . .	39
5.4.2. Reductora . . . . .	41
5.4.3. Controladores . . . . .	42
5.4.4. Placa base CNC . . . . .	43
5.4.5. Fuente de alimentación . . . . .	44
5.4.6. Rodamientos . . . . .	44
5.4.7. Finales de carrera . . . . .	45
5.4.8. Electroimán . . . . .	46
5.5. Diseño CAD . . . . .	47
5.5.1. Base principal . . . . .	47
5.5.2. Base de los motores . . . . .	49
5.5.3. Paralelogramos . . . . .	50
5.5.4. Elemento terminal . . . . .	52
5.5.5. Herramienta electroimán . . . . .	52
5.6. Impresión y montaje . . . . .	53
<b>6. Desarrollo software del manipulador</b>	<b>59</b>
6.1. Control de los actuadores . . . . .	59
6.1.1. Grbl como <i>firmware</i> del robot . . . . .	59
6.1.2. Comunicación con Grbl . . . . .	60
6.1.3. Configuración de Grbl para su uso en robótica . . . . .	62
6.2. Integración con ROS 2 . . . . .	65
6.2.1. Descripción del robot . . . . .	65
6.2.2. Integración con MoveIt 2 . . . . .	69
6.2.3. Driver para ROS . . . . .	72
6.3. Pruebas . . . . .	75
6.3.1. Capacidad de carga . . . . .	75
6.3.2. Velocidad máxima . . . . .	76
6.3.3. Consumo eléctrico . . . . .	77

<b>7. Conclusiones</b>	<b>79</b>
7.1. Objetivos cumplidos . . . . .	79
7.2. Competencias adquiridas . . . . .	80
7.3. Valoración final y líneas futuras . . . . .	81
<b>Bibliografía</b>	<b>93</b>

# Índice de figuras

---

1.1.	Robot Unimate . . . . .	2
1.2.	Robot Shakey . . . . .	3
1.3.	Robots de campo . . . . .	4
1.4.	Robots de limpieza . . . . .	4
1.5.	Robots de entretenimiento . . . . .	5
1.6.	Robots en el campo de la salud . . . . .	5
1.7.	Robots de logística . . . . .	6
1.8.	Fanuc SR-3iA . . . . .	7
1.9.	Robot articulados . . . . .	7
1.10.	Robots basados en paralelogramos . . . . .	8
1.11.	Robots cartesianos . . . . .	8
1.12.	Robots en institutos . . . . .	9
1.13.	Robots usados en universidades . . . . .	10
1.14.	Robots de bajo coste . . . . .	11
2.1.	Robot HydraX . . . . .	12
2.2.	Robot UIArm . . . . .	14
2.3.	Robot Niryo One . . . . .	15
2.4.	Robots formados a partir de paralelogramos . . . . .	17
4.1.	Logo de Grbl . . . . .	24
4.2.	Logo de FreeCAD . . . . .	25
4.3.	Logotipo de ROS 2 Humble . . . . .	26
4.4.	Plugin de MoveIt para el visualizador de ROS, Rviz2 (Franka Emika Robot) . . . . .	27
4.5.	Esquema de la arquitectura hardware . . . . .	28
4.6.	Placa base MakerBase DLC32 . . . . .	29
4.7.	Motores Nema 17 de 2.1A . . . . .	29
4.8.	Controlador TMC2209 . . . . .	30

4.9.	Métodos usados para alimentar el robot . . . . .	31
5.1.	Espacio tridimensional . . . . .	32
5.2.	Tipos de articulaciones más usadas en robótica . . . . .	33
5.3.	Pinza paralela con 1 grado de libertad . . . . .	34
5.4.	Principio fundamental del brazo MeArm . . . . .	35
5.5.	Modelo alámbrico de G-Arm . . . . .	36
5.6.	Bocetos realizados . . . . .	37
5.7.	Prueba de concepto . . . . .	38
5.8.	Dinámica del manipulador completamente extendido . . . . .	39
5.9.	Dimensiones de los motores Nema17 . . . . .	40
5.10.	Ejemplo de correa GT2 . . . . .	41
5.11.	Controladores existentes en el mercado . . . . .	42
5.12.	Placas base candidatas . . . . .	43
5.13.	Rodamientos de tipo brida . . . . .	45
5.14.	Final de carrera económico . . . . .	45
5.15.	Electroimán D20H15 3KG . . . . .	46
5.16.	Base principal . . . . .	47
5.17.	Base de los motores . . . . .	49
5.18.	Conjunto de los paralelogramos . . . . .	50
5.19.	Contorno de la polea de 100 dientes . . . . .	51
5.20.	Ambos lados de la palanca de la articulación 3 . . . . .	51
5.21.	Elemento terminal . . . . .	52
5.22.	Vistas del elemento terminal . . . . .	53
5.23.	Herramienta electroimán . . . . .	54
5.24.	Ender-3 Pro V1 2017 . . . . .	55
5.25.	Ensamble CAD completo del robot final . . . . .	57
5.26.	Montaje real del robot . . . . .	58
6.1.	Terminal gráfica de Cutecom al conectarse a una placa con Grbl . . . . .	60
6.2.	Interfaz de UniversalGcodeSender . . . . .	62
6.3.	Elementos del formato URDF . . . . .	66
6.4.	Cadenas cinemáticas . . . . .	67
6.5.	Ventana de RViz visualizado el robot creado . . . . .	69
6.6.	Rviz al lanzar el <i>demo.launch.py</i> . . . . .	71
6.7.	Planificando una trayectoria simple . . . . .	71
6.8.	Relación de <i>move_group</i> con el resto del <i>Firmware</i> . . . . .	72

6.9. Diagrama de actividades del driver . . . . .	74
6.10. Fotos extraídas del vídeo de las pruebas de carga . . . . .	75
6.11. Fotos extraídas del vídeo de las pruebas de velocidad . . . . .	77
6.12. Fotos extraídas del vídeo de las pruebas de consumo . . . . .	77
7.1. Resultado de ejecutar <i>tree -d -L 1</i> en la carpeta <i>workspace</i> . . . . .	84
7.2. Activación de la opción Show Trail de un link en Rviz . . . . .	88
7.3. Trayectorias realizadas . . . . .	88
7.4. Herramienta porta lápices . . . . .	89
7.5. Prueba en el robot real . . . . .	90
7.6. Puntos del ejercicio . . . . .	92

# Listado de códigos

---

4.1. Programa para realizar una trayectoria cuadrada . . . . .	25
6.1. API de la clase <i>Grbl</i> . . . . .	61
6.2. Campos del tipo de mensaje <i>sensor_msgs/msg/JointState</i> . . . . .	73
7.1. Uso básico de PyMoveIt2 para moverse a un punto . . . . .	87
7.2. Uso básico de PyMoveIt2 para moverse a un punto . . . . .	91

## Listado de ecuaciones

# Índice de cuadros

---

4.1. Especificaciones técnicas de los motores utilizados . . . . .	30
4.2. Especificaciones técnicas del TMC2209 . . . . .	31
5.1. Parámetros del modelo alámbrico de G-Arm . . . . .	36
5.2. Comparación de especificaciones de los controladores existentes . . . . .	42
5.3. Componentes hardware necesarios . . . . .	54
5.4. Tornillos necesarios . . . . .	55
5.5. Tuercas necesarias . . . . .	55
5.6. Arandelas necesarias . . . . .	56
5.7. Varillas roscadas necesarias . . . . .	56
5.8. Piezas necesarias . . . . .	56
6.1. Comandos utilizados en la realización del proyecto . . . . .	61
6.2. Parámetros Grbl usados en este trabajo . . . . .	64
6.3. Resultados de los diferentes resultados en la prueba de carga . . . . .	76
6.4. Consumos eléctricos en distintos escenarios . . . . .	78

---

# Capítulo 1

## Introducción

---

*El éxito es la capacidad de ir de fracaso en fracaso sin perder el entusiasmo.*

Winston Churchill

En la actualidad, los robots desempeñan un papel fundamental en el ámbito educativo, tanto en institutos como en universidades. Su incorporación ha demostrado ser muy beneficiosa para el desarrollo de habilidades STEM y la enseñanza del pensamiento computacional entre los estudiantes. No obstante, es esencial que estos robots no tengan un coste elevado, permitiendo así que las escuelas puedan hacerse con una cantidad significativa de ellos. De esta forma, cada alumno tendría la oportunidad de disfrutar de un aprendizaje enriquecedor e individualizado.

En este capítulo introductorio se abordará el contexto de la robótica, explorando sus diversas disciplinas y aplicaciones. Esto nos ayudará a comprender mejor el ámbito en el cual se enmarca este trabajo, proporcionando una base sólida para comprender los conceptos y desarrollos que se presentarán a lo largo del documento.

### 1.1. Robótica

La robótica es la ciencia dedicada al diseño, construcción y programación de robots. Se entiende por robot a una máquina programable capaz de llevar a cabo tareas de forma autónoma, usando sensores para percibir su entorno y actuadores para interactuar con él.

Ya en la antigüedad se encuentran referencias a autómatas y dispositivos mecánicos que tratan de imitar acciones humanas. Sin embargo, no es hasta mediados del siglo XX cuando surge el concepto de robot tal y como lo conocemos hoy en día. En 1954, George Devol y Joseph Engelberger desarrollaron el primer robot industrial

programable llamado *Unimate* (Figura 1.1), el cual fue utilizado en la fábrica General Motors para realizar tareas de soldadura en la línea de producción de automóviles. Este hito marcó el comienzo de la automatización industrial y sentó las bases para el desarrollo futuro de robots industriales.

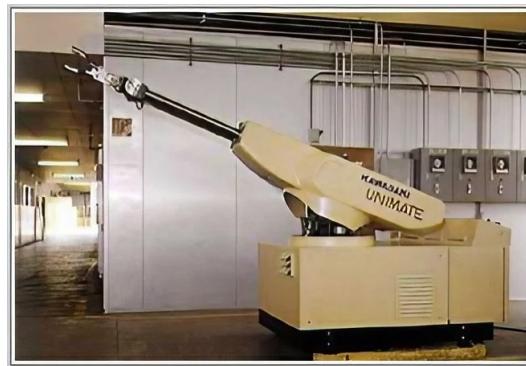


Figura 1.1: Robot Unimate

En las décadas posteriores se produjeron numerosos avances en robótica; se desarrollaron robots cada vez más sofisticados que cubrían una amplia gama de tareas más allá de la automatización industrial. Un claro ejemplo de ello fue el robot *Shakey* (Figura 1.2), desarrollado por el laboratorio de investigación de la Universidad de Stanford en 1966. Se trataba de un robot cuyo único propósito era navegar autónomamente en una sala con obstáculos gracias al uso de sensores y algoritmos de planificación.

En la década de 1970, el uso de robots industriales se había disparado en todo el mundo. Estos robots, conocidos como robots manipuladores, fueron utilizados para realizar todo tipo de tareas repetitivas y peligrosas en fábricas.

En las últimas décadas, la robótica ha seguido avanzando a pasos agigantados. Los avances en la capacidad de cómputo, el aprendizaje automático y la visión artificial han permitido el desarrollo de robots cada vez más inteligentes.

En la actualidad, la robótica está presente en numerosos sectores, abarcando una gran variedad de aplicaciones. Esta versatilidad ha llevado al surgimiento de diversos grupos y categorías que nos permiten clasificar las diferentes tipos de robots. Un claro ejemplo de ello es la robótica de servicio, que integra robots en aplicaciones cotidianas.

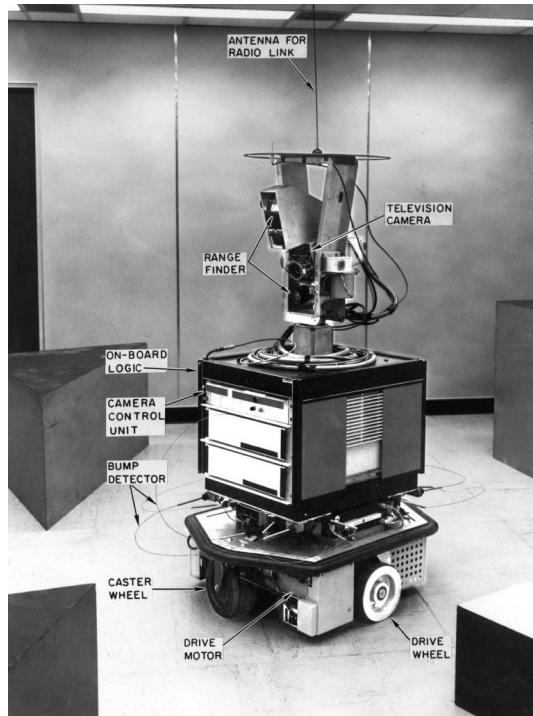


Figura 1.2: Robot Shakey

## 1.2. Robótica de servicio

Un robot de servicio es un tipo de robot diseñado para realizar tareas en beneficio de los seres humanos. Estos robots están destinados a interactuar directamente con las personas, por lo que están equipados con gran variedad de sensores, actuadores y sistemas de inteligencia artificial, que les permiten percibir y comprender el entorno que los rodea. En función de su ámbito de uso, pueden llegar a realizar una amplia gama de tareas, como limpieza y mantenimiento del hogar, asistencia en la intervención médica, entrega de alimentos y productos, cuidado de personas mayores, entre otros. A continuación, veremos algunos ejemplos de estos robots.

### Robots de campo

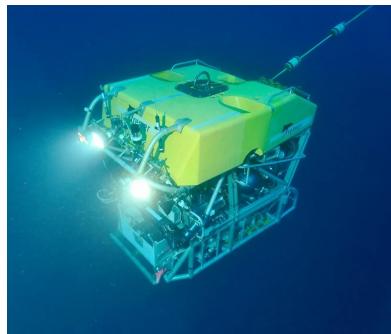
Se considera robot de campo a un robot diseñado para su uso en exteriores; lo que significa que trabajará bajo unas duras condiciones. Se trata de un sector heterogéneo en el cual se engloban los robots espaciales, agrícolas, búsqueda y rescate, inspección de instalaciones, minería, submarinos y conducción autónoma. En la Figura 1.3 se muestran algunos ejemplos.



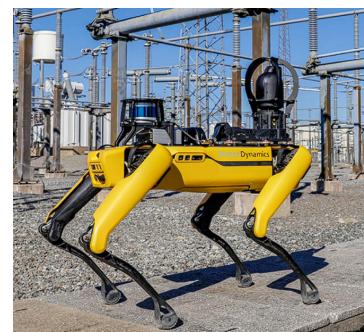
(a) NASA Opportunity



(b) Tractor autónomo



(c) ROV Victor 6000



(d) Boston Dynamics Spot

Figura 1.3: Robots de campo

### Robots de limpieza

Son aquellos robots creados para eliminar la suciedad en hogares y empresas. En función de sus características, pueden ser usados para aspirar y fregar el suelo, o incluso, para limpiar los cristales exteriores de los edificios. Se trata de una tecnología asentada y robusta que a día de hoy cuenta con más de 20 años en el mercado. Pese a esto, se pueden encontrar diferentes categorías de robot en función de las necesidades. En la actualidad, integran cada vez más sensores para realizar una limpieza más eficaz y en entornos más imprevisibles (cables, animales, objetos tirados, etc). En la Figura 1.4 se muestran algunos de los más usados.



(a) Roomba J7+



(b) Roomba Braava



(c) Hobot 388

Figura 1.4: Robots de limpieza

## Robots de entretenimiento

Los robots de entretenimiento son máquinas diseñadas para proporcionar diversión y compañía a las personas. Están equipados con capacidades y características específicas que los hacen adecuados para interactuar con las personas. Tienen una apariencia amigable y agradable, como los mostrados en la Figura 1.5. Están pensados para evitar el efecto del *Valle Inquietante*. Esta teoría sugiere que a medida que los robots se vuelven más humanos en apariencia y comportamiento, la respuesta emocional de las personas hacia ellos es más negativa, antes de volver a ser positiva a medida que se vuelven prácticamente indistinguibles de los seres humanos reales.



(a) Aibo Sony



(b) Foca Paro

Figura 1.5: Robots de entretenimiento

## Robots en salud

Los robots en el campo de la salud están diseñados para proporcionar ayuda en entornos médicos y de atención sanitaria. Desempeñan tareas variadas como pueden ser la cirugía (Figura 1.6(a)), la rehabilitación (Figura 1.6(b)) y la trasferencia de pacientes (Figura 1.6(c)), entre otros.



(a) Da Vinci

(b) Atlas  
Bionics

(c) Riba 2

Figura 1.6: Robots en el campo de la salud

## Robots en logística

Los robots de logística están enfocados a tareas de transporte y organización de mercancías en almacenes y centros de distribución. Son capaces de moverse de manera

autónoma, cargar y descargar objetos ágilmente y bajo demanda optimizando los procesos logísticos. Un ejemplo claro de ellos son los robots utilizados en Amazon (Figura 1.7(a)).

Más allá de su uso de almacenes, cabe destacar los llamados robots de "última milla". Son aquellos usados en el reparto de comida y paquetes en las ciudades. Aunque actualmente se encuentra en fase de desarrollo, ya han sido probados algunos prototipos como el usado por Glovo en Londres para repartir comida (Figura 1.7(b))



(a) AMR Amazon Robotics



(b) Robot de reparto de Glovo

Figura 1.7: Robots de logística

### 1.3. Robótica industrial

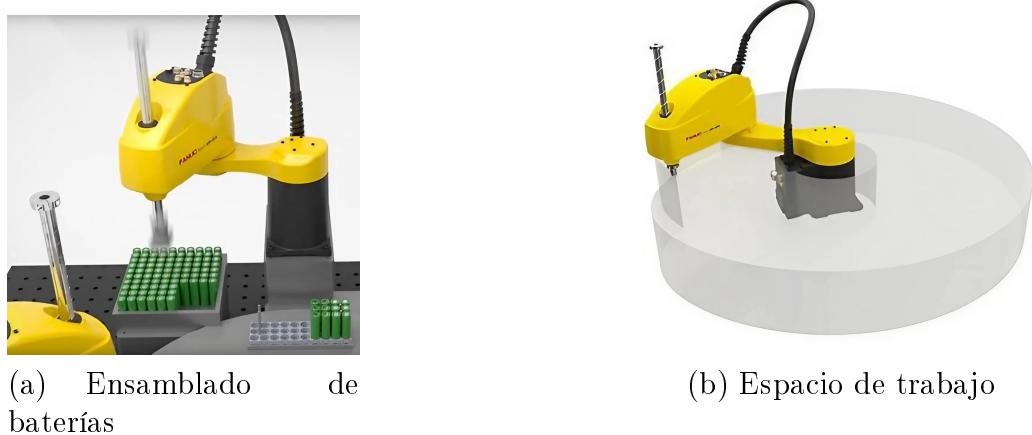
Se entiende por robot industrial a una máquina automatizada diseñada específicamente para llevar a cabo tareas en entornos industriales. Disponen de numerosas articulaciones y una gran capacidad de maniobrabilidad. Su objetivo principal es reemplazar a un operario humano en tareas aburridas, sucias, peligrosas y exigentes (*4D: Dull, Dirty, Dangerous and Demanding*). En función de el número de articulaciones y la geometría del manipulador, se pueden clasificar en diferentes categorías.

#### SCARA

Los robots de tipo SCARA se caracterizan por la disposición de sus 3, o 4, articulaciones con todos los ejes de rotación perpendiculares al suelo. Su capacidad para realizar movimientos rápidos y precisos en un plano horizontal lo hacen ideal para su uso en aplicaciones de ensamblaje electrónico, operaciones de *pick and place* (coger componentes y situarlos) y empaquetado de productos, entre otras.

---

<sup>1</sup><https://www.fanuc.eu/uk/en/robots/robot-filter-page/scara-series/scara-sr-3ia>

Figura 1.8: Ejemplo: FANUC SR-3iA <sup>1</sup>

### Articulados

Es el tipo de brazo robótico más usado en la industria. Están formados por una estructura similar a la de un brazo humano. Poseen entre 4 y 6 articulaciones de tipo revolución (giran en torno a un eje) permitiendo realizar movimientos complejos.



Figura 1.9: Robot articulados

### Paralelos

Son un tipo de robot industrial que se caracteriza por tener una estructura mecánica en la cual varios brazos o cadenas cinemáticas se conectan de forma paralela a una plataforma móvil o base. Estos robots son rápidos y livianos, por lo que son ideales para aplicaciones que requieren alta velocidad, como en la industria alimentaria o en la selección y empaquetado de productos.

<sup>2</sup><https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industria/kr-iontec>

<sup>3</sup><https://new.abb.com/products/robotics/robots/articulated-robots/irb-6700>



(a) Robot Delta ABB



(b) FANUC M-410iC

Figura 1.10: Robots basados en paralelogramos

## Cartesianos

Los robots pertenecientes a esta categoría, se mueven a lo largo de unas guías lineales en la dirección de los tres ejes cartesianos (X, Y, Z) y cuyos ejes forman ángulos rectos entre sí.

Se trata de un tipo de máquina muy común en la industria y es usada en tareas de *pick and place* y fresado, entre otras.



(a) VP-2500DP<sup>4</sup>



(b) OpenBuilds CNC Lead 1010<sup>5</sup>

Figura 1.11: Robots cartesianos

En resumen, la robótica es un campo en constante evolución que busca crear máquinas autónomas e inteligentes capaces de realizar una gran variedad de tareas cada vez más complejas. Sin embargo, a medida que la demanda de habilidades en

<sup>4</sup><https://www.smallsmt.biz/bestueckungsautomat/>

[5 https://openbuilds.com/builds/lead-cnc-1010-40-x-40.7832/](https://openbuilds.com/builds/lead-cnc-1010-40-x-40.7832/)

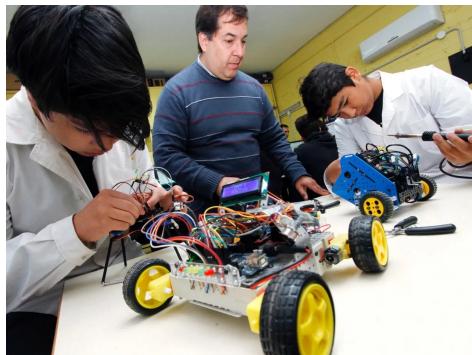
robótica y automatización aumenta, se hace evidente la necesidad de formar a más personas en este campo, siendo la robótica educativa una herramienta fundamental para ello.

## 1.4. Robótica educativa

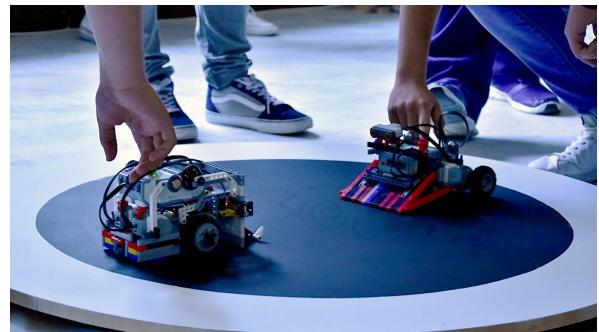
La robótica educativa es una disciplina que ha adquirido una gran relevancia en los últimos años debido a la creciente necesidad de formar a las nuevas generaciones en competencias tecnológicas.

### 1.4.1. Robótica en la educación secundaria

En los institutos, la robótica se ha convertido en una herramienta pedagógica eficaz para desarrollar habilidades y conocimientos en áreas como la programación, la matemática, la electrónica y la resolución de problemas. Esto es conocido como *Science, Technology, Engineering and Mathematics* (STEM). Gracias a esta formación, los estudiantes aprenden a diseñar, construir y programar robots simples para llevar a cabo una tarea específica, lo que les ayuda a comprender los conceptos de ciencia y tecnología de una manera más práctica e interactiva.



(a) Instituto en Argentina<sup>6</sup>



(b) Institutos en Toledo<sup>7</sup>

Figura 1.12: Robots en institutos

### 1.4.2. Robótica en la universidad

A nivel universitario, la robótica se ha convertido en una disciplina esencial para formar a los futuros ingenieros. Los estudiantes aprenden a diseñar y construir robots

<sup>6</sup><https://www.diariodecuyo.com.ar/suplementos/Robotica-la-ciencia-de-aprender-jugando-20170907-0087.html>

<sup>7</sup><https://www.uclmtv.uclm.es/institutos-de-camarena-y-madridejos-ganan-el-viii-orneo-de-robotica-organizado-por-la-uclm-en-toledo/>

más avanzados, y refuerzan sus habilidades de programación y control de sistemas complejos. En estos centros se hace uso de varios tipos de robots, como pueden ser, brazos industriales y plataformas robóticas móviles como los mostrados en la Figura 1.13. Al tratarse de sistemas usados en el mundo profesional, los estudiantes pueden aprender con robots similares a los que usarán en un futuro.

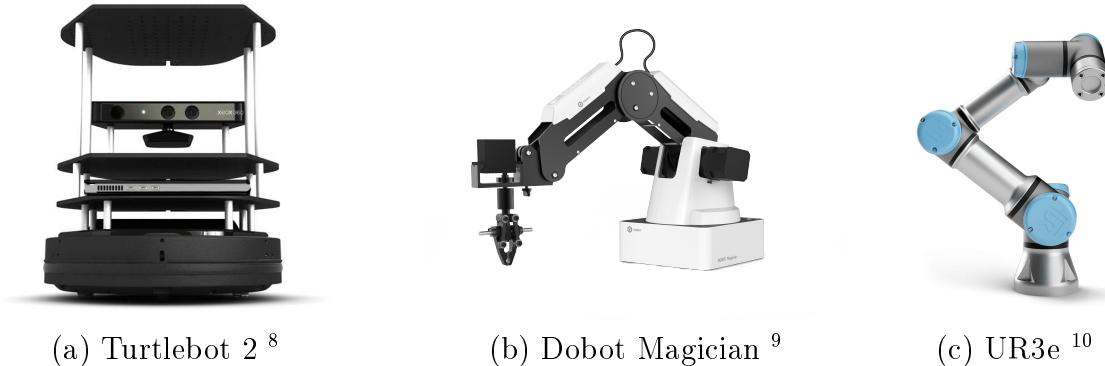


Figura 1.13: Robots usados en universidades

Por contra, aunque es verdad que las universidades disponen de algunas unidades, no siempre son accesibles para el estudiante por diversas razones. La mayor de ellas es su elevado coste que reduce la cantidad de unidades que se puede permitir la universidad. Es por esto que existe la creciente necesidad de crear robots asequibles para su uso en enseñanza.

## 1.5. Robótica de bajo coste

La robótica de bajo coste es el área de la robótica enfocada en el diseño y desarrollo de robots accesibles y asequibles. Tiene como objetivo conseguir desarrollar tecnología robótica barata, reduciendo la complejidad de los sistemas empleados y haciendo uso de materiales más económicos.

La disponibilidad de herramientas de fabricación de bajo coste, como la impresión 3D y el corte láser, ha hecho posible que los usuarios puedan crear piezas y componentes robóticos personalizados a un precio más bajo que el de la fabricación tradicional.

En este campo de la robótica se hace patente la Ley de Moore, una ley teórica que establece que la capacidad de procesamiento de los microchips se duplica cada dos años. Esta observación, que se planteó hace casi 60 años, sigue siendo válida en la actualidad y ha sido fundamental para el desarrollo de componentes electrónicos

<sup>8</sup><https://www.turtlebot.com/turtlebot2/>

<sup>9</sup><https://www.robotlab.com/store/dobot-robotic-arm>

<sup>10</sup><https://www.universal-robots.com/es/productos/robot-ur3/>

cada vez más pequeños, eficientes y potentes. Gracias a esto, ha surgido la filosofía del «Hazlo tú mismo» o *Do It by Yourself* (DIY), que se enfoca en la creación de proyectos personalizados y asequibles utilizando tecnología de bajo coste y materiales comunes.

