



## **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*

# Índice general

---

<b>1. Introducción</b>	<b>1</b>
1.1. Robótica industrial . . . . .	1
1.2. Robótica en educación . . . . .	3
1.2.1. Robótica en colegios e institutos . . . . .	3
1.2.2. Robótica en universidades . . . . .	4
1.3. Robótica de bajo coste . . . . .	5
<b>2. Estado del arte</b>	<b>7</b>
<b>3. Objetivos</b>	<b>12</b>
3.1. Descripción del problema . . . . .	12
3.2. Requisitos . . . . .	13
3.3. Metodología . . . . .	14
3.4. Plan de trabajo . . . . .	14
<b>4. Plataforma de desarrollo</b>	<b>15</b>
4.1. Software . . . . .	15
4.1.1. Python . . . . .	15
4.1.2. Grbl . . . . .	16
4.1.3. Código G . . . . .	16
4.1.4. FreeCAD . . . . .	17
4.1.5. ROS 2 . . . . .	18
4.1.6. MoveIt! . . . . .	18
4.2. Hardware . . . . .	19
4.3. MKS DLC32 . . . . .	20
4.4. Motores Nema 17 . . . . .	21
4.5. Controlador TMC2209 . . . . .	22
4.6. Fuente de alimentación genérica . . . . .	24

<b>5. Desarrollo hardware del manipulador</b>	<b>25</b>
5.1. Concepto . . . . .	25
5.2. Modelo alámbrico . . . . .	26
5.2.1. En qué consiste . . . . .	26
5.2.2. Modelo alámbrico del brazo MeArm 2.3(a) . . . . .	26
5.3. Dinámica . . . . .	26
5.4. DH . . . . .	26
5.5. Bocetos . . . . .	26
5.6. Diseño CAD . . . . .	26
5.7. Diseño CAD . . . . .	26
5.8. Verbatim . . . . .	27
5.9. Ecuaciones . . . . .	27
5.10. Tablas o cuadros . . . . .	28
<b>6. Desarrollo software del manipulador</b>	<b>29</b>
6.1. Control de los actuadores . . . . .	29
6.1.1. Control de motores paso a paso . . . . .	29
6.2. Comunicación ordenador ⇄ robot . . . . .	29
6.2.1. En qué consiste . . . . .	29
6.3. Verbatim . . . . .	29
6.4. Ecuaciones . . . . .	30
6.5. Tablas o cuadros . . . . .	31
<b>7. Conclusiones</b>	<b>33</b>
7.1. Conclusiones . . . . .	33
7.2. Corrector ortográfico . . . . .	34
<b>Bibliografía</b>	<b>35</b>

# Índice de figuras

---

1.1.	Robots en líneas de ensamblaje . . . . .	2
1.2.	Robot Dobot M1 realizando una soldadura con estaño . . . . .	2
1.3.	Robots en el ámbito científico . . . . .	3
1.4.	Robots educativos Lego Mindstorms en escuelas. . . . .	4
1.5.	Robots Kobuki Turtlebot 2 usado en la URJC. . . . .	4
1.6.	Robot IceBot que utiliza una FPGA libre Alhambra II. . . . .	6
2.1.	Robot HydraX . . . . .	8
2.2.	Robot UIArm . . . . .	9
2.3.	Robots formados a partir de paralelogramos . . . . .	10
4.1.	Logo de Grbl . . . . .	16
4.2.	Logo de FreeCAD . . . . .	18
4.3.	Logotipo de ROS 2 Humble . . . . .	19
4.4.	Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot) . . . . .	19
4.5.	Placa base MakerBase DLC32 . . . . .	20
4.6.	Motores Nema 17 de 2.1A . . . . .	21
4.7.	Controlador TMC2209 . . . . .	22
4.8.	Métodos usados para alimentar el robot . . . . .	24
5.1.	Pinza paralela con 1 grado de libertad . . . . .	26
6.1.	Pinza paralela con 1 grado de libertad . . . . .	30

# Listado de códigos

---

5.1. Función para buscar elementos 3D en la imagen . . . . .	27
5.2. Cómo usar un Slider . . . . .	27
6.1. Función para buscar elementos 3D en la imagen . . . . .	30
6.2. Cómo usar un Slider . . . . .	31

