



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 licenciatante.
- *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*

Índice general

1. Introducción	1
1.1. Robótica	1
1.1.1. Robots de servicio	1
1.1.2. Robots industriales	1
1.2. Robótica industrial	2
1.2.1. SCARA	2
1.2.2. Articulados	2
1.2.3. Paralelos	2
1.2.4. Cartesianos	2
1.3. Robótica educativa	2
1.3.1. Robótica en institutos	2
1.3.2. Robótica en universidades	2
1.4. Robótica de bajo coste	2
2. Estado del arte	3
3. Objetivos	8
3.1. Descripción del problema	8
3.2. Requisitos	9
3.3. Metodología	10
3.4. Plan de trabajo	10
4. Plataforma de desarrollo	11
4.1. Software	11
4.1.1. Python	11
4.1.2. Grbl	12
4.1.3. Código G	12
4.1.4. FreeCAD	13
4.1.5. ROS 2	14

4.1.6. MoveIt!	14
4.2. Hardware	15
4.3. MKS DLC32	16
4.4. Motores Nema 17	17
4.5. Controlador TMC2209	18
4.6. Fuente de alimentación genérica	20
5. Desarrollo hardware del manipulador	21
5.1. Concepto	21
5.2. Modelo alámbrico	24
5.2.1. En qué consiste	24
5.2.2. Modelo alámbrico del brazo MeArm 2.3(a)	25
5.3. Dinámica	25
5.4. Bocetos	25
5.5. Diseño CAD	26
5.6. Impresión y montaje	26
6. Desarrollo software del manipulador	28
6.1. Control de los actuadores	28
6.1.1. Control de motores paso a paso	28
6.2. Comunicación ordenador ⇄ robot	28
6.2.1. Interfaces de comunicación de la placa MKS DLC32??	29
6.2.2. Adaptación de grbl para su uso en robótica	30
6.3. Denavit-Hartenberg	32
6.4. Integración con ROS 2	32
6.4.1. Descripción del robot	32
6.4.2. Nodos	34
6.5. Integración en MoveIt 2	34
6.6. Pruebas	34
6.6.1. Estabilidad y vibración	34
6.6.2. Capacidad de carga	35
6.6.3. Velocidad y tiempo de respuesta	35
6.6.4. Exactitud y repetitividad	35
7. Conclusiones	37
7.1. Conclusiones	37
7.2. Corrector ortográfico	38

Bibliografía

39

Índice de figuras

2.1. Robot HydraX	4
2.2. Robot UIArm	5
2.3. Robots formados a partir de paralelogramos	6
4.1. Logo de Grbl	12
4.2. Logo de FreeCAD	14
4.3. Logotipo de ROS 2 Humble	15
4.4. Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot)	15
4.5. Placa base MakerBase DLC32	16
4.6. Motores Nema 17 de 2.1A	17
4.7. Controlador TMC2209	18
4.8. Métodos usados para alimentar el robot	20
5.1. Tipos de articulaciones más usadas en robótica	22
5.2. Fanuc SR-3iA	23
5.3. REBEL-4DOF-01	23
5.4. Robots basados en paralelogramos	24
5.5. Ejemplos de robots cartesianos	24
5.6. Pinza paralela con 1 grado de libertad	25
5.7. Ender-3 Pro V1 2017	26
6.1. Diagramas de tiempo de UART	29

Listado de códigos

Listado de ecuaciones

6.1. Cálculo de pasos por grado en Grbl	31
---	----

Índice de cuadros

4.1. Especificaciones técnicas de los motores utilizados	18
4.2. Especificaciones técnicas del TMC2209	19
5.1. Parámetros del modelo alámbrico de G-Arm	25
5.2. Componentes hardware necesarios	27
5.3. Tornillería necesaria	27
5.4. Piezas necesarias	27
6.1. Parámetros Grbl usados en este trabajo	31
6.2. Tabla resultante de Denavit-Hartenberg	32
6.3. Evaluación de la estabilidad	35
6.4. Evaluación de la capacidad de carga	35
6.5. Evaluación de la velocidad y tiempo de respuesta	35

Capítulo 1

Introducción

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

Winston Churchill

1.1. Robótica

donde viene este término de las esntanforms y primero robot

Los robots se pueden clasificar en dos grupos.

1.1.1. Robots 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 y ayudar en diversas actividades. Debido a la necesidad de interactuar con los humanos, 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 en rescates

Robots en el espacio

Robots en medicina

1.1.2. Robots industriales

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 remplazar a un operario humano en tareas aburridas, sucias, peligrosas y exigentes (*4D: Dull, Dirty, Dangerous and Demanding*).

1.2. Robótica industrial

1.2.1. SCARA

1.2.2. Articulados

1.2.3. Paralelos

1.2.4. Cartesianos

1.3. Robótica educativa

1.3.1. Robótica en institutos

1.3.2. Robótica en universidades

1.4. Robótica de bajo coste

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.

