

DESARROLLO DE UN APLICATIVO PARA LA PROGRAMACIÓN DE
TRAYECTORIAS DEL ROBOT TIPO DELTA DE LA UNIVERSIDAD SANTO
TOMÁS.

CRISTIAN ALEJANDRO MONTENEGRO AGUIRRE

UNIVERSIDAD SANTO TOMÁS
INGENIERÍA MECÁNICA
BOGOTÁ. D.C.
2021

DESARROLLO DE UN APLICATIVO PARA LA PROGRAMACIÓN DE
TRAYECTORIAS DEL ROBOT TIPO DELTA DE LA UNIVERSIDAD SANTO
TOMÁS.

CRISTIAN ALEJANDRO MONTENEGRO AGUIRRE

Trabajo de grado

DIRECTOR: ING. JORGE ANDRES GARCIA BARBOSA

UNIVERSIDAD SANTO TOMÁS
INGENIERÍA MECÁNICA
BOGOTÁ D.C
2021

Nota de Aceptación

Presidente del Jurado

Jurado

Jurado

Bogotá 04-11-2021

DEDICATORIA:

A familia:

Que me han brindado la vida, y su presencia me ayudó a crear la persona que hoy soy. La cual me ha apoyado a lo largo de mi vida y de mi formación con todo su esfuerzo y corazón a pesar de las adversidades.

A mi pareja:

Que me ha brindado un apoyo incondicional a lo largo de mi carrera y durante el tiempo de creación del proyecto.

A mis compañeros:

Gracias a ellos mi experiencia en esta etapa académica fue grata a pesar de los buenos y malos momentos que llegamos a vivir con lo que algunos llegaron a ser más que compañeros y se convirtieron en amigos y futuros colegas.

A mis amigos:

Ya que me dieron ánimo y apoyo desde incluso antes de entrar a la Universidad y aún siguen estando a mi lado a pesar de los años que pasan.

A mis profesores:

Quienes me guiaron y enseñaron muchas cosas a lo largo de mi formación académica, tanto profesional como personal.

Gracias

AGRADECIMIENTOS

A la universidad Santo Tomás la cual me abrió sus puertas, dándome la oportunidad de formarme como ingeniero mecánico y ser una mejor persona.

Al Ingeniero Jorge Andrés García, por su guía, su paciencia e interés y apoyo desde el inicio del proyecto.

Al ingeniero Ubaldo García de XMART PLM, por sus recomendación y explicaciones sobre la Cinemática Inversa.

Al ingeniero Francisco Pinto de XMART PLM, por sus recomendaciones y ayuda en la creación del código.

Y a todas las personas que me colaboraron en el desarrollo de éste proyecto.

Gracias

CONTENIDO

0. RESUMEN	11
1. INTRODUCCIÓN	12
2. OBJETIVOS.....	13
2.1 OBJETIVO GENERAL.....	13
2.2 OBJETIVOS ESPECÍFICOS.....	13
3. REQUERIMIENTOS Y VARIABLES DE UN ROBOT DELTA.....	14
3.1 Dimensiones Robot delta.....	14
3.2 Volumen útil de trabajo.....	15
3.3 Controlador y servomotores.....	16
3.3.1 Controlador.....	16
3.3.2 Buffer.....	17
3.3.3 PVT	18
3.3.4 Servomotores	18
3.4 Cinemática directa.....	19
3.4.1 Singularidades o indeterminaciones	22
3.4.1.1 Caso 1	22
3.4.1.2 Caso 2	23
3.4.1.3 Caso 3	23
3.4.1.4 Caso 4	23
3.4.1.5 Caso 5	24
3.5 Interpolación lineal entre 2 puntos	24
3.6 Perfil de suavizado.	25
3.7 Cinemática Inversa.....	27
3.7.1 Caso 1	27
3.7.2 Caso 2	28
3.7.3 Caso 3	29
3.8 Sistema de variables de entrada y salida.....	31
4. PROGRAMACIÓN DE ALGORITMOS BASES PARA EL APLICATIVO DE COMUNICACIÓN ENTRE VISUAL STUDIO Y GALIL DMC 40X3	32
4.1. Programa y lenguaje del aplicativo	32
4.1.1 Conceptos utilizados.....	33

4.4 Programación Cinemática directa.....	34
4.2 Programación Interpolación lineal.....	35
4.3 Programación Perfiles Suavizados	37
4.5 Programación Cinemática Inversa.....	38
4.6 Comunicación controlador DMC	39
4.2 Programación interfaz.....	41
5. PRUEBAS DEL APLICATIVO	43
5.1 Trayectoria lineal A – B en el espacio.....	43
6. CONCLUSIONES	57
7. RECOMENDACIONES	58
BIBLIOGRAFÍA.....	59
ANEXO A: SOLUCIÓN CINEMÁTICA DIRECTA	60
A.1 Singularidades	63
ANEXO B: PERFILES SUAVIZADOS	66
ANEXO C: CÓDIGO INTERFAZ	67
ANEXO D: CÓDIGO CINEMÁTICA DIRECTA	86
ANEXO E: CÓDIGO CINEMÁTICA INVERSA	92
ANEXO F: CÓDIGO INTERPOLACIÓN Y PERFIL SUAVIZADO	97
ANEXO G: OTRAS TRAYECTORIAS.....	101
G.1 Trayectoria lineal A – B vertical.....	101
G.2 Trayectoria lineal A – B horizontal.....	107

LISTA DE TABLAS

Tabla 1.	Longitudes conocidas del robot delta de la Universidad Santo Tomás.	14
Tabla 2.	Valores establecidos del volumen de control del robot delta de la Universidad Santo Tomás	16
Tabla 3.	Variables de salida del sistema.	18
Tabla 4.	Especificaciones servomotores	19
Tabla 5.	Variables Cinemática Directa	24
Tabla 6.	Nomenclatura utilizada en la interpolación.	25
Tabla 7.	Valores obtenidos de la Cinemática Inversa.	30
Tabla 8.	Desplazamiento Angular del motor 1	49
Tabla 9.	Desplazamiento Angular del motor 2	50
Tabla 10.	Desplazamiento Angular del motor 3	50
Tabla A.	Desplazamiento Angular del motor 3	102
Tabla B.	Desplazamiento Angular del motor 1	109

LISTA DE FIGURAS

<i>Figura 1.</i>	Esquema del manipulador de tres grados de libertad	14
<i>Figura 2.</i>	Volumen de trabajo de robots tipo delta.	15
<i>Figura 3.</i>	Volumen de trabajo del robot tipo delta de la Universidad Santo Tomás.	16
<i>Figura 4.</i>	Controlador Galil DMC-4143.	17
<i>Figura 5.</i>	Buffer FIFO.	17
<i>Figura 6.</i>	Puntos PVT.	18
<i>Figura 7.</i>	Servomotores.	19
<i>Figura 8.</i>	Posición angular de los brazos.	20
<i>Figura 9.</i>	Generación de esferas a partir de los brazos.	21
<i>Figura 10.</i>	Punto de intersección de las esferas.....	22
<i>Figura 11.</i>	Movimiento de Punto A a Punto B	24
<i>Figura 12.</i>	Posición, velocidad, aceleración y jeck de un perfil unitario trapezoidal.	26
<i>Figura 13.</i>	Posición, velocidad, aceleración y jeck de un perfil unitario suavizado con la función coseno.....	27
<i>Figura 14.</i>	Consideración geometría para $0 \leq P_{xi} \leq rA$	28
<i>Figura 15.</i>	Consideración geometría para $P_{xi} < 0$	29
<i>Figura 16.</i>	Consideración geometría para $P_{xi} > rA$	30
<i>Figura 17.</i>	Flujo de Variables de entrada y salida.	31
<i>Figura 18.</i>	Diagrama para un movimiento de A a B.....	32
<i>Figura 19.</i>	Programa Visual Studio 2019.	33
<i>Figura 20.</i>	Cinemática directa.	34
<i>Figura 21.</i>	Código de variables (Cinemática Directa).	35
<i>Figura 22.</i>	Interpolación lineal.	35
<i>Figura 23.</i>	Código de variables (Interpolación).....	36
<i>Figura 24.</i>	Perfil Suavizado.	37
<i>Figura 25.</i>	Constantes de integración (Perfil suavizado).	38
<i>Figura 26.</i>	Cinemática inversa.	38
<i>Figura 27.</i>	Código para los valores de posición (Cinemática Inversa).	39
<i>Figura 28.</i>	Clase necesaria para el funcionamiento de las librerías.	40
<i>Figura 29.</i>	Administrador de referencias Visual Studio.....	40
<i>Figura 30.</i>	Funcionamiento de la interfaz de usuario.....	41
<i>Figura 31.</i>	Interfaz gráfica de usuario.....	42
<i>Figura 32.</i>	Representación Trayectoria lineal A – B en el espacio.	43

<i>Figura 33.</i>	Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición -200,-200,-600.....	44
<i>Figura 34.</i>	Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 200, 200,-850	44
<i>Figura 35.</i>	Posición A y posición B de la trayectoria.....	45
<i>Figura 36.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 1.	45
<i>Figura 37.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 2.	46
<i>Figura 38.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 3.	47
<i>Figura 39.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 1.....	48
<i>Figura 40.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 2.....	48
<i>Figura 41.</i>	Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 3.....	49
<i>Figura 42.</i>	Posición Y vs tiempo de desplazamiento en trayectoria 3.....	51
<i>Figura 43.</i>	Velocidad Y vs tiempo de desplazamiento en trayectoria 3.....	52
<i>Figura 44.</i>	Aceleración Y vs tiempo de desplazamiento en trayectoria 3.....	52
<i>Figura 45.</i>	Jerk Y vs tiempo de desplazamiento en trayectoria 3	53
<i>Figura 46.</i>	Posición angular motor 2 vs tiempo de desplazamiento en trayectoria 3	53
<i>Figura 47.</i>	Velocidad angular motor 2 vs tiempo de desplazamiento en trayectoria 3	54
<i>Figura 48.</i>	Aceleración motor 2 vs tiempo de desplazamiento en trayectoria 3	55
<i>Figura 49.</i>	Jerk motor 2 vs tiempo de desplazamiento en trayectoria 3.....	55
<i>Figura A.</i>	Valores constantes y variables conocidas.....	62
<i>Figura B.</i>	Código utilizado para llegar a la solución en <i>Wolfram Mathematica</i>	63
<i>Figura C.</i>	Solución del código ingresado en <i>Wolfram Mathematica</i>	63
<i>Figura D.</i>	Análisis geométrico caso 1 de Cinemática directa	63
<i>Figura E.</i>	Análisis geométrico caso 2 de Cinemática directa	64
<i>Figura F.</i>	Análisis geométrico caso 3 de Cinemática directa	65
<i>Figura G.</i>	Representación Trayectoria 1: Trayectoria lineal A – B vertical.....	101
<i>Figura H.</i>	Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 0, 0, -850.....	101
<i>Figura I.</i>	Comparación posición A y B Motor 3.....	102
<i>Figura J.</i>	Posición Z vs tiempo de desplazamiento en trayectoria 1.....	103
<i>Figura K.</i>	Velocidad Z vs tiempo de desplazamiento en trayectoria 1	103
<i>Figura L.</i>	Aceleración Z vs tiempo de desplazamiento en trayectoria 1	104

Figura M.	Jerk Z vs tiempo de desplazamiento en trayectoria 1.....	104
Figura N.	Posición angular motor 1 vs tiempo de desplazamiento en trayectoria 1	105
Figura O.	Velocidad angular motor 1 vs tiempo de desplazamiento en trayectoria 1 ...	105
Figura P.	Aceleración motor 1 vs tiempo de desplazamiento en trayectoria 1	106
Figura Q.	Jerk motor 1 vs tiempo de desplazamiento en trayectoria 1	106
Figura R.	Representación Trayectoria 2: Trayectoria lineal A – B horizontal.	107
Figura S.	Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición -564,0,-800.....	107
Figura T.	Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 564,0,-800.....	108
Figura U.	Comparación de la posición antes y después del movimiento de la Trayectoria 2 Trayectoria lineal A – B horizontal del motor 1.	108
Figura V.	Posición X vs tiempo de desplazamiento en trayectoria 2.....	109
Figura W.	Velocidad X vs tiempo de desplazamiento en trayectoria 2.....	110
Figura X.	Aceleración X vs tiempo de desplazamiento en trayectoria 2.....	110
Figura Y.	Jerk X vs tiempo de desplazamiento en trayectoria 2	111
Figura Z.	Posición angular motor 3 vs tiempo de desplazamiento en trayectoria 2	111
Figura AA.	Velocidad angular motor 3 vs tiempo de desplazamiento en trayectoria 2 ...	112
Figura BB.	Aceleración motor 3 vs tiempo de desplazamiento en trayectoria 2	112
Figura CC.	Jerk motor 3 vs tiempo de desplazamiento en trayectoria 2.....	113

LISTA DE ANEXOS

ANEXO A: SOLUCIÓN CINEMÁTICA DIRECTA	60
ANEXO B: PERFILES SUAVIZADOS	66
ANEXO C: CÓDIGO INTERFAZ	67
ANEXO D: CÓDIGO CINEMÁTICA DIRECTA	86
ANEXO E: CÓDIGO CINEMÁTICA INVERSA	92
ANEXO F: CÓDIGO INTERPOLACIÓN Y PERFIL SUAVIZADO	97
ANEXO G: OTRAS TRAYECTORIAS	101

0. RESUMEN

Un robot de arquitectura paralela tipo delta está conformado por tres cadenas cinemáticas que se unen en un efector final. Este tipo de disposición otorga al robot la capacidad de moverse a altas velocidades y aceleraciones, siendo utilizado principalmente en operaciones de pick and place. Para que el efector final se desplace en una línea recta dentro del volumen de trabajo, es necesario realizar transformaciones cinemáticas al dominio de los actuadores, en este caso rotacionales. En la generación de estas trayectorias es necesario el uso de la cinemática directa e inversa aplicando perfiles suavizados de velocidad. La nube de puntos obtenida mediante un código desarrollado en C# es enviada a un controlador galil DMC 4143. La trayectoria calculada es transmitida a los servomotores para la generación de movimientos en las extremidades del robot que convergen al efector final, en el cual se encuentra el punto de análisis cinemático. A fin de verificar la generación de trayectorias lineales con perfiles suavizados de velocidad por el código, se implementa un comando de consola el cual consulta las variables internas del aplicativo y las muestra en forma de lista, de este modo obtenemos los datos de posición y tiempo en el punto a analizar. Al graficar los puntos obtenidos se comprueba que la trayectoria tiene periodos de aceleración y desaceleración acordes a lo planteado. Al tener certeza que el código genera trayectorias lineales correctas, se sabe que el robot no tendrá movimientos que comprometan su estructura y buen funcionamiento.

1. INTRODUCCIÓN

En las últimas décadas, la tendencia ha sido la de automatizar de manera exponencial todos los procesos de producción. Esta tendencia seguirá siendo posible gracias al desarrollo de las tecnologías de programación, automatización e inteligencia artificial [1]. La evidencia de lo anterior se ve reflejada en los robots industriales, que han influido de manera directa en la competitividad, productividad, eficiencia y rentabilidad en las empresas [2].

Por lo cual la Facultad de Ingeniería Mecánica de la Universidad Santo Tomás viene desarrollando un robot de arquitectura paralela tipo delta para aplicaciones pick and place, el cual ya cuenta con un prototipo funcional conectado a un control de movimiento Galil DMC-41x3. Pero ya que este cuenta con características dimensionales y cinemáticas propias hacen que sus movimientos no puedan ser programados por sistemas CAM comerciales dado que no hay un lenguaje estandarizado para programar robots, debido a que las empresas y universidades trabajan independientemente en el desarrollo de estos. Por ello aún no ha sido posible adquirir algún aplicativo para realizar los movimientos de dicho prototipo dando como consecuencia el uso nulo del mismo y la pérdida de oportunidades de avanzar por este camino poco explorado en Colombia.

Por esto contar con una herramienta que permita la programación del robot tipo delta de la Universidad Santo Tomás, sin tener conocimientos avanzados sobre procesos de automatización y control sería importante para los estudiantes que se inician en el tema siendo un incentivo el poder utilizar un robot, puesto que no todas las Universidades cuenta con uno, sumado a que con ello se podrían implementar nuevos temas en la línea de automatización y control que tiene la Facultad de Ingeniería Mecánica llegando a ser un motivo para el que apliquen nuevos estudiantes.

Por esta razón desarrollar una interfaz hombre/máquina que permita manipular el control del robot tipo delta sería necesario para aumentar los aportes futuros de estudiantes que decidan utilizar dicho robot para alguna aplicación en la Universidad Santo Tomás.

Sumado a trabajar en la programación, automatización y control de un robot de arquitectura paralela tipo delta como lo es el de la Universidad abre las puertas a explorar y aumentar los conocimientos de estos temas, dando así futuras oportunidades para poder realizar trabajos de esta o mayor magnitud en el área profesional llegando a dar reconocimiento a la Universidad en este campo de la ingeniería.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar una aplicativo para la programación de trayectorias del robot tipo delta de la Universidad Santo Tomás.

2.2 OBJETIVOS ESPECÍFICOS

Definir los requerimientos del aplicativo de control en términos de variables o señales de entrada y salida.

Programar el algoritmo base del aplicativo considerando las variables de entrada y salida sobre una plataforma virtual.

Validar el aplicativo desarrollado mediante pruebas de las trayectorias.

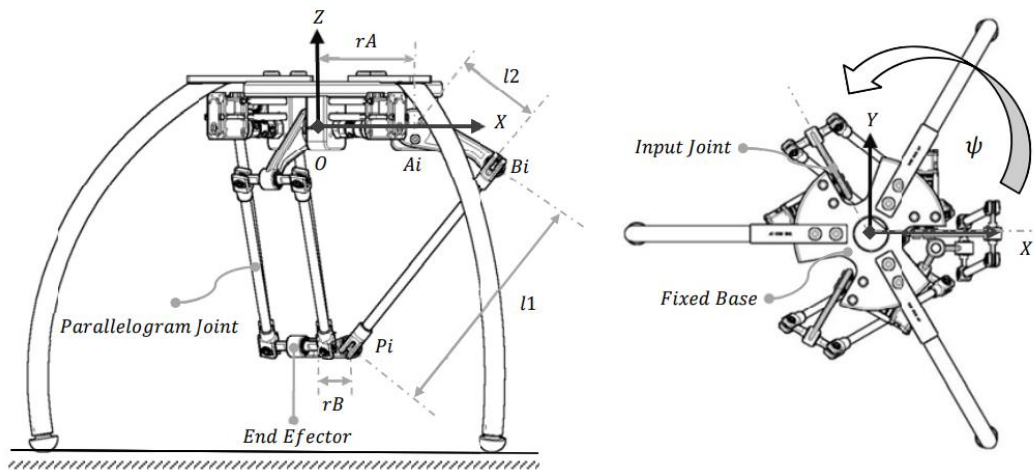
3. REQUERIMIENTOS Y VARIABLES DE UN ROBOT DELTA

Para la generación de trayectorias del robot delta de la universidad Santo Tomás se deben tener en cuenta varios factores antes de empezar el proceso de programación. Principalmente el conocer y tener claras tanto sus variables y constantes de entrada como de salida del sistema, además del conocimiento de los conceptos matemáticos.

3.1 Dimensiones Robot delta.

Las dimensiones del robot delta serán el primer paso para determinar todas las variables y constantes que se utilizarán a lo largo de este proyecto. Al conocerse especificaciones de diseño óptimas determinadas con anterioridad [3] estas se aplicarán como las dimensiones ideales.

Figura 1. Esquema del manipulador de tres grados de libertad



Fuente: [4].

En la Figura 1 se puede apreciar el esquema con el que cuenta el robot, lográndose ver en este sus cotas geométricas las cuales serán fundamentales ya que serán un requisito en el desarrollo del proyecto. En la Tabla 1 se hará un recuento de estas cotas para tener mayor claridad de sus valores al momento de programar los algoritmos.

Tabla 1. Longitudes conocidas del robot delta de la Universidad Santo Tomás.

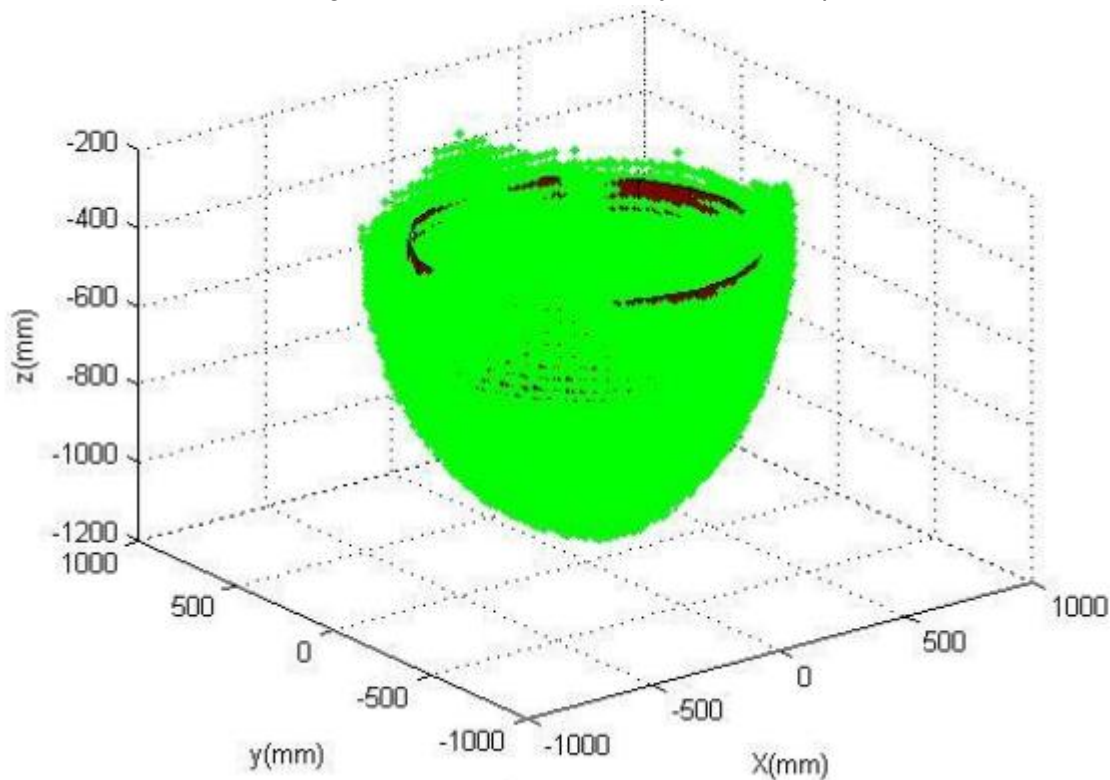
Nomenclatura	Referencia	Valor
L_1	Longitud del paralelogramo	840 mm
L_2	Longitud del brazo	310 mm
rA	Distancia entre el sistema de referencia hasta el eje de los motores.	100 mm
rB	Distancia entre el punto central del efector hasta la unión de los paralelogramos.	50 mm

Fuente: [3] Autor.

3.2 Volumen útil de trabajo.

Al tener las restricciones dimensionales definidas se puede generar el espacio de trabajo con en el cual podrá moverse, para los robots tipo delta el volumen de trabajo que puede alcanzar es bastante amplio como se puede observar en la Figura 2 teniendo una forma de la mitad de una elipse extruida alrededor del eje z.

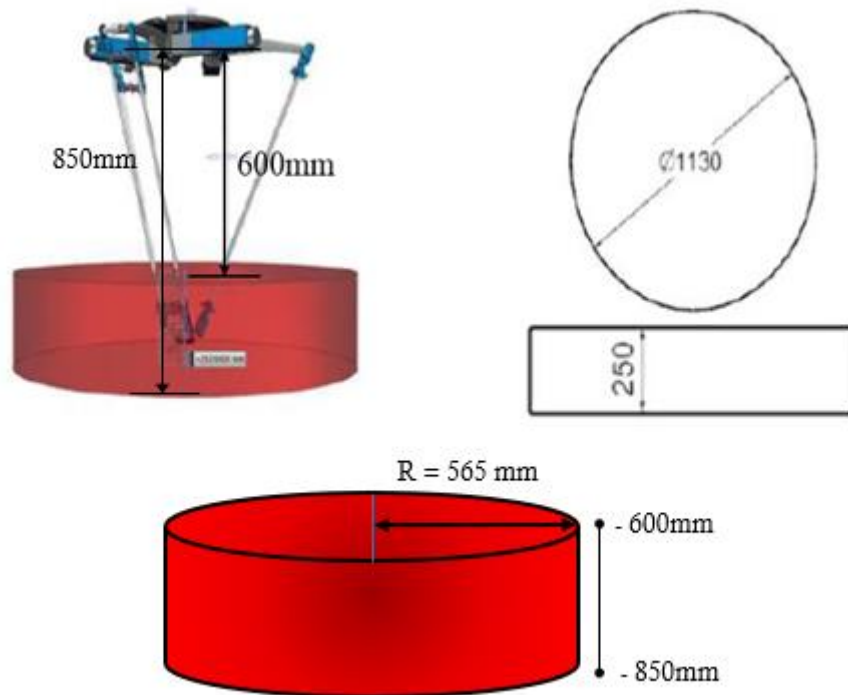
Figura 2. Volumen de trabajo de robots tipo delta.



Fuente: [3].

Al realizarse todas las pruebas de diseño se define un volumen útil que fue delimitado con la geometría de un cilindro, esto se puede ver en la Figura 3, así se asegura que el robot se podrá ubicar en cualquier punto al estar dentro del volumen del movimiento además de que el efector final se puede mover con completa libertad asegurando así el alcanzar su máximo potencial y no genere colisiones con los demás elementos.

Figura 3. Volumen de trabajo del robot tipo delta de la Universidad Santo Tomás.



Fuente: [3].

En la Tabla 2 se puede apreciar los valores utilizados para la definición del volumen útil dentro del código, siendo este otro requisito en la programación de los algoritmos.

Tabla 2. Valores establecidos del volumen de control del robot delta de la Universidad Santo Tomás

Nomenclatura	Referencia	Valor
R_{ctrl}	Radio del volumen de control.	565 mm
A_{ctrl}	Altura del volumen de control	250 mm

Fuente: [3] Autor.

3.3 Controlador y servomotores

3.3.1 Controlador

El robot está conectado a un controlador de movimiento DMC-41x3 es uno de los controladores de movimiento independientes de última generación de Galil este se puede ver en la Figura 4. El cual está disponible como un controlador de movimiento a nivel de caja o tarjeta. La unidad funciona de forma independiente o se conecta a una PC con Ethernet o USB. Incluye entradas y salidas analógicas. El DMC-41x3 está disponible en formatos de uno a ocho ejes, y cada eje es configurable por el usuario para la operación paso a paso o servomotor. Los modos de movimiento incluyen

posicionamiento punto a punto, seguimiento de posición, jogging, interpolación lineal y circular, contorno, engranaje electrónico y ecam.

Figura 4. Controlador Galil DMC-4143.

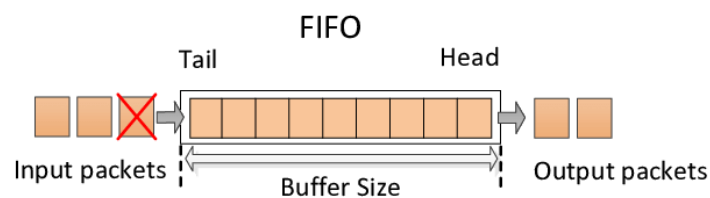


Fuente: Autor.

3.3.2 Buffer

El controlador cuenta con un buffer FIFO ("First In - First Out", "Primero en entrar - primero en salir") el cual es una memoria en la que se almacenan datos temporalmente como se puede ver en la Figura 5, su principal uso es para evitar que el controlador se quede sin datos durante una transferencia (entrada/salida) de datos irregular o por la velocidad del proceso.

Figura 5. Buffer FIFO.



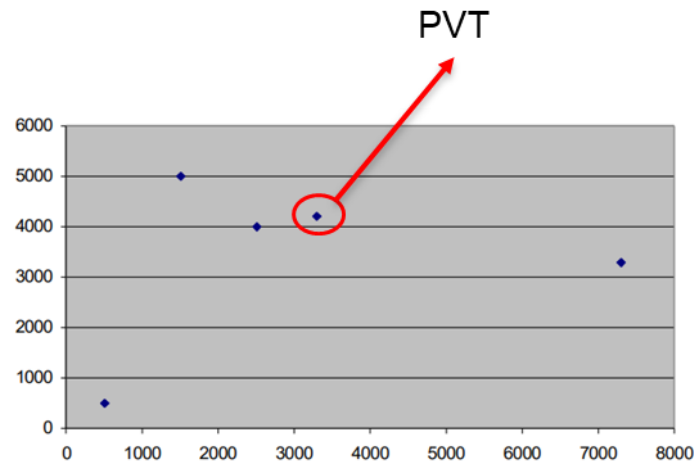
Fuente: [10] Autor.

También es importante resaltar que el espacio con el que cuenta el buffer es bastante grande, aunque por motivos prácticos se utilizó uno de 255 segmentos para todos los puntos PVT de los 3 ejes.

3.3.3 PVT

Los controladores DMC-41x3 ahora admiten un modo de movimiento denominado "PVT" ("Posición, Velocidad, Tiempo"). Los perfiles se definirán por posición, velocidad y tiempo individualmente en los ejes como se puede apreciar en la Figura 6. Este movimiento está diseñado para sistemas donde la carga debe atravesar una serie de coordenadas sin discontinuidades en la velocidad.

Figura 6. Puntos PVT.



Fuente: [10] Autor.

Como se dijo anteriormente los segmentos PVT deben ingresarse un eje a la vez mediante la distancia a la que se moverá en cts, la velocidad que se obtendrá en cts / segundo y el período de tiempo especificado en samples (1024 samples = 1 segundo). Con lo cual estos valores serán los que se generarán a lo largo del proyecto es decir nuestras variables de salida del sistema y se representarán de la siguiente forma como se puede apreciar en la Tabla 3.

Tabla 3. Variables de salida del sistema.

Nomenclatura	Referencia	Valor
P	Posición angular del motor	Variable
V	Velocidad del motor	Variable
T	Tiempo	Variable

Fuente: Autor.

3.3.4 Servomotores

Se cuenta con 3 servomotores lo cuales se pueden ver en la Figura 7, estos funcionan por medio de modulación de ancho de pulso (PWM) generados por sus encoder. Todos los servos disponen de tres cables, dos para alimentación Vcc y Gnd (4.8 a 6 [V]) y un tercero para aplicar el tren de pulsos de control, que hace que el circuito de

control diferencial interno ponga el servo en la posición indicada. Un encoder es un dispositivo de detección que proporciona una respuesta. Los Encoder convierten el movimiento en una señal eléctrica que puede ser leída por algún tipo de dispositivo de control en un sistema de control de movimiento.

Figura 7. Servomotores.



Fuente: Autor.

Las características relevantes de los servomotores serán las que se podrán ver en la Tabla 4.

Tabla 4. Especificaciones servomotores

Marca	Magmotor	-
Serial	CP0000003	-
Fabricación	Estados Unidos	-
Potencia	750	Watts
Frecuencia	50/60	Hz
Encoder	4000	cts

Fuente: Autor.

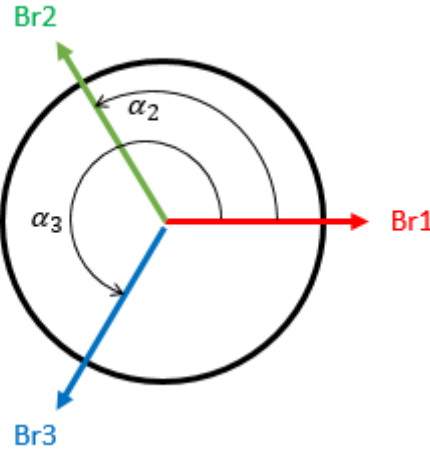
Los servomotores llevan consigo reductores de velocidad con una relación de 25 a 1. Teniendo en cuenta lo anterior para que los servomotores realicen un giro de 360° se le debe suministrar una señal de 100000 cts.

3.4 Cinemática directa.

Para la cinemática directa la posición de los ángulos de los motores (q_1, q_2, q_3) ya está determinada gracias a unos encoders los cuales están conectados al controlador y el problema está en encontrar la posición del efector final.

Para esto se inicia creando una matriz de rotación de los brazos.

Figura 8. Posición angular de los brazos.



Fuente: Autor.

Por lo cual al tener en cuenta que $\alpha_1 = 0^\circ$, $\alpha_2 = 120^\circ$ y $\alpha_3 = 240^\circ$ son los ángulos en los que están ubicados cada brazo respectivamente como se puede ver en la Figura 8 se procede a generar la matriz de rotación en el eje Z (Rot_{zi}) obteniendo lo siguiente:

$$Rot_{zi} = \begin{vmatrix} \cos \alpha_i & -\sin \alpha_i & 0 \\ \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (1).$$

Seguido a esto se genera un vector que describe la posición de cada uno de los brazos (P_{bi}).

$$P_{bi} = [R + l_a \cos q_i \quad 0 \quad l_a \sin q_i] \quad (2).$$

Con lo cual al generar una multiplicación entre la matriz Rot_{zi} y el vector P_{bi} se creará una matriz P_b la cual describe la posición respecto a la base fija de los brazos

$$Rot_{zi} = \begin{vmatrix} (\cos \alpha_1)(R + l_a \cos q_1) & -(\cos \alpha_1)(R + l_a \cos q_1)(\sin \alpha_1) & -l_a \sin q_1 \\ (\cos \alpha_2)(R + l_a \cos q_2) & -(\cos \alpha_2)(R + l_a \cos q_2)(\sin \alpha_2) & -l_a \sin q_2 \\ (\cos \alpha_3)(R + l_a \cos q_3) & -(\cos \alpha_3)(R + l_a \cos q_3)(\sin \alpha_3) & -l_a \sin q_3 \end{vmatrix} \quad (3).$$

Teniendo esto se pasa a encontrar el punto (X, Y, Z) del efector final, para esto se representan 3 esferas teniendo como centro la articulación entre el brazo y el paralelogramo con un radio igual a la longitud de L_1 como se puede notar en la Figura 9.

Figura 9. Generación de esferas a partir de los brazos.



Fuente: Autor.

Una vez se tiene presente esto se puede generar un sistema de ecuaciones que representan estas tres esferas (c_1, r_1) , (c_2, r_2) y (c_3, r_3) , con lo cual los c_i se podrán representar como un vector del centro de cada una de estas siendo $c_1 = [x_1, y_1, z_1]$, $c_2 = [x_2, y_2, z_2]$, $c_3 = [x_3, y_3, z_3]$ los radios r_1, r_2, r_3 son valores conocidos, esto nos lleva a lo siguiente:

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 &= r_3^2 \end{aligned} \quad (4).$$

Aplicando la geometría y particularidades a este sistema de ecuaciones se obtiene

$$\begin{aligned} (x - x_1)^2 + (y)^2 + (z - z_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 &= r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 &= r_3^2 \end{aligned} \quad (5).$$

De este sistema de ecuaciones las incógnitas son x, y, z , las cuales serán los puntos de intersección.

Al solucionar este sistema de ecuaciones se llega a lo siguiente:

$$z_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (6).$$

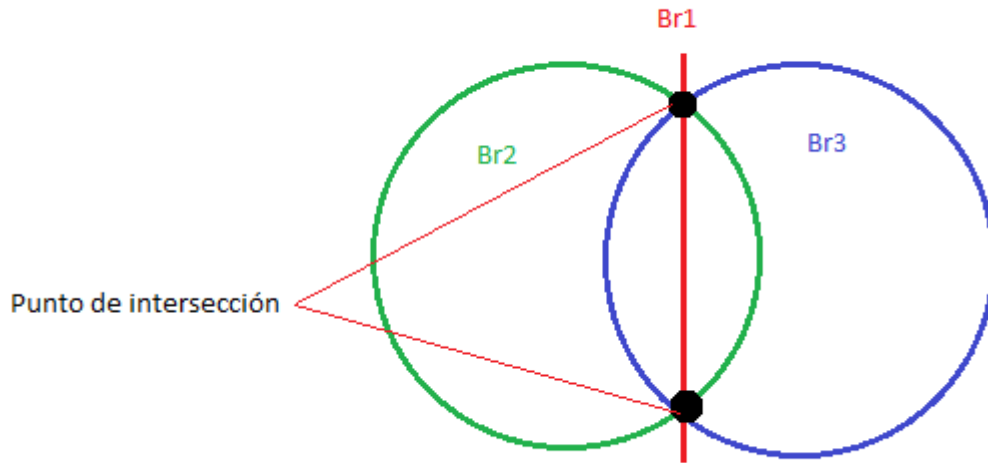
$$y_{\pm} = a_6 z_{\pm} + a_7 \quad (7).$$

$$x_{\pm} = a_4 z_{\pm} + a_5 \quad (8).$$

En donde las variables y solución para llegar a las ecuaciones (6), (7) y (8) se pueden encontrar en el ANEXO A: SOLUCIÓN CINEMÁTICA DIRECTA.

