UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

mim máster en ingeniería mecatrónica

Control de trayectoria de la simulación de un brazo robot de 5 grados de libertad, controlado mediante la plataforma C2000 Piccolo LAUNCHXL-F28027F

MEMORIA PRESENTADA POR:

John Germán Vera Luzuriaga

DIRECTOR:

Francisco Gimeno Sales

Valencia-España, enero 2017

AGRADECIMIENTOS

En primer lugar, agradezco a Dios por darme esta oportunidad de ver y sentir toda esta gran experiencia.

A Gabriela Santillán por ser la amiga que me apoyo para poder realizar mis estudios.

A mi esposa Lorena Peña León por ser el soporte de mi familia en mi ausencia de todo mi periodo de estudios académicos.

Agradezco a mis amigos de Valencia – España por su apoyo y por todas esas noches de estudio y discusión. Por sus palabras de aliento en este complejo trayecto de mi vida académica.

A mi primo Gabriel Larrea y Roberto Larrea por estar presentes apoyándome.

A mis Padres por ayudarme en mis momentos de necesidad tanto a la distancia como a mi familia.

Y finalmente, agradezco a Ford Motors Valencia por darme la oportunidad de conocer el manejo de un manipulador a nivel industrial.

DEDICATORIA

Dedico este trabajo a mi familia y a mi esposa que me apoyo en mis decisiones, y por ser un soporte para nuestra familia.

A mis padres por el esfuerzo y apoyo para mi crecimiento profesional.

También lo dedico a mis amigos y maestros de Ecuador y España que me ayudaron hacer posible culminar esta etapa de mi vida.

A mis amigos de España que en los pocos momentos de distracción hicimos una estrecha relación de amistad.

RESUMEN

El objetivo de este trabajo es realizar el control cinemático de un brazo robot de 5 grados de libertad, controlado por el DSC Piccolo C2000 F28027F, brazo robot basado en el manipulador de desarrollo MELFA RV-2AJ de Mitsubishi Industrial Robot y que se encuentra alojado en la página web de Proyectos Robóticos[1]. Siendo este brazo robot modificado en sus aspectos dimensionales e implementada la simulación en FreeBasic.

Se desarrolla el modelo cinemático desde la obtención de su tabla de parámetros de Denavit Hartenberg, estándar para desarrollar la cinemática directa (apartado 1.2), la misma que tiene como datos de entrada, ángulos de los ejes y devuelve posiciones cartesianas. Seguido del desarrollo algebraico de la cinemática inversa (apartado 1.3) que tiene como datos de entrada posiciones cartesianas y devuelve valores angulares.

Posteriormente se desarrolla el Jacobiano (apartado 1.4) del manipulador, con el objetivo de determinar las singularidades (apartado 1.5) del brazo por medio de la determinante del Jacobiano, teniendo en cuenta otros factores para su mayor control.

Determinado la cinemática directa e inversa se procede a realizar la planeación de la trayectoria (apartado 2), aplicando un polinomio de orden superior para generar la trayectoria, para que se desarrolle en el espacio articular como cartesiano.

Realizado el cálculo, se procesan estos datos y se los implementa en el software utilizando el programa Matlab para desarrollar los diferentes scripts que componen la cinemática del brazo robot (apartado 3).

Con el desarrollo cinemático y la implementación en Matlab se programa en Code Composer Studio (apartado 4) para programar el DSC. Se realiza la comprobación del control cinemático del brazo robot simulado dando tres distintas trayectorias para el manipulador.

[1] https://sites.google.com/site/proyectosroboticos/cinematica-inversa-iii

SUMMARY

The objective of this work is to perform the kinematic control of a robot arm of 5 degrees of freedom (DOF), controlled by the DSC Piccolo C2000 F28027F, robot arm based on the manipulator of development MELFA RV-2AJ of Mitsubishi Industrial Robot. Being this robot arm modified in its dimensional aspects and implemented the simulation in FreeBasic.

The kinematic model is developed from obtaining its Denavit Hartenberg parameter table, standard for the development of direct kinematics (Section 1.2), which has axes angles as input data and returns Cartesian coordinates. Right after, it's followed by the algebraic development of inverse kinematics (section 1.3) which has Cartesian coordinates as input data and returns angular values.

Subsequently, the Jacobian (Section 1.4) of the manipulator is developed, aiming at determining the singularities (Section 1.5) of the arm using the Jacobian determinant, taking into account other factors for its greater control.

Determining the direct and inverse kinematics, we proceed to plan the trajectory (Section 2), applying a higher-order polynomial to generate the trajectory, so that it develops in both, articular space and Cartesian.

After the calculation, these data are processed and implemented in the software using the Matlab program to develop the different scripts that make the robot arm kinematics up (Section 3).

Having the kinematic development and the implementation done in Matlab, we proceed to program in Code Composer Studio (Section 4) to program the DSC. Finally, the kinematic control of the simulated robot arm is performed, giving three different trajectories for the manipulator.

RESUM

L'objectiu d'aquest treball es realitzar el control cinemàtic d'un braç robot de 5 graus de llibertat, controlat per el DSC Piccolo C2000 F28027F, braç robot basat en el manipulador de desenrotllament MELFA RV-2AJ de Mitsubishi Industrial Robot y que es troba allotjat en la pàgina web de "Proyectos Robóticos". Sent aquest braç robot modificat en els seus aspectes dimensionals e implementat la simulació en FreeBasic.

Es desenrotlla el model cinemàtic des de l'obtenció de la seua tabla de paràmetres de Denavit Hartenberg, estàndard per a desenvolupar la cinemàtica directa (apartat 1.2), la mateixa que te com a dades de entrada, angles dels eixos i retorna posicions cartesianes. Seguit del desenvolupament algebraic de la cinemàtica inversa (apartat 1.3) que te com a dades de entrada posicions cartesianes i retorna valors angulars.

Posteriorment es desenrotlla el Jacobià (apartat 1.4) del manipulador, amb l'objectiu de determinar les singularitats (apartat 1.5) del braç per mitjà de la determinant del Jacobià, tenint en conter altres factors per al seu major control.

Determinat la cinemàtica directa i inversa es procedeix a realitzar el planejament de la trajectòria (apartat 2), aplicant un polinomi d'orde superior per a generar la trajectòria, per a que es desenrotlle en l'espai articulat com a cartesià.

Realitzat el càlcul, es processen aquestes dades i se'ls implementa en el software utilitzant el programa Matlab per a desenrotllar els diferents scripts que componen la cinemàtica del braç robot (apartat 3).

Amb el desenrotllament cinemàtic i la implementació en Matlab es programa mitjançant Code Composer Studio (apartat 4) per a programar el DSC. Es realitza la comprovació del control cinemàtic del braç robot simulat donant tres distintes trajectòries per al manipulador.

CONTENIDO

TABLA DE ILUSTRACIONES

TABLAS

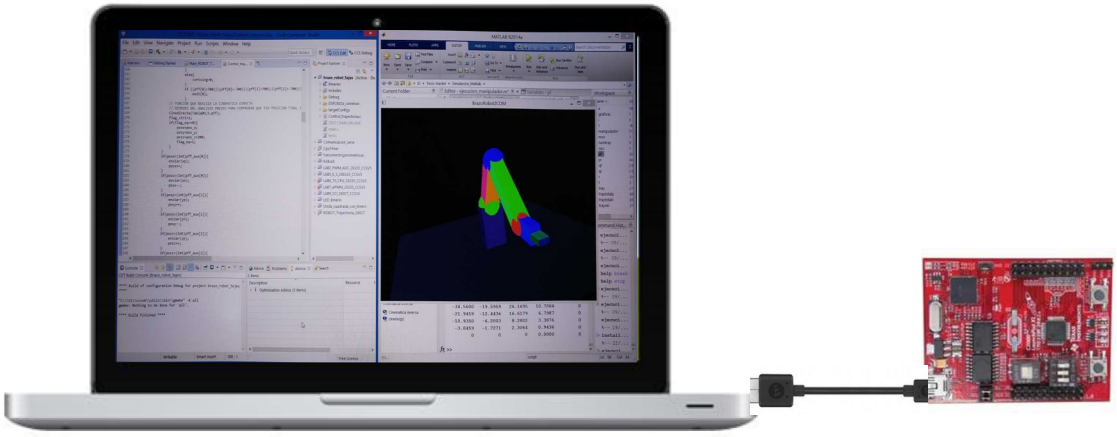
OBJETIVOS

OBJETIVO PRINCIPAL

El objetivo principal del presente proyecto es controlar la trayectoria de un brazo robot de 5 GDL simulado por computadora con el software FreeBasic, controlado por la plataforma de desarrollo C2000 Piccolo LAUNCHXL-F28027F de National Instruments mediante la comunicación serie y programado en lenguaje C con el Code Composer Studio. Para demostrar la generación de la trayectoria, por medio de un algoritmo basado en un polinomio de quinto orden en función de las posiciones y ángulos, generados por los métodos de cinemática directa e inversa.

OBJETIVOS SECUNDARIO:

- Realizar la comunicación serie RS232 entre el DSC y la computadora para el control de trayectoria.
- Desarrollo del análisis cinemático directo e inverso para el movimiento del brazo robot de 5 ejes en el espacio articular.
- Por medio del uso de Matlab programar todos los módulos para llevar acabo el movimiento del manipulador en una simulación.



ANTECEDENTE Y APORTACIÓN DEL TEMA

El presente trabajo es un esfuerzo para demostrar los diferentes conocimientos adquiridos de la ingeniería mecatrónica mediante el uso de diferentes simulaciones y plataforma de desarrollo electrónico para controlar la trayectoria de un brazo robot, dado los puntos cartesianos por el usuario. Por medio del análisis matemático y de algoritmos para su correcto funcionamiento y para su futura implementación.

En la evolución de los robots industriales y de servicio para realizar una o varias tareas específicas, estos tienen una estrecha relación con los últimos cambios económicos mundiales, con efecto sobre la automatización de los procesos de manufactura y necesidades del ser humano o de la industria (CRAIG , 2006). Siendo esta la razón que en los últimos años la industria ha empezado a utilizar robots para la automatización de procesos repetitivos, peligrosos, de precisión, de fuerza, entre otros. Los robots industriales se los llaman manipuladores mecánicos o robots manipuladores, para referirse a este trabajo los llamaremos manipulador o brazo robot. Usando como referencia para el desarrollo de esta memoria, con la valiosa información de los libros como Robotics, vision and control Fundamental Algorithms in Matlab (Corke, 2013), Robótica (CRAIG , 2006), (Manual del microcontrolador TMS320F2802X Piccolo, 2016) e Introducción a la Robótica (Subir Kumar , 2010)

En la actualidad los robots se están desarrollando para su uso tanto en la educación como para otras tareas con interacción del ser humano, a estos últimos son llamados robots colaborativos (cobot) que están siendo insertados en el trabajo y vida cotidiana del ser humano. Los robots colaborativos funcionan de la misma manera que los manipuladores industriales con la diferencia que tienen menor par, un sistema de control y sensores para controlar los movimientos de su entorno para salvaguardar la integridad del ser humano y poder trabajar junto a él.

Mediante el uso de los DSC, en este caso de la familia C2000 de Texas Instruments. La placa de desarrollo Piccolo LAUNCHXL-F28027F (Manual del microcontrolador TMS320F2802X Piccolo, 2016) se programa para generar y controlar cada servo motor o eje de un brazo robot, sin embargo, como parte de la ingeniería primero se realiza el diseño en este caso el control cinemático de una simulación de un brazo robot programado en FreeBasic para temas didácticos o desarrollo.

Con este aporte se pretende hacer más fácil la comprensión a un nivel un poco más profundo de un robot manipulador desde el estudio del movimiento hasta el control de la trayectoria para su concepción y futura implementación según la tarea o la necesidad que el ser humano necesite suplir por simplificar un trabajo, reducir tiempos del mismo o por el peligro que este puede tener.

ESTRUCTURA DEL TRABAJO ACADÉMICO

El presente trabajo se desarrolla en 4 capítulos, el primer capítulo se trata del análisis mecánico del manipulador basado en el brazo robot RV-2AJ de 5 grados de libertad de la empresa Mitsubishi, la obtención de los parámetros de Denavit-Hartenberg para realiza la cinemática directa, para desarrollar la cinemática inversa por medio de una solución geométrica, seguido de la determinación de la matriz Jacobiana que determina las velocidades del efector final del manipulador y por ultimo obtener las singularidades del brazo robot y una breve referencia a la estática.

En el segundo capítulo se habla sobre la planeación de la trayectoria que es el control cinemático del robot para realizar la trayectoria en el espacio articular, entender el espacio cartesiano, tipos de trayectoria en el espacio articular y para poder generar la trayectoria que perfil es el más adecuado.

El tercer capítulo es la programación del brazo robot por bloques o varios scripts para su simulación en Matlab y por último el cuarto capítulo que es el objetivo de este trabajo es la programación del DSC **C2000 Piccolo LAUNCHXL-F28027F** basándose en la programación previa en Matlab para el control de la trayectoria del brazo simulado en tiempo real.

El control de la trayectoria de la simulación del brazo robot de 5 grados de libertad antes mencionado se limita en el análisis de su parte cinemática teniendo por razón que no se va a implementar físicamente, solo se tiene de información la posición inicial y final del efector final y ángulos de cada articulación para poder generar el control de la trayectoria usando un DSC.

INTRODUCCIÓN

Los robots a lo largo de la historia han sido y son algo que puede ser desde un juguete hasta los más grandes y complejos en las fábricas. A que existan mecanismos a nuestra semejanza que manipulen e interactúen con las tareas cotidianas, es la tendencia para el servicio del ser humano como una aspiradora automática o un brazo robot industrial.

Los robots no solo pueden ser semejantes al ser humano en alguna parte de su fisiología como un brazo sino también puede ser semejanza al de un animal o simplemente automatizar una necesidad, sin embargo, en este trabajo solo se referencia sobre el análisis del brazo robot de 5 grados de libertad.

Hoy por hoy gracias a que su estudio se ha profundizado en la automatización y desempeño en varias tareas, siendo la palabra robot usada para definir un conjunto de mecanismos y dispositivos para realizar una tarea específica, por esta razón ha tenido un gran impacto y un gran crecimiento en la industria en especial en la automotriz.

Como hecho histórico la palabra 'robot' viene de la palabra checa "Robota" que es trabajo duro o forzado. Escrita en la obra de teatro Rossum's Universal Robots (RUR) en 1921 escrita por el checo Karel Capek.

El escritor de ciencia ficción Isaac Asimov se imaginó al robot como un ayudante o sirviente de la humanidad, pero para que se cumple esa tarea un robot debería cumplir las famosas tres leyes de la robótica:

I. Un robot no debe dañar a un ser humano ni, por su inacción, dejar que un ser humano sufra daño.

II. Un robot debe obedecer las órdenes que le son dadas por un ser humano, excepto si estas entran en conflicto con la primera ley.

III. Un robot debe proteger su propia existencia, a menos que esta entre en conflicto con las dos primeras.

Con la creciente demanda de robots en las industrias, James L. Fuller (1999) con su libro "Robótica: Introducción, Programación y Proyectos", introdujo una cuarta ley:

IV. Un robot podrá tomar el trabajo de un ser humano, pero no debe dejar a esta persona sin empleo.

La robótica es una de las ciencias que integra varias áreas que son la mecánica, el control, la eléctrica, la electrónica e informática. Por lo que para su desarrollo en el ámbito industrial se requiere de un equipo de ingenieros. Que cada día investigan y desarrollan más funcionalidades que proporcionan más utilidades al robot para que puedan llegar a trabajar en las tareas más complicadas para el ser humano.

Los robots industriales se pueden dividir por las diferentes combinaciones en su estructura mecánica a continuación, se nombran 5 tipos de robot clasificados por su estructura mecánica:

- Robot cartesiano,
    - Se mueve entorno a sus ejes X, Y, Z de forma lineal, estos formando ángulos rectos.
- Robot cilíndrico,
    - Se mueve en su base de forma rotacional y sus otros dos ejes de forma lineal perpendiculares entre estos.
- Robot SCARA (Selective Compliant Articulated Robot Arm),
    - Es un robot de cuatro grados de libertad, es una combinación de las ventajas de los robot cartesianos y cilíndricos, siendo estos muy rápidos y flexibles en los ejes X e Y.
- Robot Polar o esférico,
    - Tiene dos articulaciones rotacionales y una articulación prismática.
- Robot antropomórfico (Es el brazo robot en estudio en el presente trabajo)
    - Este robot es el más usado en la industria, varia en configuraciones con más o menos grados de libertad según la necesidad y solo tiene articulaciones rotacionales.

CAPÍTULO 1. ANALISIS CINEMÁTICO

1. MECÁNICA DE UN BRAZO ROBOT

El análisis mecánico de un robot se lo realiza con la matemática en base a las posiciones y orientaciones en el espacio tridimensional, seguido de la cinemática en su forma directa e inversa, donde solo se estudia el movimiento del brazo robot sin considerar las fuerzas que lo provocan. Por último, el análisis dinámico del robot que es el análisis de las fuerzas y momentos necesarios para que se produzca el movimiento requerido del robot como ejemplo el brazo robot Melfa RV-2AJ de 5 grados de libertad de la empresa Mitsubishi. La simulación a controlar se basa en el brazo robot antes mencionado con cambios en las dimensiones de sus eslabones, siendo este una configuración de estructura mecánica antropomórfica o angular. Hay que tener en cuenta que el manipulador en estudio no tiene el cuarto eje de articulación (balanceo o yaw) en el efector final como se muestra en la siguiente *Ilustración 1*, solo tiene 5 grados de libertad correspondientes a los dos últimos a pitch(J_5) y a roll(J_6) ver en la *Ilustración 2*.

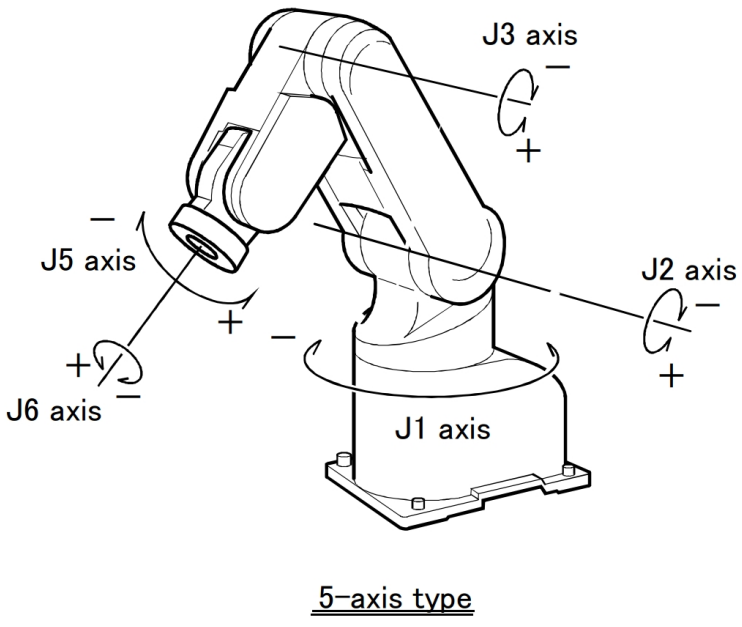


Ilustración 1, Robot RV-2AJ

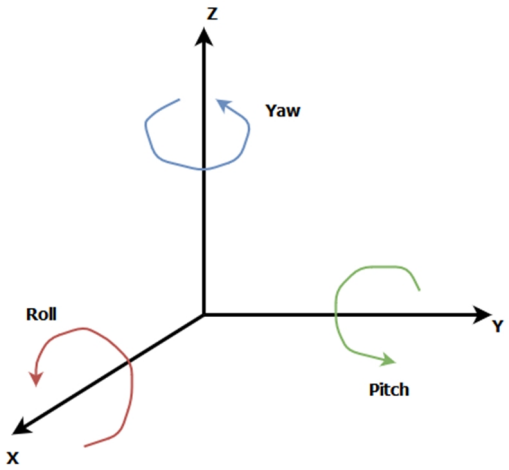


Ilustración 2, Roll, Pitch y Yaw coordenadas del efector final del manipulador (ángulos de Euler)

Donde,

Roll es el rotación o giro del efector final.

Pitch es el cabeceo o elevación del efector final.

Yaw es la balanceo o desviación en el efector final.

El problema de la cinemática directa es encontrar la posición y orientación del efector final o herramienta dado los ángulos en las articulaciones existiendo una sola solución, mientras que el problema cinemático inverso es un proceso un poco más complejo, se analiza cómo encontrar los ángulos de las articulaciones dada la posición y orientación de la herramienta por métodos geométricos o algebraicos pudiendo existir varias soluciones. La estructura mecánica del manipulador tiene una relación a la anatomía del brazo del ser humano que va desde su hombro hasta su mano, ver *Ilustración 3.*

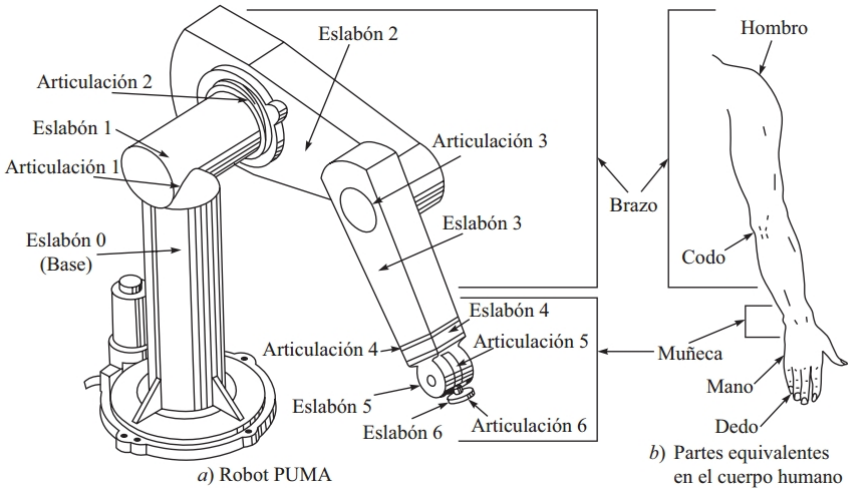


Ilustración 3, Equivalencia de un brazo robot de cadena abierta con un brazo humano. (Subir Kumar , 2010), imagen tomada del libro Introducción a la Robótica de Subir Kumar Saha, pág. 16.

Por medio del uso del criterio de Grubler-Kutzbach se comprueba que el manipulador RV-2AJ tiene 5 GDL (grados de libertad), el mismo criterio tiene la siguiente ecuación:

$$n = s(r - p - 1) + \sum_{i}^{p} n_i \qquad (1)$$

En donde los valores de los parámetros son:

s depende de la dimensión del espacio de trabajo, para mecanismos en el plano será 3 y para mecanismos en el espacio será 6.

r es el número de eslabones.

p es el número de articulaciones.

n_i es los grados de libertad relativos de cada articulación.

n son los grados de libertad de todo el mecanismo.

$$n = 6 \times (6 - 5 - 1) + 5 \times 1 = 5G \qquad (2)$$

En el valor de r se le da un valor de 6 por lo que se incluye la base fija como se aprecia en la *Ilustración 1*.

Este criterio determina los grados de libertad en las cadenas cinemáticas, es decir determina los grados de movilidad de un mecanismo. Este criterio tiene ciertas restricciones, según el tipo de mecanismo sin embargo para el presente mecanismo que es una cadena abierta el análisis es funcional.

1.1. CINEMÁTICA DEL ROBOT MANIPULADOR

1.1.1. CINEMÁTICA DIRECTA

En el siguiente desarrollo la cinemática directa nos ayudara a expresar el movimiento del brazo robot de 5 grados de libertad de tal manera que con los ángulos para las 5 articulaciones poder determinar la posición y orientación del efector final.

Los brazos robot o también llamados manipuladores se recomienda que tengan un solo grado de libertad por cada articulación para poder aplicar la notación o convención de Denavit-Hartenberg que se describirá más adelante.

Para el diseño mecánico de un robot hay que tener las siguientes consideraciones:

La mayoría de robots utilizan articulaciones angulares o de rotación como es el caso del RV-2AJ y/o combinados con articulaciones de deslizamiento o prismáticas.

Los vínculos o eslabones son los que unen a las articulaciones, se enumeran desde la base fija del robot con el numero 0 hasta el último eslabón n.

Los brazos robot tienen una configuración de cadena cinemática abierta, es decir tiene libertad para moverse en el espacio. Existen otras configuraciones de cadena cinemática cerrada como son los robots paralelo delta, cartesiano, cilíndrico entre otros en el cual su movimiento está en función a sus eslabones.

En el desarrollo matemático se usan las matrices de rotación y traslación para poder llegar a la matriz homogénea con tamaño de 4x4, es la que expresa la posición y orientación de un sistema de referencia O'UVW con respecto a uno fijo OXYZ.

$$ {}^{i-1}_{i}T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ f_{1x3} & W_{1x1} \end{bmatrix} = \begin{bmatrix} R & \text{ón} & T & \text{ón} \\ P & & E & \end{bmatrix} = \begin{bmatrix} R\ ta\ \text{ón} & Tr & \text{ón} \\ 0\ 0\ 0 & 1 & \end{bmatrix} \quad (3) $$

1.1.2. MATRICES DE ROTACIÓN

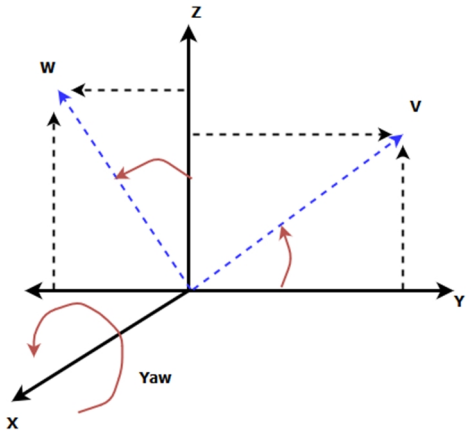
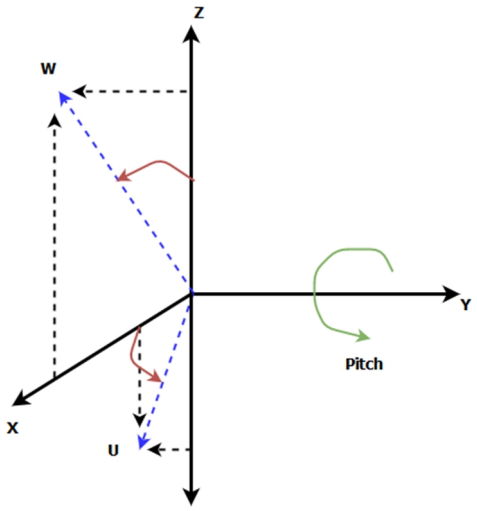


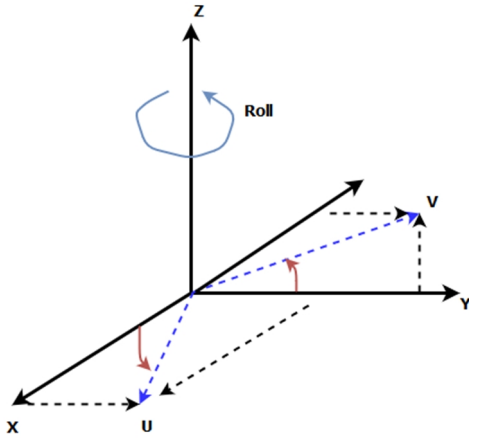
Ilustración 4, Rotación en el eje X

$$ R_{(x,\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad (4) $$



$$R_{(y,\varphi)} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \quad (5)$$

Ilustración 5, Rotación en el eje Y



$$R_{(z,\theta)} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Ilustración 6, Rotación en el eje Z

1.1.3. VECTOR DE TRASLACIÓN

Es un vector que contiene los valores de las posiciones de los ejes X, Y y Z.