En base a lo descrito en estos artículos se han extraído 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

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

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.

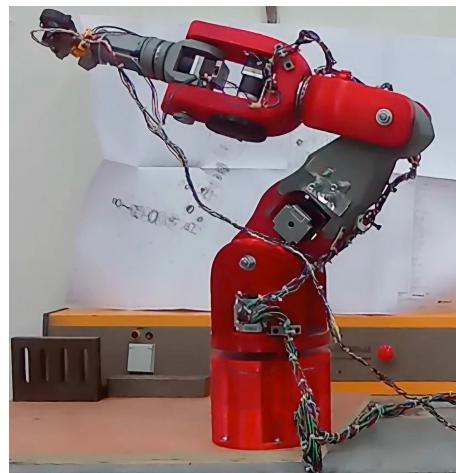


Figura 2.1: Robot HydraX

- 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. 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.

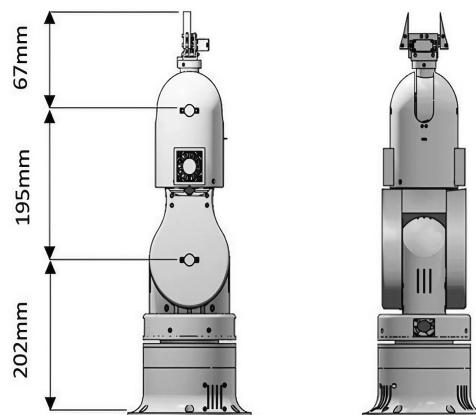
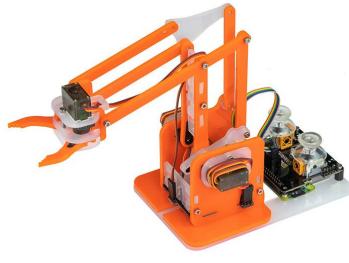


Figura 2.2: Robot UIArm

- 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 Thingiverse². 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 2.3: Robots formados a partir de paralelogramos

²<https://www.thingiverse.com/search?q=mearm&page=1&type=things&sort=relevant>

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 montaje³ que circulan por internet.
- 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.

³<https://docs.rs-online.com/2bb1/0900766b81593e58.pdf>

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 nuestra universidad, donde la escasez de robots disponibles y la dificultad para acceder a ellos han limitado nuestra experiencia práctica en robótica. Estos dispositivos son costosos, lo que limita su disponibilidad, y su fragilidad impide que los estudiantes interactúen plenamente con ellos por temor a dañarlos. Además, la cantidad limitada de robots en relación con el número de carreras que quieren utilizarlos, crea una gran competencia por su uso. Sumado a ello, los profesores tienen que hacer frente a la burocracia asociada al uso de los mismos. La solución propuesta busca superar estas limitaciones al proporcionar a la enseñanza un robot casero impreso en 3D, que será económico, accesible y útil en el aprendizaje práctico de los estudiantes. Por lo tanto, 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 asequible y fácilmente replicable haciendo uso de las impresoras 3D de la universidad.

Con el fin de alcanzar esta meta, se ha dividido el proyecto en los siguientes subobjetivos:

1. Realizar una investigación acerca de los robots que actualmente están disponibles y que cumplan con las características y objetivos deseados. Estos robots deberán

tener un tamaño y costo similar, y preferiblemente haber sido impresos en 3D. Se dará mayor relevancia a aquellos que utilicen un software y hardware libres.

2. Explorar diversas opciones de diseño para determinar la forma ideal del robot. Se analizará cuidadosamente qué tipo de robot se ajusta mejor al uso que se le pretende dar, así como los grados de libertad necesarios.
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 de construir un robot eficiente y funcional. Se llevará a cabo un análisis detallado de los precios y características de cada componente, para seleccionar aquellos que ofrezcan la mejor relación calidad-precio. Una vez evaluados todos los aspectos, se elegirá un conjunto final de componentes para construir el robot de manera efectiva.
4. Realizar el *Diseño Asistido por ordenador* (CAD) del brazo. Se pretende hacer uso de FreeCAD, entre otras herramientas, para diseñar en 3D cada pieza que constituirá el robot. Puesto a que debe ser imprimible en una impresora FDM convencional, debe estar pensado para no necesitar soportes y utilizar el mínimo material posible.
5. Emplear una impresora 3D convencional para materializar los diseños realizados anteriormente.
6. Programar el software necesario para poder controlar el robot desde el ordenador.
7. Realizar la integración del robot realizado en el ecosistema ROS.

