UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

Robot Tool: Un nuevo enfoque basado en CNC

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADO EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

P R E S E N T A :

IBZAN PÉREZ HERNÁNDEZ

D I R E C T O R

DR. JUAN CARLOS MARTÍNEZ ROSAS

Ciudad de México, noviembre de 2023.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Dedicatoria

Este trabajo está dedicado principalmente a mis padres, Gumesindo Pérez Rosales y Hermila Hernández Hernández, quienes me han apoyado de manera incondicional, dándome consejos y alentándome a seguir hacia delante con todos mis proyectos. También dedico este trabajo a mis hermanos que me han brindado su ayuda y consejos. A mis familiares que de una u otra manera estuvieron apoyándome.

Agradecimientos

Agradezco primeramente a Dios por permitirme la vida y permitirme llegar hasta este momento tan importante de mi formación profesional. A mi familia, por todo el apoyo y motivación que me han brindan y gracias a ellos he llegado a este punto de mi vida donde puedo cumplir con mis metas y realizar lo que verdaderamente me gusta y que en ocasiones parecieran ser imposibles de cumplir, sin embargo, con su apoyo es posible cumplirlas.

Le agradezco a la Universidad Autónoma de la ciudad de México (UACM) por brindarme la oportunidad de continuar con mis estudios universitarios en la licenciatura de Ingeniería en Sistemas Electrónicos Industriales, de igual manera por todo el equipo e instalaciones que fueron brindadas para el desarrollo de este proyecto tan fabuloso. Por proporcionar el Robot *Scorbot ER-VII*, las impresoras 3D así como también todo el material necesario para construir las impresiones 3D, por brindarnos un espacio para trabajar (Laboratorio B403).

Agradezco a mi asesor de tesis, el Dr. Juan Carlos Martínez Rosas, quien fue un excelente docente en mi formación universitaria y quien sentó las bases y el interés por los temas relacionados con la robótica y la metalmecánica, donde puedo realizar cosas interesantes, como el mismo trabajo que se presenta en esta tesis, le agradezco además por la paciencia y la atención que me brindo durante todo el desarrollo de este trabajo, ya que sin su ayuda no hubiera sido posible llevarlo a cabo.

A mis amigos y compañeros de clase con los cuales la estancia en la universidad fue más llevadera, que junto a ellos pasé momentos buenos y momentos de inmensa presión, pero seguíamos siempre hacia adelante, teniendo una mentalidad positiva, pensando en conseguir nuestras metas y objetivos.

A la empresa CIARobotics por abrirme las puertas a sus instalaciones, brindarme el préstamo de equipo especializado, herramientas, robots, tarjetas de desarrollo y asesoramiento de su equipo de trabajo, que siempre estuvieron pendientes y dispuestos a ayudarme.

Agradezco a la Universidad Autónoma de la Ciudad de México (UACM), que por medio de Coordinación de Servicios Estudiantiles (CSE), me fue otorgado el apoyo para la impresión y empastado de este trabajo de tesis.

Resumen

El presente trabajo está enfocado en la conversión de robots industriales a sistemas CNC. Con el fin de poder utilizar los robots CNC como una herramienta multipropósito y poder ser utilizados en casi cualquier proceso industrial.

En el capítulo 1 se plantean los objetivos generales así como los objetivos específicos y cómo se planean desarrollar, también se propone la Hipótesis a demostrar, la metodología que se seguirá a lo largo del desarrollo de este trabajo de tesis. De forma general se dan a conocer las principales herramientas con las que se cuentan, tanto en el laboratorio así como también el material proporcionado por la empresa CIARobotics.

En el capítulo 2 se describen las características principales de los robots, similitudes entre las partes de un robot y el cuerpo humano, tipo de articulaciones que se pueden encontrar en un robot, la clasificación de los robots dependiendo de su configuración y espacio de trabajo. También se da una breve explicación sobre el funcionamiento de los encoders, ya que son los sensores más utilizados para medir las posiciones de las articulaciones. Tipo de transmisiones comúnmente utilizadas en maquinaria industrial, así como una descripción general de las transmisiones armónicas las cuales son utilizadas en brazos robóticos.

En el capítulo 3 se describen la tecnología CAD/CAM y qué diferencias existen entre estas, así como la importancia y las ventajas de utilizarlas. Se describe el uso de las máquinas CNC y su principio de funcionamiento, el uso del código G y M. Se describen algunas máquinas CNC y en qué aplicaciones son utilizadas, así como sus ventajas y desventajas de utilizar este tipo de tecnología. Se describe el principio básico de funcionamiento de programación de robots industriales.

En el capítulo 4 se describen las principales herramientas matemáticas utilizadas en el modelado de robots, operaciones con vectores como lo son la suma, resta, producto punto y producto entre matrices. Representaciones espaciales y cómo son utilizadas dependiendo de la configuración del robot. Se describen los conceptos de matrices de rotación, traslación y las matrices de transformaciones homogéneas, se utiliza un cubo para representar gráficamente como trabajan las matrices de transformación.

En el capítulo 5 se caracteriza por completo el robot *Scorbot ER-VII*. Es un robot que se encuentra en el laboratorio de Robótica de la UACM, del plantel San Lorenzo Tezonco. Se describen el tipo de encoders que utiliza, las relaciones de transiciones, el espacio de trabajo donde es capaz de moverse adecuadamente.

Se muestran las conexiones para cada una de las articulaciones. Se determinan las medidas reales de cada eslabón.

En el capítulo 6 se muestra el desarrollo para la obtención de la cinemática directa e inversa para un robot planar de 2 GDL y para el robot *Scorbot ER-VII* de 5 GDL, haciendo uso de la convención de *Denavit-Hartenberg*. Una vez obtenidas las ecuaciones se hace uso del software *Matlab* para validar que todas las ecuaciones calculadas y elegidas para que el robot tome una pose en particular sean correctas. Se da un ejemplo del uso de las matrices de rotación para orientar el efector final para cualquier robot.

En el capítulo 7 se muestran las características generales de la tarjeta controladora Kflop y SnapAmp las cuales se utilizarán para implementar las conversiones de los robots a sistemas CNC. Se muestra el desarrollo para la conversión a CNC del robot planar de 2 GDL y del *Scorbot ER-VII*, por otro se muestran las pruebas y resultados obtenidos utilizando código G. También se muestra el desarrollo e implementación de las herramientas que se utilizaran como efector final para el robot *Scorbot ER-VII*, utilizando un láser y un dremel 3000 y finalmente se hace una comparación entre los diferentes tipos de maquinaria CNC.

Índice

Lista de figuras

Capítulo 1

Introducción

Como se ha visto a lo largo de los años, las máquinas y herramientas que ha usado el ser humano se han modernizado gracias al desarrollo tecnológico. Una de las herramientas que más llaman la atención son los robots industriales, por sus infinitas aplicaciones en donde se pueden utilizar, áreas como lo son la industria automotriz, minería, agricultura, medicina, etc. Los robots desarrollan trabajos que el ser humano no podría desempeñar de manera eficiente o simplemente no podría llevarlos a cabo por su dificultad para ejecutarlo. El término robot apareció en 1921, proviene del vocablo checo "*Robota*" que significa trabajo arduo, repetitivo y monótono, el término fue empleado en la obra de teatro R.U.R. (Rossum's Universal Robots), escrita por Karel Capek, [Capek, 2004]. Esta obra trata de la creación de robots, los cuales sustituían el trabajo de los humanos y los desempeñaban con una mejor eficiencia gracias a que estos carecían de emociones, sin embargo, los robots se terminaron rebelando contra sus creadores, destruyendo así a la raza humana. Como se puede ver, el término robot fue empleado para describir un mecanismo o ente que desarrolla las tareas del ser humano con una mayor eficiencia.

Sin dejar a un lado los Robots, las máquinas CNC como los tornos o fresadoras también son sumamente importantes, ya que con estas se desarrollan infinidad de piezas mecánicas con una muy buena precisión, tomando en cuenta su fácil programación. Algunas de estas máquinas hacen uso de los sistemas CAD/CAM, diseño asistido por computadora y manufactura asistida por computadora, donde la parte CAD se encarga de la visualización de forma detallada de la pieza a fabricar y la parte CAM es la parte donde se tiene el código necesario para mecanizar o fabricar la pieza, es decir esta se encarga de realizar los cálculos de las trayectorias para que la pieza final quede con las especificaciones geométricas requeridas.

Ahora imaginemos que se tiene las mejores características de los Robots y la

fácil programación de los sistemas CAD/CAM, se tendría una herramienta multipropósito y de fácil programación.

Esta tesis reporta una de las muchas maneras de poder convertir un robot de uso industrial a un sistema CNC, teniendo en cuenta que el sistema a implementar es de bajo coste y además que este sistema se podrá utilizar para aplicaciones industriales.

1.1 Objetivos generales

En este trabajo se tiene como objetivo demostrar que es posible convertir robots industriales en sistemas de Control Numérico Computarizado (CNC) sin hacer uso de softwares especializados, los cuales necesitan el controlador original del robot para lograr que trabaje con código G, en otras palabras, hacer uso únicamente de la estructura mecánica, sensores y actuadores del sistema original. En este caso se utilizará un robot *Scorbot ER-VII*, el cual se tiene en el laboratorio B403, en la UACM, plantel San Lorenzo Tezonco, se modelará matemáticamente y las ecuaciones obtenidas se cargarán a la tarjeta controladora Kflop la cual tiene software gratuito para procesar código G.

1.2 Objetivos específicos

Se busca realizar la conversión de un robot industrial a CNC, para obtener una herramienta multipropósito, haciendo uso de la filosofía CNC, es decir, utilizar el control numérico asistido por computadora (CNC). Se empleará el robot *Scorbot ER-VII*, el cual no cuenta con su sistema de control, por lo tanto, se harán mediciones de las relaciones de engranajes que se tiene en cada articulación, la cantidad de pulsos que entrega cada encoder por cada revolución, señales de los finales de carrera (*switch*) para delimitar restricciones físicas, también se obtendrán mediciones de longitudes de eslabones.

Se obtendrá la cinemática directa e inversa y se simulará en *Matlab* para validar que las ecuaciones obtenidas son correctas. Finalmente, con las ecuaciones obtenidas se procederá a cargarlas en la tarjeta controladora Kflop, la cual es utilizada principalmente para controlar maquinaria CNC. Se realizarán pruebas de movimientos para validar que el *Scorbot ER-VII* puede ser controlado por medio de código G y hace el seguimiento de trayectorias de forma correcta.

1.3 Hipótesis

Se puede convertir cualquier robot industrial a una herramienta CNC multipropósito bajo la filosofía CNC haciendo uso de la tarjeta controladora Kflop. El robot CNC podrá ser utilizado en casi todas las aplicaciones industriales como podrían ser soldadura, pintura, fresado y torneado de piezas de diversos materiales, tallado de piedra, elaboración de PCBs, tallado en madera, corte por plasma, corte por chorro de agua, paletización, entre otras aplicaciones industriales.

1.4 Metodología

Para el desarrollo de esta tesis se seguirá la siguiente metodología:

Se utilizará la tarjeta controladora Kflop para convertir cualquier robot industrial a un sistema CNC, por lo tanto, se investigarán los pasos a seguir para poder hacer uso de la cinemática utilizando la tarjeta Kflop, ya que esta tarjeta es utilizada comúnmente para routers CNC, cortadoras de plasma, automatización o máquinas cartesianas.

Se obtendrá el modelo cinemático directo e inverso para el robot planar de 2 GDL, ya que al ser un sistema relativamente sencillo se podrán hacer pruebas de funcionamiento de forma rápida y nos servirá para ver si el uso de la cinemática de cualquier modelo de robot es posible cargarlo al controlador Kflop.

Se obtendrán las características principales para el robot *Scorbot ER-VII*, como podrían ser longitudes de los eslabones, tipo de motores, tipo de encoder, relaciones de transmisión, área de trabajo, etc. En pocas palabras caracterizar el robot *Scorbot ER-VII*.

Se obtendrán el modelo cinemático directo e inverso para el robot *Scorbot ER-VII* y se hará uso de *Matlab* para poder comprobar que las ecuaciones obtenidas son correctas, una vez comprobadas las ecuaciones se procederá a crear la clase cinemática haciendo uso de Visual Estudio, finalmente se cargará nuestra nueva clase a la tarjeta controladora Kflop para hacer pruebas de funcionamiento.

Como el robot *Scorbot ER-VII* no cuenta con su efector final, se diseñarán y construirán 2 herramientas para probar el funcionamiento correcto del *Scorbot ER-VII* ya convertido a un sistema CNC. Para diseñar las herramientas se hará uso de Software (CAD) gratuito o en sus versiones de prueba, donde se puedan obtener sus modelos en 3D para poder imprimirlos en impresoras tridimensionales, una

vez se tengan los modelos en 3D se imprimirán con las impresoras 3D con las que se cuenta en el laboratorio de robótica de la UACM, del plantel San Lorenzo Tezonco.

El laboratorio de robótica de la Universidad Autónoma de la Ciudad de México (UACM) en conjunto con la empresa CIA Robotics ponen a disposición sus instalaciones, herramientas y asesoría para el desarrollo de este trabajo. Algunos equipos disponibles con los que se cuenta son:

- Robot *Scorbot ER-VII*.
- Robot Motoman SK16.
- Impresora de resina (Mega 8k).
- Impresora filamento (Anycubic Chiron).
- Tarjeta controladora Kflop.
- Amplificador de potencia SnapAmp.
- Licencia para software *Matlab*.
- Software libres *KmotionCNC, Kmotion* y Visual estudio 2022.

En este primer capítulo se plantean los objetivos generales así como los objetivos específicos y cómo se planean desarrollar, también se propone la Hipótesis a demostrar, la metodología que se seguirá a lo largo del desarrollo de este trabajo de tesis. De forma general se dan a conocer las principales herramientas con las que se cuentan, tanto en el laboratorio así como también el material proporcionado por la empresa CIARobotics.

Capítulo 2

Características de los robots

En este capítulo se describirá de forma general algunas de las características de los robots. La morfología se refiere a la forma en como está construido un robot, su estructura mecánica, algunos switches para limitar movimientos, encoders, etc. También se describen elementos que permiten transmitir pares o convertir movimientos rotacionales a lineales o viceversa, como las transmisiones mecánicas.

2.1 Similitudes entre robots manipuladores y el cuerpo humano

Los robots manipuladores son semejantes al cuerpo humano, haciendo una comparación como se muestra en la (Figura 2.1), en donde la primera articulación se puede comparar con la cintura, la segunda articulación con el hombro, la tercera al codo y la cuarta a la muñeca, la parte verde es comparable con el brazo y la parte azul con el antebrazo.

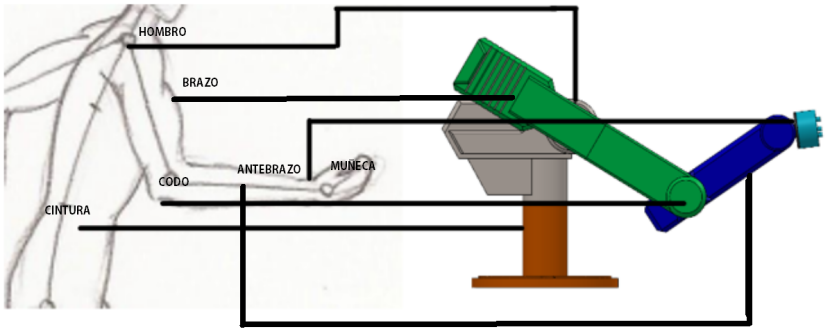


Figura 2.1: Similitudes entre robot manipulador y brazo humano.

2.2 Tipos de articulaciones

Los robots manipuladores son brazos articulados, están esencialmente formados por articulaciones y eslabones, los cuales están conectados entre sí, gracias a las articulaciones los eslabones se pueden mover relativamente entre sí como se observa en la (Figura 2.2). Las articulaciones son las encargadas de transmitir la energía necesaria para producir movimiento en los eslabones.

La primera articulación está asociada a la base, donde puede producir diferentes movimientos dependiendo el tipo de articulación. El último eslabón no está conectado a ninguna articulación, sin embargo, en este se colocará la herramienta necesaria para desarrollar alguna tarea en particular. Como el final del robot no está conectado con la base, desde el punto de vista mecánico se dice, que es una cadena de cinemática abierta, [Cortés, 2020].

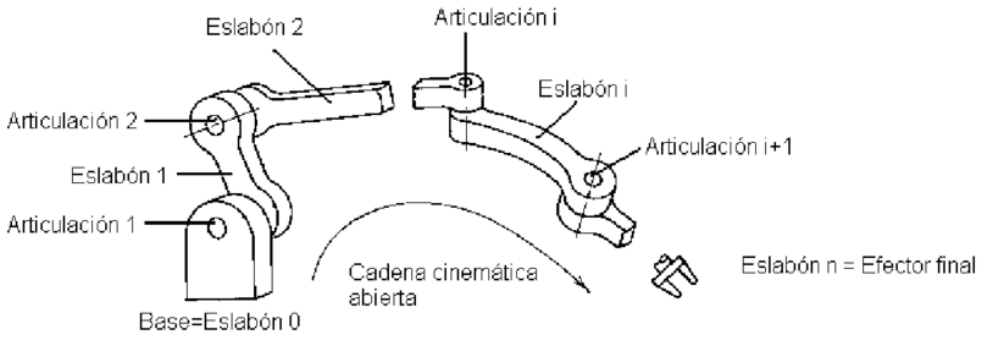


Figura 2.2: Estructura de un robot, articulaciones y eslabones.
Tomado de [Vila-Rosado and Domínguez-López,].

Existen dos tipos de articulaciones principales, una produce un movimiento rotacional y, por lo tanto, es llamada articulación rotacional, la segunda produce un movimiento lineal y se conoce como articulación prismática o lineal, (Figura 2.3). Haciendo uso de los dos tipos de articulación se pueden generar distintos tipos de movimientos. Un movimiento producido por una articulación puede ser de 2 tipos, como se mencionó anteriormente, de revolución o prismática, o una combinación de ambas, cada movimiento independiente en las articulaciones proporciona un grado de libertad (GDL), [Baturone, 2005]. En la (Figura 2.3) se muestran diferentes tipos de articulaciones, donde en la primera columna se muestra una pequeña representación gráfica, en la segunda columna se presenta una breve descripción

de la articulación y en la última columna se observa los grados de libertad proporcionados por la articulación.

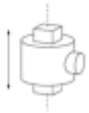
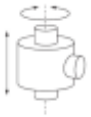
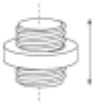
ESQUEMA	ARTICULACION	GRADO DE LIBERTAD
	**ROTACIONAL.** Suministra un grado de libertad, consistente en una rotación alrededor del eje de la articulación (es la más empleada).	1
	**PRISMÁTICA.** El grado de libertad consiste en una traslación a lo largo del eje de la articulación.	1
	**CILÍNDRICA.** Existen 2 grados de libertad que son: Uno es rotación y el otro es traslación.	2
	**PLANAR.** Se caracteriza por el movimiento de desplazamiento en un plano, existiendo 2 grados de libertad.	2
	**ESFÉRICA O RÓTULA.** Combinan 3 giros en 3 direcciones perpendiculares al espacio.	3
	**TORNILLO.** El grado de libertad consiste en la traslación a lo largo de un eje roscado.	1

Figura 2.3: Tipo de articulaciónes.

2.3 Clasificación de robots por configuración y espacio de trabajo

Se clasificarán en cuatro clases principales, estas son las configuraciones más utilizadas; cartesiano, cilíndrico, esférico (polar) y articulado (antropomorfo) y dependiendo de la clase se tiene un espacio de trabajo determinado. El área de trabajo es el volumen al cual el extremo final del robot puede posicionarse, cabe recalcar que si el extremo final del robot bien puede trabajar dentro de este volumen, en ocasiones no es posible orientar de forma correcta el efector final.

Cartesiano: Este tipo de robots se mueve en el espacio tridimensional de forma lineal. Se desplaza en las direcciones de las coordenadas (x, y, z), coordenadas rectangulares. Su espacio de trabajo está delimitado por las distancias de los eslabones, el espacio de trabajo forma un cubo o prisma rectangular, (Figura 2.4). Este tipo de robots se emplean en aplicaciones de manejo y montaje de diversos materiales, ensamblado de componentes diversos, inspección de dispositivos montando una cámara de visión como herramienta de trabajo, etcétera.

Cilíndrico: Este tipo de robot se diferencian por tener una articulación de revolución y dos prismáticas. Los puntos alcanzables pueden especificarse en coordenadas cilíndricas (radio, ángulo, altura) que es lo mismo que (r,θ,z). Su espacio de trabajo está formado por dos cilindros concéntricos como se muestra en la (Figura 2.5). Se utilizan para mover objetos, aplicaciones de soldadura y operaciones de ensamblaje.

Polar o esférico: Este tipo de robots tiene 2 articulaciones de revolución y una prismática, donde la primera articulación girará θ grados, la segunda articulación se desplazará una distancia z y la última articulación gira ϕ grados respecto a la vertical que pasa por el centro del robot, la posición final del robot se describe principalmente en coordenadas polares o esféricas (r, θ, ϕ), donde r es la distancia de la base al punto final, θ es el giro respecto a la base y ϕ es un giro respecto a la vertical que pasa por el centro del robot, como se muestra en la (Figura 2.6).

Antropomorfos: Este tipo de robots contienen solo movimientos rotacionales entre sus articulaciones y su área de trabajo es de la forma de una esfera, (Figura 2.7). Este tipo de robots tiene infinidad de aplicaciones, por ejemplo, soldadura, pintura, acomodo de piezas, etcétera.

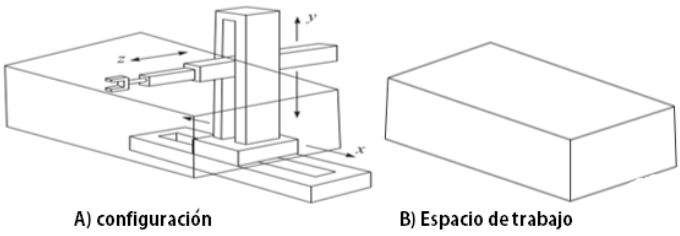


Figura 2.4: Robot cartesiano y espacio de trabajo.

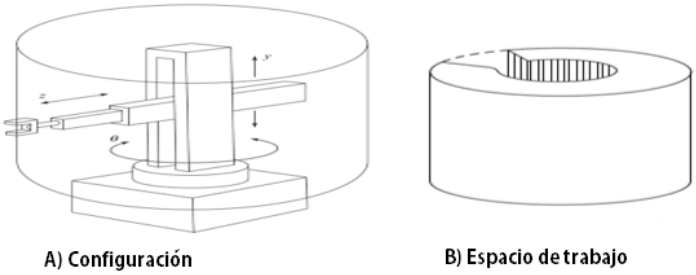


Figura 2.5: Robot cilíndrico y espacio de trabajo.

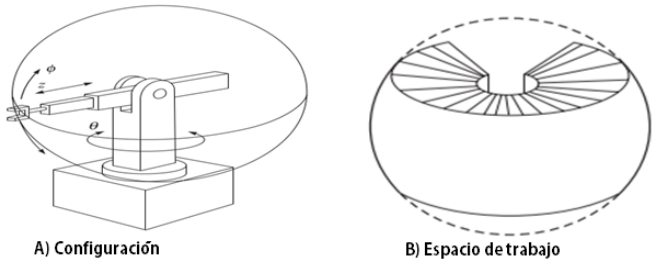


Figura 2.6: Robot polar y espacio de trabajo.

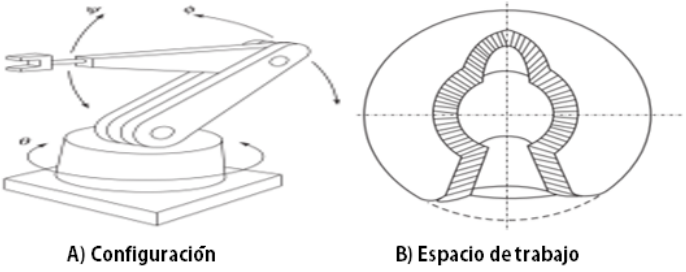


Figura 2.7: Robot antropomorfo y espacio de trabajo.

2.4 Sensores

En general, es necesario medir las posiciones, velocidades y aceleraciones de las articulaciones de los robots, con la finalidad de hacer que se realicen los movimientos deseados de manera óptima. De este modo, se pueden utilizar infinidad de sensores con su respectivo acondicionamiento para obtener dichos parámetros, sin embargo, los más utilizados actualmente son los decodificadores, también llamados encoders.

Los encoders más utilizados son los ópticos, donde el principio del funcionamiento consiste en un diodo emisor de luz, el cual emite un haz de luz que pasará a través de un disco con ranuras distribuidas uniformemente como se muestra en la (Figura 2.8), si el haz de luz pasa por la rendija será detectado por un fotorreceptor el cual generará una señal lógica TTL.

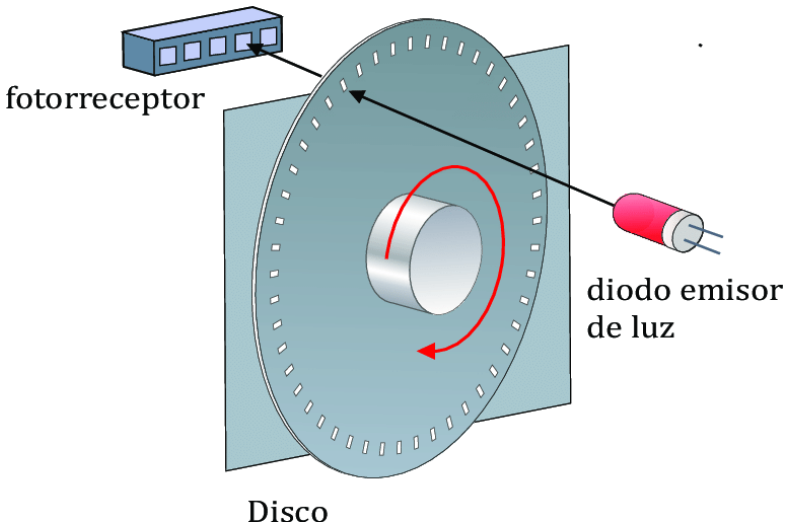


Figura 2.8: Principio de funcionamiento de un encoder óptico.

Dependiendo las señales que entregue un encoder a su salida puede ser considerado del tipo absoluto o incremental. Si para cualquier instante de tiempo se tiene una salida el encoder será absoluto, el otro caso es cuando solo se obtiene información cuando el encoder está en movimiento, este se denomina encoder incremental.

2.4.1 Encoders incrementales

Los encoders incrementales constan de un disco de plástico al cual se le superpone un disco con marcas opacas distribuidas radialmente y equidistantemente entre sí, para llevar a cabo una medición el encoder se acopla al eje del sistema a medir. Una vez que el eje comienza a girar, el haz de luz llega al fotorreceptor, al detectarse un haz de luz se producirá un estado lógico TTL, estos estados se pueden contar utilizando algún microcontrolador y así saber la posición del eje. Sin embargo, se tiene otro problema cuando se tiene un solo disco dentado, puesto que si el eje gira en sentido contrario, el microcontrolador seguirá contando los mismos pulsos y no estaríamos realizando una medición correcta. Para solucionar dicho problema se coloca otro sensor, el cual se coloca desfasado 90 grados respecto al anterior, [ROSALENY, 2021].

En ocasiones se coloca una marca adicional, la cual servirá para indicar cuando se ha completado un giro o si se tuviera un corte de energía, se podría utilizar como referencia para iniciar a contar.

Por lo tanto, podemos decir que los encoders pueden clasificarse en 2 tipos, incrementales o bidireccionales, en el primero caso no se reconoce el giro del eje, en el segundo se tendrán 2 salidas A y B, las cuales estarán desfasadas 90 grados una respecto a la otra y servirán para determinar el sentido de giro además de que mejorará la resolución del encoder, como se puede observar en la (Figura 2.9).

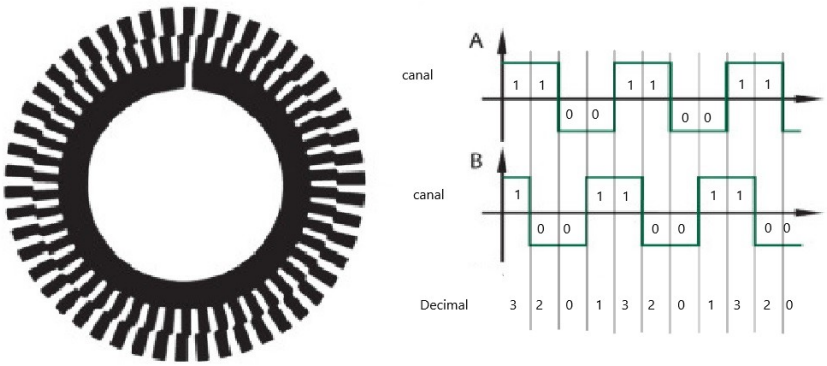


Figura 2.9: Encoder incremental Bidireccional.

La (Tabla 2.1) muestra los valores en decimal que se obtendrán dependiendo de los estados lógicos que se encuentren del canal A y B los cuales representaran un valor binario, el bit más significativo estará asignado al canal A y el menos significativo al canal B. La (Tabla 2.1) se rellenó conforme a la (Figura 2.9).

Tabla de estados		
	Bit más significativo	Bit menos signifiativo
valor en decimal	Canal A	Canal B
0	0	0
2	1	0
3	1	1
1	0	1

Tabla 2.1: Valor decimal asignado dependiendo del estado de los canales.

En la (Tabla 2.2) se asigna un número a la variable estado, el valor correspondiente es igual al valor decimal de los dos canales, también se puede observar el orden en que el eje del motor está girando, esto está indicado con las flechas que se encuentran en los costados.

Tomando como referencia la (Tabla 2.2), se procede a rellenar la (Tabla 2.3), en esta última se hace la combinación de todos los posibles casos a ocurrir, para determinar la posición, sentido de giro o algún error (falla). En la (Tabla 2.3) se hace la comparación del estado anterior y el estado actual, por ejemplo, si las señales de los canales son ambas cero estará en el estado 0, si se gira el eje las señales de los canales cambian y en conjunto pasarán a ser el estado 2, el eje estará girando en sentido positivo además de que como se observa el estado 0 pasa a ser el anterior y el actual sería el estado 2, luego el estado 2 pasará a ser el estado anterior y así sucesivamente.

En donde los símbolos de la (Tabla 2.3) tendrán el siguiente significado:
X=no gira
+= giro positivo
-=giro negativo
E= no sucede (error)

La información de la (Tabla 2.3) se puede utilizar para que se cargue a algún microcontrolador para que realice la lectura de las señales de los canales y obtener la información requerida y utilizarla, por ejemplo, mostrar la velocidad a la que se mueve el eje, controlar la velocidad del giro del eje o detener el giro del eje si se llegara a detectar un error.

GIRO (+)	ESTADOS	A	B	GIRO (-)
	2	1	0	
	3	1	1	
	1	0	1	
	0	0	0	

Tabla 2.2: Asignación de estados.

ANTERIOR / ESTADOS ACTUAL			0		1		2		3	
			A	B	A	B	A	B	A	B
	A	B	0	0	0	1	1	0	1	1
0	0	0	X		-		+		E	
1	0	1	+		X		E		-	
2	1	1	-		E		X		+	
3	1	0	E		+		-		X	

Tabla 2.3: Cambio de estados, sentido de giro y errores en un encoder incremental.

2.4.2 Encoders absolutos

Los encoders absolutos darán la posición angular del eje en todo momento. El principio de funcionamiento es similar al encoder incremental, solo que para este se tiene n emisores y n receptores. Este tipo de encoder está formado por un disco, el cual está dividido en secciones transparentes u obscuras, (Figura 2.10), las cuales están acomodadas de forma que el conjunto de señales hacer referencia a una posición particular. Lo más usual es que los discos se dividan en potencias de 2, con la finalidad de decodificar de forma binaria.

En los encoders absolutos se podrá detectar el sentido del giro del eje sin necesidad de tener algún sistema extra que lo detecte, esto porque cada posición del eje está especificada por una sección del encoder, es decir, estará codificado de manera absoluta, [ROSALENY, 2021].

La salida que algunos encoders absolutos presentan están dadas en código BCD

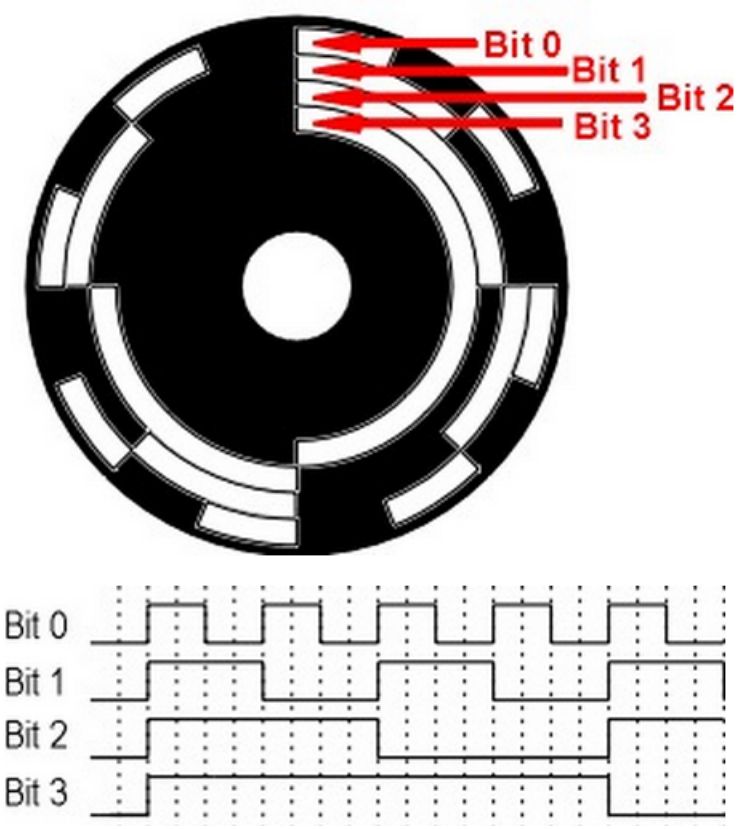


Figura 2.10: Encoder absoluto.

o código Gray. En el código BCD (Binario Codificado a Decimal), se trata del código binario, solo que ahora se utiliza la codificación del 0 al 9 en decimal o 0000 al 1001 en binario. El código Gray se utiliza debido a que sirve para evitar errores, ya que en cada cambio a la salida solo un bit cambiará.

2.5 Actuadores

Los actuadores proporcionan los pares necesarios para dotar de movimiento las partes mecánicas del robot, por ejemplo, mover un eslabón. Entre los actuadores que podemos encontrar en el mercado se encuentran del tipo neumáticos, hidráulicos o eléctricos, según si principio de funcionamiento [Kumar Saha, 2010]. Dentro de los actuadores se pueden encontrar los motores de corriente continua (DC), corriente alterna (AC), motores paso a paso o servomotores. Actualmente se están utilizando de forma más común los servomotores.

Se puede decir que un servomotor es un motor que cuenta con algunas características especiales, una de las principales características es que cuenta con un sistema de retroalimentación (encoder), el cual le indica al servo drive (controlador del servomotor) la posición del eje del motor y este corrige el error de posición requerida (referencia), (Figura 2.11). Con estos sistemas se busca corregir el error y obtener una alta precisión. Estos servo sistemas pueden utilizarse para controlar la velocidad, posición o par, [YASKAWA, 2002].

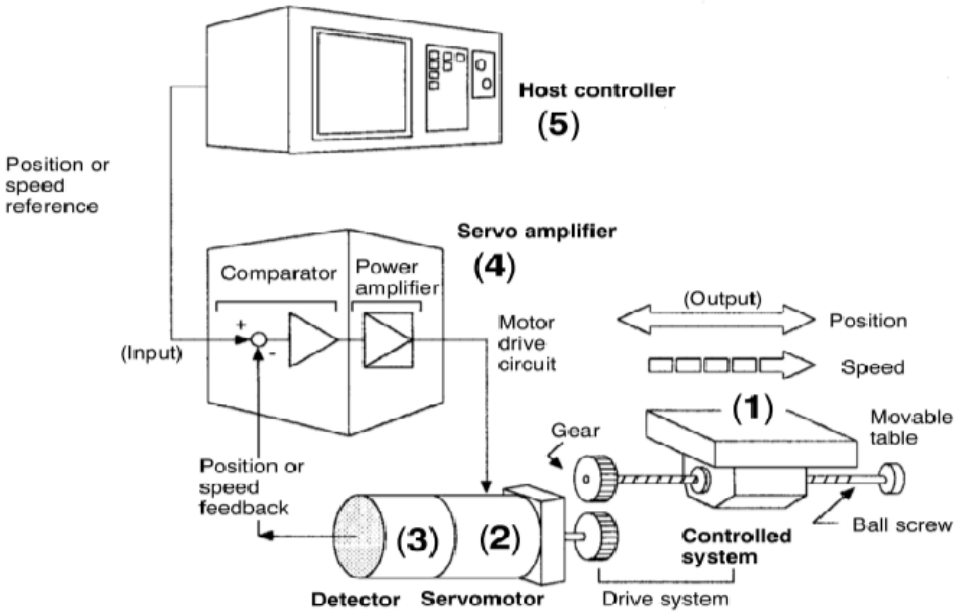


Figura 2.11: Sistema general para servomotores.
Tomada de [YASKAWA, 2002].

2.6 Elementos de transmisión

Los elementos de transmisión se encargan de transmitir el movimiento de los actuadores (motores) a las articulaciones o herramientas (efector final) del robot.

Usualmente, se utilizan cajas de transmisión como elementos de transmisión, estas se encuentran formadas por un conjunto de engranes que reducen la velocidad del motor y aumentan el par, siempre cumpliendo con la ley de la conservación de la energía. Las cajas de transmisión más sencilla solo están conformadas por un par de engranes, si se necesita un mayor par se agregan más etapas de engranajes. En la (Figura 2.12) se muestran los elementos principales de una caja de transmisión.

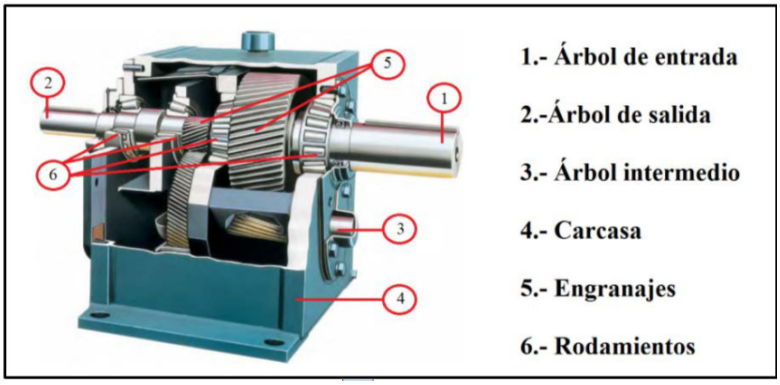


Figura 2.12: Elementos de un sistema de engranajes.
Imagen tomada de [Duque, 2017].

Otros elementos para transmitir pares y de igual forma muy utilizados en la industria son las correas dentadas, sistema piñón cremallera e incluso una cadena. Haciendo una comparación entre los sistemas de cajas de engranajes y los demás elementos se pueden presentar las siguientes ventajas y desventajas [Duque, 2017].

