

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



**Diseño, implementación, simulación y control de un brazo
robótico para el Rover UVG**

Trabajo de graduación presentado por José Rodrigo Díaz Díaz para
optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



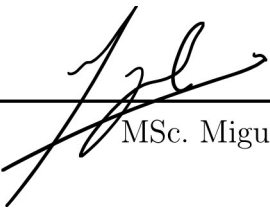
**Diseño, implementación, simulación y control de un brazo
robótico para el Rover UVG**

Trabajo de graduación presentado por José Rodrigo Díaz Díaz para
optar al grado académico de Licenciado en Ingeniería Mecatrónica


Guatemala,

2023

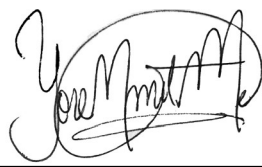
Vo.B0.:

(f) 
MSc. Miguel Zea

Tribunal Examinador:

(f) 
MSc. Miguel Zea

(f) 
Ing. Kurt Kellner

(f) 
Ing. Yosemite Meléndez

Fecha de aprobación: Guatemala, 4 de enero de 2023.

Quiero darle gracias en primer lugar a Dios por ayudarme a terminar este proyecto. También quiero agradecer a mis papás, quienes siempre me apoyaron y me asistieron durante este proceso. De la misma manera, quiero agradecer a mi asesor Msc. Miguel Zea, quien proporcionó apoyo durante el semestre para lograr ejecutar este trabajo de graduación.

También quiero agradecer a mis amigos y compañeros de la carrera. De manera especial a Darío, Diego, Emilio, Marco, Mario y Helder; quienes fueron de gran apoyo con sus sugerencias, observaciones y asistencia.

Espero que este trabajo sea un aporte valioso y pueda ser de utilidad para los siguientes estudiantes trabajando en el Rover UVG.

Prefacio	v
Lista de figuras	x
Lista de cuadros	xi
Resumen	xiii
Abstract	xv
1. Introducción	1
2. Antecedentes	3
3. Justificación	9
4. Objetivos	11
5. Alcance	13
6. Marco teórico	15
6.1. Proceso de diseño mecánico	15
6.1.1. Identificar la necesidad	15
6.1.2. Investigación preliminar	15
6.1.3. Planteamiento de objetivos	16
6.1.4. Especificaciones de desempeño	16
6.1.5. Generación de ideas e invención	16
6.1.6. Análisis	16
6.1.7. Selección	16
6.1.8. Diseño detallado	17
6.1.9. Creación de prototipos y pruebas	17
6.1.10. Producción	17
6.2. Robótica	17
6.2.1. Robots manipuladores	18

6.2.2. Cinemática en manipuladores seriales	19
6.2.3. Representación de ángulos	22
6.2.4. Transformación homogénea	23
6.2.5. Cinemática directa	24
6.2.6. Cinemática diferencial	26
6.2.7. Cinemática inversa	28
6.3. Mecanismos	29
6.4. Motores y controladores Lynxmotion	29
6.4.1. Lynxmotion ST1	29
6.4.2. Lynxmotion Smart Servo Adapter Board	30
6.5. ROS	30
6.6. Simulación	32
6.6.1. Modelos URDF	32
6.6.2. RVIZ	32
7. Diseño electromecánico del brazo	35
7.1. Primer grado de libertad	37
7.2. Segundo y tercer grados de libertad	38
7.3. Cuarto grado de libertad	40
7.4. Quinto grado de libertad	41
7.5. Efector final	42
7.6. Base del robot	43
7.7. Movimiento de los motores	43
8. Análisis cinemático del robot	47
8.1. Cinemática directa	47
8.2. Cinemática diferencial	48
8.3. Cinemática inversa	49
8.4. Límites en las juntas	50
9. Simulación y control en ROS	53
10. Resultados de pruebas	61
10.1. Diseño mecánico	61
10.2. Ganancia de torque	62
10.3. Repetibilidad	63
11. Conclusiones	67
12. Recomendaciones	69
13. Bibliografía	71
14. Anexos	73
15. Glosario	75

Lista de figuras

1. Modelo físico entregado, [2].	4
2. Sistema para localización y auto piloto, [1].	4
3. Sistema mecánico humana, [3].	5
4. Modelo físico de RESCUER, [5].	5
5. Modelo simulación en <i>MoveIt!</i> de RESCUER, [5].	6
6. Modelo simulación de UR5, [6].	6
7. Modelo simulación de IRB120 en <i>MoveIt!</i> , [7].	7
8. Modelo simulación de IRB210 en <i>RobotStudio</i> , [7].	7
9. EEZYbotARM mk2, [8].	7
10. Ejemplo matriz de decisión, [9].	17
11. Diagrama simplificado diseño de ingeniería, [10].	18
12. Componentes básicos del sistema robótico, [11].	18
13. Manipulador serial, [11].	19
14. Espacio de trabajo de robot SCARA, [11].	20
15. Algoritmo cinemática inversa, [11].	28
16. Ejemplo de nodos de ROS, [17].	31
17. Estructura de árbol URDF, [12].	33
18. Modelo URDF, [12].	33
19. Primer grado de libertad, vista lateral.	37
20. Primer grado de libertad, vista inferior.	38
21. Segundo y tercer grados de libertad, vista isométrica.	39
22. Segundo y tercer grados de libertad, vista lateral.	40
23. Cuarto grado de libertad, vista isométrica.	41
24. Cuarto grado de libertad, vista lateral.	42
25. Quinto grado de libertad.	42
26. Efecto final, vista isométrica.	43
27. Efecto final, vista inferior.	44
28. Base del robot.	45
29. Aplicación Lynxmotion.	45
30. Código para leer datos y mandar el dato a la función de la librería Lynxmotion	45

31. Cinemática directa con Matlab	48
32. Jacobiano del vector de configuración q_1	49
33. Histórico de las iteraciones del vector de configuración para llegar a la posición p_1 .	49
34. Límites de la junta 1	50
35. Límites de la junta 5 y el gripper	50
36. Límites de las juntas 2, 3 y 4	51
37. Configuración para inicializar simulación en launch.	54
38. Argumentos de inicialización en launch.	54
39. Inicialización nodos en launch.	55
40. Inicialización acciones en launch.	55
41. Ejemplo de junta y eslabón del URDF.	56
42. Cajas de colisiones en RVIZ.	57
43. Visualización del modelo URDF en RVIZ.	58
44. Nodos presentes en la simulación.	58
45. Relaciones padre-hijo entre las juntas.	59
46. Almacenamiento de valores de juntas en nodo suscriptor.	59
47. Posible ruptura de las cajas.	62
48. Ganancia de torque junta 3, en kg-cm.	63
49. Ganancia de torque junta 4, en kg-cm.	63
50. Gráfica de ganancia de torque junta 3, en kg-cm en función del largo del eslabón.	64
51. Gráfica de ganancia de torque junta 4, en kg-cm en función del largo del eslabón.	64
52. Ejemplo de medición con Tracker para obtener los ángulos reales.	65
53. Valores para los ángulos medidos y los reales para la junta 1.	65
54. Promedio de la diferencia de ángulos y desviación estándar para la junta 1.	65
55. Valores para los ángulos medidos y los reales para las juntas 2 y 3.	65
56. Promedio de la diferencia de ángulos y desviación estándar para las juntas 2 y 3.	65
57. Valores para los ángulos medidos y los reales para la junta 4.	66
58. Promedio de la diferencia de ángulos y desviación estándar para la junta 4.	66
59. Valores para los ángulos medidos y los reales para la junta 5.	66
60. Promedio de la diferencia de ángulos y desviación estándar para la junta 5.	66
61. Valores para los ángulos medidos y los reales para el gripper.	66
62. Promedio de la diferencia de ángulos y desviación estándar para el gripper.	66
63. Propuesta entregada del brazo.	73
64. Brazo montado sobre el Rover UVG.	74

Lista de cuadros

1. Matriz Denavit Hartenberg, medidas en milímetros.	48
--	----

Este proyecto consiste en el diseño mecánico e implementación de un brazo robótico de 5 grados de libertad para colocar en el Rover UVG. Se realizó una investigación para conocer el tipo de juntas utilizadas y cómo transmitir la potencia de los motores así como los principios básicos de ROS. Se optó por un diseño sencillo, para que sea una base en la cual se pueda realizar trabajos posteriores. Es por esto que se utilizaron los mecanismos de 4 barras como sistema de transmisión de potencia y métodos de manufactura rápida como impresión 3D y corte láser para fabricar las piezas. Para aumentar el torque de las primeras juntas, se utilizaron cajas armónicas. También se realizó el diseño de un cojinete axial impreso en 3D con balines para realizar la rotación principal. Se buscó versatilidad en el efector final, por lo que este se puede intercambiar.

En cuanto al diseño de ROS, se logró implementar un paquete que funcione en el espacio de trabajo del Rover capaz de simular las juntas del robot utilizando su modelo URDF y RVIZ. Para poder mandar los datos de forma serial a los motores, se realizó un nodo subscriptor de la topic joint state para poder leer la información de las juntas y mandarlas con formato json.

Se logró reducir el torque producido por el peso de los motores por 0.73 kg-cm al utilizar ambos mecanismos de cuatro barras. En las pruebas de repetibilidad, se encontró un error promedio menor a 1.5 grados en las juntas y una desviación estándar menor a 3.5 grados. El alcance máximo del brazo es de 40 cm.

This project consists of the mechanical design and implementation of a robotic arm with 5 degrees of freedom to be placed in the Rover UVG. An investigation was carried out to know the type of joints typically used and how to transmit the power of the motors as well as the basic principles of ROS. A simple design was chosen, so that it is a base on which subsequent work can be carried out. This is why 4-bar mechanisms were used as the power transmission system and rapid manufacturing techniques like 3D printing and laser cutting were used. To increase the torque of the first joints, harmonic boxes were used. The design of an axial bearing printed with pellets to carry out the main rotation was designed to obtain a smooth rotation. Versatility was sought in the end effector, so it can be interchanged easily.

Regarding the ROS design, it was possible to implement a package that works in the Rover workspace capable of simulating the joints of the robot using its URDF and RVIZ model. In order to send the data serially to the motors, a subscriber node of the topic joint state was created to be able to read the information from the joints and send them in json format.

It was possible to reduce the torque produced by the weight of the motors by 0.73 kg-cm when using both four-bar mechanisms. In repeatability tests, a maximum average error of 1.5 degrees was found in the joints and the standard deviation was less than 3.5 degrees. The maximum reach of the arm is 40 cm.

CAPÍTULO 1

Introducción

Este proyecto forma parte de la nueva iteración del Rover UVG. En esta nueva etapa se busca incrementar la autonomía del Rover y es por esta razón que se busca implementar un brazo robótico. Los manipuladores seriales son utilizados tanto a nivel industria como a nivel académico y de consumidor final para ayudar al usuario a realizar tareas sencillas de forma repetitiva y con bajo error.

Este proyecto no tenía antecedentes dado que nunca se había contado con un brazo robótico para el Rover. Es por eso que se comenzó con la etapa de diseño e investigación. Se buscaron los mejores mecanismos y sistemas de transmisión de potencia para lograr mover las juntas sujeto a las restricciones de los servomotores empleados. Dado que la universidad cuenta con máquinas de impresión 3D y corte láser, fue con estas herramientas que se construyó este proyecto. Para poder unir estas piezas, se utilizaron piezas estándar como tornillos y cojinetes.

El Rover UVG es un proyecto que se lleva trabajando por varios años en la UVG, con aplicaciones desde recoger basura en la playa hasta detectar obstáculos y ser controlado de forma remota. Su diseño mecánico ha sido modificado a lo largo de las iteraciones para ajustarse a cada aplicación. Sin embargo, nunca ha tenido un brazo robótico, lo que permitiría interactuar con su entorno. El brazo robótico será controlado a través de ROS y su diseño mecánico será realizado con técnicas de manufactura rápida.

2.1 Robot explorador modular UVG

En [1], Archila trabajó en configurar un autopiloto y agregar sensores de inclinación, temperatura, altura y posición así como poder transmitir video a través de internet o radio frecuencia. El auto piloto utilizado fue Pixhawk. Esto permitió controlar las entradas, salidas y sensores configurado con el programa Mission Planner. Se utilizó una Raspberry Pi 3 con sistema operativo APSync para el manejo y envío de datos a través de internet. Para que el rover se pudiera movilizar, se utilizaron dos motores DC con caja reductora y controlados por un driver que funciona con PWM.

En [2], Sagastume se encargó de rediseñar el sistema mecánico de la última versión del rover. Se rediseñó el chasis del rover utilizando *Autodesk Inventor*, para poder proteger sus componentes esenciales. Se decidió conservar el sistema de orugas para poder manejarse en diversos terrenos. Se utilizó *Altium Designer* para poder fabricar las PCB necesarias y así facilitar la conexión de los sensores. Al finalizar, se logró reducir el tamaño del rover, brindar mayor protección y crear espacios para implementar más módulos en el futuro.

2.2 Diseño mecánico de brazo Humana UVG

En [3], Calderón se encargó del diseño mecánico de un brazo robótico para la automatización de un sistema VarioGuide-Brainlab para realizar cirugías estereotácticas. Se utilizaron



Figura 1: Modelo físico entregado, [2].



Figura 2: Sistema para localización y auto piloto, [1].

motores stepper NEMA 11 y se rediseñó el sistema de transmisión de potencia. Se diseñaron reducciones por medio de engranajes planetarios para los primeros grados de libertad para poder alcanzar un mayor torque. Para el primer grado de libertad se utilizó un sistema de tornillo sinfín-corona para lograr un autobloqueo y para el segundo se utilizó un sistema de husillo de bola para convertir movimiento rotacional en movimiento lineal. Para cada grado de libertad, se tuvo que analizar la fuerza máxima que este podía ejercer y así se fue modificando el diseño o los motores utilizados.

2.3 Aplicación de herramientas de aprendizaje reforzado y aprendizaje profundo en simulaciones de robótica de enjambre con restricciones físicas

En [4], Izeppi trabajó en establecer las bases para poder desarrollar proyectos más complicados dentro del entorno de robótica. Aunque su objetivo principal fue aplicar un algoritmo que utiliza métodos de aprendizaje automático para poder evadir obstáculos de forma aleatoria, se evalúa la viabilidad de realizar simulaciones de sistemas multi-robot en ROS. Se



Figura 3: Sistema mecánico humana, [3].

utilizaron nodos de ROS para realizar la comunicación entre los diferentes componentes. Al terminar, se pudo enviar instrucciones al robot y recibir información de sus sensores por medio de los nodos de comunicación.

2.4 Extendiendo las capacidades del robot RESCUER en ROS

En, [5], León Pérez trabajó en el brazo robótico para el rover RESCUER de la Universitat Jaume I debido a que antes no tenía ningún manipulador y así poder realizar tareas de agarre. Su proyecto consistió en realizar tareas de planificación automática para mover el brazo robótico, integrar el robot en un nodo de ROS para realizar tareas de forma autónoma y crear simulaciones para probar los movimientos. Se creó el modelo URDF para poder utilizar *MoveIt!* y poder trazar con diferentes poses del robot. La simulación se realizó en Gazebo y allí se diseñaron controladores PID. Dado que el brazo ya estaba construido a la hora de realizar este trabajo, no se especifican los servo motores utilizados, pero las dimensiones del robot se pueden extraer del modelo URDF.

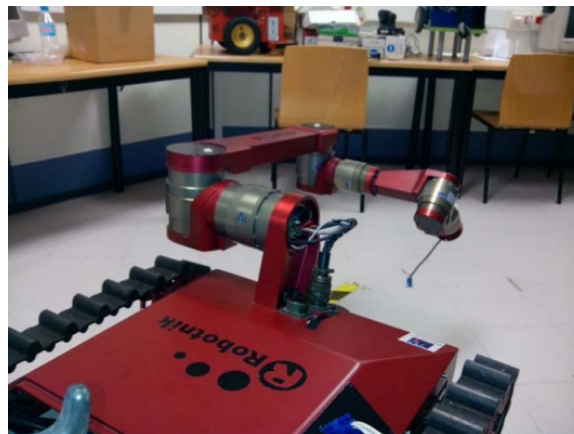


Figura 4: Modelo físico de RESCUER, [5].

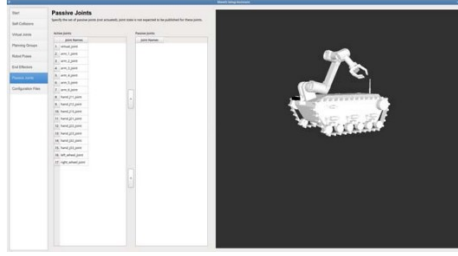


Figura 5: Modelo simulación en *MoveIt!* de RESCUER, [5].

2.5 Operación de remachado mediante robot manipulador industrial basado en ROS

En [6], Gómez Tejada trabajó en resolver la tarea de realizar una tarea de remachado a una pieza de trabajo mediante un robot manipulador industrial. Se programó el robot mediante ROS para realizar la tarea, evaluando su rapidez y eficiencia. Se utilizó el manipulador UR5, simulado en *Gazebo* y *RVIZ* para visualizar los datos de los sensores e información del sistema. Creando el modelo URDF para la simulación, se logró mover el robot a través de las herramientas de ROS, identificar y localizar el agujero a remachar y lograr llevar al robot a su posición final.

2.6 Integración del brazo robot IRB120 en entorno ROS-MATLAB

En [7], Gómez Cuadrado trabajó en un sistema de control de trayectoria para el brazo ABB IRB120 a través del entorno de Matlab. El proyecto fue capaz de recibir el estado en el que se encuentra el robot y mandar distintas posiciones al brazo a través de MATLAB para que las alcance. Se utilizó *MoveIt!* para calcular las trayectorias del robot y se utilizó *RobotStudio* para simular los resultados.

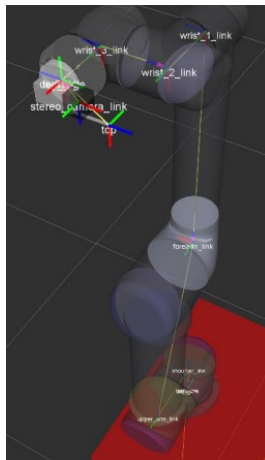


Figura 6: Modelo simulación de UR5, [6].

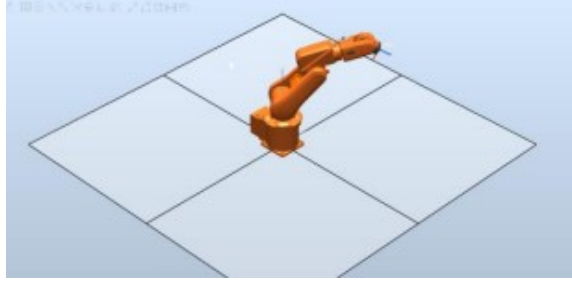


Figura 7: Modelo simulación de IRB120 en *MoveIt!*, [7].