En resumen, la robótica de bajo coste es una consecuencia directa del crecimiento de la capacidad de computación, el abaratamiento de los precios de los componentes electrónicos y el enfoque DIY. El desarrollo de esta tecnología, permite que más personas puedan experimentar en este área de la ingeniería y desarrollar sus propias soluciones robóticas.

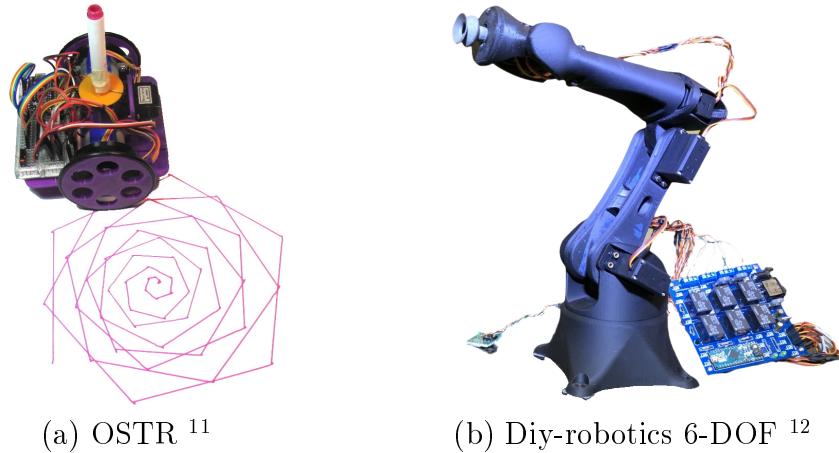


Figura 1.14: Robots de bajo coste

El presente trabajo se centra en el desarrollo de un robot industrial de tamaño reducido y bajo coste, diseñado con el propósito de facilitar su uso y control. Este robot ha sido completamente impreso en 3D y se encuentra integrado con herramientas ampliamente utilizadas en el ámbito de la robótica. Esta combinación permite que cualquier estudiante tenga acceso a él y, al replicar el proyecto, aprenda de manera práctica sobre el apasionante mundo de la impresión 3D.

En el siguiente capítulo se expondrán diferentes trabajos e investigaciones llevadas a cabo que guardan una cierta similitud con el tipo de robot que se ha desarrollado.

---

<sup>11</sup><https://www.instructables.com/Low-Cost-Arduino-Compatible-Drawing-Robot/>

<sup>12</sup>[https://diy-robotics.com/educative-robot-cell-free-package/?utm\\_source=Instructables.com&utm\\_medium=referrer&utm\\_campaign=Educative%20Cell](https://diy-robotics.com/educative-robot-cell-free-package/?utm_source=Instructables.com&utm_medium=referrer&utm_campaign=Educative%20Cell)

---

## Capítulo 2

# Estado del arte

---

En este capítulo se describen algunos de los trabajos más relevantes publicados sobre robots industriales en educación.

En [Krimpenis et al., 2020] se presenta una solución industrial barata para crear un robot capaz de hacer operaciones de mecanizado (fabricación de piezas mediante operaciones de corte). Este robot se llama HydraX y está dotado de 6 grados de libertad. Tiene un alcance máximo de casi un metro, un peso que ronda los 13 Kg y está fabricado mediante impresión 3D. Además, tiempo después, se publicó la segunda parte de este artículo [Papapaschos et al., 2020], en el cual se aborda el diseño software y de control que se ha desarrollado para controlar dicho brazo.



Figura 2.1: Robot HydraX

Los puntos fuertes del proyecto son los siguientes:

- Tiene una repetitividad de  $\pm 0.04$  mm.
- Tiene un espacio de trabajo muy amplio.
- Tener más de 3 grados de libertad le permiten alcanzar gran cantidad de puntos con distintas orientaciones.
- Según el artículo, tiene una capacidad de carga máxima de 12 kilogramos. A pesar de esto, se comenta que en la práctica el peso en el extremo del robot debe ser menor a 5 Kg, por lo que su carga útil rondará los 3 Kg para un funcionamiento aceptable. Esto lo sitúa a la par del robot comercial ABB IRB 120<sup>1</sup>

Y estos son los puntos débiles que cabe mencionar:

- Carece de integración en *Robot Operating System* (ROS), plataforma de desarrollo de la que se habla en la Sección 4.1.5.
- Su coste en materiales supera los 1000 €, por lo que no es lo suficientemente asequible para su uso académico.
- Según este artículo, se necesitan 200 horas para poder imprimir y montar el brazo, lo que implica que es costoso en tiempo crear varias unidades, además de que requiere de cierta habilidad para construirlo correctamente.
- En este artículo no se proporcionan los ficheros necesarios para poder crearlo. Además, se menciona que las piezas han sido diseñadas mediante el software privativo SolidWorks®, por lo que lo hace más difícil y costoso de editar.

---

<sup>1</sup><https://new.abb.com/products/es/3HAC031431-001/irb-120>

Existen soluciones más sencillas y reproducibles, como por ejemplo, la presentada en [Adediran et al., 2023]. En este trabajo se describe un brazo robótico cuyo propósito es separar botellas de plástico en un proceso de reciclaje. Más allá de la aplicación que se le pretende dar, lo mas importante es que se hace uso de un manipulador de tamaño reducido, impreso en 3D y con 4 *Degrees Of Freedom* (DOF).

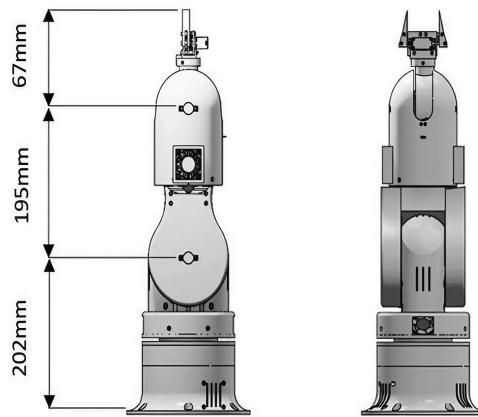


Figura 2.2: Robot UIArm

A partir de la información proporcionada en este artículo, se han extraído los siguientes puntos fuertes:

- El diseño de las piezas se ha realizado mediante el uso de la herramienta de diseño FreeCAD, esto supone una ventaja respecto a [Krimpenis et al., 2020], que utilizaba una herramienta privativa.
- Utiliza motores de pequeño tamaño con una reductora integrada, lo que aumenta su torque y abarata en gran medida los costes del robot. Debido a esto, el consumo de energía es menor pudiendo hacer uso de una electrónica compacta y menos costosa.
- Al estar compuesto por menos piezas y tener pocos grados de libertad, se requiere menos tiempo para imprimir y construir este robot.

Y estos son puntos negativos que cabría destacar:

- El espacio de trabajo (puntos del espacio alcanzables por el extremo del brazo) del robot es demasiado reducido. Esto es debido a la disposición de los grados de libertad. Utiliza dos de ellos para cambiar la orientación del extremo del robot, dejando solo dos para el posicionamiento, por lo que es imposible en la práctica abarcar todo el espacio 3D al alcance del brazo.

- Utiliza engranajes impresos en 3D para rotar la segunda articulación comenzando desde la base. Debido a esto, pierde exactitud en función de la posición del brazo debido a la holgura de este tipo de transmisión.
- Carece de integración con ROS y el autor no proporciona otro tipo de software que permita desarrollar programas para este robot perjudicando la continuidad del proyecto.

Otro desarrollo *low-cost* que ha adquirido gran popularidad gracias a la plataforma KickStarter<sup>2</sup> es el robot Niryo One, que es un brazo robot impreso en 3D con 6 grados de libertad, en la cual fue capaz de recaudar 80.000€ de los 20.000 € necesarios.

Este proyecto busca dar una alternativa de bajo coste al robot convencional usado por *makers*, estudiantes y pequeñas compañías. Para lograrlo, apuesta por el uso de componentes comunes y la filosofía *Open Source*, la cual se basa en la idea de compartir, colaborar y democratizar el acceso al conocimiento y al software. Por ello, su código fuente y planos están disponibles de forma gratuita para que cualquier persona interesada pueda acceder a ellos.

Aunque este modelo no es el más nuevo de la marca Niryo, sí es el que más comunidad tiene y más extendido está. Este robot y los modelos posteriores, pueden ser adquiridos a través de su página oficial<sup>3</sup>.



Figura 2.3: Robot Niryo One

---

<sup>2</sup><https://www.kickstarter.com/projects/niryo/niryo-one-an-open-source-6-axis-robotic-arm-just-f/description>

<sup>3</sup><https://niryo.com/products-cobots/niryo-one/>

Este proyecto destaca positivamente por:

- El proyecto completo es de código abierto y se puede encontrar toda la electrónica necesaria y documentos para imprimirla en su repositorio de github<sup>4</sup>.
- Tiene integración con ROS 1 y MoveIt.
- Cuenta con 6-DOF, lo que permite alcanzar gran cantidad de puntos con distintas orientaciones.
- Cuenta con una repetitividad de 0.5mm y puede levantar hasta 300g.
- Permite adaptar una gran variedad de herramientas diferentes.
- Es capaz de detectar colisiones gracias a sus motores codificados mediante sensores magnéticos.

En cambio, tiene los siguientes inconvenientes:

- Hace uso de una placa Raspberry Pi en vez de una placa Arduino/Esp32, lo que incrementa el coste del equipo.
- Hablando del precio, el coste de este equipo comprado en su versión en aluminio ronda los 3500€. El coste de fabricación de este robot mediante el uso de impresión 3D ronda los 1000€ según este artículo<sup>5</sup>
- Aunque proporcionan los ficheros fuentes del diseño, este ha sido realizado mediante SolidWorks® por lo que no puede ser editado sin tener que pagar por el programa.
- Pese a tener integración con ROS1, el proyecto no ha sido adaptado de forma oficial para ROS2, por lo que es necesario usar un sistema operativo antiguo (Ubuntu 16.04) del año 2016. O bien utilizar Ros Bridge<sup>6</sup> para comunicar el ROS 1 de la placa con un ordenador moderno con ROS 2.

---

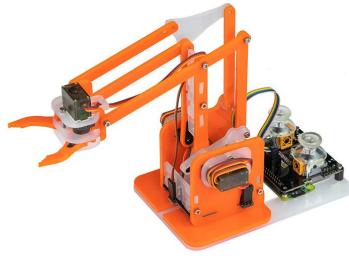
<sup>4</sup>[https://github.com/NiryoRobotics/niryo\\_one](https://github.com/NiryoRobotics/niryo_one)

<sup>5</sup><https://www.zdnet.com/article/this-mini-industrial-robot-is-less-than-1k/>

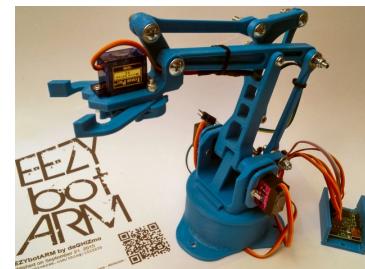
<sup>6</sup>[https://github.com/ros2/ros1\\_bridge](https://github.com/ros2/ros1_bridge)

En el ámbito de los robots caseros, se puede destacar un referente reconocido, MeArm. Se trata de un brazo mecánico de diseño simple y completamente *OpenSource*. De hecho, tal es su repercusión, que miles de personas se han impreso el suyo o creado su propia versión mejorada. Prueba de ello es la cantidad de modificaciones que se han ido subiendo a páginas como Thingiverse<sup>7</sup>. Además, existen robots con diferentes nombres basados en el mismo concepto, como puede ser EEZYBotArm, una versión más robusta y atractiva que el original.

Consta de 3 grados de libertad y de una pinza simple. Todos los motores del robots son servomotores de 9 gramos y se basa en el uso de paralelogramos para transferir el movimiento de los motores al extremo del robot, manteniendo el peso de los mismos en la base.



(a) MeArm



(b) EEZYBotArm MK1

Figura 2.4: Robots formados a partir de paralelogramos

Se trata de un proyecto muy extendido, con una comunidad que ha creado muchas variantes de él, por lo que para evaluar los puntos fuertes y débiles del robot, se va a hacer referencia al robot original MeArm V1. Los aspectos significativos de este robot son:

- Se trata de un proyecto totalmente accesible y replicable, ya que usa pocos materiales y estos son fáciles de adquirir.
- El centro de masa del robot se concentra en la base, reduciendo la inercia del brazo y permitiendo así movimientos más rápidos.
- No requiere de apenas tiempo de impresión y el montaje es sencillo gracias a las numerosas guías de montaje<sup>8</sup> que circulan por internet.
- Al usar servomotores, se puede establecer el ángulo de cada articulación fácilmente sin necesitar de ningún tipo de sensor externo.

<sup>7</sup><https://www.thingiverse.com/search?q=mearm&page=1&type=things&sort=relevant>

<sup>8</sup><https://docs.rs-online.com/2bb1/0900766b81593e58.pdf>

- Debido a su forma y materiales utilizados, tiene un peso muy reducido pudiéndose acoplar a robots móviles sin afectar a su rendimiento.

En cuanto a sus limitaciones, se deben mencionar:

- Se suele prescindir del uso de rodamientos en sus articulaciones. Esto reduce el tiempo de vida del brazo debido a que el propio rozamiento de los ejes acaba desgastando el plástico, creando holguras que afectan directamente en la precisión. Esta holgura obliga a apretar en exceso los tornillos que hacen de ejes de las articulaciones, aumentando el rozamiento y reduciendo fuerza útil de los pequeños motores.
- Este robot está limitado en cuanto a fuerza debido al uso de servomotores. Usa los típicos SG90 o similares con un torque de apenas 0.16 Newton-metro. El uso de servos más grandes aumenta significativamente el precio del dispositivo.
- Los servomotores padecen de vibraciones al conectarle una carga. Esto hace que el conjunto tiemble en exceso durante los movimientos.

Una vez se han presentado los proyectos más relevantes en este campo, se procederán a definir una serie de requisitos y objetivos concretos para dar solución al problema que se plantea en este trabajo.

---

# Capítulo 3

## Objetivos

---

*Un objetivo sin un plan es solo un deseo.*

Antoine de Saint-Exupéry

Tras haber enmarcado el contexto en el cual se encuentra este trabajo de fin de grado, se procede a realizar una descripción del problema, requisitos, metodología y plan de trabajo usado.

### 3.1. Descripción del problema

Este trabajo de fin de grado nace de la necesidad de abordar el problema existente en las universidades, donde la escasez de robots disponibles, y la dificultad para acceder a ellos, limitan la experiencia práctica en robótica.

Este problema es causado en gran medida por elevado coste y fragilidad de los robots utilizados, limitando la interacción plena de los estudiantes por temor a dañarlos. Además, la cantidad limitada de ellos en relación con el número de carreras que quieren utilizarlos, limita su disponibilidad. Sumado a ello, los profesores tienen que hacer frente a la burocracia asociada a su uso.

La solución propuesta busca superar estas limitaciones, proporcionando un robot casero impreso en 3D, que es económico, accesible y útil en el proceso de aprendizaje práctico de los estudiantes. Concretamente, el objetivo principal de este trabajo de fin de grado es desarrollar un brazo robótico que pueda ser empleado en la asignatura de Robótica Industrial de esta universidad. Asimismo, se busca que el sistema creado sea barato y fácilmente replicable, para que se pueda disponer de un número elevado de estos en el aula.

Con el fin de alcanzar este objetivo, se consideran los siguientes subobjetivos:

1. Realizar una investigación acerca de los robots que actualmente están disponibles y que cumplan con las características y objetivos deseados.
2. Explorar diversas opciones de diseño para determinar la forma del robot.
3. Realizar una investigación exhaustiva sobre los componentes de hardware disponibles en el mercado, con el fin de seleccionar aquellos que mejor se adapten a las necesidades y objetivos.
4. Realizar el diseño CAD de las piezas mediante el uso de herramientas libres.
5. Emplear una impresora 3D convencional para materializar las piezas realizadas.
6. Realizar el software necesario para poder controlar el robot desde el ordenador.
7. Integrar el robot en el ecosistema ROS2/MoveIt.

### 3.2. Requisitos

Con el fin de solucionar el problema descrito, se han establecido los siguientes requisitos:

1. La fabricación del brazo robótico no debe suponer un coste superior a 200€.
2. La mayoría de las partes que componen el robot deben ser imprimibles en cualquier impresora 3D convencional.
3. A fin de poder garantizar la portabilidad del robot, este debe tener un consumo inferior a 25 vatios.
4. En cuanto a sus dimensiones, se busca un tamaño idóneo para su uso sobre un escritorio. Esto implica que no necesariamente tiene que estar unido al suelo, permitiendo así su fácil traslado.
5. Debe ser capaz de levantar cargas de hasta 300 gramos.
6. Es necesario que sea simple de montar y esté compuesto del menor número de piezas posibles, con el fin de poder crear varias unidades en poco tiempo.
7. Se busca continuidad en el proyecto a largo plazo, por lo que debe tener integración con el ecosistema ROS 2.

### 3.3. Metodología

Durante el desarrollo del trabajo se ha establecido un protocolo de reuniones semanales con el tutor a través de la plataforma Teams, con el objetivo de compartir los avances realizados y recibir retroalimentación sobre el trabajo. Además, cada semana se han propuesto las actividades a realizar, asegurando así una adecuada planificación y coordinación del proyecto.

Para el desarrollo del sistema se ha utilizado un repositorio en la plataforma GitHub<sup>1</sup>, en el cual se ha ido subiendo el código y diseños generados a lo largo del proyecto. Adicionalmente, en este mismo repositorio se ha incluido una Wiki<sup>2</sup> con las explicaciones detalladas de todas las actividades llevadas a cabo durante estos meses de trabajo. De esta manera, se ha creado un registro completo y accesible de todo el proceso de desarrollo del sistema.

### 3.4. Plan de trabajo

El desarrollo del TFG ha estado dividido en las siguientes etapas:

1. *Investigación del estado del arte.* Fase inicial en la que se realizaron diferentes búsquedas en plataformas online como Google Scholar con el fin de encontrar posibles soluciones al problema descrito. Se buscaban palabras clave como “DIY”, “educational”, “low-cost”, “robot”, “arm”, “DOF” y “3D printed”. Además se priorizó aquellos trabajos que utilizaban software y hardware libre.
2. *Encontrar la forma del robot ideal.* Se analizó cuidadosamente qué tipo de robot se ajusta mejor a los requisitos propuestos. Se establecieron los grados de libertad necesarios y su principio de funcionamiento. Para ello, se investigó los posibles tipos de articulaciones y su configuración.
3. *Análisis del mercado de componentes.* En esta fase, se realizó un análisis detallado de los precios y características técnicas de cada componente, para seleccionar aquellos que ofrezcan la mejor relación calidad-precio. Una vez evaluados todos los aspectos, se eligió el conjunto final de componentes que componen el robot.

---

<sup>1</sup><https://github.com/RoboticsURJC/tfg-vperez>

<sup>2</sup><https://github.com/RoboticsURJC/tfg-vperez/wiki>

4. *Diseño CAD.* Tras haber definido el concepto y haber seleccionado los componentes físicos que lo componen, se comenzó con el diseño del manipulador. Primero de todo, se realizaron numerosos bocetos a mano alzada para establecer la posición espacial de cada pieza. Posteriormente se concretó la forma exacta de cada pieza teniendo en cuenta que dichas piezas debían ser impresas en 3D y requerir de la menor cantidad de material posible. Finalmente, se hizo uso de la herramienta FreeCAD para diseñar por ordenador todas y cada una de las piezas.
5. *Impresión 3D.* En esta fase del proyecto, se materializaron los diseños previos mediante la técnica de impresión 3D haciendo uso de una impresora de filamento (tipo FDM) convencional. Previamente, se habían realizado pruebas de tolerancias para garantizar que las piezas impresas cumplían con las medidas especificadas en el diseño y calibrar la dilatación térmica en función de eso. Esta fase finalizó con el ensamblado de todos los componentes y piezas.
6. *Desarrollo del software de control.* Una vez fue construido el robot, se desarrolló toda la programación de bajo nivel necesaria para poder controlar el robot desde el ordenador. Para ello se investigó las posibles opciones de comunicación con la electrónica elegida. Además, se calculó todas la matemáticas necesarias para su control numérico.
7. *Integración en ROS2 y MoveIt.* Tras poder controlar el robot mediante el ordenador, se realizó una integración en el ecosistema ROS. Se describió el robot mediante el formato URDF y se integraron los controladores de articulaciones de ROS necesarios para su control mediante *topics*. Posteriormente, se realizó la integración con el *framework* MoveIt a partir del paquete de descripción generado previamente.
8. *Evaluación del desempeño.* En esta etapa final, se evaluó las distintas capacidades técnicas del manipulador con el software final. Se encontraron las limitaciones físicas del robot y se generaron una serie de tablas con los resultados obtenidos.

---

# Capítulo 4

# Plataforma de desarrollo

---

*Las herramientas adecuadas en las manos adecuadas  
pueden cambiar el mundo*

Steve Jobs

Tras haber establecido los objetivos que se pretenden alcanzar en este trabajo de fin de grado, se procede a describir las herramientas *software* y *hardware* utilizadas para lograrlos.

## 4.1. Software

En esta sección se describen el conjunto de programas y librerías externas que han sido necesarias en el desarrollo de este trabajo.

### 4.1.1. Python

Python es un lenguaje de programación interpretado de alto nivel. Cuando se dice que un lenguaje es de alto nivel, se hace referencia a su nivel de abstracción y facilidad de uso en comparación con los lenguajes de bajo nivel, es decir, aquellos que te permiten mayor control sobre los componentes físicos del sistema. Decimos que es un lenguaje interpretado debido a que no es necesario convertir el texto escrito por el humano en instrucciones entendibles por un procesador previamente a su ejecución.

Una de las características más destacadas de Python es su amplia librería<sup>1</sup> estándar, que proporciona una gran variedad de módulos y funciones para realizar multitud de tareas. Además, como cualquier otro lenguaje de programación, cuenta con una gran cantidad de librerías de terceros que amplían sus capacidades, como SciPy en la ciencia de datos, Django en el desarrollo web, Numpy para operar con matrices, TensorFlow el aprendizaje automático, entre otros.

---

<sup>1</sup>conjunto de código destinado a un objetivo concreto

### 4.1.2. Grbl

Grbl<sup>2</sup> es un *firmware*<sup>3</sup> de código abierto utilizado para controlar máquinas CNC. Está pensado para poderse ejecutar en microcontroladores de bajos recursos, como pueden ser los Arduinos y los ESP32.



Figura 4.1: Logo de Grbl

Básicamente, convierte una serie de instrucciones de código G (posteriormente se hablará de él) en señales eléctricas que se envían a los motores de la máquina. Además de eso, comprueba los sensores de la máquina, para poder conocer y establecer los límites físicos de cada movimiento.

Grbl es flexible, por lo que podemos cambiar la configuración para adaptarla a un caso de uso concreto. De hecho, aunque principalmente está pensado para realizar movimientos lineales, en este trabajo se abordará la configuración necesaria para poder usarse en articulaciones rotativas.

### 4.1.3. G-code

También conocido como RS-274, es el nombre del lenguaje de programación más usado en máquinas de *Control Numérico por Computadora* (CNC).

Su programación, precisa de una sintaxis específica en la que se emplean letras y números para representar diferentes comandos y parámetros. Por ejemplo, la letra G seguida de un número indica un comando de movimiento, mientras que la letra M seguida de un número se utiliza para comandos misceláneos, como el encendido o apagado del láser. En el Código 4.1 se muestra un ejemplo muy sencillo, para trazar un cuadrado de 20mm de lado.

---

<sup>2</sup><https://github.com/gnea/grbl>

<sup>3</sup>software específico para el hardware elegido

```

G90      ; Establecer modo de posicionamiento absoluto
G21      ; Establecer unidad de medida en milímetros
F200     ; Establecer velocidad de avance a 200 unidades por minuto

G0 X0 Y0 ; Mover a la posición inicial (esquina inferior izquierda)
G1 X20   ; Mover horizontalmente hacia la derecha 20 mm
G1 Y20   ; Mover verticalmente hacia arriba 20 mm
G1 X0    ; Mover horizontalmente hacia la izquierda 20 mm
G1 Y0    ; Mover verticalmente hacia abajo 20 mm
G0 X0 Y0 ; Volver a la posición inicial (esquina inferior izquierda)

M2       ; Fin del programa

```

Código 4.1: Programa para realizar una trayectoria cuadrada

#### 4.1.4. FreeCAD

FreeCAD<sup>4</sup> es un programa de diseño asistido por ordenador de código abierto y gratuito, utilizado para modelar piezas industriales. Está dirigido al mundo de la ingeniería mecánica y el diseño de productos. Utiliza el enfoque del modelado paramétrico, que se basa en el uso de parámetros y relaciones entre ellos para construir modelos 3D que pueden ser modificados fácilmente.



Figura 4.2: Logo de FreeCAD

Una de las ventajas de FreeCAD es la disponibilidad de diversos bancos de trabajo especializados, conocidos como *workbenches*. Cada uno de ellos, proporciona herramientas y funciones específicas para tareas de diseño particulares. Además, es altamente personalizable, ya que incorpora un administrador de complementos que permite a la comunidad agregar nuevas herramientas y funcionalidades al software. Gracias a ella, el software mejora continuamente y resulta sencillo de aprender debido a la cantidad de documentación y tutoriales que existen.

---

<sup>4</sup>[https://www.freecad.org/index.php?lang=es\\_ES](https://www.freecad.org/index.php?lang=es_ES)

#### 4.1.5. ROS 2

ROS, por sus siglas en inglés, significa *Robot Operating System*. Se trata de una plataforma de código abierto utilizada para desarrollar y controlar sistemas robóticos. Su objetivo principal es facilitar el desarrollo de sistemas robóticos al proporcionar una infraestructura que permite a los desarrolladores centrarse en la lógica y el comportamiento específico de los robots, sin tener que preocuparse por la infraestructura de comunicación y control. Además, proporciona una colección de bibliotecas, herramientas y convenios que permiten a los desarrolladores reutilizar y modularizar el software para robots. También incorpora programas para la visualización de datos, depuración y simulación de robots creados por una comunidad activa de usuarios y desarrolladores.

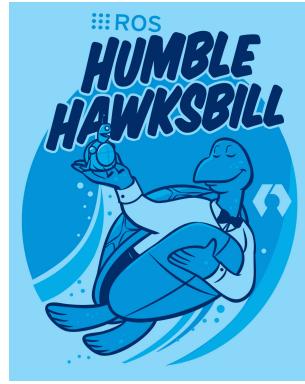


Figura 4.3: Logotipo de ROS 2 Humble

Una de las características distintivas de ROS es su arquitectura basada en nodos. Los nodos son procesos independientes que se comunican entre sí mediante mensajes. Cada nodo puede realizar tareas específicas, en función de la lógica atribuida por el programador.

Concretamente, en este trabajo se utiliza ROS 2, la versión sucesora de ROS. Esta versión incorpora una gran variedad de mejoras respecto a su predecesor. Por ejemplo, los nodos no dependen de un componente *Master* para comunicarse ya que las comunicaciones están distribuidas mediante el uso de *Data Distribution Service* (DDS). Esto incrementa la robustez del sistema y reduce significativamente los tiempos de latencia. La distribución de ROS utilizada en este trabajo es *ROS 2 Humble*. Se trata de la última versión de ROS estable disponible para el sistema operativo Ubuntu 22.04 LTS.