- Ventajas

    1. Mejor regulación de velocidad y pares.
    2. Mayor eficiencia al momento de transmitir la potencia del motor.
    3. Confiabilidad.
    4. Reducidos costos de mantenimiento.
    5. Mayores transmisiones de potencia en menores volúmenes.
    6. Trabajo en ambientes industriales.

- Desventajas

    1. Costos elevados de producción y diseño.
    2. Requiere sistemas de control.
    3. Cambios de lubricantes.
    4. Generación de ruidos mientras están en funcionamiento.

Existe una cantidad grande de sistemas de transmisión, pero podemos clasificarlas principalmente por el tipo de engranajes que lo conforman y la disposición en la que se encuentran acomodados los ejes y engranes, (Figura 2.13).

Transmisión de ejes coaxiales: Los ejes de entrada y el eje de salida son colineales. Este tipo de transmisión está constituido principalmente con engranajes rectos o helicoidales, puede estar compuesto de múltiples etapas según la relación que se necesite. Se pueden tener relaciones de 1 a 8 por etapa y se pueden tener rendimientos de 93 % a 99% dependiendo del tipo de materiales, lubricación, etcétera, (Figura 2.13 a).

Transmisión de ejes Paralelos: El eje de entrada y el eje de salida se encuentran dispuestos de forma paralela entre ellos, este tipo de transmisión está constituido principalmente con engranajes rectos o helicoidales, una ventaja de este tipo de transmisiones es que puede tener múltiples ejes de salida. Se pueden tener relaciones de 1 a 8 por etapa y se pueden tener rendimientos de 93% a 99% dependiendo del tipo de materiales, lubricación, etcétera, (Figura 2.13 b).

Transmisión de ejes perpendiculares: Los ejes de entrada y salida se encuentran dispuestos de forma perpendicular. Este tipo de transmisiones está constituido principalmente por engranajes cónicos. Se pueden obtener relaciones de 1 a 6 por etapa y conseguir un rendimiento entre 90% y 95%, (Figura 2.13 c).

Transmisión de corona sinfín: El eje de entrada y de salida se encuentran dispuestos paralelamente. Solo contienen una etapa de reducción. Se pueden tener relaciones de 1 a 100 por etapa y se pueden tener rendimientos de entre 45% a 90%, (Figura 2.13 d).

Un tipo de transmisión un poco diferente a las anteriores es la *Harmonic-Drive*,

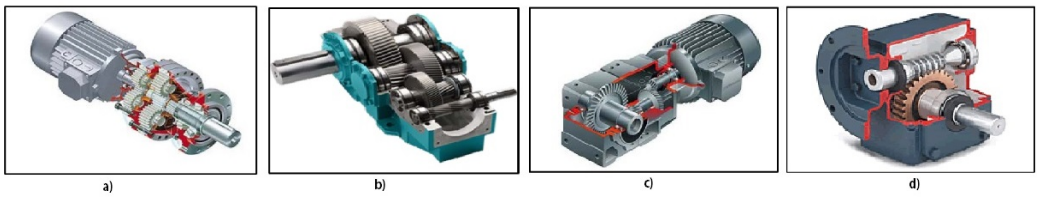


Figura 2.13: Tipo de cajas de transmisión. a) Transmisión de ejes coaxiales, b) Transmisión de ejes paralelos, c)Transmisión de ejes perpendiculares y d) Transmisión de corona sinfín.

transmisión armónica, estas presentan una muy alta precisión y una baja holgura (*backlash*). Este tipo de transmisiones son muy utilizadas en brazos robóticos y dispositivos de exploración de entornos espaciales, por mencionar algunos ejemplos.

Se basa en el principio llamado, deformación de onda de engranaje (*strain-wave gearing*), su nombre es derivado por el elemento encargado de la transmisión de par, *flexspline* [Sandin, 2003]. Este tipo de transmisiones consta de tres elementos principales, el *circular spline, flexspline, wave generator,* (Figura 2.14). El *circular spline* es un círculo sólido y no giratorio, con dientes en su parte interna, el *flexspline* es un círculo dentado en la parte exterior el cual es flexible y de un diámetro menor que el *circular spline*, esto con el fin de poder se deformarmado por el generador de ondas *(wave generator)* y poder hacer que los dientes del *flexspline* y del *circular spline* se acoplen entre ellos.

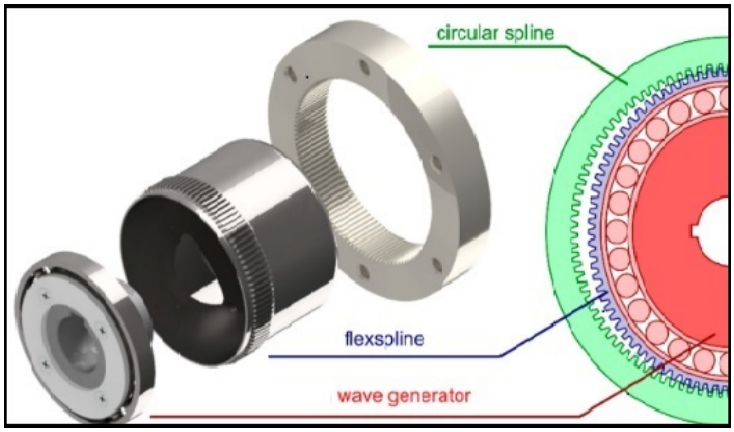


Figura 2.14: Elementos de una transmisión *Harmonic Drive*.

En la (Figura 2.15) se observa el funcionamiento de la transmisión armónica. Los dientes más alejados del centro del *flexspline* se enganchan con los dientes del *circular spline* de esta forma se transferirá el par, (Figura 2.15a), al mover el generador de ondas se consigue que los dientes del *flexspline* se desplacen y se acoplen a los dientes siguientes del *circular spline* (Figura 2.15b), al hacer que el *flexspline* tenga un diámetro menor que el *circular spline* se consiguen relaciones de transmisión, como se puede ver en la (Figura 2.15c), donde al completar un giro el *flexspline* las flechas no coinciden, esto significa que tiene una relación diferente de 1.

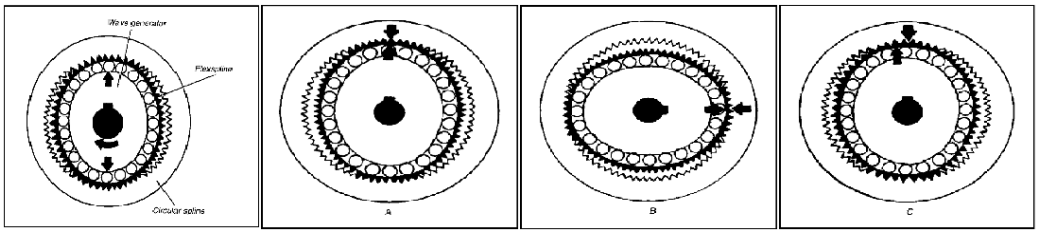


Figura 2.15: Funcionamiento de una transmisión armónica.

Capítulo 3

Tecnología CAD/CAM

En la actualidad una gran cantidad de proyectos, principalmente en metalmecánica, son elaborados y comprobados para posteriormente mandarlos a fabricación, ayudándose de herramientas de diseño asistido por computadora y manufactura asistida por computadora, también conocidos como CAD/CAM.

Esta tecnología se ayuda de las computadoras para diseñar y fabricar infinidad de productos, con esto se consigue una mayor precisión y costos de producción reducidos.

La parte de Diseño Asistido por Computadora (CAD- *Computer Aided Design*), es donde el diseñador representa gráficamente la pieza, en 2 o 3 dimensiones, utilizando un software adecuado de dibujo y modelado de sólidos, donde se crean entidades tales como líneas, arcos, círculos, polígonos, elipse, entre otros, (Figura 3.1).

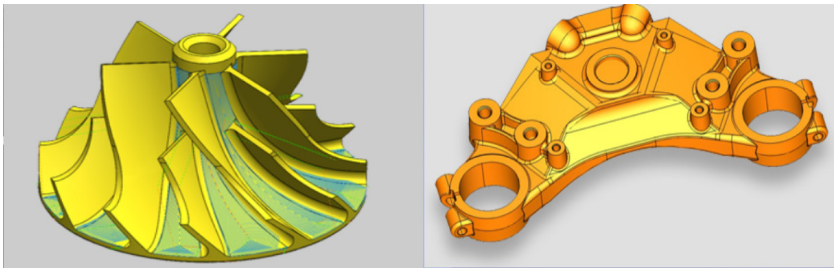


Figura 3.1: Diseño Asistido por Computadora (CAD).

La parte de manufactura asistida por computadora (CAM- *Computer Aided Manufacturing*), es donde se selecciona las entidades de la pieza, se ingresan parámetros de corte, avance, revoluciones de giro de la herramienta, profundidad de corte, etc.

También se realiza la simulación del proceso de mecanizado, la traducción de la simulación al lenguaje de Control Numérico, para finalmente transmitir el programa a la máquina CNC, (Figura 3.2).

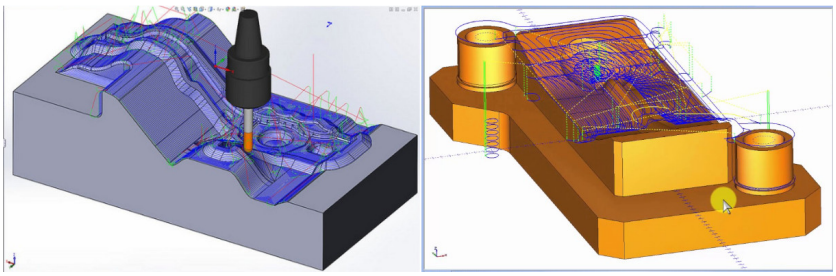


Figura 3.2: Manufactura asistida por computadora (CAM).

3.1 Control Numérico Computarizado (CNC)

El control numérico computarizado (CNC) es un conjunto de códigos que sirve para controlar con precisión las operaciones a realizar de una máquina, estas operaciones son instrucciones que se forman de un conjunto de letras, números y símbolos, las cuales hacen referencia a posiciones, distancias, movimientos o funciones específicas [García, 2006]. La unidad de control de la máquina CNC es la que se encarga de traducir las instrucciones y enviar señales a los servomotores para generar los movimientos específicos. Para obtener un programa de control numérico se pueden seguir 2 métodos, programando manualmente o haciendo uso de un software CAD/CAM.

El siguiente ejemplo muestra un pequeño programa de control numérico el cual se envía o introduce a la máquina CNC, para que esta intérprete el código y genere los movimientos necesarios para maquinar de forma precisa la pieza, (Figura 3.3).

Donde los códigos hacen referencia a:

G90 : Programación en coordenadas absolutas.
G71: Unidades en milímetros.
S: Revoluciones de giro de la herramienta cortante.
GO: Movimiento rápido, movimiento en línea recta moviendose a la maxíma velocidad.
G1: Avance de corte. Los ejes se mueven de tal forma que la herramienta se

Figura 3.3: Código de control numérico de ejemplo.
Tomado de [Vila-Rosado and Domínguez-López,].

mueve a lo largo de una línea recta con un avance determinado por F.
G2: Interpolación circular en sentido horario.
M3: Accionamiento del giro del husillo principal.
M5: Desactivación del giro del husillo principal.
M2: Fin del código

La interpolación sirve para generar movimientos coordinados de todos los ejes de la máquina y generar un movimiento en línea curva o movimientos en forma de arcos.

La programación para maquinaria CNC está estandarizada y utiliza principalmente los códigos G y M, esto permite utilizar casi los mismos programas sin importar el tipo de máquina CNC utilizada, se puede cargar el programa directamente a la máquina o en ocasiones se hacen pequeños cambios al código.

Los códigos G y M es un lenguaje de bajo nivel, secuencial que contiene la información necesaria para realizar el maquinado de una pieza a través de alguna máquina CNC. Los nombres G y M se refieren a instrucciones Generales y Misceláneas, los códigos G especifican funciones de movimiento, avances, ciclos, pausas, interpolación, etc. Los códigos M especifican funciones misceláneas, por ejemplo, arranque del husillo, paro del husillo cambio de herramienta, etc. Algunos ejemplos de códigos G y M se dan a continuación, sin embargo, para una correcta programación se deberá de revisar el manual de cada uno de los equipos CNC.

Códigos Generales
G00: Posicionamiento rápido (sin maquinar)
G01: Interpolación lineal (maquinando)
G02: Interpolación circular (horaria)
G03: Interpolación circular (antihoraria)

G20: Comienzo de uso de pulgadas
G21: Comienzo de uso de milímetros
G28: Volver al home de la máquina
G32: Maquinar una rosca en una pasada
G36: Compensación automática de herramienta en X
G37: Compensación automática de herramienta en Z
G41: Compensación de radio de curvatura de herramienta a la izquierda
G42: Compensación de radio de curvatura de herramienta a la derecha
G76: Maquinar una rosca en múltiples pasadas

Códigos Misceláneos
M00: Parada opcional
M01: Parada opcional
M02: Reset del programa
M03: Hacer girar el husillo en sentido horario
M04: Hacer girar el husillo en sentido antihorario
M05: Frenar el husillo
M06: Cambiar de herramienta
M07: Abrir el paso del refrigerante B
M08: Abrir el paso del refrigerante A
M09: Cerrar el paso de los refrigerantes
M10: Abrir mordazas
M11: Cerrar mordazas
M13: Hacer girar el husillo en sentido horario y abrir el paso de refrigerante
M14: Hacer girar el husillo en sentido antihorario y abrir el paso de refrigerante
M31: Incrementar el contador de partes
M37: Frenar el husillo y abrir la guarda
M38: Abrir la guarda
M39: Cerrar la guarda
M62: Activar salida auxiliar 1
M65: Desactivar salida auxiliar 2
M66: Esperar hasta que la entrada 1 esté en ON
M76: Esperar hasta que la entrada 1 esté en OFF
M77: Esperar hasta que la entrada 2 esté en OFF
M98: Llamada a subprograma

Instrucciones complementarias
F: Velocidad de avance
S: Velocidad de giro del husillo
T: Tipo de Herramienta
C: Achaflanado de aristas

R: Redondeo controlado de aristas

3.2 Máquinas CNC

El Control numérico se puede aplicar a infinidad de máquinas como se muestra en la (Figura 3.4), por ejemplo en tornos, fresadoras, centros de mecanizados, rectificadoras, punzonadoras, dobladoras, máquinas de soldar, máquinas de oxicorte, máquinas de corte láser, plasma por chorro de agua, robots manipuladores, etc.

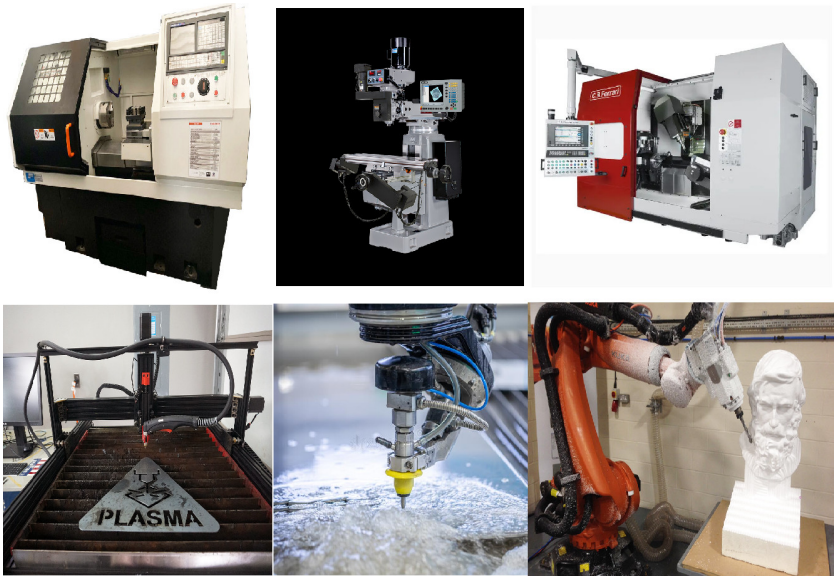


Figura 3.4: Máquinas CNC.

3.2.1 Ventajas de las máquinas CNC

Al utilizar maquinaria CNC se tienen ventajas y desventajas, haciendo una comparación de la utilización del Control Numérico con la maquinaria convencional.
Ventajas

- Reducción de tiempo de fabricación de piezas
- Utilización de un gran número de operaciones
- Reducción de personal

- Reducción de residuos
- Alta precisión
- Repetibilidad
- Elaboración de piezas complejas
- Reducción de áreas de trabajo
- Reducción de costos al producir en grandes cantidades
- Mejor productividad, de 8 a 10 veces mayor a la tecnología convencional
- Mayor seguridad para los operarios

Desventajas

- Elevados costos de adquisición de maquinaria
- Elevados costos de mantenimiento
- Genera gastos de especialización de empleados

En algunas máquinas CNC el cambio de herramientas ya viene integrado en el sistema, esto ayuda a reducir el tiempo de maquinado ya que el cambio de herramienta manualmente llevaría un mayor tiempo. Principalmente se utilizan la torreta giratoria y el carrusel, la primera se utiliza comúnmente en los tornos y la segunda en las fresadoras, (Figura 3.5).

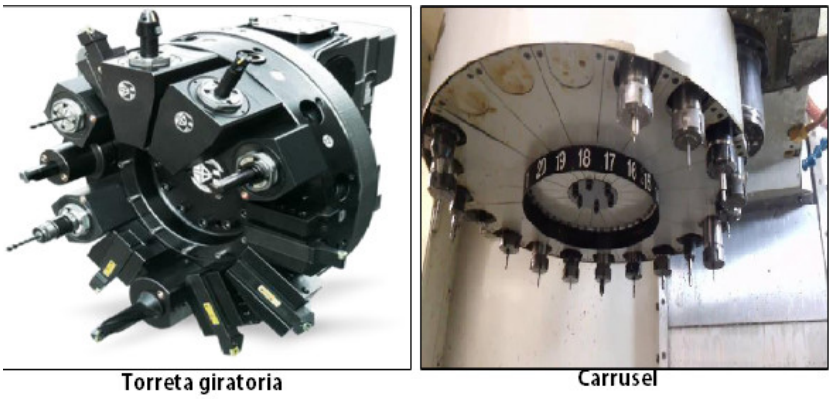


Figura 3.5: Torreta y carrusel porta herramientas.

3.3 Principio básico de programación de robots industriales

En forma general y muy compacta, los robots industriales cuentan principalmente con un controlador, estructura mecánica del robot, efector final y un *Teach Pendant*, (Figura 3.6).

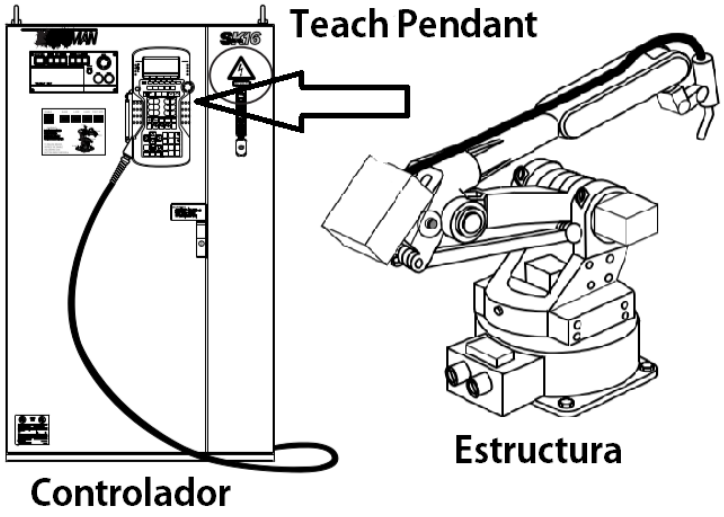


Figura 3.6: Partes principales de un robot industrial.

El controlador es el "cerebro" del robot, en él se llevan a cabo todas operaciones y cálculos de los movimientos para todas las articulaciones, etcétera, la estructura mecánica es el conjunto de eslabones, transmisiones y motores, el efector final es la herramienta a utilizar y esta depende de la aplicación, el *Teach Pendant* es un dispositivo que funciona como interfaz entre el usuario y todo el sistema del robot, por medio de este se pueden manipular las trayectorias que debe seguir la herramienta, velocidad de avance, insertar o editar programas.

Existen 4 movimientos principales: *Joint, Lineal, Circular* y *Spline* [MOTOMAN, 1998].

Joint (MOVJ): Se indica al robot moverse al punto programado de la manera más fácil posible, este movimiento nunca será en línea recta. Se utiliza principalmente para posicionar el robot cerca a la pieza de trabajo, en general para hacer movimientos donde no se esté realizando ningún trabajo. El primer y último paso se realizan con este tipo de movimiento.

Lineal (MOVL): Este tipo de movimiento indica al robot que debe moverse en línea recta al punto programado. Es un tipo de movimiento interpolado, por lo cual si el punto central de la herramienta se moverá en línea recta, sin importar si la herramienta cambia de ángulo durante su movimiento.

Circular (MOVC): este movimiento indica al robot hacer una trayectoria en forma circular. Para realizar una trayectoria de este tipo es necesario tener como mínimo 3 puntos de referencia consecutivos. Para un medio círculo se ocupan 3 puntos, para un círculo completo se ocupan 5 puntos consecutivos.

Spline (MOVS): Este tipo de movimientos indica al robot moverse en una trayectoria parabólica, se ocupan 3 puntos consecutivos similar al medio círculo. El segundo punto intersecta a la mitad a la línea que une a los puntos 1 y 3.

También existen instrucciones para poder utilizar operaciones aritméticas, timers, comentarios, *jump* (saltos de línea), condicionales IF, etc. Con estas instrucciones se pueden hacer programas un poco elaborados, sin embargo, para hacer figuras más complejas con este tipo de instrucciones estaríamos limitados.

Se dice la forma de programación de robots es punto a punto, ya que todos los movimientos se hacen con referencia a puntos, por ejemplo, para realizar las trayectorias de la (Figura 3.7), para el cuadrado se utilizan los puntos de las cuatro esquinas y se utilizan movimientos lineales para ir de un punto al otro, para el círculo se utilizan 5 puntos y se secciona el movimiento circular de forma similar para las parábolas [Álvarez, 2020].

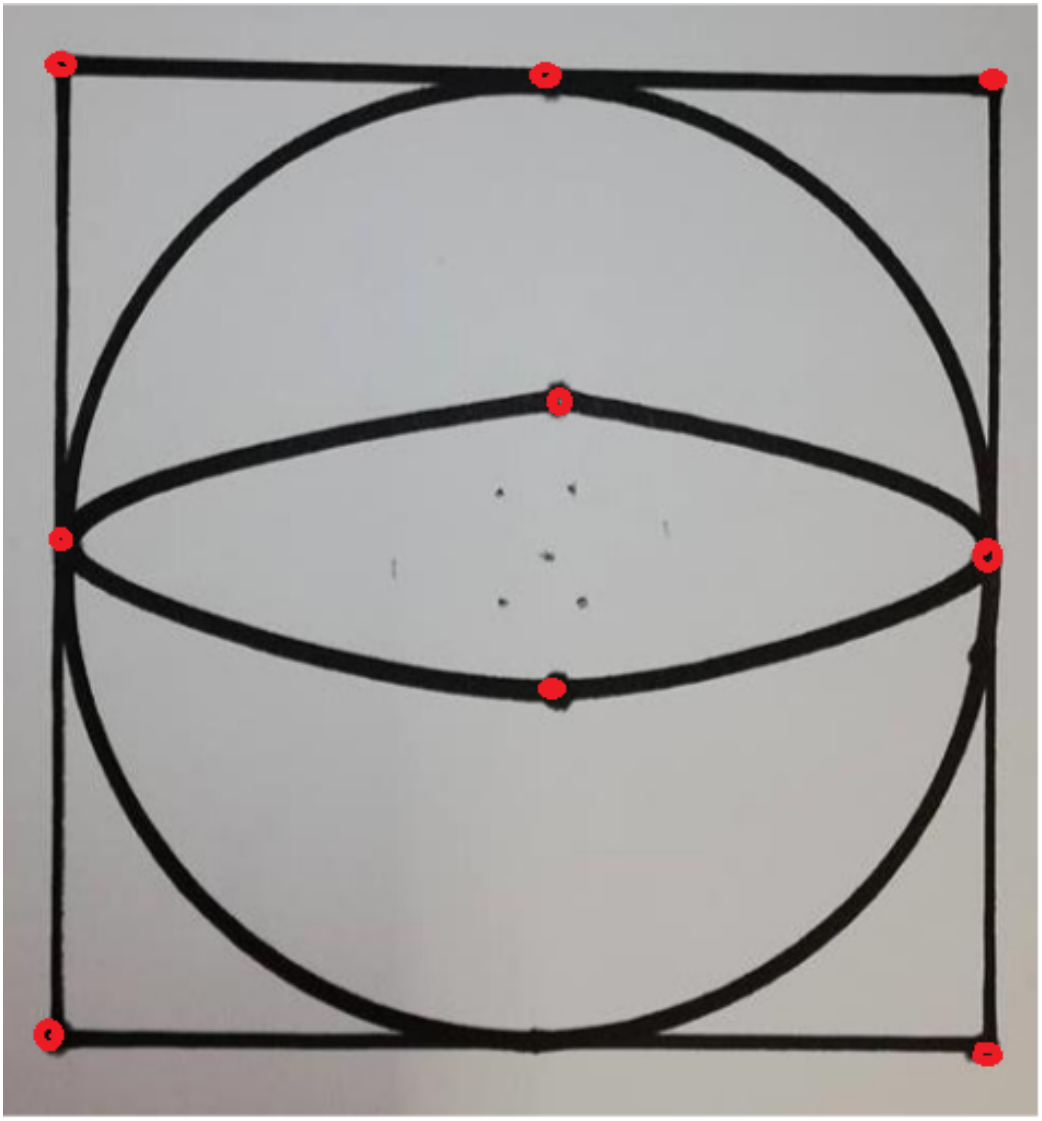


Figura 3.7: Trayectorias utilizando diferentes tipos de movimientos.
Tomada de Manual [Álvarez, 2020]

Capítulo 4

Herramientas matemáticas

En este capítulo se describirán conceptos matemáticos, principalmente los relacionados con operaciones con vectores y matrices, las cuales nos servirán para el estudio del posicionamiento y la orientación del extremo final del robot. Se da una introducción a las operaciones de traslación y rotación, sistemas de referencias, cambios de sistemas de referencia y transformaciones de matrices homogéneas. También se utiliza el software de *Matlab* para poder visualizar las operaciones realizadas y poder tener más claros los conceptos.

El lenguaje de las matemáticas es un lenguaje universal muy utilizado para describir y predecir fenómenos físicos. Sin embargo, en cualquier área de la ingeniería las matemáticas son una herramienta fundamental, y el área de la robótica no queda excuida. En la robótica se utiliza para analizar, diseñar, implementar robots, diseñar algoritmos de control y planeación de trayectorias, por eso es de suma importancia tener claro los conceptos matemáticos, [Baturone, 2005].

4.1 Operaciones con vectores y matrices

Un vector renglón de n componentes se define como un conjunto ordenado de n números y se define como la ecuación (4.1), un vector columna se define como en la ecuación (4.2). Una matriz $\mathbf{A}$ de mxn es un arreglo de la forma rectangular de mxn números acomodados en m renglones y n columnas como se muestra en (4.3) donde el símbolo mxn se lee m por n y hacen referencia al tamaño de la matriz, i y j hacen referencia a renglones y columnas respectivamente, [Grossman, 2008].

$$\vec{\mathbf{x}} = \left(x_1, x_2, ..., x_n\right) \tag{4.1}$$

$$\vec{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{4.2}$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & & a_{ij} & & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix} \tag{4.3}$$

Cabe recalcar que los vectores son tipos de matrices especiales. Por lo tanto, son aplicables las propiedades de las matrices. Si en una matriz $m = n$ se dice que es una matriz cuadrada. Una matriz donde todos sus elementos son igual a cero se denomina matriz cero de tamaño *mxn*. Una matriz **A** es igual a una matriz **B** solo si elemento a elemento son iguales y las matrices son de la misma dimensión, [Grossman, 2008].

4.1.1 Localización de componentes en matrices

Para localizar los elementos de una matriz primero se escribe el número de renglón y luego el número de columna para posteriormente localizarlo dentro de las matrices. Por ejemplo, localizando los componentes (1,2) y (3,3) se realiza como se ve en la (Figura 4.1).

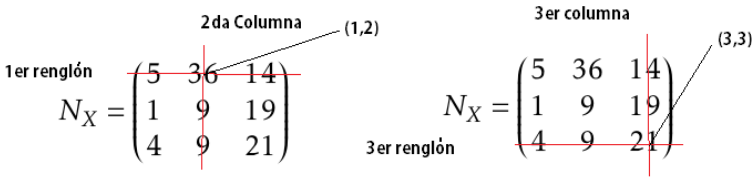


Figura 4.1: Ejemplo para localización de elementos en una matriz.

4.1.2 Suma de matrices

Si se tienen las matrices **A** y **B** de tamaño *mxn*, se puede aplicar la operación de adición, solo si las dos matrices son del mismo tamaño, se suma componente a

componente como se observa en la ecuación (4.4).

$$\mathbf{A+B} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1j}+b_{1j} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2j}+b_{2j} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1}+b_{i1} & a_{i2}+b_{i2} & & a_{ij}+b_{ij} & & a_{in}+b_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mj}+b_{mj} & \dots & a_{mn}+b_{mn} \end{pmatrix} \tag{4.4}$$

4.1.3 Mutiplicación de un escalar por una matriz

Si **A** es una matriz de *mxn* y α un escalar, la multiplicación de $\alpha\mathbf{A}$ está dada por la multiplicación de α por cada uno de los elementos de la matriz como se ve en la ecuación (4.5).

$$\alpha\mathbf{A} = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \dots & \alpha a_{1j} & \dots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \dots & \alpha a_{2j} & \dots & \alpha a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ \alpha a_{i1} & \alpha a_{i2} & & \alpha a_{ij} & & \alpha x a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \dots & \alpha a_{mj} & \dots & \alpha a_{mn} \end{pmatrix} \tag{4.5}$$

4.1.4 Producto punto o producto escalar

Sea $\vec{a}$ y $\vec{b}$ dos vectores, los cuales se colocan de modo que sus orígenes coincidan (Figura 4.2 A), formando un ángulo entre ellos, el ángulo entre dos vectores viene dado por (4.6).

$$\theta = cos^{-1}(\frac{\vec{a}\cdot\vec{b}}{||\vec{a}||||\vec{b}||}) \tag{4.6}$$

Despejando $\vec{a}\cdot\vec{b}$ obtenemos (4.7), llamado comúnmente producto punto.

$$\vec{a}\cdot\vec{b} = ||\vec{a}||||\vec{b}||cos(\theta) \tag{4.7}$$

El producto punto entre dos vectores $\vec{a}\cdot\vec{b}$ también viene dado por la ecuación (4.8), donde es aplicable para *n* dimensiones.

$$\vec{a}\cdot\vec{b} = (a_1b_1 + a_2b_2 + ... + a_nb_n) \tag{4.8}$$

La ecuación (4.7) se puede analizar tomándola como si fuera un “sensor”, donde se podrían detectar 3 cosas:

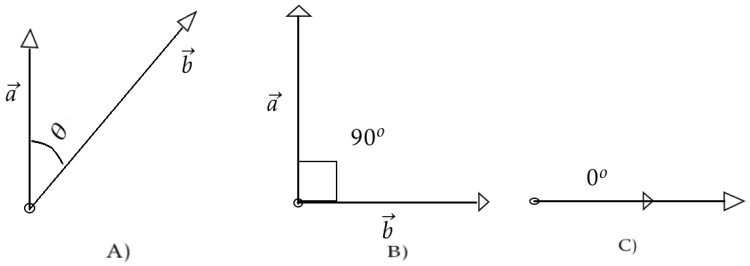


Figura 4.2: Ángulo entre vectores.

- Si existe un ángulo θ entre ellos, (Figura 4.2 A).
- Si dos vectores son ortogonales (perpendiculares) entre ellos, esto ocurre si y solo si $\vec{a} \cdot \vec{b} = 0$. Por ejemplo, cuando el ángulo entre los dos vectores es de 90^o, (Figura 4.2 B).
- Si dos vectores son paralelos entre ellos, esto ocurre si y solo si $\vec{a} \cdot \vec{b} = ||\vec{a}||||\vec{b}||$. Por ejemplo, cuando el ángulo entre los dos vectores es de 0^o o 180^o, (Figura 4.2 C).

4.1.5 Producto entre dos matrices

Sea **A** una matriz *mxn* y **B** una matriz *nxp*, el resultado del producto de **A** y **B** es una matriz **C** de *mxp* como se muestra en (4.9). Esta operación solo se puede llevar a cabo cuando el número de columnas de la primera matriz es igual al número de filas de la segunda matriz.

$$\mathbf{C} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & & a_{ij} & & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1j} & \dots & a_{1p} \\ b_{21} & b_{22} & \dots & b_{2j} & \dots & b_{2p} \\ \vdots & \vdots & & \vdots & & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nj} & \dots & b_{np} \end{pmatrix} \quad (4.9)$$

Para calcular los valores para $\mathbf{C}_{ij}$ de la matriz **C** se resuelve como si fuera el producto de dos vectores, es decir $\mathbf{C}_{ij}$ = (renglón i de **A**) $\cdot$ (columna j de **B**). El producto de **AB** es diferente a **BA**, se dice que el producto de matrices es no conmutativo.

4.2 Representaciones espaciales

Al trabajar con robots es necesario conocer la posición y orientación del efector final, conociendo estos podemos hacer infinidad de tareas, como por ejemplo mover una pieza de un lugar a otro, introducir la herramienta anclada al efector final en lugares poco accesibles, o seguir ciertas trayectorias preprogramadas. Por lo tanto, es muy importante tener un sistema de referencia.

4.2.1 Coordenadas cartesianas

Hay muchas maneras de representar un punto en el espacio, se tiene el sistema de coordenadas cartesianas, en este sistema de referencia, si se trabaja en 2 dimensiones se tendrían un vector con 2 componentes (x, y) si el punto en el espacio se encuentra en el espacio tridimensional se representará con vector con 3 componentes (x, y, z), en la (Figura 4.3) se puede observar dos sistemas cartesianos, en a) se muestra para 2 dimensiones y en b) para 3 dimensiones.

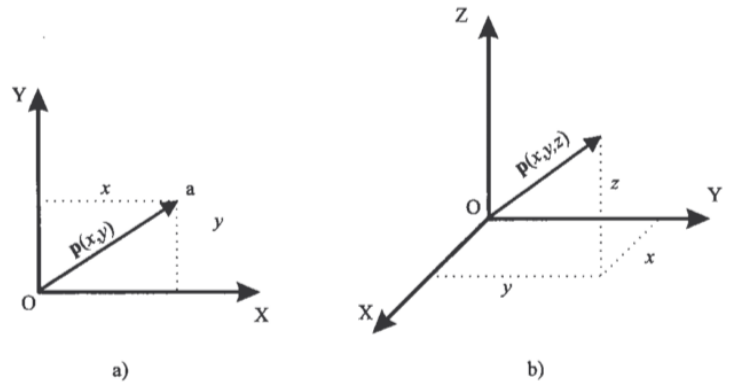


Figura 4.3: Representación crtesiana de un punto en el espacio, para 2 y 3 dimensiones.

4.2.2 Coordenadas polares o cilíndricas

Otra forma de localizar un punto en el espacio con respecto a un sistema cartesiano es mediante coordenadas polares $p(r,\theta,z)$, donde r es la proyección del vector del punto p sobre el plano xy, θ es el ángulo medido desde el eje x hasta r, z es la distancia medida a través del eje z. Del mismo modo que en las coordenadas

cartesianas existe la representación en 2 y 3 dimensiones, esto se puede observar en la (Figura 4.4) donde en a) es la representación para 2 dimensiones y b) para 3 dimensiones.

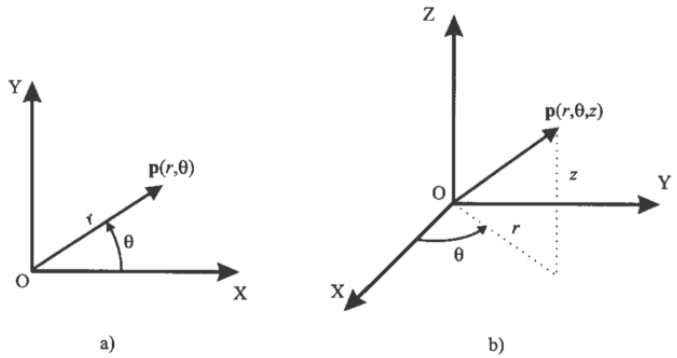


Figura 4.4: Representación polar o cilíndrica, para 2 y 3 dimensiones.

4.2.3 Coordenadas esféricas

Una forma más para representar un punto en el espacio son las coordenadas esféricas, solo que estas solo se pueden utilizar para representar puntos en el espacio tridimensional. El punto P se estará representado con $p(r, \theta, \phi)$, donde r representa la distancia del origen hasta el punto P, θ es la proyección del vector P en el plano XY y por último ϕ es el ángulo formado por el vector P y el eje z, como se muestra en la (Figura 4.5).

4.3 Matrices de transformaciones

Para tener completamente definido un punto en el espacio es necesario definir las coordenadas de posición, sin embargo, para el caso de un sólido, también es necesario especificar las coordenadas de orientación con respecto a un sistema de referencia. En los robots es necesario indicar la posición y orientación del efector final. Por ejemplo, si se requiere que el robot desbaste material para la elaboración de una escultura, en la parte donde se presentan ángulos agudos, la herramienta tiene que colocarse en una posición de tal manera que al ir desbastando siga las curvas suaves, por eso es necesario que se oriente la herramienta.

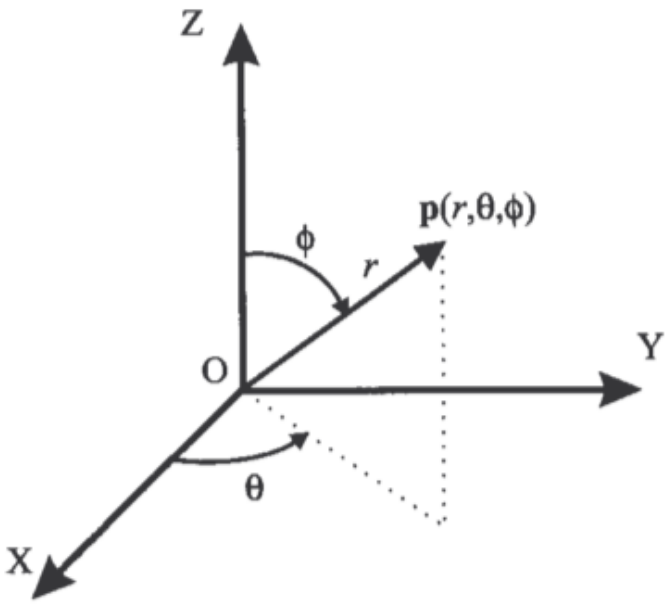


Figura 4.5: Representación esférica de un punto en el espacio.

Para describir la orientación de un sólido con respecto a un sistema de referencia, es habitual asignar un nuevo sistema de referencia al sólido para después estudiar la relación que existe entre los dos sistemas. Las matrices de rotación son la forma más utilizada para describir orientaciones respecto a diferentes sistemas de referencia dados [Rodriguez, 2007].