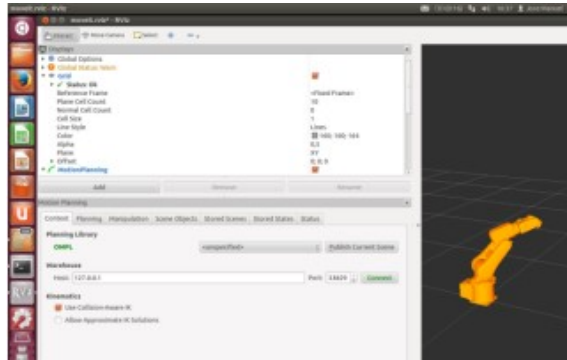


Figura 8: Modelo simulación de IRB210 en *RobotStudio*, [7].

2.7 EEZYbotARM MK2

En [8], Franciscone realiza la segunda iteración del EEZYbotARM. Este, al igual que su predecesor, busca armar un brazo robótico simple utilizando solamente piezas impresas y tornillos. Su objetivo es presentar una opción educativa para que los estudiantes puedan tener sus primeras experiencias con brazos robóticos. Utiliza un sistema cinemático a escala del robot industrial ABB IRB460.

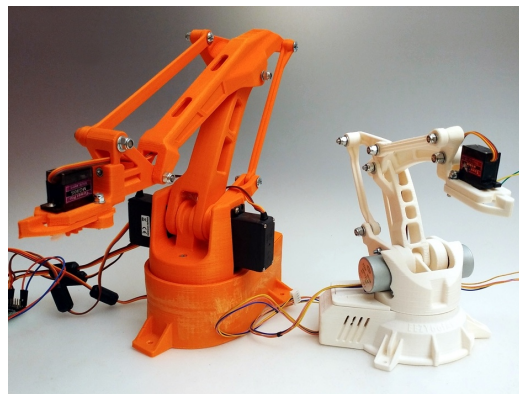


Figura 9: EEZYbotARM mk2, [8]

El proyecto del Rover ya es capaz de movilizarse y ser controlado de forma remota, pero esto todavía está bastante lejos de lograr la autonomía. Para llegar más allá de la locomoción, se propone diseñar e implementarle un brazo robótico. Se optó por utilizar un manipulador serial porque estos son capaces de realizar tareas repetitivas con bajo error. Con este brazo robótico, el Rover UVG podrá interactuar con su entorno y así incrementar su autonomía, capaz de realizar tareas de recolección, limpieza, reparaciones, exploración u otro tipo de asistencia.

Dado que la versión anterior del Rover no cuenta con ningún manipulador, se realizará un modelo impreso en 3D, corte láser u otro método de manufactura rápida con 5 grados de libertad. Este funcionará como un nodo de ROS para ser incorporado con el resto del Rover y así lograr llegar a sus posiciones deseadas. Se escogió utilizar ROS porque es una herramienta poderosa utilizada en la industria, siendo un estándar para la simulación y control de robots. Para lograr esto, primero se hará el modelo URDF para poder simular el brazo por software antes de implementarlo físicamente.

4.1 Objetivo general

Diseñar un manipulador robótico para el Rover UVG de 5 GDL controlado por medio de ROS.

4.2 Objetivos específicos

- Simular y controlar al manipulador diseñado, empleando ROS.
- Utilizar los servo motores inteligentes Lynxmotion para lograr los movimientos de las juntas rotacionales.
- Diseñar e implementar el sistema mecánico del brazo robótico con técnicas de manufactura rápida.

El enfoque de este proyecto no es implementar rutinas complejas de control, sino desarrollar una plataforma compatible con ROS2 Foxy que permita realizar esto en trabajos posteriores. Este proyecto se puede seguir iterando y mejorando ya sea en conjunto con el Rover o de manera individual.

En cuanto al diseño del robot, se tienen limitantes de materiales y de su fabricación. Está compuesto por partes impresas en 3D, corte láser y tornillos métricos estándar. Esto se realizó con el fin de tener reproductibilidad del diseño y poder realizar modificaciones más rápido. Sin embargo, la disponibilidad de impresoras y espera de la fabricación de piezas son factores que afectan el tiempo entre una iteración en el diseño y la siguiente. Es por esto que el sistema mecánico puede ser aún mejorado.

Otra limitante fueron los motores proporcionados. Aunque los Lynxmotion ST1 son adecuados para realizar proyectos y tienen una interfaz amigable con la aplicación, su capacidad de torque no es la mejor. Se tuvo que sacrificar bastante el tamaño del brazo para poder cumplir las restricciones del torque de cada motor.

6.1. Proceso de diseño mecánico

La palabra diseño es muy común en temas de ingeniería debido a que se debe buscar nuevas formas de innovar y mejorar los sistemas ya existentes. El diseño de ingeniería se puede definir como el proceso de aplicar las diversas técnicas y principios científicos con el propósito de definir un dispositivo, proceso o sistema con suficientes detalles que permitan su realización, [9]. Los ingenieros con frecuencia deben buscar la forma de estructurar estos problemas. El primer punto importante es recordar que el proceso de diseño no es lineal, sino que necesita de varias iteraciones para lograr su objetivo. Después de cada uno de los pasos planteados a continuación, se debe analizar si no surgieron nuevos problemas y si realmente se está solucionando el problema inicial.

6.1.1. Identificar la necesidad

Usualmente esta parte es breve y sin muchos detalles. Solamente se plantea el problema a resolver.

6.1.2. Investigación preliminar

Este paso es de los más importantes, pero en muchos casos no se les da la importancia necesaria dado que es un proceso largo y en ocasiones tedioso. Se busca reunir información sobre que se ha hecho anteriormente en casos similares y buscar formas de como mejorar estas ideas.

A la hora de realizar la investigación para un trabajo formal, se debe tomar en cuenta que las fuentes sean confiables y robustas para contar con información fidedigna. Otro problema común es perderse en la investigación y comenzar a resolver el problema equivocado.

Siempre hay que asegurarse que se está totalmente preparado antes de comenzar a resolver el problema.

6.1.3. Planteamiento de objetivos

Se buscan tres características principales: ser conciso, general y funcional. Esto evita confusiones y genera una trayectoria clara para poder seguir trabajando la solución del problema.

6.1.4. Especificaciones de desempeño

Estas definen lo que el sistema debe hacer. No se deben confundir con las especificaciones de diseño, las cuales definen cómo se realizará el proyecto. No se trata de formular de inmediato como se lograrán todas las tareas necesarias, sino es más una formulación de los puntos importantes que se deben llegar a resolver. Estas especificaciones deben ser realistas y verificables.

6.1.5. Generación de ideas e invención

Este paso es posiblemente el más complicado, porque pone a prueba tanto la creatividad como la toma de decisiones del ingeniero. El proceso creativo incluso se puede considerar como un subconjunto del proceso de diseño debido a la complejidad que puede llegar a tener.

Primero se debe tener la generación de ideas. Se sugiere no juzgar la calidad de las ideas sino solamente plantearlas. Lo que se busca es tener cantidad. Cuando se tiene un equipo, se pueden realizar lluvias de ideas mientras que si se trabaja solo se pueden emplear analogías e inversiones para generar nuevos puntos de vista. Se sugiere dar unos días para este paso porque no todas las ideas surgen en las primeras horas.

6.1.6. Análisis

Se aplican técnicas de análisis a las ideas generadas en el paso anterior. En esta etapa se decide cuales de estas son viables y cumplen los requisitos necesarios. Si es necesario, se debe iterar los pasos iniciales varias veces hasta poder garantizar el éxito.

6.1.7. Selección

Dentro del grupo de diseños que lograron pasar la etapa de análisis, se debe elegir una opción para realizar los prototipos. Se puede hacer uso de una matriz de decisión.

	Costo	Seguridad	Desempeño	Confiabilidad	RANGO
Factor de ponderación	.35	.30	.15	.20	1.0
Diseño 1	3 1.05	6 1.80	4 .60	9 1.80	5.3
Diseño 2	4 1.40	2 .60	7 1.05	2 .40	3.5
Diseño 3	1 .35	9 2.70	4 .60	5 1.00	4.7
Diseño 4	9 3.15	1 .30	6 .90	7 1.40	5.8
Diseño 5	7 2.45	4 1.20	2 .30	6 1.20	5.2

Figura 10: Ejemplo matriz de decisión, 9.

6.1.8. Diseño detallado

En este paso usualmente se utilizan herramientas de CAD para generar ensamblajes del prototipo escogido en el paso anterior. Todas las piezas deben diseñarse con las medidas que se esperan tener físicamente. Es posible que se encuentren errores que necesiten de nuevo una iteración.

6.1.9. Creación de prototipos y pruebas

Se construye el modelo físico que se había diseñado en CAD. Puede que este resulte ser el paso más caro, pero sigue siendo más económico que probar con el modelo final. Es preferible hacerlo en la escala real. Las pruebas pueden comenzarse en ambientes controlados, pero deben finalizar en el ambiente donde se espera que opere el diseño.

6.1.10. Producción

Este ya es el último paso. Después que se lograron pasar todas las pruebas necesarias, se construye la versión final del diseño. Se espera que para este paso ya se hayan hecho las iteraciones necesarias para que el diseño realice las tareas que se plantearon al inicio cumpliendo con todas las especificaciones.

6.2. Robótica

La robótica comenzó por el deseo de construir máquinas capaces de replicar acciones humanas de una forma más rápida y eficaz. Después de los avances alcanzados a lo largo de los años, se establecieron 3 leyes fundamentales:

1. Un robot no puede dañar a un ser humano o habilitarlo para causar daño.

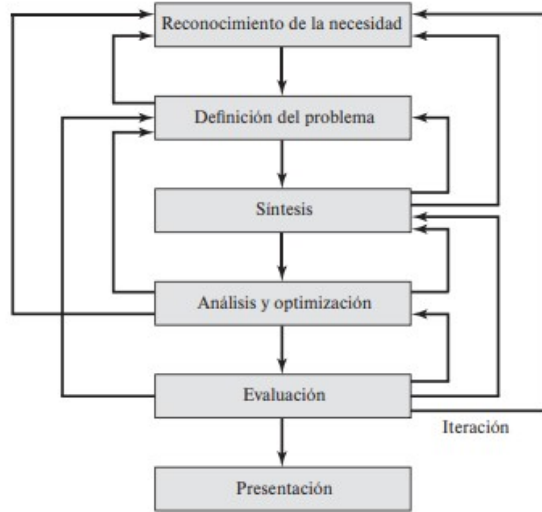


Figura 11: Diagrama simplificado diseño de ingeniería, [10].

2. Un robot debe obedecer las ordenes dadas por un ser humano, a no ser que cause conflicto con la primera regla.
3. Un robot debe proteger su existencia, siempre y cuando no exista conflicto con las primeras dos reglas.

Los componentes esenciales de un robot incluyen el sistema mecánico, los aparatos de locomoción y los manipuladores. La capacidad de ejercer una acción la proveen los actuadores. Asimismo los sensores permiten que el robot pueda percibir. El sistema de control permite ejecutar comandos acorde a los objetivos y a las medidas percibidas. Adicionalmente, se pueden planificar trayectorias e imponer límites de funcionamiento. Es por esto que se considera una multidisciplina que contiene aspectos mecánicos, de control, programación y electrónica. [11]

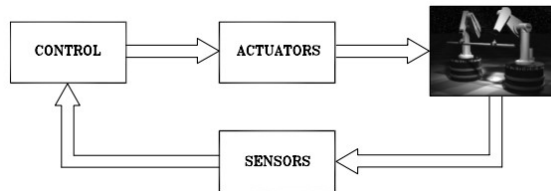


Figura 12: Componentes básicos del sistema robótico, [11].

6.2.1. Robots manipuladores

Este tipo de robots consisten en un conjunto de cuerpos rígidos interconectados por medio de articulaciones. Se tienen tres partes principales. El brazo son los primeros eslabones y ayudan que el robot pueda mobilizarse, después se encuentra la muñeca que ayuda en la destreza del robot y finalmente el efector final realiza la tarea de manipulación. Este debe tener las dimensiones necesarias para poder realizar la tarea provista.

La estructura es una cadena cinemática abierta. Esto significa que solo hay una secuencia que conecta la base con el efector final. Este tipo de estructura se utiliza cuando la carga no es muy grande y se quiere más espacio de trabajo. Las articulaciones pueden ser tanto revolutas como prismáticas. Las juntas primáticas tienen un movimiento traslacional mientras que las revolutas tienen movimiento rotacional y suelen ser más compactas y seguras. Los grados de libertad es el número mínimo de parámetros para definir un robot y estos deben estar distribuidos de una forma correcta para maximizar el área de funcionamiento. El número de grados de libertad se puede calcular utilizando la ecuación de Grübler. [12]

$$dof = m(N - 1 - J) + \sum_{i=1}^J f_i(x). \quad (1)$$

Donde $m = 3$ para mecanismos planares, $m = 6$ para mecanismos espaciales, N es el número de eslabones, J es el número de juntas y f_i es el número de grados de libertad para cada eslabón.

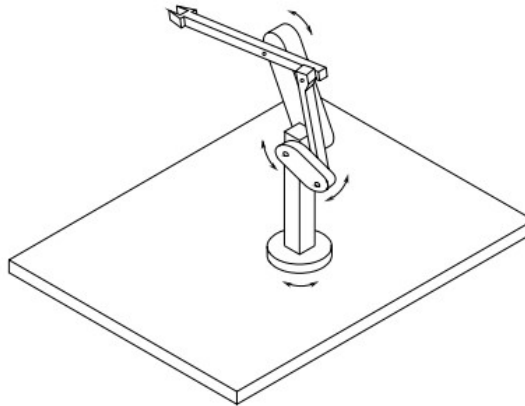


Figura 13: Manipulador serial, [11].

El espacio de trabajo representa la porción del espacio donde puede llegar el efector final. Este espacio es definido dependiendo de las juntas elegidas, donde se encuentran en el robot y el tamaño de los eslabones. El espacio de tarea es el espacio donde el efector final del robot debe llegar para realizar una tarea específica, [12].

6.2.2. Cinemática en manipuladores seriales

Como se había mencionado anteriormente, un manipulador puede ser representado como una cadena cinemática de cuerpos rígidos que están conectados por juntas revolutas o prismáticas. El comienzo de la cadena está en la base y el final está en el efector final. Por lo tanto, el movimiento de cada cuerpo rígido depende de la composición de los cuerpos que van antes en la cadena.

La pose de un cuerpo rígido sirve para describir tanto su posición como su orientación respecto a un marco de referencia. Este marco de referencia puede situarse en cualquier lugar del espacio. Usualmente se trabaja con un marco de referencia inercial, que dicta los

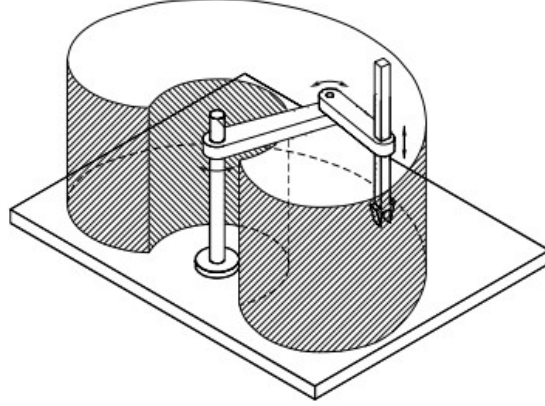


Figura 14: Espacio de trabajo de robot SCARA, [11].

ejes y dirección del lugar donde se está trabajando, un marco de referencia de la base y otro para cada cuerpo rígido en movimiento. Este marco de referencia no debe estar en un punto específico de cada cuerpo rígido pero debe ser estacionario relativo al cuerpo. [12]

Movimiento en 2D

Para poder describir una configuración, se debe especificar tanto su posición como orientación respecto a un marco de referencia inercial. Se puede expresar en término de los ejes de coordenadas [2], o en término del ángulo con respecto al eje x [3].

$$p = p_x \hat{x}_s + p_y \hat{y}_s. \quad (2)$$

$$\begin{aligned} \hat{x}_b &= \cos \theta \hat{x}_s + \sin \theta \hat{y}_s. \\ \hat{y}_b &= \cos \theta \hat{y}_s - \sin \theta \hat{x}_s. \end{aligned} \quad (3)$$

De forma simplificada, se puede expresar un punto como un vector columna [4].

$$p = \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \quad (4)$$

Al sustituir [3] en [4] se obtiene una matriz de rotación. Cada columna de una matriz de rotación debe ser un vector unitario y las columnas deben ser ortogonales. La matriz [5] representa la orientación de b con respecto al marco inercial.

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}. \quad (5)$$

Las matrices de rotación y vectores posición tienen 3 roles principales:

- Representar una configuración de un cuerpo rígido.
- Cambiar el marco de referencia en el que un vector o pose está representado.
- Desplazar un vector o pose.

Representación en 3D

Los principios del movimiento y rotación en 2D se pueden aplicar para el caso en 3D.

$$p = p_x \hat{x}_s + p_y \hat{y}_s + p_z \hat{z}_s. \quad (6)$$

$$\begin{aligned} \hat{x}_b &= r_{11} \hat{x}_s + r_{21} \hat{y}_s + r_{31} \hat{z}_s. \\ \hat{y}_b &= r_{12} \hat{x}_s + r_{22} \hat{y}_s + r_{32} \hat{z}_s. \\ \hat{z}_b &= r_{13} \hat{x}_s + r_{23} \hat{y}_s + r_{33} \hat{z}_s. \end{aligned} \quad (7)$$

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \quad (8)$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \quad (9)$$

Como se puede observar en (8) y en (9), ahora se necesitan 12 parámetros para poder describir totalmente la posición y orientación de un cuerpo rígido en 3D. Sin embargo, solamente 3 de los 9 elementos de R pueden ser escogidos de forma independiente debido a que la orientación tiene tres grados de libertad.

Las rotaciones elementales alrededor de cada uno de los ejes se muestran a continuación. Estas son positivas si se realizan en contra de las agujas del reloj alrededor del eje.

$$Rot(\hat{x}, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}. \quad (10)$$

$$Rot(\hat{y}, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}. \quad (11)$$

$$Rot(\hat{z}, \theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (12)$$

6.2.3. Representación de ángulos

Como se mencionó anteriormente, las matrices de rotación dan información redundante debido a que solamente 3 de sus parámetros son independientes para cumplir con la ortogonalidad. Una rotación genérica puede realizarse al multiplicar 3 rotaciones elementales, garantizando que no habrán dos rotaciones seguidas en un eje paralelo. Existen 12 combinaciones y se denominan como sets de ángulos de Euler. 12

Ángulos ZYZ

Se realizan las siguientes rotaciones:

- R_z alrededor del eje z.
- R_y alrededor del eje y' .
- R_z alrededor del eje z'' .