Debido a que la solución es un sistema de ecuaciones cuadrático se llega a dos soluciones, una positiva y otra negativa que corresponden a los puntos de intersección de las 3 esferas como se puede apreciar en la Figura 10. Teniendo en cuenta la forma geométrica del robot el punto de intersección elegido será el negativo.

Figura 10. Punto de intersección de las esferas.



Fuente: Autor.

3.4.1 Singularidades o indeterminaciones

Ahora como se dijo con anterioridad los componentes del sistema de ecuaciones es cuadrático, con lo cual al momento de su solución se indetermina dependiendo del valor que arrojen los encoders, esto puede cambiar en su totalidad el procedimiento de la solución creando así ciertas singularidades. Se analizaron 5 casos distintos en donde dependiendo de esto se aplicó una serie de soluciones prácticas las cuales se pueden encontrar en el ANEXO A: SOLUCIÓN CINEMÁTICA DIRECTA.

3.4.1.1 Caso 1

El primer caso en el que se encontrará una indeterminación es cuando los 3 valores arrojados por los encoders son 0, es decir cuando $q_1 = 0$ & $q_2 = 0$ & $q_3 = 0$.

Las ecuaciones que se utilizaron son

$$z_{\pm} = -1 * (\sqrt{(L_1)^2 - (Cat)^2}) \quad (9).$$

$$y_{\pm} = 0 \quad (10).$$

$$x_{\pm} = 0 \quad (11).$$

3.4.1.2 Caso 2

El segundo caso en el que se encontró una indeterminación es cuando los 3 valores arrojados por los encoders son mayores a 0 e iguales, es decir cuando $q_1 = q_2 = q_3$ y $q_1 > 0 \& q_2 > 0 \& q_3 > 0$

Las ecuaciones que se utilizaron son

$$z_{\pm} = -1 * (z_1 - z_2) \quad (12).$$

$$y_{\pm} = 0 \quad (13).$$

$$x_{\pm} = 0 \quad (14).$$

3.4.1.3 Caso 3

El tercer caso en el que se encontró una indeterminación es cuando los 3 valores arrojados por los encoders son menores a 0 e iguales, es decir cuando $q_1 = q_2 = q_3$ y $q_1 < 0 \& q_2 < 0 \& q_3 < 0$

Las ecuaciones que se utilizaron son

$$z_{\pm} = -1 * (z_1 + z_2) \quad (15).$$

$$y_{\pm} = 0 \quad (16).$$

$$x_{\pm} = 0 \quad (17).$$

3.4.1.4 Caso 4

El cuarto caso en el que se encontró una indeterminación es cuando los valores arrojados por el motor 2 y 3 son iguales, es decir $q_2 = q_3$

Las ecuaciones que se utilizaron son

$$z_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (18).$$

$$y_{\pm} = a_6 z_{\pm} + a_7 \quad (19).$$

$$x_{\pm} = a_4 z_{\pm} + a_5 \quad (20).$$

3.4.1.5 Caso 5

Este caso es cuando los valores obtenidos de los 3 motores son distintos, con lo cual se aplican las ecuaciones que se realizaron inicialmente, es decir $q_1 \neq q_2 \neq q_3$

Con esto finalmente se concretan las variables que nos da la cinemática directa las cuales estarán vinculadas a la posición del efector final y el valor angular de los motores, estas se podrán apreciar en la Tabla 5.

Tabla 5. Variables Cinemática Directa

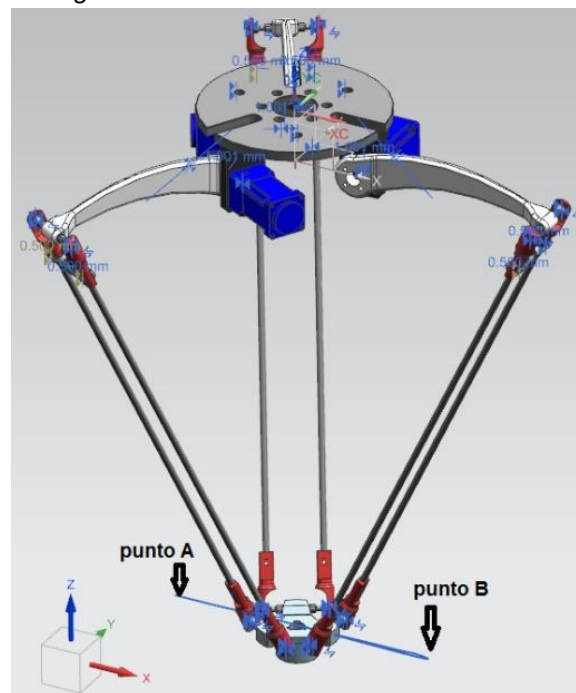
Nomenclatura	Referencia	Valor
x	Coordenada x del centro del efector final.	Variable
y	Coordenada y del centro del efector final.	Variable
z	Coordenada z del centro del efector final.	Variable
q_1	Valor angular del motor 1	Variable
q_2	Valor angular del motor 2	Variable
q_3	Valor angular del motor 3	Variable

Fuente: Autor.

3.5 Interpolación lineal entre 2 puntos

Ya que el robot irá de un punto A a un punto B como se puede apreciar en la Figura 11, es necesario determinar ciertas variables que se ven presentes en este proceso.

Figura 11. Movimiento de Punto A a Punto B



Fuente: [11] Autor.

Para esto se tiene en cuenta la posición inicial ($P_A = |x_i \ y_i \ z_i|$) la cual es encontrada por la cinemática directa y a la posición a la que se quiere ir siendo esta digitada por el usuario ($P_B = |x_f \ y_f \ z_f|$). Estas posiciones se definen vectorialmente (X,Y,Z), para así determinar el movimiento vectorial entre estos puntos.

$$V_M = |P_B - P_A| \quad (21).$$

Una vez conseguido este vector de movimiento se pasa a encontrar su magnitud, la cual sería equivalente a la longitud del movimiento.

$$L = \sqrt{(V_{Mx})^2 + (V_{My})^2 + (V_{Mz})^2} \quad (22).$$

Con este valor y el de la velocidad ingresada por el usuario se encontrará el tiempo en el que el robot recorre esta longitud. Ya con estos valores junto a la capacidad del buffer del controlador se pasa a encontrar la cantidad de particiones en la cuales se dividirá la trayectoria esto con la finalidad de un mejor proceso de envío de datos al controlador.

De este proceso se rescatan las siguientes variables y constantes mostradas en la Tabla 6.

Tabla 6. Nomenclatura utilizada en la interpolación.

Nomenclatura	Referencia	Valor
x_i	Coordenada inicial x del centro del efector final.	Variable
y_i	Coordenada inicial y del centro del efector final.	Variable
z_i	Coordenada inicial z del centro del efector final.	Variable
x_f	Coordenada final x del centro del efector final.	Variable
y_f	Coordenada final y del centro del efector final.	Variable
z_f	Coordenada final z del centro del efector final.	Variable
V_{Mx}	Valor del movimiento vectorial en el eje x	Variable
V_{My}	Valor del movimiento vectorial en el eje y	Variable
V_{Mz}	Valor del movimiento vectorial en el eje z	Variable
L	Longitud total del movimiento	Variable
T	Tiempo que tarda el movimiento	Variable
N	Particiones de la longitud	Variable

Fuente: Autor.

3.6 Perfil de suavizado.

En la generación de trayectorias de movimientos de máquinas y robots de precisión y velocidades altas, un perfil trapezoidal de velocidad no es el adecuado ya que este no tiene una transición o comportamiento en cuanto a velocidades y aceleraciones suaves. Con lo cual se aplica a este perfil trapezoidal la función coseno a su velocidad y una integración de sus ecuaciones, se puede encontrar en el ANEXO B: PERFILES

SUAVIZADOS, dando como resultado un sistema de ecuaciones de posición [7] que asegura que tanto sus velocidades, aceleraciones y jerk sean continuas y suaves ya que físicamente sus derivadas correspondientes son estas, consiguiendo que se reduzcan las vibraciones en sistemas mecánicos.

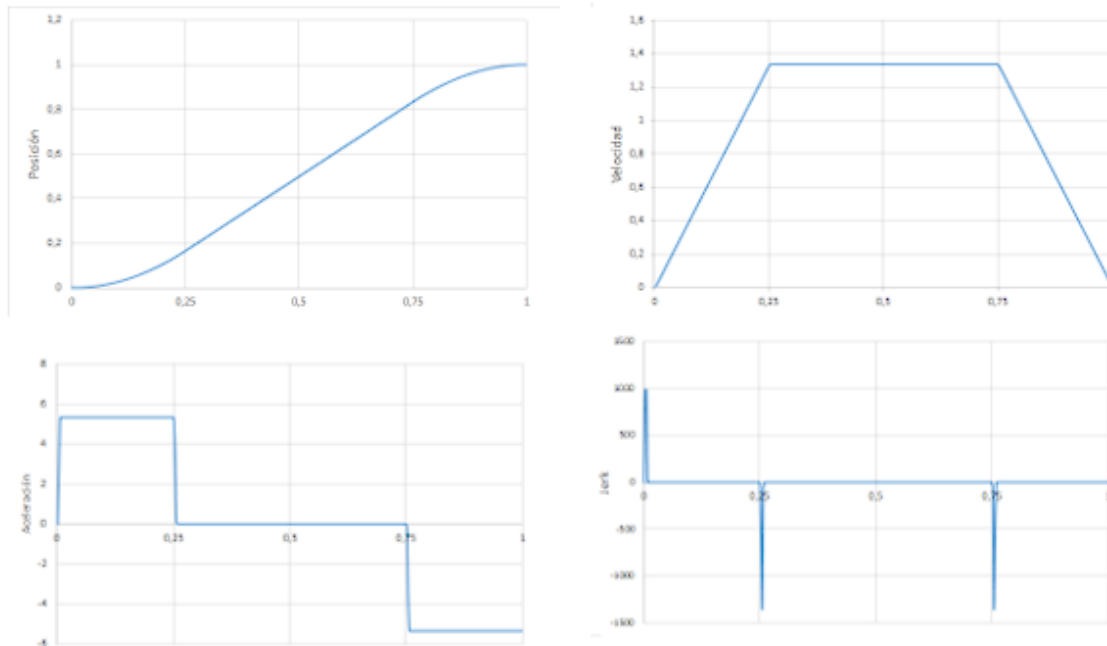
$$x_1 = c_1 * (t - \frac{t_1}{\pi} * \sin(\frac{\pi * t}{t_1})) + k_1 \quad (23).$$

$$x_2 = c_2 * t + k_2 \quad (24).$$

$$x_3 = c_3 * (t + \frac{(t_2 - 1)}{\pi} * \sin \sin(\frac{\pi * (t - t_2)}{t_2 - 1})) + k_3 \quad (25).$$

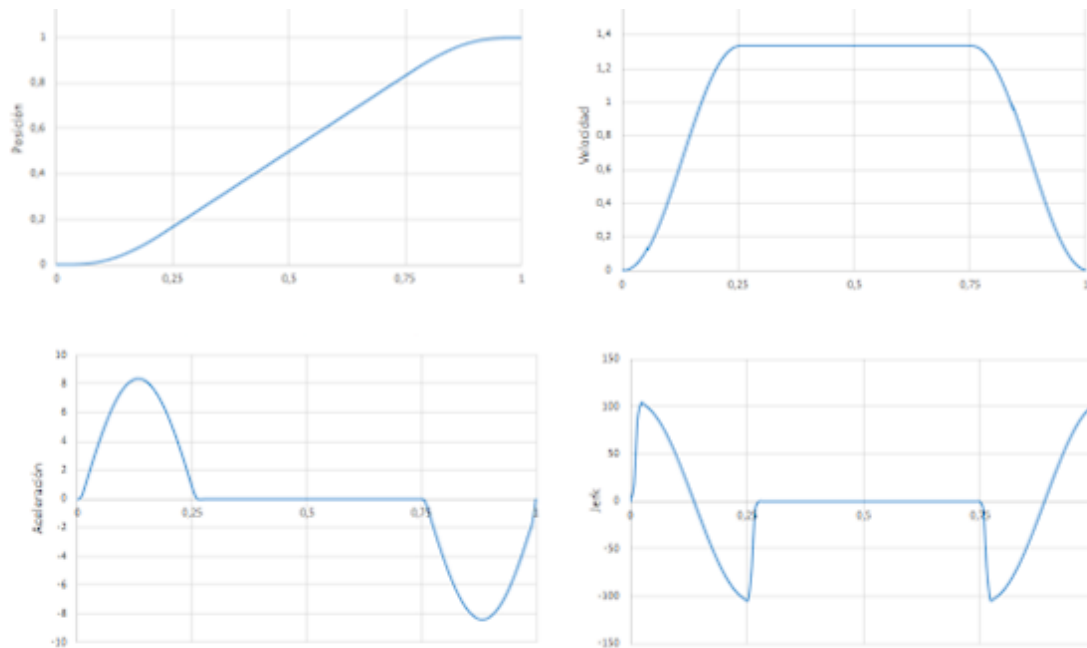
En la Figura 12 se puede apreciar un perfil trapezoidal en donde se muestra su posición, velocidad, aceleración y jerk, en este perfil se ve como los cambios de dirección y magnitud son abruptos. Ahora en la Figura 13 se puede ver un perfil suavizado, en donde se muestra posición, velocidad, aceleración y jerk, de este perfil lo que se resalta es que se logra apreciar las transiciones entre sus etapas de aceleración, velocidad máxima crucero y su desaceleración de una manera muy suave, sin cambios abruptos de magnitud o dirección.

Figura 12. Posición, velocidad, aceleración y jeck de un perfil unitario trapezoidal.



Fuente: Autor.

Figura 13. Posición, velocidad, aceleración y jek de un perfil unitario suavizado con la función coseno.



Fuente: Autor.

3.7 Cinemática Inversa.

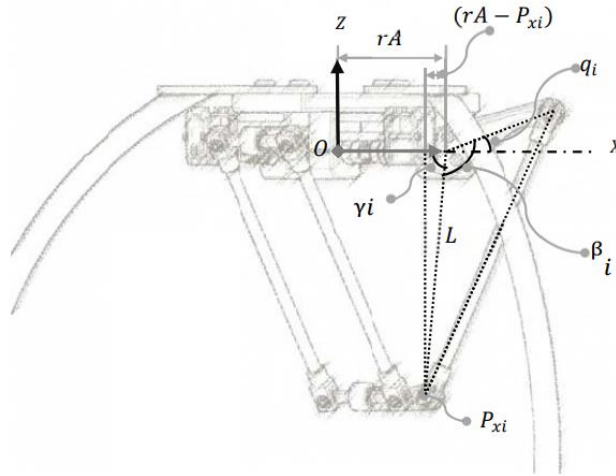
Para la cinemática inversa la posición del efector final está determinada puesto que es la ubicación a la que se quiere ir, el problema a solucionar en esta ocasión es encontrar los ángulos de los motores para llegar al lugar deseado.

Entonces para la solución de la cinemática inversa se tienen en cuentas la geometría del robot para generar las ecuaciones que determinan la relación que tienen los ángulos de los motores con respecto a la posición del efector final, estas están divididas en tres casos debido a que el análisis de cinemática inversa implementado fue desarrollado mediante un método geométrico, por lo cual consta de algunas condiciones o casos específicos [4].

3.7.1 Caso 1

El primer caso ocurre cuando $0 \leq P_{xi} \leq rA$ como se puede ver en la Figura 14.

Figura 14. Consideración geometría para $0 \leq P_{xi} \leq rA$



Fuente: [4]

El ángulo q_i se calcula implementando la siguientes fórmulas:

$$L = \sqrt{(rA - P_{xi})^2 + (P_{zi})^2} \quad (26).$$

$$\gamma = -1 * \tan^{-1}\left(\frac{P_{zi}}{(rA - P_{xi})}\right) \quad (27).$$

$$\theta = \text{Sen}^{-1}\left(\frac{P_{yi}}{l_1}\right) \quad (28).$$

$$\beta = \text{Cos}^{-1}\left(\frac{L^2 + l_2^2 - (l_1 \cos \theta)^2}{2Ll_2}\right) \quad (29).$$

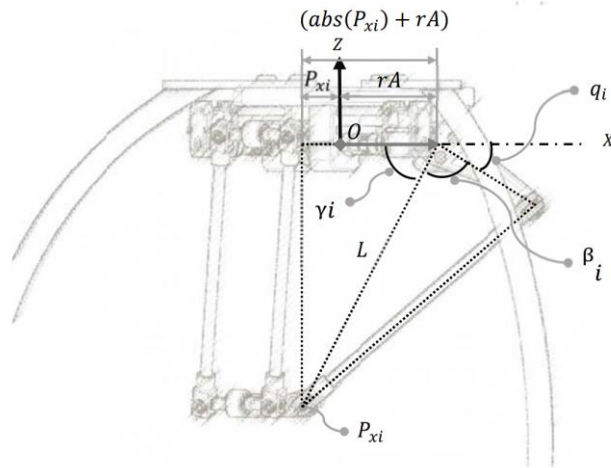
$$q_i = \pi - (\gamma + \beta) \quad (30).$$

$$(31).$$

3.7.2 Caso 2

El segundo caso ocurre cuando $P_{xi} < 0$ como se puede ver en la Figura 15.

Figura 15. Consideración geometría para $P_{xi} < 0$



Fuente: [4]

El ángulo q_i se calcula implementando la siguientes fórmulas:

$$L = \sqrt{(abs(P_{xi}) + rA)^2 + (P_{zi})^2} \quad (32).$$

$$\gamma = -1 * \tan^{-1}\left(\frac{P_{zi}}{(abs(P_{xi}) + rA)}\right) \quad (33).$$

$$\theta = \text{Sen}^{-1}\left(\frac{P_{yi}}{l_1}\right) \quad (34).$$

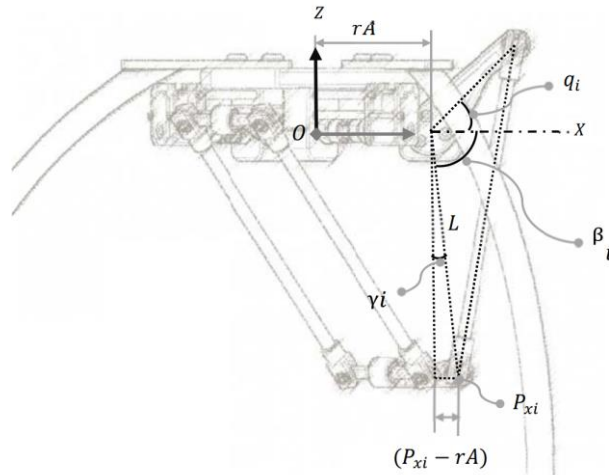
$$\beta = \text{Cos}^{-1}\left(\frac{L^2 + l_2^2 - (l_1 \cos \theta_i)^2}{2Ll_2}\right) \quad (35).$$

$$q_i = \pi - (\gamma + \beta) \quad (36).$$

3.7.3 Caso 3

El tercer caso ocurre cuando $P_{xi} > rA$ como se puede ver en la Figura 16.

Figura 16. Consideración geometría para $P_{xi} > rA$



Fuente [4]

El ángulo q_i se calcula implementando la siguientes fórmulas:

$$L = \sqrt{(P_{xi} - rA)^2 + (P_{zi})^2} \quad (37).$$

$$\gamma = -1 * \tan^{-1}\left(\frac{P_{xi} - rA}{P_{zi}}\right) \quad (38).$$

$$\theta = \text{Sen}^{-1}\left(\frac{P_{yi}}{l_1}\right) \quad (39).$$

$$\beta = \text{Cos}^{-1}\left(\frac{L^2 + l_2^2 - (l_1 \cos \theta)^2}{2Ll_2}\right) \quad (40).$$

$$q_i = \frac{\pi}{2} - (\gamma + \beta) \quad (41).$$

De esto se obtienen las variables necesarias para dar solución a la Cinemática Inversa en la programación del algoritmo, estas se pueden apreciar en Tabla 7 esto con la finalidad de tener mayor facilidad al utilizarlas en el algoritmo.

Tabla 7. Valores obtenidos de la Cinemática Inversa.

L	Longitud que hay desde el sistema de referencia al punto central de la herramienta.	Variable
-----	---	----------

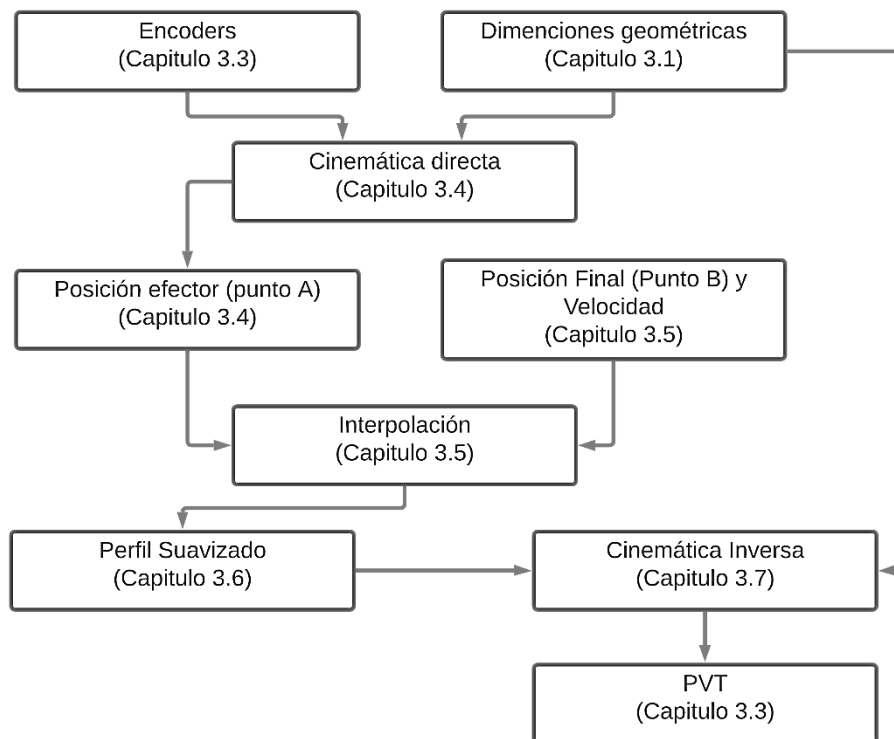
$P_{x,y,z}$	Hace referencia a la posición (x,y,z) de la unión entre el efector final y el paralelogramo.	Variable
q_i	Ángulo de los motores.	Variable
β_i	Ángulo entre el eje x y la línea de referencia L .	Variable
θ_i	Ángulo entre el eje z y el paralelogramo.	Variable

Fuente: [4] Autor.

3.8 Sistema de variables de entrada y salida.

En la Figura 17 se puede ver como en sus primer fila de recuadros se encuentran las variables de entrada de todo el sistema, siendo estas las dimensiones geométricas y los encoders de los servomotores y en su ultima fila estaría el PVT, las cuales son las variables de salida del sistema, y en su núcleo estarían las variables internas del sistema que vendrían a ser los requisitos que se tendrán para su correcto comportamiento como podrían ser las creadas en base a las Cinemáticas o al Perfil Suavizado.

Figura 17. Flujo de Variables de entrada y salida.



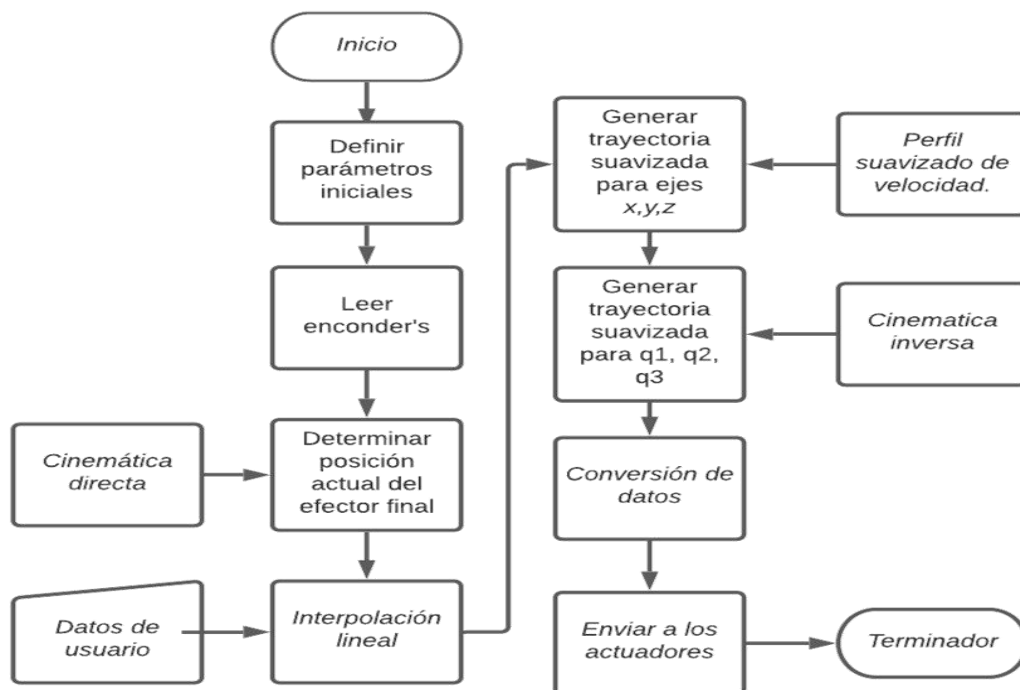
Fuente: Autor.

4. PROGRAMACIÓN DE ALGORITMOS BASES PARA EL APLICATIVO DE COMUNICACIÓN ENTRE VISUAL STUDIO Y GALIL DMC 40X3

Como bien se sabe para realizar una programación de cualquier tipo se necesita tener en claro qué se hará o tener una idea de esto, por lo cual el determinar un programa en el cual se realizará el aplicativo y el lenguaje de programación que se tendrá en cuenta para su desarrollo serán vitales, además de esto se realizarán varios diagramas en este proyecto, esto con la finalidad de tener control de todo lo que se quiere realizar en los algoritmos o códigos para así, de ser necesario lograr intervenir oportunamente en caso de querer realizar actualizaciones o cambios futuros.

La idea primordial entonces fue generar un movimiento de un punto inicial a un punto final, por ende, se desarrolla el primer diagrama (Figura 18) en el cual se plantea de forma sencilla y práctica el cómo se implementará esta idea.

Figura 18. Diagrama para un movimiento de A a B.



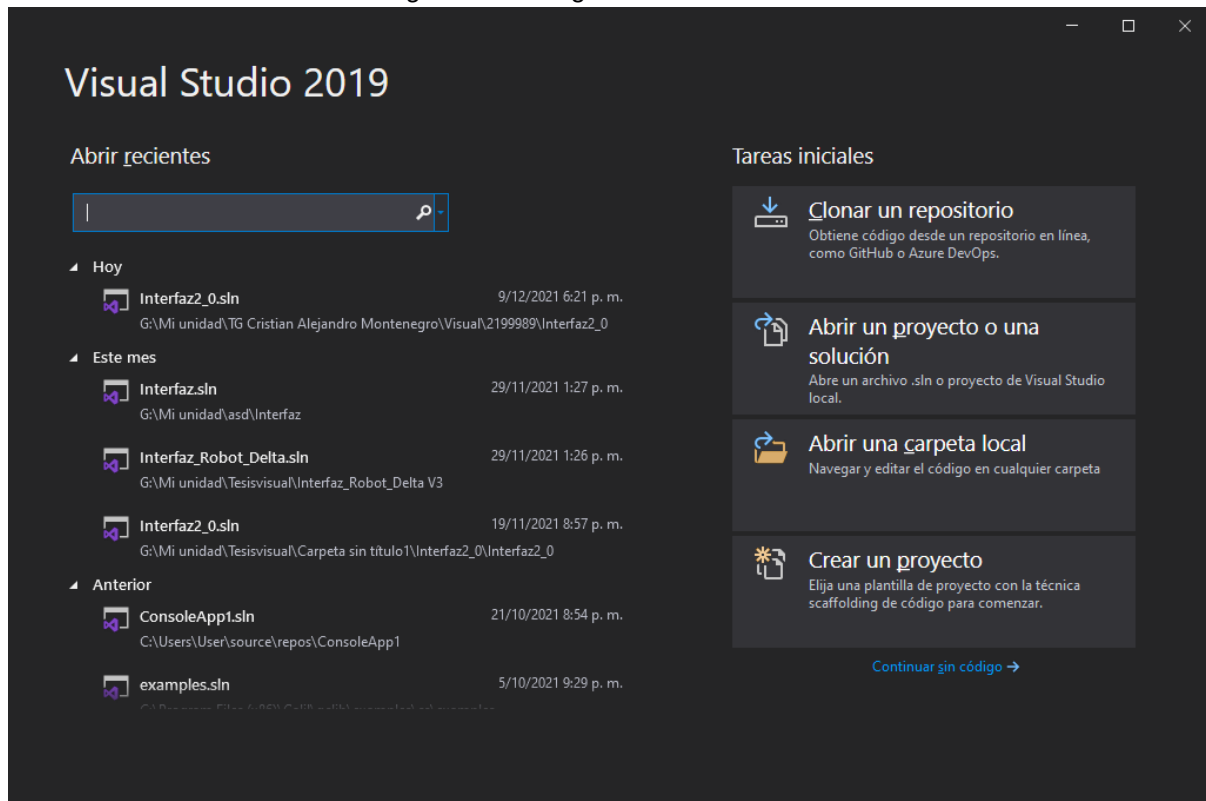
Fuente: Autor.

4.1. Programa y lenguaje del aplicativo

Como programa base del aplicativo se utiliza el programa *Microsoft Visual Studio* el cual es un entorno de desarrollo integrado para Windows o MacOS como se puede ver en la Figura 19. El cual es compatible con lenguajes de programación como lo son C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP al igual que entornos de desarrollo web como ASP.NET MVC, Django, entre otros [12]. Esto lo convierte en un

programa capacitado en crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

Figura 19. Programa Visual Studio 2019.



Fuente: Autor.

Para el lenguaje de programación se utilizó C#, el cual está orientado a objetos y componentes permitiéndole a los desarrolladores crear muchos tipos de aplicaciones seguras y sólidas que se ejecutan en .NET [12]. Además de ser compatible con el programa seleccionado y librerías otorgadas por el controlador.

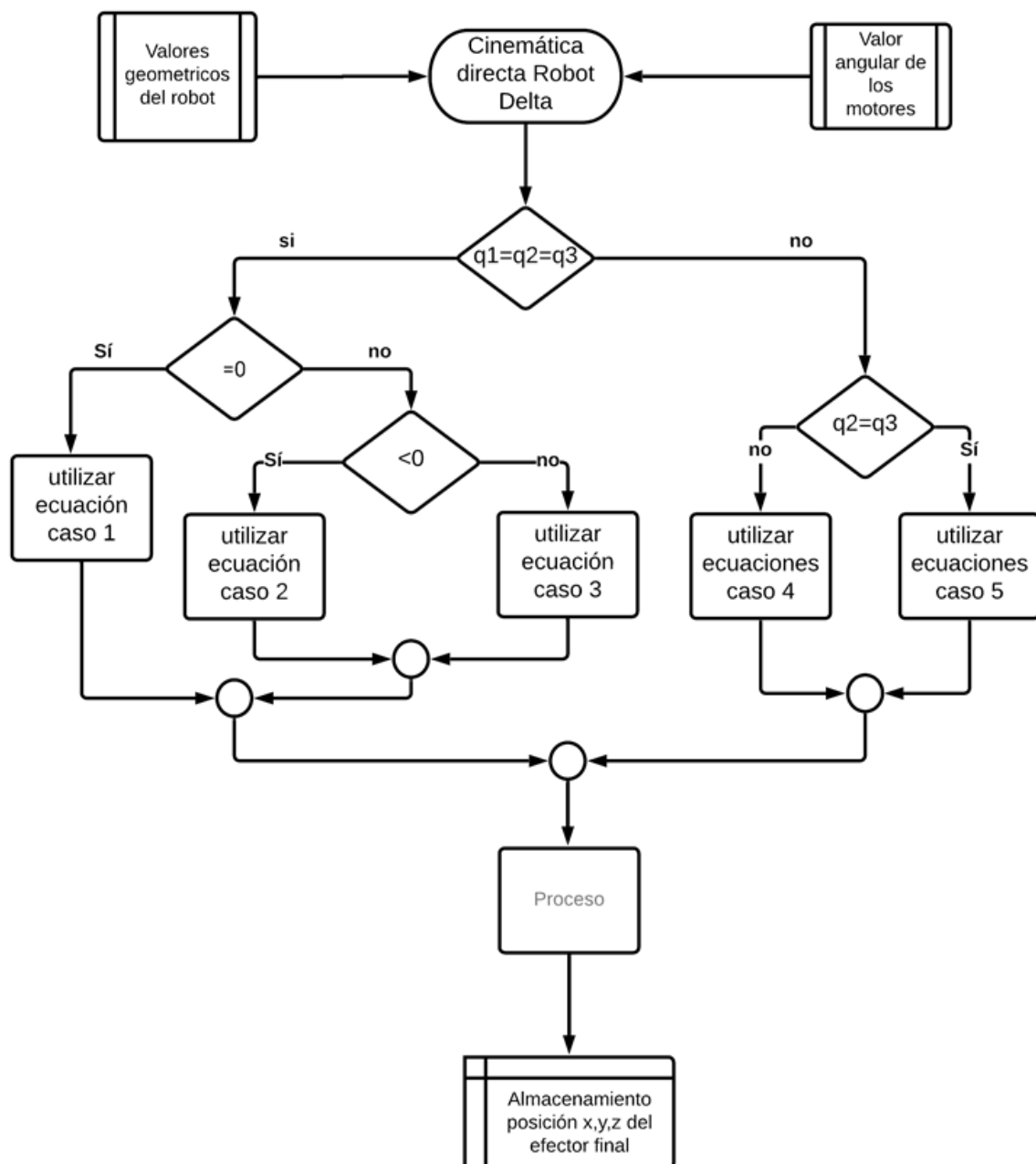
4.1.1 Conceptos utilizados.

Los conceptos clave en C # son los siguientes: programas, espacios de nombres, tipos, miembros y ensamblados. Los programas declaran tipos, que contienen miembros y se pueden organizar en espacios de nombres. Las clases, estructuras e interfaces son ejemplos de tipos. Los campos, métodos, propiedades y eventos son ejemplos de miembros. Cuando se compilan los programas de C #, se empaquetan físicamente en ensamblados. Los ensamblados suelen tener la extensión de archivo .exe o .dll, dependiendo de si implementan aplicaciones o bibliotecas, respectivamente [12].

4.4 Programación Cinemática directa.

La programación de la cinemática directa se desarrolló en base a la Figura 20, como se puede apreciar se llama la clase de Cinemática directa esta recibe los valores de los ángulos de los motores y valores geométricos del robot, luego de esto empieza definir en qué caso está dependiendo de la singularidad en la que se encuentra según la posición angular de los 3 motores.

Figura 20. Cinemática directa.



Fuente: Autor.

Cuando el código determina en qué caso se encuentra el movimiento se aplicará las ecuaciones respectivas y posteriormente guardará la posición encontrada en la que estará el efector en las coordenadas x, y, z.

En la Figura 21 se puede apreciar el momento en el que ingresan los valores angulares de los motores y geométricos del robot además de las conversiones necesarias de unidades para su correcta lectura en la programación. El código completo se puede ver en el ANEXO D: CÓDIGO CINEMÁTICA DIRECTA.