Supóngase que se tienen dos sistemas de referencia $\sum_0$ y $\sum_1$ ambos situados en el mismo origen O y donde el sistema $\sum_0$ estará fijo y el sistema $\sum_1$ se puede mover, como se muestra en la (Figura 4.6).

El punto P puede estar definido con respecto a cualquiera de los sistemas de referencia, por lo tanto, se tendría definido a P como:

$$P_0(x_0, y_0) = P_{x_0}.i_0 + P_{y_0}.j_0 \tag{4.10}$$

$$P_1(x_1, y_1) = P_{x_1}.i_1 + P_{y_1}.j_1 \tag{4.11}$$

En la ecuación (4.10) está definido el punto con respecto al sistema de referencia $\sum_0$ y la ecuación (4.11) está definido el punto con respecto al sistema de referencia $\sum_1$. El problema radica en encontrar la relación que existe entre las coordenadas del punto referido de un sistema de referencia a otro.

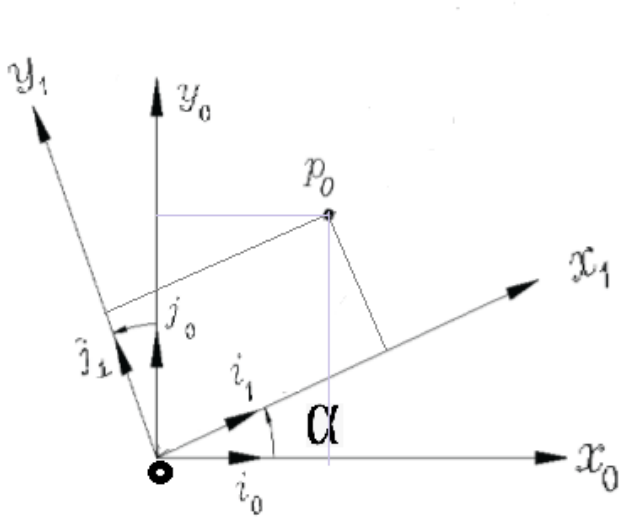


Figura 4.6: Sistemas de referencia fijo $\sum_0$ y móvil $\sum_1$ con origen común.

Analizando la (Figura 4.6) se observa que para referir de un sistema a otro es necesario multiplicar las coordenadas dadas referidas al sistema $\sum_1$ por una matriz de vectores unitarios, la cual hace referencia a cómo se encuentran proyectados los vectores unitarios del sistema $\sum_1$ con respecto del sistema fijo $\sum_0$, a esta matriz se le llama matriz de rotación **R**.

$$P_0(x_0, y_0) = \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix} = \mathbf{R} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} \tag{4.12}$$

$$\mathbf{R} = \begin{pmatrix} i_0 i_1 & i_0 j_1 \\ j_0 i_1 & j_0 j_1 \end{pmatrix} \tag{4.13}$$

La matriz de rotación para el caso de dos dimensiones está dada por la ecuación (4.14), considerando que el sistema $\sum_1$ se gira un ángulo α respecto al sistema $\sum_0$. En el caso de que $\alpha = 0$, es decir, que los dos sistemas coinciden, la matriz **R** será una matriz identidad.

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{4.14}$$

Para sistemas en 3 dimensiones, como se observa en la (Figura 4.7), se sigue el mismo procedimiento que para 2 dimensiones. Para este caso la matriz de rotación queda definida en términos de vectores unitarios como $\mathbf{R}_0^1$, donde el subíndice superior hace referencia al sistema móvil y el subíndice inferior al sistema referido.

La relación que existe entre las coordenadas referidas a un sistema de referencia con otro sistema al cual se quieren referir queda definida como en la ecuación (4.15).

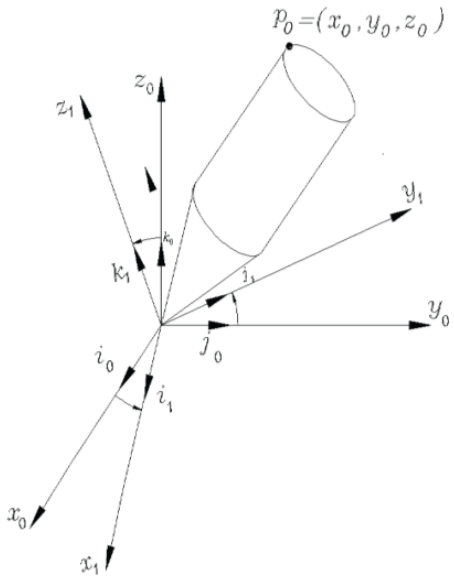


Figura 4.7: Sistemas de referencia fijo $\sum_0$ y móvil $\sum_1$ con origen común para 3 dimensiones.

Figura recuperada de [Cortés, 2020].

$$P_0(x_0, y_0, z_0) = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \mathbf{R}_0^1 \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \tag{4.15}$$

$$\mathbf{R}_0^1 = \begin{pmatrix} i_1 \cdot i_0 & j_1 \cdot i_0 & k_1 \cdot i_0 \\ i_1 \cdot j_0 & j_1 \cdot j_0 & k_1 \cdot j_0 \\ i_1 \cdot k_0 & j_1 \cdot k_0 & k_1 \cdot k_0 \end{pmatrix} \tag{4.16}$$

La matriz $\mathbf{R}_0^1$ es una matriz de transformación de coordenadas del punto P referenciado al sistema $\sum_1$ para transformarlas o referirlas al sistema $\sum_0$. En otras palabras, dadas las componentes del $P1$ referidas al sistema $\sum_1(x1, y1, z1)$, entonces $\mathbf{R}_0^1 P_1$ representa el mismo vector referenciado al sistema $\sum_0(x0, y0, z0)$, [Cortés, 2020].

4.3.1 Matrices de rotación sobre un eje principal

Las matrices de rotación sobre un solo eje o una combinación de estas son una de las herramientas más utilizadas en robótica, pues con estas se puede hacer

girar cierto ángulo un sistema de referencia respecto a otro, teniendo en claro que ambos sistemas tienen el mismo origen.

Matriz de rotación sobre los eje z_0, x_0 y y_0

Considere dos sistemas de referencia $\sum_0$ y $\sum_1$ donde el primero se mantendrá inmóvil y el segundo se gira un ángulo θ alrededor del eje z_0. Los ejes z_0 y z_1 se mantendrán paralelos como se muestra en la (Figura 4.8). Se sigue la regla de la mano derecha para determinar los ángulos.

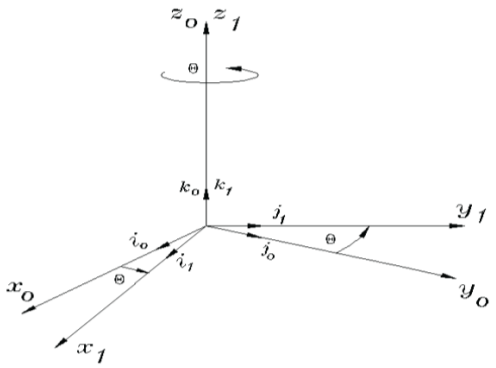


Figura 4.8: Rotación sobre el eje z_0 un ángulo θ.
Figura recuperada de [Cortés, 2020].

Proyectando los vectores unitarios del sistema $\sum_1$ en $\sum_0$ se consigue la matriz de rotación $\mathbf{R}_z(\theta)$ donde reduciendo términos podemos obtener (4.18).

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & \cos(90+\theta) & \cos(90) \\ \cos(90-\theta) & \cos(\theta) & \cos(90) \\ \cos(90) & \cos(90) & \cos(0) \end{pmatrix} \tag{4.17}$$

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.18}$$

Para obtener la matriz de rotación $\mathbf{R}_x(\theta)$ se mantienen los ejes x_0 y x_1 paralelos entre ellos como se muestra en la (Figura 4.9). Para obtener $\mathbf{R}_y(\theta)$ mantienen los ejes y_0 y y_1 paralelos entre ellos como se observa en la (Figura 4.10). Donde se obtendrán las matrices de rotación (4.19) y (4.20).

$$\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{4.19}$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{4.20}$$

Para ejemplificar el uso de las matrices de rotación se realizó un programa en

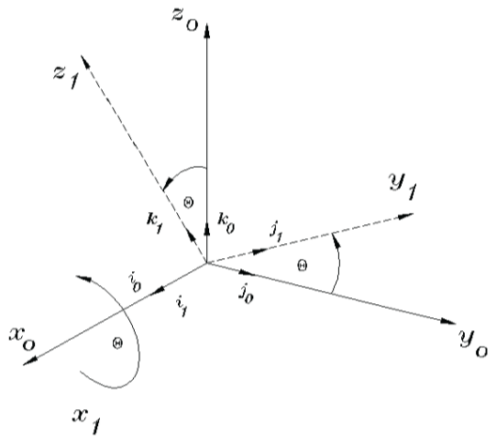


Figura 4.9: Rotacíon sobre el eje x_0 un ángulo θ.
Figura recuperada de [Cortés, 2020]

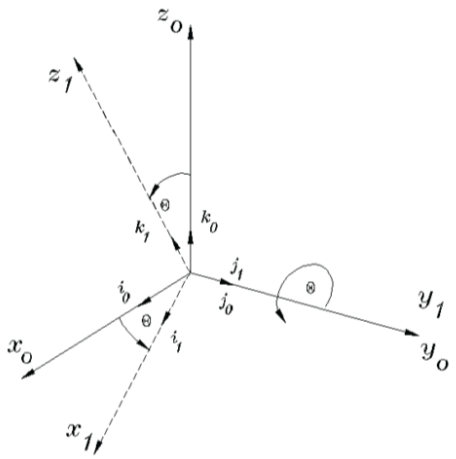


Figura 4.10: Rotacíon sobre el eje y_0 un ángulo θ.
Figura recuperada de [Cortés, 2020].

Matlab (Anexo A.1). En el cual se construyó un cubo a partir de las coordenadas que conforman los vértices de un cubo. Es un cubo de dimensión unitaria como se observa en la (Figura 4.11), luego se hace girar el cubo 15^o, 45^o y 90^o manteniendo uno de los 3 ejes fijo para realizar la rotación sobre este eje. En la (Figura

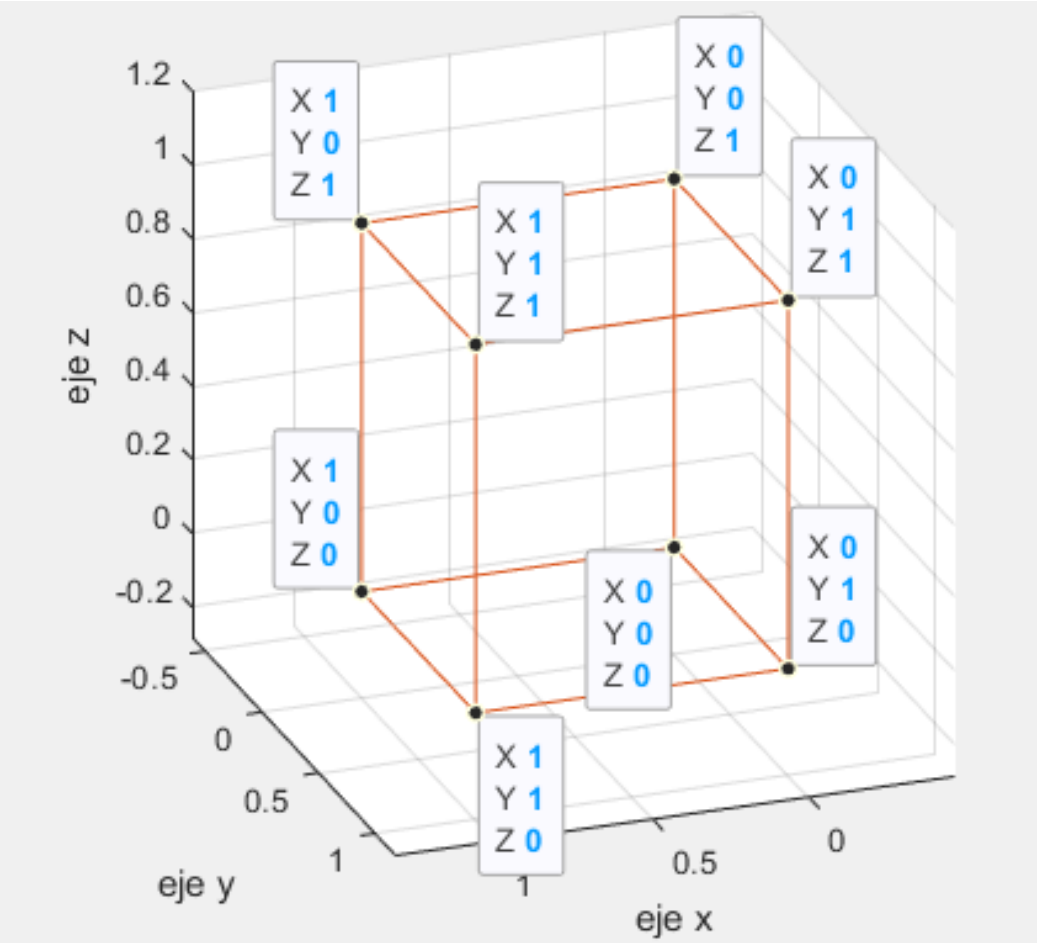


Figura 4.11: Cubo para aplicarle rotación sobre los ejes x, y y z.

4.12) se hace girar al cubo alrededor de x, en la (Figura 4.13) se hace girar al cubo alrededor de y en la (Figura 4.14) se hace girar al cubo alrededor de *z*.

En este caso solo se hizo girar al cubo sobre un solo eje, pero se pueden aplicar giros simultáneos, esto se consigue multiplicando las matrices de rotación de las rotaciones simultÁneas a realizar.

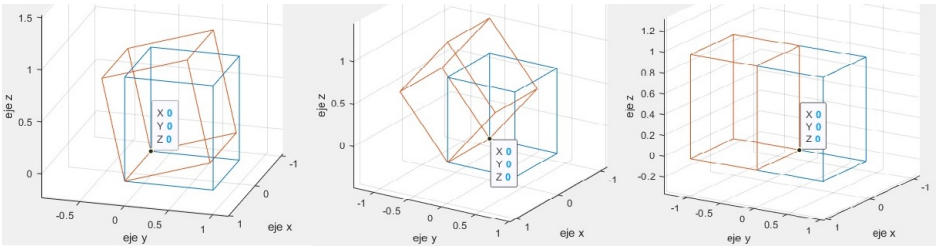


Figura 4.12: Rotación de cubo 15^o, 45^o y 90^o sobe el eje x.

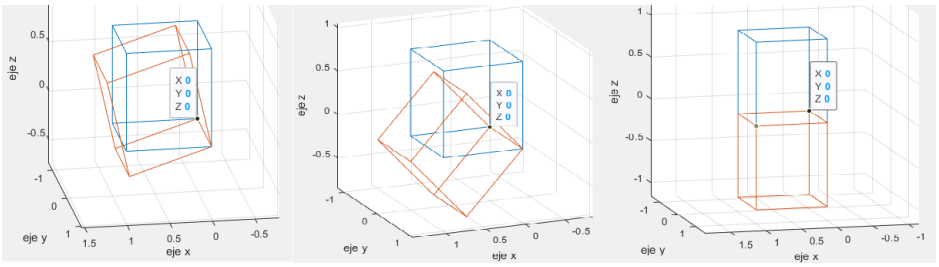


Figura 4.13: Rotación de cubo 15^o, 45^o y 90^o sobe el eje y.

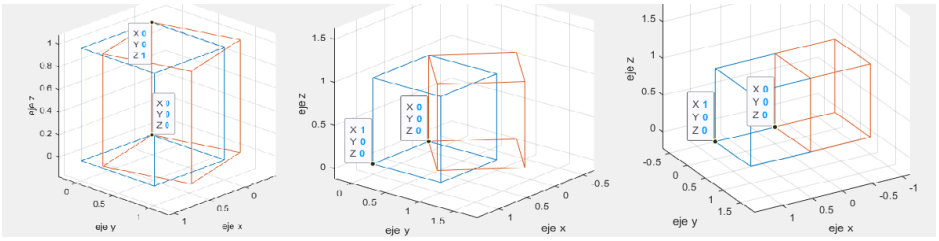


Figura 4.14: Rotación de cubo 15^o, 45^o y 90^o sobe el eje z.

4.4 Transformaciones de traslación

Considerando un sistema de referencia fijo $\sum_0$ y el sistema $\sum_1$ móvil, donde los orígenes son no coincidentes. El origen del sistema $\sum_1$ se encuentra desplazado una cantidad $d = d_0^1$ con respecto al origen del sistema $\sum_0$ como se muestra en la (Figura (4.15).

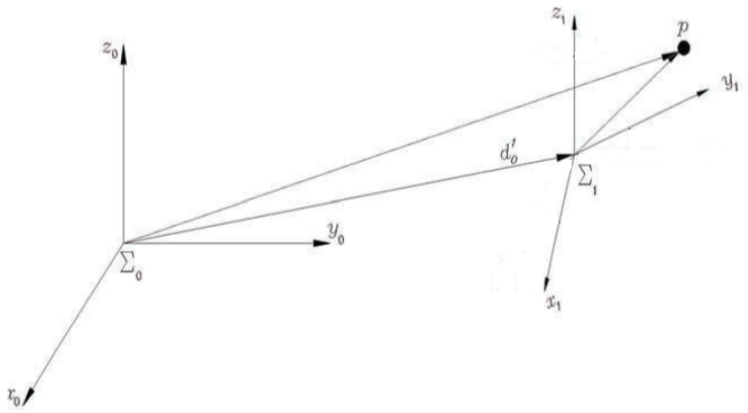


Figura 4.15: Transformaciones de traslación del sistema $\sum_1$ con respecto al sistema $\sum_0$.

Figura recuperada de [Cortés, 2020].

El punto p puede estar referenciado al sistema $\sum_0$ o al sistema $\sum_1$. El vector d_0^1 está referenciado al sistema $\sum_0$ y hace referencia al vector que va del origen del sistema $\sum_0$ al origen del sistema $\sum_1$. Por otro lado, suponiendo que los orígenes de los dos sistemas son coincidentes y el sistema $\sum_1$ está rotado cierto ángulo con respecto al sistema $\sum_0$ y el punto P está referido al sistema $\sum_1$ es decir p_1 y se requiere referirlo al sistema $\sum_0$ hace uso de las matrices de rotación. Efectuando la multiplicación del punto p_1 por una matriz de rotación se obtiene $\mathbf{R}_0^1 P_1$. Por último, sumando el vector d_0^1 con $\mathbf{R}_0^1 P_1$ se consigue la relación $d_0^1 + \mathbf{R}_0^1 P_1$. De forma general, para referenciar un punto en el espacio, referenciado a un sistema $\sum_1$ el cual se encuentra trasladado y rotado con respecto a un sistema fijo $\sum_0$ se tiene la siguiente relación (4.21).

$$P_0 = d_0^1 + \mathbf{R}_0^1 P_1 \tag{4.21}$$

$$P_0 = \begin{pmatrix} d_{0x}^1 \\ d_{0y}^1 \\ d_{0z}^1 \end{pmatrix} + \mathbf{R}_0^1 \begin{pmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{pmatrix} \tag{4.22}$$

En el caso de que se tengan 3 sistemas de referencia como, se muestra en la (Figura 4.16), se puede obtienen las siguientes expresiones:

$$P_0 = d_0^1 + \mathbf{R}_0^1 P_1 \tag{4.23}$$

$$P_1 = d_1^2 + \mathbf{R}_1^2 P_2 \tag{4.24}$$

Sustituyendo (4.24) en (4.23) se tiene:

$$P_0 = d_0^1 + \mathbf{R}_0^1 (d_1^2 + \mathbf{R}_1^2 P_2) \tag{4.25}$$

$$P_0 = d_0^1 + \mathbf{R}_0^1 d_1^2 + \mathbf{R}_0^1 \mathbf{R}_1^2 P_2 \tag{4.26}$$

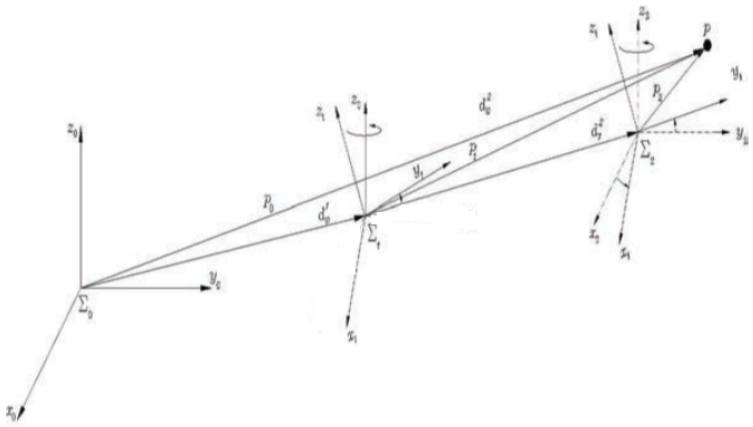


Figura 4.16: Transformaciones de traslación y rotación para 3 sistemas de referencia.

Figura recuperada de [Cortés, 2020]

4.5 Transformaciones homogéneas

Otra manera más común de representar las transformaciones de traslación y rotaciones es de forma matricial y se conoce como matriz de transformación homogénea y se representa de la siguiente manera:

$$\mathbf{H}_0^1 = \begin{pmatrix} \mathbf{R}_0^1 & d_0^1 \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} \text{Matriz de rotación} & \text{vector de traslación} \\ \text{Vector transpuesto de ceros} & 1 \end{pmatrix} \tag{4.27}$$

Donde $\mathbf{R}_0^1$ pertenece a $SO(3)$ y d_0^1 es el vector de traslación, en el último renglón se coloca el vector de ceros y al final del renglón aparece un 1 [Cortés, 2020].

Como ya hemos mencionado anteriormente, las matrices de rotación son una herramienta muy útil para orientar el extremo final de robot; sin embargo, es necesario también saber la posición del extremo final de robot. La herramienta que nos servirá para definir tanto posición y orientación de forma sencilla serán las matrices de transformaciones homogéneas (4.27).

4.5.1 Matrices de tranformación homogénea de traslación

La matriz de transformación homogénea de traslación, es decir, sin aplicar rotaciones, toma la forma de la ecuación (4.28), donde dx, dy y dz hacen referencia a los valores de desplazamiento respecto a los ejes x, y y z respectivamente.

$$\mathbf{H} = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{pmatrix} \tag{4.28}$$

Para ejemplificar el uso de esta matriz se utiliza un programa en *Matlab* (Anexo A.2). En donde se hace uso de nuevo del cubo empleado en el ejemplo de matrices de rotación, pero en este caso se hace un desplazamiento de 1, 2 y 3 unidades sobre un respectivo eje. Para el primer caso se hace un desplazamiento sobre el eje x, aplicando la multiplicacion de la matriz (4.29) con los respectivos puntos del cubo, a la variable dx se incrementa de uno en uno para poder observar como se desplaza el cubo sobre el eje, (Figura 4.17).

$$\mathbf{H}_x = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} dx \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{pmatrix} \tag{4.29}$$

Para hacer un desplazamiento únicamente sobre el eje y, dx y dz se les asigna un valor 0 y dy toma los valores 1, 2 y 3. Como se puede observar en la (Figura 4.18) el cubo se desplaza ahora únicamente sobre el eje y.

$$H_y = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 \\ dy \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{pmatrix} \tag{4.30}$$

Para hacer un desplazamiento sobre el eje z, d_x y dy se les asigna un valor 0 y d_z toma los valores 1, 2 y 3, ahora el desplazamiento se verá reflejado sobre el eje z, (Figura 4.19).

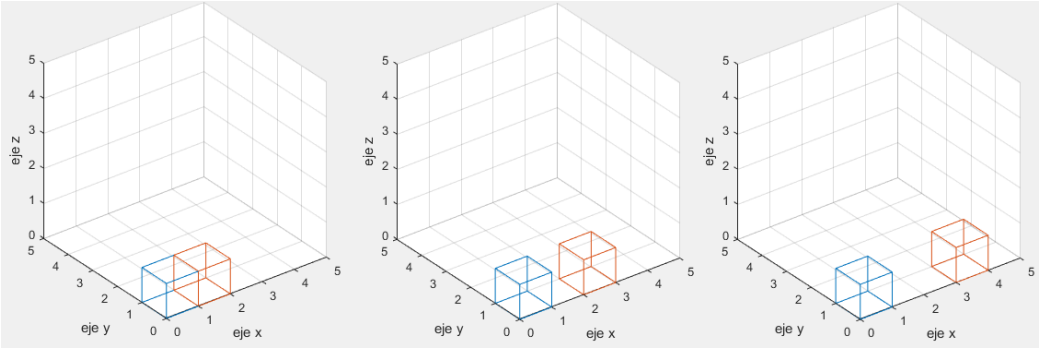


Figura 4.17: Traslación de 1, 2 y 3 unidades sobre el eje x.

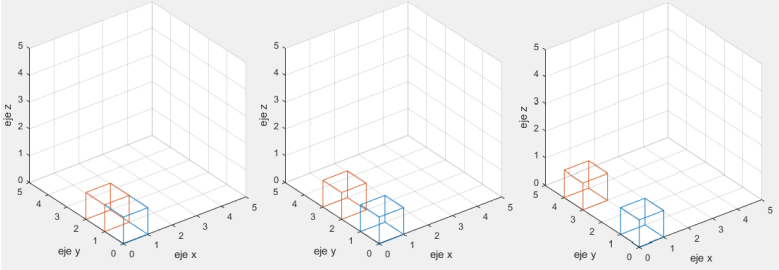


Figura 4.18: Traslación de 1, 2 y 3 unidades sobre el eje y.

$$H_z = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ dz \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} & 1 \end{pmatrix} \tag{4.31}$$

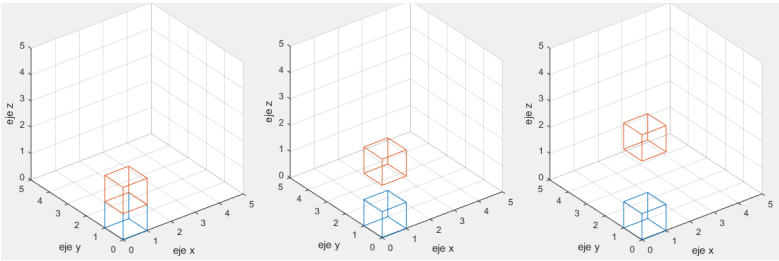


Figura 4.19: Traslación de 1, 2 y 3 unidades sobre el eje z.

La matriz de transformación homogénea se pueden conformar por una matriz de rotación sobre los ejes x, y y z, o alguna combinación de estas. También pueden contener una matriz de rotación y un vector de desplazamiento el cual hará que la

coordenada a convertir se desplace sobre los ejes x, y y z, o alguna combinación de desplazamientos en los ejes. Cabe mencionar que hay que tener cuidado al utilizarlas, ya que la multiplicación de matrices no es conmutativa. Por ejemplo, no es lo mismo rotar el cubo y luego trasladarlo, que si primero se traslada y luego se hace rotar.

Capítulo 5

Caracterización del Robot *Scorbot ER-VII*

En este capítulo se caracterizará por completo el *Scorbot ER-VII*, (Figura 5.1), para posteriormente utilizar la información obtenida para obtener el modelo matemático del robot. Este robot *Scorbot ER-VII* es uno de los robots con los que se cuenta en el laboratorio de robótica de la UACM, plantel San Lorenzo Tezonco; sin embargo, no se cuenta con su sistema de control, únicamente se tiene la estructura mecánica, sensores, motores de DC y algunos arneses para comunicación o alimentación de entradas y salidas, por lo tanto, es necesario conocer qué tipo de sensores, switches, transmisiones y motores utiliza, así también como parámetros de medidas de los eslabones.

5.1 Especificaciones generales de las articulaciones

El robot *Scorbot ER-VII* cuenta con 5 grados de libertad. El diseño de este robot permite posicionar y orientar la herramienta final dentro de un gran número de configuraciones. En la (Figura 5.2) se muestran las distintas partes que componen la estructura para el *Scorbot ER-VII*, siendo los 5 grados de libertad proporcionados por el giro de la base, hombro, codo, muñeca (inclinación) y un efector (rotación), [ROBOTEC, 1998].

5.2 Mecanismos de transmisión de movimiento

Cada articulación mostrada en la (Figura 5.2) está conformada físicamente por un motor de 12 V DC, una transmisión armónica, correas dentadas y poleas, (Figura 5.3). Las relaciones de transmisión de engranajes de cada uno de los ejes así como

Figura 5.1: Robot *Scorbot ER-VII.*

la cantidad de pulsos por revolución entregados por el encoder se muestran en la (Tabla 5.1).

CARACTERÍSTICAS GENERALES PARA *Scorbot ER-VII*		
Eje	Relación motor-polea/Transmisión-armónica	PPR
1 (Base)	3:1 / 160:1	184320
2 (Hombro)	3:1 / 160:1	184320
3 (Codo)	3:1 / 160:1	184320
4 (Muñea/Inclinación)	3:1 / 100:1	115200
5 (Efector/Rotación)	1:1 / 100:1	38400

Tabla 5.1: Relaciones de transmisiónes.

EL *Scorbot ER-VII* contiene en todas sus articulaciones un codificador incremental HEDS-5500 k11. Este es un codificador incremental óptico de dos canales de alto rendimiento, bajo costo, alta confiabilidad, alta resolución y fácil montaje. Entregan una señal cuadrada en cada canal A y B, las cuales están desfasadas 90^o entre ellas y trabajan a un voltaje de 5V. Tienen una resolución de 96 pulsos por revolución. Su rango de operación de temperatura va de -40^o a 100^o [TECHNOLOGIES, 2023].

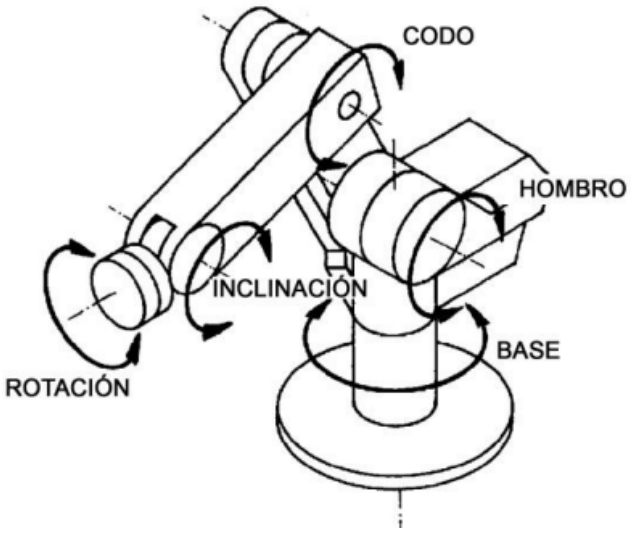


Figura 5.2: Arquitectura del *Scorbot ER-VII.*

Otro elemento importante es la transmisión armónica [ROBOTEC, 1998] la cual está conformada por 4 partes principales, (Figura 5.4).

- Anillo circular (*Circular spline*): Anillo circular anclado a la articulación del robot.
- Generador de onda (*Wave generator*): Disco rígido y liso con forma elíptica, conectada al eje del motor, éste contiene rodamientos de bolas.
- Anillo flexible (*Flexspline*): Cilindro dentado y de material flexible éste está conectado al eje de salida.
- Anillo dinámico (*Dynamic spline*): Anillo circular rígido dentado internamente.

Las relaciones de transmisión se encuentran enlistadas en la (Tabla 5.1), donde se enlistó la relación de transmisión que se tiene en cada transmisión referente a cada una de las articulaciones.

Por último, pero no menos importante se encuentran los motores. El *Scorbot ER-VII* cuenta con motores de corriente continua de imanes permanentes de DC a 12V. Para las articulaciones 1, 2 y 3 pueden alcanzar una velocidad máxima de 6151 rpm y para 4 y 5 de 5592 rpm, un par de 0.226 N·m y 0.088269 N·m respectivamente.

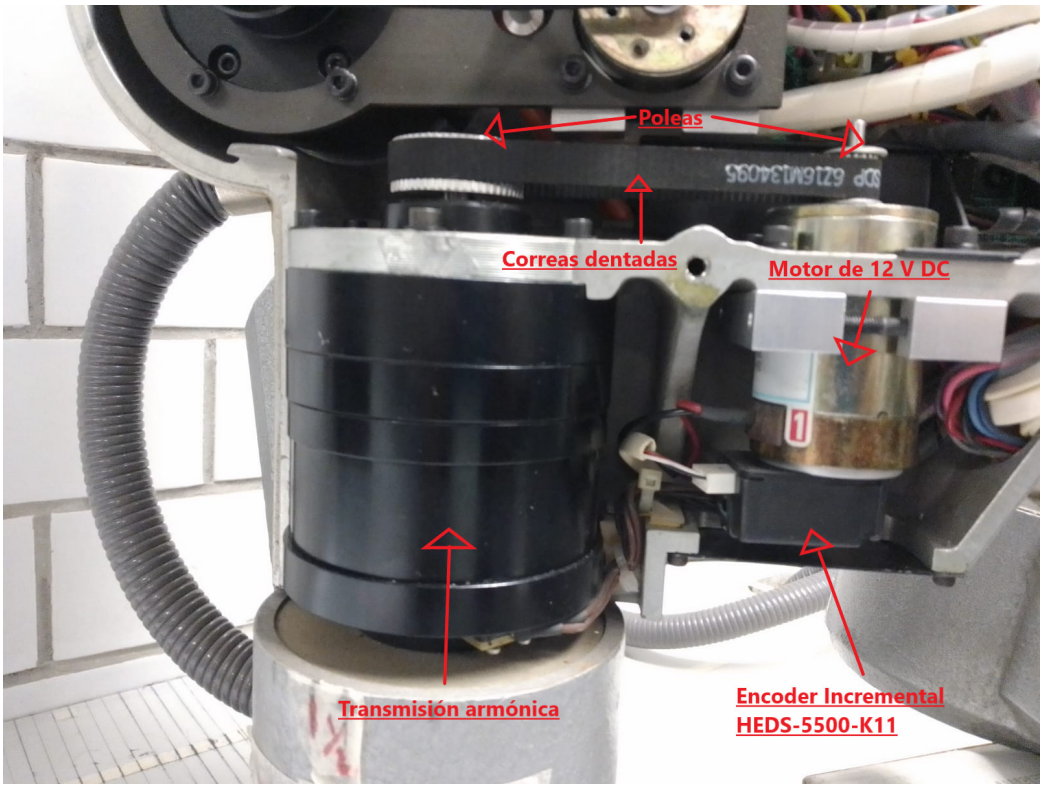


Figura 5.3: Sistema de transmisiones para las articulaciones del *Scorbot ER-VII.*

5.3 Características dimensionales

En la (Figura 5.5) se muestran las longitudes de los eslabones, las cuales son necesarias para el estudio de la cinemática del robot. También se puede observar que el área de trabajo puede cambiar dependiendo de las dimensiones de la herramienta, sin embargo, si no se toma en cuenta la herramienta, el robot podrá estirarse dentro de un radio de 695 mm.

Aunque las articulaciones pareciera que se pueden desplazar libremente la realidad es otra, los movimientos de cada articulación están limitados por la estructura mecánica y cada articulación tendrá un mayor o menor ángulo de movimiento. En la (Tabla 5.2) se enumeran la cantidad de grados que se puede desplazar cada articulación.

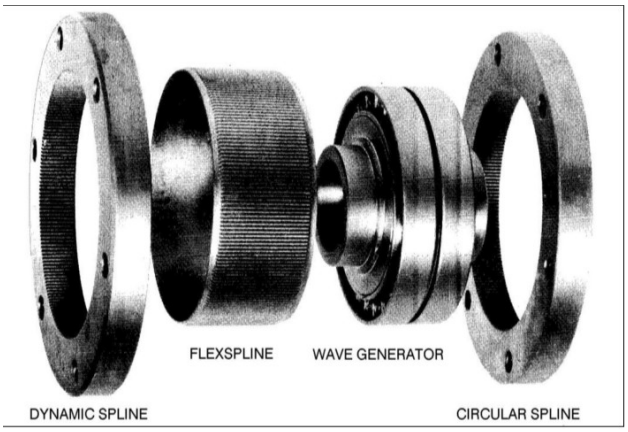


Figura 5.4: Estructura para la transmisión armónica.
Tomada del [ROBOTEC, 1998].

Movimientos de las articulaciones (Scorbot ER-VII)	
Articulación	Movimiento de articulaciónes
1 (Base)	310°
2 (Hombro)	170°
3 (Codo)	225°
4 (Muñea/Inclinación)	180°
5 (Efector/Rotación)	360°

Tabla 5.2: Tabla: Limitaciones de movimientos de articulaciones.

5.4 Switchs de límite de movimiento y Home

El *Scorbot ER-VII* utiliza interruptores (switch) que impiden que el robot se mueva más allá de sus límites. Los interruptores son parte integral de un circuito electrónico del robot y son muy independientes del sistema de control.

Los ejes del 1 al 4 utilizan 2 interruptores de límite de movimiento, el eje 5 no tiene interruptores de límite.

Los interruptores de límite están sujetos a un disco anclado al robot, cuando alguno de los interruptores es presionado el movimiento del eje se detendrá y únicamente se podrá mover cuando se invierta el movimiento del giro del motor, (Figura 5.6).

Para detectar el home se utiliza un interruptor óptico, el cual está fijado al chasis del robot como se muestra en la (Figura 5.6).

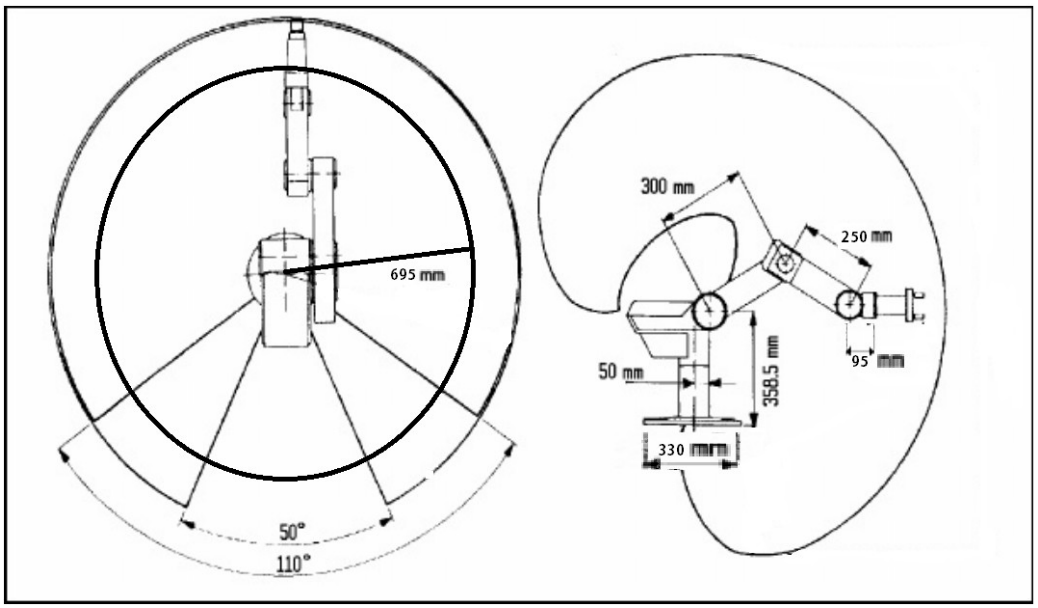


Figura 5.5: Dimensiones de articulaciones.

