

# Control y trazado de trayectorias es un robot de 4 ejes: Diseño y simulación

Alulema Paredes, Marco Bryan  
uo304238@uniovi.es

**Resumen—** Este proyecto aborda el diseño y control de un robot simulado de 4 grados de libertad para el trazado de trayectorias en un tablero de dibujo. El robot, impulsado por servomotores, tiene como objetivo trazar trayectorias precisas y eficientes, manteniendo siempre la pluma en posición vertical y dentro de los límites permitidos del plano de trabajo.

El proyecto utiliza técnicas de conversión de unidades de ángulo, control de servos y cinemática directa e inversa. El control de servos se realiza mediante funciones que convierten ángulos de robot en señales PWM. Los cálculos de alcance permiten delimitar el espacio de trabajo para la pluma, mientras que la cinemática directa calcula las coordenadas exactas para las trayectorias. La cinemática inversa proporciona los ángulos necesarios para que el robot llegue a posiciones específicas.

El diseño de trayectorias rectilíneas, circulares y rectangulares se basa en cálculos geométricos y trigonométricos. Se lleva a cabo un estudio detallado para equilibrar precisión y velocidad, ajustando los parámetros para optimizar el rendimiento del robot. Los resultados demuestran que el sistema puede ejecutar trayectorias con precisión y eficiencia, ofreciendo una base sólida para futuras aplicaciones en robótica y automatización.

**Palabras Clave—** Cinemática, servomotores, trayectorias, robótica, automatización

## I. INTRODUCCIÓN.

El desarrollo de sistemas robóticos ha tenido un impacto significativo en una amplia gama de campos, desde la manufactura hasta la medicina. El uso de simuladores permite a los desarrolladores y estudiantes diseñar y probar sistemas robóticos en un entorno controlado antes de implementarlos en el mundo real. Este informe presenta un proyecto en el que se utiliza el simulador AlMarDrawingRobot para controlar un robot de 4 ejes y realizar trazados de trayectorias en un tablero de dibujo.

El simulador AlMarDrawingRobot simula un brazo robótico con 4 grados de libertad, compuesto por cuatro servomotores y tres brazos que se unen en las articulaciones base, hombro, codo y muñeca. El objetivo del proyecto es desarrollar un programa en MATLAB que permita el trazado de trayectorias en el tablero, manteniendo siempre el trazador en posición vertical y evitando trazar en la zona donde el robot no puede llegar. Para lograr esto, se emplean técnicas de control de servos,

cinemática directa e inversa, y diseño de trayectorias.

El sistema de simulación tiene características similares al robot educativo Tinkerkit Braccio, un robot de tipo brazo que utiliza servomotores para su movimiento. El sistema incluye un tablero para el trazado de figuras y un conjunto de comandos básicos para interactuar con el robot.

El proyecto incluye las siguientes tareas obligatorias y ampliaciones voluntarias:

Conversión de unidades de ángulo: Implica la transformación entre diferentes sistemas de ángulos.

Control de servos: Se requiere controlar la posición de los servos para mover el robot.

Cálculos de alcance: Determinar el alcance del robot y las limitaciones de movimiento.

Cinemática directa e inversa: Utilizar cálculos matemáticos para determinar las posiciones y trayectorias del robot.

Diseño y ejecución de trayectoria rectilínea: Crear un programa que permita el trazado de trayectorias rectilíneas.

Además, el estudio detallado de precisión y velocidad según parámetros de tiempo y distancia, la realización de trayectorias rectangulares y circulares, y otros estudios sobre la precisión y velocidad del robot.

El informe detalla el proceso seguido para el desarrollo del programa, incluyendo las herramientas utilizadas, el diseño del código y los resultados obtenidos. También se discuten las dificultades encontradas. El objetivo es proporcionar una visión completa del proceso de desarrollo y los resultados obtenidos al controlar el robot simulador AlMarDrawingRobot.

## II. CONVERSIÓN DE UNIDADES DE ÁNGULO.

Para la conversión entre ángulos de robot y ángulos de servomotores se debe tener en cuenta los 4 servomotores que dispone el robot y los ángulos de montaje ( $\theta_{mount}$ ) de cada uno. Como se muestra en la Figura 1 y su relación con ángulos de robot.