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \quad (7)$$

1.1.4. MATRIZ HOMOGÉNEA

La matriz homogénea es el producto de matrices de transformación de rotación y traslaciones (es el producto entre la matriz identidad y el vector traslación) como se define a continuación:

En donde dos articulaciones se relacionan por un vínculo o eslabón por la matriz de rotación alrededor del eje Z_{i-1} con ángulo θ_i, por el producto de la matriz de traslación (matriz unitaria o identidad en producto por vector de traslación) a lo largo del Z_{i-1} con una distancia d_i las otras posiciones de los otros ejes son cero, luego por el producto de la matriz de traslación a lo largo de X_i con una distancia de a_i y por ultimo un producto por la matriz de rotación alrededor del eje X_i con ángulo α_i.

$$ {}^{i-1}_{i}A(\theta_i, d_i, a_i, \alpha_i) = T_{(z,\theta_i)} \times T_{(0,0,d_i)} \times T_{(a_i,0,0)} \times T_{(x,\alpha_i)} \quad (8) $$

$$ {}^{i-1}_{i}A = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9) $$

$$ {}^{i-1}_{i}A = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10) $$

1.1.5. ALGORITMO DE DENAVIT-HARTENBERG

Para el desarrollo de la cinemática directa se debe obtener los parámetros de Denavit-Hartenberg, teniendo en cuenta que las articulaciones i conectan a los eslabones $i-1$ a i para que la articulación i mueva al eslabón i. Los parámetros de Denavit-Hartenberg se pueden determinar de dos maneras, con el método estándar o modificado (presentado por John Craig en 1986), el presente trabajo se desarrolla usando el método estándar sin embargo más adelante se explica en que consiste el método modificado. El algoritmo de Denavit-Hartenberg tiene los siguientes pasos para su construcción sistemática:

1) Identificar cuantas articulaciones tiene el robot, (es según los grados de libertad que tiene el robot).
2) El número de articulaciones comienza en 1 hasta n comenzando por la base fija y terminando con el efector final.
3) Establecer el sistema de coordenadas en la base según la regla de la mano derecha (dextrógiro), en donde z_0 está situado a lo largo del eje de movimiento de la articulación 1.
4) Enumerar los eslabones comenzando con 0 como el eslabón de la base fija hasta el eslabón n.
5) Asignar el eje z de rotación para cada articulación rotativa.

6) Asignar los ejes x_i en la dirección perpendicular o normal común entre z_{i-1} y z_i, cuando están paralelos.
7) Asignar los ejes y de tal manera que cumpla con la regla de la mano derecha.
8) Para el ultimo sistema i-1 no aplica la convención mencionada ya que no existe un eslabón i por lo que puede elegirse el sistema de forma arbitraria.
9) Determinar los parámetros de Denavit-Hartenberg, en donde el eslabón están definido por dos parámetros (a_i para la longitud y α_i para la torsión) y la articulación está definido por dos parámetros (d_i la distancia de un sistema de coordenadas del eslabón hasta la articulación y θ_i el ángulo de rotación de la articulación). Se los obtiene siguiendo los siguientes puntos:
   a) a_i es la distancia de z_{i-1} a z_i medida sobre el eje x_i, longitud del eslabón.
   b) α_i es el ángulo formado entre z_{i-1} a z_i medido sobre el eje x_i, ángulo de torsión.
   c) d_i es la distancia de x_{i-1} a x_i o desde el origen del sistema $i-1$ a x_i medido a lo largo del eje z_{i-1}, distancia de desplazamiento de la articulación.
   d) θ_i es el ángulo formado entre x_{i-1} a x_i medido sobre el eje z_{i-1}, ángulo de la articulación.

Según como estén los ejes z_{i-1} y z_i ver la Ilustración 7, z_{i-1} y z_i no ejes son paralelos en donde z_{i-1} y z_i no son ejes paralelos o cuando z_{i-1} y z_i son ejes paralelos como se aprecia en la Ilustración 8, z_{i-1} y z_i son ejes paralelos, para determinar los parámetros de Denavit Hartenberg.

1.1.6. MÉTODO ESTÁNDAR DE DENAVIT-HARTENBERG

El origen $\{i-1\}$ se encuentra en la articulación i y a_i representa la longitud del eslabón i medida de z_{i-1} a z_i como se muestra en la Ilustración 9, Método estándar de Denavit-Hartenberg.

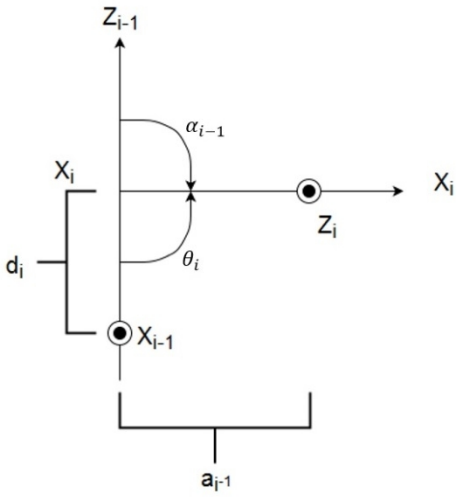


Ilustración 7, z_{i-1} y z_i no ejes son paralelos

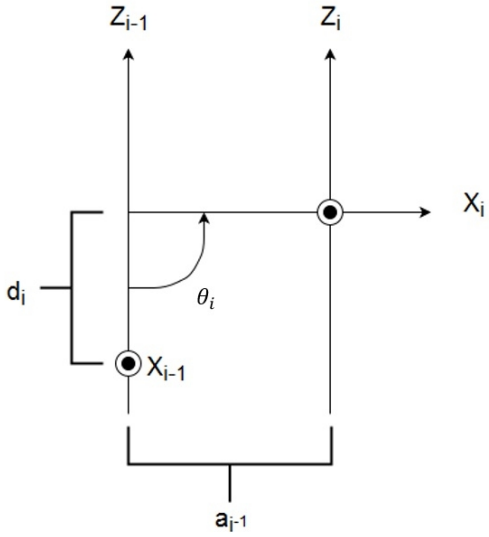


Ilustración 8, z_{i-1} y z_i son ejes paralelos

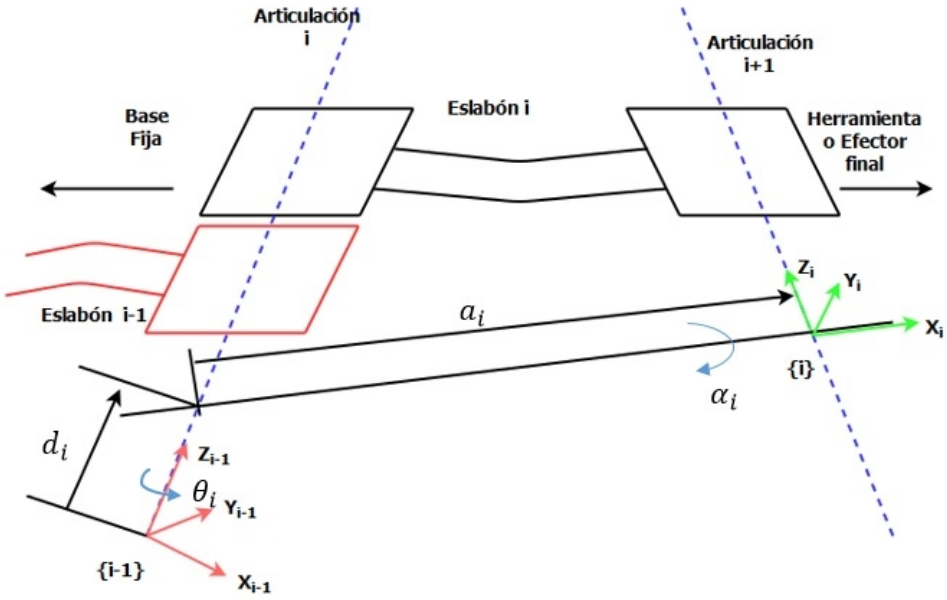


Ilustración 9, Método estándar de Denavit-Hartenberg

En donde la convención de la matriz de transformación homogénea para el método estándar de Denavit-Hartenberg es definida en la ecuación ***(10)***.

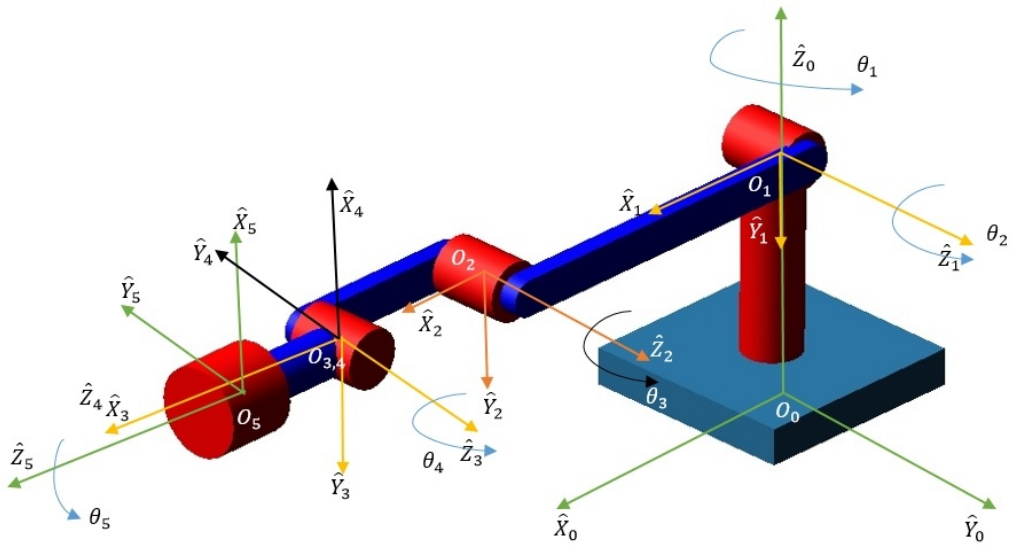


Ilustración 10, Posición inicial del brazo robot

i	a_i	α_i	d_i	Θ_i
1 (J_1)	0	$-\pi/2$	d_1	θ_1 (0)
2 (J_2)	a_2	0	0	θ_2 (0)
3 (J_3)	a_3	0	0	θ_3 (0)
4 (J_5)	0	$-\pi/2$	0	θ_4 $(-\pi/2)$
5 (J_6)	0	0	d_5	θ_5 (0)
T (Tool)	0	0	d_T	θ_5 (0)

Tabla 1, parámetros de Denavit-Hartenberg

En la Tabla 1, parámetros de Denavit-Hartenberg obtenida por el método estándar de Denavit-Hartenberg se puede apreciar como los ángulos de las articulaciones tienen entre paréntesis un ángulo, ese valor es tomado de la posición en la que se encuentra el manipulador según Ilustración 10, Posición inicial del brazo robot. Se podría agregar una fila más que sería la herramienta que tendría el valor del parámetro de desplazamiento de la articulación d_T y su ángulo es el mismo que el de la articulación 5.

Una vez determinada la tabla de DH se procede a obtener la matriz homogénea de cada articulación, en donde se procede a pre-multiplicar la anterior y post-multiplicar la actual articulación en donde se determinará la posición y orientación de la herramienta.

$$ {}^{0}_{1}A = \begin{bmatrix} C\theta_1 & 0 & -S\theta_1 & 0 \\ S\theta_1 & 0 & C\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11) $$

$$ {}^{1}_{2}A = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12) $$

$$ {}^{2}_{3}A = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & a_3C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & a_3S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13) $$

$$ {}^{3}_{4}A = \begin{bmatrix} C\theta_4 & 0 & -S\theta_4 & 0 \\ S\theta_4 & 0 & C\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14) $$

$$ {}^{4}_{5}A = \begin{bmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ S\theta_5 & C\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15) $$

$$ {}^{5}_{T}A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16) $$

$$ {}^{0}_{5}T = {}^{0}_{1}A \cdot {}^{1}_{2}A \cdot {}^{2}_{3}A \cdot {}^{3}_{4}A \cdot {}^{4}_{5}A \cdot {}^{5}_{T}A = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (17) $$

Como resultado del producto de las matrices tenemos la matriz de transformación T homogénea, donde P es el vector de posición y la matriz que contiene n, o y a es la de rotación o también llamada de orientación (n es el vector normal, o es el vector de orientación y a es el vector de aproximación asociado al eje z, los tres ceros de la última fila es la perspectiva y por último el escalado con valor unitario que está en la última fila y columna). Los elementos de la matriz de rotación son vectores ortonormales unitarios que describe la orientación del sistema móvil con respecto del fijo.

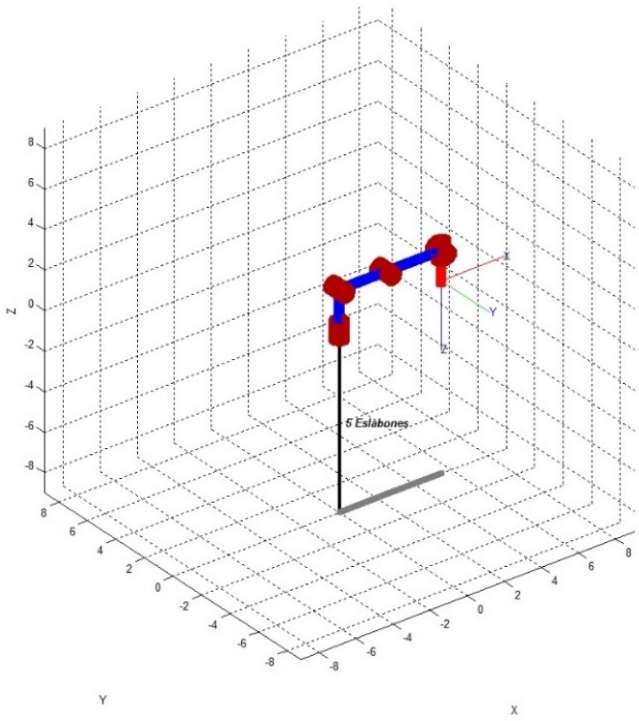


Ilustración 11, Resultado de los parámetros de DH

Resolución manual:

$T =$

$$n_x = S_1S_5 + C_5\big(C_4(C_1C_2C_3 - C_1S_2S_3) - S_4(C_1C_2S_3 + C_1C_3S_2)\big) \quad (18)$$

$$n_y = -C_1S_5 + C_5\big(C_4(S_1S_2S_3 - S_1C_2C_3) + S_4(S_1C_2S_3 + S_1C_3S_2)\big) \quad (19)$$

$$n_z = -C_5\big(C_4(C_2S_3 - S_2C_3) + S_4(C_2C_3 - S_2S_3)\big) \quad (20)$$

$$o_x = S_1C_5 - S_5\big(C_4(C_1C_2C_3 - C_1S_2S_3) - S_4(C_1C_2S_3 + C_1C_3S_2)\big) \quad (21)$$

$$o_y = -C_1C_5 + S_5\big(C_4(S_1S_2S_3 - S_1C_2C_3) + S_4(S_1C_2S_3 + S_1C_3S_2)\big) \quad (22)$$

$$o_z = S_5\big(C_4(C_2S_3 - S_2C_3) + S_4(C_2C_3 - S_2S_3)\big) \quad (23)$$

$$a_x = -C_4(C_1C_2S_3 - C_1S_2C_3) - S_4(C_1C_2C_3 - C_1S_2S_3) \quad (24)$$

$$a_y = S_4(S_1S_2S_3 - S_1C_2C_3) - C_4(S_1C_2S_3 + S_1S_2C_3) \quad (25)$$

$$a_z = S_4(C_2S_3 + S_2C_3) - C_4(C_2C_3 - S_2S_3) \quad (26)$$

$$P_x = a_2C_1C_2 - d_5\big(C_4(C_1C_2S_3 + C_1S_2C_3) + S_4(C_1C_2C_3 - C_1S_2C_3)\big) + a_3(C_1C_2C_3 - C_1S_2S_3) \quad (27)$$

$$P_y = a_2S_1C_2 - d_5\big(C_4(S_1C_2S_3 + S_1S_2C_3) - S_4(S_1S_2S_3 - S_1C_2C_3)\big) + a_3(S_1C_2C_3 - S_1S_2S_3) \quad (28)$$

$$P_z = d_1 - a_2S_2 - d_5\big(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + S_2C_3)\big) - a_3(C_2S_3 - S_2C_3) \quad (29)$$

Por Matlab,

$T =$

$$n_x = \sin\theta_1 \times \sin\theta_5 + \cos(\theta_2 + \theta_3 + \theta_4) \times \cos\theta_1 \times \cos\theta_5 \quad (30)$$

$$o_x = \cos\theta_5 \times \sin\theta_1 + \cos(\theta_2 + \theta_3 + \theta_4) \times \cos\theta_1 \times \sin\theta_5 \quad (31)$$

$$a_x = \sin(\theta_2 + \theta_3 + \theta_4) \times \cos\theta_1 \quad (32)$$

$$P_x = \cos\theta_1 \times (a_3 \times \cos(\theta_2 + \theta_3) + a_2 \times \cos\theta_2 - d_5 \times \sin(\theta_2 + \theta_3 + \theta_4)) \quad (33)$$

$$n_y = \cos(\theta_2 + \theta_3 + \theta_4) \times \cos\theta_5 \times \sin\theta_1 - \cos\theta_1 \times \sin\theta_5 \quad (34)$$

$$o_y = -\cos\theta_1 \times \cos\theta_5 - \cos(\theta_2 + \theta_3 + \theta_4) \times \sin\theta_1 \times \sin\theta_5 \quad (35)$$

$$a_y = \sin(\theta_2 + \theta_3 + \theta_4) \times \sin\theta_1 \quad (36)$$

$$P_y = \sin\theta_1 \times (a_3 \times \cos(\theta_2 + \theta_3) + a_2 \times \cos\theta_2 - d_5 \times \sin(\theta_2 + \theta_3 + \theta_4)) \quad (37)$$

$$n_z = -\sin(\theta_2 + \theta_3 + \theta_4) \times \cos\theta_5 \quad (38)$$

$$o_z = \sin(\theta_2 + \theta_3 + \theta_4) \times \sin\theta_5 \quad (39)$$

$$a_z = -\cos(\theta_2 + \theta_3 + \theta_4) \quad (40)$$

$$P_z = d_1 - a_3 \times \sin(\theta_2 + \theta_3) - a_2 \times \sin\theta_2 - d_5 \times \cos(\theta_2 + \theta_3 + \theta_4)) \quad (41)$$

En donde los C_1 es $\cos\theta_1$, S_2 es $\sin\theta_2$ y $\sin(\theta_2 + \theta_3)$ es una función trigonométrica de doble ángulo, para el seno y coseno tenemos:

$$\sin(\theta_1 \pm \theta_2) = \sin\theta_1 \cos\theta_2 \pm \cos\theta_1 \sin\theta_2$$

$$\cos(\theta_1 \pm \theta_2) = \cos\theta_1 \cos\theta_2 \mp \sin\theta_1 \sin\theta_2$$

$$^{0}_{5}T = \begin{bmatrix} 1 & 0 & 0 & 198 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 345 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (42)$$

La matriz homogénea resultante (42) de los parámetros de Denavit-Hartenberg, se puede ver como los valores corresponden a la orientación y posición del efector final de la Ilustración 11, Resultado de los parámetros de DH. La matriz tiene valor de uno positivo en x, de uno negativo en y y en z con respecto al sistema de coordenadas fijo de la base, y en la posición que es la última columna varía según la tabla de longitudes del robot simulado.

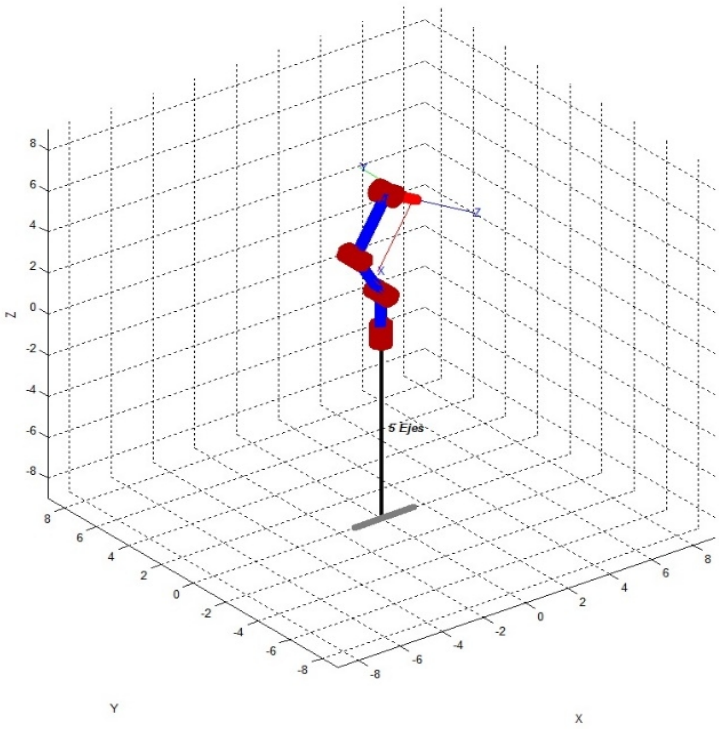


Ilustración 12, Robot en posición de reposo

$$ {}^{0}_{5}T = \begin{bmatrix} -0.5 & 0 & 0.866 & 154.9 \\ 0 & 1 & 0 & 0 \\ -0.866 & 0 & -0.5 & 601 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(43)} $$

Matriz **(*43*)** homogénea resultante de la Ilustración 12, Robot en posición de reposo, tiene la posición y orientación del efector final

1.1.7. MÉTODO MODIFICADO DE DENAVIT-HARTENBERG

El origen $\{i-1\}$ se encuentra en la articulación $i-1$ y a_i representa la longitud del eslabón i medida de z_i a z_{i+1} como se ilustra a continuación :

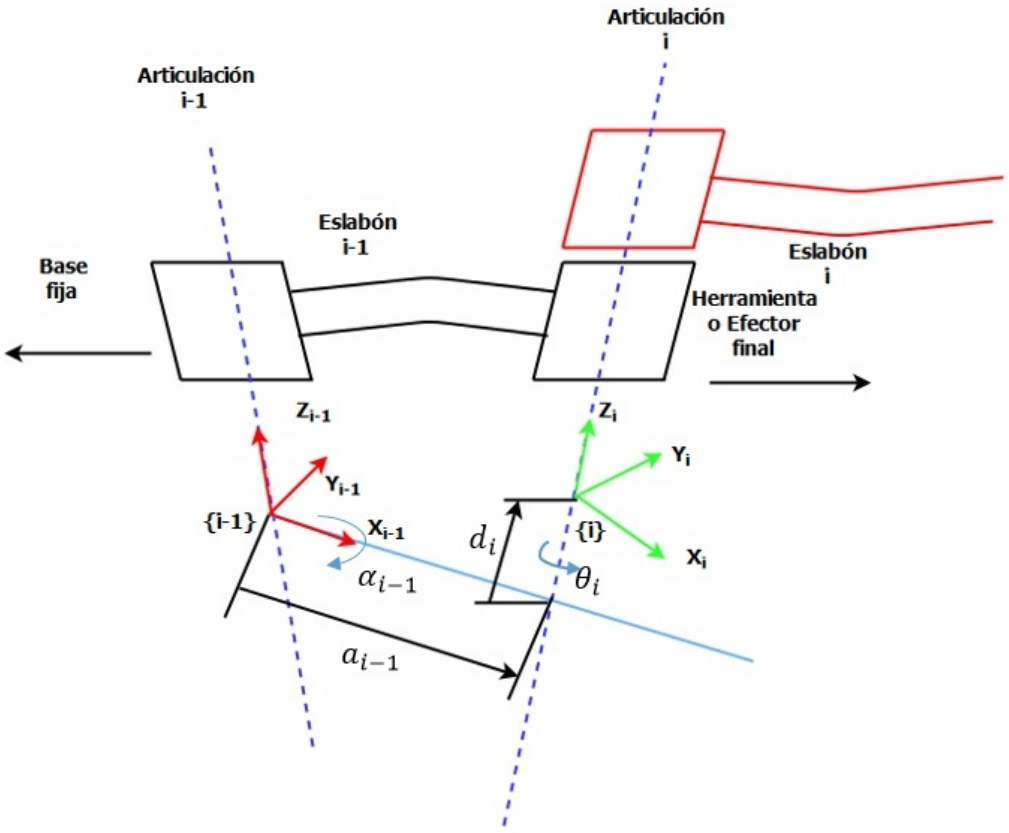


Ilustración 13, Método modificado de Denavith Hartenberg

En donde la convención de la matriz de transformación homogénea es definida por Craig de la siguiente manera:

$$ {}^{i-1}_{i}A(\alpha_{i-1}, a_{i-1}, \theta_i, d_i) = T_{(x,\alpha_{i-1})} \times T_{(a_{i-1},0,0)} \times T_{(z,\theta_i)} \times T_{(0,0,d_i)} \quad (44) $$

$$ {}^{i-1}_{i}A = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_i \\ C\alpha_{i-1}S\theta_i & -C\alpha_{i-1}C\theta_i & -S\alpha_{i-1} & -d_iS\alpha_{i-1} \\ S\alpha_{i-1}S\theta_i & -S\alpha_{i-1}C\theta_i & C\alpha_{i-1} & d_iC\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (45) $$

1.2. CINEMÁTICA INVERSA

En este apartado de la cinemática que es la que se usa en la industria por la razón que sabemos la posición y orientación del efector final, sin embargo es necesario determinar los ángulos de las articulaciones que son usados por el sistema de control, para el caso del robot ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$), en θ_5 no es necesario realizar cálculos para encontrarlo porque no afecta al movimiento del brazo, ya que es el ángulo de giro de la muñeca también conocido como "roll wrist".

Existen varios métodos resolución para la cinemática inversa siendo esta no sistemática, es específica para las diferentes combinaciones de un brazo robot.

Para este trabajo se escogió el método geométrico, el cual es analizar los tres primeros grados de libertad del brazo sin embargo para un reducido número de robot se puede obtener también por este método los últimos ángulos de las articulaciones del efector final o utilizar la solución por el método del desacoplo cinemático, para este caso no es necesario usarlo.