5.5 Identificación de pines E/S

El *Scorbot ER-VII* tiene un arnes con 6 conectores del tipo DB9 macho. Cada conector está asignado a una articulación, (Figura 5.7). En estos conectores se pueden obtener las señales de los codificadores, alimentación de codificadores, switchs de límites de movimiento y alimentación para los motores.

La asignación de pines está dada como se muestra en la (Tabla 5.3). Para todas las articulaciones se tiene la misma asignación de pines por conector DB9. Cabe mencionar que el *Home switch* se utiliza para detectar cuando la articulación se encuentra en su estado de Home. Los motores se deben alimentar a 12 V DC y los encoder a 5 V.

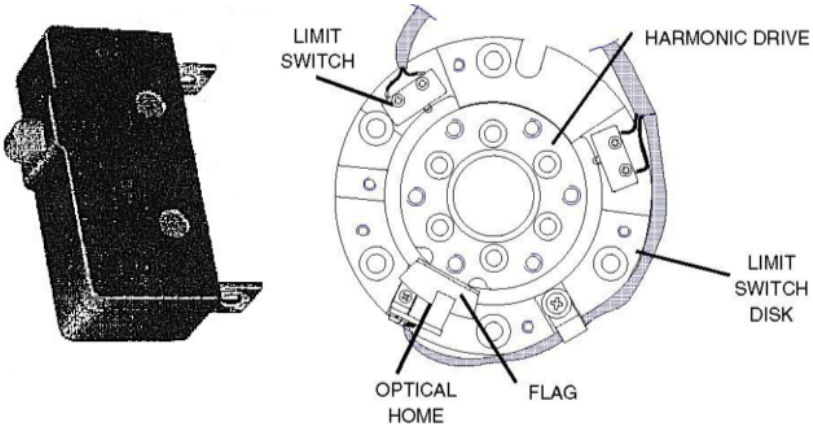


Figura 5.6: Switchs de límite de movimiento y Home.

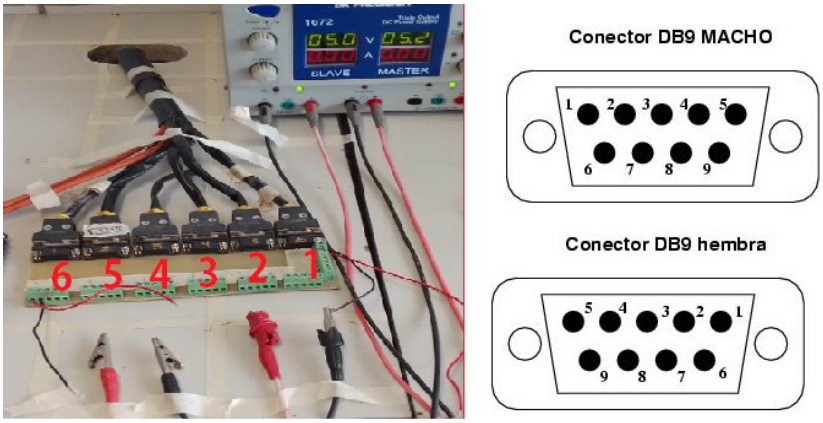


Figura 5.7: Conectores DB9 para alimentación y señales referente a cada articulación.

Asignación de pines (*Scorbot ER-VII*)		
Articualacíon	PIN	Función
1,2,3,4,5 y 6	3	+5V
	4	Home switch
	5	Común
	6	Encoder canal A
	8	Encoder canal B
	9	Alimentación motor +
	1	Alimentación motor -

Tabla 5.3: Tabla: Asignación de pines.

Capítulo 6

Cinemática directa e inversa

La cinemática es la rama de la física que estudia el movimiento de los objetos sólidos sin tomar en cuenta las fuerzas que lo originan. Esta estudia principalmente la posición, velocidad y aceleración [Rodd, 1987]. El estudio de la cinemática directa e inversa en los robots manipuladores se enfocan en estudiar el posicionamiento y orientación del efector final del robot con respecto a un sistema de referencia fijo.

Existen tres problemas principales respecto a la cinemática, los son la cinemática directa, la cinemática inversa y la relación entre estas como se muestra en la (Figura 6.1). La cinemática directa consiste en determinar la posición y orientación del efector final del robot, conocidos los valores de los ángulos de las articulaciones y los parámetros geométricos del robot. La cinemática inversa se refiere a determinar los ángulos que debe adoptar cada una de las articulaciones conocida la posición y orientación del efector final [Rodriguez, 2007].

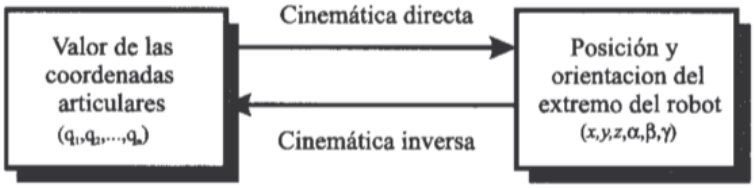


Figura 6.1: Relación entre cinemática directa e inversa.

Para resolver el problema de la cinemática directa existen muchos métodos, por ejemplo, el método geométrico, este método es muy útil cuando el robot es de pocos grados de libertad, pero a medida que se aumentan grados de libertad el

método geométrico se vuelve una herramienta complicada. Otro método es el propuesto por Denavit y Hartenberg en 1955, es un método sencillo de usar y útil para deducir las ecuaciones de cinemática directa para los robots manipuladores, donde se hace uso de las matrices de transformaciones homogéneas.

6.1 Convención *Denavit-Hartenberg*

La convención de *Denavit-Hartenberg* consiste en determinar un conjunto de parámetros relacionados con la geometría de cada uno de los eslabones del robot. Se deben escoger adecuadamente los sistemas de referencia coordenados para determinar los parámetros para cada eslabón y utilizarlos con las matrices de transformaciones homogéneas y poder determinar la posición y orientación de un robot.

Son 4 las trasformaciones básicas que se utilizan en este método, las cuales consisten en una sucesión de rotación, traslación, traslación y rotación [Cortés, 2020]. Estas transformaciones permiten referenciar el elemento i con el elemento $i-1$ siempre y cuando se cumpla que los sistemas de referencia S_{i-1} y S_i se hayan definido de acuerdo al método.

Transformaciónes D-H

- Rotación alrededor del eje z_{i-1} un ángulo θ_i
- Traslación a lo largo de z_{i-1} una distancia d_i
- Traslación a lo largo de x_i una distancia a_i
- Rotación alrededor del eje x_i un ángulo α_i

Como ya sabemos el producto de matrices no es conmutativo, cambiar el orden de transformaciones cambiara el resultado final. De forma general se tiene:

$$\mathbf{A}_{i-1}^{i} = \mathbf{R}(z_{i-1},\theta_i)\mathbf{T}(z_{i-1},d_i)\mathbf{T}(x_i,a_i)\mathbf{R}(x_i,\alpha_i) \tag{6.1}$$

$$\mathbf{A}_{i-1}^{i} = \begin{pmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta_i & -S\theta_i & 0 \\ 0 & S\theta_i & C\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.2}$$

$$\mathbf{A}_{i-1}^{i} = \begin{pmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.3}$$

La matriz (6.3) queda en términos de los parámetros θ_i, d_i, a_i y α_i los cuales son los parámetros D-H referentes al eslabón i. Para relacionar varios eslabones es cuestión de determinar los parámetros D-H para cada eslabón y sustituirlos en la matriz (6.3) para luego multiplicar sucesivamente las matrices obtenidas.

6.2 Algoritmo para determinar parámetros *Denavit-Hartenberg*

Asignación de sistemas de referencia

1. Fijar un sistema de referencia 0 en el origen del robot (base). Los ejes x_0 y y_0 se situarán de modo que formen un sistema dextrógiro con z_0, en otras palabras que se cumpla la regla de la mano derecha.

2. Enumerar los eslabones comenzando por el número 1. Se enumera como eslabón 0 a la base fija del robot.

3. Enumerar las articulaciones comenzando con 1 hasta n, donde n son los grados de libertad (GDL).

4. Localizar el eje de rotación de cada articulación. Si la articulación es del tipo prismática el eje z se asignará a lo largo del movimiento del desplazamiento (colocar en el extremo final), si la articulación es rotativa el eje z se asignará sobre el eje de giro.

5. Colocar el eje z_i empezando desde $i = 0$ hasta n-1,se coloca el eje z_i sobre la articulación $i+1$.

6. Situar los orígenes de los sistemas de referencia $\sum_i$, desde 1 hasta n-1. Encontrar una línea normal común a los ejes z_{i-1} y z_i y colocar el origen del sistema $\sum_i$ en la intersección entre el eje z_i y la normal común a los ejes z_{i-1} y z_i (infinitas posibilidades). Si ambos ejes z_{i-1} y z_i se cortasen, se colocará el sistema de referencia $\sum_i$ en el punto de corte entre la normal común y el eje de z_i. Si fuesen paralelos z_{i-1} y z_i $\sum_i$ se situará en la articulación i+1.

7. Colocar x_i sobre la línea normal cómun a z_{i-1} y z_i . Colocar y_i de forma que se cumpla la regla de la mano derecha.

8. Colocar el sistema de referencia $\sum_n$ en el extremo final del robot, de modo que z_n coincida con z_{n-1} y se cummpla que x_n sea normal a z_{i-1} y z_i.

Obtención de parámetros D-H

1. Obtener θ_i como el ángulo que hay que girar en torno a z_i para que x_{i-1} y x_i queden paralelos.

2. Obtener d_i como distancia, medida a lo largo de z_{i-1}, que habría que desplazar el sistema $\sum_{i-1}$ para que x_{i-1} y x_i queden alineados.

3. Obtener a_i como la distancia, medida a lo largo de x_i que habría que desplazar $\sum_{i-1}$ para que su origen coincida con $\sum_i$.

4. Obtener α_i como el ángulo que habría que girar en torno a x_i, desde z_{i-1} hasta que coincida con el eje z_i.

NOTA: Realizar los giros y desplazamientos de los sistemas en el mismo orden de lista anterior para poder encontrar θ_i, d_i, a_i y α_i consecutivamente.

6.2.1 Cinemática directa para robot Planar (2 GDL) y *Scorbot ER-VII*

Utilizando el algoritmo D-H se resolverá la cinemática directa para un robot planar de 2 GDL y el robot Scorbot ER-VII de 5 GDL.

Siguiendo el algoritmo D-H para asignar los sistemas de referencia adecuados, se asignan como se muestra en la (Figura 6.2), luego se obtienen los parámetros θ_i, d_i, a_i y α_i y son asignados como en la (Tabla 6.1).

Parámetros D-H para Robot planar 2 GDL				
Articulación	θ_i	d_i	a_i	α_i
1	θ_1	0	a_1	0
2	θ_2	0	a_2	0

Tabla 6.1: Parámetros D-H para Robot planar 2 GDL.

Remplazando los valores de la (Tabla 6.1) en las matrices de trasformaciones D-H, en la matriz (6.3) se obtienen.

$$\mathbf{H}_1 = \begin{pmatrix} C\theta_1 & -S\theta_1 & 0 & a_1 C\theta_1 \\ S\theta_1 & C\theta_1 & 0 & a_1 S\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.4}$$

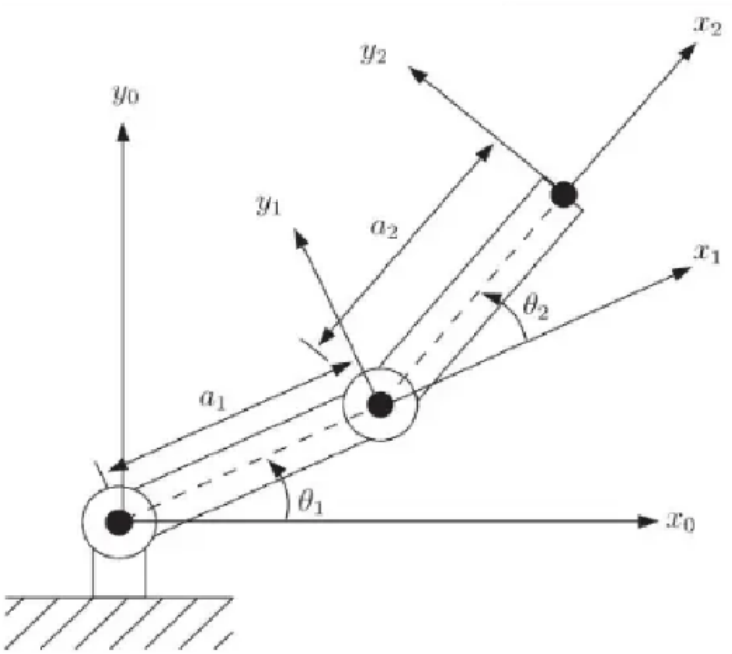


Figura 6.2: Robot planar de 2 GDL.

$$\mathbf{H}_2 = \begin{pmatrix} C\theta_2 & -S\theta_2 & 0 & a_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.5}$$

$$\mathbf{H}_1^2 = \mathbf{H}_1\mathbf{H}_2 = \begin{pmatrix} C(\theta_1+\theta_2) & -S(\theta_1+\theta_2) & 0 & a_1C\theta_1 + a_2C(\theta_1+\theta_2) \\ S(\theta_1+\theta_2) & C(\theta_1+\theta_2) & 0 & a_1S\theta_1 + a_2S(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.6}$$

Donde $C_i = cos(\theta_i)$ y $S_i = sin(\theta_i)$

La cinemática directa para el robot planar de 2GDL queda conformado por las componentes de la última columna de la matriz (6.6):

$$X(\theta) = a_1C\theta_1 + a_2C(\theta_1+\theta_2) \tag{6.7}$$

$$Y(\theta) = a_1S\theta_1 + a_2S(\theta_1+\theta_2) \tag{6.8}$$

$$Z(\theta) = 0 \tag{6.9}$$

6.2.2 Cinemática directa para *Scorbot ER-VII*

El modelo *Scorbot ER-VII* de *Intelitek*, (Figura 6.3), es un robot de 5 grados de libertad. Es un robot industrial para múltiples aplicaciones, desarrollado principalmente para la enseñanza en el aula.

Figura 6.3: Robot *Scorbot ER-VII* de 5 GDL.

Tomando las dimensiones del *Scorbot ER-VII* mostradas en la (Figura 5.5) y colocando los sistemas de referencia de acuerdo al algoritmo D-H (Figura 6.4) se obtienen los parámetros D-H mostrados en la (Tabla 6.2).

Remplazando los valores de la (Tabla 6.1) en las matrices de trasformaciones DH, en la matriz (6.3) se obtienen.

$$\mathbf{H}_1 = \begin{pmatrix} C\theta_1 & 0 & -S\theta_1 & L_2C\theta_1 \\ S\theta_1 & 0 & C\theta_1 & L_2S\theta_1 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.10}$$

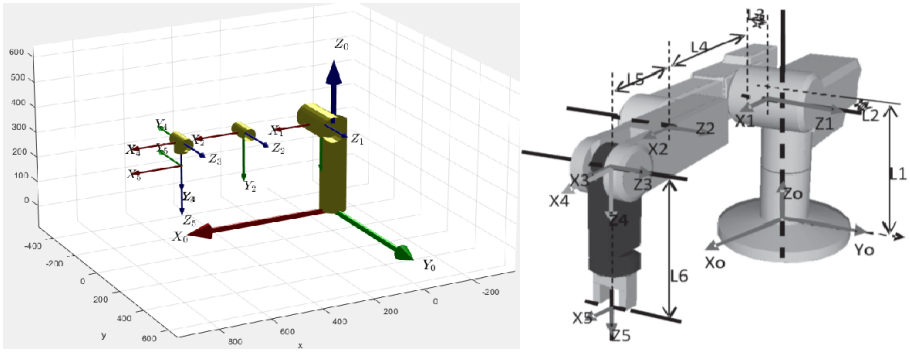


Figura 6.4: Asignación de ejes de referencia para Robot *Scorbot ER-VII.*

Parámetros D-H para Robot Scorbot ER-VII				
Articulación	θ_i	$d_i(mm)$	a_i	α_i
1	θ_1	$L_1 = 358.5$	$L_2 = 50$	-pi/2
2	θ_2	$-L_3 = -36$	$L_4 = 300$	0
3	θ_2	0	$L_5 = 250$	0
4	θ_2	0	0	-pi/2
5	θ_2	$L_6 = 95$	0	0

Tabla 6.2: Parámetros D-H para Robot *Scorbot ER-VII.*

$$\mathbf{H}_2 = \begin{pmatrix} C\theta_2 & -S\theta_2 & 0 & L_4C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & L_4S\theta_2 \\ 0 & 0 & 1 & -L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.11}$$

$$\mathbf{H}_3 = \begin{pmatrix} C\theta_3 & -S\theta_3 & 0 & L_5C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & L_5S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.12}$$

$$\mathbf{H}_4 = \begin{pmatrix} C\theta_4 & 0 & -S\theta_4 & 0 \\ S\theta_4 & 0 & C\theta_4 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.13}$$

$$\mathbf{H}_5 = \begin{pmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ S\theta_5 & C\theta_5 & 0 & 0 \\ 0 & 0 & 1 & L_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.14}$$

$$\mathbf{T}_0^5 = \begin{pmatrix} s_1.s_5 + c_1.c_{2+3+4}.c_5 & -c_1.c_{2+3+4}.s_5 + s_1.c_5 & -c_1.s_{2+3+4} & -L_6.c_1.s_{2+3+4} + L_3.s_1 + \\ & & & c_1.(L_5.c_{2+3} + L_4.c_2) + L_2.c_1 \\ s_1.c_{2+3+4}.c_5 - c_1.s_5 & -c_1.c_5 - c_{2+3+4}.s_1.s_5 & -s_1.s_{2+3+4} & -s_1.s_{2+3+4}.L_6 - L_3.c_1 + \\ & & & s_1.(L_5.c_{2+3} + L_4.c_2) + L_2 s_1 \\ -s_{2+3+4}.c_5 & s_{2+3+4}.s_5 & -c_{2+3+4} & L_1 - L_5.s_{2+3} - L_4.s_2 - L_6.c_{2+3+4} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.15}$$

La cinemática directa para el *Scorbot ER-VII* queda conformado por las componentes de la última columna de la matriz (6.15):

$$X(\theta) = -L_6.c_1.s_{2+3+4} + L_3.s_1 + c_1.(L_5.c_{2+3} + L_4.c_2) + L_2.c_1 \tag{6.16}$$

$$Y(\theta) = -s_1.s_{2+3+4}.L_6 - L_3.c_1 + s_1.(L_5.c_{2+3} + L_4.c_2) + L_2 s_1 \tag{6.17}$$

$$Z(\theta) = L_1 - L_5.s_{2+3} - L_4.s_2 - L_6.c_{2+3+4} \tag{6.18}$$

6.3 Cinemática inversa

La cinemática inversa consiste en determinar la configuración que debe tomar cada articulación, es decir, el ángulo de giro para conseguir posicionar y orientar el efector final [Arias and Fonseca, 2012]. En otras palabras, conocidos la posición (P_X, P_Y, P_Z) y los ángulos de orientación del efector final (α, β, γ), se puede determinar los ángulos θ que debe girar cada articulación para posicionar y orientar la herramienta.

La solución de la cinemática inversa es un punto esencial para el estudio o diseño de los robots, es un proceso difícil, ya que se pueden encontrar múltiples soluciones debido a que se hace uso de ecuaciones cinemáticas no lineales, [Kumar Saha, 2010]. Existen diversos métodos, los más utilizados son el método geométrico, algebraico y el método por desacople [Garay Molina et al., 2001]. Para encontrar la solución para el robot planar emplearemos el método geométrico y para el *Scorbot ER-VII* se optó por el método algebraico, ya que debido a la configuración mecánica y el número de grados de libertad el método geométrico se vuelve complicado.

6.3.1 Cinemática inversa para robot planar de 2 GDL

Partiendo de que son conocidas las coordenadas (P_x, P_y) y se requiere obtener los valores de nuestras articulaciones θ_1 y θ_2 para llegar al punto de interés, se puede observar en la (Figura 6.5) que se pueden llegar a obtener 2 posibles soluciones. Analizaremos el caso de la configuración codo abajo. Haciendo uso del teorema de Pitágoras se encuentra C, donde este es la hipotenusa del triángulo remarcado con rojo como se muestra en la (Figura 6.6) paso seguido trazamos un triángulo

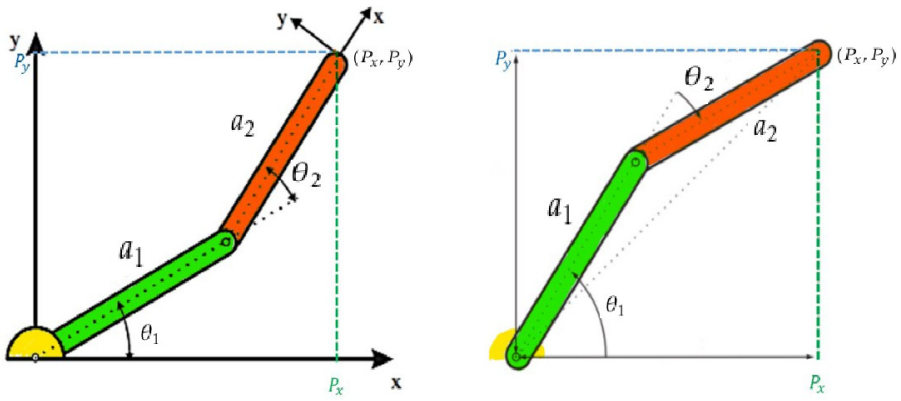


Figura 6.5: Posibles configuraciones para un robot planar de 2 GDL.

con las articulaciones donde se forman los ángulos β, γ y α, (Figura 6.6). Utilizando la ley de los cosenos se tiene:

$$C^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\gamma) \tag{6.19}$$

$$-\cos(\gamma) = \frac{C^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{6.20}$$

$$-\cos(\gamma) = \frac{(P_x^2 + P_y^2) - a_1^2 - a_2^2}{2a_1a_2} \tag{6.21}$$

Observando la (Figura 6.6) encontramos la siguiente relación $\gamma = \pi - \theta_2$ y sabiendo que la funcíon cos es par, se tiene:

$$-\cos(\gamma) = -\cos(\pm\pi - \theta_2) \tag{6.22}$$

$$-\cos(\gamma) = -\cos(-\theta_2) \tag{6.23}$$

$$\cos(\gamma) = \cos(\theta_2) \tag{6.24}$$

Sustituyendo (6.21) en (6.24)

$$\cos(\theta_2) = \frac{(P_x^2 + P_y^2) - a_1^2 - a_2^2}{2a_1a_2} \tag{6.25}$$

Haciendo uso de funciones trigonométricas y despejando θ_2 obtenemos:

$$\theta_2 = \arctan\left(\frac{\sqrt{(1 - (\frac{P_x^2+P_y^2-a_1^2-a_2^2}{2a_1a_2})^2}}{\frac{(P_x^2+P_y^2)-a_1^2-a_2^2}{2a_1a_2}}\right) \tag{6.26}$$

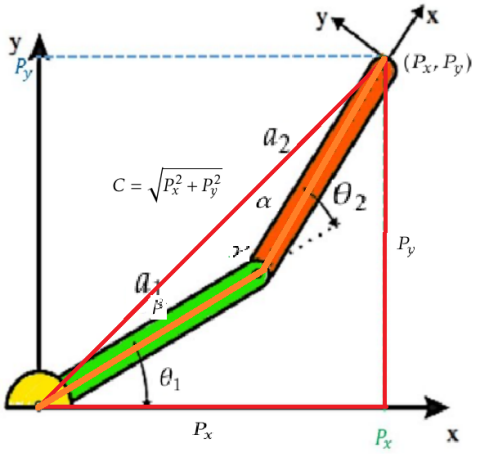


Figura 6.6: Configuración codo abajo.

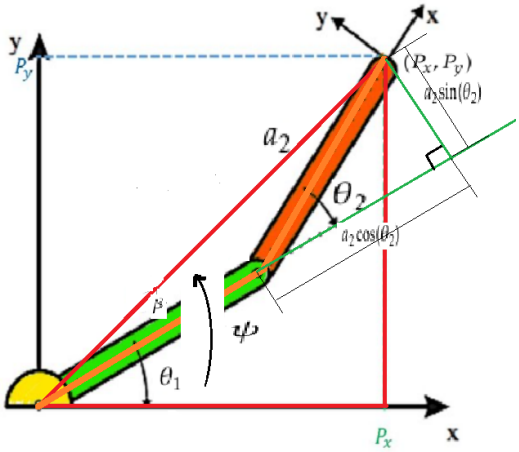


Figura 6.7: Triángulos formados para encontrar θ_1.

Para obtener θ_1 nos ayudamos de la (Figura 6.7) y utilizando el teorema de Pitágoras obtenemos la relación de ángulos siguiente $\theta_1 = \psi - \beta$ donde:

$$\psi = \arctan(\frac{P_y}{P_x}) \tag{6.27}$$

$$\theta_1 = \arctan(\frac{P_y}{P_x}) - \beta \tag{6.28}$$

β se obtiene de manera sencilla utilizando las proyecciones del eslabón a_2

$$\beta = \arctan(\frac{a_2 \sin(\theta_2)}{a_1 + a_2 \cos(\theta_2)}) \tag{6.29}$$

Sustituyendo (6.29) en (6.28)

$$\theta_1 = \arctan(\frac{P_y}{P_x}) - \arctan(\frac{a_2 \sin(\theta_2)}{a_1 + a_2 \cos(\theta_2)}) \tag{6.30}$$

6.3.2 Cinemática inversa para el robot *Scorbot ER-VII*

Para encontrar el modelo cinemático inverso se recurre al método algebraico. Partiendo de la matriz de transformación homogénea (6.15) la cual expresada de forma genérica se representa de la siguiente manera:

$$\mathbf{T}_0^5 = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{H}_1.\mathbf{H}_2.\mathbf{H}_3.\mathbf{H}_4.\mathbf{H}_5 = \mathbf{T}_0^1.\mathbf{T}_1^2.\mathbf{T}_2^3.\mathbf{T}_3^4.\mathbf{T}_4^5 \tag{6.31}$$

De forma general, una matriz de transformaciones homogéneas inversa está dada por:

$$\mathbf{H}^{-1} = \begin{pmatrix} n_x & n_y & n_z & -[n_x\ n_y\ n_z]^T[p_x\ p_y\ p_z] \\ o_x & o_y & o_z & -[o_x\ o_y\ o_z]^T[p_x\ p_y\ p_z] \\ a_x & a_z & ya_z & -[a_x\ a_y\ a_z]^T[p_x\ p_y\ p_z] \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.32}$$

Por lo tanto, obtenenemos $(\mathbf{T}_0^1)^{-1}$

$$(\mathbf{T}_0^1)^{-1} = \begin{pmatrix} cos(\theta_1) & sin(\theta_1) & 0 & -L_2 \\ 0 & 0 & -1 & L_1 \\ -sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.33}$$

Multiplicando (6.15) por la inversa de $(\mathbf{T}_0^1)^{-1} = \mathbf{T}_1^0$ se obtiene $\mathbf{T}_1^5$

$$\mathbf{T}_1^5 = \begin{pmatrix} c_{2+3+4}.c_5 & -c_{2+3+4}.s_5 & -s_{2+3+4} & -L_6.s_{2+3+4} + L_5.c_{2+3} + L_4.c_2 \\ s_{2+3+4}.c_5 & -s_{2+3+4}.s_5 & c_{2+3+4} & L_6.c_{2+3+4} + L_5.s_{2+3} + L_4.s_2 \\ -s_5 & -c_5 & 0 & -L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.34}$$

Por otro lado, multiplicamos (6.31) por la inversa de $(\mathbf{T}_0^1)^{-1} = \mathbf{T}_1^0$

$$\mathbf{T}_1^5 = \begin{pmatrix} cos(\theta_1) & sin(\theta_1) & 0 & -L_2 \\ 0 & 0 & -1 & L_1 \\ -sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.35}$$

$$\mathbf{T}_{15} = \begin{pmatrix} n_x.c_1 + n_y.s_1 & o_x.c_1 + o_y.s_1 & a_x.c_1 + a_y.s_1 & p_x.c_1 - a_1 + p_y.s_1 \\ -n_z & -o_z & -a_z & L_1 - p_z \\ n_y.c_1 - n_x.s_1 & o_y.c_1 - o_x.s_1 & a_y.c_1 - a_x.s_1 & p_y.c_1 - p_x.s_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.36}$$

Para encontrar las ecuaciones que relacione los ángulos de las articulaciones con las coordenadas (P_x, P_y, P_z) y los ángulos de orientación se procede a igualar (6.34) y (6.36) de la siguiente forma:

$$\begin{pmatrix} n_x.c_1 + n_y.s_1 & o_x.c_1 + o_y.s_1 & a_x.c_1 + a_y.s_1 & p_x.c_1 - a_1 + p_y.s_1 \\ -n_z & -o_z & -a_z & L_1 - p_z \\ n_y.c_1 - n_x.s_1 & o_y.c_1 - o_x.s_1 & a_y.c_1 - a_x.s_1 & p_y.c_1 - p_x.s_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c_{2+3+4}.c_5 & -c_{2+3+4}.s_5 & -s_{2+3+4} & -L_6.s_{2+3+4} + L_5.c_{2+3} + L_4.c_2 \\ s_{2+3+4}.c_5 & -s_{2+3+4}.s_5 & c_{2+3+4} & L_6.c_{2+3+4} + L_5.s_{2+3} + L_4.s_2 \\ -s_5 & -c_5 & 0 & -L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Igualando los elementos entre ambas matrices, iniciamos tomamos el elemento de la tercera fila cuarta columna y despejando todo lo que dependa de θ_1 obtenemos:

$$-p_x.s_1 + P_y.c_1 = -L_3 \tag{6.37}$$

Para encontrar θ_1 se hacen las siguientes sustituciones trigonométricas

$$p_x = \beta.cos(\eta) \tag{6.38}$$

$$p_y = \beta.sin(\eta) \tag{6.39}$$

$$\beta = \sqrt{p_x^2 + p_y^2} \tag{6.40}$$

$$\eta = atan2(p_y, p_x) \tag{6.41}$$

Remplazando (6.38) y (6.39) en (6.37) se tiene:

$$-\beta.cos(\eta).s_1 + \beta.sin(\eta).c_1 = -L_3 \tag{6.42}$$

$$-cos(\eta).s_1 + sin(\eta).c_1 = \frac{-L_3}{\beta} \tag{6.43}$$

Se tienen las siguientes relaciones trigonométricas

$$sin(\ominus^{+}_{-}\eta) = sin(\ominus).cos(\eta)^{+}_{-}cos(\ominus).sin(\eta) \tag{6.44}$$

$$cos(\ominus^{+}_{-}\eta) = cos(\ominus).cos(\eta)^{-}_{+}sin(\ominus).sin(\eta) \tag{6.45}$$

$$(cos(\ominus))^2 + (sin(\ominus))^2 = 1 \tag{6.46}$$

La ecuación (6.43) se reduce de la siguiente forma

$$sin(\eta - \theta_1) = \frac{-L_3}{\beta} \tag{6.47}$$

$$cos(\eta - \theta_1) =^{+}_{-} \sqrt{1 - (\frac{-L_3}{\beta})^2} \tag{6.48}$$

Luego:

$$tan(\eta - \theta_1) = \frac{sin(\eta - \theta_1)}{cos(\eta - \theta_1)} = \frac{\frac{-L_3}{\beta}}{^{+}_{-}\sqrt{1 - (\frac{-L_3}{\beta})^2}} \tag{6.49}$$

$$\eta - \theta_1 = atan2(\frac{-L_3}{\beta},^{+}_{-} \sqrt{1 - (\frac{-L_3}{\beta})^2}) \tag{6.50}$$

Despejando θ_1 y sustituyendo (6.41) en (6.50) obtenemos 2 soluciones, dependiendo el signo que se tome.

$$\theta_1 = atan2(p_y, p_x) - atan2(-L_3,^{+}_{-} \sqrt{p_x^2 + p_y^2 - L_3^2} \tag{6.51}$$

Conociendo los valores de p_x, p_y y L_3 se obtiene fácilmente el valor de θ_1.
Para determinar el valor de θ_5 se utiliza el elemento (3,1) y (3,2) se obtienen

$$-n_x.sin(\theta_1) + n_y.cos(\theta_1) = -sin(\theta_5) \tag{6.52}$$

$$-o_x.sin(\theta_1) + o_y.cos(\theta_1) = -cos(\theta_5) \tag{6.53}$$

$$tan(\theta_5) = \frac{sin(\theta_5)}{cos(\theta_5)} \tag{6.54}$$

Se sustituye (6.52) y (6.53) en (6.54) y se despeja θ_5

$$\theta_5 = atan2(n_x.sin(\theta_1) - n_y.cos(\theta_1), o_x.sin(\theta_1) - o_y.cos(\theta_1)) \tag{6.55}$$

Conocidos los valores para θ_1 y θ_5 se procede a calcular $(\theta_2+\theta_3+\theta_4)$, se utilizan los elementos (1,1) y (2,1)

$$cos(\theta_2+\theta_3+\theta_4) = \frac{n_x.cos(\theta_1)+n_y.sin(\theta_1)}{cos(\theta_5)} \tag{6.56}$$

$$sin(\theta_2+\theta_3+\theta_4) = \frac{-n_z}{cos(\theta_5)} \tag{6.57}$$

$$tan(\theta_2+\theta_3+\theta_4) = \frac{\frac{-n_z}{cos(\theta_5)}}{\frac{n_x.cos(\theta_1)+n_y.sin(\theta_1))}{cos(\theta_5)}} \tag{6.58}$$

Se obtiene:

$$\theta_2+\theta_3+\theta_4 = atan2(\frac{-n_z}{cos(\theta_5)}, \frac{n_x.cos(\theta_1)+n_y.sin(\theta_1)}{cos(\theta_5)}) \tag{6.59}$$

Para obtener el valor de θ_3 se toman los elementos (1,4) y (2,4)

$$-L_6.s_{2+3+4}+L_5.c_{2+3}L_4.c_2 = p_x.c_1-L_2+p_y.s_1 \tag{6.60}$$

$$L_6.c_{2+3+4}+L_5.s_{2+3}L_4.s_2 = L_1-p_z \tag{6.61}$$

Se hace un cambio de variable para simplificar cálculos

$$k_1 = L_5.c_{2+3}+L_4.c_2 \tag{6.62}$$

$$k_1 = p_x.c_1-L_2+p_y.s_1+L_6.s_{2+3+4} \tag{6.63}$$

$$k_2 = L_5.s_{2+3}+L_4.s_2 \tag{6.64}$$

$$k_2 = L_1-p_z-L_6.c_{2+3+4} \tag{6.65}$$

Luego se elevan al cuadrado k_1 y k_2 se suman para posteriormente despejar $cos(\theta_3)$

$$k_1^2+k_2^2 = (L_5.s_{2+3}+L_4.s_2)^2+(L_1-p_z-L_6.c_{2+3+4})^2 \tag{6.66}$$

$$k_1^2+k_2^2 = L_5^2+L_4^2+2.L_4.L_5.(c_{2+3}.c_2+s_{2+3}.s_2) \tag{6.67}$$

Utilizando la identidad trigonométrica (6.45) se tiene:

$$cos(\theta_3) = (c_{2+3}.c_2+s_{2+3}.s_2) = cos(\theta_2+\theta_3-\theta_2) \tag{6.68}$$

Sustituyendo (6.68) en (6.66) y despejando θ_3 se tiene:

$$\theta_3 = cos^{-1}(\frac{k_1^2 + k_2^2 - L_5^2 - L_4^2}{2.L_4.L_5}) \tag{6.69}$$

Como ya se conoce θ_3 se pueden utilizar las ecuaciónes (6.62) y (6.64) para encontrar θ_2. Se hace uso de funciones trigonométricas y se obtienen

$$k_1 = L_5.(c_2.c_3 - s_2.s_3) + L_4.c_2 \tag{6.70}$$

$$k_1 = (L_5.c_3 + L_4).c_2 - (L_5.s_3).s_2 \tag{6.71}$$

$$k_2 = L_5.(s_2.c_3 + c_2.s_3) + L_4.s_2 \tag{6.72}$$

$$k_2 = (L_5.c_3 + L_4).s_2 + (L_5.s_3).c_2 \tag{6.73}$$

Resolviendo el sistema de ecuaciones (6.71) y (6.73) encontramos $cos(\theta_2)$ y $sin(\theta_2)$ y como ya es conocido θ_3 se tiene:

$$cos(\theta_2) = \frac{k_1.(L_5.c_3 + L_4) + k_2.L_5.s_3}{L_5^2 + L_4^2 + 2L_4.L_5.c_3} \tag{6.74}$$

$$sin(\theta_2) = \frac{k_2.(L_5.c_3 + L_4) - k_1.L_5.s_3}{L_5^2 + L_4^2 + 2L_5.L_4.c_3} \tag{6.75}$$

$$\theta_2 = atan2(\frac{k_2.(L_5.c_3 + L_4) - k_1.L_5.s_3}{L_5^2 + L_4^2 + 2L_5.L_4.c_3}, \frac{k_1.(L_5.c_3 + L_4) + k_2.L_5.s_3}{L_5^2 + L_4^2 + 2L_4.L_5.c_3}) \tag{6.76}$$

Por último despejaremos θ_4 de (6.58)

$$\theta_4 = (\theta_2 + \theta_3 + \theta_4) - \theta_2 - \theta_3 \tag{6.77}$$

6.4 Configuraciones posibles para el *Scorbot ER-VII*

Una vez obtenidas todas las ecuaciones para la cinemática inversa del robot *Scorbot ER-VII*, para determinar los ángulos θ_i, tenemos que tener cuidado con algunas ecuaciones, si bien se pueden utilizar los signos positivo y negativo en estas ecuaciones, los cálculos obtenidos serían correctos, sin embargo, estos alterarán la configuración que adoptara el robot [Preisz et al., 2009]. Algunas configuraciones son posibles, pero hay otras que debido a la estructura física son imposibles de implementar, (Figura 6.8).

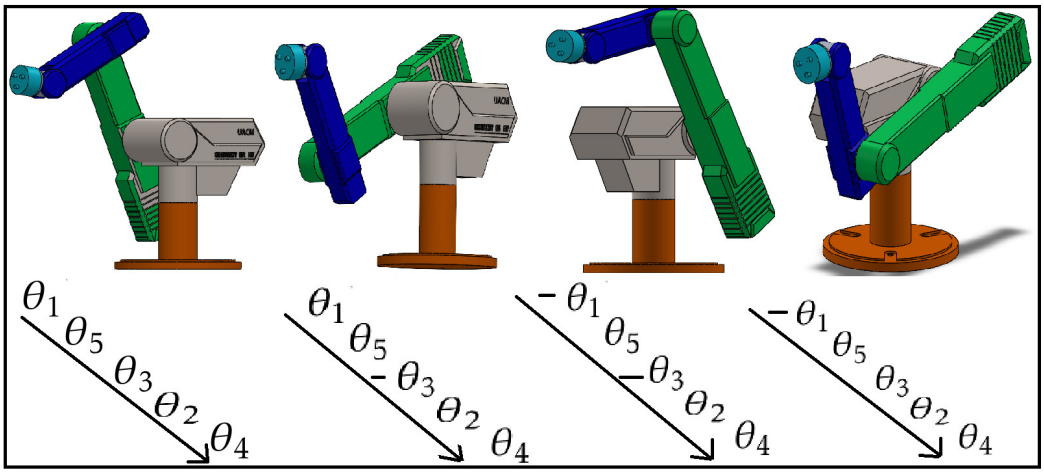


Figura 6.8: Configuraciones posibles para el *Scorbot ER-VII.*

6.5 Orientación del efector final

Para determinar los valores que deben adoptar los parámetros $n_x, n_y, n_z, o_x, o_y, o_z, a_x,$
a_y y a_z de la matriz de transformación homogénea (6.31) y con ellos determinar la orientación del efector final de un brazo articulado (robot) respecto a su origen, se pueden utilizar las matrices de rotación (4.18), (4.20) y (4.19) compuesta por rotaciones sobre los ejes x, y y z.