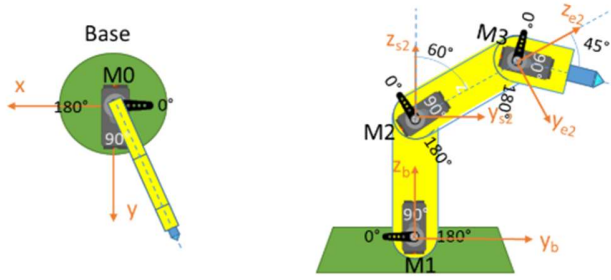


Figura 1. Montaje de servomotores

Los ángulos de robot son los ángulos que, independientemente del ángulo de montaje del robot, los eslabones del brazo se posicionan. Como se muestra en los 2 ejemplos de la Tabla 1, se puede identificar que los ángulos de robot son diferentes a los servos y a su vez se puede evidenciar en la Figura 2(a) la respuesta ante los ángulos  $[0\ 0\ 0\ 0]$  y en la Figura 2(b), la respuesta ante los ángulos  $[0\ 0\ 90\ 90]$  respectivamente.

Tabla 1. Ejemplo de relación entre ángulos de robot y ángulos de servomotor

N.	$\theta_{robot}$	S0	S1	S2	S3
1	$[0\ 0\ 0\ 0]$	90	90	30	45
2	$[0\ 0\ 90\ 90]$	90	90	120	135

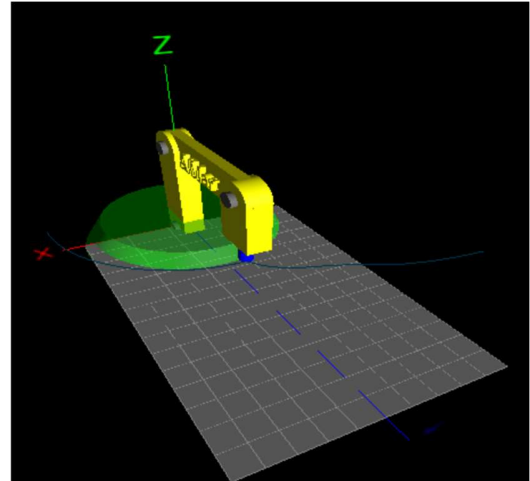
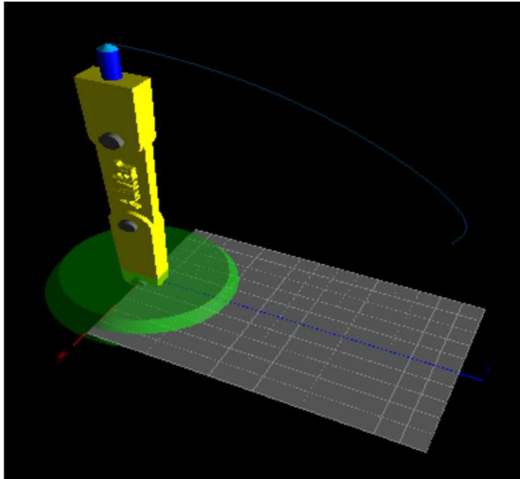


Figura 2. Posicionamiento del robot

Para la conversión de las unidades de ángulo entre los servomotores y ángulo de robot se ha calculado la ecuación de la recta según la Figura 3.

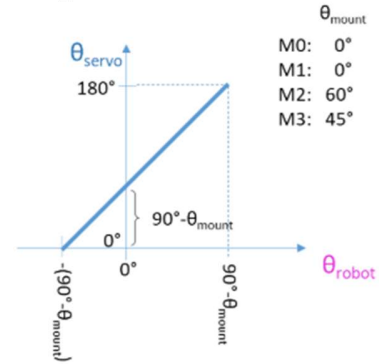


Figura 3. Relación de los ángulos de montaje en coordenadas de robot

Para esto la ecuación que responde a la gráfica es:

$$\theta_{servo} = \theta_{robot} + 90 + \theta_{mount} \quad (1)$$

Considerando que  $\theta_{robot}$  es un dato se obtiene el ángulo del servo correspondiente al  $\theta_{mount}$  según el servomotor que se necesite.