Figura 21. Código de variables (Cinemática Directa).

```

9  class Cinematica_Directa
10 {
11     public double znd, xnd, ynd;
12     0 referencias
13     public void CineDirecta(double PosicionA, double PosicionB, double PosicionC, double L1, double L2, double R)
14     {
15         double la = L2;
16         double r1 = L1;
17         double ang1 = (0 * System.Math.PI) / 180; // angulo del brazo 1
18         double ang2 = (120 * System.Math.PI) / 180; // angulo del brazo 2
19         double ang3 = (240 * System.Math.PI) / 180; // angulo del brazo 3
20         double q1 = PosicionA * 360 / 100000;
21         double q2 = PosicionB * 360 / 100000;
22         double q3 = PosicionC * 360 / 100000;
23         double q11 = (-q1 * System.Math.PI) / 180;
24         double q22 = (-q2 * System.Math.PI) / 180;
25         double q33 = (-q3 * System.Math.PI) / 180;
26     }

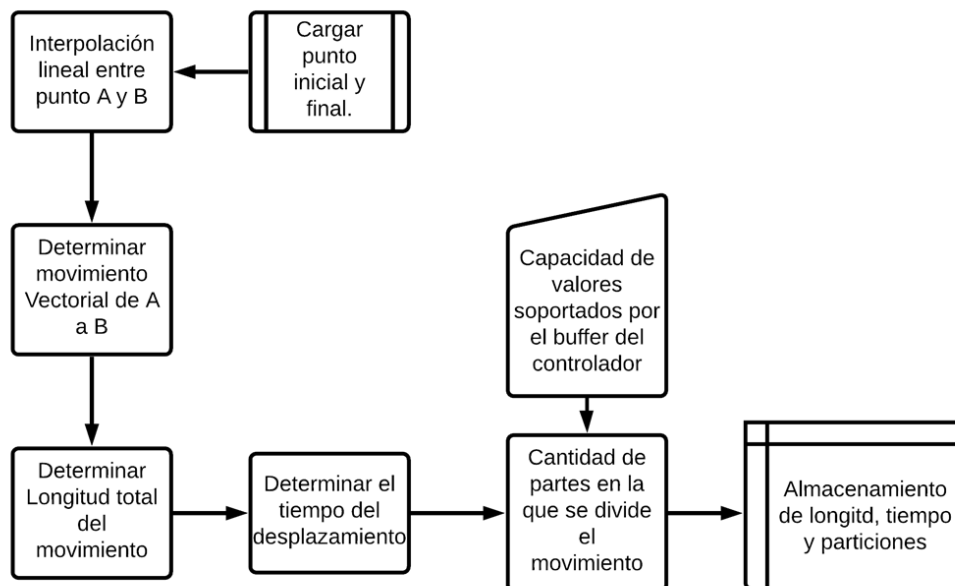
```

Fuente: Autor.

4.2 Programación Interpolación lineal

Para la programación de la interpolación lineal se desarrolló en base al Figura 22.

Figura 22. Interpolación lineal.



Fuente: Autor.

Con lo cual siguiendo el diagrama de la Figura 22 se logra ver que una vez se llama la clase de interpolación esta recibe los puntos tanto inicial como final de la trayectoria, teniendo esto el programa determina el movimiento vectorial que hace la trayectoria en x, y, z, teniendo estos valores se aplican las ecuaciones correspondientes para establecer la longitud total del movimiento y seguido a esto se determinará el tiempo que tomará el desplazamiento de A a B.

Finalmente se ingresa en el código la capacidad con la que cuenta el buffer del controlador para sacar las particiones que tendrá el movimiento y su posterior almacenamiento.

Figura 23. Código de variables (Interpolación).

```
public class Interpolacion
{
    public double tiemdespl, delX, delY, delZ, N, vecdx, vecdy, vecdz;
    1 referencia
    public void Inter(double xf, double yf, double zf, double vel, double xi, double yi, double zi)
    {
        vecdx = Math.Abs(xf - xi);
        vecdy = Math.Abs(yf - yi);
        vecdz = Math.Abs(zf - zi);
        double L = Math.Sqrt((Math.Pow(vecdx, 2)) + (Math.Pow(vecdy, 2)) + (Math.Pow(vecdz, 2)));
        tiemdespl = L / vel;
    }
}
```

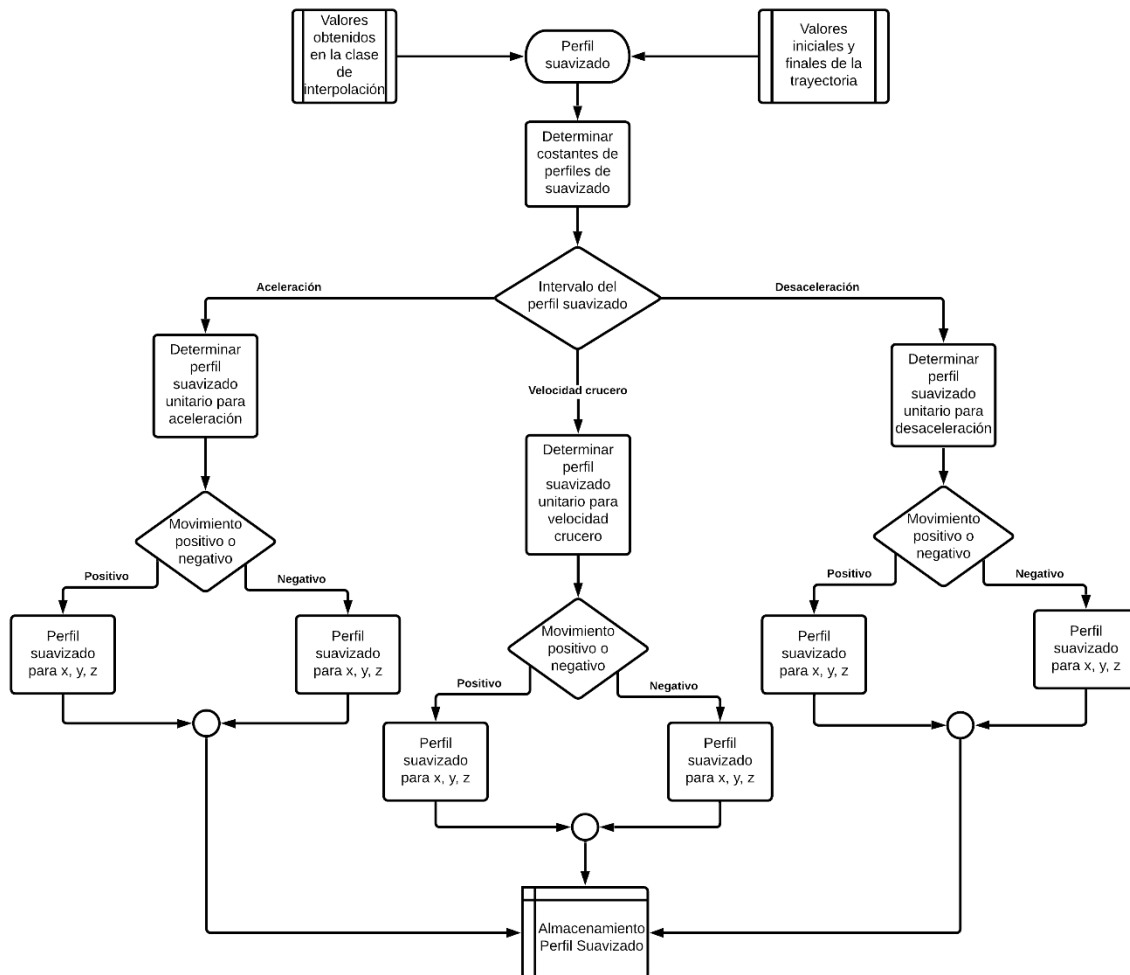
Fuente: Autor.

En la Figura 23 se puede notar como se definen algunas variables las cuales serán almacenadas posteriormente como lo pueden ser “L” que sería la Longitud y “tiemdespl” que es el tiempo que tarda en desplazarse del punto inicial al final. El código completo se puede ver en el ANEXO F: CÓDIGO INTERPOLACIÓN Y PERFIL SUAVIZADO.

4.3 Programación Perfiles Suavizados

Para la programación del perfil suavizado se desarrolló en base al Figura 24. Se puede apreciar que inicialmente la clase de Perfil suavizado recibe los valores de los puntos inicial y final de la trayectoria y a su vez los valores obtenidos de la clase Interpolación, con estos valores se aplicarán las ecuaciones correspondientes para encontrar las constantes que se pueden ver en (24).

Figura 24. Perfil Suavizado.



Fuente: Autor.

Siguiendo el flujo del diagrama se determina si la trayectoria está en su etapa de aceleración, velocidad crucero o desaceleración, luego de esto dependiendo si el movimiento se genera hacia la dirección positiva o negativa de cualquier eje teniendo en cuenta el sistema coordenado se pasará a la aplicación de las ecuaciones esto con la finalidad de crear el perfil suavizado.

Figura 25. Constantes de integración (Perfil suavizado).

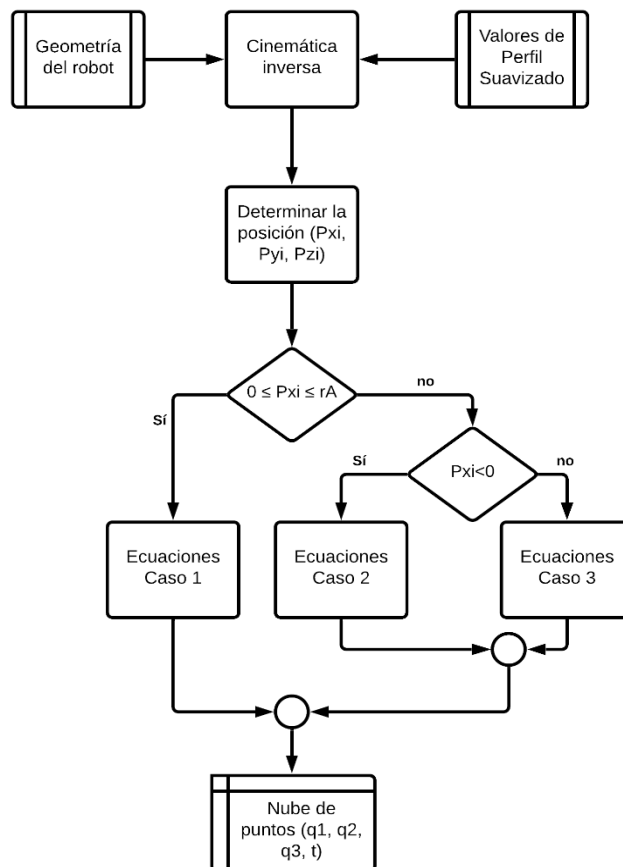
```
double n = N;
int N1 = Convert.ToInt32(n);
double delT = 1 / N;
double t1 = 1 * 0.1;
double t2 = 1 - 1 * 0.1;
double k1 = 0;
double c1 = (1 - k1) / (1 + t2 - t1);
double c2 = 2 * c1;
double k2 = k1 - t1 * c1;
double c3 = c1;
double k3 = k1 + c1 * (t2 - t1);
cont = 0;
double i.. = 0;
perS = new double[4, N1 + 2];
```

Fuente Autor.

En la Figura 25 se puede observar cómo se calculan las constantes necesarias para aplicar las ecuaciones del perfil suavizado además de cómo se crea la matriz en donde se almacenarán los valores de este. El código completo se puede ver en el ANEXO F: CÓDIGO INTERPOLACIÓN Y PERFIL SUAVIZADO.

4.5 Programación Cinemática Inversa

Figura 26. Cinemática inversa.



Fuente: Autor.

La programación de la cinemática inversa se desarrolló en base a la Figura 26. En este se puede ver como se traen los valores obtenidos de la clase de Perfil Suavizado y a su vez los valores geométricos del robot, luego de esto se identifican los valores de la posición en donde se une cada brazo con el efector final, con los cuales se identifica en qué caso se encuentra el robot aplicando así las ecuaciones correspondientes a estos dando como resultado los valores angulares de los motores los cuales son guardados y el tiempo.

Figura 27. Código para los valores de posición (Cinemática Inversa).

```
// Eje 2 (Rotación plano -240°)

double cos1 = -0.5;
double cos2 = -0.5;
double sen1 = 0.866025403784438;
double sen2 = -0.866025403784438;

Posi[7, cont1] = (Posi[4, cont1] * cos1) + (Posi[5, cont1] * sen1); //px2
Posi[8, cont1] = (Posi[4, cont1] * sen2) + (Posi[5, cont1] * cos2); //py2
Posi[9, cont1] = perS1[3, cont1]; //pz2

// Eje 3
Posi[10, cont1] = perS1[1, cont1] - 25;
Posi[11, cont1] = perS1[2, cont1] - 43.30127019;
Posi[12, cont1] = perS1[3, cont1];

// Eje 3 (Rotación plano -120°)

double cos3 = -0.5;
double cos4 = -0.5;
double sen3 = -0.866025403784438;
double sen4 = 0.866025403784438;
```

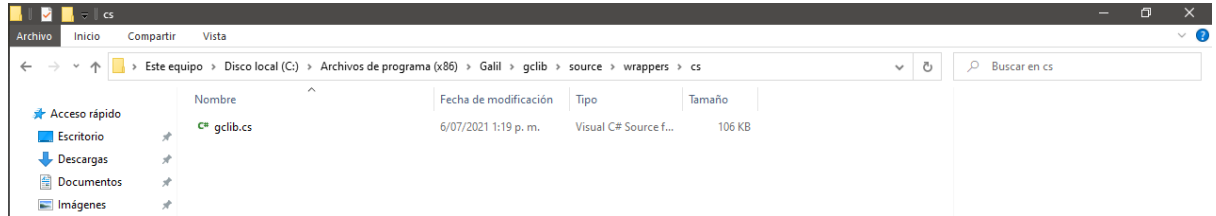
Fuente: Autor.

En la Figura 27 se muestra una parte del código en donde se encuentra la posición en x, y, z de donde se une los brazos 2 y 3 con el efector final. El código completo se puede ver en el ANEXO E: CÓDIGO CINEMÁTICA INVERSA.

4.6 Comunicación controlador DMC

El primer paso para realizar la comunicación entre Visual Studio y el controlador Galil DMC 41x3 es instalar “Galil Library Docs”, mediante un ejecutable proporcionado por Galil siendo estas las bibliotecas que proporcionan los códigos necesarios para el envío de datos, una vez se tenga esto instalado en el dispositivo se agrega en el proyecto una clase llamada “gclib” mediante el explorador de soluciones de este, la ruta en donde se encuentra esta clase será la siguiente “C: \ProgramFiles(x86) \Galil\gclib\source\wrappers\cs” como se puede apreciar en la Figura 28 [9]. Cabe recalcar que esta ruta se crea luego de instalar correctamente las librerías.

Figura 28. Clase necesaria para el funcionamiento de las librerías.

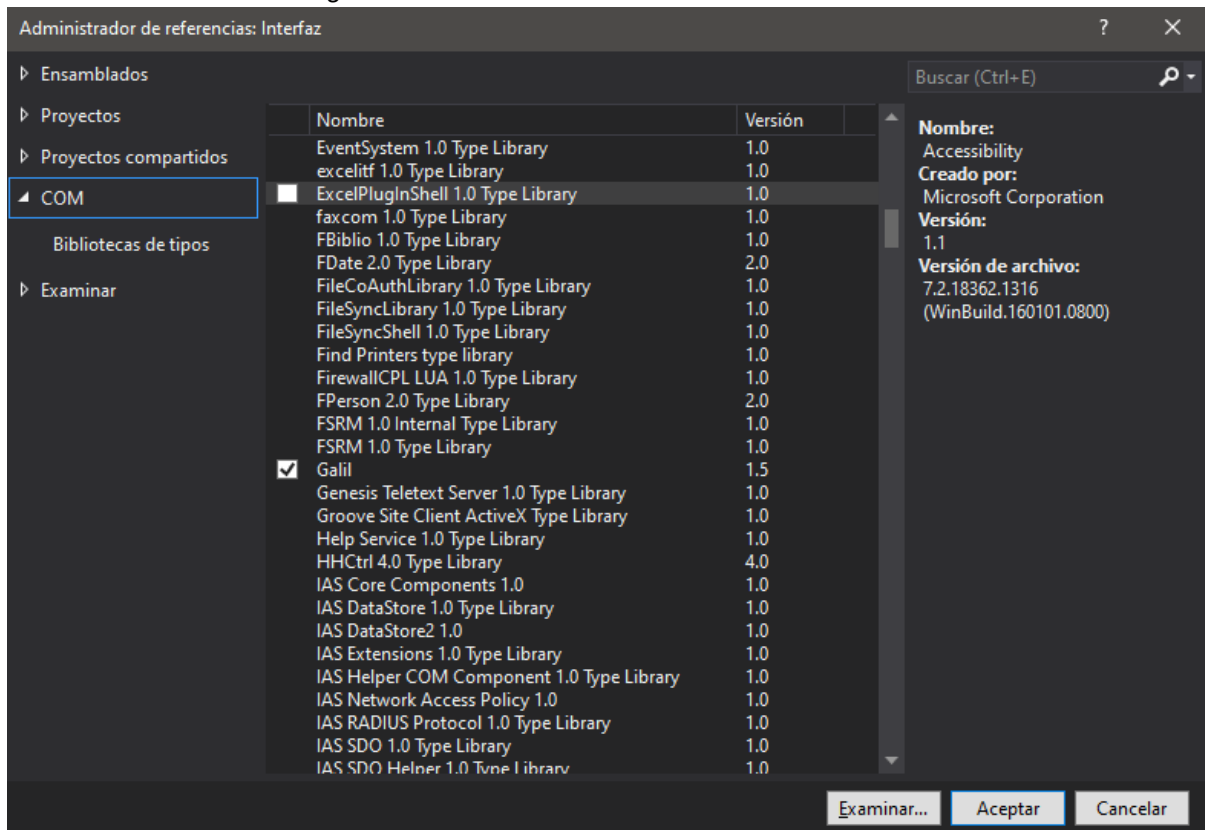


Fuente: Autor.

Una vez terminado este paso ya se podrá llamar valores de esta clase en el proyecto principal, esto referenciando la clase de la siguiente forma: `gclib gclib = new gclib();`. Esto hará posible que siempre al nombrar “gclib” dentro del código se pueda utilizar libremente como por ejemplo encender los motores mediante el código: `gclib.GCommand("SH ABC");`.

Terminado eso se pasará a instalar el programa “GalilTools” el cual al igual que las librerías el ejecutable de instalación es proporcionado directamente por la página de Galil. Al instalarse el programa se procederá a ingresar en el proyecto la referencia del controlador el cual ayudará en la recepción de valores del controlador, esto se hará mediante el explorador de soluciones en la opción de “referencias COM” y se seleccionará la opción llamada “Galil” como se puede ver en la Figura 29 [8].

Figura 29. Administrador de referencias Visual Studio.



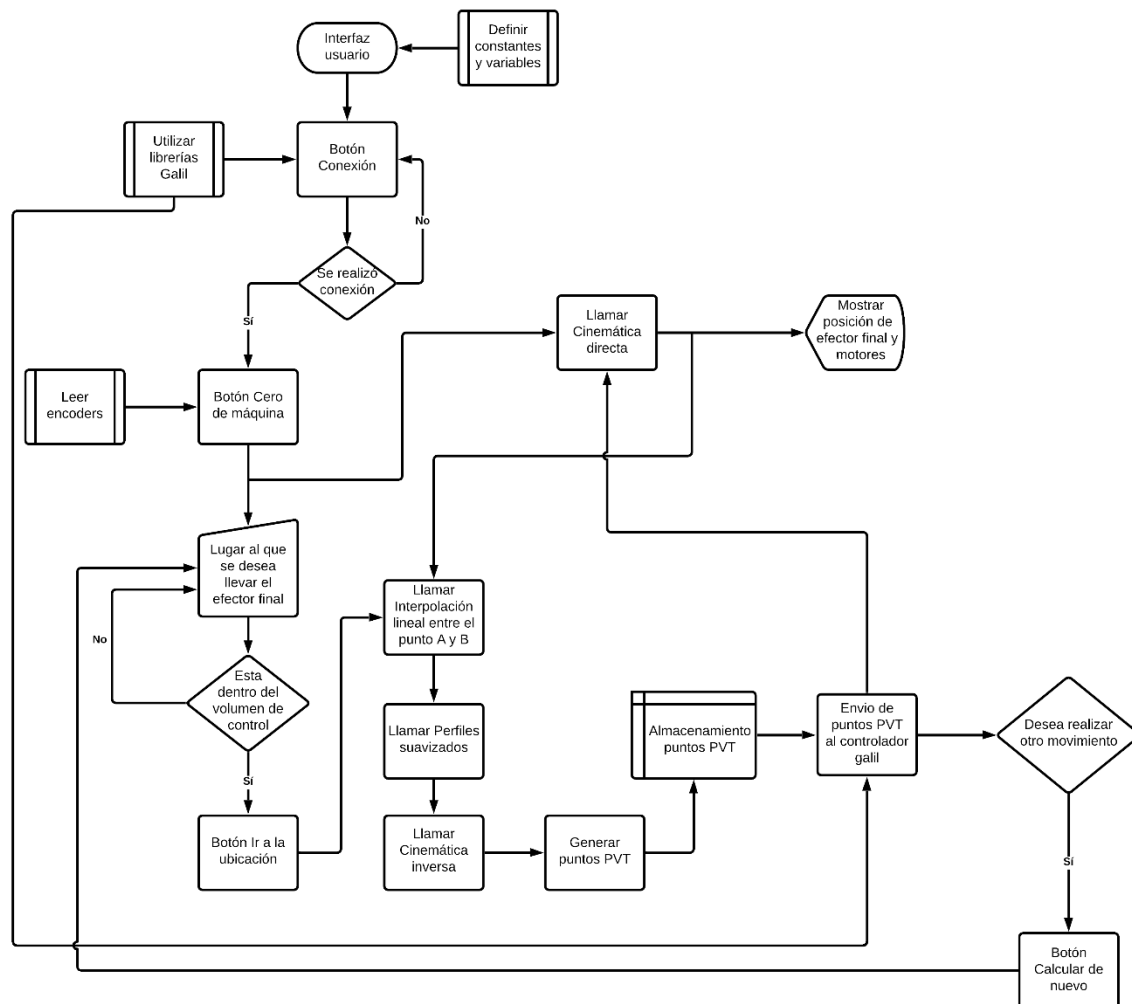
Fuente: Autor

Con la referencia COM agregada ya se podrá traer al dispositivo los valores que arroja el controlador, esto se logrará al referenciarlo en el proyecto de la siguiente forma “Galil.Galil g = new Galil.Galil();”. Esto hará posible que siempre al nombrar “g” dentro del código se pueda utilizarla libremente como por ejemplo el vincular los valores de los encoder de los motores a una variable mediante el siguiente comando “double PosicionA = g.commandValue("TPA")”;

4.2 Programación interfaz

La interfaz de usuario se creó siguiendo la Figura 30 en donde se establecieron procesos predefinidos como el definir constantes, variables y los códigos ofrecidos por las librerías proporcionadas por el controlador Galil, así una vez se inicia la interfaz lo primero que se podrá hacer será interactuar con el botón de conexión el cual utilizará el comando desde las librerías de Galil, siendo este puente entre nuestra interfaz y el controlador, logrando así la conexión en tiempo real con el controlador.

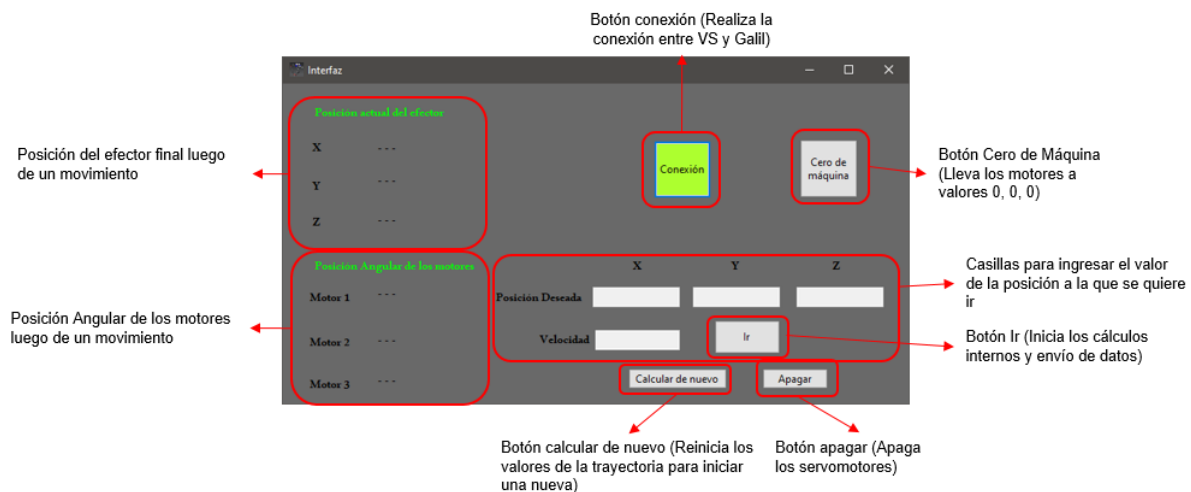
Figura 30. Funcionamiento de la interfaz de usuario.



Fuente: Autor.

Una vez se logre este puente se desbloqueará el siguiente botón “Cero de máquina” el cual leerá los encoders proporcionados por el controlador para así determinar el cero de máquina en dónde los motores tienen como valor angular 0, una vez realice esto hará dos cosas, la primera de estas es llamar a la clase Cinemática directa, con la finalidad de mostrar en la interfaz (pantalla) la posición actual tanto de los motores como del efector final, determinando así la posición inicial de la trayectoria, la otra cosa que hará será habilitar los TextBox que serán las casillas en donde se podrá digitar la posición a la que se quiere ir y el botón “Ir” como se puede apreciar en la Figura 31.

Figura 31. Interfaz gráfica de usuario.



Fuente: Autor.

Una vez se llenen las casillas y se opriman el botón ir él determinará si la posición a la que se quiere ir está dentro del volumen útil, de no estar en este saldrá un aviso y se pedirá que se ingrese de nuevo la posición a la que se quiere ir, si se esta dentro de este volumen útil empezará a interactuar internamente llamando la clase de interpolación lineal, terminado los cálculos correspondientes se dirigirá a la clase de Perfiles suavizados, al ejecutar el código pasará a la cinemática inversa en donde definirá los valores de los motores mediante las ecuaciones, seguido de esto pasará crear la nube de puntos de Posición, Velocidad y Tiempo, la cual es almacenada y preparada para su posterior envío al controlador.

Al ser enviados los valores al controlador volverá a leer los encoders de los motores y llamar la Cinemática directa, con la finalidad de que muestre en la pantalla la posición en la que se encuentra los motores y efector final luego de terminar el movimiento, a su vez activará el botón “Calcular de nuevo” el cual habilitará la opción de digitar valores para una nueva trayectoria, el código creado se puede ver en el ANEXO C: CÓDIGO INTERFAZ.

5. PRUEBAS DEL APLICATIVO

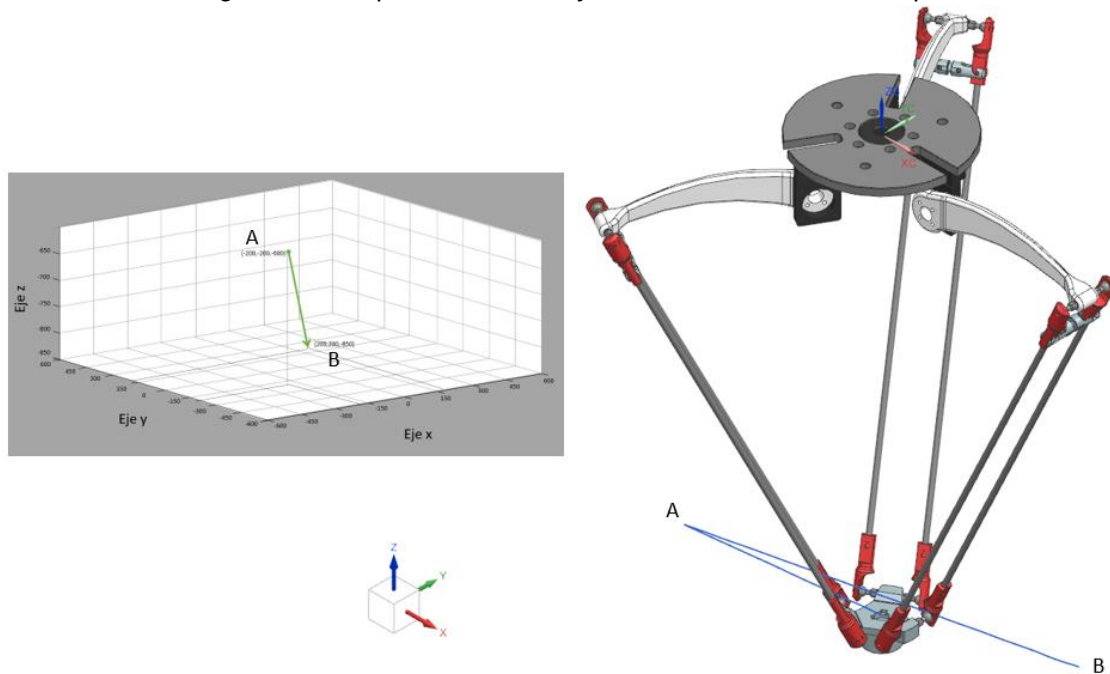
Para verificar el buen funcionamiento de la interfaz creada, el funcionamiento óptimo de la programación de tanto la cinemática inversa como la cinemática directa y el desempeño del robot delta de la Universidad Santo Tomás se realizaron trayectorias lineales en el espacio, las cuales ya han sido propuestas en otros trabajos [3][6].

Las trayectorias que se demostrarán tienen como referencia su sistema cartesiano ubicado en el centro de la plataforma fija 0,0,0 siendo este el punto de origen y ubicando el punto central del efector final en 0,0,-758,9466 con lo cual los valores arrojados de la posición serán con respecto al sistema cartesiano de referencia.

5.1 Trayectoria lineal A – B en el espacio.

Esta trayectoria consiste en un desplazamiento desde (-200, -200, -600) hasta (200, 200,-850), siendo el recorrido de 618,47 mm como se puede ver en la Figura 32.

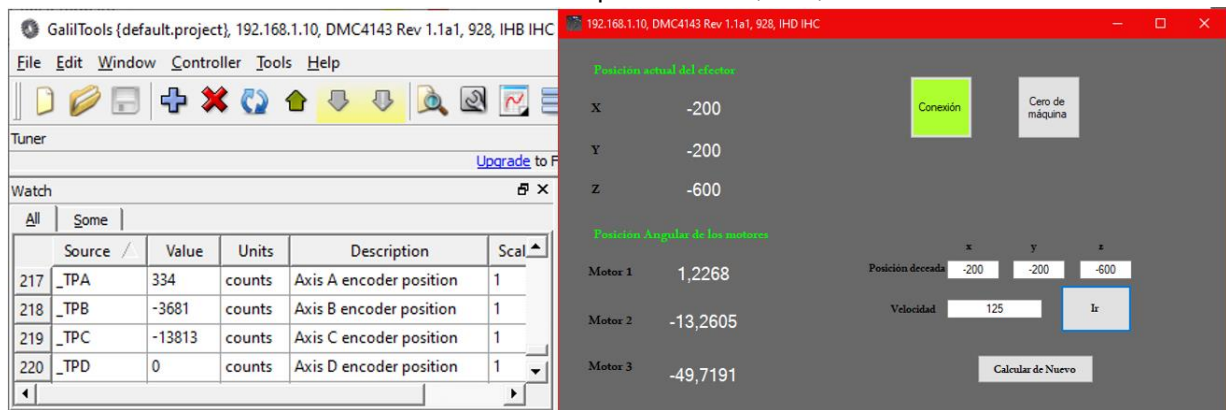
Figura 32. Representación Trayectoria lineal A – B en el espacio.



Fuente: [11] Autor.

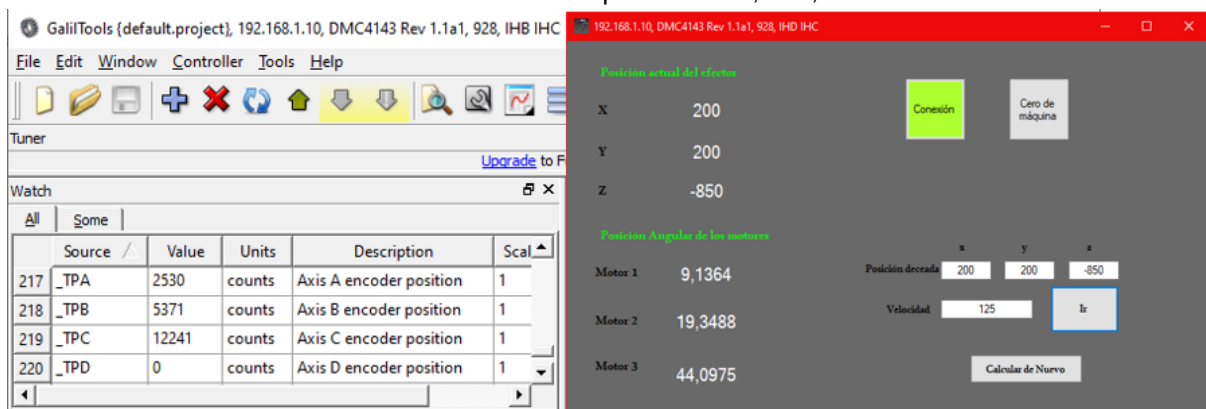
Al realizar este movimiento con los motores reales del robot se revisaron los valores de los ángulos proporcionados por el controlador Galil y la interfaz como se puede apreciar en la Figura 33 y Figura 34.

Figura 33. Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición -200,-200,-600



Fuente: Autor.

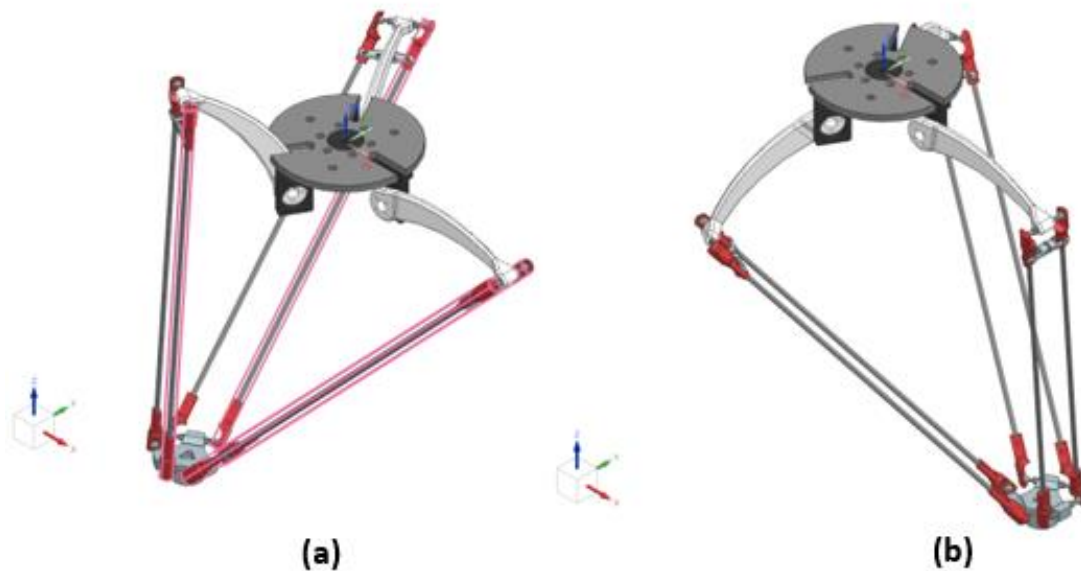
Figura 34. Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 200, 200,-850



Fuente: Autor.

Para tener una mayor seguridad de que el robot se moverá a la posición deseada se utilizó el gemelo virtual obtenido de [11] en donde se hace la misma trayectoria en donde se puede apreciar en la Figura 35 (a) la ubicación de todos sus motores y el efector final en la posición inicial y a su vez en la Figura 35 (b) la ubicación de todos sus motores y el efector final en la posición final.