3.2. Requisitos

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

1. Se espera un brazo robot de tipo industrial de bajo coste cuya fabricación completa esté por debajo de 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 de 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 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. //TODO

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

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

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

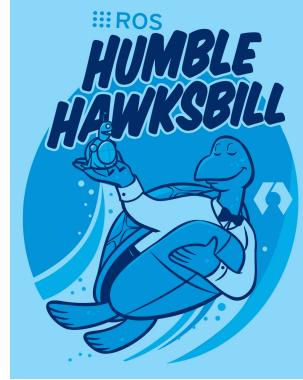


Figura 4.3: Logotipo de ROS 2 Humble

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.

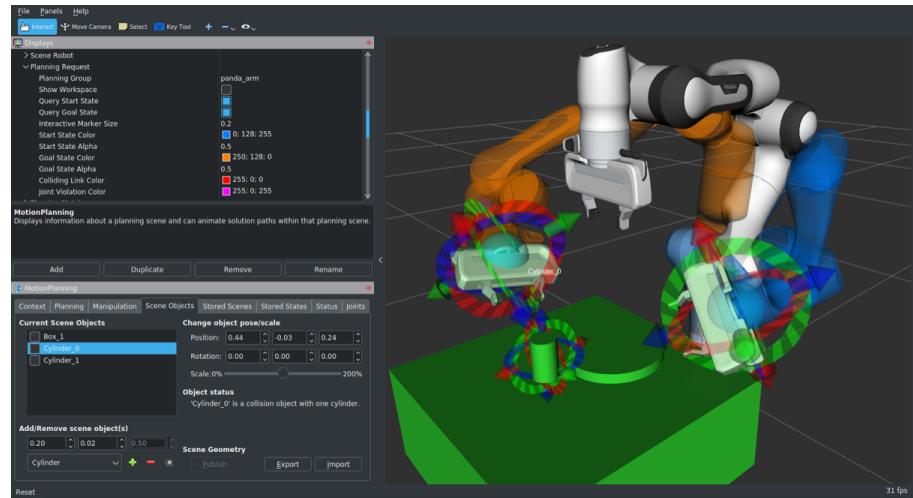


Figura 4.4: Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot)

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.

4.3. 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.

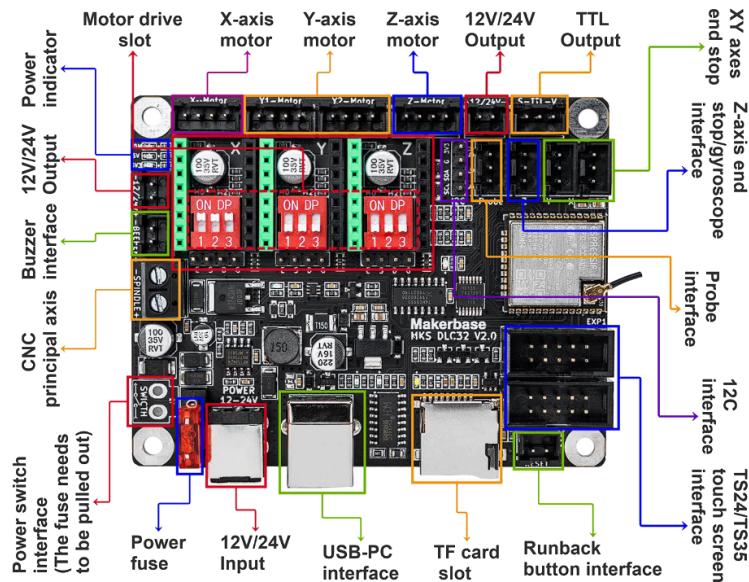


Figura 4.5: Placa base MakerBase DLC32

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

4.4. 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

4.5. 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.

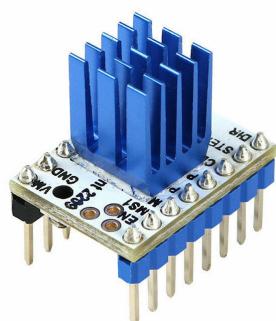


Figura 4.7: Controlador TMC2209

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

4.6. 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.

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo (y quizás alguno más) es donde, por fin, describes detalladamente qué has hecho y qué experimentos has llevado a cabo para validar tus desarrollos.

5.1. Concepto

En esta sección se expone como ha sido el proceso de encontrar y definir la idea fundamental del robot, en función de los objetivos propuestos, evaluando las distintas opciones para encontrar el que mejor se adapte. Se trata de balance

Primero, se debe conocer el numero de grados de libertad que se ajuste a 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 decidido limitar los grados de libertad a un máximo de 4.

hablamos del tipo de joints que existente Existen muchos tipos de joints entre los que destacan:

- Revolución: Este tipo de *joint* permite el movimiento de rotación alrededor de un eje fijo. Está restringido a 1 DOF. Es comúnmente usado en todo tipo de robots, sobre todo en industriales.
- Prismático: En este tipo de *joint*, las partes del robot se pueden desplazar linealmente a lo largo de un eje específico. Está restringido a 1 DOF. Es usado principalmente en los carros de avance de robot industriales y en máquinas CNC.