# Listado de ecuaciones

---

5.1. Ejemplo de ecuación con fracciones . . . . .	28
5.2. Ejemplo de ecuación con array y letras y símbolos especiales . . . . .	28
6.1. Ejemplo de ecuación con fracciones . . . . .	31
6.2. Ejemplo de ecuación con array y letras y símbolos especiales . . . . .	32

# Índice de cuadros

---

4.1. Especificaciones técnicas de los motores utilizados . . . . .	22
4.2. Especificaciones técnicas del TMC2209 . . . . .	23
5.1. Parámetros intrínsecos de la cámara . . . . .	28
6.1. Parámetros intrínsecos de la cámara . . . . .	31

---

# Capítulo 1

## Introducción

---

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

Winston Churchill

La tendencia de la industria hacia la automatización total ha ido en aumento en los últimos años, haciendo que la demanda de robots industriales se dispare en todo el mundo. Un ejemplo de ello, son las megafactorías en el sector automovilístico. Se tratan de inmensas fábricas con un componente humano mínimo y con una automatización cada día mayor, mediante el uso de la robótica industrial.

No solo se hace uso de grandes robots pesados, sino también de flotas de pequeños pero ágiles robots que desempeñan tareas en cintas transportadoras, ensamblado de placas base, entre otras. Debido al aumento en su uso y del progreso en estas tecnologías, es necesario formar cada vez más ingenieros que ejercerán en este campo. Es por esto que se necesitan herramientas adaptadas a las nuevas tecnologías y fácilmente accesibles para estudiantes y centros.

Por otro lado, la robótica educativa ha nacido como una herramienta innovadora y poderosa en el ámbito de la enseñanza. Mediante la integración de robots en las aulas, los estudiantes pueden experimentar el aprendizaje de una manera más interactiva y participativa. Desde los centros de educación primaria hasta las universidades, la incorporación de la robótica educativa ha experimentado un crecimiento significativo. Esta tecnología se adapta de manera excepcional a las necesidades y capacidades de cada etapa educativa.

### 1.1. Robótica industrial

La robótica industrial es la rama de la ingeniería dedicada al diseño, construcción, operación y mantenimiento de robots utilizados en la automatización de procesos

industriales. Estos robots pueden realizar una amplia variedad de tareas, desde la manipulación y transporte de materiales, hasta el ensamblaje y soldadura de piezas. Entre las aplicaciones más importantes encontramos las siguientes:

- *Automatización de líneas de ensamblaje.* Esta tecnología se encuentra ampliamente asentada en las líneas de producción del sector automotriz. Un ejemplo son las factorías de Tesla.



Figura 1.1: Robots en líneas de ensamblaje

- *Soldadura.* Es utilizada para realizar la unión de estructuras mecánicas debido a su alta precisión y capacidad de realizar la misma soldadura perfecta una y otra vez. Además, también son usados para soldar placas de circuitos.



Figura 1.2: Robot Dobot M1 realizando una soldadura con estaño

- *Investigación y desarrollo en laboratorios.* Los brazos robóticos realizan tareas de laboratorio repetitivas y precisas, lo que puede acelerar el proceso de investigación y desarrollo de nuevos productos médicos. Uno de los usos más frecuentes es preparar y procesar muestras en investigaciones científicas. En concreto, pueden desempeñar tareas como la extracción de *Ácido Desoxirribonucleico* (ADN), la separación de componentes, la adición de reactivos y el análisis de muestras químicas, entre otros. Esto ayuda a reducir los errores humanos y garantizar la integridad y reproducibilidad de los experimentos llevados a cabo.



Figura 1.3: Robots en el ámbito científico

## 1.2. Robótica en educación

La robótica en la educación 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 colegios e institutos

En las escuelas, 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). 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.



Figura 1.4: Robots educativos Lego Mindstorms en escuelas.

