



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Análisis cinemático e
implementación de brazo
robot educativo**

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Omar Servin Comunidad

DIRECTORA DE TESIS

M.I. Gloria Mata Hernández



Ciudad Universitaria, Cd. Mx., 2024

Índice

Resumen	4
Abstract	5
Objetivo general	6
Objetivos específicos.....	6
Justificación.....	6
Alcance	7
Presentación	8
Capítulo 1 Cinemática de robots manipuladores.....	10
Definición de robot	10
Clasificación de los robots manipuladores.....	11
¿Qué es la cinemática?.....	13
Cinemática de los robots.....	14
Cinemática directa	15
Cinemática inversa	19
Capítulo 2 Descripción y Caracterización del brazo robot.....	22
Características físicas del robot.....	23
Características mecánicas	27
Características eléctricas y Servomotores	32
Plataforma LabVIEW	39
Capítulo 3 Análisis de la cinemática directa e inversa.....	46
Implementación de algoritmo Denavit Hartenberg para robot bajo estudio.....	46
Obtención de la cinemática inversa para el robot bajo estudio	59
Ángulos de Euler.....	76
Capítulo 4 Desarrollo de algoritmos	84
Programa para control de un servomotor mediante ancho de pulso	85
Programa de interpolación	87
Máquina de estados.....	89
Programa para crear una rutina secuencial	91
Programa para cambio de estado	100
Programa para ejecutar rutina secuencial	102
Programa para la implementación de la cinemática directa	106
Programa para la implementación de la cinemática inversa	110

Programa para convertir ángulos de Euler a matriz de orientación	115
Programa para obtener las variables articulares q_4 y q_5	117
Programa para seleccionar las ternas de variables articulares q_1, q_2, q_3	120
Capítulo 5 Pruebas y resultados	124
Repetitividad con rutinas secuenciales.....	124
Simulación de la cinemática directa y modelo 3D del robot manipulador.....	128
Implementación de rutina secuencial para recoger y dejar pelotas.....	139
Validación de la cinemática inversa	144
Implementación de rutina para dibujar letras y figuras	153
Conclusión.....	165
Trabajos citados	166
Anexos	168
Programas Tiva	168
Programa de simulación del robot LeArm en Matlab.....	173
Algoritmo para obtener ternas de las variables articulares q_1, q_2, q_3 en la cinemática directa implementada para el robot LeArm de 5GDL	175
Índice de figuras	191

Resumen

En el siglo XXI, los robots han adquirido una creciente importancia en diversos sectores, como el industrial, educativo, de investigación, minería, medicina y otros ámbitos. Con el continuo avance en la automatización, los robots manipuladores han experimentado un aumento tanto en su cantidad como en su demanda. En este contexto, resulta imperativo contar con profesionales y técnicos capacitados para programar, construir, diseñar y operar estos dispositivos.

El análisis matemático de un robot manipulador constituye una etapa crucial previa a su manipulación. Este análisis permite anticipar el comportamiento que exhibirá durante su operación y, en caso de ser necesario, realizar correcciones de acuerdo con las tareas asignadas.

El estudio de la cinemática de los robots manipuladores facilita la comprensión de la relación entre su estructura física, compuesta por eslabones, articulaciones y herramienta final, y el espacio de trabajo donde se llevan a cabo las operaciones. De esta manera, es posible diseñar trayectorias, manipular la herramienta final del robot y operarla eficientemente.

Existen dos enfoques principales para abordar los desafíos relacionados con el análisis cinemático de los robots manipuladores: la cinemática directa y la cinemática inversa.

La cinemática directa busca obtener los datos de posición y orientación de la herramienta o extremo final del manipulador a partir de las coordenadas articulares del propio robot. Estas coordenadas quedan definidas en relación con un sistema de referencia fijo.

Por otro lado, la cinemática inversa se centra en determinar las coordenadas articulares que posicionan y orientan la herramienta o extremo final del manipulador en el lugar de interés.

En este proyecto, se aborda el análisis de un robot manipulador de 5 grados de libertad con una pinza como herramienta final. Se resuelven los problemas cinemáticos directo e inverso para este robot en particular, detallando sus características mecánicas, eléctricas y otros componentes, como servomotores, articulaciones y herramientas de trabajo. Además, se presentan los algoritmos utilizados para la manipulación, junto con las pruebas de desempeño y los resultados obtenidos al alcanzar puntos específicos en el espacio para el manejo de piezas ligeras en tareas de recoger, soltar y escribir.

PALABRAS CLAVE: brazo robot, cinemática, cinemática directa, cinemática inversa.

Abstract

In the 21st century, robots have assumed an increasingly pivotal role across various sectors, including industrial, educational, research, mining, and medicine. The automation sector's progress has led to a surge in both the quantity and demand for manipulator robots. Consequently, there is a growing need for professionals and technicians proficient in programming, constructing, designing, and operating these robots.

The mathematical analysis of a robot manipulator stands as a crucial precursor to its manipulation. By scrutinizing a manipulator robot, one can anticipate its behavior during operation and, when necessary, make corrections based on the assigned tasks.

The analysis of robot manipulators' kinematics enables a comprehensive exploration of the relationship between their physical structure (comprising links, joints, and tools) and the workspace where manipulation occurs for task execution. This allows for the creation of trajectories and the manipulation of the robot's working tool for tasks such as object handling.

Two distinct approaches address the challenges encountered in kinematic analysis for robot manipulators: forward kinematics and inverse kinematics. Forward kinematics aim to derive the position and orientation data of the tool or end of the manipulator based on the joint coordinates of the robot. These coordinates are defined within a fixed reference system. Conversely, inverse kinematics involves determining the joint coordinates that position and orient the tool or end of the manipulator in a specific manner.

This study focuses on the analysis of a manipulator robot with 5 degrees of freedom, featuring a clamp as its final tool. The forward and inverse kinematic problems for the examined robot are solved, accompanied by a detailed description of its mechanical and electrical characteristics, including components such as servomotors, joints, and work tools. Additionally, the presented algorithms for manipulation, performance tests, and results showcase the robot's ability to reach specific points in space for tasks like picking up, dropping off, and writing with precision.

KEY WORDS: robot arm, kinematics, forward kinematics, inverse kinematics.

Objetivo general

Desarrollar e implementar el modelo de cinemática directa e inversa de un brazo robot educativo con 5 GDL para determinar la posición y orientación que adopta el actuador final con respecto a su sistema de coordenadas de referencia, permitiendo que ejecute acciones de manipulación, maniobrabilidad, interacción y recolección de objetos.

Objetivos específicos

- Desarrollar la cinemática directa para el robot LeArm con 5 grados de libertad.
- Desarrollar la cinemática inversa para el robot LeArm con 5 grados de libertad.
- Desarrollar algoritmos de la cinemática directa e inversa que puedan ser programados.
- Con los algoritmos planteados, crear interfaces gráficas para que el usuario interactúe con el robot físico y observe directamente los resultados obtenidos por la cinemática inversa y directa.
- Validar los resultados obtenidos de la cinemática directa e inversa.
- Implementar rutinas y/o tareas que hagan uso de la cinemática directa e inversa.

Justificación

Los temas relacionados con la robótica son temas relevantes en muchas investigaciones, debido a que han tenido gran impacto en su avance actual. Aun cuando se han venido desarrollando desde hace décadas, la robótica encuentra en la actualidad otras herramientas como la inteligencia artificial que permiten seguir perfeccionándose. Además, los campos de aplicación como la automatización incentivan la investigación, su desarrollo y su implementación. En la licenciatura de ingeniería Eléctrica Electrónica se imparte la asignatura de Instrumentación Virtual que proporciona al estudiante un enfoque práctico y experimental donde es posible aplicar de manera conjunta los conocimientos adquiridos durante la carrera.

En la variedad de aplicaciones a desarrollar en la asignatura, se tiene la manipulación de robots móviles, entre los que se encuentran el Mindstrom Lego, TETRIX Lego y DANI2.0 NI.

Con la incorporación del robot LeArm, un brazo robot manipulador de cinco grados de libertad, creado para fines educativos, se pueden implementar los algoritmos como parte de la formación práctica de los estudiantes. La finalidad de esto es que se conozcan algunos aspectos fundamentales del diseño mecánico del robot, los algoritmos desarrollados como resultado de un análisis de la cinemática del robot, y los programas desarrollados a partir de los algoritmos para poder maniobrar y manipular al robot.

El robot LeArm por su tamaño y características, hace factible su ejecución en entornos pequeños como laboratorios y aulas, haciendo posible llevar a cabo las tareas planeadas para este manipulador.

Como parte de actividades formativas en la ingeniería aplicada se plantea desarrollar diversas rutinas que permitan el conocimiento, manejo de los diferentes elementos y aspectos involucrados en el brazo robot, encontrándose entre estos la estructura mecánica, los actuadores eléctricos, la ejecución de rutinas secuenciales, de las dimensiones tridimensionales, así como la manipulación de la posición y orientación en su entorno espacial.

De esta manera, a partir de los resultados obtenidos se proponen actividades que se puedan llevar a cabo en la asignatura de Instrumentación Virtual por representar un buen complemento ligado a los temas relacionados con el área de instrumentación, control y robótica, que expanden los conocimientos y las habilidades de los alumnos en dicha área.

Alcance

- Implementación del robot LeArm de cinco grados de libertad como etapa inicial para el desarrollo de material para prácticas de laboratorio en las asignaturas de la carrera de Ingeniería Eléctrica-Electrónica de la Facultad de Ingeniería de la UNAM.
- Desarrollo preliminar de prácticas para la materia de “Instrumentación Virtual” del módulo “control y robótica” de la carrera de Ingeniería Eléctrica-Electrónica de la Facultad de Ingeniería de la UNAM.

Presentación

La presente tesis está organizada en 5 capítulos y una sección de anexos. El **Capítulo 1 Cinemática de robots manipuladores**, presenta los conceptos básicos de cinemática y su aplicación a los robots manipuladores. Se muestra la teoría de los dos principales problemas cinemáticos: la cinemática directa e inversa. Adicionalmente se repasa brevemente la historia de los robots manipuladores, así como algunas configuraciones mecánicas típicas de estas máquinas.

En el **Capítulo 2 Descripción y Caracterización del brazo robot** se presentan los diversos sistemas y subsistemas que hacen posible el funcionamiento del robot manipulador bajo estudio. Se presentan las características eléctricas de los servomotores, las diversas piezas mecánicas que conforman el cuerpo del robot, la parte de comunicación entre los servomotores y TIVA. Se muestra el entorno de programación gráfico "LabVIEW" y algunas de sus herramientas que hacen posible el desarrollo de las aplicaciones y experimentos que se exhiben en capítulos posteriores. Se menciona el concepto de área de trabajo y se da a conocer el propio para el robot LeArm.

En el **Capítulo 3 Análisis de la cinemática directa e inversa** se desarrollan matemáticamente la cinemática directa e inversa y se obtienen las soluciones de acuerdo con el análisis hecho al robot LeArm. Se hace, también, el desarrollo del cálculo de los ángulos de Euler y el cálculo de la matriz de orientación a partir de los ángulos de Euler. La matriz de orientación y ángulos de Euler son parámetro importantes que permiten hacer los cálculos de cinemática inversa cuando se requiera una representación adicional de la rotación.

En el **Capítulo 4 Desarrollo de algoritmos** se muestran los diversos algoritmos desarrollados en el entorno de programación gráfica. Se presentan algunos pequeños subprogramas que son base para la estructura de los programas más complejos, como son los que resuelven los problemas cinemáticos directo e inverso. Se introduce el concepto de máquinas de estados que permite desarrollar los algoritmos y programas que integran varios procesos y que dependen de un usuario que interactúa con estos.

Para el **Capítulo 5 Pruebas y resultados** se dan a conocer los resultados y experimentos llevados a cabo mediante los algoritmos programados en el capítulo 4. Se hacen comparaciones entre pruebas simuladas y físicas, así como los experimentos que muestran de forma práctica la aplicación de la cinemática directa e inversa.

Se crea una sección extra llamada **Anexos**, que agrega información adicional del algoritmo para controlar los servomotores mediante la tarjeta TIVA. Se incorpora, también, la programación de la simulación del robot bajo estudio en la plataforma Matlab. Se muestra la forma en que se forman las ternas para los tres primeros grados de libertad y de a partir de esto se crea el algoritmo y su respectiva

programación que es usada en el algoritmo programado para resolver la cinemática inversa.

Capítulo 1 Cinemática de robots manipuladores

Definición de robot

La palabra “robot” fue usada por primera vez en una obra de teatro escrita por el checo Karel Čapek en el año 1921. (Saha, 2010)

El escritor de ciencia ficción Asimov inventó el concepto y la palabra “robótica” para describir una disciplina ficticia encargada del estudio de los robots, años antes de que estas máquinas fueran una realidad.

Los robots forman parte de la sociedad actual del siglo XXI, hay algunos que caminan, otros hacen gestos humanos y simulan el habla, existen los diseñados para explorar planetas, etc. Además, los robots son usados en infinidad de actividades como la minería, la industria automotriz, medicina, industria nuclear, educación y demás.

Sin embargo, los primeros robots no eran nombrados de esta manera. Por ejemplo, algunas máquinas que se usaban en los años cuarenta para manejar elementos radiactivos eran llamados manipuladores teleoperados. En un principio, estos manipuladores teleoperados eran accionados mecánicamente y hasta los años cincuenta se implementó un accionamiento eléctrico.

En la misma década de los cincuenta, un inventor estadounidense, George C. Devol patentó un brazo robot al que denominó “programmed article handling device” cuyas tareas programadas se almacenaban en memorias de tambor. Esta máquina sería conocida más tarde como “unimate”, obsérvese en la figura 1. Devol, junto con su compañero Joseph Engelberger fundaron una empresa de robótica, “Unimation”, que vendió sus primeras creaciones a la empresa automotriz “General Motors”. En dicha empresa, el “unimate” fue utilizado en tareas de fundición a presión. A partir de entonces el término robot industrial fue acuñado y popularizado. (Invent, 2023) (Langer, 2023)

Los sindicatos estadounidenses vieron una amenaza para los empleos a los robots de Devol quien junto a su compañero Engelberger, firmaron acuerdos con la compañía japonesa “Kawasaki Heavy Industries”. Fue en Japón donde tuvo más aceptación, un posterior éxito y auge del desarrollo de robots industriales japoneses, el cual superó a la industria estadounidense. Incluso en 1972 se fundó la “Asociación de Robótica Industrial de Japón” (JIRA), dos años antes que el “Instituto de Robótica de América” (RIA). (Langer, 2023) (Barrientos, Peñín, Belaguer, & Aracil, 2007)

Las organizaciones de robótica alrededor del mundo han establecido su definición de robot industrial manipulador.

Entidades como la Organización Internacional Para la Estandarización, ISO, o la Federación internacional de Robótica, RIA, definen a los robots industriales como manipuladores multifuncionales reprogramables, formados por elementos en serie

articulados entre sí, con la capacidad de mover, posicionar y orientar materiales, piezas o herramientas, según trayectorias. De acuerdo la definición de la ISO en la norma ISO-8373-1998, un robot manipulador industrial debe tener al menos 3 o más ejes. (Barrientos, Peñín, Belaguer, & Aracil, 2007)

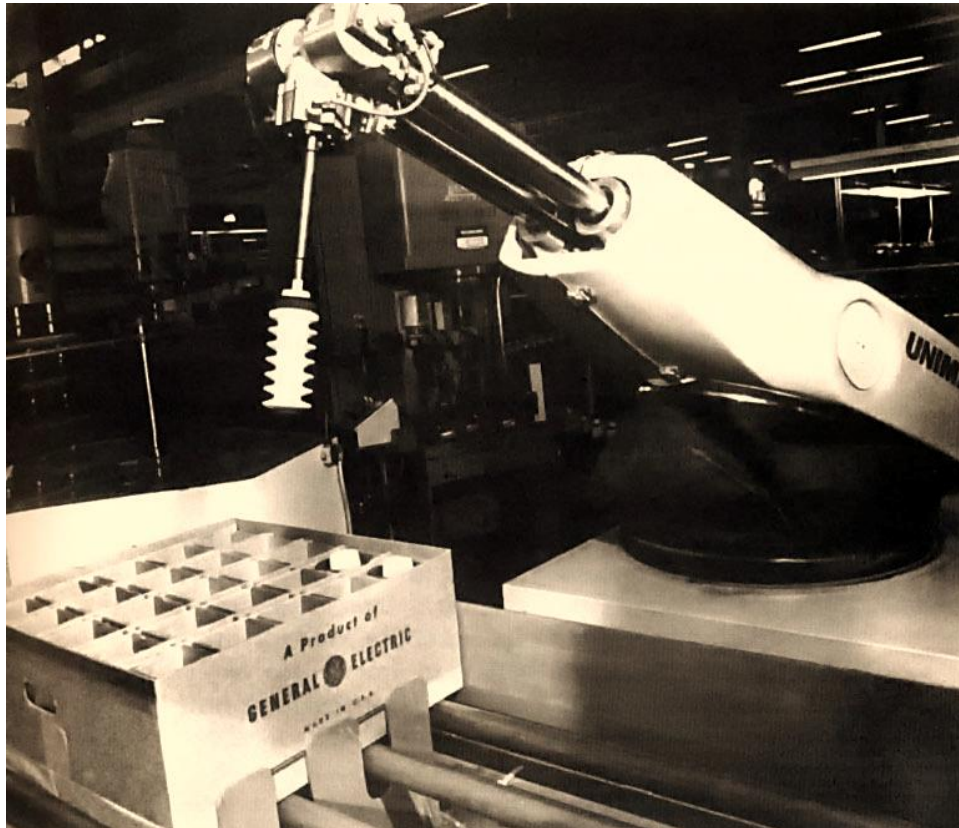


Figura 1 Robot Unimate creado por la empresa UNIMATION

Clasificación de los robots manipuladores

Los robots industriales pueden clasificarse por su área de aplicación, por su sistema de potencia, ya sea eléctrico, neumático o hidráulico, por su forma de control, por su método de programación, por la generación de robots a la que pertenecen, por su tipo de actuadores, etc. (Saha, 2010) Un sistema de clasificación muy importante, es el que se hace de acuerdo con sus sistemas de coordenadas.

Con la clasificación por coordenadas es posible saber el volumen de trabajo en el espacio tridimensional, que puede ser alcanzado el efector final del robot. También es nombrada clasificación por configuración de brazo o por volumen geométrico de trabajo. (Saha, 2010)

Por configuración de brazo se entiende a la combinación de tipos de articulaciones que componen al robot. Existen dos articulaciones básicas, la prismática y la rotacional.

La articulación rotacional es en la que se tienen dos eslabones que giran uno respecto al eje del otro. Mientras que las articulaciones prismáticas son formadas por un par de eslabones en las que uno de los eslabones se desliza a lo largo del eje del otro. Obsérvese los dos tipos de movimiento de las articulaciones en la figura 2. (Reyes Cortés, Control de robots manipuladores, 2011)

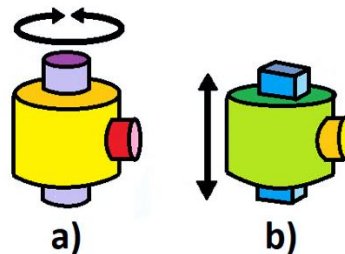


Figura 2 Tipos articulaciones. a)rotacional. b)prismática.

- El robot cartesiano se conforma de tres articulaciones prismáticas y su espacio de trabajo es un prisma rectangular.
- El robot cilíndrico está compuesto por una articulación rotacional y dos prismáticas. Su espacio de trabajo es un cilindro.
- El robot esférico tiene por configuración dos articulaciones rotacionales y una prismática. Su espacio de trabajo es una esfera.
- El robot escara tiene articulaciones rotacionales y una prismática. Se diferencia del robot esférico por la orientación que toman sus articulaciones. Su espacio de trabajo es un cilindro.
- El robot angular o antropomórfico se constituye por tres articulaciones rotacionales. Su espacio de trabajo es una esfera. (Reyes Cortés, Control de robots manipuladores, 2011)

Las anteriores configuraciones se visualizan en la figura 3.

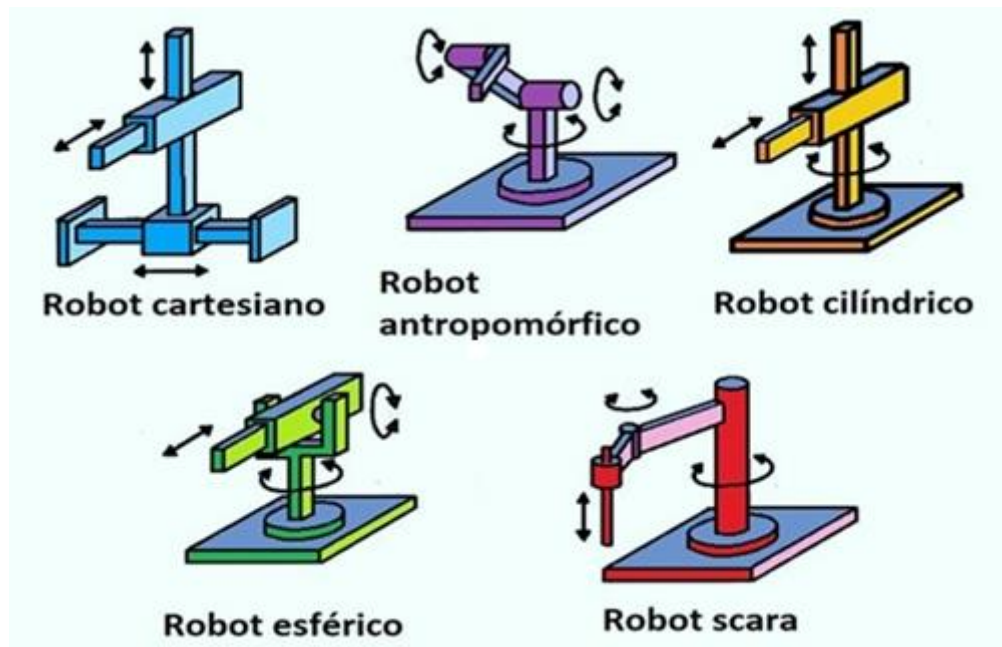


Figura 3 Robots manipuladores de acuerdo a su configuración de brazo y espacio de trabajo.

¿Qué es la cinemática?

En el campo de la ciencia física se encuentra como una rama importante, la mecánica clásica. La mecánica clásica se ocupa estudiar de los cuerpos en reposo o movimiento sometidos por fuerzas. A su vez, la mecánica se encuentra ramificada en otras áreas como la estática, la dinámica, la cinemática, la cinética, etcétera. (Hibbeler, 2010)

La cinemática analiza el movimiento de los cuerpos rígidos (los objetos) y de su trayectoria en función del tiempo, sin tener en consideración las fuerzas que originan el movimiento. Algunos conceptos estudiados en la cinemática son la posición, la velocidad y la aceleración. (Beer, Johnston, & Cornwell, 2010)

La posición permite ubicar a un objeto en un plano o un espacio tridimensional con respecto a un marco de referencia que puede ser un punto de origen u otros objetos. La velocidad permite conocer la variación de la posición de un objeto en función del tiempo. Así mismo, la aceleración se entiende como el cambio de velocidad en función del tiempo de un objeto.

El concepto de posición, por su relación, da lugar también al concepto de trayectoria. La trayectoria se puede entender si vemos a un conjunto de posiciones como un conjunto de puntos en el plano o el espacio tridimensional en el que esté nuestro objeto de análisis. Al unir esos puntos en lo que ha estado el objeto, se forma una trayectoria que marca el movimiento que ha tenido el objeto a lo largo del tiempo. Puede tratarse también una trayectoria deseada, la cual se requiere sea realizada por el objeto.

Cinemática de los robots

Al revisar los conceptos manejados en la cinemática, se observa que son aplicables a los robots manipuladores. Esto es comprensible, porque algunos utilizan herramientas como efectores finales para efectuar tareas de soldar, pintar, imprimir, etcétera. Y para ejecutar correctamente las tareas es fundamental llevar a cabo el posicionamiento y orientación correctos de los efectores finales.

Se tiene el caso de los manipuladores equipados con una herramienta de corte láser donde el posicionamiento de esta es clave para que los cortes efectuados sean precisos. Aunque, también la dinámica y la teoría de control en conjunto hacen posible tal precisión, la cinemática es el punto de partida para comprender el movimiento de un brazo robot.

La aplicación de la cinemática directa permite diseñar trayectorias porque se requiere que este realice tareas que conllevan una rutina secuencial moviéndose dentro de su espacio de trabajo.

Se tiene el ejemplo de un robot que recoge tornillos y lo atornilla en determinada posición. Para cumplir la tarea debe pasar por el lugar donde se encuentran los tornillos, coge uno y de haberlos, sortea obstáculos para no chocar con algún objeto al seguir su trayectoria. Debe llegar al sitio final donde atornilla al girar su herramienta de trabajo.

Los principales problemas de la cinemática a resolver para los robots manipuladores son la cinemática directa y la cinemática inversa. Se pueden estudiar individualmente o complementarse y tener un estudio más completo.

La finalidad de la cinemática directa es relacionar la posición final del extremo del manipulador, o en su caso de la herramienta, en función de las variables articulares, respecto a un punto fijo de referencia. Las variables articulares son parámetros fijos o variables que son información de la estructura física del robot (longitud de eslabones) o se presentan como parámetros que existen dada una relación angular entre sus eslabones. (Reyes Cortés, MATLAB aplicado a Robótica y mecatrónica, 2012)

En cambio, la cinemática inversa tiene como objetivo conocer la o las combinaciones de coordenadas articulares (ángulos de las articulaciones del robot) que su conjunto da como resultado una posición y orientación determinados. Es decir, la posición y orientación son datos conocidos y a partir de estos se determinan los valores angulares de las articulaciones. A diferencia de la cinemática directa, en la cinemática inversa puede haber más de una solución. (Reyes Cortés, MATLAB aplicado a Robótica y mecatrónica, 2012)

Cinemática directa

La cinemática directa determina la posición y orientación del extremo o efector final de robot a partir de los valores geométricos y de las variables articulares del robot, esto con respecto a un sistema de coordenadas de referencia.

Existen varios métodos que permiten dar solución a la cinemática directa entre los que se encuentran el método geométrico, el método de Denavit-Hartenberg y uso de cuaternios. (Barrientos, Peñín, Belaguer, & Aracil, 2007) Para efectos de este documento solo se abordan los dos primeros métodos mencionados

En el caso del método geométrico, se usan las relaciones geométricas y de las variables articulares para obtener las expresiones que relacionan el sistema de ejes en la base del robot con el sistema de ejes en el extremo del robot. Este método tiene algunos inconvenientes. Se vuelve sencillo de implementar para robots con pocos grados de libertad y no es un método sistemático que permita una solución rápida y sencilla.

En el caso del método de Denavit-Hartenberg se puede sistematizar su uso a casi cualquier configuración de robot manipulador. Por sistematizar se entiende que usa un método que guía paso a paso. Esto funciona porque los robots manipuladores son cadenas cinemáticas. Una cadena cinemática es un conjunto de eslabones unidos por articulaciones. (Reyes Cortés, Control de robots manipuladores, 2011)

Para un robot manipulador que cuenta con ' n ' grados de libertad, su cadena cinemática estará formada por ' n ' eslabones que se unen por n articulaciones. (Reyes Cortés, Control de robots manipuladores, 2011)

La configuración formada por articulación-eslabones, forma un grado de libertad como puede observarse en la figura 4.

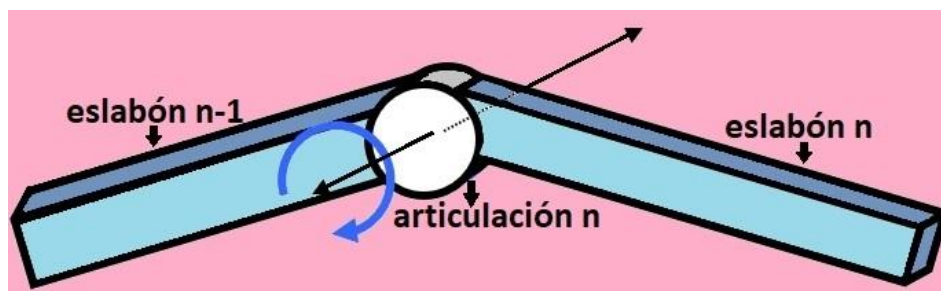


Figura 4 Grado de libertad.

El método de Denavit-Hartenberg hace uso de matrices y vectores para poder describir la localización de los elementos de un robot manipulador en el espacio tridimensional.

Se le asigna a cada eslabón de la cadena cinemática, un sistema de referencia O_i y mediante el método de Denavit-Hartenberg se obtienen unas matrices que

describen las transformaciones en el espacio. Es decir las rotaciones y translaciones necesarias para pasar de un sistema de referencia a otro.

Por cada eslabón se obtiene una matriz de transformación. Estos sistemas de referencia están asociados al eslabón analizado y su adyacente próximo.

Al ser un robot una cadena cinemática, se puede usar las propiedades de las matrices para multiplicar cada una de las obtenidas para cada eslabón y obtener una sola matriz de transformación homogénea que represente la posición y orientación del sistema de referencia en el extremo del robot con respecto a un sistema de referencia fijo que se encuentra en la base del mismo.

Una matriz transformación que representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se expresa como en la ecuación (1).

$${}^{i-1}_i H \quad (1)$$

De la ecuación (1), 'i' hace referencia al eslabón analizado e 'i-1' alude al eslabón adyacente o próximo anterior en la cadena cinemática.

De esta manera, la matriz ${}^0_1 H$ describe la posición y orientación del sistema de referencia del primer eslabón '1' con respecto al sistema de referencia base '0', ${}^1_2 H$ describe la posición y orientación del eslabón '2' con respecto del eslabón '1'.

Teniendo en cuenta lo anterior se deduce que el eslabón '2' del robot con respecto al sistema de coordenadas de la base '0' se expresa mediante la matriz, ${}^0_2 H$, donde el producto de matrices de transformación que la componen se muestra en la ecuación (2).

$${}^0_2 H = {}^0_1 H * {}^1_2 H \quad (2)$$

Considerando a todos los eslabones, se obtiene una matriz ${}^0_n H$. Esta matriz de transformación homogénea total se expresa como se dicta en la ecuación (3).

$$T = {}^0_n H \quad (3)$$

La anterior ecuación es el resultado de la multiplicación de las matrices de transformación homogénea para cada uno de los eslabones, como se muestra en la ecuación 4.

$$T = {}^0_1H * {}^1_2H * {}^2_3H * \dots * {}^{i-1}_iH \quad (4)$$

Se considera que cada matriz de transformación homogénea asociada a un eslabón está compuesta por cuatro transformaciones básicas que dependen exclusivamente de los parámetros geométricos de cada eslabón. Estas transformaciones son rotaciones y traslaciones que relacionan el sistema de referencia del eslabón 'i-1' con el sistema del eslabón 'i'. Esto se expresa matemáticamente en la ecuación (5). (Barrientos, Peñín, Belaguer, & Aracil, 2007)

$${}^{i-1}_iH = \mathbf{ROTZ}(\theta_i) \mathbf{\Gamma}(0,0,d_i) \mathbf{\Gamma}(a_i,0,0) \mathbf{ROTX}(\alpha_i) \quad (5)$$

Donde:

- **ROTZ**(θ_i) Es una matriz que representa Rotación de un ángulo θ_i alrededor del eje Z_{i-1} . Donde Z_{i-1} está asociado al sistema de ejes del eslabón 'i-1'.
- **$\Gamma(0,0,d_i)$** Es una matriz que representa una traslación de una distancia d_i a lo largo del eje Z_{i-1} . Donde Z_{i-1} está asociado al sistema de ejes del eslabón "i-1".
- **$\Gamma(a_i,0,0)$** Es una matriz que representa una traslación de una distancia a_i a lo largo del eje X_i . Donde X_i está asociado al sistema de ejes del eslabón 'i',
- **ROTX**(α_i) Es una matriz que representa la rotación de un ángulo α_i alrededor del eje X_i . Donde X_i está asociado al sistema de ejes del eslabón 'i'. (Barrientos, Peñín, Belaguer, & Aracil, 2007)

En la ecuación 6 se muestra cada matriz elemental de la ecuación 5, desarrollada con todos sus términos y respectivos parámetros.

$${}^{i-1}_iH = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dots \dots \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Al multiplicar las matrices la ecuación (6) se reduce a una sola matriz como se expresa en la ecuación (7).

$${}^{i-1}H = \begin{bmatrix} \cos \theta_i & -\cos \alpha_i * \sin \theta_i & \sin \alpha_i * \sin \theta_i & a_i * \cos \theta_i \\ \sin \theta_i & \cos \alpha_i * \cos \theta_i & -\sin \alpha_i * \cos \theta_i & a_i * \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

En la ecuación (7) se observa que es una generalización de una matriz de transformación homogénea donde solo queda obtener los valores de los parámetros articulares θ_i , d_i , a_i y α_i . (Barrientos, Peñín, Belaguer, & Aracil, 2007)

Al tener una matriz de transformación homogénea generalizada ya no es necesario hacer todas las operaciones matemáticas para tener cada matriz individual. Solo se necesita conocer los parámetros articulares asociados a cada eslabón de la cadena cinemática.

Es aquí donde el método de Denavit-Hartenberg propone un algoritmo con el que se obtienen los parámetros. Este algoritmo se presenta con detalle en el capítulo 3.

Una vez obtenida cada matriz de transformación homogénea, se obtiene la matriz de transformación homogénea total " T " multiplicando cada matriz asociada a cada eslabón que compone al robot manipulador.

De la ecuación (7) que expresa a la matriz de transformación homogénea generalizada, se representa como una matriz de tamaño 4X4 (cuatro por cuatro) lo que significa que la componen 16 elementos. Esta ecuación se describe como en la ecuación (8).

$$T = {}^0H * {}^1H * {}^2H * \dots * {}^{i-1}H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

La matriz T define la orientación (submatriz de rotación) y la posición (submatriz de traslación) del extremo referido a la base, en función de las variables articulares α_i , a_i , d_i , θ_i .

Los elementos 'nx', 'ny', 'nz', 'ox', 'oy', 'oz', 'ax', 'ay' y 'az' definen la orientación del sistema de ejes asociada a la herramienta del robot respecto al eje de referencia en la base del robot manipulador. Mientras que los elementos 'px', 'py', 'pz', representan la posición del sistema de ejes asociada a la herramienta del robot respecto al eje de referencia en la base del manipulador.

Al obtener el valor de cada elemento de la matriz de la ecuación 8 queda resuelto el problema cinemático directo.

Cinemática inversa

La cinemática inversa es un análisis de la cinemática de los robots manipuladores que determina, a partir de una posición y orientación del efector final del robot, la posición angular que deben adoptar cada una de las articulaciones del manipulador.

La cinemática inversa, al contrario de la cinemática directa, no puede obtenerse mediante algún algoritmo ya establecido. Existen procedimientos que permiten la resolución del problema, pero que se vuelven particulares para cada robot y a diferencia de la cinemática directa, se pueden obtener múltiples soluciones o ninguna solución. En la figura 5 se muestra un robot manipulador, considerando dos eslabones 'l₂' y 'l₃' que, en conjunto, alcanzan la posición 'p'. Se observa que en ambas configuraciones se logra la misma posición, pero con distintos valores de las variables articulares 'θ₂₁', 'θ₃₁', 'θ₂₂' y 'θ₃₂'. Esto demuestra que mediante la cinemática inversa se pueden encontrar varias soluciones.

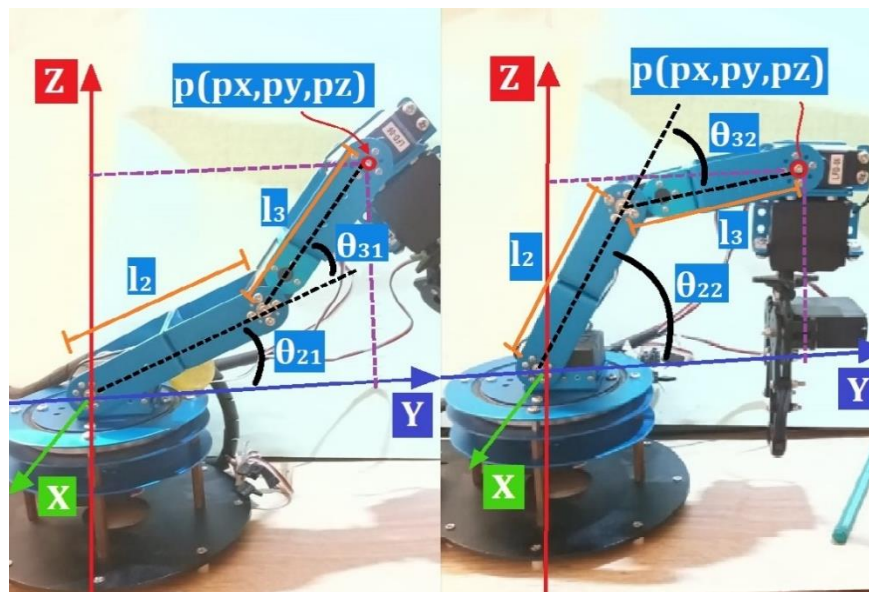


Figura 5 Mismo punto en el espacio, pero diferente configuración de robot manipulador.

Al igual que en la cinemática directa, en la cinemática inversa se utilizan métodos geométricos que son sencillos de implementar para robots manipuladores con pocos grados de libertad. También se hace uso de las matrices de transformación homogénea adquiridas en la cinemática directa para obtener los valores de las "n" variables articulares, despejándolas de las expresiones de tales matrices. Esto es complejo porque implica resolver sistemas de ecuaciones, generalmente no lineales, que crecen en número dependiendo de los grados de libertad del robot a analizar.

Como alternativa a los métodos anteriormente mencionados, uno de los procedimientos para la solución al problema cinemático inverso se denomina "Desacoplo cinemático". Este método se aplica a los robots manipuladores en los que se tiene tres grados de libertad para posicionar a la herramienta final del

manipulador y otros últimos grados de libertad correspondientes a la muñeca que orienta la herramienta del robot. Este tipo de robots suelen ser de configuración antropomórfica.

El desacoplo cinemático consiste en separar el problema cinemático inverso en dos procedimientos, como se observa en la figura 6. En la primera parte se resuelven los primeros grados de libertad mediante métodos geométricos. En la segunda fase se hace uso de la matriz de transformación homogénea, obtenida en la solución de la cinemática directa, para formular ecuaciones que expresen las variables articulares que aún se desconocen, en función de las ya conocidas (tras aplicar primeramente, el método geométrico). (Barrientos, Peñín, Belaguer, & Aracil, 2007)

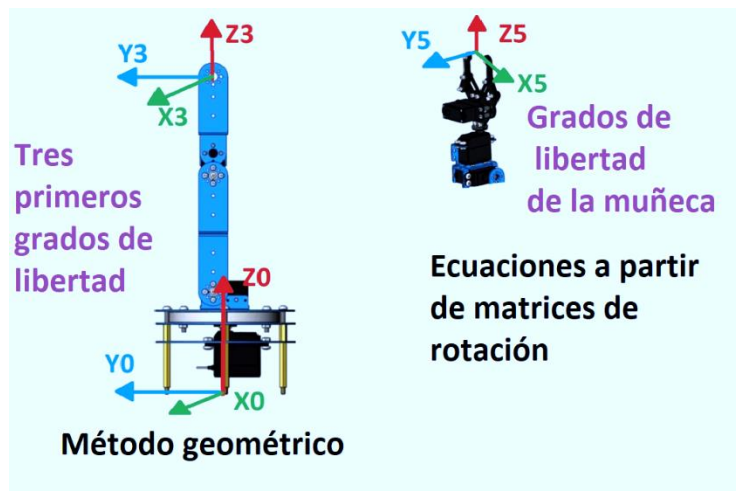


Figura 6 División del problema cinemático inverso en dos partes.

Para que el método de desacoplo cinemático funcione debe calcularse, previamente, el punto medio. El punto medio marca el final del tercer eslabón del robot que, junto a los eslabones posteriores, posicionan a la herramienta del robot. También marca el inicio de la muñeca del robot y es importante para encontrar las tres primeras variables articulares.

Se recuerda que los datos conocidos en la cinemática inversa son posición y orientación de la herramienta final del robot. Contar con estos datos es insuficiente para aplicar el método geométrico. Por esta razón se calcula el punto medio que, desde otra perspectiva, es el punto final del robot cuando se consideran únicamente los tres primeros grados de libertad.

En la figura 7 se ejemplifica la localización del punto medio en la configuración de un robot antropomórfico que cuenta con una herramienta. La posición de la herramienta es el dato conocido en la cinemática inversa. La posición final es respecto al origen del robot, al igual que el punto medio.

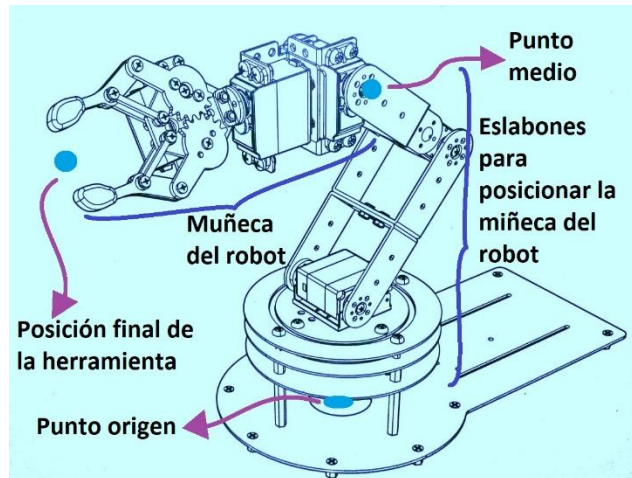


Figura 7 Ubicación del punto medio en un robot manipulador de cinco grados de libertad.

En resumen, para solucionar la cinemática inversa mediante el desacoplo cinemático se necesita calcular el punto medio del robot. Como segundo paso, se encuentran las tres primeras variables articulares, correspondientes a los tres primeros grados de libertad, utilizando el método geométrico. La última parte consiste en hallar las variables articulares restantes, correspondientes a la muñeca del robot, usando las matrices de transformación homogénea para encontrar las expresiones que estén en función de las variables articulares obtenidas mediante el método geométrico.

Capítulo 2 Descripción y Caracterización del brazo robot

En el presente capítulo se exponen las cualidades principales del robot bajo estudio. Se muestran las características físicas que lo conforman, las dimensiones de sus eslabones, el número de eslabones y de articulaciones, entre otras cualidades. A partir de conocer las articulaciones del robot se puede estudiar su espacio de trabajo, que delimita el entorno en el que puede realizar las actividades para las que fue construido.

Se aborda, también, las características de los servomotores, los cuales son los que realizan movimiento de las articulaciones del robot. Estos requieren de una alimentación eléctrica y una señal PWM para posicionarse angularmente.

Las señales PWM son generadas a través de una placa de desarrollo con un microcontrolador. A su vez, la placa se comunica a través de un puerto USB con una computadora para enviar instrucciones al microcontrolador para mover los servos y con ello las articulaciones del robot.

Se muestra en la figura 8 un esquema general de los diversos sistemas y componentes que operan conjuntamente para hacer posible el movimiento del manipulador.

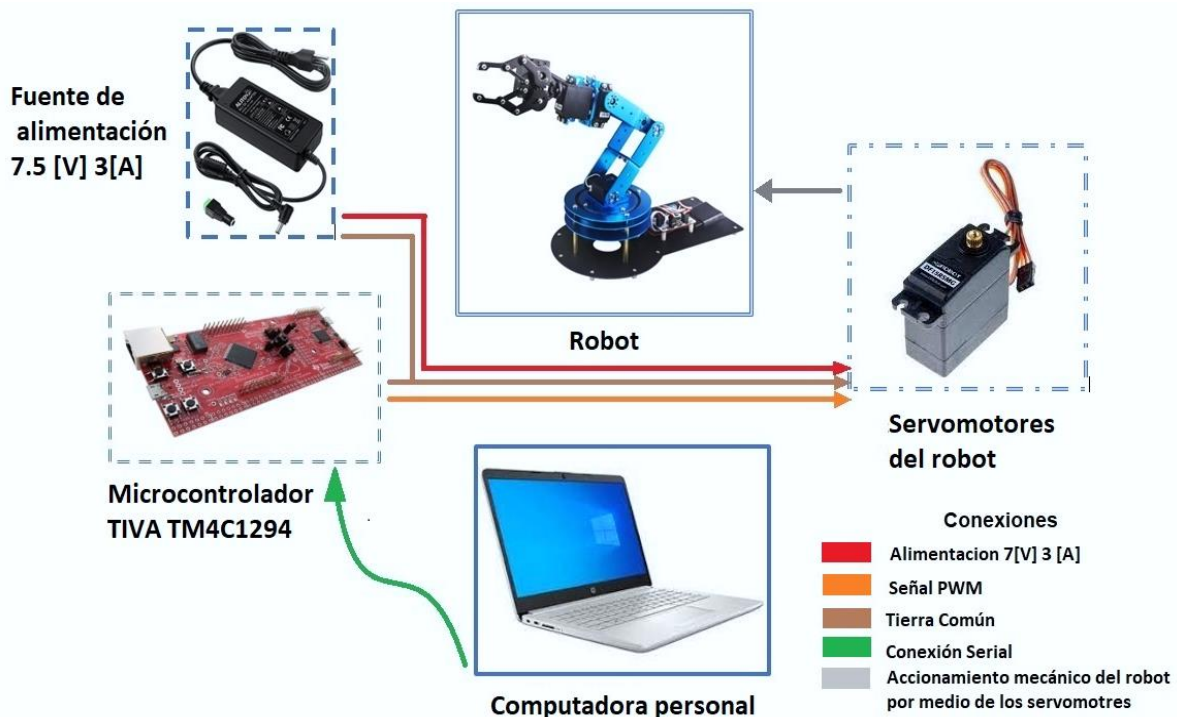


Figura 8 Diagrama de conexiones para la manipulación del robot de cinco grados de libertad.

Características físicas del robot

El robot bajo estudio que se utiliza para su análisis e implementación es un robot fabricado por una empresa China, "Hiwonder". El modelo del robot tiene por nombre "LeArm: Smart 6DOF Robotic Arm".



Figura 9 Compañía que fabrica el robot LeArm.

Este robot es presentado por el fabricante, como un robot educativo. Su estructura está compuesta por una cadena cinemática abierta, con configuración antropomórfica y de seis grados de libertad (tomando en cuenta el movimiento del efector final). El robot está constituido por una base, dos eslabones, haciendo la analogía con un brazo humano, se tratan del brazo y antebrazo respectivamente. Y también una muñeca, con dos grados de libertad, y una pinza como efector final.

Todas articulaciones de este robot son movimientos rotacionales dados mediante seis servomotores.

En la figura 10 se muestran los eslabones que componen el robot LeArm y también la base sobre la que se encuentra montada la estructura.



Figura 10 partes que componen al robot manipulador LeArm de 5 grados de libertad.

La estructura del robot es de aleación de aluminio que lo hace ligero. Sus colores son en tono azul metálico y negro. Con un peso total de 1.68 [Kg], considerando toda la estructura. Y tiene un peso de 1.24 [Kg] sin tener en cuenta la base sobre la que se monta, es decir, considerando únicamente los eslabones que componen la configuración antropomórfica del robot.

Considérese para el robot LeArm una configuración de posiciones de los eslabones tal que, la base pueda tener un ángulo de rotación cualquiera; el segundo eslabón tomando un ángulo de articulación de 90 grados; la articulación del codo, igualmente, un ángulo de 90 grados; el cuarto grado de libertad con un ángulo de 90 grados; el quinto grado de libertad en cualquier posición de ángulo; y con la pinza o efecto final en posición cerrada. Se obtiene una configuración en la que el robot alcanza su máxima extensión de altura y esta es de 0.465 [m]. Esto se visualiza en la figura 11, considerando la configuración descrita.

La base sobre la que se monta la estructura del robot tiene como medidas: 0.12 [m] de ancho y 0.285[m] de largo.



Figura 11 Medidas de la base y altura en una configuración vertical del robot.

La base correspondiente al primer eslabón, que mueve toda la estructura del robot, tiene como diámetro 0.12[m] y está conformada por tres círculos del mismo diámetro. En cambio, semicírculo de la base fija, donde se monta toda la estructura, tiene un diámetro de 0.16[m]. Se muestra esta comparación en la figura 12.

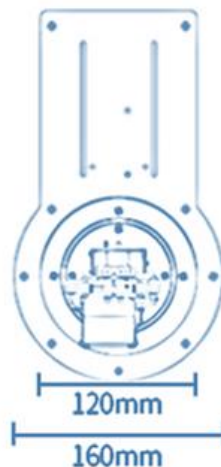


Figura 12 Medidas de los diámetros de la base fija y del primer eslabón del robot.

La longitud del primer eslabón, midiendo desde la base al segundo grado de libertad, es de 0.096[m]

Para el segundo eslabón su medida es de 0.105[m], teniendo en cuenta la medición va desde la articulación del hombro al codo.

El codo es el tercer grado de libertad, mismo que mueve el tercer eslabón, y al final de este se encuentran los restantes dos grados de libertad para la orientación y el efector final. La medida del tercer eslabón es de 0.089[m], tomando en cuenta que se considera desde el inicio de la articulación del codo al punto donde comienza la articulación de cabeceo de la muñeca.

Seguida de la cadena cinemática de posición, se encuentra la subestructura para la orientación del robot, la cual hace uso también de dos grados de libertad. Está integrada por una muñeca y un elemento final de sujeción.

El primer grado de libertad de orientación o cabeceo mueve la subestructura donde se encuentran los elementos para la orientación que corresponden a muñeca y pinza o elemento de sujeción.

La segunda articulación de orientación es la de rotación de la muñeca, cuya función es girar la herramienta de sujeción. En total, la subestructura para orientación tiene una longitud de 0.175 [m] teniendo en consideración que esta medida se toma desde el comienzo de la articulación que realiza el movimiento de cabeceo hasta el punto de agarre de la pinza, considerando que esta última está en posición cerrada.

En la figura 13 se resumen las medidas de los eslabones del robot learn, la altura de la base a la segunda articulación y la medida de la subestructura de la muñeca.

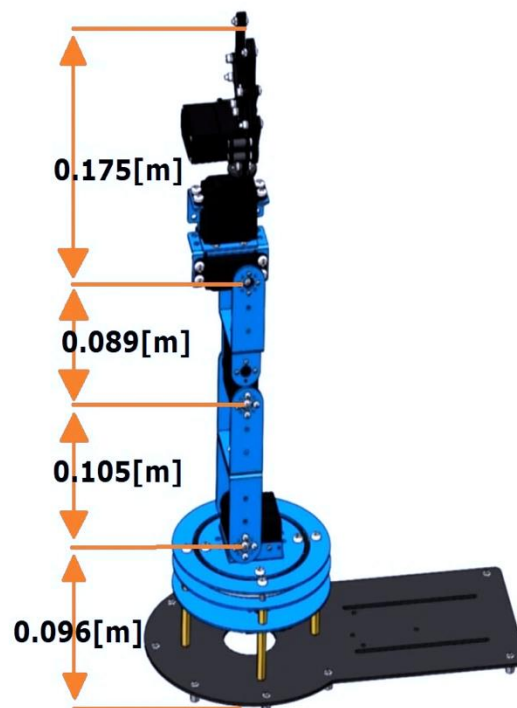


Figura 13 Medias de los eslabones del robot manipulador de cinco grados de libertad.

Por último, se encuentra la herramienta de trabajo o efector final. Se trata de una pinza que realiza la acción de abrir y cerrar para llevar a cabo tareas de sujetar y soltar objetos. En su máxima apertura se tiene una medida de 0.057[m]. Abierta, la herramienta tiene una anchura máxima de 0.098 [m] y alcanza una altura de 0.084[m]. Cerrada la herramienta, la anchura máxima de la herramienta es de 0.048[m] y una altura de 0.105[m]. Estas características se visualizan en la figura 14.

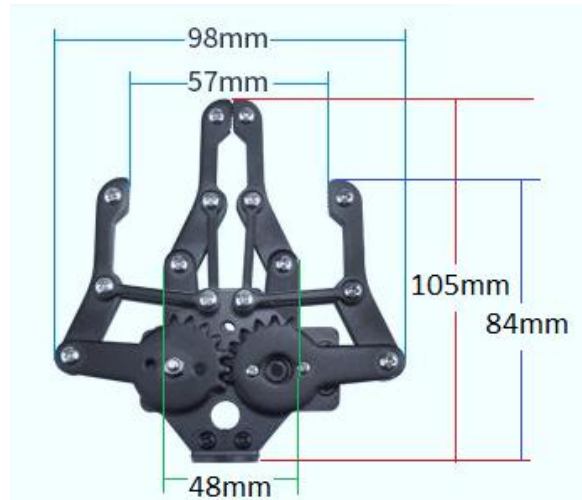


Figura 14 Medidas de la pinza o efector final del robot.

Características mecánicas

Las articulaciones realizan desplazamientos angulares por la acción del movimiento rotacional de los servomotores instalados en cada articulación, incluyendo la herramienta de trabajo.

De acuerdo con las características de los servomotores, las articulaciones tienen un rango de movimiento que va desde los 0 hasta los 180 grados. El servomotor montado en la herramienta final, es una excepción. Este último, opera en un rango que va de los 90 a los 180 grados.

De la figura 15 a 20 se hace muestra el movimiento de cada articulación del robot LeArm.

El servomotor de la base opera en un rango de 0° a 180° y el movimiento que realiza la articulación sobre la que está montado, se muestra en la figura 15.

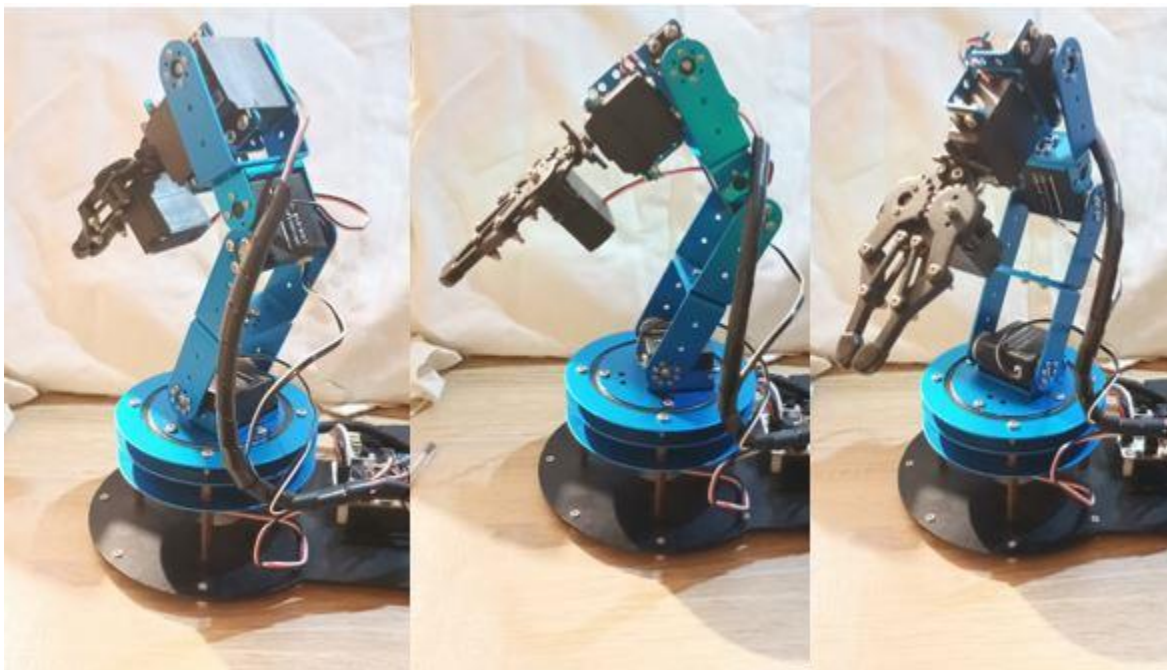


Figura 15 Rango de movimiento del primer grado de libertad.