- Esférico: Esta articulación permite la rotación en cualquier dirección. Está restringido a 3 DOF y suele ser usado como articulación pasiva, es decir, que su movimiento depende de restricciones externas.
- Cilíndrica: Esta articulación 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. Está restringido a 2 DOF. Es usado en el extremo de los robots tipo *Scara*.

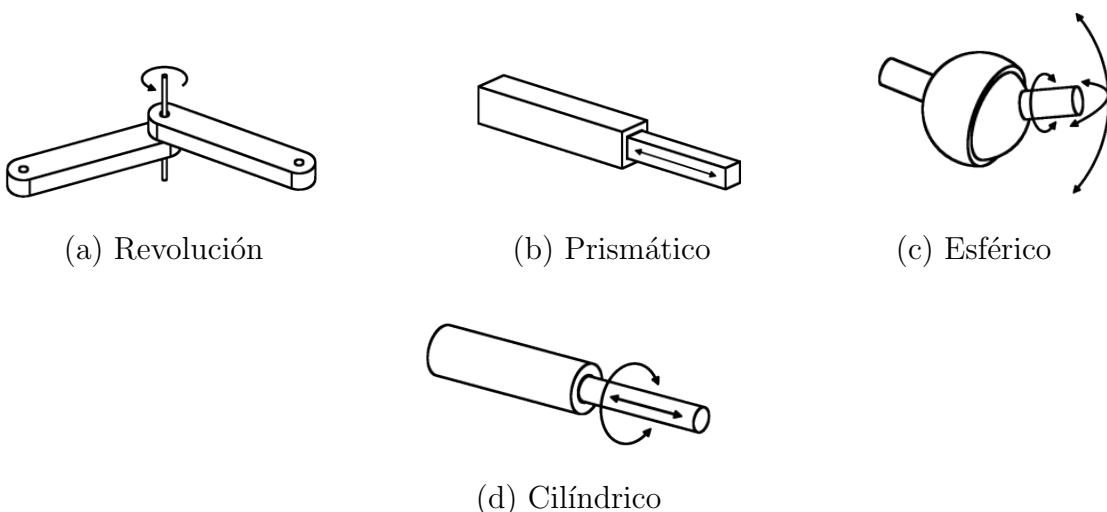


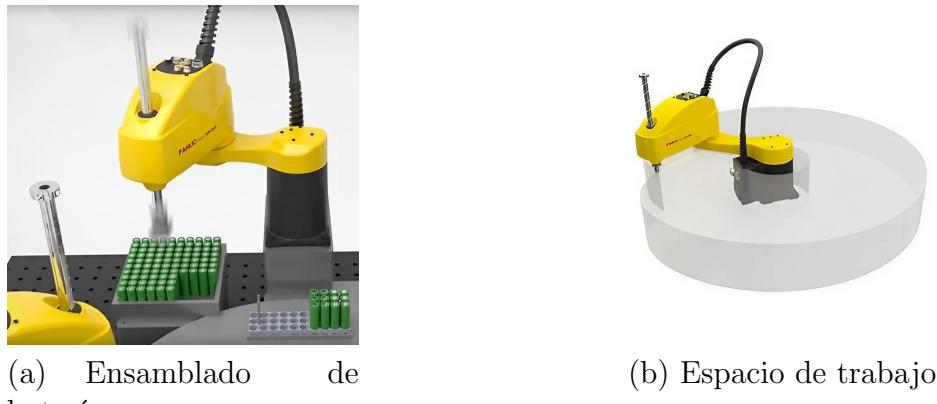
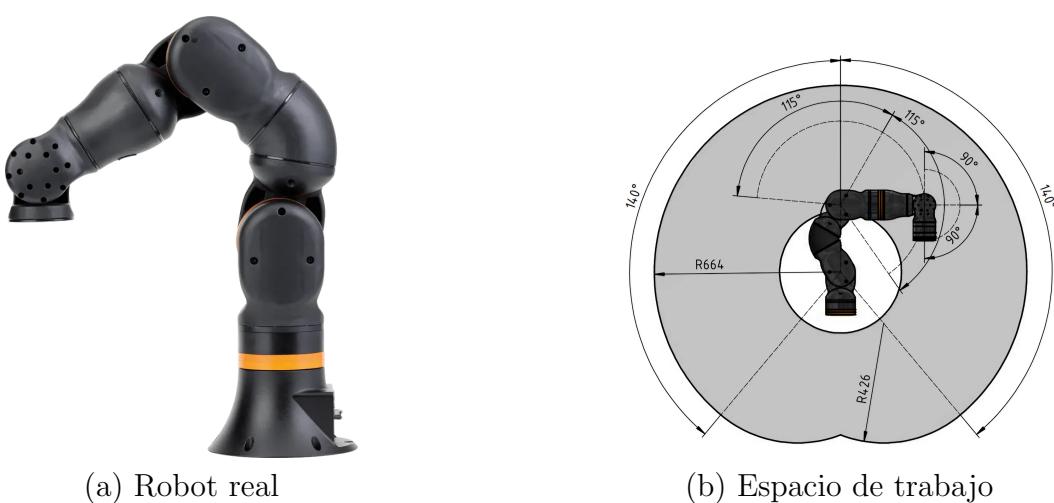
Figura 5.1: Tipos de articulaciones más usadas en robótica