### 1.2.2. Robótica en universidades

En el 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. Además se hace uso de más tipos de robots, como pueden ser, industriales o plataformas robóticas móviles reales. Al ser sistemas usados en el mundo profesional, los estudiantes pueden aprender con el robot que usarán en un futuro. Aunque es verdad que las universidades disponen de algunas unidades, no siempre son accesibles para el estudiante por diversas razones.



Figura 1.5: Robots Kobuki Turtlebot 2 usado en la URJC.

En resumen, la robótica en la educación es una herramienta poderosa para fomentar el aprendizaje y la innovación en las nuevas generaciones. Desde la escuela hasta la universidad, la robótica se ha convertido en una disciplina clave para formar a los futuros líderes tecnológicos del mundo.

### 1.3. Robótica de bajo coste

La robótica de bajo coste es un área de la robótica enfocada en el diseño y desarrollo de robots accesibles y asequibles. El objetivo principal de esta, es 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 costo, como la impresión 3D y el corte láser, han hecho posible que los usuarios puedan crear piezas y componentes robóticos personalizados a un precio más bajo que el de la fabricación tradicional.

Este campo de la robótica se beneficia enormemente de 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 costo y materiales comunes.

En resumen, la robótica de bajo costo es una consecuencia directa del crecimiento de la capacidad de computación, el abaratamiento de los precios de los componentes electrónicos y del enfoque DIY. Esto ha permitido la creación de robots avanzados y accesibles para el público general. El desarrollo de esta tecnología, permite que más personas puedan experimentar en este área de la ingeniería y desarrollar sus propias soluciones robóticas.

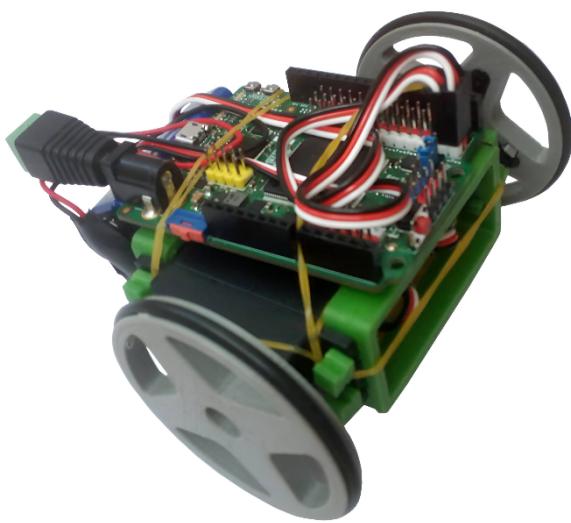


Figura 1.6: Robot IceBot que utiliza una FPGA libre Alhambra II.

---

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

---

<sup>1</sup><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.

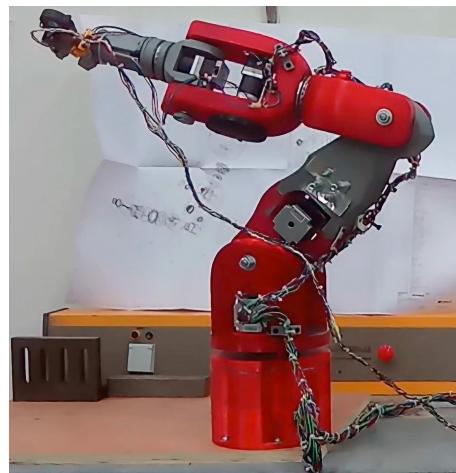


Figura 2.1: Robot HydraX

- 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.

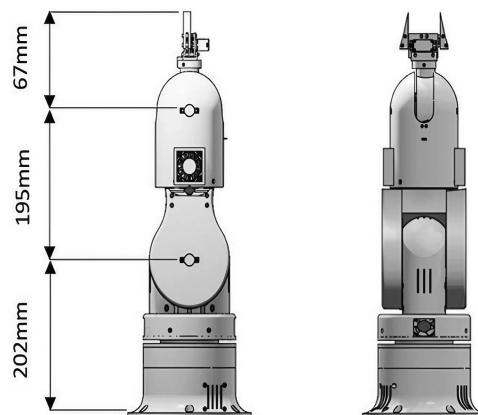
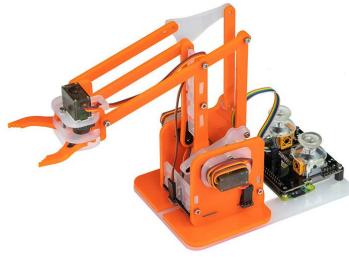


Figura 2.2: Robot UIArm

- 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 Thingiverse<sup>2</sup>. Además, existen robots con diferentes nombres basados en el mismo concepto, como puede ser EEZYBotArm, una versión más robusta y atractiva que el original.