6.2.3.2 Ángulos ZYX (*Roll-Pitch-Yaw*)

Esta representación se utiliza en el ámbito aeronáutico porque denotan los cambios de altitud. Ahora las rotaciones son:

- R_x alrededor del eje x.
- R_y alrededor del eje y' .
- R_z alrededor del eje z'' .

Representación eje-ángulo

Se pueden utilizar cuatro parámetros para describir la rotación de un ángulo alrededor de un eje. Se toma el siguiente vector r y se realizan 3 pasos.

$$r = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}. \quad (13)$$

- Se debe alinear r con el eje z. Esto se alcanza rotando $-\alpha$ alrededor de z y $-\beta$ alrededor de y.
- Rotar ϕ alrededor del eje z.
- Realignar con la dirección inicial de r . Esto se alcanza rotando β alrededor de y y α alrededor de z.

Cuaterniones

Esta es otra representación de cuatro parámetros, pero donde se busca resolver el problema de la no unicidad. Se define $Q = \eta, \epsilon$ donde:

$$\eta = \cos \frac{\phi}{2}. \quad (14)$$

$$\epsilon = \sin \frac{\phi}{2} r. \quad (15)$$

Donde η es la parte escalar y $\epsilon = [\epsilon_x, \epsilon_y, \epsilon_z]^T$ es la parte vectorial. Estas deben seguir la condición:

$$\eta^2 + \epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 = 1. \quad (16)$$

Con esto, se puede observar que se logra arreglar el problema de la no unicidad.

6.2.4. Transformación homogénea

Con lo planteado anteriormente, se conoce que un punto P con el marco de referencia 1 respecto al marco de referencia 0 se puede expresar de la siguiente manera:

$$p^0 = o_1^0 + R_1^0 p^1. \quad (17)$$

La transformación inversa se obtiene al premultiplicar ambos lados por $R^0 T_1$.

$$p^1 = -R_0^1 o_1^0 + R_0^1 p^0. \quad (18)$$

Sin embargo, para poder representar de forma compacta la relación entre los dos marcos inerciales, se debe utilizar la representación homogénea del vector p como se observa en (19).

$$\bar{p} = \begin{bmatrix} p \\ 1 \end{bmatrix}. \quad (19)$$

Ahora sí se puede representar la transformación como:

$$A_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix}. \quad (20)$$

Esto se conoce como la matriz de transformación homogénea. Esta incluye el vector de traslación y la matriz de rotación. Para la inversa, se debe recordar que:

$$A^{-1} \neq A^T. \quad (21)$$

$$A_1^0 = \begin{bmatrix} R_0^1 & -R_0^1 o_1^0 \\ 0^T & 1 \end{bmatrix}. \quad (22)$$

Para realizar varias transformaciones, se aplica el mismo concepto que las matrices de rotación.

$$\bar{p}^0 = A_1^0 A_2^1 \dots A_n^{n-1} \bar{p}^n. \quad (23)$$

6.2.5. Cinemática directa

El propósito de la cinemática directa es computar la pose del efector final como una función de las variables de las juntas. Esta expresión se puede representar con la siguiente matriz:

$$T_e^b(q) = \begin{bmatrix} n_e^b(q) & s_e^b(q) & a_e^b(q) & p_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (24)$$

Donde q es un vector de $(n \times 1)$, n , s y a son los vectores unitarios del marco de referencia del efector final y p es la posición del vector del origen con respecto al origen. Los vectores a , s y n son normales entre si, a representa el acercamiento mientras que s representa el deslizamiento. Este problema se vuelve más complicado mientras incrementa el número de juntas y todavía más complicado si existen cadenas cinemáticas cerradas. [11]

Una vez se tiene la matriz que describe el efecto de cada una de las juntas, se debe considerar la posición del robot con respecto al marco inercial. En la Ecuación (25), se puede observar cómo la transformación de base y transformación del efector final pueden afectar a la pose del robot con respecto al marco inercial. ${}^I T_B$ representa la transformación de la base del robot respecto al marco inercial, mientras que ${}^E T_F$ representa la transformación del efector final respecto a la última junta del robot.

$${}^I T_F(q) = {}^I T_B \times {}^B T_E(q) \times {}^E T_F \quad (25)$$

Convención Denavit-Hartenberg

Para realizar la cinemática directa de una cadena cinemática abierta, se sigue la ecuación (23). Se desarrolló un método que define la posición y orientación relativa de dos eslabones consecutivos. El mayor problema es determinar los marcos de referencia que representan a cada eslabón. Dado que se pueden escoger de forma arbitraria, la convención Denavit-Hartenberg define lo siguiente:

- Se escoge el eje z a lo largo de la junta $i + 1$.
- Se determina el origen de i como la intersección de los ejes z_i con la normal común de z_{i-1} y z_i .
- Se escoge el eje x como la normal común entre los ejes z_{i-1} y z_i con una dirección positiva de la junta i hacia la junta $i + 1$.
- Se escoge el eje y para que cumpla la regla de la mano derecha.

Se tienen los siguientes casos especiales:

- Para el marco de referencia 0, solamente se especifica el eje z , el origen y eje x se pueden escoger arbitrariamente.
- Para el marco de referencia n , como no existe $n + 1$, no se define z_n pero x_n debe ser normal a z_{n-1} . Si es una revoluta, se puede alinear con z_{n-1} .
- Cuando dos ejes consecutivos son paralelos, la normal común entre ellos no es definida de forma única.
- Cuando dos ejes consecutivos se intersecan, la dirección positiva de x_i es arbitraria.
- Cuando la junta i es prismática, solamente la dirección z_{i-1} es dada.

Dos de los parámetros, a_i y α_i siempre son constantes, y dependen solamente de la geometría de la conexión entre las juntas consecutivas establecidas en el eslabón i . De los dos parámetros que faltan, solamente uno depende del tipo de junta que conecta $i-1$ a i .

- Si i es revoluta, la variable es el ángulo ζ .
- Si i es prismática, la variable es d_i .

Ahora ya se puede expresar las coordenadas de la transformación entre el marco i y el marco $i-1$.

- Se escoge un marco alineado con el marco $i-1$.
- Se traslada el marco por una distancia d_i a lo largo del eje z_{i-1} y se rota por ζ_i alrededor del eje z_{i-1} . Este marco se alinea con el marco i' .
- Se traslada el marco alineado con i' por a_i y se rota por α_i alrededor del eje x_i . Este marco se alinea con el marco i .
- La matriz de transformación se logra al multiplicar estas dos transformadas.

Debe notarse que la transformada es una función solamente de q_i y ζ_i si es revoluta o d_i si es prismática. En resumen, la convención Denavit-Hartenberg permite la construcción de cinemática directa con la composición de transformadas individuales para formar una transformada homogénea. 11

6.2.6. Cinemática diferencial

Este tipo de cinemática presenta la relación entre las velocidades de las juntas con las velocidades del efector final. Este mapeo se realiza mediante la matriz del jacobiano geométrico, la cual depende de la configuración del manipulador. Si se da la pose del efector final, es posible computar el jacobiano por medio de la cinemática directa con respecto a las variables de las juntas. Este jacobiano es el diferencial. El jacobiano ayuda a determinar singularidades, redundancia y determinar algoritmos de cinemática inversa.

Jacobiano geométrico

La cinemática directa para un manipulador de n grados de libertad se expresa en (26).

$$T_e(q) = \begin{bmatrix} R_e(q) & p_e(q) \\ 0^T & 1 \end{bmatrix}. \quad (26)$$

Donde $\mathbf{q} = [q_1 \dots q_n]^T$ es el vector de las variables de las juntas. Tanto la posición como orientación del efector final varían mientras cambia \mathbf{q} .

El objetivo de la cinemática diferencial es encontrar la relación entre las velocidades de las juntas \mathbf{q} y las velocidades lineales \dot{p}_e y angulares $\dot{\omega}_e$ del efector final.

$$\dot{p}_e = J_P(q)\dot{q}. \quad (27)$$

$$\dot{\omega}_e = J_O(q)\dot{q}. \quad (28)$$

Tanto J_P como J_O con matrices de $(3 \times n)$, y se pueden escribir de forma compacta en (29)

$$v_e = \begin{bmatrix} \dot{p}_e \\ \dot{\omega}_e \end{bmatrix} = J(q)\dot{q}. \quad (29)$$

Esto representa el manipulador diferencial de la ecuación de cinemática. La matriz J de $(6 \times n)$ se llama el jacobiano geométrico.

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}. \quad (30)$$

Singularidades

Estas ocurren cuando \mathbf{J} es de rango deficiente en la ecuación, (29).

- Las singularidades representan configuraciones en las que la movilidad de la estructura es reducida.
- Cuando la estructura está en una singularidad, existen infinitas soluciones para la cinemática inversa.
- En las vecindades de la singularidad, pequeñas velocidades en el espacio de operación causan grandes velocidades en las juntas.

Se pueden categorizar como de frontera o internas. Las de frontera con las que ocurren cuando el manipulador está totalmente estirado o retractado. Las internas ocurren cuando se alinean dos o más ejes de los eslabones. Estas son las más peligrosas porque no se detectan con facilidad y se encuentran dentro del espacio de trabajo. [11]

Jacobiano analítico

La pose del efector final es especificada en término del mínimo número de parámetros en el espacio de operación. La velocidad traslacional del marco del efector final puede ser expresada como:

$$\dot{p}_e = J_P(q)\dot{q}. \quad (31)$$

En cuanto a la velocidad rotacional:

$$\dot{\phi}_e = J_\phi(q)\dot{q}. \quad (32)$$

El jacobiano no se puede operar directamente, sino que requiere computación de elementos relativos a la matriz de rotación. Con estas premisas, se puede llegar a la cinemática diferencial con:

$$\dot{x}_e = J_A(q)\dot{q}. \quad (33)$$

Es posible encontrar la relación entre la velocidad angular ω y la velocidad rotacional $\dot{\phi}$.

$$\omega_e = T(\phi_e)\dot{\phi}_e. \quad (34)$$

El significado de ω es más intuitivo que el de $\dot{\phi}$, porque sus tres componentes son respecto al marco base mientras que el marco de $\dot{\phi}$ cambia siempre que cambie la pose del efector final. La relación entre el jacobiano analítico y geométrico es la siguiente:

$$J = T_A(\phi)J_A. \quad (35)$$

Esto sugiere que en la mayoría de escenarios serán diferentes. El geométrico será utilizado cuando es necesario referirse a las cantidades con significado físico mientras que el analítico se utiliza cuando es necesario referirse a las cantidades diferenciales de las variables definidas en el espacio de operación. [11]

6.2.7. Cinemática inversa

Este enfoque consiste en determinar las variables de las juntas correspondientes a una pose del efector final dada. Este tipo de problemas son comunes cuando se desea llegar a ciertos puntos y para más adelante lograr definir trayectorias. Se presentan las siguientes complicaciones:

- Las ecuaciones son no lineales, por lo que no siempre se puede encontrar una solución.
- Pueden existir múltiples soluciones.
- Pueden existir infinitas soluciones, cuando es redundante.
- Pueden haber soluciones no admisibles debido a la estructura o ambiente.

Dado que la cinemática diferencial representa un mapeo entre las velocidades de las juntas y las velocidades en el espacio de operación, se pueden utilizar estos métodos para resolver el problema. La fórmula general es:

$$q(t_{k+1}) = q(t_k) + J^{-1}(q(t_k))v_e(t_k)\Delta t. \quad (36)$$

Con el objetivo de solucionar el problema que las velocidades \dot{q} no coincidan en el tiempo continuo, se calcula el error de espacio de operación:

$$e = x_d - x_e. \quad (37)$$

A partir de estos principios, se realiza el algoritmo de cinemática inversa. Este relaciona \dot{q} con el error e para describir la evolución del error en el tiempo. Se debe asegurar que se escoja una relación que permita que e converja a 0. Esto se puede ver de una forma más clara en la Figura (15).

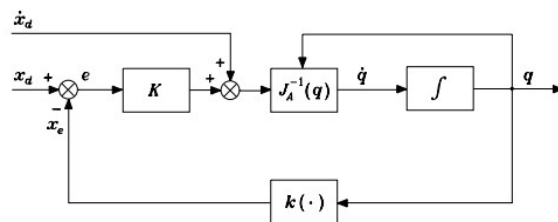


Figura 15: Algoritmo cinemática inversa, [11]

Un caso particular es el jacobiano de la pseudo-inversa. En este caso se asume que J_A es una matriz cuadrada y no singular. Por lo que (38) se puede simplificar a (39).

$$\dot{q} = J_A^{-1}(q)(\dot{x}_d + Ke). \quad (38)$$

$$\dot{e} + Ke = 0. \quad (39)$$

Si K es una matriz positiva y usualmente diagonal, el sistema es asintóticamente estable. El error tiende a 0 y depende de la magnitud de los eigenvalores de K , mientras mayor sean, más rápido converge. Dado que se toma en tiempo discreto, el límite superior depende de los eigenvalores. [11]

6.3. Mecanismos

El grado de libertad de un grupo de eslabones determina si puede ser considerado un mecanismo. Si el grado de libertad es mayor a 0, puede ser un mecanismo porque tendrá movimiento, de lo contrario solamente es una estructura. Las juntas son los puntos donde se unen dos eslabones. Así como se definió anteriormente, las juntas más comunes son las revolutas y las prismáticas. En mecanismos de dos dimensiones, solamente se puede tener alguna de estas dos o una combinación de ellas.

El mecanismo de 4 barras es el mecanismo articulado más simple posible para movimiento controlador con grado de libertad simple. La simplicidad es un signo de buen diseño y es por esto que se reconoce como uno de los dispositivos más comunes en la maquinaria. Otro aspecto positivo es la variedad de movimientos que se pueden generar.

Las configuraciones del mecanismo de 4 barras incluyen manivela balancín, manivela corredera, doble manivela, doble balancín, entre otros. Una configuración que es muy utilizada en robots es la del paraleloramo. Este tipo de configuración duplica con exactitud el movimiento rotatorio de la manivela impulsadora en la impulsada. En este mecanismo, se conserva el mismo ángulo y velocidad angular para las manivelas. [9]

6.4. Motores y controladores Lynxmotion

Para este proyecto, se utilizarán los productos de Lynxmotion. Se usarán los *smart servos* ST1 y el adapter board para poder controlarlos. Un servomotor inteligente es un motor configurable que realiza conjeturas para poder alinearse y ajustar sus parámetros.

6.4.1. Lynxmotion ST1

Es un servo motor configurable y modular. Puede llegar hasta los 60 rpm sin carga, tiene un torque estático de 14 Kg-cm y un máximo torque dinámico de 2.8 Kg-cm. Se alimenta con 12V, tiene una rotación de 360 grados y una precisión de 1.5 grados.

Están diseñados para utilizarse en robots con varios grados de libertad, animatrónicos y proyectos RC. Los parámetros son configurables por el usuario y tiene un sensor de feedback de voltaje, corriente, temperatura, posición y rpm que sirve de protección. Aunque funcionan mejor cuando se controlan desde un microcontrolador, se pueden utilizar como actuadores RC.

El protocolo de comunicación serial utilizado es el SSC-32/32U. Esto presenta una opción fácil e intuitiva de utilizar. [13]

6.4.2. Lynxmotion Smart Servo Adapter Board

Esta placa es necesaria porque no se pueden conectar los motores directamente. El problema es que los servomotores ST1 no tienen una interfaz de comunicación incluida, y por eso es necesario este componente.

Esto permite una conexión fácil y se puede realizar el control de los servomotores inteligentes. Idealmente debería ser alimentado por una batería LiPo con un conector XT60. Es compatible con los escudos de Arduino y también es compatible con una Raspberry Pi. [14]

Las especificaciones incluyen:

- Control de los acutadores LSS están en el mismo bus.
- Bee Socket, USB y conexión a Arduino.
- Selección de fuente de poder automática
- Corriente por conector de 3 A.
- Reguladores de 5V y 3.3V incorporados.
- Conector USB mini
- USB a serial incorporado por medio de FTDI COM Chip
- Hoyos para montaje de 3.1 mm.

6.5. ROS

El Robot Operating System (ROS) es un conjunto de librerías y herramientas de fuente abierta para el desarrollo del software de robots. Tiene sus orígenes en las investigaciones de inteligencia artificial, pero en las últimas décadas agarró su propia importancia y se desarrolló más a profundidad. Su arquitectura modular permite construir robots sin licencias y es compatible con cualquier componentes que tenga una interfaz de software. Con ROS se puede trabajar con una simulación del robot antes de realizar las pruebas con el sistema físico. [15]

Existen 3 variantes principales de ROS. ROS Noetic Nimjumys es la última versión de ROS 1 LTS y está dirigido para Ubuntu 20.04, aunque otros sistemas también son compatibles. ROS Foxy Fitzroy es la última versión de ROS 2 LTS y es compatible con Ubuntu Focal, macOS y Windows 10. ROS Galactic Geochelone también tiene la compatibilidad de Foxy, pero no cuenta con soporte de largo plazo. Esto significa que no es tan estable y tiene menos pruebas que Foxy. Existen muchas más versiones de ROS porque lanzan versiones cada año, y versiones con soporte de largo plazo cada 2 años. ROS 2 trabaja con microROS, una variante de ROS que corre de forma nativa en sistemas embebidos de tiempo real. [15]

Los procesos de ROS funcionan por nodos. Estos están conectados por temas (topics) para poder intercambiar información. Los nodos pueden enviar o recibir mensajes, guardar o recuperar datos y pedir algún servicio a otro nodo. Se tiene un proceso llamado maestro que registra todos los nodos y configura la comunicación de nodo a nodo para poder controlar las actualizaciones. El sistema de intercambio de información (middleware) ahorra tiempo al organizar los detalles de comunicación entre nodos por medio de patrones anónimos de publicar y suscribir. Esto facilita las interfaces y reutilización de código. [16]

Su arquitectura está basada en grafos, donde los nodos pueden recibir, mandar o multiplexar mensajes de los sensores, estados y actuadores. Aunque ofrece muchas de las funcionalidades de un sistema operativo, no se considera como tal. Es un kit de software de desarrollo que provee los componentes necesarios para construir aplicaciones robóticas. [16]

ROS tiene dos partes principales, el sistema operativo y el pkg. El pkg consiste en un conjunto de paquetes aportados por la contribución de usuarios, organizados en stacks, que implementan las funcionalidades como localización, planificación, simulación, entre otros. [17]

Un nodo es un proceso que realiza ciertos cálculos, conectados por temas y que están destinados a operar a una escala definida. Por lo general, se separan por su función. Algunos están encargados de leer la información de los sensores y mandar estos datos, mientras que otros reciben la información y en base a un algoritmo deciden que hacer con los actuadores. También se pueden tener nodos para realizar control con retroalimentación. [17]

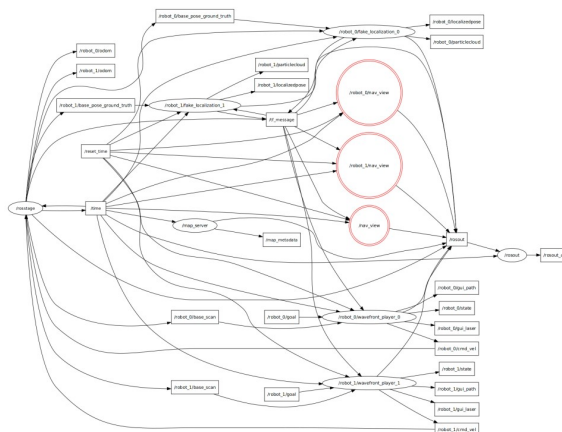


Figura 16: Ejemplo de nodos de ROS, [17].