Con hasta 4 grados de libertad podemos realizar los siguientes robots: Tipos:

1. Robots SCARA: son una categoría de robots industriales ampliamente utilizados en aplicaciones de ensamblaje electrónico, operaciones de *pick and place* (coger componentes y situarlos en una determinada posición) y empaquetado de productos, entre otras. Se caracterizan por su diseño de brazo articulado y su capacidad para realizar movimientos rápidos y precisos en un plano horizontal. Este tipo de robot suele tener tres o cuatro grados de libertad, en función de si en el extremo del robot puede rotar sobre si mismo o no.
2. Robots articulados: estos robots tienen una estructura similar a un brazo humano con múltiples articulaciones y ejes rotatorios. Suelen tener entre 4 y 6 ejes y permiten movimientos complejos y flexibles.
3. Robots basados en paralelogramos:

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

²<https://www.igus.es/product/20962?artNr=REBEL-4DOF-01>

Figura 5.2: Ejemplo: FANUC SR-3iA¹Figura 5.3: Ejemplo: REBEL-4DOF-01²

- El robot Delta es un tipo de robot paralelo de 3 (o más) grados de libertad compuesto por dos bases unidas por tres cadenas cinemáticas basadas en el uso de paralelogramos. La base superior se encuentra fija mientras la base inferior, donde se ubica el efecto final, es móvil y siempre está paralela a la base fija. Son rápidos y precisos, por lo que se utilizan en aplicaciones que requieren alta velocidad, como en la industria alimentaria o en la selección y empaquetado de productos.

M-410iC/185

<https://www.fanuc.eu/uk/en/robots/robot-filter-page/m-410-series/m-410ic-185>

4. Robots cartesianos: se mueven sobre rieles/guías a lo largo de tres ejes cartesianos (X, Y, Z) y cuyos ejes forman ángulos rectos unos respecto de los otros. Se trata

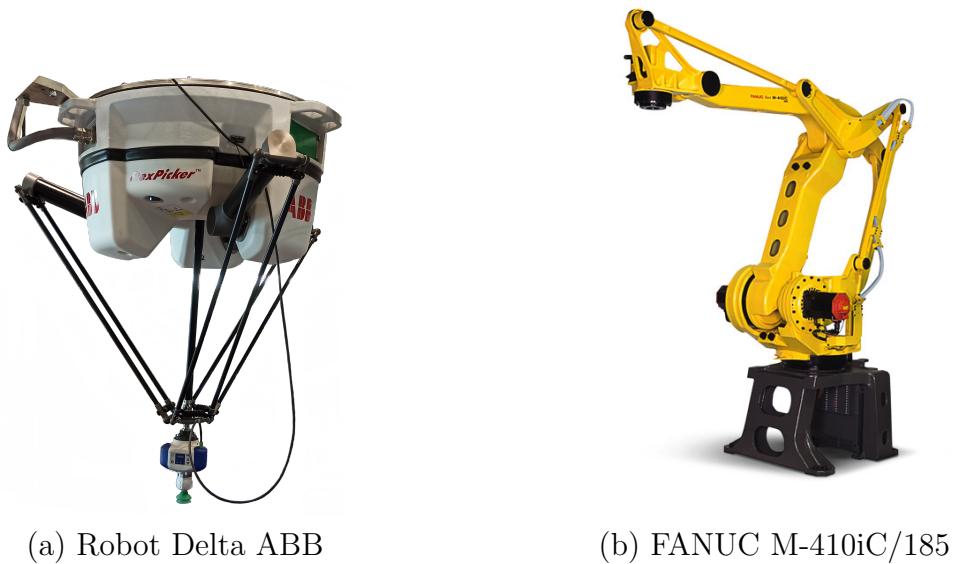


Figura 5.4: Robots basados en paralelogramos

de un tipo de máquina muy común en la industria.