La segunda articulación u hombro se mueve en un rango de 0° a 180° y el movimiento que realiza se observa en la figura 16.

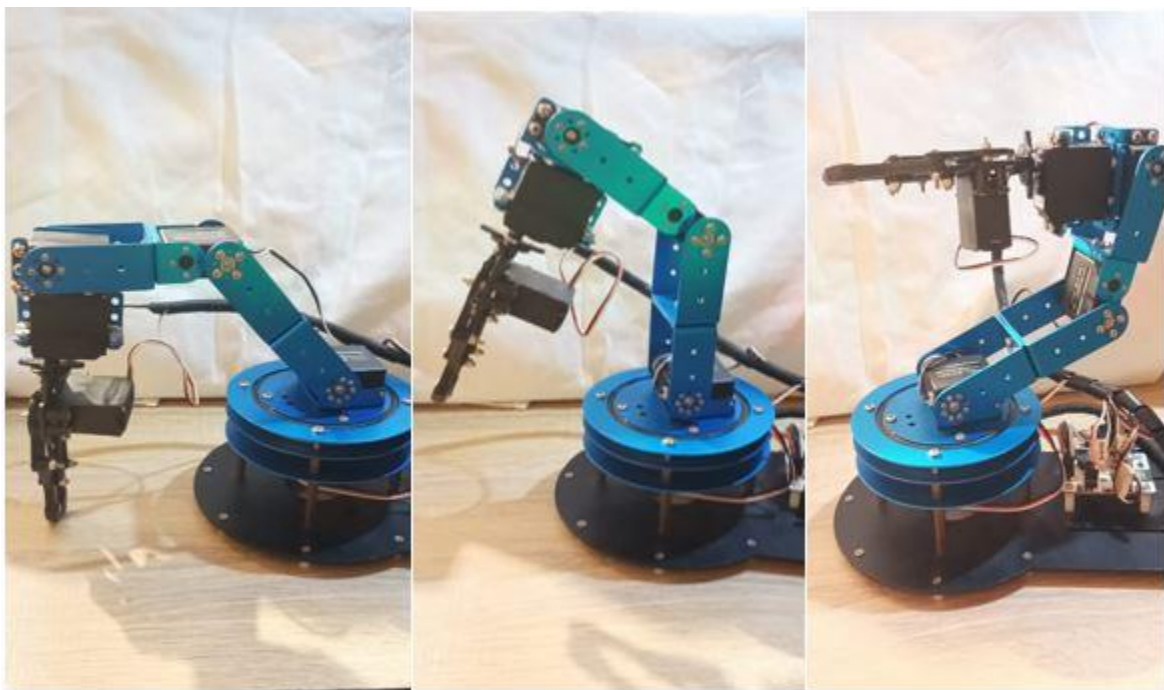


Figura 16 Rango de movimiento del segundo grado de libertad.

La tercera articulación o codo se mueve en un rango de 0° a 180° y el movimiento que realiza se visualiza en la figura 17.



Figura 17 Rango de movimiento del tercer grado de libertad.

La cuarta articulación de la cadena y primera de la muñeca tiene un rango de movimiento que va desde los 0° a los 180° . El movimiento realizado se muestra en la figura 18.



Figura 18 Rango de movimiento del cuarto grado de libertad.

El quinto grado de libertad y el segundo grado de libertad de la muñeca corresponde al de la rotación de la herramienta de trabajo. Su rango de movimiento va de los 0° a los 180° y sigue un movimiento como se observa en la figura 19.

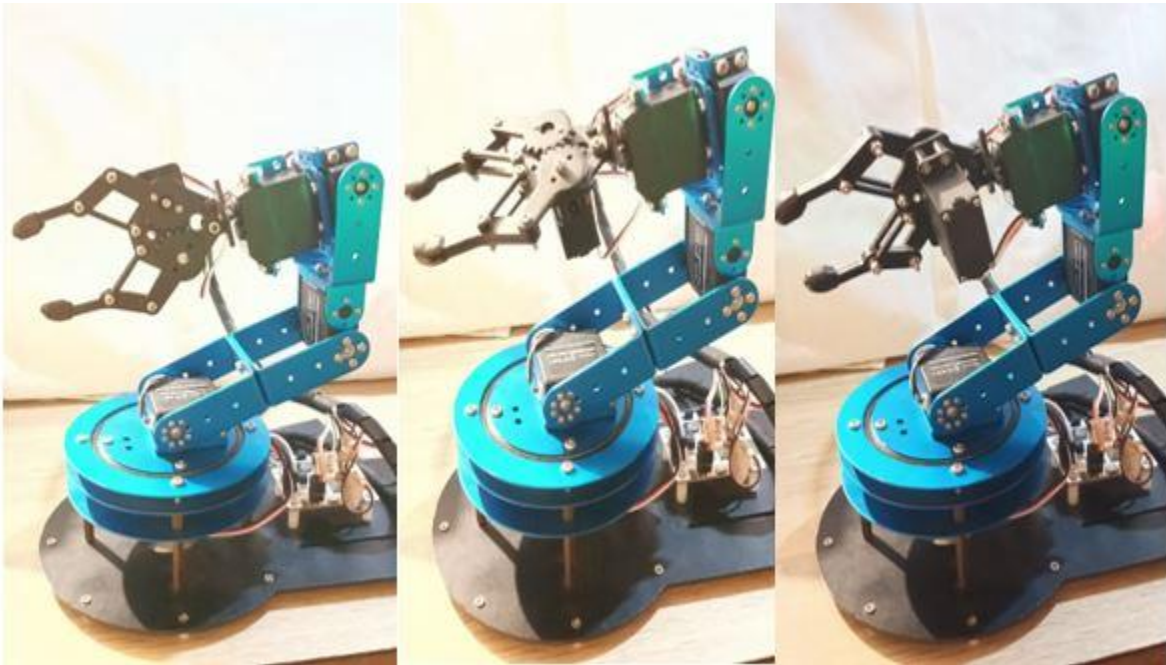


Figura 19 Rango de movimiento del quinto grado de libertad.

La herramienta de trabajo opera en un rango de 90° a 180° . Cuando se posiciona el servomotor a 90° , la pinza está en su máxima apertura. Cuando se posiciona el servomotor a 180° la pinza se encuentra totalmente cerrada. El movimiento es como se muestra en la figura 20.

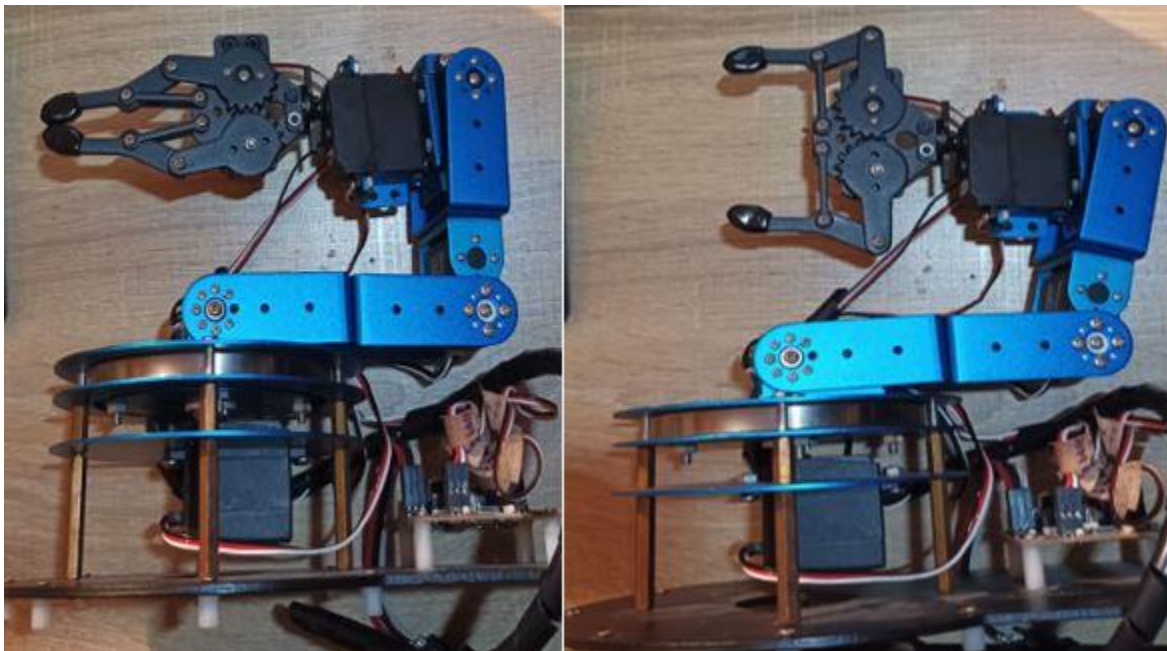


Figura 20 Rango de movimiento de la pinza o efector final del robot.

Estudiados los rangos de movimiento de cada articulación, del robo LeArm, es posible entender el espacio de trabajo de este manipulador.

El espacio de trabajo es e

l volumen alcanzable por el efector final como consecuencia del rango de movimiento de sus articulaciones.

El robot LeArm es antropomórfico, y, al igual que otros de su tipo, el espacio de trabajo es de forma esférica.

Este caso particular de estudio, el robot LeArm es capaz de moverse en un volumen compuesto por una semiesfera superior de 0.369[m] de radio y por una porción de una segunda semiesfera inferior que tiene una altura de 0.096[m]. La Figura 21 ilustra este espacio de trabajo.

Las dos porciones (superior e inferior) del espacio de trabajo, son partes de una esfera que no se visualiza en la figura 21, pero que se intuye. Esta esfera imaginaria tendría un centro que pasa por el eje de rotación de la segunda articulación del robot LeArm.

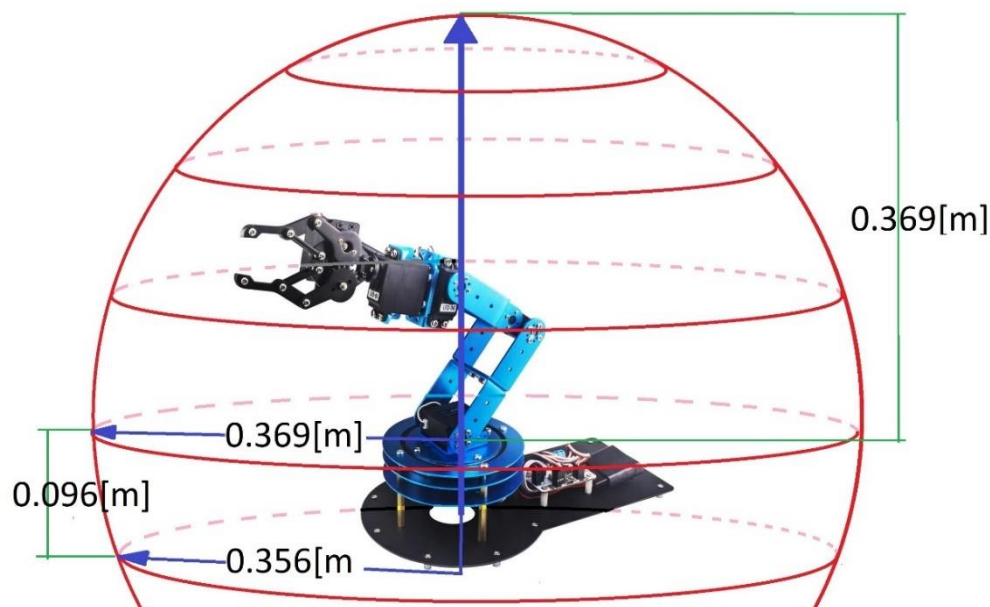


Figura 21 Espacio de trabajo del robot de cinco grados de libertad.

Lo presentado como espacio de trabajo en la figura 21, se reduce para efectos de las aplicaciones y actividades planteadas en los capítulos posteriores. Una razón para trabajar con una porción del espacio de trabajo es porque hay una zona donde se encuentra la base sobre la que se monta el robot. En esta porción de la base está instalada una placa donde se colocan algunos componentes del robot y de la que también sobresalen los cables que conectan los servomotores. Por estos componentes instalados es complicado manipular objetos. Para evitar estos

inconvenientes se reduce el volumen a la mitad, trabajando solo con las partes frontales de la semiesfera y la porción de la semiesfera.

El espacio de trabajo queda reducido como se observa en la figura 22.

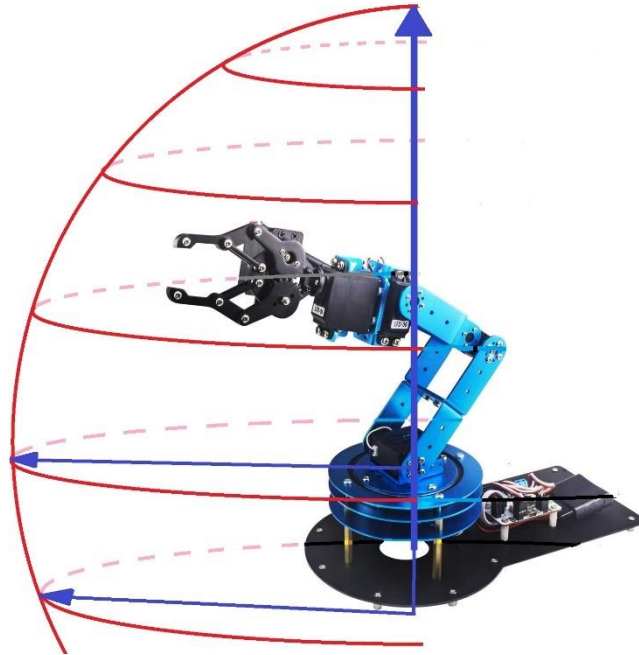


Figura 22 Espacio de trabajo con el que se trabaja en el presente trabajo de tesis.

Características eléctricas y Servomotores

Cada articulación es accionada mediante servomotores, los cuales operan en un rango de 0 a 180 grados. Para dar la indicación a un servo de posicionarse en cierto ángulo, se envía una señal eléctrica con ancho de pulso (PWM).

Esta señal varía de 500 a 2500 milisegundos y se corresponden con los extremos angulares que el servo puede alcanzar, 0 y 180 grados, respectivamente. Se tiene una excepción para el servomotor encargado del movimiento de la pinza, que opera en el rango de 90 a 180 grados o su equivalente en señal de ancho de pulso de 1500 a 2500 milisegundos.

En la figura 23 se visualiza la relación del ancho de pulso con la posición angular de los servomotores, considerando los ángulos principales 0° , 90° y 180° . Se muestra que la distancia entre ancho de pulso es de 20 milisegundos [ms] y es igual para cada una de las señales.

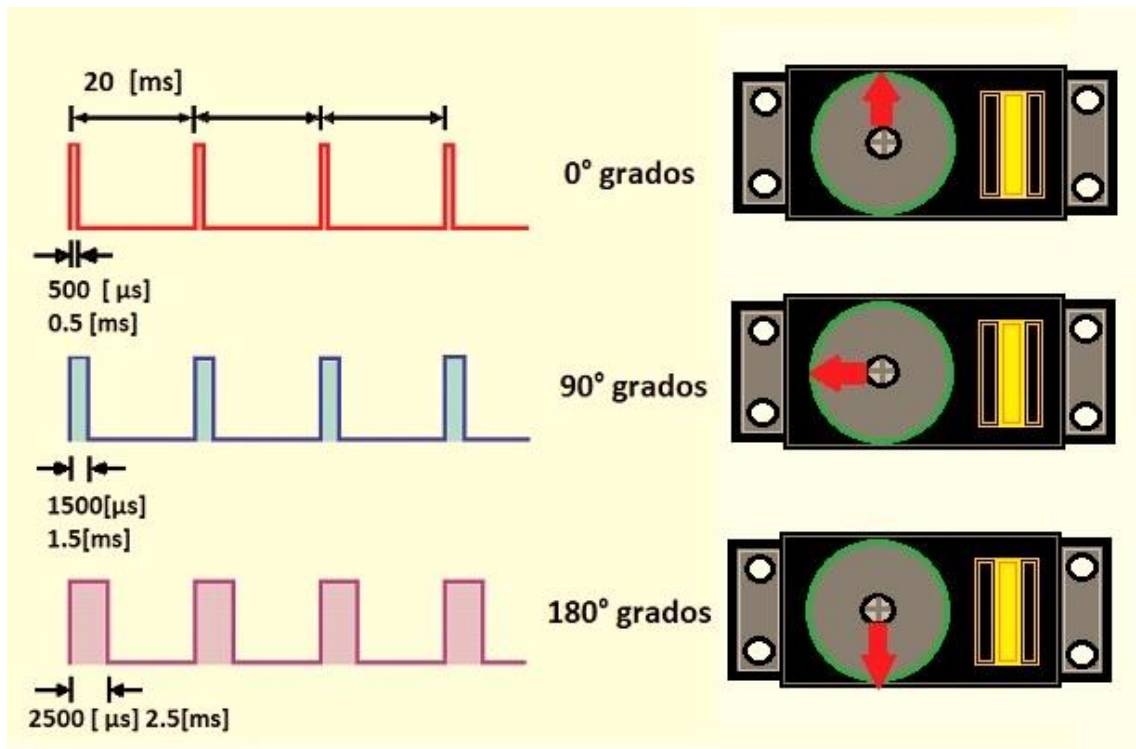


Figura 23 Ancho de pulso que posiciona a un servomotor en 0, 90 y 180 grados, respectivamente.

Los servomotores utilizados en el robot son:

- El modelo LD-1510MG es empleado para la articulación de la base. Este servomotor tiene un alto torque de hasta 20 [Kg*cm] cuando se alimenta con una tensión de 7 volts. Este alto torque es de gran relevancia para mover el peso de los eslabones acoplados al eslabón base.
- El modelo LDX-218 es utilizado para las articulaciones de hombro y codo. Este modelo de servomotor tiene un torque máximo de hasta 17 [Kg*cm] cuando la tensión de entrada es de 7 volts.
- El modelo LFD-06 es usado para las articulaciones de la muñeca (cabeceo y rotación) y la herramienta de trabajo. No hay datos acerca de este servomotor, pero se estima en comparación con otros servomotores, que tiene un torque de hasta 9 [Kg*cm] cuando es alimentado con una tensión de 7 volts.

Otras características e información adicionales de los servomotores, se muestra en las tablas 1,2 y 3.

Tabla 1 Datos para el servomotor LDX-2018.

Nombre:	LDX-218 Digital servo
Marca:	Hiwonder
Peso:	56 [g]
Medidas:	40*20*51.4 [mm]
Voltaje de trabajo:	6-8.4 [v]
Velocidad:	0.16sec/60° 7.4 [V]
Torque:	15 [Kg*cm] 6[V];17 [Kg*cm] 7.4[V]
Rotación:	0°- 180°
Corriente:	2.4-3 [A]
Método control:	PWM
PWM rango de pulso:	500- 2500 [μs]

Tabla 2 Datos para el servomotor LFD-06.

Nombre:	LFD-06 Digital servo
Marca:	Hiwonder
Peso:	62 [g]
Medidas:	40*20*50[mm]
Voltaje de trabajo:	4-7 [v]
Velocidad:	0.12sec/60° 7.4 [V]
Torque:	15 [Kg*cm] 6[V];17 [Kg*cm] 7[V]
Rotación:	0°- 180°
Corriente:	2.4-3 [A]
Método control:	PWM
PWM rango de pulso:	500- 2500 [μs]

Tabla 3 Datos para el servomotor LD-1501MG.

Nombre:	LD-1501MG Digital servo
Marca:	Hiwonder
Peso:	63 [g]
Medidas:	40.7*20.5*39.5[mm]
Voltaje de trabajo:	4.8-7.2 [v]
Velocidad:	0.14sec/60° 7.4 [V]
Torque:	17 [Kg*cm] 6[V];20 [Kg*cm] 7[V]
Rotación:	0°- 180°
Corriente:	2.4-3 [A]
Método control:	PWM
PWM rango de pulso:	500- 2500 [μs]

La fuente de alimentación principal es un adaptador con una alimentación de entrada de 100-240 [V] a 50 o 60 [Hz] y una salida de 7.5 [V] con corriente de 3 [A]. Este adaptador, mostrado en la figura 24, da potencia a cada uno de los servomotores.



Figura 24 Adaptador de 7.5 volts que permite alimentar con corriente a los servomotores del robot LeArm.

TIVA tm4c1294

Para ser controlados, los servomotores reciben una señal PWM y, como se ha descrito anteriormente, el ancho de esta señal posiciona el servo en un ángulo específico.

La señal PWM puede ser generada en un circuito electrónico en el que, el ancho de pulso se modifica por medio de un potenciómetro. También, existen otras formas de producir este tipo de señales sin necesidad de manipulación física y con precisión del ancho de pulso.

Los microcontroladores y tarjetas de desarrollo son dispositivos programables que permiten generar señales de salida PWM. Pueden ser tantas señales como se necesiten y permita el dispositivo usado.

Existen algunas placas comerciales que son utilizadas para estos fines. Una de las opciones principales es “Arduino” y en específico “Arduino uno”, que es una placa de desarrollo pequeña.

Para el trabajo presente, se decidió usar la tarjeta de la compañía “TEXAS Instruments”, la TIVA TM4C1294 que se visualiza en la figura 25. Una característica destacada de esta, es la velocidad de 120 [MHz] de su procesador. Siete veces la velocidad del “Arduino uno” que cuenta con una velocidad de 16 [MHz].

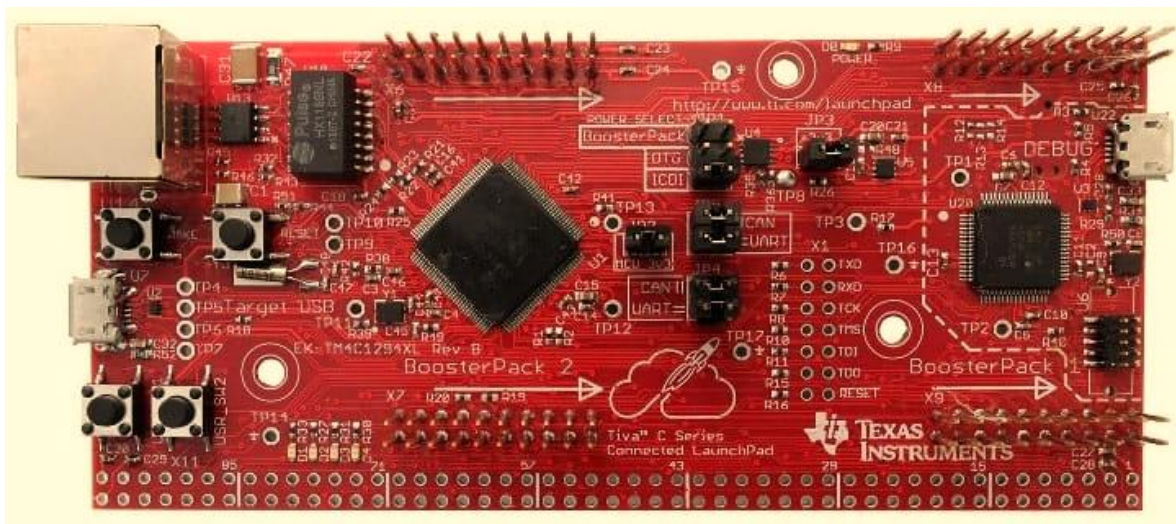


Figura 25 Tarjeta TIVA TM4C129

La TIVA LaunchPad TM4C1294 es una tarjeta de desarrollo ideal para aplicaciones industriales que incluyen monitoreo remoto, automatización en red, redes de control-comunicación industrial, pruebas y mediciones, entre otras aplicaciones.

Además, la TIVA LaunchPad TM4C1294 tiene como características principales las siguientes:

- 120 [MHz] 32-bit ARM Cortex-M4 CPU.

- 1 [MB] Flash, 256KB SRAM, 6 [KB] EEPROM.
- 10/100 Ethernet MAC+PHY.
- Data protection hardware, 8x 32-bit timers.
- 2 módulos CAN.
- Interfaces QSSI/UART/I2C.
- 2 Puertos USB 2.0 integrados.
- 40 pines más 40 “headers” adicionales para que el usuario los use a conveniencia. Pines que pueden ser usados como entradas/salidas; digitales o análogas para lectura y escritura de datos; para comunicación UART, I2C, SPI; señales PWM; salidas de voltaje 5 [V], 3.3 [V] y tierra.

Otra característica destacada la TIVA, es que sus pines pueden ser usados como analógicos o digitales, según convenga.

Para programar la TIVA se puede usar un compilador y entorno proporcionado por la compañía “TEXAS nstruments” que da la opción de emplear lenguaje ensamblador o “C”. Existe una alternativa, la plataforma “Energía”, de libre desarrollo. Basada en un entorno similar al de la programación de Arduino. Esta, cuenta con librerías que simplifican el proceso de programación.

El entorno “Energía” fue elegido como plataforma de programación. Sus ventajas son la sencillez del código de programación y la utilización de librerías para simplificar procesos, haciendo correcciones instantáneas sin analizar mucho código. En la figura 26 se observa la interfaz de entorno de energía.

```
Energia_Rocks.ino | Energia 1.6.10E18
Energia_Rocks.ino
1 #define LED RED_LED
2
3 // the setup routine runs once when you press reset:
4 void setup() {
5   // initialize the digital pin as an output.
6   pinMode(LED, OUTPUT);
7 }
8
9 // the loop routine runs over and over again forever:
10 void loop() {
11   digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
12   delay(1000); // wait for a second
13   digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
14   delay(1000); // wait for a second
15 }
Done Saving.
15 RED LaunchPad w/ msp432 EMT (48MHz) on /dev/cu.usbmodemM4321001
```

Figura 26 Entorno de programación “Energía”.

Comunicación serial USB

La tarjeta TIVA TM4C tiene dos puertos USB. Uno de ellos sirve para comunicarse con un dispositivo con esta entrada, usualmente una computadora. El microcontrolador TM4C1294NCPDT hace posible este tipo de comunicación, que entre otras cosas, permite programar la tarjeta

La conexión USB es de tipo serial, siendo capaz de recibir y transmitir datos entre este dispositivo y la computadora. Los conectores USB tienen 4 hilos sobre los que se transmite la información y la alimentación eléctrica.

Los pines de conexión son:

1. Fuente de alimentación de +5 [V] (VBUS) corriente máxima 100 [mA]
2. Datos (D-)

3. Datos (D+)
4. Conexión a tierra (GND)

Se puede observar en la figura 27 como están distribuidos físicamente cada pin de un conector USB.

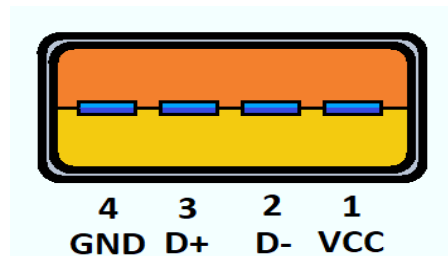


Figura 27 Esquema conector USB.

La capacidad de recibir y transmitir datos es relevante para poder controlar los servomotores mediante la tarjeta TIVA. De esta manera, se envían los datos de posición angular para cada uno de los servos en diversos tiempos o instantes, cambiando esos valores con la velocidad y precisión, requeridos.

Controlarlos desde una computadora abre distintas posibilidades. Una manera es que un usuario cambie manualmente los anchos de pulso de las señales PWM. Otra forma, es controlar el ancho de pulso automáticamente mediante programación.

Plataformas de software de computadora como LabVIEW, son capaces de crear comunicaciones entre usuario y hardware mediante una programación gráfica.

Plataforma LabVIEW

LabVIEW es un software con un entorno de programación gráfica que sirve para el desarrollo de sistemas de pruebas automatizadas; la creación de aplicaciones de automatización, control, medición. También es usado para el análisis y registro de datos. Los datos pueden ser monitorizados y provenir de mediciones de sonido, procesamiento de imágenes, análisis de frecuencias, etcétera. Su alcance es tal que se utiliza para aplicaciones de automatización industrial a gran escala. (Ni, 2023)

Los programas desarrollados con LabVIEW se llaman instrumentos virtuales. Este nombre es apropiado, teniendo en cuenta la posibilidad del software de conectarse con sensores, tarjetas de adquisición de datos y otro hardware mediante los buses de comunicación de una computadora. Además, la interfaz gráfica de usuario permite visualizar los datos, graficarlos, e interactuar mediante botones, perillas y otros objetos que simulen un panel de mando de algún instrumento. (Lajara Vizcaíno & Pelegrí Sebastián, 2018)

Existen dos ventanas principales en este software. La primera posibilita la interacción del usuario, se llama Panel de control. Se pueden desplegar controles e

indicadores que se arrastran al panel y se sueltan en el lugar deseado para generar la interfaz a criterio del desarrollador.

Los controles son aquellos elementos que simulan dispositivos de entrada para dar un suministro de datos. Estas entradas se encuentran en forma de perillas, pulsadores, botones y otros.

Por otra parte, los indicadores son aquellos elementos que van a simular dispositivos de salida. Estos pueden presentarse en forma de LED, medidores, cuadros de texto, etcétera.

En la figura 28 se observa un ejemplo de panel posterior en el que se hace una simulación de un brazo robot de tres grados de libertad. Se visualizan algunos controles con los que es posible cambiar la perspectiva angular de la simulación y también mover las articulaciones del manipulador.

En la figura 29 se tiene el diagrama de bloques correspondiente al programa de la figura 28. Se observan algunos bloques que hacen posible la simulación del robot. Y también la interacción entre estos por medio de los cables de conexión.

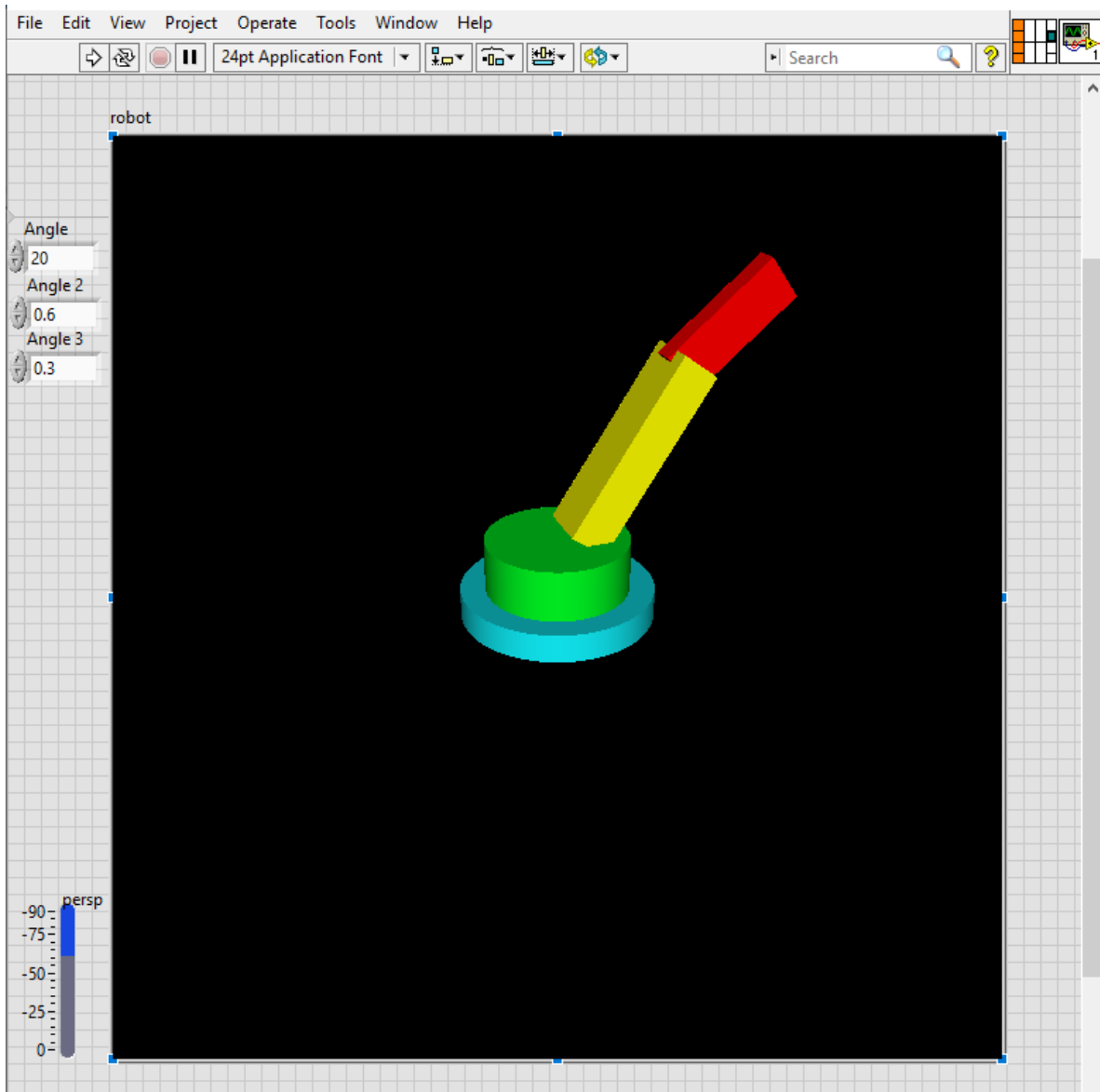


Figura 28 Panel de Control del entorno de programación de LabVIEW. Se ejemplifica con un programa para la manipulación de un robot de tres grados de libertad.

La segunda ventana llamada Panel posterior o Diagrama de bloques, se muestra el código utilizado en este software en forma de bloques. Esta sintaxis de programación por su configuración facilita la visualización de los diversos elementos y su interacción entre ellos.

Se cuenta con diversos bloques con operaciones, ya sean aritméticos, lógicos, matriciales. Otros como los conocidos ciclos “for”, “while” usados en algunos lenguajes de programación textual. E incluso, hay unos más complejos que dan color y orientación a figuras geométricas.

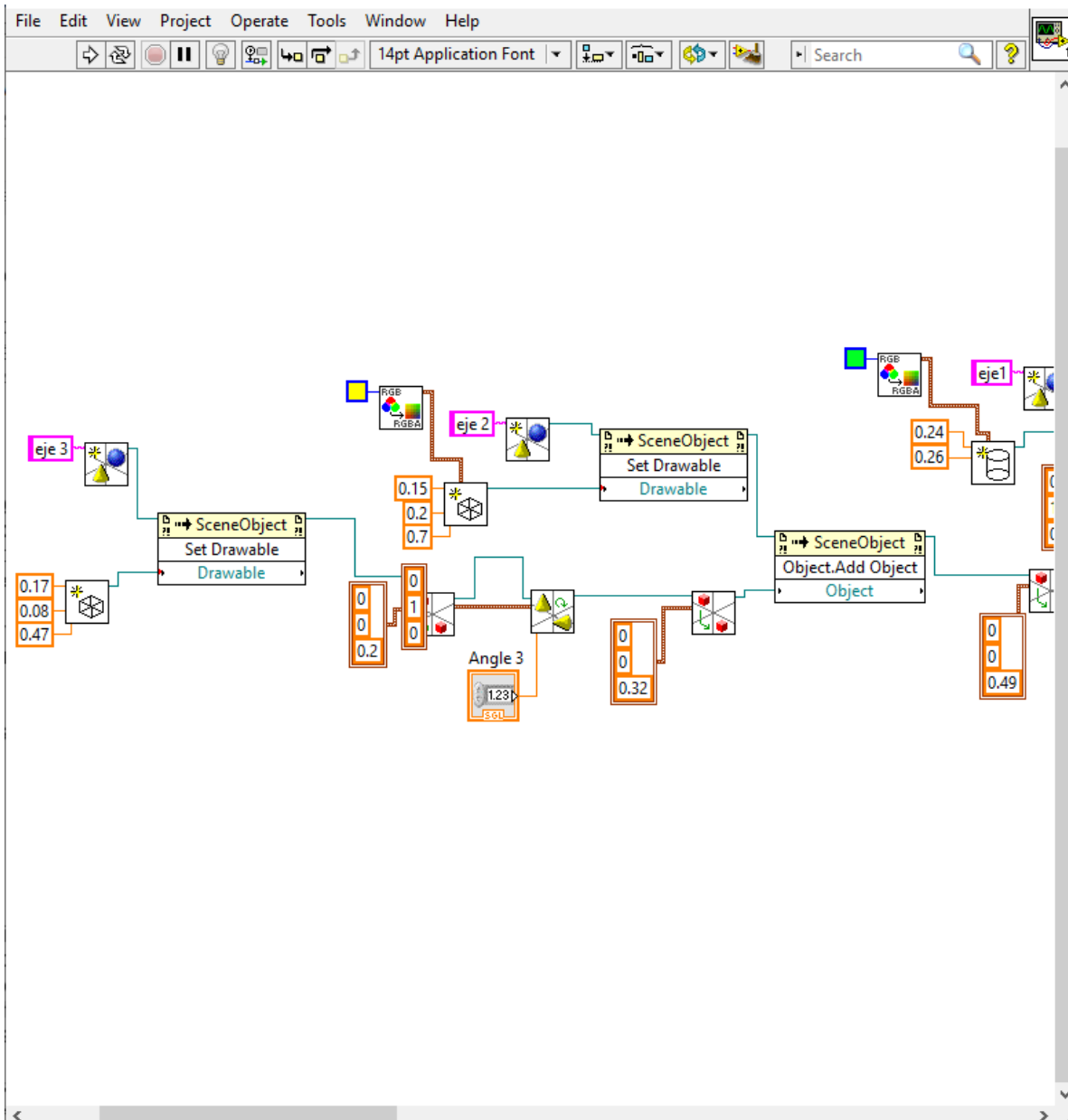


Figura 29 Diagrama de bloques del entorno de programación de LabVIEW. Se ejemplifica con un programa para la manipulación de un robot de tres grados de libertad.

Las ventajas de usar un software gráfico son la sintaxis intuitiva que permite una gran interacción con los elementos en comparación con los lenguajes de programación basados en texto.

Otra ventaja es la facilidad para encontrar y dar solución a los errores de programación sin necesidad de hacer algún tipo de compilación de código. Se tiene, por tanto, un rendimiento del sistema que permite desarrollar tareas con gran rapidez.

La velocidad de ejecución dada por este software también es vital para las aplicaciones que se dan en la industria, la investigación científica y técnica, de esta

manera es posible desarrollar y dar solución a las necesidades de alta exigencia de esos sectores.

Librería Visa

LabVIEW da la posibilidad de integración de diversos sensores, instrumentos, módulos y demás hardware. Esto hace necesario el uso de los buses y entradas de una computadora. Para crear una comunicación con estos dispositivos el software ofrece controladores.

VISA es un controlador de instrumentos con una implementación estándar de entradas/salidas. Configura, programa y depura sistemas de instrumentación. Sin importar el tipo de interfaz, VISA usa las mismas operaciones para comunicarse.

Sin necesidad de aprender protocolos de comunicación para cada tipo de interfaz, se proporcionan las herramientas que facilitan la comunicación con los instrumentos conectados. Las funciones inician comunicación con el puerto, escriben, leen datos y cierran comunicación. Estas son: "VISA Configure Serial Port", "VISA Read Function", "VISA Write Function" y "VISA Close Function". (Ni, 2023)

VISA Configure Serial Port

Esta herramienta de programación gráfica, que se muestra en la figura 30, inicializa el puerto serial especificado por "VISA Resource name" y la configuración para los demás pines como la tasa de baudios, el número de bits transferidos a través del puerto, la paridad. En el caso de las aplicaciones que se usaron para los experimentos llevados a cabo, no es necesario editar cada uno. Se configuró el pin de "VISA Resource" para que el usuario pueda escoger el puerto en el cual se conectó la tarjeta TIVA u otro instrumento. Se estableció también, con una constante, el número de baudios con un valor de velocidad de comunicación serial de 9600 baudios. Los demás pines no se configuran y estos toman valores por efecto. En la figura 31 se observa como son las conexiones para este bloque.

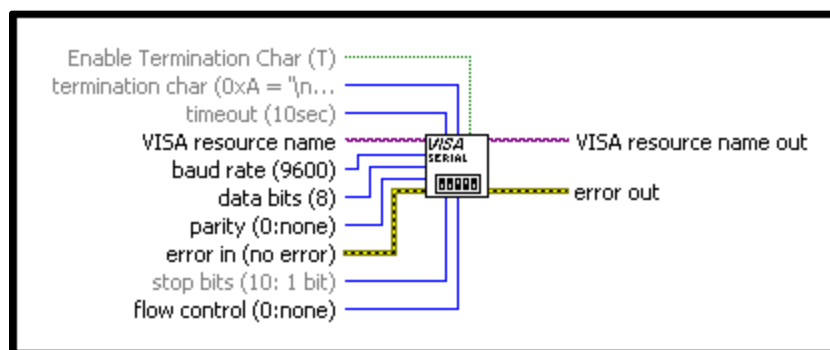


Figura 30 "VISA configure Serial Port" permite configurar un puerto serial.

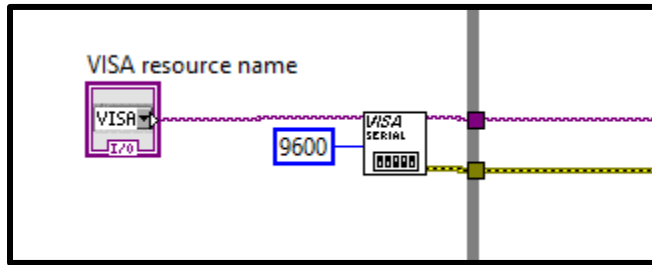


Figura 31 Selección de puerto serial y de velocidad de datos.

VISA Read Function

Este bloque, como se visualiza en la figura 32, lee el número preciso de bytes del dispositivo o interfaz especificado por el "VISA Resource name" y devuelve los datos en el búfer de lectura (Read buffer). Cuando no se especifica el número de bytes a leer, se toma todos los datos por defecto.

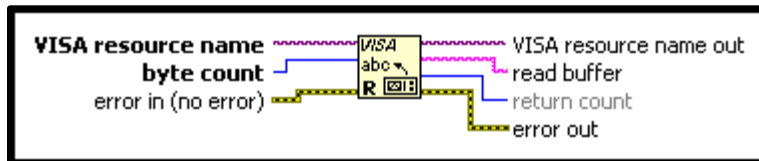


Figura 32 Bloque de "VISA Read Function".

VISA Write Function.

Este bloque, que se observa en la figura 33, escribe los datos del búfer de escritura en el dispositivo o interfaz especificado por "VISA Resource name".

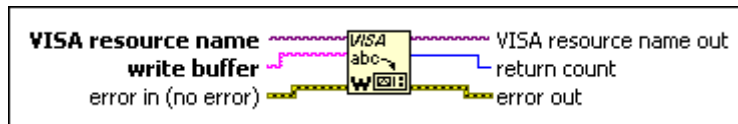


Figura 33 Bloque de "VISA Write Function".

VISA Close Function

Este bloque, con el aspecto físico que se muestra en la figura 34, cierra una sesión de dispositivo o un objeto de evento especificado por el nombre del "Visa Resource", es decir, se termina la comunicación serial.

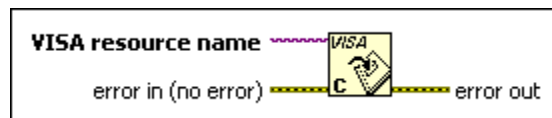


Figura 34 Bloque "VISA Close Function".

Para ejemplificar la escritura y lectura de datos por medio del puerto serial, se muestra en las figuras 35 y 36, el diagrama de bloques que se tiene que programar para realizar las respectivas tareas. Cuentan, adicionalmente, con un botón "stop" que realiza el paro total del programa.

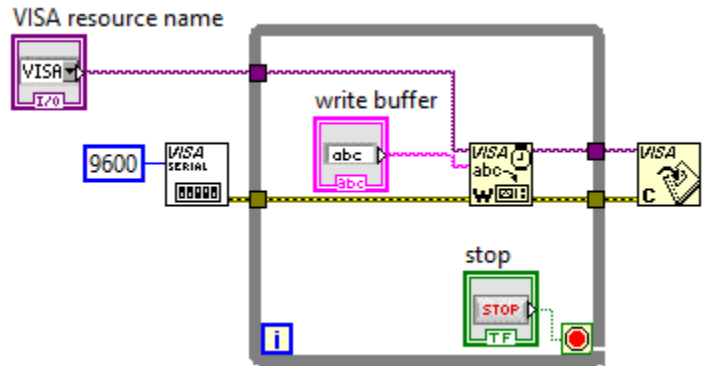


Figura 35 Configuración básica para escribir datos en el puerto serial, con funciones de VISA.

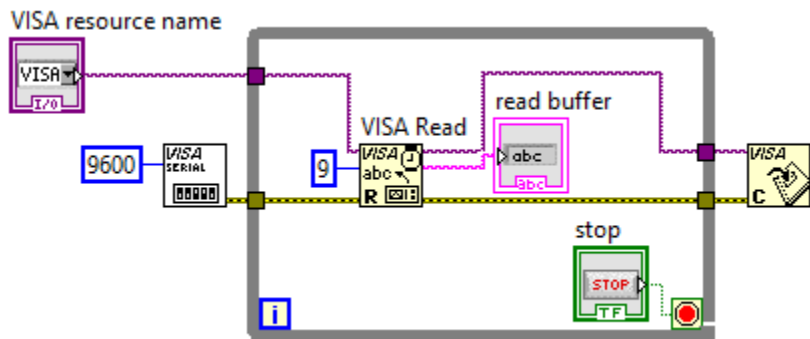


Figura 36 Configuración básica para leer datos en el puerto serial, con funciones de VISA

Capítulo 3 Análisis de la cinemática directa e inversa.

Hasta este capítulo se ha analizado el tema de la cinemática, explicando las dos formas de abordar el problema en lo que se refiere a los robots manipuladores: la cinemática directa y la cinemática inversa; Posteriormente se han detallado las características del “robot manipulador”, siendo este un robot antropomórfico con cinco grados de libertad, el cual cuenta con cinco eslabones rotacionales y posee una pinza como herramienta final, por lo que en este capítulo se describe el análisis cinemático directo e inverso.

Implementación de algoritmo Denavit Hartenberg para robot bajo estudio.

El objetivo de implementar el algoritmo de Denavit-Hartenberg es determinar las variables articulares (θ_i , d_i , a_i y α_i) asociadas a cada eslabón que forma la cadena cinemática del robot manipulador. Con las variables articulares es posible formar matrices de transformación homogénea, una por cada eslabón del robot manipulador. Para el caso de este robot antropomórfico, LeArm, con 5 grados de libertad se requiere de 5 matrices de transformación homogénea para que al ser multiplicadas se obtenga una matriz general que contiene la información de posición y orientación de la herramienta final del robot. Al obtener la matriz de transformación homogénea total, se tiene la solución al problema cinemático directo.

El algoritmo de Denavit-Hartenberg asigna sistemas de ejes en las articulaciones, en la herramienta final y en la base del robot, siendo esta última, el sistema origen. La orientación que tome cada eje de los sistemas de ‘n’ ejes O_n , influyen en los valores o existencia de las variables articulares. Por tanto, el algoritmo de Denavit-Hartenberg guía en la asignación de la ubicación y orientación de cada sistema de ejes para obtener un diagrama similar al que se muestra en la figura 37.

Se hace el análisis de los ejes para determinar los valores de las variables articulares.

Se recuerda que cada variable tiene las siguientes características, considerando que se analiza el eslabón ‘i’.

θ_i es el ángulo que hay que girar en torno al eje z_{i-1} para que los ejes x_{i-1} y x_i queden paralelos.

d_i es la distancia, medida a lo largo del eje z_{i-1} , que hay que desplazar el origen O_{i-1} para que los ejes x_i y x_{i-1} queden alineados.

a_i es la distancia medida a lo largo del eje x_i (que ahora coincide con el eje x_{i-1}) que hay que desplazar el nuevo origen O_{i-1} para que coincida con el origen O_i .

α_i como el ángulo que hay que girar en torno al eje x_i , para que el nuevo origen O_{i-1} coincida totalmente con O_i .

Es posible que algunas características no se cumplan y, por tanto, no exista algunas de las variables articulares asociadas a algún eslabón analizado.

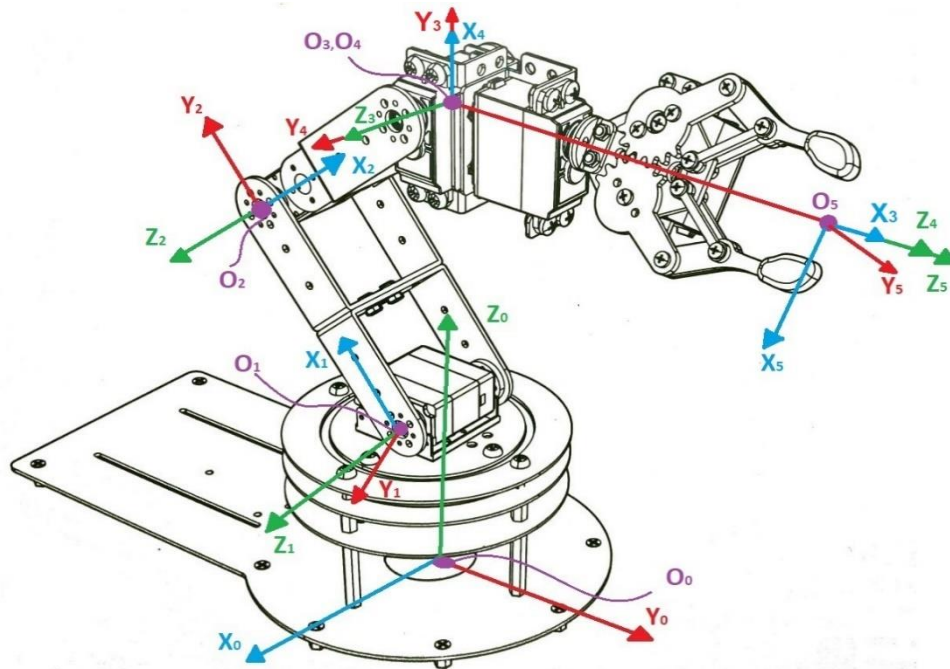


Figura 37 Asignación de sistemas de ejes para robot de cinco grados de libertad.

Para resolver las variables articulares con base en la cinemática del robot “LeArm” haciendo uso del algoritmo Denavit Hartenberg se lleva a cabo el siguiente procedimiento:

El algoritmo se basa en el expuesto en la referencia . (Barrientos, Peñín, Belaguer, & Aracil, 2007)

1. Se asigna el número de eslabón ‘0’ a la base.
2. Se enumeran los eslabones de la cadena cinemática asignando el ‘1’ al primer eslabón, el ‘2’ al segundo eslabón y así consecutivamente hasta llegar al último eslabón ‘n’, que comúnmente es la herramienta final, como se muestra en la figura 38.

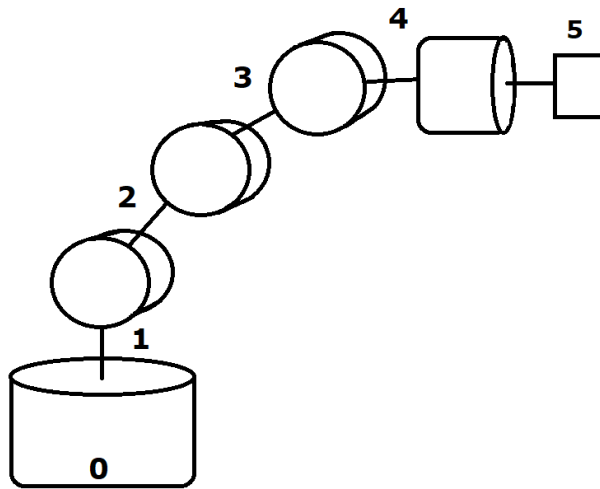


Figura 38 Enumeración de eslabones para el robot LeArm.

3. Se enumera cada articulación comenzando por la primera que está en la base, asignándole el número '1' y acabando en la articulación 'n'. Se debe recordar que cada articulación corresponde a un grado de libertad, como se indica en la figura 39.

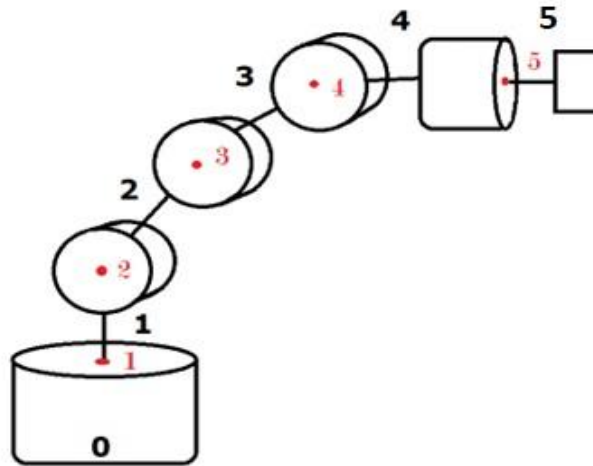


Figura 39 Enumeración de articulaciones para el robot LeArm.

- Se localiza el eje de cada articulación, como se visualiza en la figura 40. Para articulaciones rotativas será su propio eje en que ocurre el giro. Para articulaciones prismáticas, será el eje a lo largo del cual hay desplazamiento.

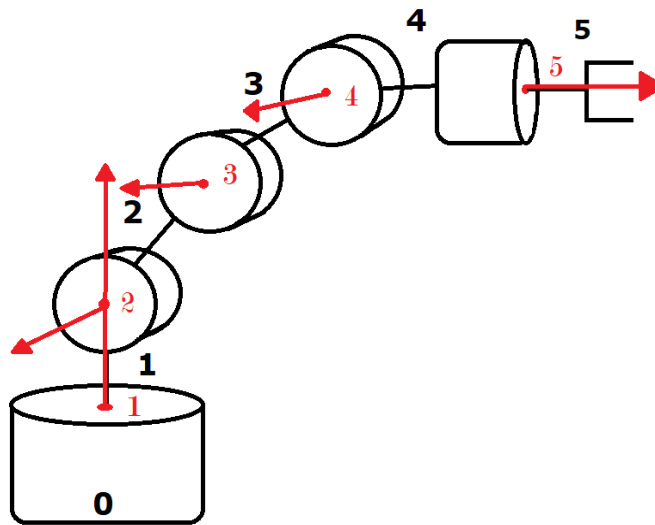


Figura 40 Localización del eje de giro para cada articulación del robot LeArm.

- Se coloca el eje z_i , como se indica en la figura 41, sobre el eje de la articulación $i+1$, donde i comienza en '0' y termina en 'n-1'.