### III. CONTROL DE SERVOS

Para la conversión por software de los ángulos se ha realizado una función:

**function** [Ang] = Conv\_ToXY(TB,TS,TE,TW)

Donde:  $T_B: \theta_0$   
 $T_S: \theta_1$   
 $T_E: \theta_2$   
 $T_W: \theta_3$

En la que se tiene como entradas los ángulos de robot de: Base ( $\theta_0$ ), Shoulder ( $\theta_1$ ), Elbow ( $\theta_2$ ), Wrist ( $\theta_3$ ); y da como resultado un vector de 4 posiciones con los ángulos de servomotores en el orden correspondiente, este valor ya convertido en PWM, considerando que 1ms Ton =  $0^\circ$  servo y 2ms Ton =  $180^\circ$  servo.

Básicamente, se transforma el valor en grados a PWM para enviar a los diferentes servomotores. Se lo puede evidenciar en la Figura 2, el movimiento descrito desde la posición 1 y 2 de la Tabla 1.

Para ejecutar este movimiento se lo realiza con la función en Matlab de:

```
servos.SetPwm(0:3,Ang);
```

En esta función se envía al simulador un vector con 4 señales de diferentes ordenadas según el servo, de 0 a 3, respectivamente.

#### IV. CÁLCULOS DE ALCANCE

Para el cálculo del alcance máximo para graficar con la pluma, primero se definen las dimensiones del robot, en este caso los diferentes eslabones, como se muestra en la Figura 4.

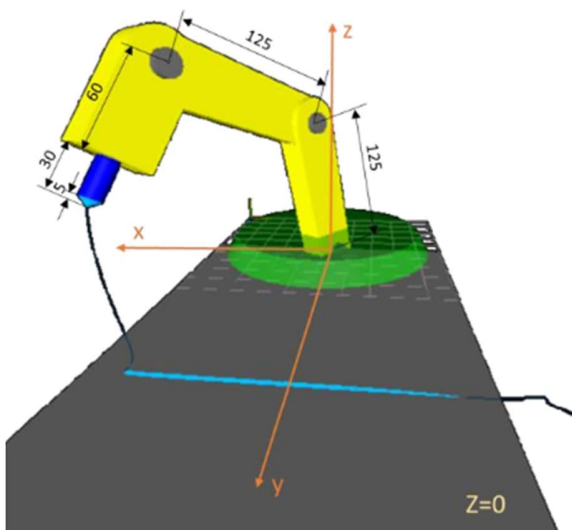


Figura 4. Dimensiones del robot

Se considera la restricción de que el eslabón 3 siempre debe estar paralelo al eje de coordenadas Z, para esto, se ha graficado y se obtiene que el alcance máximo de la pluma en el eje W es de 233.24, lo cual se lo puede evidenciar en la Figura 5, tomando en cuenta que la punta de la pluma está en la coordenada 0 del eje Z.

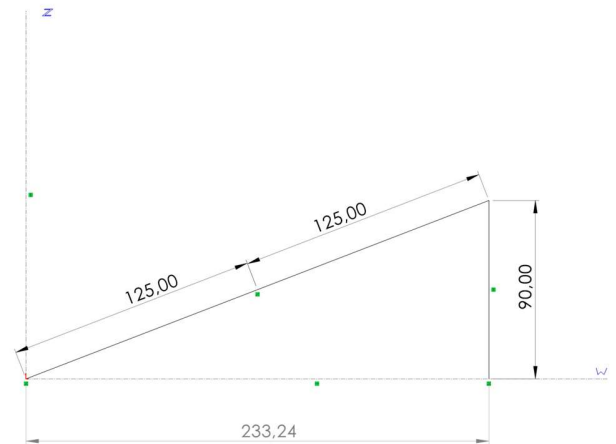


Figura 5. Alcance máximo, eje W

Este valor transferido a los ejes XY, se obtiene lo de la Figura 6, lo que se considera el alcance máximo en el radio de 233,24 en un ángulo de  $180^\circ$ .