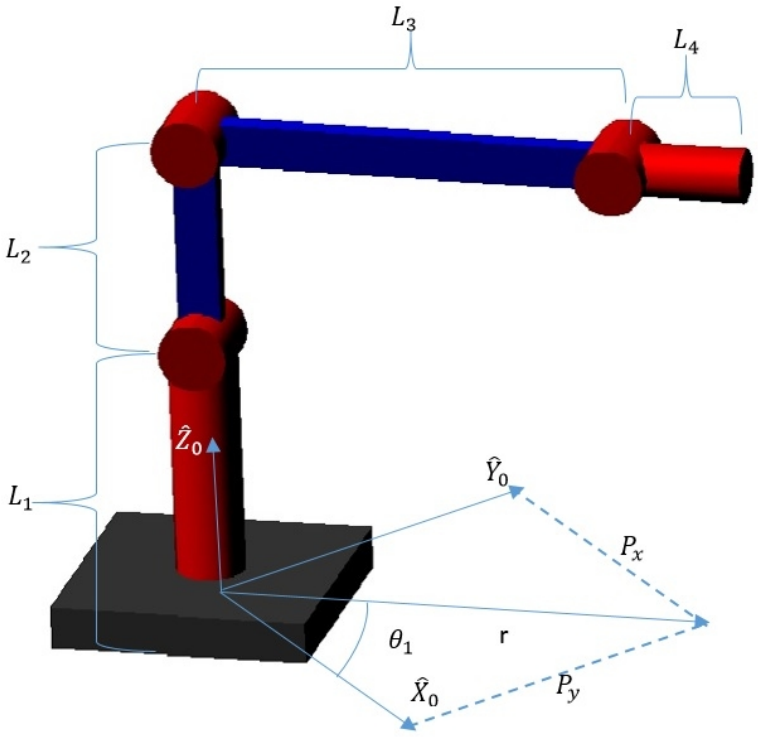


Ilustración 14, Dimensiones del brazo robot

El robot RV-2AJ tiene las siguientes longitudes:

Variable	Longitud (mm)
L_1 (Altura del hombro)	300
L_2 (Longitud del brazo)	250
L_3 (Longitud del antebrazo)	160
L_4 (L_4 Longitud de la muñeca + L_T Longitud de la herramienta)	195 (72+123)

El robot simulado tiene las siguientes longitudes:

Variable	Longitud (mm)
L_1 (Altura del hombro)	200
L_2 (Longitud del brazo)	250
L_3 (Longitud del antebrazo)	300
L_4 (L_4 Longitud de la muñeca + L_T Longitud de la herramienta)	150 (100+50)

1.2.1. SOLUCIÓN GEOMÉTRICA

1.2.2. CALCULO DE θ_1

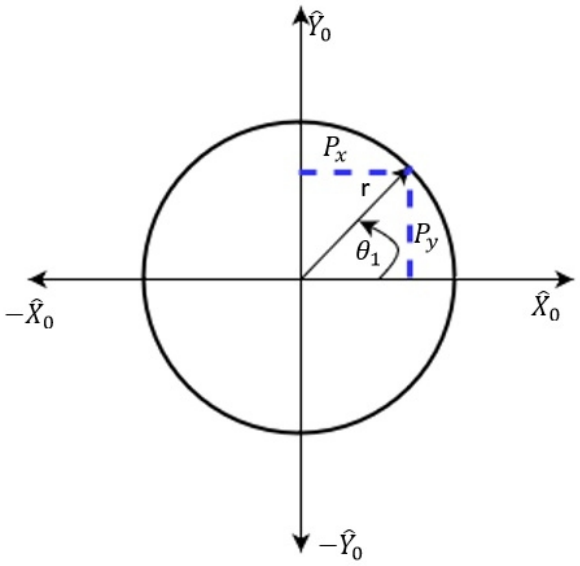


Ilustración 15, ángulo θ_1

Como la base tiene un giro que va desde los -150º a los 150º grados debemos obtener el arco tangente del ángulo θ_1 formado por los ejes P_x, P_y , hallamos el modulo "r" del mismo.

$$r = \sqrt{P_x{}^2 + P_y{}^2} \quad (46)$$

$$\sin\theta_1 = \frac{P_y}{r} \quad (47)$$

$$\cos\theta_1 = \frac{P_x}{r} \quad (48)$$

$$\theta_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right) \quad (49)$$

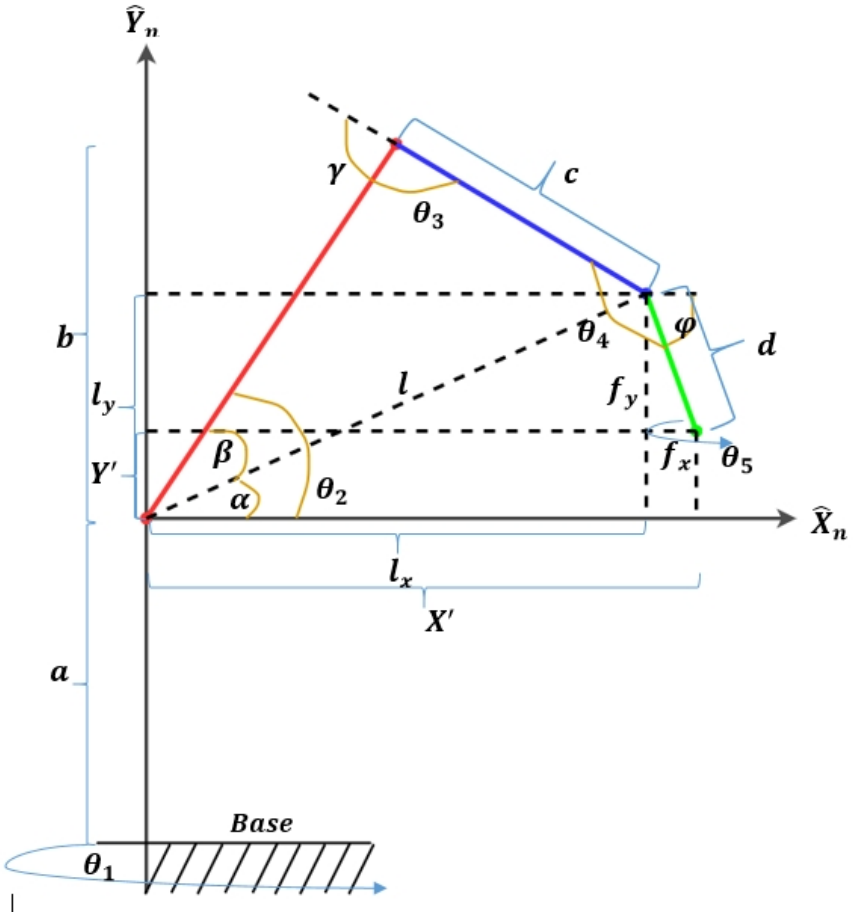


Ilustración 16, Análisis geométrico para obtener los ángulos

Variable	Descripción
a	Altura o longitud del hombro
b	Longitud del brazo
c	Longitud del antebrazo
d	Longitud de la muñeca
θ_4	Pitch o cabeceo
θ_5	Roll o rotación

1.2.3. CALCULO DE θ_2

Tenemos que determinar primero los valores de α y β, para determinar α por medio del arco tangente del triángulo rectángulo formado por l_x y l_y, y mientras que en el caso de β debemos aplicar el ley de cosenos y la hipotenusa del triángulo que da como resultado las siguientes ecuaciones a continuación:

$$f_x = d \times \cos(\theta_{4i}) \quad (50)$$

$$l_x = X' - f_x \quad (51)$$

$$f_y = d \times \sin(\theta_{4i}) \quad (52)$$

$$l_y = Y' - f_y - a \quad (53)$$

$$l = \sqrt{{l_x}^2 + {l_y}^2} \quad (54)$$

$$\alpha = \tan^{-1}\left(\frac{l_y}{l_x}\right) \quad (55)$$

$$\beta = \cos^{-1}\left(\frac{b^2 + l^2 - c^2}{2b}\right) n = s(r - p - 1) + \sum_{i}^{p} n_i \quad (56)$$

$$\theta_2 = \alpha + \beta = \tan^{-1}\left(\frac{P_z}{r}\right) + \cos^{-1}\left(\frac{b^2 + l^2 - c^2}{2b}\right) \quad (57)$$

El ángulo θ_{4i} (pitch o cabeceo) es el valor inicial que damos para el manipulador así como su posición.

Sin embargo, por coste computacional se recomienda dejar las ecuaciones trigonométricas en función de arco tangente, en el caso del programa que simula al robot que es similar al RV-2AJ, en la programación para poder determinar el β se usa el arco coseno.

1.2.4. CALCULO DE θ_3

Para calcular el ángulo θ_3 de la articulación que une el eslabón 1 y 2, se utiliza la ley de cosenos ya que el triángulo de análisis es un triángulo escaleno, esto puede variar entre cada configuración de brazos robots.

$$l^2 = b^2 + c^2 - 2b \times \cos(\gamma) \quad (58)$$

$$\gamma = \cos^{-1}\left(\frac{b^2 + c^2 - l^2}{2b}\right) \quad (59)$$

$$\theta_3 = -(180 - \gamma) \quad (60)$$

Normalmente se recomienda que se encuentre los últimos ángulos de robot por el método de desacoplo cinemático, pero como la configuración del robot tiene menos de 6 GDL es posible determinarlo por el método geométrico.

1.2.5. CALCULO DE θ_4

Es el más simple de determinar, en el programa de simulación el ángulo φ se da como valor inicial cero y como ya se calcularon los dos ángulos anteriores θ_2 y θ_3 podremos saber el valor de θ_4, como se describe a continuación:

$$\theta_4 = \varphi - \theta_2 - \theta_3 \quad (61)$$

1.2.6. CALCULO DE θ_5

El ángulo de rotación del manipulador no afecta o interviene en el resto de los cálculos por lo que es el rotación, es decir gira en su propio eje la herramienta en el rango de $-200° \leq \theta_5 \leq 200°$. Estas restricciones limites se toman del robot RV-2AJ, sin embargo, como es una simulación se puede omitir, porque las restricciones dependen de los servos motores que se vayan a usar porque ese análisis va relacionado con el análisis dinámico por lo que en el presente trabajo no se lo va a realizar.

1.3. MATRIZ JACOBIANA

La cinemática directa e inversa son las posiciones del brazo robot mientras que la matriz Jacobiana nos permite conocer las velocidades del efector final (Jacobiana Inversa) o extremo del robot a partir de las velocidades de las articulaciones (Jacobiana Directa).

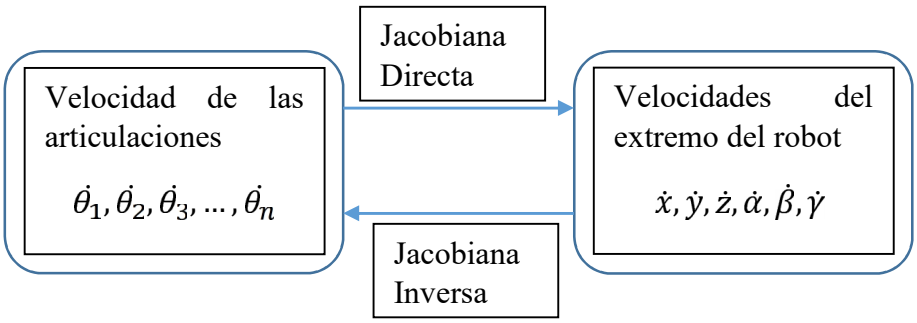


Ilustración 17, Diagrama del método del Jacobiano

Por medio de derivadas parciales con respecto al tiempo se obtiene la matriz Jacobiana que es de dimensión $6 \times n$ donde n son los grados de libertad del manipulador, para el caso de estudio $n = 5$, siendo esta matriz útil para:

I. Encontrar y analizar configuraciones singulares.
II. Encontrar y analizar la redundancia.
III. Relación entre las velocidades de las articulaciones y del extremo del robot.
IV. Relación entre las fuerzas del extremo del robot y de los pares de torsión en las articulaciones, es decir estudiar la estática del robot.

Entre otros casos, la matriz Jacobiana está formada por velocidades lineales y angulares del efector final.

El método del Jacobiano directo dependerá de los ángulos de las articulaciones, velocidades angulares, para poder determinar la velocidad del efector final, teniendo en cuenta que será de valor diferente en cada instante en el espacio de la trayectoria.

Como en nuestro caso el análisis del brazo robot es de articulaciones rotacionales (antropomórfico) tendremos las siguientes ecuaciones:

$$V = J_V \dot{\theta} \qquad (62)$$

$$V_i = V_{i-1} + \omega_i \times P_{i-1} \quad (63)$$

$$\omega = J_\omega \dot{\theta} \quad (64)$$

$$\omega_i = \omega_{i-1} + Z_{i-1} \quad (65)$$

Donde,

V_i Es la velocidad lineal de la articulación i.

ω_i Es la velocidad angular de la articulación i.

Z_i Es el vector unitario de proyección del eje de rotación de la articulación i.

P_i Es el vector posición de la matriz homogénea de la articulación i.

$$^{i-1}_{i}T(\theta) = \begin{bmatrix} R(\theta) & P(\theta) \\ 0_{1x3} & 1 \end{bmatrix} \quad (66)$$

$$\theta = [\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5]^T \quad (67)$$

$$J = \begin{bmatrix} J_V \\ J_\omega \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (68)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = J \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_5 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} (n = s(r-p-1) + \sum_{i}^{p} n_i) \quad (69)$$

La ecuación (69) n es valor de los numero de grados de libertad del mecanismo, es el criterio de Grubler-Kutzbach ver ecuación (1).

1.3.1. MÉTODO ANALÍTICO (POR DERIVADAS PARCIALES)

Se extrae de la matriz de transformación $^{i-1}_{i}T(\theta)$ la posición en $P(\theta)$ y la orientación en $R(\theta)$, donde $^{0}_{n}z$ es el vector k en la matriz de orientación por cada articulación hasta la articulación $i-1$ y el vector inicial es $^{0}_{0}z = [0\ 0\ 1]^T$

$$J\dot{\theta} = \begin{bmatrix} J_V\dot{\theta} \\ J_\omega\dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\partial^0_5\vec{r}}{\partial\theta_1} & \frac{\partial^0_5\vec{r}}{\partial\theta_2} & \frac{\partial^0_5\vec{r}}{\partial\theta_3} & \frac{\partial^0_5\vec{r}}{\partial\theta_4} & \frac{\partial^0_5\vec{r}}{\partial\theta_5} \\ ^0_0z & ^0_1z & ^0_2z & ^0_3z & ^0_4z \end{bmatrix} \quad (70)$$

1.3.2. MÉTODO GEOMÉTRICO:

$$\begin{bmatrix}\dot{p}\\ w\end{bmatrix} = \begin{bmatrix}J_P(\theta)\\ J_O(\theta)\end{bmatrix}\dot{\theta} \quad (71)$$

$${}^0_5\vec{r} = \begin{bmatrix}P_x\\ P_y\\ P_z\end{bmatrix} \quad (72)$$

$${}^0_4T(\theta_1,\theta_2,\theta_3,\theta_4) = \begin{bmatrix}n_x & o_x & a_x & P_x\\ n_y & o_y & a_y & P_y\\ n_z & o_z & a_z & P_z\\ 0 & 0 & 0 & 1\end{bmatrix} \quad (73)$$

$${}^0_4z = \begin{bmatrix}a_x\\ a_y\\ a_z\end{bmatrix} \quad (74)$$

$$J\dot{\theta} = \begin{bmatrix}{}^0_0z \times \left({}^0_5\vec{r} - {}^0_0\vec{r}\right) & {}^0_1z \times \left({}^0_5\vec{r} - {}^0_1\vec{r}\right) & {}^0_2z \times \left({}^0_5\vec{r} - {}^0_2\vec{r}\right) & {}^0_3z \times \left({}^0_5\vec{r} - {}^0_3\vec{r}\right) & {}^0_0z \times \left({}^0_5\vec{r} - {}^0_4\vec{r}\right)\\ {}^0_0z & {}^0_1z & {}^0_2z & {}^0_3z & {}^0_4z\end{bmatrix} \quad (75)$$

Siendo el vector ${}^0_5\vec{r}$ la posición P de la matriz Transformación homogénea de θ_1 a θ_5, y los otros vectores son las posiciones de cada matriz de articulación posmultiplicada con la siguiente, el vector de posición ${}^0_5\vec{r} - {}^0_0\vec{r}$ es un vector relativo que describe la posición relativa del sistema de coordenadas fija al sistema del efector final. Se debe tomar el vector de la tercera columna con las tres primeras filas de cada matriz de transformación hasta la matriz $i-1$ para que se pueda cumplir la diferencia de vectores de posición $r_i - r_{i-1}$. Se puede realizar de cualquiera de las dos formar algebraicamente o aplicar las derivadas parciales cómo se muestran en las ecuaciones (*70*) y (*76*).

$$J = \begin{bmatrix}-S_1(a_3C_2 + a_2C_2) + d_5S_2 & S_1 & -C_1(a_3S_2 + a_2S_2 + d_5C_2) & -C_1(a_3S_2 + d_5C_2) & -d_5C_2 & C_1 & 0\\ C_1(a_3C_2 + a_2C_2) - d_5S_2 & C_1 & -S_1(a_3S_2 + a_2S_2 + d_5C_2) & -S_1(a_3S_2 + d_5C_2) & -d_5C_2 & S_1 & 0\\ 0 & & d_5S_2 - a_2C_2 - a_3C_2 & d_5S_2 - a_3C_2 & d_5S_2 & & 0\\ 0 & & -S_1 & -S_1 & -S_1 & -S_2 & C_1\\ 0 & & C_1 & C_1 & C_1 & -S_2 & S_1\\ 1 & & 1 & 0 & 0 & -C_2 & \end{bmatrix}$$

(76)

Donde, por ejemplo:

S_1 es el $\sin\theta_1$.

C_1 es el $\cos\theta_1$.

S_2 es el $\sin(\theta_2+\theta_3+\theta_4)$

La matriz Jacobiana anterior fue obtenida por procedimientos computacionales iterativos en MATLAB en sus dos métodos tanto geométrico como por derivadas parciales, el código del programa se encuentra en los ANEXOS, 8.2 Script para la obtención de la matriz Jacobiana.

Para el caso de un robot que tenga articulaciones prismáticas la ecuación (*77*) cambiaria con la siguiente:

$$J = \begin{bmatrix} J_V \\ 0 \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ V_z \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \textbf{(77)}$$

No tendría velocidad angular ya que su movimiento es lineal.

Para el caso del método del Jacobiano inverso debemos calcular las velocidades angulares a partir de las velocidades del efector final, invirtiendo la matriz como se muestra a continuación:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_5 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = J^{-1} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \quad \textbf{(78)}$$

Sin embargo, por la complejidad que exige en el cálculo este procedimiento no es tan recomendable calcularlo.

1.4. SINGULARIDADES

Si la matriz Jacobiana determinada es invertible entonces no tiene singularidad, pero para el caso contrario que no tiene inversibilidad la matriz es singular en esa ubicación en el espacio.

También llamadas singularidades del mecanismo, son los límites que tiene el robot en su espacio de trabajo tanto en el exterior como en el interior, es decir cuando el manipulador pierde movilidad o puntos no alcanzables.

1. **Singularidad en el espacio de trabajo exterior:** Es cuando el brazo robot está completamente extendido o se doble sobre sí mismo, donde el efector final está cerca o en los límites del espacio de trabajo.
2. **Singularidad en el espacio de trabajo interior:** Es cuando el brazo robot tiene alineados dos o más ejes de las articulaciones fuera de los límites del espacio de trabajo.

Si sucede alguno de los dos casos anteriores el manipulador pierde uno o más grados de libertad haciendo imposible que moverse en cierta dirección en el espacio cartesiano.

Para el cálculo de la singularidad del brazo robot, debemos obtener la determinante de la matriz Jacobiana la misma que únicamente cuando det(J) = 0, esa configuración es singular. También se puede especificar los límites del mecanismo para un control del rango de movilidad como por ejemplo el $-60° \leq \theta_2 \leq 120°$ donde ya está limitado por ese rango.

Para tener un mejor control con la determinante Jacobiana, cuando los puntos alcanzar por el manipulador son fuera de los limites la determinante Jacobianan nos devuelve números complejos que pueden ser controlados con condiciones para evitar daños al brazo robot al esforzar al motor para alcanzar ese punto.

Para tener mayor control con respecto a los tipos de singularidades que la determinante devuelve un valor de cero es necesario que incluya sentencias o condicionales de control para determinar limites admisibles y no admisibles del brazo robot por ejemplo si se requiere que el manipulador alcance su límite de espacio de trabajo extendiendo completamente el brazo no quiere decir que por esa razón se deba acabar el proceso ya que el valor devuelto por la determinante es de cero, en el caso que se doble completamente sobre sí mismo entra en una singularidad pero no es admisible.

Para el manipulador de 5 GDL como la matriz jacobiana no es cuadrada puede ser calculado siguiendo los puntos tomados del libro de “Fundamentos de Robótica – Antonio Barrientos, pág. 129”, que son los siguientes:

I. Identificar la articulación correspondiente al grado de libertad perdido.
II. Eliminarla fila de la Jacobiana correspondiente al grado de libertad perdido y/o la columna correspondiente a la articulación causante (si esta fuese necesaria).

III. Con la nueva Jacobiana reducida (rango n-1) obtener las velocidades de todas las articulaciones, a excepción de la eliminada, necesarias para conseguir las velocidades cartesianas deseadas. La velocidad de la articulación eliminada se mantendrá en cero.

La singularidad del manipulador de 5 GDL está definida por la siguiente expresión:

$$\begin{aligned} J_d \quad = -(a_2 \times a_3(a_3\cos(\theta_3+\theta_4) - a_3\cos(2\theta_2+\theta_3+\theta_4) \\ - d_5\sin(2\theta_2+3\theta_3+2\theta_4) - a_2\cos\theta_4 \\ + a_2\cos(2\theta_2+2\theta_3+\theta_4) + a_3\cos(2\theta_2+3\theta_3+\theta_4) \\ + 2d_5\sin\theta_3 + d_5\sin(2\theta_2+\theta_3+2\theta_4) - a_2\cos(2\theta_2+\theta_4) \\ + a_2\cos(2\theta_3+\theta_4) - a_3\cos(\theta_3-\theta_4)))/4 \end{aligned} \quad (79)$$

Obtenida de la matriz de Jacobiana eliminando la última fila por no ser un manipulador de 6 GDL.

1.5. ESTÁTICA

Es la que analiza las fuerzas y momentos que actúan sobre el sistema que se encuentra en equilibrio o estado estático. En el caso del manipulador serial en estudio se busca la relación entre pares de torsión/fuerza, los momentos y fuerzas aplicadas al efector final sea por métodos de vectores, principio de trabajo virtuales, etc.

Para aplicar la estática el brazo robot debe realizar una determinada tarea como mover un objeto de un sitio a otro. En ese movimiento para esa tarea se ejerce una fuerza y momento, que son las reacciones que causan ese movimiento. Siendo muy importante este tema para determinar el tamaño de los eslabones y actuadores necesarios para un correcto funcionamiento.

CAPÍTULO 2. CONTROL CINEMÁTICO

2. PLANEACIÓN DE TRAYECTORIA

2.1. GENERACIÓN DE TRAYECTORIA

Para describir la trayectoria del movimiento de un brazo robot se tiene que obtener la posición, velocidad y aceleración de cada articulación o grado de libertad para que dé como resultado la trayectoria del efector final.

La planeación de la trayectoria o generación de trayectoria debe generar una secuencia de puntos vía deseados que proporcionen información de la ruta, es decir la posición y orientación descrita por cada punto vía o trama.

Esta secuencia de puntos vía es llamada puntos ruta que contiene la información y restricciones del movimiento realizado por las articulaciones en el espacio de trabajo.

Para la generación de trayectoria tenemos dos métodos, que son los siguientes:

I. Espacio de articulaciones o articular.
II. Espacio cartesiano u operacional.

Por el tipo de perfil de la trayectoria pueden ser catalogados como los siguientes:

1. Trayectoria trapezoide
2. Perfil en “S”
3. Perfil polinomial

2.2. ESPACIO ARTICULAR

Para este método supondremos que se conocen las variables iniciales, que son la posición inicial y final, proporcionado por la cinemática inversa. Por medio de la utilización de polinomios en función del tiempo, es decir su trayectoria es por generadores polinomiales.

Para que el movimiento se pueda dar se requiere de restricciones sobre $\theta(t)$. Para una articulación con ángulo inicial θ_i en un tiempo $t_i = 0$ y un ángulo final de θ_f en un tiempo $t = t_f$, tenemos:

$$\theta(t_i) = \theta(0) = \theta_i \quad (80)$$

$$\theta(t_f) = \theta_f \quad (81)$$

Siendo estas mis dos restricciones en la posición y mis otras dos restricciones son la velocidad y aceleración que están condicionadas por las siguientes ecuaciones:

$$\dot{\theta}(t_i) = \dot{\theta}_i = 0 \quad \dot{\theta}(t_f) = \dot{\theta}_f = 0 \quad (82)$$

$$\ddot{\theta}(t_i) = \ddot{\theta}_i = 0 \quad \ddot{\theta}(t_f) = \ddot{\theta}_f = 0 \quad (83)$$

Con las restricciones impuestas debemos utilizar un polinomio que cumpla para la solución de las seis incógnitas, por lo que se debe usar un polinomio de orden superior, en este caso de quito orden, como la ecuación a continuación:

$$\theta(t) = a + b\ + ct^2 + dt^3 + et^4 + ft^5 \quad (84)$$

$$\dot{\theta}(t) = b + 2c\ + 3dt^2 + 4et^3 + 5ft^4 \quad (85)$$

$$\ddot{\theta}(t) = 2c + 6d\ + 12et^2 + 20ft^3 \quad (86)$$

Mediante las restricciones ya señaladas tanto de la velocidad como de la aceleración para que las articulaciones tengan un movimiento acelerado y desacelerado que cumpla con un paro completo del efector final al punto deseado. Para el desarrollo de las seis incógnitas se debe resolver el sistema de forma matricial:

$$\begin{bmatrix} {t_i}^5 & {t_i}^4 & {t_i}^3 & {t_i}^2 & t_i & 1 \\ 5{t_i}^4 & 4{t_i}^3 & 3{t_i}^2 & 2t_i & 1 & 0 \\ 20{t_i}^3 & 12{t_i}^2 & 6t_i & 2 & 0 & 0 \\ {t_f}^5 & {t_f}^4 & {t_f}^3 & {t_f}^2 & t_f & 1 \\ 5{t_f}^4 & 4{t_f}^3 & 3{t_f}^2 & 2t_f & 1 & 0 \\ 20{t_f}^3 & 12{t_f}^2 & 6t_i & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f \\ e \\ d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \theta_i \\ 0 \\ 0 \\ \theta_f \\ 0 \\ 0 \end{bmatrix} \quad (87)$$

Para la resolución de la matriz se debe seguir el siguiente procedimiento:

$$A \cdot B = C \quad (88)$$

$$A^{-1} \cdot A \cdot B = A^{-1} \cdot C \quad (89)$$

$$B = A^{-1} \cdot C \quad (90)$$

Recordando que la no es conmutativa, $A^{-1} \cdot C \neq C \cdot A^{-1}$

Desarrollando la forma matricial del sistema de ecuaciones con seis incógnitas tendremos las siguientes ecuaciones:

$$a = \frac{\theta_i {t_f}^3 \left({t_f}^2 - 5t_i t_f + 10{t_i}^2\right) - \theta_f {t_i}^3 \left(10{t_f}^2 - 5t_i t_f + {t_i}^2\right)}{\left(t_f - t_i\right)^5} \quad (91)$$

$$b = \frac{30{t_i}^2 {t_f}^2}{\left(t_f - t_i\right)^5}\left(\theta_f - \theta_i\right) \quad (92)$$

$$c = \frac{30t_i t_f \left(t_i + t_f\right)}{\left(t_f - t_i\right)^5}\left(\theta_i - \theta_f\right) \quad (93)$$

$$d = \frac{\left(10{t_i}^2 + 40t_i t_f + 10{t_f}^2\right)}{\left(t_f - t_i\right)^5}\left(\theta_f - \theta_i\right) \quad (94)$$

$$e = \frac{\left(15t_i + 15t_f\right)}{\left(t_f - t_i\right)^5}\left(\theta_i - \theta_f\right) \quad (95)$$

$$f = \frac{6}{\left(t_f - t_i\right)^5}\left(\theta_f - \theta_i\right) \quad (96)$$

Trayectoria del brazo robot para tres puntos finales distintos generando una secuencia continua polinomial de quinto orden.