Para tener una mejor percepción de cómo se orientará el efector final del robot, con respecto a los ejes coordenados de la base se hace uso de *Matlab* (Anexo A.3), donde el ángulo A es un giro sobre el eje X, B es un giro sobre el eje Y y C es el ángulo de giro sobre el eje Z. En la (Figura 6.9 A) se muestra en los ejes coordenados que están orientados en la misma dirección, es decir los ángulos de giro son iguales a cero, los ejes coordenados coinciden, en la (Figura 6.9 B) se realiza un giro sobre el eje x de 180 grados. Si se observa de manera minuciosa las matrices de trasformación homogéneas, la parte que corresponde a las matrices de rotación se podrá ver que los valores de n,o y a toman un valor determinado de acuerdo a la orientación del efector final que se requiera.

Por ejemplo, en la (Figura 6.10) se tienen los siguientes ángulos de giro $A = 180^o$, $B = -45^o$ y $C = 45^o$, para que la herramienta final del robot adquiera dicha orientación los valores de n, o y a se igualan con los valores de resultantes de matriz

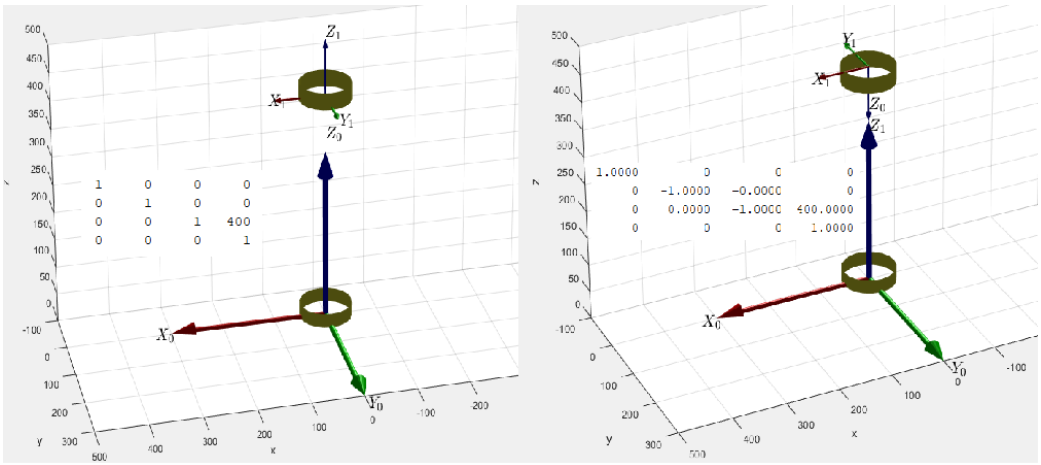


Figura 6.9: Parámetros para la orientación del efector final.

de rotación, es decir $n_x = 0.50$, $n_y = 0.5$, $n_z = 0.7071$, $o_x = 0.7071$, etc.

$$\mathbf{T} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.78}$$

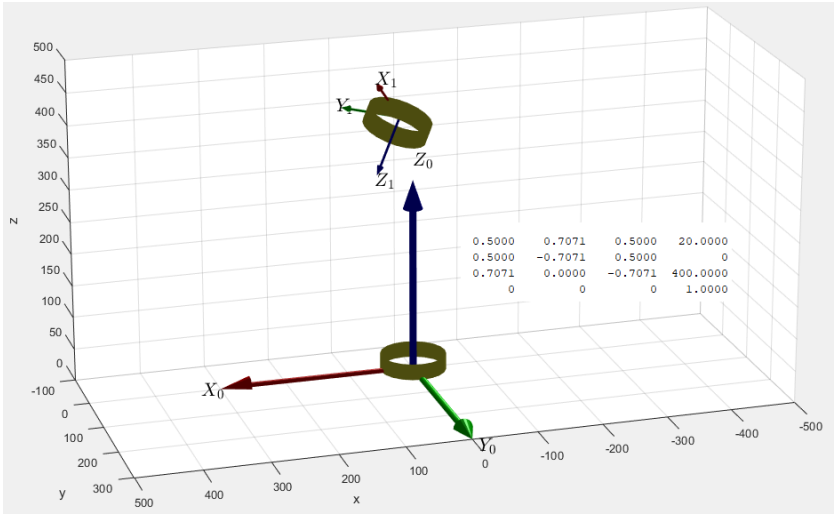


Figura 6.10: Orientación del efector final con especto a un sistema fijo.

6.6 Comprobación de ecuaciones utilizando *Matlab*

Se utiliza el software Matlab para comprobar que las ecuaciones obtenidas para la cinemática directa e inversa son correctas y servirán para cargarlas a la tarjeta Kflop. Como primer paso se instala el *Toolbox, Simple Robotics Toolbox*, referirse al (Anexo C.3) para obtener más información sobre la correcta instalación del *Tolboox*. Con esta herramienta podemos representar el diagrama cinemático de cualquier tipo de robot, el diagrama se puede representar mediante piezas sólidas haciendo uso de archivos .STL, o únicamente con líneas simples [Chavez, 2023].

En nuestro caso utilizamos cilindros para representar las articulaciones del robot *Scorbot ER-VII*, también se muestran los ejes coordenados para cada articulación y para el punto final. El código con las ecuaciones de cinemática se programa en *Matlab* como se muestra en el (Anexo A.8), donde la primera parte contiene la cinemática inversa y la segunda parte la cinemática directa. Los parámetros (x, y, z) y los ángulos de orientación son conocidos, es decir, es la posición y orientación a la que se desea llegar, con estos datos se obtienen los ángulos a los que se debe de mover cada una de las articulaciones para posicionar el efector final de manera correcta, una vez obtenidos los ángulos estos se dan como parámetros de entrada para la cinemática inversa la cual deberá de arrojar como resultado la posición y orientación deseada que se utilizaron para la cinemática directa, si ambos valores son iguales significa que nuestras ecuaciones son correctas.

Para calcular los ángulos para posicionar el efector final se hace uso de las matrices de rotación, en nuestro caso la relación de orientación de la herramienta es con respecto al eje coordenado de la base del robot.

En la (Figura 6.11) se muestra una posición conocida y fácil de analizar para observar si los valores obtenidos son correctos, los valores deseados son (300, -36, 563.5) y un giro de 180 grados alrededor del eje x, observando las flechas de colores para los ejes coordenados del punto final y la base del robot se obtiene que los ejes x coinciden, pero no ocurre lo mismo para el eje y y z, estos dos últimos van en sentido contrario, si relacionamos esto con la matriz de transformación homogénea en la parte que corresponde a la matriz de rotación, la columna 1 en la posición de x esto hace referencia que los ejes coinciden en la dirección, en el caso de la segunda columna en la posición x y z tienen un valor de cero, pero en la posición de y se tiene un -1 lo que hace referencia a que el eje y del sistema de referencia del efector final va en sentido contrario al eje y del sistema coordenado de la base, ocurre lo mismo para el eje z. Los valores de la posición deseada coinciden con los calculados en *Matlab* (Anexo A.8) Como se puede observar en

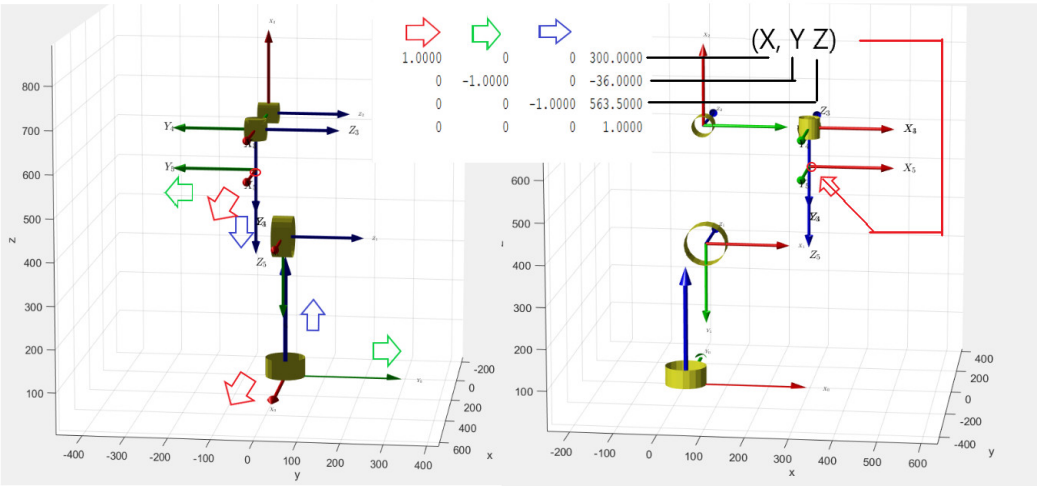


Figura 6.11: Validación de ecuaciones utilizando *Matlab*.

la (Figura 6.11), en la parte superior, la matriz de transformación homogénea nos da los valores correctos para la posición deseada.

Como se puede observar en la (Figura 6.11), en la parte superior, en la sección de desplazamientos de la matriz de transformación homogénea nos da los valores correctos para la posición que deseamos de (x, y, z). Por lo tanto, nuestras ecuaciones calculadas para la cinemática directa e inversa fueron obtenidas de manera correcta. Cabe recalcar que esta posición la utilizaremos como posición de *Home* para el *Scorbot ER-VII*.

Como segunda prueba se desea posicionar el efector final en las coordenadas (300, 100, 563.5) y un ángulo de inclinación de -45 grados girado sobre el eje *y* como se observa en la (Figura 6.12). Como se puede observar las ecuaciones fueron calculadas correctamente, tanto la orientación como la posición son correctas. Estas ecuaciones se utilizarán para crear la Clase Cinemática para la conversión del robot *Scorbot ER-VII* a CNC.

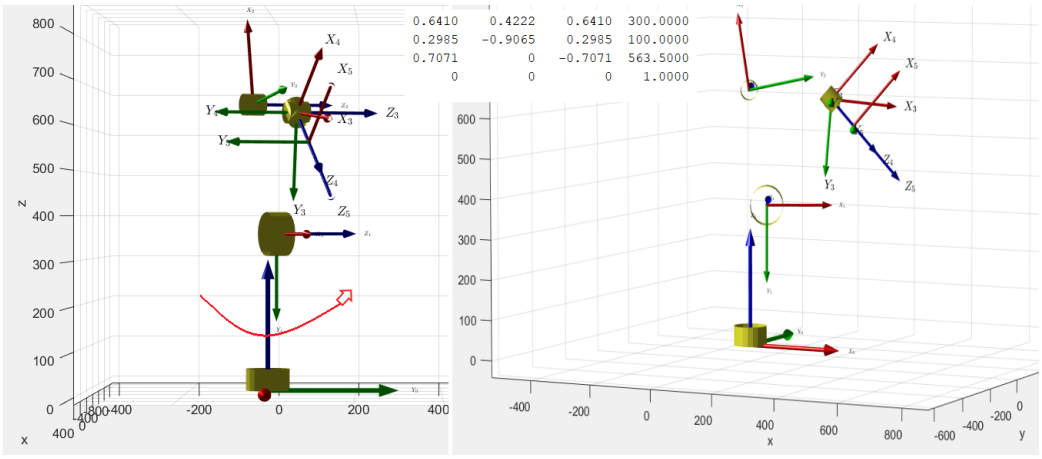


Figura 6.12: Validación de ecuaciones para una posición diferente de *Home*.

Capítulo 7

Implementación y resultados

En este capítulo se presentan los pasos a realizar para poder convertir el robot *Scorbot ER-VII* a un sistema CNC, el tipo de tarjeta controladora a utilizar y algunas características importantes de esta. Por otro lado, se mostrarán los resultados obtenidos del sistema CNC.

7.0.1 KFLOP como tarjeta controladora CNC

Kflop es controlador de movimientos muy potente, (Figura 7.1), el cual trabaja en tiempo real gracias a que puede almacenar datos en el búfer y ejecutar múltiples subprocesos simultáneos. Kflop está basado en FPGA (Una matriz de puertas lógicas programable en campo) y un DSP (Procesador de señales digitales). Esta tarjeta puede procesar 1.2 GFLOP de cálculos por segundo, es decir, mil doscientos millones de cálculos por segundo. Puede controlar hasta 8 ejes con una actualización de 90us para todos los ejes, una gran cantidad de E/S, planificador avanzado de trayectoria, capacidad para introducir cinemática, planificación de rutas con G-Code y la capacidad de programar en C [Dynomotion, 2023].

Cabe mencionar que para poder utilizar esta tarjeta no es necesario comprar licencias o software de paga, ya que tiene su propio software gratuito que contiene rutinas preestablecidas que permiten una capacidad de ajuste completa para aplicaciones que necesitan una alta precisión.

7.0.2 Aplicaciones

- Mecanizados CNC.
- Actualización de maquinaria CNC.

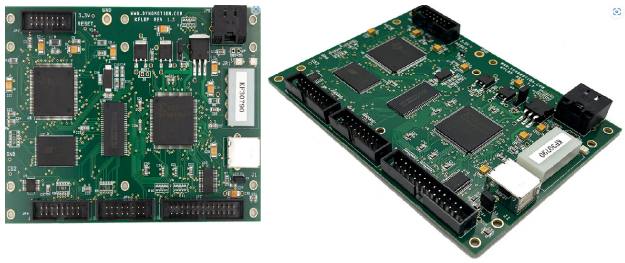


Figura 7.1: Tarjeta controladora Kflop.

- Rauters
- Sierras
- Automatización
- Robótica
- Tornos
- Fresadoras

7.0.3 SnapAmp-Amplificador (Complemento para KFLOP)

SnapAmp permite que el controlador esté en todo tiempo monitoreando todos los parámetros tales como la posición, la corriente, voltajes y temperatura en tiempo real, (Figura 7.2).

Figura 7.2: Tarjeta SnapAmp.

7.0.4 Características pricipales

- 8 entradas optoaisladas
- 4 entradas de codificador diferencial

- 14 entradas o salidas de propósito general LVTTL
- Medición de corriente digital (10 bits)
- Límites de corriente programables
- Protección contra carga en cortocircuito
- Medición de voltaje de alimentación del motor
- Suministro de motor independiente en cada lado
- Límite de corriente programable
- Medición digital de temperatura

KmotionCNC es un software gratuito con el cual se pueden editar, ejecutar y visualizar programas de código G. En *KmotionCNC* se utilizan varias plantillas en las cuales se pueden visualizar, por ejemplo, las posiciones para cada uno de los ejes, indicador de entradas y salidas, indicador de archivo G que se está ejecutando, visualizador gráfico de código G, además se tienen botones para realizar movimientos para todos los ejes, botón de paro de emergencia, selector de tipo de herramienta a utilizar, tipo de coordenadas a utilizar, entre muchas otras funciones, (Figura 7.3).

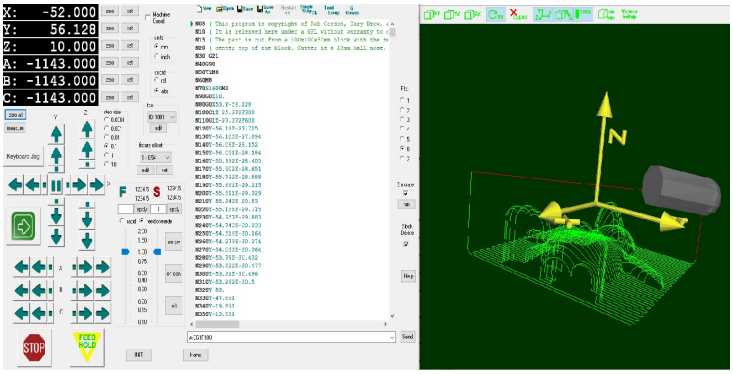


Figura 7.3: Software para G-Code, *KmotionCNC*.

Kmotion es parte del software que ofrece *Dynomotion* con el cual podemos visualizar parámetros tales como la posición real de los ejes, destino deseado para cada eje, modo de entrada y salida para pines, ejes habilitados. Tiene diferentes plantillas con las cuales podemos modificar parámetros (Figura 7.4) como lo son los canales de entrada y salida donde conectaremos nuestros encoders y motores,

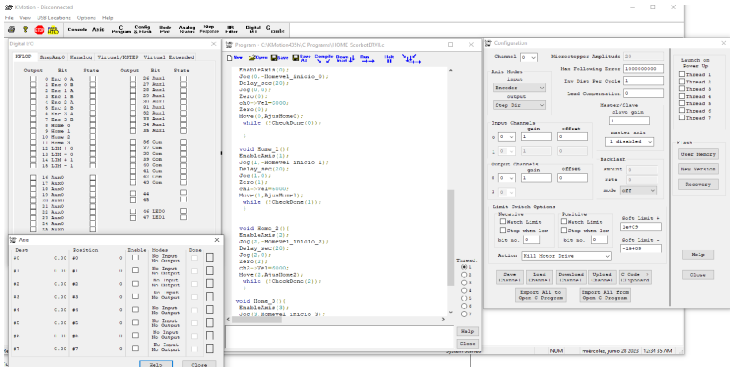


Figura 7.4: Software para configuración/visualización de parametros, *Kmotion.*

tipo de motores utilizar, calibrar el control PID para que nuestros motores trabajen de forma correcta, abrir y editar programas C, entre otras funciones.

7.0.5 Sistema de control para robots utilizando Kflop

El control PID (proporcional-integral-derivativo) es el algoritmo de control más utilizado en la industria, a pesar de que se tienen técnicas de control avanzadas, esto se debe a su simplicidad estructural, amplia gama de aplicaciones y sus capacidades de desempeño [Villarreal-Cervantes, 2014].

El PID está conformado por tres coeficientes: proporcional, integral y derivativo. Estos coeficientes pueden cambiar dependiendo del sistema a controlar. Para lograr un control óptimo es necesario sintonizar el controlador, es decir, ajustar las ganancias.

La (Figura 7.5) muestra un sistema de control general de control PID, donde la señal de referencia en el caso de los robots es el valor de las posiciones deseadas (destino) a las que se deben desplazar las articulaciones. La salida del sistema serían los ángulos que debe girar cada motor para que el robot tome una configuración particular, los sensores (encoders) se encargan de convertir las señale de salida para que pueda ser comparada con la referencia. El error es la diferencia entre el valor de referencia menos el valor medido, dependiendo del valor del error el controlador mandará una señal al sistema para que se igualen las señales de salida con las de referencia, es decir que el error se vuelva cero.

El sistema de control para un robot industrial convertido a CNC, utilizando el controlador Kflop, se muestra en la (Figura 7.6). El funcionamiento del sistema

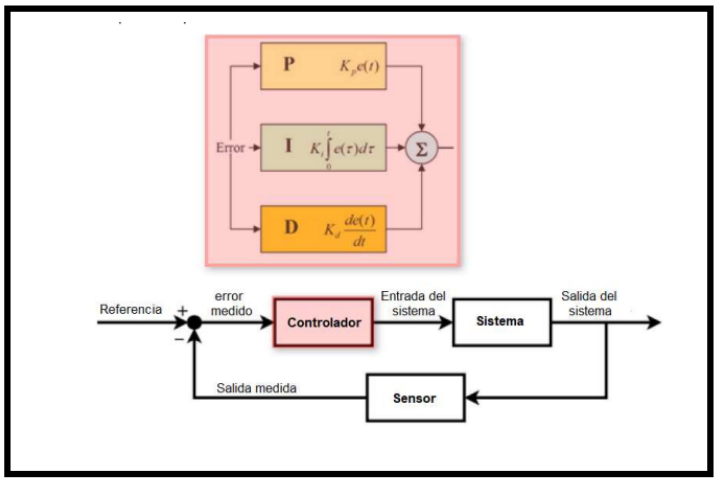


Figura 7.5: Estructura general para sistema de control PID.
Figura recuperada de [Vásquez Rojas, 2020]

es bastante simple, una vez generado el código G en algún programa CAD/CAM (*SolidCAM, Sprutcam, Aspire, Mastercam, etcétera.*), se envía por medio de un puerto USB hacia la tarjeta Kflop donde las coordenadas tomarán otros valores dependiendo de las dimensiones de la herramienta seleccionada para trabajar. Con las nuevas coordenadas, se calculan los ángulos deseados para cada articulación, esto con la ayuda de la cinemática inversa. Comparando los ángulos de las articulaciones deseados y reales se obtienen las señales de error las cuales se emplean para que el controlador PID accione el amplificador SnapAmp que a su vez mandan las señales de voltaje y corriente a los motores para que giren de tal forma que el error se vuelva cero. Los sensores, en este caso son encoders incrementales, mandan las señales a Kflop por medio de SnapAmp y se usan para la retroalimentación.

Obtenidos los ángulos para cada articulación, ángulos censados, se hace uso de la cinemática directa y los parámetros de la herramienta para devolver las posiciones y orientaciones del efector final las cuales se mostrarán en la PC, (Figura 7.6).

7.0.6 Creación de bibliotecas para distintas cinemáticas

La cinemática es un elemento más de la biblioteca de movimiento coordinado que utiliza el intérprete de G-code. Al crearse una cinemática nueva, agregar las ecuaciones de la cinemática del robot a la Kflop, el intérprete de G-code realiza una transformación de las posiciones CAD a las posiciones que necesita moverse cada actuador para que el robot se mueva a esas posiciones CAD deseadas.

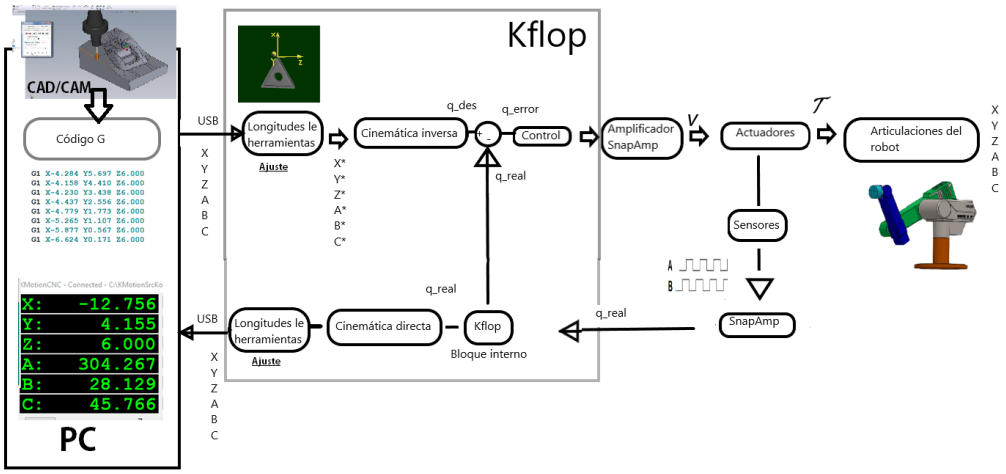


Figura 7.6: Estructura para sistema de control de Robots CNC utilizando KFlop.

Para agregar una nueva cinemática es necesario agregar una nueva clase de cinemática al proyecto GCodeInterpreter.

Las bibliotecas Kmotion vienen incluida en la descarga del software. Sin embargo, para modificarlas, crearlas, compilarlas o depurarlas se utiliza *Visual Studio, VS 2008 Standard*, incluida la versión gratuita de *Microsoft* de *VS 2013 Community*, en nuestro caso utilizaremos *Visual estudio Community 2022*. Para poder crear bibliotecas se utiliza código C++.

El primer paso es obtener la cinemática inversa de nuestro robot, en nuestro caso se obtuvieron para el robot planar de 2 grados de libertad y para *Scorbot ER-VII*, referirse al capítulo 6.

Se procede a agregar un archivo de texto con el nombre de (Kinematics.txt), el cual debe de contener una cadena de caracteres ”nombre de nuestra clase cinemática”, (Figura 7.7), la clase CCoordMotion utiliza este nombre para determinar la clase de cinemática que se utilizará, si este archivo no es creado o el nombre de la cinemática será errónea y se utilizará la cinemática lineal ortogonal por defecto. Como siguiente paso es ir a Disco local (C) y buscamos la carpeta con el nombre, KMotion435h, donde encontraremos el directorio de raíz de instalación que contiene una solución de *Visual Estudio*, con el nombre BuildAllLibs.sln, (Figura 7.8), para compilar las bibliotecas KMotion.

Al abrir la solución nos dirigiremos al lado derecho donde encontraremos una lista de soluciones, (Figura 7.9), luego nos dirigimos al archivo GCodeInterpreter

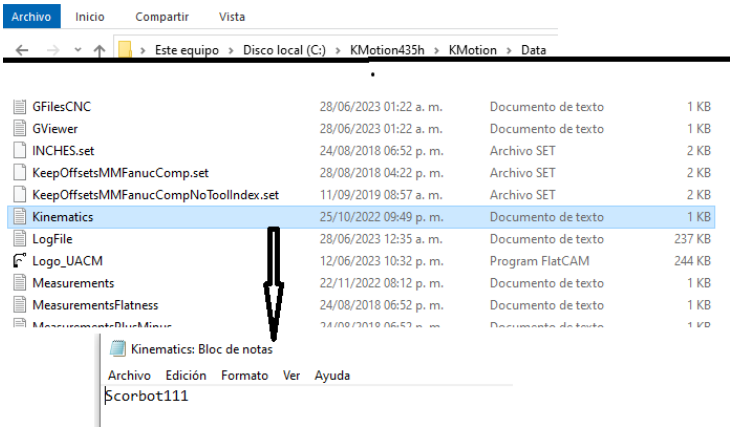


Figura 7.7: Cinemática a utilizar.

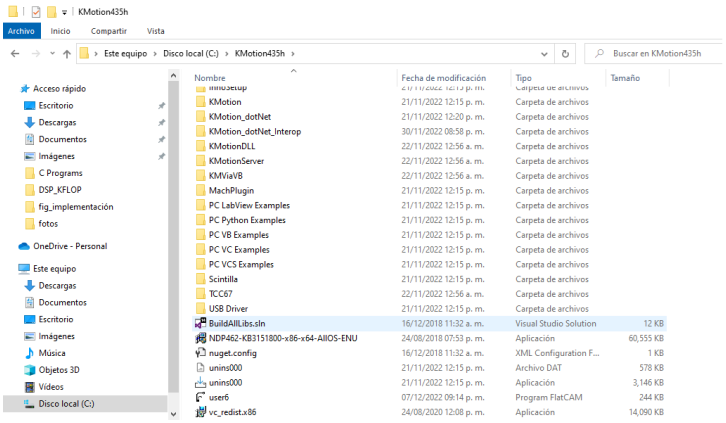


Figura 7.8: Soluciones de *Visual Estudio*, BuildAllLibs.sln.

y luego a ”Headers file” donde crearemos nuestro archivo (nombre.h) el cual le asignaremos un nombre, por ejemplo ”KinematicsScorbotER-VII.h”, luego buscamos ”StdAfx.h” donde incluiremos nuestra clase, agregando una simple línea de código con el nombre de nuestra clase cinemática, por ejemplo (# include ”KinematicsScorbotER-VII.h”).

En el mismo archivo GCodeInterpreter nos dirigimos a ”Source files” donde crearemos un archivo ”nombre.ccp” con un nombre referente a nuestra cinemática, ejemplo ”kinematicsScorbotER-VII.ccp”, una vez creado el archivo buscamos el archivo CoordMotion.ccp donde buscamos las líneas de código como las que aparecen en el código de la (Figura 7.10) recuadro, donde agregaremos nuestra clase cinemática, este último paso es para poder utilizar nuestra cinemática cuando sea necesario, la cinemática que se utiliza dependerá del nombre que con-

tenga nuestro archivo .txt.

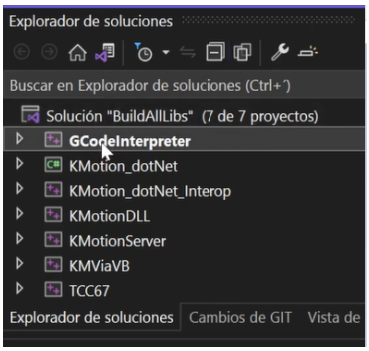


Figura 7.9: Lista de soluciones.

```
FILE *f = fopen((CString)MainPath + "\\Data\\Kinematics.txt","rt
    ");
if (f)
{  char s[81];
        fgets(s, 80, f);
        Kinematics = new CKinematicsGeppettoExtrude;
        else if (strstr(s, "Geppetto") != NULL)
        Kinematics = new CKinematicsGeppetto;
        else if (strstr(s, "Scara") != NULL)
        Kinematics = new CKinematicsScara;
        else if (strstr(s, "Scorbot111") != NULL)
        Kinematics = new CKinematicsScorbotER71AA;
        else if (strstr(s, "Kinematics2AxisRobot") != NULL)
        Kinematics = new CKinematics2AxisRobot;
        else if (strstr(s, "KinematicksMotomanSK16") != NULL)
        Kinematics = new CKinematicksMotomanSK16;
        else if (strstr(s, "KinematicsScorbotERVII") != NULL)
        Kinematics = new CKinematicsScorbotERVII;
        else
        Kinematics = new CKinematics3Rod;
                fclose(f);
}
else
{

        m_TCP_affects_actuators = false;
        Kinematics = new CKinematics;
}
```

Figura 7.10: Lista de clases cinemáticas.

Todas las ecuaciones respecto a la cinemática directa e inversa son colocadas en los archivo ”nombre.ccp”.

En el (Anexo A.4) se muestra el código utilizando como ejemplo la cinemática para el robot *Scorbot ER-VII*, se crea la clase KinematicsScorbotERVII y se seguiría la misma lógica para cargar cualquier otra cinemática, el archivo “.h” se puede encontrar en el (Anexo A.5).

7.1 Parámetros de ajuste

Dynomotion utiliza el término canal para referirse a una forma de seleccionar varios elementos idénticos [Foro-Wiki, 2023]. Kflop cuenta con 8 canales para controlar los 8 ejes, donde un canal es una entidad capaz de realizar una operación de control. La configuración para cada canal determina qué tipo de dispositivos utilizarán (motores, encoder, etc), no determina los pines de E/S que se utilizarán. Los pines de E/S son conectores fijos, sin embargo, estos se pueden asignar a un canal de los 8 disponibles, (Figura 7.11). Por ejemplo, es posible asignar la entrada 0 al canal 7 y al mismo tiempo asignar la salida 5, como puede verse se pueden asignar E/S en el orden que lo prefiera.

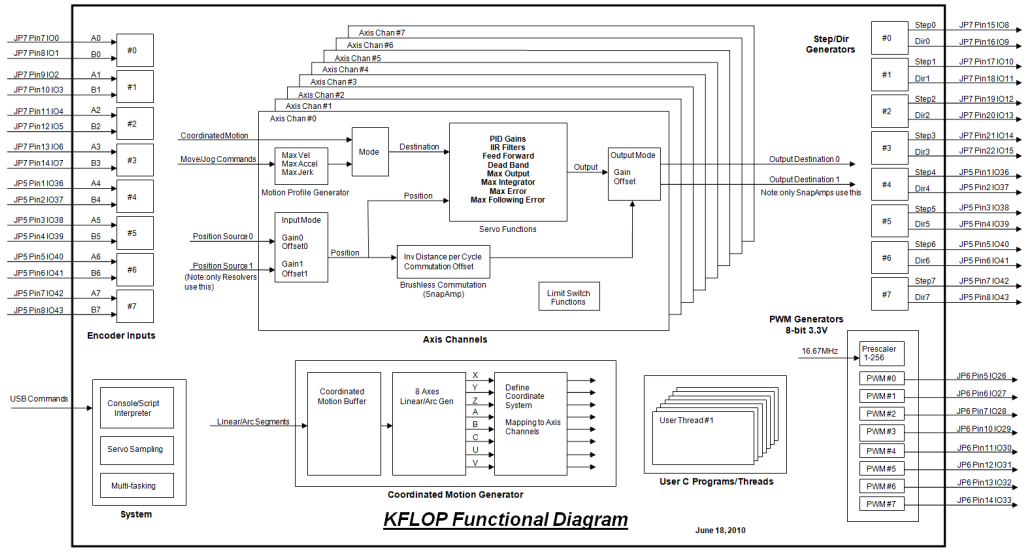


Figura 7.11: Diagrama funcional para los 8 canales que ofrece Kflop

Para cada articulación del *Scorbot ER-VII* se deben ajustar los siguientes parámetros.

Modos de entrada y salida, en nuestro caso todos los ejes tienen encoder el cual será nuestra entrada y la salida es un motor de DC. Por lo tanto, en el programa C se colocan las siguientes líneas de código.

```
ch0->InputMode = ENCODER_MODE;          ch2->InputMode =
    ENCODER_MODE;
```

```
ch0->OutputMode = DC_SERVO_MODE;      ch2->OutputMode =
    DC_SERVO_MODE;
ch1->InputMode = ENCODER_MODE;        ch3->InputMode =
    ENCODER_MODE;
ch1->OutputMode = DC_SERVO_MODE;      ch3->OutputMode =
    DC_SERVO_MODE;
```

Donde ch hace referencia al canal donde estará conectado el encoder y el motor de cada uno de los ejes. Las ganancias para el control PID fueron ajustadas a prueba y error. Como se puede ver en la siguiente lista de parámetros, únicamente usando la ganancia proporcional el control funciona correctamente.

```
ch0->P = 10;      ch2->P = 10;
ch0->I = 0;       ch2->I = 0;
ch0->D = 0;       ch2->D = 0;
ch1->P = 10;      ch3->P = 10;
ch1->I = 0;       ch3->I = 0;
ch1->D = 0;       ch3->D = 0;
```

Un amplificador SnapAmp agrega cuatro amplificadores de puente completo PWM. Los cuatro PWM se identifican como 8, 9, 10, 11, [Dynomotion, 2023]. Por lo tanto, se asignaron los canales de la forma siguiente.

```
ch0->InputChan0 = 8;          ch2->InputChan0 = 10;
ch0->InputChan1 = 0;          ch2->InputChan1 = 0;
ch0->OutputChan0 = 8;         ch2->OutputChan0 = 10;
ch0->OutputChan1 = 0;         ch2->OutputChan1 = 0;
ch1->InputChan0 = 9;          ch3->InputChan0 = 11;
ch1->InputChan1 = 0;          ch3->InputChan1 = 0;
ch1->OutputChan0 = 9;         ch3->OutputChan0 = 11;
ch1->OutputChan1 = 0;         ch3->OutputChan1 = 0;
```

7.1.1 Paro de emergencia

Las paradas de emergencia son una función que todo sistema debe de tener implementado, ya que son un recurso para detener el sistema en su totalidad y así prevenir un accidente o prevenir acciones que pongan en riesgo a las personas. Al presionar el botón de emergencia en seguida la máquina se detendrá sin importar la acción que esté realizando. En cada estación de trabajo debe de haber un botón de paro de emergencia al alcance del operador listo para ser activado en cuanto se presente una emergencia.

Por lo tanto, se implementó un botón de paro de emergencia para el Robot *Scorbot ER-VII* CNC, donde el botón E-Stop estará conectado en el Pin 26 del conector JP6-Aux-1 de Kflop. Para hacer uso E-stop es necesario utilizar el código C (Anexo A.6) proporcionado por *Dynomotion*. Este programa se anexó al programa *INIT* (programa de parámetros para los ejes), debe ejecutarse en el hilo 1, ya que este hilo siempre seguirá ejecutándose, en los otros hilos se detendrán los programas que se estén ejecutando al instante al presionar el botón E-Stop. Kflop puede ejecutar hasta 7 programas C, cada programa se ejecuta en un hilo.

7.1.2 Uso de Gamepad USB

También es posible utilizar un Gamepad USB para mover el sistema o ejecutar algunas funciones. Primero se debe de seleccionar la opción (Enable Gameapad) en el software *KmotionCNC* enseguida se conectará el Gameapad a la PC, automáticamente se vincularán. Hay que tener en cuenta que no se pueden personalizar las acciones que se realizan cuando un botón es presionado, las acciones ya están configuradas por defecto, sin embargo, dependiendo del gamepad utilizada podrían cambiar las acciones realizadas cuando es presionado un botón.

El *joystick* izquierdo controla los movimientos sobre el eje X y Y, el botón X sirve para dar inicio a la ejecución del G code, el botón A al ser presionado ejecutará solo una línea de G code, el *joystick* de botones de lado izquierdo sirve para deshabilitar todos los ejes, en la parte de atrás del gamepad se encuentran 2 gatillos, serven para realizar un movimiento en Z positivo y otro un movimiento en Z negativo, el botón superior a la derecha del control servirá para parar todo (Stop), (Figura 7.12).

7.1.3 Posicionamiento en Home para *Scorbot ER-VII*

Cuando el sistema CNC es encendido por primera vez no se sabe la posición en la que se encuentra cada articulación, puesto que no se tiene un sistema de memoria que capture los datos y los guarde. Para que se pueda tomar una referencia y a partir de esta saber las posiciones reales de cada articulación se implementó una función de Home.

Para saber la posición inicial en la que se encuentran todas las articulaciones del *Scorbot ER–VII* cada vez que se enciende por primera vez el sistema es necesario implementar una función la cual posicione todas las articulaciones siempre en la misma posición para que esta sea tomada como referencia para empezar a contar los pulsos que entrega cada encoder y así poder determinar el ángulo que adoptan todas las articulaciones. Para el *Scorbot ER-VII* la posición *Home* que se decidió

Figura 7.12: Gamepad USB.

adoptar se muestra en la (Figura 7.13). Se hace uso de un programa en C (Anexo A.7) y es invocado por el botón *Home* en la pantalla de *KmotionCNC*, este funciona de la siguiente forma al ser activado deshabilitaran todas las articulaciones y luego habilitara la articulación 2 para que esta gire hasta su límite de movimiento, a partir de allí el contador de pulsos se pone a cero, luego se envía un comando con la cantidad de pulsos necesarios para que la articulación se posicione de forma vertical, se sigue el mismo procedimiento para las otras articulaciones. Los valores de posición y orientación del efector final (X,Y,Z,A,B,C) cuando el robot esté en *Home*, son las que se mostraran en *KmotionCNC*.