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



(a) MeArm



(b) EEZYBotArm MK1

Figura 2.3: Robots formados a partir de paralelogramos

<sup>2</sup><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 montaje<sup>3</sup> que circulan por internet.
- Al usar servomotores, se puede establecer el ángulo de cada articulación fácilmente sin necesitar de ningún tipo de sensor externo.
- 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.

---

<sup>3</sup><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 GitHub<sup>1</sup>, en el cual se ha ido subiendo el código y diseños generados a lo largo del proyecto. Adicionalmente, en este mismo repositorio se ha incluido una Wiki<sup>2</sup> con las explicaciones detalladas de todas las actividades llevadas a cabo durante estos meses de trabajo. De esta manera, se ha creado un registro completo y accesible de todo el proceso de desarrollo del sistema.

### 3.4. Plan de trabajo

El desarrollo del TFG ha estado dividido en dos etapas. La primera, comenzó en octubre y fue abandonada en enero. //TODO

---

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

<sup>2</sup><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

Grbl<sup>1</sup> 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

---

<sup>1</sup><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

FreeCAD<sup>2</sup> 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.

---

<sup>2</sup>[https://www.freecad.org/index.php?lang=es\\_ES](https://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

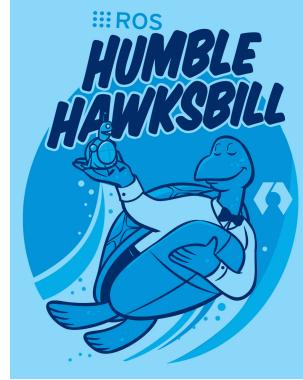


Figura 4.3: Logotipo de ROS 2 Humble

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.

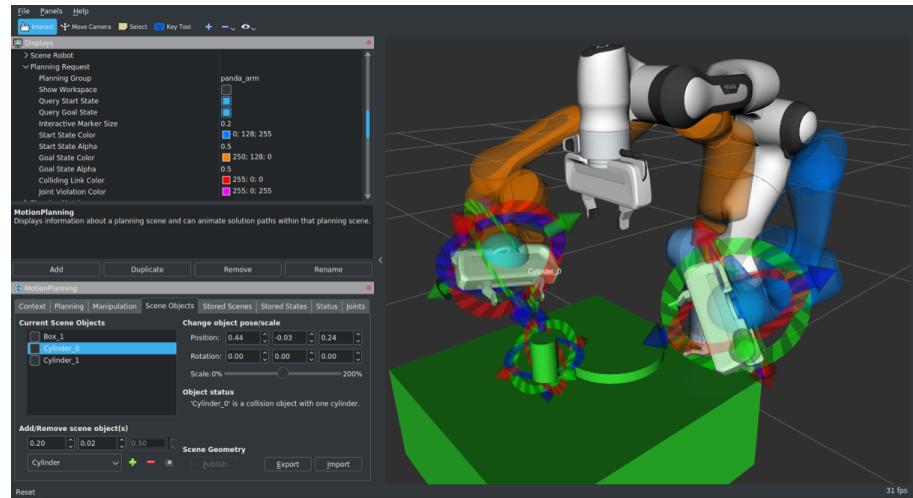


Figura 4.4: Aspecto del plugin de MoveIt para Rviz2 (Franka Emika Robot)

## 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 Github<sup>3</sup>. 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.