6.6. Simulación

6.6.1. Modelos URDF

El formato de descripción de robot universal es un archivo de tipo XML utilizado por ROS para describir la cinemática, propiedades inerciales y comportamiento de los eslabones de un robot en particular.

Las juntas conectan dos eslabones. Uno de ellos se debe denominar el principal y el otro se llama secundario. Las juntas más utilizadas son prismáticas, revolutas, continuas y fijas. Cada junta tiene un origen que se sitúa en el eje del eslabón principal para poder definir la posición y orientación del secundario.

Los eslabones sirven para definir las propiedades de la masa. Es por esto que se debe definir el centro de masa y una matriz de inercia.

Un modelo URDF puede definir cualquier robot con una estructura de árbol. Las orientaciones relativas se alcanzan utilizando las coordenadas RPY. Se necesita tener un marco de referencia para cada junta.

Además de esto, también se puede representar la apariencia de un robot para poder realizar simulaciones de planificación de trayectorias y colisiones. [\[12\]](#)

6.6.2. RVIZ

RVIZ es el visualizador 3D de datos nativo de ROS. Este permite suscribirse a tópicos que se estén publicando en ROS y mostrarlos en un sistema de coordenadas. Se pueden visualizar datos en tiempo real, estos pueden estar relacionados con un robot, un sensor o un algoritmo. Presenta una forma sencilla de escribir a ciertos tópicos y utilizar plugins.

Esta herramienta permite saber cómo el robot está interactuando con su entorno. Se tiene información del estado sensorial para recopilar información y marcadores de visualización para poder visualizar los cambios. Al unir estas dos herramientas es que se mide la capacidad del robot.

Uno de los paquetes más populares es MoveIt!. Este se encarga de resolver la cinemática del robot, planificar trayectorias, comprobar colisiones e interactuar con el robot. [\[18\]](#)

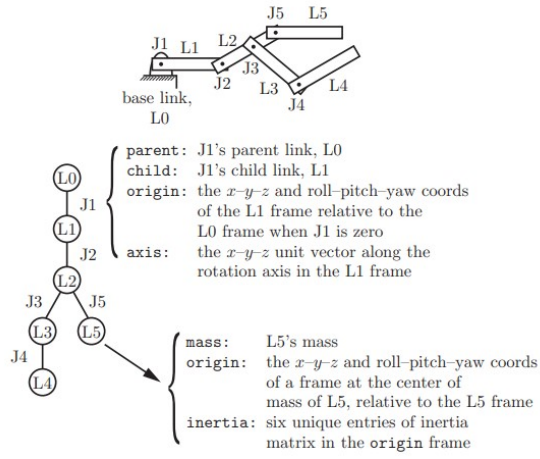


Figura 17: Estructura de árbol URDF, [12].

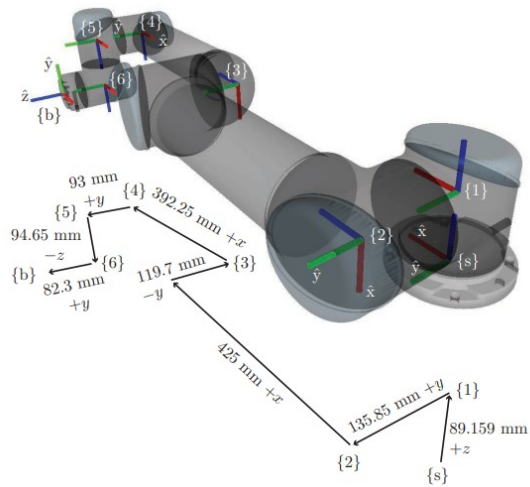


Figura 18: Modelo URDF, [12].

Diseño electromecánico del brazo

Como todo proceso de diseño, se realizaron varias versiones en la etapa de modelado y prototipado antes de llegar a la versión que se entregará. La necesidad que se busca satisfacer en este proyecto es dotar al Rover UVG con un brazo para incrementar su autonomía. Sin embargo, se tienen limitaciones en cuanto a los motores que se obtuvieron y a los materiales disponibles.

Se realizó una investigación preliminar donde se conoció sobre algunas técnicas para la construcción de brazos robóticos industriales y caseros. Se notó que en los brazos industriales, donde los motores son más pesados, se tiene más importancia por reducir el torque producido por su peso ya sea cambiando su posición o utilizando contrapesos. Al mover los motores de lugar se debe tener un sistema de transmisión de potencia para lograr entregar el movimiento desde el motor hasta la junta respectiva.

El objetivo del diseño es el mismo del proyecto general, pero después de realizar la investigación se tomó como parámetro limitante el torque que puede entregar cada motor y se sacrificó el tamaño del brazo. Sin embargo, las funciones debían ser las mismas que se plantearon en los objetivos originales. En cuanto al desempeño, se espera que el robot sea capaz de llegar a las posiciones indicadas, pero la velocidad y aceleración serán factores limitantes para no forzar los motores. Deberán emplearse velocidades bajas para no requerir una aceleración alta por parte de los motores. Para el control por medio de ROS no se espera que se tengan sistemas de trayectorias avanzadas sino que se tenga la base para poder realizar el control de un brazo por medio de ROS.

Durante la generación de ideas, se supuso que la mejor solución sería utilizar sistemas de poleas para lograr colocar los motores lo más cerca posible del eje central del robot y así disminuir el torque de su peso. Adicionalmente fue un sistema de transmisión de potencia que permitía aumentar el torque de salida. Estas poleas podían ser con fajas o con cuerdas. También se consideraron sistemas de engranajes y mecanismos de barras. Se decidió que los

eslabones largos serían cortados con láser mientras que las juntas serían piezas impresas en 3D por ser de geometría más complicada. Se buscó la forma de mantener las juntas en donde deberían estar según la convención Denavit Hartenberg sin la necesidad de poner eslabones fijos adicionales. Se tomó esta decisión de complicar el diseño mecánico pero simplificar la programación del controlador.

Durante la etapa de análisis del proceso de diseño, se determinó que el sistema de fajas no sería conveniente para utilizarlo en la primera iteración tomando en cuenta las siguientes consideraciones. Primero, solamente se encontraron fajas ya cortadas a nivel local, por lo que se pueden producir fuentes de error en la unión manual de estas. Se deben utilizar varios cojinetes para tener un eje de entrada, uno de salida y en ciertos casos ejes intermedios. También los piñones impresos no suelen dar los mejores resultados y pueden presentar otra fuente de error. Además de esto, se debe tener un sistema para tensar la faja para poder realizar el montaje y desmontaje del sistema. El sistema de engranes tampoco fue viable dado que se necesitaban múltiples ejes para tener la reducción deseada con un tren de engranes o tener unos de diámetro mayor que interrumpían el movimiento.

Por estas razones se decidió cambiar el sistema de transmisión de potencia a un sistema de cuatro barras, dado que las complicaciones de la propuesta original superaron sus beneficios. Aun cuando la propuesta del mecanismo de cuatro barras no puede aumentar el torque, este no presentó complicaciones para montar el sistema y permitió colocar a los motores en una posición más cercana al eje central. Estos mecanismos debían ser paralelos, para conservar la velocidad angular y el ángulo de rotación. Este sistema se implementó para los grados de libertad tres y cuatro que corresponden al codo y a la muñeca. Las demás juntas tienen a los motores en el mismo lugar donde se ubican. Para el gripper, se buscó tener versatilidad por lo que se implementó un sistema de sujeción para que el efector final sea intercambiable y que el brazo pudiera tener más de una función. El gripper que se empleó en esta primera iteración fue uno planar, basado en un mecanismo de cuatro barras de tipo manivela balancín. Se consideró el uso de cojinetes en la base para que el peso no caiga sobre el motor y en el quinto grado de libertad para que no hubiese fricción entre las piezas impresas. Para poder incrementar el torque de las juntas 2 y 3, se optó por utilizar cajas reductoras de tipo armónicas. Esto es porque se tiene un alto torque de salida, se puede tener un diseño compacto y se tiene un alto grado de precisión. Los diseños no son propios, sino basados en la guía elaborada por Darío Marroquín en su trabajo de graduación. Se observaron los cambios necesarios y se fue iterando hasta llegar a una solución aceptable.

Después se realizó el diseño CAD y se generaron los ensamblajes para darse una buena idea de cómo funcionaría el brazo robótico. Una vez se tuvo eso terminado, se comenzó con la construcción de los prototipos. Durante la construcción de los prototipos se percató que se debía reducir aún más el tamaño del brazo para cumplir con las especificaciones de los motores, ya teniendo los pesos de las piezas impresas y cortadas.

En cuanto al diseño electrónico, no hubieron complicaciones mayores dado que el sistema de control de los motores está diseñado para ser simple. La placa adaptadora ya tenía las conexiones necesarias para utilizar los motores.

Uno de los problemas se tuvo al momento de controlar cada motor por separado. Esto fue porque cada motor tiene un identificador (ID), pero todos tienen el ID 0 de fábrica y no hay una forma fácil de cambiarlo por medio de software. La forma convencional es utilizando su

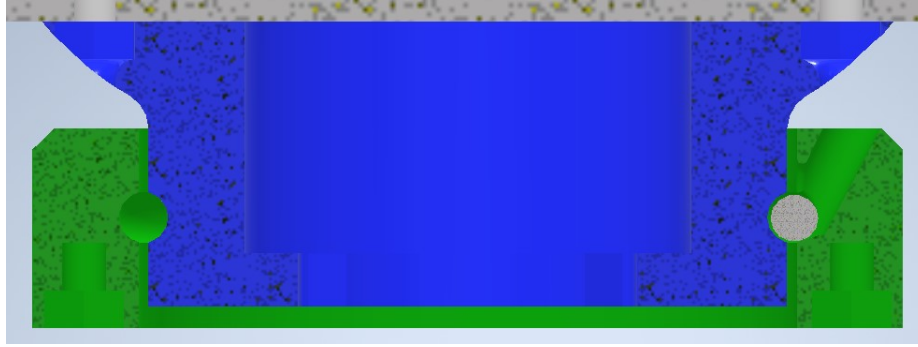


Figura 19: Primer grado de libertad, vista lateral.

aplicación, por lo que se necesita un cable mini USB para realizar la comunicación entre la computadora y los motores. De esta manera se puede cambiar el ID, permitiendo controlarlos a todos por separado.

Para la manufactura del robot, se imprimieron las partes complicadas en 3D utilizando las impresoras Ultimaker 3 de la universidad. Los eslabones largos se cortaron de MDF utilizando la cortadora láser PLS4.5. Los tornillos utilizados fueron adquiridos o reutilizados pero siempre cumpliendo con el estándar métrico. El cojinete del quinto grado de libertad fue adquirido y adecuado para su función mientras que el del primer grado de libertad fue hecho manualmente para que pudiera cumplir con las especificaciones de diseño.

En cuanto a las velocidades, se limitó la velocidad máxima para las juntas sin reducción a 10 rpm y para las que tienen reducción a 30 rpm. Después de la reducción, las juntas con caja reductora tendrán una velocidad de 1.5 rpm. Se realizaron estas modificaciones con el propósito de minimizar las aceleraciones y así disminuir el torque requerido. Además de esto, por la naturaleza de los materiales, se tiene una mayor estabilidad de movimiento a estas velocidades menores.

7.1. Primer grado de libertad

El primer grado de libertad es el que permite girar el robot respecto a su base. Lo que se buscó en este grado de libertad fue que el peso del brazo no cayera sobre el motor. Se consideró utilizar cojinetes radiales horizontalmente para que soportaran las cargas y tener un movimiento fluido. Sin embargo, esta idea sobrecomplicaba el diseño. Por esta razón, se optó por utilizar un solo cojinete axial en el centro. Para evitar utilizar partes adicionales al cojinete para poder acoplarlo tanto al motor, como al brazo y a la base, se imprimieron piezas que se asemejan a un cojinete y que ya tengan estas características. De esta manera, se facilita el ensamblaje de esta primer junta.

Como se observa en la Figura (19), se tiene una pieza superior en azul que se conecta a la base del robot. En la Figura (20), se observa como esta pieza tiene una estructura para unirse con la pieza de unión con el motor. Esto se realizó por medio de ocho uniones trapezoidales que permiten movimiento axial pero no radial. Se diseñó de esta manera para no introducir una fuente de error en la rotación, pero que se pueda ensamblar y desensamblar

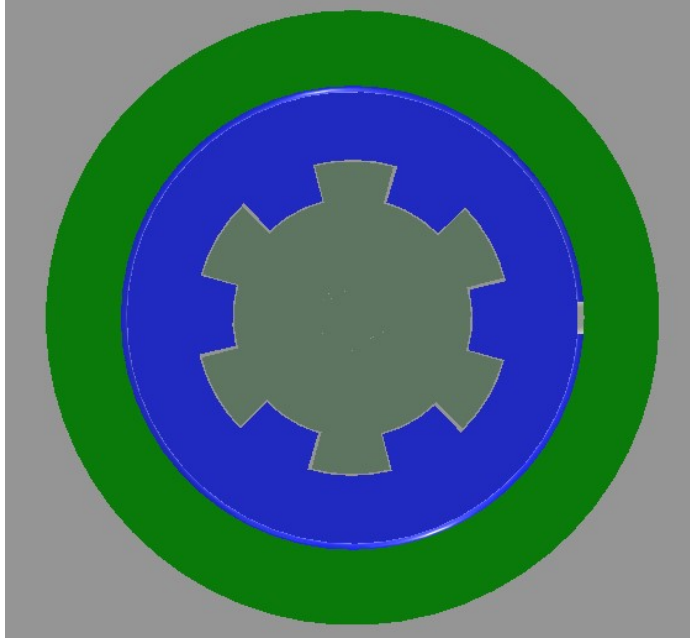


Figura 20: Primer grado de libertad, vista inferior.

con facilidad.

Tanto la pieza superior como la pieza inferior en la Figura (19) tienen espacios para introducir tornillos M4 para poder realizar la unión con las demás partes del brazo. Adicionalmente, ambas piezas tienen un carril para permitir el paso de balines de 4.5mm de diámetro. No obstante, el carril no es un semicírculo perfecto, sino está corrido medio milímetro para que el carril sea menos profundo en ambas piezas. Esto hace que exista una separación entre las piezas y no se genere fricción. Dado que el movimiento entre la parte superior y la unión con el motor permite el movimiento axial, se asegura que el peso no caiga sobre el motor, sino sobre los balines colocados entre estas piezas. En la parte inferior, se tiene una rampa con un agujero para poder introducir y sacar los balines cuando sea necesario.

7.2. Segundo y tercer grados de libertad

Como se puede observar en la Figura (21), para el tercer grado de libertad se utilizó por primera vez el mecanismo de 4 barras paralelo. Eso se realiza con el objetivo de mantener el motor de este grado de libertad en la base en lugar del codo (pieza azul). Como ya se explicó con anterioridad, el torque ejercido por el motor se mantiene, así como la velocidad angular y el ángulo de movimiento. En el caso del tercer grado de libertad (motor derecho), cuando el motor derecho rota el primer eslabón, se traslada el movimiento de rotación hacia el eslabón fijo en el codo. Esto solamente se alcanza si el motor izquierdo no presenta movimiento. Dado que se conserva el ángulo de rotación entre estos puntos, no existe mayor fuente de error al utilizar este mecanismo.

En el segundo grado de libertad (motor izquierdo) el movimiento se complica dado que

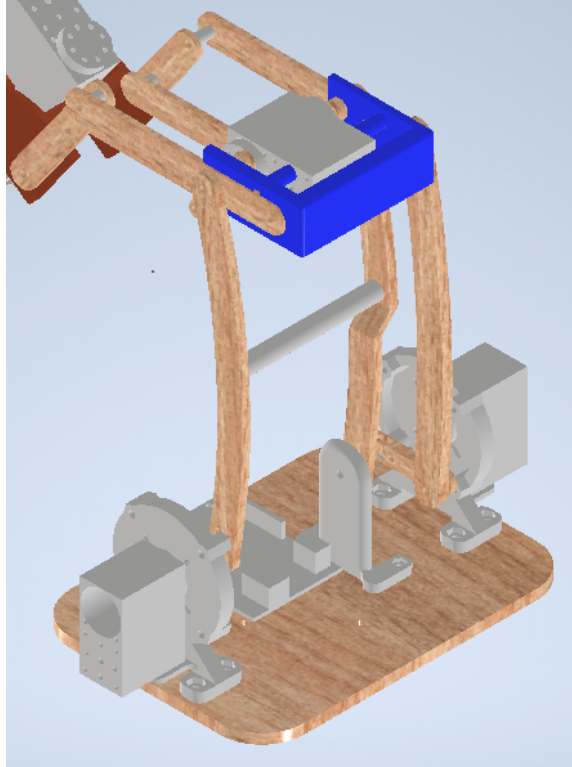


Figura 21: Segundo y tercer grados de libertad, vista isométrica.

solamente con rotar el motor izquierdo, se mueve el eslabón que tiene conexión directa pero no se conserva el ángulo en el eslabón fijo del codo. Esto fue problemático porque se debe calcular el error que puede causar y este puede llegar a ser impredecible. La solución propuesta fue mover tanto el motor de la derecha como el de la izquierda la misma cantidad de grados para que el cuadrilátero se vea obligado a mantener sus ángulos internos. Por lo tanto, la rutina del motor derecho debe ser idéntica que el izquierdo al momento de realizar el movimiento del segundo grado y después se realizan movimientos adicionales para el tercer grado de libertad solamente para el motor derecho.