7.1.4 Implementación de robots CNC.

Como primer experimento y para observar el correcto funcionamiento del uso de la cinemática en la tarjeta KPLOP, se decido utilizar un robot en su forma más simple, en este caso un robot planar de 2 grados de libertad, el cual se muestra su

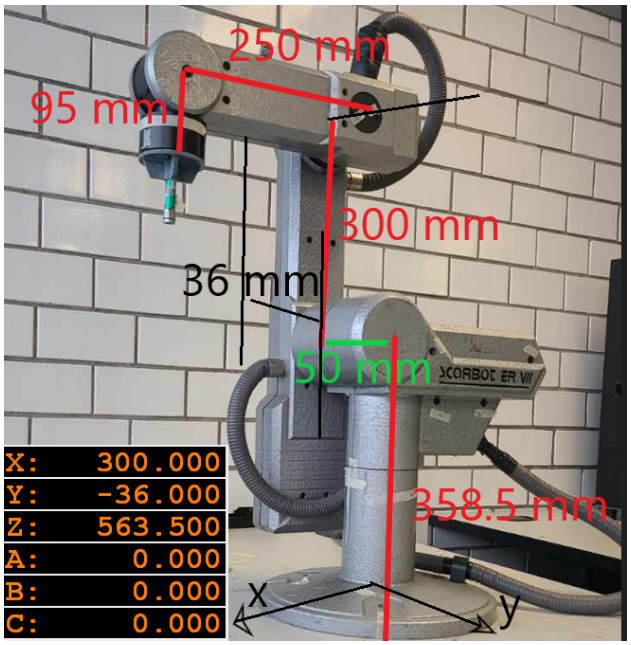


Figura 7.13: Home *Scorbot ER-VII*.

modelo matemático en el capítulo 6.

En la (Figura 7.14) se muestra la conexión de los dos motores de DC y su respectivo encoder a la tarjeta Kflop y SnapAmp, así como un robot planar de 2 GDL improvisado, para ver más a detalle las características de los motores refiérase al (Anexo B.0), si se desea ver de manera más clara las conexiones de los motores y encoder referirse al (Anexo B.1), donde se muestra el diagrama eléctrico de forma más detallada.

Una vez cargada la clase cinemática y la configuración a la Kflop se procedió a abrir el software *KmotionCNC* y se hicieron pruebas de movimiento sobre los ejes x y y utilizando G-code con el fin de observar que los dos motores se movieran simultáneamente, (Figura 7.15), ya que para mover el punto final sobre el eje x o y es necesario que los dos motores se muevan en conjunto para conseguir que el punto final se mueva solo sobre un eje. Al observar que los motores se comportaban de forma correcta se construyó una pequeña maqueta improvisada, con eslabones de plástico y sujetado con cinta blanca los motores, cabe recalcar que se hizo de esta manera para validar de forma sencilla y rápida que se podía utilizar la cinemática en conjunto con este tipo de tarjeta para convertir un robot a CNC.

Sin embargo, debido a que las transmisiones acopladas a los motores tenían mu-

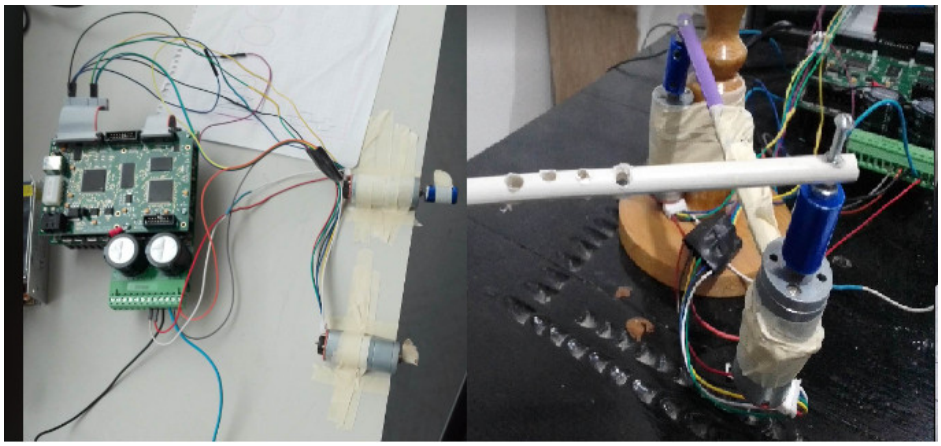


Figura 7.14: Primeras pruebas para convertir robot planar de 2GDL a CNC.

cho *BLACKLAS* (pequeño espacio entre dos piezas mecánicas) y los eslabones no eran lo bastante rígidos se decidió por implementar un robot de más grados de libertad en este caso se decidió trabajar con el robot *Scorbot ER-VII*.

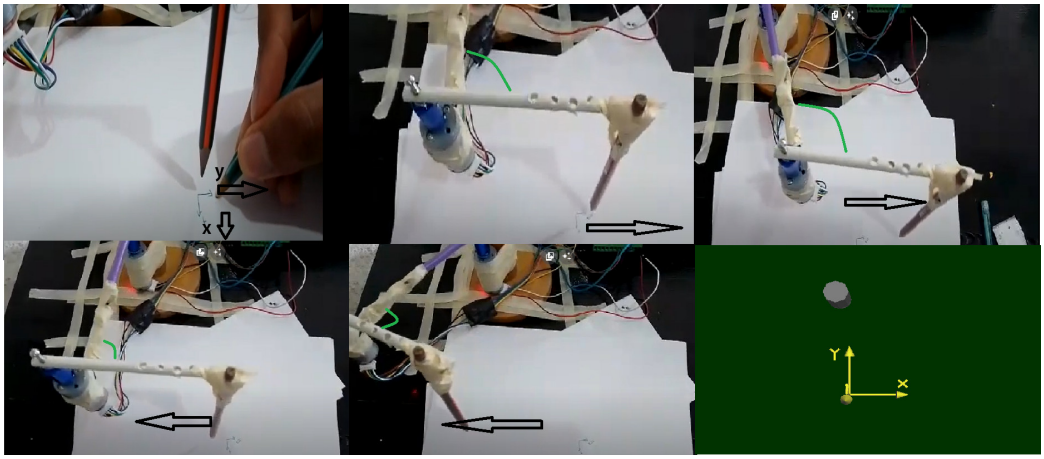


Figura 7.15: Pruebas de movimiento sobre el eje y utilizando *KmotionCNC*.

Implementación del *Scorbot ER-VII* como CNC

Como ya se ha mencionado, el *Scorbot ER-VII* es de 5 grados de libertad, sin embargo, debido a que el amplificador de potencia SnapAmp solo soporta 4 motores de DC, la última articulación (articulación 5) se mantiene fija a la 4 articulación, por lo tanto, solo se contemplarían 4 grados de libertad para este caso.

Utilizando las ecuaciones de la cinemática directa e inversa y creando la clase de cinemática para este robot en particular se procede a hacer pruebas para observar el correcto funcionamiento de los comandos de posición, y observar que se muevan cada uno de los motores los grados necesarios para posicionar y orientar el robot, (Figura 7.16), se hace de esta manera para prevenir accidentes, ya que si se envía un comando de posición errónea podría hacer que el robot se mueva bruscamente o sin control.

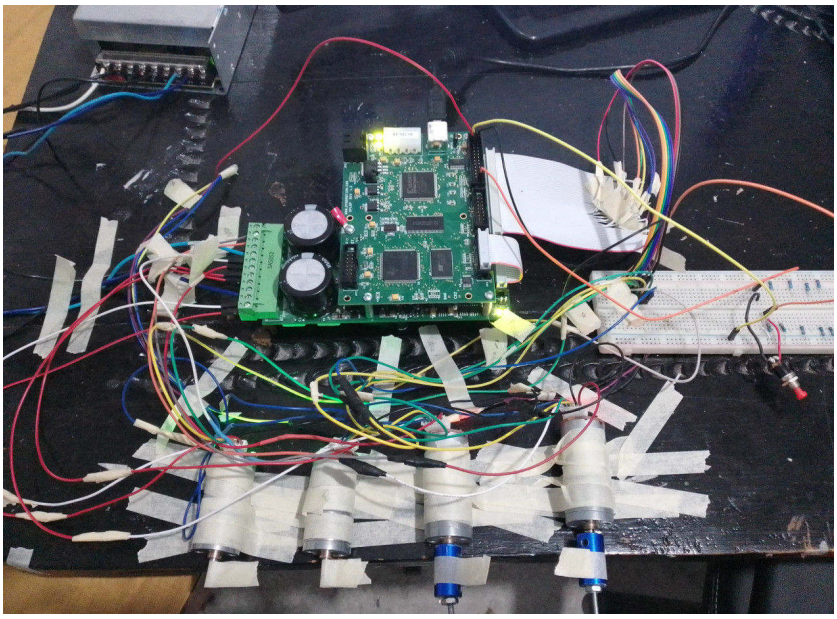


Figura 7.16: Prueba de correcto funcionamiento de comandos de movimiento.

Una vez comprobado el correcto funcionamiento de los comandos de posición se procede a diseñar las herramientas que servirán como efector final. Se utilizan dos herramientas para hacer distintas pruebas, un láser y un Dremel 3000, la primera fue construida con un láser de juguete el cual se adaptó a una base impresa en 3D, (Figura 7.17), se utilizaron una impresora de resina (Mega 8k) y una de filamento (Anycubic Chiron) que se encuentran en el laboratorio de Robótica en la UACM, cabe recalcar que para imprimir en 3D se pidió ayuda a expertos, estos nos asesoraron en como calibrar de forma correcta las impresoras para conseguir imprimir las piezas en 3D de forma correcta, se utilizó la versión gratuita de *Ultimaker cura*, para ajustar los parámetros de las impresoras, ya que es un software completamente gratuito.

Como segunda herramienta se adaptó la herramienta, Dremel 3000, referirse al

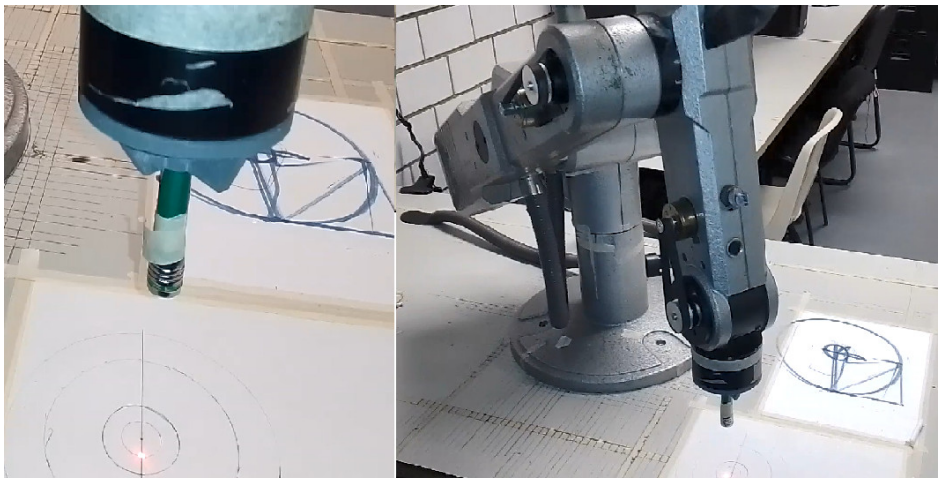


Figura 7.17: Láser como adaptado como efector final.

(Anexo B.3) para más detalles sobre el efector final, para esto se imprimió una base en filamento PLA, en la cual se fijó el Dremel como se muestra en la (Figura 7.18).

Figura 7.18: Dremel 3000 adaptado como efector final.

Teniendo el robot con su herramienta de trabajo y con las conexiones necesarias, los diagramas de las conexiones eléctrica se muestran en el (Anexo B.2), se hecen las siguientes pruebas, como primera prueba se genera el código G para hacer un cuadrado de 5 cmx5 cm, (Figura 7.19). Se dibujó el cuadrado en una hoja de papel, la cual se fijó a la mesa donde está fijado el robot y se coloca la punta del láser de tal manera que coincida con el origen del cuadrado y se procede a correr

el código G, como se muestra en la (Figura 7.21) se observa que el láser sigue de forma correcta el cuadrado dibujado en la mesa, cumpliendo con las medidas de 5 cm por lado que se requiere.

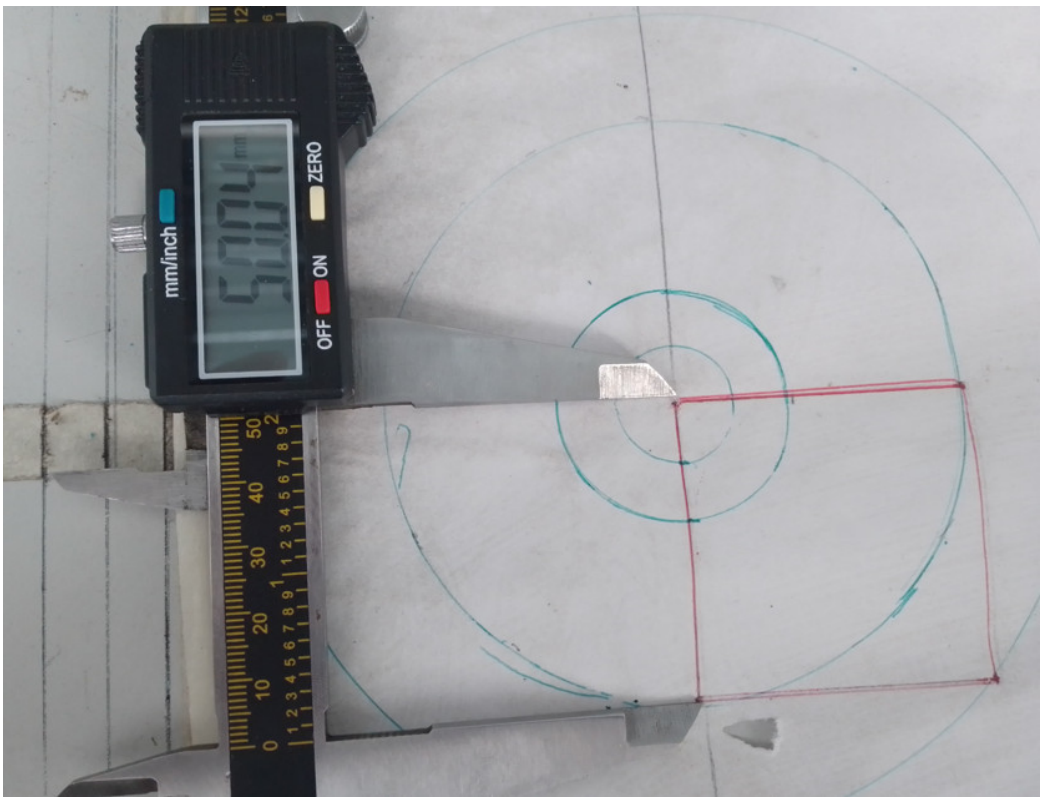


Figura 7.19: Medición de cuadrado plasmado en la mesa de trabajo para hacer pruebas.

La siguiente prueba fue realizar círculos con diferentes radios, (Figura 7.20), que van desde 1 cm, 2 cm, 5 cm y 7 cm. En la (Figura 7.22) se muestra que el láser sigue de forma correcta el círculo de 7 cm, de igual manera se hizo la prueba para cada uno de los diferentes círculos. Se decidió hacer pruebas con círculos, ya que estos necesitan de movimientos coordinados de todas las articulaciones para seguir la trayectoria circular.

Cambiando la herramienta de trabajo, es decir, cambiando el láser por el dremel 3000, se realiza una prueba más elaborada que consiste en tallar en un pedazo de madera el logo de la UACM, para más detalles sobre el código G dirigirse al (Anexo C.1). Se procedió a realizar el código G utilizando el Software *Inkscape*,

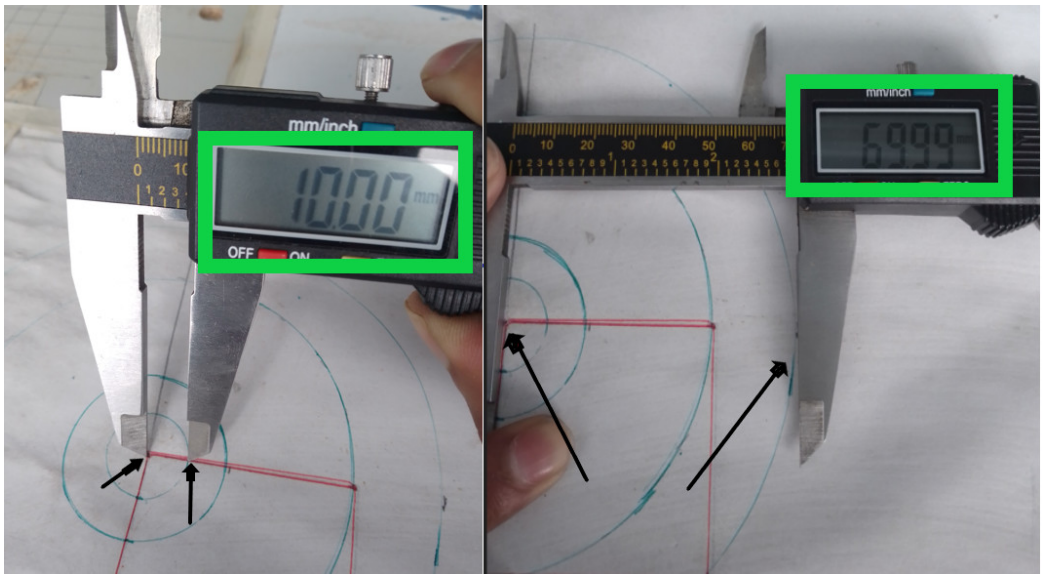


Figura 7.20: Medición de círculos plasmado en la mesa de trabajo para hacer pruebas.

luego se cargó el código G en el software *KmotionCNC* y se pone en marcha para que el robot efectúe el tallado en madera como se observa en la (Figura 7.23).

Se realizó el tallado del logo de la UACM con el fin de mostrar que es posible maquinar piezas más complejas, en este caso a partir de una imagen que contenía el logo "UACM" se generó el código G utilizando *Inkscape*, sin embargo, se puede hacer uso de otros programas CAM.

Código G para cuadrado de 5cmX5cm

```
1 G0  X0  Y0  Z0
2 F400
3 G1  X50
4 G1  Y50
5 G1  X0
6 G1  Y0
7 M2
```

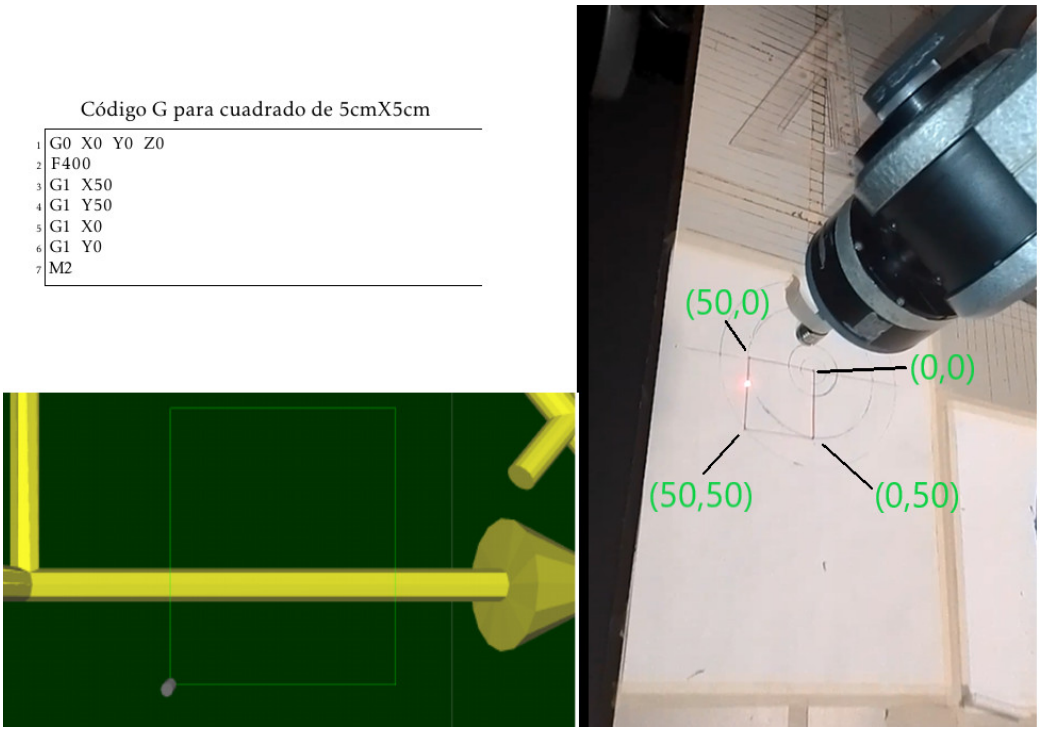


Figura 7.21: Prueba para hacer un cuadrado de 5cmx5cm por lado.

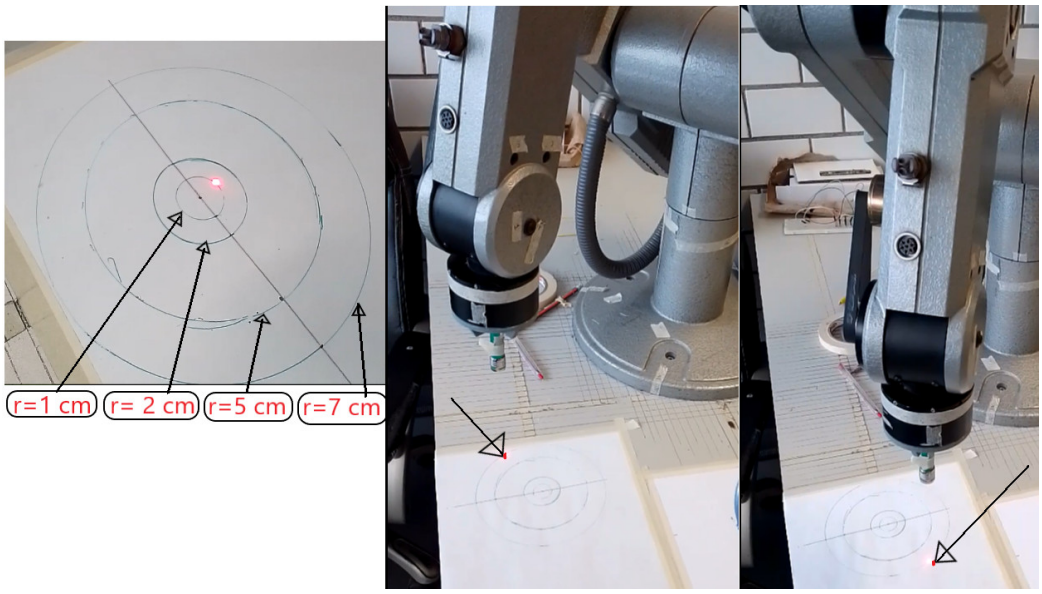


Figura 7.22: Pruebas con trayectorias circulares.

Figura 7.23: Logo UACM realizado con código G, utilizando el *Scorbot ER-VII*.

7.2 Comparaciones y costos de sistemas CNC.

Si hacemos una pequeña comparación entre las ventajas y desventajas de un robot CNC y una máquina CNC tendremos que:

Un robot CNC puede trabajar en un gran número de posiciones y orientaciones, mientras que una máquina CNC está muy limitada.

Un robot CNC puede ser utilizado casi en cualquier proceso industrial, solo haría falta cambiar la herramienta de trabajo.

El área de trabajo de una máquina CNC está limitada a un cubo, a comparación de un robot CNC que puede llegar a tener un área de trabajo mucho más grande, principalmente una esfera.

Un robot CNC podría llegar a tener 6GDL a diferencia de las máquinas CNC que podría tener hasta 5 GDL Máximo.

En la (Tabla 7.1) se muestran los costos aproximados de equipos CNC, como se puede ver a medida que la máquina CNC tiene más grados de libertad es más costosa. Si comparamos el costo de conversión a CNC para el *Scorbot ER-VII* y los otros sistemas, tendríamos un costo bastante bajo, aunque en este caso el *Scorbot ER-VII* podríamos considerarlo un robot pequeño y económico, se podría sustituir por otro con mayores prestaciones y el costo aumentaría considerablemente, sin embargo, el costo total de la implementación sería menor al valor de los otros equipos.

Precios de quipos CNC				
Scorbot ER-VII	Robot CNC	CNC	CNC	CNC
4 GDL	6GDL	3 GDL	4 GDL	5 GDL
Kflop =$5031.24 MN SnapAmp = $8385.40 MN Scorbot ER-VII sin controlador = $20000 MN-$30000 MN Fuente conmutada 24V =$300 MN Impresión 3D =$350 MN Varios=$200 MN	DENSO VP-6442G =$250,000 MN Software =$ 120,600 MN			
Total=$34,266.64 MN	$370,600 MN	$30,000MN - -270,000 MN	$30,000 MN - -600,000 MN	$250,000 MN - -2,000,000

Tabla 7.1: Comparación entre precios de equipos CNC.

Capítulo 8

Conclusiones

- Es posible convertir cualquier tipo de robot industrial a un sistema CNC utilizando el controlador Kflop.

- El robot *Scorbot ER-VII* pudo convertirse a un sistema CNC, el cual ya puede posicionarse en las coordenadas (x, y, z) deseadas y además puede adoptar algunas orientaciones; sin embargo, al estar limitado a 5 grados de libertad no puede adoptar todas las orientaciones que realiza un robot CNC industrial.

- El ajuste para el control PID no es tan sencillo debido a que se debe ajustar motor por motor (articulación) y dependiendo las posiciones de los eslabones podría afectar el ajuste del controlador, este cambio se da por el peso de los eslabones y los pares que se generan, no es lo mismo dar un ajuste con todas las articulaciones posicionadas en la posición de home que si se tiene todas las articulaciones estiradas completamente. Finalmente se ajustaron las ganancias de forma que el control trabaje de forma correcta.

- Para convertir cualquier robot a CNC utilizando este método es necesario obtener la cinemática directa e inversa para posteriormente generar la clase de cinemática siguiendo los pasos antes mencionados, utilizando el software Visual Estudio. Hay que tener en cuenta que se debe tener mucho cuidado al momento de probar o cargar la cinemática nueva y enviar algún comando a la tarjeta KFLOP puesto que si el robot adopta una posición a la cual se llega a alguna singularidad, el robot podría moverse de forma brusca y podría dañar a los operarios o a el propio mecanismo del robot.

- Es posible utilizar un robot CNC como una herramienta multipropósito, ya que únicamente cambiando la herramienta de trabajo se podrán realizar diferentes procesos.

Capítulo 9

Trabajos futuros

9.1 Conversión de robot MOTOMAN SK16 (CNC)

Actualmente se está trabajando con un robot Motoman SK16, (Figura 9.1), de la empresa CIARobotics, el cual es un robot industrial de 6 grados de libertad. Principalmente se pretendía retirar todo el sistema de control y utilizar únicamente los servopack originales del robot (drivers y motores) y remplazar el controlador por la tarjeta de control Kflop, investigando la forma en como poner en marcha los servopack encontramos dificultades de comunicación debido a que Yaskawa no proporcionó información referente al tipo de comunicación que utilizan para mandar las señales correspondientes de control. Luego se procedió a intentar utilizar únicamente los motores con otro tipo de servo drivers comerciales pero resulta ser que los encoders acoplados a los motores (originales) envían la información por medio de algún protocolo de comunicación especial y que únicamente Yaskawa lo tiene documentado, por lo tanto, se decidió hacer un cambio de servo drivers por unos comerciales (controladores de los motores y motores), pero debido a que es necesario acoplar de forma precisa cada uno de los nuevos motores a la estructura del robot,es necesario que un tornero realice este trabajo, ya que nosotros no estamos capacitados para realizar este tipo de modificaciones.

Estamos en espera de que se terminen de acoplar los motores a la estructura mecánica del robot,por lo tanto, por el momento no nos fue posible hacer la implementación física y se decidió dejarlo para trabajos futuros.

Figura 9.1: Robot industrial Motoman SK16 de 6 GDL.

9.2 Drones CNC

Intentar aplicar esta misma filosofía CNC a sistemas conformados con drones y darles aplicaciones en concreto, por ejemplo, se podrían utilizar para formar figuras 3D en el cielo de forma similar a los espectáculos que ofrece INTEL mostrado en la (Figura 9.2), espectáculo de luces con drones (*Drone Light Shows*).

Se podría también aplicar a la agricultura, si se desea fumigar un área de ter-

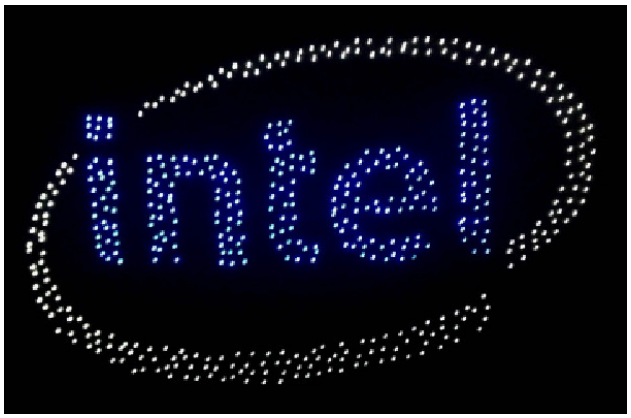


Figura 9.2: Espectáculo de drones presentado por, INTEL (Noviembre de 2016).

reno y esta se realiza de forma manual podría llegar a ser muy tedioso dado que es necesario que el operador esté siempre pendiente de los movimientos que realiza el dron, por el contrario, si se utiliza un dron CNC simplemente se genera el

G-code para indicar la trayectoria que debe seguir el dron, como se muestra en la (Figura 9.3), se muestra a modo de ejemplo dos trayectorias que podría seguir el dron. Para poner en marcha el dron simplemente se lecargaría o enviaria el código G y se pondría en marcha el programa, el dron realizaría el fumigado de forma automática y eficientemente.

Figura 9.3: Aplicación CNC en el área de la agricultura.

9.3 Ruedas omnidireccionales y robot CNC

También como trabajo futuro se planea implementar un sistema que cuente con ruedas omnidireccionales donde se montará el robot CNC, (Figura 9.4), donde el sistema podría desplazarse dentro de un área de trabajo muy grande. Algunas aplicaciones donde se podría implementar este sistema podrían ser para soldadura de estructuras, pintura o estiba de materiales, por mencionar algunos ejemplos.

Figura 9.4: Ruedas omnidireccionales y robot CNC.

Anexos

A.1 Cubo rotado en el espacio 3D

Este programa es escrito en ***Matlab*** **versión R2020b**, teniendo como función poder visualizar de manera gráfica cómo se pueden aplicar las matrices de rotación en el espacio 3D. Se genera un cubo (unitario) utilizando las coordenadas de sus vértices y luego se unen para así formar el cubo, cada coordenada se multiplicará por una matriz de rotación para finalmente poder observar un cubo rotado en el espacio 3D.

```
clc;
clear all;
close all;
%  DEFINIR VARIABLES
%  _____valores para el plano___
x1=-1.5; y1=-1.5; z1=-1; x2=1.5; y2=1.5; z2=1.5;
%  *puntos de vertices del cubo*
px0=[0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0];
py0=[0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1];
pz0=[0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0];
plot3( px0,py0,pz0)
% dibuja los puntos en el plano y los une
ang1= 0  % \angulo para girar en x
ang2=0   % angulo para girar en y
ang3=0   % angulo para girar en z
%****** MATRICES DE ROTACION******
theta1=ang1*(pi/180);
theta2=ang2*(pi/180);
theta3=ang3*(pi/180);
Rx_theta=([1 0 0;
0 cos(theta1) -sin(theta1);
0 sin(theta1) cos(theta1)]);
Ry_theta=([cos(theta2) 0 sin(theta2);
0  1 0;
-sin(theta2) 0 cos(theta2)]);
Rz_theta=([cos(theta3) -sin(theta3) 0;
          sin(theta3) cos(theta3) 0;
          0 0 1;]);
sigma1=   Rx_theta*Ry_theta*Rz_theta*[px0;py0;pz0];
px1=sigma1(1,:);%EXTRAE FILA 1 Y TODAS LAS COLUMNAS Y GUARDA EN
    PX1
py1=sigma1(2,:);%EXTRAE FILA 2 Y TODAS LAS COLUMNAS Y GUARDA EN
    PXY
pz1=sigma1(3,:);%EXTRAE FILA 3 Y TODAS LAS COLUMNAS Y GUARDA EN
    PXZ
plot3(px0,py0,pz0,px1,py1,pz1),grid;
         % formato del plano
axis([x1,x2,y1,y2,z1,z2]);  %limites de los ejes para mostrar
axis square;                %cuadricula cuadrada
```

```
grid on;                    % mostrar la cuadricula
view (3);                   %para ver en 3D
xlabel('eje x');            % da nombre al eje x
ylabel('eje y');            % da nombre al eje y
zlabel('eje z');            % da nombre al eje z
```

A.2 Traslaciones y rotaciones en el espacio 3D

Este programa es escrito en ***Matlab*** **versión R2020b**, el cual tiene la función de poder demostrar cómo aplicar traslaciones y rotaciones en el espacio 3D. Se genera un cubo (unitario) utilizando las coordenadas de sus vértices y luego se unen para así formar el cubo. Cada coordenada se multiplicará por una matriz transformación homogénea de rotación y traslación para finalmente ver un cubo rotado y trasladado en el espacio 3D.

```

%_____valores para el plano___
a=5; x1=0; y1=0; z1=0; x2=a; y2=a; z2=a;
      %  *puntos de vertices del cubo*
px0=[0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0];
py0=[0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1];
pz0=[0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0];

d01=[0;0;3];            %desplazamiento en el espacio
plot3(px0,py0,pz0)    % dibuja los puntos en el plano y los une

ang1= 0  % angulo para girar en x
ang2=0   % angulo para girar en y
ang3=0   % angulo para girar en z
%****** MATRICES DE ROTACION******
theta1=ang1*(pi/180);
theta2=ang2*(pi/180);
theta3=ang3*(pi/180);
Rx_theta=([1 0 0;
0 cos(theta1) -sin(theta1);
0 sin(theta1) cos(theta1)]);

Ry_theta=([cos(theta2) 0 sin(theta2);
0  1 0;
-sin(theta2) 0 cos(theta2)]);

Rz_theta=([cos(theta3) -sin(theta3) 0;
sin(theta3) cos(theta3) 0;
0 0 1;]);
%   conversion del sistema 0 al sistema 1

```

```
32 sigma1=   Rx_theta*Ry_theta*Rz_theta*[px0;py0;pz0];
33 px1=sigma1(1,:);%EXTRAE FILA 1 Y TODAS LAS COLUMNAS Y GUARDA EN
       PX1
34 py1=sigma1(2,:);%EXTRAE FILA 2 Y TODAS LAS COLUMNAS Y GUARDA EN
       PXY
35 pz1=sigma1(3,:);%EXTRAE FILA 3 Y TODAS LAS COLUMNAS Y GUARDA EN
       PXZ
36 plot3(px0,py0,pz0,px1,py1,pz1),grid;
37
38 % formato del plano
39 axis([x1,x2,y1,y2,z1,z2]);  %limites de los ejes para mostrar
40 axis square;                %cuadricula cuadrada
41 grid on;                    % mostrar la cuadricula
42 view(3);                    %para ver en 3D
43 xlabel('eje x');            % da nombre al eje x
44 ylabel('eje y');            % da nombre al eje y
45 zlabel('eje z');            % da nombre al eje z
```

A.3 Orientación entre dos sistemas de referencia

Este programa es escrito en **textitMatlab versión R2020b**, el cual sirve para aplicar rotación en el espacio 3D utilizando los ángulos de Euler, es decir aplicar un giro en X un giro en Y y un giro en Z. Se representa la orientación entre dos sistemas de referencia haciendo uso de las matrices de rotación. Útil para darle valores a los parámetros $n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y$ y a_z, con ellos orientar el efector final de un brazo articulado (Robot).

```
1
2 clc
3 clear all
4 H0=eye(4)
5 %syms psi1 psi2 psi3
6 anguloA=0
7 %Distancias entre origenes de los sistemas de referencia
8 d_x=50
9 d_y=100
10 d_z=100
11 %************************************
12 psi1_x=0 %%giro en x
13 psi2_y=0%%giro  en y
14 psi3_z=pi/3%%giro en z
15 %************************************
16 Tz =[ cos( psi3_z), -sin( psi3_z),0 ;
17 sin( psi3_z),      cos( psi3_z),0;
18 0,                          0,          1;]
19 Tx =[ 1, 0,  0;
```

```
0,      cos(psi1_x),-sin(psi1_x);
0,                  sin(psi1_x),        cos(psi1_x);]
Ty =[ cos(psi2_y),0, sin(psi2_y);
0,      1,0;
-sin(psi2_y),                   0,          cos(psi2_y);]
%**********************************
%T01=(Tz*Ty*Tx)
T01 =[cos(psi2_y)*cos(psi3_z), cos(psi3_z)*sin(psi1_x)*sin(
    psi2_y) - cos(psi1_x)*sin(psi3_z), sin(psi1_x)*sin(psi3_z) +
    cos(psi1_x)*cos(psi3_z)*sin(psi2_y),d_x;
cos(psi2_y)*sin(psi3_z), cos(psi1_x)*cos(psi3_z) + sin(psi1_x)*
    sin(psi2_y)*sin(psi3_z), cos(psi1_x)*sin(psi2_y)*sin(psi3_z)
    - cos(psi3_z)*sin(psi1_x),d_y;
-sin(psi2_y),                                   cos(psi2_y)*sin(
    psi1_x),                                  cos(psi1_x)*cos(
    psi2_y),d_z;
0,0,0,1];
%**********************************
figure(1) %Figura donde se va a trabajar
clf %limpio la grafica de la figura
hold on % permanescan las graficas conforme se agregan mas
    figuras
%PERMITE DIBUJAR UN SISTEMA DE REFERENCIA
[origen0]=Sistema3D(H0,500,'X_0','Y_0','Z_0',16);
[origen1]=Sistema3D(T01,300,'X_1','Y_1','Z_1',16);

 %Eslabon(origen0,origen1,'k',2)
cilindro(H0,365,50,0) %Altura, Diametro Y
cilindro(T01,200,50,100)

 %CARTESIANO EN 3d  con la matriz de transformacion  asociada HO
% (1 =magnitud de las  unidad de los ejes) (16 = tamano de
    fuente)
grid on % muestre cuadricula
axis equal % proporcion de los ejes iguales
axis([-1000 1000 -900 900  -100 1000]); % limites de la figura
cameratoolbar % barra de herramientas de carmara
view([0,0]) %para verlo desde z hacia el plano xy
camlight('left') %efectos de luces para ver mejor la imagen
camproj('perspective') % VISTA DE PERSPECTIVA DE CAMARA
title('Demo') %TITULOS
xlabel('x')
ylabel('y')
zlabel('z')
```