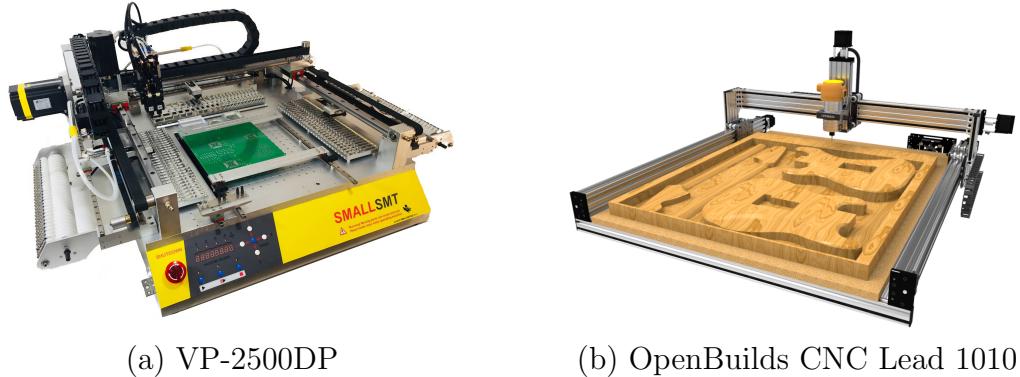


Figura 5.5: Ejemplos de robots cartesianos

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

<https://www.orion-industry.com/pick-and-place/VP2500DP-WNE-eng.pdf>

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.

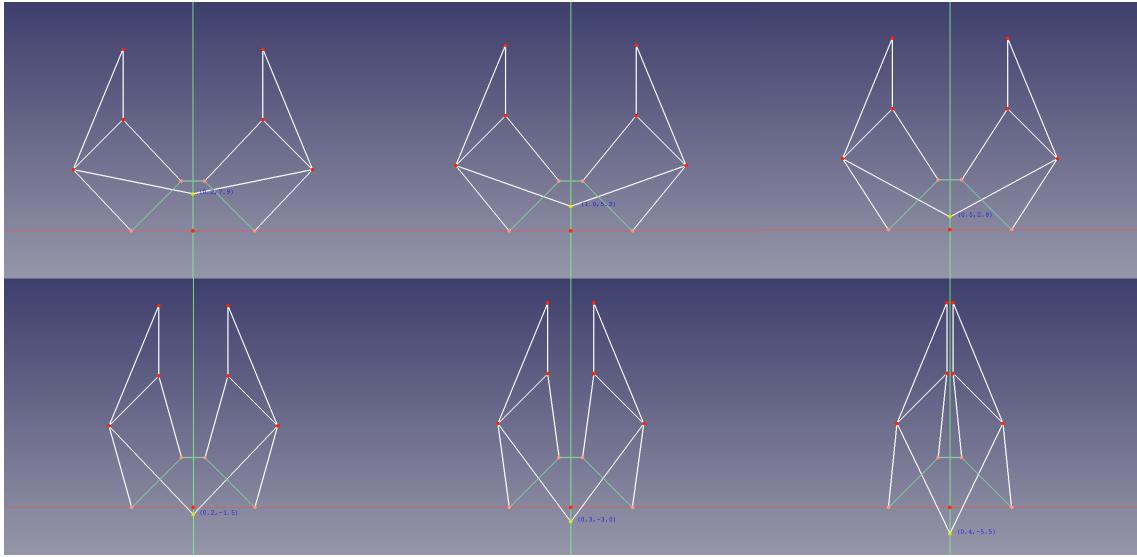


Figura 5.6: Pinza paralela con 1 grado de libertad

5.2.2. Modelo alámbrico del brazo MeArm 2.3(a)

En esta sección se analiza el comportamiento del modelo alámbrico del manipulador en el que se basa este proyecto. Está compuesto por una serie de restricciones que definen la dinámica del brazo.

Parámetros	Valores
L1	170mm
L2	170mm
P1	35mm
P2	35mm
P3	25mm
Ángulo P2	135º

Cuadro 5.1: Parámetros del modelo alámbrico de G-Arm

5.3. Dinámica

5.4. 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.

5.5. Diseño CAD

En esta sección se va a hablar de las particularidades del diseño y diversas cosas que hay que tener en cuenta a la hora de diseñar cualquier pieza mecánica que posteriormente será impresa en plástico. Además se divaga sobre las posibles piezas mecánicas que se pueden usar para lograr la mejor relación resultados-precio. 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 FreeCad?? para cumplir el objetivo de ser totalmente *Open Source* y parametrizado, para que cualquier persona pueda investigarlo, modificarlo y utilizarlo de forma gratuita. El diseño 3D se ha ido contruyendo en a partir de los bocetos y modificando ligeramente en función de como iba quedando.

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 Pro³ y un rollo de 1Kg de filamento PLA Rojo convencional.