Adicionalmente, después de las primeras iteraciones se observó que es necesario colocar un punto de pivote. Este conecta un eslabón a la parte derecha del codo, siguiendo la misma geometría que el resto de los eslabones. El punto de pivote debe estar en el mismo eje de rotación de los motores en la base para no afectar el movimiento de las juntas. Al introducir los cambios se observó que aumenta la estabilidad de la estructura al colocar puntos de soporte de ambos lados y se eliminó la fuente de error generada por la fuerza que ejerce el peso de la estructura.

En la Figura (22), se tiene mejor visualización sobre el cuadrilátero que se forma con los eslabones de estos dos grados de libertad. Cuando los eslabones 1 y 4 se mueven la misma cantidad de grados, los otros dos eslabones deben mantener su ángulo relativo.

Dado que estos dos grados de libertad son los que exigen más torque, fue en estos donde se aplicaron las cajas reductoras armónicas. Se utilizaron cajas armónicas impresas en PLA con un diámetro externo de 50 mm y con cojinetes de diámetro externo de 13 mm y diámetro

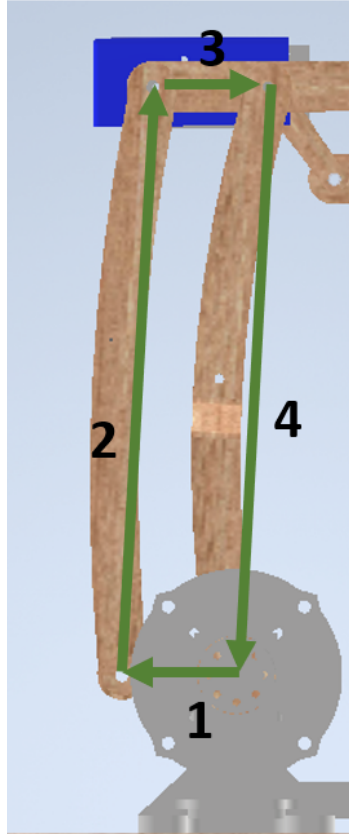


Figura 22: Segundo y tercer grados de libertad, vista lateral.

interno de 4mm. Las cajas tienen una reducción de 20:1. Dado que la velocidad máxima de los motores se configuró de 30 rpm, la velocidad de salida es de 1.5 rpm. Dado que usualmente no se tiene un movimiento mayor a 90 grados, el movimiento no debería de tardar más de 6 segundos en realizarse. La reducción requerida para soportar el peso del brazo es de 3, pero con la reducción de 20 se logra un factor de seguridad de 5. Sin embargo, las cajas pueden llegar a fallar porque los dientes de PLA no logran soportar la carga máxima teórica antes de llegar a la fractura. Se aumentó el área efectiva de los dientes de PLA lo más posible sin sacrificar el tamaño de las cajas para evitar las fallas por ruptura de dientes. También se experimentó utilizando diferentes porcentajes de relleno, llegando a un resultado satisfactorio con el 20 % donde también se consideró el ahorro de materiales. Estas no fueron diseñadas, sino se utilizó la guía de la tesis de Darío Marroquín. Se observaron los cambios necesarios y se fue iterando hasta llegar a una solución aceptable.

7.3. Cuarto grado de libertad

En este grado de libertad se utilizó el segundo mecanismo de cuatro barras paralelo. Este sistema fue más simple porque solamente debía manejar el movimiento de una junta. Se aprovechó la estructura del motor para colocar una copia de los eslabones para tener un refuerzo. Dado que los motores tienen un extremo actuado y el otro libre, se colocaron eslabones de las mismas dimensiones del lado libre del motor y estos siguen a los movimientos

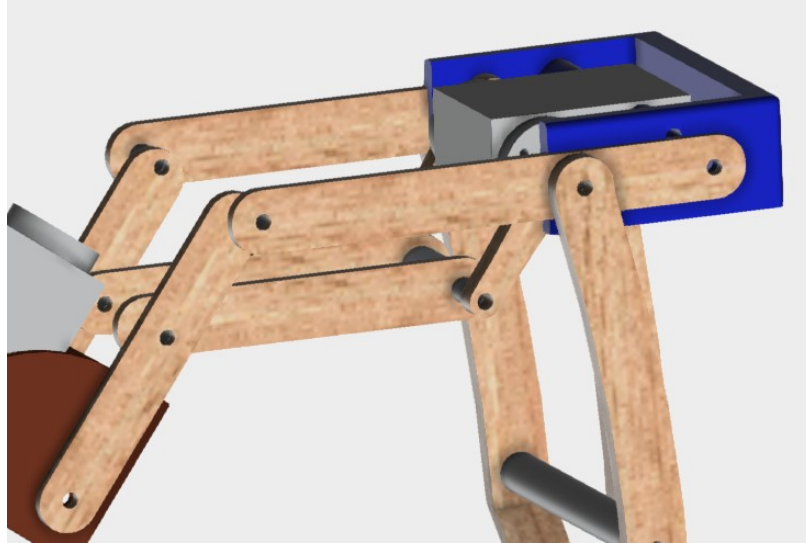


Figura 23: Cuarto grado de libertad, vista isométrica.

de los eslabones actuados por el motor.

Como se puede observar en la Figura (23), este mecanismo permite colocar el motor de la cuarta junta en el codo en lugar de la muñeca. La pieza azul del codo permite colocar el motor de tal forma que el movimiento rotacional salga exactamente del punto donde termina la segunda junta. De esta manera, se puede utilizar el eslabón de la tercer junta para formar el mecanismo de cuatro barras.

De nuevo se presenta una vista lateral en la Figura (24), en donde se puede observar que el movimiento angular del eslabón 1 será el mismo que el eslabón 3. Esto permite controlar la junta desde el codo con una relación de movimiento 1:1. Otro aspecto importante a considerar es que se necesita un eje de unión para los eslabones de manivela. Esto es necesario dado que los mecanismos de 4 barras tienen dos configuraciones y con el eje libre se pueden cambiar de una configuración a otra cuando se sobrepasa el punto de equilibrio del mecanismo.

En la configuración actual, se tiene una longitud efectiva de 80 mm para los dos eslabones, pero esto puede aumentar con las especificaciones de los motores, dado que todavía se disponen de 1.2 Kg-cm para llegar a su límite.

7.4. Quinto grado de libertad

Este grado de libertad permite el movimiento rotacional en el eje axial de la muñeca. Para este movimiento, se colocó el motor directamente en la junta. Tratar de mover el motor de esta junta al codo complicaba el diseño y se necesitaría una transmisión de potencia que lograra mantener su ángulo con el movimiento de la muñeca.

Se realizó esta geometría en la pieza de la muñeca porque de lo contrario se complicaba la conexión del motor. Para que el peso no cayera completamente en el motor, se colocó un cojinete de bolas para que este soportara la mayoría del peso del efector final y no se

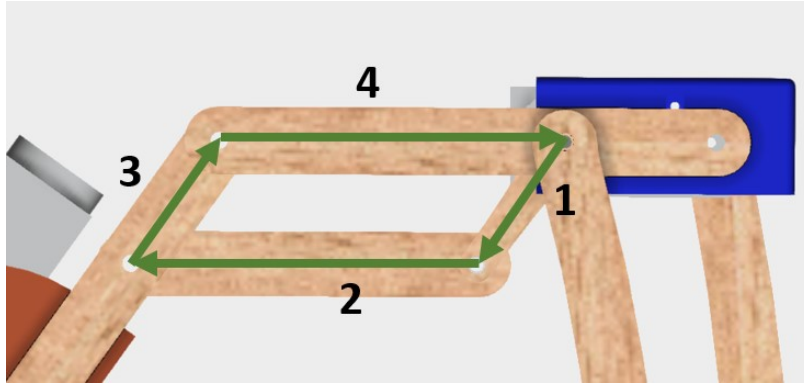


Figura 24: Cuarto grado de libertad, vista lateral.

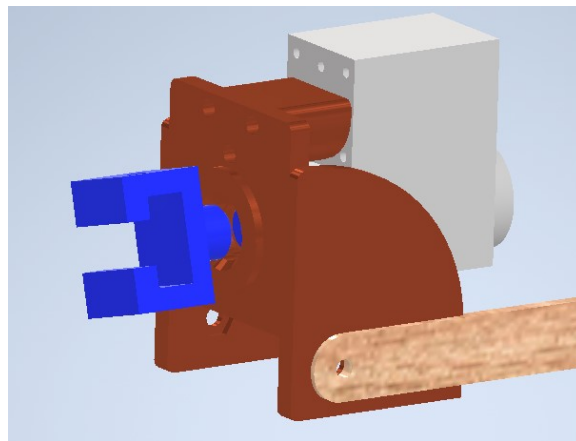


Figura 25: Quinto grado de libertad.

dañe el motor. Adicionalmente, esto ayuda a que el motor requiera un esfuerzo mínimo para poder hacer rotar el efector final. Solamente se debe aplicar el torque suficiente para mover el cojinete. Dado que se tiene una limitante de corriente de 3 A para cada cadena de motores, resulta beneficioso que esta junta no requiera muchos recursos.

7.5. Efector final

Como primera opción de efector final se fabricó un **gripper** lineal. Este consiste de dos mecanismos de cuatro barras manivela corredera simétricos para abrir y cerrar las pinzas. Como se observa en la Figura (26), se tiene dos eslabones rectos que conectan a los eslabones con dientes a la base del gripper. El motor se coloca en esta base y se tiene un agujero para poder realizar el movimiento rotacional.

En la Figura (27) se puede observar con más facilidad el mecanismo de funcionamiento. Se tienen dos engranajes con un pequeño eslabón saliendo de ellos. No se tienen dientes en todo el engranaje para establecer los límites de rotación del mecanismo. No se busca que se sigan cerrando las pinzas una vez entran en contacto los dientes y a su vez se establece un límite de qué tanto se abren las pinzas. La bancada del mecanismo consiste en el eslabón

que se forma entre los puntos de rotación que se encuentran en la base del **gripper**. Dado que los dos mecanismos son idénticos, solamente reflejado sobre el plano medio, se mueven de la misma manera en direcciones opuestas.

Los dientes están diseñados de tal manera que los de un lado encajen justo con los del otro y así llegar a un cierre total. Sin embargo, prácticamente nunca se buscará tener un cierre total porque no se cuenta con un sensor para saber cuándo dejar de cerrar el gripper. Por esto, se debe conocer el tamaño de la pieza que se desea agarrar de antemano para saber que tanto se debe mover el gripper para agarrar la pieza deseada.

7.6. Base del robot

Esta parte del robot no presenta movimiento, pero sirve como base del motor y posible posición para colocar la placa que controla a los motores. Se tiene una pieza que ayuda a asegurar al motor para el primer grado de libertad, pero también con facilidad de desensamblaje. Adicionalmente, se tiene una pieza que se coloca en la rueda del motor con la unión trapezoidal para poder unirse con el cojinete de la primera junta.

Los pilares que sostienen la plancha superior sirven para establecer la altura entre las planchas de material cortado con láser. La plancha superior sirve para colocar el cojinete de la primera junta mientras que la inferior sirve para colocar el motor, la placa de Arduino y la placa de Lynxmotion para el control de los motores.

7.7. Movimiento de los motores

Los motores deben tener un ID específico para poder tener un movimiento independiente de los demás. Como se puede observar en la Figura (29), aquí fue donde se determinó el ID de cada motor. Esta aplicación es la oficial de Lynxmotion y es la forma más sencilla de manipular a los motores. Sin embargo, no es de utilidad para la aplicación de la construcción

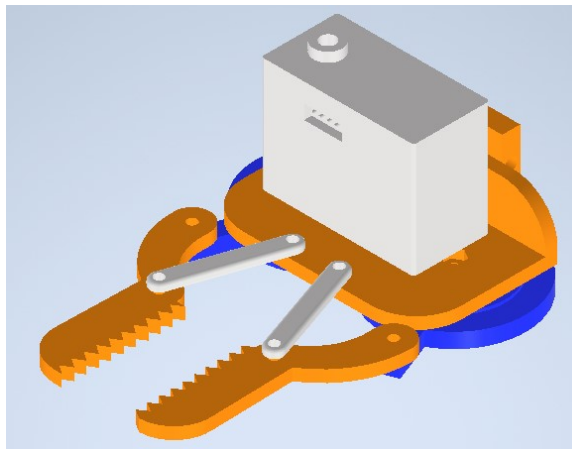


Figura 26: Efector final, vista isométrica.

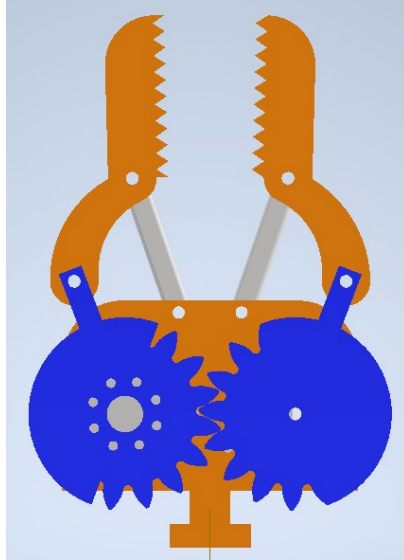


Figura 27: Efector final, vista inferior.

del brazo, ya que debe seguir la cinemática planteada. Es por esto que se utiliza un Arduino y la placa de Lynxmotion como un shield de este. Esto permitió controlar los motores por medio del Arduino. Este código debe recibir los comandos de movimiento del nodo de ROS y mover los motores de forma que representen el movimiento deseado.

Se tiene una librería en Arduino que permite controlar el movimiento de los motores tanto con posición angular como tiempo de rotación. Adicionalmente se puede observar la telemetría con información sobre la corriente, voltaje, temperatura, velocidad y posición de cada motor. Esto permitió ejecutar todas las acciones que se realizan desde la aplicación de Lynxmotion pero de una manera más dinámica y en el momento que se desea cada una de las acciones. El control sencillo de estos motores viene con las limitaciones que se debe trabajar alrededor de ellos en lugar de diseñar un sistema para ellos. Sin embargo, se logró una precisión de 1.5 grados como se presentará más adelante. La mayor complicación fue realizar el control de las juntas que tienen cajas reductoras. Esto es porque se manejan ángulos mayores a 360 grados y esto puede confundir a la telemetría. Se desarrolló un script básico en Arduino que sirve para configurar los motores y mandarles las posiciones deseadas. Este se también tiene implementado un sistema para leer los mensajes json mandados desde ROS y leer los valores de las juntas. Esto se puede observar en la Figura [30](#).

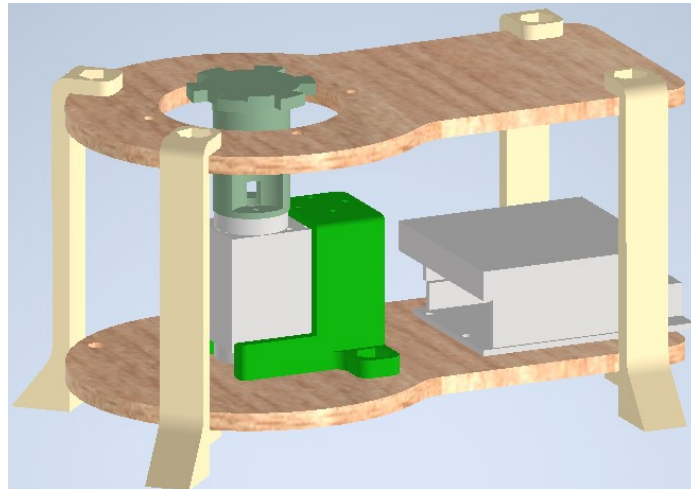


Figura 28: Base del robot.

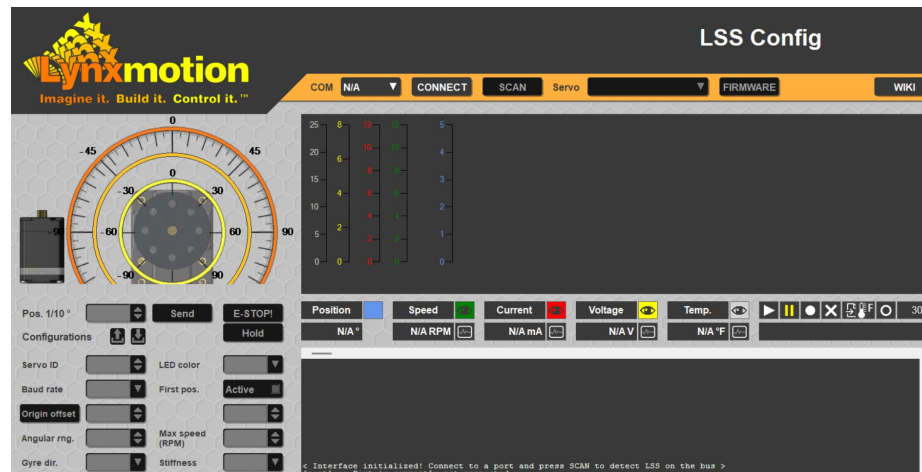


Figura 29: Aplicación Lynxmotion

```

if (Serial.available()>0) {
  lectura=Serial.readStringUntil('');
  lectura.concat('');

  DeserializationError error = deserializeJson(doc, lectura);
  q11 = float(doc["q1"]);
  lectura_CMDFun();
}

if (q11 > 0){
  digitalWrite(led, HIGH);
}
else {
  digitalWrite(led, LOW);
}

myLSS1.move(q11*10);

```

Figura 30: Código para leer datos y mandar el dato a la función de la librería Lynxmotion

8.1. Cinemática directa

Siguiendo las consideraciones y definiciones en la sección de cinemática directa del marco teórico, se obtienen los siguientes resultados.

En ${}^I T_B$ es donde se toma en consideración cómo está instalado el brazo con respecto al origen del Rover. Dado que se puede cambiar de posición, este es un parámetro ajustable. En este caso ${}^E T_F$ representa la transformación del gripper lineal. Esta es la distancia que existe entre la última junta hasta las pinzas del gripper. En la Ecuación (40) se puede observar la matriz de transformación, la cual representa que las pinzas del gripper agarran al objeto a 6 cm de la última junta del brazo. Dado que solamente se tienen cinco grados de libertad en el brazo, no se puede alcanzar cualquier pose en 3D porque no es redundante.

$${}^E T_F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.06 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (40)$$

Siguiendo los pasos de la sección de Denavit Hartenberg en el marco teórico, la matriz de Denavit Hartenberg para este proyecto se puede observar en el Cuadro (1). Se utilizó el Toolbox de Peter Corke en Matlab para simular este brazo. Para la cinemática directa, se utilizaron los valores del Cuadro (1) para armar el modelo del robot. Después de esto, se utilizó la función de teach para poder desplegar una representación 3d del brazo robótico. Como se puede observar en la Figura (31), se puede tener la representación del brazo en 3d utilizando la función plot con los ángulos encontrados en teach para llegar a cierta posición. Para este ejemplo, se utilizó el vector de configuración $[pi/2, -pi/4, pi/6, pi/3, -pi/2]$.