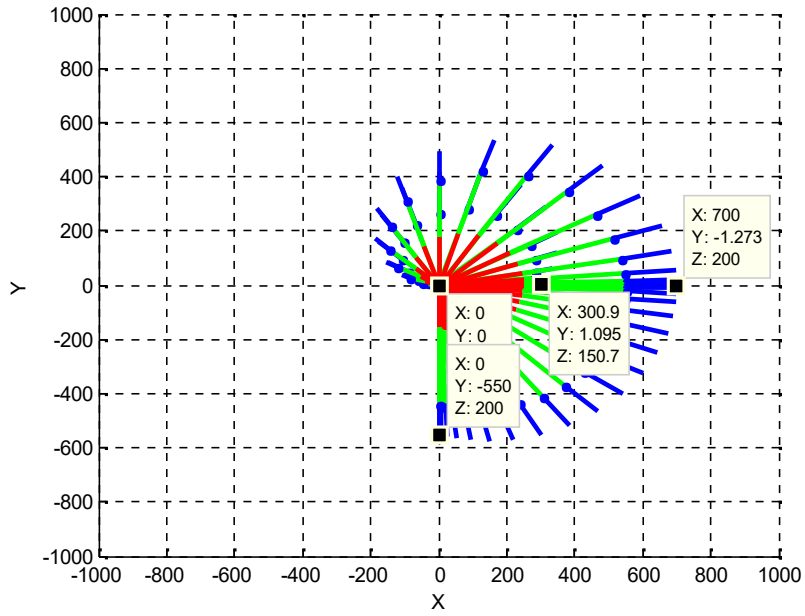


Ilustración 18, Trayectoria vista en el plano XY en MatLab

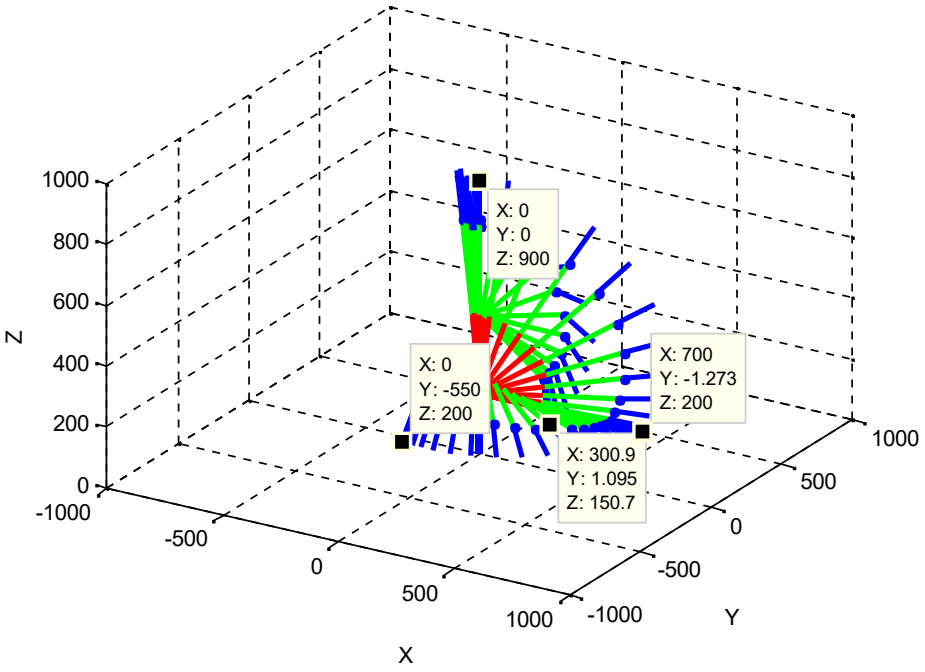


Ilustración 19, Trayectoria vista en 3D en MatLab

2.3. TIPOS DE TRAYECTORIAS EN EL ESPACIO ARTICULAR

2.3.1. TRAYECTORIA CON PUNTO INICIAL AL PUNTO FINAL

Cuando se cuenta con los puntos iniciales (comienza en el reposo) y finales (paro total), es decir existe una trayectoria que tiene una etapa de aceleración, de velocidad contante y desaceleración hasta el paro como en la Ilustración 20, Trayectoria punto inicial al punto final.

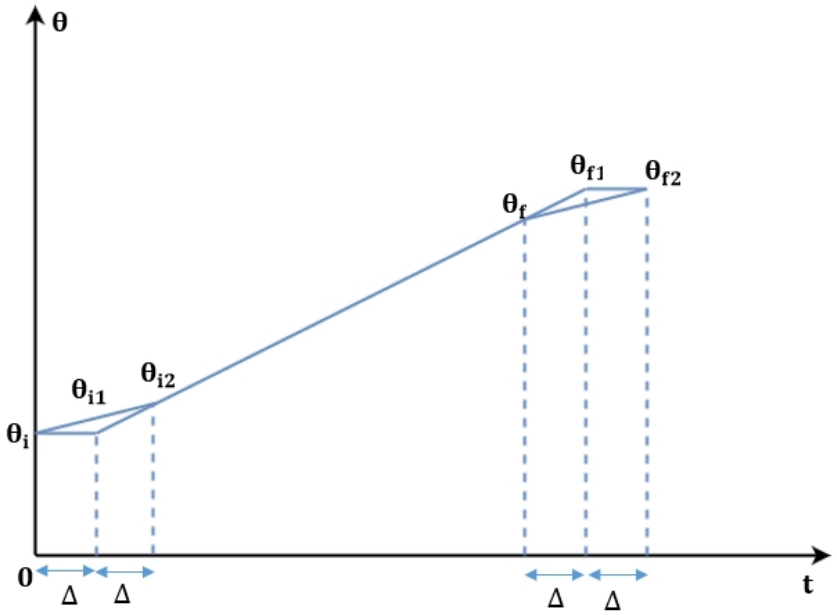


Ilustración 20, Trayectoria punto inicial al punto final

2.3.2. TRAYECTORIA CON PUNTO INICIAL, PUNTOS INTERMEDIOS Y PUNTO FINAL

En este tipo de trayectoria al igual que la trayectoria anterior tiene puntos iniciales (comienza en el reposo), finales (paro total) y puntos intermedios, donde dan lugar a segmentos en la trayectoria. Con varios puntos para la ruta intermedios el mejor método es mediante polinomios usando líneas rectas para conectar los puntos como se representa en la Ilustración 21, Trayectoria con puntos intermedios que es del análisis de la posición de la primera articulación del manipulador en estudio.

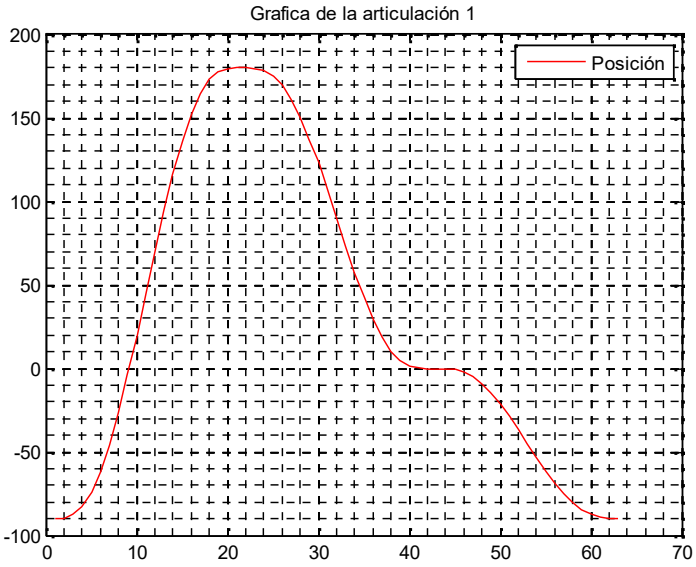


Ilustración 21, Trayectoria con puntos intermedios

2.4. ESPACIO CARTESIANO

El espacio articular describen los puntos de inicio, de ruta y de destino. Sin embargo, la ruta no siempre es una línea recta en el espacio, sino una ruta más compleja dependiendo del análisis cinemático que se haya realizado para el brazo robot.

La ruta no es más que un conjunto de puntos que tienen una posición y orientación en el espacio de trabajo del manipulador. En el plano cartesiano usaremos la cinemática inversa para calcular estos puntos donde debe especificar la ruta del efector final del brazo robot.

En la generación de la ruta en el espacio cartesiano, requiere mayor poder de cómputo y cálculo de la cinemática inversa en el cambio constante del efector final en su posición (cada periodo de muestreo) para determinar y llegar al punto especificado por el usuario.

El movimiento que se puede lograr es una línea recta para que el efector final se mueva en el espacio cartesiano, al especificar muchos puntos sobre la ruta, es el movimiento cartesiano. Al generar las rutas en línea recta es recomendable usar funciones lineales para la interpolación lineal para especificar la posición, para el caso de la orientación del efector final tomaremos los ángulos de Euler o los ángulos de cabeceo y rotación para el brazo robot en estudio sin embargo si fuera de 6 ejes sería más el de balanceo.

Dando como resultado la forma espacial de la ruta del efector final un trazado que no es una línea recta a través del espacio de trabajo.

El manejo de las coordenadas en el espacio cartesiano es más fácil que el manejo de los valores angulares generados para el espacio articular, sin embargo, para el manejo interno de los ángulos como posiciones para el control cinemático y generar el polinomio es mejor manejar con los ángulos de cada articulación. Por lo que es conveniente manejar los datos de ingreso en cartesiano y su procesamiento en angular.

2.5. TIPOS DE PERFIL DE LA TRAYECTORIA

2.5.1. TRAYECTORIA PERFIL TRAPEZOIDAL

El perfil trapezoidal es comúnmente usado en la industria, para los variadores o drivers para controlar los motores. Para medir la posición, velocidad y aceleración en un tiempo determinado para la interpolación lineal, donde se trata de mantener constante la velocidad, la aceleración en el arranque y en la desaceleración.

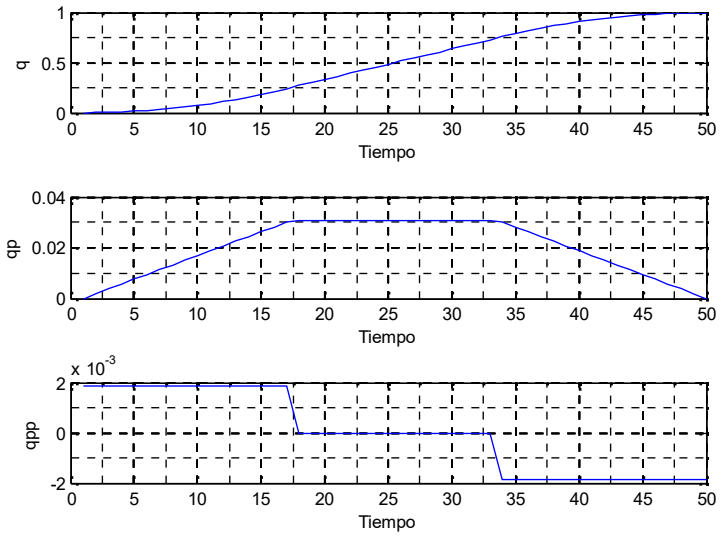


Ilustración 22, Perfil Trapezoidal

2.5.2. TRAYECTORIA PERFIL EN 'S'

Parecido al perfil trapezoidal con la diferencia que la velocidad en el tiempo para alcanzar su valor esperado tiene una forma de "S" y que la etapa de aceleración es instantánea al igual que la desaceleración.

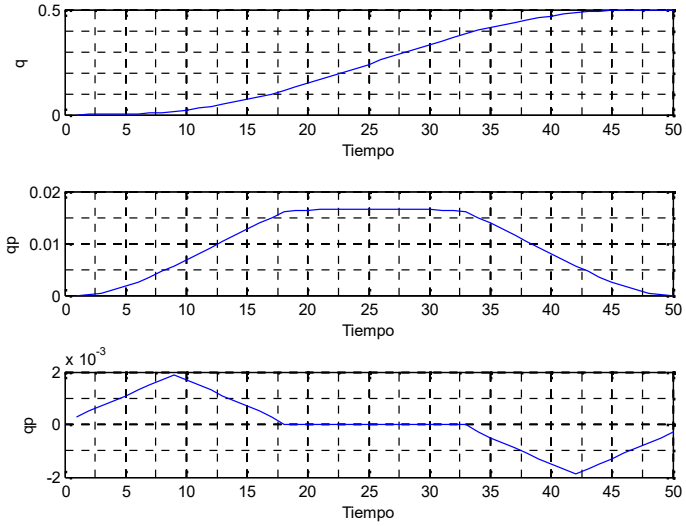


Ilustración 23, Perfil en S

2.6. TRAYECTORIA PERFIL POLINOMIAL

El perfil polinomial que es el más usado para los brazos robots por el polinomio que es de quinto orden es el que genera los puntos para que la ruta sea un movimiento continuo, que debe seguir la articulación en la Ilustración 24, Perfil polinomial, se aprecia que en un tiempo de 60 segundos realizo dos movimientos y 20 segundos paso en reposo.

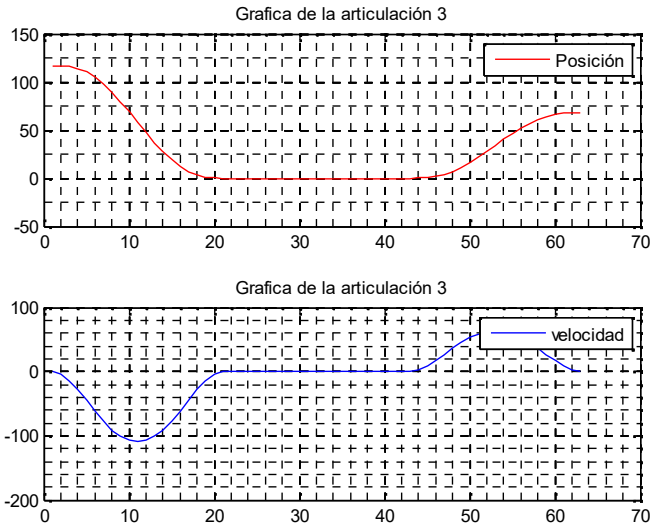


Ilustración 24, Perfil polinomial

2.7. COMPARACIÓN ENTRE PERFILES

La trayectoria con perfil polinomial es el más comúnmente usado por la interpolación lineal y curvilínea que existe para hacer una trayectoria continua. Para poder tener un

mayor control del brazo robot a diferencia de los otros perfiles que tienen aceleraciones instantáneas como la de perfil en “S”, el manipulador tendría movimientos abruptos siendo un perfil peligroso. En el caso del trapezoidal se supone que tiene una aceleración igual a la desaceleración pudiendo servir para los brazos robot, pero no se podría generar una trayectoria tan continua como lo hace la polinomial por lo que no se lo suele usar comúnmente.

CAPÍTULO 3. MATLAB

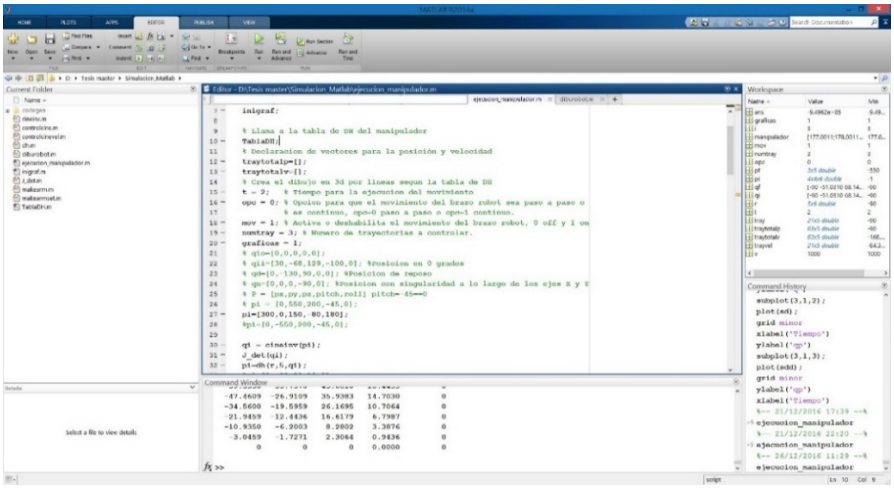
3. PROGRAMACIÓN EN MATLAB

Uno de los ambientes de desarrollo integrado usados en este trabajo es Matlab que es un software matemático con su propio lenguaje llamado M, teniendo similitudes al lenguaje C. Matlab es un lenguaje que trabaja con vectores y matrices, funciones, y programación orientada a objetos.[2] Este lenguaje es comúnmente usado para la resolución de problemas matemáticos, para el modelado, simulación y control de eventos físicos.

Matlab trabaja con la compilación de varios bloques o scripts para ejecutar las diferentes funciones, así como para que el desarrollador pueda programar las funciones necesarias para un determinado proyecto, ver Ilustración 25, Interfaz Software Matlab.

En este apartado se explicará los diferentes módulos que se desarrollaron en Matlab para generar una simulación de cómo debería ser el control cinemático controlado por el DSC al brazo robot simulado.

El flujograma de la programación en Matlab está dado por la Ilustración 26, Flujograma del programa principal en Matlab, en donde se debe especificar el punto de inicio en este caso la posición inicial del brazo robot seguido del o de los puntos finales que quiere llegar el brazo.



[2] https://es.wikipedia.org/wiki/MATLAB

Ilustración 25, Interfaz Software Matlab

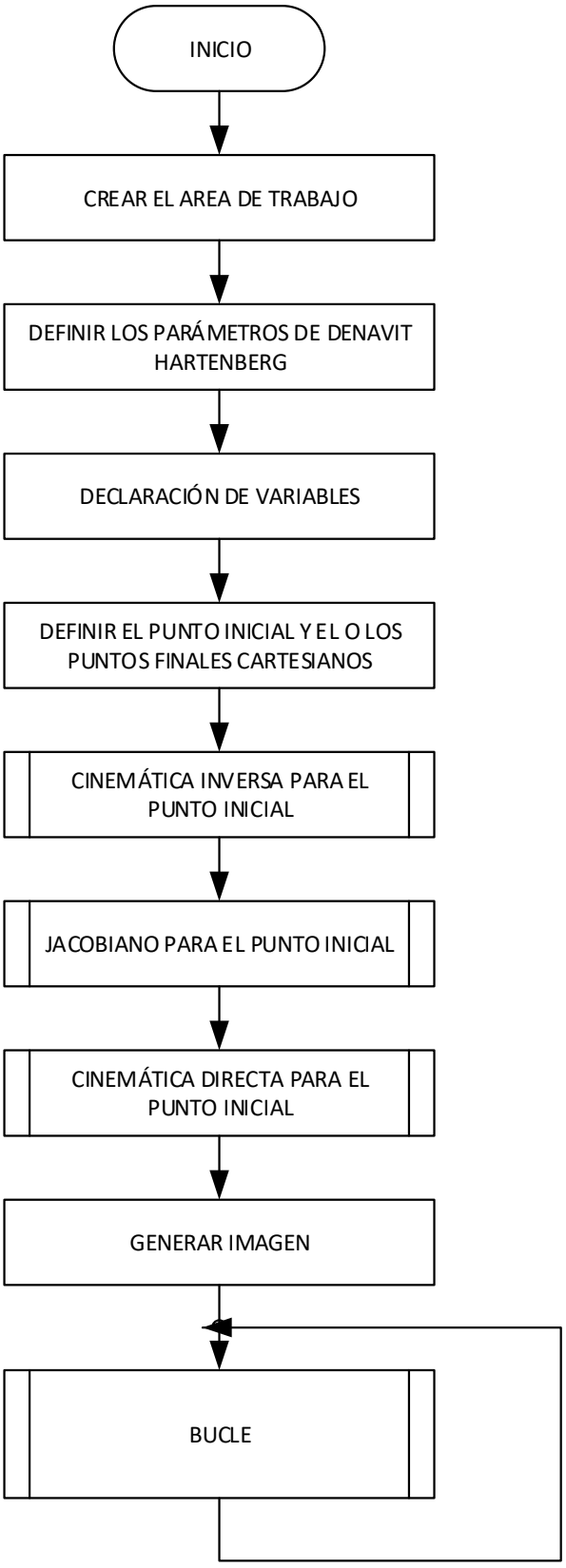


Ilustración 26, Flujograma del programa principal en Matlab

El bucle sin fin contiene las mismas sub-rutinas que en la Ilustración 26, Flujograma del programa principal en Matlab, pero para realizar las operaciones con respecto a los puntos finales para que el resultado final se almacene en la posición inicial para repetir el ciclo las veces que sean definidas por el usuario que sean igual a los puntos definidos, como se aprecia en el flujograma de la Ilustración 27, Flujograma del bucle sin fin.

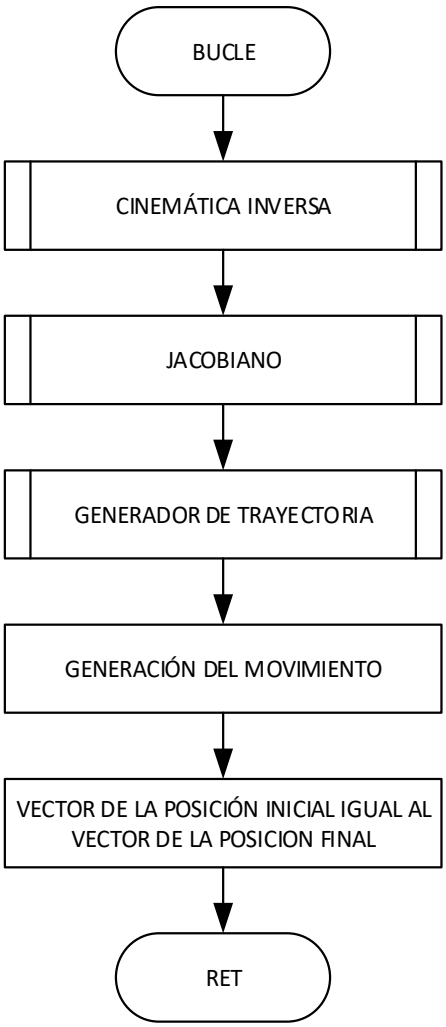


Ilustración 27, Flujograma del bucle sin fin

A continuación, el programa principal que llama a las diferentes funciones para el control cinemático de un brazo robot de 5 ejes desarrollado en Matlab:

```
% Crea el área de trabajo en 3 dimensiones x,y,z
entorno;

% Llama a la tabla de DH del manipulador
TablaDH;

% Declaración de vectores para la posición y velocidad
traytotalp=[];
traytotalv=[];

% Crea el dibujo en 3d por líneas según la tabla de DH
t = 2;   % Tiempo para la ejecución para la resolución del movimiento

opc = 1; % Opcion para que el movimiento del brazo robot sea paso a paso o
         % es continuo, opc=0 paso a paso o opc=1 continuo.
mov = 1; % Activa o deshabilita el movimiento del brazo robot, 0 off y 1 on
numtray = 1; % Numero de trayectorias a controlar.
% Esta opción cuando es 1 habilita la generación de graficas con el
% movimiento total de cada articulación, cuando es 0 la deshabilita
graficas = 1;
pi=[300,0,150,-80,180];
qi = cineinv(pi);
J=J_det(qi);
pi=dh(r,5,qi);

manipulador = makearm(r,qi');
pf = [[0,0,900,90,0];[700,0,200,0,0];[0,-550,200,-45,0]];
i=0;
if(mov==1)
    for i=1:1:numtray

        qf = cineinv(pf(i,1:5));
        J=J_det(qf);
        if (imag(J)~=0)
            fprintf('Esta en una posición singular fuera del espacio de
trabajo \n');
            graficas=0;
            break;
        end
       if (pf(i,4)>120)||(pf(i,4)<-120)
            fprintf('Esta en una posición singular \n');
            graficas=0;
            break;
        end
        tray = controlcine(qi,qf,t);
        trayvel = controlcinevel(qi,qf,t);
        diburobot(r,manipulador,tray,opc);
        qi=qf;
        traytotalp=vertcat(traytotalp,tray);
        traytotalv=vertcat(traytotalv,trayvel);
    end
    if(graficas==1)
            figure(2);
            subplot(2,1,1);
            plot(traytotalp(:,1),'r');
            grid minor;
            title('Grafica de la articulación 1');
            legend('Posición');
            xlabel('Tiempo'); ylabel('\theta');
            subplot(2,1,2);
            plot(traytotalv(:,1),'b');
            grid minor;
            title('Grafica de la articulación 1');
            legend('velocidad');
            xlabel('Tiempo'); ylabel('\theta\prime');

            figure(3);
            subplot(2,1,1);
```

```
            plot(traytotalp(:,2),'r');
            grid minor;
            title('Grafica de la articulación 2');
            legend('Posición');
            xlabel('Tiempo'); ylabel('\theta');
            subplot(2,1,2);
            plot(traytotalv(:,2),'b');
            grid minor;
            title('Grafica de la articulación 2');
            legend('velocidad');
            xlabel('Tiempo'); ylabel('\theta\prime');

            figure(4);
            subplot(2,1,1);
            plot(traytotalp(:,3),'r');
            grid minor;
            title('Grafica de la articulación 3');
            legend('Posición');
            xlabel('Tiempo'); ylabel('\theta');
            subplot(2,1,2);
            plot(traytotalv(:,3),'b');
            grid minor;
            title('Grafica de la articulación 3');
            legend('velocidad');
            xlabel('Tiempo'); ylabel('\theta\prime');
    end
end
```

El bloque de la cinemática inversa se lo puede deducir en el siguiente flujograma Ilustración 28, Flujograma de la cinemática inversa, basado en las ecuaciones que van de la ***(46)*** a la ***(61)***. Para poder generar los ángulos a partir de los puntos cartesianos programados por el usuario.