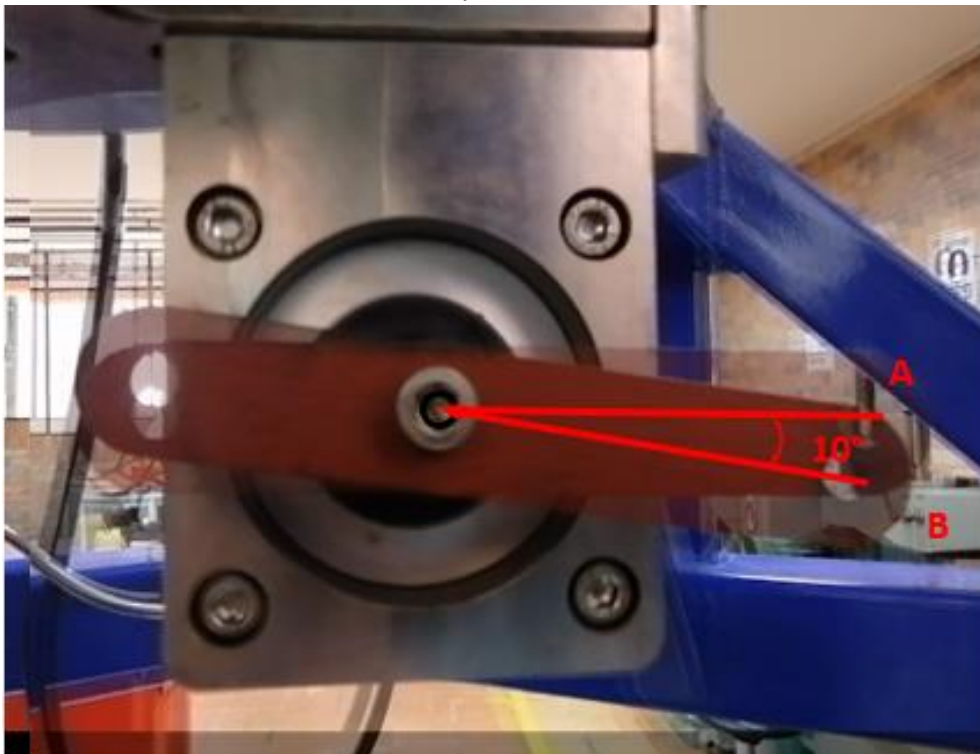
Figura 35. Posición A y posición B de la trayectoria.



Fuente: [11] Autor.

Para revisar si estos valores del Galil Tools e interfaz se reflejaban correctamente en la realidad se tomaron varias fotografías además de grabarse un video al momento de hacer la trayectoria, siguiendo la posición de los tres motores, en donde se compara la posición antes y después de terminar el movimiento con la finalidad de identificar la distancia que hay entre los motores al final de la trayectoria.

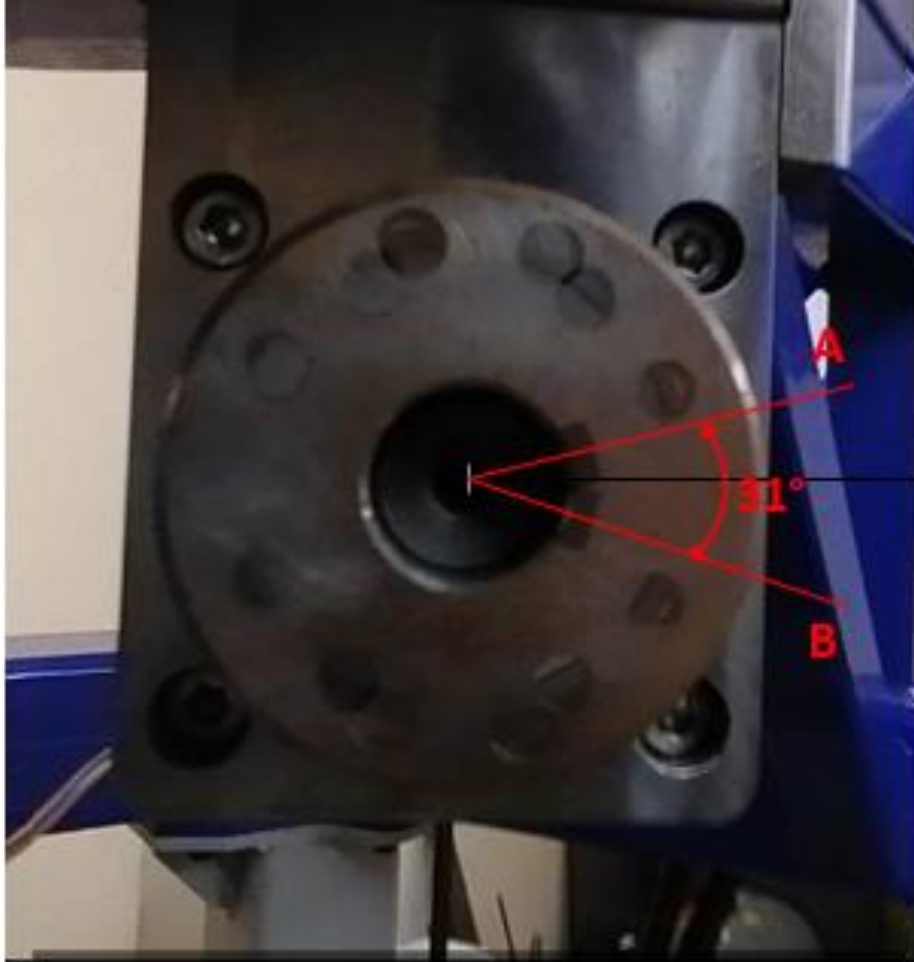
Figura 36. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 1.



Fuente: Autor.

En la Figura 36 se puede observar como la posición del motor 1 en A y B está por debajo de su centro, con lo cual se puede afirmar que es una posición similar a la que dicen los valores de la Figura 33 y Figura 34 en el motor 1. además de que al analizarse mediante programas como lo son Solid Edge y Tracker se consigue el ángulo que hay entre las dos posiciones siendo este de 10° .

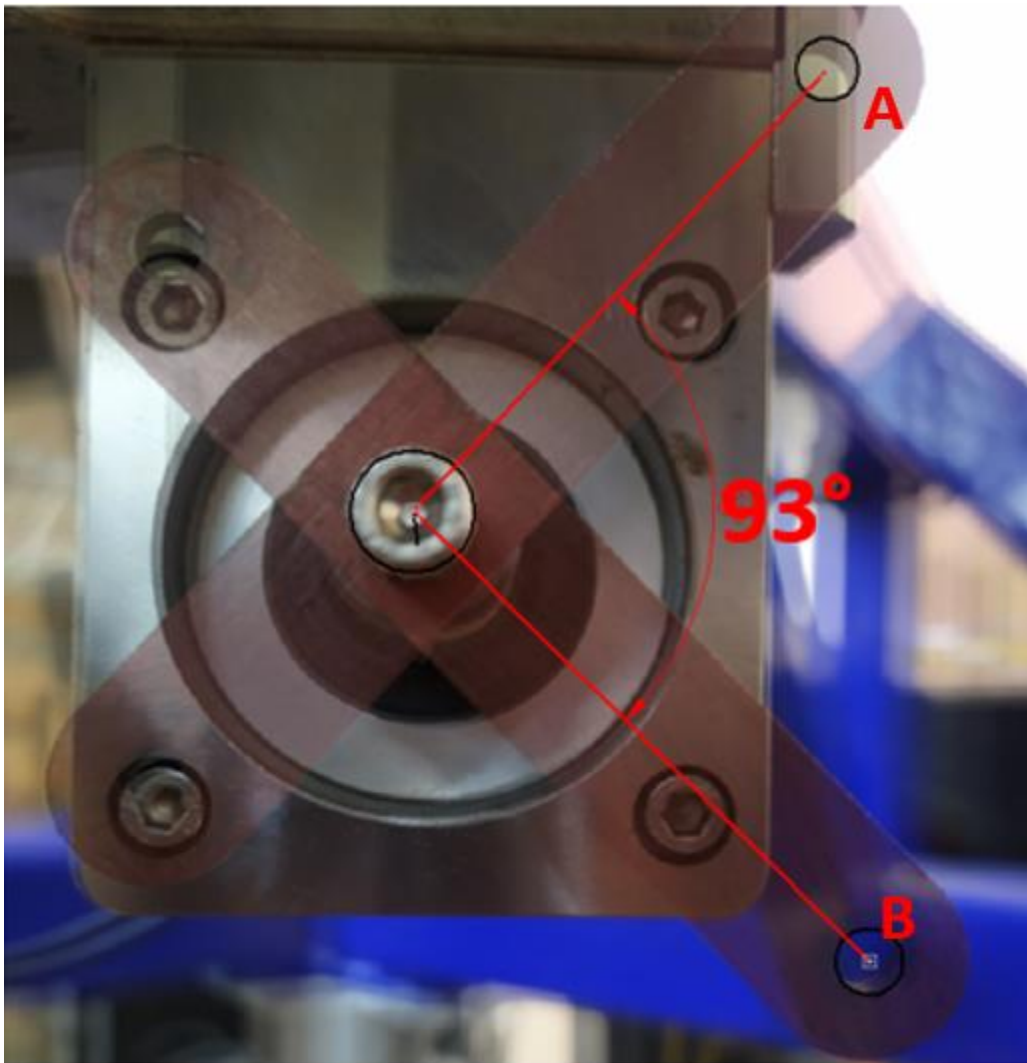
Figura 37. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 2.



Fuente: Autor.

En la Figura 37 se observa como la posición del motor 2 en A está por encima de su eje, dando la seguridad que efectivamente se encuentra en los valores negativos teniendo sentido si se tiene en cuenta los valores de la Figura 33 y a su vez en la posición B se ve por debajo del eje, concordando con lo que se aprecia en la Figura 34 además de que al analizarse se llega a que el ángulo que hay entre las dos posiciones es de 31° .

Figura 38. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor real 3.

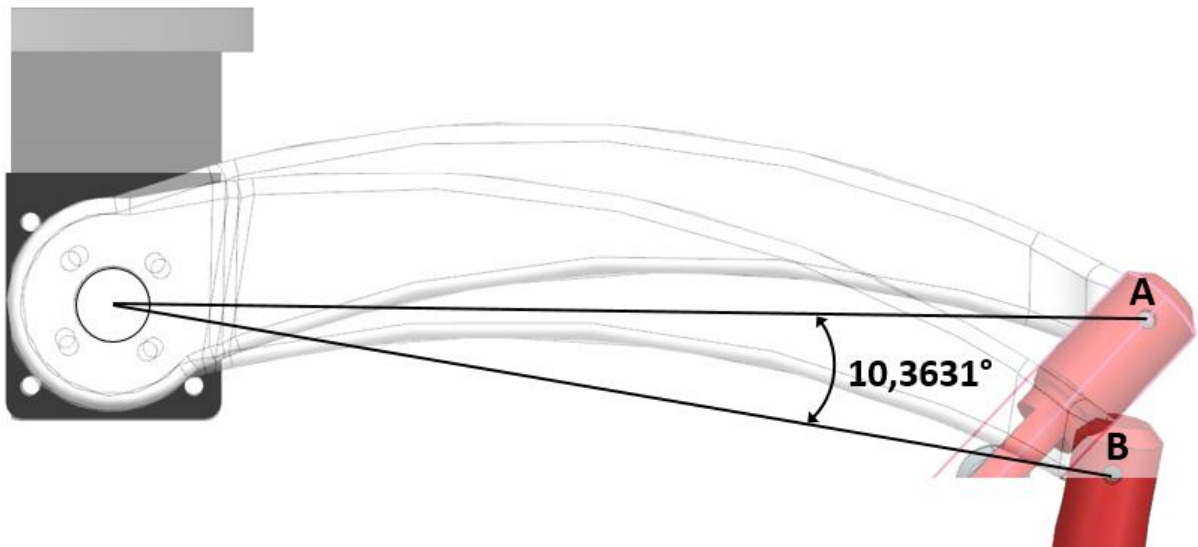


Fuente: Autor.

En la Figura 38 se logra ver como la posición del motor 3 en A está por muy encima de su eje, y a su vez la posición de B está muy por debajo, dando la seguridad que se encuentra en los valores correspondientes de la Figura 33 y Figura 34, al analizar estas posiciones se llegó a que el ángulo que hay 93°

Para comprobar si la posición de los motores se hace la misma comparación que en la realidad, pero esta vez a los motores virtuales, esto con la finalidad de corroborar que los valores que se envían a los motores reales convergen correctamente en un futuro ensamble completo ya que en el gemelo virtual como se aprecia en la Figura 35 se llega a las posiciones deseadas.

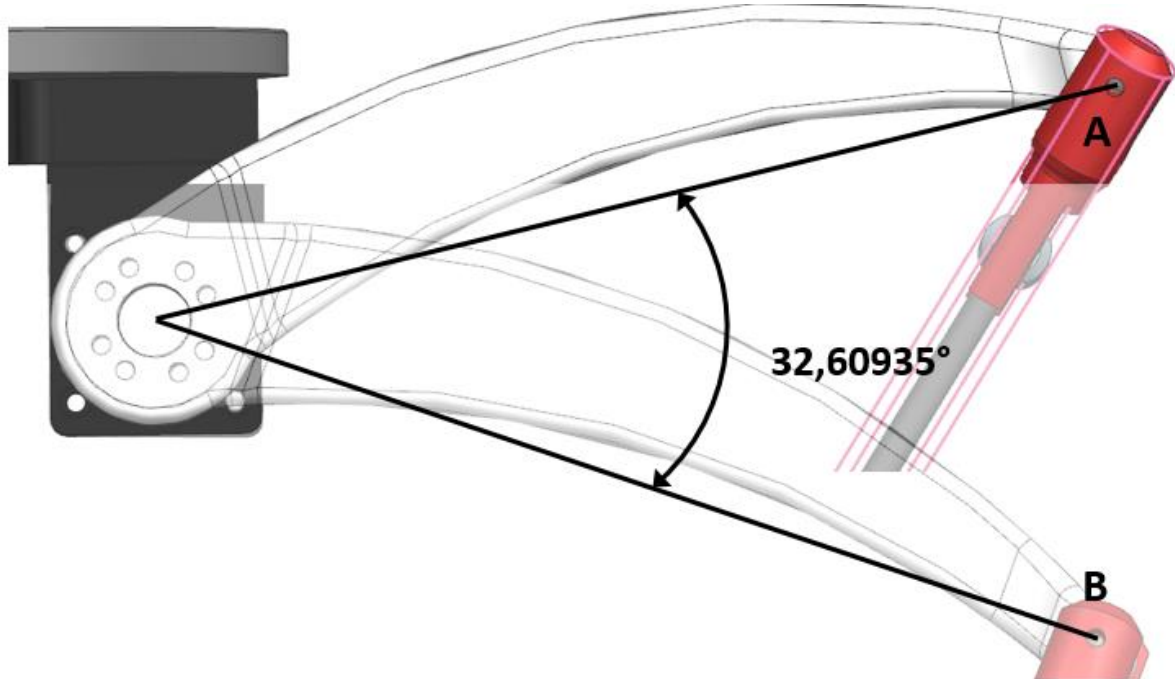
Figura 39. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 1.



Fuente: [11] Autor.

En la Figura 39 se puede observar como la posición del motor 1 virtual y su brazo en A y B están prácticamente iguales que como se puede ver en la Figura 36 dando una seguridad bastante grande al determinar la longitud que hay entre las posiciones.

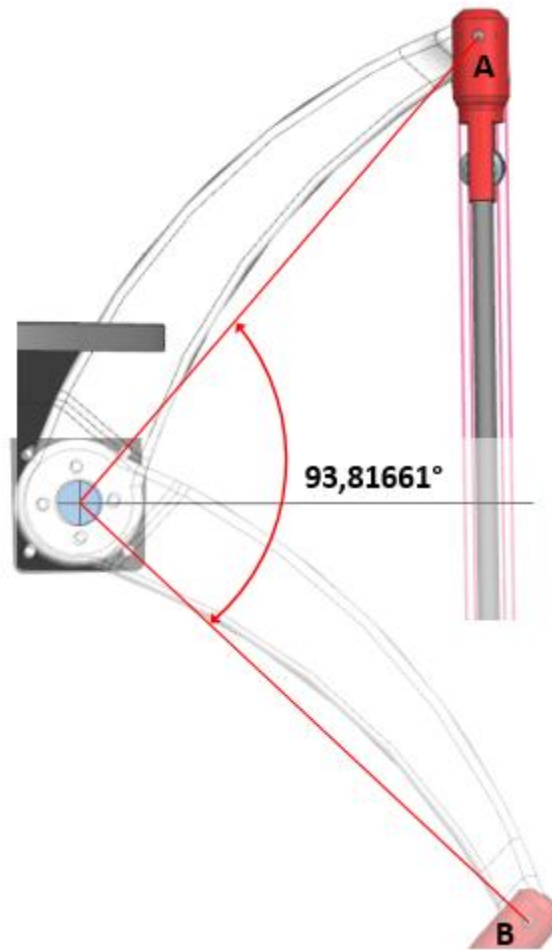
Figura 40. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 2.



Fuente: [11] Autor.

En la Figura 40 se contempla como la posición del motor 2 en A está por encima de su eje, y a su vez en la posición B se ve por debajo del eje, esto se logra apreciar igualmente en la Figura 37.

Figura 41. Comparación de la posición antes y después del movimiento de la Trayectoria lineal A – B espacio del motor virtual 3.



Fuente: [11] Autor.

En la Figura 41 se nota como la posición del motor 3 en A está muy por encima de su eje, al igual que en la posición B se ve muy por debajo, siendo esta en donde se logra apreciar más la similitud que se tiene con los motores reales por su distancia entre posiciones.

Al realizar estas comparaciones entre las diferentes herramientas se plantea el realizar tablas en donde se pueda apreciar más la exactitud del movimiento de los motores.

Tabla 8. Desplazamiento Angular del motor 1

	Desplazamiento angular del motor 1		
	Valor inicial	Valor final	Desplazamiento total
GalilTools	1.2024°	9.108°	10.3104°
Interfaz	1.2267°	9.1364°	10.3631°
Fotografías	1°	9°	10°
NX	1.226787°	9.1364°	10.363187°

Fuente: Autor.

De la Tabla 8 se puede determinar que la diferencia de posición en el motor analizado es increíblemente pequeña, teniendo una diferencia de $0,000870^\circ$ entre lo obtenido en la interfaz y el NX, además de que la diferencia entre la interfaz y el GalilTools es de $0,057^\circ$ y finalmente la diferencia entre la interfaz y lo que arroja el análisis de los motores es de $0,3631^\circ$ siendo esta la más grande en este motor, posiblemente por errores causados por no tener una base fija al momento de grabar y tomar las tomas, dando paso a que la persona que hizo estas tareas se moviera o cambiara ligeramente el ángulo de la cámara.

Tabla 9. Desplazamiento Angular del motor 2

	Desplazamiento angular del motor 2		
	Valor inicial	Valor final	Desplazamiento total
GalilTools	-13.2516	19.3356	32.5872
Interfaz	-13.2605	19.3488	32.5793
Fotografías	-13	18	31
NX	-13.260522	19.348833	32.609355°

Fuente: Autor.

De la Tabla 9 se puede determinar que la diferencia de posición en el motor 2 es igualmente pequeña pero no tanto como en la anterior, teniendo una diferencia de $0,030055^\circ$ entre lo obtenido en la interfaz y el NX, además de que la diferencia entre la interfaz y el GalilTools es de $0,0079^\circ$ y finalmente la diferencia entre la interfaz y lo que arroja el análisis de los motores es de $1,5793^\circ$ siendo esta la más grande en este motor y en toda la comparación, posiblemente por errores causados por no tener una base fija al momento de grabar y tomar las tomas, dando paso a que la persona que hizo estas tareas se moviera o cambiara ligeramente el ángulo de la cámara además que al no tener un punto de referencia delimitado como en los otros motores esto pudo hacer que aumentará esta diferencia.

Tabla 10. Desplazamiento Angular del motor 3

	Desplazamiento angular del motor 1		
	Valor inicial	Valor final	Desplazamiento total
GalilTools	-49,7268°	44,0676°	93,7944°
Interfaz	-49,7191°	44,0975°	93,8166°
Fotografías	-49°	44°	93,0000°
NX	-49,71908°	44,097537°	93,816618°

Fuente: Autor.

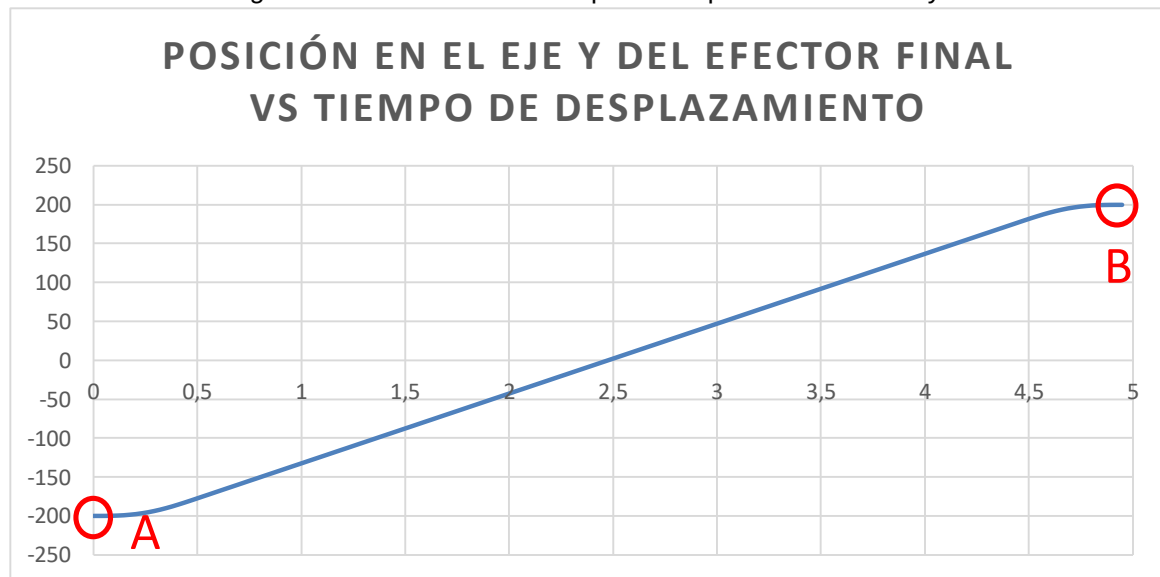
De la Tabla 10 se puede determinar que la diferencia de posición en el motor 3 es incluso más pequeña que la del motor 1, teniendo una diferencia de $0,000018^\circ$ entre lo obtenido en la interfaz y el NX, además de que la diferencia entre la interfaz y el GalilTools es de $0,0222^\circ$ y finalmente la diferencia entre la interfaz y lo que arroja el análisis de los motores es de $0,8166^\circ$ siendo esta la más grande en este motor,

posiblemente por errores causados por no tener una base fija al momento de grabar y tomar las tomas, dando paso a que la persona que hizo estas tareas se moviera o cambiara ligeramente el ángulo de la cámara.

Ya con la comparación de la posición de los motores el siguiente paso a comprobar sería la correcta aplicación de los perfiles suavizados a las trayectorias y motores, siendo esto un requisito importante al momento de generar las trayectorias del robot delta.

En la Figura 42 se analizará la posición del efector final a lo largo del movimiento visto desde el eje Y, esto se debe a que es el eje que más se aprecia el cambio de posición en la trayectoria. De esta figura se puede apreciar como en el punto A, se encuentra en su posición inicial de -200 y tiempo 0, lográndose ver como a través del tiempo atraviesa el 0 de su centro coordenado pasando de negativo a positivo, llegando a su posición final en 200 en un tiempo de 4.95 segundos.

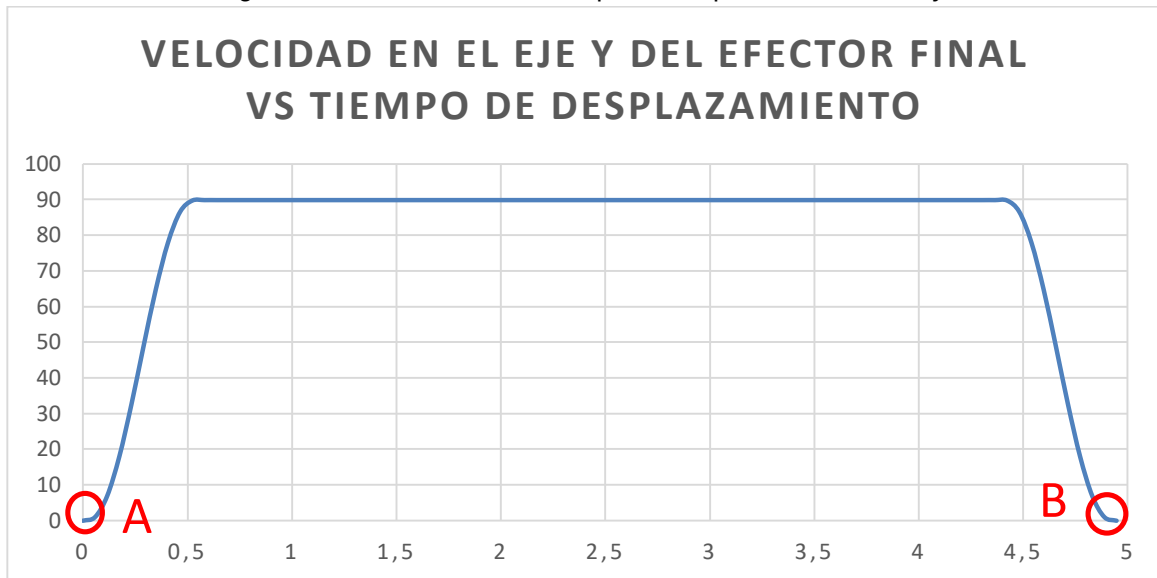
Figura 42. Posición Y vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

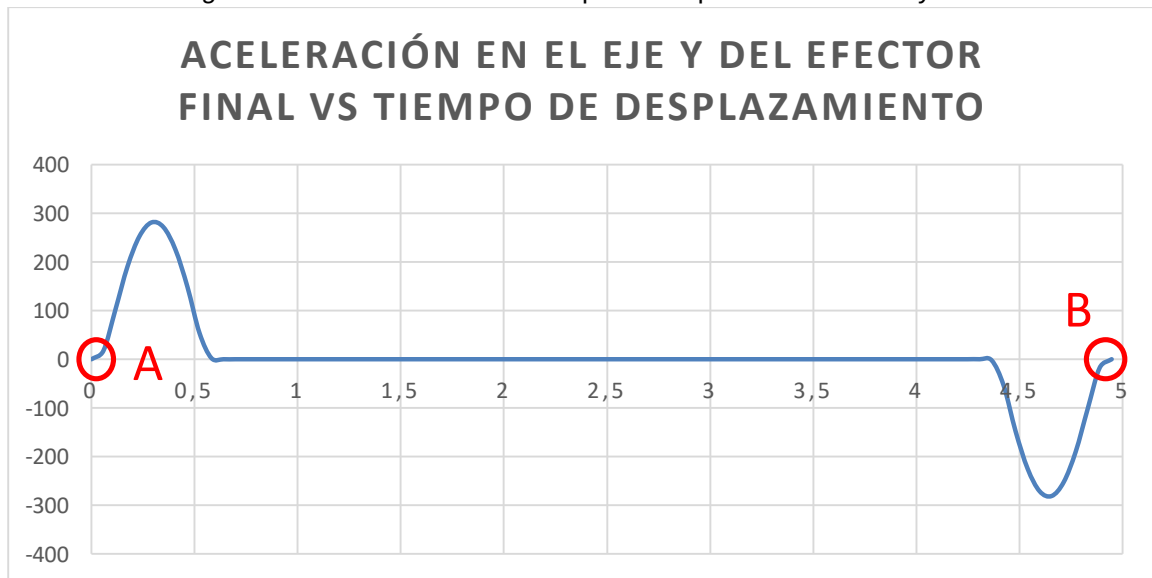
En la Figura 43 estará enfocada en la velocidad que obtiene el efector final a lo largo de la trayectoria, en donde se puede apreciar como inicialmente sube la velocidad senoidalmente hasta llegar a un punto, este es la velocidad cruceo o velocidad máxima del sistema, luego de un periodo su velocidad descende hasta 0 igualmente senoidalmente coincidiendo en el tiempo de 4.95 segundos.

Figura 43. Velocidad Y vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

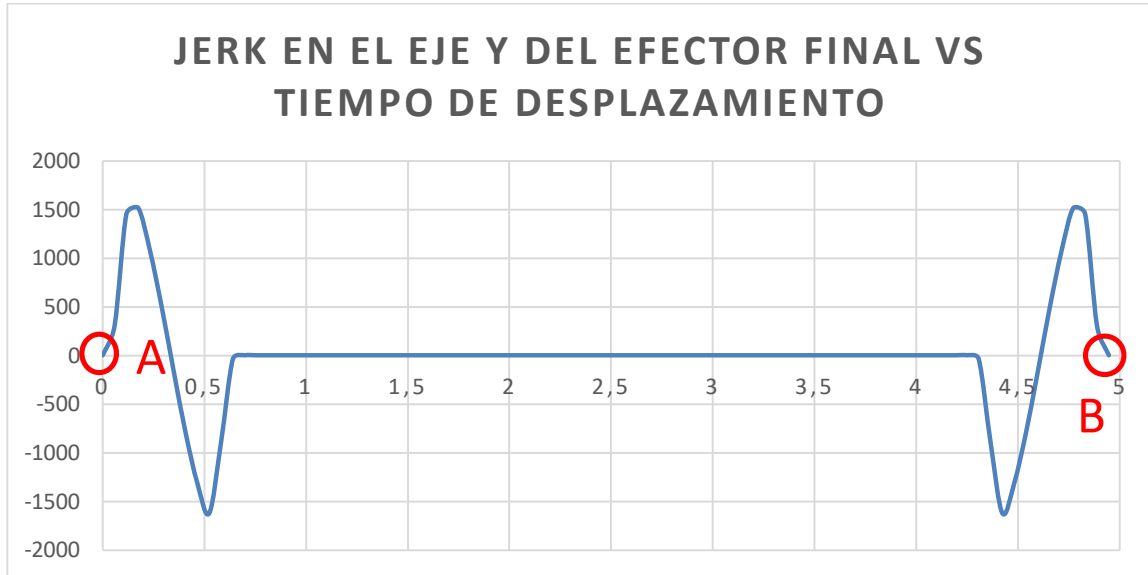
Figura 44. Aceleración Y vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

En la Figura 44 se logrará apreciar la aceleración del efector final, en donde en su primer tramo se ve una aceleración suavizada, llegando a el mismo punto de la velocidad crucero, con lo cual para este caso la aceleración es 0 y finalmente se llega al punto de desaceleración del sistema el cual cumple de igual manera con el perfil suavizado.

Figura 45. Jerk Y vs tiempo de desplazamiento en trayectoria 3

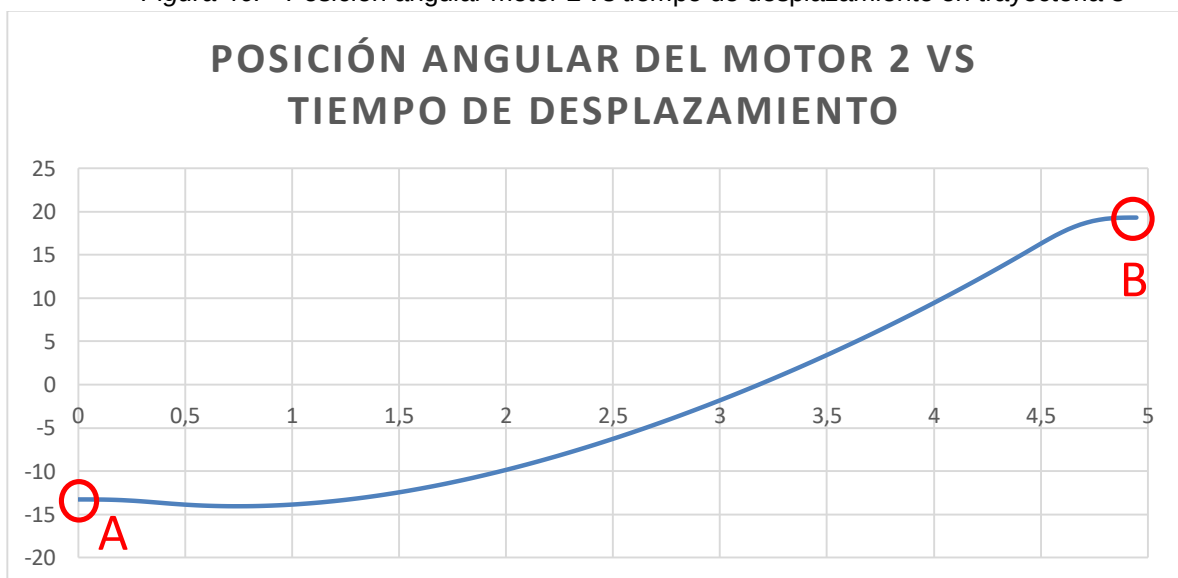


Fuente: Autor.

Finalmente en la Figura 45 se logra apreciar el Jerk, el cual es la derivada de la aceleración respecto al tiempo, con lo cual podemos ver como en la etapa de aceleración el jerk se comporta de forma esperada, siendo una curva senoidal y evitando los picos infinitos, al igual que en la aceleración en la etapa de velocidad crucero se mantiene en 0 y finaliza en su etapa de desaceleración con una curva senoidal para terminar en 0 en el tiempo de 4.95 segundos.

Después de pasar los valores de todos los perfiles suavizado por la cinemática inversa se crean una nueva nube de puntos, la cual está ligada a los 3 motores, por cuestiones prácticas se analizará únicamente el motor 2.

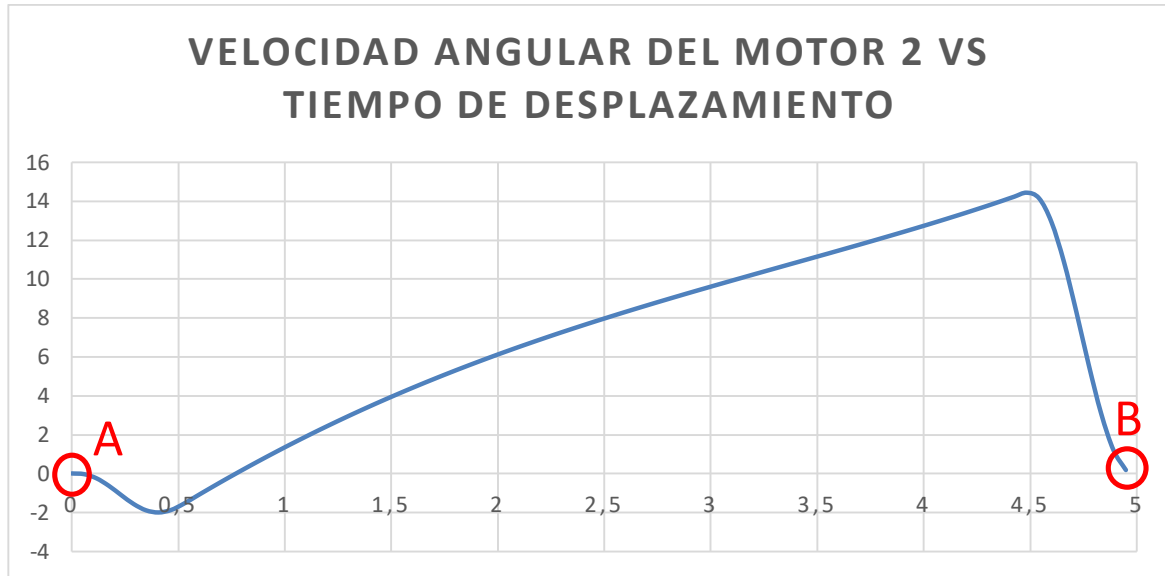
Figura 46. Posición angular motor 2 vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

En la Figura 46 se puede ver la posición angular del motor 2 a lo largo del tiempo, en donde se aprecia su comportamiento suavizado en todo el movimiento, la apariencia de este no es idéntico al visto en el efector y esto se debe a que al pasar por la cinemática inversa se ve afectado por su función senoidal pero no interfiere en su propósito.

Figura 47. Velocidad angular motor 2 vs tiempo de desplazamiento en trayectoria 3

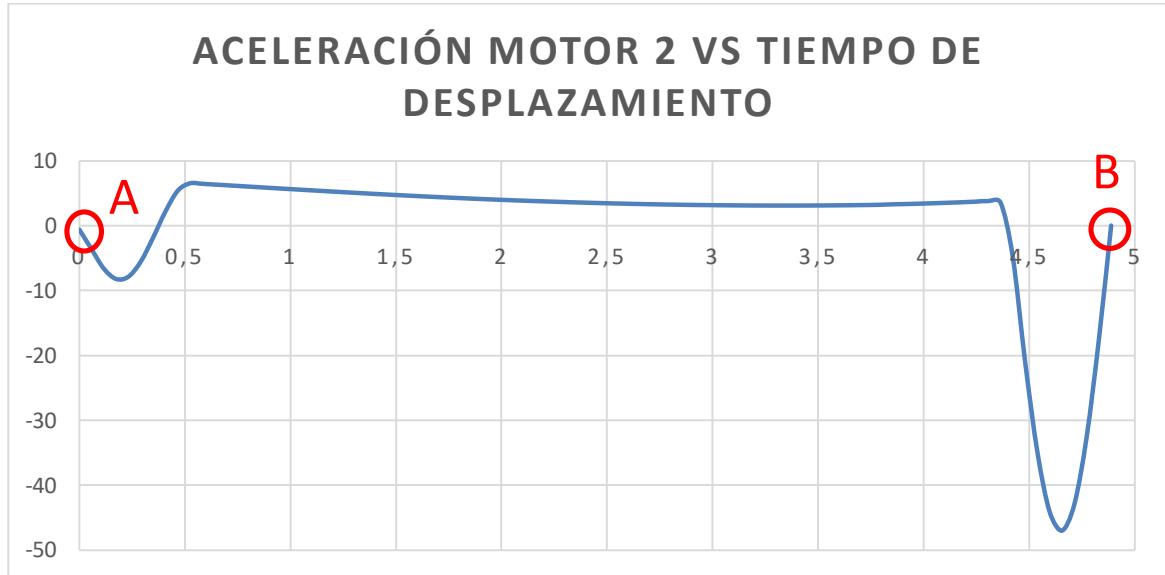


Fuente: Autor.