θ_j	d_j	a_j	α_j
q_1	118	0	$\pi/2$
q_2	0	-180	0
q_3	0	-80	0
$q_4 - \pi/2$	0	0	$\pi/2$
q_5	80	0	0

Cuadro 1: Matriz Denavit Hartenberg, medidas en milímetros.

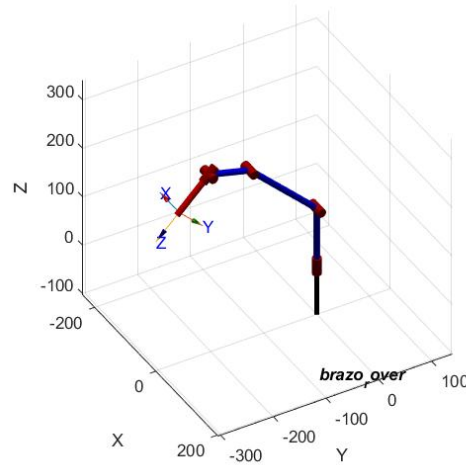


Figura 31: Cinemática directa con Matlab

8.2. Cinemática diferencial

Se deben seguir los pasos y consideraciones en la sección de cinemática diferencial del marco teórico. Con esta información del marco teórico, y la matriz obtenida en la sección de cinemática directa, se puede determinar el jacobiano del sistema.

Se vuelve a utilizar el Toolbox de Peter Corke en Matlab para poder visualizar los resultados. Se utiliza el mismo modelo de cinemática directa, se premultiplica por la transformación de base y se multiplica por la transformación del efector final. Con esto, se obtiene una matriz de transformación homogénea a partir de un vector de configuración. De esta matriz de transformación homogénea se extrae la matriz de rotación para obtener el jacobiano. El jacobiano correspondiente al vector de configuración $q_1 = [\pi/2, \pi/2, 0, 0, 0]$ se puede observar en la Figura (32).

```

>> r17J(q1)

ans =

    0    0.0000    0.7500    0.3750     0     0
  1.0000     0     0     0     0     0
    0     0   -0.0000   -0.0000     0     0
    0    0.0000   -0.0000   -0.0000   -0.0000   -0.0000
    0    0.0000    1.0000    1.0000    1.0000   -0.0000
    0   -1.0000    0.0000    0.0000    0.0000    1.0000

```

Figura 32: Jacobiano del vector de configuración q_1

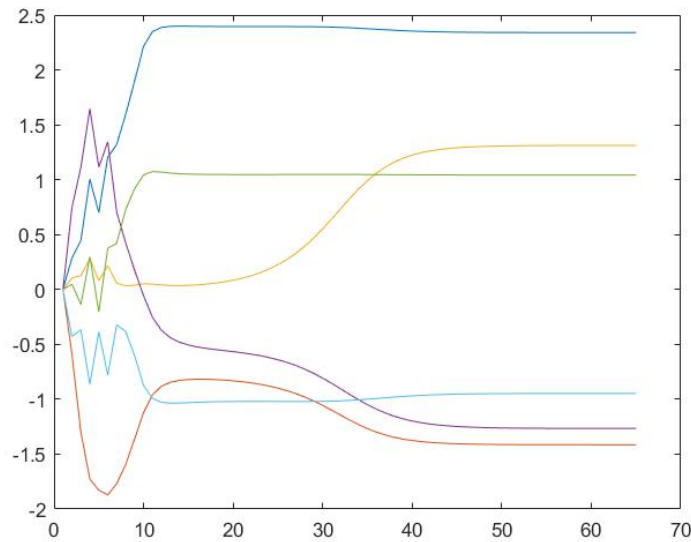


Figura 33: Histórico de las iteraciones del vector de configuración para llegar a la posición p_1 .

8.3. Cinemática inversa

Nuevamente se deben seguir los pasos y consideraciones de la sección de cinemática inversa del marco teórico.

Una vez más, se utiliza el Toolbox de Peter Corke en Matlab para realizar las pruebas necesarias. Ahora se tiene una función que tiene como entrada la pose deseada y la configuración inicial. Esta regresa la configuración deseada y el histórico de cómo fue cambiando conforme al algoritmo. Se utiliza el error de posición para determinar las veces que se debe iterar el algoritmo. Se utiliza el método de la pseudo-inversa para calcular la aproximación de la inversa del jacobiano y así ir actualizando la solución. En la Figura (33), se observa el histórico del vector de configuración para llegar a la posición $p_1 = (0.5, -0.3, 0.6)$.

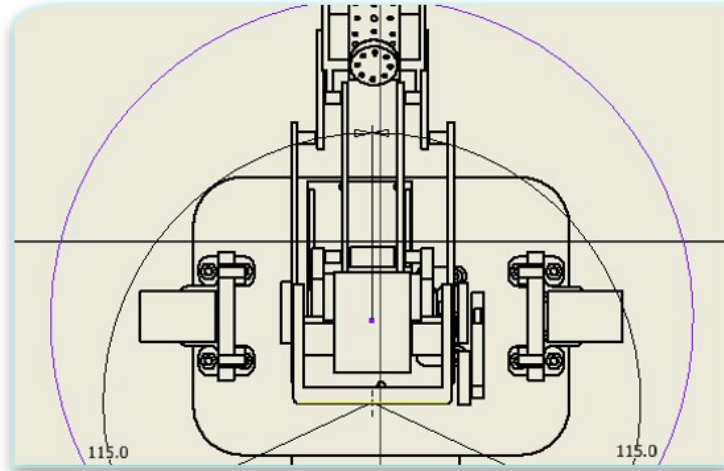


Figura 34: Límites de la junta 1

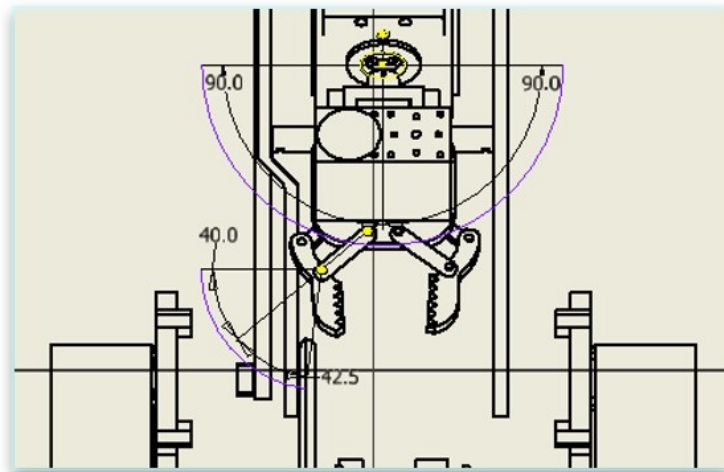


Figura 35: Límites de la junta 5 y el gripper

8.4. Límites en las juntas

Se realizaron bosquejos con los límites de cada una de las juntas después de probar los movimientos utilizando el código de Arduino. El límite de las juntas uno y cinco solamente se estableció por comodidad. Para la segunda y tercer junta se tienen límites debido a el diseño de la estructura. Para la cuarta junta se tienen límites debido al mecanismo de 4 barras, evitando que el mecanismo espejo se cambie de configuración. Estos límites se pueden observar en las Figuras (34), (36) y (35).

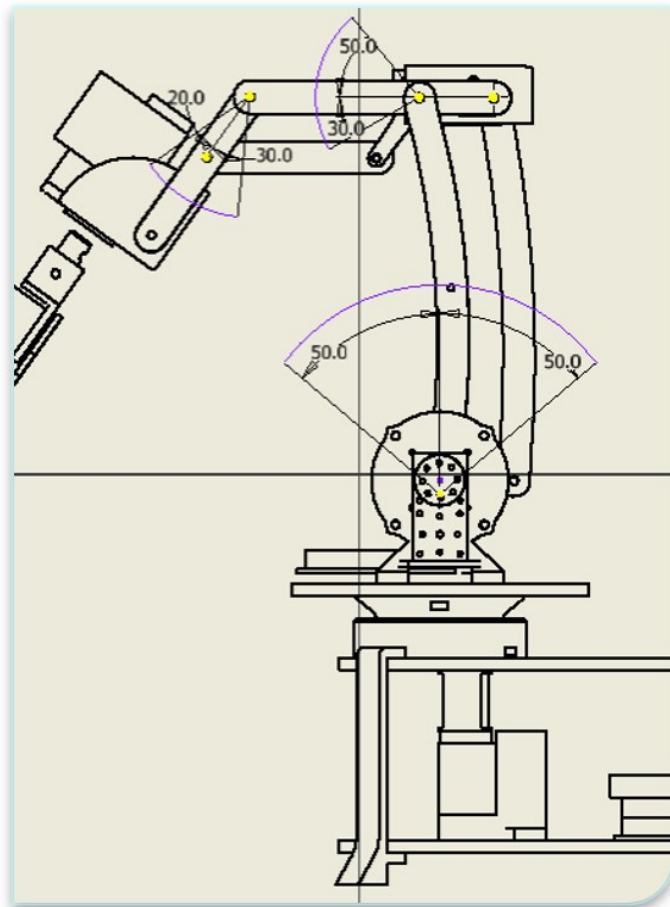


Figura 36: Límites de las juntas 2, 3 y 4

Simulación y control en ROS

Para poder comenzar con los trabajos de ROS, primero se tuvo que instalar una máquina virtual para poder utilizar Ubuntu 20.04. Una vez se tuvo esto, se procedió a descargar ROS2 Foxy Fitzroy. Se decidió utilizar esta versión de ROS porque tiene soporte a largo plazo y aumenta la probabilidad que siga siendo compatible en iteraciones posteriores del Rover. Seguido de esto, se creó un espacio de trabajo para poder almacenar los paquetes específicos del brazo. Se siguieron los tutoriales de la página oficial de ROS2 Foxy para aprender a utilizar los nodos, tópicos, servicios y acciones de una forma básica.

Después de esto, se configuró un archivo launch. Este sirve para especificar el paquete donde se puede buscar el modelo URDF y las configuraciones de RVIZ para inicializar la simulación. Esta parte del código se puede observar en la Figura (37), donde se especifica que se debe buscar el archivo *brazo2-urdf* como el modelo y el archivo *urdf-brazo-config.rviz* para inicializar RVIZ. Después de esto, se deben declarar los argumentos para cada uno de los paquetes a utilizar. Los argumentos utilizados se pueden observar en la Figura (38). En este caso, todos los paquetes se inicializan con los valores estándar.

En este archivo también se inicializa la interfaz gráfica con la finalidad de poder modificar los valores de las juntas en tiempo real durante la simulación. Los nodos utilizados se pueden observar en la Figura (39). En esta parte se inicializa un nodo para robot state publisher, uno para joint state publisher y uno para joint state publisher gui. Cada uno se inicializa con el ejecutable por defecto dado que estos paquetes ya están creados y se buscan utilizar sus funciones básicas. Finalmente, en la Figura (40), se puede observar que se inicializan acciones para cada uno de los nodos. Esto se realiza con el objetivo de fijar objetivos y establecer comunicación los nodos hasta cumplir los objetivos establecidos. Se realizan las actualizaciones necesarias tanto en la simulación en RVIZ como en los datos de los tópicos mencionados anteriormente.

Una vez se tiene el archivo de launch configurado correctamente, se debe asegurar que

```

pkg_share = FindPackageShare(package='basic_mobile_robot').find('basic_mobile_robot')
default_launch_dir = os.path.join(pkg_share, 'launch')
default_urdf_model_path = os.path.join(pkg_share, 'models/brazo_v2.urdf')
robot_name_in_urdf = 'basic_mobile_bot'
default_rviz_config_path = os.path.join(pkg_share, 'rviz/urdf_brazo_config.rviz')

# Launch configuration variables specific to simulation
urdf_model = LaunchConfiguration('urdf_model')
rviz_config_file = LaunchConfiguration('rviz_config_file')
use_robot_state_pub = LaunchConfiguration('use_robot_state_pub')
use_rviz = LaunchConfiguration('use_rviz')
use_sim_time = LaunchConfiguration('use_sim_time')

```

Figura 37: Configuración para inicializar simulación en launch.

```

declare_urdf_model_path_cmd = DeclareLaunchArgument(
    name='urdf_model',
    default_value=default_urdf_model_path,
    description='Absolute path to robot urdf file')

declare_rviz_config_file_cmd = DeclareLaunchArgument(
    name='rviz_config_file',
    default_value=default_rviz_config_path,
    description='Full path to the RVIZ config file to use')

declare_use_joint_state_publisher_cmd = DeclareLaunchArgument(
    name='gui',
    default_value='True',
    description='Flag to enable joint_state_publisher_gui')

declare_use_robot_state_pub_cmd = DeclareLaunchArgument(
    name='use_robot_state_pub',
    default_value='True',
    description='Whether to start the robot state publisher')

declare_use_rviz_cmd = DeclareLaunchArgument(
    name='use_rviz',
    default_value='True',
    description='Whether to start RVIZ')

declare_use_sim_time_cmd = DeclareLaunchArgument(
    name='use_sim_time',
    default_value='false',
    description='Use simulation (Gazebo) clock if true')

```

Figura 38: Argumentos de inicialización en launch.

el archivo del URDF esté configurado de forma correcta. En este caso, se utilizó xacro para poder simplificar el código. Esta es una herramienta que permite utilizar macros y realizar parametrización de variables. Para las geometrías, se utilizaron los archivos stl generados por *Autodesk Inventor* en la etapa de modelado. Esto sirve para darle una figura que represente de forma correcta el modelo real del robot y que se calculen tanto la cinemática como colisiones de manera correcta. Para las colisiones no se utilizaron los meshes, sino geometrías rectangulares con dimensiones un poco más grandes de los archivos stl como se puede observar en la Figura (42). Esto se realizó con el objetivo de simplificar los cálculos del algoritmo de identificación de colisiones y que la simulación sea más fluida. Para realizar todas las juntas, se utilizó la función de junta continua y para cada una se establecieron sus límites de posición, velocidad y esfuerzo específicos. Para que se haga una revolución, se deja vacío el vector xyz y se determina un eje para que gire alrededor de él. Un ejemplo de esto se puede observar en la Figura (41).

Se puede apreciar el modelo URDF completo en RVIZ en la Figura (43). Adicionalmente, se puede observar las relaciones padre-hijo entre las juntas en la Figura (45).

```

start_robot_state_publisher_cmd = Node(
    condition=IfCondition(use_robot_state_pub),
    package='robot_state_publisher',
    executable='robot_state_publisher',
    parameters=[{'use_sim_time': use_sim_time,
                  'robot_description': Command(['xacro ', urdf_model])}],
    arguments=[default_urdf_model_path])

start_joint_state_publisher_cmd = Node(
    condition=UnlessCondition(gui),
    package='joint_state_publisher',
    executable='joint_state_publisher',
    name = 'joint_state_publisher',
    parameters=[{'use_sim_time': use_sim_time,
                  'robot_description': Command(['xacro ', urdf_model])}],
    arguments=[default_urdf_model_path])

start_joint_state_publisher_gui_node = Node(
    condition=IfCondition(gui),
    package='joint_state_publisher_gui',
    executable='joint_state_publisher_gui',
    name = 'joint_state_publisher_gui',
    parameters=[{'use_sim_time': use_sim_time,
                  'robot_description': Command(['xacro ', urdf_model])}],
    arguments=[default_urdf_model_path])

# Launch RViz
start_rviz_cmd = Node(
    condition=IfCondition(use_rviz),
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    output='screen',
    arguments=['-d', rviz_config_file])

```

Figura 39: Inicialización nodos en launch.

Para el archivo de inicialización de RVIZ, no se deben configurar nodos sino configurar las ventanas y displays que se pueden observar al momento de inicializar el archivo de launch. Esto no es completamente necesario, pero es bastante cómodo no tener que modificar ciertos parámetros cada vez que se inicializa RVIZ. Para este caso, se debe indicar que se utilizarán los plugins RobotModel, TF y Grid para poder realizar la simulación y visualización del modelo URDF. Además de esto, se van a poder visualizar sus marcos de referencia y así verificar que están hechos de forma correcta. Finalmente se deben inicializar los plugins de defecto para poder interactuar con la cámara e interactuar con el modelo.

Una vez que se tuvieron todos estos archivos listos, se copiaron del paquete original al que contiene todos los archivos de la simulación del Rover. Esto se realizó para comenzar el proceso de unificación de los trabajos y que en el futuro puedan funcionar en conjunto.

```

# Declare the launch options
ld.add_action(declare_urdf_model_path_cmd)
ld.add_action(declare_rviz_config_file_cmd)
ld.add_action(declare_use_joint_state_publisher_cmd)
ld.add_action(declare_use_robot_state_pub_cmd)
ld.add_action(declare_use_rviz_cmd)
ld.add_action(declare_use_sim_time_cmd)

# Add any actions
ld.add_action(start_joint_state_publisher_cmd)
ld.add_action(start_joint_state_publisher_gui_node)
ld.add_action(start_robot_state_publisher_cmd)
ld.add_action(start_rviz_cmd)

```

Figura 40: Inicialización acciones en launch.

```

<link name="upper_base">
  <visual>
    <origin xyz="0.0 0.0 121.0" rpy="{pi/2} {pi} {pi/2}"/>
    <geometry>
      <mesh filename="package://basic_mobile_robot/meshes/base_superior.stl" />
    </geometry>

    <material name="white">
      <color rgba="1.0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <box size="{base_width} {base_length} {base_height}"/>
    </geometry>
  </collision>

</link>

<joint name="base_to_upper_base" type="continuous">
  <axis rpy="0 0 0" xyz="0 0 1.0"/>
  <parent link="base_link"/>
  <child link="upper_base"/>
  <origin rpy="0 0 0" xyz="0 0 0"/>
</joint>

```

Figura 41: Ejemplo de junta y eslabón del URDF.

Aunque se encuentran en el mismo paquete, los archivos de launch y de inicialización son diferentes al de la simulación del Rover.