A.4 Código para cargar la cinemática del robot *Scorbot ER-VII*

Se muestra el código para poder cargar las ecuaciones referentes a la cinemática directa e inversa para el robot *Scorbot ER-VII*, a la tarjeta controladora **KFLOP**, se seguiría la misma lógica para cargar cualquier otra cinemática. El programa está escrito en **código C++** utilizando el Software **Visual Studio Community 2022**, en su **versión 17.3.4.**

```
1  #include "stdafx.h"
2  #include "KinematicsScorbotERVII.h"
3  #include <iostream>
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include "canon.h"
7  #define sqr(x) ((x)*(x))
8  ////////////////////////////////////
9  // Construction/Destruction
10 ////////////////////////////////////      CKinematicsScorbotERVII
       ::CKinematicsScorbotERVII()
11 {
12 L1 = 358.5 / (10.0 * 2.54);
13 L2 = 50 / (10.0 * 2.54);//60
14 L3 = 36 / (10.0 * 2.54);
15 L4 = 300 / (10.0 * 2.54);
16 L5 = 250 / (10.0 * 2.54);
17 L6 = 95 / (10.0 * 2.54);
18 // copied from gepetto
19 }
20 CKinematicsScorbotERVII::~CKinematicsScorbotERVII()
21 {
22 }
23 int CKinematicsScorbotERVII::TransformCADtoActuators(double x,
       double y, double z, double a, double b, double c, double*
       Acts, bool NoGeo)
24 {
25 GeoCorrect(x, y, z, &x, &y, &z);
26 double  ux = 1;
27 double  uy = 0;
28 double  uz = 0;
29 double  vx = 0;
30 double vy = -1;
31 double  vz = 0;
32 double wx = 0;
33 double wy = 0;
34 double wz = -1;
35 double px = x ;//+ a11+a33;
```

```
36 double py = y ;//-d22-d33;
37 double pz = z ;// +d11+a22-d55;
38 double theta1 = 0.0;
39 double theta2 = 0.0;
40 double theta3 = 0.0;
41 double theta4 = 0.0;
42 double theta5 = 0.0;
43 double  theta234 = 0.0;
44 double k1 = 0.0;
45 double k2 = 0.0;
46 double  psi1 =- PI;//giro en x
47 double  psi2 = b * PI / 180;//giro  en y
48 double  psi3 = c;
49 theta1 = atan2(py, px) - atan2(-L3, sqrt((px * px) + (py * py) -
       (L3) * (L3)));
50 psi3 = theta1;
51 ux = cos(psi2) * cos(psi3);
52 uy = cos(psi2) * sin(psi3);
53 uz = -sin(psi2);
54 vx = cos(psi3) * sin(psi1) * sin(psi2) - cos(psi1) * sin(psi3);
55 vy = cos(psi1) * cos(psi3) + sin(psi1) * sin(psi2) * sin(psi3);
56 vz = cos(psi2) * sin(psi1);
57 wx = sin(psi1) * sin(psi3) + cos(psi1) * cos(psi3) * sin(psi2);
58 wy = cos(psi1) * sin(psi2) * sin(psi3) - cos(psi3) * sin(psi1);
59 wz = cos(psi1) * cos(psi2);
60 theta5 = atan2(sin(theta1) * ux - cos(theta1) * uy, sin(theta1)
      * vx - cos(theta1) * vy);
61 theta234 = atan2(-uz, (cos(theta1) * ux + sin(theta1) * uy));
62 k1 = px * cos(theta1) + py * sin(theta1) - L2 + L6 * sin(
     theta234);
63 k2 = -pz + L1 - cos(theta234) * L6;
64 theta3 = acos(((k1 * k1 + k2 * k2 - L5 * L5 - L4 * L4) / (2 * L4
       * L5)));
65 theta2 = atan2(k2 * (L5 * cos(theta3) + L4) - k1 * L5 * sin(
     theta3), k1 * (L5 * cos(theta3) + L4) + k2 * L5 * sin(theta3)
     );
66 theta4 = theta234 - theta2 - theta3;
67 double the1 = (theta1 * 180.00 / PI);
68 double the5 = (theta5 * 180.00 / PI);
69 double the3 = (theta3 * 180.00 / PI) - 90.0;
70 double the2 = (theta2 * 180.00 / PI) + 90.0;
71 double the4 = (theta4 * 180.00 / PI);
72 double the6 = (theta5 * 180.00 / PI);
73 if (the1 > 125) {
74 the1 = 125;
75 {AfxMessageBox("L{\'i}mite m{\'a}ximo de giro para la articulaci
     {\'o}n 1 \r",
76 MB_ICONEXCLAMATION | MB_OK);
77 }
```

```
78 else␣if(the1␣<␣-125)␣{
79 the1␣=␣-125;
80 AfxMessageBox("L{\'i}mite m{\'i}nimo␣de␣giro␣para␣la␣articulaci
       {\'o}n 1\r",
81 MB_ICONEXCLAMATION | MB_OK);
82 }
83 else {}
84 if (the2 > 150) {
85 the2 = 150;
86 AfxMessageBox("L{\'i}mite␣m{\'a}ximo de giro para la articulaci
       {\'o}n␣2\r",
87 MB_ICONEXCLAMATION␣|␣MB_OK);␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
88 }
89 else␣if␣(the2␣<␣-35)␣{
90 the2␣=␣-35;
91 AfxMessageBox("L{\'i}mite m{\'i}nimo␣de␣giro␣para␣la␣articulaci
       {\'o}n 2\r",
92 MB_ICONEXCLAMATION | MB_OK);
93 }
94 else {}
95 if (the3 > 50) {
96 the3 = 50;
97 AfxMessageBox("L{\'i}mite␣m{\'a}ximo de giro para la articulaci
       {\'o}n␣3\r",
98 MB_ICONEXCLAMATION␣|␣MB_OK);␣␣␣␣␣␣␣␣␣␣␣
99 }
100 else␣if␣(the3␣<␣-179)␣{
101 the3␣=␣-179;
102 AfxMessageBox("L{\'i}mite m{\'i}nimo␣de␣giro␣para␣la␣articulaci
       {\'o}n 3\r",
103 MB_ICONEXCLAMATION | MB_OK);
104 }
105 else {}
106 if (the4> 5) {
107 the4 = 5;
108 AfxMessageBox("L{\'i}mite␣m{\'a}ximo de giro para la articulaci
       {\'o}n␣4\r",
109 MB_ICONEXCLAMATION␣|␣MB_OK);
110 }
111 else␣if␣(the4␣<␣-180)␣{
112 the4=␣-180;
113 AfxMessageBox("L{\'i}mite m{\'i}nimo␣de␣giro␣para␣la␣articulaci
       {\'o}n 4\r",
114 MB_ICONEXCLAMATION | MB_OK);
115 }
116 else {}
117 if (the5 > 180) {
118 the5 = 180;
119 AfxMessageBox("L{\'i}mite␣m{\'a}ximo de giro para la articulaci
```

```
    {\'o}n 5\r",
120 MB_ICONEXCLAMATION | MB_OK);
121 }
122 else if (the5 < -180) {
123 the5= -180;
124 AfxMessageBox("L{\'i}mite m{\'i}nimo de giro para la articulaci
    {\'o}n 5\r",
125 MB_ICONEXCLAMATION | MB_OK);
126 }
127 else {}
128 Acts[0] = the1 * m_MotionParams.CountsPerInchX;
129 Acts[1] = the2 * m_MotionParams.CountsPerInchY;
130 Acts[2] = the3 * m_MotionParams.CountsPerInchZ;
131 Acts[3] = the4 * m_MotionParams.CountsPerInchA;
132 Acts[4] = the5 * m_MotionParams.CountsPerInchB;
133 Acts[5] = the6 * m_MotionParams.CountsPerInchC;
134 return 0;
135 }
136 int CKinematicsScorbotERVII::TransformActuatorstoCAD(double*
    Acts, double* xr, double* yr, double* zr, double* ar, double*
     br, double* cr, bool NoGeo)
137 {
138 double aux1 = Acts[0] / m_MotionParams.CountsPerInchX;
139 double aux2 = Acts[1] / m_MotionParams.CountsPerInchY;
140 double aux3 = Acts[2] / m_MotionParams.CountsPerInchZ;
141 double aux4 = Acts[3] / m_MotionParams.CountsPerInchA;
142 double aux5 = Acts[4] / m_MotionParams.CountsPerInchB;
143 double aux6 = Acts[5] / m_MotionParams.CountsPerInchC;
144 double theta1 = (aux1 * PI / 180.00);
145 double theta2 = (aux2 * PI / 180.00) - PI / 2.0;
146 double theta3 = (aux3 * PI / 180.00) + PI / 2.0;
147 double theta4 = (aux4 * PI / 180.00);
148 double theta5 = aux5;
149 double theta6= aux6;
150 double xx = -cos(theta1) * sin(theta2 + theta3 + theta4) * L6 +
    cos(theta1) * (L5 * cos(theta2 + theta3) + L4 * cos(theta2))
    + L3 * sin(theta1) + L2 * cos(theta1);
    double yy = -sin(theta1) * sin(theta2 + theta3 + theta4) * L6
     + sin(theta1) * (L5 * cos(theta2 + theta3) + L4 * cos(theta2
    )) - L3 * cos(theta1) + L2 * sin(theta1);
151 double zz = -cos(theta2 + theta3 + theta4) * L6 - (L5 * sin(
    theta2 + theta3) + L4 * sin(theta2)) + L1;
152 // find the coordinates (x2,y2) - the end point
153 *xr = xx ;//-(a11 + a33);
154 *yr = yy ;// -(-d22 - d33);
155 *zr = zz ;//+d55-d11-a22;
156 *ar = aux6;
157 *br = (aux2 + aux3 + aux4);
158 * cr = aux1;
```

```
159 return 0;
160 }
161 // Check for out of range for ACos
162 double CKinematicsScorbotERVII::Confangle(double psi1, double
    psi2, double psi3)
163 {
164 return 0;
165 }
```

A.5 Creación de la clase cinemática

Se crea la **clase** KinematicsScorbotERVII, en el archivo **.h**. Se define la clase a utilizar y se declaran las constantes, es decir, las longitudes de los eslabones, en este caso son las dimensiones de los eslabones del robot *Scorbot ER-VII*. También se declaran los métodos (funciones). El programa está escrito en **código C++** utilizando el Software ***Visual Studio Community* 2022**, en su **versión 17.3.4.**

```
1  // KinematicsScorbotERVII.h: interface for the CKinematicsScara
       class.
2  //
3  #if !defined(
4  AFX_KinematicsScorbotERVII
5  _H__876A0A72_6EC3_48D0_9040_60AE3DA2F3C7__INCLUDED_)
6  #define AFX_KinematicsScorbotERVII
7  _H__876A0A72_6EC3_48D0_9040_60AE3DA2F3C7__INCLUDED_
8  #if _MSC_VER > 1000
9  #pragma once
10 #endif // _MSC_VER > 1000
11 #include "stdafx.h"
12 class CKinematicsScorbotERVII : public CKinematics
13 {
14 public:
15 CKinematicsScorbotERVII();
16 virtual ~CKinematicsScorbotERVII();
17 virtual int TransformCADtoActuators(double x, double y, double z
       , double a, double b, double c, double* Acts, bool NoGeo =
       false);
18 virtual int TransformActuatorstoCAD(double* Acts, double* x,
       double* y, double* z, double* a, double* b, double* c, bool
       NoGeo = false);
19 double CKinematicsScorbotERVII::Confangle(double psi_1, double
       psi_2, double psi_3);
20 double L1;
21 double L2;
22 double L3;
23 double L4;
```

```
24 double L5;
25 double L6;
26
27 double a1;
28 double a2;
29 double a3;
30 double d1;
31 double d2;
32 double d3;
33 double d5;
34
35 double ny;
36 double nz;
37 double ox;
38 double oy;
39 double oz;
40
41 double  ax;
42 double  ay;
43 double  az;
44 double ux;
45 double uy;
46 double uz;
47
48 double vx;
49 double vy;
50 double vz;
51
52 double  wx;
53 double  wy;
54 double  wz;
55 double psi1;
56 double psi2;
57 double psi3;
58 double auxp;
59 */
60 };
61 #endif // !defined(AFX_KinematicsScorb
```

A.6 Paro de emergencia

El siguiente código es necesario para implementar el paro de emergencia. También es útil para agregar algunos botones externos que pueden servir para activar funciones principales como correr un programa, detenerse de forma gradual el movimiento de la máquina entre otras funcionalidades. El código debe cargarse en la tarjeta controladora **KFLOP**, en el **Hilo 1**. Se utiliza el software ***KmotionCNC*** en su

versión 435h, el código es escrito en **lenguaje C++**, donde se utilizan también algunas funciones propias de ***Dynomotion***.

```
#include "KMotionDef.h"
#define TMP 10 // which spare persist to use to transfer data
#include "KflopToKMotionCNCFunctions.c"//para enviar dialogos de
    advertencias
#define FEEDHOLDBIT 46
#define CYCLESTARTBIT 47
#define ESTOP 26
#define HALTBIT 27
#define RESTARTBIT 28
#define ZEROALLBIT 29
// function prototypes for compiler
int DoPC(int cmd);
int DoPCFloat(int cmd, float f);
int Debounce(int n, int *cnt, int *last, int *lastsolid);
// state variables for switch debouncing
int flast=0,flastsolid=-1,fcount=0;
int clast=0,clastsolid=-1,ccount=0;
int elast=0,elastsolid=-1,ecount=0;
int hlast=0,hlastsolid=-1,hcount=0;
int rlast=0,rlastsolid=-1,rcount=0;
int zlast=0,zlastsolid=-1,zcount=0;
main()
{   int result;
        for (;;) // loop forever
        {
                WaitNextTimeSlice();

                // Handle FeedHold/Resume
                result = Debounce(ReadBit(FEEDHOLDBIT),&fcount,&
                    flast,&flastsolid);
                if (result == 1)
                {
                        if (CS0_StoppingState == 0)
                        StopCoordinatedMotion();
                        else
                        ResumeCoordinatedMotion();
                }

                // Handle Cycle Start
                result = Debounce(ReadBit(CYCLESTARTBIT),&ccount
                    ,&clast,&clastsolid);
                if (result == 1)
                {
                        DoPC(PC_COMM_EXECUTE);
                }

```

```
44                  // Handle ESTOP
45                  result = Debounce(ReadBit(ESTOP),&ecount,&elast
                        ,&elastsolid);
46                  if (result == 1)
47                  {     MsgBox("EMERGENCY STOP \n", MB_ICONHAND|
                        MB_OK);// Abre un cuadro de advertencia
48
49                        DoPC(PC_COMM_ESTOP);
50                  }
51
52                  // Handle HALT
53                  result = Debounce(ReadBit(HALTBIT),&hcount,&
                        hlast,&hlastsolid);
54                  if (result == 1)
55                  {
56                        DoPC(PC_COMM_HALT);
57                  }
58
59                  // Handle RESTART
60                  result = Debounce(ReadBit(RESTARTBIT),&rcount,&
                        rlast,&rlastsolid);
61                  if (result == 1)
62                  {
63                        DoPC(PC_COMM_RESTART);
64                  }
65
66                  // Handle ZERO ALL
67                  result = Debounce(ReadBit(ZEROALLBIT),&zcount,&
                        zlast,&zlastsolid);
68                  if (result == 1)
69                  {
70                        DoPCFloat(PC_COMM_SET_X,0.0);
71                        DoPCFloat(PC_COMM_SET_Y,0.0);
72                        DoPCFloat(PC_COMM_SET_Z,0.0);
73                  }
74            }
75 }
76
77 // Put a Float as a parameter and pass the command to the App
78 int DoPCFloat(int cmd, float f)
79 {
80            int result;
81            persist.UserData[PC_COMM_PERSIST+1] = *(int*)&f;
82            return DoPC(cmd);
83 }
84
85
86 // Pass a command to the PC and wait for it to handshake
87 // that it was received by either clearing the command
```

```
88 // or changing it to a negative error code
89 int DoPC(int cmd)
90 {
91         int result;
92
93         persist.UserData[PC_COMM_PERSIST]=cmd;
94
95         do
96         {
97                 WaitNextTimeSlice();
98         }while (result=persist.UserData[PC_COMM_PERSIST]>0);
99
100        printf("Result = %d\n",result);
101
102        return result;
103 }
104
105
106
107
108 // Debounce a bit
109 //
110 // return 1 one time when first debounced high
111 // return 0 one time when first debounced low
112 // return -1 otherwise
113 #define DBTIME 300
114
115 int Debounce(int n, int *cnt, int *last, int *lastsolid)
116 {
117         int v = -1;
118
119         if (n == *last)  // same as last time?
120         {
121                 if (*cnt == DBTIME-1)
122                 {
123                         if (n != *lastsolid)
124                         {
125                                 v = *lastsolid = n;  // return
                                    debounced value
126                         }
127                 }
128                 if (*cnt < DBTIME)       (*cnt)++;
129         }
130         else
131         {
132                 *cnt = 0;  // reset count
133         }
134         *last = n;
135         return v;
```

```
136 }
```

A.7 Programa para posicionar en Home el robot *Scorbor ER-VII*

El Robot *Scorbot ER-VII* debido a que no tiene un sistema de memoria para guardar las posiciones de las articulaciones, cuando se le corta el suministro de energía, es necesario que de alguna forma se posicione en una pose donde se conozca la posición real de cada articulación y a partir de allí mover de forma correcta las articulaciones. El siguiente código es necesario para posicionar en la posición (Home) el *Scorbor ER-VII*. Se utiliza ***KmotionCNC*** en su **versión 435h**, el código es escrito en **lenguaje C++**, donde se utilizan también algunas funciones propias de ***Dynomotion***.

```
1  #include "KMotionDef.h"
2
3  #define Homevel_0 6000    //velocidad a la que se movera el eje
       para encontrar home
4  #define Homevel_inicio_0 6000  //velocidad a la que se movera el
       eje para encontrar home
5  #define Homevel_inicio_1 6000
6  #define Homevel_inicio_2 6000
7  #define Homevel_inicio_3 6000
8
9
10 #define AjusHome0  78925 //ajuste para llegar a la posici{\'o}n
       de home_1 /135 GRADOS  79043
11 //78518
12 #define AjusHome1   15246
13 #define AjusHome2   89039
14 #define AjusHome3   0
15
16 void Home_0(void);
17 void Home_1(void);
18 void Home_2(void);
19 void Home_3(void);
20 void Home_4(void);
21
22 int main()
23 {
24
25         DisableAxis(0);
26         DisableAxis(1);
27         DisableAxis(2);
28         DisableAxis(3);
```

```
29          DisableAxis(4);
30          DisableAxis(5);
31          Home_1();
32          Home_2();
33          Home_3();
34          Home_0();
35          EnableAxis(4);
36          EnableAxis(5);
37          //Home_4();
38          //printf("bit %d \n",ReadBit(BitHome1) );
39          ch0->Vel=10000;
40          ch1->Vel=10000;
41          ch2->Vel=10000;
42          ch3->Vel=10000;
43          ch4->Vel=10000;
44
45  }
46
47
48  void Home_0(){
49          EnableAxis(0);
50          Jog(0,-Homevel_inicio_0);
51          Delay_sec(20);
52          Jog(0,0);
53          Zero(0);
54          ch0->Vel=6000;
55          Zero(0);
56          Move(0,AjusHome0);
57          while (!CheckDone(0));
58
59  }
60
61  void Home_1(){
62          EnableAxis(1);
63          Jog(1,-Homevel_inicio_1);
64          Delay_sec(20);
65          Jog(1,0);
66          Zero(1);
67          ch1->Vel=6000;
68          Move(1,AjusHome1);
69          while (!CheckDone(1));
70  }
71
72
73  void Home_2(){
74          EnableAxis(2);
75          Jog(2,-Homevel_inicio_2);
76          Delay_sec(20);
77          Jog(2,0);
```

```
78          Zero(2);
79          ch2->Vel=6000;
80          Move(2,AjusHome2);
81          while (!CheckDone(2));
82  }
83
84  void Home_3(){
85          EnableAxis(3);
86          Jog(3,Homevel_inicio_3);
87          Delay_sec(20);
88          Jog(3,0);
89          Zero(3);
90          ch3->Vel=6000;
91          Move(3,AjusHome3);
92          while (!CheckDone(3));
93
94  }
```

A.8 Código para la comprobación de ecuaciones utilizando *Matlab*, para *Scorbor ER-VII*

El siguiente código es escrito en ***Matlab*** **versión R2020b**, utilizando ***Robotics Toolbox***. Es importante comprobar que nuestras ecuaciones obtenidas para la cinemática directa e inversa fueron calculadas de manera correcta para poder cargarlas a la tarjeta KFLOP y no nos genere ningún movimiento extraño el robot al momento de correr algún código G.

```
1  clc
2  clear all
3  H0=eye(4);
4  anguloA=-45*(pi/180) %valor dado por el usuario
5  psi1=-pi             %giro en x constante/ no gira la muneca
6  psi2 =anguloA        %giro  en y
7  %psi3=30*(pi/180)    %giro  en Z
8  alpha1=-pi/4;
9  alpha2=0;
10 alpha3=0;
11 alpha4=pi/2;
12 alpha5=0;
13 a1=50;     %L2
14 a2=300;    %L4
15 a3=250;    %L5
16 d1=358.5; %L1
17 d2=36      %L3
18 d5=95      %L6
```

```
19 px=300
20 py=100;%-36
21 pz=358.5-95+300
22 hold on
23 if (px^2+py^2-(d2*d2)) > 0,
24 theta1=(atan2(py,px) - atan2(-d2,sqrt(px^2+py^2-(d2*d2))));
25 psi3=theta1 % Se suma el {\'a}ngulo theta 1 al giro en z  ;
26 else
27 disp('Punto no alcanzable (muy cercano al la base principal)');
28 Error=1;
29 return;
30 end;
31 nx=cos(psi2)*cos(psi3);
32 ny=cos(psi2)*sin(psi3);
33 nz=-sin(psi2);
34 ox=cos(psi3)*sin(psi1)*sin(psi2)-cos(psi1)*sin(psi3);
35 oy=cos(psi1)*cos(psi3)+sin(psi1)*sin(psi2)*sin(psi3);
36 oz=cos(psi2)*sin(psi1);
37 ax=sin(psi1)*sin(psi3)+cos(psi1)*cos(psi3)*sin(psi2);
38 ay=cos(psi1)*sin(psi2)*sin(psi3)-cos(psi3)*sin(psi1);
39 az=cos(psi1)*cos(psi2);
40 %theta1=atan2(py,px)-atan2(-L3,sqrt((px*px)+(py*py)-(L3)*(L3)));
41 theta5=atan2(sin(theta1)*nx-cos(theta1)*ny,sin(theta1)*ox-cos(
       theta1)*oy);
42 theta234=atan2(-nz/cos(theta5),(cos(theta1)*nx+sin(theta1)*ny)/
       cos(theta5));
43 k1=px*cos(theta1)+py*sin(theta1)-a1+d5*sin(theta234);
44 k2=-pz+d1-cos(theta234)*d5;
45 aaaaaa=((k1*k1+k2*k2-a3*a3-a2*a2)/(2*a2*a3))
46 if abs((k1^2+k2^2-a3*a3-a2*a2)/(2*a2*a3))<= 1,
47 theta3=acos((k1^2+k2^2-a3*a3-a2*a2)/(2*a3*a2));
48 else
49 disp('Punto no alcanzable (externo al espacio alcazable)');
50 Error=2;
51 return;
52 end;
53 theta2=atan2(k2*(a3*cos(theta3)+a2)-k1*a3*sin(theta3),k1*(a3*cos
       (theta3)+a2)+k2*a3*sin(theta3));
54 theta4=theta234-theta2-theta3;
55 TT05=[cos(theta1)*cos(theta2+theta3+theta4)*cos(theta5)+sin(
       theta1)*sin(theta5), -cos(theta1)*cos(theta2+theta3+theta4)*
       sin(theta5)+sin(theta1)*cos(theta5),-cos(theta1)*sin(theta2+
       theta3+theta4),-cos(theta1)*sin(theta2+theta3+theta4)*d5+cos(
       theta1)*(a3*cos(theta2+theta3)+a2*cos(theta2))+d2*sin(theta1)
       +a1*cos(theta1);
56 sin(theta1)*cos(theta2+theta3+theta4)*cos(theta5)-cos(theta1)*
       sin(theta5),-sin(theta1)*cos(theta2+theta3+theta4)*sin(theta5
       )-cos(theta1)*cos(theta5),-sin(theta1)*sin(theta2+theta3+
       theta4),-sin(theta1)*sin(theta2+theta3+theta4)*d5+sin(theta1)
```

```
    *(a3*cos(theta2+theta3)+a2*cos(theta2))-d2*cos(theta1)+a1*sin
    (theta1);-sin(theta2+theta3+theta4)*cos(theta5),sin(theta2+
    theta3+theta4)*sin(theta5),-cos(theta2+theta3+theta4),-cos(
    theta2+theta3+theta4)*d5-(a3*sin(theta2+theta3)+a2*sin(theta2
    ))+d1;
0,0,0,1];
px
xx=-cos(theta1)*sin(theta2+theta3+theta4)*d5+cos(theta1)*(a3*
    cos(theta2+theta3)+a2*cos(theta2))+d2*sin(theta1)+a1*cos(
    theta1)
py
yy=-sin(theta1)*sin(theta2+theta3+theta4)*d5+sin(theta1)*(a3*cos
    (theta2+theta3)+a2*cos(theta2))-d2*cos(theta1)+a1*sin(theta1)
pz
zz=-cos(theta2+theta3+theta4)*d5-(a3*sin(theta2+theta3)+a2*sin(
    theta2))+d1
the1=(theta1*360)/(2*pi);
the5=(theta5*360)/(2*pi);
the3=(theta3*360)/(2*pi)-90;
the2=(theta2*360)/(2*pi)+90;
the4=(theta4*360)/(2*pi);
psi11=0.0;
psi22=the2+the3+the4;
psi33=the1;
T1=[cos(theta1),0,-sin(theta1),a1*cos(theta1);
sin(theta1),0,cos(theta1),a1*sin(theta1);
0,-1,0,d1;
0,0,0,1];
T2=[cos(theta2),-sin(theta2),0,a2*cos(theta2);
sin(theta2),cos(theta2),0,a2*sin(theta2);
0,0,1,-d2;
0,0,0,1];
T3=[cos(theta3),-sin(theta3),0,a3*cos(theta3);
sin(theta3),cos(theta3),0,a3*sin(theta3);
0,0,1,0;
0,0,0,1];
T3_3=[cos(theta3),-sin(theta3),0,a3*cos(theta3);
sin(theta3),cos(theta3),0,a3*sin(theta3);
0,0,1,0;
0,0,0,1];
T4=[cos(theta4),0,-sin(theta4),0;
sin(theta4),0,cos(theta4),0;
0,-1,0,0;
0,0,0,1];
T5=[cos(theta5),-sin(theta5),0,0;
sin(theta5),cos(theta5),0,0;
0,0,1,d5;
0,0,0,1];
T01=H0*T1;
```

```
T02=T1*T2;
T03=T1*T2*T3;
T04=T1*T2*T3*T4;
T05=H0*T1*T2*T3*T4*T5
%
figure(1)%Figura donde se va a trabajar
clf%limpio la grafica de la figura
hold on% permanezcan las gr{\'a}ficas conforme se agregan m{\'a}s figuras
[origen0]=Sistema3D(H0,300,'X_0','Y_0','Z_0',5);%PERMITE DIBUJAR UN PLANO
[origen1]=Sistema3D(T01,200,'X_1','Y_1','Z_1',5);%PERMITE DIBUJAR UN PLANO
[origen2]=Sistema3D(T02,200,'X_2','Y_2','Z_2',5);
[origen3]=Sistema3D(T03,200,'X_3','Y_3','Z_3',10);
[origen4]=Sistema3D(T04,200,'X_4','Y_4','Z_4',10);
[origen5]=Sistema3D(T05,200,'X_5','Y_5','Z_5',10);
%Eslab{\'o}n(origen0,origen1,'k',2)
cilindro(H0,50,50,0)%ALTURA, DIAMETRO Y desplazamiento
cilindro(T01,50,50,25)
cilindro(T02,50,25,25)
cilindro(T03,50,25,25)
cilindro(T04,50,25,25)
cilindro(T05,5,5,2.5)
%CARTECIANO EN 3d  con la matriz de transformaci{\'o}n  asociada HO
%(1 =magnitud de las  unidad de los ejes) (16 = tamano de fuente)
grid on% muestre cuadr{\'i}cula
axis equal % proporci{\'o}n de los ejes iguales
axis([-1500 1500 -1500 1500 -1000 1000]);% l{\'i}mites de la figura
cameratoolbar% barra de herramientas de carmara
view([0,0])%para verlo desde z hacia el plano xy
camlight('left')%efectos de luces para ver mejor la imagen
camproj('perspective')
title('Demo') %Titulos
xlabel('x')
ylabel('y')
zlabel('z')
%*****************************
```

B.0 Características para motores de DC JGA25-370

En este anexo se describen las características principales de los motores utilizados en las primeras pruebas con el robot planar, de 2 grados de libertad. Se utilizaron

motores de DC marcado con el modelo JGA25-370, el cual está dotado con un encoder de dos canales el cual entrega 11 pulsos por revolución, cabe recalcar que se tiene una transmisión acoplada al eje del motor con una relación de 1 a 100, [Naylampmechatronics, 2022].

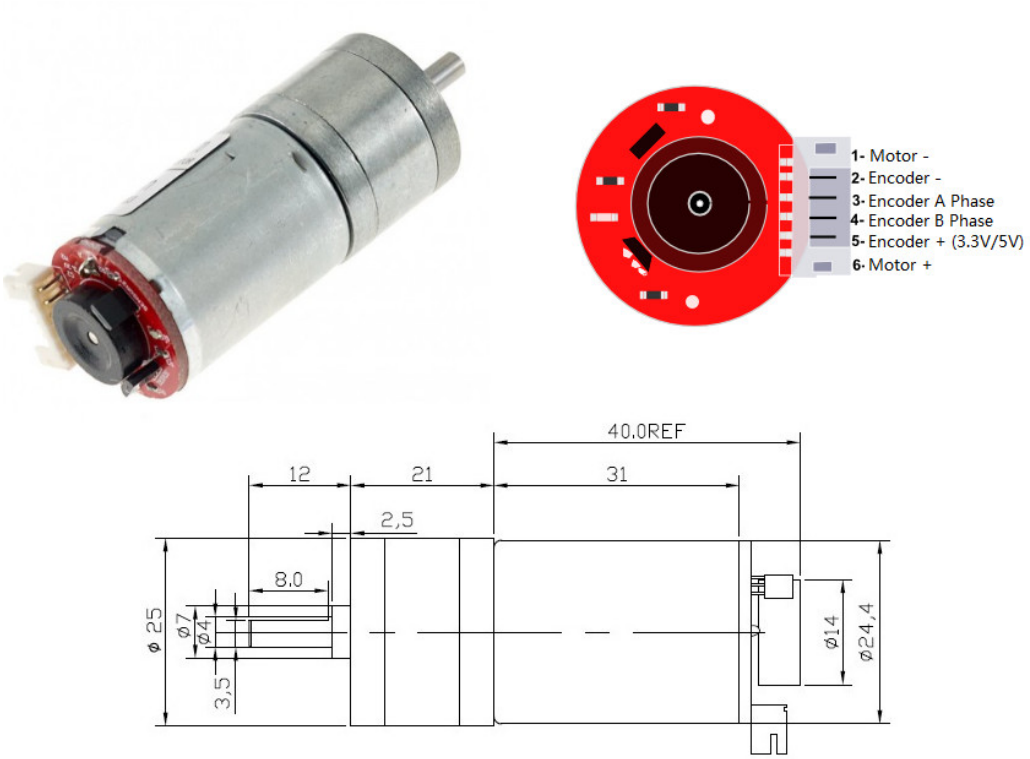


Figura 9.5: Características de motor de DC modelo JGA25-370.

Características técnicas

1. Voltaje de alimentación del motor: 6V - 12V DC
2. Voltaje de alimentación del encoder: 3.3V - 5V DC
3. corriente sin carga: 280mA
4. corriente nominal: 400mA
5. Velocidad de rotación: 6V - 165 RPM / 12V - 350 RPM
6. Relación en engranes: 1:100

7. Encoder:De efecto Hall

8. Peso: 95 gramos

B.1 Diagrama eléctrico para robot planar de 2 DGL

En la (Figura 9.6) se muestra el diagrama eléctrico, donde los motores se conectan a la tarjeta SnapAmp, los motores, por otro lado, se conectan el encoder al conector JP7 de la tarjeta KPLOP.

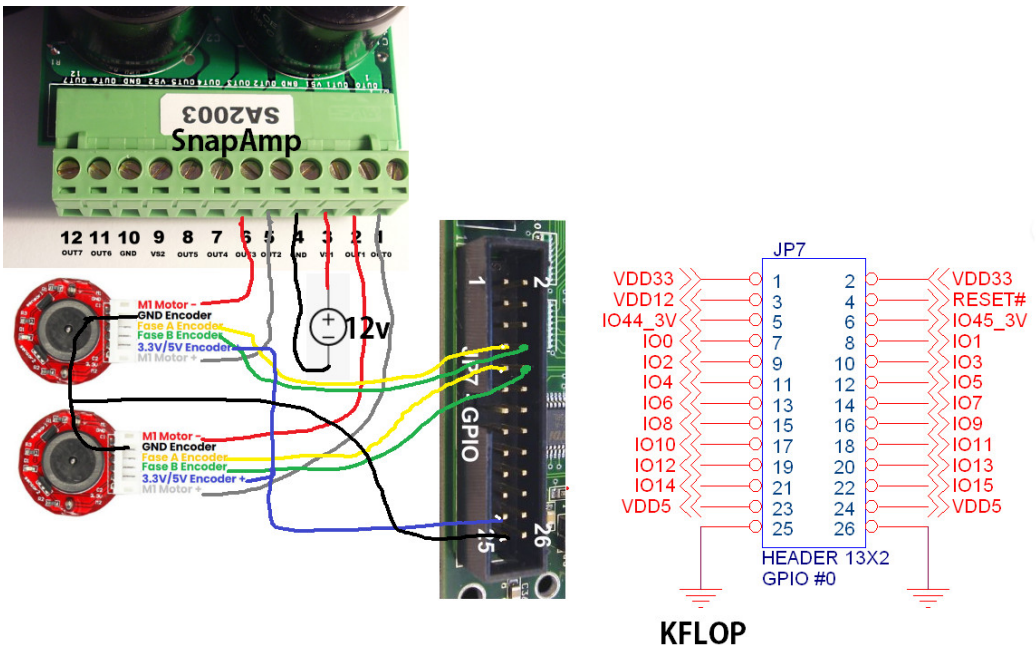


Figura 9.6: Diagrama de conexiones para el robot planar de 2 grados de libertad.

B.2 Conexiones eléctricas para el robot *Scorbot ER-VII*

En la (Figura 9.7) se muestra un diagrama general interno de las conexiones de los actuadores y de los motores para cada articulación. Está disponible un conector DB9 para cada articulación el cual interconecta las distintas señales, mostradas en la (Figura 9.11). Se utilizó una PCB anteriormente diseñada por otros alumnos la cual tiene la función de hacer más fácil la conexión de las señales hacia el controlador, se enumeraron las señales de cada articulación con un color rojo, (Figura

9.8).

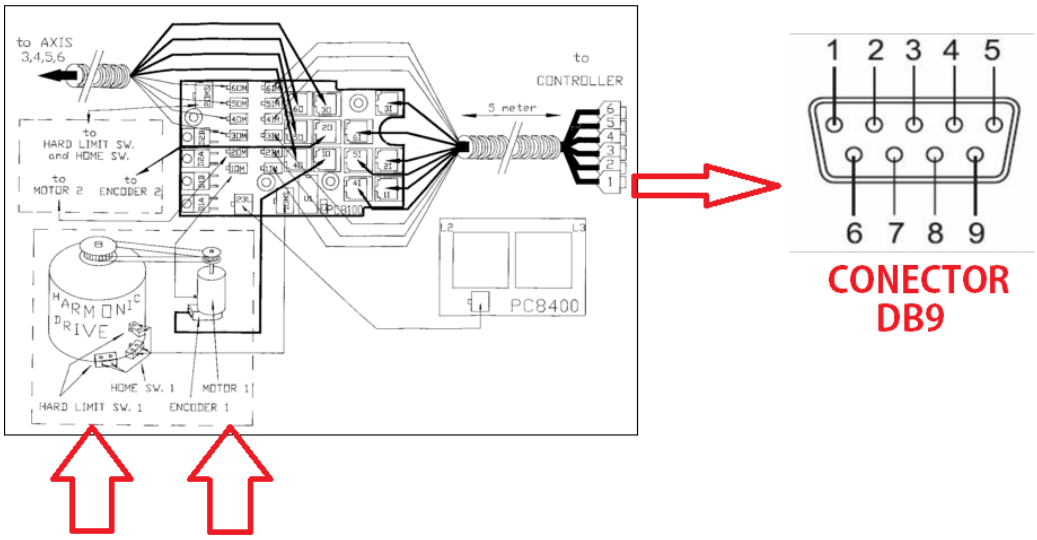


Figura 9.7: Diagrama de conexiones internas para el robot planar de 2 grados de libertad.

Los pines 1 y 9 de los conectores DB9 los cuales llevan las señales de los motores se conecta como se muestra en la (Figura 9.9), la fuente de alimentación de 12 V es únicamente para alimentar los motores las señales de control o de encoders están alimentados de forma independiente.

En la (Figura 9.10) se muestra la conexión de los encoders hacia la tarjeta SnapAmp.

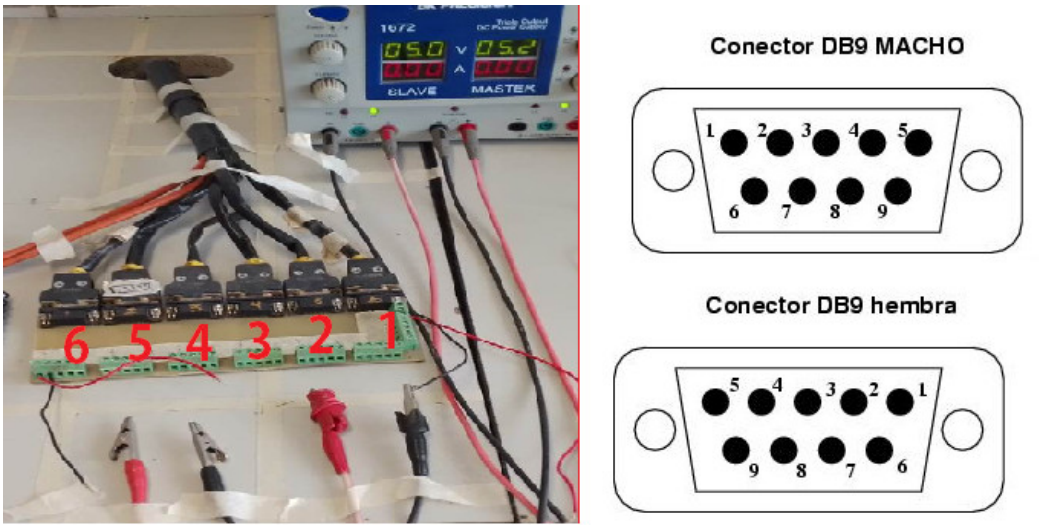


Figura 9.8: Diagrama de conexiones para el robot Scorbot ER-VII.

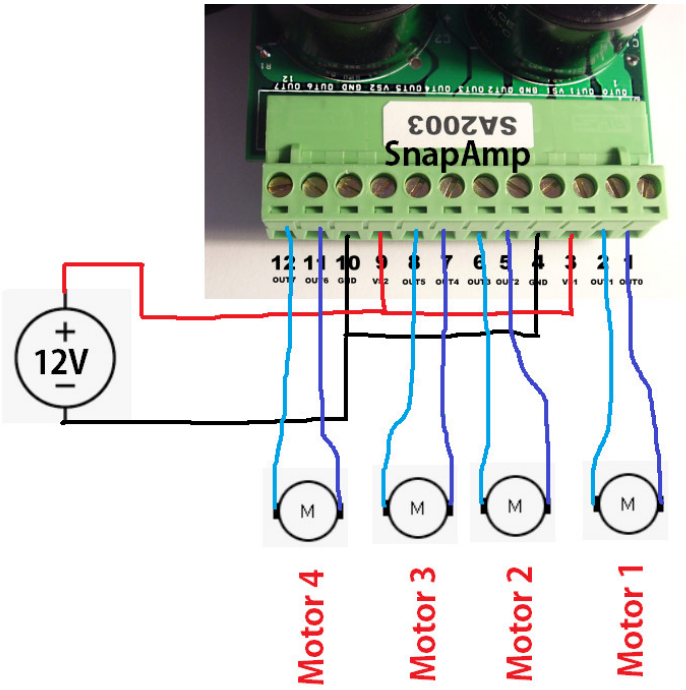


Figura 9.9: Señales de alimentación para motores.

