



GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela de Ingeniería de Fuenlabrada

Curso académico 2022-2023

Trabajo Fin de Grado

Diseño y fabricación de un brazo robótico industrial impreso en 3D, de bajo costo y código abierto con soporte para ROS 2.

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

Unas bonitas palabras...

Quizás un segundo párrafo esté bien. No te olvides de nadie.

Un tercero tampoco viene mal para contar alguna anécdota...

¿Alguien más? Aunque sean *actores* secundarios.

Un quinto párrafo como colofón.

*A alguien especial;
si no, tampoco pasa nada*

Madrid, xx de xxxxxx de 20xx

Tu nombre

Resumen

Escribe aquí el resumen del trabajo. Un primer párrafo para dar contexto sobre la temática que rodea al trabajo.

Un segundo párrafo concretando el contexto del problema abordado.

En el tercer párrafo, comenta cómo has resuelto la problemática descrita en el anterior párrafo.

Por último, en este cuarto párrafo, describe cómo han ido los experimentos.

Acrónimos

ADN *Ácido Desoxirribonucleico*

DOF *Degrees Of Freedom*

STEM *Science, Technology, Engineering and Mathematics*

DIY *Do It by Yourself*

CAD *Diseño Asistido por ordenador*

FDM *Modelado por Deposición Fundida*

ROS *Robot Operating System*

CNC *Control Numérico por Computadora*

CC *Corriente Contínua*

PWM *Pulse Width Modulation*

DDS *Data Distribution Service*

CAM *Fabricación Asistida por Computadora*

SCARA *Selective Compilant Assembly Robot Arm*

URDF *Unified Robot Description Format*

XML *eXtensible Markup Language*

UGS *Universal Gcode Sender*

UART *Universal Asynchronous Receiver/Transmitter*

AGV *Automatic Guided Vehicle*

AMR *Autonomous Mobile Robots*

BLCD *Autonomous Mobile Robots*

STL *Standard Tessellation Language*

Índice general

1. Introducción	1
1.1. Robótica	1
1.1.1. Robótica de servicio	3
1.1.2. Robótica industrial	7
1.2. Robótica educativa	10
1.2.1. Robótica en la educación secundaria	10
1.2.2. Robótica en la universidad	11
1.3. Robótica de bajo coste	12
2. Estado del arte	13
3. Objetivos	20
3.1. Descripción del problema	20
3.2. Requisitos	21
3.3. Metodología	22
3.4. Plan de trabajo	22
4. Plataforma de desarrollo	24
4.1. Software	24
4.1.1. Python	24
4.1.2. Grbl	25
4.1.3. Código G	25
4.1.4. FreeCAD	26
4.1.5. ROS 2	27
4.1.6. MoveIt!	27
4.2. Hardware	28
4.2.1. MKS DLC32	29
4.2.2. Motores Nema 17	30
4.2.3. Controlador TMC2209	31

4.2.4. Fuente de alimentación genérica	33
5. Desarrollo hardware del manipulador	34
5.1. Eligiendo la geometría del manipulador	34
5.2. Modelo alámbrico	36
5.2.1. En qué consiste	36
5.2.2. Desarrollo del modelo alámbrico de G-Arm	37
5.3. Bocetos	39
5.4. Elección de componentes hardware	39
5.4.1. Motores	39
5.4.2. Reductora	41
5.4.3. Controladores	41
5.4.4. Placa base	42
5.4.5. Fuente de alimentación	42
5.4.6. Rodamientos	44
5.4.7. Finales de carrera	44
5.4.8. Electroimán	45
5.5. Diseño CAD	46
5.5.1. Base principal	46
5.5.2. Base de los motores	48
5.5.3. Paralelogramos	49
5.5.4. Elemento terminal	50
5.5.5. Herramienta electroimán	51
5.6. Impresión y montaje	53
6. Desarrollo software del manipulador	55
6.1. Control de los actuadores	55
6.1.1. Grbl como <i>firmware</i> del robot	55
6.1.2. Comunicación con Grbl	56
6.1.3. Configuración de Grbl para su uso en robótica	58
6.2. Integración con ROS 2	61
6.2.1. Descripción del robot	61
6.2.2. Integración con MoveIt 2	66
6.2.3. Nodo ROS	69
6.3. Pruebas	71
6.3.1. Estabilidad y vibración	71
6.3.2. Capacidad de carga	71

6.3.3. Velocidad y tiempo de respuesta	71
6.3.4. Exactitud y repetitividad	72
7. Conclusiones	73
7.1. Conclusiones	73
7.2. Corrector ortográfico	74
Bibliografía	75

Índice de figuras

1.1.	Robot Unimate	1
1.2.	Robot Shakey	2
1.3.	Robots de campo	3
1.4.	Robots de limpieza	4
1.5.	Robots de entretenimiento	4
1.6.	Robots en el campo de la salud	5
1.7.	Robots de logística usados en almacenes	6
1.8.	Robot de reparto de Glovo	6
1.9.	Fanuc SR-3iA	7
1.10.	Robot articulados	8
1.11.	Robots basados en paralelogramos	8
1.12.	Robots cartesianos	9
1.13.	Robots en institutos	10
1.14.	Robots usados en universidades	11
1.15.	Robots de bajo coste	12
2.1.	Robot HydraX	13
2.2.	Robot UIArm	15
2.3.	Robot Niryo One	16
2.4.	Robots formados a partir de paralelogramos	18
4.1.	Logo de Grbl	25
4.2.	Logo de FreeCAD	27
4.3.	Logotipo de ROS 2 Humble	28
4.4.	Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot)	28
4.5.	Placa base MakerBase DLC32	29
4.6.	Motores Nema 17 de 2.1A	30
4.7.	Controlador TMC2209	31
4.8.	Métodos usados para alimentar el robot	33

5.1.	Espacio tridimensional	34
5.2.	Tipos de articulaciones más usadas en robótica	35
5.3.	Pinza paralela con 1 grado de libertad	36
5.4.	Parámetros del modelo alámbrico	37
5.5.	Modelo alámbrico de G-Arm	38
5.6.	Diferentes categorías de motor paso a paso Nema ¹	40
5.7.	Rodamientos de tipo brida	44
5.8.	Base principal	46
5.9.	Base de los motores	48
5.10.	Elemento terminal	50
5.11.	Vistas del elemento terminal	50
5.12.	Herramienta electroimán	51
5.13.	Contorno de la polea de 100 dientes	52
5.14.	Contorno de la polea de extruido	53
5.15.	Ender-3 Pro V1 2017	53
6.1.	Interfaz de UniversalGcodeSender ²	58
6.2.	Elementos del formato URDF ³	62
6.3.	Rviz al lanzar el <i>demo.launch.py</i>	68
6.4.	Planificando una trayectoria simple	68
6.5.	Relación de <i>move_group</i> con el resto del <i>Firmware</i>	69

¹https://filament2print.com/es/blog/139_motores-nema.html

²https://winder.github.io/ugs_website/download/

³<http://library.isr.ist.utl.pt/docs/roswiki/urdf%282f%29XML%282f%29Joint.html>

Listado de códigos

Listado de ecuaciones

6.1. Cálculo de pasos por grado en Grbl	59
---	----

Índice de cuadros

4.1. Especificaciones técnicas de los motores utilizados	31
4.2. Especificaciones técnicas del TMC2209	32
5.1. Parámetros del modelo alámbrico de G-Arm	39
5.2. Comparación de especificaciones de los controladores existentes	42
5.3. Componentes hardware necesarios	54
5.4. Tornillería necesaria	54
5.5. Piezas necesarias	54
6.1. Comandos utilizados en la realización del proyecto	56
6.2. Parámetros Grbl usados en este trabajo	60
6.3. Evaluación de la estabilidad	71
6.4. Evaluación de la capacidad de carga	71
6.5. Evaluación de la velocidad y tiempo de respuesta	72

Capítulo 1

Introducción

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

Winston Churchill

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.

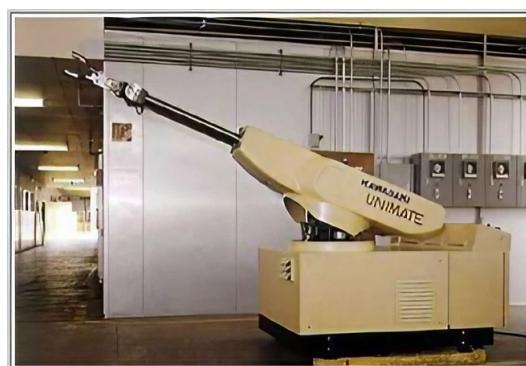


Figura 1.1: Robot Unimate

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.

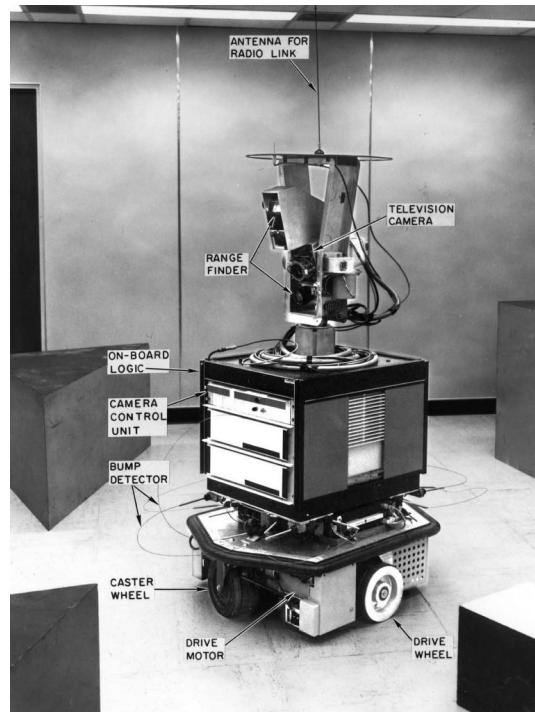


Figura 1.2: Robot Shakey

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.

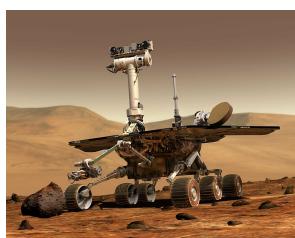
En la actualidad, la robótica está presente en numerosos sectores, abarcando una gran variedad de aplicaciones. Esta versatilidad ha llevado al surgimiento de diversos grupos y categorías que nos permiten clasificar las diferentes tipos de robots.

1.1.1. 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.

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.



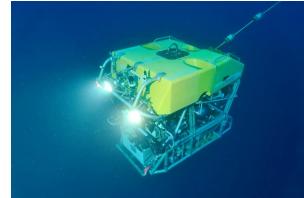
(a) NASA
Opportunity



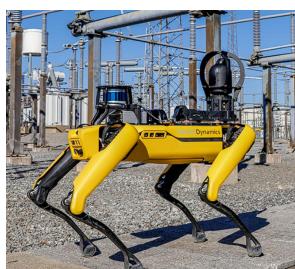
(b) Tractor
autónomo



(c) Drone de rescate



(d) ROV Victor 6000



(e) Boston Dynamics
Spot



(f) Waymo

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).



Figura 1.4: Robots de limpieza

Robots de entretenimiento

Los robots de entretenimiento son máquinas diseñadas para proporcionar diversión y compañía a las personas. Estos robots están equipados con capacidades y características específicas que los hacen adecuados para interactuar con las personas. Tienen una apariencia amigable y agradable. 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.

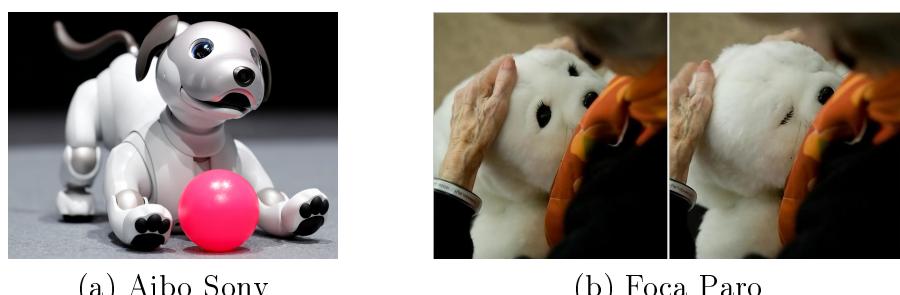


Figura 1.5: Robots de entretenimiento

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 1.6(a), la rehabilitación 1.6(b) y la trasferencia de pacientes 1.6(c), entre otros.



(a) Da Vinci



(b) Atlas Marsi 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.

Los robots de logística destinados a usarse en almacenes pueden ser clasificados en 2 categorías:

- *Automatic Guided Vehicle (AGV)*: Son robots filoguiados, es decir, solo se pueden desplazar a lo largo de un carril preestablecido. Este carril puede ser una línea pintada en el suelo o un riel metálico. Son baratos pero son poco flexibles.
- *Autonomous Mobile Robots (AMR)*: Son la evolución de los AGV. Están dotados con una gran variedad de sensores para poder auto-localizarse y

detectar obstáculos. Son capaces de navegar autonomamente transportando cargas pesadas, como *palets* o estanterías, en colaboración con el resto de la flota. Además son adaptables y fáciles de desplegar pero su coste es más elevado.