#### 4.1.6. MoveIt!

MoveIt es una plataforma de desarrollo (*framework*) de código abierto que permite implementar aplicaciones de manipulación complejas utilizando ROS.

Además, proporciona una amplia gama de herramientas y bibliotecas que facilitan el control y la planificación de movimientos. Con este software, se puede desarrollar fácilmente aplicaciones que permitan a los robots realizar tareas de manipulación avanzadas, como agarrar objetos, ensamblar piezas y maniobrar el brazo en entornos complejos.

Es altamente flexible gracias a su diseño modular y se puede utilizar con una gran variedad de robots. De hecho, incorpora la herramienta *MoveIt Setup Assistant*, que se trata de un asistente de configuración con interfaz gráfica, que te permite generar los ficheros necesarios para poder incorporar tu propio robot en este ecosistema.

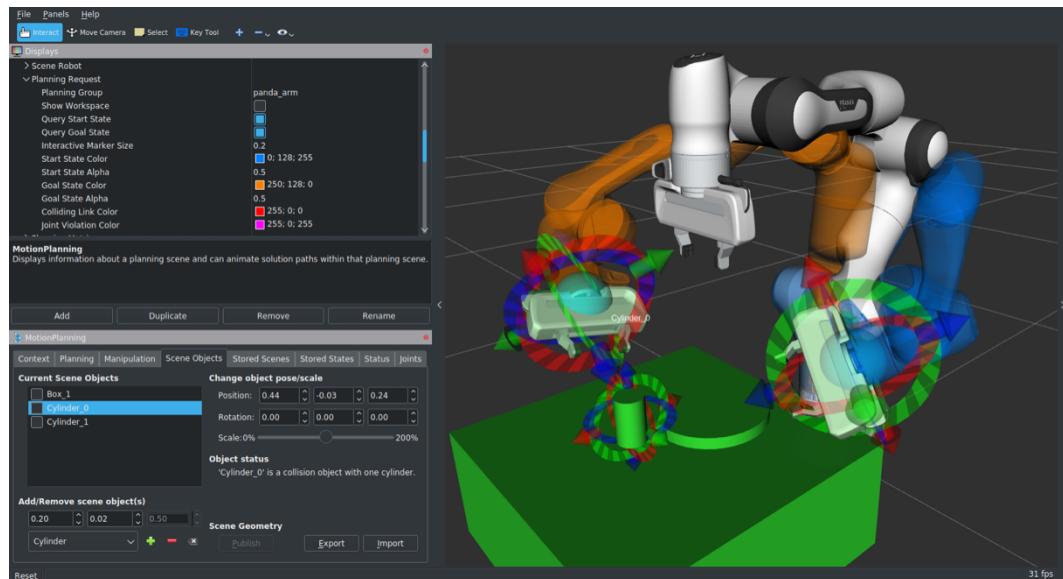


Figura 4.4: Plugin de MoveIt para el visualizador de ROS, Rviz2 (Franka Emika Robot)

## 4.2. Hardware

A continuación se describen todos los componentes físicos utilizados en trabajo. El plantamiento en la adquisición de estos, ha sido: funcionalidad, precio y disponibilidad. Para tener una visión general de los componentes hardware, se presenta la Figura 4.5.

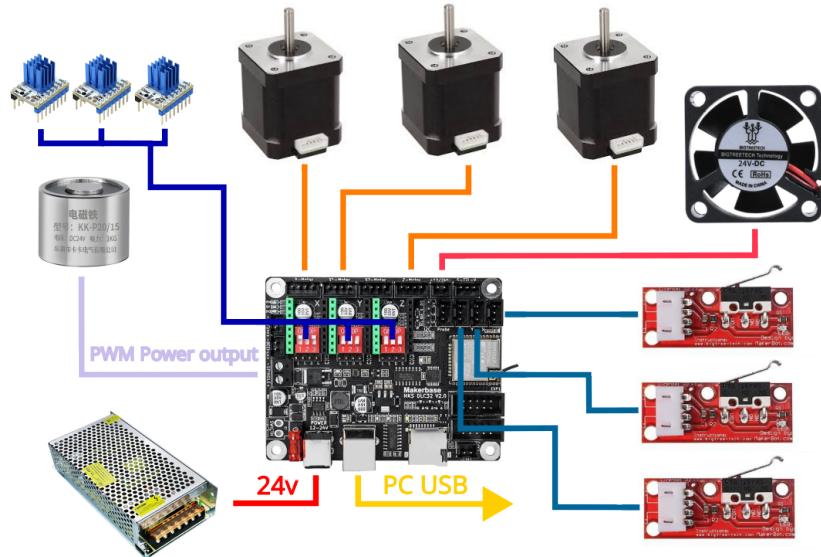


Figura 4.5: Esquema de la arquitectura hardware

A continuación se procede a describir en detalle aquellos que tienen un mayor peso en esta arquitectura hardware.

### 4.2.1. MKS DLC32

Se trata de una placa base<sup>5</sup> destinada al mundo de las máquinas de grabado láser y creada por *MakerBase* con carácter *open hardware*, por lo que toda la información de la placa puede encontrarse en su repositorio de Github<sup>6</sup>. Aun así, es fácil de comprar en páginas web como *Aliexpress* por un precio que ronda los 16€.

Está basada en el microcontrolador ESP32. Éste, es un dispositivo muy asentado en la comunidad *maker* debido a su bajo coste e integración en el ecosistema Arduino que, sumado a su conectividad wifi, es una opción cada vez más popular entre los fabricantes de este tipo de placas.

Como cabe esperar, es compatible con Grbl e incorpora una serie de características que la hacen ideal para este proyecto. Es capaz de controlar hasta 3 motores junto con

<sup>5</sup>componente esencial que proporciona conexiones y circuitos para que los demás componentes se comuniquen entre sí

<sup>6</sup><https://github.com/makerbase-mks/MKS-DLC32>

una señal de *Pulse Width Modulation* (PWM) para añadirle un servomotor, electroimán o láser.

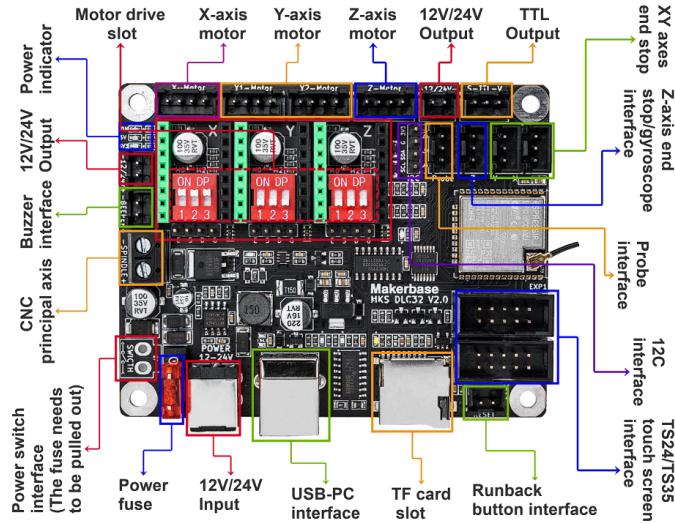


Figura 4.6: Placa base MakerBase DLC32

#### 4.2.2. Motores Nema 17

Un motor paso a paso es un tipo de motor que se mueve en pequeños incrementos discretos en lugar de girar continuamente. Esta posibilidad de tener un control sobre la rotación del motor hace que sea la opción idónea para mover las articulaciones del robot.

Concretamente, se han usado 3 motores Nema 17 de 60 milímetros de largo como los mostrados en la Figura 4.7. Se tratan de motores bipolares de 2.1 amperios y una resistencia de 1,6 ohmios con un torque máximo de 0.65 newton-metro. Estos, han sido comprados en *Amazon* por un precio de 55€.



Figura 4.7: Motores Nema 17 de 2.1A

En la Tabla 4.1 se muestran las especificaciones técnicas de los motores elegidos.

Parámetros	Valores
Nombre del motor	17HS24-2104S
Tipo de motor	Bipolar
Ángulo de paso	1.8°
Resistencia de fase	1.6Ω
Corriente máx. de fase	2.1A
Torque de sujeción	0.65Nm.
Dimensiones	42x42x60mm
Peso	500g
Diámetro del eje	5mm
Longitud del eje	24mm

Cuadro 4.1: Especificaciones técnicas de los motores utilizados

#### 4.2.3. Controlador TMC2209

Un controlador paso a paso es el módulo hardware capaz de transformar las señales lógicas que le envía el procesador, en una serie de pulsos de potencia que excitarán ambas bobinas del motor en un cierto orden para hacerlo girar. En concreto, se han utilizado 3 controladores TMC2209, compatibles con la placa mencionada anteriormente. Este modelo específico, es capaz de entregar mayor corriente que su competencia e incluye una serie de tecnologías que reducen el ruido en los motores y las vibraciones. Además, cuenta con una serie de protecciones para evitar dañarse en caso de un mal uso. Aún con todo esto, su precio por unidad es de apenas 3€.

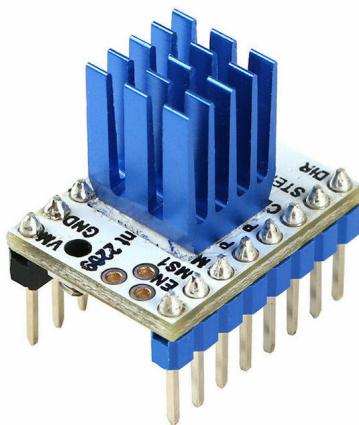


Figura 4.8: Controlador TMC2209

Parámetros	Valores
Nombre del controlador	TMC2209
Voltaje lógico	3 - 5V
Voltaje de alimentación	5.5 - 28V
Microsteps	Hasta 1/256
Corriente máx. de fase (RMS)	2A
Pérdida de conducción (RDS)	0.2Ω
Interfaz de comunicación	Pines CFG y UART

Cuadro 4.2: Especificaciones técnicas del TMC2209

#### 4.2.4. Fuente de alimentación genérica

Para alimentar el brazo se va a hacer uso de una fuente de alimentación genérica de 24 voltios y 5 amperios (Figura 4.9(a)). Este tipo de fuentes chinas son fáciles de encontrar en páginas como *Aliexpress* y su precio rondan los 15€. A pesar de haber usado esta, no es necesario utilizar una fuente tan potente. De hecho, puede ser usada cualquier tipo de fuente capaz de entregar más de 20 vatios de potencia en el rango de voltaje 12-24v. Por lo que es completamente viable utilizar cargadores de ordenadores portátiles como el de la Figura 4.9(b) para alimentar el robot.



(a) Fuente de alimentación 24v 5A

(b) Cargador de portátil genérico

Figura 4.9: Métodos usados para alimentar el robot

---

# Capítulo 5

# Desarrollo hardware del manipulador

---

*La simplicidad es la máxima sofisticación*

Leonardo da Vinci

En este capítulo, se describe el proceso de desarrollo completo llevado a cabo desde la concepción inicial de la idea hasta la materialización de un brazo robótico completamente funcional. A lo largo de estas páginas, se desglosará cada paso realizado y las dificultades encontradas, así como sus soluciones.

## 5.1. Geometría del manipulador

En esta sección se detalla el proceso llevado a cabo para definir el concepto y forma del robot, en base a los requisitos propuestos en el Capítulo 3.

Primeramente, se debe escoger el número de grados de libertad<sup>1</sup> del manipulador. En relación al espacio tridimensional (Figura 5.1), este cuenta con 6, estando 3 de ellos ligados al posicionamiento (X, Y, Z) y los otros 3 a las orientaciones (RX, RY, RZ). Es por esto que 3 es el mínimo número de grados de libertad necesarios para poder posicionar el extremo del robot un punto cualquiera del espacio.

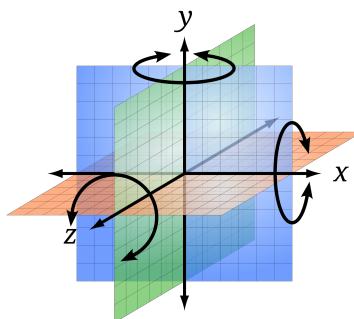


Figura 5.1: Espacio tridimensional

---

<sup>1</sup>número de movimientos independientes que puede realizar

Teniendo esto en cuenta, se ha optado por usar ese número, y no uno mayor, debido a que esto encarecería los costes totales del robot, no pudiendo cumplir así los requisitos 1 y 6 que limitan en cuanto a precio y complejidad.

Seguidamente, se debe elegir el tipo de articulación a utilizar. Aunque en robótica existen varias, en la práctica destacan dos:

- Revolución: Permite el movimiento de rotación alrededor de un eje fijo.
- Prismático: Las partes del robot se pueden desplazar linealmente a lo largo de un eje específico.

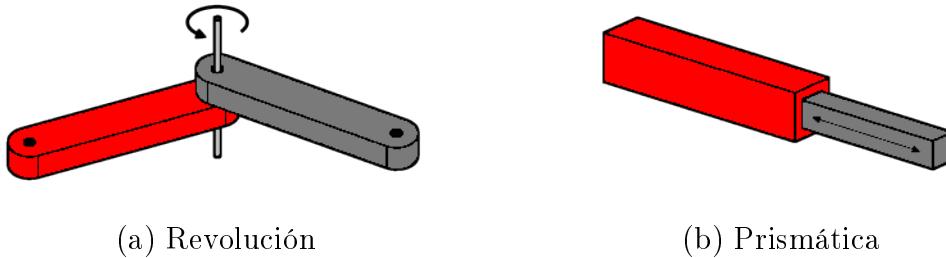


Figura 5.2: Tipos de articulaciones más usadas en robótica

Finalmente, se ha optado por emplear articulaciones de tipo revolución, debido a que son más sencillas de implementar y están constituidas por un menor número de partes.

En cuanto a la geometría del manipulador, se debe contar con la limitación existente a la hora de tener únicamente 3 grados de libertad. Esta es que, aunque podemos alcanzar cualquier punto del espacio, no podemos controlar la orientación del extremo del robot en dicho punto. En base a esto, se ha investigado en Internet qué tipos de robot existen con ese número de grados de libertad y como lo solventan. Principalmente destacan dos soluciones: los robots SCARA y los robots basados en paralelogramos (presentados en la Sección 1.3). Ambos, tienen en común la característica de mantener su extremo siempre paralelo al suelo, por lo que se tiene conocimiento sobre 2 de las 3 orientaciones en ese punto.

En la última orientación (Rotación en Z) se le suele incluir un cuarto grado de libertad, que en nuestro caso no tenemos. Aún así puede utilizarse con herramientas tipo electroimán o ventosas para tareas sencillas de movimiento de objetos. Concretamente, se pretende implementar un robot basado en paralelogramos que utilice el mismo principio de funcionamiento que los utilizados en tareas de paletizado<sup>2</sup>.

---

<sup>2</sup>Colocar objetos en un *palet* de forma ordenada y eficiente

## 5.2. Modelo alámbrico del manipulador

El modelo alámbrico es una forma de analizar el movimiento de un sistema mecánico compuesto por ejes y eslabones. Este enfoque simplifica la representación visual al destacar las relaciones espaciales entre las diferentes partes del sistema mediante líneas y conexiones simbólicas.

En la Figura 5.3 se muestra un ejemplo creado específicamente para ilustrar este concepto.

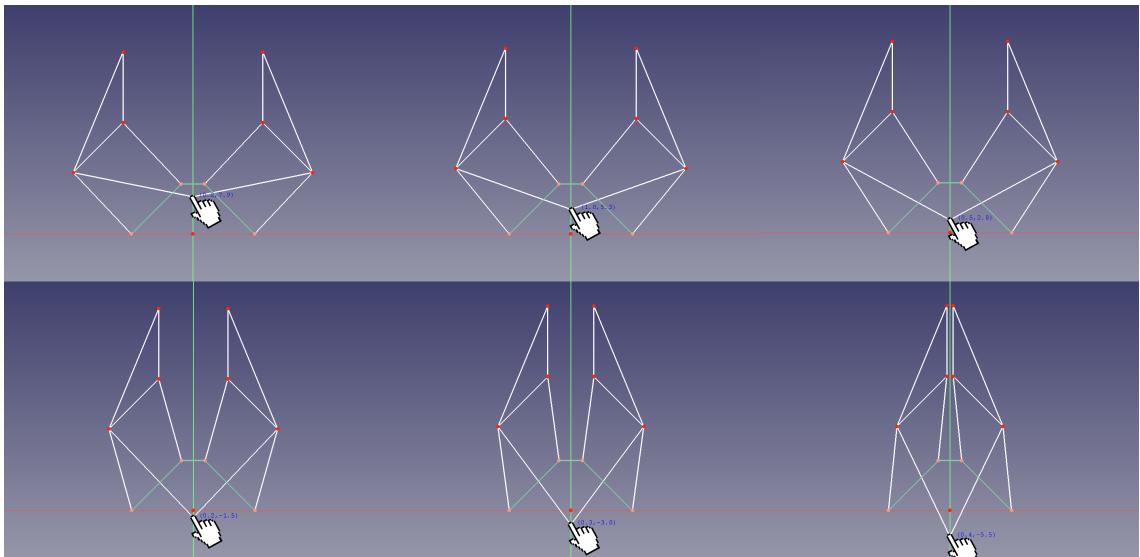


Figura 5.3: Pinza paralela con 1 grado de libertad

A la hora de crear el modelo alámbrico de este robot, se ha tomado como referencia el del robot casero MeArm (Figura 2.4(a)), en concreto el modelo alámbrico usado por Juan González en la asignatura de Mecatrónica<sup>3</sup> de esta universidad.

Este robot está definido por una serie de parámetros a los que se les ha puesto nombre: L1, L2, P1, P2, P3 y A (mostrados en la Figura 5.4). En función de los valores de estos parámetros, el comportamiento del brazo será diferente. Para encontrar los adecuados, no queda otra que utilizar la experimentación. Aún así, si se sigue un cierto orden, el proceso resulta realmente sencillo.

Es por esto que se ha seguido el siguiente procedimiento:

---

<sup>3</sup>[https://github.com/myTeachingURJC/Mecatronica/wiki/S3:-Estructuras-mec%C3%A1nicas-\(II\)](https://github.com/myTeachingURJC/Mecatronica/wiki/S3:-Estructuras-mec%C3%A1nicas-(II))

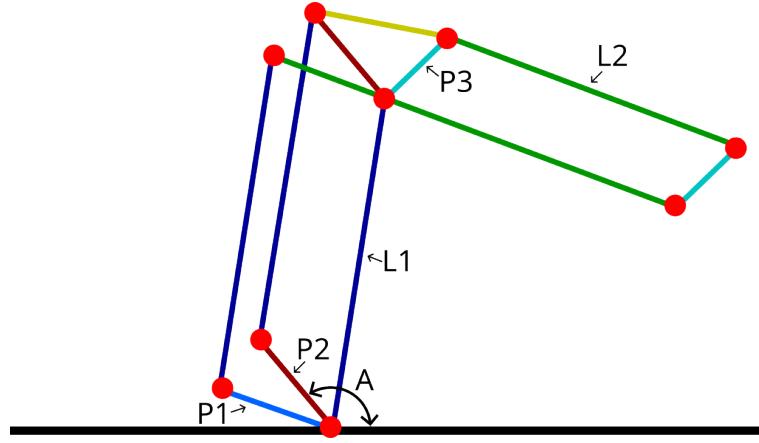


Figura 5.4: Principio fundamental del brazo MeArm

1. Se ha comenzado estableciendo el largo de los eslabones primarios L1 y L2. En base al tamaño de escritorio que se le pretende dar al robot, 17 centímetros por eslabón es un buen punto de partida.
2. El siguiente paso es elegir P1, P2 y P3. Estos son los lados cortos de los paralelogramos. Es importante jugar con estos valores de tal manera que se consiga obtener los más largos posibles y estos no intersequen entre sí. Esto es debido a que las piezas reales ocupan un cierto espacio y si el paralelogramo es muy pequeño no realizarán todo el recorrido ya que chocarán entre sí.
3. El ángulo A restringe el paralelogramo que mantiene el extremo del robot paralelo al suelo. Este ángulo hay que variarlo ligeramente. Un buen punto de partida es  $120^{\circ}$ . Este valor es crítico ya que afecta significativamente a la accesibilidad del brazo a lugares próximos a su base.

En la Figura 5.5 se muestra el modelo alámbrico final de este manipulador, bautizado como G-Arm<sup>4</sup>, cuya tabla de parámetros se puede ver en el Cuadro 5.1

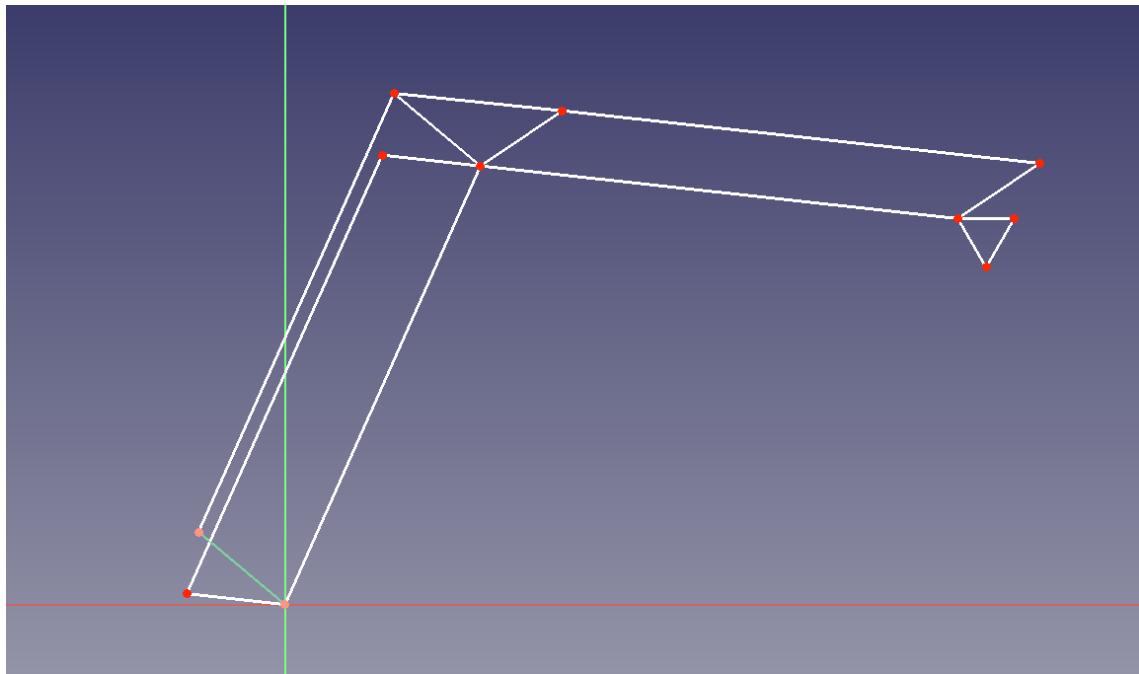


Figura 5.5: Modelo alámbrico de G-Arm

Parámetros	Valores
L1	170mm
L2	170mm
P1	35mm
P2	35mm
P3	25mm
A	135°

Cuadro 5.1: Parámetros del modelo alámbrico de G-Arm

---

<sup>4</sup>Bautizado así debido al lenguaje de programación que será utilizado posteriormente para comunicarse con él

### 5.3. Bocetos

Realizar bocetos es una fase clave del diseño que se lleva a cabo antes de comenzar a modelar en 3D. Con ellos, se busca tener una idea clara de la forma y posición de los elementos que constituyen el robot. Para el desarrollo de este brazo, se han realizado numerosos bocetos, de los cuales se conservan aquellos dibujados digitalmente en un iPad. En la Figura 5.6 se muestran algunos de ellos.

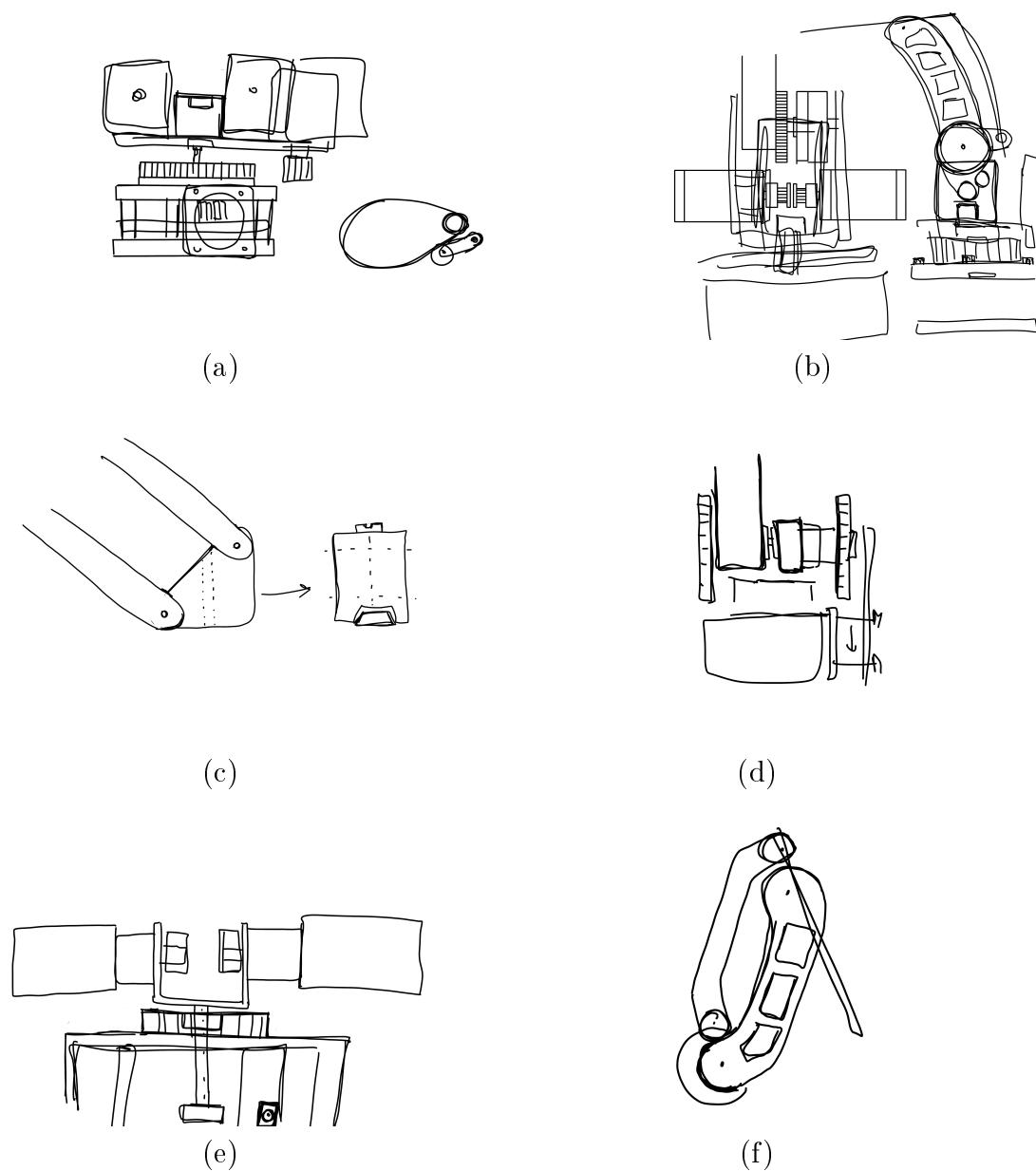


Figura 5.6

De hecho, antes pasar directamente al diseño CAD, se realizó una versión en miniatura como prueba de concepto. Primeramente se hizo un boceto detallado en base al modelo alámbrico de la sección anterior. Posteriormente se llevó al ordenador para finalmente, imprimirla en 3D. El proceso llevado a cabo, se puede ver en la Figura 5.7.