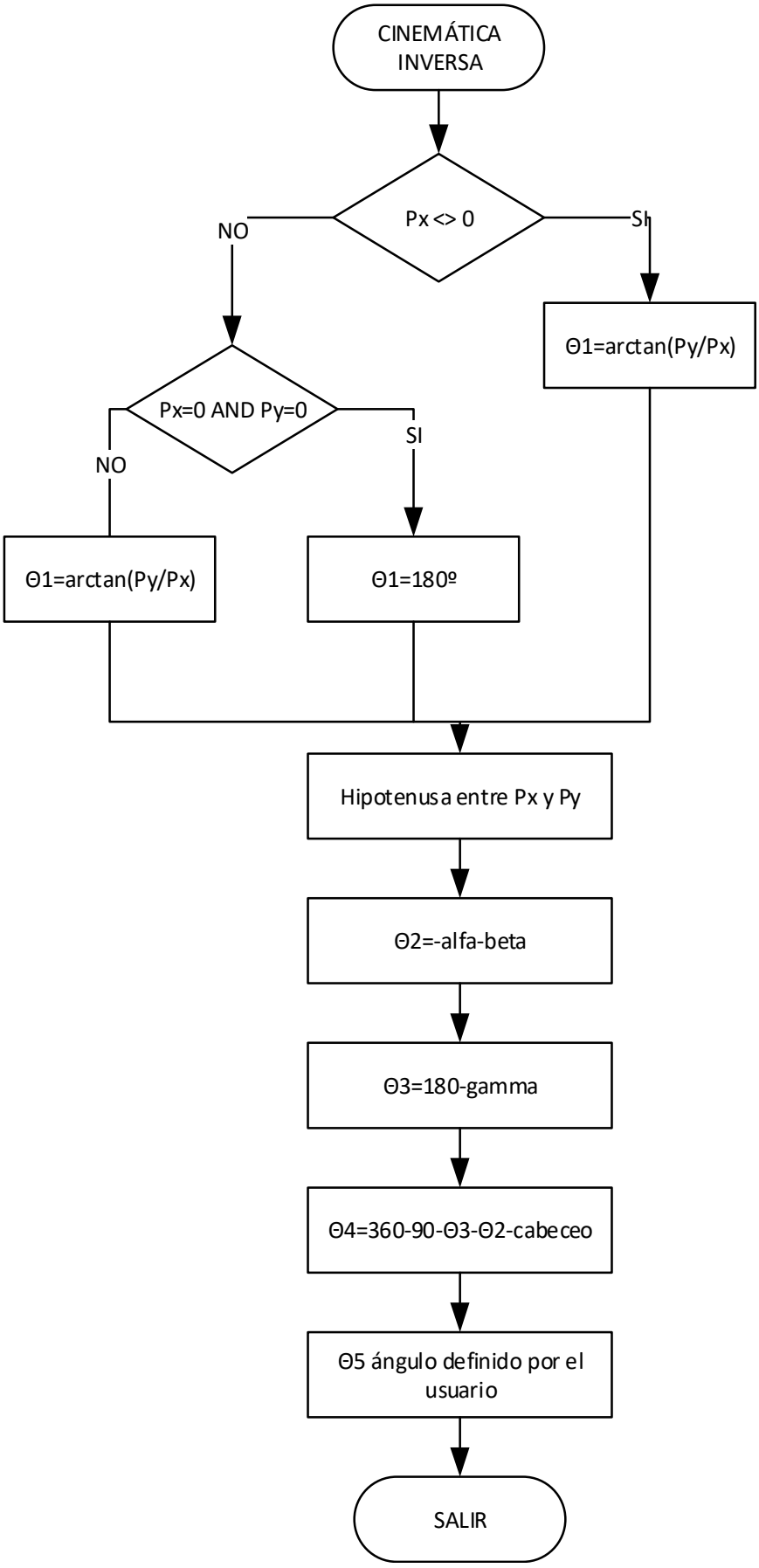


Ilustración 28, Flujograma de la cinemática inversa

El script de la cinemática inversa está basado en los ángulos límites del brazo robot RV-2AJ, para las restricciones físicas del servo motor. Sin embargo, para tratar a la posterior simulación te lo realiza con el bloque de la determinante del Jacobiano. El siguiente código se desarrolla con procedimientos geométricos descritos en el capítulo 1 en la solución geométrica de la cinemática inversa.

```
%% Cinematica inversa
function invp=cineinv(p)
a = 200; %Longitud del hombro
b = 250; %Longitud del brazo
c = 300; %Longitud del antebrazo
d = 150; %Longitud de la muñeca + herramienta
%espacio = b+c+d;

% % Valores iniciales
%Toma de los valores de la posicion y orientacion del efector final
px=p(1);
py=p(2);
pz=p(3);
pitch=p(4);
roll=p(5);

%Calculo theta 1
if (px~=0)
theta1 = atan2d(py,px);
  if (theta1<=-150)&&(theta1>=150)
      display('Error singularidad en theta1 valor no alcanzable');
      display('Recalculando valor');
      if theta1<-150;
        theta1=-150;
      else
        theta1=150;
      end
      %return;
  end
else
theta1 = atand(py/px);
end
if (px==0)&&(py==0)
        %theta1 = 180;
        theta1 = 180;
end
%Hipotenusa que crea el brazo robot con respecto a las articulaciones Q2,Q3
%y Q4
r = sqrt(abs(px^2)+abs(py^2));

%Calculo theta 2
fx = d*cosd(pitch);
lx = r-fx;
fy = d*sind(pitch);
ly = pz-fy-a;
l = sqrt(lx^2+ly^2); %Hipotenusa del triangulo formado por b y c
alfa = atand(ly/lx);
beta = acosd((b^2+l^2-c^2)/(2*b*l));
    theta2 =(-(alfa+beta));
    if (px==0)&&(py==0)
        theta2 =(-90);
    end
%Calculo theta 3
gamma = (acosd((b^2+c^2-l^2)/(2*b*c)));
    theta3 = 180-gamma;
    if (px==0)&&(py==0)
        theta3 = 0;
```

```
    end
%Calculo theta 4
    theta4 = 360-90+(-pitch-theta2-theta3);
theta5 = roll;
invp=[theta1,theta2,theta3,theta4,theta5];

end
```

El bloque de la función para determinar la singularidad del manipulador por medio de la determinante del jacobiano. Si el resultado es igual cero el brazo está en singularidad. El desarrollo del jacobiano se puede resolver por uno de dos métodos expuestos en el capítulo 1 matriz jacobiana al igual que la obtención de su determinante.

```
% Funcion para determinar singularidad
function J=J_det(q)
    th1=q(1);th2=q(2);th3=q(3);th4=q(4);th5=q(5);
a1=0; a4=0; a5=0; al1=-pi/2; al2=0; al3=0; al4=-pi/2; al5=0; d2=0; d3=0;
d4=0; th6=th5;
a2=250; a3=300; d1=200; d5=150;

J=-(a2*a3*(a3*cos(th3 + th4) - a3*cos(2*th2 + th3 + th4) - d5*sin(2*th2 +...
      +3*th3 + 2*th4) - a2*cos(th4) + a2*cos(2*th2 + 2*th3 + th4) +...
  a3*cos(2*th2 + 3*th3 + th4) + 2*d5*sin(th3) + d5*sin(2*th2 + th3 + 2*th4)...
  - a2*cos(2*th2 + th4) + a2*cos(2*th3 + th4) - a3*cos(th3 - th4)))/4;

if(J==0)
    fprintf('Esta en una posición singular \n');

end

end
```

El siguiente bloque resuelve la cinemática directa por medio de la matriz de Denavit Hartenberg del brazo robot. Usando la matriz de transformación homogénea ***(10)***. La función genera una matriz por cada una de las articulaciones por lo que se debe usar matrices de tres dimensiones para almacenar los valores para su posterior uso.

```
function [matriz]=den_hat(robot,eje,q)

N=eje;
Ai(:,:,1) = eye(4);
for i=2:N+1
    a=robot(i-1,1);
    alfa=robot(i-1,2);
    %phi = 0 es revolucion phi = 1 es prismatico
    if robot(i-1,6) == 0
        theta=q(i-1);
        d=robot(i-1,3);
    else
        d=q(i-1);
        theta=robot(i-1,4);
    end
    A=[cosd(theta)        -cosd(alfa)*sind(theta)        sind(alfa)*sind(theta)
a*cosd(theta);...
         sind(theta)       cosd(alfa)*cosd(theta)       -sind(alfa)*cosd(theta)
a*sind(theta);...
          0 sind(alfa) cosd(alfa) d;...
          0 0 0 1];
    Ai(:,:,i)=Ai(:,:,i-1)*A;
end
```

```
for i=1:N
    Ai(:,:,i)=Ai(:,:,i+1);
end
    matriz=Ai;
end
```

Por último, el script que genera la trayectoria para el movimiento del brazo robot por medio de un polinomio de quinto orden, donde tenemos las restricciones de contorno para reducir el polinomio como se puede apreciar en las ecuaciones **(80)** al **(83)**.

```
%% Generacion de trayectoria o control cinematico
% Función que genera la trayectoria punto a punto
% Usando un polinomio de cuarto orden

function [q] = controlcine(qi,qf,tf)

poli = [];
polip = [];
q = [];
qp = [];
% Condiciones de contorno thetap0=thetapp0=thetappf=0
% thetaf-theta0=(tf/2)(thetapf+thetap0)

for i=0:0.1:tf
    for j=1:1:length(qi)
        a=(qi(j)*tf^3*(tf^2)/(tf^5));
        b=0;
        c=0;
        d=(qf(j)*(10*tf^2)/tf^5)-(qi(j)*(10*tf^2)/tf^5);
        e=(qi(j)*(15*tf)/tf^5)-(qf(j)*(15*tf)/tf^5);
        f=6*qf(j)/tf^5-6*qi(j)/tf^5;
        poli(j) = (f*i^5 + e*i^4 + d*i^3 + c*i^2 + b*i + a);
        polip(j) = (5*f*i^4 + 4*e*i^3 + 3*d*i^2 + 2*c +b);
    end
[q] = [q;poli]; %Matriz de la trayectoria
[qp] = [qp;polip]
end
end
```

Los cuatro bloques antes mencionados son los más importantes para poder resolver la conversión de los valores cartesianos a angulares o articulares y procesarlos para que los resultados poderlos convertir a cartesianos y después para generar la trayectoria para el movimiento del manipulador.

La posición inicial que está dada por la simulación a controlar es 300 unidades en el eje X, 0 unidades en el eje Y, 150 unidades en el eje Z, -80º en la articulación 4 y 180º en la articulación 5 como se puede ver en la Ilustración 29, Posición inicial brazo robot, para el primer punto programado de la primera trayectoria es 0 unidades en el eje X, 0 unidades en el eje Y, 900 unidades en el eje Z, 90º en la articulación 4 y 0º en la articulación 5 como en la Ilustración 30, Primera posición programada. Para el segundo punto programado se tiene 700 unidades en el eje X, 0 unidades en el eje Y, 200 unidades en el eje Z, 0º en la articulación 4 y 5 ver la Ilustración 31, Segunda posición

programada. Por ultimo para la tercera posición programada la trayectoria está dada por el punto final con 0 unidades en el eje X, -550 unidades en el eje Y, 200 unidades en el eje Z, -45º en la articulación 4 y 0º en la articulación 5 como se aprecia en la Ilustración 32, Posición final programada.

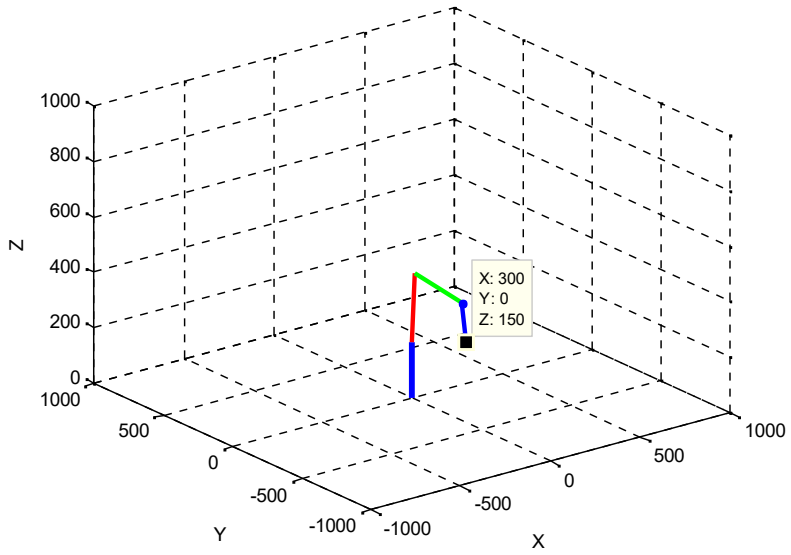


Ilustración 29, Posición inicial brazo robot

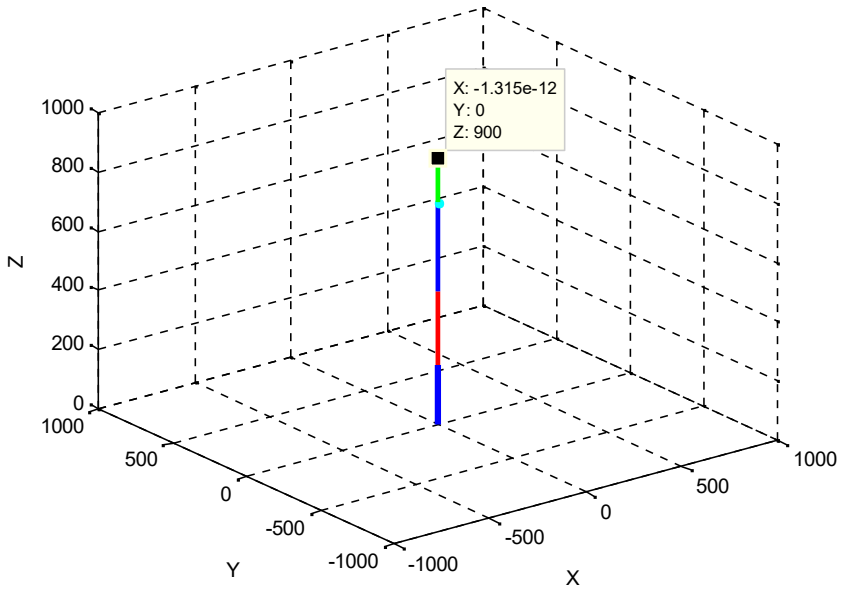


Ilustración 30, Primera posición programada

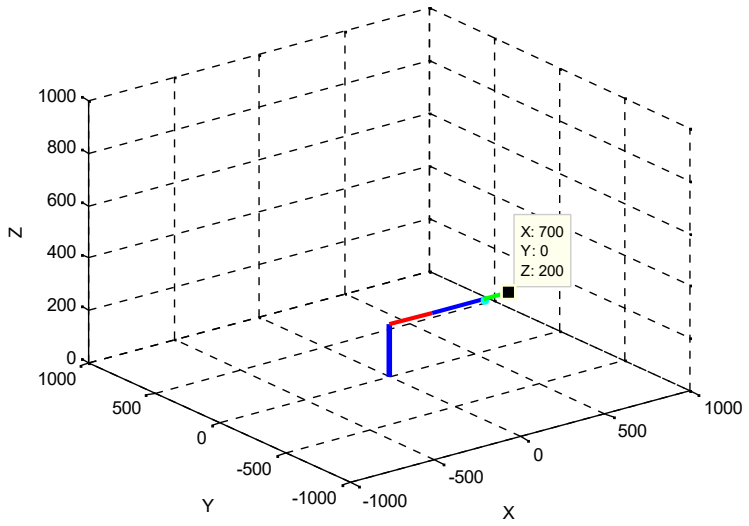


Ilustración 31, Segunda posición programada

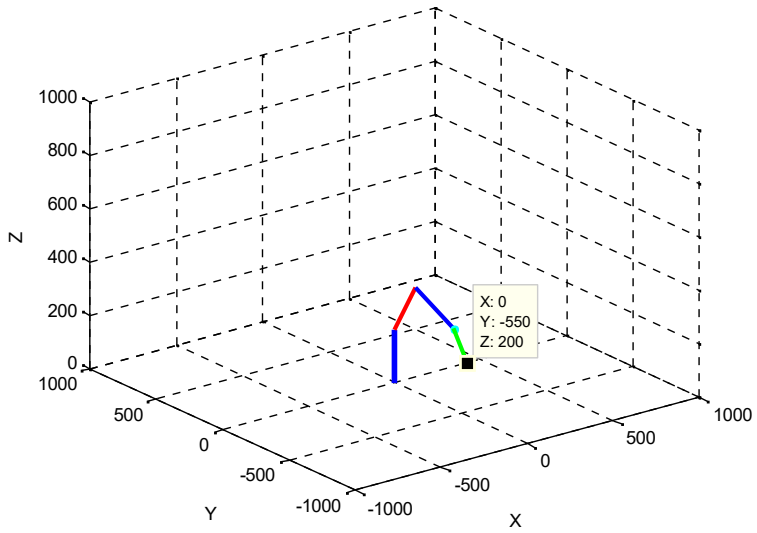


Ilustración 32, Posición final programada

En la generación de la trayectoria se genera un perfil polinomial, dado por los puntos especificados anteriormente, para llegar a cada posición especificada, cada articulación debe seguir una trayectoria diferente como se aprecia en la Ilustración 33, Trayectoria de cada articulación. La trayectoria es continua de polinomios con interpolaciones de restricción en su velocidad especificada en la sección del espacio articular, se puede ver un historial en el tiempo de cómo fue ese movimiento si ha sido constante o con variaciones en su velocidad de eso dependerá del tipo de tarea que deba realizar.

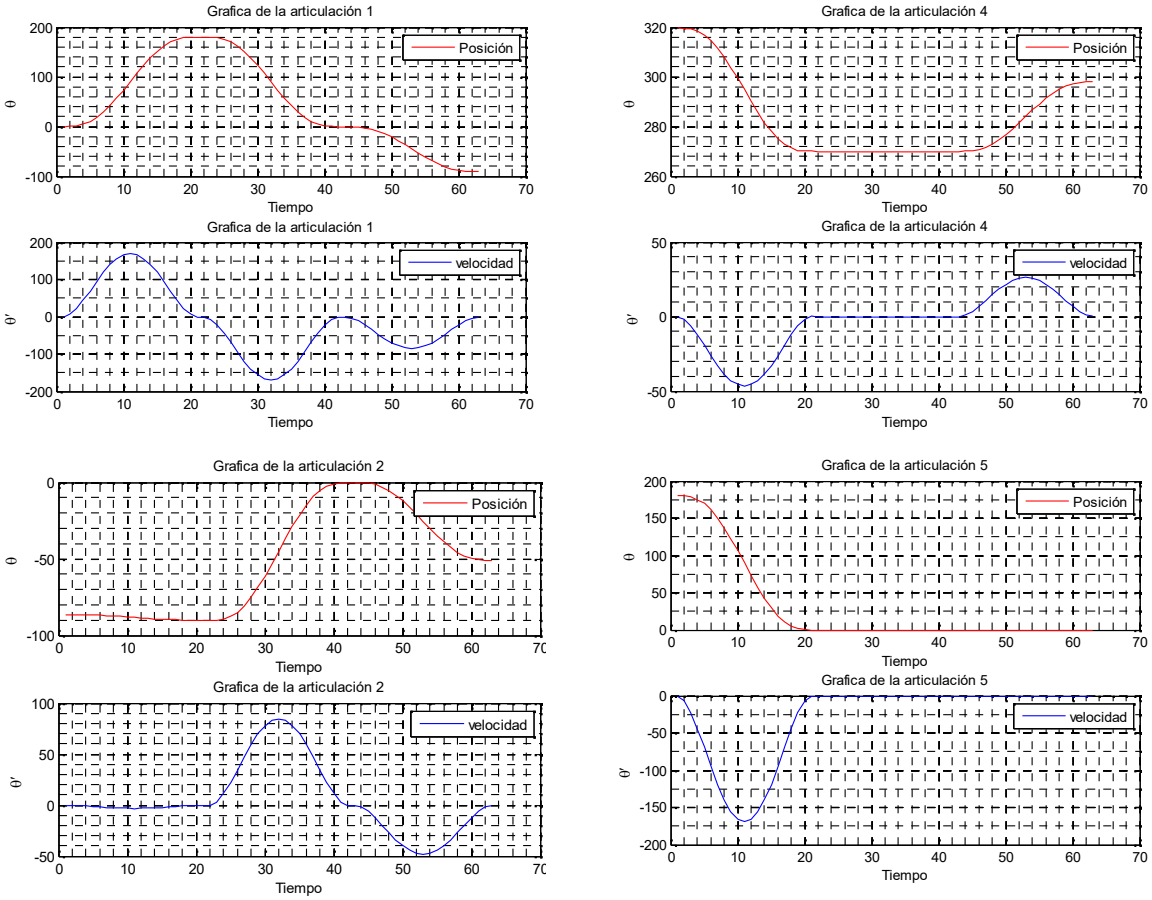


Ilustración 33, Trayectoria de cada articulación

CAPÍTULO 4. CODE COMPOSER STUDIO

4. PROGRAMACIÓN CON CODE COMPOSER STUDIO

En este capítulo se complementa con el análisis del control cinemático de los capítulos anteriores por medio de la programación previa en Matlab para simular y realizar los diferentes módulos a ser implementados en el DSC LaunchXL-F28027F Piccolo de la familia C2000.

Para la implementación en el microcontrolador se usa el Code Composer Studio que es un ambiente de desarrollo integrado (IDE), que soporta los microcontroladores y sistemas embebidos de Texas Instruments. Code Composer Studio integra un motor compilador de C/C++ optimizado además de integrar las ventajas del software Eclipse para depurar los diferentes sistemas embebidos ver Ilustración 34, Interfaz Code Composer Studio.

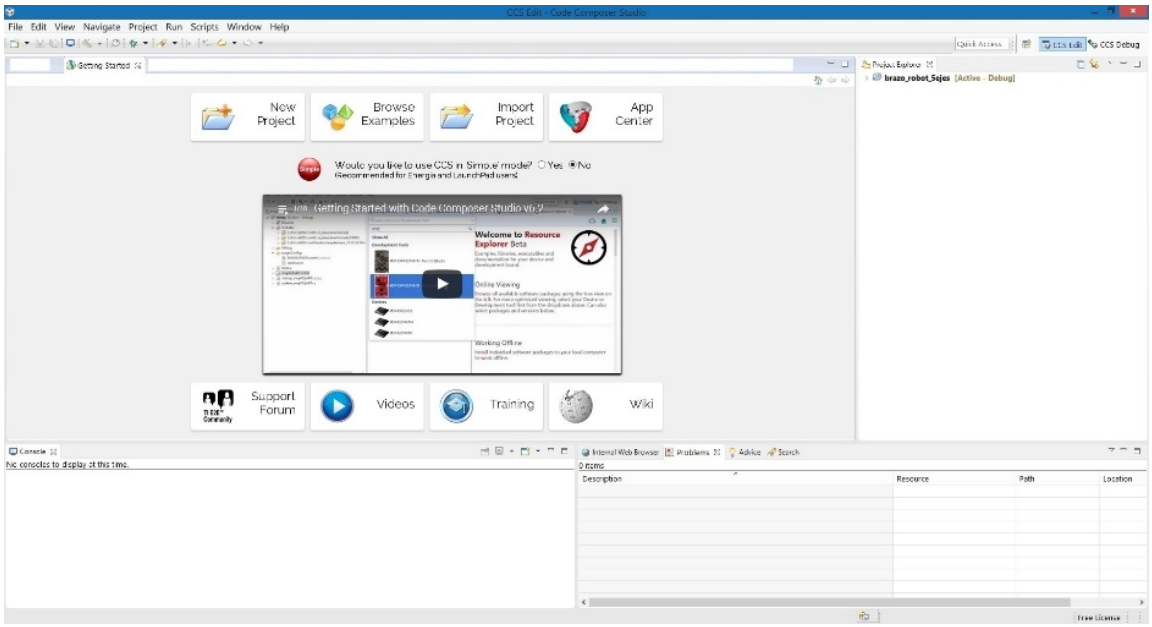


Ilustración 34, Interfaz Code Composer Studio

4.1. PROCESADOR DIGITAL DE SEÑAL

4.1.1. ARQUITECTURA DE LOS PROCESADORES DIGITALES

Los DSP (procesador digital de señales o procesamiento digital de señales) son microcontroladores diseñados para poder procesar algoritmos de señales digitales donde puede incluir multiplicaciones, operaciones matemáticas a alta velocidad, también tienen la capacidad de funcionar en aplicaciones en tiempo real. [3]

Con arquitectura de bus Harvard para separar los bloques de memoria de datos y de programas, así se puede usar bus de acceso independiente.

El DSP puede procesar instrucciones de forma paralela, su velocidad de procesamiento es más baja que la de un procesador de computador convencional, pero para las aplicaciones que debe realizar es suficiente.

Los DSP's son diseñados en una gran parte para los sistemas embebidos, es decir para sistemas autónomos, como teléfonos móviles, cámaras de fotografiar digitales, etc. Un microprocesador de computadora posee una velocidad de procesamiento mayor que la de un DSP, pero como no tiene instrucciones específicas para un tipo de operación o aplicación es más lento que el DSP para esas operaciones específicas. En la Ilustración 35, Arquitectura básica del DSP, se puede apreciar la arquitectura básica de un DSP y su arquitectura de bus Harvard que se explicara más adelante. [4]

[3] http://ocw.uv.es/ingenieria-y-arquitectura/sistemas-electronicos-para-el-tratamiento-de-la-informacion/seti_materiales/seti3_ocw.pdf

[4] http://www.electronicasi.com/wp-content/uploads/2013/04/dspElectronica-avanzada.pdf

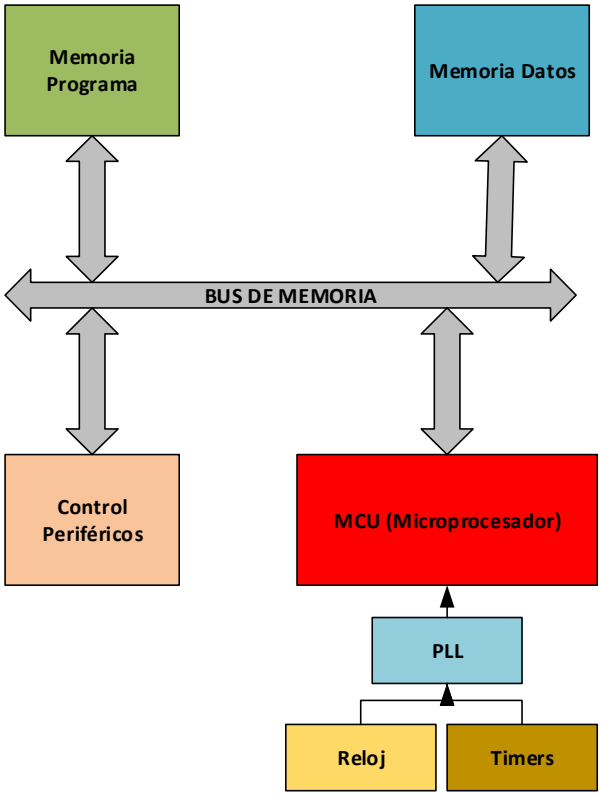
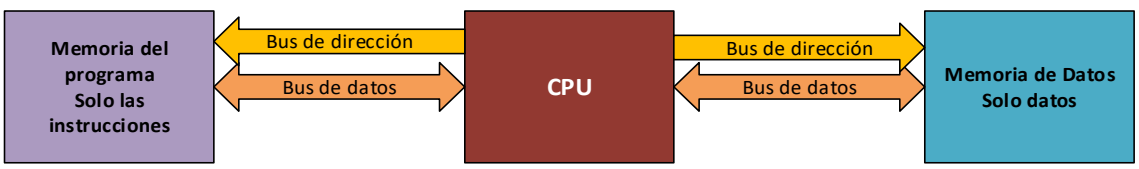


Ilustración 35, Arquitectura básica del DSP

Adicionalmente los DSP tienen hardware para realizar las operaciones matemáticas con un módulo llamado MAC (Multiply, Add y Accumulate) que multiplica, suma y acumula en un ciclo de reloj.

La memoria tiene varios buses para acceso a datos y a las direcciones, es decir arquitectura Harvard llamado así por su desarrollo en la universidad de Harvard en 1940 por Howard Aiken (1900-1973). Como se muestra en la Ilustración36, Arquitectura Harvard el cual muestra como la memoria de datos y la del programa están separados con buses independientes para que puedan funcionar al mismo tiempo o según la necesidad del procesador, es una arquitectura de doble bus, mejorando el rendimiento sobre la velocidad de las arquitecturas que son de un solo bus como la es la Von Neumann [5].