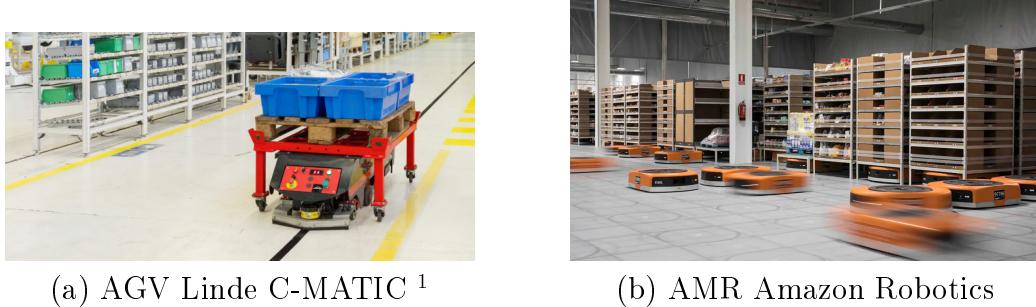


Figura 1.7: Robots de logística usados en almacenes

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 1.8: Robot de reparto de Glovo

¹<https://www.linde-mh.es/es/Productos/Carretillas-automatizadas/C-MATIC/>

1.1.2. 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 del 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 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.

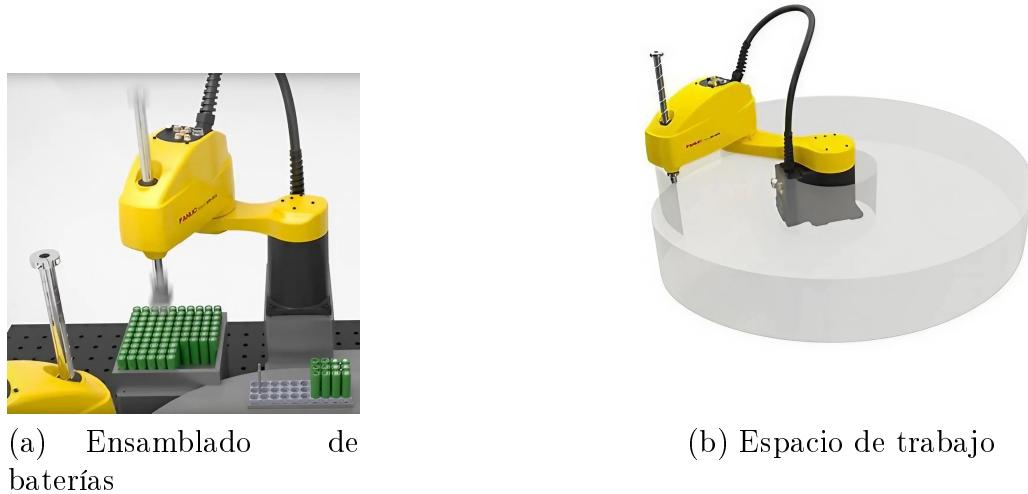


Figura 1.9: Ejemplo: FANUC SR-3iA ²

²<https://www.fanuc.eu/uk/en/robots/robot-filter-page/scara-series/scara-sr-3ia>

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.

(a) KUKA IONTEC ³(b) ABB IRB 6700 ⁴

Figura 1.10: 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.



(a) Robot Delta ABB



(b) FANUC M-410iC

Figura 1.11: Robots basados en paralelogramos

³<https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industria/kr-iontec>

⁴<https://new.abb.com/products/robotics/robots/articulated-robots/irb-6700>

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 ⁵

(b) OpenBuilds CNC Lead 1010 ⁶

Figura 1.12: 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 robótica y automatización aumenta, se hace evidente la necesidad de formar a más personas en este campo.

⁵<https://www.smallsmt.biz/bestueckungsautomat/>

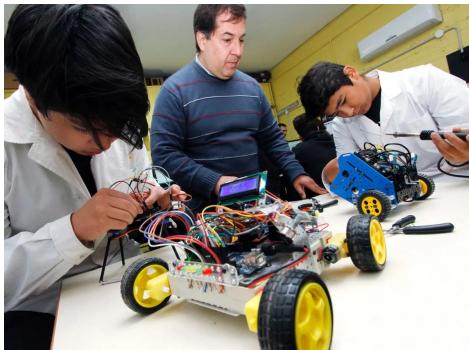
⁶<https://openbuilds.com/builds/lead-cnc-1010-40-x-40.7832/>

1.2. Robótica educativa

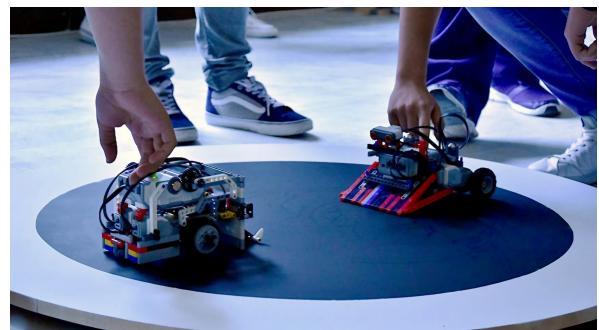
La robótica educativa es una disciplina que ha adquirido una gran relevancia en los últimos años debido a la creciente necesidad de formar a las nuevas generaciones en competencias tecnológicas.

1.2.1. 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 ⁷



(b) Institutos en Toledo ⁸

Figura 1.13: Robots en institutos

⁷<https://www.diariodecuyo.com.ar/suplementos/Robotica-la-ciencia-de-aprender-jugando-20170907-0087.html>

⁸<https://www.uclmtv.uclm.es/institutos-de-camarena-y-madridejos-ganan-el-viii-torneo-de-robotica-organizado-por-la-uclm-en-toledo/>

1.2.2. 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 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 1.14. Al tratarse de sistemas usados en el mundo profesional, los estudiantes pueden aprender con robots similares a los que usarán en un futuro.

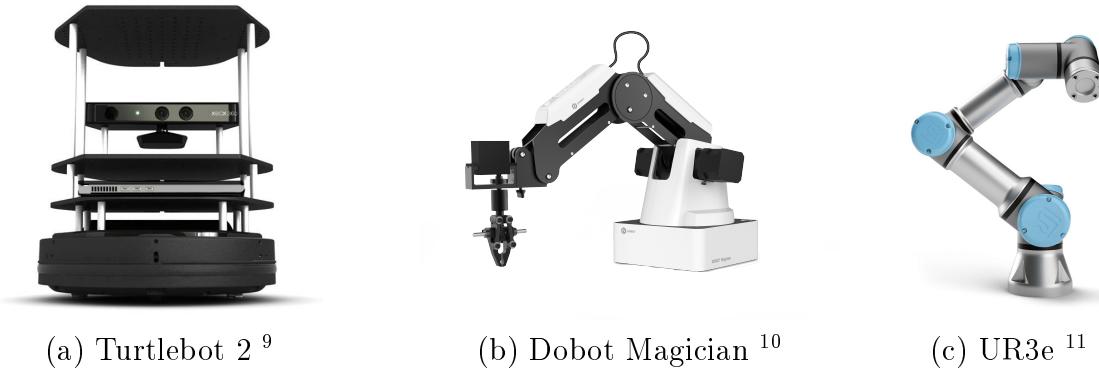


Figura 1.14: Robots usados en universidades

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.

⁹<https://www.turtlebot.com/turtlebot2/>

¹⁰<https://www.robotlab.com/store/dobot-robotic-arm>

¹¹<https://www.universal-robots.com/es/productos/robot-ur3/>

1.3. 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 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.

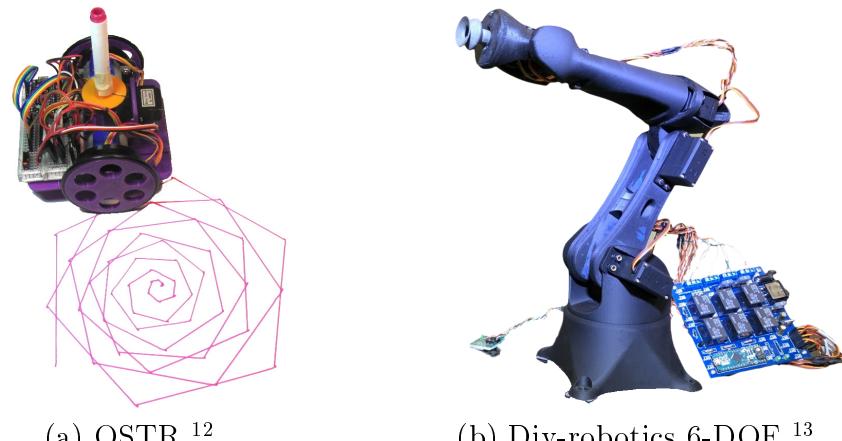


Figura 1.15: Robots de bajo coste

¹²<https://www.instructables.com/Low-Cost-Arduino-Compatible-Drawing-Robot/>

¹³https://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 expone el estado del arte de los robots industriales en educación. Esta fase fundamental de la investigación se ha completado gracias a la búsqueda en diversas fuentes de renombre, con el fin de recopilar información que pueda ser de utilidad para desarrollar el presente proyecto de fin de grado. Como resultado, se han seleccionado una serie de trabajos relevantes y significativos en la materia, que se procederán a analizar a continuación.

- 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). Es llamado Hydra y está dotado de 6 *Degrees Of Freedom* (DOF). Tiene un alcance máximo de casi un metro y un peso que ronda los 13 Kg. Pese a todas estas funciones, 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.

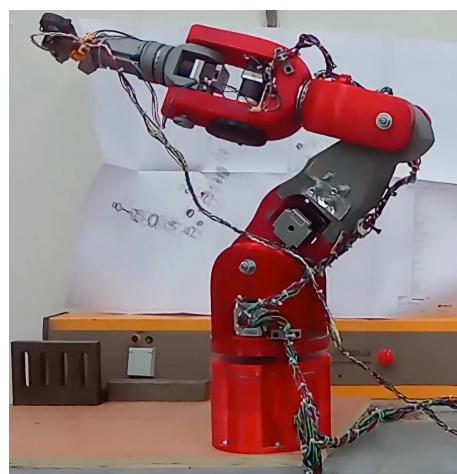


Figura 2.1: Robot HydraX

En base a lo descrito en estos artículos se han extraido los siguientes puntos fuertes del proyecto:

- Tiene una repetitividad de ± 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 120¹

En cambio, también se deben mencionar los siguientes puntos débiles:

- Carece de integración en *Robot Operating System* (ROS), plataforma de desarrollo de la que se habla en 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 y requiere de cierta habilidad para construirlo correctamente.
- En este artículo no se proporcionan los ficheros necesarios para poder replicarlo. 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.

¹<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]. Se trata de 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, se hace uso de un manipulador de tamaño reducido, impreso en 3D y con 4 DOF.

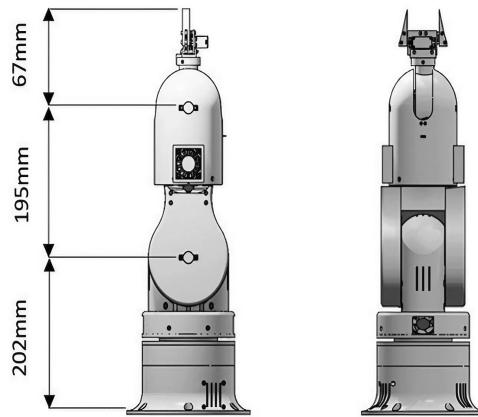


Figura 2.2: Robot UIArm

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.

Cabe destacar los siguientes puntos negativos:

- 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.
- Existe un brazo robot impreso en 3D de 6 grados de libertad que ha ganado gran popularidad gracias a la plataforma KickStarter² 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 lograr esto, apuesta por el uso de componentes comunes y la filosofía *Open Source*.

Aunque este modelo no es el más nuevo de la marca Niryo, si es el que más comunidad tiene y más extendido está. Este robot y los modelos posteriores, pueden ser adquiridos mediante la página oficial³.

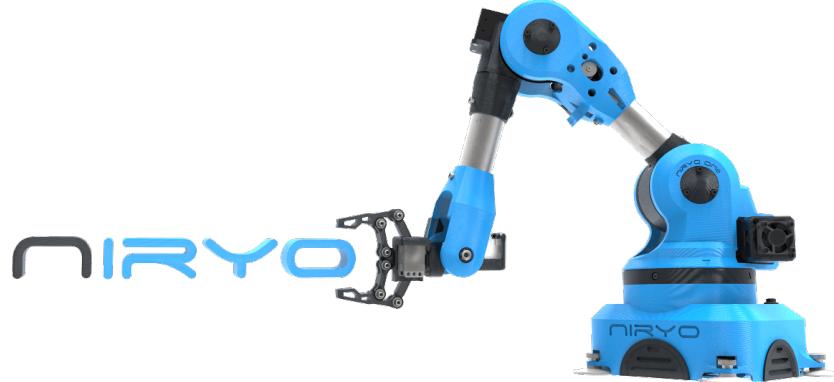


Figura 2.3: Robot Niryo One

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

³<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 github⁴.
- 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 artículo⁵
- 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 Bridge⁶ para comunicar el ROS 1 de la placa con un ordenador moderno con ROS 2.

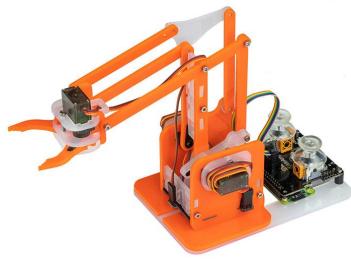
⁴https://github.com/NiryoRobotics/niryo_one

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

⁶https://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*, lo que significa que su código fuente y planos están disponibles de forma gratuita para que cualquier persona interesada pueda acceder a ellos. 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 Thingiverse⁷. 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 montaje⁸ que circulan por internet.