En la Figura 47 se aprecia la velocidad angular que tiene el motor 2 a lo largo del tiempo, lográndose ver su inicialmente como su velocidad desciende senoidalmente hasta valores negativos, esto se debe a que el motor está girando en el sentido contrario a lo usual, luego se vería como empieza a subir constantemente hasta llegar a su velocidad máxima, en donde pasa a su etapa final en donde su velocidad desciende hasta 0 igualmente senoidalmente coincidiendo en el tiempo de 4.95 segundos.

De la Figura 48 se puede ver como en su primer tramo se genera una desaceleración y aceleración, esto se debe al cambio de dirección como se nombró anteriormente, pero esto lo hace de manera suavizada, evitando que este cambio de dirección genere saltos en las aceleraciones de 0 a valores gigantescos, en su etapa final podemos ver una desaceleración más fuerte ya que detiene el motor, pero esto se hace controladamente como se espera de un perfil suavizado.

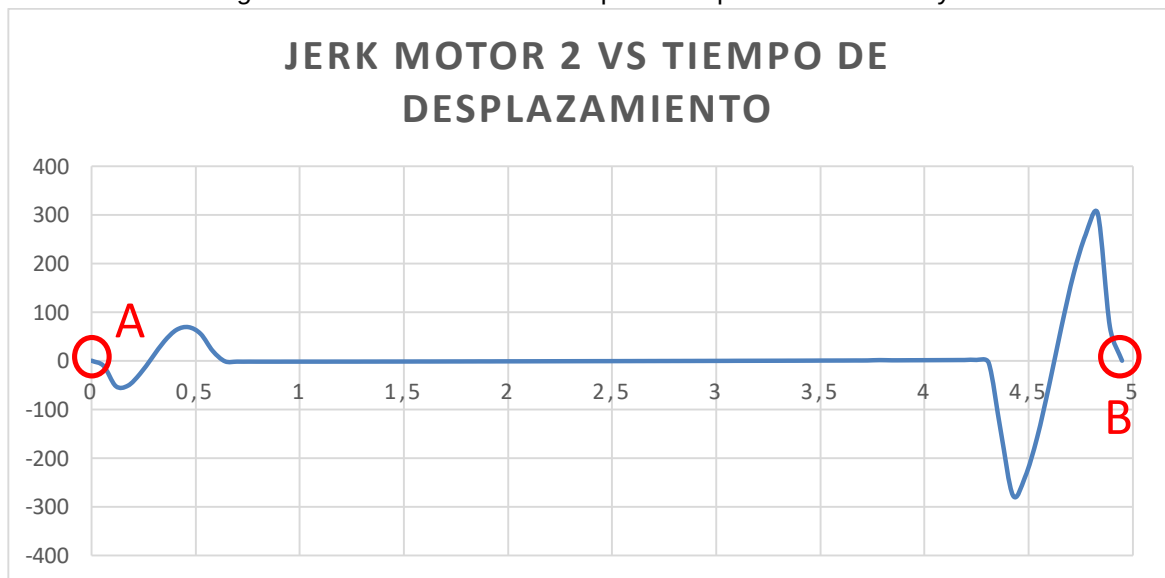
Figura 48. Aceleración motor 2 vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

Finalmente en la Figura 49 apreciamos el Jerk generado en el motor 2, en este al igual que en todo lo anterior su etapa inicial se ve afectada por la dirección en la que gira el motor dando una curva similar a la de su etapa final, pero con distinta magnitud, esto se debe a que su aceleración es mayor en su última etapa.

Figura 49. Jerk motor 2 vs tiempo de desplazamiento en trayectoria 3



Fuente: Autor.

Con los resultados obtenidos a lo largo de las gráficas se observa que el modelo de perfiles suavizados fue bien aplicado en los movimientos del robot, por lo que se

aprecia como en todas tienen las 3 etapas siendo la aceleración y desaceleración lo que más resalta de estas.

Para apreciar más gráficas de perfiles suavizados puede ir a ANEXO G: OTRAS TRAYECTORIAS.

6. CONCLUSIONES

- La detección de las variables de entrada y salida del robot fue cumplida en su totalidad dando la posibilidad de parametrizar y generar así posibles aplicaciones a diferentes prototipos de robots paralelos tipo delta siempre que cumplan con la misma geometría y cuenten con el mismo controlador.
- Los requerimientos primordiales para generar trayectorias en un robot de arquitectura paralela fueron analizados y determinados de manera correcta.
- La implementación del aplicativo que se realizó mediante el software Visual Studio es eficiente para poder definir movimientos de trayectorias lineales en el espacio.
- Las variables de entrada y salida, al igual que los requerimientos se lograron aplicar en su totalidad dentro del aplicativo.
- Los valores que arroja el aplicativo corresponden a la posición del robot teniendo en cuenta la cinemática inversa y a su gemelo virtual.
- Los perfiles suavizados fueron bien implementados en las trayectorias, logrando comprobarse en la posición, velocidad, aceleración y jerk en movimientos analizados desde un eje en concreto.
- Los perfiles suavizados en los movimientos de los motores tienen un excelente comportamiento, logrando así prolongar la vida útil de los componentes del robot.
- A pesar de que no está el ensamble completo del robot, gracias a las fotografías, videos tomados y comparaciones con el gemelo virtual se puede tener la certeza de que el movimiento de los actuadores convergerá correctamente en la posición del efector final, la exactitud de esto dependerá de la calidad del montaje final del robot tipo delta.

7. RECOMENDACIONES

- Siempre que termine de ejecutar una trayectoria y antes de apagarlo llevar el robot al cero de máquina ya que los encoders no reconocen su posición luego de que el controlador sea apagado, con lo cual podría generar problemas en un futuro ensamble del robot.
- Se recomienda un estudio más a profundidad sobre los Buffers que tiene el controlador, con la finalidad de enviarle más puntos de posición, velocidad y tiempo, logrando una exactitud aún más amplia de la que tiene actualmente.
- Ya que la interfaz actual aún no tiene un método capaz de realizar una trayectoria compleja, se recomienda implementar para mejorar el rendimiento del robot.

BIBLIOGRAFÍA

- [1] R. Sanchis, R. Julio, and A. Carlos, Automatización industrial. 2010.
- [2] Rafael Cisneros Limón, "Modelo matemático de un robot paralelo de seis grados de libertad," no. May 2006, p. 251, 2006, [Online]. Available: https://www.researchgate.net/publication/37613187_Modelo_Matematico_de_un_Robot_Paralelo_de_Seis_Grados_de_Libertad.
- [3] J. D. N. Y. William Zamir Jr. Muñoz Camelo, «Diseño mecánico de un robot de arquitectura paralela tipo delta de 3 dof,» Bogotá, 2014.
- [4] U. G. Zaragoza, «Torque analyzis of a three translational degrees of freedom parallel manipulator,» Bogotá, 2012.
- [5] H. C. J. G. Jaime Cortez, «Diseño óptimo dimensional de una máquina -robot de arquitectura paralela tipo delta para aplicaciones pick and place,» CIMM, 2013.
- [6] J. E. Marulanda, «Puesta en marcha virtual aplicada al robot delta de la Universidad Santo Tomás,» Bogotá, 2015.
- [7] Córdoba, E. Paternina, J. García, J. «Control de Movimiento en Manufactura Automatización CNC Fundamentos de Diseño y Modelamiento experimental,» Bogotá, 2013.
- [8] Galil, «GalilTools Communication Library,» [En línea]. Available: <https://www.galil.com/download/manual/galiltools/hello.html>.
- [9] Galil, «gclib,» [En línea]. Available: https://www.galil.com/sw/pub/all/doc/gclib/html/cs_examples_instructions.html.
- [10] Galil, «DMC-41x3 User Manual,» [En línea]. Available: https://www.a2v.fr/galil/doc/cartes/dmc-41x3/man_41x3.pdf
- [11] Ariza Cifuentes, C. D. (2022). Puesta En Marcha Virtual, Del Robot Delta De La Universidad Santo Tomás, Utilizando El Módulo Mechatronics Concept Designer Del Software Nx Universidad Santo Tomás.
- [12] Microsoft «A tour of the C# language,» [En línea]. Available:<https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>

ANEXO A: SOLUCIÓN CINEMÁTICA DIRECTA

$$\begin{aligned}
 (S.1) \quad & (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = r_1^2 \\
 & (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = r_2^2 \\
 & (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = r_3^2
 \end{aligned}$$

$$\begin{aligned}
 (S.2) \quad & (x - x_1)^2 + (y)^2 + (z - z_1)^2 = r^2 \\
 & (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = r^2 \\
 & (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = r^2
 \end{aligned}$$

De este sistema de ecuaciones las incógnitas son x , y , z , las cuales sería nuestro punto de intersección.

Para la solución de este sistema de ecuaciones, primero se expandirá cada término cuadrático, dejando todos los valores cuadráticos a un lado de la igualdad, mientras que en el otro se dejarán lo demás. Teniendo esto procederá a restar la tercera ecuación de la primera de (S.2) y la tercera de la segunda de (S.2) con la finalidad de eliminar los valores cuadráticos de las incógnitas

$$(E.1) \quad a_{11}x + a_{12}y + a_{13}z = b_1$$

$$(E.2) \quad a_{21}x + a_{22}y + a_{23}z = b_2$$

Donde:

$$\begin{aligned}
 a_{11} &= 2(x_1 - x_3) & a_{21} &= 2(x_2 - x_3) \\
 a_{12} &= 2(y_1 - y_3) & a_{22} &= 2(y_2 - y_3) \\
 a_{13} &= 2(z_1 - z_3) & a_{23} &= 2(z_2 - z_3)
 \end{aligned}$$

$$\begin{aligned}
 b_2 &= x_2^2 + y_2^2 + z_2^2 - x_3^2 - y_3^2 - z_3^2 \\
 b_1 &= x_1^2 + y_1^2 + z_1^2 - x_3^2 - y_3^2 - z_3^2
 \end{aligned}$$

Solucionando (E.1) y (E.2) para y obtiene:

$$(E.3) \quad y = \frac{b_1}{a_{12}} - \frac{a_{13}}{a_{12}}z - \frac{a_{11}}{a_{12}}x$$

$$(E.4) \quad y = \frac{b_2}{a_{22}} - \frac{a_{23}}{a_{22}}z - \frac{a_{21}}{a_{22}}x$$

Igualando (E.3) con (E.4) se eliminará y obteniendo una ecuación de tipo $x = f(z)$

$$\frac{b_1}{a_{12}} - \frac{a_{13}}{a_{12}}z - \frac{a_{11}}{a_{12}}x = \frac{b_2}{a_{22}} - \frac{a_{23}}{a_{22}}z - \frac{a_{21}}{a_{22}}x$$

$$(E.5) \quad x = a_4z + a_5$$

Donde:

$$\begin{aligned} a_1 &= \frac{a_{11}}{a_{12}} - \frac{a_{21}}{a_{22}} & a_4 &= \frac{-a_2}{a_1} \\ a_2 &= \frac{a_{13}}{a_{12}} - \frac{a_{23}}{a_{22}} & a_5 &= \frac{-a_3}{a_1} \\ a_3 &= \frac{b_2}{a_{22}} - \frac{b_1}{a_{12}} \end{aligned}$$

Ahora se sustituye (E.5) en (E.4) para eliminar x obteniendo una ecuación de tipo $y = f(z)$

$$\begin{aligned} y &= \frac{b_2}{a_{22}} - \frac{a_{23}}{a_{22}}z - \frac{a_{21}}{a_{22}}(a_4z + a_5) \\ y &= \frac{b_2}{a_{22}} - \frac{a_{23}}{a_{22}}z - \frac{a_{21}}{a_{22}}a_4z - \frac{a_{21}}{a_{22}}a_5 \end{aligned}$$

$$(E.6) \quad y = a_6z + a_7$$

Donde:

$$a_6 = \frac{-a_{23} - a_{21}a_4}{a_{22}} \quad a_7 = \frac{b_2 - a_{21}a_5}{a_{22}}$$

$$(a_4z + a_5 - x_1)^2 + (a_6z + a_7 - y_1)^2 + (z - z_1)^2 = r^2$$

Al expandir esta ecuación se obtiene

$$(E.7) \quad az^2 + bz + c = 0$$

Donde:

$$\begin{aligned}a &= a_4^2 + 1 + a_6^2 \\b &= 2a_4(a_5 - x_1) - 2y_1 + 2a_6(a_7 - z_1) \\c &= a_5(a_5 - 2x_1) + a_7(a_7 - 2z_1) + x_1^2 + y_1^2 + z_1^2 - r^2\end{aligned}$$

Llegando a:

$$(E.8) \quad z_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Para completar la solución se sustituyeron los valores de (E.8) en (E.6) y (E.5) para obtener los valores de “x” y “y” generando así dos soluciones para cada ecuación siendo estas.

$$(E.9) \quad y_{\pm} = a_6 z_{\pm} + a_7$$

$$(E.10) \quad x_{\pm} = a_4 z_{\pm} + a_5$$

Teniendo en cuenta que la solución tiene dos puntos de intersección las esferas se tomará el punto más negativo siendo la solución:

$$P = [x_- \ y_- \ z_-]$$

Para la comprobación de esto se aplicó la matriz P_b en (S.1) obteniendo:

$$\begin{aligned}(x - (\cos \alpha_1)(R + l_a \cos q_1))^2 + (y - (-(R + l_a \cos q_1)(\sin \alpha_1)))^2 + (z - (-l_a \sin q_1))^2 &= l_b^2 \\(x - (\cos \alpha_2)(R + l_a \cos q_2))^2 + (y - (-(R + l_a \cos q_2)(\sin \alpha_2)))^2 + (z - (-l_a \sin q_2))^2 &= l_b^2 \\(x - (\cos \alpha_3)(R + l_a \cos q_3))^2 + (y - (-(R + l_a \cos q_3)(\sin \alpha_3)))^2 + (z - (-l_a \sin q_3))^2 &= l_b^2\end{aligned}$$

Y solucionando este sistema de ecuaciones en el software *Wolfram Mathematica* se logra obtener valores para las tres incógnitas.

Figura A. Valores constantes y variables conocidas.

$$\begin{aligned}R &= 50 & q3 &= 0 \\l_a &= 310 & a1 &= 0 \\l_b &= 840 & a2 &= \left(\frac{2 * \pi}{3}\right) \\q1 &= 0 & a3 &= \left(\frac{4 * \pi}{3}\right) \\q2 &= 0\end{aligned}$$

Fuente Autor

Figura B. Código utilizado para llegar a la solución en Wolfram Mathematica

```

In[ ]:= Solve[ { (x - (Cos[a1] * (la * Cos[q1] + R)))^2 + (y - (- (R + la * Cos[q1])) * Sin[a1])^2 + (z - (-la * Sin[q1]))^2 == lb^2,
[resuelve]      [coseno]      [coseno]      [coseno]      [seno]      [seno]
(x - (Cos[a2] * (la * Cos[q2] + R)))^2 + (y - (- (R + la * Cos[q2])) * Sin[a2])^2 + (z - (-la * Sin[q2]))^2 == lb^2,
[coseno]      [coseno]      [coseno]      [seno]      [seno]
(x - (Cos[a3] * (la * Cos[q3] + R)))^2 + (y - (- (R + la * Cos[q3])) * Sin[a3])^2 + (z - (-la * Sin[q3]))^2 == lb^2}, {x, y, z}
[coseno]      [coseno]      [coseno]      [seno]      [seno]

```

Fuente Autor

Figura C. Solución del código ingresado en Wolfram Mathematica

```

In[ ]:= N[ { {x -> 0, y -> 0, z -> -120 * Sqrt[49 - 9 Cos[240]^2 - 9 Sin[240]^2]}, {x -> 0, y -> 0, z -> 120 * Sqrt[49 - 9 Cos[240]^2 - 9 Sin[240]^2]} } ]
[valor numérico]

Out[ ]:= { {x -> 0., y -> 0., z -> -758.947}, {x -> 0., y -> 0., z -> 758.947} }

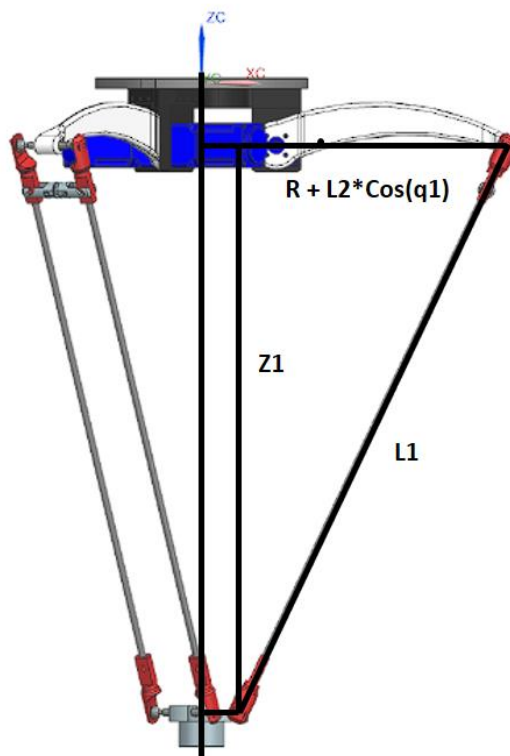
```

Fuente Autor

Como se puede observar en la Figura C el valor más negativo de la solución concuerda con el valor en z del efector en el caso de que los tres ángulos de los motores son 0.

A.1 Singularidades

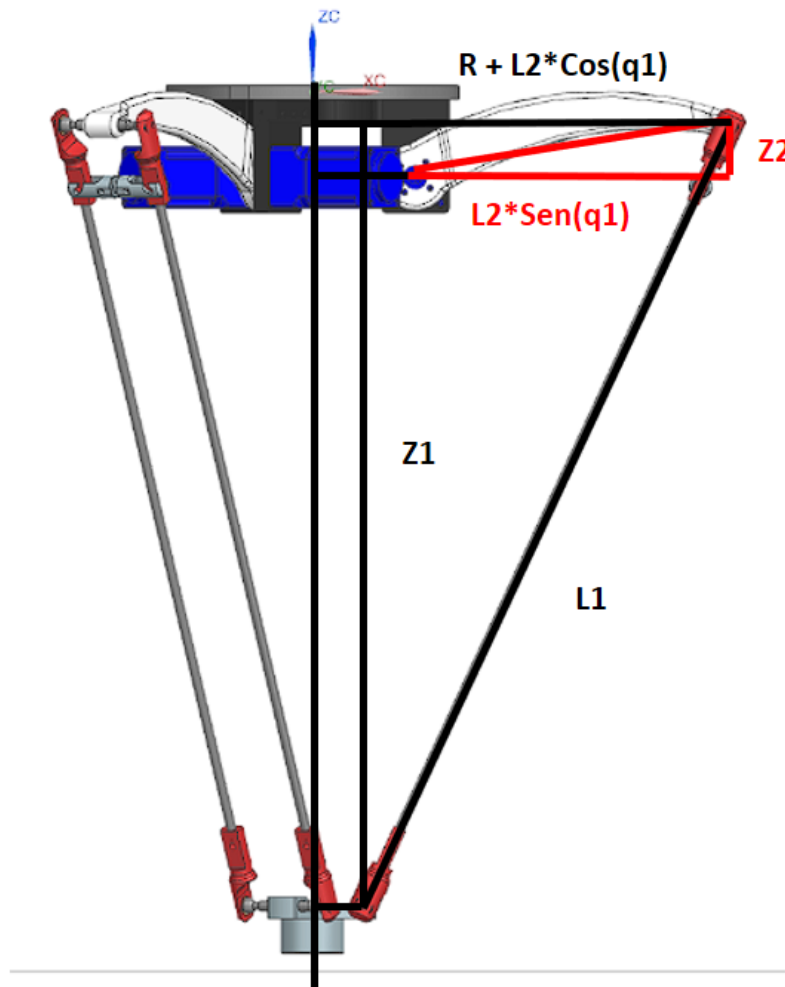
Figura D. Analisis geométrico caso 1 de Cinemática directa



Fuente: Autor.

En la Figura D se posicionan los 3 motores en el valor de 0, y se hace un análisis geométrico de la posición de los componentes del robot en donde se generará un triángulo en donde uno de sus catetos e hipotenusa son conocidos y el problema radica en encontrar el cateto faltante el cual será la posición de Z.

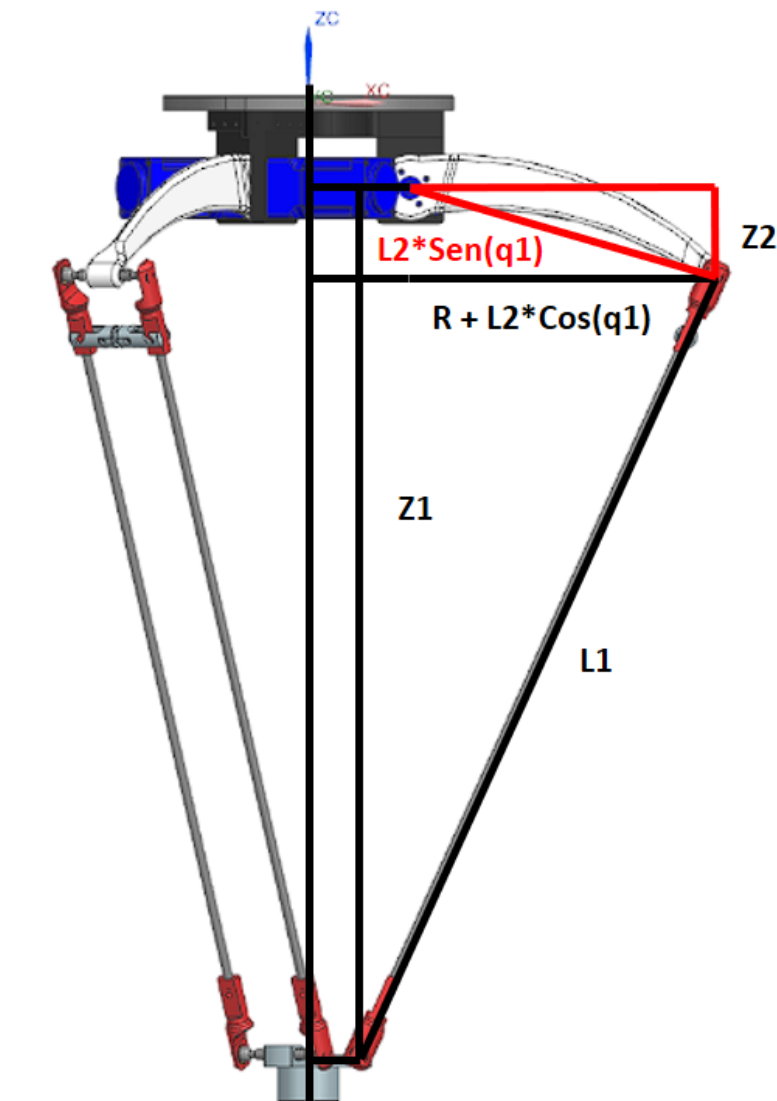
Figura E. Análisis geométrico caso 2 de Cinemática directa



Fuente: Autor.

En la Figura E se posicionan los 3 motores en valores superiores a 0 e iguales, y se hace un análisis geométrico de la posición de los componentes del robot en donde se generarán dos triángulos en donde en el primero uno de sus catetos e hipotenusa son conocidos y el problema radica en encontrar el cateto faltante y en el segundo se necesita la posición de un solo cateto el cual es conocido, la diferencia de estos catetos será la posición de Z.

Figura F. Analisis geométrico caso 3 de Cinemática directa



Fuente: Autor.

En la Figura F se posicionan los 3 motores en valores inferiores a 0 e iguales, y se hace un análisis geométrico de la posición de los componentes del robot en donde se generán dos triángulos en donde en el primero uno de sus catetos e hipotenusa son conocidos y el problema radica en encontrar el cateto faltante y en el segundo se necesita la posición de un solo cateto el cual es conocido, la suma de estos catetos será la posición de Z.

ANEXO B: PERFILES SUAVIZADOS

En la generación de trayectorias de movimientos de máquinas y robots de precisión y velocidades altas, un perfil trapezoidal de velocidad no es el adecuado ya que este no tiene unas transiciones o comportamientos en cuanto a velocidades y aceleraciones suaves. Con lo cual se aplica a este perfil trapezoidal la función coseno para así conseguir que tanto su velocidad y su aceleración sea continua y suave:

Partiendo de las ecuaciones de cada tramo (B.1) obtenidas del libro *“Control de movimiento en manufactura. Automatización CNC fundamentos de diseño y modelamiento experimental”*

$$\begin{aligned} v_1 &= -c_1 * \left(\cos \left(\frac{\pi * t}{t_1} \right) - 1 \right); v_2 = c_2; \\ v_3 &= c_3 * \left(\cos \left(\frac{\pi * (t - t_2)}{t_2 - 1} \right) + 1 \right) \end{aligned} \quad (B.1)$$

Se procede a desarrollar el modelo integrando las funciones a tramos de la velocidad para encontrar las ecuaciones de posición en donde se asegura que sus velocidades y aceleraciones serán continuas y suaves.

$$\int_0^t v_i * dt \quad \text{para } i = 1, 2 \text{ y } 3 \quad (B.2)$$

$$\begin{aligned} x_1 &= c_1 * \left(t - \frac{t_1}{\pi} * \sin \left(\frac{\pi * t}{t_1} \right) \right) + k_1; \quad x_2 = c_2 * t + k_2; \\ x_3 &= c_3 * \left(t + \frac{(t_2 - 1)}{\pi} * \sin \left(\frac{\pi * (t - t_2)}{t_2 - 1} \right) \right) + k_3 \end{aligned} \quad (B.3)$$

Solucionando estas ecuaciones para encontrar las constantes de forma y de integración se obtienen las siguientes funciones para el perfil suavizado de movimiento.

$$\begin{aligned} c_1 &= \frac{(1 - k_1)}{(1 + t_2 - t_1)}; \quad c_2 = 2 * c_1; \quad c_3 = c_1 \\ k_1 &= 0; \quad k_2 = k_1 - t_1 * c_1; \quad k_3 = k_1 + c_1 * (t_2 - t_1) \end{aligned} \quad (B.4)$$

Al reemplazar las ecuaciones B.4 en B.3 se obtiene la solución para el perfil suavizado.

ANEXO C: CÓDIGO INTERFAZ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Interfaz2_0
{
    public partial class Form1 : Form
    {
        Galil.Galil g = new Galil.Galil();
        gclib gclib = new gclib();

        public double delX, delY, delZ, N, tiempodespla, n, PosicionA, PosicionB,
        PosicionC, L1, L2, Ra, Rb, R, znd, xnd, ynd, q1, q2, q3, Rcontrol;
        public double[,] perS1;
        public double[] CIq1, CIq2, CIq3,T, P1, P2, P3, V1, V2, V3, CIq11, CIq22,
        CIq33;
        private static System.Timers.Timer timer_TCPSendData;
        private static System.Timers.Timer timer_TCPSendData1;

        public Form1()
        {
            InitializeComponent();
            textBox1.Text = "- - -";
            textBox2.Text = "- - -";
            textBox3.Text = "- - -";
            textBox4.Text = "- - -";
            textBox5.Text = "- - -";
            textBox6.Text = "- - -";
            button2.Enabled = false;
            xir.Enabled = false;
            yir.Enabled = false;
            zir.Enabled = false;
            Velocidad.Enabled = false;
            L1 = 840;
            L2 = 310;
            Ra = 100;
            Rb = 50;
            R = Ra - Rb;
            Conexion.UseVisualStyleBackColor = true;
            CheckForIllegalCrossThreadCalls = false;
            gclib.GOpen("192.168.1.10");
        }

        private void Conexion_Click(object sender, EventArgs e)
        {
            g.address = "192.168.1.10";
            this.Text = g.connection();

            Conexion.UseVisualStyleBackColor = false;
            button2.Enabled = true;
        }
    }
}
```

```

private void button2_Click(object sender, EventArgs e)
{
    double PosicionA = g.commandValue("TPA");
    double PosicionB = g.commandValue("TPB");
    double PosicionC = g.commandValue("TPC");

    if (PosicionA == 0 & PosicionB == 0 & PosicionC == 0)
    {
        gclib.GCommand("SH ABC");
        gclib.GCommand("PRA=" + PosicionA);
        gclib.GCommand("PRB=" + PosicionB);
        gclib.GCommand("PRC=" + PosicionC);
        gclib.GCommand("SP 10000");
        gclib.GCommand("BG ABC");
    }
    else
    {
        if (PosicionA > 0 & PosicionB > 0 & PosicionC > 0)
        {
            double PosicionA1 = -PosicionA;
            double PosicionB1 = -PosicionB;
            double PosicionC1 = -PosicionC;
            gclib.GCommand("PRA=" + PosicionA1);
            gclib.GCommand("PRB=" + PosicionB1);
            gclib.GCommand("PRC=" + PosicionC1);
            gclib.GCommand("SP 10000");
            gclib.GCommand("BG ABC");
        }
        else
        {
            double PosicionA1 = PosicionA;

            double PosicionB1 = PosicionB;
            double PosicionC1 = PosicionC;
            gclib.GCommand("PRA=" + PosicionA1);
            gclib.GCommand("PRB=" + PosicionB1);
            gclib.GCommand("PRC=" + PosicionC1);
            gclib.GCommand("SP 10000");
            gclib.GCommand("BG ABC");
        }
    }
}

SetTimer_TCPSendData();

irposi.Enabled = true;
xir.Enabled = true;
yir.Enabled = true;
zir.Enabled = true;
Velocidad.Enabled = true;
}
private void irposi_Click(object sender, EventArgs e)
{
    double xf = Convert.ToDouble(xir.Text);
    double yf = Convert.ToDouble(yir.Text);
    double zf = Convert.ToDouble(zir.Text);

    double vel = Convert.ToDouble(Velocidad.Text);
    double xi = Convert.ToDouble(textBox1.Text);
}

```

```

double yi = Convert.ToDouble(textBox2.Text);
double zi = Convert.ToDouble(textBox3.Text);

Interpolacion interpo = new Interpolacion();
interpo.Inter(xf, yf, zf, vel, xi, yi, zi);

delX = interpo.vecdx;
delY = interpo.vecdy;
delZ = interpo.vecdz;
N = interpo.N;
n = N - 1;
tiempodespla = interpo.tiemdespl;

perfilS perfilSuavizado = new perfilS();
perfilSuavizado.perfilSuavizado(delX, delY, delZ, N, tiempodespla, zi, zf,
xi, xf, yi, yf);

perS1 = perfilSuavizado.perS;

Cinematica_inver_1 CinematicInver = new Cinematica_inver_1();
CinematicInver.CineInver(perS1, N, L1, L2, Ra, Rb);
CIq1 = CinematicInver.CIq1;
CIq2 = CinematicInver.CIq2;
CIq3 = CinematicInver.CIq3;
CIq11 = CinematicInver.CIq11;
CIq22 = CinematicInver.CIq22;
CIq33 = CinematicInver.CIq33;

PVT pvt = new PVT();
pvt.pvt(CIq1, CIq2, CIq3, perS1, N);
P1 = pvt.P1;
P2 = pvt.P2;
P3 = pvt.P3;
V1 = pvt.V1;
V2 = pvt.V2;
V3 = pvt.V3;
T = pvt.T;

Rcontrol = Math.Sqrt(Math.Pow(xf, 2) + Math.Pow(yf, 2));
if (565 <= Rcontrol || zf < -850 || zf > -600)
{
    textBox1.Text = "Fuera de rango";
    textBox2.Text = "Fuera de rango";
    textBox3.Text = "Fuera de rango";
}
else
{
    gclib.GOpen("192.168.1.10");
    gclib.GCommand("SH ABC");

    gclib.GCommand("PVA=" + P1[1] + "," + V1[1] + "," + T[1]);
    gclib.GCommand("PVA=" + P1[2] + "," + V1[2] + "," + T[2]);
    gclib.GCommand("PVA=" + P1[3] + "," + V1[3] + "," + T[3]);
    gclib.GCommand("PVA=" + P1[4] + "," + V1[4] + "," + T[4]);
    gclib.GCommand("PVA=" + P1[5] + "," + V1[5] + "," + T[5]);
    gclib.GCommand("PVA=" + P1[6] + "," + V1[6] + "," + T[6]);
    gclib.GCommand("PVA=" + P1[7] + "," + V1[7] + "," + T[7]);
    gclib.GCommand("PVA=" + P1[8] + "," + V1[8] + "," + T[8]);
    gclib.GCommand("PVA=" + P1[9] + "," + V1[9] + "," + T[9]);
}

```



```

gclib.GCommand("PVA=" + P1[72] + "," + V1[72] + "," + T[72]);
gclib.GCommand("PVA=" + P1[73] + "," + V1[73] + "," + T[73]);
gclib.GCommand("PVA=" + P1[74] + "," + V1[74] + "," + T[74]);
gclib.GCommand("PVA=" + P1[75] + "," + V1[75] + "," + T[75]);
gclib.GCommand("PVA=" + P1[76] + "," + V1[76] + "," + T[76]);
gclib.GCommand("PVA=" + P1[77] + "," + V1[77] + "," + T[77]);
gclib.GCommand("PVA=" + P1[78] + "," + V1[78] + "," + T[78]);
gclib.GCommand("PVA=" + P1[79] + "," + V1[79] + "," + T[79]);
gclib.GCommand("PVA=" + P1[80] + "," + V1[80] + "," + T[80]);
gclib.GCommand("PVA=" + P1[81] + "," + V1[81] + "," + T[81]);
gclib.GCommand("PVA=" + P1[82] + "," + V1[82] + "," + T[82]);
gclib.GCommand("PVA=" + P1[83] + "," + V1[83] + "," + T[83]);

```