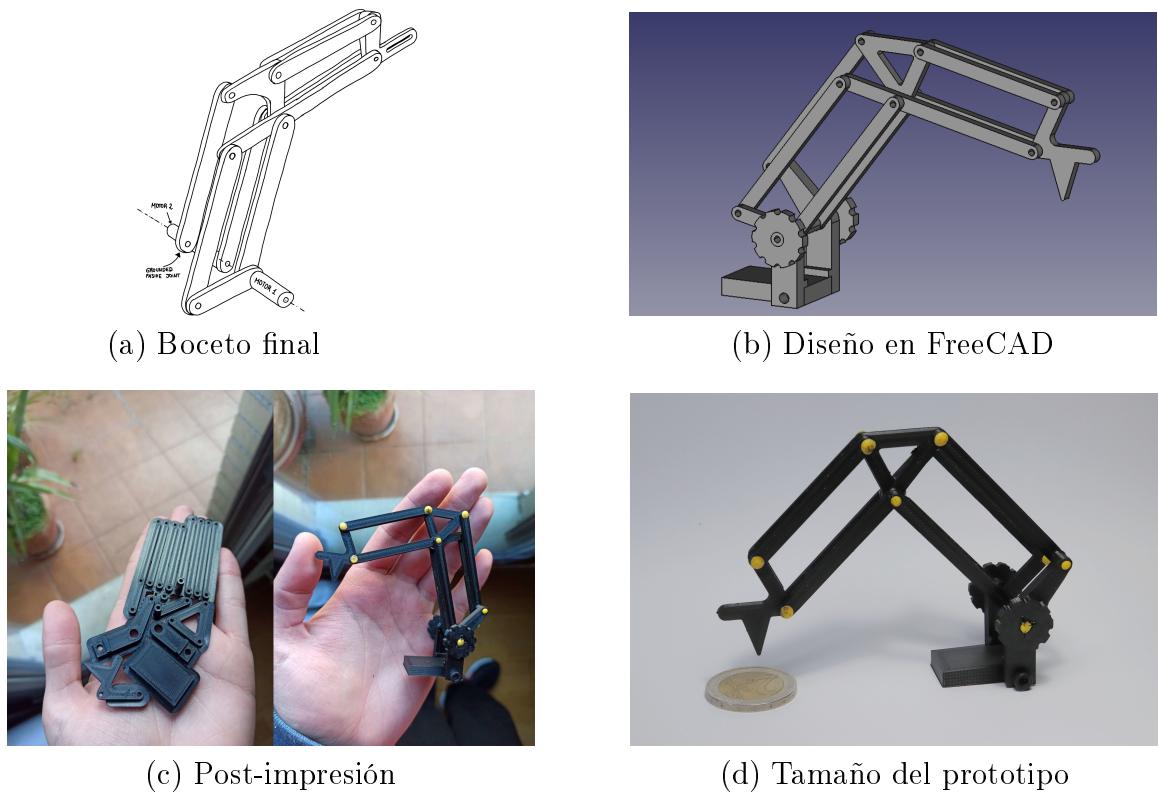


Figura 5.7: Evolución de la prueba de concepto

El pequeño prototipo, enteramente impreso en 3D, resultó funcional y bastante resistente, por lo que se dio por bueno el concepto. Gracias a haber hecho esto, se pudo tener una mejor idea de la posición de cada pieza, sirviendo de gran ayuda en la Sección 5.5.

## 5.4. Elección de componentes electromecánicos

En esta sección se eligen todos aquellos componentes que no pueden ser impresos en 3D y son necesarios para la construcción del robot. Estos componentes deben de ser asequibles y fáciles de encontrar en páginas de Internet como *Aliexpress* o *Amazon*. Con el propósito de encontrar el conjunto ideal, se evaluarán diversas opciones para seleccionar aquella que ofrezca las mejores prestaciones y se ajuste al presupuesto establecido.

### 5.4.1. Motores

El robot requiere de tres motores que deben contar con las siguientes características:

- Capacidad de entregar suficiente torque como para levantar el brazo más la carga.
- Se debe poder conocer su posición en cada instante.
- Deben poder ejercer un torque/retención estando detenidos.

Debido a este último requisito, no es viable usar motores convencionales con escobillas, ya que no son capaces de generar torque sin girar. En cambio, los motores paso a paso si son capaces de ello. Además suelen disponer de una gran fuerza y, aunque no estén codificados, se puede conocer su posición angular relativa debido a que avanzan en pequeños incrementos discretos (pasos). Como aspecto negativo, son bastante pesados. Aún así su precio no es muy elevado por lo que es una opción ideal para hacer un brazo robot.

Para estimar la fuerza necesaria, se ha realizado un análisis matemático. En la Figura 5.8 se muestra el escenario de mayor esfuerzo (brazo completamente extendido) y Ecuación 5.1 tal que da como resultado el torque mínimo necesario para mantener esa posición.

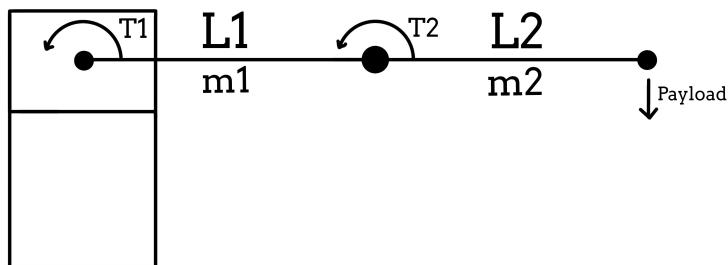


Figura 5.8: Dinámica del manipulador completamente extendido

$$T_1 = (m_1 * (L_1/2) + m_2 * (L_1 + L_2/2) + payload * (L_1 + L_2)) * g$$

$$T_2 = (m_2 * (L_2/2) + payload * (L_1 + L_2)) * g$$

Ecuación 5.1: Cálculo del torque necesario en la articulación más demandante

Teniendo en cuenta que cada eslabón mide 17cm y pesa 200g se necesitaría un torque mínimo de 1.66 N.m en la primera articulación y 0.92 N.m en la segunda.

Los motores paso a paso se dividen en diferentes categorías según su potencia y rendimiento. En este caso, se empleará un motor perteneciente a la categoría Nema, específicamente el Nema 17. Esta elección se basa en su amplia disponibilidad, facilidad para adquirirlos y su precio razonable.

Dentro de la categoría Nema 17, todos los motores comparten el mismo factor de forma, diferenciándose únicamente en su longitud, lo que afecta directamente al torque que pueden proporcionar. A mayor longitud, mayor capacidad de ejercer torque. Debido a esto, se ha optado por utilizar los más largos posibles dentro del presupuesto. Concretamente, 3 Nema 17 de 60mm (0.6 N.m) con un coste total de 55€.

Aunque el motor Nema 17 no es suficiente para suministrar por sí solo el torque necesario, se tiene previsto utilizar una reductora más adelante para multiplicar la fuerza final de la articulación. Esta estrategia evita la necesidad de utilizar un motor más grande y costoso. La reductora aumentará cinco veces el torque disponible para la articulación, permitiendo así un rendimiento adecuado sin comprometer la elección del motor Nema 17.

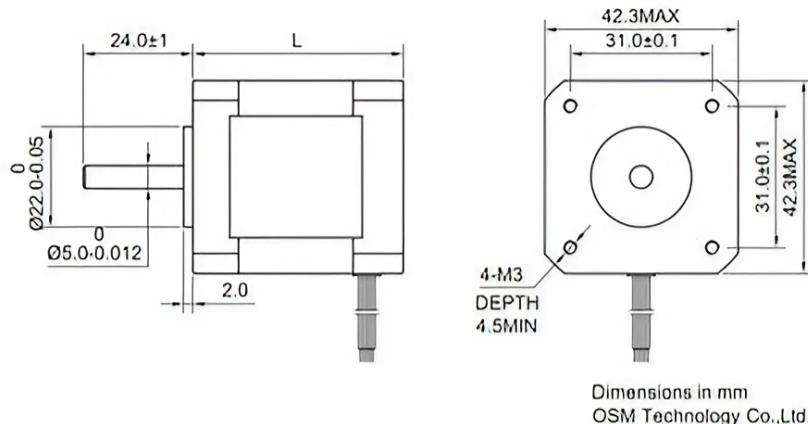


Figura 5.9: Dimensiones de los motores Nema17

### 5.4.2. Reductora

Para reducir la velocidad de giro de un motor y convertirla en fuerza, se requiere implementar una reductora. Existen varias opciones para lograrlo, como el uso de engranajes (normales, planetarios, helicoidales, etc.) o correas (lisas y dentadas).

En este caso, se ha optado por utilizar correas dentadas en lugar de engranajes, debido a que los engranajes tienden a introducir holguras en el movimiento final, ocasionando imprecisión e incertidumbre en los movimientos del robot. Por el contrario, las correas dentadas utilizan tensores para mantener siempre el contacto con ambas poleas, lo que resulta en un movimiento más preciso y confiable.

Se ha elegido el tipo de correa GT2 (Figura 5.10), ampliamente utilizado en impresoras 3D, por su bajo coste y fácil disponibilidad en el mercado. Dentro de esta categoría, se ha seleccionado un ancho de 6 mm, el más económico y adecuado para la aplicación requerida. El largo de cada correa se concretará en la Sección 5.5.



Figura 5.10: Ejemplo de correa GT2

Además, las poleas necesarias para este tipo de correas se pueden diseñar e imprimir en 3D de manera sencilla y económica. Esto permite personalizar el número de dientes de las poleas según las necesidades del robot, con un costo prácticamente nulo.

Para dos de los tres grados de libertad del robot, se ha decidido utilizar una combinación de poleas: una polea comercial de metal con 20 dientes unida al eje del motor y otra polea impresa en 3D con 100 dientes, logrando una reducción de 1:5<sup>5</sup>. En el tercer grado de libertad, se pretende emplear una polea con 120 dientes (reducción 1:6) siguiendo el mismo principio.

---

<sup>5</sup>La reducción 1:5 indica que, por cada vuelta del motor, el sistema de transmisión proporciona aproximadamente 1/5 de vuelta en el eje de salida.

### 5.4.3. Controladores

En el mundo del *maker*, existe un factor de forma común para controladores de motores paso a paso de baja corriente (hasta 2A). Debido a esto, la mayoría de los controladores disponibles en el mercado son intercambiables entre sí, aunque varían en la tecnología interna y en sus capacidades técnicas.

En el mercado actual, se pueden encontrar diversos controladores diseñados específicamente para motores paso a paso bipolares. Estos controladores se diferencian en términos de tecnología y características técnicas que ofrecen. Algunos de los controladores más comunes y populares son basados en chips como el A4988, DRV8825 o TMC2208, entre otros. Concretamente, se han seleccionado cuatro de ellos, cuyas especificaciones pueden verse en el Cuadro 5.2.

Cuadro 5.2: Comparación de especificaciones de los controladores existentes

Modelo	Voltaje de alimentación	Corriente máxima	Microstepping	Nivel de ruido	Precio
A4988	8-35V	2A	Hasta 1/16	Muy ruidoso	1€
DRV8825	8.2-45V	2.5A	Hasta 1/32	Ruido aceptable	2€
TMC2225	4.7-36V	2A	Hasta 1/32	Bajo ruido	2.8€
TMC2209	5.5-28V	2.5A	Hasta 1/256	Totalmente silencioso	3.4€



Figura 5.11: Aspecto de los controladores

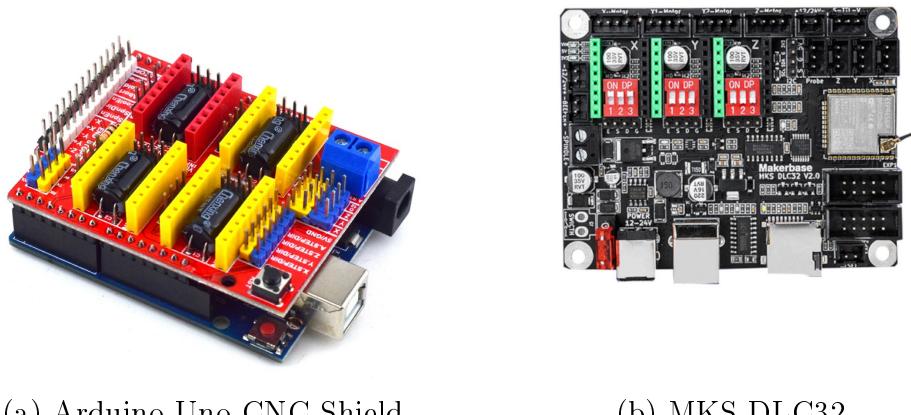
Debido a que se pretende utilizar motores Nema 17 de más de 2A, la decisión debe estar entre el DRV8825 y el TMC2209. Finalmente, se ha elegido este último debido a que es realmente silencioso e incorpora una tecnología superior que garantiza una señal más definida que evita la pérdida de pasos del motor. Además, a la hora de adquirirlo, suele venir acompañado de un disipador de calor más grande que, unido a su mayor eficiencia energética, se traduce en una menor cantidad de calor dentro de nuestro robot.

#### 5.4.4. Placa base CNC

La placa base es una tarjeta de circuito impreso que proporciona conexiones físicas y eléctricas entre los diferentes componentes y unifica estos en un mismo componente.

En el caso de aquellas utilizadas en el mundo de las CNCs, tienen la posibilidad de montar controladores paso a paso y, o bien integran un microcontrolador en la propia placa, o bien tienen la posibilidad de acoplarla a uno concreto (*shields*<sup>6</sup>).

En el mercado existen cantidad de ellas, entre las que destacan principalmente dos: Arduino CNC Shield V3 y MKS DLC32 (Figura 5.12)



(a) Arduino Uno CNC Shield

(b) MKS DLC32

Figura 5.12: Placas base candidatas

Finalmente se ha optado por la segunda ya que la diferencia entre ambas es de a penas 6 euros y en cambio, es muy superior en características. Esta placa incorpora un microcontrolador ESP32 mucho más avanzado y rápido que el AtMega328p usado en el Arduino Uno. Además incorpora un transistor MOSFET de gran potencia que permite controlar una salida de hasta 24v mediante PWM<sup>7</sup>. Por lo general, es una placa mejor acabada, con un hardware más moderno y está pensada para que su montaje eléctrico no de lugar a error.

<sup>6</sup>(placa adicional que se acopla a otras para aumentar su funcionalidad y características)

<sup>7</sup>PWM (Pulse Width Modulation) es una técnica de control que varía el ancho de un pulso eléctrico para regular la potencia entregada a un dispositivo o componente, como motores o luces, permitiendo controlar su velocidad o intensidad de manera eficiente.

### 5.4.5. Fuente de alimentación

Es el elemento encargado de suministrar la energía a todo el sistema eléctrico del robot. La elección de este componente viene dada por las características y necesidades de los componentes elegidos anteriormente.

Para conocer esto, se debe realizar una estimación del consumo en base a las especificaciones que ofrecen los fabricantes. En base a esto, la mayor parte del consumo viene dado por los propios motores que se encontrarán permanentemente excitados para no desplomar el robot. Concretamente, cada uno consume en torno a 7W, haciendo un total de 21W. A esto hay que añadirle el consumo del microcontrolador y la potencia perdida en forma de calor en los controladores (unos 4W). Sumado a ello, se deben tener en cuenta los 3W consumidos por el electroimán. Finalmente, el consumo estimado ronda los 35W.

En base a la estimación, se debe encontrar una fuente de alimentación que sea capaz de dar, al menos, esa potencia de manera continua. En cuanto al voltaje, cualquiera en el rango de 12 a 24v es válido. Se recomienda utilizar una del mayor voltaje posible ya que, en teoría, es más eficiente debido a que la corriente que debe entregar es menor, además de lidiar mejor con las caídas de tensión en movimientos rápidos.

Dicho esto, es posible también alimentar el robot mediante un cargador típico de 19v de un portátil. Aún así, la fuente adquirida para este proyecto es de 24V 150W para poder realizar todas las mediciones de consumos de la Sección 6.3.3, descartando a la fuente como factor limitante. Aún así, el precio de esta fuente es de solo 15€.

### 5.4.6. Rodamientos

Para mejorar la eficiencia de las articulaciones y garantizar que todas rotan con suavidad, se pretende hacer uso de rodamientos en cada una de ellas. Existen diferentes tipos de rodamientos en el mercado pero se ha optado por un tipo de rodamiento concreto para este proyecto. Se pretenden utilizar rodamientos brida como el de la Figura 5.13(a). La razón principal de su uso es su peculiar forma. Este tipo de rodamiento tiene un borde en el extremo que hace de borde que lo hace ideal para insertar en piezas 3D y garantizar que nunca se puedan salir de su sitio mientras exista un tornillo que lo fije (Véase la Figura 5.13(b)).

---

<sup>8</sup><https://es.aliexpress.com/item/32850989216.html>

<sup>9</sup><https://es.aliexpress.com/item/32850989216.html>

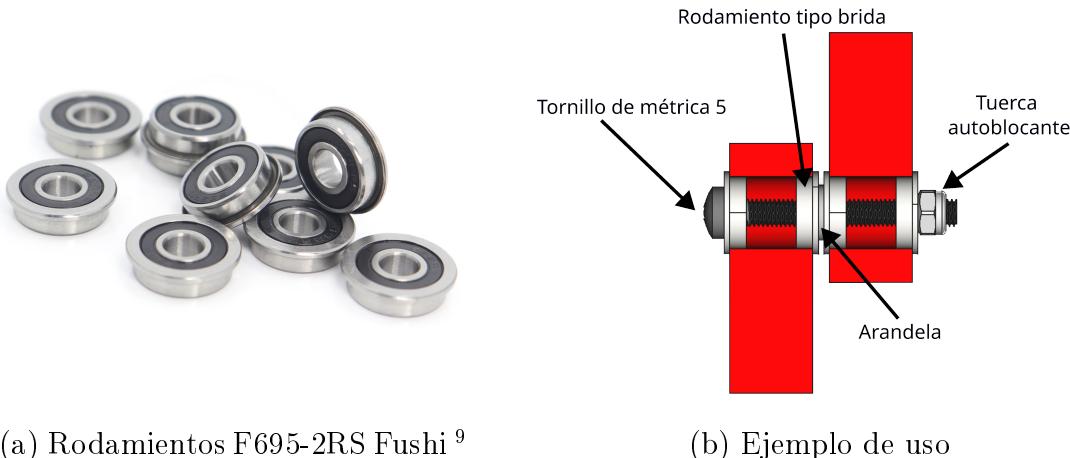


Figura 5.13: Rodamientos de tipo brida

Estos rodamientos tan particulares son ampliamente usados en las poleas pasivas de las impresoras 3D, por lo que son comunes y muy baratos, pudiendo comprar 10 de buena marca por 5€.

#### 5.4.7. Finales de carrera

Este tipo de dispositivos, son usados para conocer el final del recorrido de una articulación y poder establecer así, posiciones absolutas. Existen cantidad de ellos en el mercado con distintos principios de funcionamiento, siendo estos basados en *microswitches*<sup>10</sup> los más baratos. Dentro de esta categoría existen diferentes modelos pero se ha optado por el de la Figura 5.14 por ser el más barato y común. Además son totalmente compatibles con la placa base elegida anteriormente.

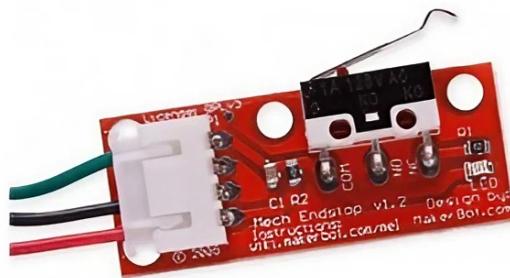


Figura 5.14: Final de carrera económico

---

<sup>10</sup>Interruptor eléctrico de acción rápida y sensible, que se activa al aplicar una ligera presión sobre su botón o palanca.

### 5.4.8. Electroimán

Para llevar a cabo las pruebas, se busca desarrollar una herramienta para el robot, consistente en un electroimán que permita el desplazamiento de objetos metálicos. En el mercado, existen diversos tipos de electroimanes con diferentes tamaños y fuerzas. Uno especialmente adecuado para esta aplicación es el *D20H15* de la Figura 5.15, cuyo nombre da a entender que tiene un diámetro de 20 mm y una altura de 15 mm. Según las especificaciones, es capaz de levantar objetos ferromagnéticos de hasta 3 kg por lo que supera la carga útil que se espera del brazo. Finalmente, se ha adquirido la versión de 24 V y 3W por 3€.



Figura 5.15: Electroimán D20H15 3KG

La lista completa de los componentes y su modelo concreto, así como su precio, se encuentra al final de este capítulo en la Sección 5.6

## 5.5. Diseño CAD

En esta sección se habla de las particularidades del diseño y diversos aspectos a tener en cuenta a la hora de diseñar cualquier pieza mecánica que posteriormente será impresa en 3D.

Para el diseño de este brazo robot, se ha utilizado dos herramientas de diseño. Inicialmente el proyecto se realizó mediante la herramienta *Fusion 360* y posteriormente se utilizó *FreeCAD4.1.4* para cumplir con el objetivo de ser totalmente *Open Source* y estar parametrizado, para que cualquier persona pueda investigarlo, modificarlo y utilizarlo de forma gratuita.

### 5.5.1. Base principal

Es la parte encargada de unir el resto del robot al suelo. Dentro de ella se encuentra la placa base junto con los controladores y toda la electrónica a excepción de la fuente de alimentación. Esto es así debido a que puede ser alimentado de múltiples formas (baterías, cargadores de ordenador, fuentes de PC, salidas de alimentación de robots móviles, etc). Además, esta electrónica debe poder estar refrigerada por un pequeño ventilador incorporado en la propia base.

Se optó por realizar dos piezas circulares y una serie de espaciadores anchos que unían el conjunto, dejando espacio en el interior para la placa base. Este tipo de piezas son muy robustas y fáciles de imprimir.

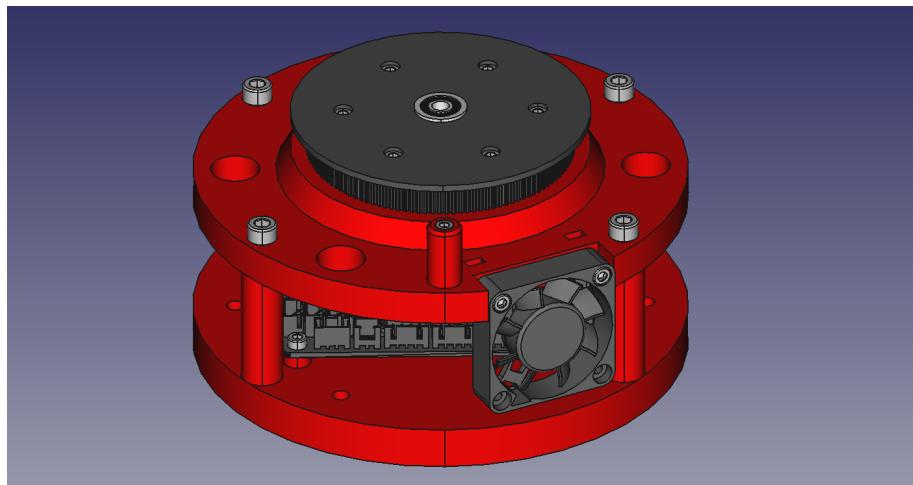


Figura 5.16: Base principal

### Pieza circular superior

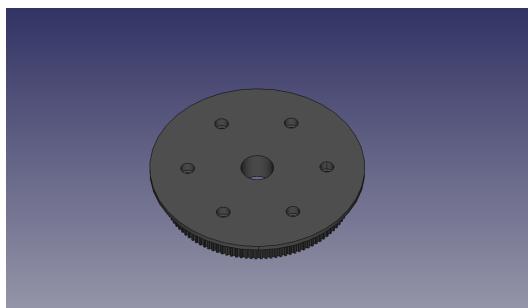
La pieza circular superior (Figura 5.17(b)), incluye una serie de agujeros y huecos hexagonales para insertar las tuercas M3 que permiten acoplar la polea de 120 dientes (Figura 5.17(a)). Además, incluye un rebaje de 40mm en un lateral para poder insertar y atornillar un ventilador 4010.

### Espaciadores

Los espaciadores (Figura 5.17(c)) constan de 4 cilindros perforados con un agujero de 5mm por el que entrarán los tornillos de métrica 5.

### Pieza circular inferior

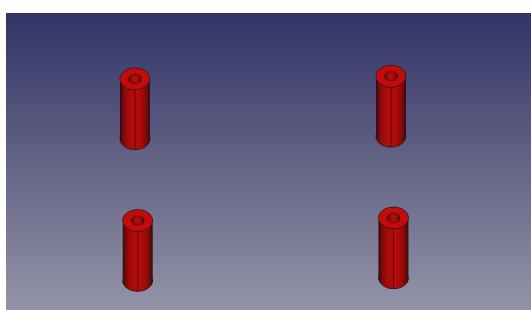
La pieza circular inferior (Figura 5.17(d)), contiene una serie de agujeros para poder atornillar la placa base. Además contiene una serie de agujeros en su perímetro para poder atornillar el robot al suelo.



(a) Polea 120 dientes



(b) Pieza superior



(c) Espaciadores



(d) Pieza inferior

### 5.5.2. Base de los motores

Este conjunto es el encargado de contener los 3 motores y rotar sobre la base principal. Consiste principalmente en 4 piezas: las dos laterales, la inferior y un espaciador de lado a lado que refuerza el conjunto. Está ideado así para solo requerir de piezas planas que serán impresas en dirección horizontal. De esta forma, se logra la mayor resistencia de pieza al incrementar la adhesión entre capas. Además se consigue la máxima exactitud en orificios y formas circulares. Todo el conjunto se une a través de varillas roscadas y tuercas (Figura 5.17).

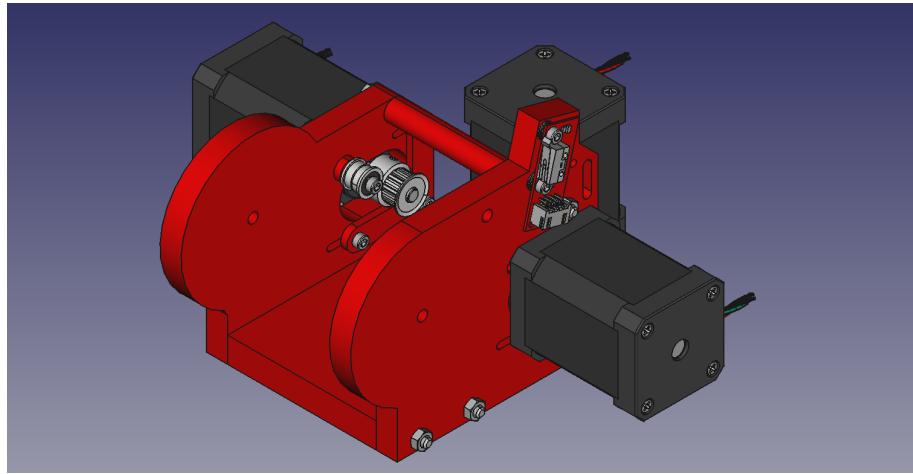


Figura 5.17: Base de los motores

Además de esto, cabe destacar el sistema de tensado de las correas, formado por una serie de orificios longitudinales que permiten deslizar el motor hasta lograr la tensión óptima, para después fijarse en su sitio. Esto último se realiza mediante una pieza que aumenta la fricción y reparte la presión de los 3 tornillos empleados.