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

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

- Al usar servomotores, se puede establecer el ángulo de cada articulación fácilmente sin necesitar de ningún tipo de sensor externo.
- 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.

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 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.

Con el fin de alcanzar esta 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. 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.
6. 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 GitHub¹, 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 Wiki² 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 dos etapas. La primera, comenzó en octubre y fue abandonada en enero.

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" "3D printed". Además se priorizó aquellos trabajos que utilizaban un 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.
4. *Diseño CAD.* Tras haber definido el concepto y haber seleccionado los componentes físicos que lo componen, se comenzó con el diseño del manipulador.

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

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

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

Llamamos *software* al conjunto de programas, instrucciones y datos que dotan de lógica al sistema.

4.1.1. Python

Python es un lenguaje de programación, es decir, una serie de reglas gramaticales bien definidas que le proporciona a una persona, en este caso el programador, la capacidad de programar una serie de instrucciones o secuencias de órdenes con el fin de crear un comportamiento lógico determinado.

Se trata de un lenguaje de alto nivel interpretado. 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. Esto agiliza el proceso de desarrollo ya que la conversión de código humano a código máquina (compilación) suele demorar un tiempo. Por el contrario, la ejecución de un programa en lenguaje compilado es más rápida.

Una de las características más destacadas de Python es su amplia librería (conjunto de código destinado a un objetivo concreto) estándar, que proporciona variedad de módulos y funciones para realizar multitud de tareas. 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.

4.1.2. Grbl

Grbl¹ es un firmware de código abierto usado para controlar máquinas llamadas CNC. Este firmware se ejecuta en un microcontrolador, que se encuentra dentro de la controladora de la máquina CNC, en nuestro caso, la placa base del robot.

Básicamente, convierte las instrucciones de código G, que posteriormente hablaremos de él, en señales eléctricas que se envían a los motores de la máquina. Además, comprueba los diferentes sensores de la máquina, como pueden ser los finales de carrera, para 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 solo soporta movimientos lineales, en este trabajo se abordará la configuración necesaria para poder usar las articulaciones rotativas del nuestro robot.



Figura 4.1: Logo de Grbl

4.1.3. Código G

G-code, también conocido como RS-274, es el nombre del lenguaje de programación más usado en máquinas de *Control Numérico por Computadora* (CNC). Proporciona control numérico basado en métricas para equipos controlados por *Fabricación Asistida por Computadora* (CAM), como pueden ser, fresadoras y tornos. La programación de este lenguaje 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

¹<https://github.com/gnea/grbl>

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/motor de fresado. Así es como se programan los movimientos necesarios para que la herramienta realice una trayectoria cuadrada en el plano XY:

```
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
```

4.1.4. FreeCAD

FreeCAD² es una aplicación de diseño asistido por ordenador de código abierto y gratuita, utilizada para el modelado de piezas en 3D. Está dirigida al mundo de la ingeniería mecánica y el diseño de productos, pero también es usada en arquitectura y otros campos.

FreeCAD utiliza modelado paramétrico. Se trata de un enfoque de diseño que utiliza parámetros y relaciones entre ellos para crear modelos 3D modificables, lo que permite realizar un cambio en la geometría modificando el valor de un cierto parámetro. También ofrece una variedad de bancos de trabajo (*work benches*) especializados, como *Part Design*, *Sketcher* y *Draft* para adaptarse a diferentes necesidades de diseño. Además, permite incluir nuevas herramientas y funcionalidades creadas por la comunidad, mediante su administrador de complementos.

FreeCAD cuenta con una comunidad activa de usuarios que contribuyen al desarrollo y la mejora continua del software. Gracias a esto, es sencillo aprender a utilizarlo a partir de las numerosas guías de uso y tutoriales.

²https://www.freecad.org/index.php?lang=es_ES



Figura 4.2: Logo de FreeCAD

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. El objetivo principal de ROS es facilitar el desarrollo de sistemas robóticos al proporcionar una infraestructura que permite que los desarrolladores pueden 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 programadores crear software para robots. También proporciona herramientas para la visualización de datos, simulación de robots, depuración y pruebas. Además, cuenta con una comunidad activa de usuarios y desarrolladores que contribuyen con paquetes adicionales y comparten su conocimiento.

Está diseñado para ser modular, lo que permite la creación de sistemas complejos mediante la reutilización de otros componentes existentes.

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.

En este trabajo se utiliza ROS 2, la versión sucesora de ROS. Esta versión incorpora gran variedad de mejoras respecto a su anterior versión. En esta versión los nodos no dependen de un nodo *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 utilizada en este trabajo es *ROS 2 Humble*. Se trata de la última versión de ROS estable disponible para Ubuntu 22.04 LTS.

4.1.6. MoveIt!

MoveIt es una plataforma de desarrollo (*framework*) para manipulación robótica de código abierto que permite desarrollar aplicaciones de manipulación complejas utilizando ROS. Además, proporciona una amplia gama de herramientas y bibliotecas que facilitan el control y planificación de movimientos de robots. Con MoveIt, puedes

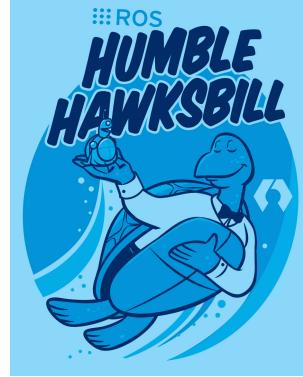


Figura 4.3: Logotipo de ROS 2 Humble

crear fácilmente aplicaciones que permitan a los robots realizar tareas de manipulación avanzadas, como agarrar objetos, ensamblar piezas y mover 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*; se trata de una asistente de configuración con interfaz gráfica que te permite generar los ficheros necesarios para utilizar tu propio robot en este ecosistema.

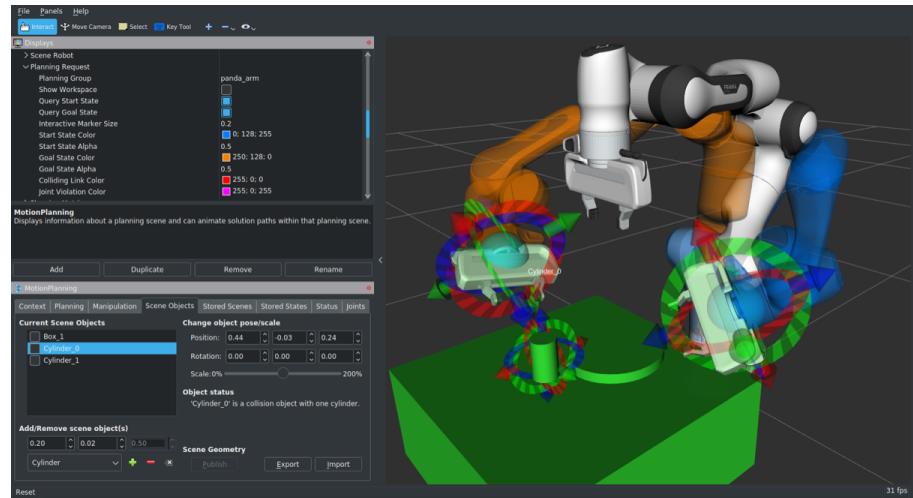


Figura 4.4: Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot)

4.2. Hardware

Cuando hablamos de *hardware*, nos referimos a todos los componentes físicos de utilizados en el dispositivo electrónico, en este caso, un robot. Estos componentes son tangibles, es decir, se pueden tocar y ver.

4.2.1. MKS DLC32

Se trata de una placa destinada al mundo de las máquinas de grabado láser. Ha sido creada por *MakerBase* y es considerada *Open Hardware* por lo que toda la información de la placa puede encontrarse en su repositorio de Github³. Es fácilmente adquirible por *Aliexpress* por un precio que ronda los 16€. Está basada en el microcontrolador de 32 bits: ESP32. Se trata de un dispositivo muy asentado en la comunidad *maker* debido a su bajo coste e integración en el ecosistema Arduino. De hecho, gracias a su conectividad wifi y bluetooth ha ganado terreno a los microcontroladores Atmega que incorporan los propios Arduinos.

Esta placa es ideal para este proyecto debido a que cuenta con la posibilidad de controlar hasta 3 motores y es completamente compatible con Grbl. Además dispone de una salida de potencia regulable controlable mediante Grbl que nos permite alimentar dispositivos. Estos podrían ser: electroimán (tipo de imán que es activado mediante electricidad), motor *Corriente Contínua* (CC) entre otros. Además se puede aprovechar las salidas *Pulse Width Modulation* (PWM) para conectar un grabador láser o un servo. El rango de funcionamiento es de 12 a 24 voltios por lo que es adecuado para ser alimentado mediante baterías y con cargadores de ordenador.

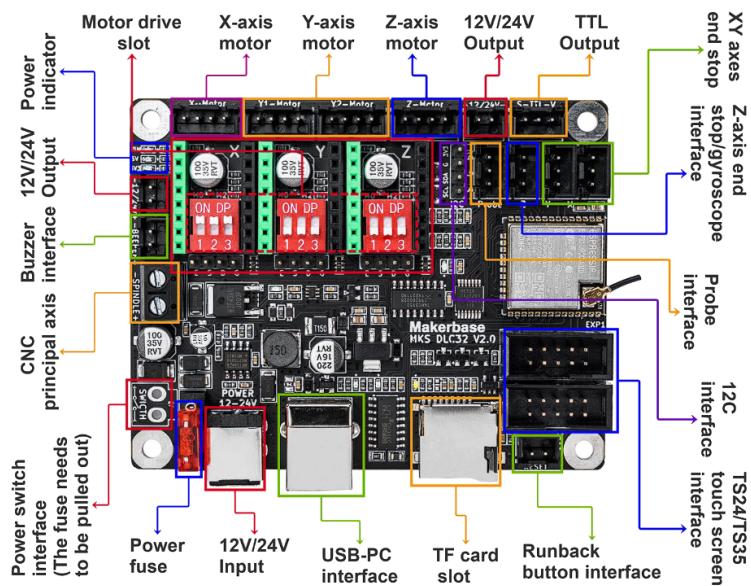


Figura 4.5: Placa base MakerBase DLC32

³<https://github.com/makerbase-mks/MKS-DLC32>

4.2.2. Motores Nema 17

Un motor paso a paso es un tipo de motor que se mueve en pequeños pasos o incrementos discretos en lugar de girar continuamente. Estos pasos son controlados por señales eléctricas que hacen girar al motor una cantidad específica de grados cada vez que se envía una señal. Debido a que se tiene control sobre su avance son una excelente opción si se quiere tener un motor que sea capaz de posicionarse en un ángulo concreto con exactitud. A pesar de su gran precisión, los motores paso a paso convencionales no tienen el conocimiento absoluto de su posición, por lo que todos los movimientos son relativos. Esto hace que se requiera de un *homing* (proceso en el cual la máquina CNC lleva las partes móviles a una posición conocida) en el arranque de la máquina para conocer su estado antes de operar. Para este trabajo, se han usado 3 motores Nema 17 de 60 milímetros de largo debido a las necesidades calculadas más adelante. Son motores bipolares de 2.1 amperios y una resistencia de 1,6 ohmios con un torque máximo de 0.65 newton-metro.



Figura 4.6: Motores Nema 17 de 2.1A

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 trasformar las señales lógicas que le envía el controlador en una serie de pulsos de potencia que excitarán las bobinas del motor en un cierto orden para lograr el movimiento. Existen motores bipolares, unipolares e híbridos. La diferencia entre ellos radica en la disposición de las bobinas de su interior. Los más usados en impresoras 3D y CNC son los bipolares. Son reconocibles debido a que tienen 4 cables. En este tipo de placas base se pueden instalar distintos modelos de controladores. Cada uno de ellos tiene unas prestaciones diferentes y por tanto un precio distinto. Unos ofrecen mayor capacidad de corriente (para controlar motores más grandes), pulsos más suaves que reducen el ruido sonoro y las vibraciones, medición en tiempo real de la corriente consumida para conocer el final de una articulación, entre otras tecnologías.

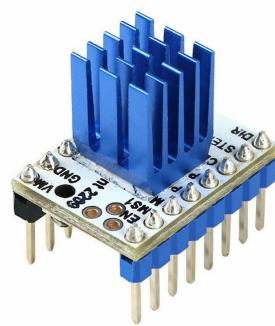


Figura 4.7: Controlador TMC2209

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 utilizar una fuente de alimentación genérica de 24 voltios y 6 amperios. Este tipo de fuentes pueden ser fácilmente adquiribles por internet por un precio aproximado de 15€. A pesar de esto, puede ser usada cualquier tipo de fuente capaz de entregar más de 20 vatios en el rango de voltaje 12-24v. Más adelante se aborda el uso de cargadores de ordenadores portátiles para alimentar el brazo.



(a) Fuente de alimentación 24v 5A



(b) Cargador de portátil genérico

Figura 4.8: Métodos usados para alimentar el robot

Capítulo 5

Desarrollo hardware del manipulador

En este capítulo se aborda el desarrollo necesario para, a partir de un concepto, acabar construyendo un prototipo real funcional. Se hace incisión en cada etapa necesaria para este cometido.

5.1. Eligiendo la geometría del manipulador

En esta sección se expone como es el proceso de encontrar y definir la forma y funcionamiento del robot, en función de los objetivos propuestos, evaluando las distintas opciones para encontrar el que mejor se adapte.