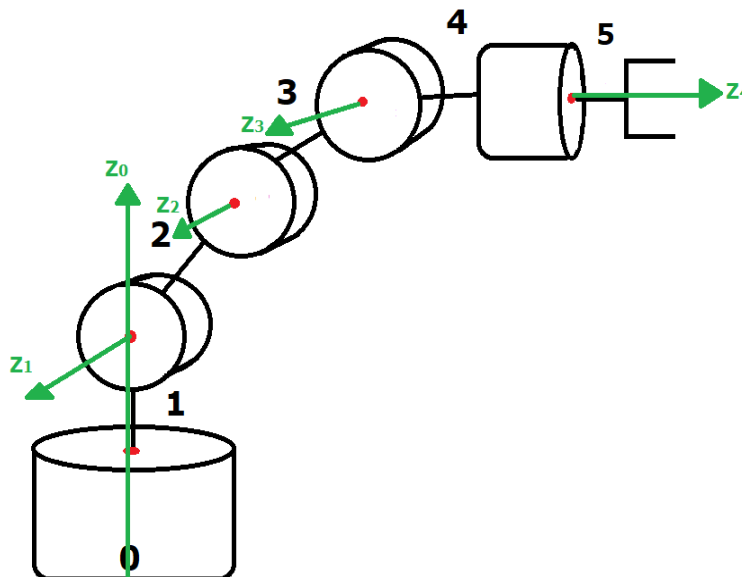


Figura 41 Asignación del eje 'z' en los ejes de cada articulación.

- Se coloca el origen del sistema de la base O_0 en cualquier punto del eje z_0 , como se visualiza en la figura 42. Los ejes x_0 y y_0 se sitúan de tal manera

que se forme un sistema dextrógiro junto con z_0 . Se puede hacer uso de la regla de la mano derecha para visualizar la dirección de los ejes, siendo el pulgar, el eje 'z'; el dedo índice, el eje 'x'; y el dedo medio, el eje 'y', tal y como se observa en la figura 43.

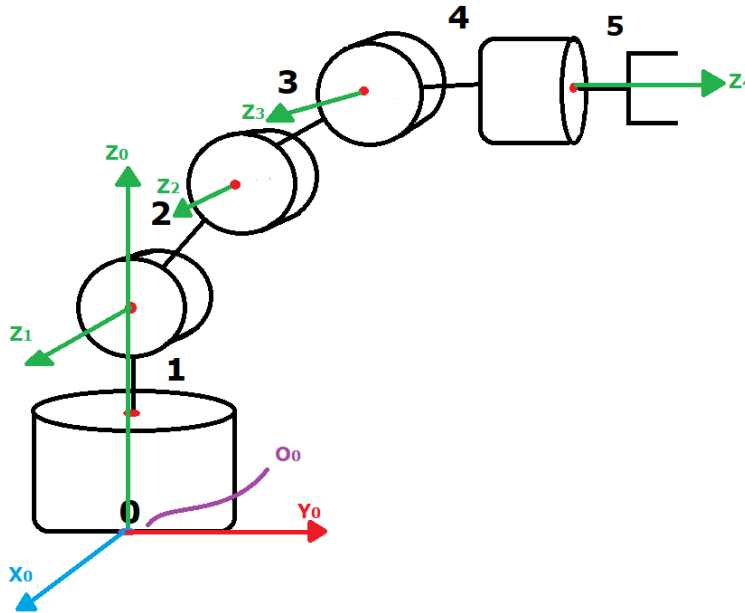


Figura 42 Asignación del origen del sistema cero y de los ejes 'x' 'y' y 'z' asociados a ese sistema, usando la regla de la mano derecha.

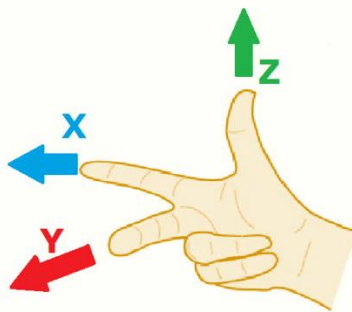


Figura 43 Regla de la mano derecha.

7. Para i que toma valores de 1 a $n-1$, se asigna el origen del sistema O_i , como se muestra en la figura 44. Se sitúa O_i en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i . Para el caso en el cual ambos ejes se corten, se coloca O_i en el punto de corte. Si ambos ejes son paralelos, O_i se pone en la articulación ' $i+1$ '.

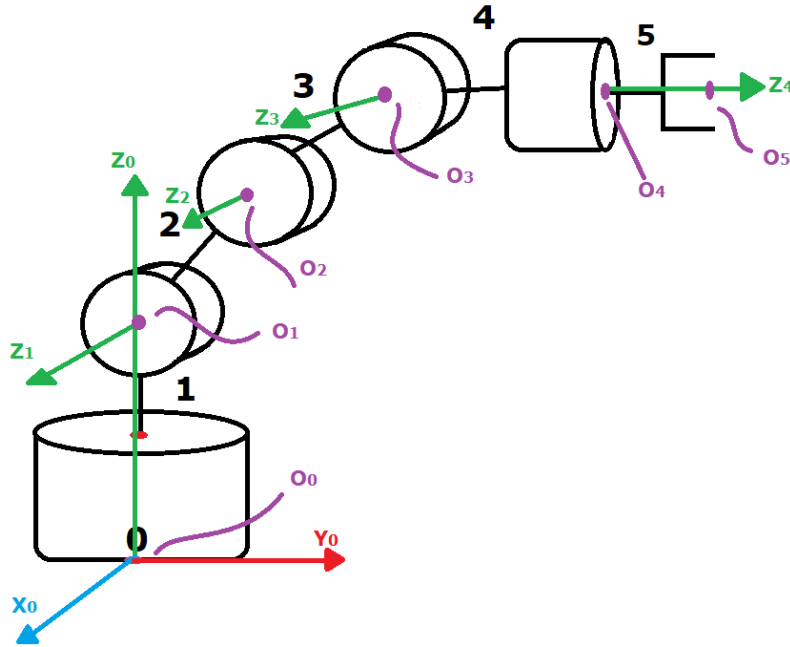


Figura 44 Asignación de los orígenes de cada sistema.

8. Para situar x_i se tiene en consideración los ejes z_{i-1} y z_i . Hay tres casos dependiendo si los ejes 'z' son paralelos, se intersecan, o ninguna de las dos anteriores. En general, se sitúa x_i en la línea normal común tanto a z_{i-1} como a z_i .
 - a) Si z_{i-1} y z_i son paralelos, existen varias posibilidades de vectores normales. Se escoge la dirección del eje x_i normal a z_i y z_{i-1} , considerando que este se dirija desde z_{i-1} a z_i .
 - b) Para el caso en que los ejes 'z' cortan en un punto. El eje x_i tiene dirección perpendicular al plano formado por los ejes 'z' y su origen se encuentra en la intersección de los ejes 'z'. Matemáticamente, se puede expresarse como la dirección de vector resultado del producto vectorial de dos vectores que tienen la misma dirección que z_i y z_{i-1} : $\vec{x}_i = \vec{z}_i \times \vec{z}_{i-1}$.
 - c) Para el tercer caso en que los ejes "z" no son paralelos ni se intersecan, la dirección del eje x_i está dada por la dirección de la normal común entre dichos ejes, esta se dirige del eje z_{i-1} al eje z_i .

La asignación de ejes para el esquema de robot LeArm queda como se muestra en la figura 45.

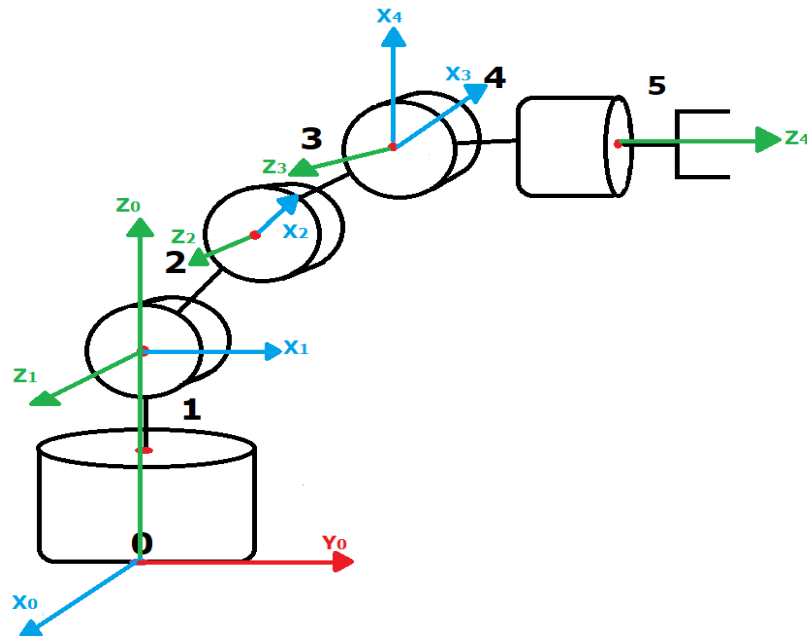


Figura 45 Asignación de cada eje 'x' en los sistemas de ejes.

9. Se sitúa y_i de tal manera que forme un sistema dextrógiro en conjunto con x_i y z_i , como se observa en la figura 46. Nuevamente, se puede apoyar en la regla de mano derecha.

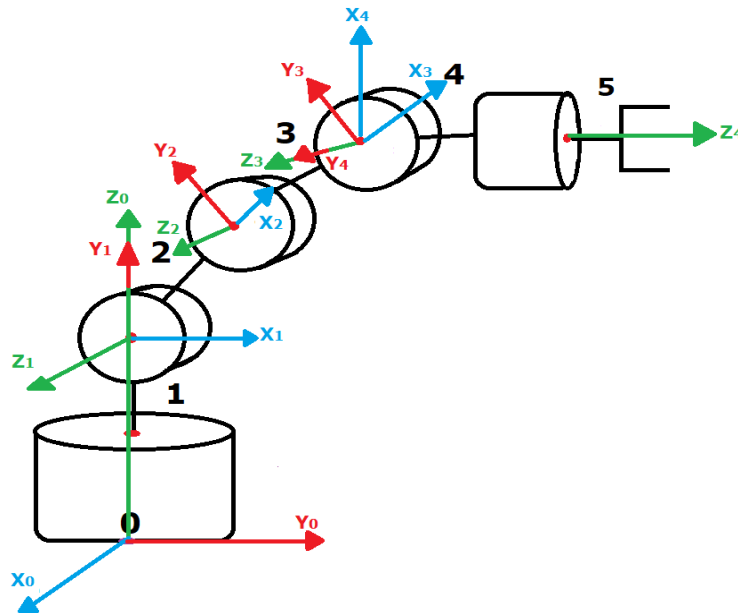


Figura 46 Asignación de los ejes 'y' en los sistemas de ejes.

10. Se sitúa el sistema O_n en el extremo del robot de tal manera que z_n coincida con la dirección de z_{n-1} y como segunda condición que x_n sea normal a z_{n-1} y z_n . Se coloca y_n de modo que se forme un sistema dextrógiro en conjunto con x_n y z_n . Es recomendable posicionar el eje y_n antes que x_n . Esto último

cuando el efector final es una pinza. En dado caso, el eje y_n se coloca en dirección de la apertura y cierre de la pinza y al final se sitúa x_n formando un sistema dextrógiro con y_n y z_n . El sistema O_n , en este caso O_5 , y el sistema de ejes x_n, y_n, z_n , que son equivalentes a x_5, y_5, z_5 , se visualizan en la figura 47.

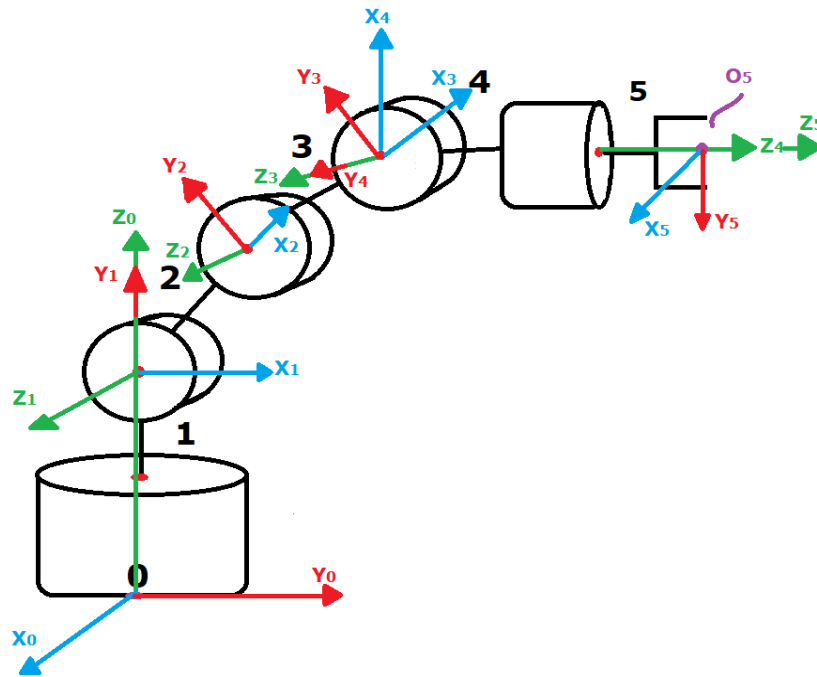


Figura 47 Asignación del origen del último sistema y sus respectivos ejes.

Con todos los sistemas de ejes asignados, se procede a obtener las variables articulares.

11. Se obtiene θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos. Se puede hacer uso de la regla de la mano derecha para poder visualizar el sentido de giro. Este parámetro θ_i es variable en las articulaciones rotatorias. Están indicados de color gris en forma de flechas que dan el sentido de giro de estos parámetros, como se muestra en la figura 48.

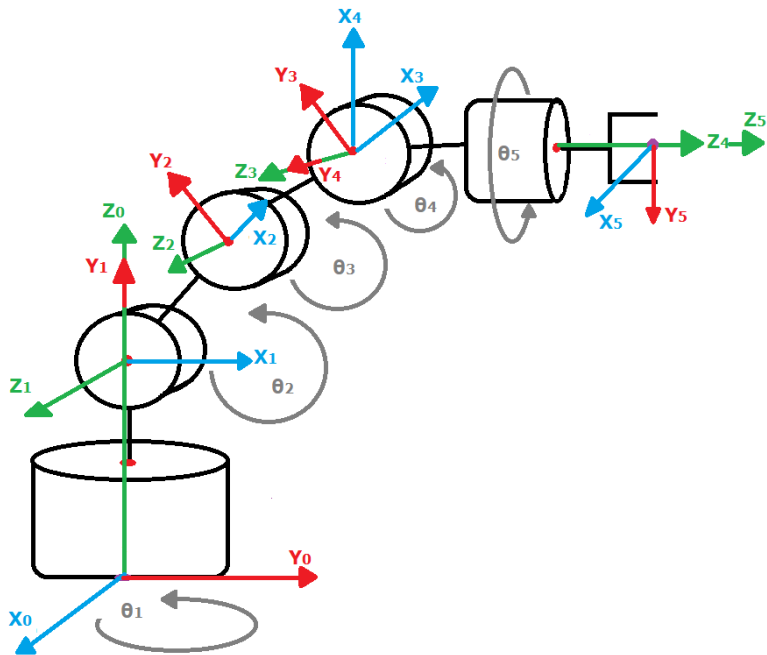


Figura 48 Obtención del parámetro ' θ '.

- Se obtiene ' d_i ' como la distancia, medida a lo largo de z_{i-1} , que hay que desplazar O_{i-1} para que coincida con O_i . Este parámetro se indica como una longitud marcada con color naranja, como se observa en la figura 49.

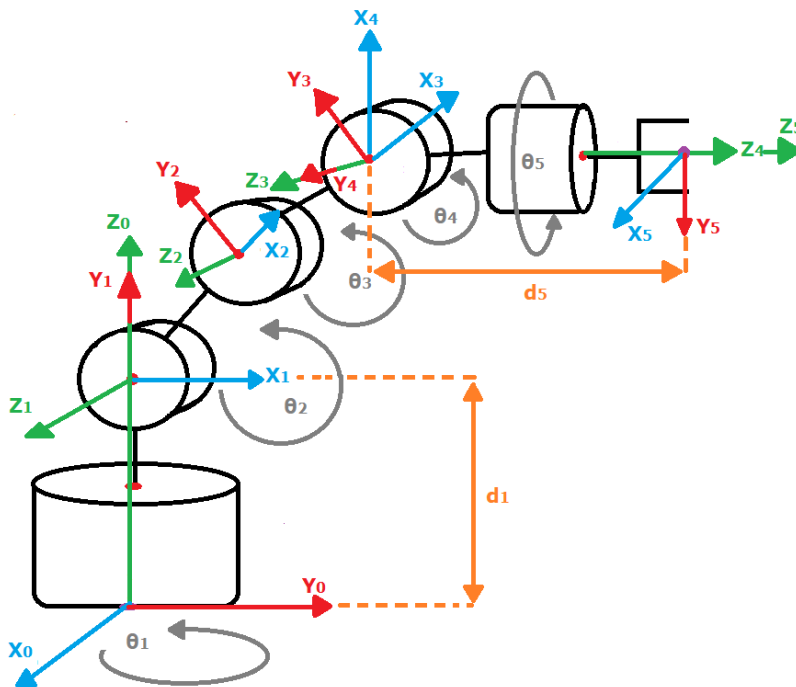


Figura 49 Obtención del parámetro ' d '.

13. Se obtiene a_i como la distancia medida a lo largo de x_i que hay que desplazar el O_{i-1} para que su origen coincida con O_i . Este parámetro está señalado como una longitud marcada con color rojo oscuro, como se observa en la figura 49.

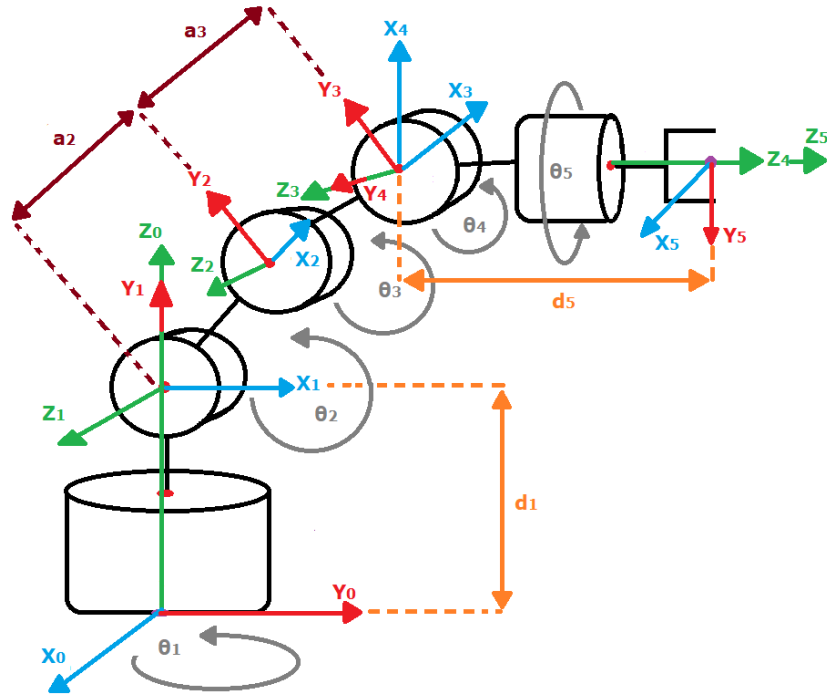


Figura 50 Obtención del parámetro 'a'.

14. Se obtiene a_i como el ángulo que hay que girar en torno a x_i , para que el eje z_{i-1} coincida con el eje z_i . Se visualizan con flechas de color morado el sentido de los parámetros como se indica en la figura 51.

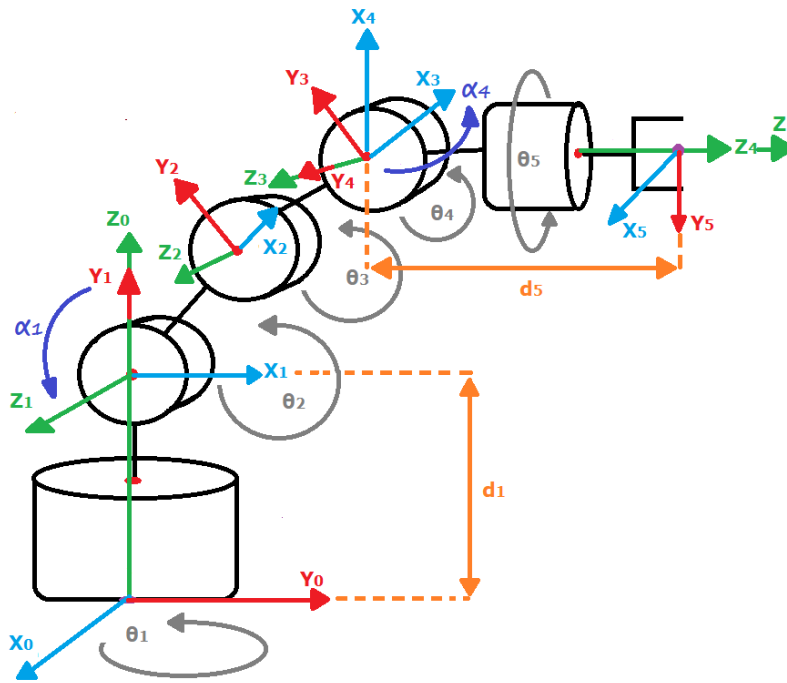


Figura 51 Obtención del parámetro 'α'.

Se realiza una tabla en la que se puedan visualizar las variables articulares asociadas a cada eslabón. Nótese en la tabla 4 que no todos los parámetros existen para cada eslabón

Tabla 4 Parámetros 'θ', 'd', 'a' y 'α', asociados a cada eslabón.

Eslabón (i)	d[cm]	a [cm]	θ[°]	α[°]
1	9.6	0	θ ₁	90
2	0	10.5	θ ₂	0
3	0	8.9	θ ₃	0
4	0	0	θ ₄	90
5	17.5	0	θ ₅	0

Obtenidos los parámetros, se obtiene las matrices de transformación: ${}^{i-1}_iH$. Se sustituyen los valores articulares de cada articulación usando como base la matriz homogénea general de la ecuación 9 para obtener las matrices 10,11,12, 13 y 14 asociadas a cada grado de libertad.

Las matrices obtenidas son:

$${}^{i-1}_iH = \begin{bmatrix} \cos \theta_i & -\cos \alpha_i * \sin \theta_i & \sin \alpha_i * \sin \theta_i & a_i * \cos \theta_i \\ \sin \theta_i & \cos \alpha_i * \cos \theta_i & -\sin \alpha_i * \cos \theta_i & a_i * \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$${}^0_1H = \begin{bmatrix} \cos \theta_1 & 0 & \sin \theta_1 & 0 \\ \sin \theta_1 & 0 & -\cos \theta_1 & 0 \\ 0 & 1 & 0 & 9.6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$${}^1_2H = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & 10.5 \cos \theta_2 \\ \sin \theta_2 & \cos \theta_2 & 0 & 10.5 \sin \theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

$${}^2_3H = \begin{bmatrix} \cos \theta_3 & -\sin \theta_3 & 0 & 8.9 \cos \theta_3 \\ \sin \theta_3 & \cos \theta_3 & 0 & 8.9 \sin \theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

$${}^3_4H = \begin{bmatrix} \cos \theta_4 & 0 & \sin \theta_4 & 0 \\ \sin \theta_4 & 0 & -\cos \theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$${}^4_5H = \begin{bmatrix} \cos \theta_5 & -\sin \theta_5 & 0 & 0 \\ \sin \theta_5 & \cos \theta_5 & 0 & 0 \\ 0 & 0 & 1 & 17.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

Posteriormente, se obtiene la matriz de transformación que relaciona el sistema de la base con el del extremo final del robot: $T = {}^0_1H * {}^1_2H * {}^2_3H * \dots * {}^{i-1}_iH$. Por tanto, T es la multiplicación de las matrices de transformación que se observan en las ecuaciones (9) a (14).

En la matriz T, queda definida la orientación y posición, submatriz de rotación y la submatriz de traslación, respectivamente, del sistema de referencia extremo referido al sistema de referencia de origen localizado en la base. Todo esto queda en función de las "n" coordenadas articulares.

Los cálculos para la obtención de la matriz de transformación homogénea final, ecuación (15), se llevaron a cabo mediante software matemático para determinar la multiplicación de las matrices indicadas en las ecuaciones (10) a (14).

Representación de la matriz de transformación homogénea general en la ecuación (15), a fin de representar cada elemento de la matriz porque no es posible representar toda la matriz con cada uno debido a su longitud.

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

Donde n_x , n_y , n_z , o_x , o_y , o_z , a_x , a_y y a_z , representan la orientación de la herramienta final respecto al eje de referencia de origen. p_x , p_y , p_z , representan la posición de la herramienta final respecto al sistema de ejes en la base del robot.

Se presentan las expresiones matemáticas para cada elemento de la matriz (15), en las ecuaciones (16), (17), (18), (19), (20), (21), (22), (23), (24), (25), (26) y (27).

$$P_x = (\cos(\theta_1) * (17.5 * \sin(\theta_2 + \theta_3 + \theta_4) + 8.9 * \cos(\theta_2 + \theta_3) + 10.5 * \cos(\theta_2))) \quad (16)$$

$$P_y = (\sin(\theta_1) * (17.5 * \sin(\theta_2 + \theta_3 + \theta_4) + 8.9 * \cos(\theta_2 + \theta_3) + 10.5 * \cos(\theta_2))) \quad (17)$$

$$P_z = (8.9 * \sin(\theta_2 + \theta_3)) - (17.5 * \cos(\theta_2 + \theta_3 + \theta_4)) + (10.5 * \sin(\theta_2)) + 9.6 \quad (18)$$

$$n_x = \sin(\theta_1) * \sin(\theta_5) + \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \cos(\theta_5) \quad (19)$$

$$n_y = \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5) * \sin(\theta_1) - \cos(\theta_1) * \sin(\theta_5) \quad (20)$$

$$n_z = \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_5) \quad (21)$$

$$\mathbf{o}_x = \cos(\theta_5) * \sin(\theta_1) - \cos(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) * \sin(\theta_5) \quad (22)$$

$$\mathbf{o}_y = -\cos(\theta_1) * \cos(\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) * \sin(\theta_5) \quad (23)$$

$$\mathbf{o}_z = -\sin(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_5) \quad (24)$$

$$\mathbf{a}_x = \sin(\theta_2 + \theta_3 + \theta_4) * \cos(\theta_1) \quad (25)$$

$$\mathbf{a}_y = \sin(\theta_2 + \theta_3 + \theta_4) * \sin(\theta_1) \quad (26)$$

$$\mathbf{a}_z = -\cos(\theta_2 + \theta_3 + \theta_4) \quad (27)$$

Con las expresiones algebraicas de cada elemento de la matriz (15), queda resuelto el problema cinemático directo.

Obtención de la cinemática inversa para el robot bajo estudio

La resolución del problema cinemático inverso se puede realizar independiente del análisis para la resolución de la cinemática directa, pero es recomendable resolver previamente esta última para obtener datos relevantes como las matrices de transformación homogénea. Las matrices de transformación homogénea son de importancia para la segunda parte del método de desacoplo cinemático. El desacoplo cinemático simplifica el análisis para robots con varios grados de libertad y evitar métodos más complejos. Si el robot bajo análisis tuviera al menos 3 grados de libertad, bastaría con utilizar el método geométrico sin importar si se hace o no el análisis cinemático directo.

El análisis cinemático inverso se lleva a cabo para el robot antropomórfico de cinco grados de libertad, LeArm.

De acuerdo con lo expuesto en el capítulo dos, es necesario encontrar el punto medio Este punto tiene representación como 'pc' y las coordenadas 'xc', 'yc' y 'zc': $p_c(x_c, y_c, z_c)$. Representan, respectivamente, al vector de posición de ese punto con respecto a sistema de origen en la base del robot.

Como se recordará, la cinemática inversa tiene como datos de entrada la posición y orientación de la herramienta final.

El punto medio es utilizado en el análisis geométrico del desacoplo cinemático como si se tratará de la posición final del robot, porque este contempla solo los tres primeros eslabones de la cadena cinemática. Al resolver por el método geométrico se encuentran las tres variables articulares 'q1', 'q2' y 'q3' que corresponden al movimiento que realizan las tres primeras articulaciones del robot.

Para encontrar el punto medio se hace un análisis de vectores de las posiciones en el espacio de la posición final de la herramienta, el punto medio y la posición de origen de la base del robot, como se muestra en el esquema de la figura 52. (Carvajal Rojas, 2006)

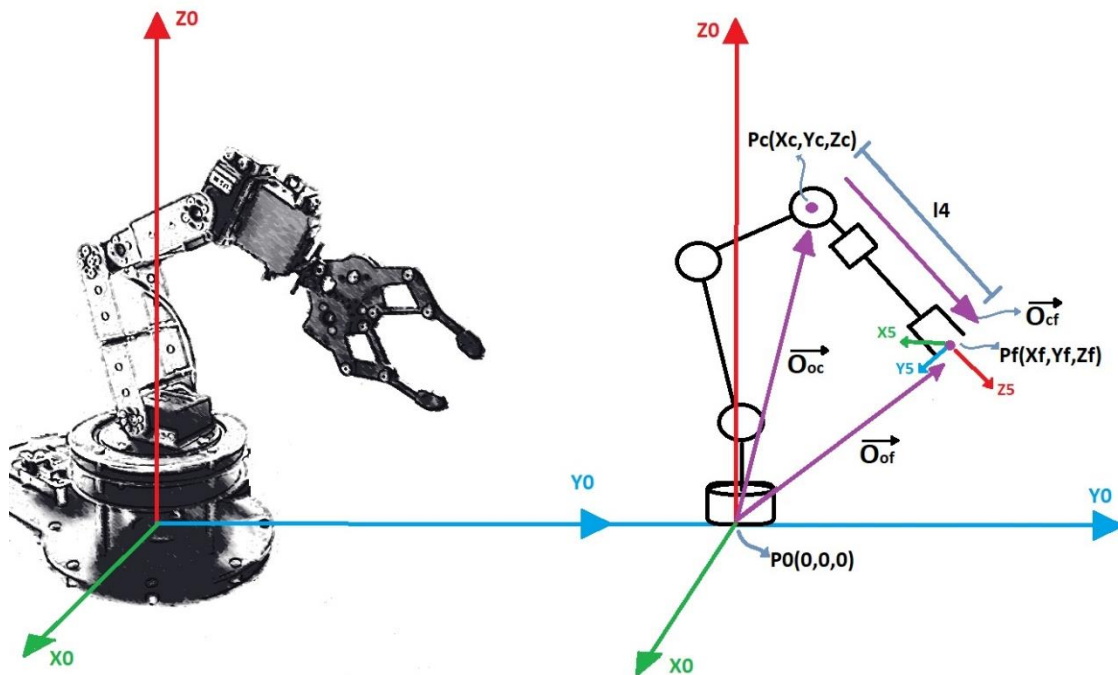


Figura 52 Esquema vectorial de las posiciones de origen, punto medio y punto final de la herramienta. ('Po', 'Pc' y 'Pf').

Por suma de vectores, se observa en la figura 53 que se cumple la ecuación (28).

$$\vec{O}_{oc} + \vec{O}_{cf} = \vec{O}_{of} \quad (28)$$

Y despejando de la Ecuación 28 el vector que representa a la posición del punto 'Pc', se obtiene la ecuación (29).

$$\vec{O}_{oc} = \vec{O}_{of} - \vec{O}_{cf} \quad (29)$$

En forma matricial, la ecuación (29) se expresa como se indica en la ecuación (30).

$$O_{oc} = O_{of} - O_{cf} \quad (30)$$

Donde 'Ocf' se puede expresar como la multiplicación de dos matrices, la matriz de rotación 'R06' y una matriz que representa al vector con magnitud 'l4' y dirección del eje 'Z5'. 'R06' es la submatriz de orientación que forma parte de la matriz de transformación homogénea final obtenida tras resolver la cinemática directa. Esta submatriz representa a la orientación del sistema de ejes de la herramienta final. 'R06' se multiplica por el vector que va sobre el eje 'z5' y que tiene magnitud 'l4' donde 'l4' es la longitud del eslabón de la muñeca del robot manipulador. 'Ocf' queda expresada en la ecuación (31).

$$O_{cf} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ l4 \end{bmatrix} = \begin{bmatrix} l4 * a_x \\ l4 * a_y \\ l4 * a_z \end{bmatrix} \quad (31)$$

Tras obtener 'Ocf' es posible expresar el vector que representa el punto medio que es representado en la ecuación (32).

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} X_f \\ Y_f \\ Z_f \end{bmatrix} - \begin{bmatrix} l4 * a_x \\ l4 * a_y \\ l4 * a_z \end{bmatrix} = \begin{bmatrix} X_f - l4 * a_x \\ Y_f - l4 * a_y \\ Z_f - l4 * a_z \end{bmatrix} \quad (32)$$

Resolución por el método geométrico

Las variables articulares q1,q2,q3 que se obtienen, tras aplicar el método geométrico, quedan expresadas en función de las coordenadas del punto medio (xc, yc ,zc).

A su vez, el punto 'pc' queda en función de las coordenadas de posición de la herramienta final.

Para implementar el método geométrico se recurre a un esquema del robot que permita ver las figuras geométricas que se forman por la posición de las articulaciones del manipulador, como se observa en la figura 53. Generalmente, las figuras que se forman son triángulos y, por tanto, es necesario hacer uso de funciones y propiedades trigonométricas.

De la figura 53 se obtiene la primera variable articular 'q1=θ1' que es el ángulo de un triángulo rectángulo en el que intervienen las coordenadas 'xc' y 'yc' y se obtiene la ecuación (33).

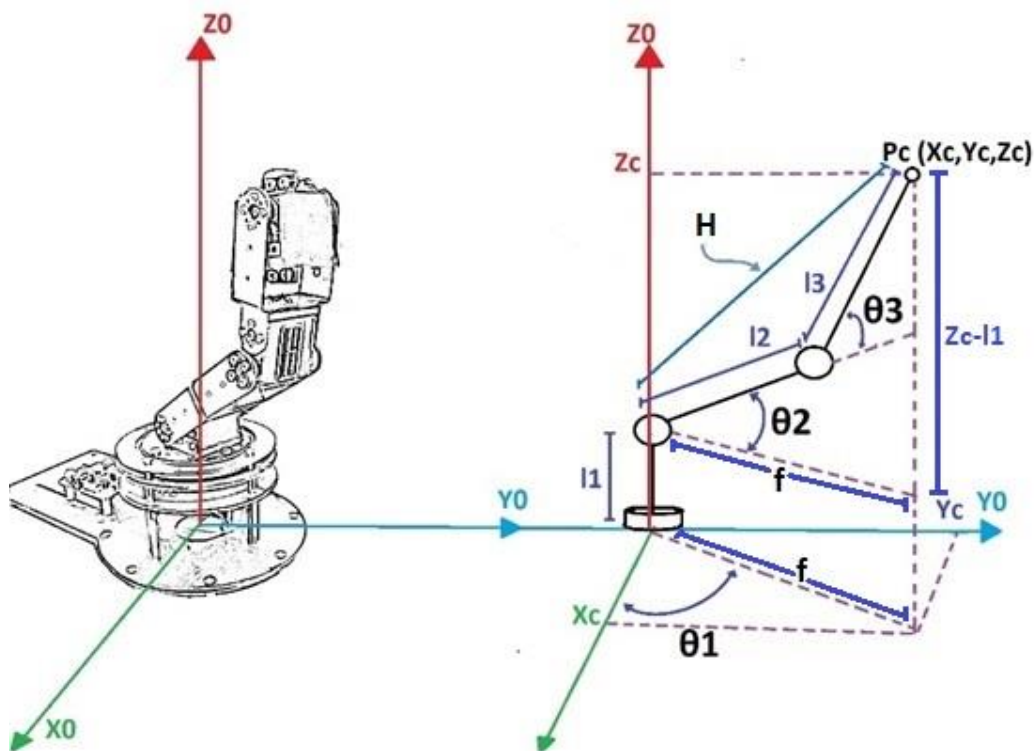


Figura 53 Resolución por el método geométrico para el robot LeArm.

$$q_1 = \theta_1 = \text{atan}\left(\frac{y_c}{x_c}\right) = \text{atan2}\left(\frac{y_c}{x_c}\right) \quad (33)$$

La variable articular 'q1' queda expresada en función de un arco tangente, pero se decidió obtener esta variable articular mediante la función 'atan2' que, a diferencia de la función "atan", considera el signo de los dos argumentos 'xc' y 'yc'. De esta manera, es posible dar como salida el ángulo correcto, ya que en la función 'atan' es imposible saber que signo tenían originalmente los argumentos en la división 'yc/xc'.

De la misma figura 53, se obtiene el valor de la tangente 'f' que está formada por los catetos 'xc' y 'yc'. Esta tangente tiene la misma longitud que un cateto que forma parte de otro triángulo junto al cateto de longitud 'zc-l1'. A las dos longitudes se les nombra indistintamente 'f' al tener la misma longitud según se aprecia en el esquema de la figura 53.

Proyectando las articulaciones y eslabones restantes sobre el plano 'ZY' se pueden obtener una vista distinta del esquema del robot. Esto ayuda a visualizar los eslabones 'l2' y 'l3' que estaban en una vista tridimensional a un plano, como se visualiza en la figura 54.

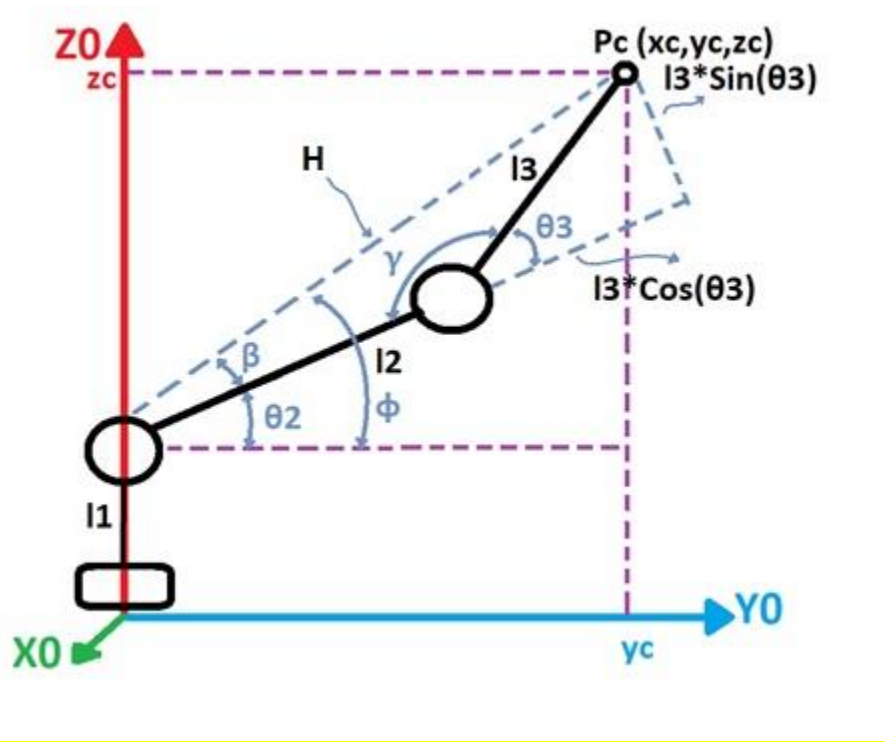


Figura 54 Obtención de las variables articulares 'q2' y 'q3' por el método geométrico.

De la figura 54 se obtiene la ecuación (34) que expresa ángulo 'β' como la resta de los ángulos 'Φ' y 'θ2'.

$$\theta_2 = \phi - \beta \tag{34}$$

También la resta de los ángulos 'θ3' y 180°(grados) ó 'π' da como resultado el ángulo 'γ' (gamma) como se expresa en la ecuación (35).

$$\gamma = \pi - \theta_3 \tag{35}$$

El triángulo formado por 'l2', 'l3', y 'H', mostrado en la figura 55 no es rectángulo, por lo que se debe usar la ley de cosenos expresada de forma general en la ecuación (36). (Swokowski, 2009)

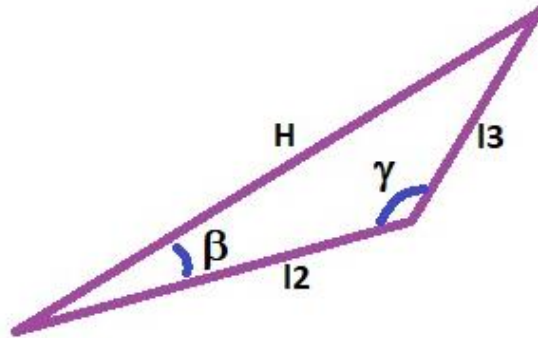


Figura 55 Triángulo obtuso formado por los lados 'l2', 'l3' y 'H'.

$$C^2 = A^2 + B^2 - 2 * A * B * \cos (\gamma) \quad (36)$$

El valor de 'H' se obtiene a partir de los triángulos observados en la figura (54), para después sustituirlo por 'C' de la ecuación (36) y obtener la ecuación (37).

$$C^2 = H^2 = f^2 + (z_c - l1)^2 = x_c^2 + y_c^2 + (z_c - l1)^2 \quad (37)$$

Los lados 'A' y 'B' para un triángulo generalizado de la ecuación (36) se igualan a los valores del triángulo de la figura 55, como se muestra en las ecuaciones (38) y (39).

$$A^2 = l_2^2 \quad (38)$$

$$B^2 = l_3^2 \quad (39)$$

Sustituyendo las ecuaciones (37), (38), (39) y (35) en la ecuación (36), se forma la ecuación (40).

$$x_c^2 + y_c^2 + (z_c - l_1)^2 = l_2^2 + l_3^2 - 2 * l_2 * l_3 * \cos (\pi - \theta_3) \quad (40)$$

Se usa la identidad trigonométrica expresada en la ecuación (41).

$$\cos(\pi - \theta) = -\cos(\theta) \quad (41)$$

Se reescribe la ecuación (40) usando la identidad de la ecuación (41) y se obtiene la ecuación (42).

$$x_c^2 + y_c^2 + (z_c - l_1)^2 = l_2^2 + l_3^2 + 2 * l_2 * l_3 * \cos(\theta_3) \quad (42)$$

Se despeja el coseno de 'θ3' de la ecuación (42) y se obtiene la ecuación (43).

$$\cos(q_3) = \cos(\theta_3) = \frac{x_c^2 + y_c^2 + (z_c - l_1)^2 - l_2^2 - l_3^2}{2 * l_2 * l_3} \quad (43)$$

Se sustituye 'θ3' por 'q3' para representar a 'q3' como una variable articular.

$$q_3 = \theta_3 \quad (44)$$

Aunque en la ecuación 43 es posible despejar la variable articular 'q3', no se puede expresar la solución de esta forma.

De acuerdo con la figura 56 que representa a la gráfica de la función coseno inverso, su dominio es $[-1, 1]$. En cambio, la función tangente inversa o arco tangente tiene un dominio en los números reales 'R' y su rango es $[-\pi, \pi]$. La función tangente inversa con su rango y dominio se observa en la figura 57. (Swokowski, 2009). Esta función representa mejor los valores de ángulo en que varían las articulaciones del robot LeArm.



Figura 56 Función coseno inverso.

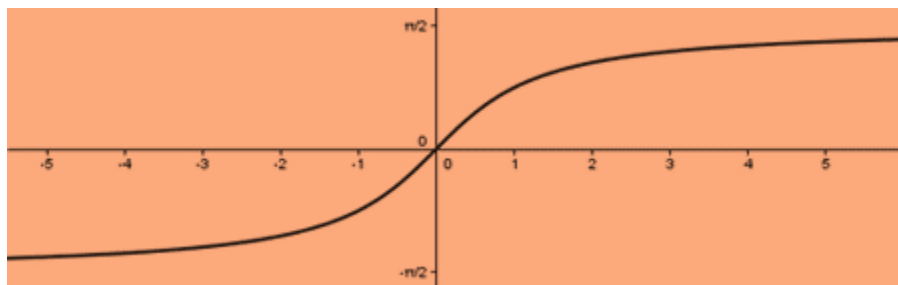


Figura 57 Función tangente inversa o arco tangente.

Se tiene la expresión 'cos(q3)' y se requiere tener 'tan (q3)' para obtener el arco tangente 'atan(q3)'. Se realizan lo siguiente para obtener una expresión en términos de la cotangente.

Se usa la identidad trigonométrica de la ecuación (45).

$$\sin^2 \theta + \cos^2 \theta = 1 \quad (45)$$

Despejando 'sin q3' en la ecuación (45) y se obtiene la ecuación 46.

$$\sin q_3 = \pm \sqrt{1 - \cos^2 q_3} \quad (46)$$

A partir de la ecuación (43), 'cos (q3)' queda representado como la variable 'D' y ahora queda expresada como se observa en la ecuación (47).

$$D = \cos(q_3) = \frac{x_c^2 + y_c^2 + (z_c - l_1)^2 - l_1^2 - l_2^2}{2 * l_1 * l_2} \quad (47)$$

Se simplifican la ecuación (46) a partir de la ecuación (47) y se tiene la ecuación (48).

$$\sin q_3 = \pm \sqrt{1 - D^2} \quad (48)$$

La expresión de la tangente es como se indica en la ecuación (49).

$$\tan \theta = \frac{\sin \theta}{\cos \theta} \quad (49)$$

'Tan q3' queda en términos de 'cos q3' o de la variable 'D' usando las ecuaciones (47) y (48) para obtener la ecuación (50).

$$\tan q_3 = \left(\pm \frac{\sqrt{1 - \cos^2 q_3}}{\cos q_3} \right) = \frac{\pm \sqrt{1 - D^2}}{D} \quad (50)$$

Y despejando 'q3' se obtiene la ecuación (51).

$$q_3 = \text{atan2} \left(\pm \frac{\sqrt{1 - D^2}}{D} \right) \quad (51)$$

Para resolver la variable articular 'q2' se usa la ecuación (34) reemplazando 'θ2' y se obtiene la ecuación (52).

$$q_2 = \phi - \beta \quad (52)$$

Tomando en cuenta la figura 54 se forman dos triángulos rectángulos, que tienen a 'Φ' y 'β' como uno de sus ángulos, respectivamente. Se muestra en figura 58, individualmente los dos triángulos que se forman.

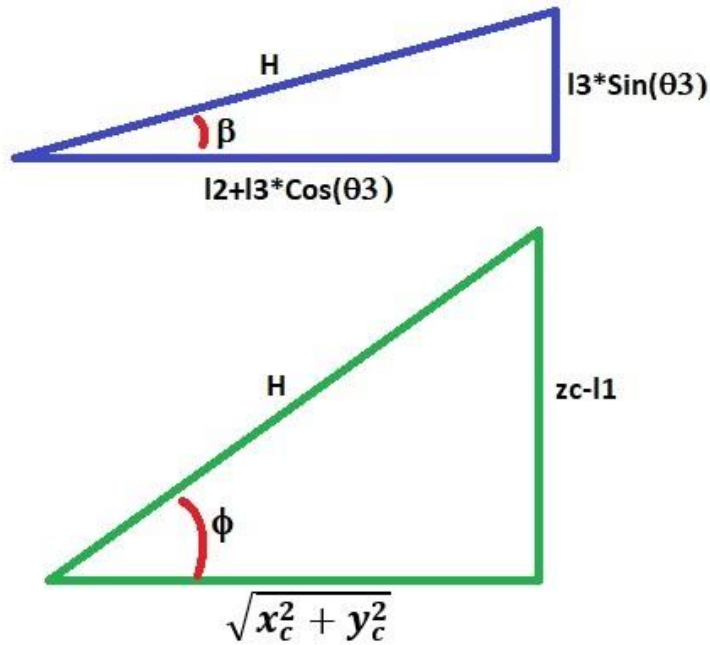


Figura 58 Triángulos rectángulos que tienen a uno de sus ángulos a ' β ' y ' ϕ ', respectivamente.

Para cada ángulo (' ϕ ' y ' β ') se expresa en términos de la tangente como se observa en las ecuaciones (53) y (55). Después se despeja el ángulo y queda en términos de la tangente inversa como se indica en las ecuaciones (54) y (56).

$$\tan \phi = \left(\frac{z_c - l1}{\pm \sqrt{z_c^2 + y_c^2}} \right) \quad (53)$$

$$\phi = \text{atan2} \left(\frac{z_c - l1}{\pm \sqrt{z_c^2 + y_c^2}} \right) \quad (54)$$

$$\tan \beta = \left(\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3} \right) \quad (55)$$

$$\beta = \text{atan2} \left(\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3} \right) \quad (56)$$

Sustituyendo las ecuaciones (54) y (56) en la ecuación (52) se obtiene la ecuación (57).

$$q_2 = \operatorname{atan2}\left(\frac{z_c - l_1}{\pm\sqrt{z_c^2 + y_c^2}}\right) - \operatorname{atan2}\left(\frac{l_3 * \sin q_3}{l_2 + l_3 * \cos q_3}\right) \quad (57)$$

Se han obtenido las tres primeras variables articulares 'q1', 'q2', 'q3' por el método geométrico expresadas en las ecuaciones (33), (51) y (57).

Obtención de las variables articulares de la muñeca del robot.

Para la resolución de las dos variables articulares restantes se hace uso de la matriz de transformación homogénea obtenida en la cinemática directa. En forma matricial y simbólica se representa en la ecuación (58). (Carvajal Rojas, 2006)

$${}^0_5H = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (58)$$

Donde los términos $n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z$ son los términos de la submatriz de rotación del sistema de referencia 0 al sistema de referencia 5. Quedando la submatriz expresada en la ecuación (59).

$${}^0_5R = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \quad (59)$$

La matriz de la ecuación (59) se puede expresar también como la multiplicación de dos matrices. Una matriz que contempla la rotación para los tres primeros grados de libertad y otra matriz que represente la rotación para los dos últimos grados de libertad como se expresa en la ecuación (60).

$${}^0_5R = {}^0_3R * {}^3_5R \quad (60)$$

Despejando y aplicando propiedades de matrices ortonormales, se puede expresar una matriz inversa como la matriz transpuesta y se obtiene expresión la matriz de rotación para los grados de libertad 3 a 5 como se indica en la ecuación (61).

$${}^3_5R = {}^0_3R^T * {}^0_5R \quad (61)$$

Donde la matriz 3_5R de la ecuación (61) queda expresada en función de las variables articulares que aún se desconocen, ' θ_4 ' y ' θ_5 '. La matriz 3_5R se representa como la multiplicación de las matrices de orientación 3_4R y 4_5R cómo se observa en la ecuación (62).

En la ecuación (63) se compara cada uno de sus elementos (una vez realizada y simplificada la multiplicación de la ecuación (62) simbólicamente con un elemento 'Bij' de otra matriz de igual tamaño (3x3). Esto con el fin de ubicar cada elemento de la matriz.

$${}^3_5R = {}^3_4R * {}^4_5R = \begin{bmatrix} \cos \theta_4 & 0 & \sin \theta_4 \\ \sin \theta_4 & 0 & -\cos \theta_4 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} \cos \theta_5 & -\sin \theta_5 & 0 \\ \sin \theta_5 & \cos \theta_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (62)$$

$${}^3_5R = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{pmatrix} = \begin{pmatrix} \cos \theta_4 * \cos \theta_5 & -\cos \theta_4 * \sin \theta_5 & \sin \theta_4 \\ \cos \theta_5 * \sin \theta_4 & -\sin \theta_4 * \sin \theta_5 & -\cos \theta_4 \\ \sin \theta_5 & \cos \theta_5 & 0 \end{pmatrix} \quad (63)$$

Del lado derecho de la igualdad de la ecuación (61), expresamos la multiplicación de matrices ${}^0_3R^T * {}^0_5R$, como se muestra en la ecuación (64).

$${}^0_3R^T * {}^0_5R = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad (64)$$

Donde ${}^0_3R^T$ es la transpuesta de 0_3R expresado simbólicamente en la ecuación 65.

$${}^0_3R = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \quad (65)$$

La ecuación (65) representada como la multiplicación de las submatrices obtenidas de las matrices de transformación homogénea, expresadas en las ecuaciones (10), (11), y (12). Se expresa en la ecuación (66).

$${}^0_3R = \begin{pmatrix} \cos \theta_1 & 0 & \sin \theta_1 \\ \sin \theta_1 & 0 & -\cos \theta_1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \cos \theta_2 & -\sin \theta_2 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta_3 & -\sin \theta_3 & 0 \\ \sin \theta_3 & \cos \theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (66)$$

De la matriz de la ecuación 65 queda expresada su matriz transpuesta en la ecuación 67.

$${}^0_3R^T = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \quad (67)$$

Los elementos de la matriz (65) desarrollados matemáticamente tras realizar las multiplicaciones de la ecuación (66) se muestran en las ecuaciones (68) a la (76).