Estos valores de alcance máximo responden solamente si la punta está en la coordenada 0 en Z y el eslabón 3 paralelo al eje Z.

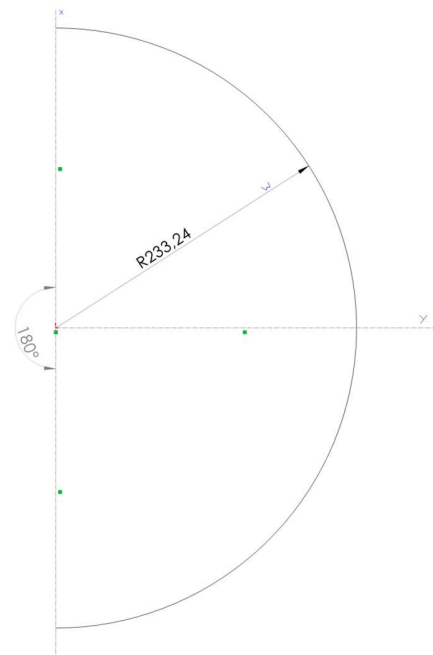


Figura 6. Alcance máximo en ejes XY

#### V. CINEMÁTICA DIRECTA

Para el cálculo de la cinemática directa, se ha creado la función:

```
function [xyz] =  
FnDirKinem(T_B,T_S,T_E,T_W,L1,L2,L3,LP)
```

Donde:  $T_B$ :  $\theta_0$

T\_S:  $\theta_1$   
 T\_E:  $\theta_2$   
 T\_W:  $\theta_3$   
 L1: Longitud eslabón 1  
 L2: Longitud eslabón 2  
 L3: Longitud eslabón 3  
 LP: Longitud pluma

Esta función da como resultado las coordenadas XYZ en las que se encuentra la punta del lápiz.

Para esto se tiene como entradas a la función los ángulos de robot en los que se encuentra en brazo, y por trigonometría se calcula la posición de la punta de la pluma tomando en cuenta los ángulos descritos por cada eslabón, como se puede observar en la Figura 7.

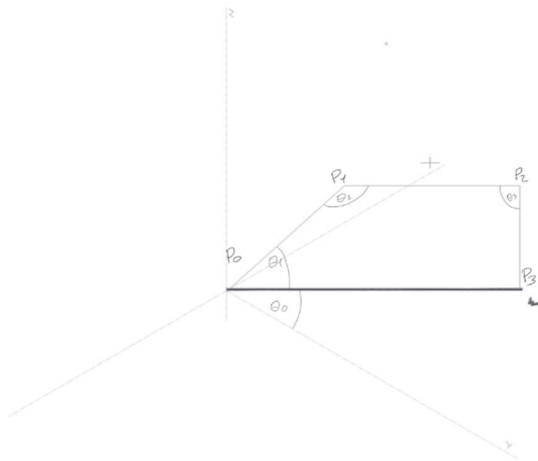


Figura 7. Representación de ángulos del robot

Para esto se han aplicado las siguientes formulas con vectores:

$$P_1[w, z] = L_1 * [\sin(\theta_1), \cos(\theta_1)] \quad (2)$$

$$P_2[w, z] = P_1 + L_2 * [\sin(\theta_1 + \theta_2), \cos(\theta_1 + \theta_2)] \quad (3)$$

$$P_3[w, z] = P_2 + L_3 * [\sin(\theta_1 + \theta_2 + \theta_3), \cos(\theta_1 + \theta_2 + \theta_3)] \quad (4)$$

En este caso ya se obtiene el punto más alejado desde el origen de coordenadas, que sería el P3[w], con este dato se calcula las componentes en X Y de la posición de la pluma y en el eje Z, vendría a ser P3[z].