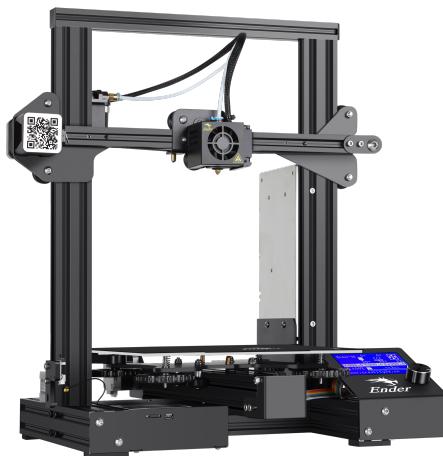


Figura 5.7: Ender-3 Pro V1 2017

³<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)4.6	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 5.2: Componentes hardware necesarios

Componente	Cantidad	Precio total
Tornillo M3 Allen	3	56€

Cuadro 5.3: Tornillería necesaria

El precio total de los componentes necesarios es: 156.3€

Identificador	Cantidad	Relleno óptimo
#1	1	15 %

Cuadro 5.4: Piezas necesarias

Capítulo 6

Desarrollo software 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.

Poner fotito UML

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo (y quizás alguno más) es donde, por fin, describes detalladamente qué has hecho y qué experimentos has llevado a cabo para validar tus desarrollos.

6.1. Control de los actuadores

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

6.1.1. Control de motores paso a paso

hablar de gbl y lo q simplifica esto del controlador hardware utilizado y que realizxa por debajo

Basado en paralelos

6.2. Comunicación ordenador ⇔ robot

En esta sección se exponen los mecanismos necesarios para permitir la comunicación entre el ordenador y el microcontrolador integrado en la base del robot, así como, la arquitectura del sistema robótico. Además se detallan las configuraciones necesarias para adaptar al mundo de la robótica las diferentes partes que lo componen.

6.2.1. Interfaces de comunicación de la placa MKS DLC32??

En base a las especificaciones del manual¹ de la placa, permite establecer comunicaciones vía USB y existe la posibilidad (reside en el propio firmware) de utilizar la funcionalidad Wi-Fi del ESP32. La comunicación obtenida mediante el uso del puerto USB se realiza sobre el protocolo *Universal Asynchronous Receiver/Transmitter* (UART). Este protocolo serie es utilizado para transmitir datos de forma asíncrona entre dispositivos. Es comúnmente usado en la interconexión de microcontroladores, periféricos y otros componentes electrónicos. El UART convierte los datos paralelos en datos serie y viceversa, utilizando bits de inicio, datos, bits de parada y posiblemente bits de paridad para la sincronización y detección de errores. Otra forma de

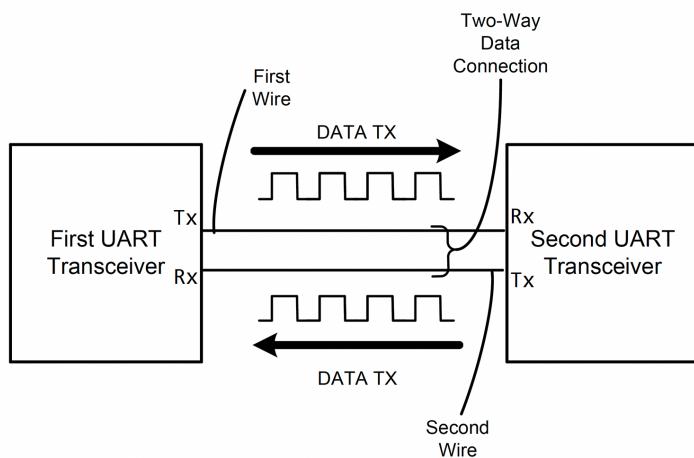


Figura 6.1: Diagramas de tiempo de UART

comunicarnos con la placa es mediante el uso de su adaptador inalámbrico Wi-Fi. Existen firmwares, como es el caso de Grbl, que tienen una versión adaptada para microcontroladores con conectividad Wi-Fi que permiten enviar comandos usando una gran variedad de protocolos convencionales usados en redes de ordenadores. En este caso, Telnet.

Además, el firmware de Grbl es capaz de levantar un punto de acceso inalámbrico al cual puedes conectarte como si de un router se tratara y se puede acceder a un servidor alojado en su dirección IP. Para acceder a él, basta con introducir 192.168.1.1 en la barra superior del navegador web para acceder a una interfaz de usuario que permite controlar los movimientos de la máquina. En este caso tiene un aspecto peculiar y no amigable con el mundo de la robótica ya que está pensado para el uso en CNC. Es por

¹<https://raw.githubusercontent.com/makerbase-mks/MKS-DLC32/main/doc/DLC32%20wiring%20manual.pdf>

esto que es necesario realizar unas ciertas configuraciones para su uso en robótica.

grbl wifi grbl cable biblioteca python usada detalles de la comunicacion via codigo G ejemplos de uso ugs

6.2.2. Adaptación de grbl para su uso en robótica

GRBL tiene ciertas limitaciones a la hora de su utilización en robótica. Es normal, debido a que está pensado para controlar máquinas CNC de 3 ejes prismáticos. Pese a esto, se pueden hacer ciertas modificaciones para adaptarlo a esta aplicación.