Se observa en la ecuación 67 que los elementos de su matriz son iguales a los de la ecuación 65, solo que estos elementos cambian de posición.

$$A_{11} = \cos(q1) * (\cos(q2) * \cos(q3) - \sin(q2) * \sin(q3)) \quad (68)$$

$$A_{12} = -\cos(q1) * (\cos(q3) * \sin(q2) + \cos(q2) * \sin(q3)) \quad (69)$$

$$A_{13} = \sin(q1) \quad (70)$$

$$A_{21} = \sin(q1) * (\cos(q2) * \cos(q3) - \sin(q2) * \sin(q3)) \quad (71)$$

$$A_{22} = -\sin(q1) * (\cos(q3) * \sin(q2) + \cos(q2) * \sin(q3)) \quad (72)$$

$$A_{23} = -\cos(q1) \quad (73)$$

$$A_{31} = \cos(q3) * \sin(q2) + \cos(q2) * \sin(q3) \quad (74)$$

$$A_{32} = \cos(q2) * \cos(q3) - \sin(q2) * \sin(q3) \quad (75)$$

$$A_{33} = 0 \quad (76)$$

La matriz 0_5R es la orientación de la herramienta final, que debe ser conocida al ser datos de entrada para la cinemática inversa. Como los elemento de esta matriz son datos conocidos, pero su valor depende de cada caso particular, se puede representar la matriz como:

$${}^0_5R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \quad (77)$$

La matriz resultado de la multiplicación expresada por ${}^0_3R^T * {}^0_5R$ se expone en la ecuación (78).

$${}^0_3R^T * {}^0_5R = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad (78)$$

Los elementos resultado de la multiplicación de la ecuación (78) se presentan en las ecuaciones (79) a (87).

$$a = R_{31} * \sin(q_2 + q_3) + R_{11} * \cos(q_2 + q_3) * \cos(q_1) + R_{21} * \cos(q_2 + q_3) * \sin(q_1) \quad (79)$$

$$b = R_{32} * \sin(q_2 + q_3) + R_{12} * \cos(q_2 + q_3) * \cos(q_1) + R_{22} * \cos(q_2 + q_3) * \sin(q_1) \quad (80)$$

$$c = R_{33} * \sin(q_2 + q_3) + R_{13} * \cos(q_2 + q_3) * \cos(q_1) + \dots \dots + R_{23} * \cos(q_2 + q_3) * \sin(q_1) \quad (81)$$

:

$$d = R_{31} * \cos(q_2 + q_3) - R_{11} * \sin(q_2 + q_3) * \cos(q_1) - R_{21} * \sin(q_2 + q_3) * \sin(q_1) \quad (82)$$

$$e = R_{32} * \cos(q_2 + q_3) - R_{12} * \sin(q_2 + q_3) * \cos(q_1) - R_{22} * \sin(q_2 + q_3) * \sin(q_1) \quad (83)$$

$$f = R_{33} * \cos(q_2 + q_3) - R_{13} * \sin(q_2 + q_3) * \cos(q_1) - \dots \dots - R_{23} * \sin(q_2 + q_3) * \sin(q_1) \quad (84)$$

$$g = R_{11} * \sin(q_1) - R_{21} * \cos(q_1) \quad (85)$$

$$h = R_{12} * \sin(q_1) - R_{22} * \cos(q_1) \quad (86)$$

$$i = R_{13} * \sin(q_1) - R_{23} * \cos(q_1) \quad (87)$$

De la matriz en la ecuación (63) se usan los elementos 'B13', 'B23', 'B31' y 'B32' para poder hacer una igualdad elemento-elemento con los de la matriz de la ecuación (78), 'c', 'f', 'g' y 'h'. Las igualdades se muestran en las ecuaciones (88), (89), (90) y (91).

$$\begin{aligned} B_{13} = c \\ \sin \theta_4 = R_{33} * \sin(q_2 + q_3) + R_{13} * \cos(q_2 + q_3) * \cos(q_1) + \dots \\ \dots + R_{23} * \cos(q_2 + q_3) * \sin(q_1) \end{aligned} \quad (88)$$

$$\begin{aligned} B_{23} = f \\ -\cos \theta_4 = R_{33} * \cos(q_2 + q_3) - R_{13} * \sin(q_2 + q_3) * \cos(q_1) - \dots \\ \dots - R_{23} * \sin(q_2 + q_3) * \sin(q_1) \end{aligned} \quad (89)$$

$$\begin{aligned} B_{31} = g \\ \sin \theta_5 = R_{11} * \sin(q_1) - R_{21} * \cos(q_1) \end{aligned} \quad (90)$$

$$\begin{aligned} B_{32} = h \\ \cos \theta_5 = R_{12} * \sin(q_1) - R_{22} * \cos(q_1) \end{aligned} \quad (91)$$

Al usar las ecuaciones (88), (89) y (90), (91) para formar tangente para "q4" y "q5", se expresan como en su forma tangencial, como se indica en las ecuaciones (92) y (93).

$$\tan q_4 = \frac{\sin q_4}{\cos q_4} \quad (92)$$

$$\tan q_5 = \frac{\sin q_5}{\cos q_5} \quad (93)$$

Se despeja 'q4' y 'q5' en las ecuaciones (92) y (93) para obtener las funciones arco tangente, respectivas. Posteriormente, se desarrolla con las expresiones de las ecuaciones (88), (89), (90) y (91) para obtener las ecuaciones (94) y (95) que representan a las variables articulares restantes, 'q4' y 'q5'.

$$q_4 = \text{atan2} \left(\frac{R_{33} * \sin(q_2 + q_3) + R_{13} * \cos(q_2 + q_3) * \cos(q_1) + R_{23} * \cos(q_2 + q_3) * \sin(q_1)}{R_{13} * \sin(q_2 + q_3) * \cos(q_1) + R_{23} * \sin(q_2 + q_3) * \sin(q_1) - R_{33} * \cos(q_2 + q_3)} \right) \quad (94)$$

$$q_5 = \text{atan2} \left(\frac{R_{11} * \sin(q_1) - R_{21} * \cos(q_1)}{R_{12} * \sin(q_1) - R_{22} * \cos(q_1)} \right) \quad (95)$$

Se ha encontrado las ecuaciones de la solución de la cinemática inversa al obtener las expresiones para las 5 variables articulares asociadas a los cinco grados de libertad del robot. Sin embargo, para la cinemática inversa no hay una única solución. Existen diversas combinaciones de las variables articulares que pueden posicionar y orientar, de una manera determinada, a la herramienta final del robot.

Esas soluciones se dan por los signos “más/menos” (+/-) que tienen algunas de las expresiones encontradas.

Hay otras cuestiones a considerar. Para la primera variable articular se tiene una única solución. Sin embargo, se toma en cuenta una segunda que no es señalada por la expresión matemática, sino por los resultados examinados tras la aplicación de la cinemática inversa en un software gráfico. Los resultados del programa se muestran en el capítulo.

Debido al rango de la función ‘atan2’ que va de $[-\pi, \pi]$, puede arrojar valores con signo negativo que no están en el rango de operación física de los servomotores, que va de $[0, 180]$ grados. Para arreglarlo se le suma al resultado 180° grados.

Las posibles soluciones para la primera variable articular son las mostradas en las ecuaciones (96) y (97).

$$q_{11} = \text{atan2} \left(\frac{y_c}{x_c} \right) \quad (96)$$

$$q_{12} = \text{atan2} \left(\frac{y_c}{x_c} \right) + 180 \quad (97)$$

Para la segunda variable articular hay cuatro posibles soluciones, de acuerdo con la ecuación (57). La primera parte de la solución tiene una raíz cuadrada con dos signos, lo que da dos soluciones. La segunda parte de la solución está en función de ‘sin (q3)’ que de acuerdo a la ecuación (48), se tiene dos soluciones por la raíz cuadrada que cuenta con dos signos. De esta manera se tienen cuatro posibles soluciones para la segunda variable articular al tener en cuenta las probables combinaciones. Las posibles soluciones a la segunda variable articular se muestran en las ecuaciones (98), (99), (100) y (101).

$$q_{21} = \text{atan2}\left(\frac{z_c - l1}{\sqrt{z_c^2 + y_c^2}}\right) - \text{atan2}\left(\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3}\right) \quad (98)$$

$$q_{22} = \text{atan2}\left(\frac{z_c - l1}{\sqrt{z_c^2 + y_c^2}}\right) - \text{atan2}\left(-\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3}\right) \quad (99)$$

$$q_{23} = \text{atan2}\left(-\frac{z_c - l1}{\sqrt{z_c^2 + y_c^2}}\right) - \text{atan2}\left(\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3}\right) \quad (100)$$

$$q_{24} = \text{atan2}\left(-\frac{z_c - l1}{\sqrt{z_c^2 + y_c^2}}\right) - \text{atan2}\left(-\frac{l3 * \sin q_3}{l2 + l3 * \cos q_3}\right) \quad (101)$$

Para la tercera variable hay dos posibles dado el doble signo en la raíz cuadrada como se muestra en la ecuación (51). Por lo tanto, las dos posibles soluciones son:

$$q_{31} = \text{atan2}\left(\frac{\sqrt{1 - D^2}}{D}\right) \quad (102)$$

$$q_{32} = \text{atan2}\left(-\frac{\sqrt{1 - D^2}}{D}\right) \quad (103)$$

Recordando que $D = \frac{x_c^2 + y_c^2 + (z_c - l_1)^2 - l_1^2 - l_2^2}{2 * l_1 * l_2}$.

Para las variables articulares 'q4' y 'q5' no hay signos que puedan indicar que hay más de una solución. Simplemente, están en función de las tres primeras variables articulares 'q1', 'q2' y 'q3' y son a causa de estas últimas que se tendrán varias posibles soluciones. Simplemente, se dejan expresadas tal cual se obtuvieron y se representan en las ecuaciones (94) y (95).

Además, se debe considerar que las posibles soluciones para las variables articulares 'q1', 'q2', 'q3', no todas son posibles físicamente para los rangos de operación de los servomotores del robot.

Ángulos de Euler

Al obtener la matriz de transformación homogénea, dentro de la misma hay una submatriz con 9 elementos que representa la rotación del sistema de la herramienta final respecto al sistema de origen.

Hay una alternativa a la matriz de orientación que, en lugar de usar nueve elementos para representar la orientación, solamente usa tres. Se trata de los ángulos de Euler.

Antes de explicar los ángulos de Euler, se muestra cómo funciona la matriz de orientación para representar la orientación. Con ello se puede hacer una comparativa entre las dos alternativas y así poder interpretar los datos de la matriz de orientación.

La matriz de orientación puede representarse como se ha mostrado con anterioridad, pero, además independiente del valor que tome cada elemento de la matriz, hay detrás una operación de vectores que se llama producto punto y se representa en la ecuación (104). (Craig, 2006)

$${}^0_N R = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix} = \begin{pmatrix} \widehat{l}_x \cdot \widehat{l}_u & \widehat{l}_x \cdot \widehat{j}_v & \widehat{l}_x \cdot \widehat{k}_w \\ \widehat{j}_y \cdot \widehat{l}_u & \widehat{j}_y \cdot \widehat{j}_v & \widehat{j}_y \cdot \widehat{k}_w \\ \widehat{k}_z \cdot \widehat{l}_u & \widehat{k}_z \cdot \widehat{j}_v & \widehat{k}_z \cdot \widehat{k}_w \end{pmatrix} \quad (104)$$

Como puede observarse, cada elemento de la matriz es un producto punto de vectores unitarios. Esto sin importar los N grados de libertad que tenga el robot examinado.

El producto punto puede representarse como se indica en la ecuación (105). (Solís, 2006)

$$\vec{A} \cdot \vec{B} = |A||B| \cos \theta \quad (105)$$

Donde \vec{A} y \vec{B} son dos vectores y $|A|$ y $|B|$ sus respectivas normas. ' θ ' representa el ángulo entre los dos vectores en cuestión.

Al tratarse de vectores unitarios, la norma de los vectores se reduce a '1' y la ecuación (105) queda expresada como se observa en la ecuación (106).

$$\vec{A} \cdot \vec{B} = \cos \theta \quad (106)$$

Se observa que el producto punto de dos vectores unitarios está determinado por el ángulo entre los dos vectores. Por ejemplo, cuando los ángulos son ortonormales, el ángulo entre ellos es 90° grados y, por tanto, el producto punto es igual a '1'. Otro ejemplo es cuando los vectores son paralelos y en consecuencia el ángulo entre ellos es 0° grados, el producto punto es igual a '0'.

Para mostrar que representan el producto punto de dos vectores unitarios, al analizar la orientación en robots manipuladores, en la figura 59 donde se observan dos sistemas de ejes. El sistema "x, y, z" representa al sistema de origen y el sistema "x', y', z'" representa al sistema de ejes de la herramienta final. Los ángulos " α , β , γ " son los ángulos que se encuentran girados, respectivamente, cada eje homólogo de los dos sistemas.

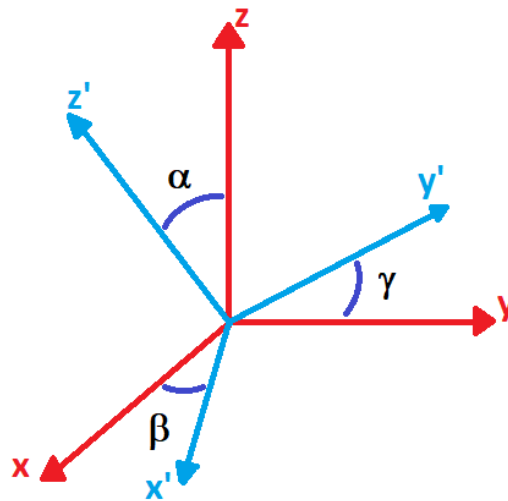


Figura 59 Ángulos de Euler para realizar la transformación de un sistema de coordenadas a otro.

Teniendo en cuenta los sistemas de ejes mostrados en la figura 59, es posible mostrar la representación de los ejes unitarios de la matriz de la ecuación (104).

Como se muestra en la figura 60, cada vector unitario representa un vector en dirección de alguno de los ejes de los dos sistemas de ejes.

Por tanto, la matriz de la ecuación (104) muestra a las combinaciones de producto punto para cada vector unitario. Los nueve elementos de la matriz son la comparación del ángulo entre vectores de cada sistema de coordenadas. Por ejemplo, el producto punto $\hat{i}_x \cdot \hat{i}_u$ compara del ángulo el eje "x'" respecto al eje "x". El producto punto de $\hat{i}_x \cdot \hat{k}_w$ compara el ángulo del eje "z'" respecto al eje "x", así sucesivamente con cada uno de los elementos de la matriz.

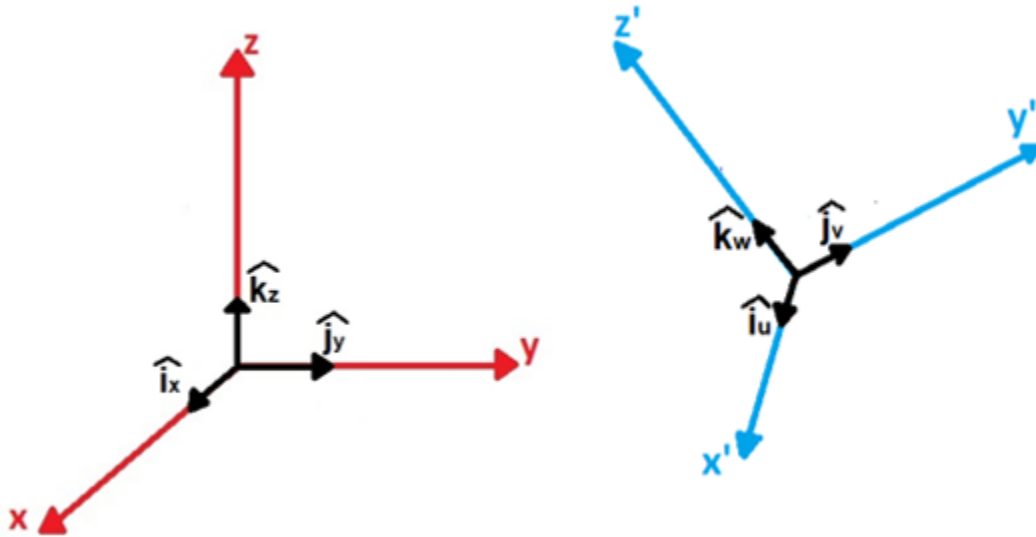


Figura 60 Ejes unitarios correspondientes a los ejes de cada sistema comparado.

Es complicado interpretar la orientación de esta manera (con la matriz de orientación) y se dificulta cuando se tienen valores que son decimales. No es de ninguna manera intuitivo. De ahí que los ángulos de Euler representen una alternativa para representar la orientación de un objeto en el espacio con solo tres valores.

Los ángulos de Euler son un conjunto de tres coordenadas angulares que permiten conocer la orientación de un sistema de ejes, que rota en el espacio tridimensional, respecto a otro sistema de ejes fijo. Esto es similar a lo que sucede con el sistema de origen de un robot y el otro sistema que pertenece a la herramienta final de un robot manipulador.

Las tres coordenadas angulares representan, cada una, una rotación del sistema móvil, dada por cierto ángulo y respecto a alguno de los ejes del sistema fijo. Cada una de las transformaciones o rotaciones se da respecto al último sistema girado.

Para observar cómo operan los ángulos de Euler se da el ejemplo siguiente.

Se tiene un eje de coordenadas fijo con los ejes “xyz” y se tiene una transformación de rotación dada por tres ángulos de Euler. El primer ángulo de Euler es un ángulo ‘ α ’ respecto al eje ‘z’. El segundo ángulo de Euler es un ángulo ‘ β ’ que es aplicado sobre el eje “y” del sistema que se giró con anterioridad, al cual se nombra como “y*”. El tercer ángulo de Euler es un ángulo “ γ ” que se gira sobre el eje “x” del sistema girado con anterioridad al que se designa como “x**”. Al realizar las rotaciones con ángulos de Euler se obtiene un nuevo sistema designado por los ejes “x***, y***, z***”. Se visualiza este ejemplo en la figura 61. Se considera que los tres ángulos de Euler tengan 90° grados cada uno para poder facilitar la observación de los giros en cada transformación.

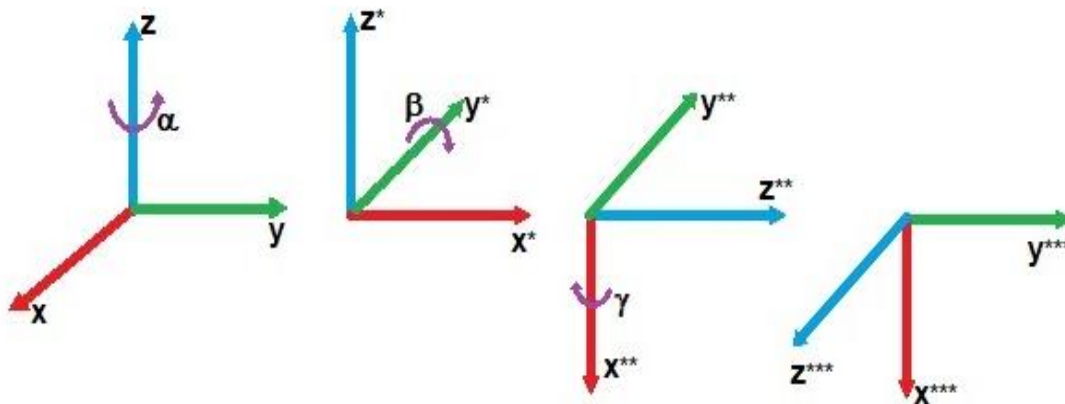


Figura 61 Ejemplo de una transformación con ángulos de Euler ‘ α ’, ‘ β ’ y ‘ γ ’ con valores de 90° grados cada uno.

Como se puede observar, hay tres transformaciones básicas. Una transformación de un ángulo ‘ α ’ respecto al eje ‘z’, una segunda transformación de un ángulo ‘ β ’ respecto al eje ‘y’, y una tercera transformación de un ángulo ‘ γ ’ respecto al eje ‘x’.

Estas transformaciones con ángulos de Euler se pueden expresar de forma matricial. Para la transformación de rotación de un ángulo ‘ α ’ respecto al eje ‘z’, se representa como se indica en la ecuación (107). (Craig, 2006)

$$RotZ(\alpha) \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (107)$$

La transformación de rotación de un ángulo ‘ β ’ respecto al eje ‘Y’, se representa como se observa en la ecuación (108).

$$RotY(\beta) \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \quad (108)$$

La transformación de rotación de un ángulo ‘ γ ’ respecto al eje x, se representa como se visualiza en la ecuación (109).

$$RotX(\gamma) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix} \quad (109)$$

Estas tres transformaciones fundamentales dan paso a que se puedan usar en cualquier combinación posible. Por ejemplo, se puede hacer una primera

transformación con respecto a 'z', una segunda respecto a 'y' y una tercera respecto a "x". A esta combinación de ángulos de Euler se le puede denominar como 'ZYX'. Pueden darse otras como la combinación 'ZXZ' o la combinación 'YZZ'.

Para el robot bajo estudio, de configuración antropomórfica con cinco grados de libertad, se implementó la combinación 'X, Y, Z'.

Desarrollado matemáticamente, esta última (X, Y, Z), esto se puede expresar como se muestra en la ecuación (110).

$$RotX(\alpha)Y(\beta)Z(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (110)$$

El resultado de la multiplicación en la ecuación (110) puede expresarse simbólicamente como se observa en la ecuación (111).

$$RotX(\alpha)Y(\beta)Z(\gamma) = \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \quad (111)$$

Donde cada elemento de la ecuación (111) se muestra en las ecuaciones (112) a (120)

$$E_{11} = \cos(\alpha) * \cos(\beta) \quad (112)$$

$$E_{12} = -\cos(\beta) * \sin(\alpha) \quad (113)$$

$$E_{13} = \sin(\beta) \quad (114)$$

$$E_{21} = \cos(\gamma) * \sin(\alpha) + \cos(\alpha) * \sin(\beta) * \sin(\gamma) \quad (115)$$

$$E_{22} = \cos(\alpha) * \cos(\gamma) - \sin(\alpha) * \sin(\beta) * \sin(\gamma) \quad (116)$$

$$E_{23} = -\cos(\beta) * \sin(\gamma) \quad (117)$$

$$E_{31} = \sin(\alpha) * \sin(\gamma) - \cos(\alpha) * \cos(\gamma) * \sin(\beta) \quad (118)$$

$$E_{32} = \cos(\alpha) * \sin(\gamma) + \cos(\gamma) * \sin(\alpha) * \sin(\beta) \quad (119)$$

$$E_{33} = \cos(\beta) * \cos(\gamma) \quad (120)$$

La matriz de la ecuación (111) es equivalente a la matriz de rotación obtenida de la matriz homogénea general, como se mostró en la ecuación (104). De esta manera también se puede obtener esa matriz de rotación, que es importante para la cinemática inversa, a partir de los ángulos de Euler. Esto es útil suponiendo que los datos de entrada en la cinemática inversa son la posición y como datos de orientación de la herramienta final, los ángulos de Euler.

Para calcular el punto medio en la cinemática inversa, también debe calcularse antes, la matriz de orientación.

O en otro caso, si se requieren calcular los ángulos de Euler a partir de la matriz de orientación, se hace lo siguiente:

Se iguala matriz de orientación, obtenida de la matriz de transformación homogénea, que se representa en la ecuación (77), con la matriz de orientación obtenida por los ángulos de Euler de la ecuación (111), teniendo en cuenta sus elementos dados por las ecuaciones (112), a (120). Esta expresión se muestra en la ecuación 121. (Craig, 2006)

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} = \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \quad (121)$$

Para obtener el ángulo de Euler 'α' se igualan los términos 'R12', 'R11', 'E12' y 'E11'. Después se hace un cociente obteniendo la ecuación (122).

$$\frac{R_{12}}{R_{11}} = \frac{-\cos(\beta) * \sin(\alpha)}{\cos(\alpha) * \cos(\beta)} = -\frac{\sin(\alpha)}{\cos(\alpha)} \quad (122)$$

Despejando el primer ángulo de Euler 'α' a partir de obtener su arco tangente, queda expresado como se observa en la ecuación (123).

$$\alpha = \text{atan2}\left(-\frac{R_{12}}{R_{11}}\right) \quad (123)$$

Para el segundo ángulo de Euler se usan los elementos 'R23', 'R33', 'E23' y 'E33'. Se iguala elemento a elemento como se indica en las ecuaciones (124) y (125).

$$R_{23} = -\cos \beta * \sin \gamma \quad (124)$$

$$R_{33} = \cos \beta * \cos \gamma \quad (125)$$

Se suman los cuadrados de las ecuaciones (124) y (125) para obtener:

$$R_{23}^2 + R_{33}^2 = (\cos \beta * \sin \gamma)^2 + (\cos \beta * \cos \gamma)^2 \quad (126)$$

Factorizando la ecuación (126) se obtiene la ecuación (127).

$$R_{23}^2 + R_{33}^2 = \cos^2 \beta (\sin^2 \gamma + \cos^2 \gamma) \quad (127)$$

Simplificando la ecuación (127) con la identidad trigonométrica $\sin^2 \theta + \cos^2 \theta = 1$, y se despeja 'cos (β)' como se observa en la ecuación (128).

$$\cos \beta = \sqrt{R_{23}^2 + R_{33}^2} \quad (128)$$

Usando el elemento 'R13' se crea un cociente para formar una expresión tangente, 'tan(β)' y queda expresado como se visualiza en la ecuación (129).

$$\tan \beta = \frac{\sin \beta}{\cos \beta} = \frac{R_{13}}{\sqrt{R_{23}^2 + R_{33}^2}} \quad (129)$$

Despejando 'β' de la ecuación (129), se obtiene el segundo ángulo de Euler que se observa en la ecuación (130).

$$\beta = \text{atan2} \left(\frac{R_{13}}{\sqrt{R_{23}^2 + R_{33}^2}} \right) \quad (130)$$

El tercer ángulo de Euler se obtiene haciendo un cociente de los elementos 'R23', 'R33', 'E23' y 'R33' como se muestra en la ecuación (131).

$$\frac{R_{23}}{R_{33}} = \frac{-\cos \beta * \sin \gamma}{\cos \beta * \cos \gamma} = -\frac{\sin \gamma}{\cos \gamma} = -\tan \gamma \quad (131)$$

Despejando al tercer ángulo de Euler 'γ' de la ecuación (131) se obtiene la ecuación (132).

$$\gamma = \text{atan2} \left(-\frac{R_{23}}{R_{33}} \right) \quad (132)$$

La obtención de los ángulos de Euler a partir de la matriz de rotación es útil, por ejemplo, en la cinemática directa. En la cinemática directa se obtiene una submatriz de posición y una submatriz de orientación de la herramienta del robot manipulador. Si se desea obtener los ángulos de Euler para facilitar la lectura de la orientación, se pueden obtener a partir de los elementos de la matriz de orientación y las ecuaciones (123), (130) y (132).

Capítulo 4 Desarrollo de algoritmos

En el capítulo 3 se realizaron los cálculos para obtener la cinemática directa e inversa del robot antropomórfico de cinco grados de libertad (LeArm).

En este capítulo se implementa de la cinemática directa e inversa, teniendo en cuenta los cálculos hechos en el capítulo 3. Además, se crean los algoritmos que permiten el movimiento del robot mediante rutinas secuenciales.

Se utilizó el software labview como entorno de programación gráfica. Se usa este software porque permite generar plataformas con las que se puede interactuar, llamadas interfaces de usuario. La interfaz de usuario se lleva a cabo en el panel frontal del software, mientras que la lógica o programación gráfica con la que interactúan los diversos elementos del programa, se establecen en el diagrama de bloques del software. Además, este software permite diversidad de herramientas, de programación en c, de lógica digital, de comunicación serial, máquinas de estados, manejo de arreglos, gráficas de funciones, por solo dar unos ejemplos. Se suma a ello la facilidad con que se puede observar todos los procesos que contiene un determinado programa, así como los elementos que se involucran y su interacción entre sí.

Los programas generados en el software de labview, pueden contener a su vez otros programas llamados subprogramas o "SubVI". Esto facilita la comprensión de la programación sin tener infinidad de elementos en la pantalla que confundan. Un subprograma puede ser un programa muy complejo por sí mismo.

Se han programado cuatro principales programas que son:

- **Programa para crear una rutina secuencial.**
- **Programa para ejecutar rutina secuencial**
- **Programa para la implementación de la cinemática directa.**
- **Programa para la implementación de la cinemática inversa.**

Los dos últimos programas resuelven el problema cinemático directo e inverso, mientras que los dos primeros permiten crear y ejecutar rutinas secuenciales para el robot manipulador bajo estudio.

Se tienen en este capítulo a otros programas o subprogramas que son usados dentro de los programas principales. Los subprocesos al estar contenidos en bloques, facilita vincularlos en los procesos principales sin tener toda su programación en el programa principal. Es decir, evitar decenas de conexiones que confundan la programación.

Estos subprogramas son:

- **Programa de interpolación.**
- **Programa para cambio de estado.**

- **Programa para convertir ángulos de Euler a matriz de orientación.**
- **Programa para obtener las variables articulares q4 y q5.**
- **Programa para seleccionar las ternas de variables articulares q1,q2,q3.**

Adicionalmente, a inicios de este capítulo se describe un programa que es básico. Su función es escribir datos en el puerto serial, para la comunicación con los servomotores del robot manipulador. Este programa se llama

- **Programa para control de un servomotor mediante ancho de pulso.**

Para la descripción de cada uno de los programas se cuenta con cuatro secciones básicas:

- **Descripción de programa**, una explicación breve del objetivo del programa.
- **Entradas**, datos que deben ser introducidos o modificados por el usuario y que son procesados por el programa.
- **Salidas**, datos que se derivan del proceso de programación y como consecuencia de los datos de entrada.
- **Otros bloques**, bloques de programación propios del software que son herramientas para manipular los datos, hacer operaciones y diversas tareas.

Algunos programas tienen una sección adicional:

- **Descripción del funcionamiento**, sección que sirve para explicar con más detalle el funcionamiento del programa. Para aquellos programas en los que se tienen varios procesos y se vuelven complejos.

Programa para control de un servomotor mediante ancho de pulso

Descripción de programa: Este programa permite el control de un servomotor por ancho de pulso a través de la comunicación serial. El usuario selecciona el ángulo deseado por medio del control “Ángulo servo” en un rango de 0 a 180 grados. Se muestra su panel frontal en la figura 62 y su diagrama de bloques la figura 63.

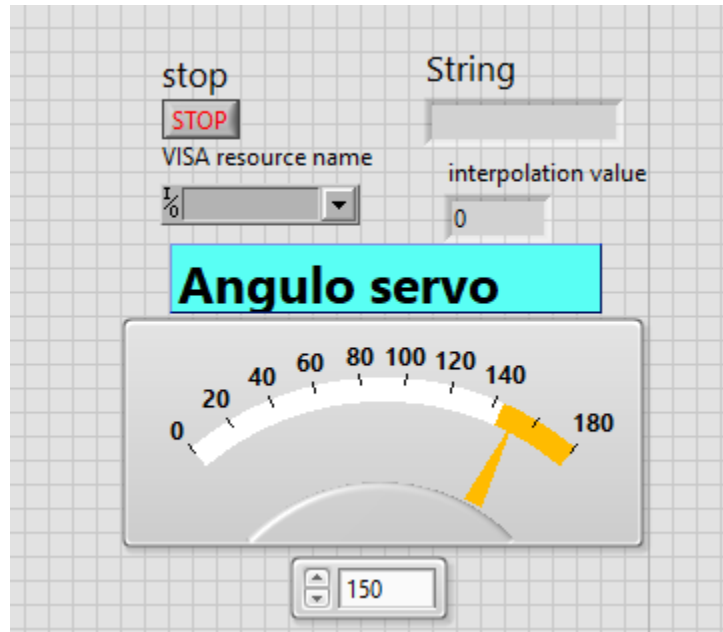


Figura 62 Panel Frontal del Programa para control de un servomotor mediante ancho de pulso.

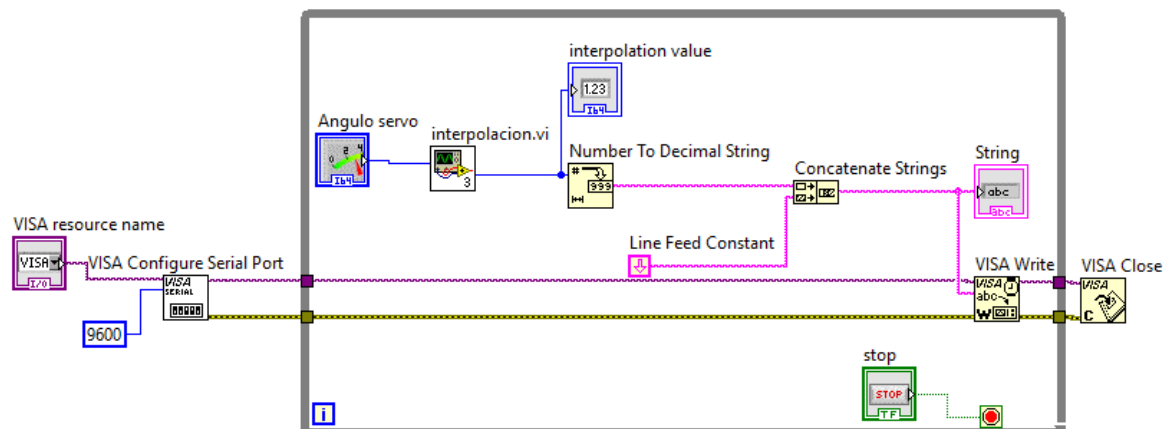


Figura 63 Diagrama de bloques del Programa para control de un servomotor mediante ancho de pulso.

Entradas:

Visa Resource name, tiene como objetivo seleccionar el puerto desde el cual se quiere leer o enviar datos.

Stop, la entrada realiza un paro del programa desde de la estructura “While Loop” en vez de ejecutar una parada forzada por medio de los botones de paro propios de LabVIEW.

Ángulo servo, este control tienen como función, ir de un intervalo de cero a ciento ochenta grados que es intervalo deseado de operación para manipular un

servomotor. El dato del ángulo seleccionado por el usuario es enviado como dato a través del puerto serial.

Salidas:

String, a través de este indicador se muestran los datos que se han enviado al puerto serial que marcan el ángulo indicado por el usuario mediante “ángulo servo”.

Interpolation value, muestra el valor del ancho de pulso obtenido a través del subprograma o SubVI de nombre, “interpolación”.

Otros bloques:

Visa Configure Serial Port, configura el puerto serial a utilizar.

Interpolacion.vi, es un subprograma. Su función es obtener el equivalente en ancho de pulso como salida, al dar como entrada un valor de ángulo dado en grados.

Number to decimal string, convierte un valor numérico decimal en otro valor de tipo “string”. Con este formato pueden ser enviados los datos a través del puerto serial.

Line Feed Constant, es un salto de línea que diferencia entre cadena de datos diferentes. El salto de línea es enviado al serial como parte de la cadena de datos. Con la programación en la tarjeta TIVA se puede hacer esa diferenciación entre datos. De esta manera, la tarjeta TIVA lee cuando el usuario tiene la intención de enviar otro valor de entrada de ángulo.

Concatenate Strings, función que concatena diferentes datos de tipo “string”. Se concatena el valor de ángulo dado por el usuario y un salto de línea.

VISA Write, función de librería VISA que permite escribir datos en el serial.

VISA Close, función de VISA que cierra la comunicación serial.

Programa de interpolación

Descripción de programa: Es un programa que asocia un ángulo con su equivalente en ancho de pulso mediante una herramienta de interpolación polinómica. El ángulo es seleccionado por el usuario con un rango de 0 a 180 grados y se obtiene como salida su equivalente en ancho de pulso. Se muestra su panel frontal en la figura 64 y su diagrama de bloques la figura 65.

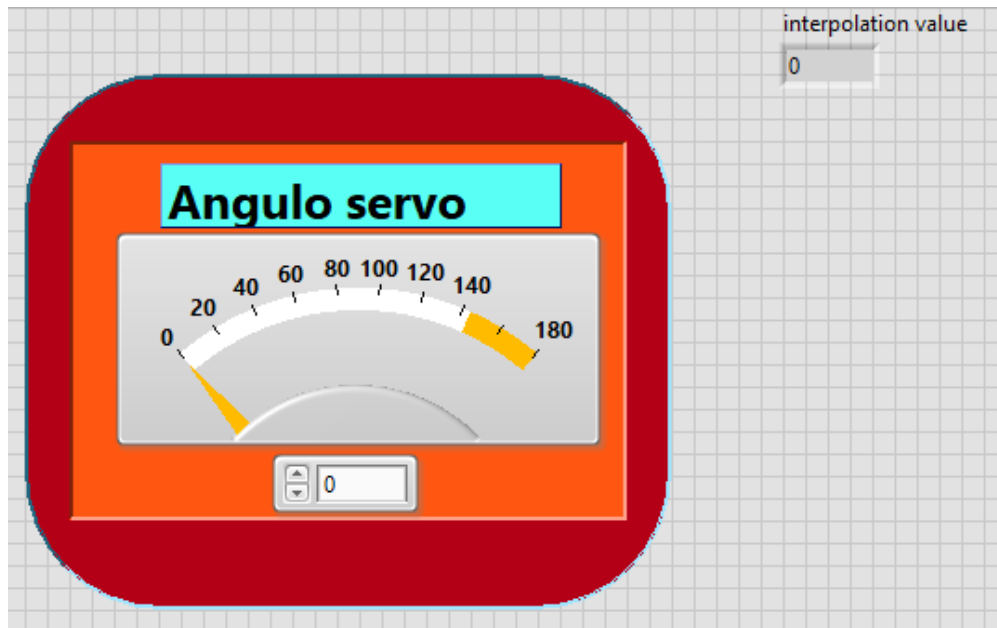


Figura 64 Panel frontal del Programa de interpolación.

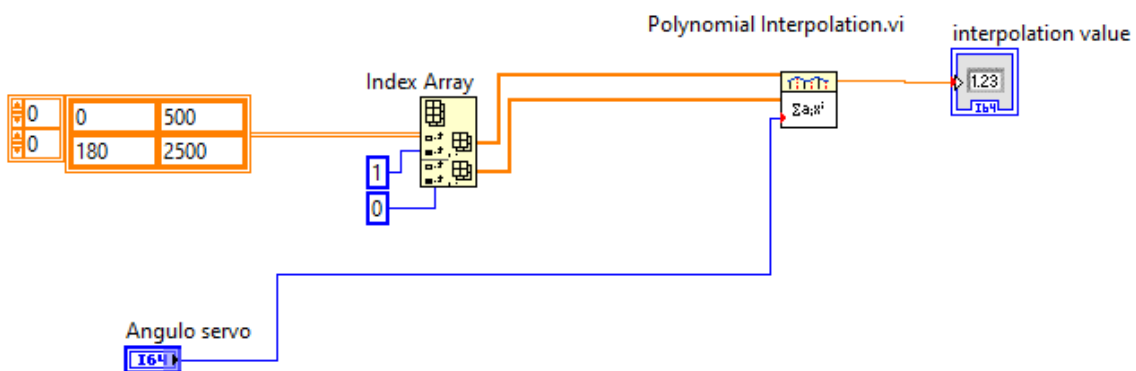


Figura 65 Diagrama de bloques del Programa de interpolación.

Entradas

Ángulo servo, es un control que varía el valor de un ángulo de 0 a 180 grados.

Salidas

Interpolation value, es un valor que es el equivalente en ancho de pulso de la entrada dada en grados.

Otros bloques

Index Array, devuelve el elemento o subarreglo de un arreglo de n dimensiones y que se define por los índices de entrada.

Polynomial Interpolation.vi, Interpola o extrapola la función f en x dado un conjunto de n puntos (x[i], y[i]), donde $f(x[i]) = y[i]$, f es cualquier función y dado un número x valor .

Máquina de estados

En el estudio de los circuitos digitales existe uno que se denomina secuencial. Para este tipo de sistemas, las salidas están determinadas en cualquier momento por las entradas en la secuencia dadas en el presente como de las del pasado. También requieren de memoria para conocer las entradas pasadas. (Alcubilla, Pons, & Bardes, 1995)

Dentro de los sistemas secuenciales existen dos clasificaciones, los síncronos y los asíncronos. Los primeros suelen controlarse o sincronizarse con un reloj. Los asíncronos no dependen de un pulso de reloj, evolucionan con el cambio de entrada.

El concepto máquina suele aplicarse a los circuitos secuenciales. Concretamente, “máquinas secuenciales Finitas”. El adjetivo “finitas” es porque una máquina de estados está conformada por un número finito de estados, es decir, que su desarrollo tiene un inicio y un fin.

Los estados son parte del proceso de una máquina de estados. Internamente, realizan tareas, evalúan datos y condiciones. El estado puede entregar algún dato de salida que de información a la máquina para tomar la decisión de a que parte del proceso ir, es decir, otros estados.

Ejemplo, se tiene un sistema que efectúa un cobro automatizado. Se tiene un estado que pide al usuario lo que desea hacer, un segundo estado que pide ingresar en número la cantidad de dinero, otro donde se pide ingresar dinero y un último donde se imprime un “ticket” y da las gracias al usuario.

Cada estado dentro del sistema hace una tarea única. Los estados tienen la capacidad de llamar a otros a partir del cumplimiento de alguna condición. La lógica de secuencia establecida por el diseñador de la máquina determina cuando pasar de un estado a otro.

Las dos máquinas de estado que son usadas con frecuencia son las máquinas de Moore y Mealy. En las máquinas de Moore las salidas del proceso están definidas a partir del estado presente. En cambio, en las máquinas de Mealy las salidas dependen del estado y entradas presentes. (Balabanian & Carlson, 2002)

Las máquinas de estados son implementables cuando los estados de un sistema a desarrollar son identificables. Es posible pensar en máquina de estados donde se necesite de algoritmos que ayuden a tomar decisiones.

Un uso de las máquinas de estado son las interfaces de usuario. Las decisiones del usuario en la interfaz, permite interactuar en las distintas partes del proceso. Se necesita monitorear las decisiones del usuario para determinar cada acción a realizar o en otras palabras, saber en qué momento transitar de un estado a otro.

Se representa gráficamente una máquina de estados mediante un diagrama de estados. En el diagrama se especifica el funcionamiento del sistema secuencial representando las relaciones de funciones y entradas por medio de nodos y arcos o flechas. Los nodos representan los estados, mientras que las flechas que unen a los nodos representan las interacciones y transiciones entre estados. (Ni, 2023)

Con los conceptos revisados hasta ahora, se puede resolver un problema práctico, no olvidando crear un diagrama de estados, previamente.

El problema en cuestión consiste en representar una interfaz de usuario para generar una tabla con posicionamientos angulares para cada articulación del robot LeArm.

La idea es formar un arreglo números con 6 columnas, una para cada articulación, contando a la herramienta del robot. Cada fila de 6 columnas representa una posición del robot o un punto de una trayectoria. Este proceso no tiene en cuenta otros aspectos de la robótica de manipuladores, como la teoría de control. También sirve para realizar pruebas de desempeño al producir trayectorias que se pueden ejecutar para el robot LeArm, una vez implementado en el software gráfico.

Como no hay forma de determinar el número de filas o de puntos de la trayectoria, la máquina de estados requiere una comunicación constante con el usuario para preguntarle si quiere generar un punto nuevo de la trayectoria.

Se muestra en la figura 66 el diagrama de estados para la solución del problema.

Entendido el diagrama de estados, se realiza la máquina de estados programada de forma gráfica en el apartado Programa para crear rutina secuencial.

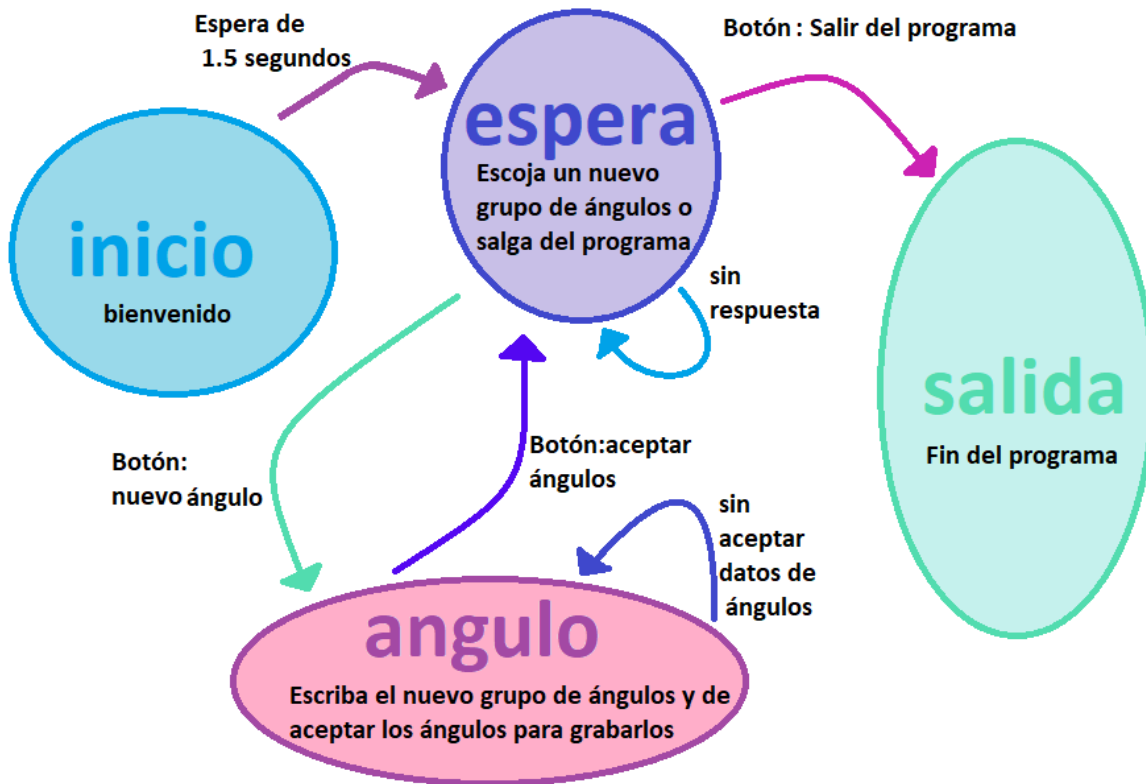


Figura 66 Diagrama de estados para una máquina de estados que representa una interfaz de usuario para crear una tabla con posicionamientos angulares de cada articulación y efector del robot manipulador de 5 grados de libertad

Programa para crear una rutina secuencial

Descripción de programa: Este programa es una interfaz de usuario con el fin de crear una rutina secuencial para un robot LeArm. El usuario ingresa los valores de los ángulos de los servomotores del robot.

Al posicionar cada servomotor del robot obtenemos una posición y orientación del efector final. Se obtienen diversas posiciones en el espacio como combinación de los valores angulares de los seis servomotores. Estos puntos al unirse conforman una trayectoria.

Por tanto, el programa obtiene del usuario una serie de seis datos que se guarda en una fila de un arreglo. Dicha fila representa el conjunto de valores para obtener una posición y orientación del efector final en un punto del espacio tridimensional. Se obtienen tantos puntos o filas como el usuario requiera.

Se adquiere como resultado un archivo “cvs” que contiene el arreglo de datos ingresado con anterioridad por el usuario. La dirección del archivo “cvs” está determinada por la programación y puede ser cambiada.

Se usa como base, el diagrama de estados de la figura 66.

Se muestra su panel frontal en la figura 67 y su diagrama de bloques la figura 68 y de manera específica cada caso que conforma a la estructura “case”, en las figuras 69, 70, 71 y 72.

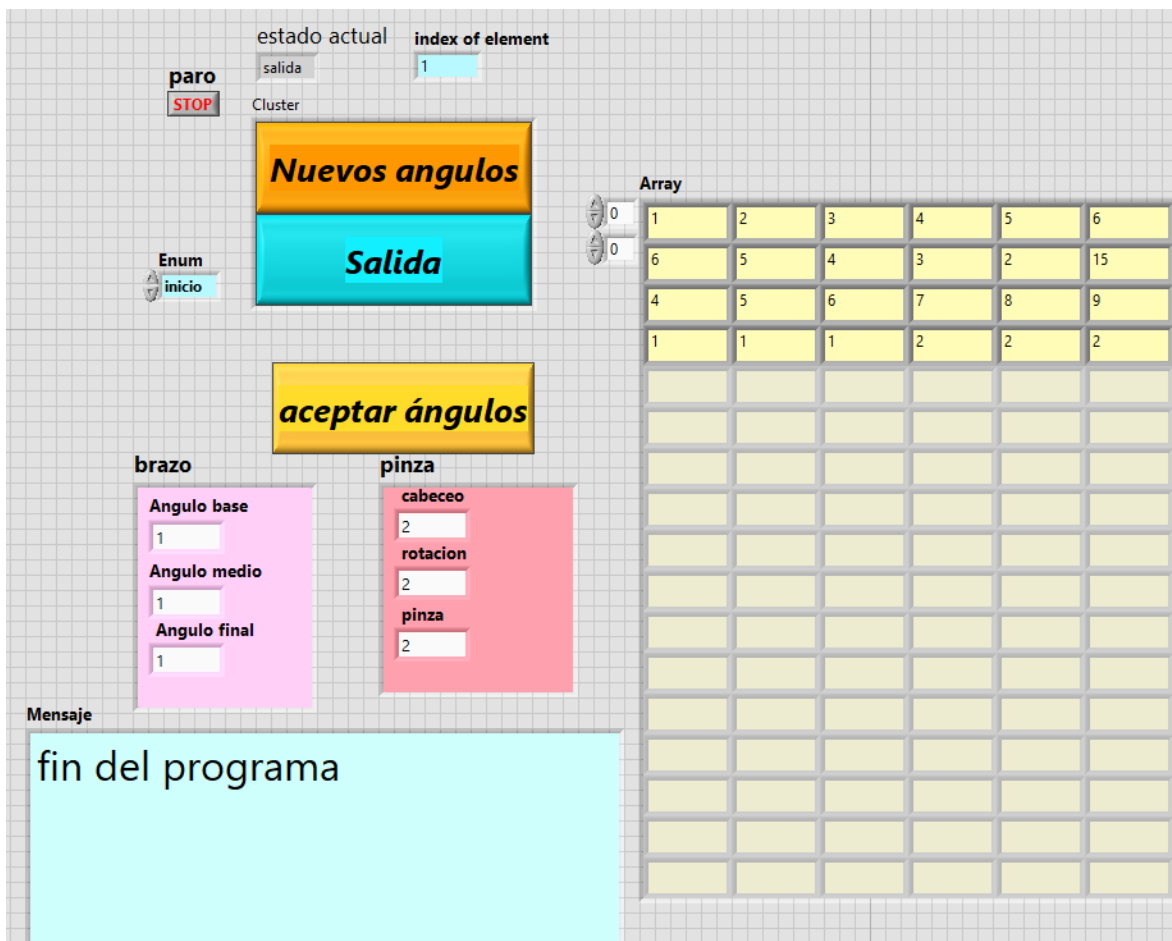


Figura 67 Panel Frontal del Programa para crear una rutina secuencial.

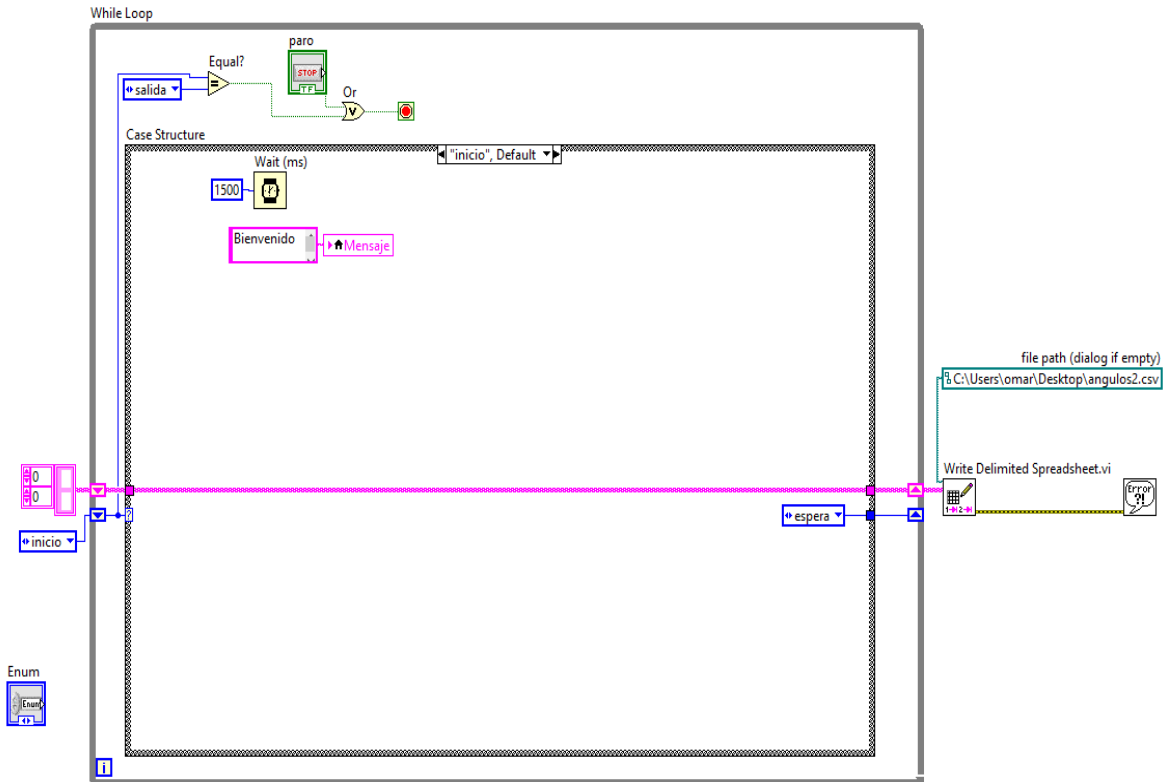


Figura 68 Diagrama de bloques del Programa para crear una rutina secuencial.

Descripción de funcionamiento del programa

El diagrama de bloques contiene la estructura de programación que hace que los diversos elementos funcionen en conjunto como una máquina de estados.

El diagrama de bloques está compuesto principalmente por una estructura “While loop” que permite el funcionamiento del sistema mientras existan las condiciones. Las condiciones son que, el mismo usuario no pare el programa manualmente o que debido a las condiciones que se den en la máquina de estados, se determine que el proceso de la máquina de estados terminó.

La estructura “While loop” contiene una estructura “case”. Esta tiene cuatro casos que contiene cada uno, las funciones a ejecutar. Los casos son equivalentes a los estados de una máquina. De acuerdo con la programación y al diagrama de estados presentado en la figura 66, se puede determinar las condiciones bajo las que se ejecuta un caso. Los cuatro estados son: inicio, espera, ángulo y salida.

“Inicio”, es el estado que aparece por defecto en la máquina de estados. Únicamente tiene un mensaje de bienvenida y una función de tiempo que sitúa al usuario 1.5 segundos en dicho de estado para cambiar automáticamente al siguiente, “espera”. El caso “inicio” se visualiza en la figura 69.

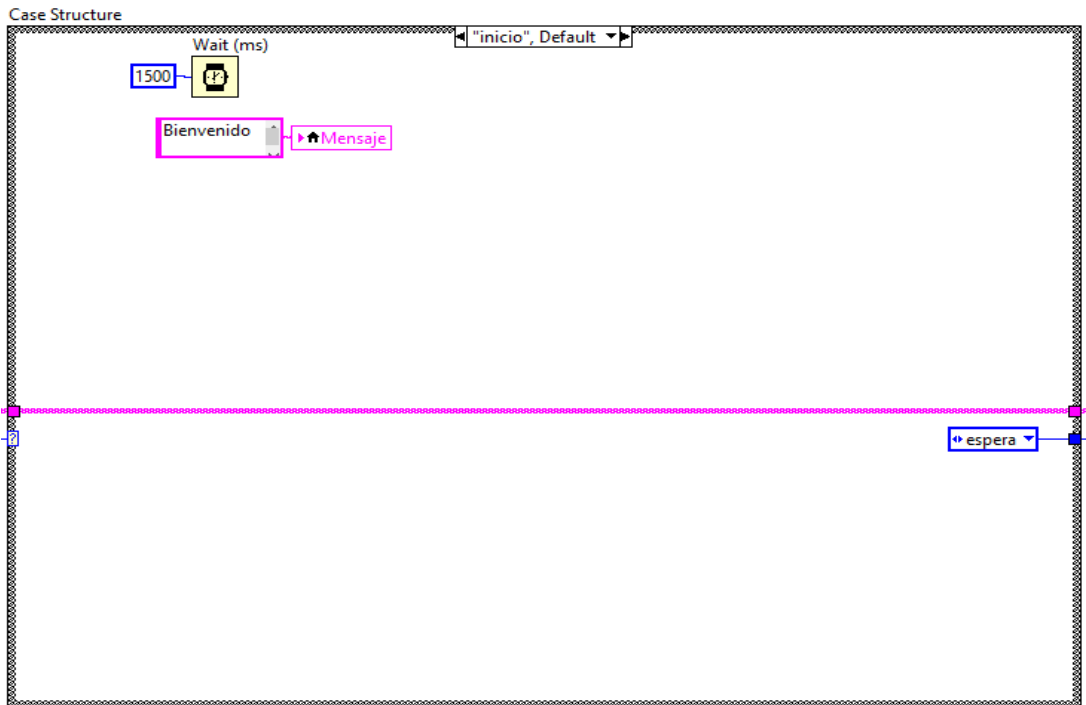


Figura 69 Vista de caso "inicio" de la estructura "Case".

El estado de espera contiene un "cluster", una estructura con diferentes tipos de elementos. En este caso los elementos son iguales y son dos botones booleanos. Los botones son "Nuevos ángulos" y "salida". De acuerdo con la elección del usuario se pasa al siguiente estado.

Para que el cambio de estado funcione, contiene un subprograma llamado "cambioEstado.vi". Se puede ver con más detalle su funcionamiento en la sección dedicada a este programa.