Por otra parte, esta pieza incorpora una serie de orificios específicamente creados para acoplar de los 3 finales de carrera del robot. Sumado a ello, se ha decidido añadir una serie de ranuras para poder realizar a *posteriori* un manejo de cables más ordenado.

### 5.5.3. Paralelogramos

Son aquellos elementos descritos en el modelo alámbrico, los cuales han sido modelados en 3D y son los encargados de llevar el movimiento de la base de los motores al extremo del robot.

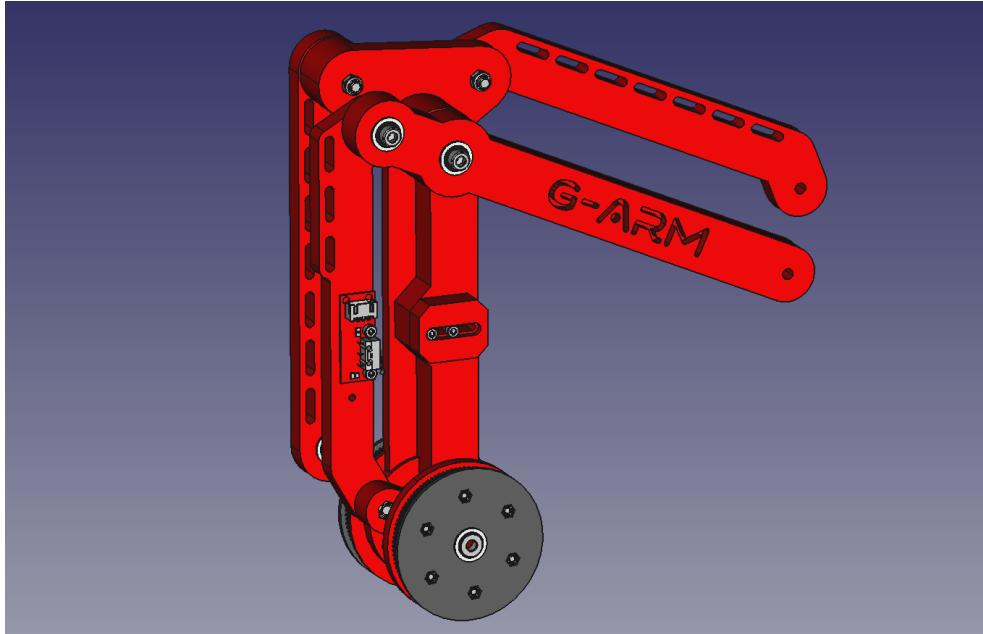


Figura 5.18: Conjunto de los paralelogramos

Como se puede apreciar, algunos de ellos incluyen (al igual que la base de los motores), las ranuras necesarias para poder guiar los cables que van hasta el extremo del robot. Además de esto, incorpora el final de carrera en un eslabón, de tal forma que se active al llegar al final de su recorrido. Para poder ajustar este punto final, se ha creado el pequeño bloque deslizante de su derecha.

Todas y cada una de las articulaciones incorpora un par de rodamientos brida (presentados en la Sección 5.4.6). Estos, deberán ser insertados a presión en las piezas reales, por lo que los orificios son ligeramente inferiores al diámetro del rodamiento.

Por otro lado, para la realización de las poleas dentadas GT2, se ha empleado la herramienta paramétrica gt2-gear-generator<sup>11</sup> para generar el contorno de la polea en formato DXF (Figura 5.19). Este formato ha sido importado en FreeCAD y extruido gracias a los espacios de trabajo *Draft* y *Part Design*.

---

<sup>11</sup>[https://avtechnik.github.io/gt2-gear-genarotor/](https://avtechnik.github.io/gt2-gear-genarator/)

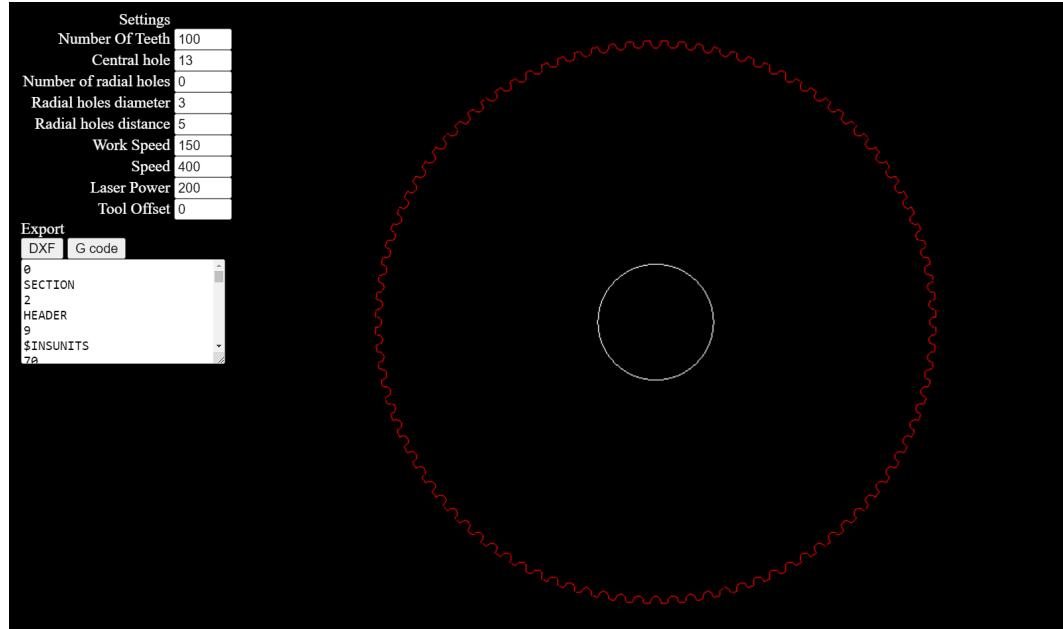


Figura 5.19: Contorno de la polea de 100 dientes

Posteriormente, se les ha incluido los agujeros y vaciados hexagonales necesarios para insertar las tuercas de métrica 3, que las unirán a su respectivo eslabón. Mostrados en la Figura 5.20.

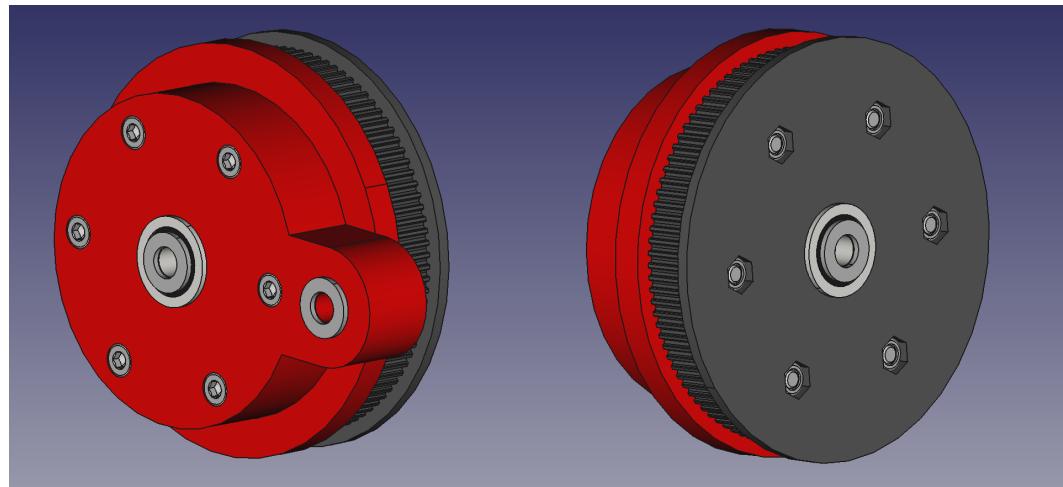


Figura 5.20: Ambos lados de la palanca de la articulación 3

Finalmente, con el objetivo de personalizar el aspecto del robot, se ha incluido su nombre en el lateral del eslabón más visible.

### 5.5.4. Elemento terminal

Esta pieza (Figura 5.21) está situada en el extremo del robot. Es la encargada de realizar la unión entre el robot y la herramienta. Por esto, se ha ideado con una forma particular que permite acoplar herramientas de una forma sólida e inequívoca.

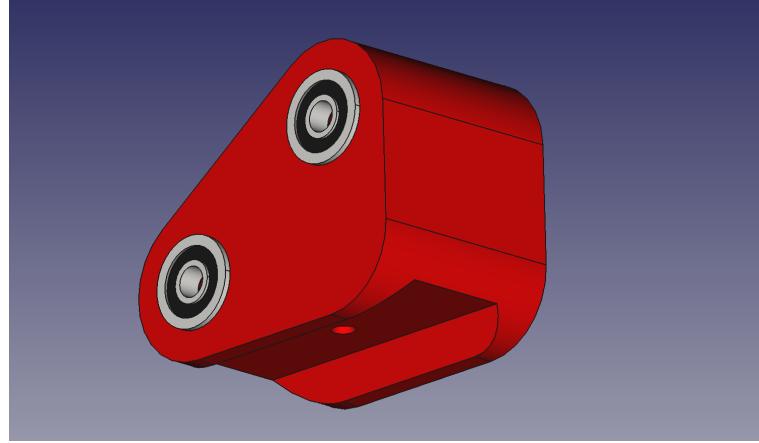


Figura 5.21: Elemento terminal

Consiste en una acanaladura a 45 grados (Figura 5.22) con un agujero que la atraviesa, por el que pasará el tornillo unirá ambas piezas. Este acople, diseñado específicamente para este proyecto, es sencillo de usar e impide que la herramienta rote o tenga algún tipo de holgura. De hecho, las paredes están en ángulo para obligar a la herramienta a centrarse en la acanaladura y hacer la unión muy robusta y precisa.

### 5.5.5. Herramienta electroimán

Para dotar al robot de una utilidad, se ha creado una herramienta que contiene el electroimán (Figura 5.23(a)). Esta herramienta debe de tener la forma de la acanaladura anterior para poder encajar en el robot. Además se ha redondeado las esquinas para que visualmente se adapte mejor a la forma del extremo del robot. Para unir el conjunto se ha utilizado el propio orificio roscado del electroimán, como se puede ver en la Figura 5.23(b)

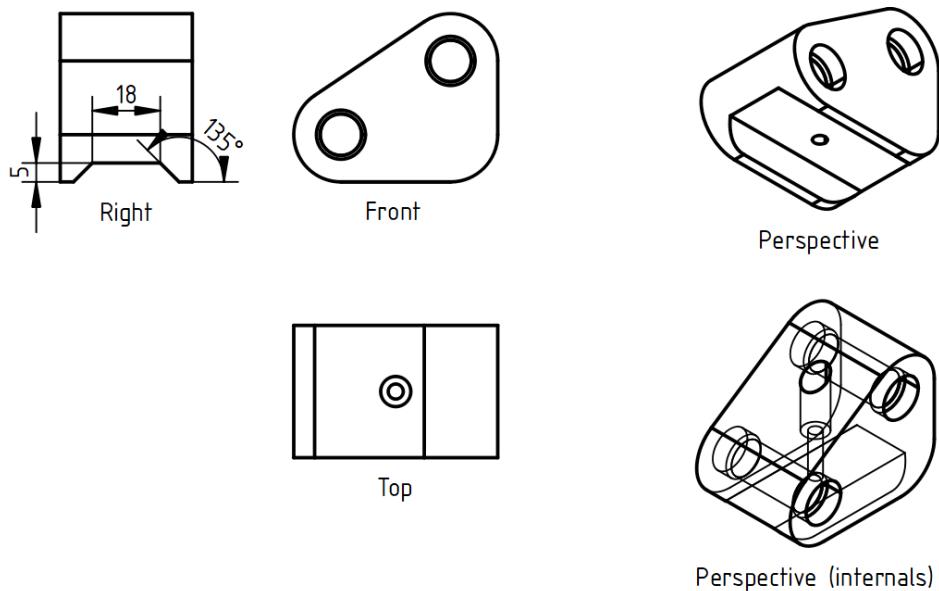


Figura 5.22: Vistas del elemento terminal

## 5.6. Impresión y montaje

En esta sección se exponen todos los detalles a tener en cuenta a la hora de querer replicar este proyecto. Para la impresión de G-Arm se ha utilizado una impresora Ender-3 Pro<sup>12</sup> y un rollo de 1Kg de filamento PLA Rojo convencional.

Con la intención de lograr la mejor organización posible, se ha decidido incluir una serie de tablas detallando los componentes necesarios para el montaje del robot. En la Tabla 5.3 se listan aquellos componentes que deben ser comprados y su precio. Se recomienda adquirirlos a través de páginas como *Aliexpress* ya que resulta significativamente más barato que comprarlos a través de *Amazon*. Por otro lado, es necesaria la tornillería de las Tablas 5.4, 5.5, 5.6 y 5.7, comprada en ferreterías locales y nacionales como *Randrade* (Online). El resto de piezas detalladas en la Tabla 5.8, son enteramente imprimibles en 3D. Se recomienda configurar el laminador con las densidades de relleno propuestas en dicha tabla. Cada pieza está diferenciada con un número y se pueden requerir varias unidades de una misma pieza (indicado en la propia tabla). Se recomienda utilizar una altura de capa de 0.12mm y tres líneas de grosor de pared.

---

<sup>12</sup><https://www.creality.com/products/ender-3-pro-3d-printer>

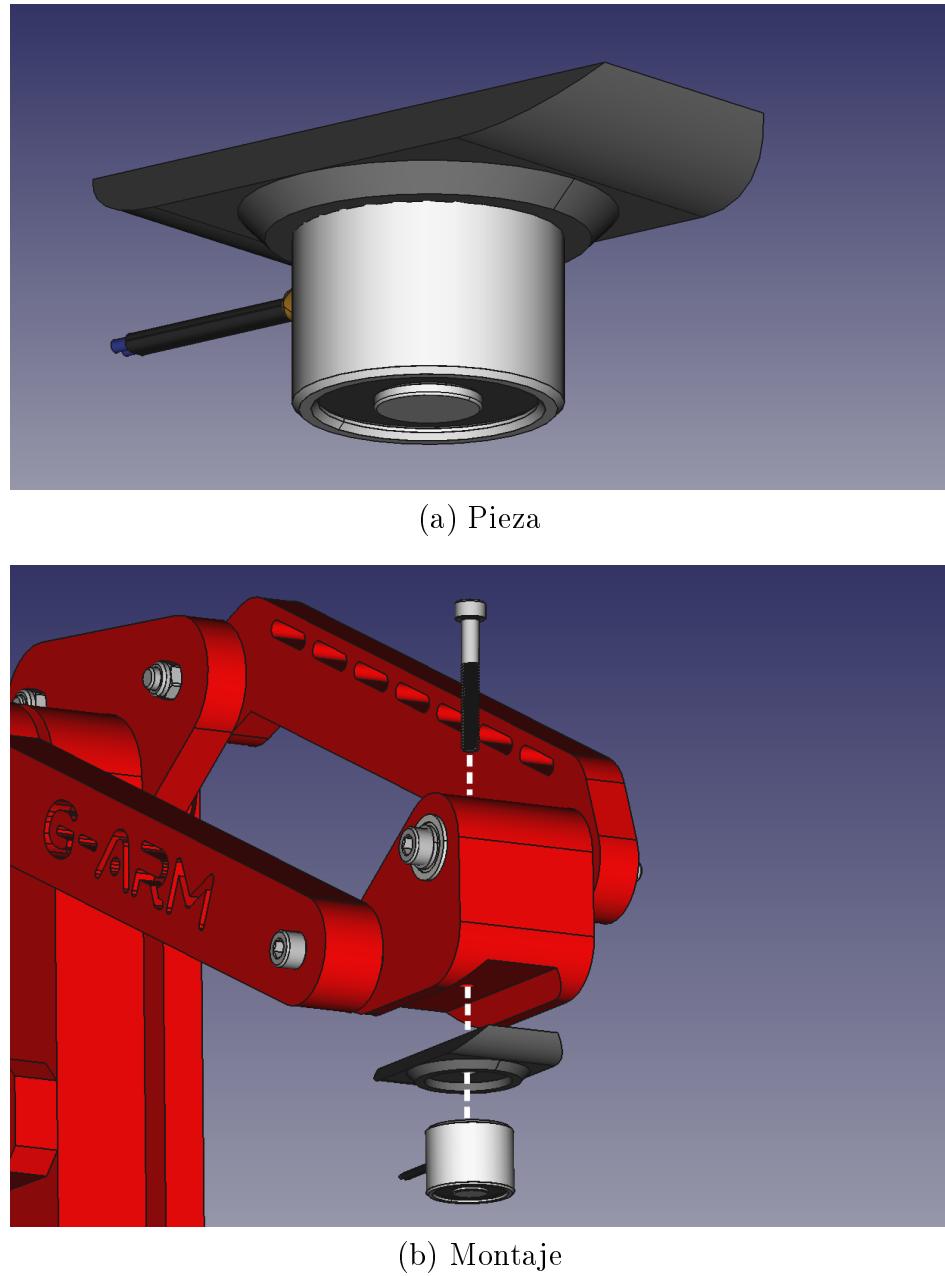


Figura 5.23: Herramienta electroimán

Componente	Modelo	Cantidad	Precio total
Motor Nema 17	17HS24-2104S	3	56€
Controlador	TMC2209	3	10€
Placa base	MKS DLC32	1	16€
Final de carrera	MakerBot (rojo)	3	5€
Fuente de alimentación	24V 5A (opcional)4.2.4	1	15€
Rodamiento	F695-2RS Fushi	22	15€
Rodamiento	F623RS Fushi	6	5.5€
Polea GT2	Correa:6mm ID:5mm	3	1.5€
Correa GT2	Correa:6mm Largo:252mm	2	3.5€
Correa GT2	Correa:6mm Largo:280mm	1	1.8€
Ventilador	24V 4010	1	2€
Electroimán	D20H15mm 3KG 24V	1	3€
Plástico para imprimir	PLA/PETG 1Kg	1	22€

Cuadro 5.3: Componentes hardware necesarios

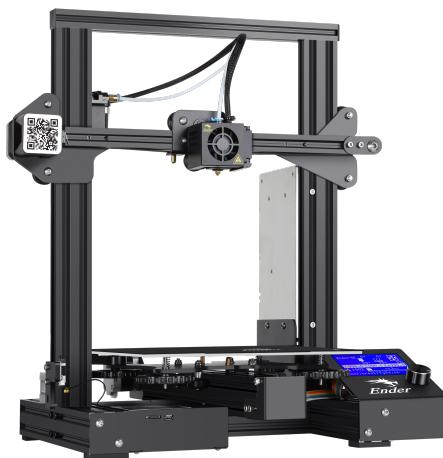


Figura 5.24: Ender-3 Pro V1 2017

Métrica	Tamaño	Cantidad
M3	12mm	8
M3	16mm	11
M3	20mm	10
M3	25mm	13
M4	30mm	3
M5	25mm	1
M5	30mm	3
M5	40mm	1
M5	50mm	5
M5	55mm	1
M5	60mm	2

Cuadro 5.4: Tornillos necesarios

Métrica	Cantidad
M3 normal	6
M3 autoblocante	29
M4 autoblocante	8
M5 normal	5
M5 autoblocante	10

Cuadro 5.5: Tuercas necesarias

Métrica	Cantidad
M3	29
M5	16

Cuadro 5.6: Arandelas necesarias

Métrica	Cantidad
M3	1 metro
M4	1 metro
M5	1 metro

Cuadro 5.7: Varillas roscadas necesarias

Identificador	Cantidad	Relleno óptimo
#1 Base inferior	1	15 %
#2 Espaciador de la base	4	100 %
#3 Base superior	1	15 %
#4 Polea 120T	1	20 %
#5 Base de los motores	1	15 %
#6 Límite de la base	1	100 %
#7 Lateral derecho	1	15 %
#8 Lateral izquierdo	1	15 %
#9 Espaciador de rodamiento	3	100 %
#10 Polea 100T	2	20 %
#11 Eslabón 1	1	15 %
#12 Espaciador de motores	1	100 %
#13 Tope de correa	1	15 %
#14 Palanca articulación 2	1	20 %
#15 Paralelo eslabón 1	1	15 %
#16 Paralelo del extremo	1	15 %
#17 Codo	1	15 %
#18 Ajuste de límite	1	15 %
#19 Eslabón 2	1	15 %
#20 Paralelo eslabón 2	1	15 %
#21 Extremo del robot	1	15 %
#22 Fijador de motores	2	100 %
#23 Herramienta electroimán	1	15 %

Cuadro 5.8: Piezas necesarias

El precio total de los componentes necesarios es de 156.3€ sumado a unos 15 en tornillería, hacen un coste total aproximado de 171€.

La impresión de todas las piezas puede llevar cerca de 60 horas. En cambio, el montaje requiere de apenas 2, o incluso, menos dependiendo de la habilidad del usuario. Para saber la posición de cada pieza y dónde montarla, se recomienda consultar el montaje completo<sup>13</sup> realizado mediante A2Plus en FreeCAD (Figura 5.25). En él, se puede ver el nombre de cada pieza, tornillo necesario y posición.

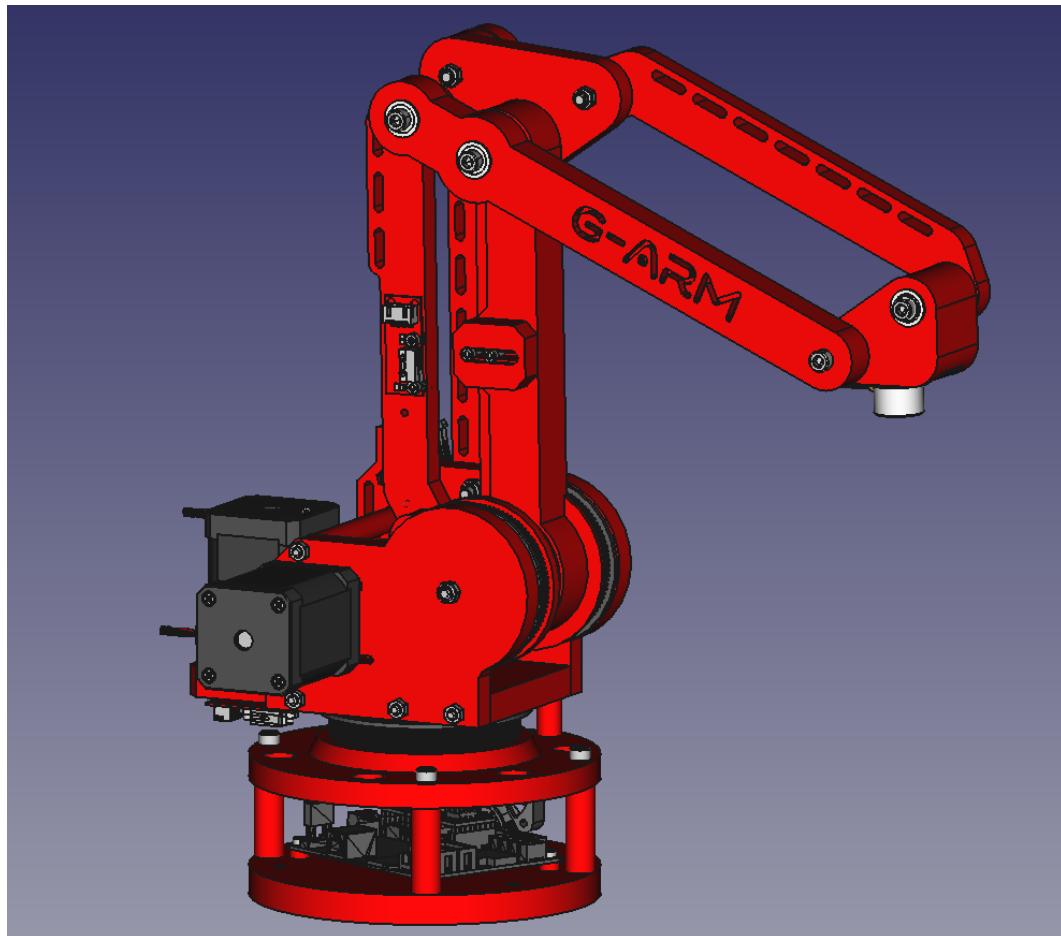


Figura 5.25: Ensamble CAD completo del robot final

En cuanto al montaje de la electrónica es realmente sencillo debido a que existen numerosos tutoriales y manuales en internet que utilizan esta placa. Por si no fuera poco, el propio circuito impreso de la placa tiene serigrafiado un nombre en cada conector. El único aspecto a tener en cuenta es la regulación de corriente de los 3 controladores TMC2209. Esta se realiza mediante un pequeño potenciómetro y requiere de un multímetro para leer los valores. En este enlace se muestra un tutorial de como hacerlo.

---

<sup>13</sup>[https://github.com/RoboticsURJC/tfg-vperez/blob/280861172bce3b1c0cfbb155a434364ea68eeb30/src/design/FreeCad/%230\\_ASSEMBLY.FCStd](https://github.com/RoboticsURJC/tfg-vperez/blob/280861172bce3b1c0cfbb155a434364ea68eeb30/src/design/FreeCad/%230_ASSEMBLY.FCStd)

En la Figura 5.26 se muestra el resultado real del robot. Como es obvio es idéntico al realizado en el diseño CAD y resulta robusto y estable. La organización de los cables ha resultado sencilla debido a las ranuras añadidas en la fase de diseño.