[5] http://www.dspguide.com/ch28/3.htm

Ilustración36, Arquitectura Harvard

4.1.2. CONTROLDOR DIGITAL DE SEÑALES

Los DSP o procesador digital de señales son solo el microprocesador, pero para evitar que el usuario implemente físicamente los periféricos como el PWM, ADC, SCI/UART, etc. Se crea el DSC o controlador digital de señales que es un hibrido entre el microcontrolador y el DSP, en otras palabras, es un DSP con periféricos para los sistemas embebidos. En donde ya puede procesar directamente los datos como son de analógico a digital o enviar de digital a analógico, comunicación, control de motores, etc.

Para la realización de este trabajo se usa el DSC TMS320F28027F Piccolo de la familia C2000 de Texas Instruments.

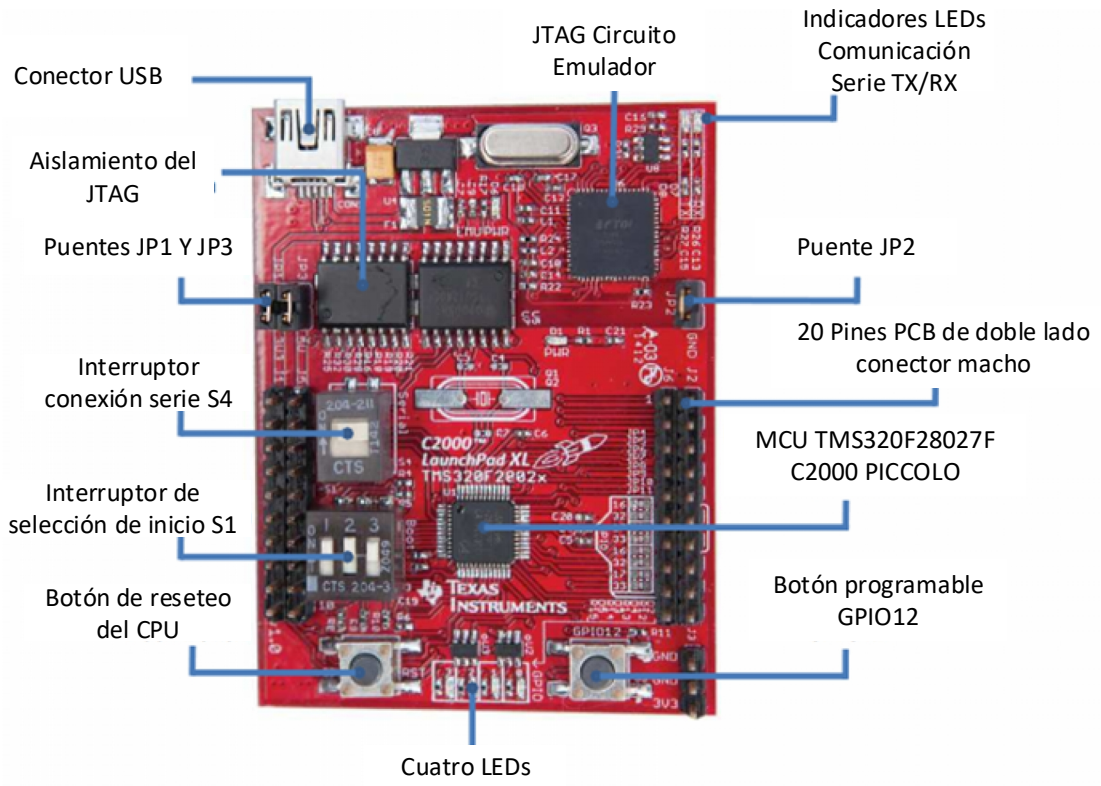


Ilustración 37, DSC LAUNCHXL-F28027F

Las características son las siguientes:

Número de parte	Descripción
TMS320F28027FPTT	Microcontrolador de 32-bits en tiempo real a 60MHz, 64kB Flash, 12kB RAM, 9 PWM, 4 HR PWM, 1 Capturador, 22 GPIOs, 13 canales, 12-bits 4.6 MSPS ADC, SCI/UART, SPI y

	I2C

4.1.3. EL MICROCONTROLADOR TMS320F28027F

La placa de desarrollo de Texas Instruments LAUNCHXL-F28027 de la familia C2000 Piccolo LaunchPad Experimenter Kit, tiene un microcontrolador de 32bits a 60 MHz, optimizado para el motor de C/C++ y para el desarrollo de algoritmos matemáticos para los sistemas de control. Con capacidades para funcionar en tiempo real, con la arquitectura de bus Harvard para la comunicación por el bus de lectura de 32 bits y de escritura de 32 bits. La ventaja de tener esta arquitectura es porque la memoria del bus puede priorizar el acceso a memoria ver Ilustración 38, Arquitectura del DSC LAUNCHXL-F28027F.

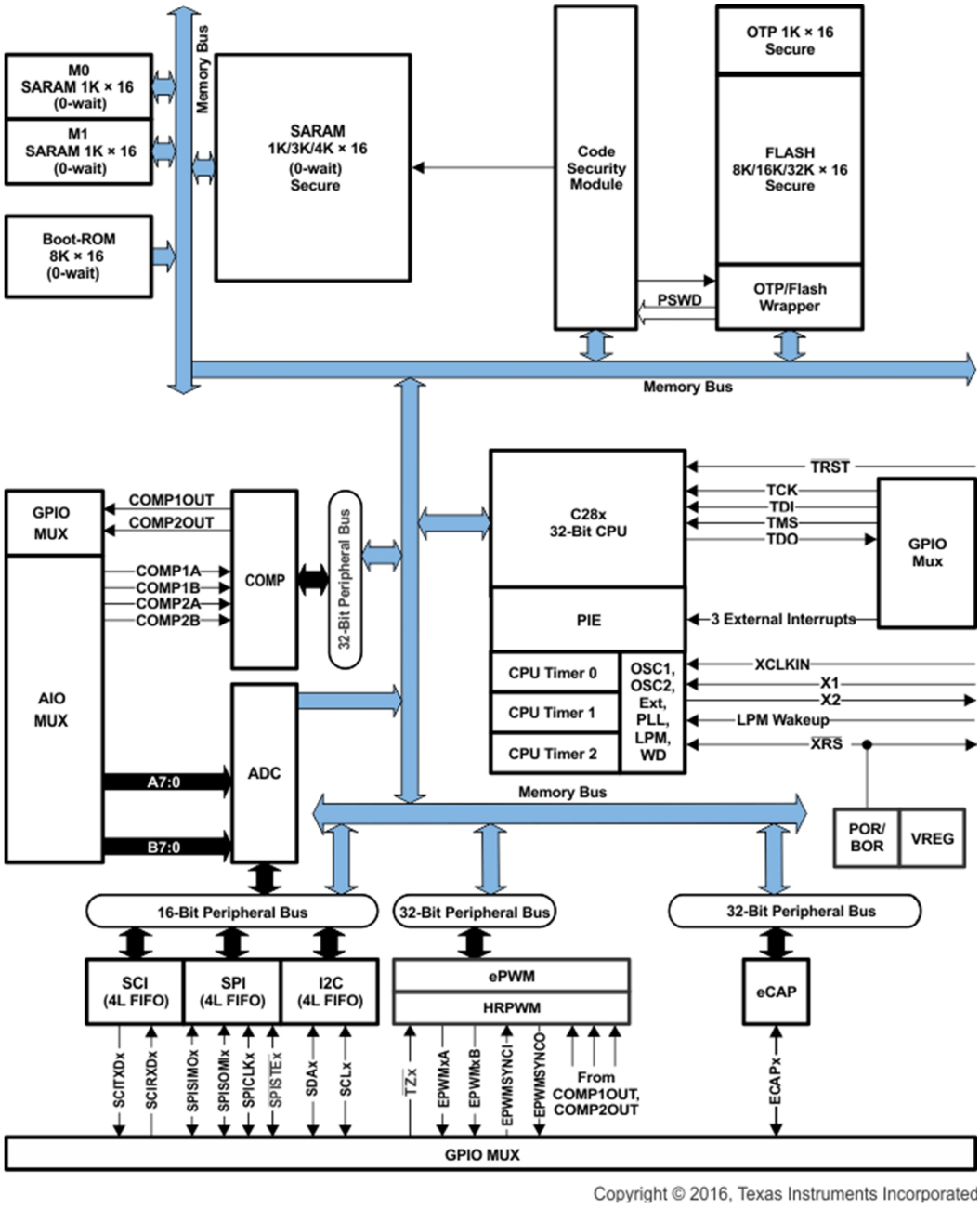


Ilustración 38, Arquitectura del DSC LAUNCHXL-F28027F

El DSC F28027F tiene el bus de periféricos de 16 y 32 bits, los módulos que lo complementan son el PWM para realizar control de la modulación del ancho del pulso, CAP este módulo es el de captura de entrada/salida usado para capturar hasta cuatro eventos y puede ser configurado para usarse como un PWM auxiliar, SCI (UART) sirve para la comunicación serie asíncrona con la computadora y se lo llama también comunicación de dos cables, SPI es un módulo de comunicación serie de alta velocidad síncrono de entrada/salida, I2C circuito de interface integrada otro módulo de comunicación serie que recibe y transmite 8 bits, GPIO y ADC que son los periféricos más 3 temporizadores del CPU.

4.1.4. PROGRAMACIÓN DEL MICROCONTROLADOR

En este apartado se va a explicar los módulos más significativos adaptados de los módulos en Matlab para el funcionamiento del brazo robot simulado como se muestra en la Ilustración 39, CCS y el manipulador de 5 ejes simulado en FreeBasic y el flujo grama que más adelante se explicara las funciones más relevantes del mismo ver Ilustración 41, Flujograma del control de trayectoria del DSC.

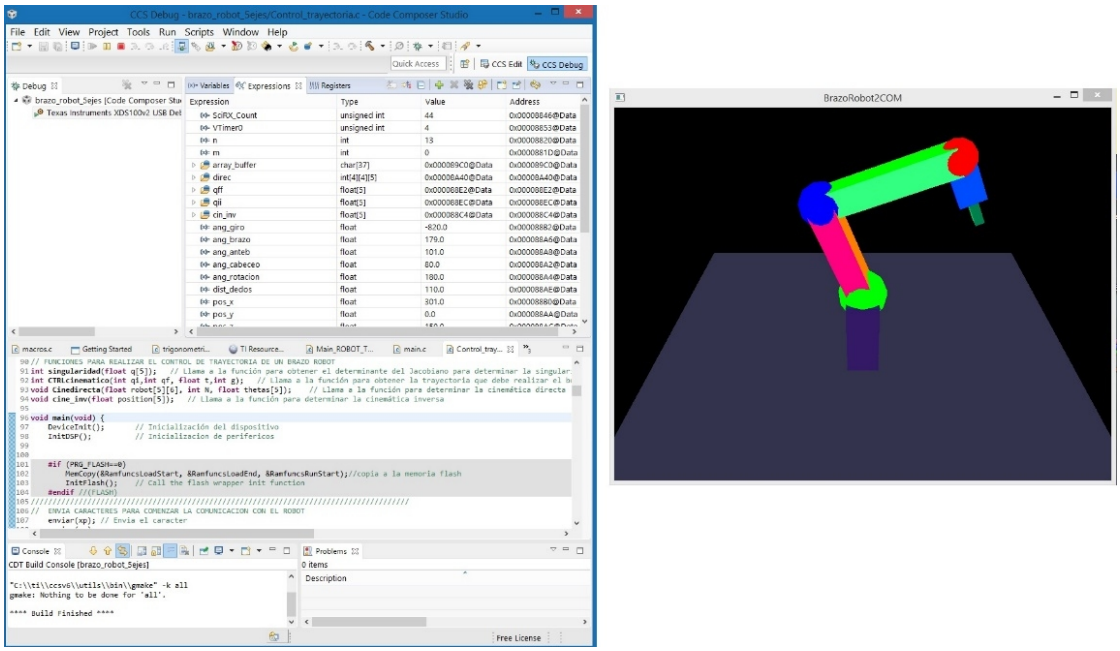


Ilustración 39, CCS y el manipulador de 5 ejes simulado en FreeBasic

El diagrama de funcionamiento del programa en la plataforma de desarrollo C2000 Piccolo F28027F tiene que gestionar las posiciones, como la inicial que es la del reposo del robot que está programada en FreeBasic con la posición 300 en X, 0 en Y y 150 en Z como punto de referencia para las posiciones finales programas. La Ilustración 40, Diagrama de funcionamiento del programa en el DSC, es un diagrama de bloques que resume todo el proceso que lleva para generar el movimiento y el control de la trayectoria del efector final del brazo robot.

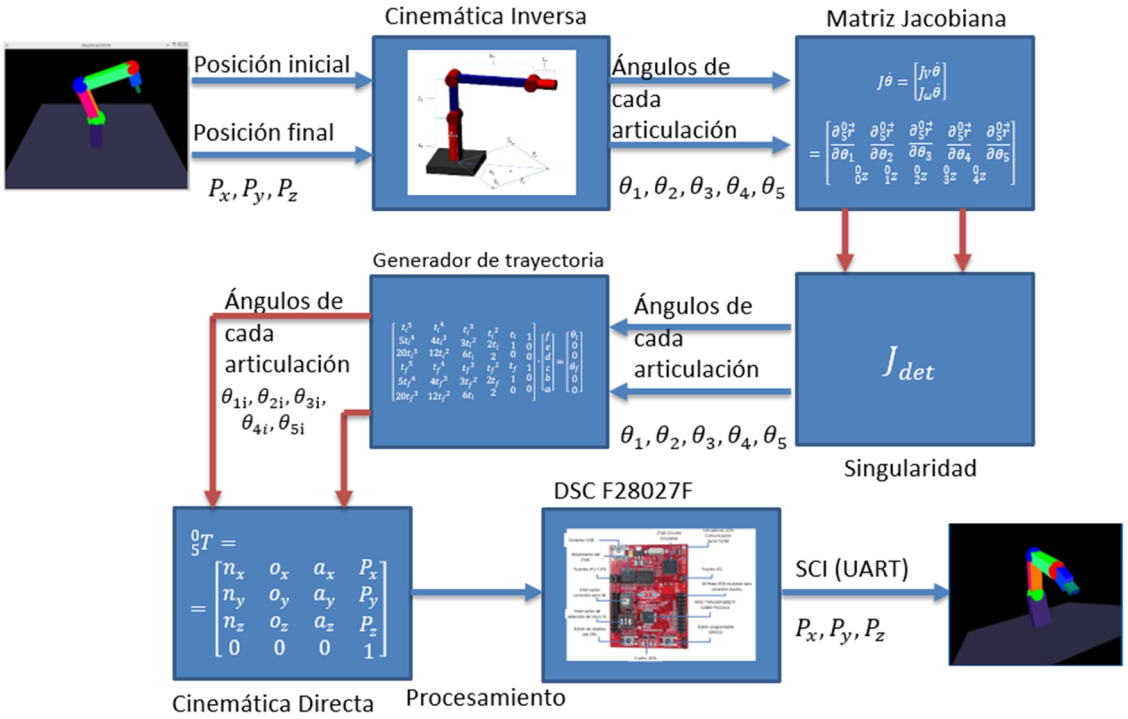
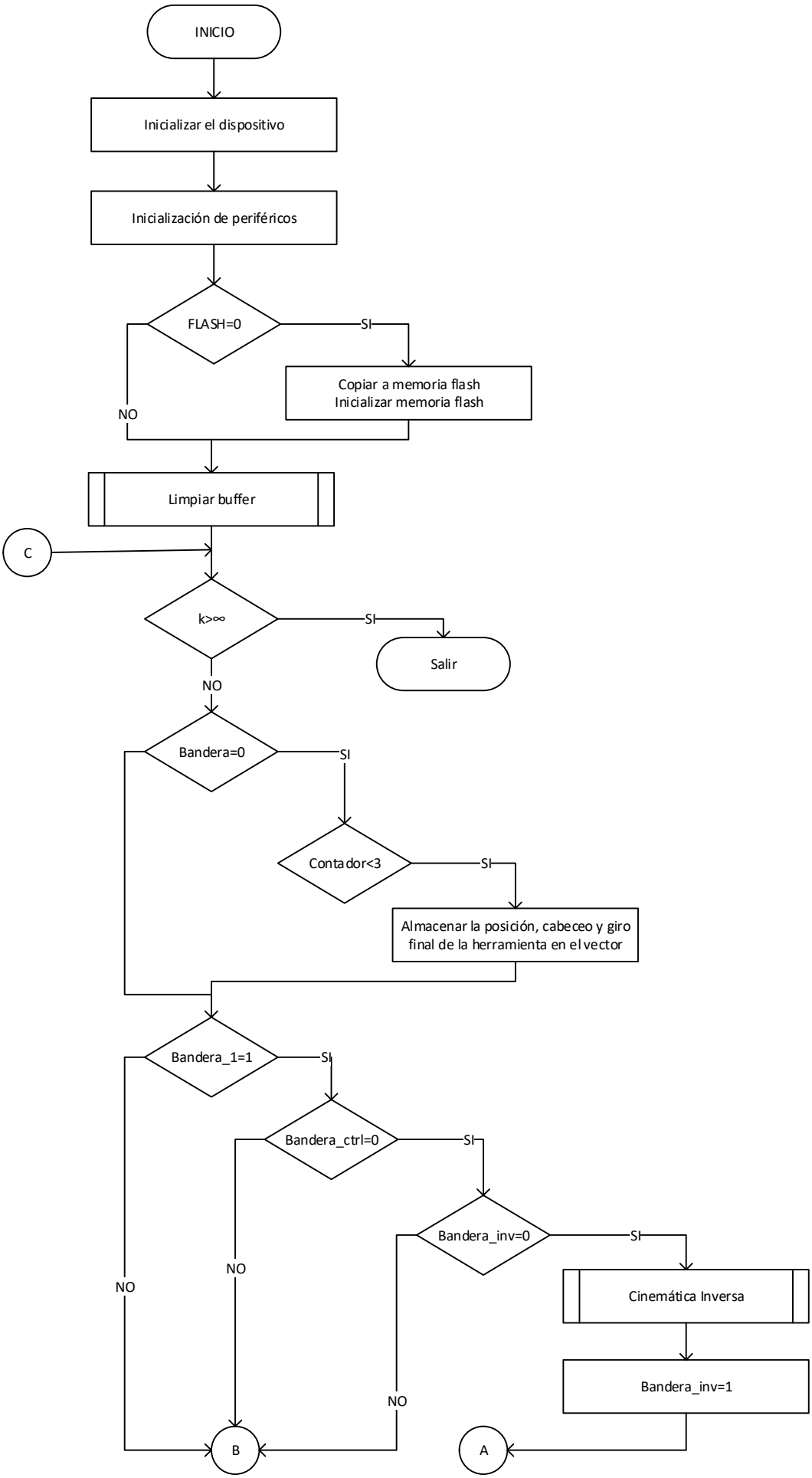


Ilustración 40, Diagrama de funcionamiento del programa en el DSC


INICIO
Inicializar el dispositivo
Inicialización de periféricos
FLASH=0
SI
NO
Copiar a memoria flash
Inicializar memoria flash
Limpiar buffer
C
k>∞
SI
Salir
NO
Bandera=0
SI
Contador<3
SI
Almacenar la posición, cabeceo y giro final de la herramienta en el vector
Bandera_1=1
SI
Bandera_ctrl=0
SI
Bandera_inv=0
SI
Cinemática Inversa
Bandera_inv=1
NO
NO
NO
NO
B
A

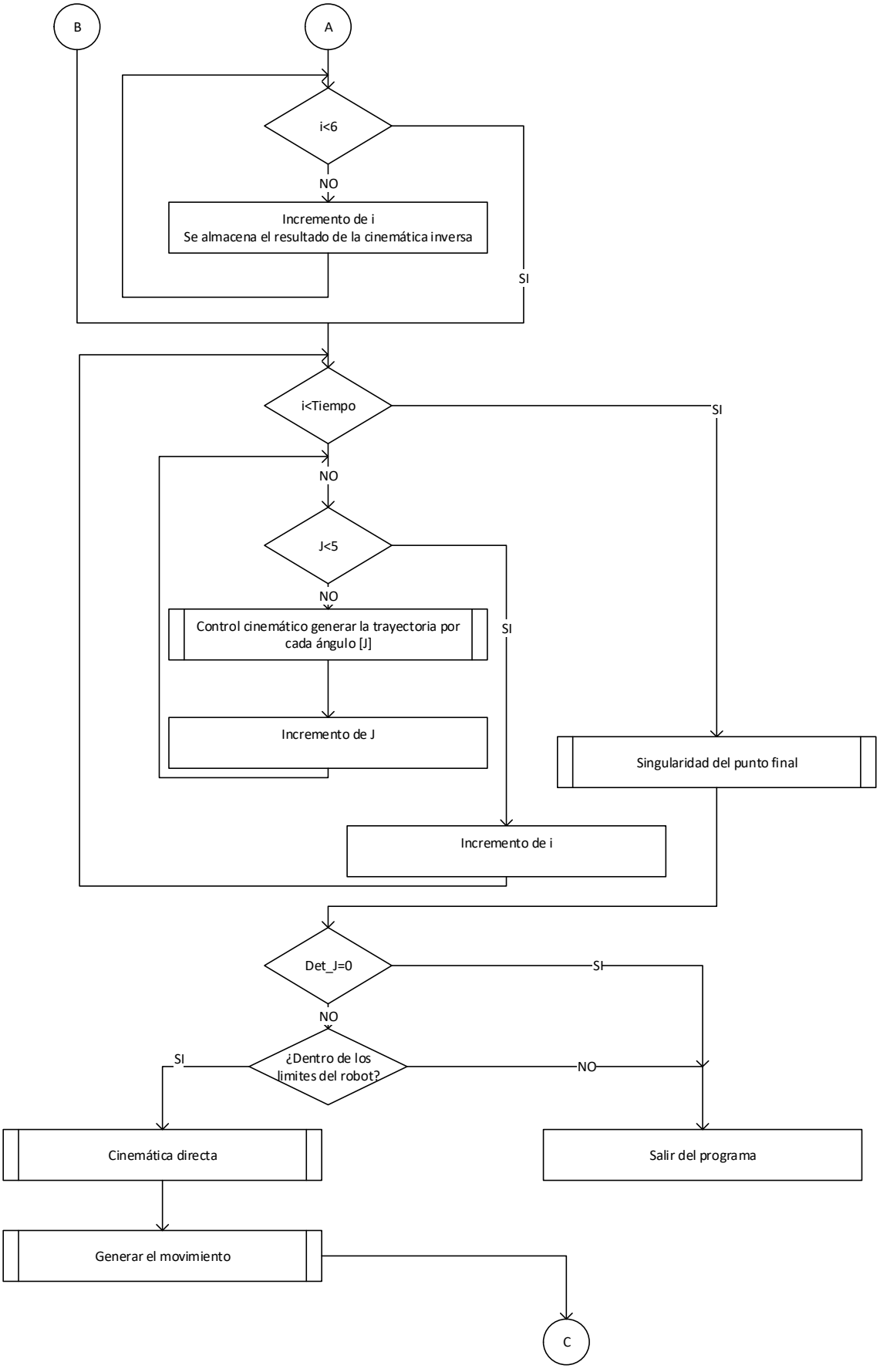


Ilustración 41, Flujograma del control de trayectoria del DSC

Las funciones creadas para la comunicación por medio de la UART o SCI al computador como la función scia_xmit( ) que permite la comunicación almacenando el carácter en el buffer de transmisión para solo transmitir. La función del while es de comparar hasta que se reciba el dato. La otra función enviar() llama tres veces a la función scia_xmit( ) para enviar el carácter correspondiente en primer lugar y los dos siguientes que son controles de caracteres, son el carácter CR (0x0D o 13) que es carriage return o retorno de carro y el ultimo el LF (0x0A o 10) que es line feed o salto de línea son terminadores de línea como protocolo de transmisión, ver para el resto de caracteres que se usan para el movimiento del manipulador en la Tabla 2, Codigo Ascii para los movimientos del robot.