El caso "espera" se observa en la figura 70.

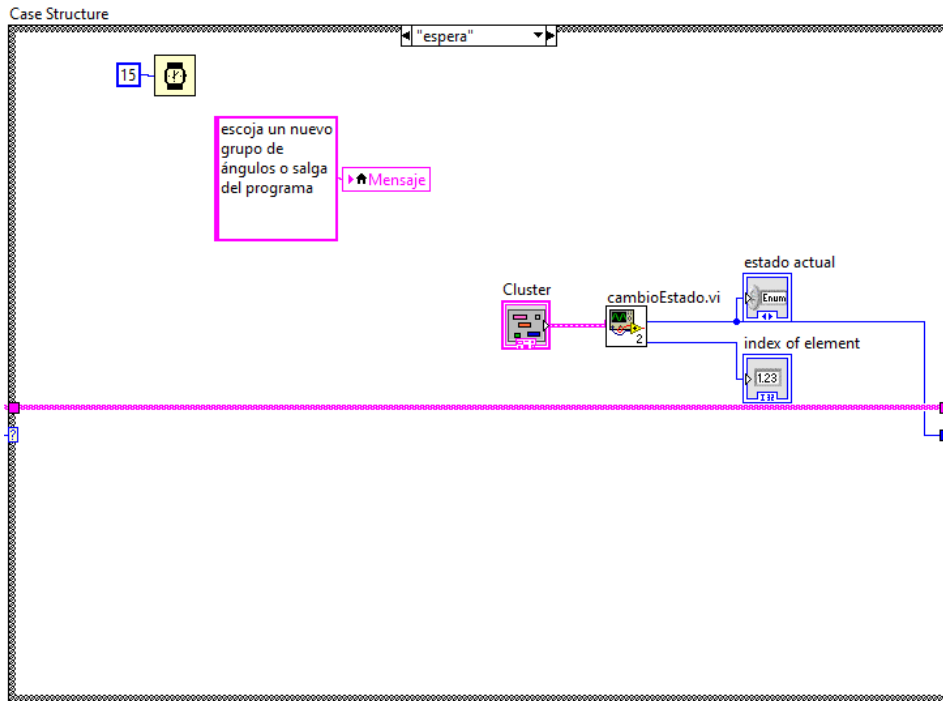


Figura 70 Vista de caso "espera" de la estructura "Case".

Cuando el usuario selecciona la opción "nuevo ángulo", es dirigido al estado "ángulo". En este, el usuario debe ingresar los valores de ángulo para posicionar los servomotores del robot. Se manda un mensaje al usuario, visible en la variable "mensaje" en el panel de control.

Los datos ingresados son enviados en forma de arreglo a la función "Write Delimited Spreadsheet.vi", la cual se encarga de crear un archivo de extensión "csv" y almacenarlos ahí. Esta función se encuentra fuera de la estructura "while loop".

Una vez ingresados los datos, el usuario debe oprimir el botón "aceptar ángulos", para que lo lleve al estado "espera" y se guarden los datos en el archivo "csv". En caso de no presionar el botón, el usuario se mantiene en el estado "ángulo" hasta que ingrese los datos.

La manera en que se cambia de estado es con una función "select" que de acuerdo con el valor booleano del botón "aceptar ángulos" es como se elige la variable tipo "enum" que cambia el estado.

El caso "ángulo" se observa en la figura 71.

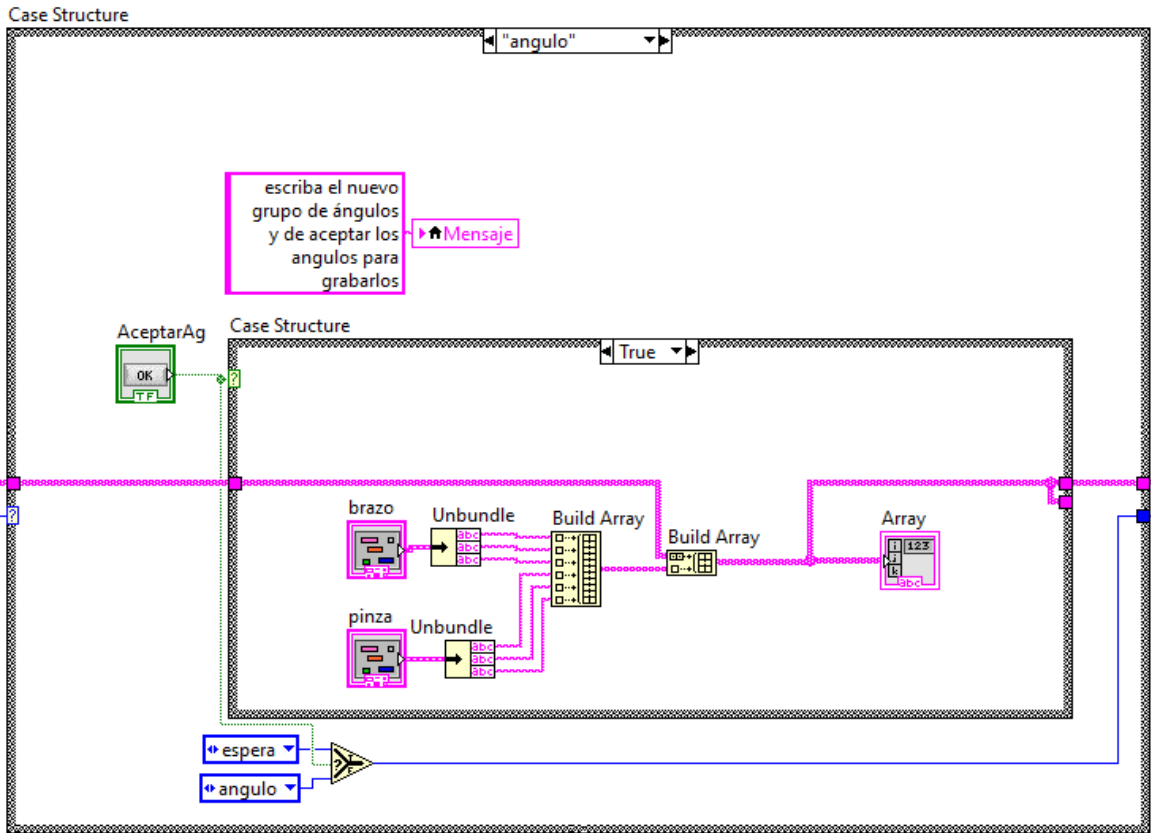


Figura 71 Vista de caso "ángulo" de la estructura "Case".

El estado "salida" indica con un mensaje que el programa terminó. Se llega a este cuando el usuario presiona el botón "salida" mientras se encuentra en el estado "espera".

El estado "salida" se visualiza en la figura 72.

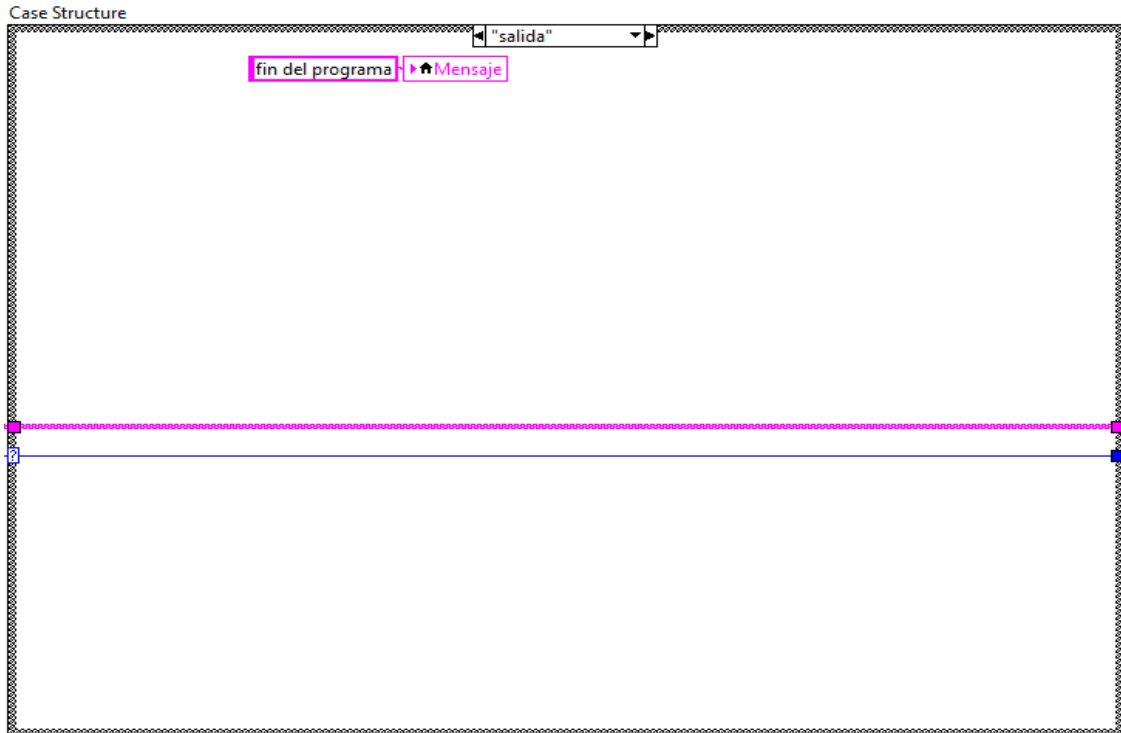


Figura 72 Vista de caso "salida" de la estructura "Case".

Entradas:

Paro, la entrada realiza un paro del programa desde de la estructura "While Loop" en vez de ejecutar una parada forzada por medio de los botones de paro propios de LabVIEW.

Cluster(Nuevos ángulos, Salida), se trata de un cluster con dos botones booleanos que representan dos opciones para el usuario. Escoger un nuevo dato de ángulo a ingresar o salir del programa y finalizar.

Enum, variable enum que inicializa el proceso en la variable que tiene seleccionada, en este caso es la variable "inicio" que nos lleva al estado "inicio".

Brazo, pinza, estos cluster tiene un conjunto de controles de variable numérica donde el usuario les da valores de acuerdo con los ángulos que desea que aparezcan en el arreglo. Ángulo base(servo base), ángulo medio (servo hombro), ángulo final (servo codo), cabeceo (servo cabeceo muñeca), rotación (servo rotación de muñeca), pinza (servo que abre y cierra pinza).

Aceptar ángulos, es un botón booleano que permite al usuario comunicar al programa que esta dé acuerdo con los datos que acaba de ingresar para que sean grabados de manera definitiva y pasar al siguiente estado.

Salidas:

Array, es un arreglo que tiene como finalidad visualizar los datos que son ingresados por el usuario.

Estado actual, tiene como finalidad mostrar al usuario el estado de la máquina estados en que se encuentra actualmente el proceso.

Index of element, es un índice que se usa en el subprograma “cambiodeestado.vi” se puede revisar la descripción del programa para más detalles. No tiene ningún efecto importante, solo es la visualización del dato para el propio subprograma y no es de interés del usuario final.

Mensaje, este string es un indicador. Da instrucciones al usuario para guiarlo de acuerdo con el estado en que se encuentra.

Otros bloques

Equal?, Devuelve un valor “true” cuando los dos valores comparados son iguales.

Or, compara dos valores “true” o “false” y regresa un valor “true” o “false” de acuerdo a la tabla de verdad de una compuerta “Or”

File path (dialog if empty), es un tipo de datos de LabVIEW que identifica la ubicación de un archivo en el disco duro.

Write Delimited Spreadsheet.vi, convierte una matriz 2D o 1D de cadenas, enteros con signo o números de doble precisión en una cadena de texto y escribe la cadena en un nuevo archivo de flujo de bytes o agrega la cadena a un archivo existente.

While Loop, es una estructura tipo “while” que ejecuta lo que contiene dentro mientras el usuario no pare la rutina mediante un botón de paro o se determine que termino el proceso de acuerdo a la propia programación.

Case structure, contiene uno o más subdiagramas (casos). El valor conectado al selector de casos determina qué caso ejecutar.

Wait(ms), espera el número especificado de milisegundos y devuelve el valor del temporizador de milisegundos.

CambioEstado.vi, Es el nombre del **programa para cambiar estado**, se explica más a detalle en la sección dedicada a este programa.

Unbundle, separa elementos de clusters por nombre, accede y organiza los elementos en un clúster.

Build Array, concatena varias matrices o añade elementos a una matriz de n dimensiones.

El cambio de estado se da por causa de una función llamada “shift register” que se encuentra en el límite la estructura “while loop”(es una flecha azul que apunta hacia

abajo). Este shift register pasa valores de iteraciones anteriores a través de un bucle a la siguiente iteración. Este valor se retroalimenta. En la figura 73 se muestra como está conectado. Está inicializado con una variable constante tipo “enum” que contiene un valor llamado “inicio”.

En el otro extremo se encuentra la terminación del “shift register” (flecha azul hacia arriba) que puede ir conectado la variable tipo “enum” del siguiente estado al que debe ir la interacción. Este valor cambia de acuerdo a las decisiones del usuario, dado por los botones con los que interactúa.

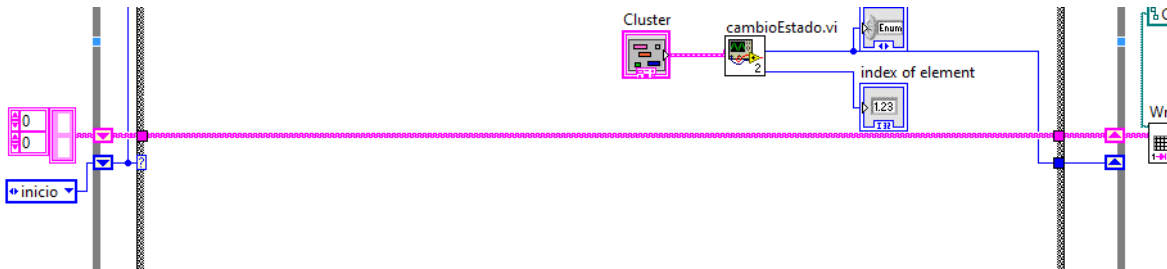


Figura 73 Vista de la variable de tipo enum y de los "shift register" que permiten el cambio de estado en la máquina de estados.

Para finalizar el programa, el estado de la máquina debe ser “salida” o el usuario necesita parar la estructura while loop. Para esto se usa la configuración como se observa en la Figura 74.

Lo que se hace, es tomar la conexión del “shift register” que contiene la información del estado en que se encuentra el programa, para compararlo mediante una función “equal” con un constante tipo “enum” de valor “salida”. De la función “equal” sale un valor booleano “true o false” que sirve para detener el proceso de la estructura “While Loop” y, por tanto, terminar el programa cuando el estado actual sea “salida”.

También se implementó un botón de paro que permite al usuario detener el proceso, independientemente del estado de la máquina. Para que estos funcionen de manera conjunta, se implementó una función “or” que simula la función de una compuerta “or”.

La compuerta or, tiene una tabla de verdad, como la que se observa en la figura 75, que contiene las combinaciones posibles de las dos condiciones y la salida al ser realizada la operación “or”. Basta con que una o ambas condiciones sean ciertas para obtener un valor booleano “true” y así activar el paro de la estructura “while loop”

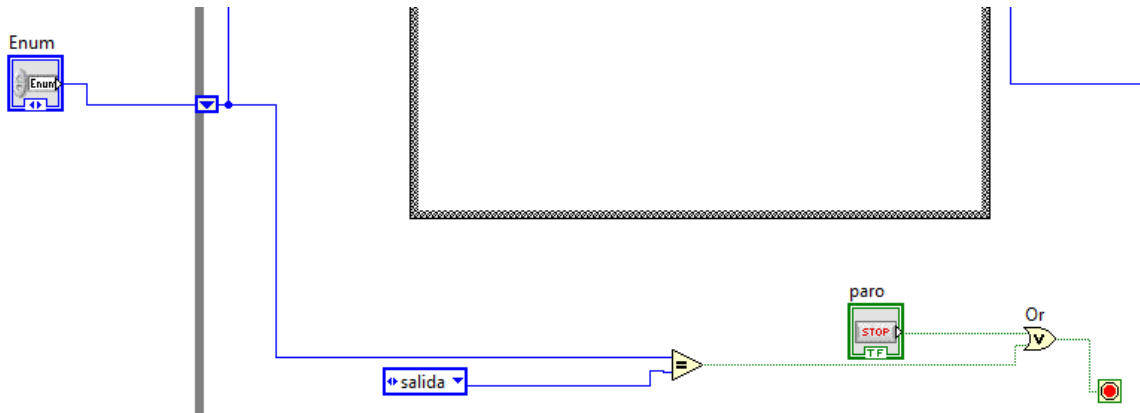


Figura 74 Vista del uso del bloque "or" para parar el programa automáticamente o manualmente.

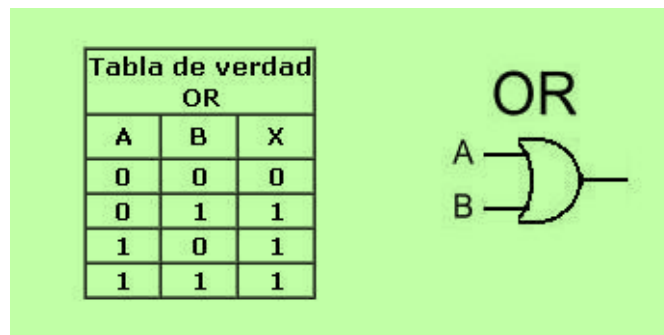


Figura 75 Compuerta "or" y su respectiva tabla de verdad.

Programa para cambio de estado

Descripción de programa: Se usa como un subprograma dentro del **programa para crear una rutina secuencial**. Su funcionamiento consiste en convertir la opción seleccionada por uno de los botones del "Cluster", en una variable de tipo "enum" de esta manera se puede interpretar como un cambio de estado cuando el usuario seleccione una opción por medio de los botones.

Se muestra su panel frontal en la figura 76 y su diagrama de bloques la figura 77.

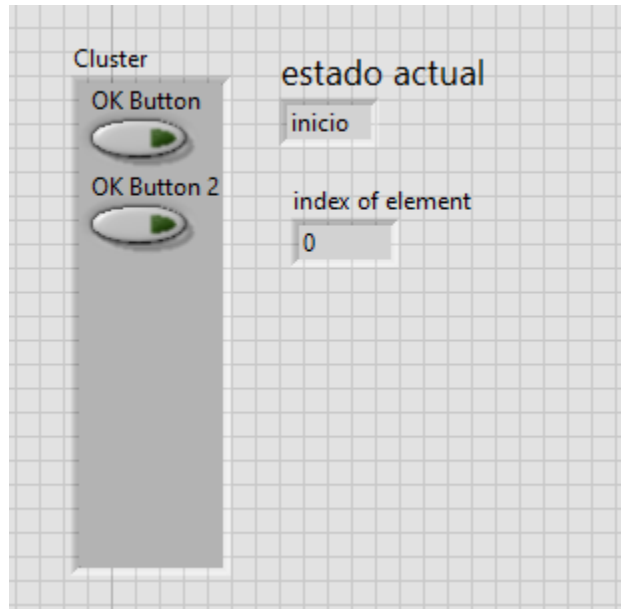


Figura 76 Panel frontal del programa para cambio de estado.

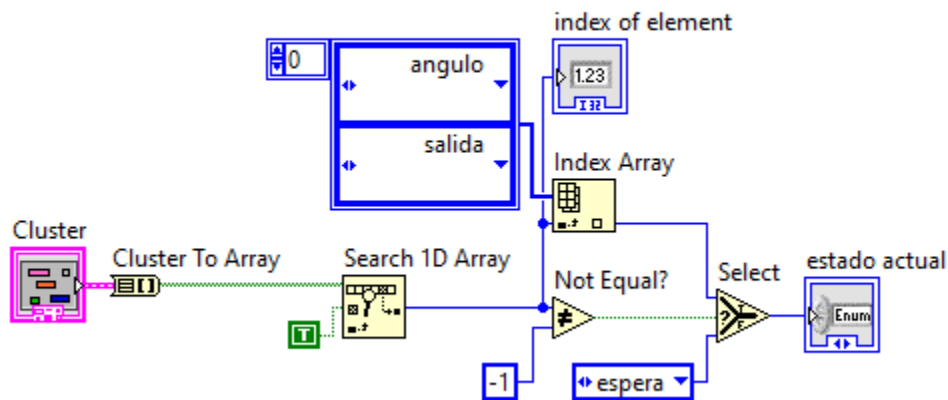


Figura 77 Diagrama de bloques del programa para cambio de estado.

Entradas

Cluster (ok button, ok button 2), es un grupo de datos que puede tener varios tipos de datos. En este caso se tienen dos botones de tipo booleano.

Salidas

Estado actual, es el resultado de todo el proceso del programa para obtener el estado al cual el programa de la máquina de estados debe ir de acuerdo a la opción seleccionada por el usuario mediante los botones del cluster.

Index of element, indica el índice del elemento obtenido por la función “Search 1D Array”.

Otros bloques

Cluster to Array, convierte la información de un cluster en un arreglo.

Search 1D Array, busca un elemento en una matriz de una dimensión que comienza en el índice de inicio (0). Debido a que la búsqueda es lineal, no necesita ordenar la matriz antes de llamar a esta función. Si no se encuentra el elemento, el índice del elemento es -1.

Not Equal?, función que hace una comparación entre dos elementos, entrega una salida "1" si los elementos son diferentes, y un "0" si los elementos son iguales. El cluster tiene transformada su información en arreglo que solo tiene dos elementos los cuales se les asigna los números "0" y "1". Cuando el usuario no selecciona ninguna opción, la función "Search 1D Array" entrega un número "-1". La función "Not Equal?", hace una comparación en una entrada. Compara la salida de "Search 1D Array" con una constante con valor "-1". Esto hace que se entregue el estado "espera" que mantiene al usuario en ese estado hasta no seleccionar alguna de las opciones del cluster.

Index Array, Es una función que devuelve un elemento de un arreglo según lo indique la entrada del índice. Se tiene dos casos posibles de un arreglo conformado por dos elementos que representan dos estados de la máquina de estados, son equivalentes a los elementos del cluster.

Select, devuelve el valor conectado a la entrada t o a la entrada f, según el valor de s. Si s es VERDADERO, esta función devuelve el valor conectado a t. Si "s" es FALSO, devuelve el valor conectado a f. En el caso del presente programa, es controlado por la función "Not Equal?". Mediante el "cluster" se elige una opción, se selecciona un elemento de un arreglo ya sea el elemento "1" o el elemento "0". Entonces se activará la opción "true" o verdadera de la función select y dará como salida un elemento del arreglo constante ("ángulo" o "salida") conectado a la función "Index Array".

En caso de que el usuario no elija un elemento del "cluster", la función arroja un número "-1". La función "Not Equal?", compara con una constante "-1" y al ser iguales, da como salida un valor "false" o falso. El valor falso es recibido por la función "Select" y da como salida una constante tipo "enum" con el valor de estado "espera".

Programa para ejecutar rutina secuencial

Descripción de programa: Este programa abre un archivo "cvs" que contiene un arreglo con valores que representan ángulos en los cuales el usuario desea posicionar seis servomotores. El archivo "cvs" tiene los datos registrados durante la ejecución del **programa para crear una rutina secuencial**. El arreglo tiene como dimensiones, 6 columnas y n filas. Son n filas porque depende de la cantidad de datos ingresados por el usuario en la ejecución del programa de la máquina de

estados. Cada fila representa un punto de una trayectoria. Una trayectoria es el conjunto de los puntos en el espacio, que, al ejecutarse de manera secuencial, forman una trayectoria continua.

Para poder ejecutar los movimientos del robot, se envía los datos del arreglo en el bus serial para ser leídos por la tarjeta y el microcontrolador. A su vez, se envía los datos a cada servo y se posiciona las articulaciones correspondientes. Las “n” interacciones se ejecutan en un ciclo infinito o hasta que el usuario pare el programa. También queda a disposición del usuario, un control de tiempo para manipular el tiempo de ejecución de cada ciclo de una estructura “For Loop”. Al controlar esto, el usuario puede variar la velocidad con la que se mueve el robot.

Se muestra su panel frontal en la figura 78 y su diagrama de bloques la figura 79.



Figura 78 Panel frontal del Programa para ejecutar rutina secuencial.

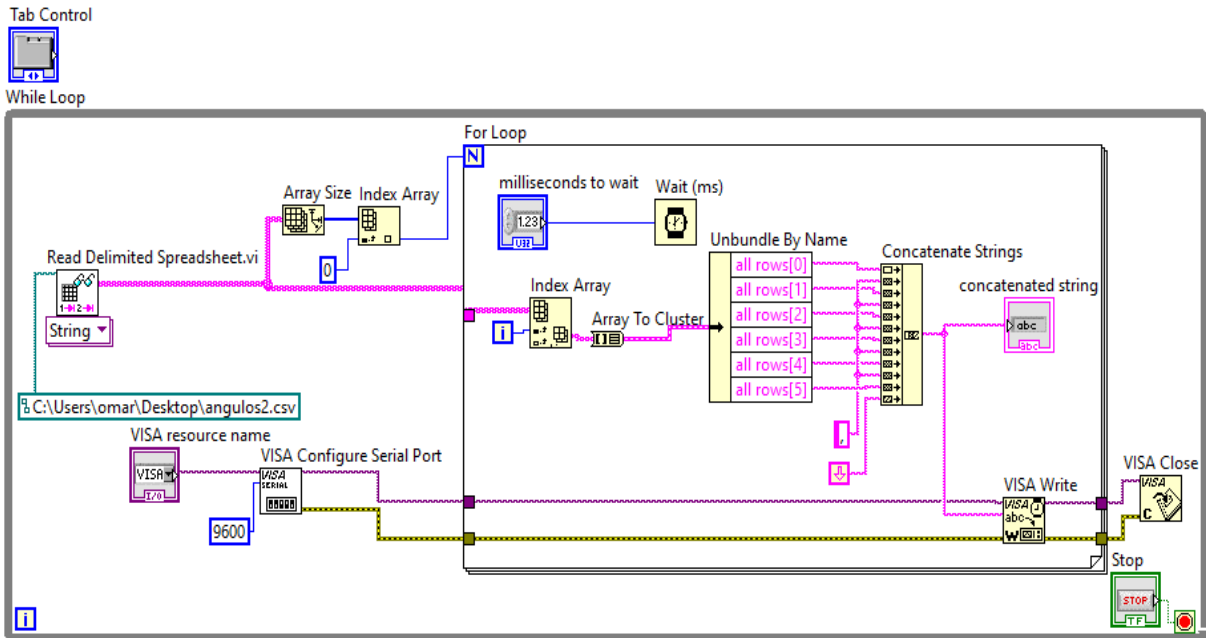


Figura 79 Diagrama de bloques del Programa para ejecutar rutina secuencial.

Entradas:

Visa Resource name, seleccionar el puerto desde el cual se quiere leer datos recibidos o enviarlos a determinado dispositivo.

Stop, ejecuta un paro de la estructura “While Loop” en vez de realizar una parada forzada del programa por medio de los botones de paro de LabVIEW.

Tab Control, es un control que consta de páginas o pestañas.

Se agregan objetos del panel frontal a las páginas del control y se seleccionan las pestañas para mostrar cada una. Para el objetivo del programa solo se usa como un objeto decorativo y no representa ningún problema en la sintaxis de la programación al no estar conectada su salida con algún otro elemento de diagrama de bloques.

Millisecons to wait, controla los milisegundos que dura cada interacción del ciclo “For Loop”. Se recomienda emplear, como constante, 600 milisegundos. Con este tiempo se han hecho pruebas y el robot corre a una velocidad aceptable. Entre más grande sea el valor, la trayectoria será lenta. Si el tiempo es más pequeño, la rutina secuencial se ejecutará rápidamente.

Salidas:

Concatenated string, muestra los valores que cada servomotor debe tomar como posición angular. Estos datos son enviados al bus serial.

Otros bloques

While Loop, es una estructura tipo “while” que ejecuta lo que contiene programado dentro mientras el usuario no pare la rutina mediante un botón de “stop”.

Read Delimited Spreadsheet, lee un número específico de líneas o filas de un archivo de texto numérico. Este comienza en un desplazamiento de caracteres específico y convierte los datos en una matriz de dos dimensiones de doble precisión de números, cadenas o enteros.

Visa Configure Serial Port, Inicializa el puerto serial con la configuración especificado por “Visa resource name”. En este programa solo se conecta las entradas de “VISA resource name” (para selección manual del puerto u usar por parte del usuario) y una constante con valor de 9600 para la velocidad de transferencia de datos (baudios).

VISA Write, Escribe los datos del búfer de escritura en el dispositivo o interfaz especificado por “VISA Resource name”.

VISA Close, Cierra una sesión de dispositivo o un objeto de evento especificado por “Visa resource name”.

Wait(ms), espera el número especificado de milisegundos y devuelve el valor del temporizador de milisegundos.

Index Array, Es una función que devuelve un elemento de un arreglo según lo indique la entrada del índice que selecciona el número de elementos del arreglo.

Array to Cluster, convierte elementos de un arreglo a elementos tipo “cluster”. Los “cluster” son grupos de datos de diferente tipo.

Unbundle By Name, devuelve los elementos del clúster por nombres específicos.

Concatenate Strings, función que permite concatenar diferentes datos de tipo “string”. En este programa se concatena el valor de ángulo dado por el usuario y un salto de línea.

For Loop, estructura que ejecuta su subdiagrama “n” veces, donde n es el valor conectado al terminal de conteo (N). El terminal de iteración (i) proporciona el recuento de iteraciones de bucle actual, que va de “0” a “n-1”.

Descripción de funcionamiento de programa

El programa contiene una configuración de escritura de datos en el serial. Se realiza por medio de los bloques de “VISA”. Con “Visa resource name” se selecciona el puerto en el cual se van a escribir los datos, es decir, el puerto USB al que se encuentra conectada la tarjeta TIVA. Los datos se escriben en el serial por la acción del bloque “VISA Write” y se cierra el ciclo de comunicación serial con “Close VISA”.

La apertura del archivo de tipo “csv” se lleva a cabo con “Read Delimited Spreadsheet”. Sobresalen dos lazos de salida, las dos contienen la información del arreglo leído. Una salida se conecta al bloque “Array size” para obtener otro arreglo con la información con el número de columnas y filas. Con index array se obtiene de “Array size”, el elemento “0” que contiene la información del número de filas. Este valor se conecta a la terminal “N” de la estructura “For Loop”, y es el número de iteraciones de esta estructura.

El segundo lazo de “Read Delimited Spreadsheet”, entra a la estructura “For Loop” y se conecta a “index array” que a cada iteración dará como salida, la fila “i” de acuerdo a lo indicado por la terminal “i” de la estructura “For loop”.

Los elementos a la salida del “Index Array” pasan a un Bloque “Array To Cluster” que conecta a “Unbundle By Name”. Este último facilita separar cada elemento de la fila de datos, por nombre. Así se puede diferenciar entre elementos del arreglo sin saber su contenido.

Estos elementos una vez separados pasan a formar una cadena de datos por medio del bloque “Concatenate Strings”. Cada valor se alterna con un carácter “coma” para diferenciarlos con la programación en el microcontrolador y diferenciar la información entre datos para un servomotor y otro. Se finaliza la cadena de datos con un carácter de salto de línea, que permite, con la programación del microcontrolador, diferenciar entre cadenas de datos enviadas en cada iteración.

La cadena de datos que sale del bloque “Concatenate Strings” contiene la información que se escribe en el bus serial por medio del bloque “VISA Write” y que también pueden visualizarse en el indicador “Concatenated String”.

La velocidad de ejecución de la rutina secuencial puede variarse con el control “milliseconds to wait”. Se recomienda ejecutar la rutina con un tiempo de 600 milisegundos. Se ha hecho pruebas con este tiempo y se ejecuta la rutina con una velocidad óptima.

Programa para la implementación de la cinemática directa

Descripción de programa: Este programa tiene como finalidad ser una interfaz de usuario que permita manipular las articulaciones del robot LeArm (físicamente). A su vez, el programa realiza los cálculos para obtener, dada cierta combinación de ángulos en cada articulación, la posición y orientación del efector final.

Se muestra su panel frontal en la figura 80 y su diagrama de bloques la figura 81.



Figura 80 Panel frontal del Programa para la implementación de la cinemática directa.

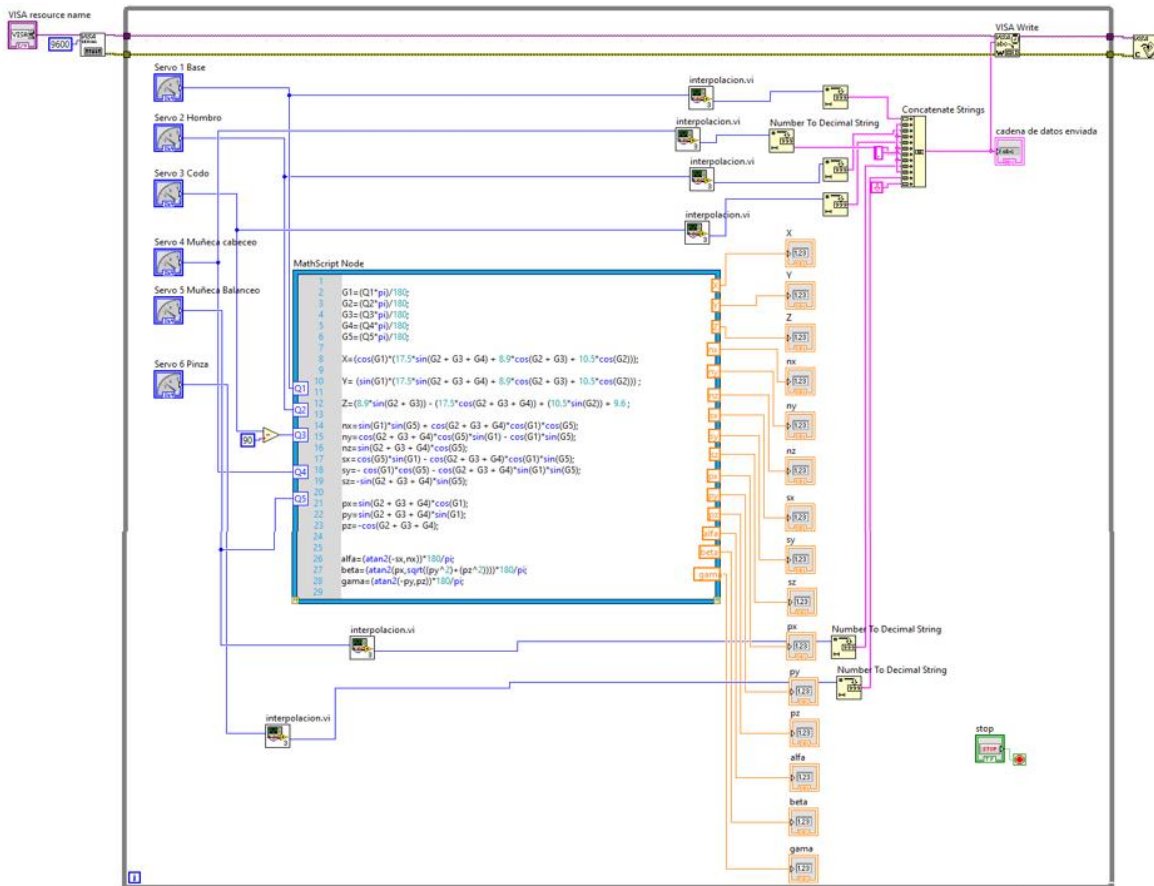


Figura 81 Diagrama de bloques del Programa para la implementación de la cinemática directa.

Entradas

VISA resource name, tiene como objetivo seleccionar el puerto desde el cual se quiere enviar datos de posicionamiento de los servos.

Stop, botón de paro general del programa.

Servo 1 Base, control para el posicionamiento del servomotor de la base del robot con un rango de 0° a 180° .

Servo 2 Hombro, control para el posicionamiento del servomotor del hombro del robot con un rango de 0° a 180° .

Servo 3 Codo, control para el posicionamiento del servomotor del codo del robot con un rango de 0° a 180° .

Servo 4 Muñeca cabeceo, control para el posicionamiento del servomotor de cabeceo de la muñeca del robot con un rango de 0° a 180° .

Servo 5 Muñeca balanceo, control para el posicionamiento del servomotor del balanceo de la muñeca del robot con un rango de 0° a 180° .

Servo 6 Pinza, control para el accionamiento del servomotor que da movimiento a la pinza. El rango va de 90° (máxima abertura de la pinza) a 180° (cierre total de la pinza)

Salidas

Cadena de datos enviada, se visualizan los valores de ancho de pulso enviados para cada servomotor del robot.

Orientación (nx, ny, nz, sx, sy, sz, px, py, pz, alfa, beta, gama), conjunto de datos que representan la orientación del robot con respecto al sistema de referencia de origen. Los primeros 9 datos corresponden a la matriz de orientación obtenida por el método de Denavit-Hartenberg. Los datos alfa, beta y gama, corresponden a los ángulos de Euler, que permiten también conocer la orientación de la pinza con menos datos en comparación con la matriz de orientación. Los elementos “sx”, “sy” y “sz” representan a los elementos “ox”, “oy” y “oz” expresados en la ecuación 8 del capítulo 1. A su vez, los elementos “px”, “py” y “pz” representan a los elementos “ax”, “ay” y “az” expresados en la misma ecuación 8. “nx”, “ny”, “nz” representan a sus símiles de la ecuación 8

Posición (x, y, z), conjunto de datos que permiten conocer la posición en el espacio tridimensional del efector final con respecto al sistema de referencia de origen. Hacen referencia a los elementos “px”, “py” y “pz” de la ecuación 8 que aparece en el primer capítulo.

Otros bloques

Interpolación.VI, subprograma que realiza un mapeo para los valores 0° a 180° y convertirlos a un valor de ancho de pulso con un rango de 500 microsegundos a 2500 microsegundos.

MathScript Node, estructura que hace los cálculos matemáticos de acuerdo a las entradas que se le proporcionan y dar datos de salida relacionados con los datos de entrada.

Number To Decimal String, bloque que convierte datos numéricos en datos tipos String. Estos últimos son el tipo de datos que pueden ser enviados por comunicación serial USB.

VISA Write, Escribe los datos del búfer de escritura en el dispositivo o interfaz especificado por “VISA Resource name”.

VISA Close, Cierra una sesión de dispositivo o un objeto de evento especificado por “Visa resource name”.

While Loop, es una estructura tipo “while” que ejecuta lo que contiene dentro mientras el usuario no detenga la rutina mediante un botón de paro.

Descripción de funcionamiento de programa

De acuerdo al diagrama de bloques, los controles para los servomotores (base, hombro, codo, cabeceo, balanceo) están conectados como entradas a la estructura "Mathscript". Estas son las variables principales para los cálculos de cinemática directa. Dentro de la estructura, se convierten a valores en radianes porque son las unidades en las que se trabaja en el "Mathscript".

A continuación, se realizan los cálculos para cada elemento de la matriz obtenida por el método de Denavit-Hartenberg. Matriz, de la cual se hizo su cálculo anteriormente. En la estructura solamente se escribe la ecuación para obtener cada elemento de la matriz. Adicionalmente, se calculan los ángulos de Euler como una alternativa para conocer la orientación de la herramienta final en función de estos términos.

Se observa, también, que los controles para los servomotores se conectan a un subprograma que convierte el dato de entrada en ángulos y da una salida equivalente en ancho de pulso. A su vez, estos datos se envían al puerto serial para poder manipular el robot físicamente al mismo tiempo que se manipulan los controles del programa.

Nota: Para el tercer servomotor (codo) se observa que se restan 90 grados antes de que ese dato entre a la estructura Mathscript. La razón es porque para efectos geométricos y matemáticos la articulación del codo hace un recorrido de -90° a 90° , pero para efectos prácticos el servomotor opera en un rango que va de 0° a 180° .

Programa para la implementación de la cinemática inversa

Descripción de programa: Este programa es una interfaz de usuario que usa como base la cinemática inversa calculada para el robot LeArm.

Se ingresan las coordenadas de posición y orientación del efector final. Como datos de salida, se obtienen dos posibles combinaciones de coordenadas angulares de los servomotores.

Se puede ingresar los datos de orientación de forma matricial o como la combinación de los ángulos de Euler, para hacerlo solo basta con marcar o desmarcar el botón llamado "Matriz/ángulos Euler". El botón encendido admite los ángulos de Euler y con el botón apagado solo pueden emplearse los datos de la matriz de orientación.

El botón de paro "Stop" detiene la ejecución del programa.

Para las salidas hay dos posibles combinaciones (grupo 1 y grupo 2). Cada grupo cuenta con un led a su costado (T1 y T2) que se encienden cuando la combinación mostrada es válida. Pueden encender ambos, una solo o ninguno.

Los indicadores de punto medio son las coordenadas en el espacio tridimensional de la posición del robot para sus primeros tres grados de libertad sin considerar los grados de la libertad de la muñeca.

Se muestra su panel frontal en la figura 82 y su diagrama de bloques la figura 83.



Figura 82 Panel frontal del Programa para la implementación de la cinemática inversa.

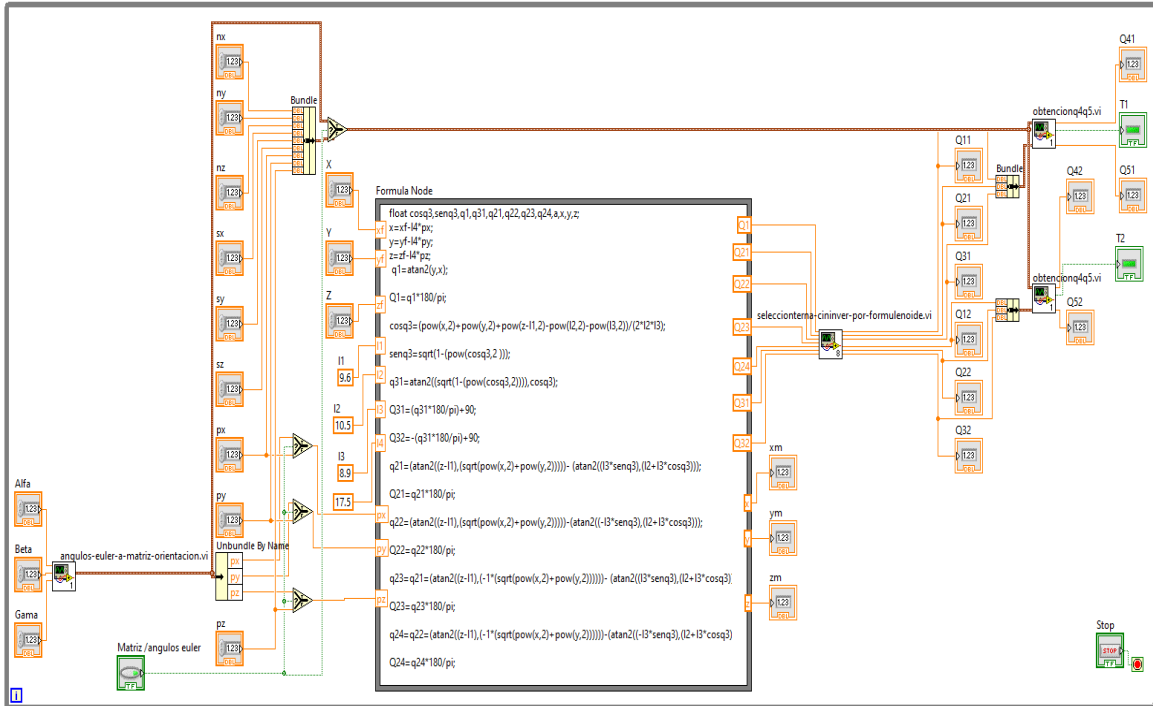


Figura 83 Diagrama de bloques del Programa para la implementación de la cinemática inversa.

Entradas

Posición (X, Y, Z), entradas para ingresar las coordenadas de posición del efector final en el espacio tridimensional con respecto al sistema de referencia de la base del robot.

Orientación, hay dos espacios dedicados a la orientación. Uno es para ingresar los elementos de la matriz de orientación y otro es para introducir los ángulos de Euler. Cualquiera representa la orientación de la herramienta del robot y se deja a criterio y libertad del usuario utilizarlo el que le parezca conveniente.

Matriz orientación: nx, ny, nx, sx, sy, sz, px, py, pz, Entradas de los 9 elementos de la matriz de orientación. Los elementos "sx", "sy" y "sz" representan a los elementos "ox", "oy" y "oz" expresados en la ecuación 8 del capítulo 1. A su vez, los elementos "px", "py" y "pz" representan a los elementos "ax", "ay" y "az". "nx", "ny", "nz" representan a sus símiles de la ecuación 8

Ángulos Euler: Alfa, Beta, Gama, entradas para ingresar los ángulos de Euler.

Matriz/ángulos Euler, botón que permite escoger al usuario entre ingresar los datos de la matriz de orientación o los ángulos de Euler, ambos válidos para representar la orientación de la herramienta final. Apagado es para teclear los datos de la matriz de orientación, encendido es para los ángulos de Euler. Solo puede ser uno u otro a la vez.

Stop, Botón que ejecuta un paro general del programa en caso de activarlo.

Salidas

Grupo 1 y 2, representan a las dos combinaciones posibles de coordenadas articulares que posicionan y orientan a la herramienta final según las entradas dadas por el usuario. Se visualiza el número “-666” cuando no existe un ángulo para los datos ingresados.

T1 y T2, son dos leds que se encuentran a lado de los grupos de posibles combinaciones de coordenadas articulares. Prenden de color verde cuando una combinación es correcta. Pueden iluminarse uno, ambos o ninguno.

Posición punto medio xm, ym, zm, coordenadas de posición para los tres primeros tres grados de libertad sin considerar los grados de libertad dedicados a orientación.

Otros bloques

ángulos-Euler-a-matriz-orientacion.vi, es el **subprograma para convertir los ángulos de Euler a matriz de orientación.**

Bundle, ensambla un clúster a partir de elementos individuales. Un cluster agrupa elementos del mismo o diferentes tipos. De esta manera, al haber varios cables de diferentes elementos podemos unir en un solo cable para tener más orden en nuestro panel de diagrama de bloques.

Unbundle by name, Devuelve los elementos del clúster cuyos nombres se especifique.

Formula node, estructura basado en texto que se usa para realizar operaciones matemáticas. Se puede hacer uso de declaraciones “if”, “ciclos while”, “ciclos for” y “ciclos do”, similares a la programación en “C”, pero la programación no es necesariamente idéntica

Seleccionterna-cininv-por-formulenoide.vi, nombre del **subprograma para seleccionar las ternas de las variables articulares q1,q2,q3**. Mediante un algoritmo selecciona que combinaciones que coordenadas son válidas. Por ejemplo, si no cumplen con el rango de operación de los servomotores (0° a 180°), se descartan tales valores para formar ternas. Son ternas porque son los valores para los tres primeros grados de libertad.

Obtencionq4q5.vi, nombre del **subprograma para obtener las variables articulares q4 y q5**. Se basa en las ecuaciones obtenidas con anterioridad en el análisis de desacoplo cinemático.

Select, Retorna el valor conectado a la entrada t o a la entrada f, según el valor de s. Si s es VERDADERO, esta función devuelve el valor conectado a t. Si s es

FALSO, esta función devuelve el valor conectado a f. En este programa se usa un botón “matriz/ángulos Euler” para determinar cuando el bloque “Select usa, ya sea los datos convertidos a partir de los ángulos de Euler en matriz de orientación o la matriz de orientación.

Descripción de funcionamiento de programa

Se tiene en medio del diagrama de bloques a la estructura “Formule node”. En esta se hacen los cálculos para obtener las coordenadas articulares de los tres primeros grados de libertad. Para conseguirlas, se necesita de la posición final de la herramienta final, así como de la primera fila de la matriz de orientación para calcular el punto medio

A partir de estos datos se hace el cálculo de 7 valores de coordenadas angulares, uno para la primera articulación, cuatro la segunda y dos la tercera. Estos valores se usarán más adelante.

El usuario ingresa como datos, la posición final y la orientación. La orientación puede ser en forma de matriz de orientación o ángulos de Euler. En el diagrama de bloques se observa las conexiones necesarias para que se lleve sin problema la ejecución del programa sin importar la forma de orientación se escoja. Los ángulos de Euler necesariamente deben convertirse a matriz de orientación porque son estos elementos de la matriz los que se emplean para todos los cálculos. A través del bloque “select” se pueden seleccionar que datos utilizar dependiendo si el botón “matriz/ángulos Euler” está presionado o no.

Como salidas de la estructura “formule node” tenemos los siete valores de variables articulares y tres valores para la posición del punto medio.

Los siete valores articulares entran al **subprograma para seleccionar las ternas de las variables articulares q1,q2,q3**, donde, mediante un algoritmo, se seleccionan las ternas o valores de los tres primeros grados de libertad. Estas ternas pueden o no existir. A veces son dos ternas, otras, solo es válida una.

Las dos ternas obtenidas entran como datos al **subprograma para obtener las variables articulares q4 y q5**, donde se realizan los cálculos de desacople cinemático para obtener las variables articulares de la muñeca. Solo entra una terna a la vez, por lo que se usa dos veces este subprograma. Además, se emplean valores de la matriz de orientación porque las ecuaciones usadas también dependen de estos términos.

Como salidas del **subprograma para obtener las variables articulares q4 y q5**, se adquieren dos pares de valores que se combinan con las ternas anteriormente obtenidas y forman dos grupos posibles de combinaciones que posicionan y orientan la herramienta del robot de acuerdo a las entradas dadas por el usuario. Se tiene también como salida los dos leds T1 y T2 que indican cuando un grupo es válido.

Programa para convertir ángulos de Euler a matriz de orientación

Descripción de programa: Programa que calcula los elementos de la matriz de orientación a partir de los ángulos de Euler. Los elementos se unen en un “cluster” para tener agrupados estos datos. Las ecuaciones escritas en la estructura “Mathscript node” provienen de los cálculos hechos en el capítulo tres y que tienen como resultado las ecuaciones (112), (113), (114), (115), (116), (117), (118), (119) y (120).

Se muestra su panel frontal en la figura 84 y su diagrama de bloques la figura 85.0

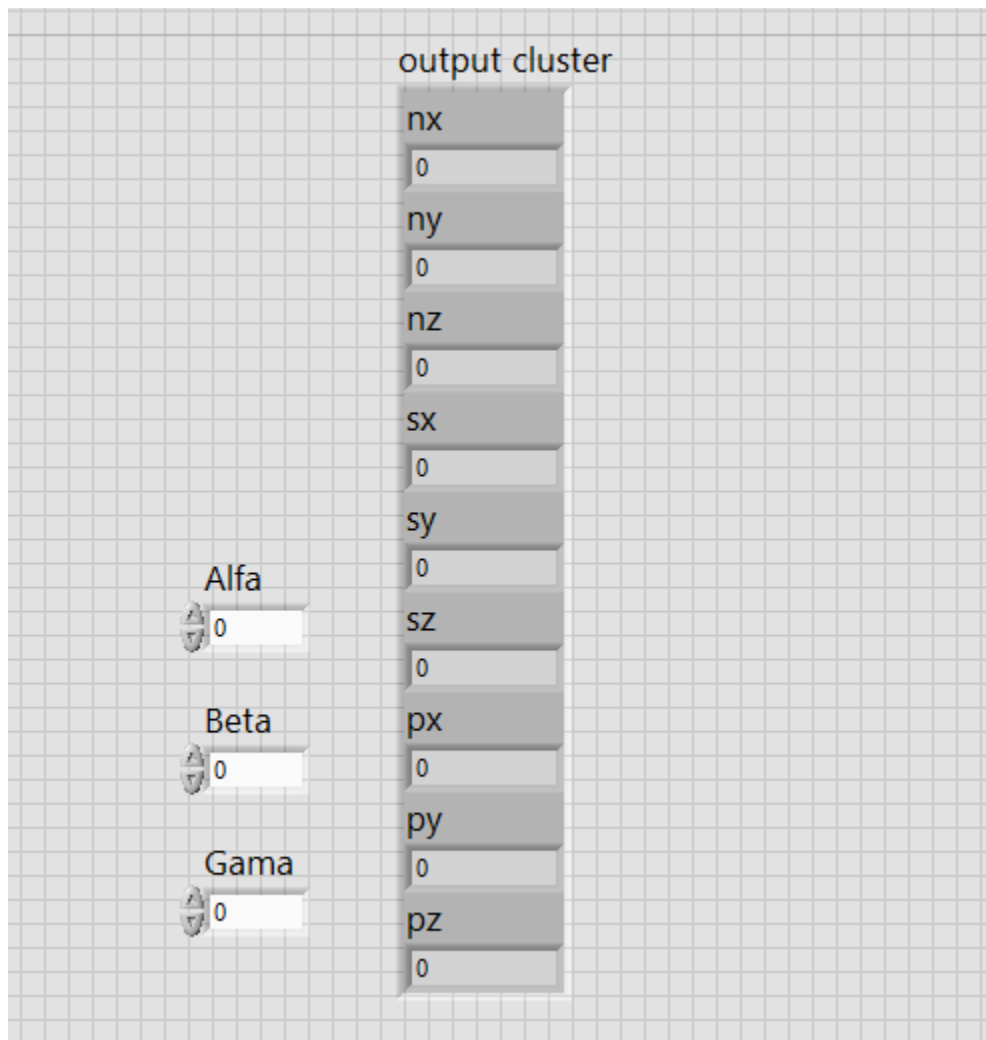


Figura 84 Panel frontal del Programa para convertir ángulos de Euler a matriz de orientación.

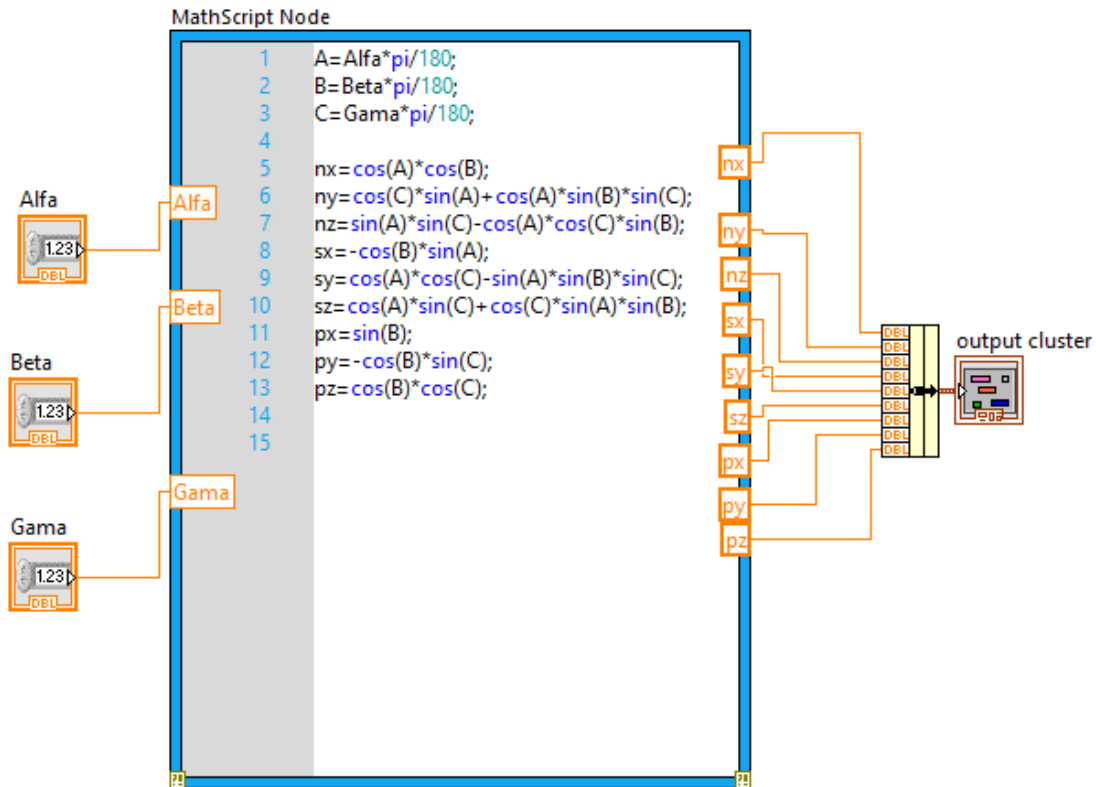


Figura 85 Diagrama de bloques del Programa para convertir ángulos de Euler a matriz de orientación.