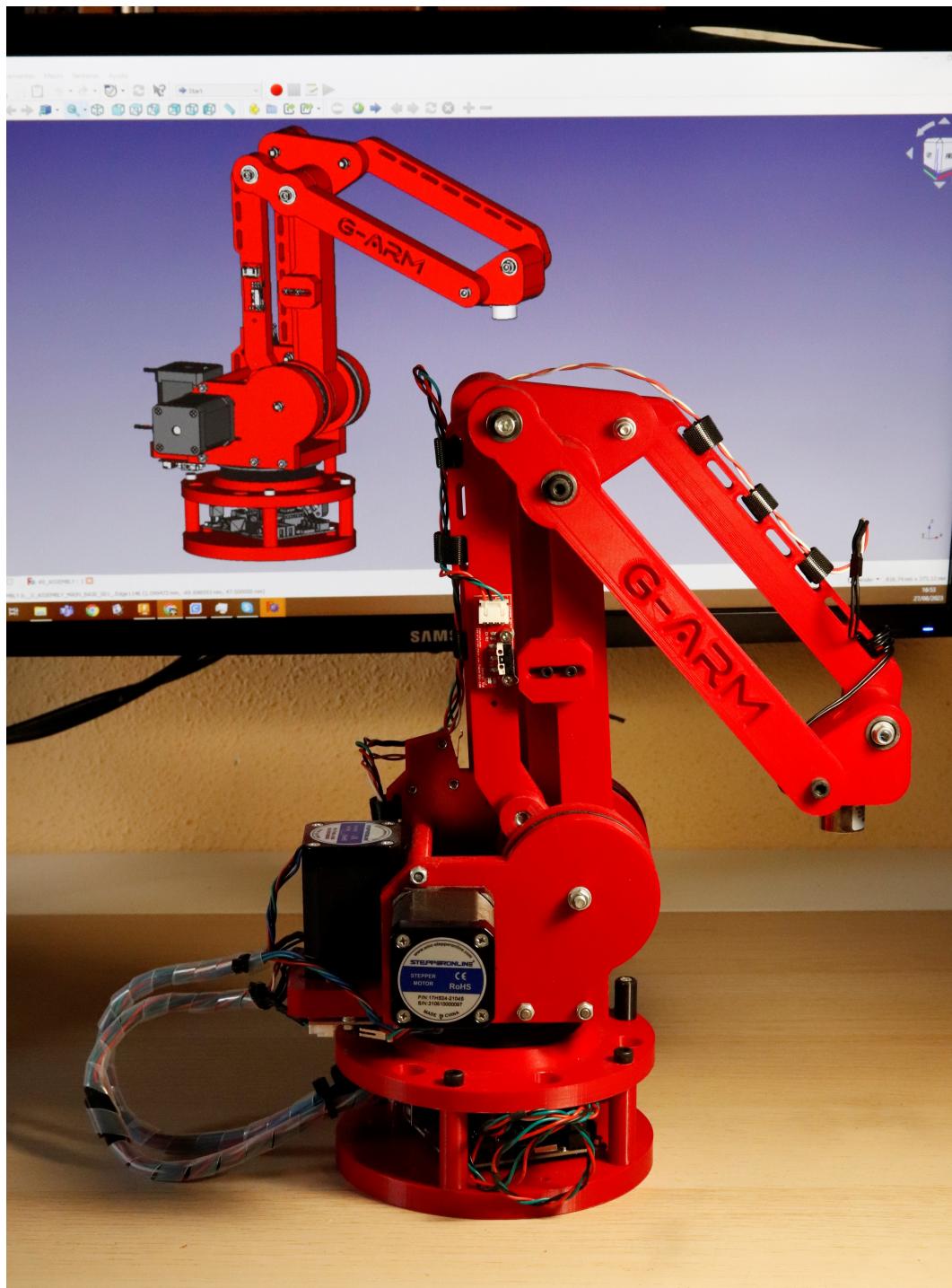


Figura 5.26: Montaje real del robot

---

# Capítulo 6

# Desarrollo software del manipulador

---

En este capítulo se aborda el desarrollo software necesario para poder usar el robot realizado tanto en simulaciones como en el mundo real.

## 6.1. Control de los actuadores

En esta sección se concretan los mecanismos y herramientas software utilizadas para poder controlar, desde un ordenador cualquiera, el hardware creado en el anterior capítulo.

### 6.1.1. Grbl como *firmware* del robot

Para realizar el control de los actuadores se ha tomado la decisión de utilizar el firmware de control numérico Grbl (Sección 4.1.2), por encima de realizar uno propio, debido a que es una solución robusta con años de desarrollo y con unas ciertas garantías difíciles de superar.

Actualmente, este firmware permite controlar hasta tres motores paso a paso simultáneamente. Además, tiene la posibilidad de utilizar un canal PWM (usualmente usado en las CNC para modificar la velocidad de la herramienta). Sobre estos motores, podemos realizar un control en posición, es decir, recibe una serie de coordenadas y automáticamente alcanza esos puntos. Este tipo de controlador es perfectamente válido para esta aplicación y simplifica bastante su uso. Aunque esta placa en concreto viene con Grbl 1.1 instalado de fábrica, existen numerosas guías en Internet acerca de como instalarlo.

Para que Grbl sepa qué posición debe tener cada motor en un momento dado, necesita recibir una serie de datos codificados a través de una de sus interfaces. Los detalles esta la comunicación se presentan en la siguiente subsección.

### 6.1.2. Comunicación con Grbl

En esta sección se exponen las distintas opciones de comunicación con el robot, así como el formato de mensaje que se debe emplear.

Debido a las características hardware de la placa utilizada, se puede establecer comunicación con Grbl a través de 2 interfaces diferentes. La primera, es a través del propio puerto USB integrado, utilizando el protocolo serie UART. Este modo de comunicación tiene la ventaja de ser rápido pero te fuerza a estar conectado al robot a través de un cable. Por otro lado, debido a las características técnicas del microcontrolador, se puede establecer una comunicación inalámbrica a través de Wifi utilizando el protocolo Telnet. Finalmente, se ha optado por utilizar la interfaz USB debido a que es más rápida y permite al usuario estar conectado a internet, en vez de a la propia red *offline* que crea la placa.

Una forma de poder saber si viene ya instalado en la placa que hemos comprado, es utilizar una terminal serie, como puede ser *Cutecom*. Establecemos una velocidad de 115200 baudios y nos conectamos al puerto que aparezca disponible al conectar la placa al ordenador. Si está instalado, veremos una serie de mensajes que indican la versión de Grbl y las distintas configuraciones internas. (Figura 6.1)



Figura 6.1: Terminal gráfica de Cutecom al conectarse a una placa con Grbl

Grbl recibe instrucciones de G-Code4.1.3 a través del puerto serie. Aunque existen una gran cantidad de ellas, solo ha sido necesario usar algunas:

Comando	Explicación
G92 X0 Y0 Z0	G92 establece los valores de posición de cada eje
G01	Establece que todos los ejes se moverán a la vez de forma lineal
G90 X1 Y2 Z3 F34	Mueve los motores a velocidad 34 hasta una coordenada absoluta
S500	Establece la velocidad del motor auxiliar a 500. Rango: 0-1000
M3	Enciende el motor auxiliar
M5	Apaga el motor auxiliar
?	Recibir el estado actual de la maquina
!	Para el movimiento actual
~	Continua con el movimiento parado anteriormente

Cuadro 6.1: Comandos utilizados en la realización del proyecto

Con el objetivo de abstraer al futuro software del robot de esta comunicación, se ha implementado una clase en Python4.1.1 llamada *Grbl*. En este software, utiliza la librería PySerial<sup>1</sup> para establecer una comunicación serie. La clase *Grbl* creada puede ser utilizada a través de los métodos mostrados en el Código 6.1. Y su código fuente puede ser encontrado en el repositorio del proyecto<sup>2</sup>.

---

```

start(puerto) # Devuelve true/false en función de si ha sido posible
    conectarse
stop() # Finaliza la comunicación
setSpindleSpeed(value) # Establece la velocidad del motor auxiliar
enableSpindle() # Activa el motor auxiliar
disableSpindle() # Desactiva el motor auxiliar
setCoordinates(x, y, z) # Establece las coordenadas x, y, z
getXYZ() # Devuelve la posición actual
asyncXYZMove(posición, velocidad, relative) # Mueve los motores a una
    posición con la velocidad dada. Permite realizar movimientos relativos y
    absolutos.

```

---

Código 6.1: API de la clase *Grbl*

A pesar de que ya podemos comandar movimientos en los distintos ejes de la máquina, se requiere realizar una serie de configuraciones para definir las características concretas de cada articulación.

<sup>1</sup><https://pypi.org/project/pyserial/>

<sup>2</sup>[https://github.com/RoboticsURJC/tfg-vperez/blob/96fc1e44bef6a31c272fb0673b5a33a7571c5ee7/src/software/g\\_arm/g\\_arm/g\\_arm\\_lib/grblAPI.py](https://github.com/RoboticsURJC/tfg-vperez/blob/96fc1e44bef6a31c272fb0673b5a33a7571c5ee7/src/software/g_arm/g_arm/g_arm_lib/grblAPI.py)

### 6.1.3. Configuración de Grbl para su uso en robótica

Grbl tiene ciertas limitaciones a la hora de usarse en robótica. Es normal, debido a que está pensado para controlar máquinas CNC de 3 ejes prismáticos. Pese a esto, se pueden unas ciertas configuraciones para adaptarlo a esta aplicación.

#### Parámetros de Grbl

Este *firmware* guarda en su memoria interna una serie de parámetros que definen su comportamiento. Si se quiere visualizar cuáles son y sus respectivos valores, hay que enviar el mensaje **\$\$** a través del puerto serie. Para modificar un cierto valor, hay que introducir una cadena con el formato: **\$identificador del parámetro=nuevoValor**.

Para modificar la configuración de una forma cómoda, se recomienda utilizar herramientas como *UniversalGcodeSender*<sup>3</sup> (Figura 6.2).

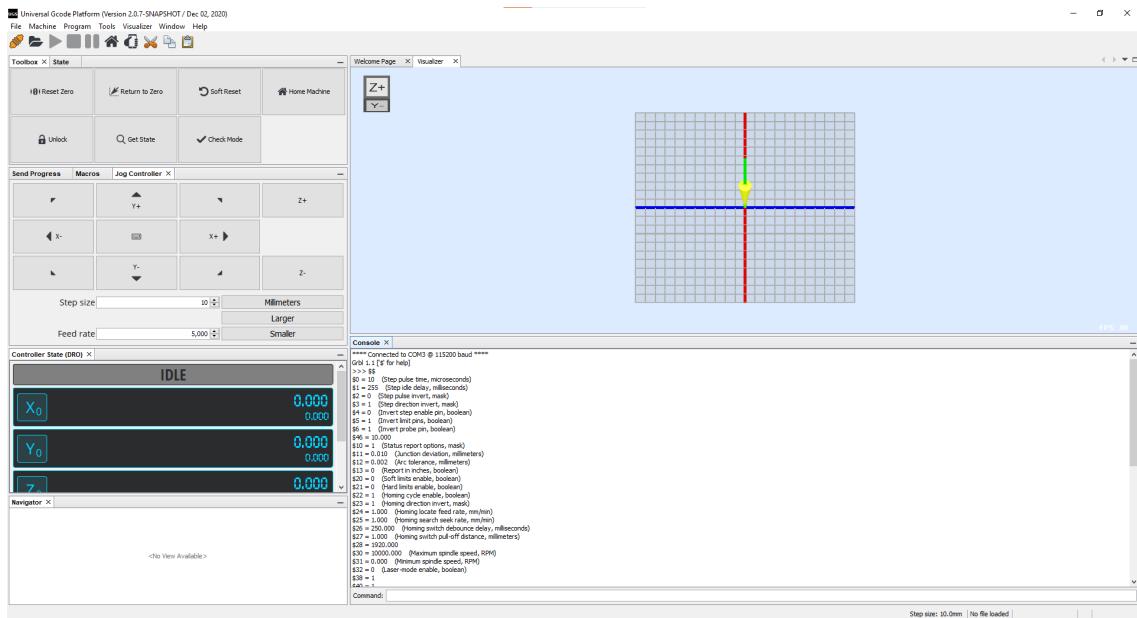


Figura 6.2: Interfaz de UniversalGcodeSender

Puesto que se quiere utilizar Grbl para controlar un brazo robot, debemos realizar las siguientes configuraciones:

- **\$1:** Retardo o tiempo de espera entre pulsos de paso cuando el motor está inactivo (en milisegundos).

Debemos configurar este parámetro en su valor máximo, en este caso 255. Este valor tiene un significado especial, haciendo que los motores paso a paso se mantengan energizados constantemente aunque no se estén moviendo. Es de vital

<sup>3</sup>[https://winder.github.io/ugs\\_website/download/](https://winder.github.io/ugs_website/download/)

importancia para evitar que el brazo se desplome al detenerse en una cierta posición.

- **\$100, \$101 y \$102:** Indican el número de pasos por unidad de movimiento para los ejes X, Y, Z respectivamente.

Por defecto está pensado para utilizar pasos por milímetro. Como se pretende utilizar articulaciones de rotación, debemos expresar esta relación en función de alguna medida angular. La unidad a utilizar podría ser: grados, radianes o vueltas, entre otras. En este trabajo se utilizan los grados debido a que en radianes y vueltas la unidad correspondía a un gran número de pasos y era difícil controlar la aceleración para incrementos de 0.1 vueltas.

$$\text{PasosPorGrado} = \frac{\text{Microstepping} * \text{Ratio}}{1,8^\circ}$$

Ecuación 6.1: Cálculo de pasos por grado en Grbl

- **\$110, \$111 y \$112:** Indican la velocidad máxima a la que puede moverse cada eje X, Y, Z en unidades por segundo. En este caso, grados por segundo. Estos valores se deben encontrar por medio de la experimentación. Se trata de una medida de seguridad en caso de que el usuario quiera mover demasiado rápido un eje pudiendo dañar el brazo.
- **\$120, \$121 y \$122:** Indican la aceleración máxima a la que puede moverse cada eje X, Y, Z en unidades por segundo cuadrado. En este caso, grados por segundo cuadrado. Estos valores también se deben encontrar por medio de la experimentación. Se trata de una medida de seguridad en caso de que el usuario quiera mover demasiado rápido un eje pudiendo dañar el brazo. Se debe encontrar una aceleración idónea para todos los movimientos, una limitación de grbl es que no se puede comandar un movimiento diciéndole una determinada aceleración.

Para profundizar más en la finalidad de cada parámetro, código y configuración se recomienda leer la documentación <sup>4</sup> <sup>5</sup>

---

<sup>4</sup><https://github.com/gnea/grbl/blob/master/doc/markdown/commands.md>

<sup>5</sup><https://github.com/gnea/grbl/blob/master/doc/markdown/settings.md>

En el caso concreto de este proyecto, se ha utilizado la configuración mostrada en la Tabla 6.2.

Parámetro	Valor
\$0	10 (Step pulse time, microseconds)
\$1	255 (Step idle delay, milliseconds)
\$2	0 (Step pulse invert, mask)
\$3	1 (Step direction invert, mask)
\$4	0 (Invert step enable pin, boolean)
\$5	1 (Invert limit pins, boolean)
\$6	1 (Invert probe pin, boolean)
\$10	1 (Status report options, mask)
\$11	0,010 (Junction deviation, millimeters)
\$12	0,002 (Arc tolerance, millimeters)
\$13	0 (Report in inches, boolean)
\$20	0 (Soft limits enable, boolean)
\$21	0 (Hard limits enable, boolean)
\$22	1 (Homing cycle enable, boolean)
\$23	1 (Homing direction invert, mask)
\$24	1,000 (Homing locate feed rate, mm/min)
\$25	1,000 (Homing search seek rate, mm/min)
\$26	250,000 (Homing switch debounce delay, milliseconds)
\$27	1,000 (Homing switch pull-off distance, millimeters)
\$30	10000,000 (Maximum spindle speed, RPM)
\$31	0,000 (Minimum spindle speed, RPM)
\$32	0 (Laser-mode enable, boolean)
\$100	26,666 (X-axis travel resolution, step/mm)
\$101	22,222 (Y-axis travel resolution, step/mm)
\$102	22,222 (Z-axis travel resolution, step/mm)
\$110	6000,000 (X-axis maximum rate, mm/min)
\$111	6000,000 (Y-axis maximum rate, mm/min)
\$112	6000,000 (Z-axis maximum rate, mm/min)
\$120	60,000 (X-axis acceleration, mm/sec^2)
\$121	60,000 (Y-axis acceleration, mm/sec^2)
\$122	60,000 (Z-axis acceleration, mm/sec^2)
\$130	450,000 (X-axis maximum travel, millimeters)
\$131	450,000 (Y-axis maximum travel, millimeters)
\$132	50,000 (Z-axis maximum travel, millimeters)

Cuadro 6.2: Parámetros Grbl usados en este trabajo

## 6.2. Integración con ROS 2

En esta sección se detalla el proceso de integración del robot G-Arm en ROS, pasando por la creación de la descripción, la visualización, la configuración y la simulación.

### 6.2.1. Descripción del robot

El primer paso en la integración de un robot en el ecosistema ROS, es lograr expresar su forma y funcionamiento de tal manera que pueda ser visualizado, simulado y controlado en el mundo virtual. Esto es conocido como describir un robot y para ello existen diferentes formatos entre los que destacan URDF y Xacro.

#### Creación del paquete de descripción

Para crear el paquete, se ha seguido el siguiente

1. En el *src* del workspace ejecutamos:

```
ros2 pkg create --build-type ament_cmake g_arm_description
```

2. Dentro de la carpeta que se ha generado, creamos 3 nuevos directorios:

```
mkdir launch rviz urdf meshes
```

3. Añadimos al fichero *package.xml*:

---

```
1 <exec_depend>joint_state_publisher</exec_depend>
2 <exec_depend>robot_state_publisher</exec_depend>
3 <exec_depend>rviz</exec_depend>
4 <exec_depend>xacro</exec_depend>
```

---

4. Compilamos el paquete desde la raíz del workspace:

```
colcon build --symlink-install
```

5. Añadimos la siguiente línea al final del .bashrc para que ROS pueda encontrar el paquete:

```
source ~/workspace/install/local_setup.bash
```

## Describir un robot mediante URDF y Xacro

*Unified Robot Description Format* (URDF) es un formato de archivo cuyo propósito es describir la estructura, cinemática y aspecto de un robot. Se trata de un estándar ampliamente utilizado en la comunidad robótica, especialmente en ROS.

En este tipo de archivo, se especifica la geometría del robot mediante la definición de eslabones (links) y articulaciones (joints). Cada eslabón es descrito por su aspecto y geometría, mientras que las articulaciones están definidas en cuanto a su tipo, recorrido y posición. Además de esto, un archivo URDF puede incluir información sobre la masa y la inercia de los eslabones, así como como texturas y modelos 3D.

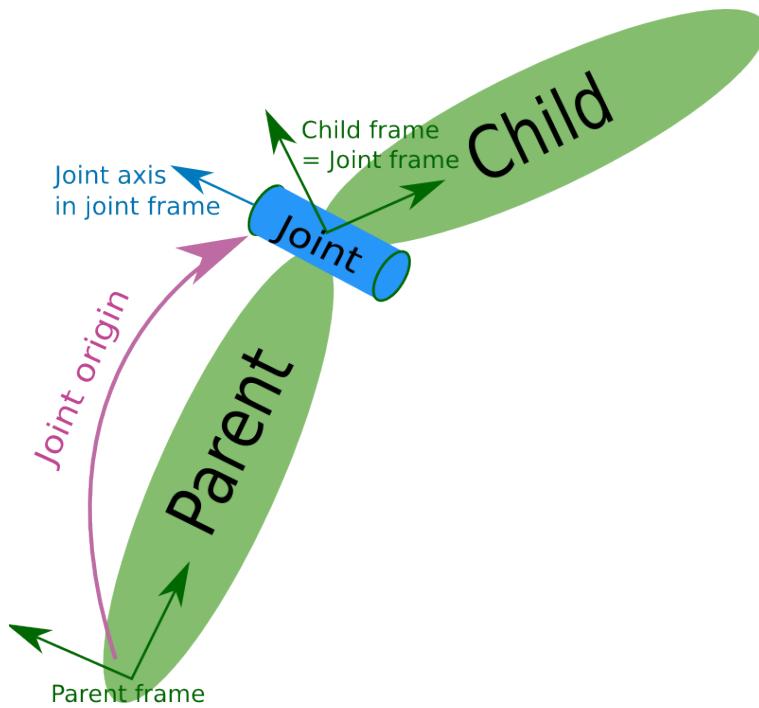


Figura 6.3: Elementos del formato URDF

Este formato se basa en el lenguaje *eXtensible Markup Language* (XML), lo que permite describir el robot de una forma estructurada y legible. Por otro lado, Xacro es un lenguaje de macros XML que simplifica la creación de descripciones URDF al permitir la reutilización de código y la parametrización de modelos. Se puede convertir un fichero Xacro a URDF ejecutando el siguiente comando:

```
xacro fichero.xacro > fichero.urdf
```

Con el objetivo de lograr una descripción lo más realista posible, se han empleado las mismas geometrías del diseño CAD. Para poder incluirlas en el URDF, es necesario exportar las piezas al formato *Collada* (.dae) desde FreeCAD. Este formato contiene información acerca de la geometría, posición, tamaño y colores, de la pieza en cuestión. Además del aspecto visual, hay que definir el volumen físico que ocupan en el espacio. Las mallas de colisión son la representación geométrica simplificada utilizada para calcular interacciones físicas y detectar colisiones en simulaciones. En este caso se va a utilizar un formato muy liviano de malla, el *Standard Tessellation Language* (STL).

Tanto los ficheros Collada como los ficheros STL, deben estar en la carpeta *meshes* creada anteriormente. En cambio; el fichero URDF debe estar en la carpeta *urdf*.

Un aspecto a tener en cuenta, es que no todos los tipos de robot pueden ser descritos con este formato. La limitación radica en que un eslabón hijo solo soporta un eslabón padre, por lo que no se pueden construir cadenas cinemáticas cerradas.



Figura 6.4: Cadenas cinemáticas

El robot de este trabajo esta constituido enteramente por cadenas cinemáticas cerradas. Pese a esto, se ha utilizado el ingenio y un tipo de articulación específico para lograr describir este tipo de robot de forma exacta. Se llama *Mimic joint* y es un tipo de articulación pasiva que imita el ángulo de otra. Con una cierta combinación de varias de ellas, y una serie de eslabones invisibles se puede lograr describir este tipo de robot. Además, ha sido necesario añadir una cuarta articulación «falsa». Esta indica la intensidad del electroimán acoplado al robot. Es de tipo revolución, y su recorrido es de un radián.

El fichero<sup>6</sup> URDF final puede ser encontrado junto al resto del proyecto en el repositorio de Git.

---

<sup>6</sup>[https://github.com/RoboticsURJC/tfg-vperez/blob/8d3b281d701ec4a57602a9dcfcdd04888955319b/src/software/g\\_arm\\_description/urdf/robot\\_electromagnet.urdf](https://github.com/RoboticsURJC/tfg-vperez/blob/8d3b281d701ec4a57602a9dcfcdd04888955319b/src/software/g_arm_description/urdf/robot_electromagnet.urdf)

### Creación de un launcher para visualizar el robot

Para visualizar el fichero que hemos creado anteriormente, podemos realizar una herramienta de ROS llamada RViz. Esta herramienta no solo permite visualizar, sino también analizar datos de sensores, modelos de robot y planificaciones de movimiento de forma interactiva. Además se le pueden añadir complementos para agregarle funcionalidades extra.

Para que se pueda cargar la descripción del robot en este visualizador, debe haber algo que la esté publicando en el topic *robot\_description*. Es por esto que existe un nodo de ROS llamado *robot\_state\_publisher* que se encarga de ello. Además, si queremos algo más que una representación estática, podemos utilizar un nodo llamado *joint\_state\_publisher\_gui*. Este nodo muestra una ventana gráfica con una serie de controles para controlar manualmente cada joint.

Teniendo en cuenta que se requiere lanzar una serie de nodos y programas, es útil crear un *launcher* de ROS. Un *launcher* es una herramienta que se utiliza para iniciar, configurar, gestionar y desplegar nodos. Este tipo de fichero terminado en *.launch.py* se almacenan en la carpeta *launch*. Para crearlo, se ha usado como inspiración el lanzador *visualize\_franka.launch.py* del robot Franka Emika<sup>7</sup>.

Finalmente, el paquete de descripción está terminado. Ejecutando el siguiente comando, se nos abrirá una ventana de RViz y una serie de controles deslizantes para cada articulación (Figura 6.5).

```
ros2 launch g_arm_description display_tool.launch.py
```

---

<sup>7</sup>[https://github.com/frankaemika/franka\\_ros2/blob/develop/franka\\_description/launch/visualize\\_franka.launch.py](https://github.com/frankaemika/franka_ros2/blob/develop/franka_description/launch/visualize_franka.launch.py)

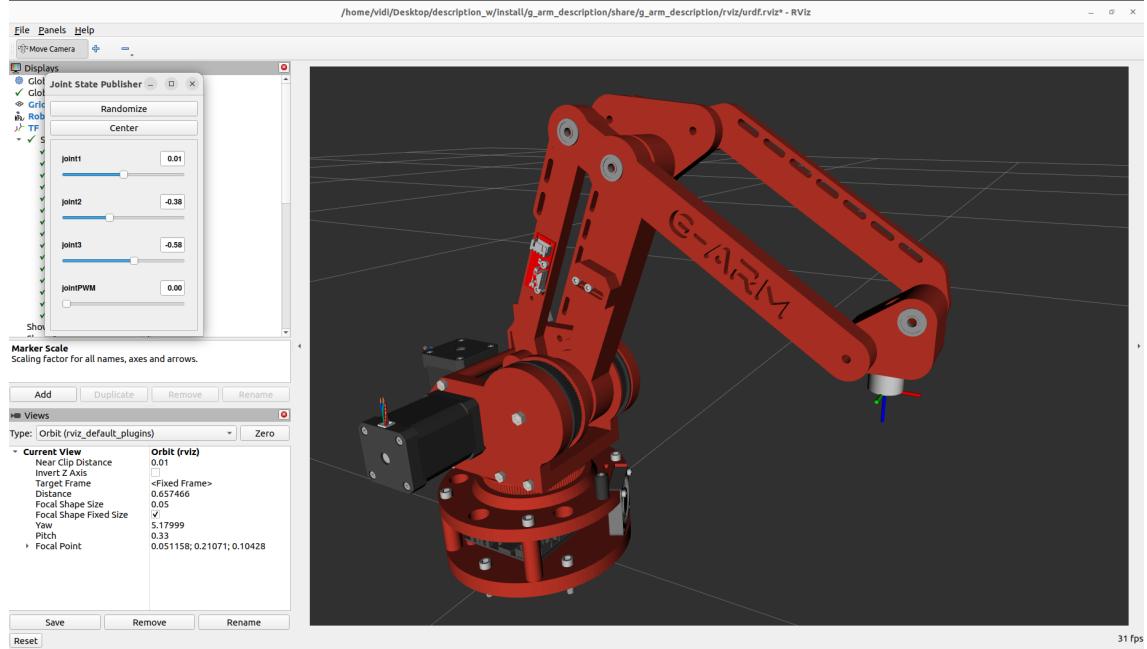


Figura 6.5: Ventana de RViz visualizado el robot creado

### 6.2.2. Integración con MoveIt 2

En esta sección se enumeran y describen los pasos necesarios para, a partir de un paquete de descripción de un robot, generar un paquete de MoveIt.

Antes de comenzar, debemos tener instalado MoveIt 2 para ROS Humble. Para ello, se ha seguido el proceso de instalación<sup>8</sup> de la documentación.

Para crear el paquete de MoveIt de este robot, se han llevado a cabo los siguientes pasos:

1. Lanzamos el asistente de configuración.

```
ros2 launch moveit_setup_assistant setup_assistant.launch.py
```

2. Pulsamos sobre «*Create New MoveIt Configuration Package*» y cargamos el URDF/Xacro del paquete de descripción del robot. Si el fichero es válido, aparecerán una serie de apartados a configurar en el lateral izquierdo del asistente.
3. Comenzamos configurando la apartado «*Self-Collisions*». Aquí se hace uso de las colisiones descritas anteriormente para generar una matriz de posibles colisiones que se pueden dar. Para ello, pulsamos sobre «*Generate Collision Matrix*».