Finalmente, se debe escribir un nodo para que comunique la información de las juntas a el Arduino por medio de comunicación serial. Para poder comenzar a analizar este problema, primero se realizó una gráfica de los nodos presentes utilizando rqt graph. Esto se puede observar en la Figura (44). En este gráfico se muestra que tanto RVIZ como joint state publisher gui obtienen la información de las juntas de robot description. Este a su vez recibe la información de robot state publisher. Robot state publisher es un paquete encargado de utilizar la información del URDF en conjunto con la información de las posiciones de las juntas para generar marcos de referencia y así poder tener una simulación. Es por esta razón que el tópico que tiene la información de las juntas es joint states. Por lo tanto, se debe buscar realizar un nodo que sea un subscriptor a este tópico para leer los valores de las juntas, almacenarlos y mandarlos por medio de comunicación serial en formato json al Arduino.

Se basó en la estructura del nodo básico de los tutoriales de la página oficial mencionados anteriormente con una comunicación serial adicional con el Arduino por medio del puerto ttyACM0. Sin embargo, para que un nodo pueda suscribirse a un tópico, se debe tener el mismo nombre, tipo de variable y tamaño de la variable. Utilizando comandos como topic list y topic echo se determinó que el mensaje es de tipo JointState. Una vez se determinó el tipo de mensaje, se realizó una función de callback. En esta se realizó un ciclo for en donde se recorre el mensaje buscando que coincida cada uno de los nombres de las juntas con el parámetro de nombre en el mensaje leído. Una vez se encuentra cada una de las juntas, se van almacenando en un diccionario. Este finalmente es enviado en formato *json* y codificación *ascii*. Para conectarse con el Arduino, se realizó una conexión con un baudaje de 115200, sin paridad, con un stop bit y un tamaño de 8 bits. La lógica utilizada se puede observar en la Figura (46).

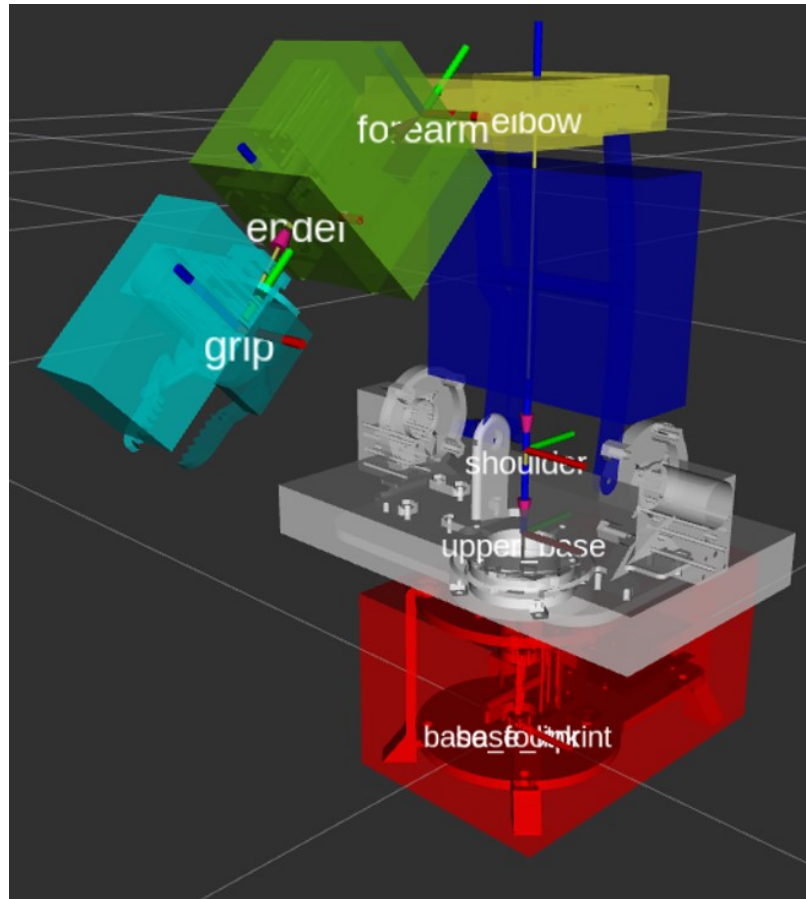


Figura 42: Cajas de colisiones en RVIZ.

No obstante, después de cierto tiempo se corta la comunicación entre ROS y el Arduino. Se intentó limitar el envío de datos cada 3 veces que ingresara en la función callback, modificar el timeout, realizar interrupciones para llamar a las funciones de los motores en el Arduino, realizar un flush del buffer del Arduino, cambio de baudrates. También se consideró resetear el nodo cada cierto tiempo, pero no se obtuvieron los resultados esperados. Los intentos de solución lograron que la duración de la conexión aumentara de 1 minuto a 5 minutos. Estos cambios permitieron realizar las pruebas necesarias, pero hay espacio para mejoras.

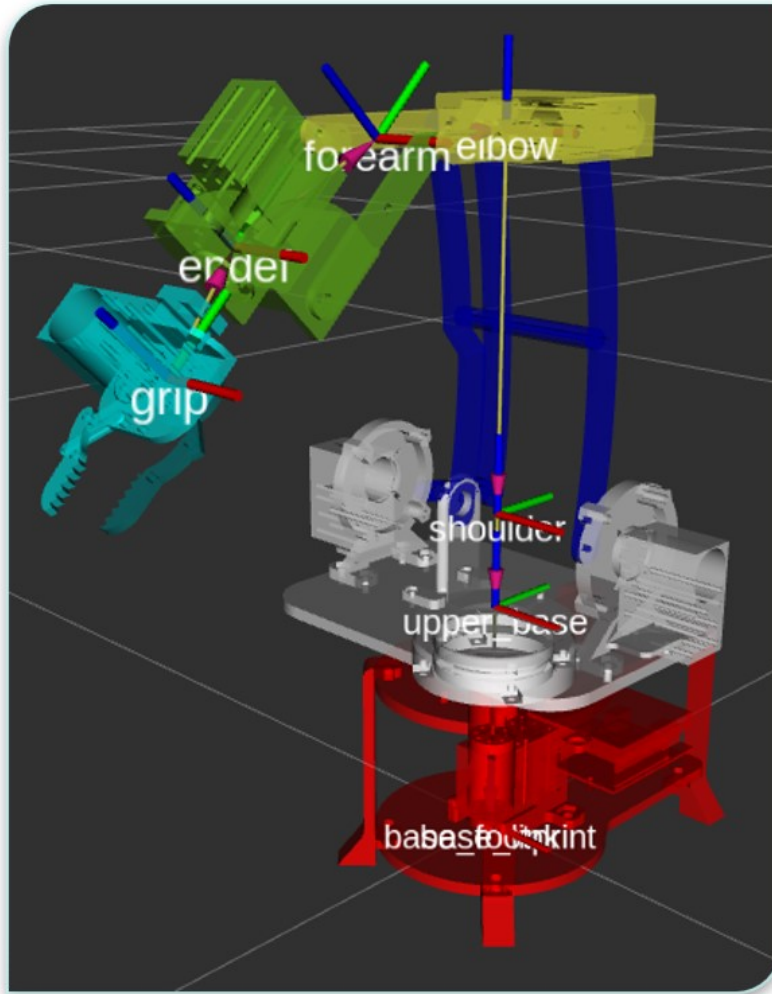


Figura 43: Visualización del modelo URDF en RVIZ.

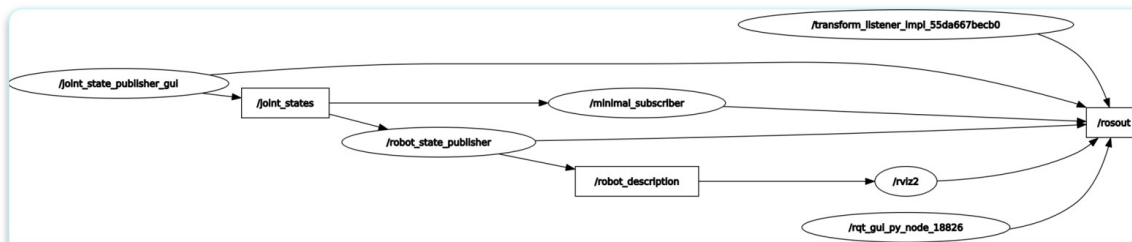


Figura 44: Nodos presentes en la simulación.

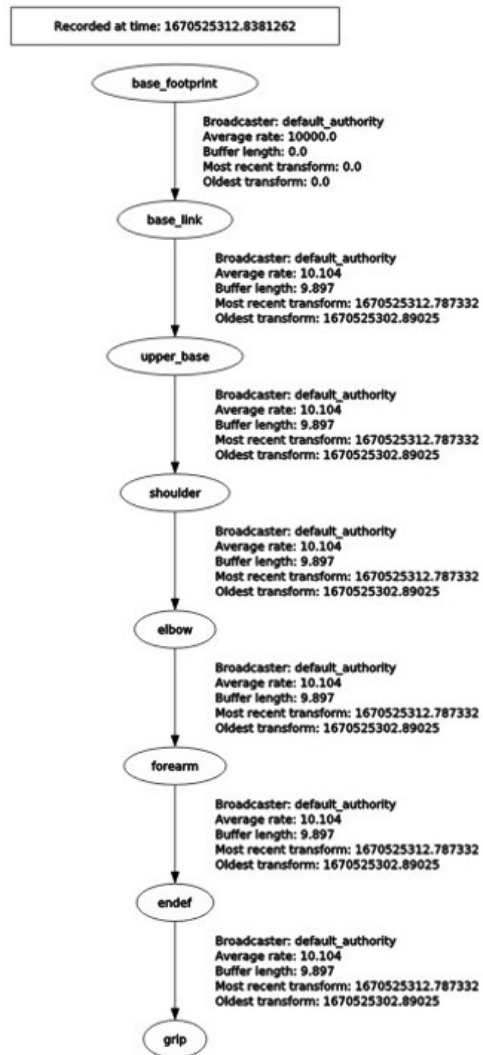


Figura 45: Relaciones padre-hijo entre las juntas.

```

def joint_state_callback(self, msg):
    for i,name in enumerate(msg.name):
        if name == "base_to_upper_base":
            pos1 = msg.position[i]
            print(pos1*180.0/(math.pi))
    data = {}
    data["q1"] = pos1*180.0/(math.pi)
    data=json.dumps(data)
    ser.write(data.encode('ascii'))
  
```

Figura 46: Almacenamiento de valores de juntas en nodo suscriptor.

10.1. Diseño mecánico

Se logró implementar mecanismos de 4 barras paralelos para mover el hombro y el codo del brazo robótico. Se logró implementar un cojinete de balines para la primera junta, permitiendo rotación sobre su eje central, minimizando la fricción. Se implementaron cajas reductoras armónicas con reducción 20:1 en las juntas 2 y 3 del hombro. Se decidió no utilizar un contrapeso dado que se desconoce la carga en el efector final y significaba utilizar espacio adicional que limitaba aún más el movimiento de las juntas 2,3 y 4. Se implementó un gripper lineal intercambiable con un mecanismo manivela corredera.

Respecto al movimiento de los motores, se encontró que las velocidades que proporcionaban más fluidez de movimiento fueron 10 rpm para las juntas sin reducción y 30 rpm para las juntas con reducción.

Los principales problemas con el sistema mecánico fueron los límites de las juntas y la implementación de las cajas reductoras. Los límites de las juntas se trabajaron modificando el diseño mecánico del brazo hasta llegar a un espacio de trabajo que permitiera llegar a las posiciones necesarias para agarrar objetos y almacenarlos. Por el otro lado, la implementación de las cajas se logró implementar de manera útil y sin ocupar más espacio de lo necesario. El problema recurrente es la posibilidad que los dientes de las cajas lleguen a la fractura. Se debe considerar que por la reducción, los dientes de entrada deben moverse 20 veces más. Es por esto que el brazo puede llegar a fallar antes de llegar a su peso máximo teórico. Después de realizar iteraciones, se logró minimizar estas ocurrencias. Sin embargo, se necesita seguir trabajando en conjunto con la tesis de cajas reductoras impresas en 3D para llegar a un modelo que no sufra de este problema. Un ejemplo de ruptura se puede observar en la Figura (47).



Figura 47: Posible ruptura de las cajas.

10.2. Ganancia de torque

En las Figuras (48) y (49), se observan las ganancias de torque, en kg-cm, que se obtuvieron al utilizar los mecanismos de cuatro barras. Estos se calcularon al identificar la diferencia entre el torque producido en su posición original y su posición final utilizada. A este dato se le debe restar el torque producido por los tornillos que se tuvieron que utilizar para realizar los mecanismos. Esto se debe realizar porque estos no estarían presentes si no se hubieran implementado los mecanismos de cuatro barras. Aunque estos no presentan un mayor torque en comparación a los pesos de los motores, en conjunto son necesarios para calcular la verdadera ganancia del sistema utilizado. Las ganancias están alrededor de los 0.73 kg-cm, lo cual es una diferencia significativa para poder agarrar objetos más pesados con el brazo.

Un aspecto importante a considerar, es que en la junta 4 pareciera que se tienen pérdidas en el torque de -0.038 kg-cm. Esta pérdida es mínima, sin embargo, no presenta beneficios sobre haber puesto el motor en su posición original. Este suceso se debe a que en la configuración final se utilizaron longitudes efectivas de 80 mm para ambos eslabones que influyen en esta junta. Estas distancias son aproximaciones de las que se habían calculado para cruzar el punto de ganancia (87 mm). El tiempo limitado no permitió seguir experimentando con las longitudes. Cualquier valor mayor a los 87 mm presentaría ganancias en el torque. Teóricamente se pueden tener ambos eslabones con una distancia de 120 mm sin sobrepasar los límites del motor. Sin embargo, se deben realizar pruebas para verificar con los materiales utilizados. Las gráficas utilizadas para calcular la ganancia de torque aproximada, se pueden observar en las Figuras (50) y (51).

Se puede apreciar que el mecanismo para la junta 3 resulta efectivo y sencillo de implementar, necesitando solamente un eslabón de 17 mm para obtener ganancias. Por el otro lado, el sistema utilizado en la junta 4 resulta efectivo para realizar su objetivo, pero no se tiene la misma magnitud de ganancia que se tiene con el primer mecanismo.

Comparación segundo grado de libertad			
Motor en su posición normal			
Pieza	Peso (gramos)	Posición (milímetros)	Torque (kg cm)
Motor LSS ST1	58	180	1.044
Motor en posición más cercana			
Pieza	Peso (gramos)	Posición (milímetros)	Torque (kg cm)
tornillo M4, 20 cm	2	35	0.007
tuerca seguridad M4	0.8	35	0.0028
roldana M4 peq X3	0.6	35	0.0021
roldana M4 gra X3	1.2	35	0.0042
eslabón 35 cm	2	17.5	0.0035
eslabón 180 cm	15	125	0.1875
tornillo M4, 16 cm	2	215	0.043
tuerca M4	0.33	215	0.007095
roldana M4 gra X2	0.8	215	0.0172
total			0.274
GANANCIA			0.770

Figura 48: Ganancia de torque junta 3, en kg-cm.

Motor en posición más cercana			
Pieza	Peso (gramos)	Posición (milímetros)	Torque (kg cm)
Motor LSS ST1	58	180	1.044
Eslabones a motor X2	3	197.5	0.05925
Uniones de eje	0.2	215	0.0043
roldanas peq X6	1.2	215	0.0258
tornillo m4, 20 cm X2	4	215	0.086
tuercas seguridad M4 X2	1.4	215	0.0301
eslabon 80 cm X2	4	255	0.102
tornillo m4, 20 cm X2	4	295	0.118
roldanas peq X6	1.2	295	0.0354
tuercas seguridad M4 X2	1.4	295	0.0413
total			1.546
GANANCIA			-0.038
GANANCIA TOTAL			0.73

Figura 49: Ganancia de torque junta 4, en kg-cm.

10.3. Repetibilidad

Después de esto se hicieron varias pruebas para medir la repetibilidad del brazo en diferentes configuraciones. Se escogieron 3 configuraciones para medir la repetibilidad de los movimientos en cada junta. Siempre se tomó el ángulo 0 como uno de los puntos para medir la repetibilidad y después se eligieron ángulos acorde a el rango de cada junta. Se tomó el mismo valor para el ángulo positivo y el negativo. Para medir el ángulo según los motores, se utilizó la telemetría incorporada con los motores LSS ST1. En la Figura (52), se puede observar el procedimiento para medir los ángulos utilizando *Tracker*. Estos valores son los que se consideraron como los ángulos reales. Se llegó a esta posición 5 veces y se compararon los valores de la telemetría con los obtenidos por *Tracker* para obtener desviación estándar y el promedio de la diferencia del ángulo obtenido con el real. Se encontró que el mayor error promedio fue de 1.2 grados. Con esto se prueba que la precisión de la configuración actual es

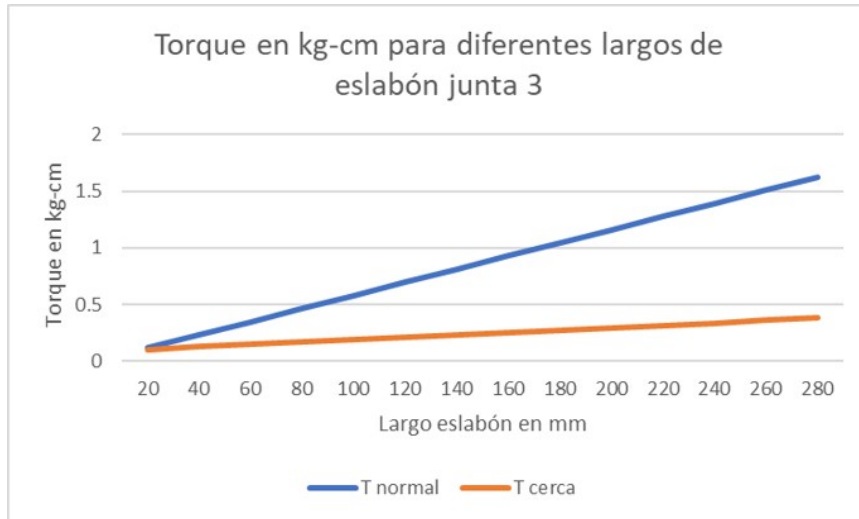


Figura 50: Gráfica de ganancia de torque junta 3, en kg-cm en función del largo del eslabón.