Primero de todo, se comenzar estableciendo el número de grados de libertad. Esto número esta limitado por los requisitos establecidos en la sección 3.2. En base a los requisitos 1 y 5, que limitan en cuanto a precio de fabricación y complejidad de los mismos, se ha optado por usar 3 grados de libertad.

En relación al espacio tridimensional, este cuenta con 6 DOF, 3 de ellos para el posicionamiento (X, Y, Z) y los otros 3 para las orientaciones (RX, RY, RZ).

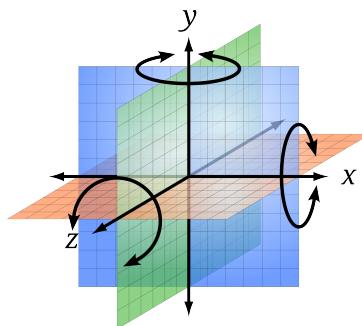


Figura 5.1: Espacio tridimensional

En base a lo anterior, con 3 grados de libertad no podremos situar el elemento terminal del manipulador en un punto del espacio y a la vez controlar la orientación de la herramienta en ese punto.

Seguidamente, se debe elegir el tipo de articulación que se pretende usar. En robótica, son ampliamente utilizados estos tipos de articulaciones:

- 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.
- Esférico: Permite la rotación en cualquier dirección. Está restringido a 3 DOF y suele ser usado como articulación pasiva.
- Cilíndrica: Permite el movimiento de rotación alrededor de un eje y también un desplazamiento lineal a lo largo del mismo. Combina los movimientos rotacionales y prismáticos.

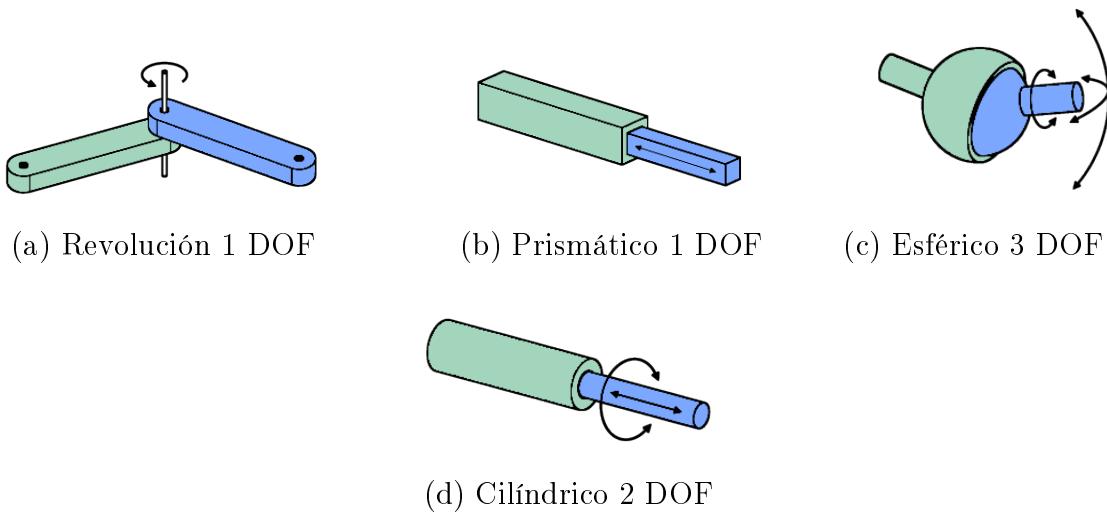


Figura 5.2: Tipos de articulaciones más usadas en robótica

Se ha optado por articulaciones de tipo revolución, debido a que son sencillas de implementar y están compuestas por pocas partes móviles.

Acto seguido, se debe establecer la forma del manipulador en función del tipo y número de articulaciones elegidas anteriormente. En base a las categorías de manipulador descritas en 1.1.2, se ha decidido implementar un robot basado en paralelogramos al estilo MeArm 2.4(a).

El tipo de brazo elegido tiene la peculiaridad de tener su elemento terminal siempre paralelo al suelo por lo que es ideal para tareas *pick and place*.

5.2. Modelo alámbrico

5.2.1. En qué consiste

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 lugar de mostrar detalles realistas del manipulador.

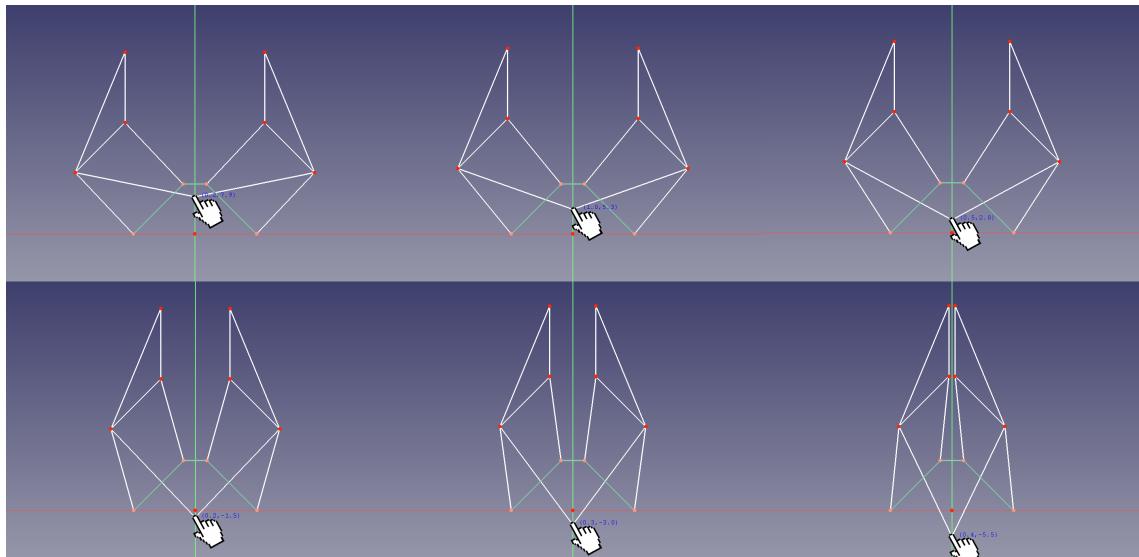


Figura 5.3: Pinza paralela con 1 grado de libertad

5.2.2. Desarrollo del modelo alámbrico de G-Arm

Para desarrollar el modelo alámbrico de este manipulador, se ha tomado como referencia el modelo alámbrico de MeArm del repositorio de github¹.

Para abordar adecuadamente la modificación de modelo, es fundamental comprenderlo en su totalidad. Por suerte, el repositorio anterior contiene una explicación detallada de la razón de ser de cada elemento.

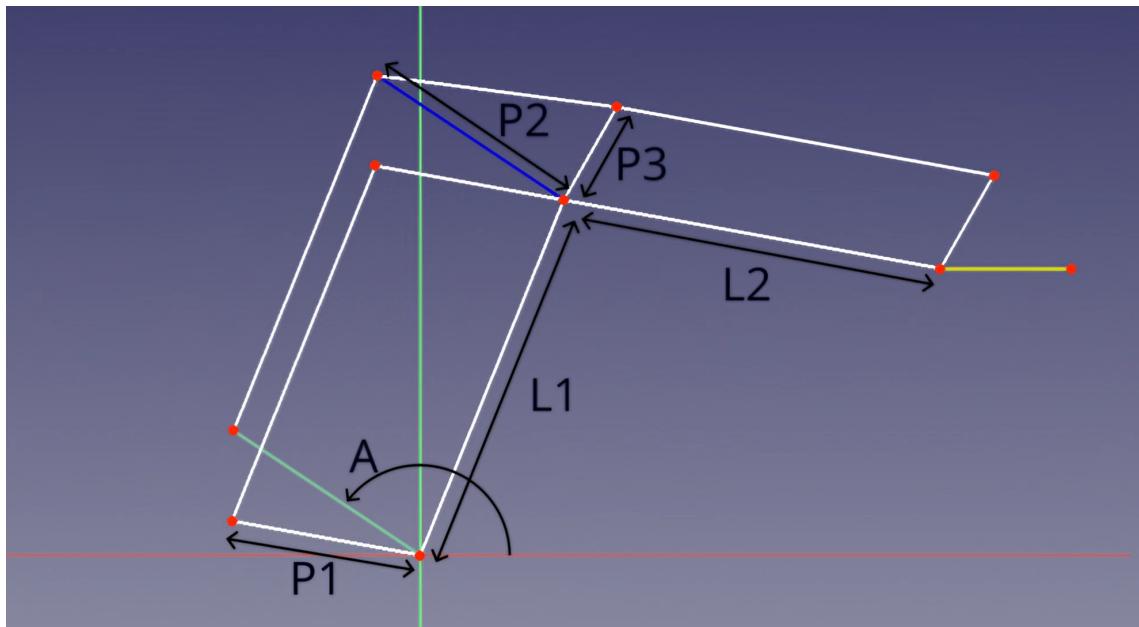


Figura 5.4: Parámetros del modelo alámbrico

Este robot está definido por una serie de parámetros (L_1 , L_2 , P_1 , P_2 , P_3 , A). Para realizar la obtención de los parámetros que definen a G-Arm se ha utilizado la experimentación. El procedimiento ha consistido en modificar una medida y evaluar como se comporta a la hora de alcanzar ciertas zonas críticas. El objetivo final es encontrar una combinación de parámetros que permita alcanzar la mayor cantidad de puntos sin que sus eslabones intersecten entre sí.

¹[https://github.com/myTeachingURJC/Mecatronica/wiki/S3:-Estructuras-mec%C3%A1nicas-\(II\)](https://github.com/myTeachingURJC/Mecatronica/wiki/S3:-Estructuras-mec%C3%A1nicas-(II))

Más concretamente, la obtención de los parámetros adecuados se realizó en el siguiente orden:

1. Una manera de comenzar, es estableciendo de antemano el largo de los eslabones primarios L1 y L2. En base al peso estimado de cada uno de ellos, y de los motores utilizados, ambos deberían medir 17cm o menos.
2. El siguiente paso es elegir P1, P2 y P3. Estos son los lados cortos del los paralelogramos. Es importante jugar con estos valores de tal manera que se consiga obtener los más largos posibles. 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. Codo
3. El ángulo A restringe el paralelogramo que mantiene el extremo del robot paralelo al suelo.

En esta imagen se muestra el modelo alámbrico de G-Arm:

Revisar parametros

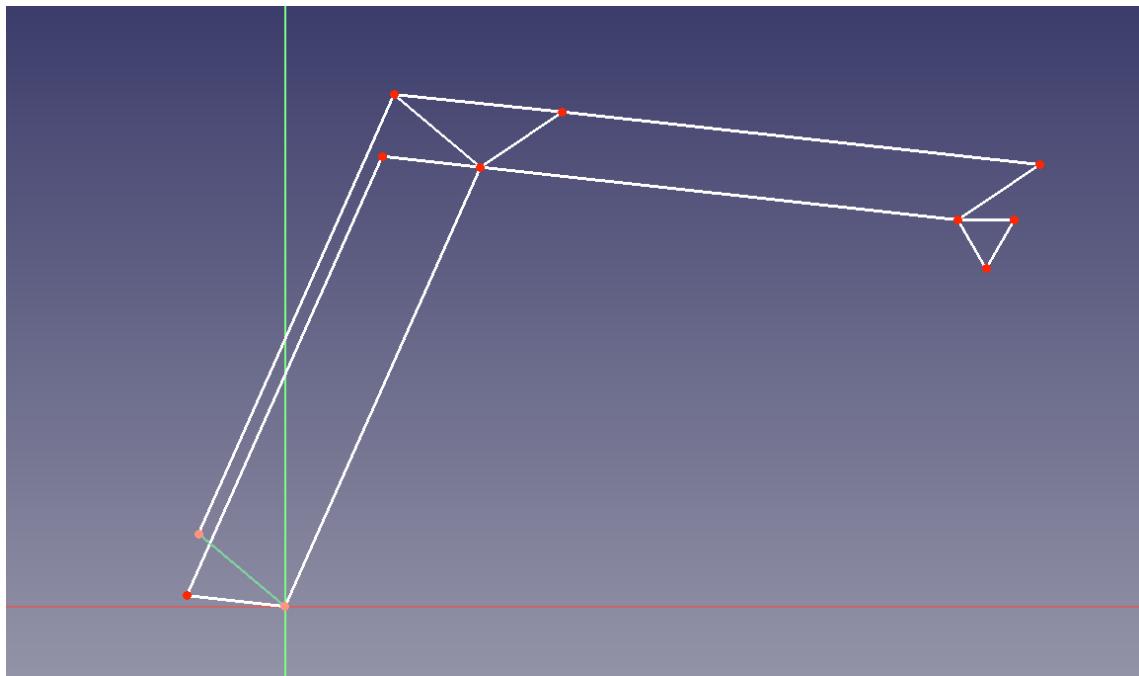


Figura 5.5: Modelo alámbrico de G-Arm

En la siguiente tabla se muestran los valores obtenidos:

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

Cuadro 5.1: Parámetros del modelo alámbrico de G-Arm

5.3. Bocetos

Una parte importante del diseño son los bocetos previos al modelado 3D. En estos bocetos se busca tener una idea clara de la forma y posición de cada pieza, así como de su lugar en el espacio. Para el desarrollo de este brazo, se han realizado numerosos bocetos, de los cuales se conservan aquellos dibujados digitalmente en un iPad.

5.4. Elección de componentes hardware

En esta sección se analiza las necesidades que deben satisfacer los componentes hardware (motores, placas, fuentes de alimentación etc.) y se busca la mejor opción rendimiento-precio existente en el mercado.