```
//motor2//
```

```

gclib.GCommand("PVB=" + P2[1] + "," + V2[1] + "," + T[1]);
gclib.GCommand("PVB=" + P2[2] + "," + V2[2] + "," + T[2]);
gclib.GCommand("PVB=" + P2[3] + "," + V2[3] + "," + T[3]);
gclib.GCommand("PVB=" + P2[4] + "," + V2[4] + "," + T[4]);
gclib.GCommand("PVB=" + P2[5] + "," + V2[5] + "," + T[5]);
gclib.GCommand("PVB=" + P2[6] + "," + V2[6] + "," + T[6]);
gclib.GCommand("PVB=" + P2[7] + "," + V2[7] + "," + T[7]);
gclib.GCommand("PVB=" + P2[8] + "," + V2[8] + "," + T[8]);
gclib.GCommand("PVB=" + P2[9] + "," + V2[9] + "," + T[9]);
gclib.GCommand("PVB=" + P2[10] + "," + V2[10] + "," + T[10]);
gclib.GCommand("PVB=" + P2[11] + "," + V2[11] + "," + T[11]);
gclib.GCommand("PVB=" + P2[12] + "," + V2[12] + "," + T[12]);
gclib.GCommand("PVB=" + P2[13] + "," + V2[13] + "," + T[13]);
gclib.GCommand("PVB=" + P2[14] + "," + V2[14] + "," + T[14]);
gclib.GCommand("PVB=" + P2[15] + "," + V2[15] + "," + T[15]);
gclib.GCommand("PVB=" + P2[16] + "," + V2[16] + "," + T[16]);
gclib.GCommand("PVB=" + P2[17] + "," + V2[17] + "," + T[17]);
gclib.GCommand("PVB=" + P2[18] + "," + V2[18] + "," + T[18]);
gclib.GCommand("PVB=" + P2[19] + "," + V2[19] + "," + T[19]);
gclib.GCommand("PVB=" + P2[20] + "," + V2[20] + "," + T[20]);
gclib.GCommand("PVB=" + P2[21] + "," + V2[21] + "," + T[21]);
gclib.GCommand("PVB=" + P2[22] + "," + V2[22] + "," + T[22]);
gclib.GCommand("PVB=" + P2[23] + "," + V2[23] + "," + T[23]);
gclib.GCommand("PVB=" + P2[24] + "," + V2[24] + "," + T[24]);
gclib.GCommand("PVB=" + P2[25] + "," + V2[25] + "," + T[25]);
gclib.GCommand("PVB=" + P2[26] + "," + V2[26] + "," + T[26]);
gclib.GCommand("PVB=" + P2[27] + "," + V2[27] + "," + T[27]);
gclib.GCommand("PVB=" + P2[28] + "," + V2[28] + "," + T[28]);
gclib.GCommand("PVB=" + P2[29] + "," + V2[29] + "," + T[29]);
gclib.GCommand("PVB=" + P2[30] + "," + V2[30] + "," + T[30]);
gclib.GCommand("PVB=" + P2[31] + "," + V2[31] + "," + T[31]);
gclib.GCommand("PVB=" + P2[32] + "," + V2[32] + "," + T[32]);
gclib.GCommand("PVB=" + P2[33] + "," + V2[33] + "," + T[33]);
gclib.GCommand("PVB=" + P2[34] + "," + V2[34] + "," + T[34]);
gclib.GCommand("PVB=" + P2[35] + "," + V2[35] + "," + T[35]);
gclib.GCommand("PVB=" + P2[36] + "," + V2[36] + "," + T[36]);
gclib.GCommand("PVB=" + P2[37] + "," + V2[37] + "," + T[37]);
gclib.GCommand("PVB=" + P2[38] + "," + V2[38] + "," + T[38]);
gclib.GCommand("PVB=" + P2[39] + "," + V2[39] + "," + T[39]);
gclib.GCommand("PVB=" + P2[40] + "," + V2[40] + "," + T[40]);
gclib.GCommand("PVB=" + P2[41] + "," + V2[41] + "," + T[41]);
gclib.GCommand("PVB=" + P2[42] + "," + V2[42] + "," + T[42]);
gclib.GCommand("PVB=" + P2[43] + "," + V2[43] + "," + T[43]);
gclib.GCommand("PVB=" + P2[44] + "," + V2[44] + "," + T[44]);
gclib.GCommand("PVB=" + P2[45] + "," + V2[45] + "," + T[45]);
gclib.GCommand("PVB=" + P2[46] + "," + V2[46] + "," + T[46]);

```

```

gclib.GCommand("PVB=" + P2[47] + "," + V2[47] + "," + T[47]);
gclib.GCommand("PVB=" + P2[48] + "," + V2[48] + "," + T[48]);
gclib.GCommand("PVB=" + P2[49] + "," + V2[49] + "," + T[49]);
gclib.GCommand("PVB=" + P2[50] + "," + V2[50] + "," + T[50]);
gclib.GCommand("PVB=" + P2[51] + "," + V2[51] + "," + T[50]);
gclib.GCommand("PVB=" + P2[52] + "," + V2[52] + "," + T[51]);
gclib.GCommand("PVB=" + P2[53] + "," + V2[53] + "," + T[52]);
gclib.GCommand("PVB=" + P2[54] + "," + V2[54] + "," + T[53]);
gclib.GCommand("PVB=" + P2[55] + "," + V2[55] + "," + T[54]);
gclib.GCommand("PVB=" + P2[56] + "," + V2[56] + "," + T[55]);
gclib.GCommand("PVB=" + P2[57] + "," + V2[57] + "," + T[56]);
gclib.GCommand("PVB=" + P2[58] + "," + V2[58] + "," + T[57]);
gclib.GCommand("PVB=" + P2[59] + "," + V2[59] + "," + T[58]);
gclib.GCommand("PVB=" + P2[60] + "," + V2[60] + "," + T[60]);
gclib.GCommand("PVB=" + P2[61] + "," + V2[61] + "," + T[61]);
gclib.GCommand("PVB=" + P2[62] + "," + V2[62] + "," + T[62]);
gclib.GCommand("PVB=" + P2[63] + "," + V2[63] + "," + T[63]);
gclib.GCommand("PVB=" + P2[64] + "," + V2[64] + "," + T[64]);
gclib.GCommand("PVB=" + P2[65] + "," + V2[65] + "," + T[65]);
gclib.GCommand("PVB=" + P2[66] + "," + V2[66] + "," + T[66]);
gclib.GCommand("PVB=" + P2[67] + "," + V2[67] + "," + T[67]);
gclib.GCommand("PVB=" + P2[68] + "," + V2[68] + "," + T[68]);
gclib.GCommand("PVB=" + P2[69] + "," + V2[69] + "," + T[69]);
gclib.GCommand("PVB=" + P2[70] + "," + V2[70] + "," + T[70]);
gclib.GCommand("PVB=" + P2[71] + "," + V2[71] + "," + T[71]);
gclib.GCommand("PVB=" + P2[72] + "," + V2[72] + "," + T[72]);
gclib.GCommand("PVB=" + P2[73] + "," + V2[73] + "," + T[73]);
gclib.GCommand("PVB=" + P2[74] + "," + V2[74] + "," + T[74]);
gclib.GCommand("PVB=" + P2[75] + "," + V2[75] + "," + T[75]);
gclib.GCommand("PVB=" + P2[76] + "," + V2[76] + "," + T[76]);
gclib.GCommand("PVB=" + P2[77] + "," + V2[77] + "," + T[77]);
gclib.GCommand("PVB=" + P2[78] + "," + V2[78] + "," + T[78]);
gclib.GCommand("PVB=" + P2[79] + "," + V2[79] + "," + T[79]);
gclib.GCommand("PVB=" + P2[80] + "," + V2[80] + "," + T[80]);
gclib.GCommand("PVB=" + P2[81] + "," + V2[81] + "," + T[81]);
gclib.GCommand("PVB=" + P2[82] + "," + V2[82] + "," + T[82]);
gclib.GCommand("PVB=" + P2[83] + "," + V2[83] + "," + T[83]);

```

```
//motor3//
```

```

gclib.GCommand("PVC=" + P3[1] + "," + V3[1] + "," + T[1]);
gclib.GCommand("PVC=" + P3[2] + "," + V3[2] + "," + T[2]);
gclib.GCommand("PVC=" + P3[3] + "," + V3[3] + "," + T[3]);
gclib.GCommand("PVC=" + P3[4] + "," + V3[4] + "," + T[4]);
gclib.GCommand("PVC=" + P3[5] + "," + V3[5] + "," + T[5]);
gclib.GCommand("PVC=" + P3[6] + "," + V3[6] + "," + T[6]);
gclib.GCommand("PVC=" + P3[7] + "," + V3[7] + "," + T[7]);
gclib.GCommand("PVC=" + P3[8] + "," + V3[8] + "," + T[8]);
gclib.GCommand("PVC=" + P3[9] + "," + V3[9] + "," + T[9]);
gclib.GCommand("PVC=" + P3[10] + "," + V3[10] + "," + T[10]);
gclib.GCommand("PVC=" + P3[11] + "," + V3[11] + "," + T[11]);
gclib.GCommand("PVC=" + P3[12] + "," + V3[12] + "," + T[12]);
gclib.GCommand("PVC=" + P3[13] + "," + V3[13] + "," + T[13]);
gclib.GCommand("PVC=" + P3[14] + "," + V3[14] + "," + T[14]);
gclib.GCommand("PVC=" + P3[15] + "," + V3[15] + "," + T[15]);
gclib.GCommand("PVC=" + P3[16] + "," + V3[16] + "," + T[16]);
gclib.GCommand("PVC=" + P3[17] + "," + V3[17] + "," + T[17]);
gclib.GCommand("PVC=" + P3[18] + "," + V3[18] + "," + T[18]);
gclib.GCommand("PVC=" + P3[19] + "," + V3[19] + "," + T[19]);
gclib.GCommand("PVC=" + P3[20] + "," + V3[20] + "," + T[20]);

```

```

gclib.GCommand("PVC=" + P3[21] + "," + V3[21] + "," + T[21]);
gclib.GCommand("PVC=" + P3[22] + "," + V3[22] + "," + T[22]);
gclib.GCommand("PVC=" + P3[23] + "," + V3[23] + "," + T[23]);
gclib.GCommand("PVC=" + P3[24] + "," + V3[24] + "," + T[24]);
gclib.GCommand("PVC=" + P3[25] + "," + V3[25] + "," + T[25]);
gclib.GCommand("PVC=" + P3[26] + "," + V3[26] + "," + T[26]);
gclib.GCommand("PVC=" + P3[27] + "," + V3[27] + "," + T[27]);
gclib.GCommand("PVC=" + P3[28] + "," + V3[28] + "," + T[28]);
gclib.GCommand("PVC=" + P3[29] + "," + V3[29] + "," + T[29]);
gclib.GCommand("PVC=" + P3[30] + "," + V3[30] + "," + T[30]);
gclib.GCommand("PVC=" + P3[31] + "," + V3[31] + "," + T[31]);
gclib.GCommand("PVC=" + P3[32] + "," + V3[32] + "," + T[32]);
gclib.GCommand("PVC=" + P3[33] + "," + V3[33] + "," + T[33]);
gclib.GCommand("PVC=" + P3[34] + "," + V3[34] + "," + T[34]);
gclib.GCommand("PVC=" + P3[35] + "," + V3[35] + "," + T[35]);
gclib.GCommand("PVC=" + P3[36] + "," + V3[36] + "," + T[36]);
gclib.GCommand("PVC=" + P3[37] + "," + V3[37] + "," + T[37]);
gclib.GCommand("PVC=" + P3[38] + "," + V3[38] + "," + T[38]);
gclib.GCommand("PVC=" + P3[39] + "," + V3[39] + "," + T[39]);
gclib.GCommand("PVC=" + P3[40] + "," + V3[40] + "," + T[40]);
gclib.GCommand("PVC=" + P3[41] + "," + V3[41] + "," + T[41]);
gclib.GCommand("PVC=" + P3[42] + "," + V3[42] + "," + T[42]);
gclib.GCommand("PVC=" + P3[43] + "," + V3[43] + "," + T[43]);
gclib.GCommand("PVC=" + P3[44] + "," + V3[44] + "," + T[44]);
gclib.GCommand("PVC=" + P3[45] + "," + V3[45] + "," + T[45]);
gclib.GCommand("PVC=" + P3[46] + "," + V3[46] + "," + T[46]);
gclib.GCommand("PVC=" + P3[47] + "," + V3[47] + "," + T[47]);
gclib.GCommand("PVC=" + P3[48] + "," + V3[48] + "," + T[48]);
gclib.GCommand("PVC=" + P3[49] + "," + V3[49] + "," + T[49]);
gclib.GCommand("PVC=" + P3[50] + "," + V3[50] + "," + T[50]);
gclib.GCommand("PVC=" + P3[51] + "," + V3[51] + "," + T[50]);
gclib.GCommand("PVC=" + P3[52] + "," + V3[52] + "," + T[51]);
gclib.GCommand("PVC=" + P3[53] + "," + V3[53] + "," + T[52]);
gclib.GCommand("PVC=" + P3[54] + "," + V3[54] + "," + T[53]);
gclib.GCommand("PVC=" + P3[55] + "," + V3[55] + "," + T[54]);
gclib.GCommand("PVC=" + P3[56] + "," + V3[56] + "," + T[55]);
gclib.GCommand("PVC=" + P3[57] + "," + V3[57] + "," + T[56]);
gclib.GCommand("PVC=" + P3[58] + "," + V3[58] + "," + T[57]);
gclib.GCommand("PVC=" + P3[59] + "," + V3[59] + "," + T[58]);
gclib.GCommand("PVC=" + P3[60] + "," + V3[60] + "," + T[60]);
gclib.GCommand("PVC=" + P3[61] + "," + V3[61] + "," + T[61]);
gclib.GCommand("PVC=" + P3[62] + "," + V3[62] + "," + T[62]);
gclib.GCommand("PVC=" + P3[63] + "," + V3[63] + "," + T[63]);
gclib.GCommand("PVC=" + P3[64] + "," + V3[64] + "," + T[64]);
gclib.GCommand("PVC=" + P3[65] + "," + V3[65] + "," + T[65]);
gclib.GCommand("PVC=" + P3[66] + "," + V3[66] + "," + T[66]);
gclib.GCommand("PVC=" + P3[67] + "," + V3[67] + "," + T[67]);
gclib.GCommand("PVC=" + P3[68] + "," + V3[68] + "," + T[68]);
gclib.GCommand("PVC=" + P3[69] + "," + V3[69] + "," + T[69]);
gclib.GCommand("PVC=" + P3[70] + "," + V3[70] + "," + T[70]);
gclib.GCommand("PVC=" + P3[71] + "," + V3[71] + "," + T[71]);
gclib.GCommand("PVC=" + P3[72] + "," + V3[72] + "," + T[72]);
gclib.GCommand("PVC=" + P3[73] + "," + V3[73] + "," + T[73]);
gclib.GCommand("PVC=" + P3[74] + "," + V3[74] + "," + T[74]);
gclib.GCommand("PVC=" + P3[75] + "," + V3[75] + "," + T[75]);
gclib.GCommand("PVC=" + P3[76] + "," + V3[76] + "," + T[76]);
gclib.GCommand("PVC=" + P3[77] + "," + V3[77] + "," + T[77]);
gclib.GCommand("PVC=" + P3[78] + "," + V3[78] + "," + T[78]);
gclib.GCommand("PVC=" + P3[79] + "," + V3[79] + "," + T[79]);
gclib.GCommand("PVC=" + P3[80] + "," + V3[80] + "," + T[80]);
gclib.GCommand("PVC=" + P3[81] + "," + V3[81] + "," + T[81]);
gclib.GCommand("PVC=" + P3[82] + "," + V3[82] + "," + T[82]);

```

```

gclib.GCommand("PVC=" + P3[83] + "," + V3[83] + "," + T[83]);
gclib.GCommand("PVC=" + 0 + "," + 0 + "," + 0);

gclib.GCommand("BT ABC");

gclib.GCommand("PVA=" + 0 + "," + 0 + "," + 0);
gclib.GCommand("PVB=" + 0 + "," + 0 + "," + 0);
gclib.GCommand("PVC=" + 0 + "," + 0 + "," + 0);
gclib.GClose();

    }
SetTimer_TCPSendData1();

    button4.Enabled = true;
}
private void SetTimer_TCPSendData()
{
    timer_TCPSendData = new System.Timers.Timer(1000); //Peridic timer in ms
    timer_TCPSendData.Elapsed += Timer_TCPSendData_Event;
    timer_TCPSendData.AutoReset = false;
    timer_TCPSendData.Enabled = true;
}

private void SetTimer_TCPSendData1()
{
    timer_TCPSendData = new System.Timers.Timer((tiempodespla*1000)+1000);
//Peridic timer in ms
    timer_TCPSendData.Elapsed += Timer_TCPSendData_Event1;
    timer_TCPSendData.AutoReset = false;
    timer_TCPSendData.Enabled = true;
}

private void Timer_TCPSendData_Event(Object source, EventArgs e)
{
    if (textBox1.Text == "Fuera de rango")
    {
    }
    else
    {

        EncoControlador Encode = new EncoControlador();

        double PosicionAAA = g.commandValue("TPA");
        double PosicionBBB = g.commandValue("TPB");
        double PosicionCCC = g.commandValue("TPC");

        double la = L2;

        double r1 = L1;
        double ang1 = (0 * System.Math.PI) / 180; // angulo del brazo 1
        double ang2 = (120 * System.Math.PI) / 180; // angulo del brazo 2
        double ang3 = (240 * System.Math.PI) / 180; // angulo del brazo 3
    }
}

```

```

q1 = PosicionAAA * 360 / 100000;
q2 = PosicionBBB * 360 / 100000;
q3 = PosicionCCC * 360 / 100000;
double q11 = (-q1 * System.Math.PI) / 180;
double q22 = (-q2 * System.Math.PI) / 180;
double q33 = (-q3 * System.Math.PI) / 180;

if (q1 == q2 && q1 == q3)
{
    if (q1 == 0 && q2 == 0 && q3 == 0)
    {

        double cat1 = R + la * (Math.Cos(q11));
        double zn = (Math.Sqrt((Math.Pow(r1, 2)) -
(Math.Pow(cat1, 2))));

        double xn = 0;
        double yn = 0;

        znd = -1 * Math.Round(zn, 4);
        ynd = Math.Round(yn, 4);
        xnd = Math.Round(xn, 4);

        textBox1.Text = xnd.ToString();
        textBox2.Text = ynd.ToString();
        textBox3.Text = znd.ToString();

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();

    }
    else
    {
        if (q1 > 0 && q2 > 0 && q3 > 0)
        {

            double cat1 = R + la * (Math.Cos(-q11));
            double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -
(Math.Pow(cat1, 2))));

            double zn2 = (Math.Sin(-q11)) * L2;

            double zn = -(zn1 + zn2);
            double xn = 0;
            double yn = 0;

            znd = Math.Round(zn, 1);
            ynd = Math.Round(yn, 1);
            xnd = Math.Round(xn, 1);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();

```

```

        textBox6.Text = Math.Round(q3, 4).ToString();

        //listBox1.Items.Add("caso tres angulos mayores
0");

    }

else
{
    double cat1 = R + la * (Math.Cos(-q11));
    double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -
(Math.Pow(cat1, 2))));

    double zn2 = (Math.Sin(-q11)) * L2;

    double zn = -(zn1 + zn2);
    double xn = 0;
    double yn = 0;

    znd = Math.Round(zn, 1);
    ynd = Math.Round(yn, 1);
    xnd = Math.Round(xn, 1);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();
    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();

    //listBox1.Items.Add("caso tres angulos menores
0");

    }
}

else
{
    if (q2 == q3)
    {
        double q111 = q11 + 0.0000000001;
        double q222 = q22 + 0.0000000001;
        double q333 = q33 + 0.000000000001;

        double x1 = (R + la * Math.Cos(q111));
        double y1 = 0;
        double z1 = (la * Math.Sin(q111));

        double x2 = (R + la * Math.Cos(q222)) *
(Math.Cos(ang2));
        double y2 = (R + la * Math.Cos(q222)) *
(Math.Sin(ang2));

        double z2 = (la * Math.Sin(q222));

        double x3 = (R + la * Math.Cos(q333)) *
(Math.Cos(ang3));
        double y3 = (R + la * Math.Cos(q333)) *
(Math.Sin(ang3));

        double z3 = (la * Math.Sin(q333));

```

```

double a11 = 2 * (x3 - x1);
double a12 = 2 * (y3 - y1);
double a13 = 2 * (z3 - z1);

double a21 = 2 * (x3 - x2);
double a22 = 2 * (y3 - y2);
double a23 = 2 * (z3 - z2);

double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

double a1 = (a11 / a13) - (a21 / a23);
double a2 = (a22 / a23) - (a12 / a13);
double a3 = (b1 / a13) - (b2 / a23);

double a4 = (a2 / a1);
double a5 = (a3 / a1);

double a6 = (-a21 * a4 - a22) / a23;
double a7 = (b2 - a21 * a5) / a23;

double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));
double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 *
(a7 - z1));
double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a
* c))) / (2 * a);
double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a
* c))) / (2 * a);

double xn = a4 * yn + a5;
double xp = a4 * yp + a5;

double zn = a6 * yn + a7;
double zp = a6 * yp + a7;

if (zn < zp)
{
    znd = Math.Round(zn, 1);
    ynd = Math.Round(yn, 1);
    xnd = Math.Round(xn, 1);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();

    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();
    //listBox1.Items.Add("q2 = q3");
}
else
{
    znd = Math.Round(zp, 1);
    ynd = Math.Round(yp, 1);
    xnd = Math.Round(xp, 1);
}

```

```

        textBox1.Text = xnd.ToString();
        textBox2.Text = ynd.ToString();
        textBox3.Text = znd.ToString();

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();

        //listBox1.Items.Add("q2 = q3");
    }
}
else
{
    double x1 = (R + la * Math.Cos(q11));
    double y1 = 0;
    double z1 = (la * Math.Sin(q11));

    double x2 = (R + la * Math.Cos(q22)) *
(Math.Cos(ang2));
    double y2 = (R + la * Math.Cos(q22)) *
(Math.Sin(ang2));
    double z2 = (la * Math.Sin(q22));

    double x3 = (R + la * Math.Cos(q33)) *
(Math.Cos(ang3));
    double y3 = (R + la * Math.Cos(q33)) *
(Math.Sin(ang3));
    double z3 = (la * Math.Sin(q33));

    double a11 = 2 * (x3 - x1);
    double a12 = 2 * (y3 - y1);
    double a13 = 2 * (z3 - z1);

    double a21 = 2 * (x3 - x2);
    double a22 = 2 * (y3 - y2);
    double a23 = 2 * (z3 - z2);

    double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
    double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

    double a1 = (a11 / a13) - (a21 / a23);
    double a2 = (a22 / a23) - (a12 / a13);
    double a3 = (b1 / a13) - (b2 / a23);

    double a4 = (a2 / a1);
    double a5 = (a3 / a1);

    double a6 = (-a21 * a4 - a22) / a23;
    double a7 = (b2 - a21 * a5) / a23;

    double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));
    double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 *
(a7 - z1));
    double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

    double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a
* c))) / (2 * a);

```

```

* c))) / (2 * a);

double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a

double xn = a4 * yn + a5;
double xp = a4 * yp + a5;

double zn = a6 * yn + a7;
double zp = a6 * yp + a7;

if (zn < zp)
{
    znd = Math.Round(zn, 0);
    ynd = Math.Round(yn, 0);
    xnd = Math.Round(xn, 0);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();

    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();

    //listBox1.Items.Add("todos distintos");

}
else
{
    znd = Math.Round(zp, 0);
    ynd = Math.Round(yp, 0);
    xnd = Math.Round(xp, 0);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();

    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();

    //listBox1.Items.Add("todos distintos");

}
}

}

private void Timer_TCPSendData_Event1(Object source, EventArgs e)
{
    if (textBox1.Text == "Fuera de rango")
    {
    }
    else

```

```

{

//EncoControlador Encode = new EncoControlador();

double PosicionAA = g.commandValue("TPA");
double PosicionBB = g.commandValue("TPB");
double PosicionCC = g.commandValue("TPC");

double la = L2;

double r1 = L1;
double ang1 = (0 * System.Math.PI) / 180; // angulo del brazo 1
double ang2 = (120 * System.Math.PI) / 180; // angulo del brazo 2
double ang3 = (240 * System.Math.PI) / 180; // angulo del brazo 3

q1 = PosicionAA * 360 / 100000;
q2 = PosicionBB * 360 / 100000;
q3 = PosicionCC * 360 / 100000;
double q11 = (-q1 * System.Math.PI) / 180;
double q22 = (-q2 * System.Math.PI) / 180;
double q33 = (-q3 * System.Math.PI) / 180;
double max = 90;
double min = -90;

if (q1 == q2 && q1 == q3)
{
    if (q1 == 0 && q2 == 0 && q3 == 0)
    {

        double cat1 = R + la * (Math.Cos(q11));
        double zn = (Math.Sqrt((Math.Pow(r1, 2)) - (Math.Pow(cat1,
2))));

        double xn = 0;
        double yn = 0;

        znd = -1 * Math.Round(zn, 4);
        ynd = Math.Round(yn, 4);
        xnd = Math.Round(xn, 4);

        textBox1.Text = xnd.ToString();
        textBox2.Text = ynd.ToString();
        textBox3.Text = znd.ToString();

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();

    }
    else
    {
        if (q1 > 0 && q2 > 0 && q3 > 0)
        {

            double cat1 = R + la * (Math.Cos(-q11));

```

```

(Math.Pow(cat1, 2))));

double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -

double zn2 = (Math.Sin(-q11)) * L2;

double zn = -(zn1 + zn2);
double xn = 0;
double yn = 0;

znd = Math.Round(zn, 1);
ynd = Math.Round(yn, 1);
xnd = Math.Round(xn, 1);

textBox1.Text = xnd.ToString();
textBox2.Text = ynd.ToString();
textBox3.Text = znd.ToString();

textBox4.Text = Math.Round(q1, 4).ToString();
textBox5.Text = Math.Round(q2, 4).ToString();
textBox6.Text = Math.Round(q3, 4).ToString();

//listBox1.Items.Add("caso tres angulos mayores 0");

}
else
{

double cat1 = R + la * (Math.Cos(-q11));
double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -

(Math.Pow(cat1, 2))));

double zn2 = (Math.Sin(-q11)) * L2;

double zn = -(zn1 + zn2);
double xn = 0;
double yn = 0;

znd = Math.Round(zn, 1);
ynd = Math.Round(yn, 1);
xnd = Math.Round(xn, 1);

textBox1.Text = xnd.ToString();
textBox2.Text = ynd.ToString();
textBox3.Text = znd.ToString();
textBox4.Text = Math.Round(q1, 4).ToString();
textBox5.Text = Math.Round(q2, 4).ToString();
textBox6.Text = Math.Round(q3, 4).ToString();

//listBox1.Items.Add("caso tres angulos menores 0");

}

}
else
{
if (q2 == q3)
{
double q111 = q11 + 0.0000000001;
double q222 = q22 + 0.0000000001;
double q333 = q33 + 0.000000000001;

```

```

double x1 = (R + la * Math.Cos(q111));
double y1 = 0;
double z1 = (la * Math.Sin(q111));

double x2 = (R + la * Math.Cos(q222)) * (Math.Cos(ang2));
double y2 = (R + la * Math.Cos(q222)) * (Math.Sin(ang2));
double z2 = (la * Math.Sin(q222));

double x3 = (R + la * Math.Cos(q333)) * (Math.Cos(ang3));
double y3 = (R + la * Math.Cos(q333)) * (Math.Sin(ang3));
double z3 = (la * Math.Sin(q333));

double a11 = 2 * (x3 - x1);
double a12 = 2 * (y3 - y1);
double a13 = 2 * (z3 - z1);

double a21 = 2 * (x3 - x2);
double a22 = 2 * (y3 - y2);
double a23 = 2 * (z3 - z2);

double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

double a1 = (a11 / a13) - (a21 / a23);
double a2 = (a22 / a23) - (a12 / a13);
double a3 = (b1 / a13) - (b2 / a23);

double a4 = (a2 / a1);
double a5 = (a3 / a1);

double a6 = (-a21 * a4 - a22) / a23;
double a7 = (b2 - a21 * a5) / a23;

double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));
double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 * (a7 -
z1));
double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);
double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);

double xn = a4 * yn + a5;
double xp = a4 * yp + a5;

double zn = a6 * yn + a7;
double zp = a6 * yp + a7;

if (zn < zp)
{
    znd = Math.Round(zn, 1);
    ynd = Math.Round(yn, 1);
    xnd = Math.Round(xn, 1);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();
}

```

```

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();
        //listBox1.Items.Add("q2 = q3");
    }
    else
    {
        znd = Math.Round(zp, 1);
        ynd = Math.Round(yp, 1);
        xnd = Math.Round(xp, 1);

        textBox1.Text = xnd.ToString();
        textBox2.Text = ynd.ToString();
        textBox3.Text = znd.ToString();

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();

        //listBox1.Items.Add("q2 = q3");
    }
}
else
{
    double x1 = (R + la * Math.Cos(q11));
    double y1 = 0;
    double z1 = (la * Math.Sin(q11));

    double x2 = (R + la * Math.Cos(q22)) * (Math.Cos(ang2));
    double y2 = (R + la * Math.Cos(q22)) * (Math.Sin(ang2));
    double z2 = (la * Math.Sin(q22));

    double x3 = (R + la * Math.Cos(q33)) * (Math.Cos(ang3));
    double y3 = (R + la * Math.Cos(q33)) * (Math.Sin(ang3));
    double z3 = (la * Math.Sin(q33));

    double a11 = 2 * (x3 - x1);
    double a12 = 2 * (y3 - y1);
    double a13 = 2 * (z3 - z1);

    double a21 = 2 * (x3 - x2);
    double a22 = 2 * (y3 - y2);
    double a23 = 2 * (z3 - z2);

    double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
    double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

    double a1 = (a11 / a13) - (a21 / a23);
    double a2 = (a22 / a23) - (a12 / a13);
    double a3 = (b1 / a13) - (b2 / a23);

    double a4 = (a2 / a1);
    double a5 = (a3 / a1);

    double a6 = (-a21 * a4 - a22) / a23;
    double a7 = (b2 - a21 * a5) / a23;

    double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));

```

```

        double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 * (a7 -
z1));
        double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

        double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);
        double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);

        double xn = a4 * yn + a5;
        double xp = a4 * yp + a5;

        double zn = a6 * yn + a7;
        double zp = a6 * yp + a7;

        if (zn < zp)
        {
            znd = Math.Round(zn, 0);
            ynd = Math.Round(yn, 0);
            xnd = Math.Round(xn, 0);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();
            textBox6.Text = Math.Round(q3, 4).ToString();

            //listBox1.Items.Add("todos distintos");
        }
        Else
        {
            znd = Math.Round(zp, 0);
            ynd = Math.Round(yp, 0);
            xnd = Math.Round(xp, 0);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();
            textBox6.Text = Math.Round(q3, 4).ToString();

            //listBox1.Items.Add("todos distintos");
        }
    }
}

}

}

private void button4_Click(object sender, EventArgs e)

```

```
{
    xir.Text = string.Empty;
    yir.Text = string.Empty;
    zir.Text = string.Empty;
    Velocidad.Text = string.Empty;
}
private void button1_Click(object sender, EventArgs e)
{
    gclib.GCommand("MO ABC");
}
}
}
```

ANEXO D: CÓDIGO CINEMÁTICA DIRECTA

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Interfaz
{
    class Cinematica_Direc
    {
        public double znd, xnd, ynd, q1, q2, q3;

        public void CineDirec2(double[] CIX1, double[] CIY1, double[] CIZ1, int cont1,
double L1, double L2, double R)
        {

            double PosicionAA = g.commandValue("TPA");
            double PosicionBB = g.commandValue("TPB");
            double PosicionCC = g.commandValue("TPC");

            double la = L2;

            double r1 = L1;
            double ang1 = (0 * System.Math.PI) / 180; // angulo del brazo 1
            double ang2 = (120 * System.Math.PI) / 180; // angulo del brazo 2
            double ang3 = (240 * System.Math.PI) / 180; // angulo del brazo 3

            q1 = PosicionAA * 360 / 100000;
            q2 = PosicionBB * 360 / 100000;
            q3 = PosicionCC * 360 / 100000;
            double q11 = (-q1 * System.Math.PI) / 180;
            double q22 = (-q2 * System.Math.PI) / 180;
            double q33 = (-q3 * System.Math.PI) / 180;
            double max = 90;
            double min = -90;

            if (q1 == q2 && q1 == q3)
            {
                if (q1 == 0 && q2 == 0 && q3 == 0)
                {

                    double cat1 = R + la * (Math.Cos(q11));
                    double zn = (Math.Sqrt((Math.Pow(r1, 2)) - (Math.Pow(cat1,
2)))));

                    double xn = 0;
                    double yn = 0;

                    znd = -1 * Math.Round(zn, 4);
                    ynd = Math.Round(yn, 4);
                    xnd = Math.Round(xn, 4);

                    textBox1.Text = xnd.ToString();
                    textBox2.Text = ynd.ToString();
                    textBox3.Text = znd.ToString();
                }
            }
        }
    }
}

```

```

        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();

    }
    else
    {
        if (q1 > 0 && q2 > 0 && q3 > 0)
        {
            double cat1 = R + la * (Math.Cos(-q11));
            double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -
(Math.Pow(cat1, 2))));

            double zn2 = (Math.Sin(-q11)) * L2;

            double zn = -(zn1 + zn2);
            double xn = 0;
            double yn = 0;

            znd = Math.Round(zn, 1);
            ynd = Math.Round(yn, 1);
            xnd = Math.Round(xn, 1);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();
            textBox6.Text = Math.Round(q3, 4).ToString();

            //listBox1.Items.Add("caso tres angulos mayores 0");

        }
    }
    else
    {
        double cat1 = R + la * (Math.Cos(-q11));
        double zn1 = (Math.Sqrt((Math.Pow(r1, 2)) -
(Math.Pow(cat1, 2))));

        double zn2 = (Math.Sin(-q11)) * L2;

        double zn = -(zn1 + zn2);
        double xn = 0;
        double yn = 0;

        znd = Math.Round(zn, 1);
        ynd = Math.Round(yn, 1);
        xnd = Math.Round(xn, 1);

        textBox1.Text = xnd.ToString();
        textBox2.Text = ynd.ToString();
        textBox3.Text = znd.ToString();
        textBox4.Text = Math.Round(q1, 4).ToString();
        textBox5.Text = Math.Round(q2, 4).ToString();
        textBox6.Text = Math.Round(q3, 4).ToString();
    }
}

```

```

        //listBox1.Items.Add("caso tres angulos menores 0");
    }
}
else
{
    if (q2 == q3)
    {
        double q111 = q11 + 0.0000000001;
        double q222 = q22 + 0.0000000001;
        double q333 = q33 + 0.0000000001;

        double x1 = (R + la * Math.Cos(q111));
        double y1 = 0;
        double z1 = (la * Math.Sin(q111));

        double x2 = (R + la * Math.Cos(q222)) * (Math.Cos(ang2));
        double y2 = (R + la * Math.Cos(q222)) * (Math.Sin(ang2));
        double z2 = (la * Math.Sin(q222));

        double x3 = (R + la * Math.Cos(q333)) * (Math.Cos(ang3));
        double y3 = (R + la * Math.Cos(q333)) * (Math.Sin(ang3));
        double z3 = (la * Math.Sin(q333));

        double a11 = 2 * (x3 - x1);
        double a12 = 2 * (y3 - y1);
        double a13 = 2 * (z3 - z1);

        double a21 = 2 * (x3 - x2);
        double a22 = 2 * (y3 - y2);
        double a23 = 2 * (z3 - z2);

        double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
        double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

        double a1 = (a11 / a13) - (a21 / a23);
        double a2 = (a22 / a23) - (a12 / a13);
        double a3 = (b1 / a13) - (b2 / a23);

        double a4 = (a2 / a1);
        double a5 = (a3 / a1);

        double a6 = (-a21 * a4 - a22) / a23;
        double a7 = (b2 - a21 * a5) / a23;

        double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));
        double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 * (a7 -
z1));
        double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

        double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);
        double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);

```

```

double xn = a4 * yn + a5;
double xp = a4 * yp + a5;

double zn = a6 * yn + a7;
double zp = a6 * yp + a7;

if (zn < zp)
{
    znd = Math.Round(zn, 1);
    ynd = Math.Round(yn, 1);
    xnd = Math.Round(xn, 1);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();

    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();
    //listBox1.Items.Add("q2 = q3");
}
else
{
    znd = Math.Round(zp, 1);
    ynd = Math.Round(yp, 1);
    xnd = Math.Round(xp, 1);

    textBox1.Text = xnd.ToString();
    textBox2.Text = ynd.ToString();
    textBox3.Text = znd.ToString();

    textBox4.Text = Math.Round(q1, 4).ToString();
    textBox5.Text = Math.Round(q2, 4).ToString();
    textBox6.Text = Math.Round(q3, 4).ToString();

    //listBox1.Items.Add("q2 = q3");
}
}

else
{
    double x1 = (R + la * Math.Cos(q11));
    double y1 = 0;
    double z1 = (la * Math.Sin(q11));

    double x2 = (R + la * Math.Cos(q22)) * (Math.Cos(ang2));
    double y2 = (R + la * Math.Cos(q22)) * (Math.Sin(ang2));
    double z2 = (la * Math.Sin(q22));

    double x3 = (R + la * Math.Cos(q33)) * (Math.Cos(ang3));
    double y3 = (R + la * Math.Cos(q33)) * (Math.Sin(ang3));
    double z3 = (la * Math.Sin(q33));

    double a11 = 2 * (x3 - x1);
    double a12 = 2 * (y3 - y1);
    double a13 = 2 * (z3 - z1);

    double a21 = 2 * (x3 - x2);
    double a22 = 2 * (y3 - y2);
    double a23 = 2 * (z3 - z2);
}