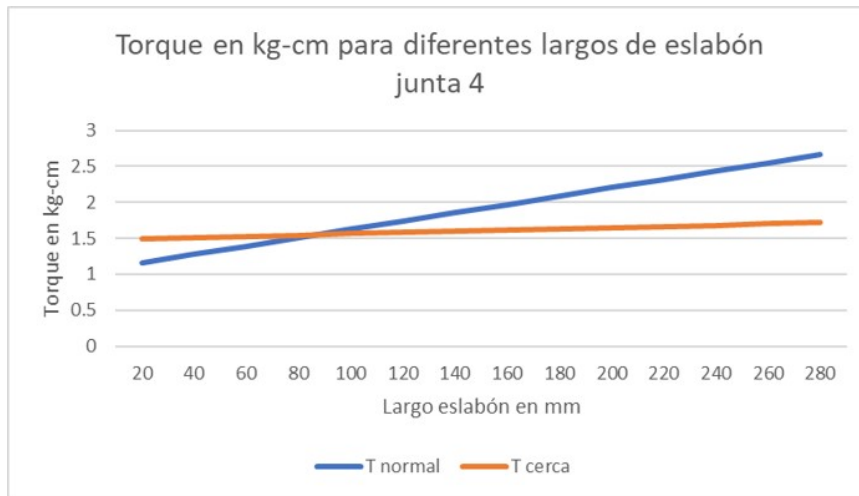


Figura 51: Gráfica de ganancia de torque junta 4, en kg-cm en función del largo del eslabón.

de alrededor de 1.2 grados para cada junta, lo cual es un resultado aceptable, pero se puede minimizar con un sistema de control. Este lazo debería tomar los datos reales de los motores y compararlos con los datos que está tirando el simulador y no avanzar hasta que se alcance cada objetivo. Movimientos bruscos pueden desestabilizar el brazo y así aumentar el error, en cambio con un lazo de control y teniendo varios puntos para una trayectoria, se tendrá un movimiento más fluido. El menor error se alcanza en las juntas que tienen reducción, dado que el ángulo de salida se convierte 20 veces más preciso. Todos los valores de promedio de error y desviación estándar se pueden observar de la Figura (53) a (62).

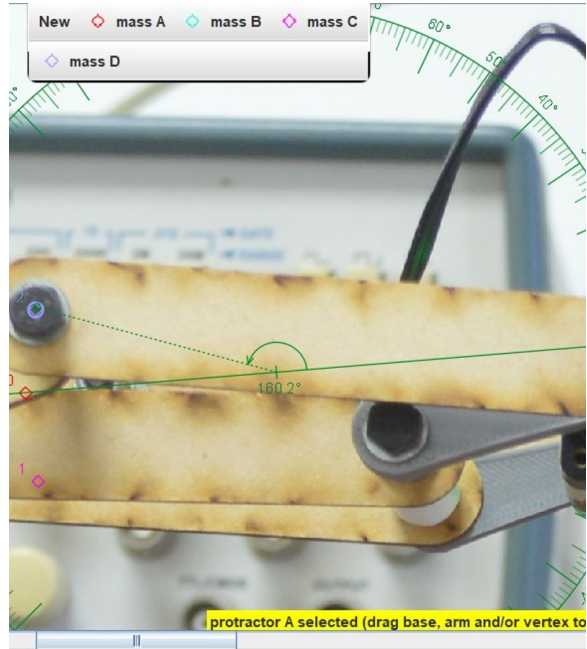


Figura 52: Ejemplo de medición con Tracker para obtener los ángulos reales.

grado 1	NOTA: EL JUEGO PUEDE LLEGAR HASTA 5 GRADOS CON LA PIEZA ACTUAL						
real	-30	30	0		30	-30	0
obtenido	-29.9	30.5	0		33.7	25.7	2.1
	-30.1	30.5	0.3		34.2	28.4	-1.6
	-30.1	30.7	0.6		28.4	33.6	-3.1
	-30	30.7	0.4		27.6	32.8	3.4
f	-30.1	30.4	0.4		33.5	27.2	2.8
				prom			
	-30.04	30.56	0.34	0.313			
	0.089443	0.134164	0.219089	0.148			

Figura 53: Valores para los ángulos medidos y los reales para la junta 1.

Mean	31.48	29.54	0.72	0.887 grados
Standard Error	1.430874	1.559359	1.292053	
Median	33.5	28.4	2.1	
Mode	#N/A	#N/A	#N/A	
Standard Deviation	3.199531	3.486832	2.889118	3.192 grados

Figura 54: Promedio de la diferencia de ángulos y desviación estándar para la junta 1.

real	-20	20	0		20	-20	0
obtenido	-20	20.01	-0.3		20.1	-19.9	-0.2
	-19.98	20	0.3		20	-19.8	0.2
	-20.01	20	0.2		20.1	-20.1	0.1
	-20.05	20.02	0.3		20.3	-20.2	0.2
f	-20.03	20.01	0.2		20.1	-20.1	0.1
				prom			
	-20.014	20.008	0.14	0.054			
	0.027019	0.008367	0.250998	0.095			

Figura 55: Valores para los ángulos medidos y los reales para las juntas 2 y 3.

Mean	20.12	-20.02	0.08	0.073 grados
Standard Error	0.04899	0.073485	0.073485	
Median	20.1	-20.1	0.1	
Mode	20.1	-20.1	0.2	
Standard Deviation	0.109545	0.164317	0.164317	0.146 grados

Figura 56: Promedio de la diferencia de ángulos y desviación estándar para las juntas 2 y 3.

grado 4							
real	30	-30	0		30	-30	0
obtenido	30.5	-29.2	0.3		31.3	-28.5	0.6
	30.5	-29.1	0.8		31.2	-28.4	1.1
	30.5	-29.2	0.7		31.1	-28.6	0.9
	30.3	-29.6	0.7		30.8	-28.2	0.8
f	30.5	-29.2	0.6		31.3	-28.7	0.6
				prom			
	30.46	-29.26	0.62	0.607			
	0.089443	0.194936	0.192354	0.159			

Figura 57: Valores para los ángulos medidos y los reales para la junta 4.

Mean	31.14	-28.48	0.8		1.153	grados
Standard Error	0.092736	0.086023	0.094868			
Median	31.2	-28.5	0.8			
Mode	31.3	#N/A	0.6			
Standard Deviation	0.207364	0.192354	0.212132		0.204	grados

Figura 58: Promedio de la diferencia de ángulos y desviación estándar para la junta 4.

grado 5	SOLAMENTE CON FEEDBACK				POST TRACKER			
	real	90	-90	0		90	-90	0
obtenido	90.3	-90.2	0.3		90.5	-90.5	1.3	
	90.4	-90.4	0.3		90.8	-91.1	1.2	
	90.4	-90.4	0.3		91	-91.2	0.6	
	90.4	-90.2	0.2		90.9	-90.4	0.4	
f	90.5	-90.3	0.2		91.2	-90.6	0.4	
				prom				
	90.4	-90.3	0.26	0.32				
	0.070711	0.1	0.054772	0.075				

Figura 59: Valores para los ángulos medidos y los reales para la junta 5.

Mean	90.88	-90.76	0.78		0.807	grados
Standard Error	0.115758	0.163095	0.195959			
Median	90.9	-90.6	0.6			
Mode	#N/A	#N/A	0.4			
Standard Deviation	0.258844	0.364692	0.438178		0.354	grados

Figura 60: Promedio de la diferencia de ángulos y desviación estándar para la junta 5.

grip	NOTA: SE TOMA EL LADO ACTUADO POR EL MOTOR							
	real	40	-40	0		40	-40	0
obtenido	40	-39.7	0.1		40.5	-39.5	0.1	
	40	-39.9	0.1		40.4	-39.6	0.2	
	40	-39.9	0.1		40.4	-39.7	0.2	
	40.1	-39.9	0.1		40.5	-39.7	0.3	
f	40	-39.9	0.6		40.4	-39.5	0.9	
				prom				
	40.02	-39.86	0.2	0.12				
	0.044721	0.089443	0.223607	0.119				

Figura 61: Valores para los ángulos medidos y los reales para el gripper.

Mean	40.44	-39.6	0.34		0.393	grados
Standard Error	0.024495	0.044721	0.143527			
Median	40.4	-39.6	0.2			
Mode	40.4	-39.5	0.2			
Standard Deviation	0.054772	0.1	0.320936		0.159	grados

Figura 62: Promedio de la diferencia de ángulos y desviación estándar para el gripper.

- Se logró construir un brazo robótico con técnicas de manufactura rápida y los motores Lynxmotion, con un alcance de 40 cm.
- Por medio de un nodo de ROS se logró realizar la comunicación entre *joint states* y los motores controlados por *Arduino*, mandando las posiciones de cada junta. Sin embargo, hay que actualizarlo para versiones posteriores de ROS.
- Se utilizaron mecanismos de cuatro barras paralelos para disminuir el torque del peso de los motores y se diseñó un cojiente axial para soportar la carga del brazo, obteniendo una rotación fluida.
- Se logró plantear un modelo URDF para simular el brazo robótico en ROS, considerando límites y colisiones.
- Se logró ahorrar 0.73 kg-cm al utilizar los mecanismos.
- Se alcanzó una precisión mayor a los 1.5 grados, con una desviación estándar menor a los 3.5 grados para la iteración presentada.
- Se logró soportar una carga máxima de 90 g al respetar los límites del proveedor y hasta 300 g al llevar a los motores al límite.

- Se recomienda realizar pruebas con contrapesos para mejorar las dimensiones del brazo y las cargas que puede aguantar. También se puede experimentar con compensación por gravedad.
- Se recomienda agregar penalización en la cinemática inversa para priorizar movimientos en las primeras juntas.
- Se recomienda realizar control al devolver las lecturas de los grados de las juntas reales.
- Se recomienda implementar generación de trayectorias en ROS con MoveIt e implementar colisiones propuestas.
- Se sugiere experimentar con más mecanismos de transmisión de potencia para minimizar las restricciones de las juntas. Sin embargo, se recomienda conservar el método de 4 barras para mover la junta 3.
- Se sugiere utilizar sistema de hilos donde los motores no están sobre el robot, disminuyendo el torque necesario para levantar los motores en su totalidad. (*Bowden Cables*)
- Se recomienda realizar un cambio de motores por unos de mayor torque o utilizar cajas de mayor capacidad para las juntas 2, 3 y 4. Se pueden utilizar motores stepper y diseñar reducciones de velocidad para aumentar su torque. De esta manera se pueden obtener torques mucho mayores y así lograr construir un brazo más grande y con más capacidad de carga sacrificando la velocidad de movimiento.
- Se recomienda mandar a fabricar fajas dentadas del largo necesario una vez se determine el largo ideal del brazo para utilizarlo con el Rover.
- Se recomienda agregar diferentes efectores finales para que el brazo pueda realizar diferentes tareas en el Rover.

-
- [1] J. E. A. Murillo, “Diseño e implementación de capacidades automáticas de navegación para un Robot Explorador Modular,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.
 - [2] H. J. S. Pinto, “Diseño Mecánico, Selección de Motores e Implementación de Sensores para un Robot Explorador Modular,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.
 - [3] H. D. C. Calderón, “Diseño mecánico de un brazo robótico para la automatización de un sistema VarioGuide-Brainlab para realizar cirugías estereotácticas.,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.
 - [4] M. A. Izeppi, “Aplicación de Herramientas de Aprendizaje Reforzado y Aprendizaje Profundo en Simulaciones de Robótica de Enjambre con Restricciones Físicas,” Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.
 - [5] J. J. León Pérez, “Extendiendo las capacidades del robot RESCUER en ROS: planificación, navegación y simulación.,” 2015.
 - [6] V. H. Gómez Tejada, “Operación de remachado mediante robot manipulador industrial basado en ROS,” 2015.
 - [7] J. M. Gómez Cuadrado et al., “Integración del brazo robot IRB120 en entorno ROS-MATLAB,” 2017.
 - [8] C. Franciscone, 2018. dirección: http://www.eezyrobots.it/eba_mk2.html.
 - [9] R. L. Norton y M. P. González, “El proceso de diseño,” en *Diseño de maquinaria: Síntesis y análisis de máquinas y mecanismos*, 5.^a ed. McGraw-Hill, Interamericana de España, 2013, págs. 6-12, 41-45.
 - [10] J. E. Shigley, C. R. Mischke, F. P. Bocanegra y C. C. Osornio, “Fases e interacciones del proceso de diseño,” en *Diseño en ingeniería mecánica*, 10.^a ed. McGraw-Hill Interamericana de México, 2018, págs. 5-8.
 - [11] B. Siciliano, G. Oriolo, L. Villani y L. Sciavicco, *Robotics: Modelling, planning and control*, 1.^a ed. Springer, 2010.

- [12] K. Lynch y F. C. Park, *Modern Robotics: Mechanics, planning, and Control*, 1.^a ed. Cambridge University Press, 2017.
- [13] RobotShop, *Lynxmotion Smart Servo (LSS) motor - standard (ST1)*, 2022. dirección: <http://www.lynxmotion.com/p-1126-lynxmotion-smart-servo-lss-motor-standard-st1.aspx>.
- [14] RobotShop, *Lynxmotion Smart Servo (LSS) - Adapter Board (USB Mini)*, 2022. dirección: <http://www.lynxmotion.com/p-1156-lynxmotion-smart-servo-lss-adapter-board-usb-mini.aspx>.
- [15] ROS, *Robot operating system*. dirección: <https://www.ros.org/>.
- [16] A. Dattalo, *ROS Introduction*, 2018. dirección: <http://wiki.ros.org/ROS/Introduction>.
- [17] M. Quigley, B. Gerkey, K. Conley et al., *ROS: an open-source Robot Operating System*. 2009, págs. 1-6.
- [18] ROS, *RVIZ*, 2018. dirección: <http://wiki.ros.org/rviz>.

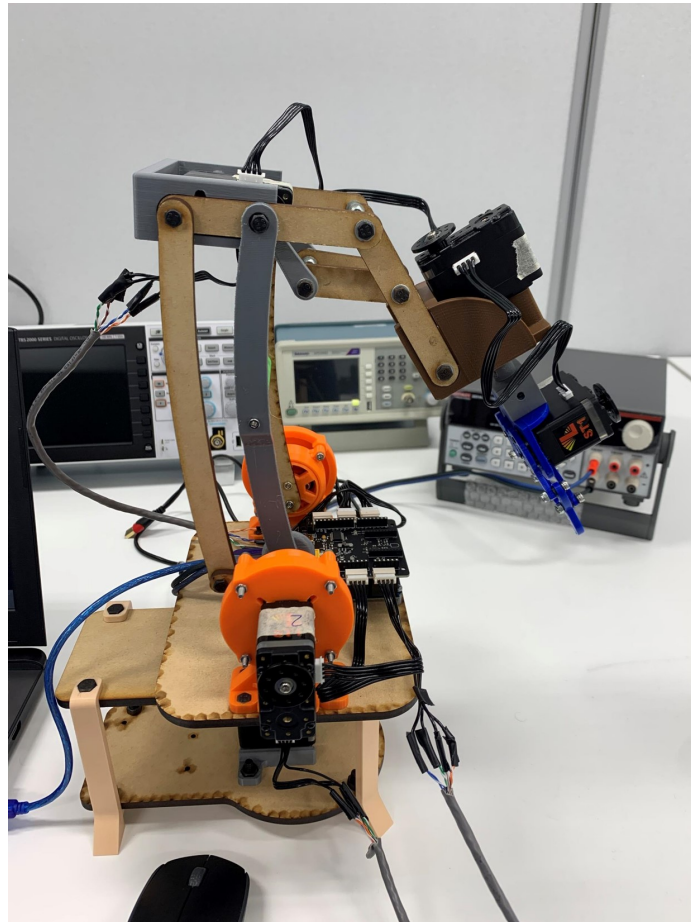


Figura 63: Propuesta entregada del brazo.

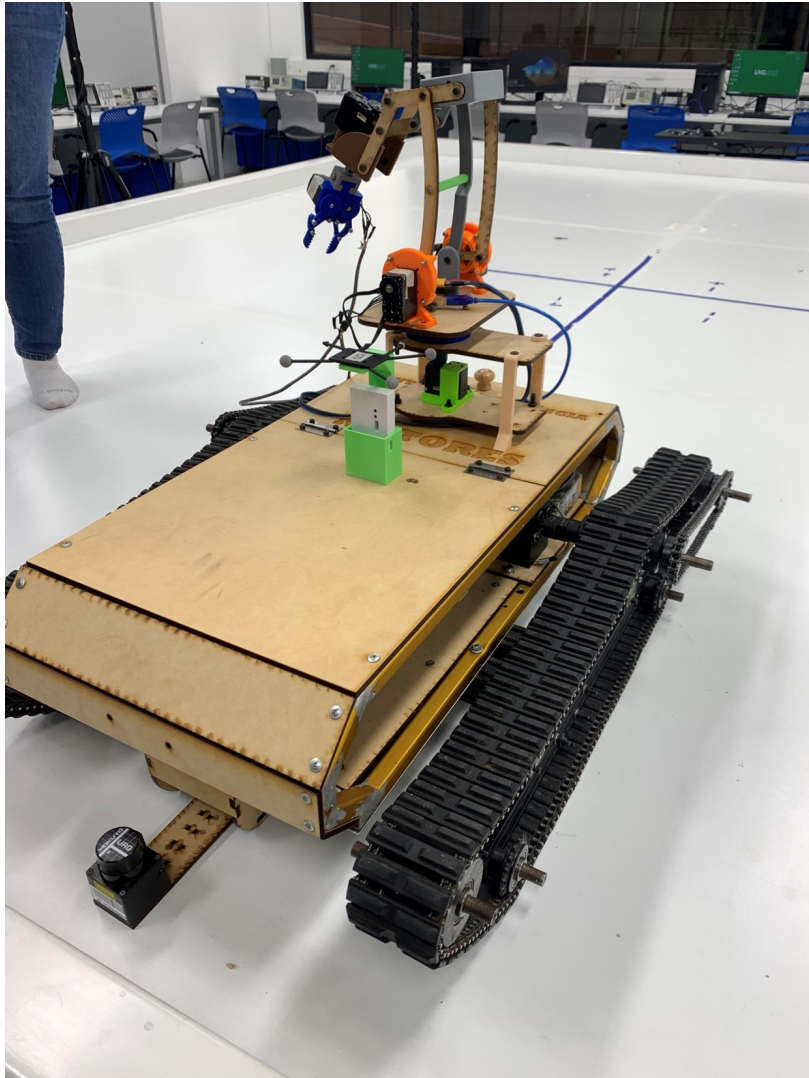


Figura 64: Brazo montado sobre el Rover UVG.

CAD Es el uso de una computadora para facilitar el proceso de creación, análisis y optimización de un diseño. Ayuda a mejorar la calidad, documentación, bases de datos y productividad en general.. [36](#)

gripper Son dispositivos que permiten a los robots agarrar y sujetar objetos. Ayudan en la automatización de procesos como ensamblaje, selección y colocación.. [42](#), [43](#)