Entradas

Alfa, Beta, Gama, Ángulos de Euler

Salidas

nx, ny, nz, sx, sy, sz, px, py, pz, elementos de la matriz de orientación. Los elementos 'sx', 'sy' y 'sz' representan a los elementos 'ox', 'oy' y 'oz' expresados en la ecuación 8 del capítulo 1. A su vez, los elementos 'px', 'py' y 'pz' representan a los elementos 'ax', 'ay' y 'az', 'nx', 'ny', 'nz' representan a sus símiles de la ecuación 8

Output cluster, agrupa los 9 elementos de la matriz de orientación en un solo elemento para manejarlos ordenadamente.

Otros bloques

Bundle, bloque que permite agrupar varios elementos en un "cluster". En este caso se agrupan los elementos de la matriz de orientación.

Descripción de funcionamiento de programa

El programa recibe como entradas los ángulos Euler y obtiene como salida un cluster con los 9 elementos de la matriz de orientación. Los cálculos se hacen en una estructura "mathscript". El cálculo se hace de cada elemento de la matriz de orientación para evitar hacer el cálculo de la matriz desde cero, lo cual puede verse afectado en el tiempo de ejecución del programa. El cálculo de la matriz de orientación es un cálculo hecho en el capítulo 3 y expresado en las ecuaciones 111 a 120.

Programa para obtener las variables articulares q4 y q5

Descripción de programa: Se tiene como objetivo obtener las dos variables articulares asociadas a la muñeca del robot (q4 y q5) a partir de los datos de orientación y los de las tres primeras variables articulares obtenidas en el método geométrico. Además, se agrega un algoritmo para indicar a través de un led si la combinación de variables articulares es un conjunto de datos válido para el rango de operación de cada uno de los servomotores del robot.

Se muestra su panel frontal en la figura 86 y su diagrama de bloques la figura 87.

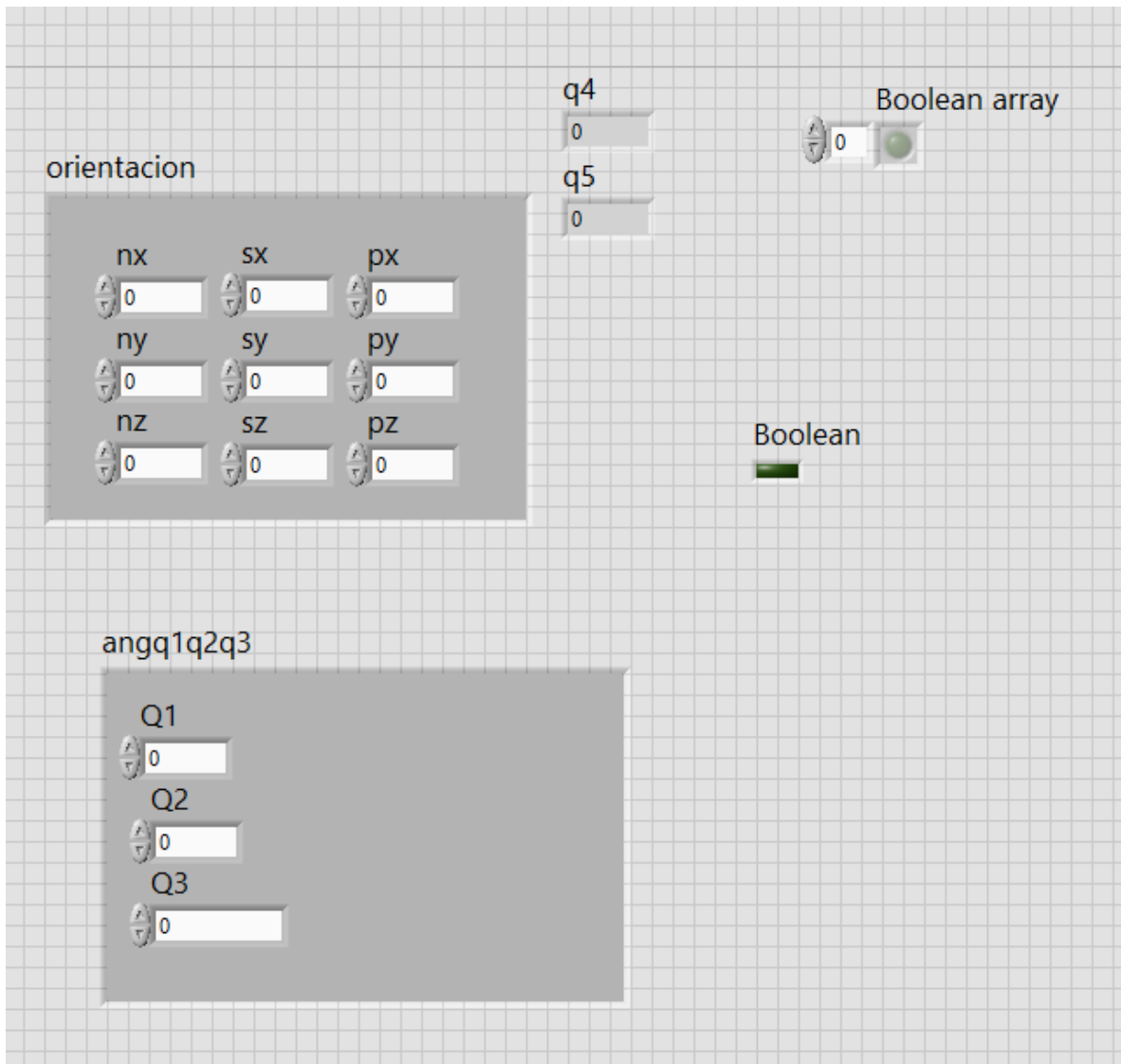


Figura 86 Panel frontal del Programa para obtener las variables articulares q4 y q5.

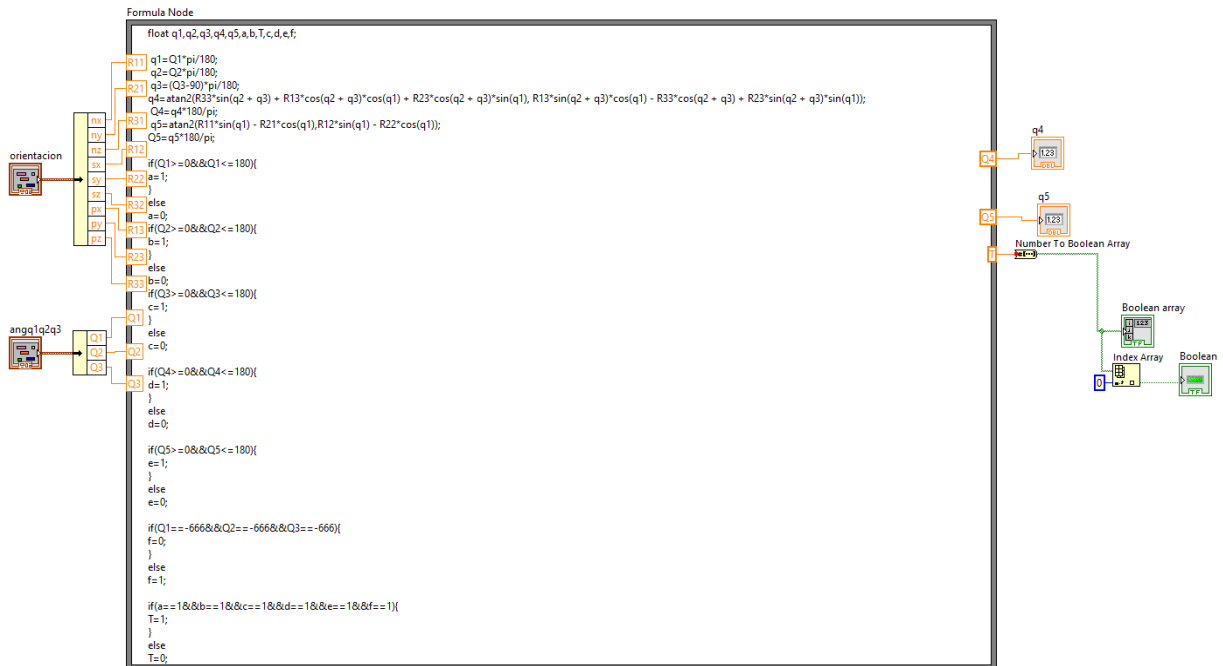


Figura 87 diagramas de bloques del Programa para obtener las variables articulares q_4 y q_5 .

Entradas

Orientación, elementos de la matriz de orientación.

Angq1q2q3, elementos de las primeras tres variables articulares para los tres primeros grados de libertad.

Salidas

q4,q5, variables articulares para los dos grados de libertad de la muñeca del robot bajo estudio.

Boolean array, arreglo de elementos booleanos, resultado de usar el bloque que convierte número a booleano (Number to boolean array). Solo tiene un elemento que representa a la variable T que se usa dentro de la estructura "Formule node".

Boolean, variable booleana en su forma de led, que indica cuando un grupo de variables articulares es válido para el rango de operación de cada uno de los servomotores.

Otros bloques

Index Array, bloque que sirve para extraer un elemento de un arreglo. Se busca extraer el valor del único del elemento del arreglo "Boolean Array", elemento "0".

Descripción de funcionamiento de programa

Se tienen como entradas “orientación” que son los elementos de la matriz de orientación y “angq1q2q3” asociados a las variables articulares para los tres primeros grados de libertad. Estos datos son necesarios porque el cálculo de q4 y q5 está en función de estas variables.

Dentro de la estructura “Formule node” se hacen los cálculos de las variables articulares restantes, las expresiones ya han sido calculadas con anterioridad y solo se sigue las ecuaciones obtenidas (ecuaciones 94 y 95)

Se crea un pequeño algoritmo en el cual se comprueba que cada variable articular este dentro del rango de operación de los servomotores, es decir, de 0 a 180 grados.

El algoritmo consiste en evaluar cada una de las variables articulares obtenidas para verificar que estén dentro del rango. Se usa una variable auxiliar en todos para la evaluación de un caso “if”. Esto se consigue asignando un “1” o “0” (verdadero, falso, respectivamente) a la variable auxiliar.

Se tiene un caso en que se evalúa si las variables articulares q1, q2,q3 tienen el valor “-666” que fue asignado con otro algoritmo, para indicar que no existen variables articulares que satisfagan criterios del punto medio como posición final. Se observa con más detalle en la descripción del **programa para seleccionar las ternas de variables articulares q1,q2,q3**”.

Programa para seleccionar las ternas de variables articulares q1,q2,q3

Descripción de programa: Se implementa un algoritmo que permite formar ternas a partir de los valores obtenidos por el cálculo de las variables articulares q1,q2,q3. Los cálculos arrojan un valor para la variable articular “q1”,se obtienen cuatro valores para la variable articular “q2” y dos para la variable “q3”. Con estos datos se forman ternas en las que únicamente podrán ser válidas las combinaciones posibles físicamente para los rangos de operación de los servomotores. Se puede obtener hasta dos ternas válidas, algunas veces se obtiene una sola y en ocasiones ninguna.

Se muestra su panel frontal en la figura 88 y su diagrama de bloques la figura 89.

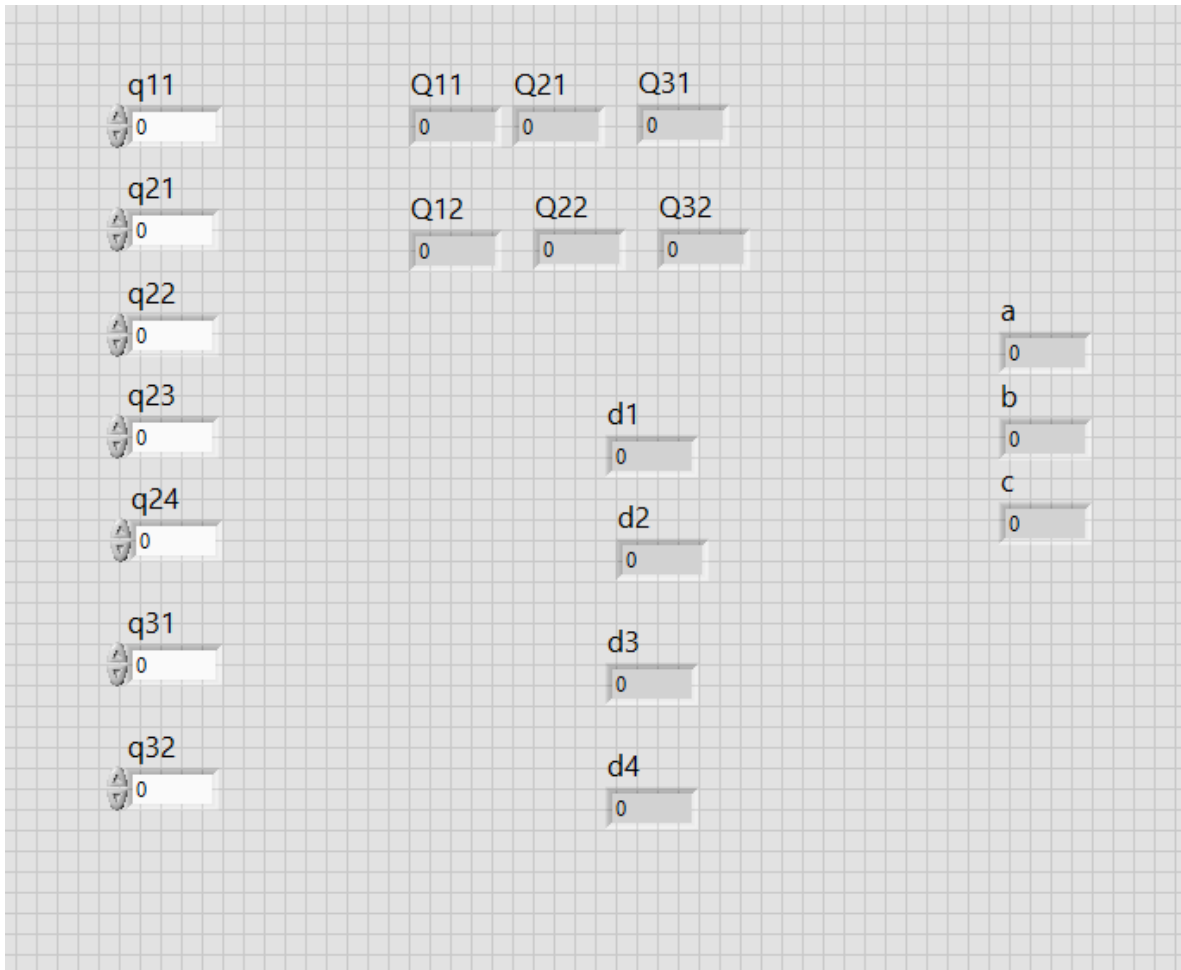


Figura 88 Panel frontal del Programa para seleccionar las ternas de variables articulares q1, q2, q3.

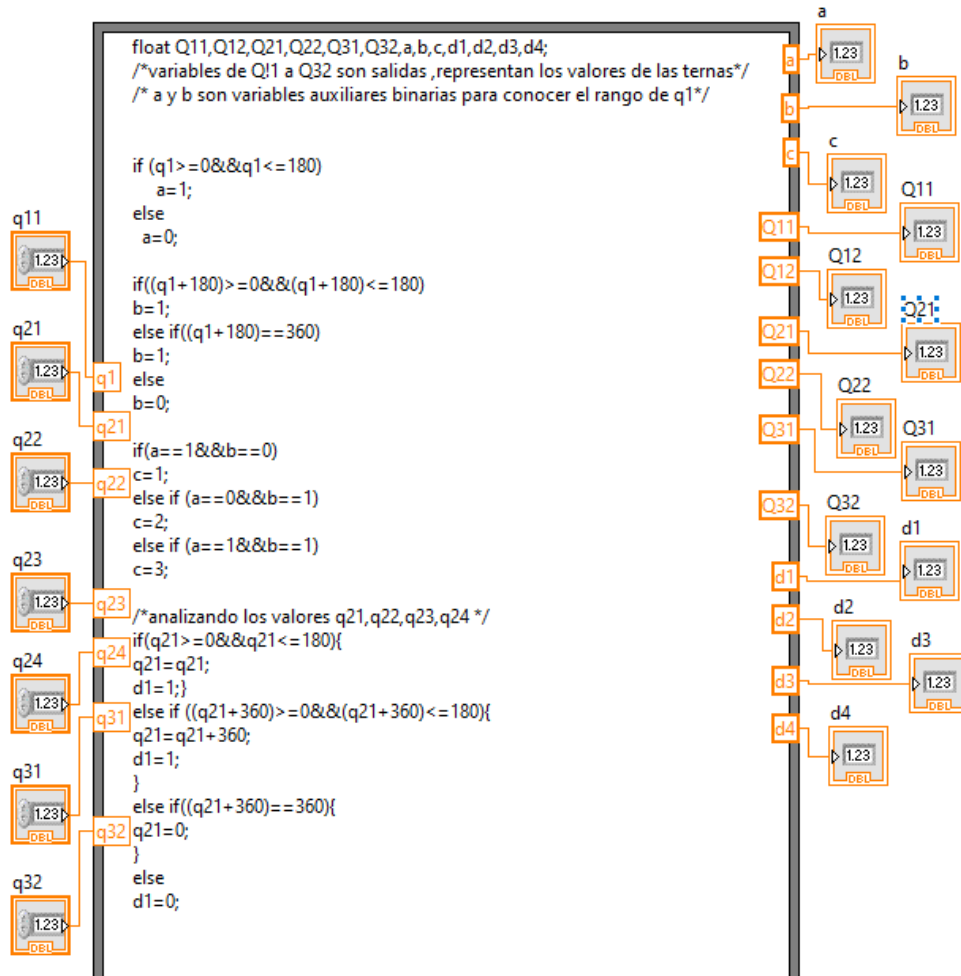


Figura 89 Diagrama de bloques del Programa para seleccionar las ternas de variables articulares q_1 , q_2 , q_3 .

Nota: El algoritmo completo se encuentra en el anexo de este documento.

Entradas

q_{11} , q_{21} , q_{22} , q_{23} , q_{24} , q_{31} , q_{32} , representan a cada uno de los valores de las variables articulares calculadas por el método geométrico.

Salidas

a, variable auxiliar que indica con un “1” si la variable articular q_1 se encuentra en un rango de 0° a 180° . En caso contrario, la variable auxiliar se le asigna el valor “0”.

b, Variable auxiliar que indica si la variable articular q_1 pertenece al rango de 0° a 180° una vez que se le suma a “ q_1 ”, 180 grados.

c, variable auxiliar que indica 3 posibles casos, un primer caso ($c=1$) cuando los valores de “a” y “b” son “1” y “0”, respectivamente. Un segundo caso ($c=2$) si los

valores de “a” y “b” son “0” y “1” respectivamente. Un tercer caso ($c=3$), si los valores de “a” y “b” son “1” en ambos casos.

d1, variable auxiliar que indica que la variable articular q_{21} pertenece al rango de 0° a 180° . Cuando es verdadero, la variable auxiliar toma un valor de “1”, caso contrario el valor es “0”

d2, variable auxiliar que indica que la variable articular q_{22} pertenece al rango de 0° a 180° . Cuando es verdadero, la variable auxiliar toma un valor de “1”, caso contrario el valor es “0”

d3, variable auxiliar que indica que la variable articular q_{23} pertenece al rango de 0° a 180° . Cuando es verdadero, la variable auxiliar toma un valor de “1”, caso contrario el valor es “0”

d4, variable auxiliar que indica que la variable articular q_{24} pertenece al rango de 0° a 180° . Cuando es verdadero, la variable auxiliar toma un valor de “1”, caso contrario el valor es “0”

Descripción de funcionamiento de programa

Los valores $q_1, q_{21}, q_{22}, q_{23}, q_{24}, q_{31}, q_{32}$ son las entradas de la estructura “formule node”. Se busca formar dos ternas posibles con los datos de entrada. Dentro de “formule node” se implementa un algoritmo que construye las ternas. El algoritmo se detalla en el **anexo** de este trabajo en el apartado **“Algoritmo para obtener ternas de las variables articulares q_1, q_2, q_3 en la cinemática directa implementada para el robot LeArm de 5GDL”**.

Como salidas se obtienen dos ternas posibles. Para indicar que alguna de las ternas es inválida, el algoritmo asigna un valor físicamente imposible para los servomotores. Se deja claro que se trata de un valor no válido, este es “-666”.

Capítulo 5 Pruebas y resultados

Las pruebas experimentales se han llevado a cabo bajo condiciones de confinamiento (debidas al COVID-19), es decir, no se ha contado con un espacio de laboratorio. Tampoco se dispuso de herramientas, equipo e instrumentos adecuados para realizar las pruebas en entornos controlados. En este sentido, se presentan las pruebas y resultados obtenidos, mismos que se consideran satisfactorios.

Las pruebas realizadas se enfocan en validar y verificar los siguientes aspectos:

- Repetitividad con rutinas secuenciales.
- Simulación de la cinemática directa y modelo 3D del robot manipulador.
- Implementación de rutina secuencial para recoger y dejar pelotas
- Validación de la cinemática inversa.
- Implementación de rutina para dibujar figuras geométricas y letras.

Para ello, se hace uso de los programas creados y explicados en el capítulo 4, así como otras herramientas de software para la validación de algunos algoritmos.

Repetitividad con rutinas secuenciales

El objetivo de esta primera prueba es analizar la repetitividad del posicionamiento del efector final del robot. Es decir, si se implementa una rutina secuencial y se quiere posicionar y orientar la herramienta del robot manipulador, se debe verificar que la herramienta del robot siempre se ubica en el mismo punto del espacio tridimensional.

Para esta prueba se adaptó una punta fina al final del efector final. Se programó una rutina secuencial realizada mediante el programa para crear una rutina secuencial, mostrada en la figura 90.



Figura 90 Creación de rutina secuencial con ayuda del programa para crear rutina secuencial.

Los datos con los que se creó la rutina secuencial se muestran en la tabla 5.

Cada fila de la tabla representa los valores angulares para que los servomotores del robot se posicionen en un momento determinado de la rutina secuencial. Son 8 filas, por tanto, son 8 eventos principales de la rutina secuencial que posicionan a los servomotores del robot LeArm.

Tabla 5 Datos de la rutina secuencial creada con 8 pasos que tiene valores para cada articulación del robot.

Evento	1er Servo Base [°]	2do Servo Hombro [°]	3er Servo codo [°]	4to Servo Muñeca [°]	5to Servo giro pinza [°]	6to servo pinza [°]
	180	90	90	90	90	180
	180	90	0	90	90	90
	180	90	0	0	180	90
	180	180	0	45	135	90
	180	90	45	45	90	90
	180	90	63	45	90	90
	180	90	63	45	71	90
	180	90	72	72	90	90

1	180	90	90	90	90	180
2	180	90	0	90	90	90
3	180	90	0	0	180	90
4	180	180	0	45	135	90
5	180	90	45	45	90	90
6	180	90	63	45	90	90
7	180	90	63	45	71	90
8	180	90	72	72	90	90

Los datos de la tabla generada con el programa para crear una rutina secuencial se almacenaron en un archivo con extensión “.cvs”, dada por el mismo programa. Este archivo “.cvs” se ejecutó con el programa para ejecutar rutina secuencial y de esta manera se pudo observar implementada físicamente en el robot LeArm.

Esta rutina puso al robot en una posición vertical al inicio de la rutina. El robot realizó otros movimientos necesarios para alcanzar una hoja que se encontraba frente a este. La herramienta del robot se posicionó sobre la hoja milimétrica con la puntilla con tinta que dejó una marca donde se posicionó.

La rutina se ejecutó de forma reiterada, tratando de cuidar las condiciones físicas iniciales durante el proceso de prueba. De esta manera se creó una secuencia con los mismos movimiento para llegar a una posición una y otra vez, como se observa en la figura 91.

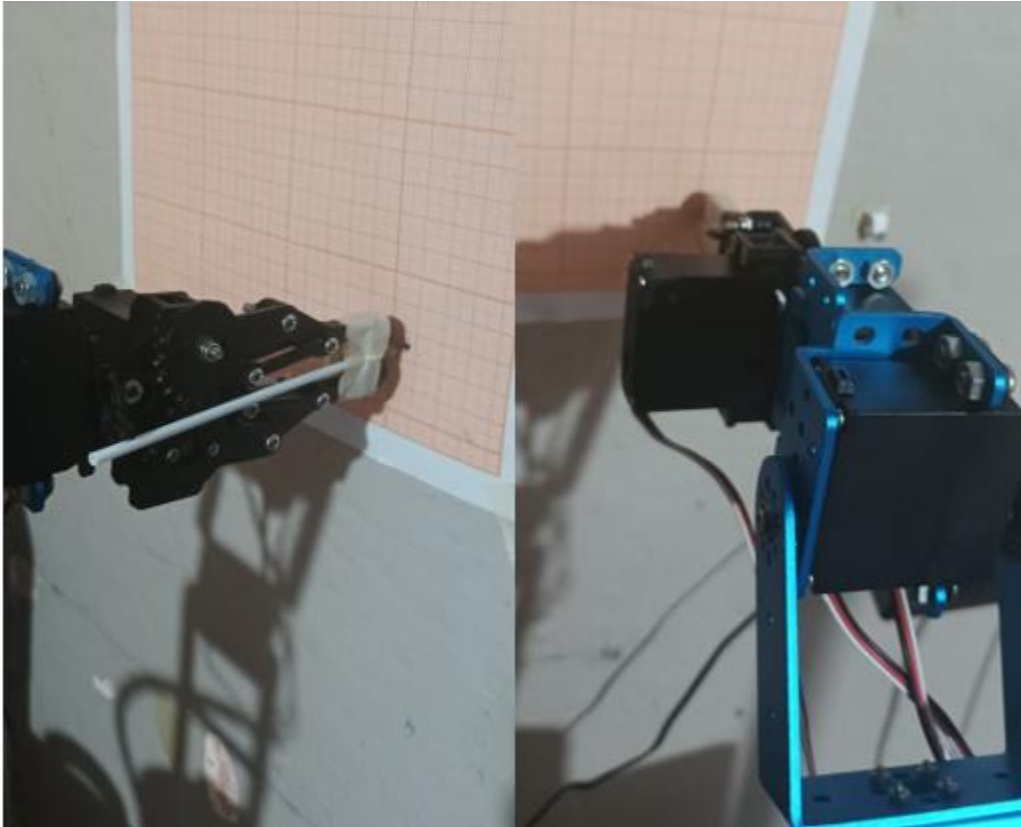


Figura 91 Robot LeArm ejecutando una rutina secuencial y dejando a su paso un patrón.

Con esta prueba se obtuvo un patrón de repetitividad creado por cada punto marcado en cada iteración en una hoja de papel en una pared. La rutina se ejecutó durante un tiempo de quince minutos. Al final de la rutina se observó un patrón ovalado delimitado por una altura de 1 [cm] por 0.5 [cm] de ancho.

Se repitió el proceso tres veces en ciclos de 15 minutos. En los patrones conseguidos se observó que cada uno fue coincidente con los demás y se pudo comprobar que la repetición del robot al posicionar el efector final no variaba más allá de 1.6 [cm] de altura por 0.6 [cm] de ancho, en cada prueba.

.Se muestran los patrones obtenidos en la figura 92.



Figura 92 Patrones dibujados por el robot LeArm en tres pruebas distintas.

Se puede observar en los patrones que aunque la herramienta no siempre se posicionó en el mismo punto del espacio, si se encuentra dentro de cierto rango de medición, en el área que se expone en los patrones.

Después de realizar observaciones en la estructura del robot, se observó que hay un huelgo en la cuarta articulación del robot manipulador. Este huelgo se da por razones del peso de la estructura y el diseño físico del robot manipulador, que no permite que el robot se mantenga donde marca el servomotor de acuerdo con las señales que le son transmitidas. Por esa razón se observa en los patrones que hay una diferencia en cuanto a la altura de la “mancha”, que es casi el doble que del ancho.

Esta prueba muestra que el robot es capaz de seguir una secuencia y moverse de acuerdo a las instrucciones dadas a sus servomotores. El error en su estructura es ajeno al buen desempeño de sus actuadores. Con esta prueba realizada, se demuestra que el robot es capaz de llevar a cabo tareas más complejas.

Simulación de la cinemática directa y modelo 3D del robot manipulador

Se planteó hacer una simulación de la cinemática inversa en un software matemático (MATLAB). Mediante un “Toolbox”, que es un paquete de funciones y herramientas sobre un tema en específico, se resolvió la cinemática directa y se simuló el modelo del robot antropomórfico con cinco grados de libertad.

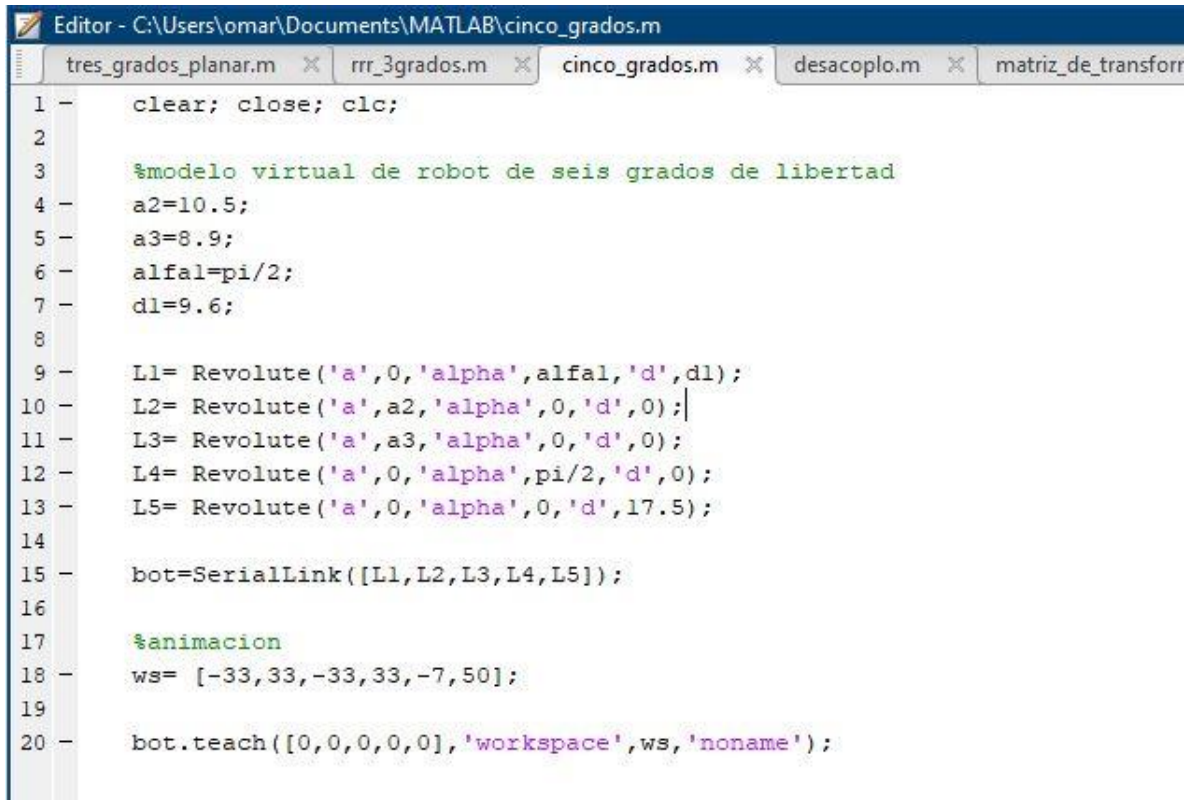
La herramienta o “Toolbox” se llama “Robotics Toolbox” creada por Peter Corke profesor en la Universidad Tecnológica de Queensland.

La herramienta es abierta para su uso, disponible para descargar en la página del profesor Peter Corke [4].

“Robotics Toolbox” permite simular y aplicar modelos de robots, la cinemática, la dinámica, creación de trayectorias, entre otras herramientas.

Se ha elegido esta herramienta que ya ha sido probada y validada, para verificar que los cálculos y resultados del algoritmo del **programa para la implementación de la cinemática directa**, son igualmente válidos y correctos.

El código programado para la simulación de la cinemática directa del robot antropomórfico bajo estudio se muestra en la figura 93.



```
Editor - C:\Users\omar\Documents\MATLAB\cinco_grados.m
tres_grados_planar.m x rrr_3grados.m x cinco_grados.m x desacoplo.m x matriz_de_transfor
1 - clear; close; clc;
2
3 %modelo virtual de robot de seis grados de libertad
4 - a2=10.5;
5 - a3=8.9;
6 - alfa1=pi/2;
7 - d1=9.6;
8
9 - L1= Revolute('a',0,'alpha',alfa1,'d',d1);
10 - L2= Revolute('a',a2,'alpha',0,'d',0);
11 - L3= Revolute('a',a3,'alpha',0,'d',0);
12 - L4= Revolute('a',0,'alpha',pi/2,'d',0);
13 - L5= Revolute('a',0,'alpha',0,'d',17.5);
14
15 - bot=SerialLink([L1,L2,L3,L4,L5]);
16
17 %animacion
18 - ws= [-33,33,-33,33,-7,50];
19
20 - bot.teach([0,0,0,0,0],'workspace',ws,'noname');
```

Figura 93 Código para emular robot antropomórfico de cinco grados de libertad en la plataforma Matlab.

La explicación del código se desarrolla en el anexo. El resultado de la simulación al ejecutarlo en la figura 94.

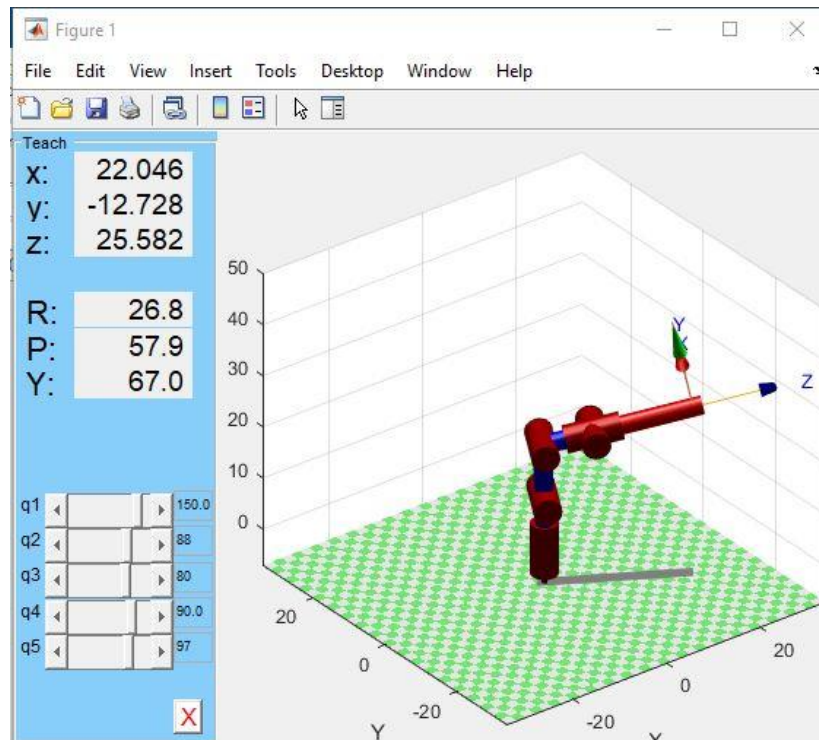


Figura 94 Simulación gráfica que se obtiene al ejecutar el código para simular un robot antropomórfico de cinco grados de libertad.

En la figura 94 se observa la simulación del modelo 3D del robot antropomórfico de cinco grados de libertad. El modelo muestra un espacio tridimensional donde se señalan los ejes “x, y, z” que son los ejes del sistema de origen. A su vez, la herramienta contiene el sistema de ejes de la herramienta del robot. Los ejes de la herramienta tienen su propia orientación respecto al eje de origen.

Del lado izquierdo, en la figura 94 se muestran la posición (x, y, z) y la orientación (R, P, Y) de la herramienta final. “R, P, Y” corresponden a los ángulos de Euler (α , β , γ), respectivamente.

Abajo y a la izquierda se muestran las variables articulares “q1, q2, q3, q4, q5”. Estas variables son datos que se pueden manipular y modificar.

Hay que aclarar que para la variable articular “q3” su rango de movimiento de $[-90^\circ, 90^\circ]$ solo para esta simulación, mientras que físicamente la articulación correspondiente varía en un rango de $[0^\circ, 180^\circ]$. Estos rangos son equivalentes y solo hay que considerar en qué rango varía cada caso, simulación o robot físico.

Se muestran las comparaciones de la ejecución del **programa para la implementación de la cinemática directa** y la simulación en el software de Peter Corke. En cada ejecución, tanto del programa como de la simulación, se introdujeron los datos de las variables articulares como entradas y se obtuvieron como salidas la orientación y posición de la herramienta final.

Además, se muestra la configuración que toma físicamente el robot manipulador al que se envían en tiempo real las entradas o ángulos que posicionan a sus servomotores. De esta manera se compara la configuración tras aplicar las entradas en la simulación, con el mismo procedimiento, pero aplicado al robot físico.

En las imágenes 95, 96, 97, 98, 99, 100, 101, 102 y 103 se exponen los resultados para tres pruebas realizadas, aunque se realizaron más. Estas muestran una ejemplificación de la prueba llevada a cabo.

Las figuras 95, 96 y 97 corresponden a una primera prueba; las figuras 98, 99 y 100 a una segunda prueba; y las figuras 101, 102 y 103 a la tercera prueba.

Primera prueba



Figura 95 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la primera prueba.

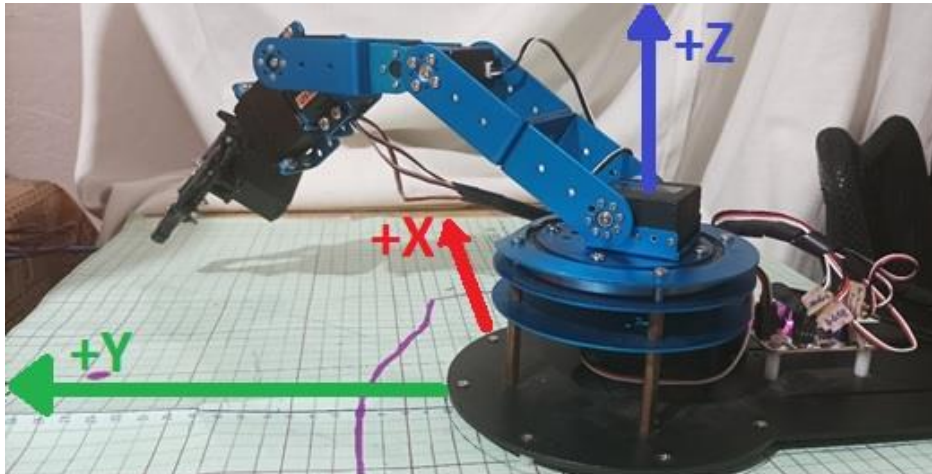


Figura 96 Configuración física que se obtiene al ingresar los datos mostrados en la figura 95 (primera prueba).

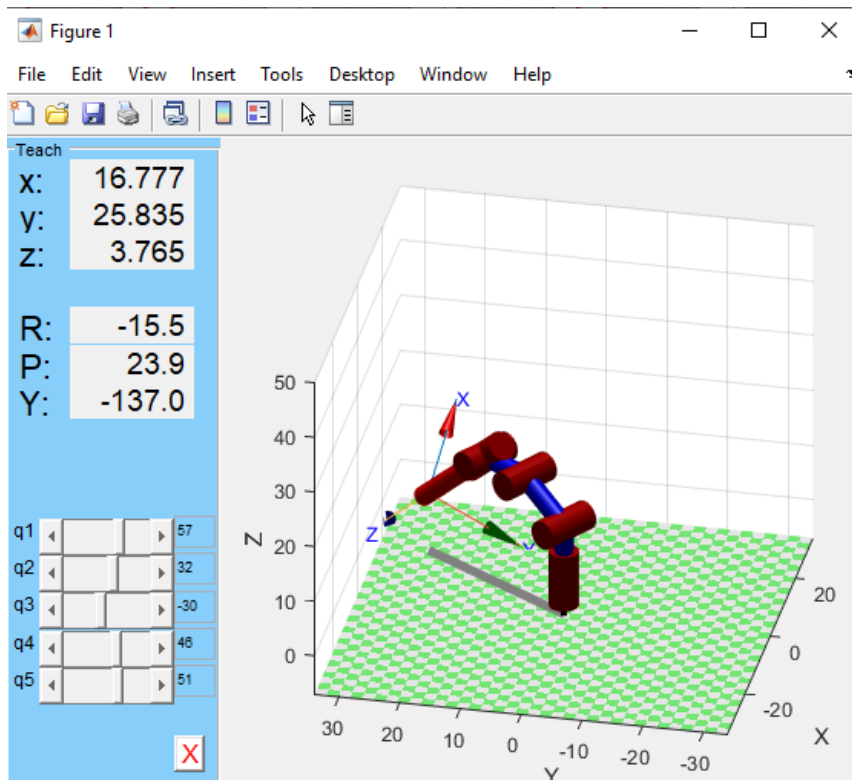


Figura 97 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 95 (primera prueba).

Segunda prueba



Figura 98 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la segunda prueba.

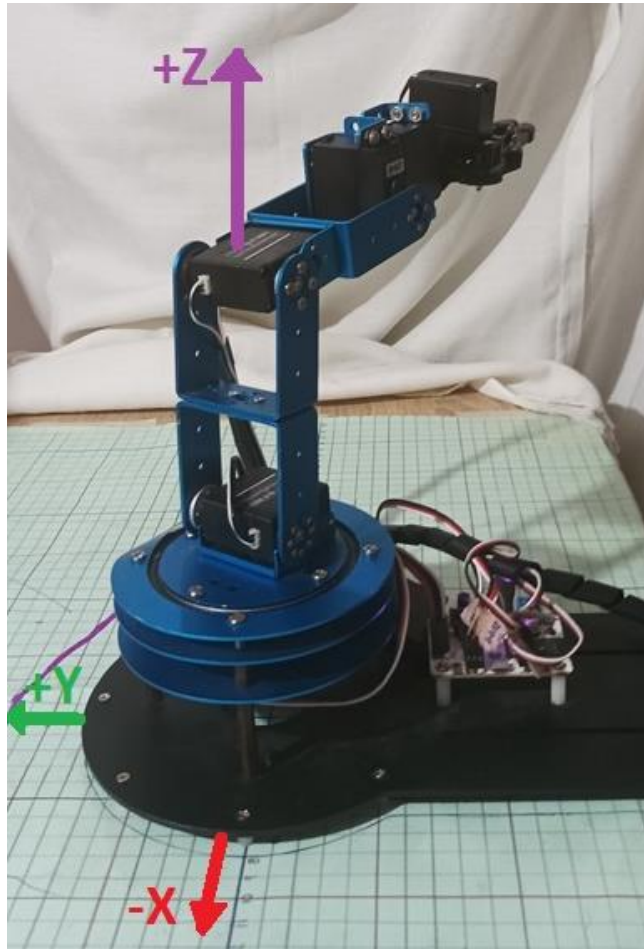


Figura 99 Configuración física que se obtiene al ingresar los datos mostrados en la figura 98 (segunda prueba).

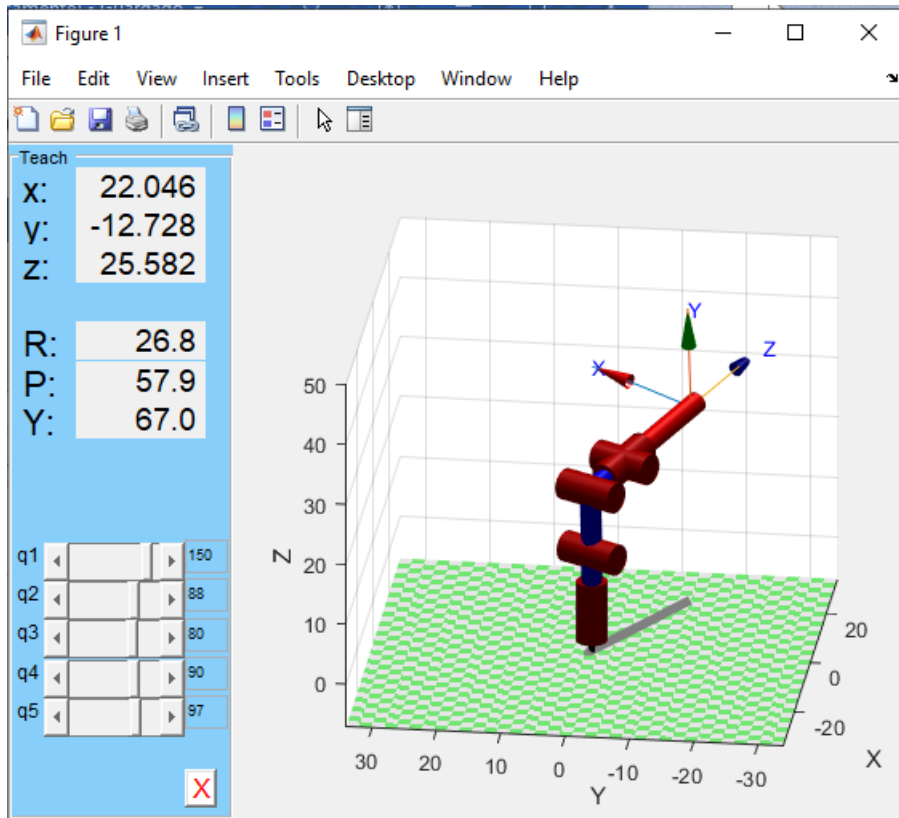


Figura 100 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 98 (segunda prueba).

Tercera prueba



Figura 101 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la tercera prueba.

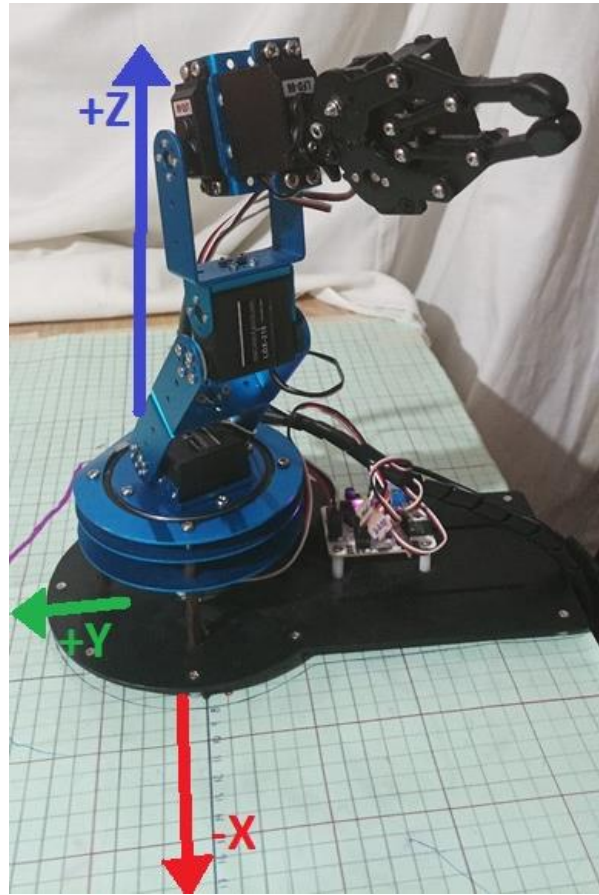


Figura 102 Configuración física que se obtiene al ingresar los datos mostrados en la figura 101 (tercera prueba).

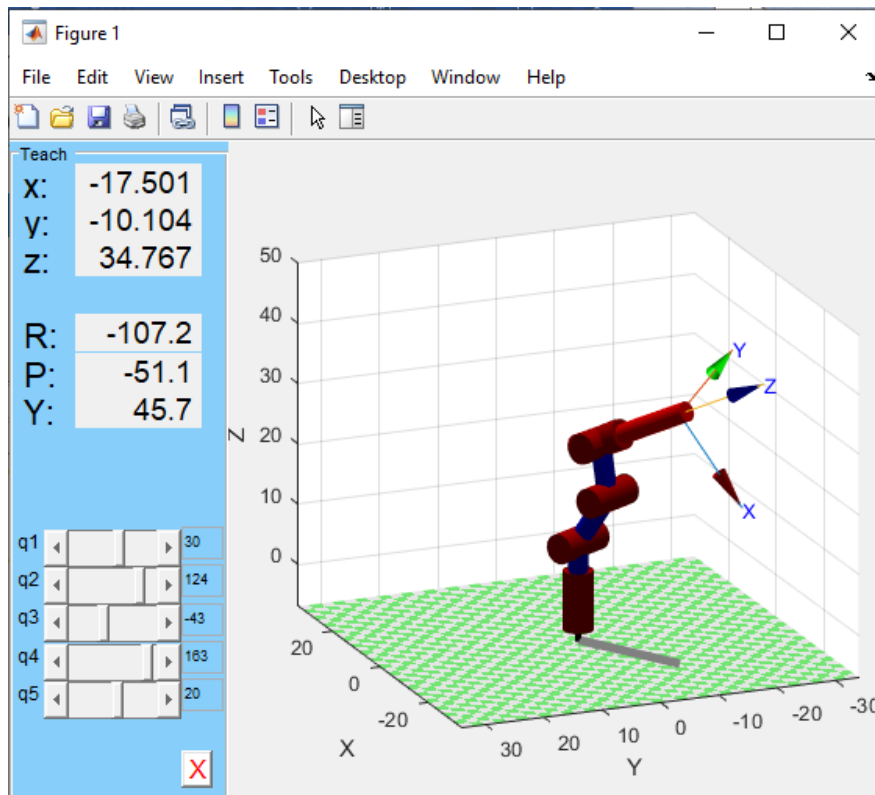


Figura 103 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 101 (tercera prueba).

Comparando los resultados tal como se observa en las figuras 95, 96, 97, 98, 99, 100, 101, 102 y 103, se puede constatar que el **programa para la implementación de la cinemática directa** es satisfactorio. Los resultados obtenidos son iguales tanto en el programa como en la simulación y en la ejecución del robot manipulador físico.

Implementación de rutina secuencial para recoger y dejar pelotas

Para implementar la cinemática directa y poner a prueba la capacidad del robot LeArm para realizar tareas, se implementó una rutina secuencial a partir de los datos obtenidos por la cinemática directa.

La tarea consistió en una rutina secuencial en la que el robot manipulador recoge, con ayuda de su herramienta de trabajo (pinza), pelotas colocadas en su espacio de trabajo para después para colocarlas en un contenedor.

El robot se montó sobre una cartulina la cual tenía trazada la silueta de la base sobre la que se montó el robot, haciendo coincidir el centro del sistema base con el centro de sistema de coordenadas dibujado. Este sistema es un plano "XY" y el eje "Z", aunque no se muestra, es perpendicular a la cartulina.

El sistema de coordenadas dibujado tiene trazos cada centímetro, permitiendo localizar coordenadas en el plano "XY" con un centímetro de precisión. Para medir

el eje "Z" se hace de manera manual apoyándose de un flexómetro midiendo desde el plano "XY" de manera perpendicular respecto a la cartulina.

Se ejemplifica el sistema "XY" dibujado en la figura 104.

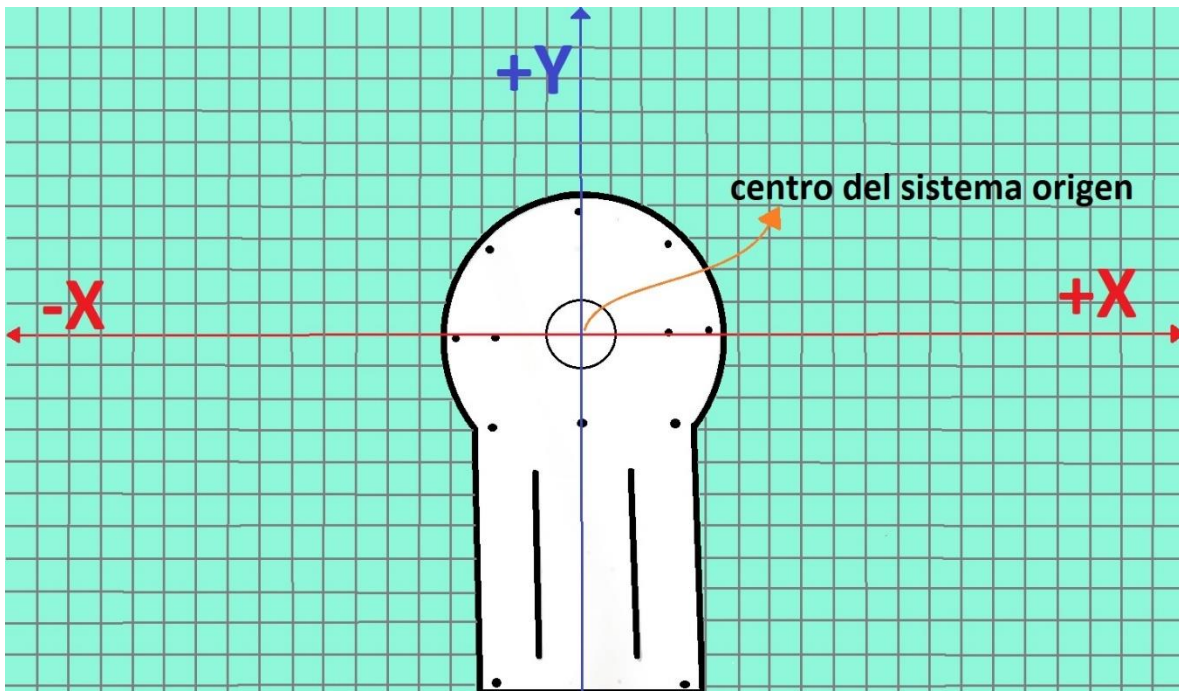


Figura 104 Plano "XY" y base del robot dibujada para colocar al robot LeArm sobre este plano.

La tarea necesitó de planeación de la rutina a ejecutarse. Con ayuda del **programa para la implementación de la cinemática directa** se manipuló el robot para llevarlo a los puntos en el espacio donde se requería que estuviera. Se tomó nota de los puntos importantes donde estaría el robot durante la rutina. Estos puntos en el espacio contemplaron la ubicación de las pelotas y del contenedor de pelotas donde serían depositadas.