Movimiento	Símbolo	Hexadecimal	Decimal
X+	D	0x0044	68
X-	A	0x0041	65
Y+	W	0x0057	87
Y-	S	0x0053	83
Z+	E	0x0045	69
Z-	Q	0x0051	81
Θ_4 Arriba	Z	0x005A	90
Θ_4 Abajo	X	0x0058	88
Θ_5 Horario	C	0x0043	67
Θ_5 Anti-horario	V	0x0056	86
Apertura pinzas	M	0x004D	77
Cerrado pinzas	N	0x004E	78

Tabla 2, Codigo Ascii para los movimientos del robot

```
// ENVIA UN CARACTER POR SCI
void scia_xmit(int a){
    while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
    SciaRegs.SCITXBUF=a;
}

// FUNCION QUE ENVIA TRES CARACTERES A LA FUNCION scia_xmit()
void enviar(int env){
       scia_xmit(env);
       scia_xmit(CR);
       scia_xmit(LF);

}
```

El bloque para la recepción de los datos enviados por el robot para tener su posición en cada movimiento, se usa funciones definidas en la librería de cabecera string.h como strtok() que nos ayuda a separar la cadena de caracteres en las que requerimos según un delimitador como en este caso es la “,” para posteriormente almacenarla en una variable y de la función atoi() definida en la librería de cabecera stdlib.h.

```
// FUNCION QUE SEPARA LOS DATOS EN LOS DIFERENTES ANGULOS Y POSICION
void recepcion(){

	msg=strtok(array_buffer, delim); // STRTOK FUNCION QUE SEPARA CADENAS POR EL
DELIMITADOR
	ang_giro = atoi(msg)-1000;  // ATOI FUNCION QUE CONVIERTE UNA CADENA EN ENTERO
	qii[0]=ang_giro+1000;
	msg=strtok(NULL, delim);
	ang_brazo = atoi(msg);
	qii[1]=ang_brazo;
	msg=strtok(NULL, delim);
	ang_anteb = atoi(msg);
	qii[2]=ang_anteb;
	msg=strtok(NULL, delim);
	ang_cabeceo = atoi(msg);
	qii[3]=ang_cabeceo;
	msg=strtok(NULL, delim);
	ang_rotacion = atoi(msg);
	qii[4]=ang_rotacion;
	msg=strtok(NULL, delim);
	dist_dedos = atoi(msg);
	msg=strtok(NULL, delim);
	pos_x = atoi(msg);
	msg=strtok(NULL, delim);
	pos_y = atoi(msg);
	msg=strtok(NULL, delim);
	pos_z = atoi(msg);
}
```

La función de la cinemática inversa usada en Matlab tiene algunas modificaciones para adaptarlo a C como usar la función abs() para obtener el valor absoluto, multiplicar por conversores a radianes y a grados.

```
// FUNCION DE LA CINEMATICA INVERSA
void cine_inv(float position[5]){
	float lh = 200; // Longitud del hombro
	float lb = 250; // Longitud del brazo
	float la = 300; // Longitud del antebrazo
	float lm = 150; // Longitud de la muñeca + herramienta
	px = position[0]; // Posición en el eje X
	py = position[1]; // Posición en el eje Y
	pz = position[2]; // Posición en el eje Z
	pitch = position[3]; // Cabeceo de la muñeca
	roll= position[4]; // Rotación de la muñeca

	// Calculo Theta 1
	if(px!= 0){
		cin_inv[0] = atan(py/px)*grad;
		if((cin_inv[0]<=-150)&&(cin_inv[0]>=150)){
			if(cin_inv[0]<-150){
				cin_inv[0] = -150;
			}
			else{
				cin_inv[0] = 150;
			}
		}
	}
	else{
		cin_inv[0] = atan(py/px)*grad;
	}
	if((px == 0)&&(py == 0)){
		cin_inv[0] = 180;
	}
```

```
        if((px!=0)&&(py!=0)){
                r = sqrt(abs(pow(px,2))+abs(pow(py,2)));
        }
        if(py==0){
                r=abs(px);
        }
        if(px==0){
                r=abs(py);
        }
        // Calculo Theta 2
        fx = lm*cos(pitch*rad);
        lx = r-fx;
        fy = lm*sin(pitch*rad);
        ly = pz-fy-lh;
        //if((lx!=0)&&(ly!=0)){
                l = sqrt((pow(lx,2))+(pow(ly,2)));
        //}
        if(ly==0){
                l=lx;
        }
        if(lx==0){
                l=ly;
        }
        alfaa = atan(ly/lx)*grad;
        beta = acosf(((pow(lb,2)+pow(l,2)-pow(la,2))/(2*lb*l)))*grad;
        cin_inv[1] = (-(alfaa+beta));
        if((px == 0)&&(py == 0)){
                cin_inv[1] = -90;
        }

        // Calculo Theta 3
        auxgam=pow(lb,2)+pow(la,2)-pow(l,2);
        auxgamd=2*(lb*la);
        gam = acos(auxgam/auxgamd)*grad;
        cin_inv[2] = (180-gam);
        if((px == 0)&&(py == 0)){
                cin_inv[2] = 0;
        }

        // Calculo Theta 4
        cin_inv[3] = 360-90+(-pitch-cin_inv[1]-cin_inv[2]);
        //cin_inv[3] = pitch-cin_inv[1]-cin_inv[2];

        cin_inv[4] = roll;
        // Theta 5 ingresa el usuario
}
```

La cinemática directa es una función fundamental al igual que la inversa aquí a partir de los datos generados por la cinemática inversa y el generador de trayectoria los convierte en datos cartesianos para dar movimiento al brazo robot. El presente código es una matriz homogénea de donde solo tomamos los valores de posición ver en Cinemática Directa Matriz Homogénea

```
// FUNCION DE LA CINEMATICA DIRECTA
void Cinedirecta(float robot[5][6], int N, float thetas[5]){
        // Limpia la matriz temporal
        for (id=0;id<4;id++){       // Bucle FOR para ingresar en la matriz temp e
igualarla a cero
                for (jd=0;jd<4;jd++){
                        temp[id][jd]=0;             // Matriz temporal para almacenar
la matriz homogenea
```

```
		}
	}
	// Crea una matriz unitaria
	for (id=0;id<4;id++){
		for (jd=0;jd<4;jd++){
			if (id != jd){
				Ai[id][jd][0] = 0;
			}
			else{
				Ai[id][jd][0] = 1;
			}
		}
	}
	// Se obtiene los valores de interes de la tabla de Denavit Hartenberg
	for (id=1;id<N+1;id++){
	    ad=robot[id-1][0];
	    alfa=robot[id-1][1];
	    //phi = 0 es revolucion phi = 1 es prismatico
	    if (robot[id-1][5] == 0){
	        theta=thetas[id-1];
	        dd=robot[id-1][2];
	    }
	    else{
	        dd=thetas[id-1];
	        theta=robot[id-1][3];
	    }
	    // Matriz homogénea DH clasica
	    A[0][0]=cos(theta*rad);              A[0][1]=-cos(alfa*rad)*sin(theta*rad);
A[0][2]=sin(alfa*rad)*sin(theta*rad); A[0][3]=ad*cos(theta*rad);
	    A[1][0]=sin(theta*rad);   A[1][1]=cos(alfa*rad)*cos(theta*rad);   A[1][2]=-
sin(alfa*rad)*cos(theta*rad); A[1][3]=ad*sin(theta*rad);
	    A[2][0]=0; A[2][1]=sin(alfa*rad); A[2][2]=cos(alfa*rad); A[2][3]=dd;
	    A[3][0]=0; A[3][1]=0; A[3][2]=0; A[3][3]=1;
	    // Multiplicación de matrices inicio
	    for (idd = 0; idd < 4; idd++){
	        for (jdd = 0; jdd < 4; jdd++){
	              temp[idd][jdd]=0;
	            for (kd = 0; kd < 4; kd++){
	                temp[idd][jdd] += Ai[idd][kd][id-1] * A[kd][jdd];
	            }
	        }
	    }
	    // Multiplicación de matrices fin
	    // Almacenado de las matrices homogeneas en la matrz auxiliar
	    for (idd = 0; idd < 4; idd++) {
	        for (jdd = 0; jdd < 4; jdd++) {
	                Ai[idd][jdd][id]=temp[idd][jdd];
	        }
	    }
	}
	// Almacena en la matriz homogénea final por cada eje
	for (id=0;id<N;id++){
		for (idd=0;idd<4;idd++){
			for (jdd = 0; jdd < 4; jdd++) {
				direc[idd][jdd][id]=Ai[idd][jdd][id+1];
			}
		}
	}
	pff_aux[0]=direc[0][3][4];
	pff_aux[1]=direc[1][3][4];
	pff_aux[2]=direc[2][3][4];
	pff_aux[3]=qff[3];
	pff_aux[4]=qff[4];
}
```

La función del generador de trayectoria es parecida al de Matlab genera un polinomio de quito orden donde cada valor se almacena en una variable.

```
// FUNCION DEL CONTROL CINEMATICO POR MEDIO DE UN POLINOMIO DE 5TH ORDEN
int CTRLcinematico(int qi,int qf, float t, int g){
   a=(qi*pow(t,3)*pow(t,2)/pow(t,5));
   b=0;
   c=0;
   d=(qf*(10*pow(t,2))/pow(t,5))-(qi*(10*pow(t,2))/pow(t,5));
   e=(qi*(15*t)/pow(t,5))-(qf*(15*t)/pow(t,5));
   f=6*qf/pow(t,5)-6*qi/pow(t,5);
   q=(f*pow(g/10.0,5) + e*pow(g/10.0,4) + d*pow(g/10.0,3) + c*pow(g/10.0,2) + b*g +
a);
   //printf("%d ",i);
   return q;
 // printf("\n");
}
```

La función para determinar la singularidad del sistema está compuesta por la determinante del Jacobiano antes explicado, aquí se almacena cada ángulo generado por la cinemática inversa en una variable para que se pueda procesar en la ecuación del determinan.

```
int singularidad(float qs[5]){
        th1=qs[0];
        th2=qs[1];
        th3=qs[2];
        th4=qs[3];
        th5=qs[4];

        a1=0; a4=0; a5=0; al1=-PI/2; al2=0; al3=0; al4=-PI/2; al5=0; d2=0; d3=0;
        d4=0; th6=th5; a2=250; a3=300; d1=200; d5=150;
        Jdet=-(a2*a3*(a3*cos(th3+th4)-a3*cos(2*th2+th3+th4)-d5*sin(2*th2+3*th3+2*th4)-
a2*cos(th4)+a2*cos(2*th2+2*th3+th4)+a3*cos(2*th2+3*th3+th4)+2*d5*sin(th3)+d5*sin(2*th
2+th3+2*th4)-a2*cos(2*th2+th4)+a2*cos(2*th3+th4)-a3*cos(th3-th4)))/4;
        return(Jdet);
}
```

El DSC tiene que inicializarse con otros módulos que se anexaran al trabajo sin embargo hay dos funciones importantes, primero cuando se inicializa la configuración del reloj del CPU del DSC se lo deja por defecto es decir a 60MHz por lo que su tiempo de muestreo es de 16.667×10^{-9} segundos o $16.667\ \eta$.

$$T = \frac{1}{F} = \frac{1}{60 \times 10^{6}} \tag{97}$$

Para la comunicación del DSC al computador está configurado a una velocidad de transmisión de 9600 Baudios que indica el número de bits que se transmiten por segundo.

Una de las funciones que es necesaria es de la interrupción de un timer del CPU para generar un tiempo de retardo de 100ms por ciclo del proceso completo, en el programa por medio de un while se realiza la espera hasta un valor decimal designado en este caso 1. El problema radica en la velocidad de transmisión puede tener perdida de datos y por consecuencia una lectura incorrecta pudiendo generarse un movimiento incorrecto por esta razón el tiempo de retardo es importante ya que es en tiempo real.

```
void InitTimers (void){

        EALLOW;
                SysCtrlRegs.PCLKCR3.bit.CPUTIMER0ENCLK = 1; // CPU Timer-0
        EDIS;

        // Asignamos los manejadores de interrupción a los timers
        EALLOW;
                PieVectTable.TINT0 = &TA0_Cpu_Timer0_isr;
        EDIS;
        // Inicializamos los perifericos de los temporizadores
        InitCpuTimers();   // For this example, only initialize the Cpu Timers

        // Configuramos los temporizadores
        ConfigCpuTimer(&CpuTimer0, 60, 150000);           //100ms


        CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit =
0

        VTimer0 = 0;   // Contador del Evento del Timer-0
}
```

La siguiente función que también puede usarse como retardo es una función que genera un " NOP" que es no operation causando que el CPU no haga nada por ese periodo especificado pero no es un tiempo real y cada llamada al "NOP" ocupa todo el tiempo de muestreo del CPU en no hacer nada es decir en 16.667 η .

```
// FUNCION DE RETARDO
void temporizador(){
        while(contwait++<50000){
                asm(" NOP"); // es un retardo de software 16.66667ns => t=1/f=1/60mhz
                EALLOW;
                SysCtrlRegs.WDKEY = 0x55; // Habilita Watchdog llave de reseteo al
registro para la escritura
                SysCtrlRegs.WDKEY = 0xAA; // Watchdog llave de reseteo al registro pone
al contador en 0
                EDIS;
        }
        contwait=0;
}
```

El resultado se puede contrastar con lo generado en Matlab y obtenemos unas posiciones similares como las siguientes:

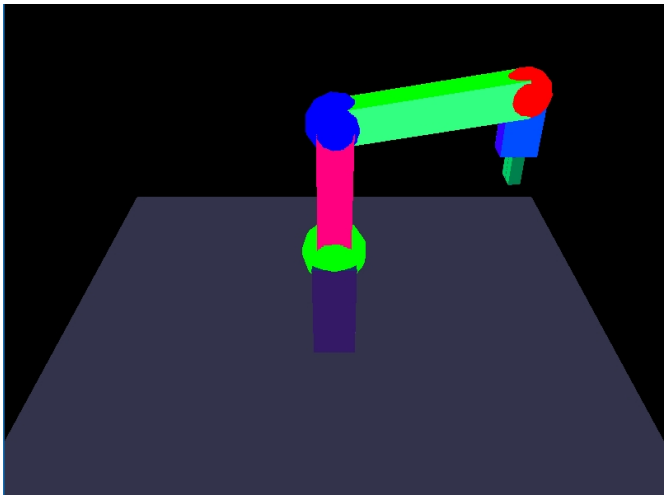


Ilustración 42, Posición inicial

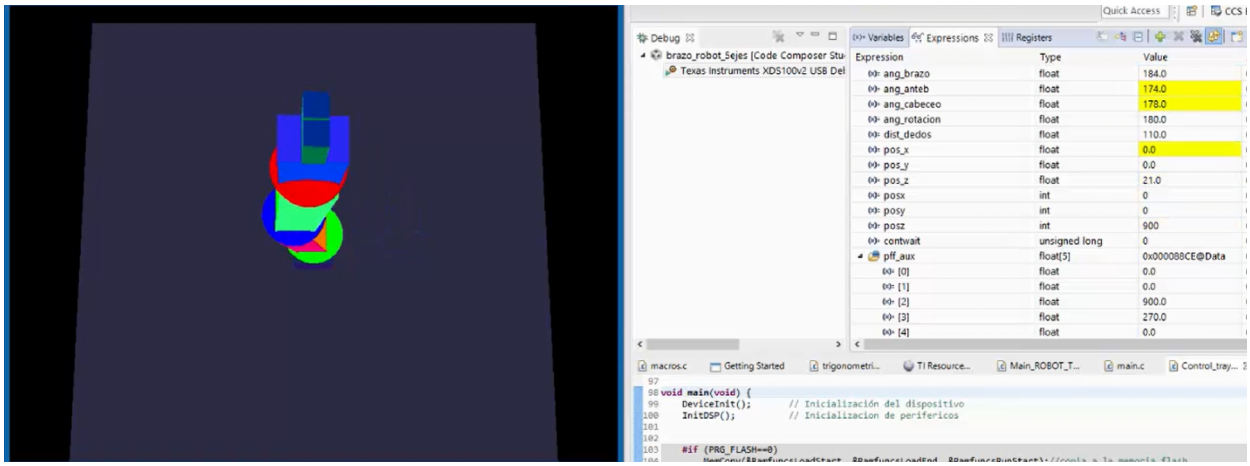


Ilustración 43, Primera posición programada del DSC

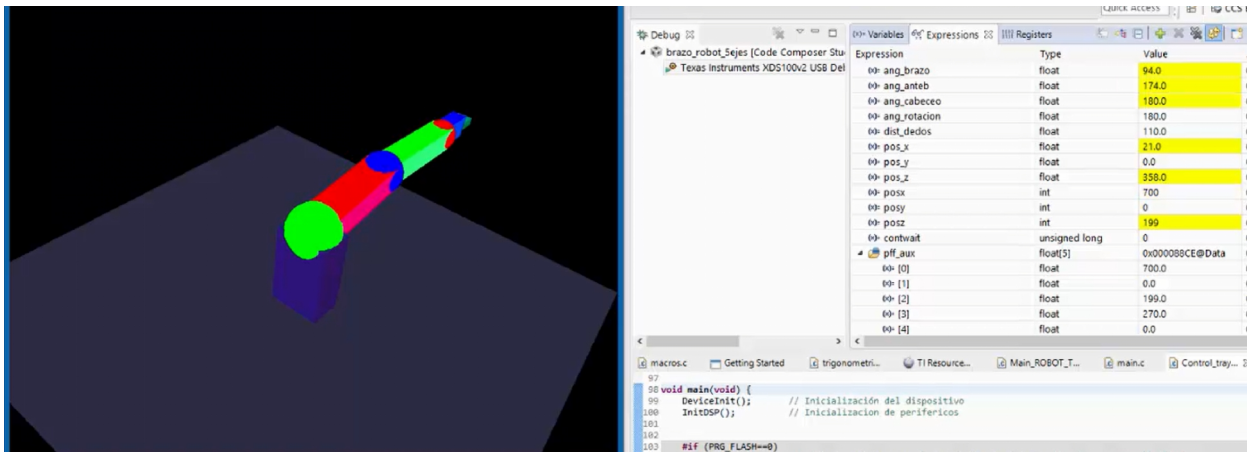


Ilustración 44, Segunda posición programada del DSC

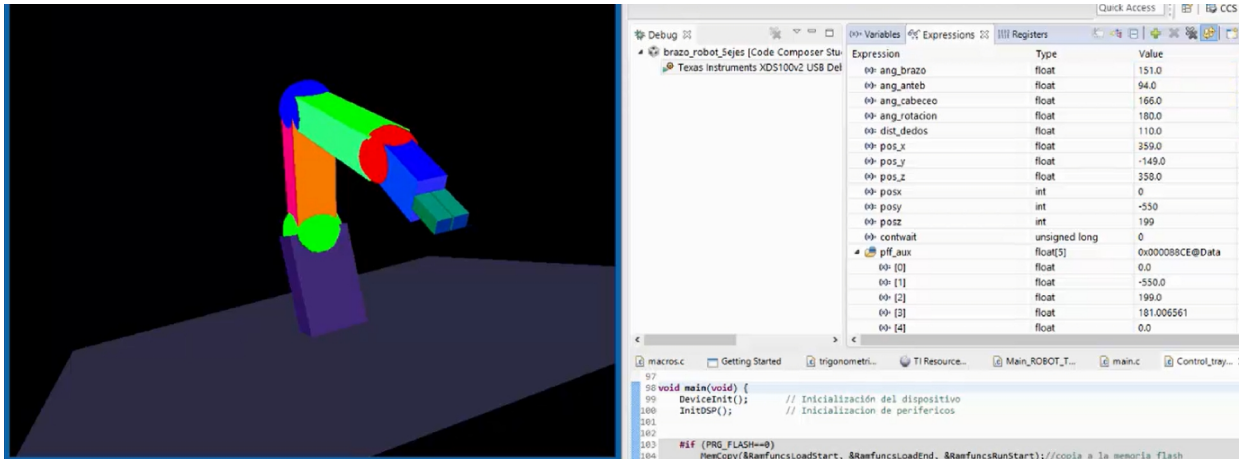


Ilustración 45, Tercera posición programada del DSC

5. CONCLUSIONES

- Una buena definición de los parámetros de Denavit-Hartenberg es muy importante, porque es la configuración de la cadena cinemática entre eslabones del brazo robot para que esta pueda ser tratada para el movimiento cinemático. La mala determinación puede causar errores en la configuración del brazo robot.
- La cinemática inversa y directa al ser bien determinadas deben devolver el valor de entrada del otro, es decir si el usuario ingresa una posición el controlador u ordenador debe procesarlo con la cinemática inversa para tener como resultado la orientación o valores angulares para después ser tratados con la cinemática directa y devolver la posición. Así el usuario va a poder manejar solo valores de posición que es mucho más simple que tratar con valores angulares.
- En la determinación de las singularidades del manipulador no se tiene un control total de los limites, por lo que se debe realizar controles adicionales de límites para definir los admisibles y no admisibles al que el brazo robot puede llegar.
- La generación de la trayectoria por medio de un polinomio puede ser simplificada de un quinto orden a un tercero dependiendo de las restricciones impuestas, sin embargo, se baja la resolución, pero se gana facilidad de procesamiento que con respecto a un mayor orden polinomial requiere más tiempo de procesamiento.
- El algoritmo de control de trayectoria en el DSC que realiza los cálculos y envía los datos para el cambio de posición del efector final es cada 150ms.
- La simulación del brazo robot de 5 ejes tiene el problema que está definido de tal forma que los valores que devuelve son desde 0 a 359, que pertenecen a los ángulos, limitando el buen desarrollo del control ya que debería devolver valores enteros perteneciente al espacio cartesiano, así también como la suma adicional del valor de la altura de la base para la posición en el eje Z. Debiéndose de realizarse cambios en el programa de Matlab para el programa del DSC.

6. PRESUPUESTO

El desembolso económico que ha supuesto la realización el trabajo es el que se detalla a continuación:

CANTIDAD	ELEMENTO	FABRICANTE	P. UNIT.	P. TOTAL
1	LAUNCHXL-F28027F	TEXAS INSTRUMENTS	30,66 €	30,66 €
CANTIDAD	**ELEMENTO**	**TIEMPO**	**P. HORA**	**P. TOTAL**
	Información de Robótica	50	11 €	550 €
	Implementación en Matlab	150	11 €	1.650 €
	Implementación en Code Composer Studio	20	11 €	220 €
	TOTAL			**2.451 €**

Tabla 3, Presupuesto

7. BIBLIOGRAFÍA

- *Asociación Española de Robotica y Automatización Tecnologías de la Producción*. (2016, 06 22). Retrieved from http://www.aeratp.com/estadisticas/internacionales/
- Corke, P. (2013). *Robotics, vision and control Fundamental Algorithms in Matlab.* Stanford : Spinger.
- Cortés Parejo, J. (2008, Marzo). *La representación Denavit-Hartenberg*. Retrieved from http://personal.us.es/jcortes/Material/Material_archivos/Articulos%20PDF/RepresentDH.pdf
- CRAIG , J. (2006). *ROBÓTICA* (TERCERA EDICION ed.). México: PEARSON EDUCATION.
- Jaramillo Botero, A. (n.d.). *Cinemática de Manipuladores Robóticos.* Retrieved from http://www.wag.caltech.edu/home/ajaramil/libro_robotica/cinematica.pdf
- Subir Kumar , S. (2010). *INTRODUCCIÓN A LA ROBÓTICA* (1ED ed.). México: Mc Graw Hill Inter-Americana.
- *TEXAS INSTRUMENTS.* (2016, JUNIO). Retrieved from http://www.ti.com/lit/ds/symlink/tms320f28027.pdf
- Universidad La Rioja. (2015, Noviembre 26). *Youtube*. (C. Elvira Izurrategui , Producer) Retrieved from https://www.youtube.com/watch?v=9Ri5i4bq7Ns&t=0s
- University, S. (n.d.). *Stanford Engineering Everywhere.* Retrieved from https://see.stanford.edu/Course/CS223A/35