5.4.1. Motores

En robótica, los actuadores son los componentes responsables de convertir las señales eléctricas en movimiento físico.

Según la geometría establecida en 5.1, el robot necesita 3 motores. Los utilizados en este proyecto deben cumplir las siguientes características:

1. Deben ser capaces de entregar el suficiente torque para levantar el brazo más una cierta la carga en su extremo.
2. Se debe poder conocer su posición en cada instante.
3. Deben poder mantener su torque sin estar en movimiento.

Debido a esta última necesidad, no es buena idea usar motores convencionales de corriente continua con escobillas, ya que no son capaces de aportar torque sin girar. Por otro lado, los motores paso a paso si son capaces de ello y, aunque no estén codificados,

se puede conocer la posición relativa ya que avanza en pequeños incrementos discretos (pasos). Además, este tipo de motor es capaz de entregar un torque considerable para su tamaño. Por contra, son bastante pesados.

Tras investigar las opciones existentes en el mercado, se ha llegado a la conclusión que los motores Nema 17 son la opción ideal para la realización de este trabajo debido a que son comunes y fáciles de encontrar, además de tener un precio aceptable. Aún así, existen motores Nema de diferentes tamaños, para realizar robots más o menos grandes.



Figura 5.6: Diferentes categorías de motor paso a paso Nema ²

Dentro de la categoría Nema 17, todos tienen el mismo factor de forma a excepción del largo del motor. Esta característica determina el torque del motor. A mayor longitud, mayor torque es capaz de ejercer.

Fotito de lo necesario

²https://filament2print.com/es/blog/139_motores-nema.html

5.4.2. Reductora

Para poder reducir la velocidad de giro de un motor y, a su vez convertirla en fuerza, es necesario implementar una reductora. Existen diferentes formas de hacerlo, como pueden ser los engranajes (normales, planetarios, helicoidales...) y las correas (lisas y dentadas).

Se ha optado por el uso de correas dentadas sobre el uso de engranajes, debido a que estos últimos siempre añaden una cierta holgura al movimiento final, haciendo el robot inexacto y añadiendo incertidumbre en los movimientos. Las correas dentadas en cambio, utilizan tensores para asegurar que siempre está en contacto con ambas poleas.

Foto de correa dentadas Se pretende usar correas de tipo GT2. Este tipo de correa es ampliamente utilizado en impresoras 3D, por lo que es realmente económico y fácil de encontrar. Dentro de esta categoría existen diferentes anchos de correa en función de la tensión soportada. En este caso, se ha elegido el ancho de 6mm puesto que es el más económico con diferencia y es más que suficiente para esta aplicación. En la construcción de robots industriales caseros y profesionales de metal y de gran tamaño, se utilizan correas y cadenas con mayor grosor y tamaño de diente. Foto de mi correa.

Además, las poleas necesarias para este tipo de correas son fáciles de diseñar e imprimir en 3D, lo que permite hacer poleas con un número de dientes concreto y por un precio casi nulo.

En este caso, para dos de los grados de libertad, se ha elegido utilizar una polea comercial de 20 dientes de metal unida al eje del motor en conjunto con otra de 100 dientes impresa en 3D, logrando así una reducción de 1:5. En el grado de libertad restante, se pretende utilizar el mismo concepto pero con una polea de 120 dientes (Reducción 1:6).

5.4.3. Controladores

Debido a que se pretende utilizar motores paso a paso bipolares, es necesario utilizar un tipo de electrónica específica para este tipo de motor. Este componente hardware es llamado controlador, y su objetivo es convertir una señal eléctrica del controlador en una serie de pulsos de mayor potencia que excitarán las bobinas del motor. Para motores paso a paso de baja corriente (hasta 2A), existe un formato de forma común en el mundo *maker*. Es por esto que la mayor parte de los controladores paso a paso que encontramos en el mercado son intercambiables entre sí. La diferencia entre ellos, reside en la tecnología de su interior y las capacidades técnicas que ofrecen.

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€
TMC2208	4-35V	2A	Hasta 1/256	Totalmente silencioso	2.8€
TMC2209	5.5-28V	2.5A	Hasta 1/256	Totalmente silencioso	3.4€

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

La placa base es una tarjeta de circuito impreso que proporciona conexiones físicas y eléctricas entre las diferentes componentes y unifica estos en un mismo componente. En el caso de aquellas usadas para montar controladores paso a paso, suelen integrar un **MCU!** (**MCU!**) o actuar como un *shield* (placa adicional que se acopla a otras para aumentar su funcionalidad y características). Foto cnc arduino shield y otras

Existen numerosas opciones en el mercado con la posibilidad de montar una distinta cantidad de controladores. En el caso de este trabajo necesitamos sólamente 3. Este tipo de placas suelen ser utilizadas para crear máquinas CNC y en este caso se ha optado por elegir una placa económica que integra todo lo necesario para este proyecto a excepción de los controladores. Se ha elegido esta y no otra puesto que tiene un **MCU!** ESP32 muy superior al AtMega328p del Arduino Uno. Además cuenta con la capacidad de poderse controlar vía wifi y la mayor ventaja es que incorpora un mosfet conectado a una salida pwm a la que podemos conectar un motor de hasta x vatios, un electroimán etc. Además incluye una serie de pines y salidas muy cómodas de utilizar para montar módulos de grabado láser y etc. Todo ello por un precio de 16€.

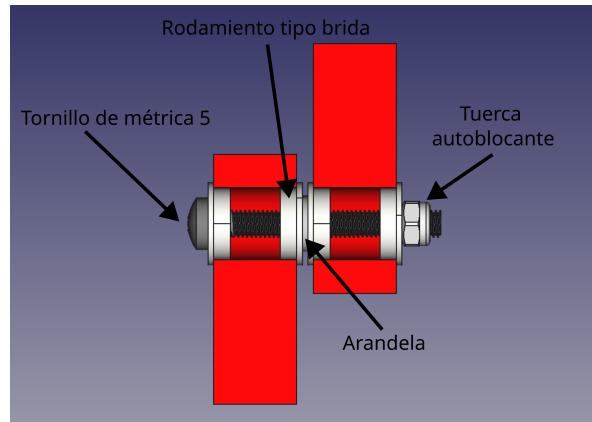
5.4.5. Fuente de alimentación

Se trata de un elemento clave del robot. Es importante conocer *a priori* las características necesarias de la fuente.

Por ello, se ha realizado una estimación del consumo del robot final. La mayor parte del consumo viene dado por los propios motores que se encontrarán permanentemente excitados con el fin de mantener el robot en una determinada posición sin desplomarse. Según las especificaciones de los motores escogidos, cada uno consume en torno a 7W. Es decir, el consumo de los motores es de 21W. A esto hay que añadir el consumo del microcontrolador y de los controladores (unos 4W). Además, si se quiere añadir un electroimán al extremo o algún servo, se necesitará de un extra de potencia. En base a la estimación, se establece que lo ideal y recomendable para alimentar este equipo es una fuente de alimentación de 12 a 24V capaz de entregar al menos 30W. Si se desea usar una fuente de baja calidad, lo ideal es elegir una un 30 % más potente para evitar forzarla y aumentar su vida útil. Debido al amplio rango de voltaje y de su bajo consumo, es perfectamente alimentable con un cargador típico de portátil de 19V sin temor a dañarlo. Pese a esto, se ha decidido utilizar una fuente de alimentacion genérica de 24V 150W para poder realizar *a posteriori* todas las mediciones de consumos descartando a la fuente como factor limitante. El precio de esta fuente es de 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 elegido un tipo de rodamiento específico para este propósito. Se ha optado por el uso de rodamientos brida como el de la Figura 5.7(a). Este tipo de rodamientos son usados en las poleas pasivas de las impresoras 3D, por lo que son muy baratos (10 de buena marca por 6-10€) y sencillos de encontrar. La razón principal de su uso es su peculiar forma. Este tipo de rodamiento tiene un borde en el extremo que hace de borde. Esto lo hace ideal para insertar en las piezas 3D a desarrollar y garantizar que nunca se puedan sacar mientras el tornillo este fijado. Además evita que el rodamiento se mueva y se gire cuando la articulación recibe un esfuerzo lateral. (Véase la Figura 5.7(b))

(a) Rodamientos F695-2RS Fushi⁴

(b) Ejemplo de uso

Figura 5.7: Rodamientos de tipo brida

5.4.7. Finales de carrera

Son útiles para conocer el final del recorrido de una articulación y poder establecer la posición absoluta al inicio de la máquina. Existen cantidad de ellos en el mercado y con distintas tecnologías. Se ha decidido utilizar unos basados en *microswitches* debido a que son baratos y cumplen su función. Además son compatibles con la placa previamente elegida.

³<https://es.aliexpress.com/item/32850989216.html>

⁴<https://es.aliexpress.com/item/32850989216.html>

5.4.8. Electroimán

Para realizar las pruebas, se pretende crear una herramienta para el robot que consiste en un electroimán para poder mover objetos metálicos de sitio. En el mercado existen muchos tipos de electroimán con distintos consumos y fuerzas. Uno ideal para esta aplicación es el *D20H15*. Su nombre indica que es un electroimán de diámetro 20mm y altura 15mm. Según las especificaciones del producto, es capaz de levantar hasta 3Kg. Además, es necesario comprar la versión de 24v para poder alimentarlo con tensiones entre 18 y 24v. En caso de alimentar el robot con 12v, se debería adquirir la versión de 12 para poder aprovechar toda su fuerza. El consumo de este electroimán es de 3W.

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.

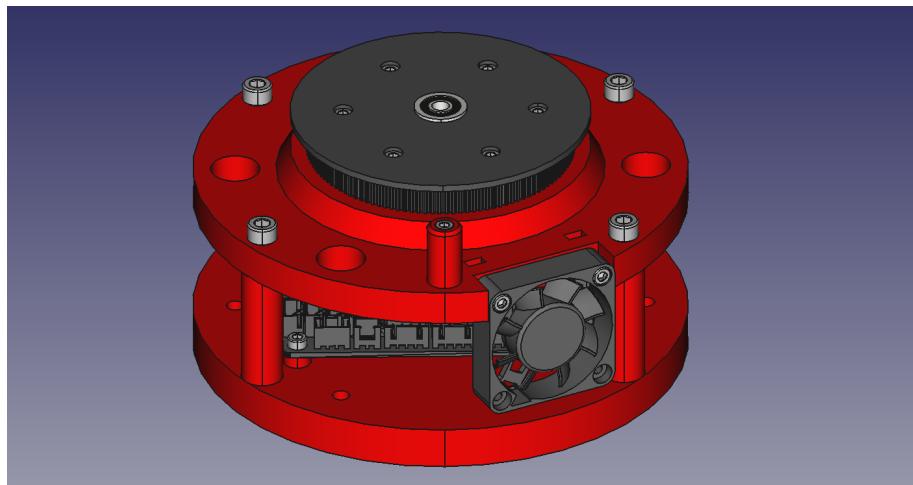


Figura 5.8: Base principal

Pieza circular superior

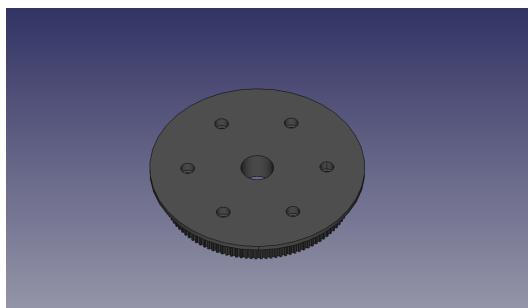
La pieza circular superior (Figura 5.9(b)), incluye una serie de agujeros y huecos hexagonales para insertar las tuercas M3 que permiten acoplar la polea de 120 dientes (Figura 5.9(a)). Además, incluye un rebaje de 40mm en un lateral para poder insertar y atornillar un ventilador 4010.

Espaciadores

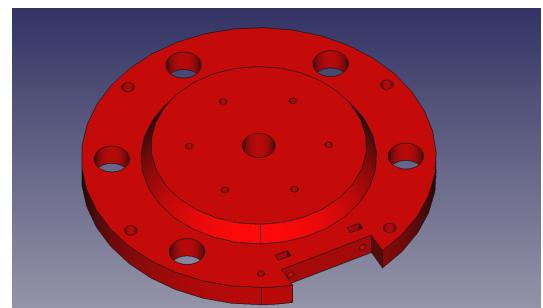
Los espaciadores (Figura 5.9(c)) constan de 4 cilindros perforados con un agujero de 5mm por el que entrarán los tornillos de métrica 5.

Pieza circular inferior

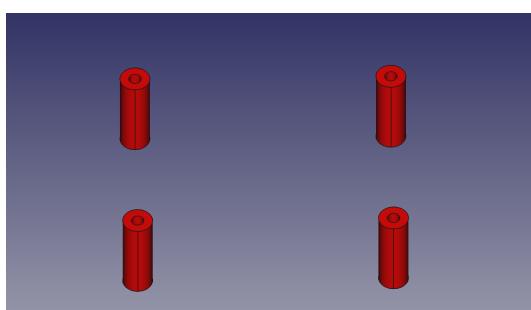
La pieza circular inferior (Figura 5.9(d)), contiene una serie de agujeros para poder atornillar la placa base. Además contiene una serie de agujeros en su perímetro para poder atornillar el robot al suelo.