```

```

        double b1 = -(Math.Pow(x1, 2)) - (Math.Pow(y1, 2)) -
(Math.Pow(z1, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));
        double b2 = -(Math.Pow(x2, 2)) - (Math.Pow(y2, 2)) -
(Math.Pow(z2, 2)) + (Math.Pow(x3, 2)) + (Math.Pow(y3, 2)) + (Math.Pow(z3, 2));

        double a1 = (a11 / a13) - (a21 / a23);
        double a2 = (a22 / a23) - (a12 / a13);
        double a3 = (b1 / a13) - (b2 / a23);

        double a4 = (a2 / a1);
        double a5 = (a3 / a1);

        double a6 = (-a21 * a4 - a22) / a23;
        double a7 = (b2 - a21 * a5) / a23;

        double a = (Math.Pow(a4, 2)) + 1 + (Math.Pow(a6, 2));
        double b = (2 * a4 * (a5 - x1)) - (2 * y1) + (2 * a6 * (a7 -
z1));
        double c = a5 * (a5 - 2 * x1) + a7 * (a7 - 2 * z1) +
(Math.Pow(x1, 2)) + (Math.Pow(y1, 2)) + (Math.Pow(z1, 2)) - (Math.Pow(r1, 2));

        double yn = (-b - (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);
        double yp = (-b + (Math.Sqrt((Math.Pow(b, 2)) - 4 * a * c))) /
(2 * a);

        double xn = a4 * yn + a5;
        double xp = a4 * yp + a5;

        double zn = a6 * yn + a7;
        double zp = a6 * yp + a7;

        if (zn < zp)
        {
            znd = Math.Round(zn, 0);
            ynd = Math.Round(yn, 0);
            xnd = Math.Round(xn, 0);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();
            textBox6.Text = Math.Round(q3, 4).ToString();

            //listBox1.Items.Add("todos distintos");
        }
        else
        {
            znd = Math.Round(zp, 0);
            ynd = Math.Round(yp, 0);
            xnd = Math.Round(xp, 0);

            textBox1.Text = xnd.ToString();
            textBox2.Text = ynd.ToString();
            textBox3.Text = znd.ToString();

            textBox4.Text = Math.Round(q1, 4).ToString();
            textBox5.Text = Math.Round(q2, 4).ToString();

```

```
        textBox6.Text = Math.Round(q3, 4).ToString();  
        //listBox1.Items.Add("todos distintos");  
    }  
}  
}
```

ANEXO E: CÓDIGO CINEMÁTICA INVERSA

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Interfaz2_0
{
    class Cinematica_inver_1
    {
        public double[,] Posi, eje1, eje2, eje3, beta;
        public double[] CIT, CIq1, CIq2, CIq3, CIq11, CIq22, CIq33;
        public int cont1, cont2, cont3, cont4;
        public void CineInver(double[,] perS1, double N, double L1, double L2, double
Ra, double Rb)
        {
            Form1 form = new Form1();
            double delT = 1 / N;
            double n = N;
            int N1 = Convert.ToInt32(n);
            Posi = new double[16, N1 + 2];
            eje1 = new double[3, N1 + 2];
            eje2 = new double[3, N1 + 2];
            eje3 = new double[3, N1 + 2];
            beta = new double[3, N1 + 2];
            CIT = new double[N1 + 2];
            CIq1 = new double[N1 + 2];
            CIq2 = new double[N1 + 2];
            CIq3 = new double[N1 + 2];
            CIq11 = new double[N1 + 2];
            CIq22 = new double[N1 + 2];
            CIq33 = new double[N1 + 2];

            cont1 = 0;
            cont2 = 0;
            cont3 = 0;
            cont4 = 0;

            for (double i = 0; i <= 1; i = i + delT)
            {
                // Tiempo
                Posi[0, cont1] = perS1[0, cont1];
                //Debug.WriteLine(Posi[0,cont1]);

                // Eje 1
                Posi[1, cont1] = perS1[1, cont1] + 50; //px1
                Posi[2, cont1] = perS1[2, cont1]; //py1
                Posi[3, cont1] = perS1[3, cont1]; //pz1

                //Debug.WriteLine(Posi[1,cont1]);

                // Eje 2
                Posi[4, cont1] = perS1[1, cont1] - 25;
                Posi[5, cont1] = perS1[2, cont1] + 43.30127019;
                Posi[6, cont1] = perS1[3, cont1];
                //Debug.WriteLine(Posi[4, cont1]);
            }
        }
    }
}
```

```

// Eje 2 (Rotación plano -240°)

double cos1 = -0.5;
double cos2 = -0.5;
double sen1 = 0.866025403784438;
double sen2 = -0.866025403784438;

Posi[7, cont1] = (Posi[4, cont1] * cos1) + (Posi[5, cont1] * sen1);
//px2
Posi[8, cont1] = (Posi[4, cont1] * sen2) + (Posi[5, cont1] * cos2);
//py2

Posi[9, cont1] = perS1[3, cont1]; //pz2
//Debug.WriteLine(Posi[7, cont1]);
// Eje 3
Posi[10, cont1] = perS1[1, cont1] - 25;
Posi[11, cont1] = perS1[2, cont1] - 43.30127019;
Posi[12, cont1] = perS1[3, cont1];

// Eje 3 (Rotación plano -120°)

double cos3 = -0.5;
double cos4 = -0.5;
double sen3 = -0.866025403784438;
double sen4 = 0.866025403784438;

Posi[13, cont1] = (Posi[10, cont1] * cos3) + (Posi[11, cont1] * sen3);
//px3
Posi[14, cont1] = (Posi[10, cont1] * sen4) + (Posi[11, cont1] * cos4);
//py3

Posi[15, cont1] = perS1[3, cont1]; //pz3

//Debug.WriteLine(Posi[14, cont1]);
//Debug.WriteLine("px3" + Posi[13, cont1]);

////////////////////////////////////
// PARA EJE 1 //
////////////////////////////////////

if (0 <= Posi[1, cont1] && Posi[1, cont1] <= Ra)
{
    eje1[0, cont1] = Math.Sqrt(Math.Pow((Ra - Posi[1, cont1]), 2.0) +
Math.Pow(Posi[3, cont1], 2.0));
    eje1[1, cont1] = -1.0 * Math.Atan(Posi[3, cont1] / (Ra - Posi[1,
cont1]));
    eje1[2, cont1] = Math.Asin((Posi[2, cont1]) / L1); //Tetta1
    beta[0, cont1] = Math.Acos((Math.Pow(eje1[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje1[2, cont1])), 2.0)) / (2.0 * eje1[0,
cont1] * L2));
    CIq1[cont1] = (((Math.PI - (eje1[1, cont1] + beta[0, cont1])) *
360) / (2 * Math.PI))*100000)/360;
    CIq11[cont1] = ((Math.PI - (eje1[1, cont1] + beta[0, cont1])) *
360) / (2 * Math.PI);
    //Debug.WriteLine(CIq1[cont1]);
}
if (Posi[1, cont1] < 0)
{

```

```

        eje1[0, cont1] = Math.Sqrt(Math.Pow((Math.Abs(Posi[1, cont1]) +
Ra), 2.0) + Math.Pow(Posi[3, cont1], 2.0));
        eje1[1, cont1] = -1.0 * Math.Atan(Posi[3, cont1] /
(Math.Abs(Posi[1, cont1]) + Ra));
        eje1[2, cont1] = Math.Asin((Posi[2, cont1]) / L1); //Tetta1
        beta[0, cont1] = Math.Acos((Math.Pow(eje1[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje1[2, cont1])), 2.0)) / (2.0 * eje1[0,
cont1] * L2));
        CIq1[cont1] = (((Math.PI - (eje1[1, cont1] + beta[0, cont1])) *
360) / (2 * Math.PI)) * 100000) / 360;
        CIq11[cont1] = ((Math.PI - (eje1[1, cont1] + beta[0, cont1])) *
360) / (2 * Math.PI);
        //Debug.WriteLine(CIq1[cont1]);
    }
    if (Posi[1, cont1] > Ra)
    {
        eje1[0, cont1] = Math.Sqrt(Math.Pow((Posi[1, cont1] - Ra), 2.0) +
Math.Pow(Posi[3, cont1], 2.0));
        eje1[1, cont1] = -1.0 * Math.Atan((Posi[1, cont1] - Ra) / Posi[3,
cont1]);
        eje1[2, cont1] = Math.Asin((Posi[2, cont1]) / L1); //Tetta1
        beta[0, cont1] = Math.Acos((Math.Pow(eje1[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje1[2, cont1])), 2.0)) / (2.0 * eje1[0,
cont1] * L2));
        CIq1[cont1] = (((Math.PI / 2) - (eje1[1, cont1] + beta[0,
cont1])) * 360) / (2 * Math.PI)) * 100000) / 360;
        CIq11[cont1] = (((Math.PI / 2) - (eje1[1, cont1] + beta[0,
cont1])) * 360) / (2 * Math.PI);
        //Debug.WriteLine(CIq1[cont1]);
    }
}

////////////////////////////////////
// PARA EJE 2 //
////////////////////////////////////

if (0 <= Posi[7, cont1] && Posi[7, cont1] <= Ra)
{
    //Debug.WriteLine("3");
    eje2[0, cont1] = Math.Sqrt(Math.Pow((Ra - Posi[7, cont1]), 2.0) +
Math.Pow(Posi[9, cont1], 2.0));
    eje2[1, cont1] = -1.0 * Math.Atan(Posi[9, cont1] / (Ra - Posi[7,
cont1]));
    eje2[2, cont1] = Math.Asin((Posi[8, cont1]) / L1); //Tetta1
    beta[1, cont1] = Math.Acos((Math.Pow(eje2[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje2[2, cont1])), 2.0)) / (2.0 * eje2[0,
cont1] * L2));
    CIq2[cont1] = (((Math.PI - (eje2[1, cont1] + beta[1, cont1])) *
360) / (2 * Math.PI)) * 100000) / 360;
    //Debug.WriteLine(eje2[0, cont1]);
    //Debug.WriteLine(beta[1, cont1]);
}
if (Posi[7, cont1] < 0)
{
    //Debug.WriteLine("1");
    eje2[0, cont1] = Math.Sqrt(Math.Pow((Math.Abs(Posi[7, cont1]) +
Ra), 2.0) + Math.Pow(Posi[9, cont1], 2.0));

```

```

        eje2[1, cont1] = -1.0 * Math.Atan(Posi[9, cont1] /
(Math.Abs(Posi[7, cont1]) + Ra));
        eje2[2, cont1] = Math.Asin((Posi[8, cont1]) / L1); //Tetta1
        beta[1, cont1] = Math.Acos((Math.Pow(eje2[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje2[2, cont1])), 2.0)) / (2.0 * eje2[0,
cont1] * L2));
        CIq2[cont1] = (((Math.PI - (eje2[1, cont1] + beta[1, cont1])) *
360) / (2 * Math.PI)) * 100000) / 360;
        //Debug.WriteLine(eje2[0, cont3]);
        //Debug.WriteLine(beta[1, cont3]);
    }
    if (Posi[7, cont1] > Ra)
    {
        //Debug.WriteLine("2");
        eje2[0, cont1] = Math.Sqrt(Math.Pow((Posi[7, cont1] - Ra), 2.0) +
Math.Pow(Posi[9, cont1], 2.0));
        eje2[1, cont1] = -1.0 * Math.Atan((Posi[7, cont1] - Ra) / Posi[9,
cont1]);
        eje2[2, cont1] = Math.Asin((Posi[8, cont1]) / L1); //Tetta1
        beta[1, cont1] = Math.Acos((Math.Pow(eje2[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje2[2, cont1])), 2.0)) / (2.0 * eje2[0,
cont1] * L2));
        CIq2[cont1] = (((Math.PI / 2) - (eje2[1, cont1] + beta[1,
cont1])) * 360) / (2 * Math.PI)) * 100000) / 360;
        //Debug.WriteLine(eje2[0, cont3]);
    }

}

////////////////////////////////////
// PARA EJE 3 //
////////////////////////////////////

if (0 <= Posi[13, cont1] && Posi[13, cont1] <= Ra)
{
    eje3[0, cont1] = Math.Sqrt(Math.Pow((Ra - Posi[13, cont1]), 2.0) +
Math.Pow(Posi[15, cont1], 2.0));
    eje3[1, cont1] = -1.0 * Math.Atan(Posi[15, cont1] / (Ra - Posi[13,
cont1]));
    eje3[2, cont1] = Math.Asin((Posi[14, cont1]) / L1); //Tetta1
    beta[2, cont1] = Math.Acos((Math.Pow(eje3[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje3[2, cont1])), 2.0)) / (2.0 * eje3[0,
cont1] * L2));
    CIq3[cont1] = (((Math.PI - (eje3[1, cont1] + beta[2, cont1])) *
360) / (2 * Math.PI)) * 100000) / 360;
    //Debug.WriteLine(CIq3[cont1]);
}
if (Posi[13, cont1] < 0)
{
    eje3[0, cont1] = Math.Sqrt(Math.Pow((Math.Abs(Posi[13, cont1]) +
Ra), 2.0) + Math.Pow(Posi[15, cont1], 2.0));
    eje3[1, cont1] = -1.0 * Math.Atan(Posi[15, cont1] /
(Math.Abs(Posi[13, cont1]) + Ra));
    eje3[2, cont1] = Math.Asin((Posi[14, cont1]) / L1); //Tetta1
    beta[2, cont1] = Math.Acos((Math.Pow(eje3[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje3[2, cont1])), 2.0)) / (2.0 * eje3[0,
cont1] * L2));

```

```

        CIq3[cont1] = (((Math.PI - (eje3[1, cont1] + beta[2, cont1])) *
360) / (2 * Math.PI)) * 100000) / 360;
        //Debug.WriteLine(CIq3[cont1]);
    }
    if (Posi[13, cont1] > Ra)
    {
        eje3[0, cont1] = Math.Sqrt(Math.Pow((Posi[13, cont1] - Ra), 2.0) +
Math.Pow(Posi[15, cont1], 2.0));
        eje3[1, cont1] = -1.0 * Math.Atan((Posi[13, cont1] - Ra) /
Posi[15, cont1]);
        eje3[2, cont1] = Math.Asin((Posi[14, cont1] / L1); //Tetta1
        beta[2, cont1] = Math.Acos((Math.Pow(eje3[0, cont1], 2.0) +
Math.Pow(L2, 2.0) - Math.Pow((L1 * Math.Cos(eje3[2, cont1])), 2.0)) / (2.0 * eje3[0,
cont1] * L2));
        CIq3[cont1] = (((Math.PI / 2) - (eje3[1, cont1] + beta[2,
cont1])) * 360) / (2 * Math.PI)) * 100000) / 360;
        //Debug.WriteLine(CIq3[cont1]);
    }
    cont1++;
}
}
}

```

ANEXO F: CÓDIGO INTERPOLACIÓN Y PERFIL SUAVIZADO

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Interfaz
{
    public class Interpolacion
    {
        public double tiemdespl, delX, delY, delZ, N, vecdx, vecdy, vecdz;
        public void Inter(double xf, double yf, double zf, double vel, double xi,
double yi, double zi)
        {

            double tiempo = 0.00025;

            vecdx = Math.Abs(xf - xi);
            vecdy = Math.Abs(yf - yi);
            vecdz = Math.Abs(zf - zi);
            Debug.WriteLine(vecdx);
            Debug.WriteLine(vecdy);
            Debug.WriteLine(vecdz);

            double L = Math.Sqrt((Math.Pow(vecdx, 2)) + (Math.Pow(vecdy, 2)) +
(Math.Pow(vecdz, 2)));

            tiemdespl = L / vel;

            double delu = vel * tiempo;

            double n = L / delu;
            N = Convert.ToInt32(n);

            delX = vecdx / N;
            delY = vecdy / N;
            delZ = vecdz / N;

            Debug.WriteLine(delX);
            Debug.WriteLine(delY);
            Debug.WriteLine(delZ);
            Debug.WriteLine(L);
            Debug.WriteLine("Este es N = " + N);
            Debug.WriteLine(delu);
            Debug.WriteLine(tiemdespl);
        }
    }
}

using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Interfaz
{
    public class perfilS
    {
        public int cont;
        public double[,] perS;
        public void perfilSuavizado(double delX, double delY, double delZ, double N,
double tiempodespla, double zi, double zf, double xi, double xf, double yi, double yf)
        {
            Debug.WriteLine("Test perfil");
            double n = N;
            int N1 = Convert.ToInt32(n);
            double delT = 1 / N;
            double t1 = 1 * 0.1;
            double t2 = 1 - 1 * 0.1;
            double k1 = 0;
            double c1 = (1 - k1) / (1 + t2 - t1);
            double c2 = 2 * c1;
            double k2 = k1 - t1 * c1;
            double c3 = c1;
            double k3 = k1 + c1 * (t2 - t1);
            cont = 0;
            double i = 0;
            perS = new double[4, N1 + 1];

            for (i = 0; i <= 1; i = i + delT)
            {
                double ñ;
                if (0 <= i && i <= t1)
                {

                    double sen1 = (Math.PI * i) / t1;
                    double sen11 = (sen1 * Math.PI) / 180;
                    double d1 = c1 * (i - (t1 / Math.PI * (Math.Sin(sen1)))) + k1;
                    perS[0, cont] = i * tiempodespla;
                    if (xi >= xf)
                    {
                        perS[1, cont] = xi - (d1 * delX);
                    }
                    else
                    {
                        perS[1, cont] = xi + (d1 * delX);
                    }
                    if (yi >= yf)
                    {
                        perS[2, cont] = yi - (d1 * delY);
                    }
                    else
                    {
                        perS[2, cont] = yi + (d1 * delY);
                    }

                    if (zi >= zf)
                    {

```

```

        perS[3, cont] = -1 * (-zi + (d1 * delZ));
    }
    else
    {
        perS[3, cont] = -1 * (-zi - (d1 * delZ));
    }
}

```

```

if (t1 <= i && i <= t2)
{
    double d2 = c2 * i + k2;

    perS[0, cont] = i * tiempodespla;
    if (xi >= xf)
    {
        perS[1, cont] = xi - (d2 * delX);
    }
    else
    {
        perS[1, cont] = xi + (d2 * delX);
    }
    if (yi >= yf)
    {
        perS[2, cont] = yi - (d2 * delY);
    }
    else
    {
        perS[2, cont] = yi + (d2 * delY);
    }

    if (zi >= zf)
    {
        perS[3, cont] = -1 * (-zi + (d2 * delZ));
    }
    else
    {
        perS[3, cont] = -1 * (-zi - (d2 * delZ));
    }
}

if (t2 <= i && i <= 1.00000001)
{
    double sen2 = (Math.PI * (i - t2)) / (t2 - 1);
    double sen22 = (sen2 * Math.PI) / 180;
    double d3 = c3 * (i + ((t2 - 1) / Math.PI) * Math.Sin(sen2)) + k3;
}

```

```

double angulo = Math.Sin((45 * Math.PI) / 180);
double x3 = d3;
double y3 = d3 * delY;
double z3 = d3 * delZ;
perS[0, cont] = i * tiempodespla;
if (xi >= xf)
{
    perS[1, cont] = xi - (d3 * delX);
}
else
{
    perS[1, cont] = xi + (d3 * delX);
}
if (yi >= yf)
{
    perS[2, cont] = yi - (d3 * delY);
}
else
{
    perS[2, cont] = yi + (d3 * delY);
}
;
if (zi >= zf)
{
    perS[3, cont] = -1 * (-zi + (d3 * delZ));
}
else
{
    perS[3, cont] = -1 * (-zi - (d3 * delZ));
}

}

cont++;
}

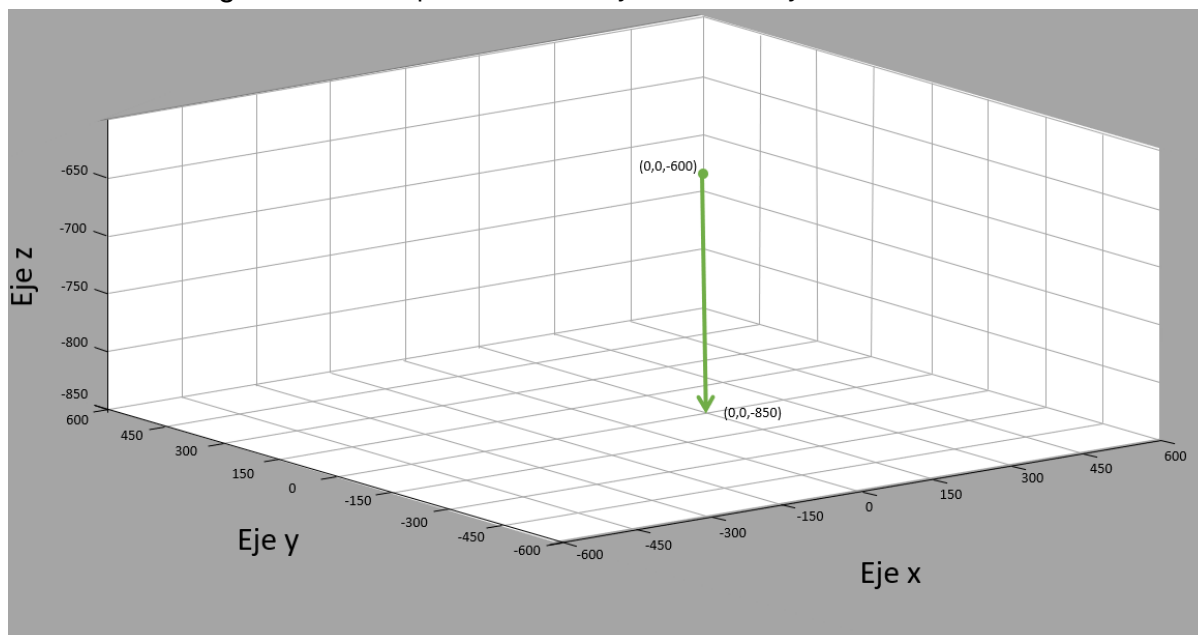
```

ANEXO G: OTRAS TRAYECTORIAS.

G.1 Trayectoria lineal A – B vertical.

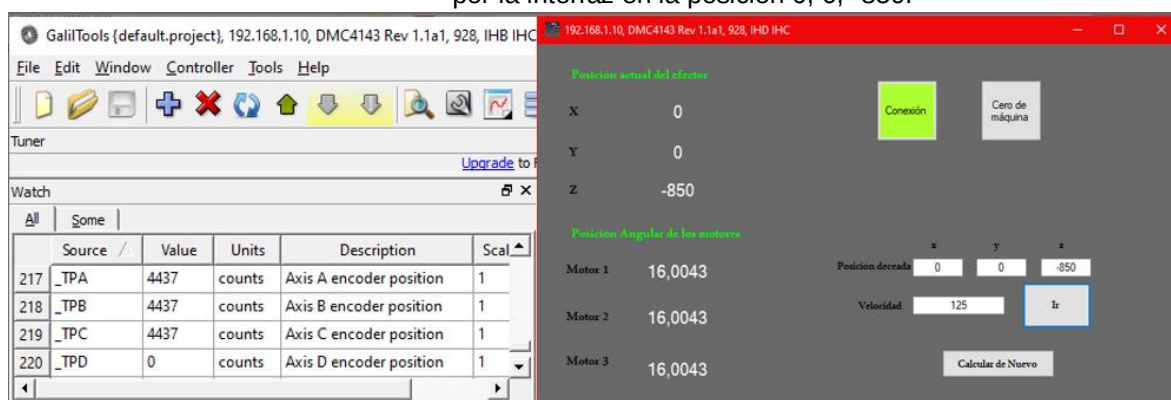
Esta trayectoria consiste en un desplazamiento desde (0,0,-600) hasta (0,0,-850), siendo el recorrido de 250 mm.

Figura G. Representación Trayectoria 1: Trayectoria lineal A – B vertical.



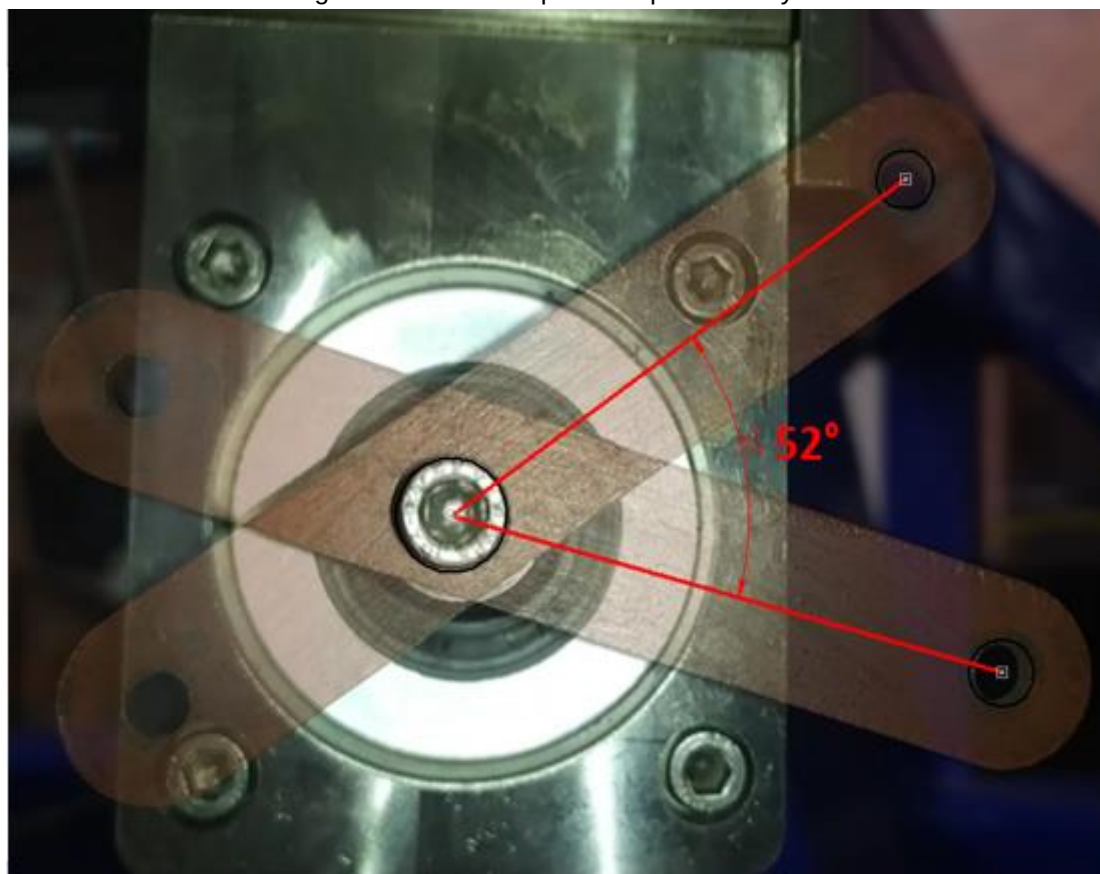
Fuente: Autor.

Figura H. Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 0, 0, -850.



Fuente Autor

Figura I. Comparación posición A y B Motor 3.



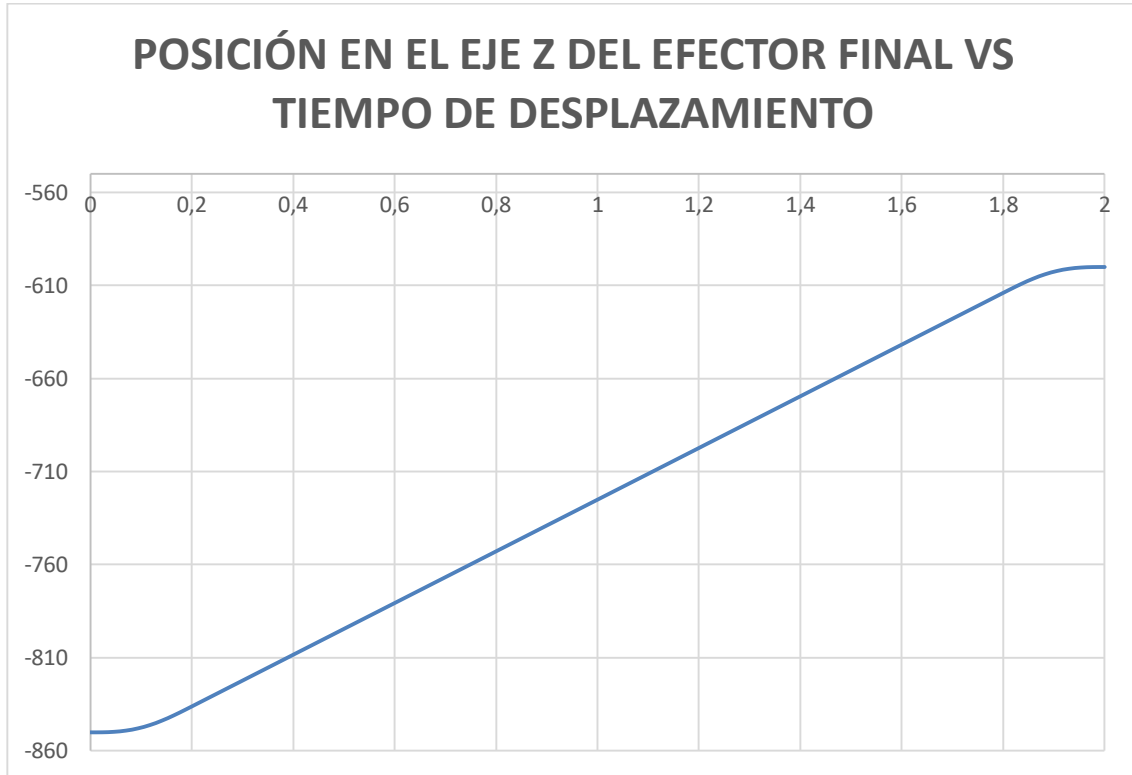
Fuente Autor.

Tabla A. Desplazamiento Angular del motor 3

	Desplazamiento angular del motor 3		
	Valor inicial	Valor final	Desplazamiento total
GalilTools	15,9732	-36,648	52,621
Interfaz	16,0043	-36,6649	52,669
Fotografías	16	-36	52,000