<sup>8</sup>[https://moveit.picknik.ai/main/doc/tutorials/getting\\_started/getting\\_started.html](https://moveit.picknik.ai/main/doc/tutorials/getting_started/getting_started.html)

4. Acto seguido se configura el apartado «**Planning Groups**», donde se establecen los eslabones y articulaciones que serán consideradas a la hora de realizar la planificación. Para ello, pulsamos sobre «**Add Group**» y acto seguido le damos un nombre. Posteriormente, pulsamos sobre «**Add Kin. Chain**» y seleccionamos como eslabón base, el eslabón que irá unido al suelo, y como eslabón *tip*, el extremo del robot (sin llegar a la herramienta). Entonces, guardamos los cambios. Ahora, se añaden todos los *links* y *joints* de la cadena anterior, al grupo de planificación creado, dando en «**Edit Selected**». Repetimos este procedimiento con el grupo de planificación de la herramienta, seleccionando como eslabón base el eslabón final de la anterior cadena, y como final de esta, el último eslabón de la herramienta.
5. En el apartado «**Robot Poses**» se pueden configurar una serie de posiciones predeterminadas para poder usarse posteriormente cuando se necesiten. Por ejemplo, es recomendable crear al menos dos; una que haga de posición *Home* (Posición cómoda para empezar a trabajar) y otra como posición de reposo (Posición que evita que el robot se desplome al cortar la electricidad). Además podemos añadir dos más para el grupo de planificación de la herramienta: *ToolON* y *ToolOFF*.
6. Usamos el apartado «**End Effectors** » para configurar el extremo de nuestro robot como tal. Para ello, seleccionamos el eslabón del extremo y el grupo de planificación creado anteriormente.
7. En «**Passive Joints**», añadimos todos aquellas articulaciones que no sean grados de libertad, es decir, todas aquellas que no sean *joint1*, *joint2*, *joint3*, *jointPWM* (*señal PWM de la herramienta*).
8. En los apartados «**ROS 2 Controllers**» y «**MoveIt Controllers**», añadimos automáticamente los controladores pulsando sobre el único botón que hay.
9. Finalmente rellenamos nuestra información personal en «**Author information**» y generamos el paquete en el *src* del *workspace*. Acto seguido, compilamos.

A la hora de usar esta herramienta, se debe de tener en cuenta que a día de la publicación de este trabajo, existe un error en el asistente que hace que el paquete no se cree bien del todo. Para corregirlo, es necesario cambiar todos los número del fichero *joints\_limits.yaml* del paquete generado a *double*. Es decir, cambiar por ejemplo un “200” por “200.0”.

Para comprobar que el paquete ha sido configurado bien, ejecutamos un *launcher* generado automáticamente por el asistente :

```
ros2 launch g_arm_moveit demo.launch.py
```

Tras ejecutarlo, se debería abrir una ventana de RViz con un aspecto similar a lo mostrado en la Figura 6.6. Podemos utilizar el propio *plugin* de MoveIt para establecer una posición aleatoria y tratar de llegar a ella (Figura 6.7).

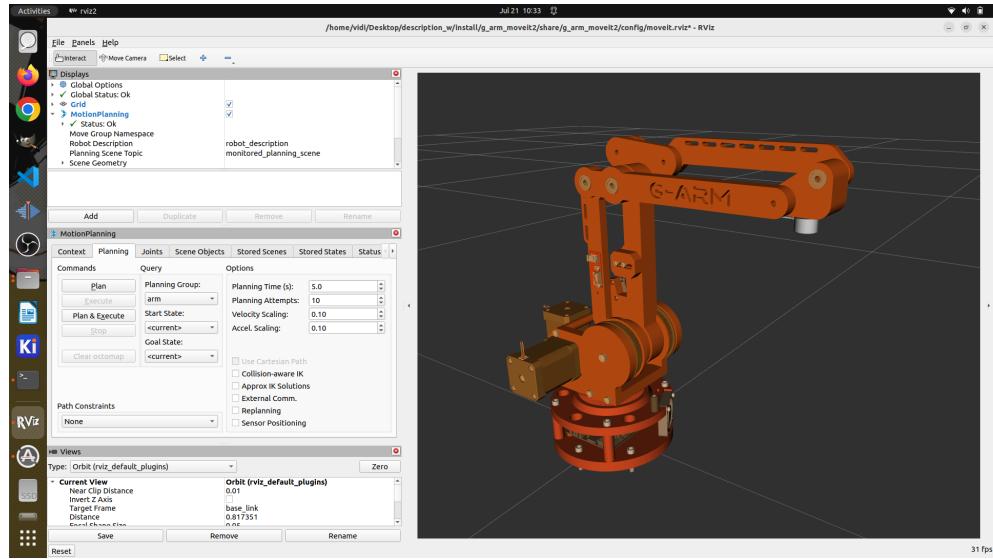


Figura 6.6: Rviz al lanzar el *demo.launch.py*

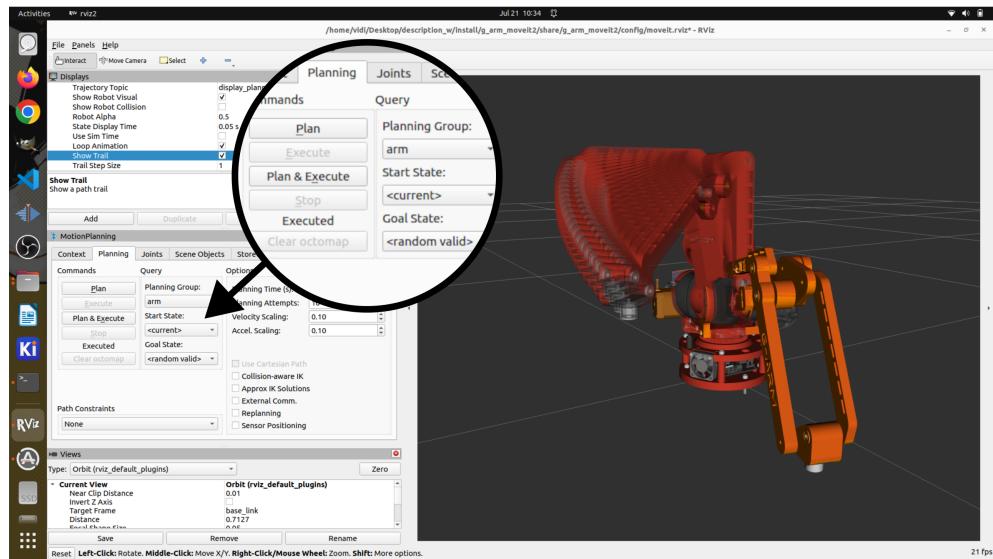


Figura 6.7: Planificando una trayectoria simple

### 6.2.3. Driver para ROS

En esta sección se detalla el funcionamiento del nodo ROS creado para ejecutar las trayectorias en el robot real.

Primeramente es necesario comprender el funcionamiento de este framework, al menos su parte más externa, aquella que se comunica con el exterior. Según la documentación<sup>9</sup>, existe un nodo llamado *move\_group* (Véase la Figura 6.8) que se comunica con resto del framework a través de los topics y acciones de ROS. De cara al exterior, puede establecer conexión con dos elementos del propio entorno del robot. El primero es la percepción, en este caso permite obtener información de la escena a través de los datos de una cámara 3D. El segundo es el controlador del robot, que se encarga de aceptar y ejecutar las trayectorias requeridas, además de devolver información acerca del progreso.



Figura 6.8: Relación de *move\_group* con el resto del *Firmware*

<sup>9</sup>[https://moveit.picknik.ai/humble/doc/concepts/move\\_group.html](https://moveit.picknik.ai/humble/doc/concepts/move_group.html)

Según la propia documentación, la forma más simple y eficaz de controlar el robot real es utilizando un controlador de *ros2\_control*<sup>10</sup> e imitar en el robot real las posiciones de los distintos joints publicadas por el controlador. En sí *ros2\_control* es un framework que incorpora una gran cantidad de herramientas para implementar sistemas de control en ROS. Dentro de él, existen los llamados controladores y según su tipo pueden controlar robots de tracción diferencial, direcciones Ackermann, entre otros. Para esta aplicación el que nos interesa es *Joint Trajectory Controller*, un controlador diseñado para ejecutar trayectorias en el espacio de articulaciones, interpolando entre uno o más puntos de referencia. Conjuntamente con esto, hay que utilizar otro nodo llamado *joint\_state\_broadcaster*, que lee las interfaces de estado del controlador y las publica en los topics */dynamic\_joint\_states* y */joint\_states*. Siendo finalmente este último topic el que deberá ser escuchado por el nodo del robot real para extraer la posición de cada articulación. En el Bloque de código 6.2 se muestran los campos del mensaje enviado a través de él.

---

```
Header header

string[] name
float64[] position
float64[] velocity
float64[] effort
```

---

Código 6.2: Campos del tipo de mensaje *sensor\_msgs/JointState*

El driver creado, hace uso de una clase intermedia entre el nodo ROS y la clase *grbl* con el fin de abstraer al nodo de la conversión de unidades. Además, esta nueva clase incorpora lo necesario para encontrar los límites físicos de cada articulación sumado a una serie de constantes usadas para hacer que la posición absoluta de las articulaciones concuerde con la usada en la descripción del robot.

Al comenzar, trata de conectarse al robot y busca el 0,0 de cada joint. Posteriormente crea una suscripción al topic */joint\_states* para recibir y guardar el último mensaje que llega a través de este topic. En paralelo a esto, existe una función llamada con una cierta frecuencia (utilizando un *timer* de ROS) que envía las posiciones del mensaje, previamente guardado, al robot real. Se ha decidido separar ambos comportamientos para tener un cierto control sobre la frecuencia de envíos de datos por el puerto serie y evitar saturar el firmware del robot.

En la Figura 6.9 se muestra el diagrama de actividades que ilustra el funcionamiento básico del driver creado.

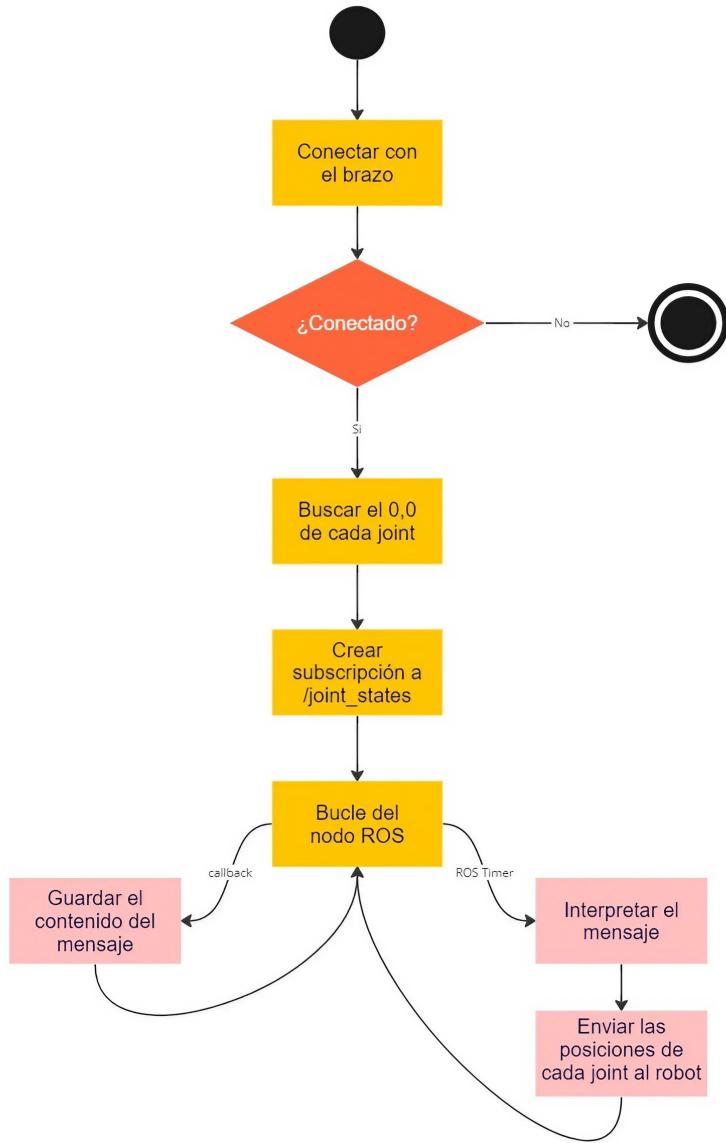


Figura 6.9: Diagrama de actividades del driver

Por suerte, el propio lanzador de demostración mencionado en la sección anterior, ya levanta estos nodos por lo que no es necesario crear uno nuevo para realizar las pruebas. Más adelante sí es recomendable hacerlo, para poder tener mayor control sobre los nodos que se están lanzando y añadir nuevos si es necesario.

## 6.3. Pruebas

En esta sección se ponen a prueba los aspectos técnicos que determinan el desempeño y la fiabilidad del brazo robot. Para ello, se han llevado a cabo una serie de pruebas, bajo distintas circunstancias, y se han apuntado los resultados. Además, se ha grabado un vídeo para cada una de ellas: capacidad de carga<sup>11</sup>, velocidad máxima<sup>12</sup> y consumo eléctrico<sup>13</sup>.

### 6.3.1. Capacidad de carga

En este apartado se pone a prueba la capacidad del robot a la hora de levantar y mover cargas con diferentes masas. Para ello, se han considerado los siguientes escenarios:

- Escenario 1: El robot comienza con el brazo completamente extendido y empieza a levantar lentamente distintas cargas. La prueba finaliza cuando los motores no son capaces de soportarlo.
- Escenario 2: Repetir el escenario 1 pero situando la carga cerca de la base.
- Escenario 3: El robot hace uso de su herramienta electroimán (teóricamente de 3Kg de fuerza) para levantar distintas cargas unidas a un objeto ferromagnético. La prueba finaliza cuando el electroimán no es capaz de sujetar la carga o bien, el robot no es capaz de levantarla.

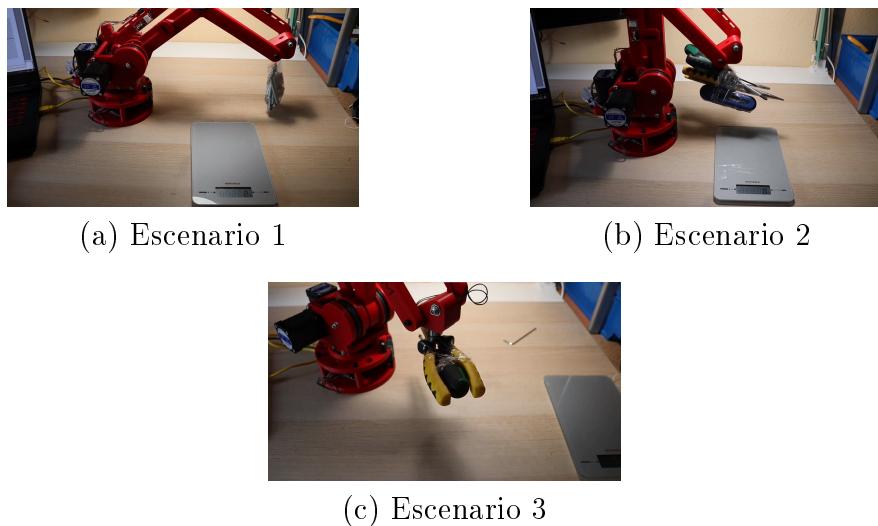


Figura 6.10: Fotos extraídas del vídeo de las pruebas de carga

<sup>11</sup><https://youtu.be/McD7kLMKMo0>

<sup>12</sup>[https://youtu.be/37KK\\_hJk\\_tI](https://youtu.be/37KK_hJk_tI)

<sup>13</sup><https://youtu.be/2MfBabEWZNo>

Los resultados obtenidos de cada escenario (Figura 6.10) se muestran en el Cuadro 6.3. De esta prueba podemos extraer que el brazo es capaz de levantar una carga significativa a pesar de su reducido tamaño. También, se ha comprobado que el electroimán que se anunciaba como capaz de sostener 3 Kg, sólamente es capaz de levantar una décima parte de ello. A pesar de esto, es capaz de levantar y manipular tornillos y objetos metálicos pequeños, incluso en movimientos rápidos. Más allá de los valores de carga máximos, el comportamiento del robot durante la prueba hace pensar que es capaz de manejar cargas de hasta 150g con total normalidad. A modo de ejemplo, en la siguiente prueba se monta un mando de televisión de 100g y se realizan movimientos rápidos con él.

Escenario	Carga máxima
Escenario 1	365 g
Escenario 2	480 g
Escenario 3	305 g

Cuadro 6.3: Resultados de los diferentes resultados en la prueba de carga

### 6.3.2. Velocidad máxima

En esta prueba se evalúa que tan rápido se puede mover cada articulación con una carga de 100g. Para realizar esta prueba se han configurado unas aceleraciones más altas de lo normal y se le ordena moverse a una velocidad inalcanzable. En esta prueba, el robot permanece unido a la mesa mediante unos sargentos para evitar que la inercia haga deslizar la base. Se ha utilizado la línea de tiempo del vídeo para saber en qué segundo y décima de segundo se comienza a mover el robot y en qué momento exacto llega a su destino. Se ha establecido un rango conocido de grados para poder hallar una velocidad media en ese tramo. En el pie de página se puede encontrar un enlace al vídeo de la prueba realizada, donde se ve la velocidad que es capaz de alcanzar este brazo. En la Figura 6.11 se pueden ver algunas imágenes extraídas del propio vídeo.

Como resultado, se ha llegado a la conclusión que la velocidad máxima de giro de la articulación de la base es de 225 grados por segundo. Aún así la velocidad media ha sido de  $90^{\circ}/s$  debido a que primero tiene que acelerar hasta alcanzar la velocidad máxima y después frenar. Este eje ha hecho un recorrido de 180 grados en a penas 2 segundos. En ese tramo es capaz de alcanzar la velocidad máxima durante al menos 90 grados. Esta es una velocidad increíble teniendo en cuenta que la velocidad máxima de la base de un robot ABB IRB120 (20.000€) es de  $250^{\circ}/s$ . En cuanto a las otras dos

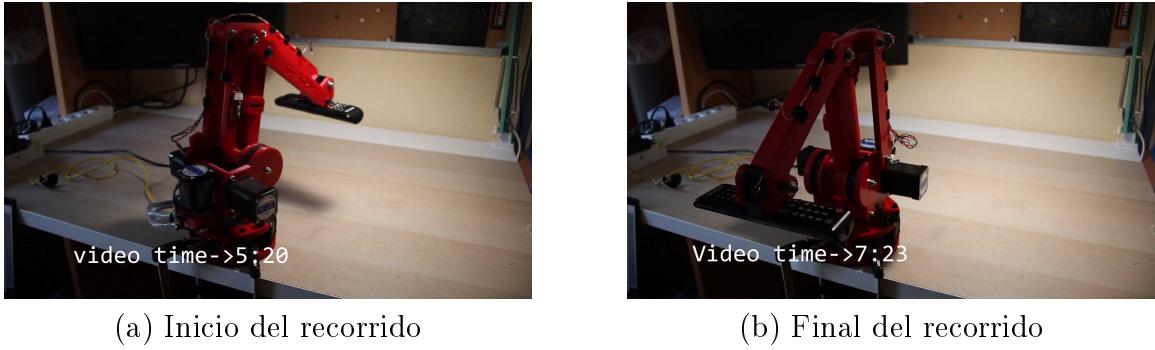


Figura 6.11: Fotos extraídas del vídeo de las pruebas de velocidad

articulaciones, al tener recorridos menores se consigue una máxima de  $120^{\circ}/\text{s}$ .

### 6.3.3. Consumo eléctrico

Este tipo de pruebas evalúan cuanta energía está consumiendo el robot bajo unas condiciones concretas. Es útil para conocer el coste de tener en funcionamiento este robot y si es viable utilizarlo sobre plataformas móviles a baterías.

Para realizar este test, se han considerado 3 escenarios:

- Escenario 1: El robot realiza movimientos lentos.
- Escenario 2: El robot realiza movimientos rápidos
- Escenario 3: El robot se encuentra bloqueado en una posición fija.

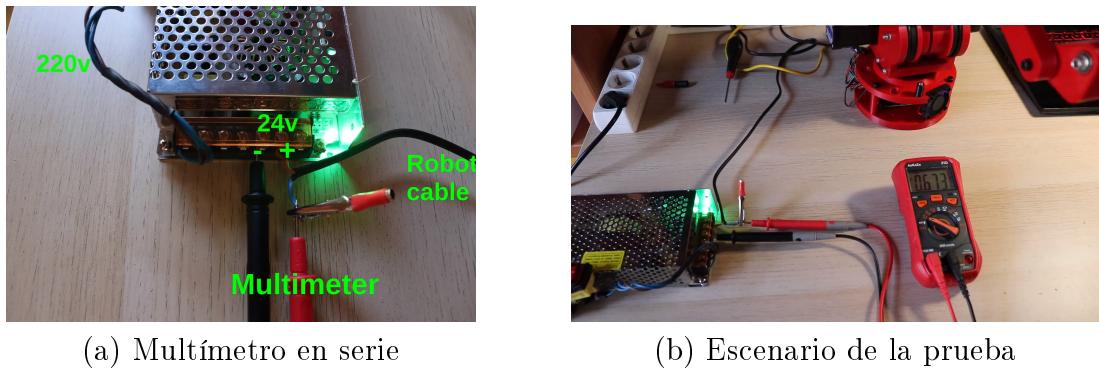


Figura 6.12: Fotos extraídas del vídeo de las pruebas de consumo

En la Figura 6.12 se puede ver cómo se ha realizado esta prueba. Los resultados obtenidos están recogidos en el Cuadro 6.4. Como se puede observar, realmente no existe una gran diferencia entre un movimiento rápido y uno lento, en cambio, la diferencia sí es notable cuando lo comparamos con permanecer en una posición fija. Hay que

recordar que aunque esté en parado, las bobinas de los motores permanecen excitadas para mantener su posición.

Gracias a esta prueba, queda comprobado que sí puede ser usado sobre plataformas móviles y ser alimentado mediante baterías. De hecho, es ideal para ser montado sobre un robot Turtlebot 2 (robot móvil empleado en las asignaturas de *Arquitecturas software para robots y Planificación y sistemas cognitivos*) utilizando la salida de 19V o 12V que incorpora en su base.

Escenario	Consumo medio
Escenario 1	14.4W
Escenario 2	17W
Escenario 3	6.8W

Cuadro 6.4: Consumos eléctricos en distintos escenarios

---

# Capítulo 7

# Conclusiones

---

*El genio se compone del 2% de talento y del 98% de perseverancia*

Beethoven

En este último capítulo se hará mención a los logros alcanzados en términos de objetivos y se presentarán las conclusiones obtenidas en este proyecto. También se discutirán las habilidades y conocimientos adquiridos durante su desarrollo, así como las posibles mejoras futuras a tener en cuenta.

## 7.1. Objetivos cumplidos

Primeramente cabe destacar que se ha logrado cumplir el objetivo principal de este trabajo: desarrollar un brazo robótico que pueda ser empleado en la asignatura de Robótica Industrial de esta universidad. De hecho el robot ha cumplido todos y cada uno de los requisitos planteados en el tercer capítulo:

1. El coste final de fabricación ha sido de 171€, siendo 200€ el límite establecido.
2. En su totalidad está impreso en 3D, a excepción de los componentes electrónicos.
3. Se ha cumplido con mantener el consumo inferior a 25 vatios, siendo siempre inferior a 20.
4. El robot final es portable y cuenta con un tamaño ideal para usarse en una mesa normal. Además se ha logrado que pueda ser usado sin necesidad de anclarlarse al suelo.
5. Es capaz de levantar los 300 gramos propuestos, incluso haciendo uso de la herramienta electromagnética desarrollada.

6. El robot creado es simple de montar y requiere de pocas piezas de tamaño medio que pueden ser impresas en cualquier impresora barata del mercado.
7. Se ha realizado la integración completa de este robot en el ecosistema ROS 2. Adicionalmente, se ha integrado en el framework MoveIt 2 para facilitar su uso.

## 7.2. Competencias adquiridas

En el desarrollo de este trabajo se han adquirido una gran cantidad de conocimientos y competencias, entre los cuales destacan:

- Conocimientos avanzados acerca de Grbl y su funcionamiento más interno.
- Se ha adquirido una amplia soltura en el manejo de la herramienta de diseño FreeCad y en multitud de sus bancos de trabajo (Draft, A2Plus, Fasteners, Mesh...).
- Se ha ganado una gran experiencia en el diseño de piezas mecánicas para su posterior impresión en 3D.
- Aumento en la capacidad de planificación de tareas y organización de los recursos disponibles.
- Manejo más rápido y descubrimiento de comandos nuevos de la herramienta Git.
- Ampliación de conocimientos en ROS 2 y dominio pleno del formato URDF (particularidades, limitaciones y puntos fuertes).
- Conocimiento del funcionamiento interno del framework MoveIt y la configuración de paquetes para usarse dentro de él.
- Generar documentación de calidad para un trabajo y comprender la documentación de otros proyectos para poder utilizarlos posteriormente en proyectos propios.
- Conocimientos avanzados en electrónica y en sistemas relacionados con el mundo de las máquinas CNC.
- Gran dominio de los parámetros de impresión, para lograr piezas con las tolerancias correctas y un acabado visual perfecto.

### 7.3. Valoración final y líneas futuras

Se ha desarrollado un robot eficaz y barato capaz de cumplir con su cometido de poder usarse en la docencia universitaria. Es por esto que se plantean las siguientes líneas futuras para continuar con el proyecto y mejorarlo todavía más:

- Añadir un cuarto grado de libertad en el extremo del robot para poder rotar objetos en el plano de trabajo.
- Crear una serie de herramientas nuevas para acoplar a este robot y aumentar así sus posibilidades.
- Integrar este robot en la asignatura de Robótica Industrial como se plantea en el Anexo II.
- Crear una firmware específico para el ESP32 de la placa base que permita controlar el robot a muy bajo nivel.

# Anexo I: Hazlo tú mismo

---

En este anexo se explica, paso a paso, todo lo necesario para poder reproducir y utilizar este robot a partir del contenido existente en el repositorio<sup>1</sup> de este proyecto.

## Obtención de los ficheros STL

Para poder imprimir las distintas piezas, primeramente se requiere obtener los ficheros STL a partir de los ficheros fuente (piezas de FreeCAD) . Estos, los podemos encontrar en la carpeta **/src/design/FreeCad** del proyecto. Para generarlos se deben realizar los siguientes pasos:

1. Abrimos la pieza correspondiente con FreeCAD (pulsando encima de ella).
2. Cambiamos al banco de trabajo *Mesh Design*.
3. Hacemos click sobre la pieza para seleccionarla y, posteriormente, pulsamos el botón *Crear malla de forma* del panel superior.
4. Introducimos un valor de superficie de desviación de 0.01mm y damos en OK.
5. Click derecho del ratón sobre el objeto de malla creado y exportamos la malla con el botón *Exportar malla*.

### Nota

No se recomienda generar directamente el STL con la opción de exportar del menú superior debido a que no se puede controlar la cantidad de polígonos de la malla y por defecto es bastante baja.

Una vez obtenidos todos los ficheros STL, se pueden importar en el laminador, en este caso se ha empleado Ultimaker Cura 5.2.1 para generar los ficheros que entiende la impresora 3D. La densidad requerida para cada pieza y la cantidad de ellas, se pueden ver en el Cuadro 5.8 del Capítulo 5.

---

<sup>1</sup><https://github.com/RoboticsURJC/tfg-vperez.git>

## Montaje

Actualmente, no se dispone de una guía detallada de montaje. Aún así el montaje resulta bastante sencillo e intuitivo. La manera recomendada de proceder es fijándose en el fichero de montaje llamado `#0_ASSEMBLY.FCStd` que se encuentra en la misma carpeta que las piezas. En él aparece cada pieza, su posición y nombre. En la Sección 5.6 se puede ver la lista de las piezas que son necesarias para poder comprarlas o fabricarlas.