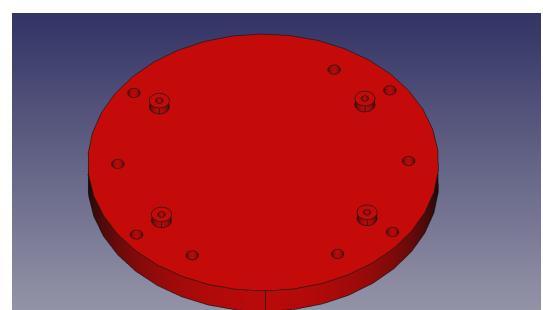
(a) Polea 120 dientes



(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.

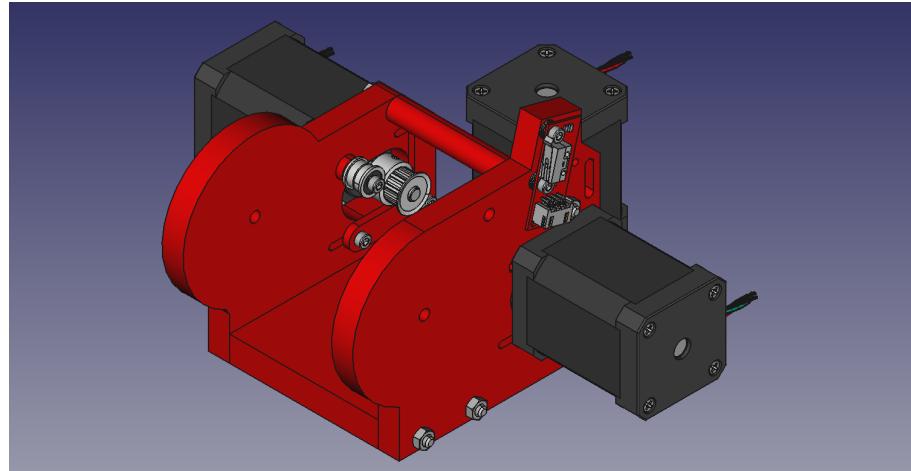


Figura 5.9: Base de los motores

Consiste principalmente en 4 piezas; las dos laterales, la inferior y un espaciador de lado a lado que refuerza el conjunto. Esta ideado así para realizar piezas que serán imprimidas en dirección horizontal que es como mayor resistencia tienen las piezas en impresora 3d por la dirección de las capas. Además se consigue la máxima exactitud en los orificios y formas circulares. Todo el conjunto se une a través de varillas roscadas y tuercas.

Cabe recalcar el sistema de tensado de las correas. Consiste en una serie de orificios longitudinalmente y una pieza que agarra el motor por el otro lado a modo de sandwich. foto de lateral que se vea eso

Además esta pieza incorpora 2 de los 3 finales de carrera del robot. Además tiene una serie de ranuras para poder poner bridales a posteriori y organizar bien los cables.

5.5.3. Paralelogramos

5.5.4. Elemento terminal

Esta pieza (Figura 5.10) 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.

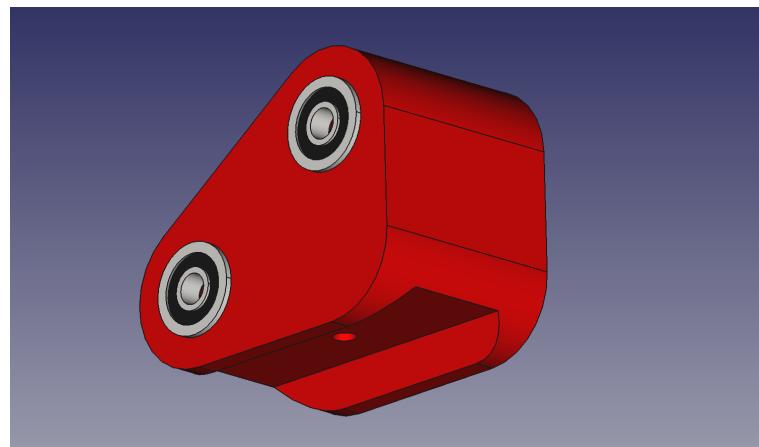


Figura 5.10: Elemento terminal

Consiste en una acanaladura a 45 grados (Figura 5.11) 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.

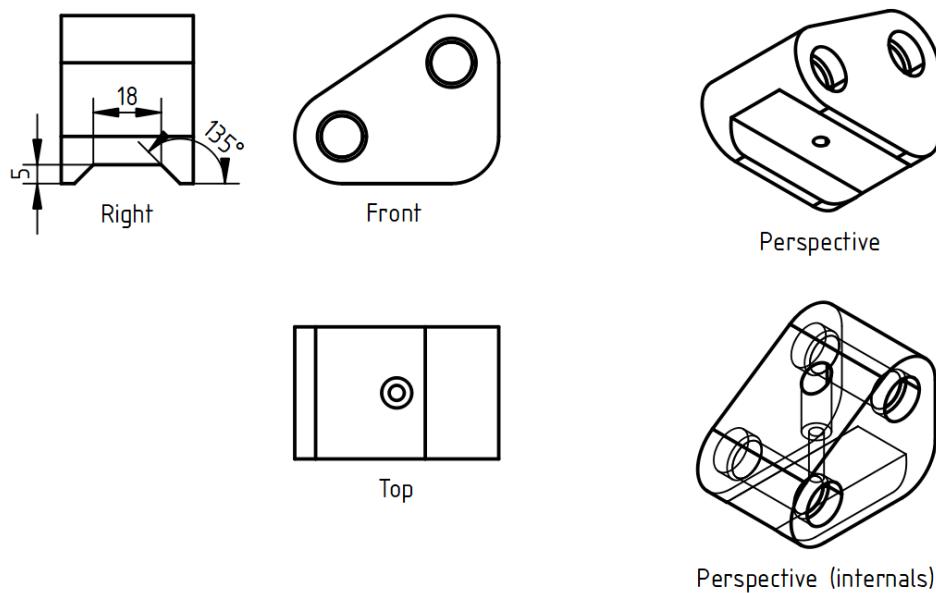


Figura 5.11: Vistas del elemento terminal

5.5.5. Herramienta electroimán

Para dotar al robot de una utilidad, se ha creado una herramienta que contiene el electroimán (Figura 5.12(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 5.12(b)

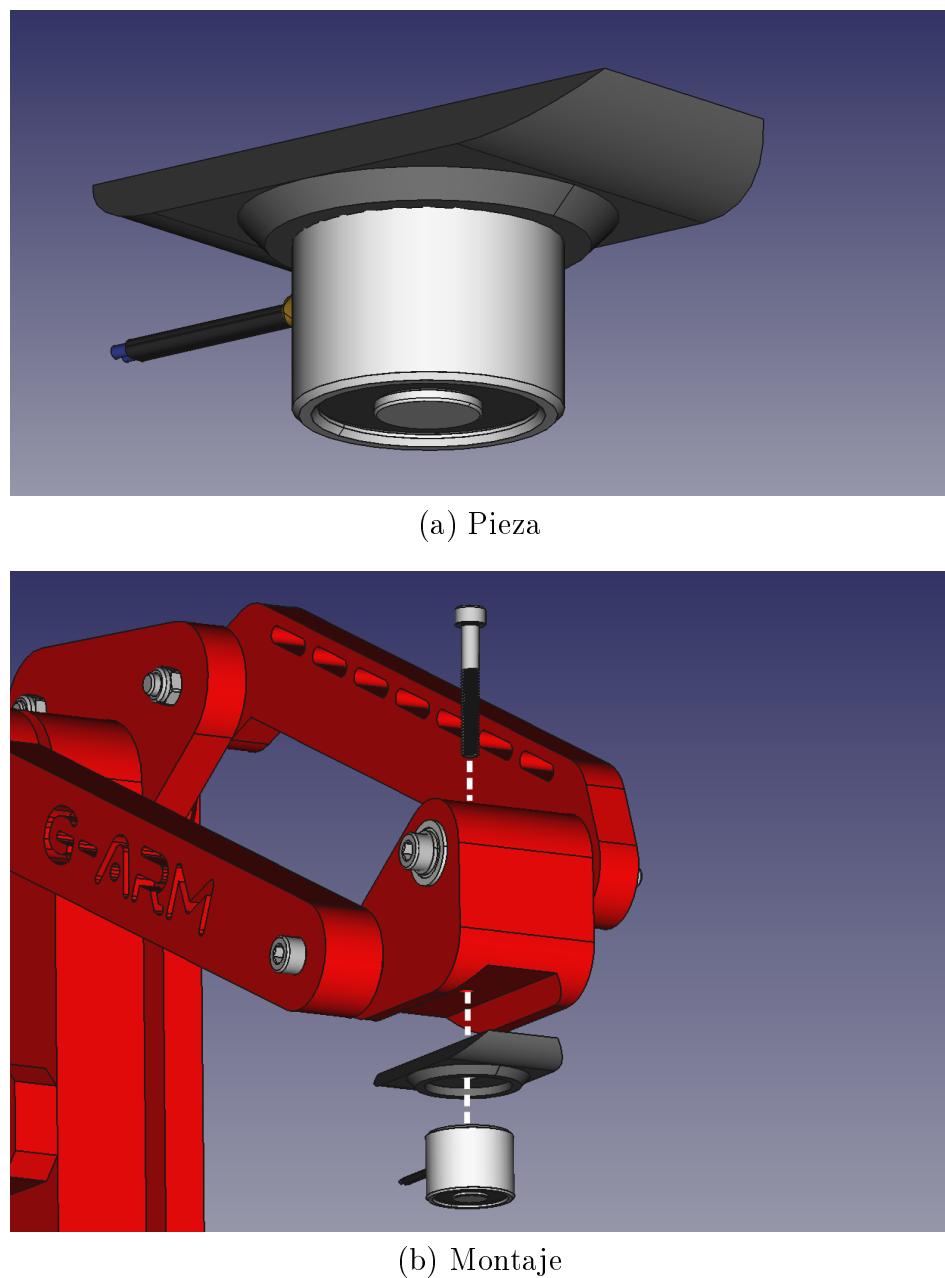


Figura 5.12: Herramienta electroimán

Diseño de las poleas dentadas

Primero, se ha utilizado la herramienta *online gt2-gear-generator*⁵ para generar de manera paramétrica el contorno de la polea en formato DXF.

Posteriormente, abrimos el fichero DXF mediante el *workbench Draft* de FreeCad y

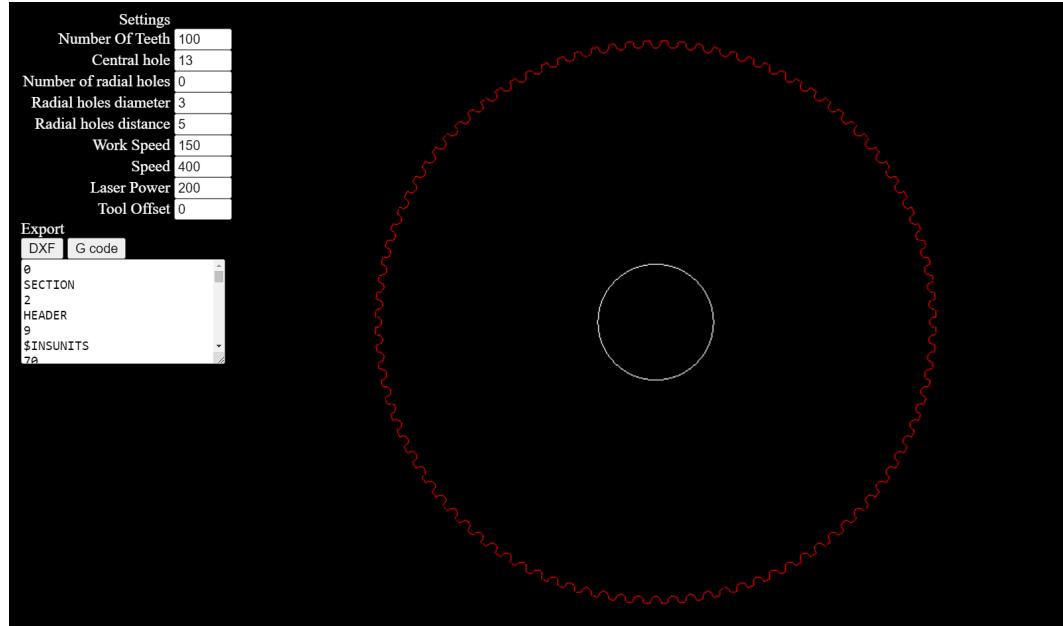


Figura 5.13: Contorno de la polea de 100 dientes

seleccionamos todos los contornos. A continuación, pulsamos en el ícono de "Borrador a Croquis" de la barra superior de herramientas y se nos generará un boceto de FreeCad que podremos extruir con el *workbench Part Design*.