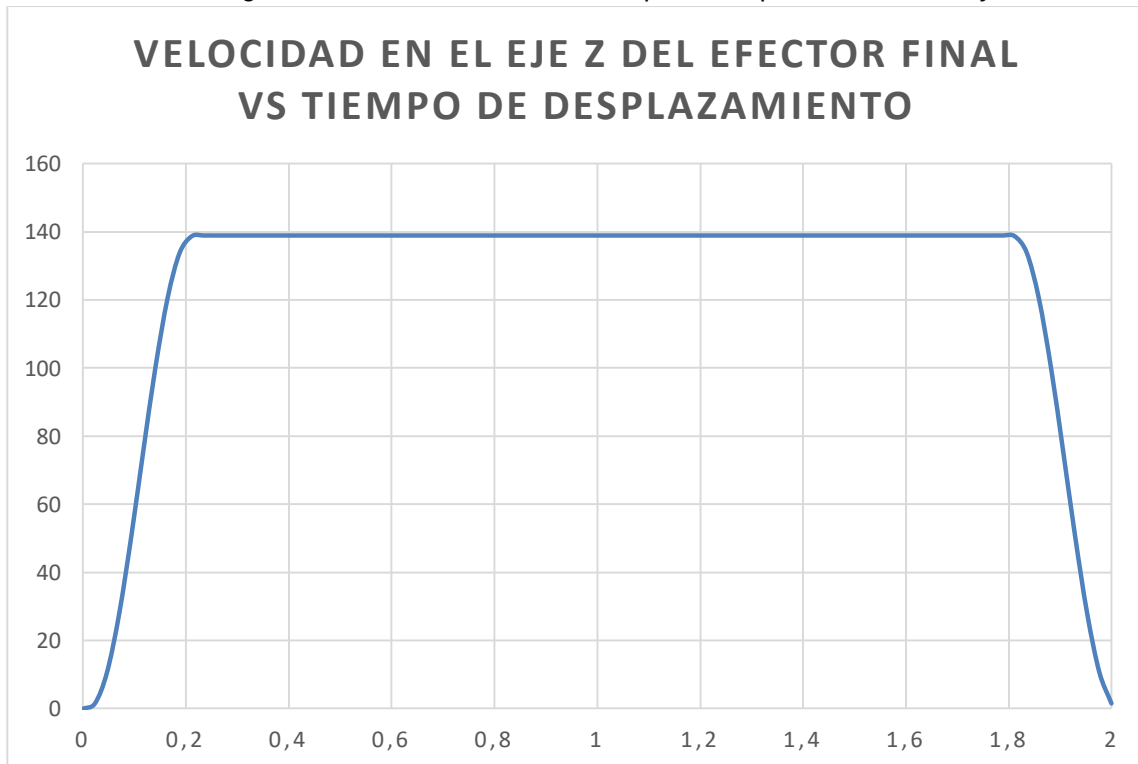
Fuente autor

Figura J. Posición Z vs tiempo de desplazamiento en trayectoria 1



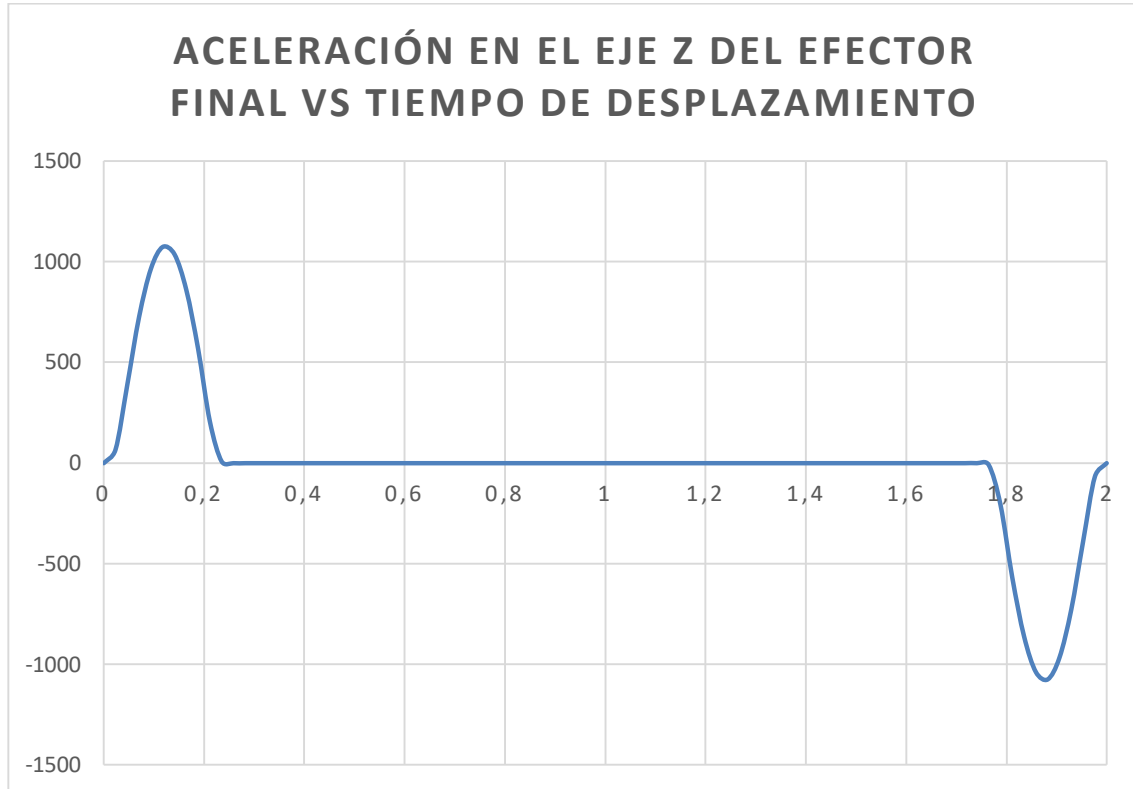
Fuente Autor

Figura K. Velocidad Z vs tiempo de desplazamiento en trayectoria 1



Fuente autor

Figura L. Aceleración Z vs tiempo de desplazamiento en trayectoria 1



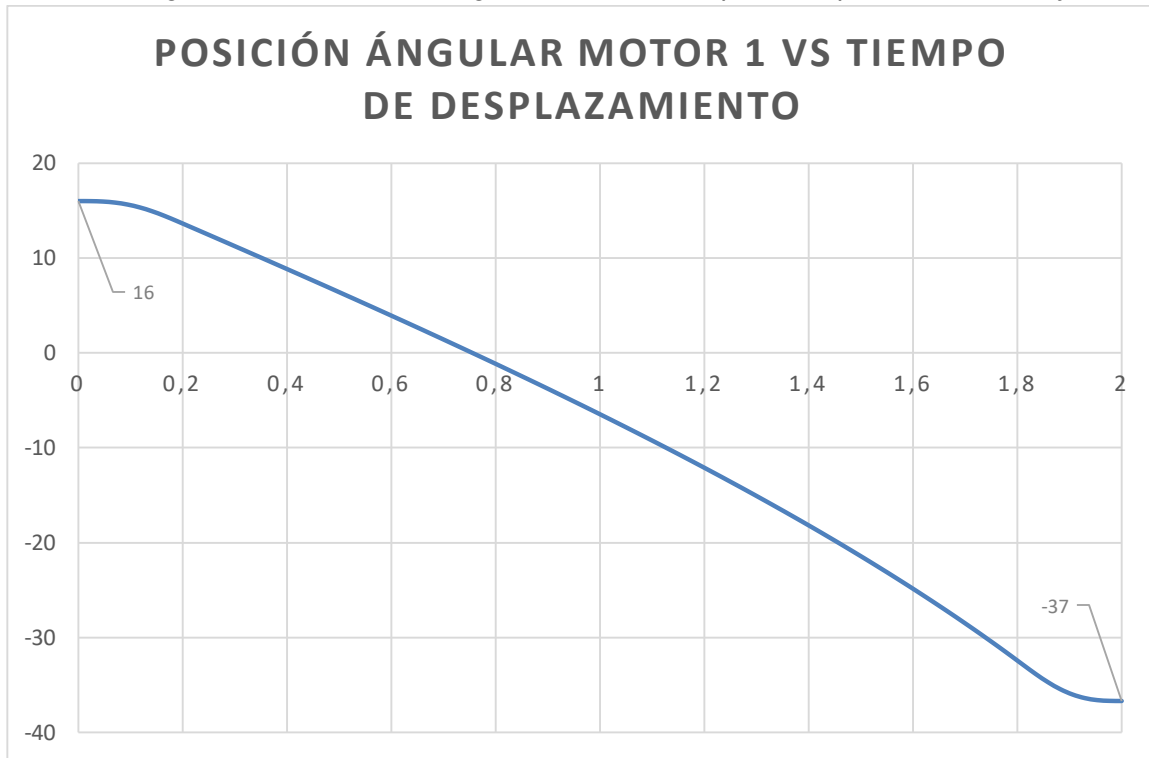
Fuente Autor

Figura M. Jerk Z vs tiempo de desplazamiento en trayectoria 1



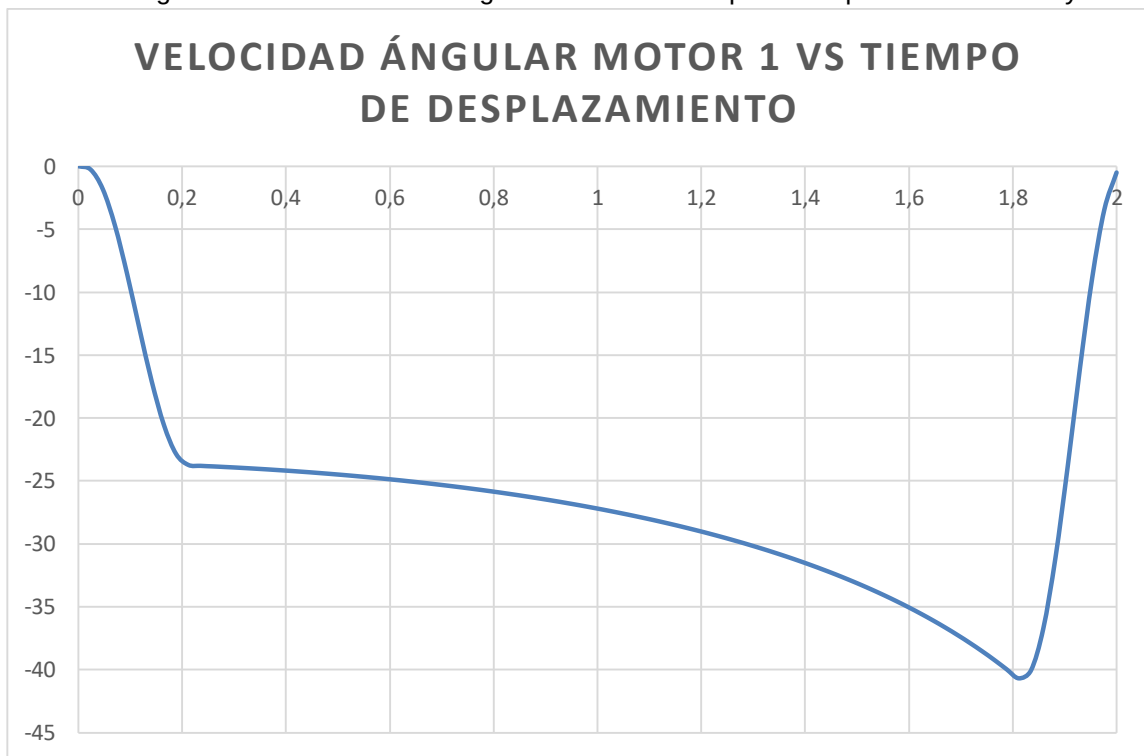
Fuente Autor

Figura N. Posición angular motor 1 vs tiempo de desplazamiento en trayectoria 1



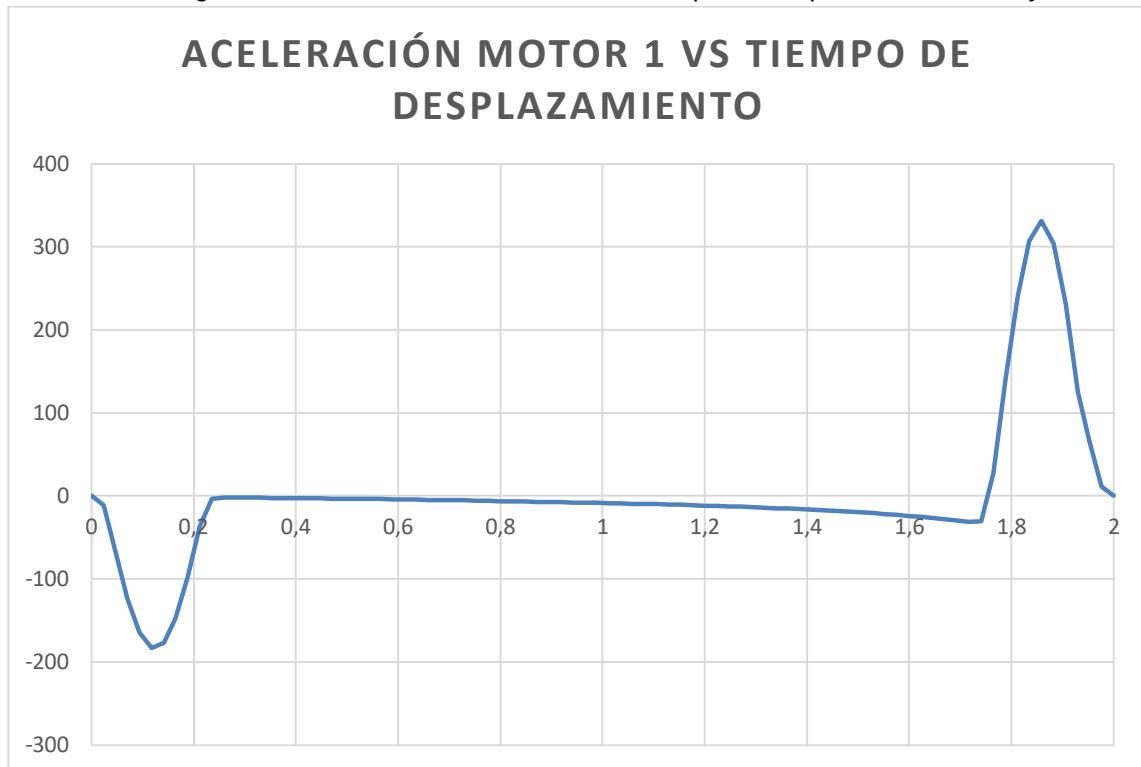
Fuente Autor.

Figura O. Velocidad angular motor 1 vs tiempo de desplazamiento en trayectoria 1



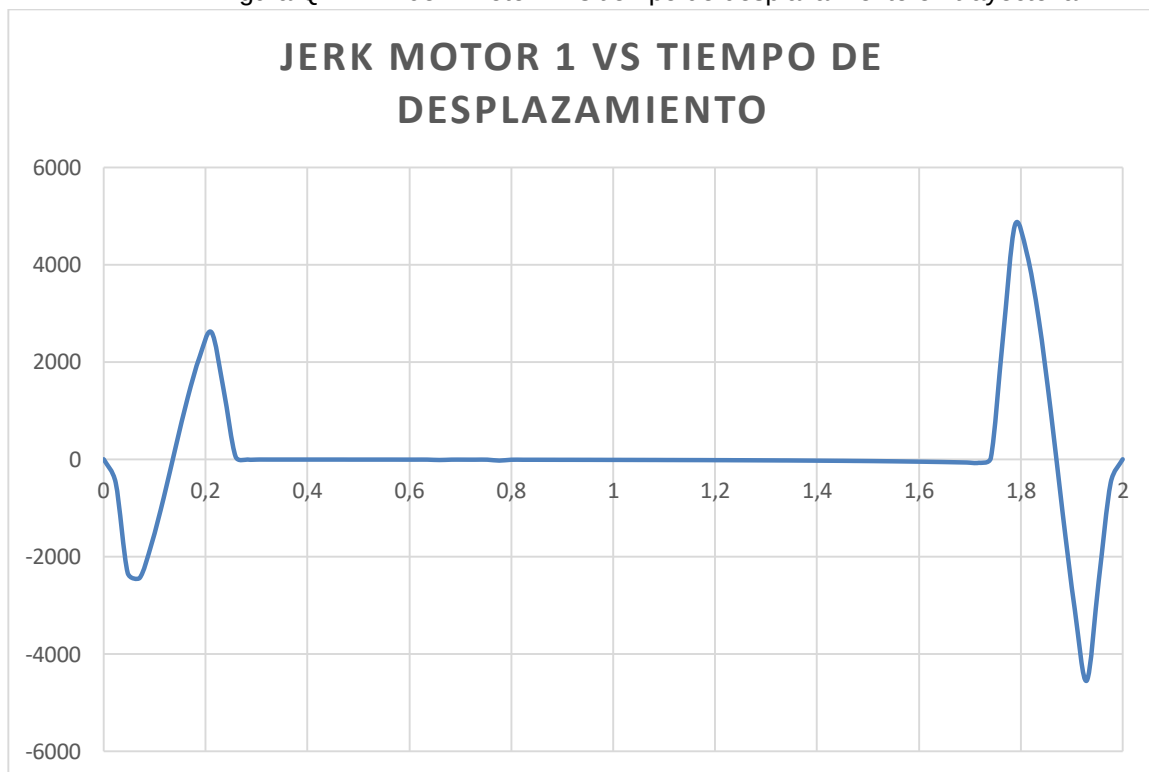
Fuente Autor.

Figura P. Aceleración motor 1 vs tiempo de desplazamiento en trayectoria 1



Fuente Autor.

Figura Q. Jerk motor 1 vs tiempo de desplazamiento en trayectoria 1

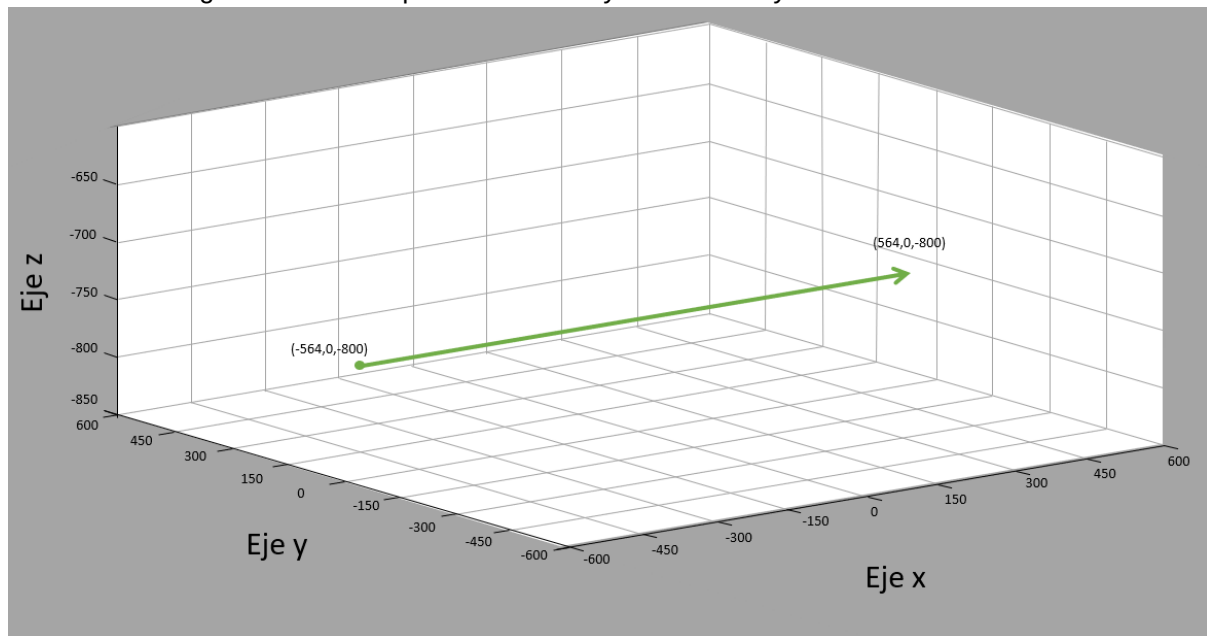


Fuente Autor

G.2 Trayectoria lineal A – B horizontal.

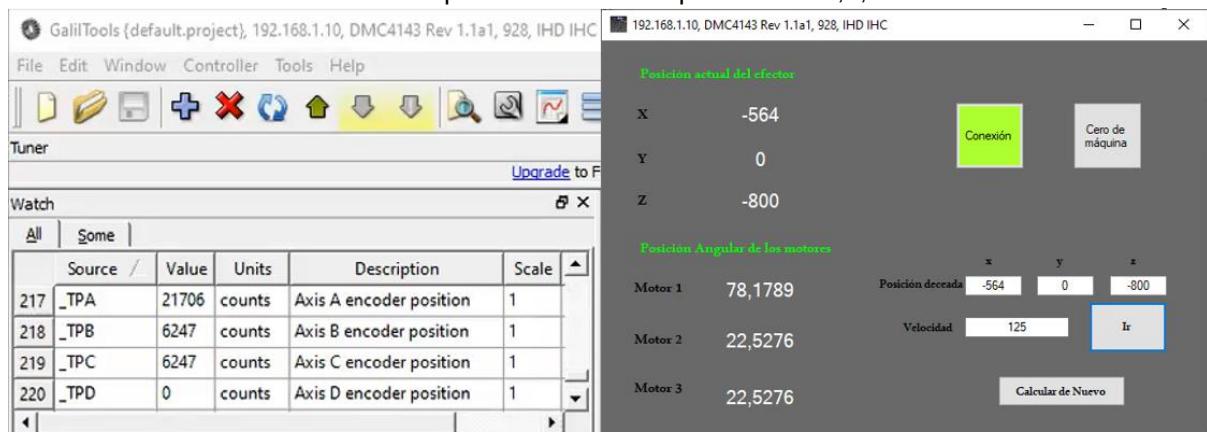
Esta trayectoria consiste en un desplazamiento desde (-564, 0, -800) hasta (564, 0, -800), siendo el recorrido de 1128 mm.

Figura R. Representación Trayectoria 2: Trayectoria lineal A – B horizontal.



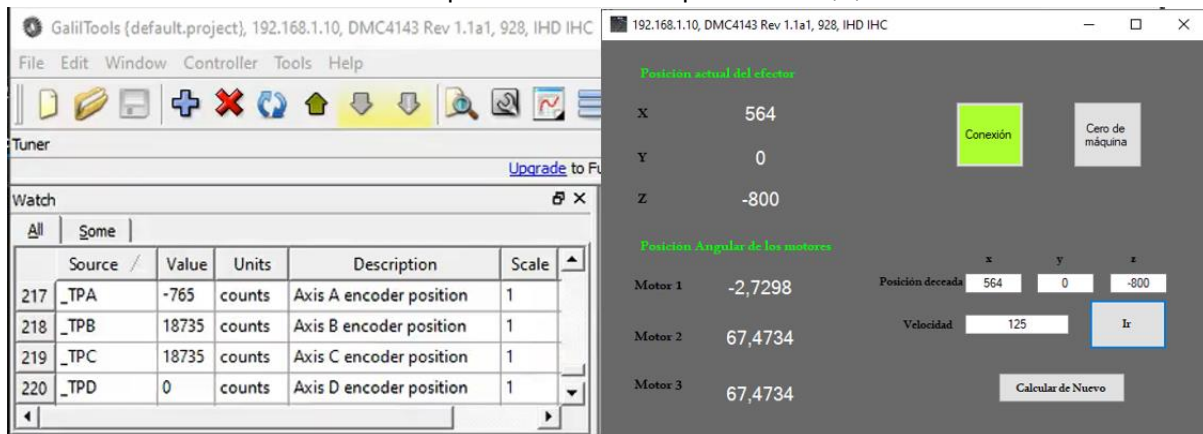
Fuente autor

Figura S. Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición -564,0,-800



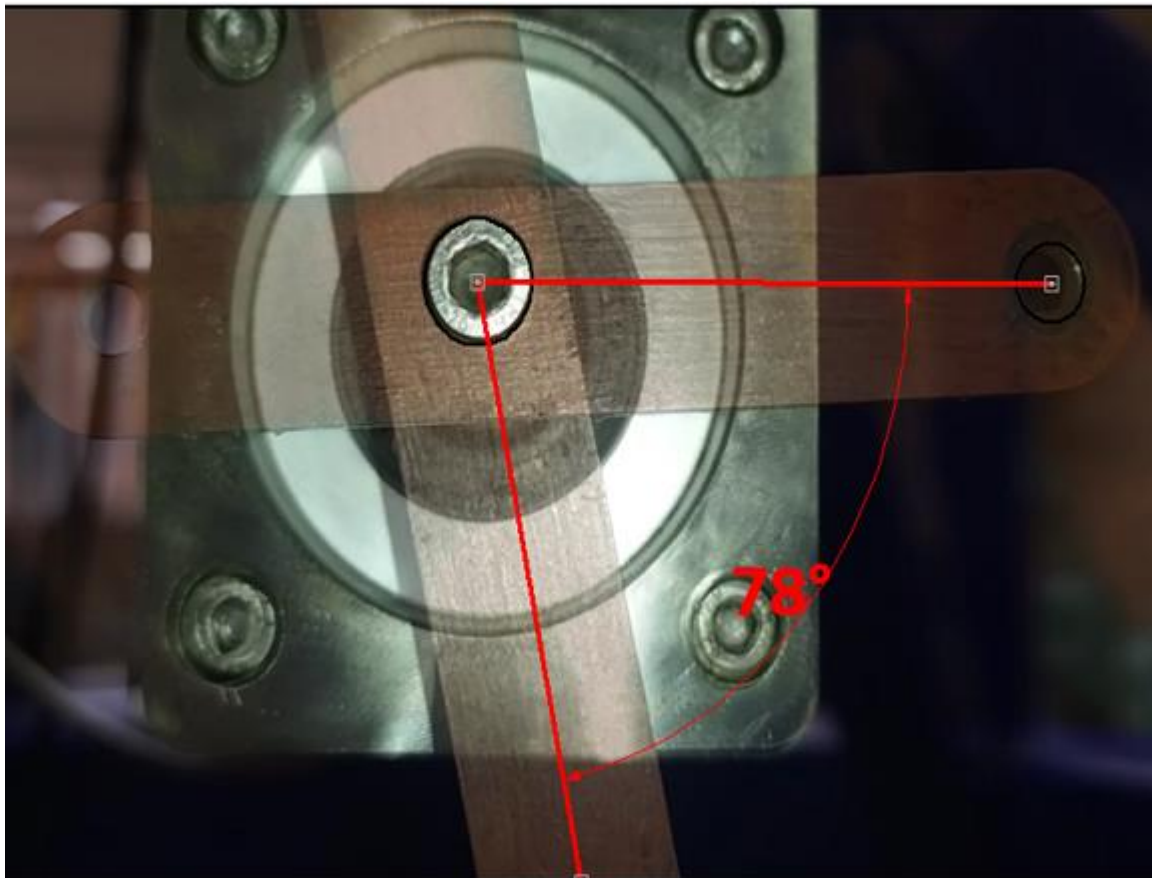
Fuente Autor.

Figura T. Valores encoders de los motores dados por GalilTools y valores arrojados por la interfaz en la posición 564,0,-800



Fuente Autor

Figura U. Comparación de la posición antes y después del movimiento de la Trayectoria 2 Trayectoria lineal A – B horizontal del motor 1.



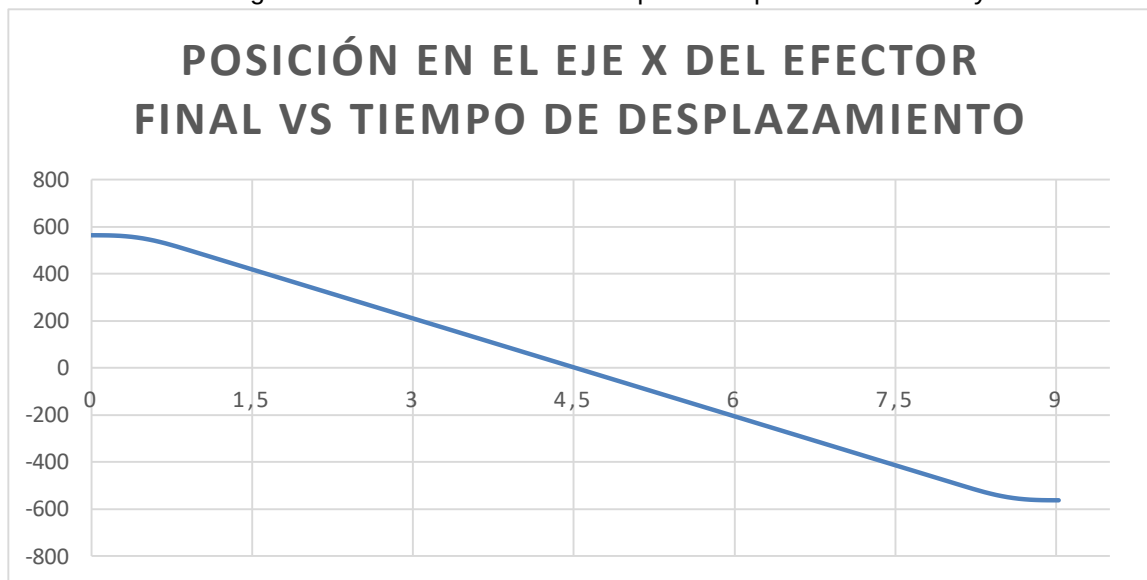
Fuente Autor.

Tabla B. Desplazamiento Angular del motor 1

	Desplazamiento angular del motor 1		
	Valor inicial	Valor final	Desplazamiento total
GalilTools	-2,754	78,1416	80,8956
Interfaz	-2,7298	78,1789	80,9087
Fotografías	-2,7	75,3	78

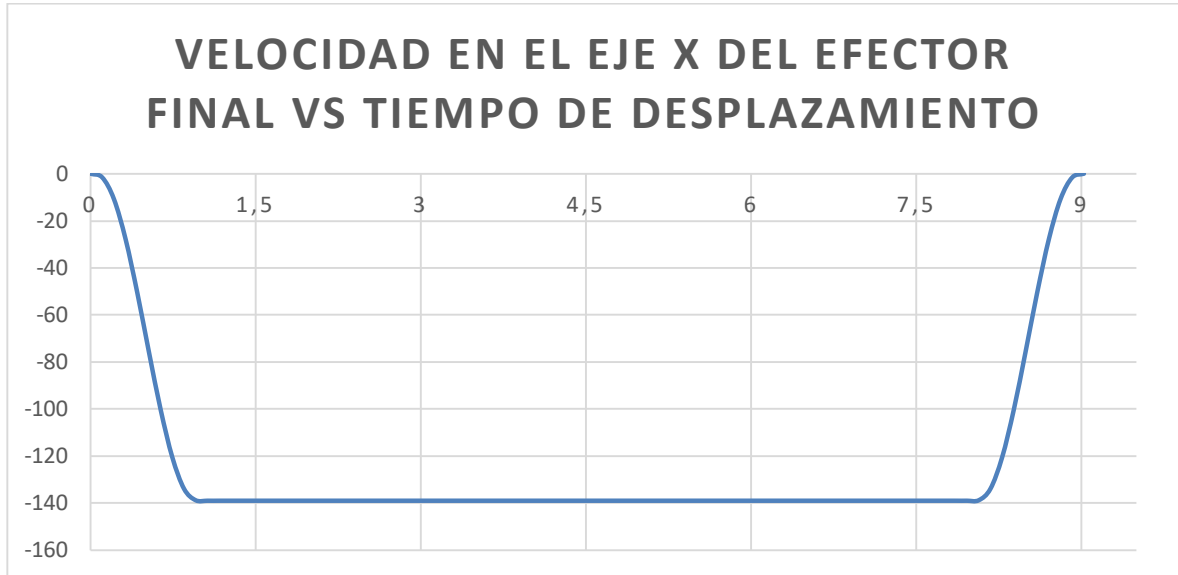
Fuente autor

Figura V. Posición X vs tiempo de desplazamiento en trayectoria 2



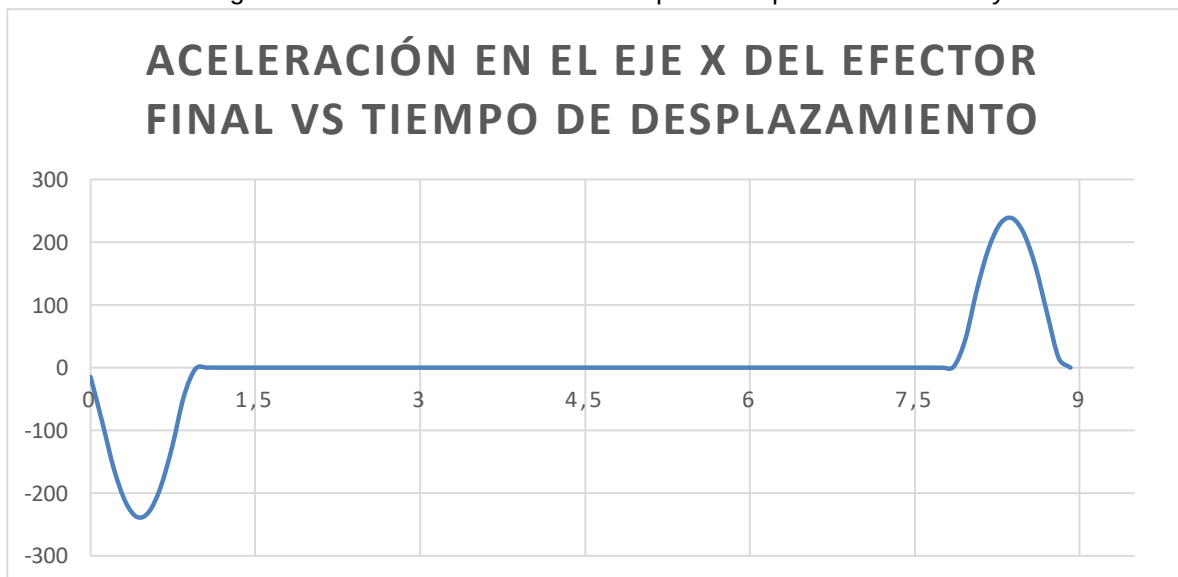
Fuente Autor.

Figura W. Velocidad X vs tiempo de desplazamiento en trayectoria 2



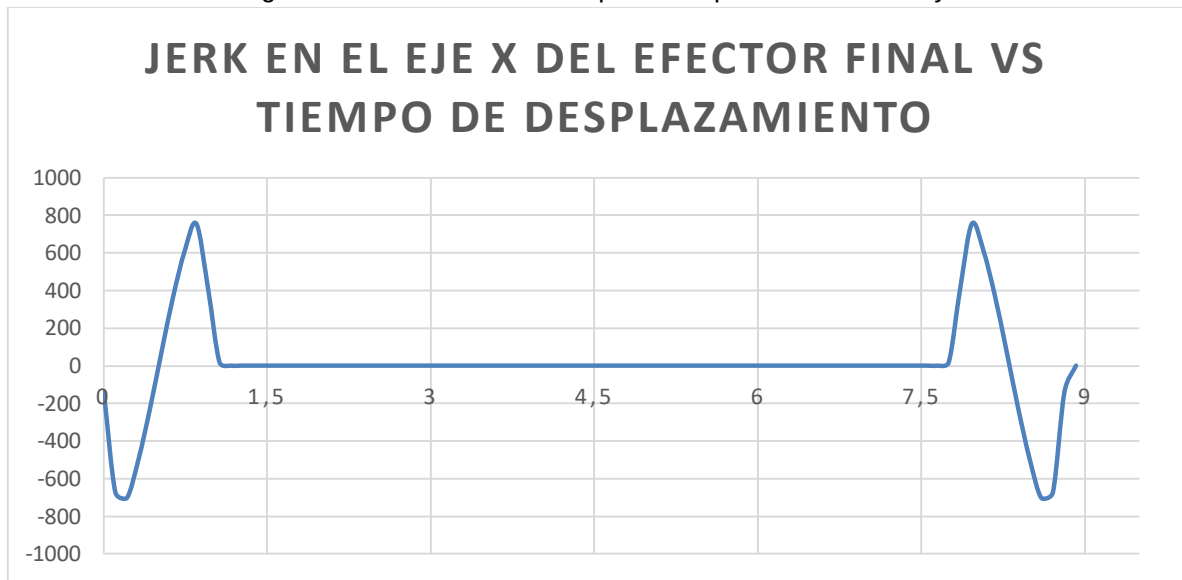
Fuente Autor.

Figura X. Aceleración X vs tiempo de desplazamiento en trayectoria 2



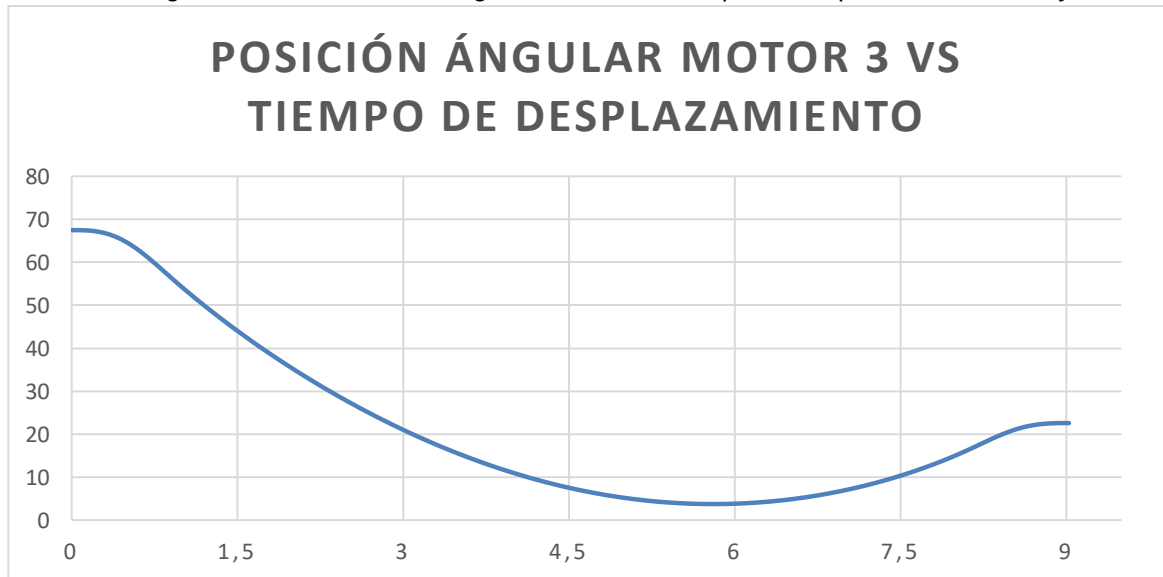
Fuente Autor.

Figura Y. Jerk X vs tiempo de desplazamiento en trayectoria 2



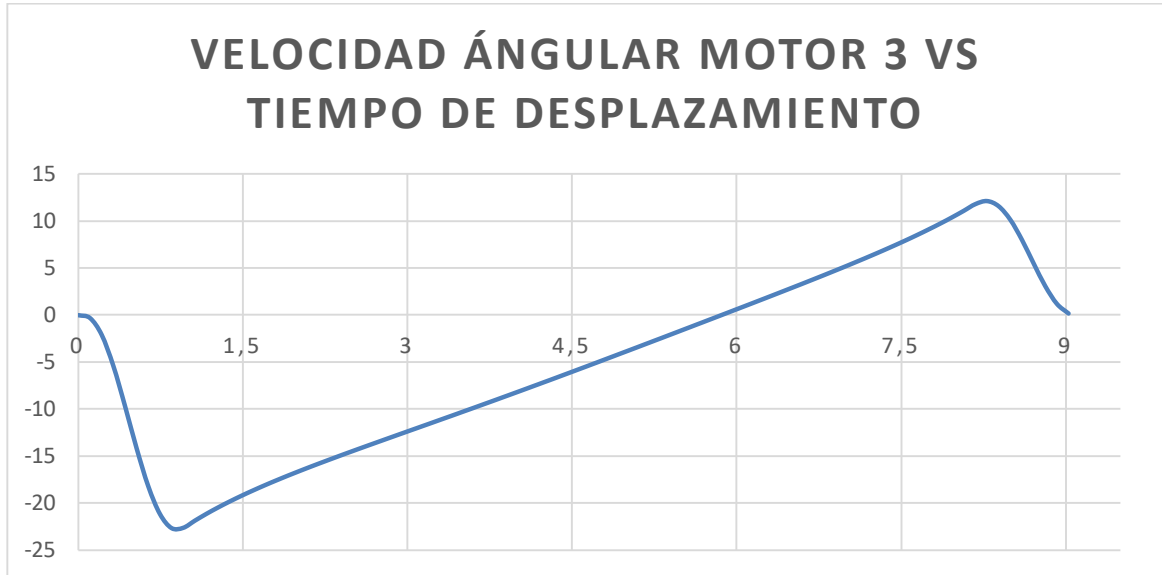
Fuente Autor

Figura Z. Posición angular motor 3 vs tiempo de desplazamiento en trayectoria 2



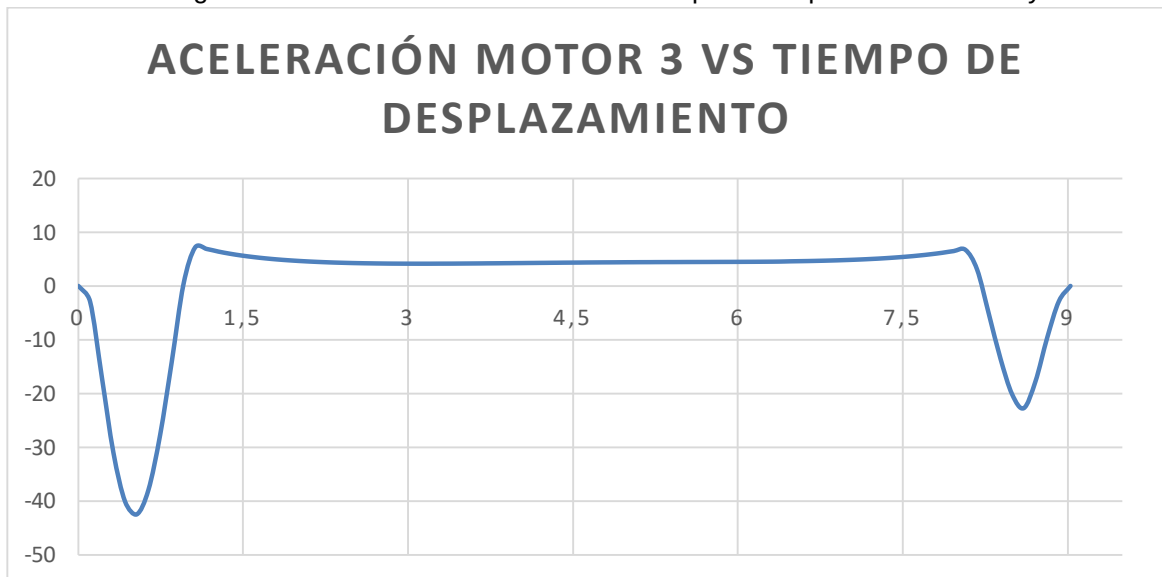
Fuente Autor.

Figura AA. Velocidad angular motor 3 vs tiempo de desplazamiento en trayectoria 2



Fuente Autor.

Figura BB. Aceleración motor 3 vs tiempo de desplazamiento en trayectoria 2



Fuente Autor.

Figura CC. Jerk motor 3 vs tiempo de desplazamiento en trayectoria 2



Fuente Autor