Se obtiene la siguiente ecuación vectorial:

$$\text{Pos}[x, y, z] = [P_3[w] * \sin(\theta_0), P_1[w] + P_2[w] + P_3[w], P_3[z]] \quad (5)$$

Esta ecuación que está involucrada en la función nos da como resultado la posición actual en el eje de coordenadas XYZ de la punta de la pluma presente en el brazo robótico

## VI. CINEMÁTICA INVERSA

Para el cálculo de la cinemática inversa, en Matlab, se ha creado la siguiente función:

```
function [Th] =
FnInvKinem(X_des,Y_des,Z_des,L1,L2,L3,LP)
```

Donde: X\_des: Coordenada en X deseada  
 Y\_des: Coordenada en Y deseada  
 Z\_des: Coordenada en Z deseada  
 L1: Longitud eslabón 1  
 L2: Longitud eslabón 2  
 L3: Longitud eslabón 3  
 LP: Longitud pluma

En este caso, a diferencia de la cinemática directa, la función tiene como entrada los puntos a los que se desea que se dirija la pluma, y da como resultado en un vector los ángulos de robot que se requieren por cada articulación para que se pueda cumplir la posición deseada.

Básicamente se tiene como entrada la posición del punto P3 en XYZ, se toma en cuenta la Figura 7 para la generación de un sub plano que viene a ser el plano WZ con rotación  $\theta_0$ , como se puede ver en la Figura 8.

Del cual se obtiene las siguientes ecuaciones:

$$W_2 = \sqrt{X_{Des}^2 + Y_{Des}^2} \quad (6)$$

$$Z_2 = Z_{Des} + L_3 + LP \quad (7)$$

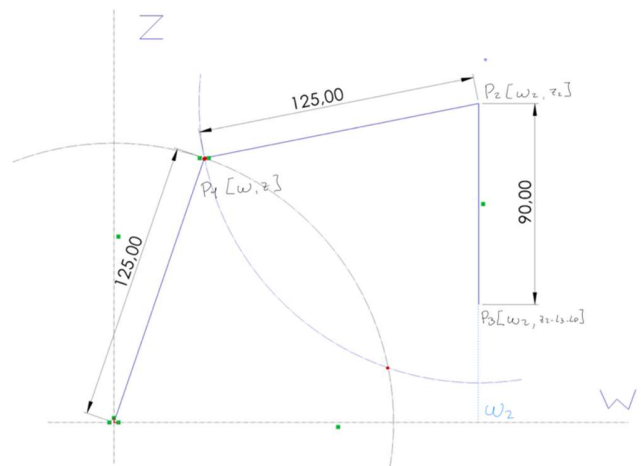


Figura 8. Representación de la intersección de circunferencias

Se toma como referencia inicial el Punto P2 y se establece lo siguiente, dentro del plano WZ:

Tabla 2. Relación de los puntos en el sub plano WZ

Punto	Plano Z	Plano Z
P3	$W_2$	$Z_2 - L_3 - L_P$
P2	$W_2$	$Z_2$
P1	$W_1$	$Z_1$

Las incógnitas, se establecen en las variables  $W_1$  y  $Z_1$ , las cuales se las resuelve con el método de intersección de circunferencias, entre la circunferencia que establece entre P0 con radio P1 y la circunferencia que se establece entre P2 con radio P1, en la Figura 8 se evidencia que se generan 2 puntos de intersección, para esto se escoge la variable  $Z_1$  de mayor valor de la solución de las ecuaciones cuadráticas, para estar seguros de que los dos eslabones estarán siempre por encima del plano XY, y la  $W_1$  se la calcula con el teorema de Pitágoras:

$$W_1 = \sqrt{L_1^2 - Z_1^2} \quad (8)$$

En este caso se llega a la problemática, de cuando  $W_1$  se coloca en el eje W negativo, ya que la fórmula 8 da como resultado un valor absoluto, así que no se puede identificar si  $W_1$  ha adquirido un valor negativo. Se lo soluciona con una pequeña validación cada vez que se calcula  $W_1$ , es verificado si la longitud del eslabón 2 ha cambiado su dimensión con la siguiente ecuación con teorema de Pitágoras:

$$L_{2\_val} = \sqrt{(W_2 - W_c)^2 + (Z_2 - Z_1)^2} \quad (9)$$

Si el valor  $L_{2\_val}$  es inferior al valor de  $L_2$ , se traduce en que el valor de  $W_1$  es un valor negativo, para lo cual se asigna el valor negativo y se realiza la del signo de  $W_1$  que corresponde.