Mecanismo de tensado de la correa

⁵<https://avtehnik.github.io/gt2-gear-genaretor/>

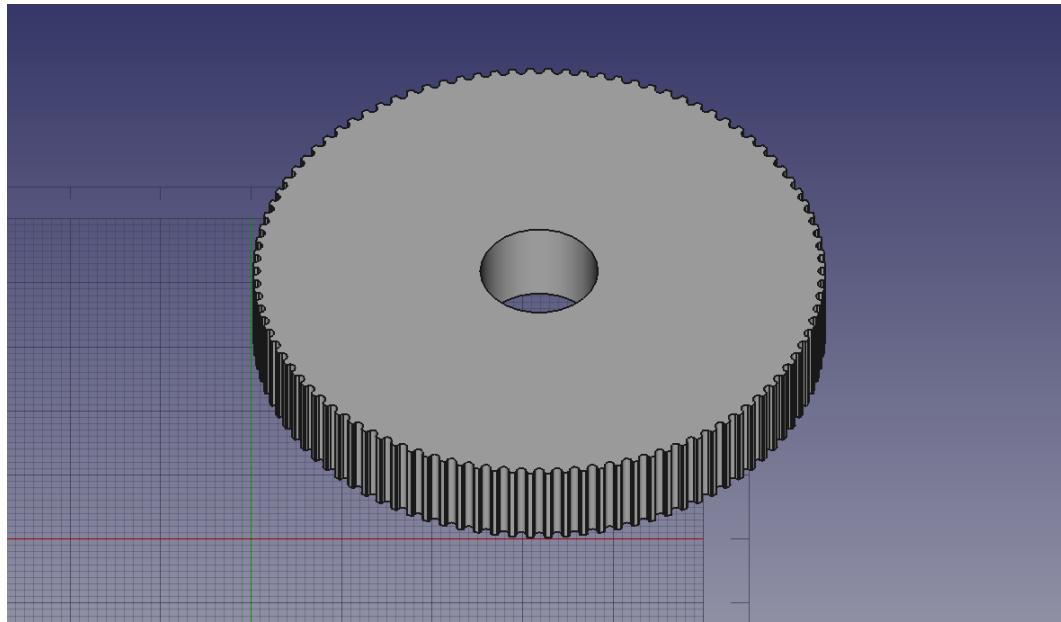


Figura 5.14: Contorno de la polea de extruido

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 Pro⁶ y un rollo de 1Kg de filamento PLA Rojo convencional.



Figura 5.15: Ender-3 Pro V1 2017

⁶<https://www.creality.com/products/ender-3-pro-3d-printer>

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

Componente	Cantidad	Precio total
Tornillo M3 Allen	3	56€

Cuadro 5.4: Tornillería necesaria

El precio total de los componentes necesarios es: 156.3€

Identificador	Cantidad	Relleno óptimo
#1	1	15 %

Cuadro 5.5: Piezas necesarias

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 Grbl4.1.2. Se ha optado por el uso de este *firmware* (software específico para el hardware elegido), 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 3 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 detallan 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 probar si está instalado y probar el funcionamiento de este, 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.

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
?	Pecibir 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 fin de poder abstraer al software del robot de esta comunicación, se ha implementado una clase en Python4.1.1 llamada *Grbl*. En este software, utiliza la librería PySerial¹ para establecer una comunicación serie debido a la facilidad que aporta. En el siguiente bloque de código, se muestra un ejemplo de su uso.

```
import serial

# Puerto '/dev/ttyUSB0' y velocidad 115200 baudios
bus = serial.Serial('/dev/ttyUSB0', 115200) # Comenzamos la comunicación

texto = 'Hola Mundo'
bus.write(bytes(texto.encode())) # Enviar mensaje

bus.close() # Terminamos la comunicación
```

La clase *Grbl* creada puede ser utilizada a través de los siguientes métodos:

```
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.
```

Su código fuente puede ser encontrado en el repositorio del proyecto².

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.

¹<https://pypi.org/project/pyserial/>

²https://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* (Figura 6.1).

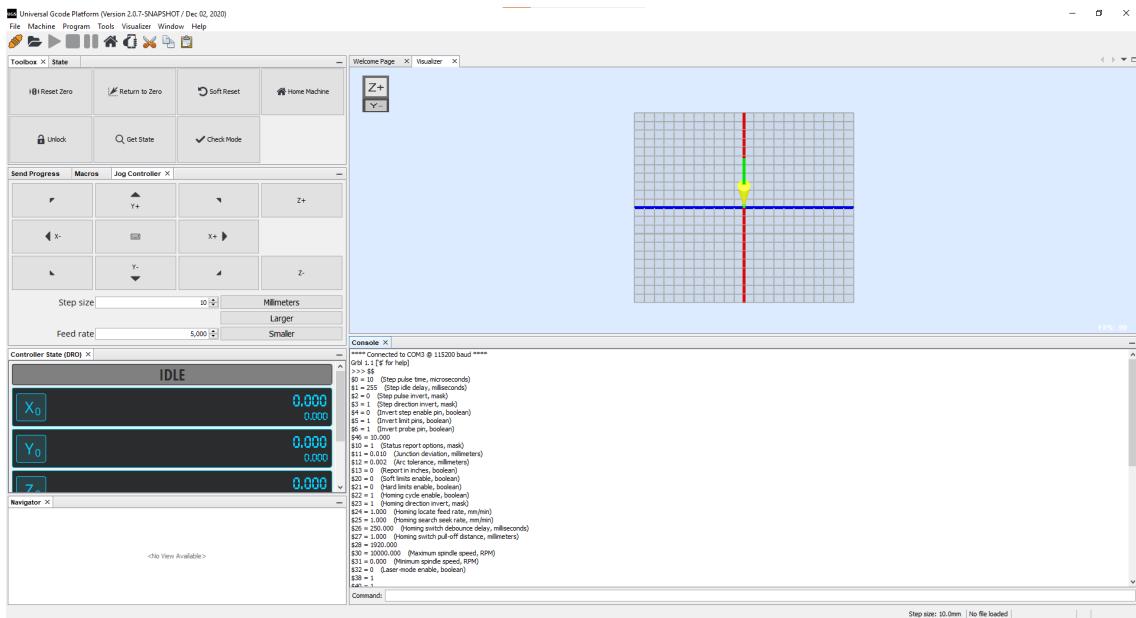


Figura 6.1: Interfaz de UniversalGcodeSender³

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 mantendrán energizados constantemente aunque no se estén moviendo. Es de

³https://winder.github.io/ugs_website/download/

vital 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.

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

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 tambien 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 ⁴ ⁵

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

⁵<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 <buildtool_depend>ament_cmake</buildtool_depend>
2
3 <exec_depend>joint_state_publisher</exec_depend>
4 <exec_depend>robot_state_publisher</exec_depend>
5 <exec_depend>rviz</exec_depend>
6 <exec_depend>xacro</exec_depend>
7
8 <test_depend>ament_lint_auto</test_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 también puede incluir información sobre la masa y la inercia de los eslabones, así como como texturas y modelos 3D.

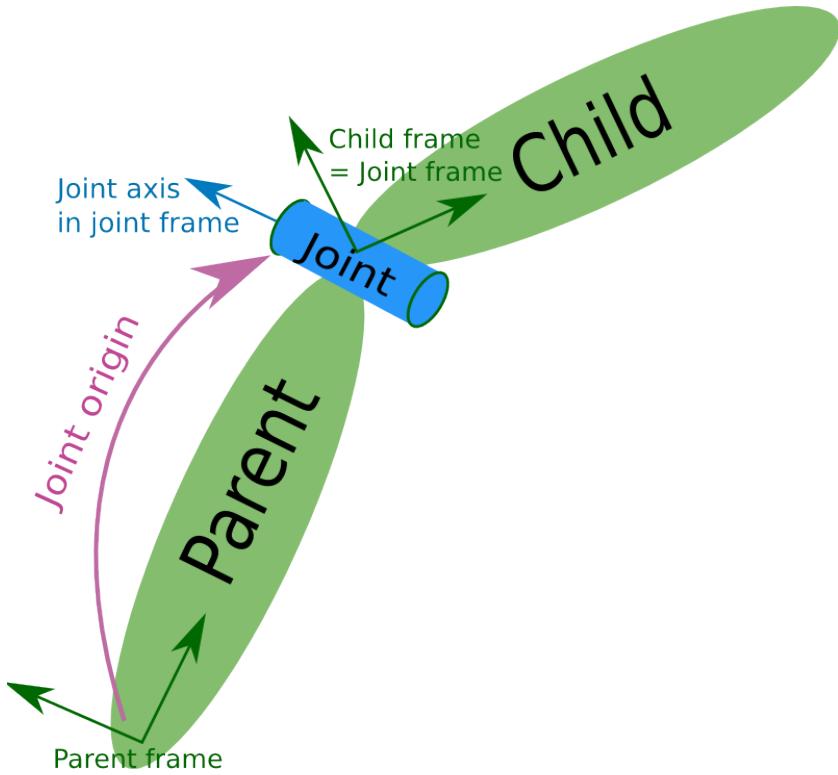


Figura 6.2: Elementos del formato URDF⁶

⁶<http://library.isr.ist.utl.pt/docs/roswiki/urdf%282f%29XML%282f%29Joint.html>

Para que la descripción del robot sea lo más realista posible, se han empleado las propias geometrías del diseño CAD del capítulo anterior. Para poder incluirlas en el URDF, es necesario exportar las piezas al formato *Collada* (.dae). 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). Para generar este tipo de mallas, al igual que los ficheros *Collada*, se puede utilizar el propio FreeCAD para exportarlo directamente a este formato.

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*.

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
```

En este código se muestra un fragmento de la descripción del robot creado que utiliza todo lo mencionado anteriormente:

```

1  <?xml version="1.0"?>
2  <robot name="g_arm">
3
4      <!-- Definition of the motors base link -->
5      <link name="motors_link">
6          <visual>
7              <origin rpy="0.0 0.0 0" xyz="0.0 0.0 0.0" />
8              <geometry>
9                  <mesh
10                     filename="package://g_arm_description/meshes/motors.dae"/>
11             </geometry>
12         </visual>
13         <collision>
14             <origin rpy="0.0 0.0 0.0" xyz="0 0 0" />
15             <geometry>
16                 <!-- Attention: STL file needs to be scaled down 1000
                     percent -->
17                 <mesh
18                     filename="package://g_arm_description/meshes/motors.stl"
19                     scale="0.001 0.001 0.001"/>
```

```

17         </geometry>
18     </collision>
19   </link>
20   <!-- Definition of the arm link 1 -->
21   <link name="link1">
22     <visual>
23       <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0" />
24       <geometry>
25         <mesh
26           filename="package://g_arm_description/meshes/link1.dae"/>
27       </geometry>
28     </visual>
29     <collision>
30       <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0" />
31       <geometry>
32         <mesh
33           filename="package://g_arm_description/meshes/link1.stl"
34           scale="0.001 0.001 0.001"/>
35       </geometry>
36     </collision>
37   </link>
38
39   <!-- Definition joint 2 -->
40   <joint name="joint2" type="revolute">
41     <origin xyz="0.035 0.0331 0.055" rpy="1.5708 0 0" />
42     <parent link="motors_link" />
43     <child link="link1" />
44     <axis xyz="0.0 0.0 1.0" />
45     <limit
46       lower="-1.5707"
47       upper="1.5707"
48       effort="200.0"
49       velocity="200.0" />
50   </joint>
51 </robot>

```

Creación de un launcher para visualizar el robot

Para poder visualizar el fichero que hemos creado anteriormente, podemos realizar tal utilizando una herramienta de ROS llamada RViz. Creamos un fichero terminado en *.launch.py* en la carpeta *launch* y le añadimos el siguiente contenido:

```

1 import os
2 from ament_index_python.packages import get_package_share_path
3
4 from launch import LaunchDescription
5 from launch.actions import DeclareLaunchArgument

```

```
6  from launch.conditions import IfCondition, UnlessCondition
7  from launch.substitutions import Command, LaunchConfiguration
8
9  from launch_ros.actions import Node
10 from launch_ros.parameter_descriptions import ParameterValue
11
12 def generate_launch_description():
13     package_path = get_package_share_path('g_arm_description')
14     default_model_path = os.path.join(package_path, 'urdf',
15         'robot.urdf.xacro')
16     default_rviz_config_path = os.path.join(package_path, 'rviz',
17         'urdf.rviz')
18
19     gui_arg = DeclareLaunchArgument(name='gui', default_value='true',
20         choices=['true', 'false'],
21             description='Flag to enable
22                 joint_state_publisher_gui')
23     model_arg = DeclareLaunchArgument(name='model',
24         default_value=str(default_model_path),
25             description='Absolute path to robot
26                 urdf file')
27     rviz_arg = DeclareLaunchArgument(name='rvizconfig',
28         default_value=str(default_rviz_config_path),
29             description='Absolute path to rviz
30                 config file')
31
32     robot_description = ParameterValue(Command(['xacro ',
33         LaunchConfiguration('model')]),
34             value_type=str)
35
36     robot_state_publisher_node = Node(
37         package='robot_state_publisher',
38         executable='robot_state_publisher',
39         parameters=[{'robot_description': robot_description}]
40     )
41
42     joint_state_publisher_node = Node(
43         package='joint_state_publisher',
44         executable='joint_state_publisher',
45         condition=UnlessCondition(LaunchConfiguration('gui'))
46     )
47
48     joint_state_publisher_gui_node = Node(
49         package='joint_state_publisher_gui',
50         executable='joint_state_publisher_gui',
51         condition=IfCondition(LaunchConfiguration('gui'))
52     )
53
54
```