8. ANEXOS

8.1 Programa del control de trayectoria para el DSC F28027F

```
#include "DSP2802x_Device.h"               // DSP2802x Headerfile Include File
#include "PeripheralHeaderIncludes.h"      // Esta funcion contiene todas las
definiciones para configurar los relojes de los perifericos asi como las
interrupciones
#include "DSP2802x_Examples.h"             // Configura el reloj del dsc por medio del
PLL
#include <string.h>                                    // Funcion que define todas
las funciones para usarlas en la construccion y operación con cadenas de caracteres
#include <stdio.h>                                     // Archivo de cabecera de
E/S que incluye las definiciones de las macros, las constantes y declaraciones de
funciones de la biblioteca standar de C
#include <stdlib.h>                                    // Archivo de cabecera que
incluye las funciones prototipo para gestión de la memoria dinámica, control de
procesos, entre otras
#include <math.h>                                      // Script de cabecera para
incluir funciones matematicas
#include <complex.h>
#include "GLOBAL_DEFS.h"                          // Definiciones globales lo del
reloj del DSC de los perifericos
#include "GLOBAL_VARS.h"                          // Variables globales contadores de
los eventos de loa timers, de los SCI, interrupciones y banderas
#include "GLOBAL_FUNCS.h"                         // Funciones globales de
inicializacion de perifericos, interrupciones y timers

#define PI   3.14159265358979323846264338327795    // Se define al valor PI
#define PRG_FLASH 1                          // Define la constante PRG_FLASH = 1
#define CR 13//0x0D                          // Carriage Return
#define LF 10//0x0A                          // Line Feed

// FUNCIONES PARA LA TRANSMISION Y RECEPCION
void scia_xmit(int a);              // Llama a la funcion que transmite caracter por
caracter por el periferico SCI
void scia_msg(char *msg);    // La funcion envia caracter por caracter del mensaje
almacenado hasta el valor NULL
void recepcion();                   // LLama a la función que almacena los ángulos y
posiciones en las variables correspondientes
void limpiar_buffer();
void enviar(int env);
// FUNCION DE RETARD
void temporizador();

// VARIABLES PARA LA COMUNICACION Y RECEPCION DE LOS DATOS
char *msg;     // Puntero del mensaje
char delim[]=","; // Variable delimitadora para separar a los datos recibidos
char array_buffer[37];       // Vector que almacena los datos recibidos en serie por
el robot
int n=0,m=0,k=0;             // Variables de uso general para las iteraciones en las
funciones y programa principal
// TABLA DE PARAMETROS DE DENAVIT HARTENBERG
float TablaDH[5][6]={ {0,-90,200,180,1,0},{250,0,0,180,2,0},
   {300,0,0,100,3,0},{0,-90,0,80,4,0},{0,0,150,180,5,0} };

// TIEMPO PARA LA RESOLUCION EN EL CONTROL CINEMATICO
int tiempo=2;

// CONSTANTES DE LOS COMANDOS PARA ENVIO DISPUESTOS EN X+,X-,Y+,Y-,Z+,Z-
,TH4up,TH4dn,TH5r,TH5l,Fo,Fc
// COMANDOS ENVIADOS SON: D, A, Q, E, W, S, Z, X, C, V, M, N en su respectivo orden
int xp=68,xn=65,yp=87,yn=83,zp=69,zn=81,th4p=90,th4n=88,th5r=67,th5l=86,fo=77,fc=78;
```

```
//int movimientos[12]={68,65,87,83,69,81,90,88,67,86,77,78};  //DEC
//int movimientos[12]={0x44,0x41,0x57,0x53,0x45,0x51,0x5A,0x58,0x43,0x56,0x4D,0x4E};
//HEX

// VARIABLES PARA LOS ANGULOS Y POSICION DE LOS EJES DEL BRAZO ROBOT
float ang_giro=0, ang_brazo=0, ang_anteb=0, ang_cabeceo=0, ang_rotacion=0,
dist_dedos=0, pos_x=0, pos_y=0, pos_z=0;

// VARIABLES PARA EL CALCULO DE LA SINGULARIDAD
float det_j, Jdet, val_J;
int th1, th2, th3, th4, th5, th6, x, y;
int a1, a2, a3, a4, a5, al1, al2, al3, al4, al5, d1, d2, d3, d4, d5;

// VARIABLES PARA EL CALCULO DEL CONTROL CINEMATICO
int a, b, c, d, e, f, g, i ,j, q;

// VARIABLES PARA EL CALCULO DE LA CINEMATICA DIRECTA
float Ai[4][4][6],A[4][4];
float temp[4][4];
int direc[4][4][5];
int id,idd,jd,jdd,kd;
float ad,alfa,theta,dd;
float rad=PI/180.0; // convierte de grados a radianes -> cos,sin,tan

// VARIABLES PARA EL CALCULO DE LA CINEMATICA INVERSA
float px, py, pz, pitch, roll, r, fx, lx, fy, ly, l, alfaa, beta, gam, cin_inv[5],
auxgam, auxgamd;
float grad=180.0/PI; // convierte de radianes a grados -> acos, asin,atan

// VARIABLES GLOBALES PARA LA POSICION FINAL DADA POR EL USUARIO
float array_pff[3][5]={{0,0,900,90,0},{700,0,200,0,0},{0,-550,200,45,0}};
float pff[5];  // Variable que almacena las posiciones XYZ, cabeceo y rotación
float pff_aux[5];     // Variable que almacena las posiciones XYZ, cabeceo y rotación
para la comparación en el programa principal

// VARIABLES PARA LOS ANGULOS INICIALES Y FINALES DEL ROBOT
float qii[5];         //={0,-81.1426,110.4873,330.6553,180.0000};
float qff[5];       //{180,90,0,270,0};
int tray[20][5];      // Variable que almacena la trayectoria generado por el control
cinematico

// VARIABLES GLOBAES
int contsing=0, flag_inv=0, flag_rx=0, flag_ctrl=0, posx=0, posy=0, posz=0,
flag_ret=0, flag_eq=0;

// VARIABLE PARA LA FUNCION DE RETARDO
unsigned long contwait=0;

// FUNCIONES PARA REALIZAR EL CONTROL DE TRAYECTORIA DE UN BRAZO ROBOT
int singularidad(float q[5]);       // Llama a la función para obtener el determinante
del Jacobiano para determinar la singularidad en el espacio de trabajo
int CTRLcinematico(int qi,int qf, float t,int g); // Llama a la función para obtener
la trayectoria que debe realizar el brazo robot
void Cinedirecta(float robot[5][6], int N, float thetas[5]);     // Llama a la función
para determinar la cinemática directa
void cine_inv(float position[5]);   // Llama a la función para determinar la
cinemática inversa

void main(void) {
       DeviceInit();        // Inicialización del dispositivo
       InitDSP();                  // Inicializacion de perifericos

       #if (PRG_FLASH==0)
              MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd,
&RamfuncsRunStart);//copia a la memoria flash
              InitFlash();  // Call the flash wrapper init function
```

```
	#endif //(FLASH)
//ENVIA CARACTERES PARA COMENZAR LA COMUNICACION CON EL ROBOT
	enviar(xp);	// Envia el caracter
	enviar(xn);
//	LIMPIA EL BUFFER DE RECEPCION
	limpiar_buffer();
//	BUCLE SIN FIN
	for(;;){
	while(VTimer0 >= 1){
		VTimer0 = 0;
		EALLOW;
		SysCtrlRegs.WDKEY = 0x55;	// service WD #1
		SysCtrlRegs.WDKEY = 0xAA;
		EDIS;
		if(flag_ret==0){
			if(k<3){
			pff[0]=array_pff[k][0];
			pff[1]=array_pff[k][1];
			pff[2]=array_pff[k][2];
			pff[3]=array_pff[k][3];
			pff[4]=array_pff[k][4];
			flag_ret=1;
			}
		}
		if(flag_rx==1){
			if(flag_ctrl==0){
// PROCESO QUE LLAMA A LA FUNCION CINEMATICA INVERSA
// PARA DE UNA POSICION DADA DEVUELVA LOS ANGULOS DE CADA EJE DEL ROBOT
			if(flag_inv==0){
				cine_inv(pff);
				flag_inv=1;
				for(i=0;i<6;i++){
					qff[i]=cin_inv[i];
				}
			}
// PROCESO QUE OBTIENE LA TRAYECTORIA A SEGUIR
// ESTE PROCESO ES EL CONTROL CINEMATICO
				for (i=0;i<(tiempo*10);i=i+1){
				for (j=0;j<5;j++){
		tray[i][j]=CTRLcinematico(qii[j],qff[j],tiempo,i);
				}
				}
// PROCESO QUE OBTIENE LA SINGULARIDAD DE LA POSICION FINAL
// EN EL CASO QUE TENGA UNA SINGULARIDAD PERSISTENTE EL PROGRAMA SE TERMINA
			det_j=singularidad(qff);
			val_J=cimag(det_j);
				if(val_J!=0){
					exit(0);
				}
		if ((pff[0]>700)||(pff[0]<-700)||(pff[1]>700)||(pff[1]<-
700)||(pff[2]>900)||(pff[2]<0)){
					exit(0);
		}
// FUNCION QUE REALIZA LA CINEMATICA DIRECTA
// DESPUES DEL ANALISIS PREVIO PARA COMPROBAR QUE ESA POSICION FINAL ES VALIDA
			Cinedirecta(TablaDH,5,qff);
			flag_ctrl=1;
			if(flag_eq==0){
			posx=pos_x;
			posy=pos_y;
			posz=pos_z+200;
			flag_eq=1;
			}
			}
				if(posx<(int)pff_aux[0]){
					enviar(xp);
```

```
                                        posx++;
                                }
                                if(posx>(int)pff_aux[0]){
                                        enviar(xn);
                                        posx--;
                                }
                                if(posy<(int)pff_aux[1]){
                                        enviar(yp);
                                        posy++;
                                }
                                if(posy>(int)pff_aux[1]){
                                        enviar(yn);
                                        posy--;
                                }
                                if(posz<(int)pff_aux[2]){
                                        enviar(zp);
                                        posz++;
                                }
                                if(posz>(int)pff_aux[2]){
                                        enviar(zn);
                                        posz--;
                                }
                                if(ang_cabeceo<((int)pff_aux[3]-90)){
                                        enviar(th4p);

                                }
                                if(ang_cabeceo>((int)pff_aux[3]-90)){
                                        enviar(th4n);
                                }

        if(((pos_x==(int)pff_aux[0])&&(pos_y==(int)pff_aux[1])&&(posz==(int)pff_aux[2]
))||((posx==(int)pff_aux[0])&&(posy==(int)pff_aux[1])&&(posz==(int)pff_aux[2]))){
                                        flag_ret=0;
                                        flag_ctrl=0;
                                        flag_inv=0;
                                        enviar(xp);
                                        enviar(xn);
                                        k++;

                                }
                                if(k==3){exit(0);}

                                //temporizador();

                        }
                }
        }

}
// FUNCION QUE LIMPIA EL VECTOR GLOBAL QUE ALMACENA LOS DATOS DE LA RECEPCION
void limpiar_buffer(){
        for(i=0;i<38;i++){
                        array_buffer[i]=NULL;
        }
}
// FUNCION DE RETARDO
void temporizador(){
        while(contwait++<50000){
                asm(" NOP"); // es un retardo de software 16.66667ns => t=1/f=1/60mhz
                EALLOW;
                SysCtrlRegs.WDKEY = 0x55; // Habilita Watchdog llave de reseteo al
registro para la escritura
                SysCtrlRegs.WDKEY = 0xAA; // Watchdog llave de reseteo al registro pone
al contador en 0
                EDIS;
        }
```

```
        contwait=0;
}
//      INTERRUPCIÓN QUE ACTIVA EL TIMER
interrupt void TA0_Cpu_Timer0_isr(void){
        VTimer0++;
        EALLOW;
                SysCtrlRegs.WDKEY = 0xAA;    // service WD #2
        EDIS;
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
//      FUNCIONES PARA ENVIO DE LOS VALORES ASCII EN DEC
/*
  @brief Con esta función se envian los valores DEC para que el brazo robot pueda
  moverse segun el codigo en el vector
*/
// FUNCION QUE ENVIA UN CARACTER POR COMUNICACION SERIE
void enviar(int env){
        scia_xmit(env);
        scia_xmit(CR);
        scia_xmit(LF);

}
// FUNCION QUE SEPARA LOS DATOS EN LOS DIFERENTES ANGULOS Y POSICION
void recepcion(){

        msg=strtok(array_buffer, delim); // STRTOK FUNCION QUE SEPARA CADENAS SEGUN EL
DELIMITADOR
        ang_giro = atoi(msg)-1000;  // ATOI FUNCION QUE CONVIERTE UNA CADENA EN ENTERO
        qii[0]=ang_giro+1000;
        msg=strtok(NULL, delim);
        ang_brazo = atoi(msg);
        qii[1]=ang_brazo;
        msg=strtok(NULL, delim);
        ang_anteb = atoi(msg);
        qii[2]=ang_anteb;
        msg=strtok(NULL, delim);
        ang_cabeceo = atoi(msg);
        qii[3]=ang_cabeceo;
        msg=strtok(NULL, delim);
        ang_rotacion = atoi(msg);
        qii[4]=ang_rotacion;
        msg=strtok(NULL, delim);
        dist_dedos = atoi(msg);
        msg=strtok(NULL, delim);
        pos_x = atoi(msg);
        msg=strtok(NULL, delim);
        pos_y = atoi(msg);
        msg=strtok(NULL, delim);
        pos_z = atoi(msg);
}
// FUNCION DE LA CINEMATICA INVERSA
/**
  @brief Funcion que devuelve los ángulos de cada articulación del brazo robot de la
posición
  cabeceo y rotación ingresados de la muñeca o efector final, datos del punto a
 */
void cine_inv(float position[5]){
        float lh = 200; // Longitud del hombro
        float lb = 250; // Longitud del brazo
        float la = 300; // Longitud del antebrazo
        float lm = 150; // Longitud de la muñeca + herramienta
        px = position[0]; // Posición en el eje X
        py = position[1]; // Posición en el eje Y
        pz = position[2]; // Posición en el eje Z
        pitch = position[3]; // Cabeceo de la muñeca
        roll= position[4]; // Rotación de la muñeca
```

```
        // Calculo Theta 1
        if(px!= 0){
                cin_inv[0] = atan(py/px)*grad;
                if((cin_inv[0]<=-150)&&(cin_inv[0]>=150)){
                        if(cin_inv[0]<-150){
                                cin_inv[0] = -150;
                        }
                        else{
                                cin_inv[0] = 150;
                        }
                }
        }
        else{
                cin_inv[0] = atan(py/px)*grad;
        }
        if((px == 0)&&(py == 0)){
                cin_inv[0] = 180;
        }
        if((px!=0)&&(py!=0)){
                r = sqrt(abs(pow(px,2))+abs(pow(py,2)));
        }
        if(py==0){
                r=abs(px);
        }
        if(px==0){
                r=abs(py);
        }
        // Calculo Theta 2
        fx = lm*cos(pitch*rad);
        lx = r-fx;
        fy = lm*sin(pitch*rad);
        ly = pz-fy-lh;
        //if((lx!=0)&&(ly!=0)){
                l = sqrt((pow(lx,2))+(pow(ly,2)));
        //}
        if(ly==0){
                l=lx;
        }
        if(lx==0){
                l=ly;
        }
        alfaa = atan(ly/lx)*grad;
        beta = acosf(((pow(lb,2)+pow(l,2)-pow(la,2))/(2*lb*l)))*grad;
        cin_inv[1] = (-(alfaa+beta));
        if((px == 0)&&(py == 0)){
                cin_inv[1] = -90;
        }

        // Calculo Theta 3
        auxgam=pow(lb,2)+pow(la,2)-pow(l,2);
        auxgamd=2*(lb*la);
        gam = acos(auxgam/auxgamd)*grad;
        cin_inv[2] = (180-gam);
        if((px == 0)&&(py == 0)){
                cin_inv[2] = 0;
        }

        // Calculo Theta 4
        cin_inv[3] = 360-90+(-pitch-cin_inv[1]-cin_inv[2]);
        //cin_inv[3] = pitch-cin_inv[1]-cin_inv[2];

        cin_inv[4] = roll;
        // Theta 5 ingresa el usuario
}
```

```
// FUNCION DE LA CINEMATICA DIRECTA
/**
  @brief Funcion que devuelve la posición X, Y y Z de los ángulos de cada
articulación
  del brazo robot de su punto final, usando la tabla de Denavit Hartenberg y matrices
  homogéneas. Para a posterior realizar las comparaciones para realizar las otras
  funciones.
 */
void Cinedirecta(float robot[5][6], int N, float thetas[5]){
       // Limpia la matriz temporal
       for (id=0;id<4;id++){        // Bucle FOR para ingresar en la matriz temp e
igualarla a cero
              for (jd=0;jd<4;jd++){
                     temp[id][jd]=0;              // Matriz temporal para almacenar
la matriz homogenea
              }
       }
       // Crea una matriz unitaria
       for (id=0;id<4;id++){
              for (jd=0;jd<4;jd++){
                     if (id != jd){
                            Ai[id][jd][0] = 0;
                     }
                     else{
                            Ai[id][jd][0] = 1;
                     }
              }
       }
       // Se obtiene los valores de interes de la tabla de Denavit Hartenberg
       for (id=1;id<N+1;id++){
           ad=robot[id-1][0];
           alfa=robot[id-1][1];
           //phi = 0 es revolucion phi = 1 es prismatico
           if (robot[id-1][5] == 0){
               theta=thetas[id-1];
               dd=robot[id-1][2];
           }
           else{
               dd=thetas[id-1];
               theta=robot[id-1][3];
           }
           // Matriz homogénea DH clasica
           A[0][0]=cos(theta*rad); A[0][1]=-cos(alfa*rad)*sin(theta*rad);
A[0][2]=sin(alfa*rad)*sin(theta*rad); A[0][3]=ad*cos(theta*rad);
           A[1][0]=sin(theta*rad); A[1][1]=cos(alfa*rad)*cos(theta*rad); A[1][2]=-
sin(alfa*rad)*cos(theta*rad); A[1][3]=ad*sin(theta*rad);
           A[2][0]=0; A[2][1]=sin(alfa*rad); A[2][2]=cos(alfa*rad); A[2][3]=dd;
           A[3][0]=0; A[3][1]=0; A[3][2]=0; A[3][3]=1;

           // Multiplicación de matrices inicio
           for (idd = 0; idd < 4; idd++){
               for (jdd = 0; jdd < 4; jdd++){
                     temp[idd][jdd]=0;
                   for (kd = 0; kd < 4; kd++){
                       temp[idd][jdd] += Ai[idd][kd][id-1] * A[kd][jdd];
                   }
               }
           }
           // Multiplicación de matrices fin
           // Almacenado de las matrices homogeneas en la matrz auxiliar
           for (idd = 0; idd < 4; idd++) {
               for (jdd = 0; jdd < 4; jdd++) {
                       Ai[idd][jdd][id]=temp[idd][jdd];
               }
           }
       }
```

```
        // Almacena en la matriz homogénea final por cada eje
        for (id=0;id<N;id++){
                for (idd=0;idd<4;idd++){
                        for (jdd = 0; jdd < 4; jdd++) {
                                direc[idd][jdd][id]=Ai[idd][jdd][id+1];
                        }
                }
        }
        pff_aux[0]=direc[0][3][4];
        pff_aux[1]=direc[1][3][4];
        pff_aux[2]=direc[2][3][4];
        pff_aux[3]=qff[3];
        pff_aux[4]=qff[4];

}
// FUNCION DEL CONTROL CINEMATICO POR MEDIO DE UN POLINOMIO DE 5TH ORDEN
/**
  @brief Funcion que mediante los 5 thetas de la posicion inicial
  y con los 5 thetas de la posicion final mas un tiempo de ejecucion
  se obtiene una matriz de angulos para cada theta o eje del brazo robot
  necesarios para sumar o restar los thetas iniciales hasta los thetas finales.
 */
int CTRLcinematico(int qi,int qf, float t, int g){
   a=(qi*pow(t,3)*pow(t,2)/pow(t,5));
   b=0;
   c=0;
   d=(qf*(10*pow(t,2))/pow(t,5))-(qi*(10*pow(t,2))/pow(t,5));
   e=(qi*(15*t)/pow(t,5))-(qf*(15*t)/pow(t,5));
   f=(6*qf/pow(t,5))-(6*qi/pow(t,5));
   q=(f*pow(g/10.0,5) + e*pow(g/10.0,4) + d*pow(g/10.0,3) + c*pow(g/10.0,2) + b*g +
a);
   //printf("%d ",i);
   return q;
 // printf("\n");
}
// FUNCION PARA EL CALCULO DE LA SINGULARIDAD DEL BRAZO ROBOT
/**
  @brief Funcion que mediante los 5 thetas de la posicion final determinara
  la singularidad del sistema mediente el determinante del Jacobiano si da
  cero dara aviso por medio de una bandera para que en una segunda vez realice
  la funcion break en el cuerpo principal y termine el programa.
 */
int singularidad(float qs[5]){
        th1=qs[0];
        th2=qs[1];
        th3=qs[2];
        th4=qs[3];
        th5=qs[4];

        a1=0; a4=0; a5=0; al1=-PI/2; al2=0; al3=0; al4=-PI/2; al5=0; d2=0; d3=0;
        d4=0; th6=th5; a2=250; a3=300; d1=200; d5=150;
        Jdet=-(a2*a3*(a3*cos(th3+th4)-a3*cos(2*th2+th3+th4)-d5*sin(2*th2+3*th3+2*th4)-
a2*cos(th4)+a2*cos(2*th2+2*th3+th4)+a3*cos(2*th2+3*th3+th4)+2*d5*sin(th3)+d5*sin(2*th
2+th3+2*th4)-a2*cos(2*th2+th4)+a2*cos(2*th3+th4)-a3*cos(th3-th4)))/4;
        return(Jdet);
}
//      INICIALIZACIÓN DE LOS PERIFERICOS DEL DSP
/**
  @brief Con esta función se inicializan los perifericos del DSC TMS320F28027

    InitTimers();     // Inicialización de los temporizadores
       InitADC();     // Inicialización del conversor analógico - digital
       InitPWM();     // Inicializacion del módulo PWM 1 y 2
       InitSci();     // Inicializamos el modulo SCI
       InitInt();     // Inicialización de las interrupciones
       @return NO se devuelve nada
```

```
*/
void InitDSP(){
	// Deshabilitamos las interrupciones de la CPU
	DINT;
	// Deshabilitamos las interrupciones de la CPU y borramos los flags
	IER = 0x0000;
	IFR = 0x0000;
	InitSci();
	InitInt();
	InitTimers(); // Inicialización de los temporizadores
}
//	INTERRUPCIÓN DE RECEPCIÓN DEL BUS SCI-A
/**
  @brief Función de interrupción de recepción (RX) del SCI-A
  Se produce esta interrupción cada vez que se reciben 4 bytes desde el
  bus SCI-A
  ////////////////////////////////////////////////////////////////
  // PETICION desde el PC-Ordenador ==> DSC ....
  // TRAMA: 0x05 - ORDEN - MENSAJE - CHKSUM
  // ORDEN: 0x51 ==> Envio_Tension
  // ORDEN: 0x52 ==> Envio_Frecuencia
  // ORDEN: 0x53 ==> Recepcion_Angulo
  ////////////////////////////////////////////////////////////////
  Utiliza los valores de los arrays: trama_RX[] y trama_TX[]
  @return NO se devuelve nada
*/
interrupt void SCIA_RXFIFO_isr(void){
	SciRX_Count++;
	// Almacenar valores de la TRAMA....
	array_buffer[n] = SciaRegs.SCIRXBUF.all;
	if((array_buffer[n-1]==CR)&&(array_buffer[n]==LF)){
		recepcion();
		flag_rx=1;
		n=0;
	}
	n++;
	// Decodificar o interpretar la ORDEN de la TRAMA....

	// Restaurar Flags para siguiente interrupción.....
	SciaRegs.SCIFFRX.bit.RXFFOVRCLR=1;   // Clear Overflow flag
	SciaRegs.SCIFFRX.bit.RXFFINTCLR=1;   // Clear Interrupt flag
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;       // Issue PIE ack
}
//	INICIALIZACIÓN DE LOS TEMPORIZADORES
/**
  @brief Función de inicialización de los Temporizadores del 28027

  Inicialización de timer 0 del DSC.
  Se configuran de la siguiente manera:
	Timer0:100ms

*/
void InitTimers (void){

	EALLOW;
		SysCtrlRegs.PCLKCR3.bit.CPUTIMER0ENCLK = 1; // CPU Timer-0
	EDIS;

	// Asignamos los manejadores de interrupción a los timers
	EALLOW;
		PieVectTable.TINT0 = &TA0_Cpu_Timer0_isr;
	EDIS;
	// Inicializamos los perifericos de los temporizadores
	InitCpuTimers();   // For this example, only initialize the Cpu Timers

	// Configuramos los temporizadores
```

```
        ConfigCpuTimer(&CpuTimer0, 60, 150000);           //100ms
        CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit =
0

        VTimer0 = 0;   // Contador del Evento del Timer-0
//      INCIALIZACIÓN DEL MODULO SCI
  @brief Función de inicialización del bus SCIA

  Inicialización del módulo de comunicaciones SCI-A. En el caso de la placa
  launchpad C2000, este módulo se corresponde con la comunicación USB, que
  genera en el PC un puerto COM mediante el cual nos podemos comunicar como
  un UART.
  Parametros de comunicacion: 9600, 8, N; 1
  Habilitamos la FIFO...
  Configuramos para que cuando haya 4 bytes en la FIFO provoque una interrupcion (RX)
void InitSci(void) {

        EALLOW;
                GpioCtrlRegs.GPAPUD.bit.GPIO28 = 0;    // Habilitamos resistencia de
PULL-UP (SCIRXDA)
                GpioCtrlRegs.GPAPUD.bit.GPIO29 = 1;    // Deshabilitamos resistencia de
PULL-UP (SCITXDA)

                GpioCtrlRegs.GPAQSEL2.bit.GPIO28 = 3;  // Entrada asincrona en GPIO28
(SCIRXDA)

                GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 1;   // Configuramos el puerto como
SCIRXDA
                GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 1;   //Configuramos el puerto como
SCITXDA

                SysCtrlRegs.PCLKCR0.bit.SCIAENCLK = 1; // Habilitamos el reloj para
SCI-A
    EDIS;

        SciaRegs.SCICCR.bit.SCICHAR = 7;    // 8 bits de datos
        SciaRegs.SCICTL1.bit.RXENA = 1;            // Habilitamos la recepción
        SciaRegs.SCICTL1.bit.TXENA = 1;            // Habilitamos la transmisión

        SciaRegs.SCICTL2.bit.RXBKINTENA = 1;       // Habilitamos la interrupción
RXRDY
        SciaRegs.SCICTL2.bit.TXINTENA = 1;         // Habilitamos la interrupción
TXRDY

        SciaRegs.SCIHBAUD = 0;
        SciaRegs.SCILBAUD = 194;            //BaudRate = 9600

        SciaRegs.SCIFFTX.bit.SCIFFENA = 1;  //Habilitamos la FIFO
        SciaRegs.SCIFFRX.bit.RXFFIENA = 1;  //Habilitamos la interrupción de la fifo de
recepción
        SciaRegs.SCIFFRX.bit.RXFFIL = 4;    //La FIFO solo tiene 4 niveles

        EALLOW;
                PieVectTable.SCIRXINTA = &SCIA_RXFIFO_isr;
        EDIS;

        SciaRegs.SCICTL1.bit.SWRESET = 1;   // Salimos del reset

}
// ENVIA UN CARACTER POR SCI
void scia_xmit(int a){
    while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {} // Espera hasta que se reciba el dato
    SciaRegs.SCITXBUF=a;

}
```

```
void scia_msg(char * msg){ // La funcion envia caracter por caracter del mensaje
almacenado hasta el valor NULL o 0\//
    int i;
    i = 0;
    while(msg[i] != '\0')
    {
        scia_xmit(msg[i]);
        i++;
    }
}

//     INICIALIZACIÓN DE INTERRUPCIONES
/**
  @brief Función de inicialización las interrupciones.

  Se habilitan e inicializan todas las interrupciones necesarias.
*/
void InitInt(void){
        //Enable Peripheral, global Ints and higher priority real-time debug events:
       IER |= (M_INT1|M_INT2);
       IER |= (M_INT3|M_INT4);
       IER |= M_INT8;
       IER |= M_INT9; // Esta deja Bloqueado al HW...
       // Activar interrupción sci fifo rx
       PieCtrlRegs.PIEIER9.bit.INTx1 = 1;   // RX SCIA
       PieCtrlRegs.PIEIER1.bit.INTx7 = 1;   // Activar Interrupción T0

       // Enable the PIE
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
       // Enables PIE to drive a pulse into the CPU
       PieCtrlRegs.PIEACK.all = 0xFFFF;
       // Enable Interrupts at the CPU level
    EINT;   // Enable Global interrupt INTM
       ERTM;   // Enable Global realtime interrupt DBGM
}
```

8.2 Script para la obtención de la matriz Jacobiana

```
N = input('Ingrese el cuantos GDL tiene el brazo robot: ');

%Variables
syms a1 a2 a3 a4 a5 a6 d1 d2 d3 d4 d5 d6 al1 al2 al3 al4 al5 al6 th1 th2...
    th3 th4 th5 th6;
%Valores a las variables que son calculadas
a1=0; a4=0; a5=0; al1=-pi/2; al2=0; al3=0; al4=-pi/2; al5=0; d2=0; d3=0;
d4=0; th6=th5;

%DH ejemplo del RV-2AJ
T=[a1 al1 d1 th1;a2 al2 d2 th2;a3 al3 d3 th3;a4 al4 d4 th4;a5 al5 d5 th5;a6 al6 d6
th6];

A13 = eye(4);
for i=1:3
    A=[cos(T(i,4)) -cos(T(i,2))*sin(T(i,4)) sin(T(i,2))*sin(T(i,4))
T(i,1)*cos(T(i,4));...
          sin(T(i,4)) cos(T(i,2))*cos(T(i,4)) -sin(T(i,2))*cos(T(i,4))
T(i,1)*sin(T(i,4));...
          0 sin(T(i,2)) cos(T(i,2)) T(i,3);...
          0 0 0 1];
    A13 = A13*A;
end
Ai = eye(4);
```

```
for i=1:N
    A=[cos(T(i,4)) -cos(T(i,2))*sin(T(i,4)) sin(T(i,2))*sin(T(i,4))
T(i,1)*cos(T(i,4));...
          sin(T(i,4)) cos(T(i,2))*cos(T(i,4)) -sin(T(i,2))*cos(T(i,4))
T(i,1)*sin(T(i,4));...
          0 sin(T(i,2)) cos(T(i,2)) T(i,3);...
          0 0 0 1];
    Ai = Ai*A;
end
fprintf('Matriz Homogenea tres primeras articulaciones: \n');
display(A13);
fprintf('Matriz Homogenea: \n');
display(Ai);

%Extraer las matrices homogeneas de cada articulacion
for i=1:N
    A=[cos(T(i,4)) -cos(T(i,2))*sin(T(i,4)) sin(T(i,2))*sin(T(i,4))
T(i,1)*cos(T(i,4));...
          sin(T(i,4)) cos(T(i,2))*cos(T(i,4)) -sin(T(i,2))*cos(T(i,4))
T(i,1)*sin(T(i,4));...
          0 sin(T(i,2)) cos(T(i,2)) T(i,3);...
          0 0 0 1];
    if i==1 A1 = A;
    end
    if i==2 A2 = A;
    end
    if i==3 A3 = A;
    end
    if i==4 A4 = A;
    end
    if i==5 A5 = A;
    end
    if i==6 A6 = A;
    end
end
if N==6
    T01 = eye(4)*A1;
    T02 = simplify(T01*A2);
    T03 = simplify(T02*A3);
    T04 = simplify(T03*A4);
    T05 = simplify(T04*A5);
    T06 = simplify(T05*A6);

    %Se obtiene los vectores de posicion
    a01v=T01(1:3,4); a02v=T02(1:3,4); a03v=T03(1:3,4); a04v=T04(1:3,4);
    a05v=T05(1:3,4);a06v=T06(1:3,4);

    %Para formar matrices de rotación desde las matrices de transformacion
    z0=[0;0;1]; z1=(T01(1:3,3)); z2=(T02(1:3,3));
    z3=(T03(1:3,3)); z4=(T04(1:3,3)); z5=(T05(1:3,3));
end
if N==5
    T01 = eye(4)*A1;
    T02 = simplify(T01*A2);
    T03 = simplify(T02*A3);
    T04 = simplify(T03*A4);
    T05 = simplify(T04*A5);

    %Se obtiene los vectores de posicion
    a01v=T01(1:3,4); a02v=T02(1:3,4); a03v=T03(1:3,4); a04v=T04(1:3,4);
    a05v=T05(1:3,4);
    %Para formar matrices de rotación a partir de las matrices de transformacion
    z0=[0;0;1]; z1=(T01(1:3,3)); z2=(T02(1:3,3));
    z3=(T03(1:3,3)); z4=(T04(1:3,3)); z5=(T05(1:3,3));
```

```
%Matriz Jacobiana
    %Derivadas parciales Jacobiano
    %Metodo analitico
    J1 = simplify(jacobian(a05v,th1));
    J2 = simplify(jacobian(a05v,th2));
    J3 = simplify(jacobian(a05v,th3));
    J4 = simplify(jacobian(a05v,th4));
    J5 = simplify(jacobian(a05v,th5));
    J = [J1 J2 J3 J4 J5;z0 z1 z2 z3 z4];

    %Metodo Geometrico
    J1=simplify(cross(z0,(a05v-[0;0;0])));
    J2=simplify(cross(z1,(a05v-a01v)));
    J3=simplify(cross(z2,(a05v-a02v)));
    J4=simplify(cross(z3,(a05v-a03v)));
    J5=simplify(cross(z4,(a05v-a04v)));
    %J6=simplify(cross(z5,(a06v-a05v)));
    Jg = [J1 J2 J3 J4 J5;z0 z1 z2 z3 z4];

fprintf('Matriz Jacobiana por derivadas parciales: ');
display(J);
fprintf('Matriz Jacobiana por metodo geometrico: ');
display(Jg);
end

%Determinante de la Matriz Jacobiana para la obtencion de la singularidad
Jdet = simplify(det(J(1:5,1:5)));
fprintf('Determinante de la Matriz Jacobiana: ');
display(Jdet);
Jdet = simplify(det(Jg(1:5,1:5)));
fprintf('Determinante de la Matriz Jacobiana: ');
display(Jdet);
```