Se colocaron tres pelotas en tres puntos del espacio, tomando en cuenta las coordenadas del sistema de referencia de la base (X, Y, Z). Los tres puntos en que se colocaron fueron:

- Pelota A (-16.2,12.7,15)
- Pelota B (1.7,19,8)
- Pelota C (20,11.5, 5.8).

El contenedor era de forma cilíndrica. El radio del contenedor era de 5 centímetros y con altura de 10 centímetros. Este fue colocado de tal manera que su centro coincidió en el punto del espacio con coordenadas (25,0,0).

Las tres pelotas tenían diferente diámetro: La pelota A tenía un diámetro de 1.8 [cm]; la pelota B tenía un diámetro de 2 [cm]; la pelota C tenía un diámetro de 2 [cm].

El diámetro de las pelotas fue importante considerarlo para tener en cuenta la apertura que debe tener la pinza o herramienta final del manipulador para agarrar correctamente cada pelota.

Con un ángulo de 158 grados en el servomotor de la herramienta final , la pinza tiene una apertura de 2 centímetros. Y con 160 grados en el servomotor de la pinza, se tiene una apertura de 1.8 centímetros. Con ayuda del **programa para la implementación de la cinemática directa** se manipuló el robot para poder moverlo en su espacio de trabajo y verificar que se ubicaba la herramienta en la posición espacial de cada pelota y el contenedor. También se usó este programa para determinar los movimientos y punto espaciales de la trayectoria que seguiría el robot para completar su tarea.

Se determinaron 24 acciones que ayudaron a realizar la rutina secuencial para la tarea encomendada, los cuales se muestran en la tabla 6. En cada acción se muestra el ángulo que debe tomar cada servomotor del robot manipulador.

Tabla 6 Información de las 24 acciones o puntos de la trayectoria para la rutina secuencial planteada. Se muestran los valores angulares de los servos para cada articulación y cada punto de la trayectoria.

No. acción de la trayectoria	servo 1 base [°]	servo 2 hombro [°]	servo 3 codo [°]	servo 4 cabeceo muñeca [°]	servo 5 balanceo muleca [°]	servo 6 pinza [°]
1	0	90	90	90	90	180
2	0	90	90	21	90	122
3	144	97	83	21	90	122
4	142	96	28	21	87	135
5	142	96	28	21	87	156
6	142	116	28	21	87	160
7	0	67	41	28	87	160
8	0	67	41	28	87	126
9	0	116	41	28	87	126
10	85	116	41	28	87	126
11	85	84	23	16	90	126
12	85	84	23	16	90	158
13	85	95	23	16	90	158
14	0	95	23	16	90	158
15	0	72	26	40	90	158
16	0	72	26	40	90	130
17	0	91	26	40	90	130

18	26	91	26	40	90	130
19	26	95	26	32	90	122
20	30	68	26	32	90	122
21	30	68	26	32	90	159
22	30	82	26	32	90	159
23	0	75	26	32	90	159
24	0	75	26	32	90	128

A su vez, cada acción de la tabla 6 posiciona y orienta la herramienta final del manipulador como se muestra en la tabla 7. Donde “x, y, z” corresponden a la posición y “Alpha, beta, gamma” corresponden a los ángulos de Euler para representar la orientación.

Tabla 7 Posición y orientación que se obtiene como salida tras ingresar los datos de la tabla 6 en la cinemática directa.

No. acción de la trayectoria	x [cm]	y [cm]	z [cm]	Alpha [°]	Beta [°]	gamma[°]
1	0	0	46.5	-90	0	0
2	16.33	0	35.27	-90	69	0
3	-12.18	8.85	35.19	26.25	-49	-56.8
4	-16.24	12.69	14.98	-39.3	-40.2	-138.7
5	-16.24	12.69	14.98	-39.3	-40.2	-138.7
6	-13.81	10.79	21.7	-21.32	-49.5	-113.52
7	25.15	0	9.85	87	46	-180
8	25.15	0	9.85	87	46	-180
9	16.3	0	28.75	-93	85	0
10	1.42	16.24	28.75	-3.43	4.98	-84.98
11	1.67	19.06	7.97	4.2	2.7	-147.1
12	1.67	19.06	7.97	4.2	2.7	-147.1
13	1.66	19.02	11.65	3.6	3.47	-136.1
14	19.09	0	11.65	90	44	-180
15	25.06	0	9.11	90	48	-180
16	25.06	0	9.11	90	48	-180
17	23.85	0	17.3	90	67	-180
18	21.44	10.46	17.3	38.7	55.8	-134
19	20.04	9.78	16.7	42.9	53.2	-139.3
20	20	11.5	5.8	54.5	30.6	-160
21	20	11.5	5.8	54.5	30.6	-160
22	20.2	11.66	11.5	48	41.6	-149
23	23.38	0	8.64	90	43	-180
24	23.38	0	8.64	90	43	-180

Los datos de la tabla 6 se ingresaron en el **programa para crear una rutina secuencial**. Se obtiene un archivo “cvs” con toda la información ingresada y este archivo se ejecuta con el **programa para ejecutar una rutina secuencial** que ejecuta la rutina para recoger las pelotas y colocarlas en el contenedor.

La rutina mostró resultados satisfactorios. La herramienta se posicionó en el lugar donde se encontraba cada pelota y se depositaban en el contenedor sin ningún problema. La cinemática directa fue fundamental para realizar esta tarea. Sin embargo, hay que manipular el robot y tantear la posición donde debe estar la herramienta, observando los datos de posición y orientación. Las figuras 105 y 106 muestran momentos de la ejecución de la rutina para recoger pelotas

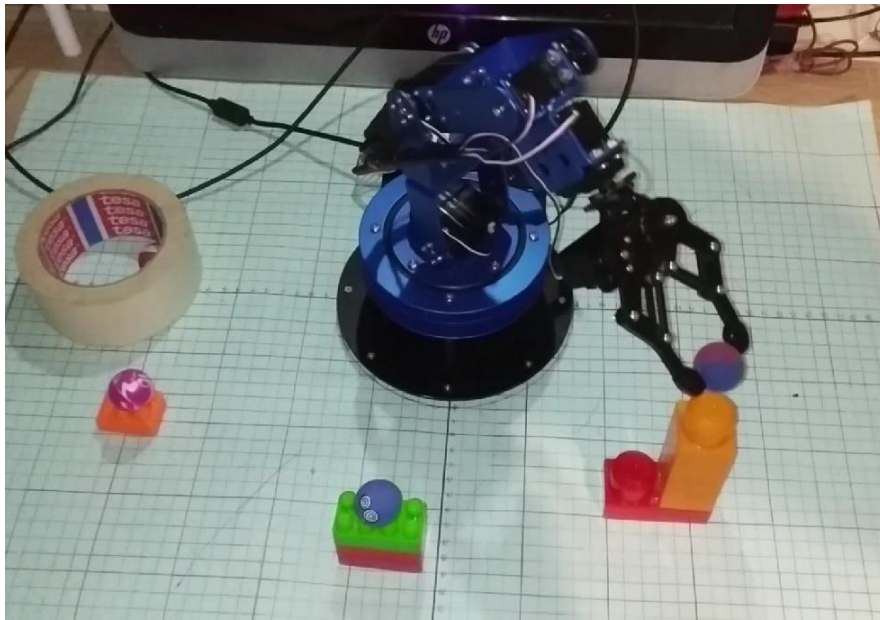


Figura 105 Vista aérea de la ejecución de la rutina secuencial para recoger pelotas.

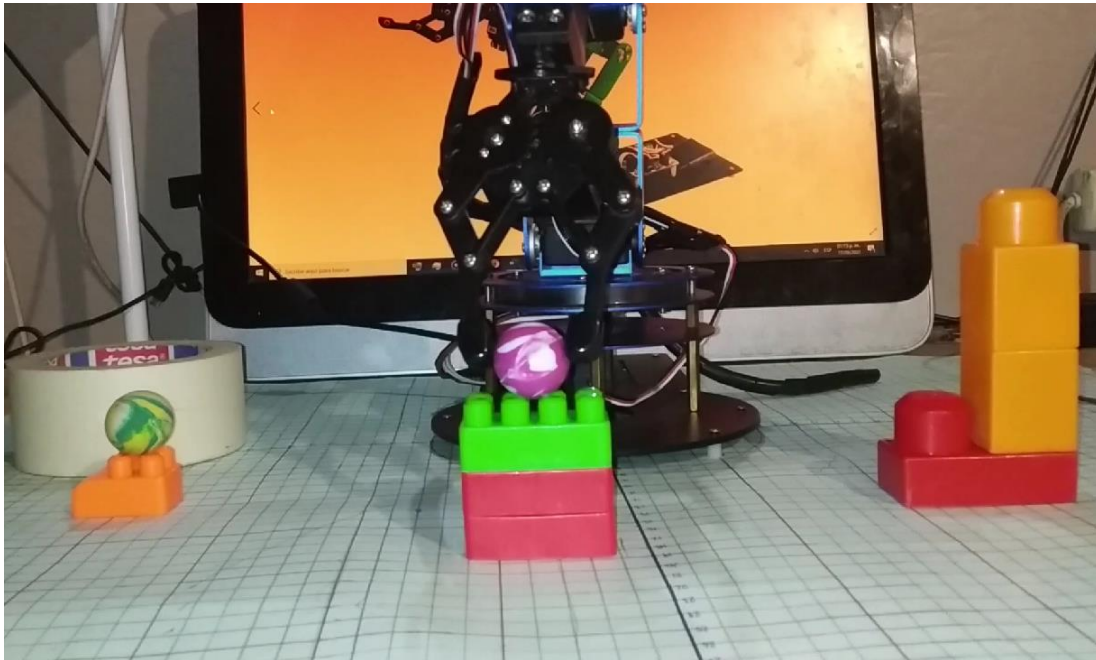


Figura 106 Vista frontal de la ejecución de la rutina secuencial para recoger pelotas.

Validación de la cinemática inversa

Para la prueba de validación de la cinemática inversa, se hizo uso del **programa para la implementación de la cinemática directa**, que anteriormente fue validado, para compararlo con el **programa para la implementación de la cinemática inversa**.

La cinemática directa tiene como datos de salida la posición y orientación de la herramienta del robot, que son la respuesta a los datos de entrada, las variables articulares del robot manipulador.

Por el contrario, la cinemática inversa tiene como salidas las variables articulares del robot y como datos de entrada, la posición y orientación de la herramienta.

Por tanto, para comprobar y validar el programa de la cinemática inversa, se compararon las entradas de la cinemática directa con las salidas de la cinemática inversa y se compararon las salidas de la cinemática directa con las entradas de la cinemática inversa

Se tienen 3 pruebas que ejemplifican la comparación entre los dos programas de cinemática. En la primera y tercer prueba se compararon la orientación mediante los ángulos de Euler y para la segunda prueba se compararon las matrices de transformación.

Los resultados de la primera prueba se muestran en las figuras 107, 108 y 109; los resultados de la segunda se muestran en las figuras 110 y 111; para la tercera, los resultados se muestran en las figuras 112, 113 y 114.

Para la primera y tercera prueba se obtuvo, en la cinemática inversa, una segunda solución. Se tiene una tercera figura en las pruebas uno y tres, las figuras 109 y 114, respectivamente. Estas representan la comprobación de que existe la segunda solución dada por el programa de cinemática inversa, y que las salidas de estas soluciones representan las entradas en la cinemática inversa de sus respectivas pruebas.

Prueba 1



Figura 107 Datos ingresados en la cinemática directa y sus respectivos datos de salida (primera prueba).



Figura 108 Datos de entrada y salida para la cinemática inversa de la primera prueba.



Figura 109 Comprobación con la cinemática directa de la segunda solución mostrada en la figura 108.

Segunda prueba



Figura 110 Datos ingresados en la cinemática directa y sus respectivos datos de salida (segunda prueba).



Figura 111 Datos de entrada y salida para la cinemática inversa de la segunda prueba.

Tercera prueba

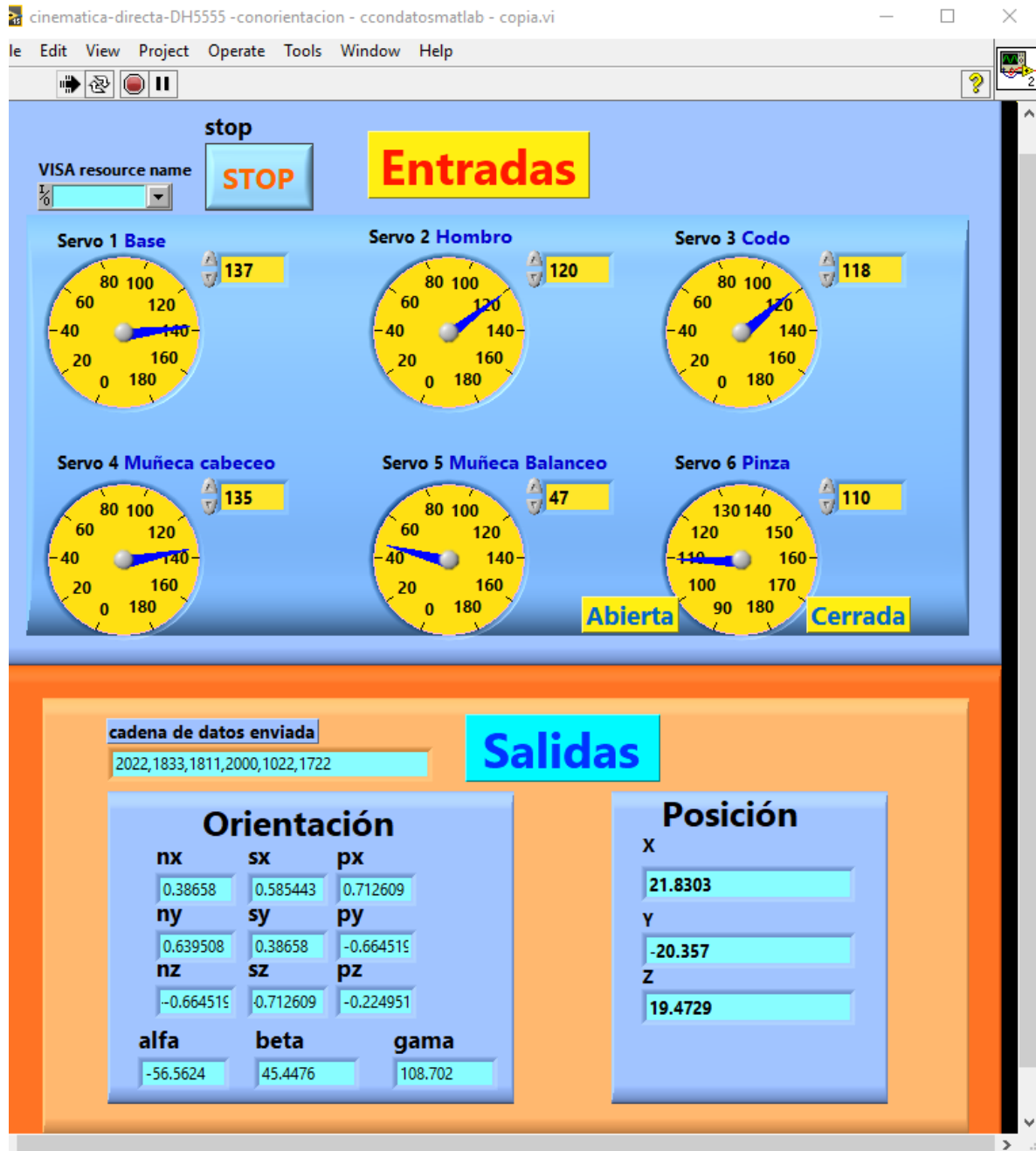


Figura 112 Datos ingresados en la cinemática directa y sus respectivos datos de salida (tercera prueba).



Figura 113 Datos de entrada y salida para la cinemática inversa de la tercera prueba.



Figura 114 Comprobación con la cinemática directa de la segunda solución mostrada en la figura 113.

Las pruebas mostradas son una representación del método usado para validar el programa de la cinemática inversa. Se hicieron más pruebas abarcando las distintas combinaciones que pueden darse. Los resultados muestran que el **programa para la implementación de la cinemática inversa** es correcto y que muestra valores congruentes que se verifican con los datos de la cinemática directa, programa validado con anterioridad. Con esto, es posible dar paso a una aplicación directa de la cinemática inversa para el robot bajo estudio.

Implementación de rutina para dibujar letras y figuras

Para la implementación de la cinemática inversa se implementaron rutinas a partir de los datos obtenidos mediante el **programa para la implementación de la cinemática inversa**.

Las rutinas tuvieron como objetivo dibujar figuras geométricas y letras. Las figuras se escribieron con la pinza del robot y un plumón sobre un pizarrón blanco.

Para escribir las letras y figuras se delimitó el área sobre el cual podía escribir el robot, la cual es un área sobre el plano "xy" y delimitada en el eje "x" por el rango [-5,5] y en el eje "y" por el rango [10,20], como se muestra en la figura 115.

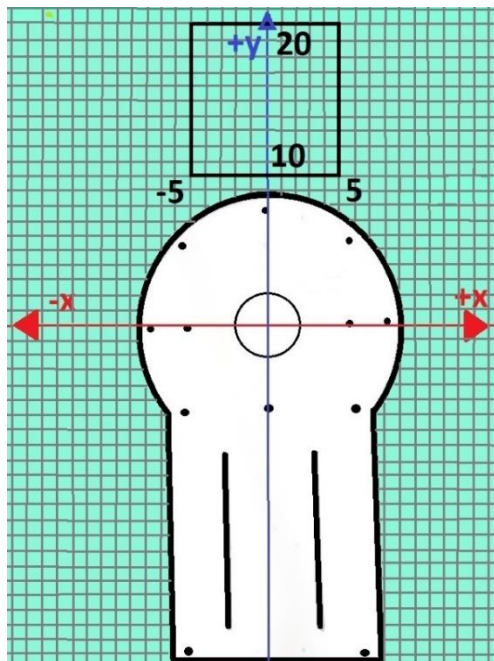


Figura 115 Área del plano XY sobre la que se dibujaron las figuras y letras.

En esta área limitada, que es un cuadrado de 10 centímetros de medida de cada lado, se dibujaron cada una de las figuras y letras para esta prueba.

Para cada prueba fue necesario localizar algunos puntos principales de cada figura y letra. En total se dibujaron 2 figuras y 4 letras: un cuadrado, un triángulo equilátero, la letra "O", la letra "M", la letra "A" y la letra "R". La unión de todos los puntos conformó cada figura y permitieron trazar una trayectoria que dibujó a la figura deseada.

En cada figura o letra, se determinaron las coordenadas de los puntos que les componen. De esta manera fue posible ingresar las coordenadas en el **programa para implementación de la cinemática inversa** y obtener los datos de las variables articulares para configurar al robot en cada punto. Las variables articulares fueron necesarias para posicionar y orientar la herramienta del robot.

Se muestra cada figura y letra con los puntos que le componen, así como las coordenadas angulares de cada punto, respectivamente. También se diseñaron unas tablas para relacionar los puntos de cada figura con los resultados tras ingresar estos datos en el **programa para la implementación de la cinemática inversa**.

Los resultados del programa de cinemática inversa son las variables articulares Q1,Q2,Q3,Q4,Q5 y Q6. Las primeras cinco variables articulares representan a cada articulación del robot, mientras que la sexta variable articular representa al ángulo que toma el servomotor que acciona la pinza o herramienta final.

Para esta actividad el ángulo del servomotor de la pinza se mantuvo constante para poder sostener un plumón para escribir. El ángulo que tomó el servomotor de la pinza fue de 169 grados con el cual la pinza abre 1.4 centímetros.

No se consideró la orientación porque hay diversas formas de orientar la herramienta del robot. Para determinar la orientación y poder ingresar estos datos en la cinemática inversa, se apoyó en la cinemática directa y del robot físico para observar la manera más conveniente de orientar la pinza. Observando la orientación en cada punto, se ingresaron junto a los datos de la posición para obtener las variables articulares.

Cuadrado

La figura dibujada fue un cuadrado con lados de 10 centímetros de longitud. Le componen 8 puntos: a, b, c, d, e, f, g, h (Figura 116). Las coordenadas angulares para cada punto se muestran en la tabla 8.

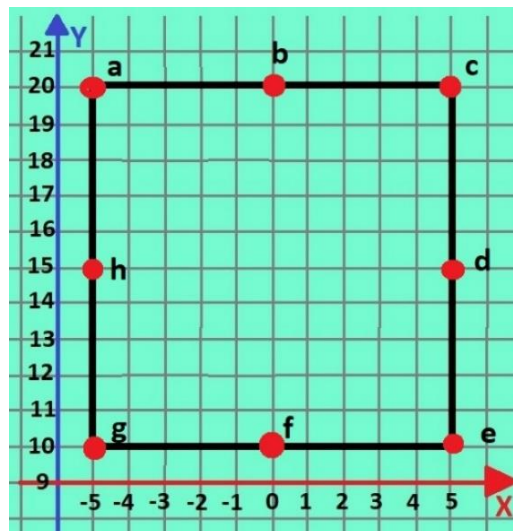


Figura 116 Figura cuadrada con sus respectivos puntos que conforman a la figura para su trazado.

Tabla 8 Valores de posición cada punto de la figura cuadrada y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(-5,20,0)	113	64	1	52	43	169
b(0,20,0)	90	65	3	46	22	169
c(5,20,0)	68	60	11	42	2	169
d(5,15,0)	69	56	29	9	133	169
e(5,10,0)	65	69	8	9	129	169
f(0,10,0)	89	67	10	4	153	169
g(-5,10,0)	110	63	17	3	174	169
h(-5,15,0)	108	53	33	6	172	169

Triángulo isósceles

La figura dibujada fue un triángulo isósceles con dos lados de 11.18 centímetros de longitud y un lado de 10 centímetros de longitud. Le componen 8 puntos: a, b, c, d, e, f, g, h, como se observan en la figura 117. Las coordenadas angulares para cada punto se muestran en la tabla 9.

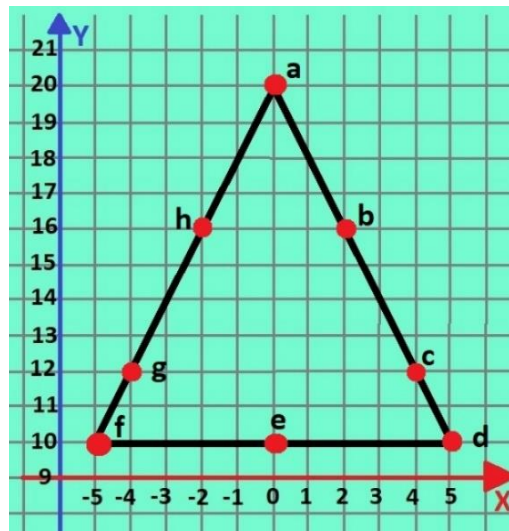


Figura 117 Figura de triángulo isósceles con sus respectivos puntos que conforman a la figura para su trazado.

Tabla 9 Valores de posición cada punto de la figura triangular y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(0,20,0)	90	65	3	46	22	169
b(2,16,0)	82	51	38	3	146	169
c(4,12,0)	69	71	7	15	133	169

d(5,10,0)	65	69	8	9	129	169
e(0,10,0)	89	67	10	4	153	169
f(-5,10,0)	110	63	17	3	174	169
g(-4,12,0)	108	72	5	16	172	169
h(-2,16,0)	96	51	37	4	160	169

Letra "O"

La letra dibujada fue la letra 'O' que se representa como un círculo con radio de 5 centímetros y centro en las coordenadas (0,15,0). Cuenta con 24 puntos que le componen: a, b, c, d, e, f, g, h, i, j, k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, como se observan en la figura 118.

La letra 'O' al ser un círculo se puede presentar su ecuación sobre el eje 'XY' como se muestra en la ecuación 133.

$$X^2 + Y^2 - 30Y + 200 = 0 \quad (133)$$

Se usó la variable x como función de la variable 'y' para obtener aquellos puntos que no es posible determinar a simple vista. Para encontrar la variable 'y' se usó la ecuación 134. La coordenada 'z' fue '0' para todos los puntos porque estos se encontraban sobre el plano 'XY'.

$$Y = \sqrt{25 - X^2} + 15 \quad (134)$$

Las coordenadas angulares para cada punto se muestran en la tabla 10.

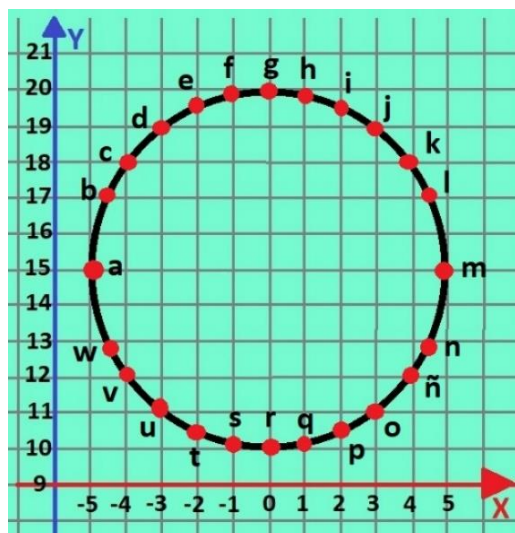


Figura 118 Letra 'O' o círculo con sus respectivos puntos que conforman a la figura para su trazado.

Tabla 10 Valores de posición cada punto de la letra 'O' y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(-5,15,0)	108	53	33	6	172	169
b(-4.5,17.12,0)	105	57	26	18	130	169
c(-4,18,0)	102	52	33	15	127	169
d(-3,19,0)	98	47	43	10	123	169
e(-2,19.58,0)	94	43	50	7	119	169
f(-1,19.89,0)	91	40	55	5	116	169
g(0,20,0)	87	39	57	4	113	169
h(1,19.89,0)	84	40	56	4	109	169
i(2,19.58,0)	81	41	53	5	106	169
j(3,19,0)	77	44	47	7	102	169
k(4,18,0)	72	50	38	11	98	169
l(4.5,17.2,0)	70	54	31	14	95	169
m(5,15,0)	64	64	14	20	90	169
n(4.5,12.87,0)	69	60	23	6	128	169
ñ(4,12,0)	70	65	14	9	129	169
o(3,11,0)	73	72	4	12	132	169
p(2,10.417,0)	78	67	11	5	138	169
q(1,10.1,0)	83	70	7	6	142	169
r(0,10,0)	83	70	6	7	142	169
s(-1,10.1,0)	92	70	6	7	151	169
t(-2,10.417,0)	97	68	9	6	156	169
u(-3,11,0)	101	64	15	4	160	169
v(-4,12,0)	104	57	26	0	163	169
w(-4.5,12.87,0)	108	67	12	14	167	169

Letra "M"

La letra dibujada fue la letra 'M'. Esta letra se compone con 7 puntos: a, b, c, d, e, f, g, como se visualizan en la figura 119. Las coordenadas angulares para cada punto se muestran en la tabla 11.

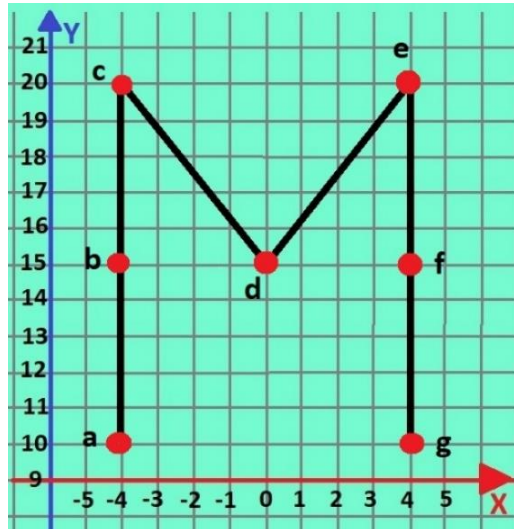


Figura 119 Letra 'M' con sus respectivos puntos que conforman a la letra para su trazado.

Tabla 11 Valores de posición cada punto de la letra 'M' y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(-4,10,0)	105	65	14	4	164	169
b(-4,15,0)	104	61	20	14	163	169
c(-4,20,0)	102	36	64	0	160	169
d(0,15,0)	87	71	4	25	147	169
e(4,20,0)	71	62	8	43	143	169
f(4,15,0)	73	52	36	3	132	169
g(4,10,0)	69	66	12	5	129	169

Letra 'A'

La letra dibujada fue la letra 'A'. Esta letra se compone con 7 puntos: a, b, c, d, e, f, g, como se visualizan en la figura 120. Las coordenadas angulares para cada punto se muestran en la tabla 12.

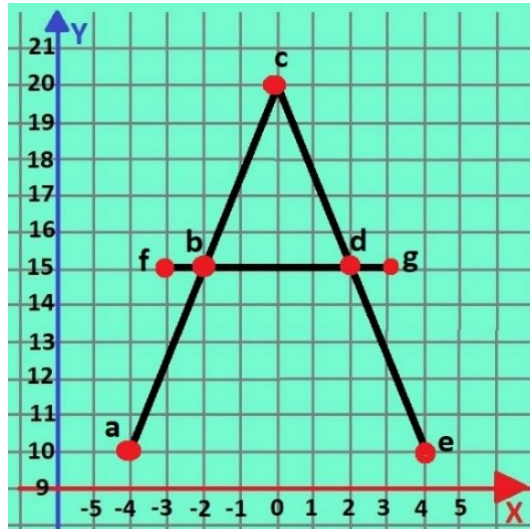


Figura 120 Letra 'A' con sus respectivos puntos que conforman a la letra para su trazado.

Tabla 12 Valores de posición cada punto de la letra 'A' y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(-4,10,0)	105	65	14	4	164	169
b(-2,15,0)	96	58	26	9	155	169
c(0,20,0)	90	65	3	46	22	169
d(2,15,0)	80	57	28	8	139	169
e(4,10,0)	69	66	12	5	129	169
f(-3,15,0)	100	57	27	8	159	169
g(3,15,0)	76	56	30	7	135	169

Letra 'R'

La letra dibujada fue la letra 'R'. Esta letra se compone con 10 puntos: a, b, c, d, e, f, g, h, i, j, como se observan en la figura 121. Las coordenadas angulares para cada punto se muestran en la tabla 13.

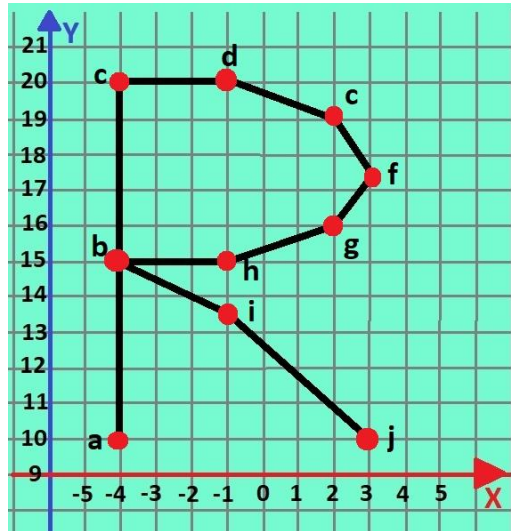


Figura 121 Letra 'R' con sus respectivos puntos que conforman a la letra para su trazado.

Tabla 13 Valores de posición cada punto de la letra 'R' y sus respectivas variables articulares para alcanzar tal posición.

Punto (X, Y, Z)	Q1 [°]	Q2 [°]	Q3 [°]	Q4 [°]	Q5 [°]	Q6 [°]
a(-4,10,0)	105	65	14	4	164	169
b(-4,15,0)	104	61	20	14	163	169
c(-4,20,0)	102	36	64	0	160	169
d(-1,20,0)	94	59	17	34	133	169
e(2,19,0)	81	64	10	37	121	169
f(3,17.5,0)	77	64	13	28	116	169
g(2,16,0)	81	65	13	22	120	169
h(-1,15,0)	95	71	4	25	134	169
i(-1,13.6,0)	95	68	11	15	135	169
j(3,10,0)	75	72	4	9	126	169

Con los datos de las coordenadas de los puntos de cada figura o letra se procedió a crear una rutina en el **programa para crear una rutina secuencial**. Cada una contiene los datos de las coordenadas articulares que posicionan y orientan la herramienta final en cada punto que forman parte de las figuras.

Se estableció una configuración inicial con la cual cada servomotor del robot a excepción del de la pinza se encontraban a 90 grados. Después de la configuración inicial, la rutina continúa con los puntos que conforman a cada figura.

La mayoría de las figuras y letras son cerradas, es decir, comienzan en un punto y terminan en el mismo punto. Dicho de otra manera, es posible dibujar sin levantar la herramienta, haciendo un trazo continuo. Para la letra 'M' aunque no es cerrada, si se pudo hacer un trazo continuo. En el caso de la letra 'R' se pasó por el punto 'b' dos veces, por lo que fue necesario considerarlo al momento de crear la rutina.

El caso de la letra 'A' es diferente a todos los demás. La letra 'A' se dibujó necesariamente despegando el plumón del pizarrón para poder realizar dos trazos diferentes que forman a la figura,

Para dibujar la letra 'A' se ingresaron los datos de la rutina de los puntos 'a-e'. Se ingresó nuevos datos a la rutina para levantar la herramienta del robot mediante la segunda articulación e interrumpir la escritura momentáneamente. Después se posicionó la herramienta en el punto 'f' para continuar con el trazado de la recta que complementaba el dibujado de la letra 'A' hasta terminar la rutina en el punto 'G'.

Se muestran los resultados de las rutinas de dibujado para cada figura y letra en las figuras 122, 123, 124, 125, 126 y 127.

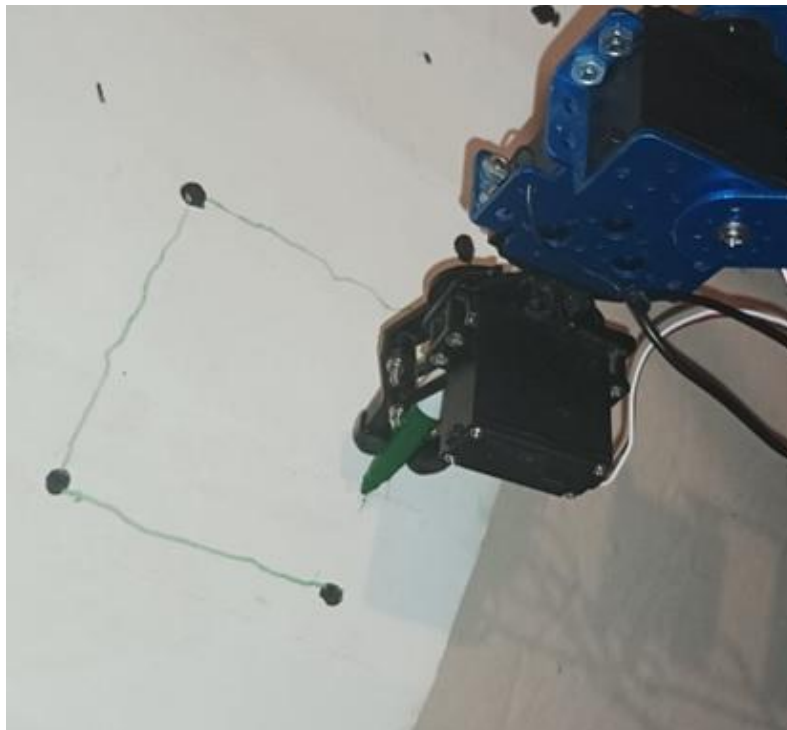


Figura 122 Trazado de figura cuadrada por el robot LeArm.

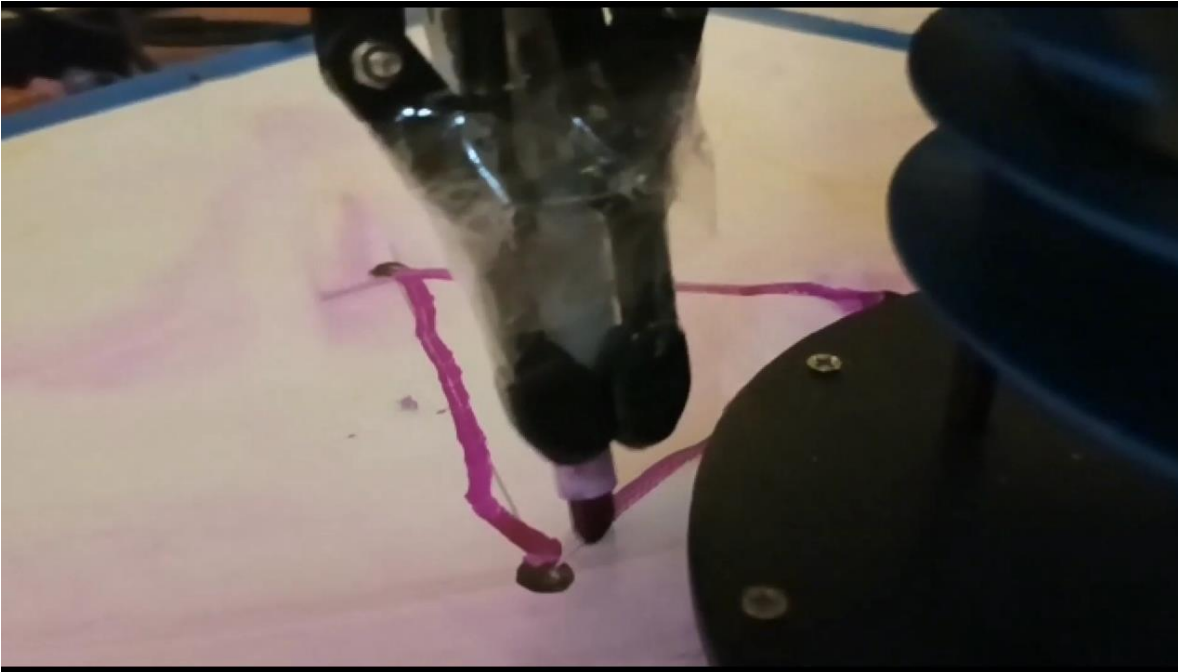


Figura 123 Trazado de figura triangular por el robot LeArm.

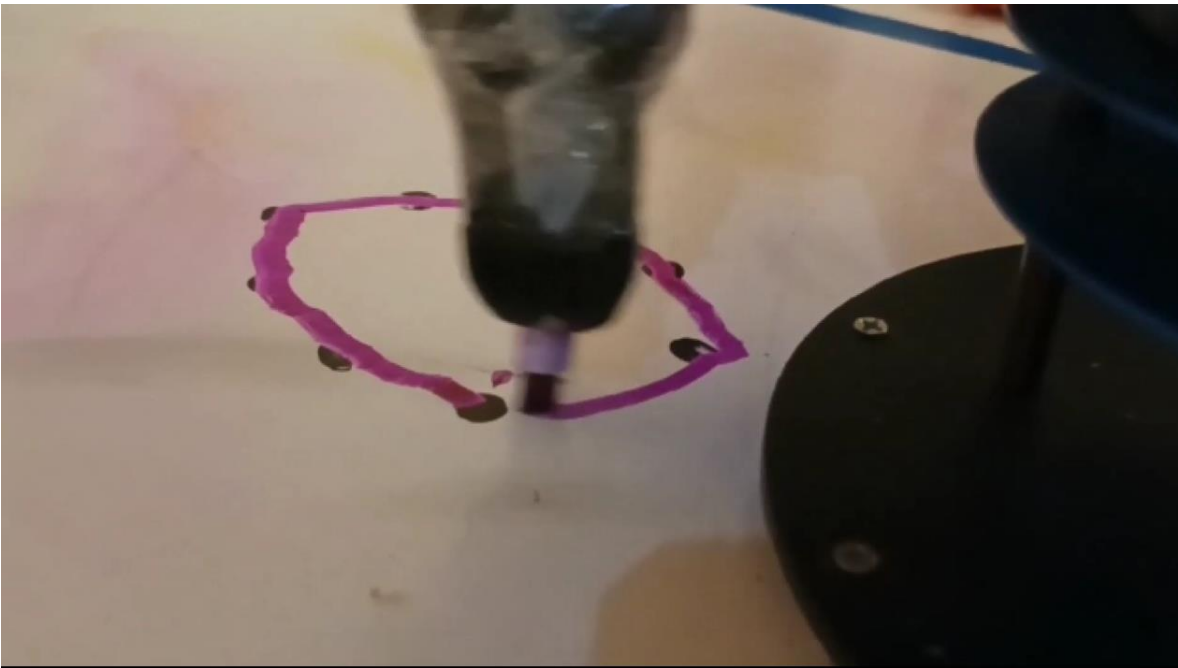


Figura 124 Trazado de letra 'O' por el robot LeArm.

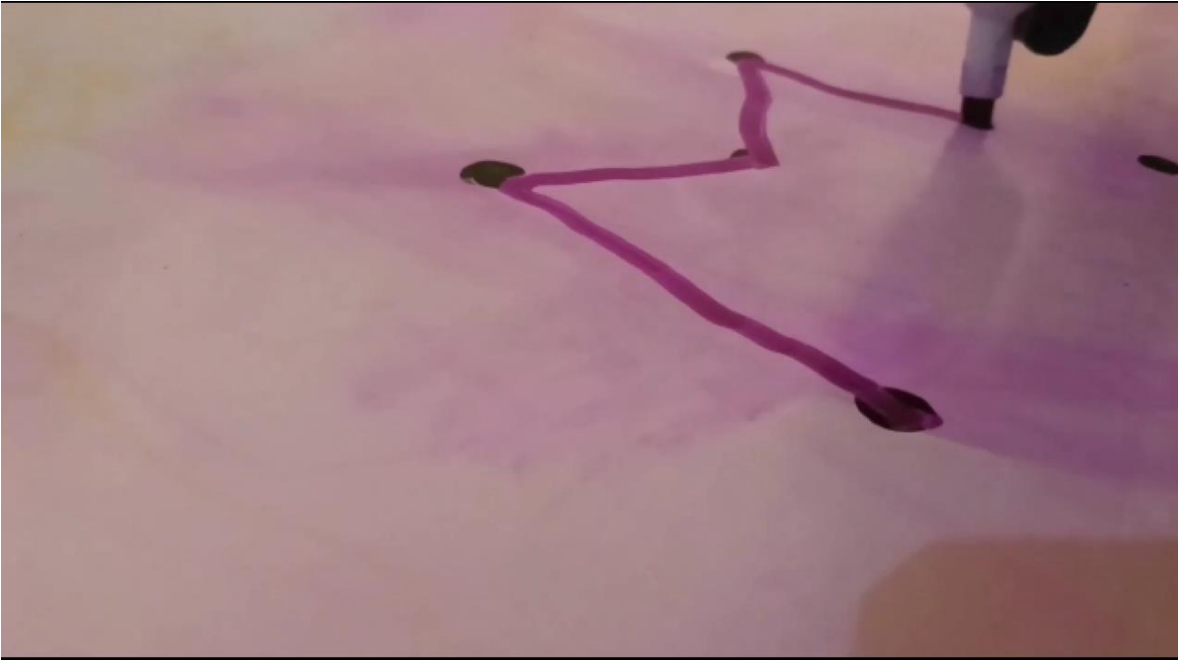


Figura 125 Trazado de letra 'M' por el robot LeArm.

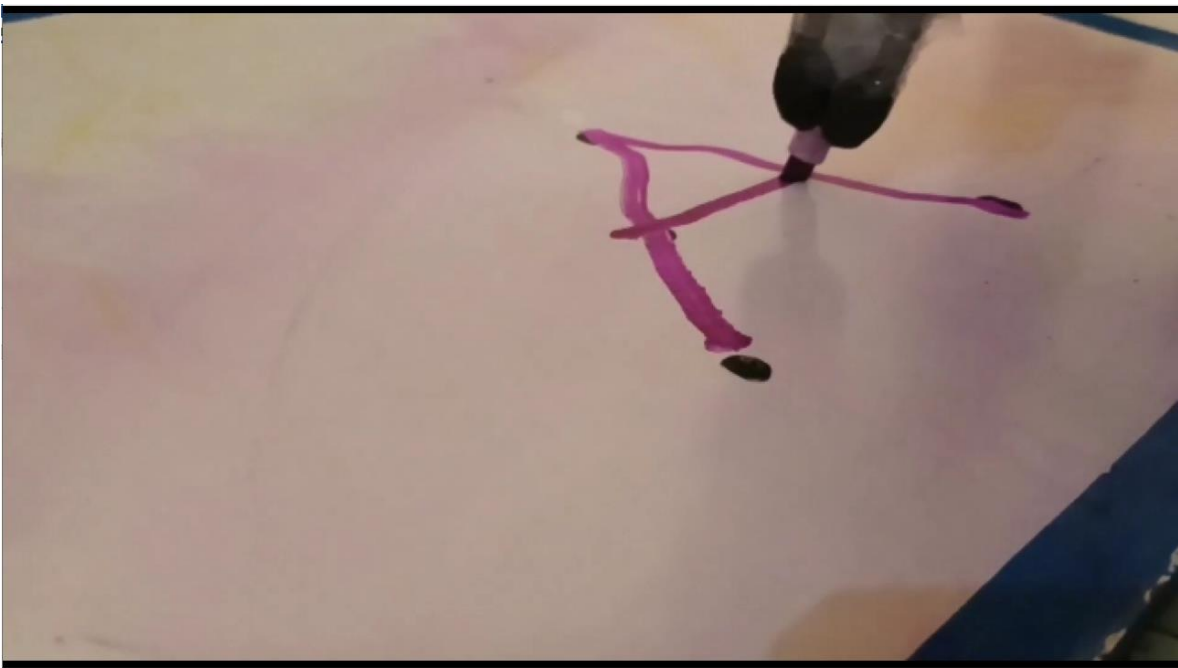


Figura 126 Trazado de letra 'A' por el robot LeArm.

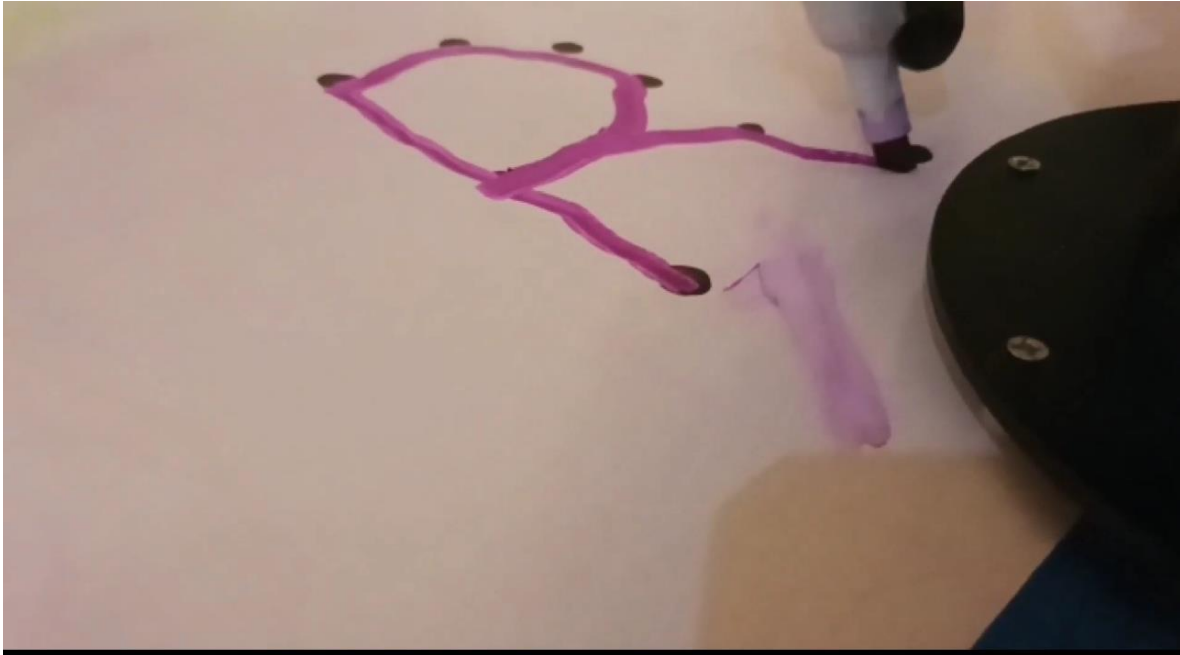


Figura 127 Trazado de letra 'R' por el robot LeArm.

De acuerdo con los resultados mostrados, se logran visualizar las figuras y letras que se contemplaron escribir o dibujar. Se observó que a mayor cantidad de puntos que componen a un trazo, se obtiene una mejor definición de las líneas dibujadas y por consecuencia una mejor claridad de la letra o figura.

Los huelgos de la estructura del robot afectaron también el trazado de las líneas, pero los errores que pudieran perjudicar a la claridad de las líneas dibujadas se compensan por la cantidad de puntos contemplados en el trazado de cada letra y figura.

Conclusión

El Prototipo de Brazo Robot Manipulador en estudio representa un proyecto analítico, experimental y práctico que consolida diversos conocimientos adquiridos durante la licenciatura en Ingeniería Eléctrica Electrónica en la Facultad de Ingeniería de la UNAM. El análisis cinemático del brazo robot manipulador implica la integración de conocimientos provenientes de áreas como ingeniería, matemáticas, física y programación.

Como resultado de este estudio, se ha logrado el cumplimiento del objetivo inicial: desarrollar e implementar el modelo de cinemática directa e inversa de un brazo robot educativo con 5 grados de libertad. Este modelo permite determinar la posición y orientación del actuador final respecto a su sistema de coordenadas de referencia, posibilitando la ejecución de acciones de manipulación, maniobrabilidad, interacción y recolección de objetos.

Se han creado diversas rutinas que facilitan la adquisición de conocimientos sobre los elementos clave de un brazo robot, abarcando desde su estructura mecánica, actuadores eléctricos y servomotores, hasta las estructuras secuenciales y las dimensiones espaciales. Además, se ha abordado la manipulación del brazo en su entorno espacial, la implementación de algoritmos de manipulación y la realización de pruebas de desempeño que han validado la fiabilidad de los algoritmos programados a partir del análisis matemático de la cinemática directa e inversa.

Mediante los algoritmos desarrollados, se llevaron a cabo rutinas secuenciales para poner a prueba el desempeño del robot, demostrando la utilidad de la cinemática inversa y directa en la realización de tareas asignadas, como la manipulación y recolección de objetos, así como la escritura o dibujo de figuras.

Es importante señalar que el desarrollo de este proyecto tuvo lugar fuera de las instalaciones universitarias. Se reconoce que la precisión de los resultados podría mejorar en condiciones más propicias para el proyecto.

A pesar de las limitaciones inherentes a la estructura mecánica del robot, se logró obtener un resultado satisfactorio. Aunque existen pequeños holguras causadas por el peso y movimiento de la estructura, generando leves errores en la calidad de la línea trazada por el robot o al posicionar la herramienta final, las figuras dibujadas son claras y la recogida de objetos es exitosa con algunas correcciones para ubicar la herramienta en el lugar exacto necesario.

A pesar de la precisión alcanzada, las pruebas de desempeño han resultado altamente satisfactorias, proporcionando un material de trabajo confiable para el desarrollo de actividades educativas y de formación profesional.

Trabajos citados

- Alcubilla, R., Pons, J., & Bardes, D. (1995). *Diseño digital una perspectiva VLSI-CMOS*. España: Ediciones de la Universitat Politècnica de Catalunya.
- Balabanian, N., & Carlson, B. (2002). *Principios de diseño lógico digital*. México: Grupo Patria Cultural.
- Barrientos, A., Peñín, L. F., Belaguer, C., & Aracil, R. (2007). *Fundamentos de Robótica*. Madrid, España: McGraw-Hill.
- Beer, F. P., Johnston, E. R., & Cornwell, P. J. (2010). *Mecánica vectorial para ingenieros Dinámica*. México: McGraw-Hill.
- Carvajal Rojas, J. (2006). *Modelamiento y diseño de robots industriales*. Medellín Colombia: Universidad de la Salle FIDAE.
- Craig, J. J. (2006). *Robótica*. México: Pearson Education.
- Hibbeler, R. C. (2010). *Ingeniería Mecánica-Dinámica*. México: McGraw-Hill.
- Invent.* (5 de septiembre de 2023). Obtenido de <https://www.invent.org/inductees/george-devol>
- Lajara Vizcaíno, J. R., & Pelegrí Sebastián, J. (2018). *LabVIEW Entorno gráfico de programación*. México: Alfaomega.
- Langer, E. (5 de septiembre de 2023). *washingtonpost*. Obtenido de https://www.washingtonpost.com/local/obituaries/george-c-devol-99-self-taught-tinkerer-who-invented-robotic-arm/2011/08/17/gIQA0Ed5LJ_story.html
- Ni.* (5 de septiembre de 2023). Obtenido de <https://www.ni.com/es/shop/labview.html>
- Ni.* (5 de Septiembre de 2023). Obtenido de <https://www.ni.com/es/support/documentation/supplemental/06/ni-visa-overview.html>
- Ni.* (5 de Septiembre de 2023). Obtenido de <https://www.ni.com/es/support/documentation/supplemental/16/simple-state-machine-template-documentation.html>
- Reyes Cortés, F. (2011). *Control de robots manipuladores*. México: Alfaomega .
- Reyes Cortés, F. (2012). *MATLAB aplicado a Robótica y mecatrónica*. México: Alfaomega.

Saha, S. K. (2010). *Introducción a la robótica*. México: Pearson Education.

Solís, R. (2006). *Geometría analítica*. México: Limusa.

Swokowski, E. L. (2009). *Álgebra y trigonometría con geometría analítica*. México: Cengage Learning.

Anexos

Programas Tiva

Se presentan los programas que se ejecutan en la tarjeta TIVA para poder establecer la comunicación entre el software de labview y los servomotores del robot manipulador.

Son tres los programas implementados en los que se usa lenguaje c y algunas librerías propias de la plataforma de programación “Energía”.

Cada programa está comentado y corresponde a los tres programas en que se hace uso de la tarjeta TIVA: programa de control de un servomotor mediante ancho de pulso, programa para la implementación de la cinemática inversa y programa para ejecutar rutina secuencial.

Para el programa para la implementación de la cinemática inversa se hace utiliza la escritura en ancho de pulso en programa de la tarjeta TIVA. En cambio, para el programa ejecutar rutina secuencial se escribe el ángulo en los servomotores en grados, sin emplear el ancho de pulso. Estas diferencias no afectan el funcionamiento.

Se implementó de esta forma para verificar que funcionan de la misma forma ambos programas porque hay servomotores que trabajan con diferente rango de ancho de pulso y pueden llegar a implementar mal los ángulos que reciben como instrucción.

Programa 1

Programa para manipular un servomotor mediante ancho de pulso, se observa en la figura 128.


```

#include<Servo.h>
//Se define a la variable servol que representa al servomotor a manipular
Servo servol;
int angl; // la variable angl guarda el valor de angulo leído a través del serial
void setup() {

Serial.begin(9600); //configuración de velocidad de datos del puerto serial, si igual que
                //en labview es 9600 baudios.
servol.attach(8,500,2500); /* configurando pin como salida a servol y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
}
//La siguiente Funcion es la que corre el código que estará ejecutando constantemente, pero
//se precinde de el para el programa actual.
void loop() {
    // put your main code here, to run repeatedly:
}
void serialEvent()//solo si el serial tiene actividad de envio o recibo de datos
                //se ejecuta el código
{
    if(Serial.available())//si el serial esta disponible se leen los datos que se
                //estan recibiendo a través de este
    {
        angl=Serial.parseInt();/*esta función lee un entero*/

        if(Serial.read()=='\n')//al detectarse el salto de linea, es decir el fin de la
                //cadena de caracteres, termina la lectura de datos del serial.
            servol.writeMicroseconds(angl);//se escribe el valor de "angulo l"(en ancho de pulso)en servo l
    }
}
}

```

Figura 128 Programa para la TIVA que se ejecuta con el programa de control de un servomotor mediante ancho de pulso.