```

45     rviz_node = Node(
46         package='rviz2',
47         executable='rviz2',
48         name='rviz2',
49         output='screen',
50         arguments=['-d', LaunchConfiguration('rvizconfig')]),
51     )
52
53     return LaunchDescription([
54         gui_arg,
55         model_arg,
56         rviz_arg,
57         joint_state_publisher_node,
58         joint_state_publisher_gui_node,
59         robot_state_publisher_node,
60         rviz_node
61     ])

```

6.2.2. Integración con MoveIt 2

En esta sección se enumeran y describen los pasos necesarios para, apartir 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 instalación⁷ 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 la colisiones descritas anteriormente para generar una matriz de posibles colisiones que se pueden dar. Para ello, pulsamos sobre «*Generate Collision Matrix*».

⁷https://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. Entonces, guardamos los cambios. Ahora, se añaden todos los *links* y *joints* al grupo de planificación creado, dando en «**Edit Selected**»
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).
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* o *joint3*.
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 probar que ha salido todo bien, ejecutamos:

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

Si todo se ha configurado bien deberíamos ver algo parecido a lo mostrado en la Figura 6.3. Para comprobar que resuelve una trayectoria sin problema, podemos establecer dos posiciones aleatorias y ejecutar la trayectoria (Figura 6.4).

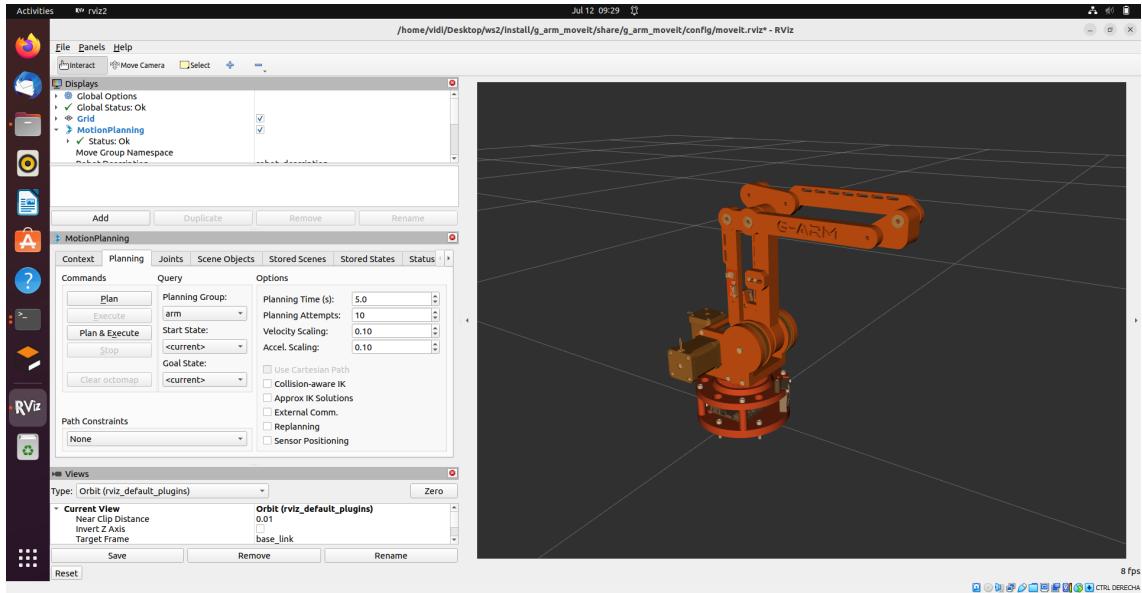
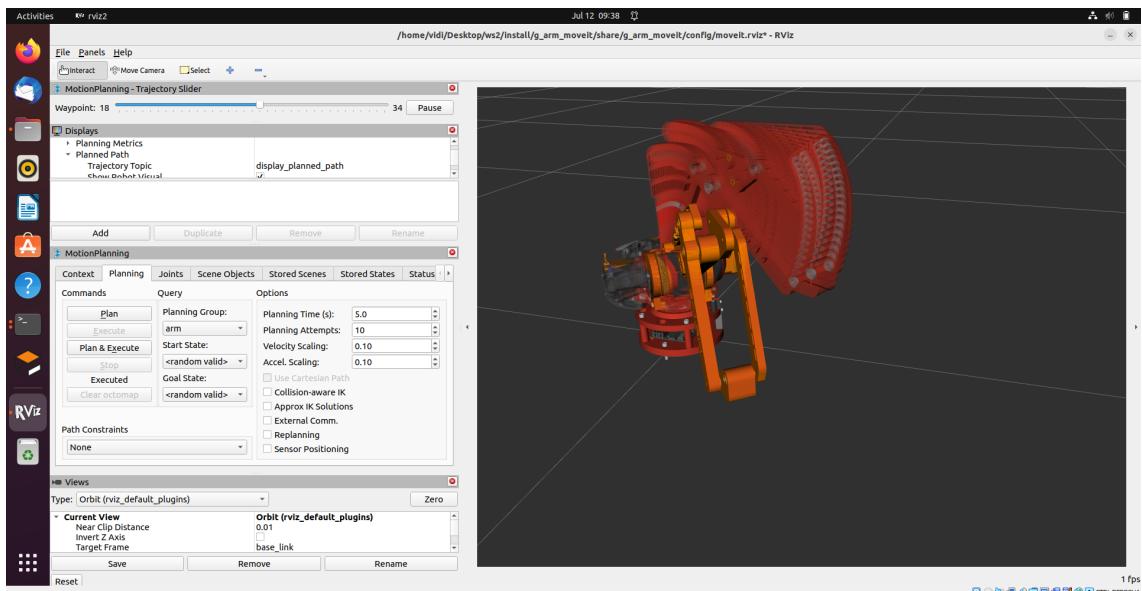
Figura 6.3: Rviz al lanzar el *demo.launch.py*

Figura 6.4: Planificando una trayectoria simple

6.2.3. Nodo ROS

En esta sección se crea un nodo ROS capaz de interpretar los mensajes resultantes de MoveIt y mover el brazo real en función de ellos.

Según la documentación⁸, existe un nodo llamado *move_group* (Véase la Figura 6.5) que se comunica con resto del framework a través de los *topics* y acciones de ROS. Además, se comunica con el robot para obtener información acerca del estado actual (posiciones de las articulaciones, etc.), para obtener nubes de puntos y otros datos de los sensores del robot, y para interactuar con sus controladores.

Principalmente hay que centrarse en un *topic* y una acción de ROS 2.

El *topic* es */joint_states*, el cual es publicado por el nodo que debemos de crear nosotros y es recibido por *move_group*. La acción es */arm_controller/follow_joint_trajectory* (el controlador ha sido nombrado como *arm_controller* por defecto).

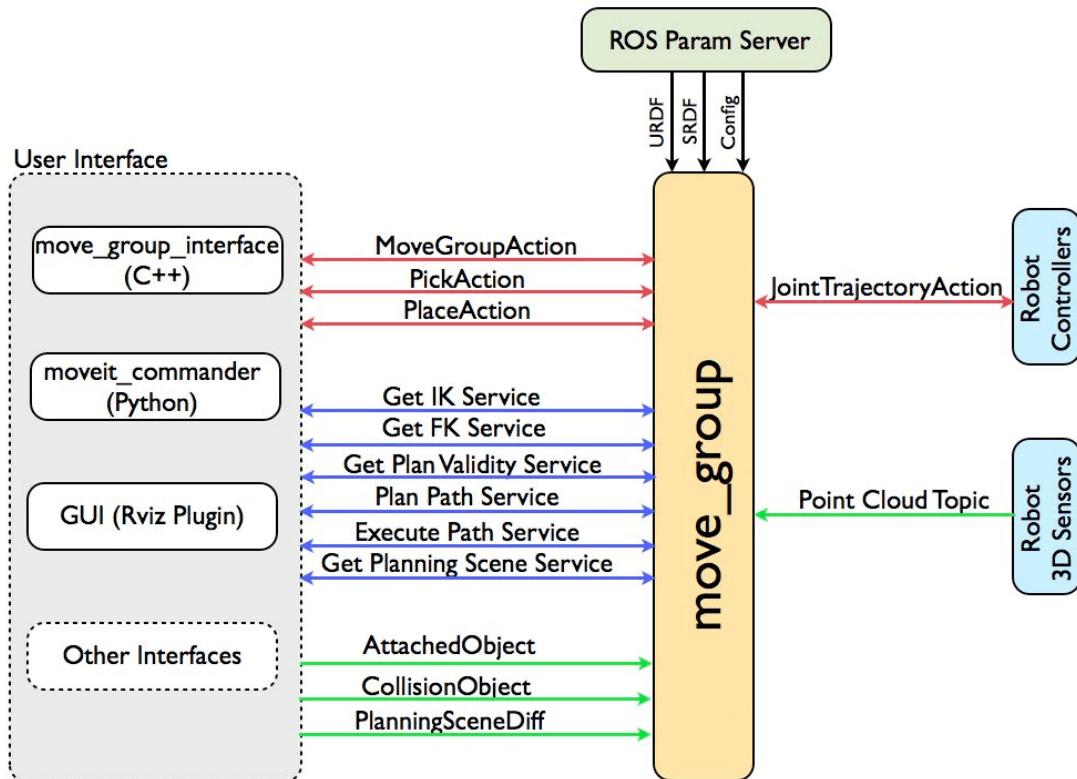


Figura 6.5: Relación de *move_group* con el resto del *Firmware*

Primeramente, es necesario crear tener *launcher*, esto es, una herramienta que se utiliza para iniciar, configurar, gestionar y desplegar nodos. Para ello, se va a hacer

⁸https://moveit.picknik.ai/humble/doc/concepts/move_group.html

uso del ya existente implementado un fichero llamado *real_robot_GUI.launch.py* que se encarga de lanzar una ventana de Rviz con todo lo necesario para controlar de forma gráfica el robot, además de lanzar el nodo *move_group*, entre otros. Este launcher se ha creado a partir de la demo que se crea automáticamente al generar el paquete. Era necesario modificar este para eliminar el controlador falso que se desplegaba para hacer pensar a moveit que había un robot conectado y poder realizar la simulación. Esto es así ya que se desea poder utilizar el robot real. Se puede lanzar este launcher con:

```
ros2 launch g_arm_moveit real_robot_GUI.launch.py
```

Por simplicidad y robustez, se ha optado por utilizar un controlador virtual de ROS Control e imitar el estado de los joints en el robot real. Esto permite desacoplar toda la lógica de control de nuestro robot.

Interpretando tal

6.3. Pruebas

En esta sección se pone a prueba los aspectos técnicos que determinan el desempeño y la fiabilidad del brazo robot. características.

6.3.1. Estabilidad y vibración

Este tipo de pruebas determinan como el movimiento del mismo afecta en su estabilidad y estructura. Además se determina el nivel de vibraciones no deseadas que puedan afectar su precisión y desempeño.

Parámetro	Valor
Capacidad de carga máxima (completamente extendido)	$\pm 9\text{mm}$
Capacidad de carga máxima (medio extendido)	$\pm 9\text{mm}$
Capacidad de carga máxima (contraido)	$\pm 9\text{mm}$
Capacidad de carga máxima (operativa)	$\pm 9\text{mm}$

Cuadro 6.3: Evaluación de la estabilidad

6.3.2. Capacidad de carga

En este tipo de pruebas se evalua la capacidad del brazo a la hora de levantar diferentes pesos. De esta manera se puede determinar el límite de carga del brazo y verificar si puede manejar objetos de manera segura y eficiente sin perder capacidades.

Parámetro	Valor
Capacidad de carga máxima (completamente extendido)	10 kg
Capacidad de carga máxima (medio extendido)	10 g
Capacidad de carga máxima (contraido)	10 g
Capacidad de carga máxima (operativa)	10 g

Cuadro 6.4: Evaluación de la capacidad de carga

Tablita de como afecta el peso a la precisión y demás

6.3.3. Velocidad y tiempo de respuesta

Es importante realizar este tipo de pruebas para evaluar las distintas velocidades que es capaz de manejar el brazo en la ejecución de diferentes movimientos. Además, se puede evaluar la capacidad del brazo para responder rápidamente a comandos y ajustar su velocidad rápidamente.

Parámetro	Valor
Velocidad máxima	10 m/s
Aceleración máxima	10 m/s
Tiempo de respuesta (señal de la herramienta)	35 ms
Tiempo de respuesta (movimiento del brazo)	35 ms

Cuadro 6.5: Evaluación de la velocidad y tiempo de respuesta

6.3.4. Exactitud y repetitividad

Se procede a realizar pruebas para evaluar la precisión del brazo robot en la ejecución de movimientos y la repetibilidad de estos movimientos. Se mide la desviación del brazo robot en comparación con las coordenadas objetivo y verificar si es capaz de alcanzar de manera consistente los mismos puntos en un cierto número de intentos.

Capítulo 7

Conclusiones

Quizás algún fragmento de libro inspirador...

Autor, Título

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo, que básicamente será una recapitulación de los problemas que has abordado, las soluciones que has prouesto, así como los experimentos llevados a cabo para validarlos. Y con esto, cierras la memoria.

7.1. Conclusiones

Enumera los objetivos y cómo los has cumplido.

Enumera también los requisitos implícitos en la consecución de esos objetivos, y cómo se han satisfecho.

No olvides dedicar un par de párrafos para hacer un balance global de qué has conseguido, y por qué es un avance respecto a lo que tenías inicialmente. Haz mención expresa de alguna limitación o peculiaridad de tu sistema y por qué es así. Y también, qué has aprendido desarrollando este trabajo.

Por último, añade otro par de párrafos de líneas futuras; esto es, cómo se puede continuar tu trabajo para abarcar una solución más amplia, o qué otras ramas de la investigación podrían seguirse partiendo de este trabajo, o cómo se podría mejorar para conseguir una aplicación real de este desarrollo (si es que no se ha llegado a conseguir).

7.2. Corrector ortográfico

Una vez tengas todo, no olvides pasar el corrector ortográfico de L^AT_EXa todos tus ficheros *.tex*. En Windows, el propio editor TeXworks incluye el corrector. En Linux, usa aspell ejecutando el siguiente comando en tu terminal:

```
aspell --lang=es --mode=tex check capitulo1.tex
```

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.
- [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).