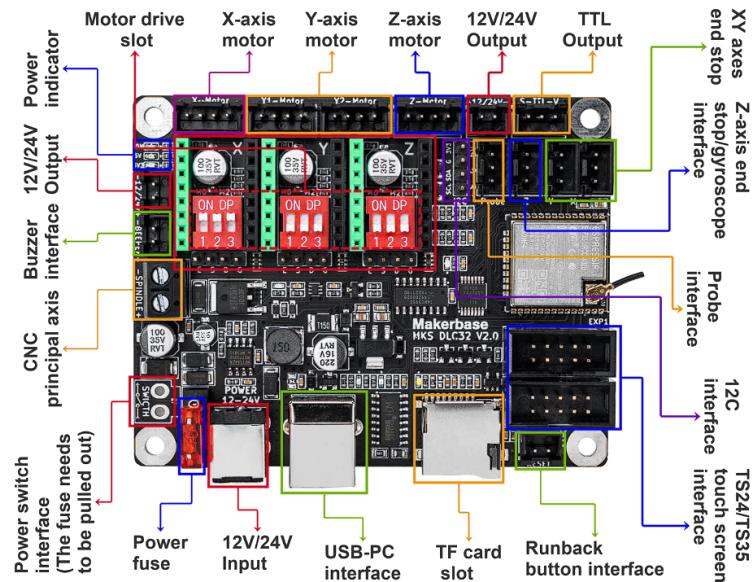


Figura 4.5: Placa base MakerBase DLC32

<sup>3</sup><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.

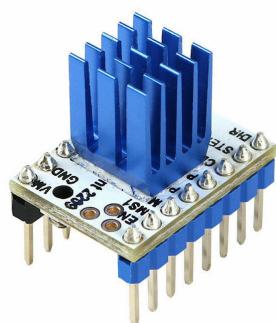


Figura 4.7: Controlador TMC2209

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

Con 4 grados de libertad vamos mal pero se puede. Tipos: Scara

RR

Basado en paralelos

## 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.

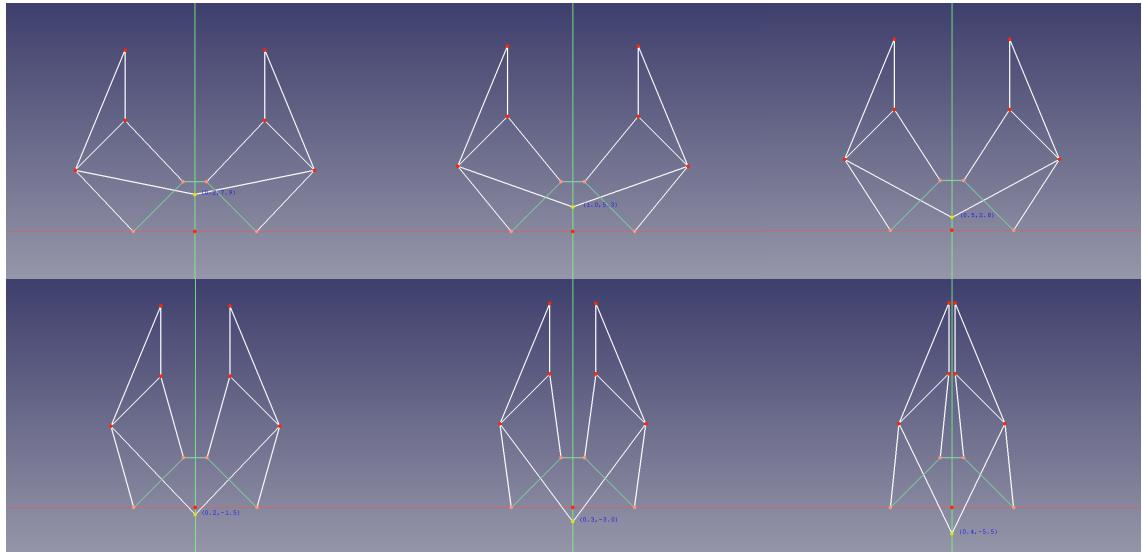


Figura 5.1: Pinza paralela con 1 grado de libertad

### 5.2.2. Modelo alámbrico del brazo MeArm 2.3(a)

## 5.3. Dinámica

## 5.4. DH

## 5.5. Bocetos

## 5.6. Diseño CAD

## 5.7. Diseño CAD

En el Código 6.2 vemos un ejemplo escrito en Python.