Programa 2

Programa para manipular los seis servomotores del robot LeArm en la implementación de los algoritmos de cinemática inversa y directa mediante ancho de pulso, véase las figuras 129 y 130.

```

#include<Servo.h>
//Se definen las variables que representan a cada uno de los servos que dan movimiento
//a las articulaciones del robot manipulador

Servo servob;//servo base (torso)
Servo servom;//servo medio (hombro)
Servo servof;//servo ultimo (codo)
Servo servopl;//servo cabeceo (orientacion)
Servo servop2;//servo rotacion (orientacion)
Servo servop3;//servo (pinza)
/*En las siguientes variables se guardan los valores de angulos enviados desde labview,
De esta forma se separan los angulos vinculados a cada servomotor */
int angba;//angulo servo base (torso)
int angme;//angulo servo medio (hombro)
int angfi; //angulo servo ultimo (codo)
int angpl; //angulo servo cabeceo (orientacion)
int angp2;//angulo servo rotacion (orientacion)
int angp3;//angulo servo pinza

//En esta funcion se configuran los pines como salidas de datos y la velocidad de transmision
//de datos
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //configuracion de velocidad de datos del puerto serial, si igual que
  //en labview es 9600 baudios.
  servob.attach(8,500,2500);/* configurando pin como salida a servo de la base y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
  servom.attach(7,500,2500);/* configurando pin como salida a servo medio y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
  servof.attach(6,500,2500);/* configurando pin como salida a servo ultimo (codo) y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
  servopl.attach(5,500,2500);/* configurando pin como salida a servo de la muñeca (cabeceo) y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
  servop2.attach(4,500,2500);/* configurando pin como salida a servo de la muñeca (rotacion) y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
  servop3.attach(3,500,2500);/* configurando pin como salida a servo de la muñeca (pinza) y definiendo el
al rango de ancho de pulso (500-2500 microsegundos)*/
}
//La siguiente Funcion es la que corre el codigo que estará ejecutando constantemente, pero
//se precinde de el para el programa actual.
void loop() {
  // put your main code here, to run repeatedly:
}
//En vez de la anterior funcion, la siguiente funcion se usa para que permita ejecutar codigo
//solo cuando el Serial este enviando o recibiendo datos.

```

Figura 129 Programa para la TIVA que se ejecuta con el programa para la implementación de la cinemática inversa (parte 1).

```

void serialEvent()//solo si el serial tiene actividad de envio o recibo de datos
                //se ejecuta el codigo
{
  if(Serial.available())//si el serial esta disponible se leen los datos que se
                        //estan recibiendo a través de este
  {
    angba=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo
    diferente de un numero, es decir, el caracter"coma" (,) y se guarda en la variable
    angba* (angulo base)*/
    angme=Serial.parseInt();//esta función lee un número entero hasta que se encuentre algo diferente
    de un numero es decir, el caracter "coma" (,) y se guarda en la variable angme*/
    angfi=Serial.parseInt();//esta función lee un número entero hasta que se encuentre algo diferente
    de un numero es decir, el caracter "coma" (,) y se guarda en la variable angfi*/
    angpl=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    de un numero es decir, el caracter "coma" (,) y se guarda en la variable angpl*/
    angp2=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    de un numero es decir, el caracter "coma" (,) y se guarda en la variable angp2*/
    angp3=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    de un numero es decir, el caracter "coma" (,) y se guarda en la variable angp3*/

    //con los datos concatenados en labview, al final se agregó un salto
    //de linea. con eso se indica que se envio toda una cadena de caracteres y sirve para
    //diferenciarla de otras diferentes con angulos diferentes
    if(Serial.read()=='\n')//al detectarse el salto de linea, es decir el fin de la
                            //cadena de caracteres, termina la lectura de datos del serial.
    {
      servob.writeMicroseconds(angba);//se escribe el valor de "angulo base"(en ancho de pulso)en servo base
      servom.writeMicroseconds(angme);//se escribe el valor de angulo medio (en ancho de pulso)en servo medio
      servof.writeMicroseconds(angfi);//se escribe el valor de angulo ultimo(en ancho de pulso)en servo ultimo
      servop1.writeMicroseconds(angpl);//se escribe el valor de angulo cabeceo(en ancho de pulso) en servo cabeceo
      servop2.writeMicroseconds(angp2);//se escribe el valor de angulo rotacion(en ancho de pulso) en servo rotacion
      servop3.writeMicroseconds(angp3);//se escribe el valor de angulo pinza(en ancho de pulso) en servo pinza
    }
  }
}

```

Figura 130 Programa para la TIVA que se ejecuta con el programa para la implementación de la cinemática inversa (parte 2).

Programa 3

Programa para manipular los seis servomotores del robot LeArm en la implementación del algoritmo para ejecutar rutinas secuenciales, como se visualiza en las figuras 131, 132 y 133. En este programa no se escribe el ancho de pulso, sino el valor de ángulo deseado, directamente.

```

#include<Servo.h>
//Se definen las variables que representan a cada uno de los servos que dan movimiento
//a las articulaciones del robot manipulador

Servo servob; //servo base (torso)
Servo servom; //servo medio (hombro)
Servo servou; //servo ultimo (codo)
Servo servoc; //servo cabeceo (orientacion)
Servo servor; //servo rotacion (orientacion)
Servo servop; //servo (pinza)

/*En las siguientes variables se guardan los valores de angulos enviados desde labview,
De esta forma se separan los angulos vinculados a cada servomotor */
int angba; //angulo servo base (torso)
int angme; //angulo servo medio (hombro)
int angul; //angulo servo ultimo (codo)
int angca; //angulo servo cabeceo (orientacion)
int angro; //angulo servo rotacion (orientacion)
int angpi; //angulo servo pinza

//En esta funcion se configuran los pines como salidas de datos y la velocidad de transmision
//de datos
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //configuracion de velocidad de datos del puerto serial, si igual que
  //en labview es 9600 baudios.

  servob.attach(2); /* configurando pin como salida a servo de la base*/
  servom.attach(3); /* configurando pin como salida a servo medio*/
  servou.attach(4); /* configurando pin como salida a servo ultimo (codo)*/
  servoc.attach(5); /* configurando pin como salida a servo de la muñeca (cabeceo)*/
  servor.attach(6); /* configurando pin como salida a servo de la muñeca (rotacion)*/
  servop.attach(7);/* configurando pin como salida a servo de la muñeca (pinza)*/

}
//La siguiente Funcion es la que corre el codigo que estará ejecutando constantemente, pero
//se precinde de el.

```

Figura 131 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 1).

```

void loop() {
  // put your main code here, to run repeatedly:
}
//En vez de la anterior funcion, la siguiente funcion se usa para que permita ejecutar codigo
//solo cuando el Serial este enviando o recibiendo datos.
void serialEvent()//solo si el serial tiene actividad de envio o recibo de datos
  //se ejecuta el codigo
{
  if(Serial.available())//si el serial esta disponible se leen los datos que se estan recibiendo a través de este
  {
    angba=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo
    diferente de un numero, es decir, el caracter"coma" (,) y se guarda en la variable  angba*(angulo base)*/
    angme=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    diferente de un numero es decir, el caracter "coma" (,) y se guarda en la variable  angme*(angulo medio)*/
    angul=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    diferente de un numero es decir, el caracter "coma" (,) y se guarda en la variable  angul*(angulo ultimo)*/
    angca=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    diferente de un numero es decir, el caracter "coma" (,) y se guarda en la variable  angca*(angulo cabeceo)*/
    angro=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    diferente de un numero es decir, el caracter "coma" (,) y se guarda en la variable  angro*(angulo rotacion)*/
    angpi=Serial.parseInt();//esta función lee un entero hasta que se encuentre algo diferente
    diferente de un numero es decir, el caracter "coma" (,) y se guarda en la variable  angpi*(angulo rotacion)*/

    //con los datos concatenados en labview, al final se agregó un salto
    //de linea. con eso se indica que se envio toda una cadena de caracteres y sirve para
    //diferenciarla de otras diferentes con angulos diferentes
  }
}

```

Figura 132 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 2).

```

if(Serial.read()=='\n')//al detectarse el salto de linea, es decir el fin de la
  //cadena de caracteres, termina la lectura de datos del serial.
{
  servob.write(angba); //se escribe el valor de "angulo base" en servo base
  servom.write(angme); //se escribe el valor de angulo medio en servo medio
  servou.write(angul); //se escribe el valor de angulo ultimo en servo ultimo
  servoc.write(angca); //se escribe el valor de angulo cabeceo en servo cabeceo
  servor.write(angro); //se escribe el valor de angulo rotacion en servo rotacion
  servop.write(angpi); //se escribe el valor de angulo balanceo en servo pinza
}
}

```

Figura 133 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 3).

Programa de simulación del robot LeArm en Matlab

El siguiente programa se desarrolló en la plataforma de Matlab y con las librerías de “Robotics Toolbox”. Se busca representar a un robot de 5 grados de libertad con configuración antropomórfica.

El programa considera que se han obtenido los parámetros de Denavit- Hartenberg con anterioridad y se hace uso de ellos vinculándolos con las articulaciones que componen al robot bajo estudio.

Las partes amarillas son la programación desarrollada en el lenguaje de la plataforma y con gris se subrayan los comentarios del programa. Se muestra en la figura 134 el mismo código implementado en la ventana de Matlab.

%modelo virtual de robot de seis grados de libertad

a2=10.5; /*valor de “a” asociado a la articulación dos.

```
a3=8.9; /*valor de "a" asociado a la articulación 3.
```

```
alfa1=pi/2; /*valor de "alfa" asociado a la articulación uno
```

```
d1=9.6; /*valor de "d" asociado a la articulación 1
```

Las siguientes expresiones "L1,L2,L3,L4,L5" representan a los grados de libertad que componen al robot. Todas son articulaciones de rotación "Revolute"

Se definen de la siguiente forma:

```
Revolute('variable "a" de la articulación', valor de la variable "a", 'variable 'alpha' de la articulación, valor de alpha, valor 'd' de la articulación , valor d);
```

```
L1= Revolute ('a',0,'alpha', alfa1,'d', d1);
```

```
L2= Revolute ('a', a2,'alpha',0,'d',0);
```

```
L3= Revolute ('a', a3,'alpha',0,'d',0);
```

```
L4= Revolute ('a',0,'alpha', pi/2,'d',0);
```

```
L5= Revolute('a',0,'alpha',0,'d',17.5);
```

```
bot=SerialLink([L1,L2,L3,L4,L5]); //se crea cadena cinemática con las articulaciones definidas anteriormente
```

```
%animación
```

```
ws= [-33,33,-33,33,-7,50]; //se definen los rangos del espacio de trabajo del robot
```

```
bot.teach([0,0,0,0,0],'workspace',ws,'noname');//Se define el lugar en que se posiciona el robot tomando un origen dentro del espacio de trabajo
```

```

Editor - C:\Users\omar\Documents\MATLAB\cinco_grados.m
cinco_grados.m  desacoplo.m  matriz_de_transformacion06.m  euler_a
1 - clear; close; clc;
2
3 %modelo virtual de robot de cinco grados de libertad
4 - a2=10.5;
5 - a3=8.9;
6 - alfa=pi/2;
7 - d1=9.6;
8
9 - L1= Revolute('a',0,'alpha',alfa,'d',d1);
10 - L2= Revolute('a',a2,'alpha',0,'d',0);
11 - L3= Revolute('a',a3,'alpha',0,'d',0);
12 - L4= Revolute('a',0,'alpha',pi/2,'d',0);
13 - L5= Revolute('a',0,'alpha',0,'d',17.5);
14 |
15 - bot=SerialLink([L1,L2,L3,L4,L5]);
16
17 %animacion
18 - ws= [-33,33,-33,33,-7,50];
19
20 - bot.teach([0,0,0,0,0],'workspace',ws,'noname');
21
22

```

Figura 134 Ventana del programa realizado en el software de Matlab para la simulación del robot LeArm de cinco grados de libertad.

Algoritmo para obtener ternas de las variables articulares q_1, q_2, q_3 en la cinemática directa implementada para el robot LeArm de 5GDL

Al algoritmo para escoger ternas como solución de cinemática inversa le precedió la elaboración de unas tablas de los valores de entradas y sus respectivas salidas para los programas de cinemática inversa y directa. Se crearon una tabla para la cinemática inversa y otra para la cinemática directa con 10 resultados cada una (ver tablas 14 y 15). Las tablas fueron llenadas a partir de los resultados obtenidos en los programas para implementar la cinemática directa e inversa, considerando únicamente los tres primeros grados de libertad del robot LeArm.

La tabla 14 contempla resultados de la cinemática directa, teniendo como entradas las variables articulares para los tres primeros grados de libertad. Como salidas se obtienen las coordenadas de posición, únicamente.

Tabla 14 Datos de entrada/salida para la cinemática directa del robot LeArm para los 3 primeros GDL.

Numero	Entradas			Salidas		
	q_1 [°]	q_2 [°]	q_3 [°]	X [cm]	Y [cm]	Z [cm]
1	45	45	135	5.25	5.25	25.925
2	0	0	0	10.5	-0	0.7
3	180	180	180	10.5	-0	0.7
4	170	150	80	15.669	-2.763	20.571
5	10	30	100	15.669	2.763	20.571

6	30	160	120	-16.135	-9.316	11.646
7	60	120	140	-7.007	-12.137	20.239
8	150	120	140	12.137	-7.007	20.239
9	150	0	160	-11.729	6.772	17.963
10	125	50	10	-8.292	11.842	13.193

Para la tabla 15 se muestran resultados de la cinemática inversa, se tienen como entradas las coordenadas de posición y las salidas son las variables articulares. Contrario a la cinemática directa, se obtienen varias soluciones posibles para cada variable articular. Para la segunda variable articular se muestran cuatro posibles; soluciones, para la tercera variable articular se muestran dos, para la primera solamente una solución.

Tabla 15 Datos de entrada/salida para la cinemática inversa del robot LeArm para los 3 primeros GDL.

Numero	Entradas			Salidas		
	X [cm]	Y [cm]	Z[cm]	q1 [°]	q2 [°]	q3 [°]
1	5.25	5.25	25.925	45	45,86.08, 93.91,134.99	44.99, -44.99
2	10.5	0	0.7	0	-80.57,0, -180, -99.42	90,-90
3	10.5	0	0.7	180	-80.57,0, -180, -99.42	90,-90
4	15.669	-2.763	20.571	-10	29.996,39.179, 140.82,150	10,-10
5	15.669	2.763	20.571	10	29.996,39.179, 140.82,150	10.01,- 10.01
6	-16.13	-9.316	11.646	-149.99	-7.469,-20, 159.95, 187.51	30,-30
7	-7.007	-12.137	20.239	-119.99	14.4,60, 120,165.59	50,-50
8	12.137	-7.007	20.239	-29.99	14.4,60, 120,165.59	50,-50
9	-11.72	6.772	17.963	149.99	0,63.39, 116.56,180	70.0,-70.0
10	-8.292	11.842	13.193	125	-22.08,50, 130,202.08	80,-80

Comparando las salidas de la cinemática directa, se observa que son iguales a los valores de entrada de la cinemática inversa. De forma contraria se comprueba también que las entradas de la cinemática directa con las salidas de la cinemática inversa corresponden con los mismos valores.

Con la comparación entre los valores de las tablas 14 y 15 se puede validar los dos programas implementados para cinemática inversa y directa.

La utilidad de las tablas permitió crear el algoritmo para escoger las ternas con los valores de las variables articulares q_1 , q_2 y q_3 . El algoritmo implementado se utilizó en el programa para implementar la cinemática inversa.

Para crear el algoritmo se requirió estudiar el patrón que permite formar las ternas con valores válidos para el rango de operación de los servomotores del robot.

Se amplió la tabla 15 para obtener un rango más amplio de posibles valores de entrada y salida. Los resultados se muestran en la tabla 16.

Se tabularon 30 resultados que contemplan de manera amplia las distintas zonas en que puede encontrarse la herramienta final del robot dentro de su volumen de trabajo.

Tabla 16 Datos de entrada/salida para la cinemática inversa del robot LeArm para los 3 primeros GDL (ampliación de la tabla 15).

No.	Entradas			Salidas				
	X [cm]	Y [cm]	Z [cm]	q_1 [°]	q_2 [°]		q_3 [°]	
1	5.25	5.25	25.925	45	45	86.084	44.994	-44.994
				45	93.915	134.997		
2	10.5	0	0.7	180	-80.57	0	90	-90
				0	-180	-99.429		
3	10.5	0	0.7	180	-80.57	0	90	-90
				0	-180	-99.429		
4	15.669	-2.763	20.571	-10	29.996	39.179	10	-10
				-10	140.821	150		
5	15.669	2.763	20.571	10	29.996	39.179	10	-10
				10	140.821	150		
6	-16.135	-9.316	11.646	-149.99	-7.469	20	30	-30
				-149.99	159.997	187.469		
7	-7.007	-12.137	20.239	60	14.405	60	50	-50
				-119.99	119.998	165.595		
8	12.137	-7.007	20.239	150	14.405	60	50	-50
				-29.99	120	165.595		
9	-11.729	6.772	17.963	*150	0	63.391	70	-70
				149.99	116.608	180		
10	-8.292	11.842	13.193	125	-22.085	50	80	-80
				125	129.999	202.086		

11	-10.5	0	0.7	360=0	-80.57	0	90	-90
				180	-180	-99.429		
12	-10.5	0	0.7	360=0	-80.57	0	90	-90
				180	-180	-99.429		
13	0	10.5	0.7	90	-80.57	0	90	-90
					-180	-99.429		
14	0	-10.5	0.7	-90	-80.57	0	90	-90
					-180	-99.429		
15	-11.577	-9.714	14.599	-140	-13.392	50	70	-70
					130	193.392		
16	-9.719	1.714	23.716	*170	30	80	55	-55
				170	99.916	150		
17	-5.78	17.788	14.467	*108	9.99	19.181	10	-10
				108	160.818	170		
18	-13.616	7.861	9.718	150	-35.139	30	72	-72
					-210	-144.86		
19	-4.289	11.794	23.798	110	37.148	59.903	24.833	-24.833
				110	120.097	142.85		
20	-9.052	10.787	4.74	130	-55.081	17	80	-80
				130	-197	-124.918		
21	3.711	14.885	8.02	76	-39.759	27.998	75	-75
					-207.999	-140.24		
22	7.681	5.378	24.489	35	35	80.595	49.997	-49.997
				35	99.404	144.998		
23	7.891	2.114	26.691	15	52.99	75.908	25	-25
				14.997	104.091	127.003		
24	0	0	29	180	90	90	0	0
				0	90	90		
25	13.25	-3.55	23.318	165	44.71	45.292	0.634	-0.634
				-14.998	134.707	135.29		
26	7.874	-16.886	14.737	115	10.831	19.996	9.991	-9.99104
				-65	160	169.168		
27	6.649	-4.655	26.234	145	47.981	79.998	34.995	-34.995
				-34.996	100	132.018		
28	-5.525	-2.011	22.046	20	24.429	104.997	89.996	-89.998
				-159.99	75.002	155.57		
29	-5.564	-3.896	24.51	35	36	95	65	-65
				-145	84.99	143.985		
30	-7.145	-10.204	20.242	55	11.011	70	65	-65

				-125	109.997	168.988		
--	--	--	--	------	---------	---------	--	--

Para leer los resultados la tabla 16, es preciso conocer que los números están subrayados con amarillo y azul cuando representan, cada uno, una terna posible físicamente que está dentro de los rangos de operación de los servomotores, descartándose completamente los valores que no están subrayados en algún color. Se usan estos colores para diferenciar una terna de otra, siendo el valor de “q1” subrayado de amarillo, el valor original (escogido por el programa de cinemática inversa) juntos con los valores de “q2” y “q3” con los que forma una terna. Se subrayó de color azul un duplicado o un valor obtenido a partir del amarillo, con los respectivos valores de “q2” y “q3” con los que forma una terna válida. Estos valores validos fueron validados por la simulación de cinemática directa del robot LeArm en Matlab para demostrar que son parte de una solución. En el resultado “24” se usó el color verde para señalar un caso especial en que la articulación “q1” puede tomar cualquier valor, mientras los valores articulares “q2” y “q3” toman valores de “90°” y “0°”, respectivamente. Con dichos valores se puede alcanzar la misma posición marcada (0,0,29) para “x”, “y” y “z”, respectivamente

Antes de mencionar el procedimiento para formar las ternas, se explica la manera en que están ordenados los valores para “q1”, “q2” y “q3” en la tabla 16.

En las tablas 17, 18 y 19 se encuentran ordenados los valores distintos para cada variable articular. Para la variable articular “q1” se obtienen dos valores “q11” y “q12”. “q11” es el valor dado por el programa de cinemática inversa y a partir de este valor se obtiene la segunda variable “q12” sumándole 180 grados a “q11”, o duplicando el valor original.

Con los valores de “q11” y “q12” es posible formar hasta dos ternas, en algunas ocasiones una terna o no es posible formar ninguna.

Para la variable “q2” se obtienen 4 valores: “q21”, “q22”, “q23” y “q24”.

Para la variable “q3” se obtienen 2 valores: “q31” y “q32”.

Tabla 17 Valores de “q1” de la tabla 16.

q1 [°]
Segundo valor formado a partir del valor sombreado en amarillo que puede ser el primer valor o el primer valor más la suma de 180°
Primer valor de q1, si es negativo, se le suma 180° para convertirlo en un valor valido dentro del rango de operación de los servos.

Tabla 18 Valores de "q2" de la tabla 16.

q2 [°]	
<p>q21</p> <p>Este valor se usa cuando al valor original de "q1 no se le ha sumado 180°. (si el valor no entra en el rango de operación de los servos se le suma 360°, si sigue estando fuera de rango se descarta valor).</p>	<p>q22</p> <p>Este valor se usa cuando al valor original de "q1 no se le ha sumado 180°. (si el valor no entra en el rango de operación de los servos se le suma 360°, si sigue estando fuera de rango se descarta valor).</p>
<p>q23</p> <p>Este valor se usa cuando al valor original de "q1 se le ha sumado 180° (Si el valor no entra en el rango de operación de los servos se le suma 360°, si sigue estando fuera de rango se descarta valor).</p>	<p>q24</p> <p>Este valor se usa cuando al valor original de "q1 se le ha sumado 180° (Si el valor no entra en el rango de operación de los servos se le suma 360°, si sigue estando fuera de rango se descarta valor).</p>

Tabla 19 Valores de "q3" de la tabla 16.

q3 [°]	
<p>q31</p> <p>Este valor forma, únicamente, terna con los valores de "q21" y "q23".</p>	<p>q32</p> <p>Este valor forma, únicamente, terna con los valores de "q22" y "q24".</p>

A partir de los valores de variables articulares de la tabla 16 se crearon los algoritmos mostrados en las figuras 135 y 136. El dato ingresado al algoritmo es el valor de "q1" obtenido de la cinemática inversa para tres grados de libertad del robot LeArm. Se ingresa tanto al algoritmo "A" como al algoritmo "B".

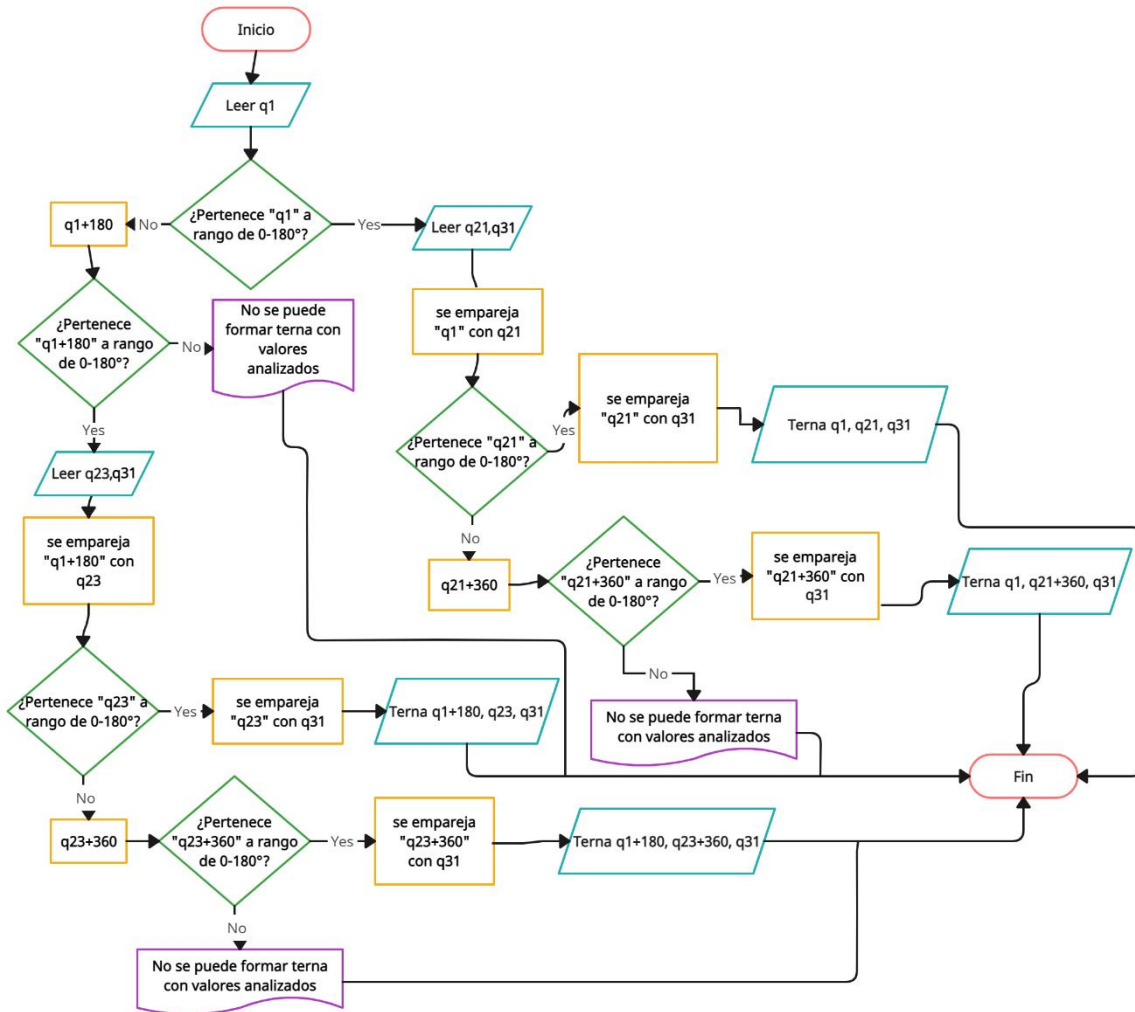


Figura 135 Algoritmo 'A' para formar ternas.

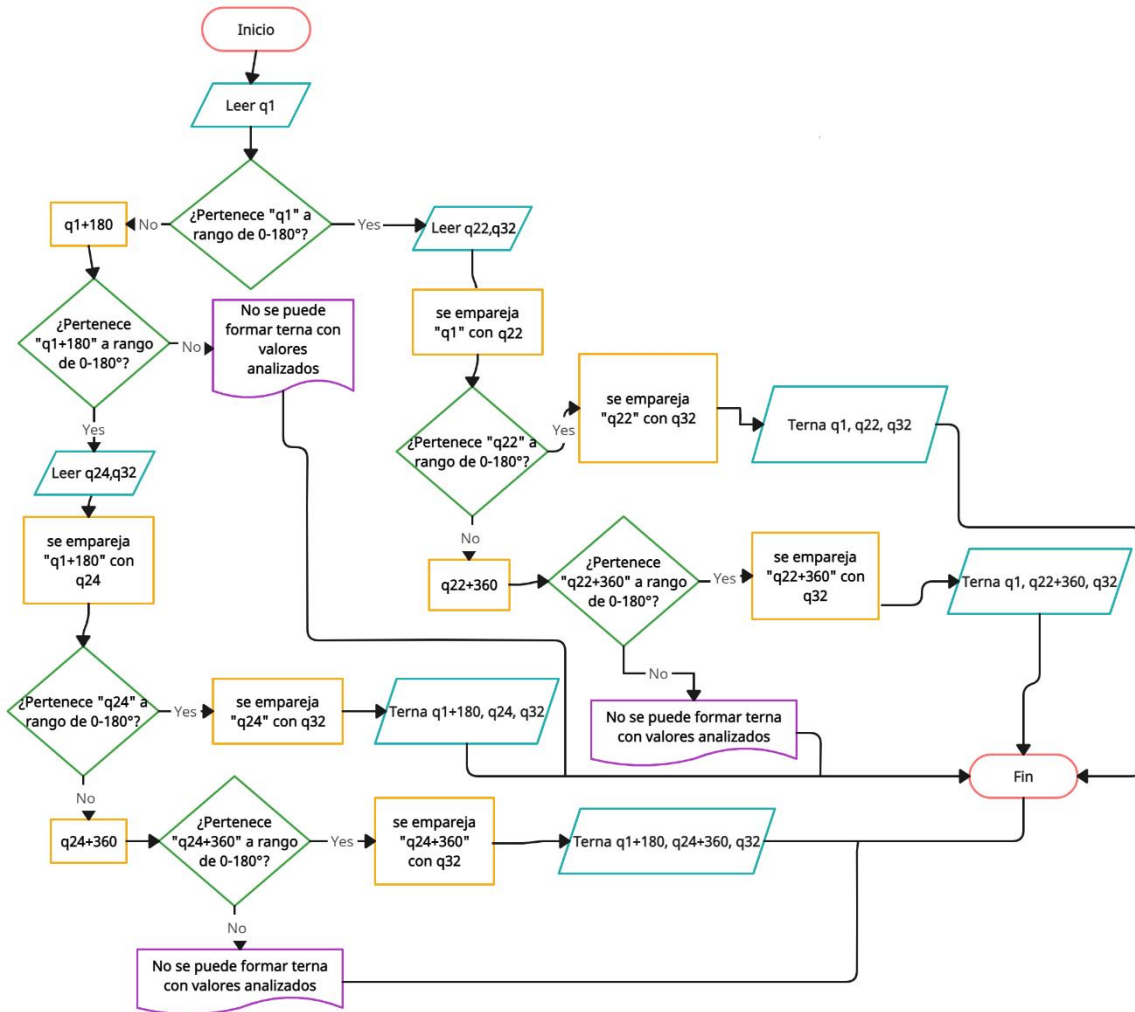


Figura 136 Algoritmo 'B' para formar ternas.

Los algoritmos "A" y "B" fueron la base para crear el algoritmo programado en "c" para el **programa para ejecutar la cinemática inversa**.

En el apartado para mostrar el panel de control del **programa para ejecutar la cinemática inversa** no se mostró el programa en "c" por ser demasiado largo para mostrar en pantalla. Se presenta el programa detallado donde se corrigen los detalles y restricciones no contempladas en los algoritmos "A" y "B". Las partes amarillas son las instrucciones en el lenguaje programado y las partes en gris son los comentarios que dan más detalles de las variables y otros elementos a especificar.

Algoritmo para seleccionar ternas programado en lenguaje "c" para el software LabVIEW.

float Q11,Q12,Q21,Q22,Q31,Q32,a,b,c,d1,d2,d3,d4;

*/*variables de Q11 a Q32 son salidas, representan los valores de las ternas.*/**

/* a y b son variables auxiliares binarias para conocer el rango de q1.*/

/* la variable "C" es una variable auxiliar que representa el valor de los tres casos posibles para obtener una o más ternas.*/

/*las variables d1,d2,d3,d4 representan a variables binarias que indican si las variables q12,q22,q23,q24 son valores posibles físicamente, es decir, que están dentro del rango especificado. */0020

/Análisis del rango en que se encuentra la variable q1. debe estar en el rango de 0 a 180 para ser físicamente posible.

if (q1>=0&&q1<=180) /si q1 está entre 0 y 180 grados.

a=1; / valor asignado 1 representa que es verdad que se encuentra en rango.

else

a=0; / valor asignado 0 representa que es falso que se encuentra en rango.

if((q1+180)>=0&&(q1+180)<=180) /analizando si al sumarle 180 grados a q1 se encuentra en rango de 0 a 180.

b=1; / valor asignado 1 representa que es verdad que se encuentra en rango.

else if((q1+180)==360) / para un valor de 360 grados se considera como 0 grados.

b=1; / valor de 1 quiere decir que como el valor de variable es 360 se considera dentro del rango al ser 0.

Else /Caso contrario.

b=0; / valor asignado 1 representa que es verdad que se encuentra en rango.

/*A partir de los valores de a y b se forman los casos posibles 1,2 y 3 asignado a la variable auxiliar "c". El primero es un caso donde hay un único valor para dos posibles valores de terna en Q11 y Q12 donde el valor para cada uno es el mismo valor dado por el valor de entrada q1. El segundo caso se distingue porque hay un único valor para las dos posibles salidas Q11 y Q12, este valor está dado por la expresión q1+180, cuando el valor de q1 es un valor negativo que es físicamente imposible y hay que sumarle 180. El tercer caso se da cuando puede haber dos valores posibles para Q11 y q12 , que son q1 y q1+180 respectivamente. */

if(a==1&&b==0) /*analizando si el único valor posible es q1 verificado previamente que está dentro de rango.*/

```
c=1; /*al ser cierto se asigna a "c" el valor de uno para que se ejecute únicamente el caso uno más adelante.*/
```

```
else if (a==0&&b==1) /*analizando si el único valor posible es q1+180 verificado previamente que está dentro de rango.*/
```

```
c=2; /*al ser cierto, se asigna a c el valor de dos para que se ejecute únicamente el caso dos más adelante.*/
```

```
else if (a==1&&b==1) /*analizando si se tienen dos valores posibles,q1 y q1+180 verificado previamente que están dentro del rango.*/
```

```
c=3; /*al ser cierto se asigna a c el valor de tres para que se ejecute únicamente el caso tres, más adelante.*/
```

```
/*Analizando los valores q21, q22, q23, q24. Analizando estos valores, es posible formar más fácilmente las ternas y determinar cuándo existirán una sola terna o las dos posibles. Las variables auxiliares d1, d2, d3, d4 que están vinculadas a los valores q21, q22, q23, q24 respectivamente, indican en caso de tener un valor de uno, que es posible utilizar los valores de q21, q22, q23, q24 para formar ternas con dichos valores, en combinación con Q12 y Q12. En caso contrario, las variables "d" tendrán un valor de cero y sus respectivos valores de "q" asociados no podrán ser utilizados para formar ternas en combinación con Q11 y Q12 */
```

```
if(q21>=0&&q21<=180){ /* analizando si q21 se encuentra en el rango de 0 a 180.
```

```
q21=q21; /* al estar dentro del rango el valor de q21 se queda igual.
```

```
d1=1;} /* al existir un valor valido de Q21 se asigna un valor de 1 a d1
```

```
else if ((q21+360)>=0&&(q21+360)<=180){ /* en caso contrario de que q21 no esté en el rango se le suma 360 y se verifica que esté en el rango de 0 a 180°. */
```

```
q21=q21+360; /* en caso de ser cierto al valor de q21 se le suma 360 grados.
```

```
d1=1; /*al existir un valor de q21 se le asigna un valor de "1" a d1.
```

```
}
```

```
else if((q21+360)==360){ /* en caso de que la suma de q21+360 grados de un valor de 360°...
```

```
q21=0; /* a q21 se le asigna un valor de cero.
```

```
}
```

```
else
```


d1=0; /* en caso de no existir un valor de q21 a d1 se le asigna un valor de cero.

if(q22>=0&&q22<=180){ /* analizando si q22 se encuentra en el rango de 0 a 180.

q22=q22; /* al estar dentro del rango el valor de q22 se queda igual.

d2=1;} /* al existir un valor se le asigna un valor de "1" a d2.

else if ((q22+360)>=0&&(q22+360)<=180){ /* en caso contrario de que q22 no esté en el rango se le suma 360 y se verifica que esté en el rango de 0 a 180°. */

q22=q22+360; /* en caso de ser cierto al valor de q22 se le suma 360 grados.

d2=1; /*al existir un valor de q22 se le asigna un valor de "1" a d2

}

else if((q22+360)==360){ /* en caso de que la suma de q22+360 grados de un valor de 360°...

q22=0; /* a q22 se le asigna un valor de cero.

}

else

d2=0; /* en caso de no existir un valor de q22 a d2 se le asigna un valor de cero.

if(q23>=0&&q23<=180){ /* analizando si q23 se encuentra en el rango de 0 a 180.

q23=q23; /* al estar dentro del rango el valor de q23 se queda igual.

d3=1;} /*al existir un valor se le asigna un valor de "1" a d3.

else if ((q23+360)>=0&&(q23+360)<=180){ /* en caso contrario de que q23 no esté en el rango se le suma 360 y se verifica que esté en el rango de 0 a 180°. */

q23=q23+360; /* en caso de ser cierto al valor de q23 se le suma 360 grados

d3=1;} /*al existir un valor de q23 se le asigna un valor de "1" a d2.

else if((q23+360)==360){ /* en caso de que la suma de q23+360 grados de un valor de 360°...

q23=0; /*a q23 se le asigna un valor de cero.

}

else

```
d3=0; /* en caso de no existir un valor de q22 a d3 se le asigna un valor de cero.
```

```
if(q24>=0&&q24<=180){ /* analizando si q24 se encuentra en el rango de 0 a 180.
```

```
q24=q24; /* al estar dentro del rango el valor de q24 se queda igual.
```

```
d4=1;} /* al existir un valor se le asigna un valor de "1" a d4.
```

```
else if ((q24+360)>=0&&(q24+360)<=180){ /* en caso contrario de que q24 no esté en el rango se le suma 360 y se verifica que esté en el rango de 0 a 180°. */
```

```
q24=q24+360; /* en caso de ser cierto al valor de q24 se le suma 360 grados.
```

```
d4=1;} /* al existir un valor de q24 se le asigna un valor de "1" a d2.
```

```
else if((q24+360)==360){ /* en caso de que la suma de q24+360 grados de un valor de 360°...
```

```
q24=0; /* a q24 se le asigna un valor de cero.
```

```
}
```

```
else
```

```
d4=0; /* en caso de no existir un valor de q24 a d4 se le asigna un valor de cero.
```

/* A continuación se analizan los casos c=1, c=2,c=3. caso c=1 cuando se tienen dos valores iguales correspondientes al valor original de q1. Caso c=2 es cuando los dos valores de q1 son igual y tiene como valor la suma de q1+180. El caso c=3 es cuando se tiene dos valores diferentes para q11 y q12, estos valores es el valor original de q1 y q1+180.

```
if(c==1){ /* en caso de que sea este caso c=1.
```

```
if(d1==1&&d2==1){ /* se verifica que existan dos valores reales de q21 y q22, se asignan los demás valores.
```

```
Q11=q1;
```

```
Q12=q1;
```

```
Q21=q21;
```

```
Q22=q22;
```

```
Q31=q31;
```

```
Q32=q32;
```

```
}
```

```
else if(d1==1&&d2==0){ /* en caso de que solo exista el valor de q21 solo se forma una terna y los demás valores se les asigna un valor de -666 que no está en el rango y se pueda verificar rápidamente por el usuario que no existen valores, puesto que no están en el rango. */
```

```
Q11=q1;
```

```
Q21=q21;
```

```
Q31=q31;
```

```
Q12=-666;
```

```
Q22=-666;
```

```
Q32=-666;
```

```
}
```

```
else if(d1==0&&d2==1){ /* en caso de que solo exista el valor de q22 solo se forma una terna y los demás valores se les asigna un valor de -666 que no está en el rango y se pueda verificar rápidamente por el usuario que no existen valores, puesto que no están en el rango. */
```

```
Q11=q1;
```

```
Q21=q22;
```

```
Q31=q32;
```

```
Q12=-666;
```

```
Q22=-666;
```

```
Q32=-666;
```

```
}
```

```
}
```

```
if(c==2){ /* analizando caso c=2.
```

```
if(d3==1&&d4==1){ /* si existen los valores q23 y q24 en el rango se asignan los demás valores.
```

```
Q11=q1+180; /* recordando que para este caso q11 y q12 son iguales y se les suma 180° al valor original.
```

Q12=q1+180;

Q21=q23;

Q22=q24;

Q31=q31;

Q32=q32;

}

else if(d3==1&&d4==0){ /* en caso de que solo exista el valor de q23 solo se forma una terna y los demás valores se les asigna un valor de -666 que no está en el rango y se pueda verificar rápidamente por el usuario que no existen valores, puesto que no están en el rango. */

Q11=q1+180;

Q21=q23;

Q31=q31;

Q12=-666;

Q22=-666;

Q32=-666;

}

else if(d3==0&&d4==1){ /* en caso de que solo exista el valor de q24 solo se forma una terna y los demás valores se les asigna un valor de -666 que no está en el rango y se pueda verificar rápidamente por el usuario que no existen valores, puesto que no están en el rango. */

Q11=q1+180;

Q21=q24;

Q31=q32;

Q12=-666;

Q22=-666;

Q32=-666;

}

```
}
```

```
if(c==3){ /* analizando el caso en que c=3.
```

```
if(d1==1&&d3==1){ /* si existen los valores q21 y q23 en el rango se asignan los demás valores.
```

```
Q11=q1;
```

```
Q12=q1+180;
```

```
Q21=q21;
```

```
Q22=q23;
```

```
Q31=q31; /* cuando se usan valores Q21 y Q23 siempre se emparejan con Q31.
```

```
Q32=q31; /* cuando se usan valores Q21 y Q23 siempre se emparejan con Q31.
```

```
}
```

```
else if(d1==1&&d4==1){ /* si existen los valores q21 y q24 en el rango se asignan los demás valores.
```

```
Q11=q1;
```

```
Q12=q1+180;
```

```
Q21=q21;
```

```
Q22=q24;
```

```
Q31=q31; /* cuando se usan valores Q21 y Q23 siempre se emparejan con Q31.
```

```
Q32=q32; /* cuando se usan valores Q22 y Q24 siempre se emparejan con Q32.
```

```
}
```

```
else if(d2==1&&d3==1){ /* si existen los valores q22 y q23 en el rango se asignan los demás valores.
```

```
Q11=q1;
```

```
Q12=q1+180;
```

```
Q21=q22;
```

```
Q22=q23;
```

```
Q31=q32; /* cuando se usan valores Q22 y Q24 siempre se emparejan con Q32.
```

```
Q32=q31; /*cuando se usan valores Q21 y Q23 siempre se emparejan con Q31.
```

```
}
```

```
else if(d2==1&&d4==1){ /* si existen los valores q23 y q24 en el rango se asignan los demás valores.
```

```
Q11=q1;
```

```
Q12=q1+180;
```

```
Q21=q22;
```

```
Q22=q24;
```

```
Q31=q32; /*cuando se usan valores Q22 y Q24 siempre se emparejan con Q32.
```

```
Q32=q32; /*cuando se usan valores Q22 y Q24 siempre se emparejan con Q32.
```

```
}
```

```
}
```

```
if(Q11==0&&Q12==0&&Q21==0&&Q22==0&&Q31==0&&Q32==00){ /* hay varios casos en que no existen valores posibles de "q" para signar a las variables entonces se les asigna un valor de -666 para visualizar fácilmente que es imposible tal valor y que, por tanto, no existe.
```

```
Q11=-666;
```

```
Q12=-666;
```

```
Q21=-666;
```

```
Q22=-666;
```

```
Q31=-666;
```

```
Q32=-666;
```

```
}
```

Índice de figuras

Figura 1 Robot Unimate creado por la empresa UNIMATION.....	11
Figura 2 Tipos articulaciones. a)rotacional. b)prismática.	12
Figura 3 Robots manipuladores de acuerdo a su configuración de brazo y espacio de trabajo.	13
Figura 4 Grado de libertad.....	15
Figura 5 Mismo punto en el espacio, pero diferente configuración de robot manipulador.	19
Figura 6 División del problema cinemático inverso en dos partes.....	20
Figura 7 Ubicación del punto medio en un robot manipulador de cinco grados de libertad.	21
Figura 8 Diagrama de conexiones para la manipulación del robot de cinco grados de libertad.	22
Figura 9 Compañía que fabrica el robot LeArm.....	23
Figura 10 partes que componen al robot manipulador LeArm de 5 grados de libertad.	24
Figura 11 Medidas de la base y altura en una configuración vertical del robot.	25
Figura 12 Medidas de los diámetros de la base fija y del primer eslabón del robot.	25
Figura 13 Medias de los eslabones del robot manipulador de cinco grados de libertad.	26
Figura 14 Medidas de la pinza o efector final del robot.	27
Figura 15 Rango de movimiento del primer grado de libertad.....	28
Figura 16 Rango de movimiento del segundo grado de libertad.	28
Figura 17 Rango de movimiento del tercer grado de libertad.....	29
Figura 18 Rango de movimiento del cuarto grado de libertad.....	29
Figura 19 Rango de movimiento del quinto grado de libertad.	30
Figura 20 Rango de movimiento de la pinza o efector final del robot.....	30
Figura 21 Espacio de trabajo del robot de cinco grados de libertad.....	31
Figura 22 Espacio de trabajo con el que se trabaja en el presente trabajo de tesis.	32
Figura 23 Ancho de pulso que posiciona a un servomotor en 0, 90 y 180 grados, respectivamente.....	33
Figura 24 Adaptador de 7.5 volts que permite alimentar con corriente a los servomotores del robot LeArm.	35
Figura 25 Tarjeta TIVA TM4C129	36
Figura 26 Entorno de programación "Energía".....	38
Figura 27 Esquema conector USB.....	39
Figura 28 Panel de Control del entorno de programación de LabVIEW. Se ejemplifica con un programa para la manipulación de un robot de tres grados de libertad.	41
Figura 29 Diagrama de bloques del entorno de programación de LabVIEW. Se ejemplifica con un programa para la manipulación de un robot de tres grados de libertad.	42
Figura 30 "VISA configure Serial Port" permite configurar un puerto serial.....	43

Figura 31 Selección de puerto serial y de velocidad de datos.....	44
Figura 32 Bloque de "VISA Read Function".	44
Figura 33 Bloque de "VISA Write Function".	44
Figura 34 Bloque "VISA Close Function"	44
Figura 35 Configuración básica para escribir datos en el puerto serial, con funciones de VISA.	45
Figura 36 Figura 35 Configuración básica para leer datos en el puerto serial, con funciones de VISA.....	45
Figura 37 Asignación de sistemas de ejes para robot de cinco grados de libertad.	47
Figura 38 Enumeración de eslabones para el robot LeArm.	48
Figura 39 Enumeración de articulaciones para el robot LeArm.....	48
Figura 40 Localización del eje de giro para cada articulación del robot LeArm.....	49
Figura 41 Asignación del eje 'z' en los ejes de cada articulación.....	49
Figura 42 Asignación del origen del sistema cero y de los ejes 'x' 'y' y 'z' asociados a ese sistema, usando la regla de la mano derecha.	50
Figura 43 Regla de la mano derecha.	50
Figura 44 Asignación de los orígenes de cada sistema.	51
Figura 45 Asignación de cada eje 'x' en los sistemas de ejes.....	52
Figura 46 Asignación de los ejes 'y' en los sistemas de ejes.	52
Figura 47 Asignación del origen del último sistema y sus respectivos ejes.	53
Figura 48 Obtención del parámetro ' θ '.	54
Figura 49 Obtención del parámetro 'd'.	54
Figura 50 Obtención del parámetro 'a'.	55
Figura 51 Obtención del parámetro ' α '.	56
Figura 52 Esquema vectorial de las posiciones de origen, punto medio y punto final de la herramienta. ('Po', 'Pc' y 'Pf').	60
Figura 53 Resolución por el método geométrico para el robot LeArm.	62
Figura 54 Obtención de las variables articulares 'q2' y 'q3' por el método geométrico.	63
Figura 55 Triángulo obtuso formado por los lados 'l2', 'l3' y 'H'.....	64
Figura 56 Función coseno inverso.	66
Figura 57 Función tangente inversa o arco tangente.	66
Figura 58 Triángulos rectángulos que tienen a uno de sus ángulos a ' β ' y ' ϕ ', respectivamente.....	68
Figura 59 Ángulos de Euler para realizar la transformación de un sistema de coordenadas a otro.	77
Figura 60 Ejes unitarios correspondientes a los ejes de cada sistema comparado.	78
Figura 61 Ejemplo de una transformación con ángulos de Euler ' α ', ' β ' y ' γ ' con valores de 90° grados cada uno.....	79
Figura 62 Panel Frontal del Programa para control de un servomotor mediante ancho de pulso.....	86
Figura 63 Diagrama de bloques del Programa para control de un servomotor mediante ancho de pulso.	86
Figura 64 Panel frontal del Programa de interpolación.	88
Figura 65 Diagrama de bloques del Programa de interpolación.....	88

Figura 66 Diagrama de estados para una máquina de estados que representa una interfaz de usuario para crear una tabla con posicionamientos angulares de cada articulación y efector del robot manipulador de 5 grados de libertad.....	91
Figura 67 Panel Frontal del Programa para crear una rutina secuencial.	92
Figura 68 Diagrama de bloques del Programa para crear una rutina secuencial..	93
Figura 69 Vista de caso "inicio" de la estructura "Case".....	94
Figura 70 Vista de caso "espera" de la estructura "Case".....	95
Figura 71 Vista de caso "ángulo" de la estructura "Case".	96
Figura 72 Vista de caso "salida" de la estructura "Case".	97
Figura 73 Vista de la variable de tipo enum y de los "shift register" que permiten el cambio de estado en la máquina de estados.	99
Figura 74 Vista del uso del bloque "or" para parar el programa automáticamente o manualmente.	100
Figura 75 Compuerta "or" y su respectiva tabla de verdad.	100
Figura 76 Panel frontal del programa para cambio de estado.....	101
Figura 77 Diagrama de bloques del programa para cambio de estado.....	101
Figura 78 Panel frontal del Programa para ejecutar rutina secuencial.	103
Figura 79 Diagrama de bloques del Programa para ejecutar rutina secuencial. .	104
Figura 80 Panel frontal del Programa para la implementación de la cinemática directa.	107
Figura 81 Diagrama de bloques del Programa para la implementación de la cinemática directa.	108
Figura 82 Panel frontal del Programa para la implementación de la cinemática inversa.....	111
Figura 83 Diagrama de bloques del Programa para la implementación de la cinemática inversa.	112
Figura 84 Panel frontal del Programa para convertir ángulos de Euler a matriz de orientación.....	115
Figura 85 Diagrama de bloques del Programa para convertir ángulos de Euler a matriz de orientación.....	116
Figura 86 Panel frontal del Programa para obtener las variables articulares q4 y q5.	118
Figura 87 diagramas de bloques del Programa para obtener las variables articulares q4 y q5.	119
Figura 88 Panel frontal del Programa para seleccionar las ternas de variables articulares q1, q2, q3.....	121
Figura 89 Diagrama de bloques del Programa para seleccionar las ternas de variables articulares q1, q2, q3.	122
Figura 90 Creación de rutina secuencial con ayuda del programa para crear rutina secuencial.	125
Figura 91 Robot LeArm ejecutando una rutina secuencial y dejando a su paso un patrón.	127
Figura 92 Patrones dibujados por el robot LeArm en tres pruebas distintas.	128
Figura 93 Código para emular robot antropomórfico de cinco grados de libertad en la plataforma Matlab.....	129
Figura 94 Simulación gráfica que se obtiene al ejecutar el código para simular un robot antropomórfico de cinco grados de libertad.	130

Figura 95 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la primera prueba.	132
Figura 96 Configuración física que se obtiene al ingresar los datos mostrados en la figura 95 (primera prueba).....	133
Figura 97 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 95 (primera prueba).	133
Figura 98 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la segunda prueba.....	134
Figura 99 Configuración física que se obtiene al ingresar los datos mostrados en la figura 98 (segunda prueba).	135
Figura 100 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 98 (segunda prueba).....	136
Figura 101 Panel frontal del programa de cinemática directa con los datos ingresados y los obtenidos para la tercera prueba.....	137
Figura 102 Configuración física que se obtiene al ingresar los datos mostrados en la figura 101 (tercera prueba).	138
Figura 103 Configuración virtual que toma el robot antropomórfico virtual al ingresar los mismos datos de la figura 101 (tercera prueba).	139
Figura 104 Plano "XY" y base del robot dibujada para colocar al robot LeArm sobre este plano.....	140
Figura 105 Vista aérea de la ejecución de la rutina secuencial para recoger pelotas.	143
Figura 106 Vista frontal de la ejecución de la rutina secuencial para recoger pelotas.	144
Figura 107 Datos ingresados en la cinemática directa y sus respectivos datos de salida (primera prueba).	145
Figura 108 Datos de entrada y salida para la cinemática inversa de la primera prueba.....	146
Figura 109 Comprobación con la cinemática directa de la segunda solución mostrada en la figura 108.....	147
Figura 110 Datos ingresados en la cinemática directa y sus respectivos datos de salida (segunda prueba).....	148
Figura 111 Datos de entrada y salida para la cinemática inversa de la segunda prueba.....	149
Figura 112 Datos ingresados en la cinemática directa y sus respectivos datos de salida (tercera prueba).	150
Figura 113 Datos de entrada y salida para la cinemática inversa de la tercera prueba.....	151
Figura 114 Comprobación con la cinemática directa de la segunda solución mostrada en la figura 113.....	152
Figura 115 Área del plano XY sobre la que se dibujaron las figuras y letras.	153
Figura 116 Figura cuadrada con sus respectivos puntos que conforman a la figura para su trazado.	154
Figura 117 Figura de triángulo isósceles con sus respectivos puntos que conforman a la figura para su trazado.....	155
Figura 118 Letra 'O' o círculo con sus respectivos puntos que conforman a la figura para su trazado.	156

Figura 119 Letra 'M' con sus respectivos puntos que conforman a la letra para su trazado.	158
Figura 120 Letra 'A' con sus respectivos puntos que conforman a la letra para su trazado.	159
Figura 121 Letra 'R' con sus respectivos puntos que conforman a la letra para su trazado.	160
Figura 122 Trazado de figura cuadrada por el robot LeArm.....	161
Figura 123 Trazado de figura triangular por el robot LeArm.....	162
Figura 124 Trazado de letra 'O' por el robot LeArm.	162
Figura 125 Trazado de letra 'M' por el robot LeArm.	163
Figura 126 Trazado de letra 'A' por el robot LeArm.....	163
Figura 127 Trazado de letra 'R' por el robot LeArm.	164
Figura 128 Programa para la TIVA que se ejecuta con el programa de control de un servomotor mediante ancho de pulso.	169
Figura 129 Programa para la TIVA que se ejecuta con el programa para la implementación de la cinemática inversa (parte 1).	170
Figura 130 Programa para la TIVA que se ejecuta con el programa para la implementación de la cinemática inversa (parte 2).	171
Figura 131 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 1).....	172
Figura 132 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 2).....	173
Figura 133 Programa para la TIVA que se ejecuta con el programa ejecutar rutina secuencial (parte 3).....	173
Figura 134 Ventana del programa realizado en el software de Matlab para la simulación del robot LeArm de cinco grados de libertad.	175
Figura 135 Algoritmo 'A' para formar ternas.....	181
Figura 136 Algoritmo 'B' para formar ternas.....	182