## Software

Todo el software necesario para hacer funcionar este robot se encuentra en la carpeta `/src/software` del repositorio. Dentro de ella existen una serie de paquetes para utilizar el robot, los cuales son:

- Paquete `g_arm`: Es un paquete de ROS 2 que contiene el software necesario para comunicarse con el robot desde el ecosistema ROS. Dentro de él se encuentra la carpeta `g_arm_lib` con las clases de python necesarias para abstraer al ejecutable (driver del robot) de las comunicaciones.
- Paquete `g_arm_description`: Es un paquete de descripción creado para representar el robot real en el mundo virtual. No contiene código, únicamente una serie de lanzadores para poder visualizarlo.
- Paquete `g_arm_moveit2`: Es un paquete de MoveIt 2 que utiliza el anterior paquete para poder controlar el robot real desde este framework.
- Paquete `g_arm_python_examples`: Es un paquete que contiene ejemplos de código para mover el robot mediante la API de PyMoveit2.

### Nota

Aunque es posible controlar las articulaciones del robot a bajo nivel enviando órdenes de código G por el puerto serie, se pierde la posibilidad de conocer la posición absoluta de cada articulación. Debido a esto, es recomendable utilizar al menos la capa de abstracción `Robot` que se puede encontrar en la carpeta `g_arm_lib` del paquete `g_arm`. Para conocer su funcionamiento, se puede analizar el código del ejecutable `driver.py` de esa misma carpeta.

Aun así, lo ideal es utilizar ROS 2 Humble y MoveIt 2 para controlar el robot. Para ello, es necesario tener instalado ambos y hacer uso de los cuatro paquetes mencionados.

Para utilizar estos paquetes es necesario añadirlos al *workspace* y compilar. En caso de que no se tenga uno, se deben seguir los siguientes pasos:

1. Se crea una carpeta llamada *workspace* en el directorio *home*, por ejemplo. Dentro de ella se debe de crear la carpeta *src*.
2. En *src*, se copian los paquetes mencionados anteriormente. Además de estos cuatro, es necesario incluir el paquete PyMoveit2<sup>2</sup>.
3. Desde el nivel de la carpeta *workspace* ejecutamos el comando *colcon build -merge-install -symlink-install*.
4. Para que ROS lo encuentre debemos añadir la siguiente línea al final del fichero oculto *.bashrc* del directorio *home*: **source ~ /workspace/install/setup.bash**

La estructura del *workspace* debería quedar igual a la mostrada en la Figura 7.1.

```

build
├── g_arm
├── g_arm_description
├── g_arm_moveit2
├── g_arm_python_examples
└── pymoveit2
install
├── lib
├── local
└── share
log
├── build_2023-09-22_19-07-44
├── build_2023-09-23_19-16-34
├── build_2023-09-23_21-08-22
├── build_2023-09-23_21-13-38
├── build_2023-09-24_11-40-36
├── build_2023-09-24_15-28-48
├── build_2023-09-24_15-38-39
├── latest -> latest_build
├── latest_build -> build_2023-09-24_15-38-39
├── latest_list -> list_2023-09-24_11-30-00
└── list_2023-09-24_11-30-00
src
├── g_arm
├── g_arm_description
├── g_arm_moveit2
├── g_arm_python_examples
└── pymoveit2

```

Figura 7.1: Resultado de ejecutar *tree -d -L 1* en la carpeta *workspace*

En cuanto al firmware interno del robot, utiliza la versión de Grbl v1.1. Existen numerosas guías en internet de cómo instalarlo. En el caso de utilizar la misma placa de este proyecto, no es necesario instalar nada ya que viene preinstalado de fábrica. Lo único a cambiar son los parámetros internos a los mostrados en el Cuadro 6.2. En esa misma sección se detalla como se cambian.

---

<sup>2</sup><https://github.com/Andrej0rsula/pymoveit2>

## Anexo II: Uso en docencia

---

En este anexo se aborda el cómo se pretende incluir este robot en la asignatura de Robótica Industrial del Grado de Robótica Software de la Universidad Rey Juan Carlos. Primeramente, se ha consultado con Julio Lora, actual profesor de esta asignatura, cómo se podría integrar este trabajo en el siguiente curso académico. Se llegó a la conclusión de que una buena manera de utilizar este robot en sus clases, sería para complementar al robot UR3, utilizado actualmente para enseñar a los alumnos a generar trayectorias de soldadura. Teniendo además el punto fuerte de aprender a programar un brazo robot a través de un framework de código abierto como es MoveIt2. En base a lo anterior, se ha planteado un posible ejercicio práctico.

### Ejercicio: Ejecutando trayectorias en MoveIt 2

Crear un programa en Python para que el robot G-Arm ejecute una trayectoria con framework de MoveIt2. La trayectoria debe consistir en una lista de puntos de paso. Además, se debe hacer uso de la herramienta *Porta lápices* para dibujar la trayectoria sobre un folio con el robot real.

### Competencias adquiridas

Al realizar este ejercicio los alumnos aprenderán las siguientes competencias:

1. Conocer la existencia del framework de MoveIt.
2. Aprender a generar los puntos de paso de distintas figuras.
3. Controlar y programar un brazo robot mediante la API de Python de MoveIt2.
4. Representar el recorrido de un eslabón con Rviz.

## Conocimientos necesarios para resolverlo

Para solucionar el ejercicio propuesto, es necesario conocer las herramientas disponibles. Una de ellas es librería PyMoveIt2, la cuál proporciona una serie de clases y métodos que facilitan la interacción con este framework. Dentro de ella, existen una serie de clases que nos permiten controlar el robot de las siguientes formas:

1. ***Joint goal***: Permite controlar el robot en el espacio de articulaciones, es decir llevar cada articulación del robot a posición angular concreta.
2. ***Pose goal***: Permite controlar el robot en el espacio cartesiano, es decir llevar al extremo del robot a una posición XYZ con una cierta orientación.
3. ***Gripper action***: Permite interactuar con la herramienta del robot a través de una serie de cómodas funciones.
4. ***Servo***: Permite mandar comandos en tiempo real (sin la previa planificación que requieren los anteriores) para controlar las velocidad lineares y angulares del extremo del robot.

Como en este caso se pretenden realizar trayectorias, es necesario utilizar las funciones relativas a ***Pose goal***.

## Desarrollo de la solución

Primeramente, se ha creado un programa sencillo en Python con cuatro funciones las cuales devuelven una lista de puntos que corresponden con los vértices de cuatro figuras: cuadrado, círculo, triángulo y un corazón. Para comprobar que los puntos son correctos, se puede utilizar la librería Turtle para dibujarlos. Por otro lado, es necesario implementar un bucle que itere sobre la lista de puntos y mueva el robot hasta el siguiente. Para hacer esto, es necesario crear una instancia de la clase *MoveIt2* y utilizar los métodos *move\_to\_pose()* y *wait\_until\_executed()* como se muestra en el fragmento de código 7.1.

---

```

from pymoveit2 import MoveIt2

# Object creation
moveit2 = MoveIt2(
    node=self._node,
    joint_names=g_arm.joint_names(),
    base_link_name=g_arm.base_link_name(),
    end_effector_name=g_arm.end_effector_name(),
    group_name=g_arm.MOVE_GROUP_ARM,
    callback_group=callback_group,
    follow_joint_trajectory_action_name="/arm_controller/follow_joint_trajectory"
    execute_via_moveit=False
)

# Target position related to frame_id variable
position = (0.25, 0.0, 0.0)

moveit2.move_to_pose(position=position, quat_xyzw=(1.0, 0.0, 0.0,
    0.0), cartesian=True,
    frame_id=g_arm.base_link_name(),
    target_link=g_arm.end_effector_name(),
    tolerance_orientation=3.14)

# Wait until there is no action running
moveit2.wait_until_executed()

```

---

Código 7.1: Uso básico de PyMoveIt2 para moverse a un punto

El código completo de la solución se encuentra, junto al resto de ejemplos, en el repositorio de github del proyecto<sup>3</sup>. Para probarlo, debemos ejecutar lo siguiente (cada comando en una terminal distinta):

```

ros2 launch g_arm_moveit2 show_end_effector_travel.launch
ros2 run g_arm_python_examples trajectory

```

---

<sup>3</sup>[https://github.com/RoboticsURJC/tfg-vperez/blob/main/src/software/g\\_arm\\_python\\_examples/g\\_arm\\_python\\_examples/trajectory.py](https://github.com/RoboticsURJC/tfg-vperez/blob/main/src/software/g_arm_python_examples/g_arm_python_examples/trajectory.py)

Para visualizar el recorrido de un determinado eslabón se debe de activar la opción *Show Trail* de *RobotModel*. En este caso, interesa activar la del extremo del robot, como se muestra en la Figura 7.2.

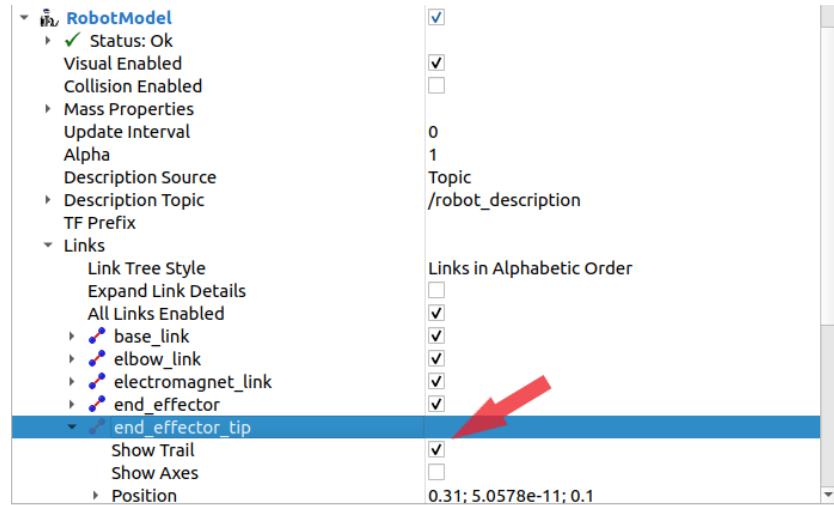


Figura 7.2: Activación de la opción Show Trail de un link en Rviz

En la Figura 7.3 se puede ver las distintas trayectorias de ejemplo ejecutadas .

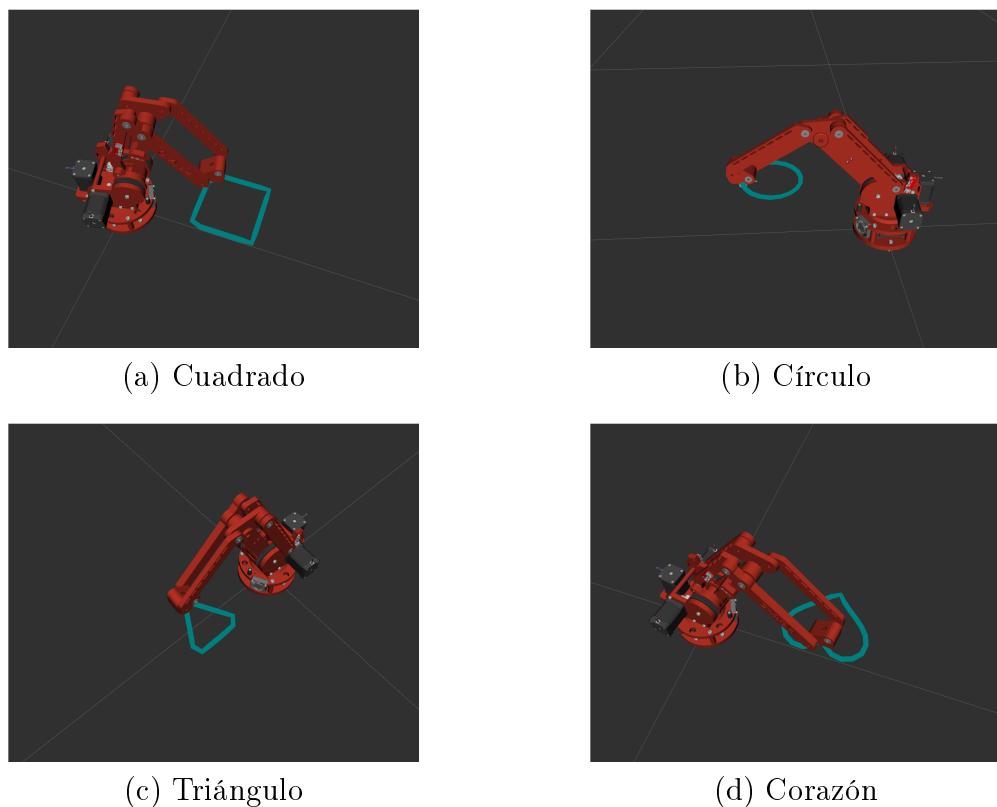
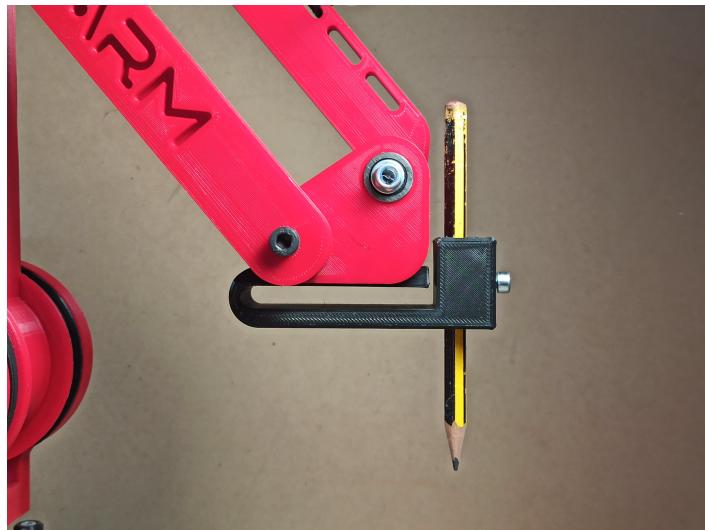
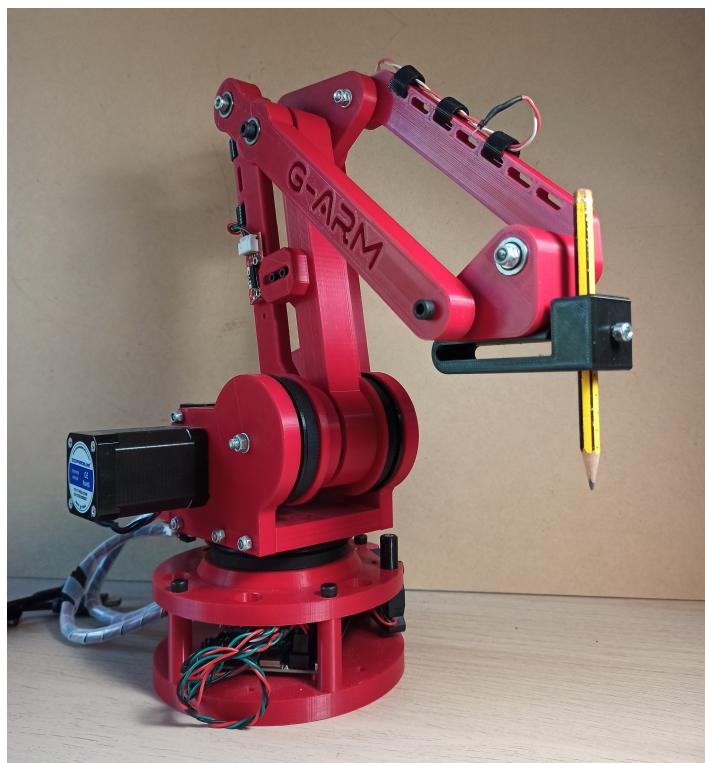


Figura 7.3: Trayectorias realizadas

Finalmente, con el objetivo de probar la trayectoria en el robot real, se ha diseñado una herramienta nueva (Figura 7.4). Se trata de un porta lápices que utiliza la flexión del PLA para acoplar un lápiz al extremo del robot de una forma robusta y sin holguras, siendo a su vez flexible. Se puede descargar el modelo desde la carpeta de piezas del repositorio (24\_TOOL\_Pen.FCStd).



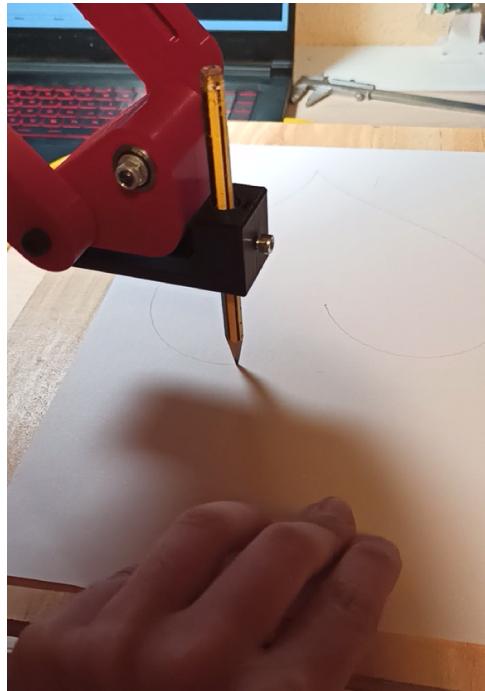
(a) Vista lateral



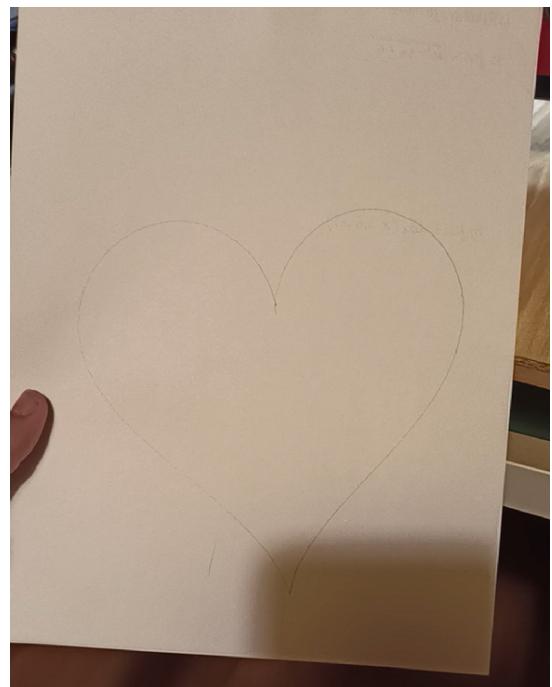
(b) Vista completa

Figura 7.4: Herramienta porta lápices

Se ha escogido un plano con una altura en Z igual a 10cm, ya que en este plano es capaz de alcanzar una gran cantidad de puntos. En él, se ha dispuesto una base plana y un folio en el que se ha dibujado la figura más compleja del ejemplo, el corazón. El resultado de todo el proceso de dibujado puede verse en el siguiente vídeo<sup>4</sup>. Adicionalmente, en la Figura 7.5 se muestra algunas imágenes extraídas de él.



(a) Durante la trayectoria



(b) Resultado final

Figura 7.5: Prueba en el robot real

Como se puede ver, el trazado es excelente, por lo que puede ser utilizado para realizar todo tipo de trayectorias asegurando unos buenos resultados.

<sup>4</sup>[https://youtube.com/shorts/3rwK\\_FV3eSs](https://youtube.com/shorts/3rwK_FV3eSs)

## Ejercicio extra: Uso de una herramienta actuada

Crear un programa en Python para llevar a cabo una tarea de manipulación con el robot G-Arm. La tarea consiste en coger una moneda de una determinada posición y depositarla en otra distinta, haciendo uso de la herramienta *Electroimán*.

### Conocimientos previos

La herramienta acoplada al robot debe ser controlada mediante un joint ficticio con un rango de movimiento de 0.0 a 1.0 radianes. Este valor es convertido por el driver de ROS en un valor entendible por la salida PWM del extremo del robot. En la configuración del paquete de MoveIt se estableció que este joint no pertenece a la cadena cinemática del brazo y no es usado a la hora de planificar. Este, se controla separadamente en un grupo separado llamado Tool. Para controlar el actuador del extremo del robot, la librería PyMoveit2 incorpora una clase llamada *MoveIt2Gripper* y principalmente dos funciones: *open()* y *close()*.

### Desarrollo

En este ejercicio se ha utilizado la clase *MoveIt2Gripper* mencionada anteriormente, tal y como se muestra en el fragmento de código 7.2.

---

```
# Create MoveIt 2 gripper interface
tool = MoveIt2Gripper(
node=self._node,
gripper_joint_names=g_arm.tool_joint_name(),
open_gripper_joint_positions=g_arm.electromagnet_on(),
closed_gripper_joint_positions=g_arm.electromagnet_off(),
gripper_group_name=g_arm.MOVE_GROUP_GRIPPER,
follow_joint_trajectory_action_name="/tool_controller/follow_joint_trajectory",
callback_group=callback_group)

tool.open() # Open gripper (electromagnet on)
tool.wait_until_executed()
tool.close() # Close gripper (electromagnet off)
tool.wait_until_executed()
```

---

Código 7.2: Uso básico de PyMoveIt2 para moverse a un punto

Posteriormente, se han establecido una serie de puntos: P0, PA1, PA2, PB1, PB2 (Figura 7.6). Y se ha utilizado los conocimientos del anterior ejercicio para moverse entre ellos.

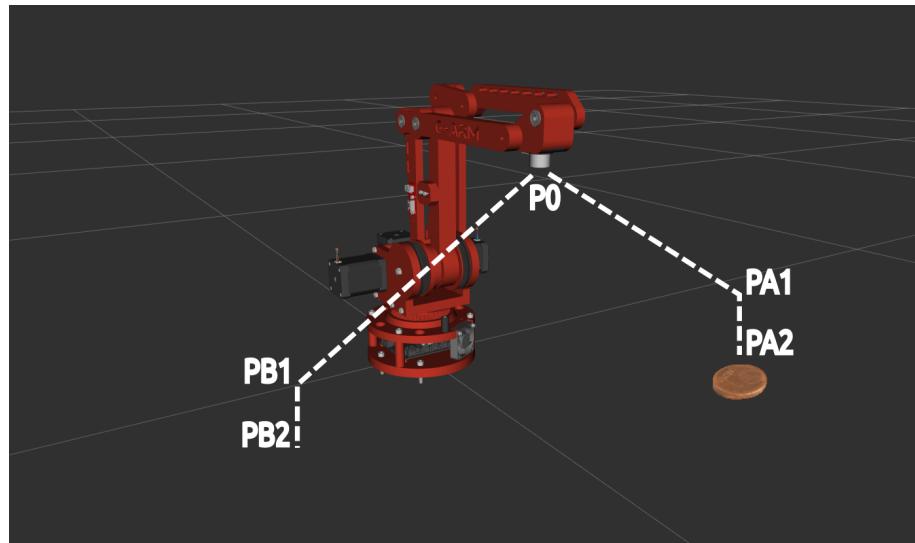


Figura 7.6: Puntos del ejercicio

El código completo de la solución se encuentra, junto al resto de ejemplos, en el repositorio de github del proyecto<sup>5</sup>. Para probarlo, debemos ejecutar lo siguiente (cada comando en una terminal distinta):

```
ros2 launch g_arm_moveit2 show_end_effector_travel.launch
ros2 run g_arm_python_examples electromagnet
```

En este enlace<sup>6</sup> se puede ver el vídeo que se ha grabado de la ejecución en el robot real.

---

<sup>5</sup>[https://github.com/RoboticsURJC/tfg-vperez/blob/main/src/software/g\\_arm\\_python\\_examples/g\\_arm\\_python\\_examples/electromagnet.py](https://github.com/RoboticsURJC/tfg-vperez/blob/main/src/software/g_arm_python_examples/g_arm_python_examples/electromagnet.py)

<sup>6</sup><https://youtu.be/iRcEg6AgwcU>

# Bibliografía

---

- [Adediran et al., 2023] Adediran, E. M., Fadare, D. A., Falana, A., Kazeem, R. A., Ikumapayi, O. M., Adedayo, A. S., Adetunla, A. O., Ifebunandu, U. J., Fadare, D. A., and Olarinde, E. S. (2023). Uiarm i: Development of a low-cost and modular 4-dof robotic arm for sorting plastic bottles from waste stream. *Journal Europeen des Systemes Automatises*, 56(1):97.
- [Armesto et al., 2016] Armesto, L., Fuentes-Durá, P., and Perry, D. (2016). Low-cost printable robots in education. *Journal of Intelligent & Robotic Systems*, 81:5–24.
- [Čehovin Zajc et al., 2018] Čehovin Zajc, L., Rezelj, A., and Skočaj, D. (2018). Open-source robotic manipulator and sensory platform. In Lepuschitz, W., Merdan, M., Koppensteiner, G., Balogh, R., and Obdržálek, D., editors, *Robotics in Education*, pages 250–256, Cham. Springer International Publishing.
- [Čehovin Zajc et al., 2019] Čehovin Zajc, L., Rezelj, A., and Skočaj, D. (2019). Teaching with open-source robotic manipulator. In Lepuschitz, W., Merdan, M., Koppensteiner, G., Balogh, R., and Obdržálek, D., editors, *Robotics in Education*, pages 189–198, Cham. Springer International Publishing.
- [Cocota et al., 2012] Cocota, J. A. N., Fujita, H. S., and da Silva, I. J. (2012). A low-cost robot manipulator for education. In *2012 Technologies Applied to Electronics Teaching (TAEE)*, pages 164–169.
- [Coleman et al., 2014] Coleman, D. T., Sucan, I. A., Chitta, S., and Correll, N. (2014). Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of software engineering in robotics*, 5(1):14.
- [Krimpenis et al., 2020] Krimpenis, A. A., Papapaschos, V., and Bontarenko, E. (2020). Hydrax, a 3d printed robotic arm for hybrid manufacturing. part i: Custom design, manufacturing and assembly. *Procedia Manufacturing*, 51:103–108. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).

- [Papapaschos et al., 2020] Papapaschos, V., Bontarenko, E., and Krimpenis, A. A. (2020). Hydrax, a 3d printed robotic arm for hybrid manufacturing. part ii: Control, calibration and programming. *Procedia Manufacturing*, 51:109–115. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).
- [Prabowo et al., 2019] Prabowo, D., Wiannastiti, M., and Hedwig, R. (2019). Simulation of mitsubishi rv-m1 robotic arms by using matlab® for high school teaching. *IOP Conference Series: Materials Science and Engineering*, 598(1):012104.
- [Quigley et al., 2011] Quigley, M., Asbeck, A., and Ng, A. (2011). A low-cost compliant 7-dof robotic manipulator. pages 6051–6058.
- [Rezeck et al., 2017] Rezeck, P. A. F., Azpurua, H., and Chaimowicz, L. (2017). Hero: An open platform for robotics research and education. In *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pages 1–6.
- [Sabri et al., 2021] Sabri, M., Fauzi, R., Fajar, M. S., Geubrina, H. S., and Sabri, F. A. M. (2021). Model and simulation of arm robot with 5 degrees of freedom using matlab. *IOP Conference Series: Materials Science and Engineering*, 1122(1):012032.
- [Vujičić et al., ] Vujičić, V., Milićević, I., Dragičević, S., and Marjanović, M. Realization of model of robotic arm s-430if for education purposes.