Parámetros de Grbl

Para acceder y modificar los parámetros de Grbl se deben realizar los siguientes pasos:

1. Debemos tener conectada al ordenador la placa con el firmware de Grbl instalado.
2. Podemos usar un programa especializado con interfaz gráfica como *Universal Gcode Sender* (UGS) o mediante una terminal serial, como puede ser Cutecom.
3. Introducimos la velocidad de trasmisión de Grbl por defecto; 115200 baudios.
4. Para preguntar al firmware sobre los parámetros que hay configurados, introducimos los caracteres: \$\$
5. Para modificar un parámetro, introducimos un texto del estilo:
\$número=nuevoValor

Para su uso en robótica nos interesa modificar los siguientes parámetros:

- **\$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 vital importancia para evitar que el brazo se desplome tras terminar un cierto movimiento.
- **\$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 también se deben encontrar por medio de la experimentación. Se trata de una medida de seguridad en caso de que el usuario quiera mover demasiado rápido un eje pudiendo dañar el brazo. Se debe encontrar una aceleración idónea para todos los movimientos, una limitación de grbl es que no se puede comandar un movimiento diciéndole una determinada aceleración.

Parámetro	Valor
\$0	0
\$1	255
\$2	Bla
\$0	0
\$0	0
\$0	0
\$0	0

Cuadro 6.1: Parámetros Grbl usados en este trabajo

En esta página oficial de documentación se profundiza más en la finalidad de cada parámetro. Para mas info: <https://github.com/gnea/grbl/blob/master/doc/markdown/commands.md>

Para mas info: <https://github.com/gnea/grbl/blob/master/doc/markdown/settings.md>

6.3. Denavit-Hartenberg

La representación Denavit-Hartenberg consiste en un procedimiento sistemático para describir la estructura cinemática de una cadena articulada constituida por articulaciones con un solo grado de libertad. El procedimiento consiste en los siguientes pasos:

1. **Numerar los eslabones:** Empezando nombrando con el **0** a la base (unida al suelo) y **1** al siguiente eslabón móvil, y así con el resto.
2. **Numerar las articulaciones:** La articulación **1** será el primer grado de libertad, y **n** el último.
3. **Localizar el eje de cada articulación:** En *joints* de revolución, es el eje de rotación y en los prismáticos será el eje por el cual se mueve el eslabón.
4. **Ejes Z:**
5. **Sistema de coordenadas 0:**
6. **Ejes X:**
7. **Ejes Y:**
8. **Sistema del extremo del robot:**

Fuente: <https://www.ciencia-explicada.com/2013/02/parametrizacion-denavit-hartenberg-para.html>

Trasformación	θ	d	a	α
$0 \rightarrow 1$	L1	170mm	a	a

Cuadro 6.2: Tabla resultante de Denavit-Hartenberg

6.4. Integración con ROS 2

En esta sección se detalla el proceso de integración del robot G-Arm en ROS.

6.4.1. Descripción del robot

1. Creamos una carpeta que será nuestro workspace:

```
cd && mkdir workspace
```

2. Creamos una carpeta src dentro de la carpeta anterior:

```
cd workspace && mkdir src
```

3. Dentro de la carpeta src creamos el paquete con:

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

4. Dentro de la carpeta g_arm_description, creamos una serie de carpetas con:

```
mkdir launch rviz urdf meshes
```

5. Podemos borrar la carpeta src que hay dentro de nuestro paquete ya que no es necesaria en la descripción del robot:

```
rm -r src
```

6. Añadimos al fichero *package.xml*, entre

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

7. Compilamos el paquete desde la carpeta workspace:

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

8. Añadimos esto al final del .bashrc:

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

Creación del paquete de descripción

Describir un robot mediante *Unified Robot Description Format* (URDF) y Xacro

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 de robótica, especialmente en ROS.

En un archivo URDF, se especifica la geometría del robot mediante la definición de partes (links) y articulaciones (joints). Cada parte se describe mediante su forma y tamaño, mientras que las articulaciones definen las restricciones de movimiento y las relaciones entre los enlaces. Además de esto, un archivo URDF también puede incluir información sobre la masa y la inercia de los enlaces, así como como texturas y modelos 3D.

El formato URDF se basa en el lenguaje *eXtensible Markup Language* (XML), lo que permite describir el robot de una forma estructurada y legible. Con la descripción en URDF, los robots pueden ser simulados, visualizados y controlados.

Para crear un paquete para describir un robot en ros2 debemos de seguir el siguiente procedimiento:

poner un tree para ver como queda todo hablar de urdf y xacro hablar de rviz y controladores

6.4.2. Nodos

6.5. Integración en MoveIt 2

Con el fin de sacar el máximo partido de nuestro robot, es buena idea implementarlo también en MoveIt. hablar setup assistant y particularidades urdf pa que vaya

6.6. Pruebas

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

6.6.1. 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

6.6.2. 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

6.6.3. 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

6.6.4. Exactitud y repetitividad

Se procede a realiza 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).