---

```
void Memory::hypothesizeParallelograms () {
    for(it1 = this->controller->segmentMemory.begin(); it1++) {
        squareFound = false; it2 = it1; it2++;
        while ((it2 != this->controller->segmentMemory.end()) && (!squareFound))
        {
            if (geometry::haveACommonVertex((*it1), (*it2), &square)) {
                dist1 = geometry::distanceBetweenPoints3D ((*it1).start, (*it1).end);
                dist2 = geometry::distanceBetweenPoints3D ((*it2).start, (*it2).end);
            }
        // [...]
    }
}
```

---

Código 5.1: Función para buscar elementos 3D en la imagen

---

```
def mostrarValores():
    print (w1.get(), w2.get())

master = Tk()
w1 = Scale(master, from_=0, to=42)
w1.pack()
w2 = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w2.pack()
Button(master, text='Show', command=mostrarValores).pack()

mainloop()
```

---

Código 5.2: Cómo usar un Slider

## 5.8. Verbatim

Para mencionar identificadores usados en el código —como nombres de funciones o variables— en el texto, usa el entorno literal o verbatim `hypothesizeParallelograms()`. También se puede usar este entorno para varias líneas, como se ve a continuación:

```
void Memory::hypothesizeParallelograms () {
    // add your code here
}
```

## 5.9. Ecuaciones

Si necesitas insertar alguna ecuación, puedes hacerlo. Al igual que las figuras, no te olvides de referenciarlas. A continuación se exponen algunas ecuaciones de ejemplo: Ecuación 6.1 y Ecuación 6.2.

$$H = 1 - \frac{\sum_{i=0}^N \frac{(\frac{d_{js}+d_{je}}{2})}{N}}{M} \quad (5.1)$$

Ecuación 5.1: Ejemplo de ecuación con fracciones

$$v(\text{entrada}) = \begin{cases} 0 & \text{if } \epsilon_t < 0,1 \\ K_p \cdot (T_t - T) & \text{if } 0,1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if } M_t < \epsilon_t \end{cases} \quad (5.2)$$

Ecuación 5.2: Ejemplo de ecuación con array y letras y símbolos especiales

## 5.10. Tablas o cuadros

Si necesitas insertar una tabla, hazlo dignamente usando las propias tablas de LATEX, no usando pantallazos e insertándolas como figuras... En el Cuadro 6.1 vemos un ejemplo.

Parámetros	Valores
Tipo de sensor	Sony IMX219PQ[7] CMOS 8-Mpx
Tamaño del sensor	3.674 x 2.760 mm (1/4"format)
Número de pixels	3280 x 2464 (active pixels)
Tamaño de pixel	1.12 x 1.12 um
Lente	f=3.04 mm, f/2.0
Ángulo de visión	62.2 x 48.8 degrees
Lente SLR equivalente	29 mm

Cuadro 5.1: Parámetros intrínsecos de la cámara

---

# 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.

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

### 6.2.1. En qué consiste

En el Código 6.2 vemos un ejemplo escrito en Python.

## 6.3. Verbatim

Para mencionar identificadores usados en el código —como nombres de funciones o variables— en el texto, usa el entorno literal o verbatim