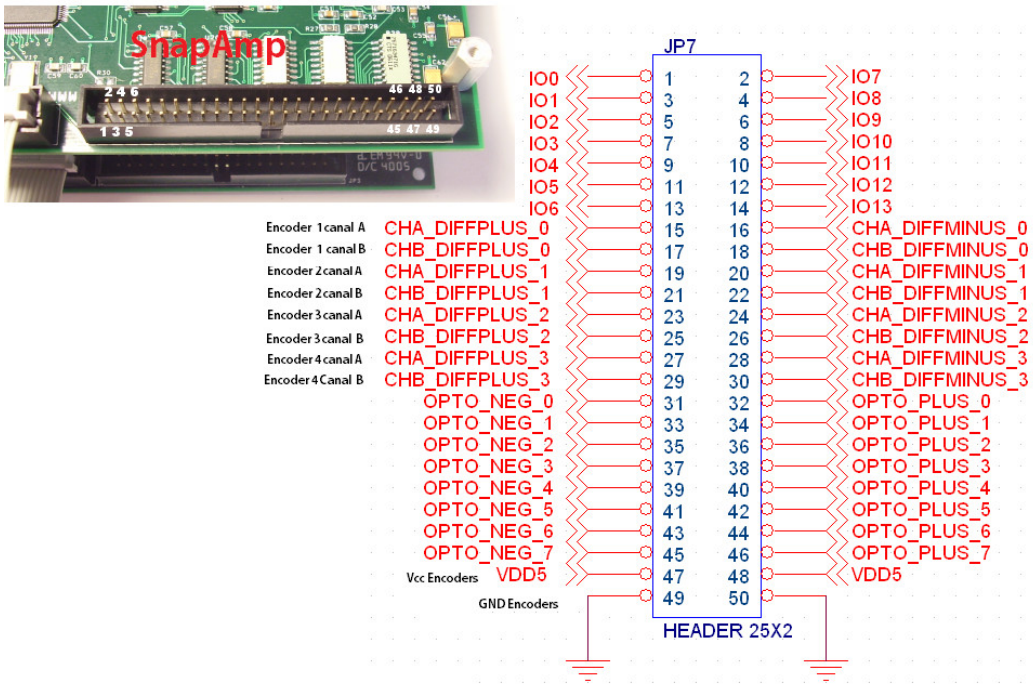


Figura 9.10: Conexión de señales de encoders hacia la tarjeta SnapAmp.

SCORBOT-ER VII Cables and Connector Pin Assignments				
Axis # (Robot Cable #)	D9 Connector Pin #	Wire Color	Connector to Robot PCB	Function
1	3	Red	Telephone	+5V
	4	Blue		Home switch
	5	Black		Common
	6	Green		Encoder channel A
	8	White		Encoder channel B
	9	Red	Molex	Motor 1 +
	1	Black		Motor 1 –
2	3	Red	Telephone	+5V
	4	Blue		Home switch
	5	Black		Common
	6	Green		Encoder channel A
	8	White		Encoder channel B
	9	Orange	Molex	Motor 2 +
	1	Brown		Motor 2 –
3	3	Red	Telephone	+5V
	4	Blue		Home switch
	5	Black		Common
	6	Green		Encoder channel A
	8	White		Encoder channel B
	9	Violet	Molex	Motor 3 +
	1	Yellow		Motor 3 –
4	3	Red	Telephone	+5V
	4	Blue		Home switch
	5	Black		Common
	6	Green		Encoder channel A
	8	White		Encoder channel B
	9	Blue	Molex	Motor 4 +
	1	Light Blue		Motor 4 –

Figura 9.11: Señales para todas las articulaciones del *Scorbot ER-VII.*

B.3 Diagrama para el efector final del *Scorbot ER-VII*

En las figuras (Figura 9.12) y (Figura 9.13) se muestran las dimensiones para el diseño de la herramienta para el robot *Scorbot ER-VII*. En este trabajo se utilizó el Software ***SolidWorks 2021*** en su **versión de prueba**.

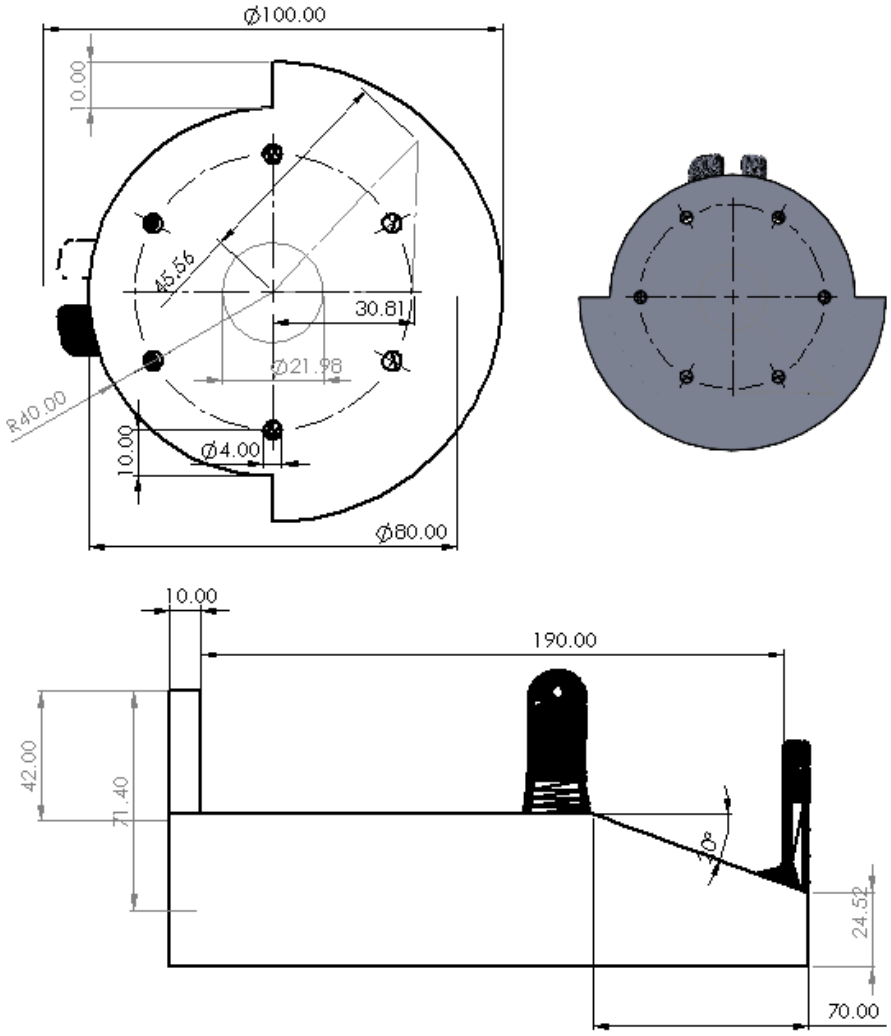


Figura 9.12: Dimensiones de la herramienta para el robot *Scorbot ER-VII*. (A)

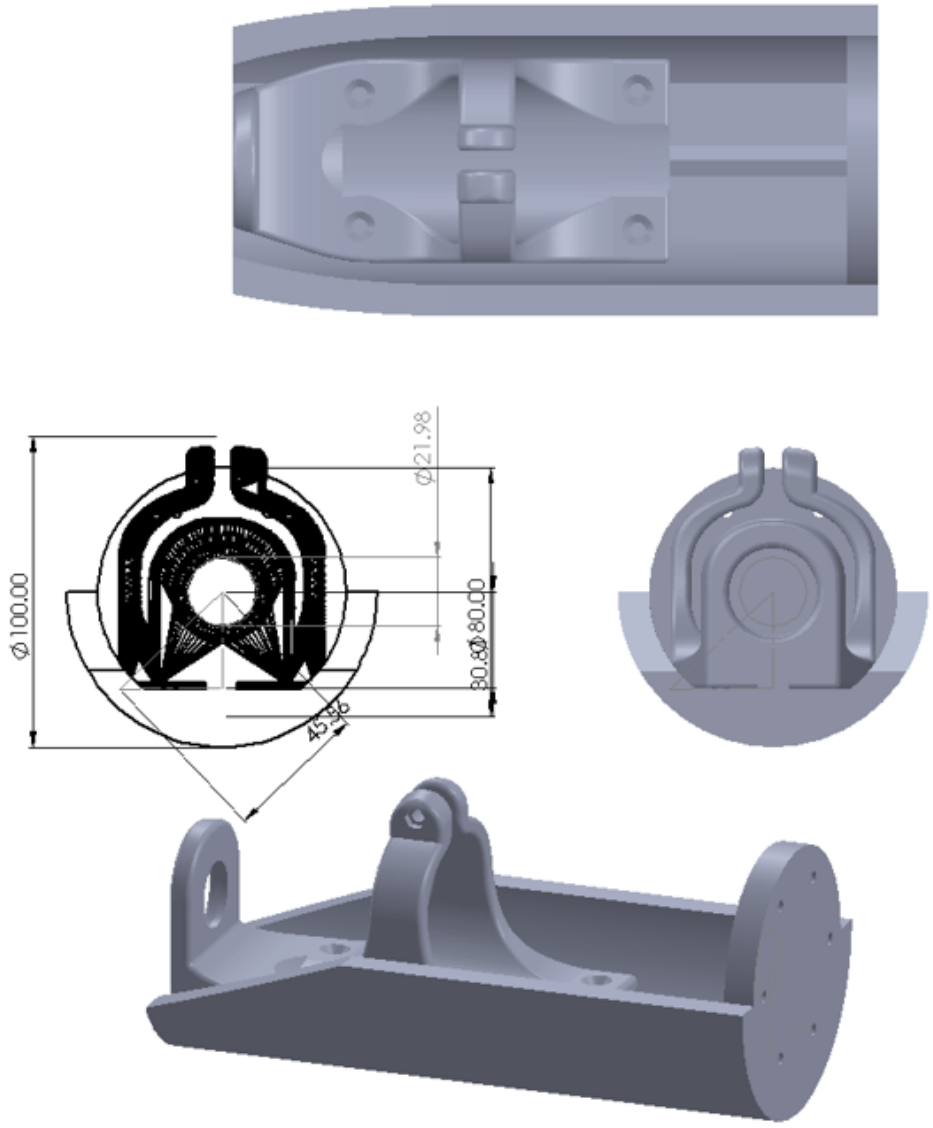


Figura 9.13: Dimensiones de la herramienta para el robot *Scorbot ER-VII.* (B)

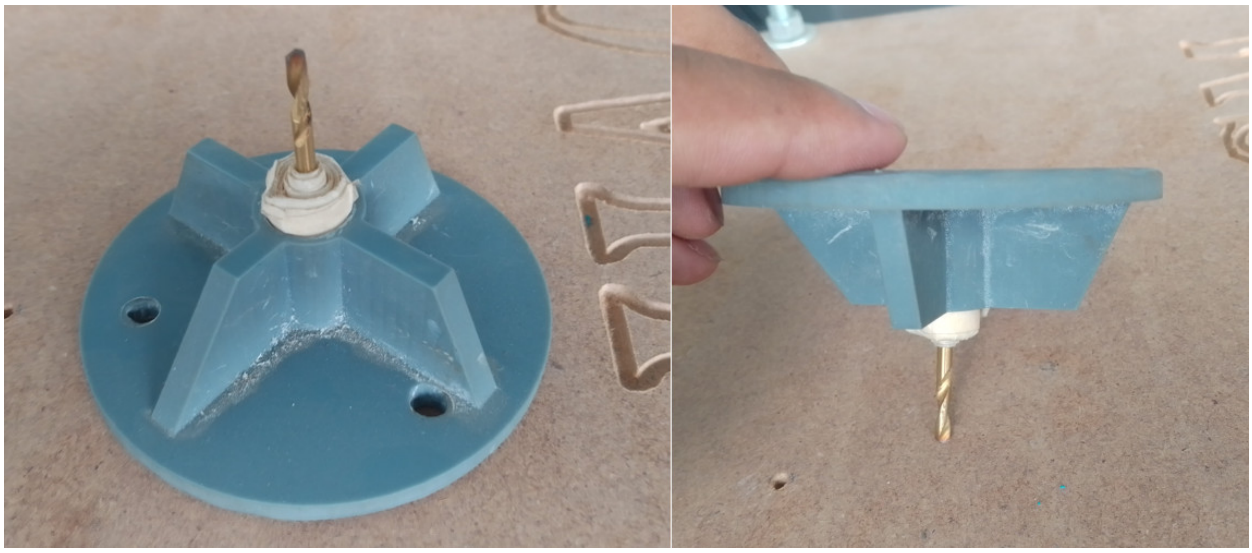


Figura 9.14: Pieza para colocación de láser en *Scorbot ER-VII.*

C.1 Generación código G con *Inkscape 1.3*

Para generar código G a partir de una imagen utilizamos el software ***Inkscape 1.3*** en su **versión de prueba**. Se muestran los pasos a seguir para generar el código G a partir de una imagen, en este caso utilizamos el logo "UACM" (Figura 9.15) como ejemplo. Se utiliza ***KmotionCNC*** en su **versión 435h** para poder simular y comprobar que el código G se genera de manera adecuada.

Figura 9.15: Logo UACM.

Paso 1. Ejecutamos el software Inkscape y se nos desplegará una ventana como se muestra en la (Figura 9.16), nos iremos a la opción archivo, luego a la opción importar, la cual desplegara otra ventana en la que seleccionaremos nuestra imagen.

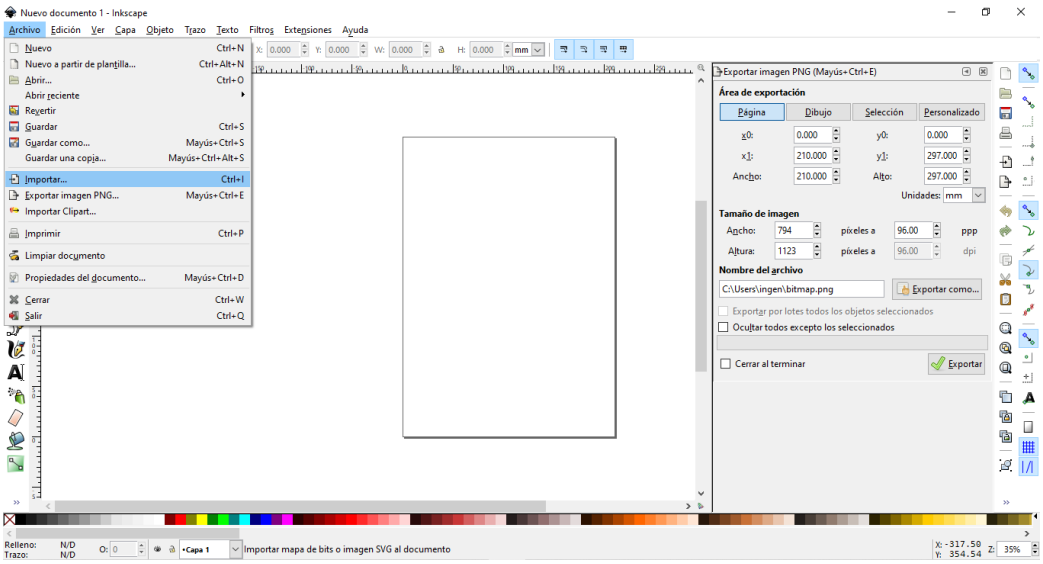


Figura 9.16: Software *Inkscape.*

Paso 2. Seleccionamos la imagen y luego seleccionamos la opción abrir, se nos

desplegara una ventana nueva en la cual daremos clic en aceptar sin modificar nada, (Figura 9.17).

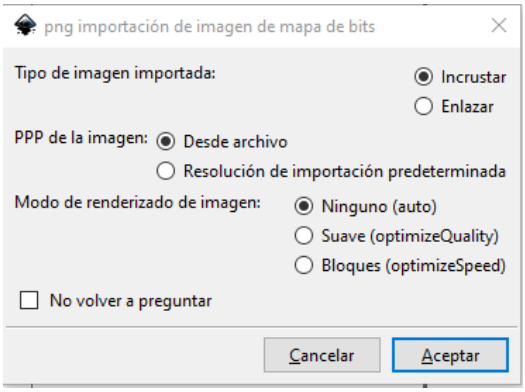


Figura 9.17: Ventana de importación.

Paso 3. Acomodamos la imagen dentro del área de trabajo, y nos dirigimos a la ventana trazo y seleccionamos la opción vectorizar mapas de bits, se desplegará otra ventana en la cual daremos clic en la opción actualizar, luego en la opción aceptar y finalmente cerramos la pestaña (Figura 9.18).

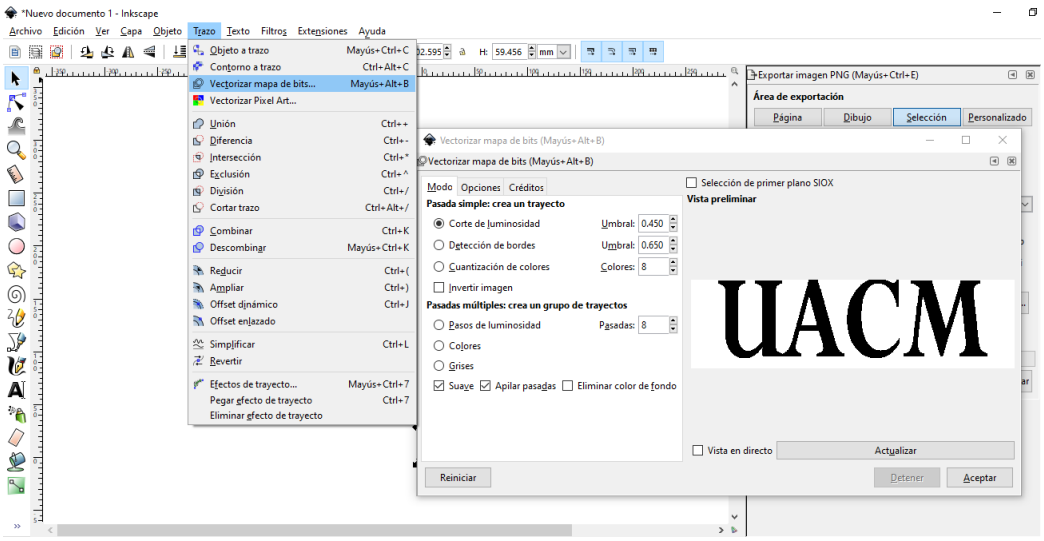


Figura 9.18: Ventana de importación de mapa de bits.

Paso 4. Procedemos a borrar la imagen original, se trabajará con la imagen que se nos desplegó. Nos dirigimos a la opción trazo, después a offset dinámico, luego nos iremos a la opción Extensiones y posteriormente a la opción Gcodetools y luego seleccionamos puntos de orientación en la cual nos desplegará un par de

coordenadas en nuestra (Figura 9.19), las cuales servirán para saber la orientación de nuestra imagen. Luego regresamos a la pestaña Gcodetools, después a biblioteca de herramientas, le damos clic a la opción cilindro, luego a aplicar y cerramos la pestaña. En seguida se nos abrirá una pestaña donde podemos modificar parámetros referentes a la herramienta (Figura 9.19).

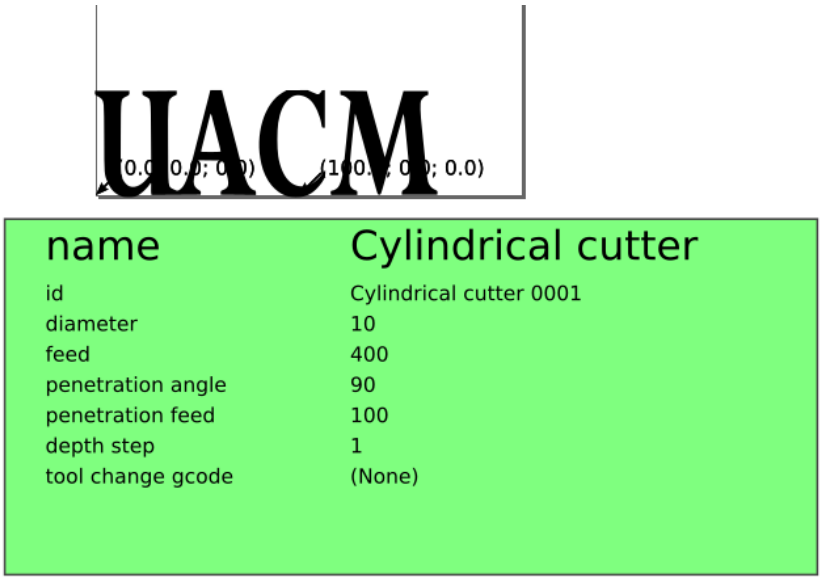


Figura 9.19: Ventana de configuración de parámetros generales.

Paso 5. Una vez colocado los parámetros de la herramienta, seleccionamos nuestra imagen, luego nos dirigimos a la opción Extensiones, Gcodetools y por último la opción Trayecto GCode, en la cual se nos despegara una pestaña donde podremos asignar el nombre a nuestro archivo, profundidad de corte, la altura en Z segura para movimientos G0 y la dirección donde se guardara nuestro archivo (Figura 9.20).

Ya generado el archivo que contiene el código G, se simula en el software *KmotionCNC* para estar seguros de que el código G se generó de manera correcta. En algunas ocasiones se deben modificar algunas líneas de código, en las Figuras (Figura 9.21) y (Figura 9.22) se pueden apreciar las trayectorias utilizando el visor de código G de *KmotionCNC*.

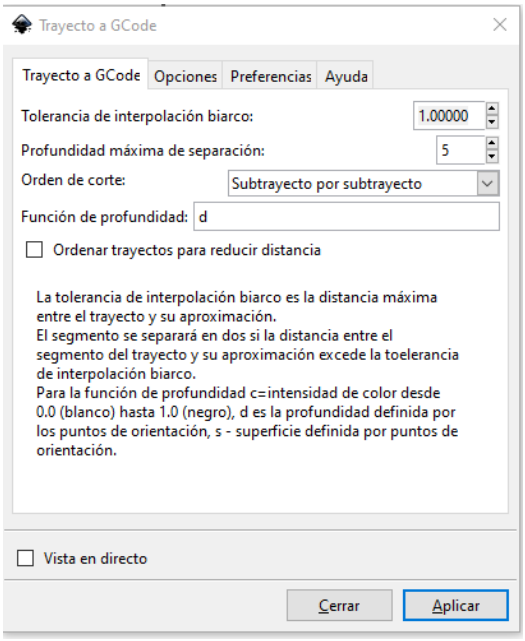


Figura 9.20: Ventana de configuración de parámetros de herramienta.

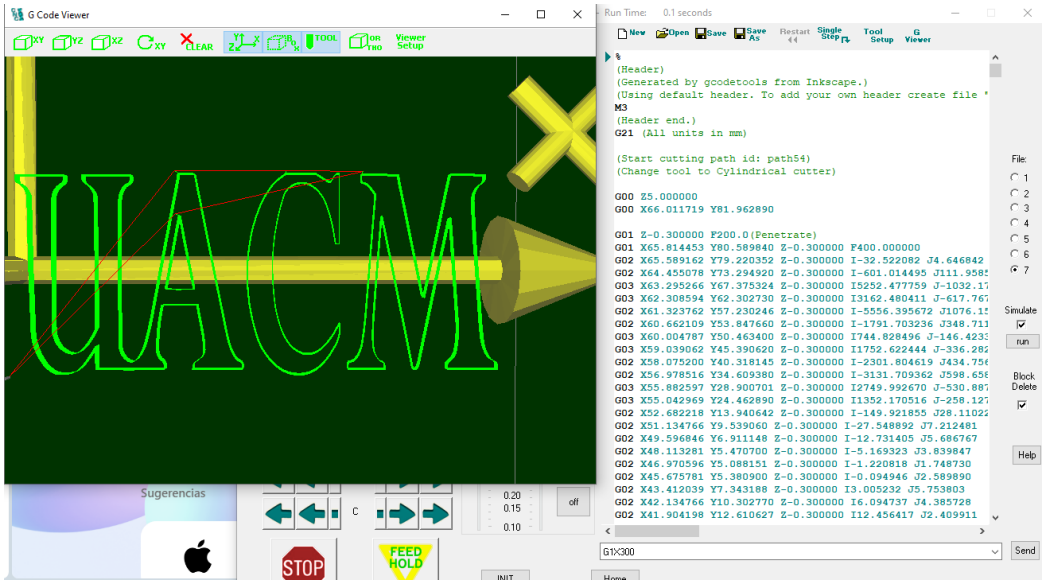


Figura 9.21: Simulación de Gcode utilizando *KmotionCNC*.

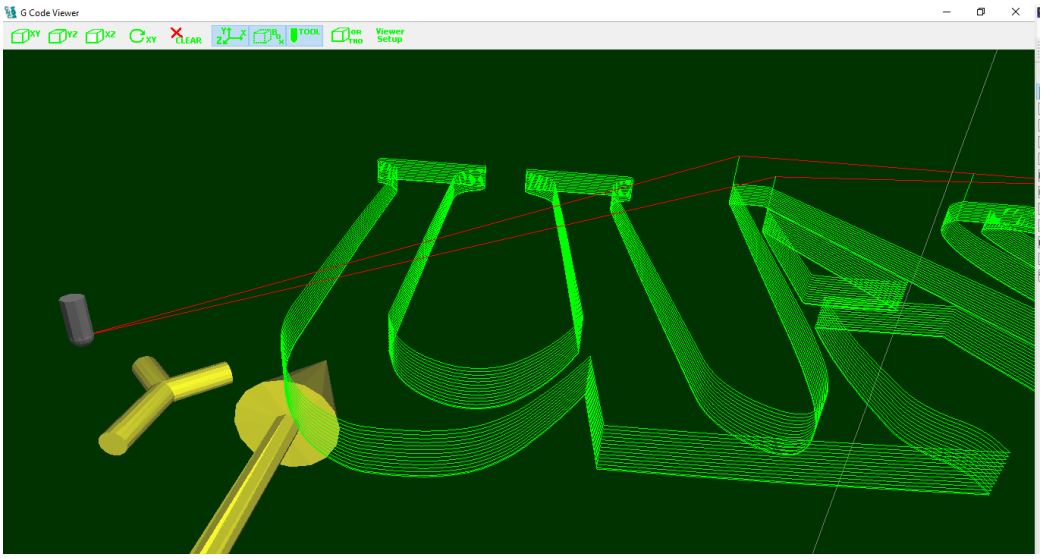


Figura 9.22: Simulación de trayectiorias Gcode.

C.2 Código G para puebas

En este Anexo se muestra el código G con el cual se realizaron las pruebas de movimiento para el robot *Scorbot ER-VII*. Se muestra el código G para un cuadrado de 5cmX5cm. Se utiliza ***KmotionCNC*** en su **versión 435h** para poder simular y comprobar que el código G se genera de manera adecuada.

```
G0 X0 Y0 Z0
F400
G1 X50
G1 Y50
G1 X0
G1 Y0
M2
```

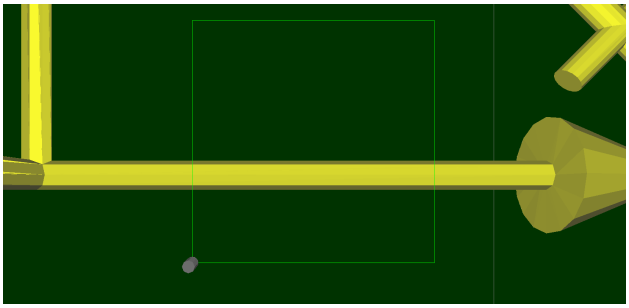


Figura 9.23: Simulación de cuadrado de 5cmX5cm en visor de Gcode, *Kmotion-CNC*.

Código G para circulo de diámetro 5cm

```
M3
(Header end.)
G21 (All units in mm)
G00 X0 Y0 Z0
F40
G0 X0 Y0 Z-20          (Baja en z 20mm)
G2 X50 Y0 I25 J0       (punto final del medio circulo X,Y)
                        (Centro del medio circulo I,J )
G2 X0 Y0 I-25 J0       (-25 En x moviendose sobre el )
M2                     (eje x en direccion hacia el Centro)
```

C.3 Instalación de *Simple Robotics Toolbox*

Este *Toolbox* se utiliza para representar diagramas cinemáticos para cualquier robot utilizando representaciones con sólidos en el espacio 3D. En seguida se

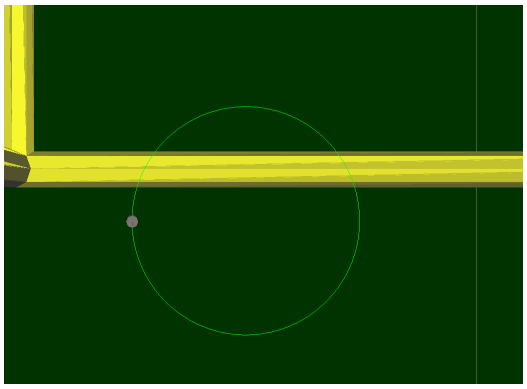


Figura 9.24: Simulación de círculo de diámetro igual a 5cm en visor de Gcode, *KmotionCNC*.

muestran los pasos a seguir para su correcta instalación.

Nos dirigimos a la página de **MathWorks (www.mathworks.com)** donde pedirá nuestro correo, (Figura 9.25), una vez ingresado nuestro correo de forma correcta nos dirigimos a la barra de búsqueda donde colocaremos, **Simple Robotics Toolbox** y descargamos el archivo .Zip (Figura 9.26).

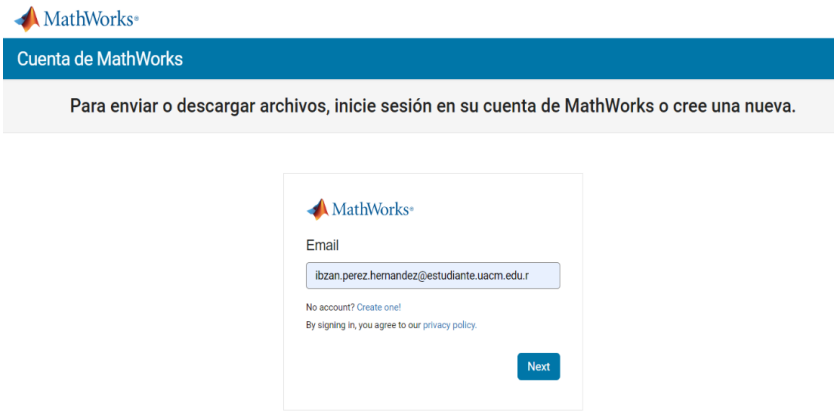


Figura 9.25: Ingresar correo institucional.

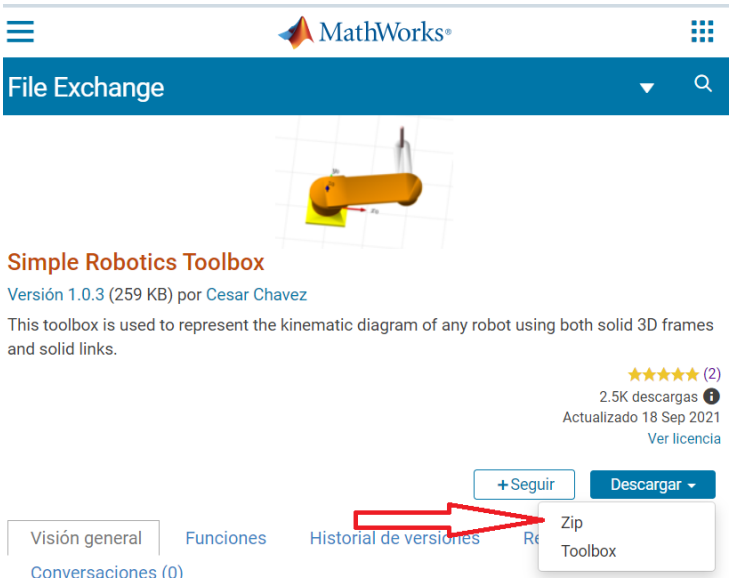


Figura 9.26: Descarga de archivo .Zip.

Después abrimos *Matlab* y seleccionamos la pestaña *HOME*, luego la opción de *Set Path*, (Figura 9.27). Una vez descomprimido nuestro archivo. Zip, procedemos a agregar todas nuestras carpetas y por último guardamos los cambios, (Figura 9.27), realizados estos pasos ya podemos hacer uso del *toolbox*.

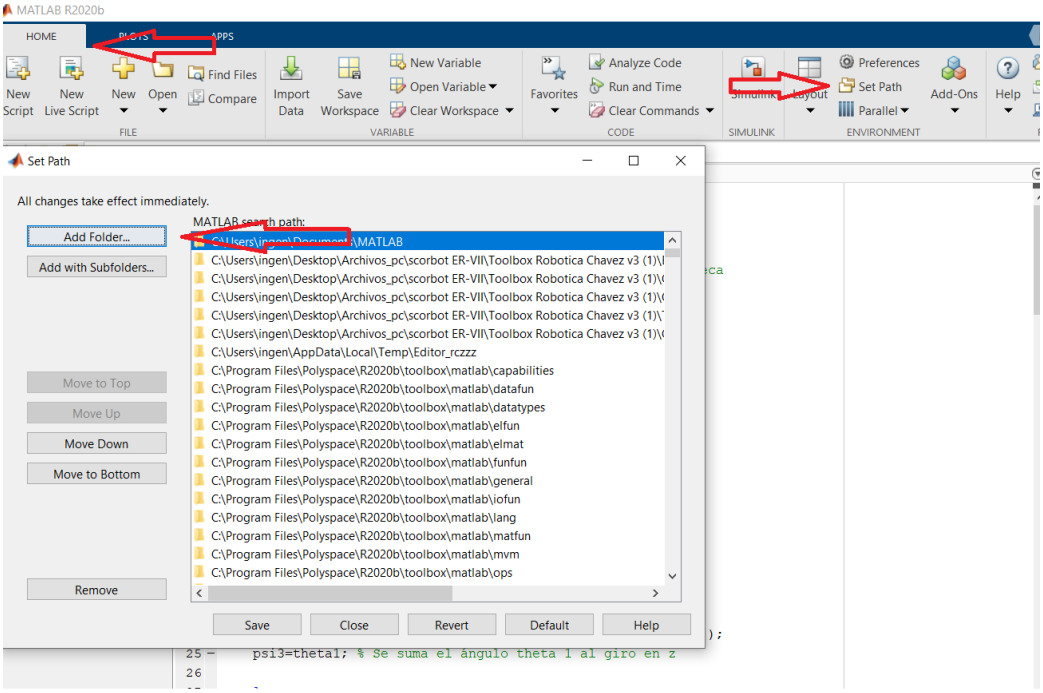


Figura 9.27: Agregar Carpetas para instalación del *toolbox*.

Bibliografía

[Arias and Fonseca, 2012] Arias, J. L. R. and Fonseca, A. R. (2012). Modelamiento matemático de la cinemática directa e inversa de un robot manipulador de tres grados de libertad. *Ingeniería solidaria*, 8(15):46–52.

[Baturone, 2005] Baturone, A. O. (2005). *Robótica: manipuladores y robots móviles*. Marcombo.

[Capek, 2004] Capek, K. (2004). *RUR (Rossum's universal robots)*. Penguin.

[Chavez, 2023] Chavez, C. (2023). *Simple Robotics Toolbox*. MATLAB Central File Exchange.

[Cortés, 2020] Cortés, F. R. (2020). *Robótica: control de robots manipuladores*. Marcombo.

[Duque, 2017] Duque, P. (2017). Guía técnica para el diseño y cálculo de engranajes para reductores de velocidad. *Memoria de titulación para pregrado de Ingeniería Mecánica, Departamento de Ingeniería Mecánica, Universidad Técnica Federico Santa María, Valparaíso, Chile*.

[Dynomotion, 2023] Dynomotion (2023). Dynomotion user manual.

[Dynomotion, 2023] Dynomotion (2023). Using SnapAmp.

[Foro-Wiki, 2023] Foro-Wiki (2023). Channels channels channels.

[Garay Molina et al., 2001] Garay Molina, O. A. et al. (2001). Control de posición de un robot industrial sobre superficies arbitrarias bajo el enfoque de desacoplamiento cinemático. *REPOSITORIO NACIONAL CONACYT*.

[García, 2006] García, J. C. C. (2006). Tecnología avanzada del diseño y manufactura asistidos por computador-cad/cam. *Prospectiva*, 4(1):75–81.

[Grossman, 2008] Grossman, S. I. (2008). *Algebra lineal, Sexta Edicion*. McGraw Hill Educación.

[Kumar Saha, 2010] Kumar Saha, S. (2010). *Introducción a la robótica*. McGraw Hill Educación.

[MOTOMAN, 1998] MOTOMAN (1998). *SK16 Dual MRC Manipulator Manual*. MOTOMAN.

[Naylampmechatronics, 2022] Naylampmechatronics (2022). Motor dc jga25-370 - 12v/350rpm con encoder.

[Preisz et al., 2009] Preisz, E. N., Weisz, R. M., Bauer, J. M., and Gentiletti, G. G. (2009). Educación en robótica: modelos cinemáticos del scorbot er-ix. *Revista Argentina de Enseñanza de la Ingeniería*, 10:51–63.

[ROBOTEC, 1998] ROBOTEC, E. (1998). *SCORBOT ER-VII Manual de Usuario*.

[Rodd, 1987] Rodd, M. G. (1987). *Introduction to robotics: Mechanics and control: John J. Craig*. Pergamon.

[Rodriguez, 2007] Rodriguez, J. (2007). *Fundamentos De Robotica Antonio Barrientos*. Biblioteca Hernán Malo González.

[ROSALENY, 2021] ROSALENY, R. T. (2021). Encoders ópticos. *Descargado de http://www. infoplc. net/files/documentacion/instrumentacion_deteccion/infoPLC_net_ENCODERS_OPTICOS. pdf*.

[Sandin, 2003] Sandin, P. (2003). *Robot mechanisms and mechanical devices illustrated*. McGraw Hill Professional.

[TECHNOLOGIES, 2023] TECHNOLOGIES, A. (2023). Quick assembly two and three channel optical encoders.

[Vásquez Rojas, 2020] Vásquez Rojas, P. M. (2020). Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control pid.

[Vila-Rosado and Domínguez-López,] Vila-Rosado, D.-E. N. and Domínguez-López, A. El arte de diseñar robots manipuladores.

[Villarreal-Cervantes, 2014] Villarreal-Cervantes, M. G. (2014). Control pid robusto de un robot manipulador con base en un problema de optimización dinámica. pages 1428–1443.

[YASKAWA, 2002] YASKAWA (2002). ∑ Series User´s Manual AC Servomotors and Drivers SGM/SGMP Servomotors.

[Álvarez, 2020] Álvarez, M. A. C. (2020). *Manual de programación básica para MOTOMAN YASNAC MRC SK16 y SK120*.