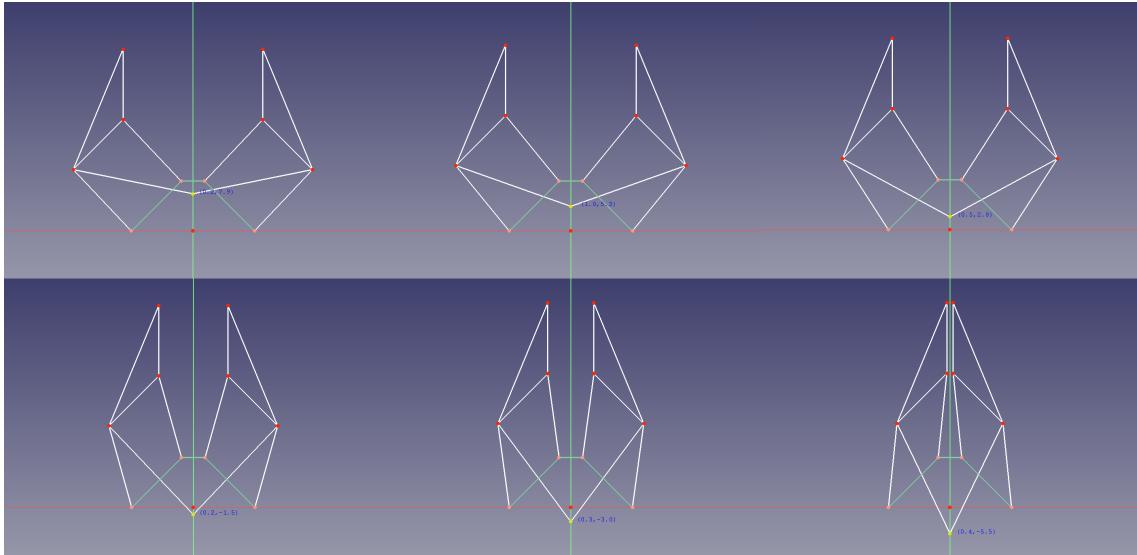


Figura 6.1: Pinza paralela con 1 grado de libertad

---

```
void Memory::hypothesizeParallelograms () {
    for(it1 = this->controller->segmentMemory.begin(); it1++) {
        squareFound = false; it2 = it1; it2++;
        while ((it2 != this->controller->segmentMemory.end()) && (!squareFound))
        {
            if (geometry::haveACommonVertex((*it1), (*it2), &square)) {
                dist1 = geometry::distanceBetweenPoints3D ((*it1).start, (*it1).end);
                dist2 = geometry::distanceBetweenPoints3D ((*it2).start, (*it2).end);
            }
        // [...]
    }
}
```

---

Código 6.1: Función para buscar elementos 3D en la imagen

`hypothesizeParallelograms()`. También se puede usar este entorno para varias líneas, como se ve a continuación:

```
void Memory::hypothesizeParallelograms () {
    // add your code here
}
```

## 6.4. Ecuaciones

Si necesitas insertar alguna ecuación, puedes hacerlo. Al igual que las figuras, no te olvides de referenciarlas. A continuación se exponen algunas ecuaciones de ejemplo: Ecuación 6.1 y Ecuación 6.2.

---

```

def mostrarValores():
    print (w1.get(), w2.get())

master = Tk()
w1 = Scale(master, from_=0, to=42)
w1.pack()
w2 = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w2.pack()
Button(master, text='Show', command=mostrarValores).pack()

mainloop()

```

---

Código 6.2: Cómo usar un Slider

$$H = 1 - \frac{\sum_{i=0}^N \frac{(\frac{d_{js}+d_{je}}{2})}{N}}{M} \quad (6.1)$$

Ecuación 6.1: Ejemplo de ecuación con fracciones

## 6.5. Tablas o cuadros

Si necesitas insertar una tabla, hazlo dignamente usando las propias tablas de L<sup>A</sup>T<sub>E</sub>X, no usando pantallazos e insertándolas como figuras... En el Cuadro 6.1 vemos un ejemplo.

Parámetros	Valores
Tipo de sensor	Sony IMX219PQ[7] CMOS 8-Mpx
Tamaño del sensor	3.674 x 2.760 mm (1/4"format)
Número de pixels	3280 x 2464 (active pixels)
Tamaño de pixel	1.12 x 1.12 um
Lente	f=3.04 mm, f/2.0
Ángulo de visión	62.2 x 48.8 degrees
Lente SLR equivalente	29 mm

Cuadro 6.1: Parámetros intrínsecos de la cámara

$$v(\text{entrada}) = \begin{cases} 0 & \text{if } \epsilon_t < 0,1 \\ K_p \cdot (T_t - T) & \text{if } 0,1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if } M_t < \epsilon_t \end{cases} \quad (6.2)$$

Ecuación 6.2: Ejemplo de ecuación con array y letras y símbolos especiales

---

# **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<sup>A</sup>T<sub>E</sub>Xa 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).