Finalmente se establecen en las siguientes ecuaciones para el cálculo de ángulos de robot que corresponde a cada servomotor:

$$\theta_0 = \tan^{-1} \left( \frac{X_{Des}}{Y_{Des}} \right) \quad (10)$$

$$\theta_1 = \sin^{-1} \left( \frac{W_1}{L_1} \right) \quad (11)$$

$$\theta_2 = 90 - \theta_1 - \sin^{-1} \left( \frac{Z_2 - Z_1}{L_2} \right) \quad (12)$$

$$\theta_3 = 180 - \theta_1 - \theta_2 \quad (13)$$

Básicamente, estos valores se los guarda en un vector y es la salida de la función de la cinemática inversa.

Adicional se ha generado una función para enviar los datos al robot hacia donde debe posicionar la punta de la pluma:

**function** [] = MoveRobot(X,Y,Z)

Donde: X: Coordenada en X  
Y: Coordenada en Y

## Z: Coordenada en X

Dentro de esta función se hace el llamado a la función de cinemática inversa, por lo cual se calculan los ángulos y se envía esta información en PWM a los motores del robot.

### VII. DISEÑO Y EJECUCIÓN DE TRAYECTORIA RECTILÍNEA

Para el diseño de la trayectoria rectilínea, se establece solicitando el punto de inicio y final de esta bajo la siguiente función en Matlab:

**function** [] = FnLineDr(X\_1,Y\_1,X\_2,Y\_2)

Donde: X\_1: Coordenada en X inicial  
Y\_1: Coordenada en Y inicial  
X\_2: Coordenada en X final  
Y\_1: Coordenada en Y final

Para esto, lo primero en calcular es la longitud de la línea a graficar, lo cual se lo realiza con Pitágoras:

$$L_{rect} = \sqrt{(Y_2 - Y_1)^2 + (X_2 - X_1)^2} \quad (14)$$

Previamente se ha establecido que se realizaran pasos de 8 unidades cada 0.2 segundos, es decir que cada este intervalo de tiempo se generará nuevamente otro punto con una distancia equivalente a 8 unidades.

En este caso se divide la longitud a realizar con relación al paso y se obtiene la cantidad de puntos a generar (n) y a su vez se calcula la longitud en X y en Y de cada paso, para esto:

$$X_{stp} = \frac{X_2 - X_1}{n} \quad (15)$$

$$Y_{stp} = \frac{Y_2 - Y_1}{n} \quad (16)$$

Se aplica la siguiente fórmula para obtener las coordenadas en X Y con el step previamente mencionado.

$$\sum_{i=1}^n x(i) = i * X_{stp} + X_1 \quad (17)$$

$$\sum_{i=1}^n y(i) = i * Y_{stp} + Y_1 \quad (18)$$

A continuación, se escribe estas coordenadas para ser enviadas al robot haciendo uso de función la cinemática inversa, ya que se tiene los datos en XYZ, las funciones previamente mencionadas calculan los ángulos y distancia a mover cada eslabón, se debe tomar en cuenta que la punta de la pluma debe estar en Z -2 para que la gráfica sea visible.

En este caso cada 0.2 segundos se recalculan las siguientes

coordenadas y se envía al robot con la función MoveRobot, dando como resultado una línea como se lo puede ver en la Figura 9, donde la punta se ha movido entre los puntos:

XYZ: -100, 200, -2

XYZ: -100, 200, -2

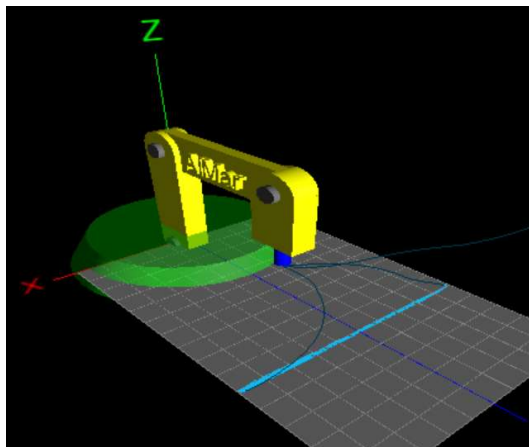


Figura 9. Trazado de trayectoria rectilínea

Adicional se puede evidenciar en la Figura 10, en la sección de color gris, mientras se grafica la recta, la punta se mantiene paralela al eje z, con mínimas variaciones menor a 1 grado de error y en la sección de color verde, se evidencia que la punta se mantiene en -2 del eje Z mientras se realiza la gráfica de la trayectoria rectilínea.

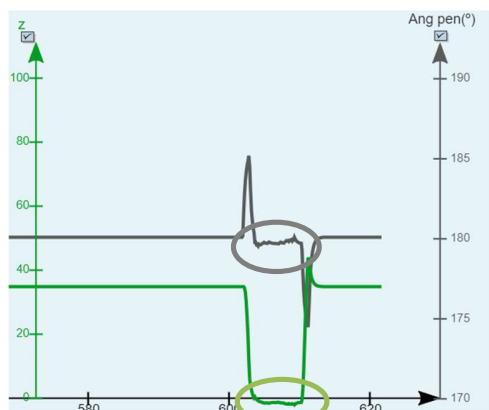


Figura 10. Posicionamiento de la pluma

#### VIII. ESTUDIO DETALLADO DE PRECISIÓN Y VELOCIDAD SEGÚN LOS PARÁMETROS TIEMPO Y DISTANCIA DE CADA PASO PARA TRAYECTORIA RECTILÍNEA

Para el estudio detallado de precisión y velocidad se ha realizado varios experimentos, entre ellos se destacan los que se muestran en la Tabla 3, en la que se puede evidenciar el error que se genera en los ejes Y Z, tomando en cuenta que la punta de pluma se debe desplazar entre los puntos:

XYZ: -100, 200, -2

XYZ: 100, 200, -2

Estratégicamente se ha seleccionado la trayectoria en Y de 200 constante, para facilitar el cálculo del error a través del movimiento de la pluma, ya que esta debe mantenerse en Y=200 durante todo el trazado. De esta manera se puede calcular el error en Y, a su vez el error absoluto al graficar trayectorias rectilíneas.

Tabla 3. Comparativa del error en diferentes configuraciones del timer y step

Time step	Step	Tiempo [ms]	Error en Y		Error en Z		Error Pen	
			Absoluto	Relativo	Absoluto	Relativo	Absoluto	Relativo
0,2	8	7846	0,65	0,327%	0,315	15,7%	0,283	0,2%
0,1	5	8002	0,59	0,295%	0,304	15,2%	0,286	0,2%
0,08	5	7270	0,65	0,323%	0,319	16,0%	0,293	0,4%
0,3	5	18551	0,35	0,173%	0,261	13,1%	0,286	0,2%

En este caso se ha seleccionado la configuración mencionada en la primera línea ya que se evidencia equilibrio entre error y velocidad para graficar la línea recta.

Se puede tomar en cuenta que la última fila de la Tabla 3, es la más precisa con relación a las anteriores, sin embargo, el tiempo de trazado es más del doble con relación a la primera, razón por la cual se ha seleccionado la primera configuración como la predeterminada.

Adicional se puede evidenciar en la última columna el error que existe al mantener el eslabón 3 paralelo al eje Z.

#### IX. REALIZACIÓN DE TRAYECTORIA RECTANGULAR

Para la generación de la trayectoria rectangular, básicamente se utiliza 4 veces la función del trazado de trayectorias rectilíneas, para esto, el usuario debe ingresar el punto XY inicial y el punto XY extremos del rectángulo, y se lleva a cabo la lógica mostrada en la Tabla 4.

Tabla 4. Lógica de trayectoria rectangular

Punto	Punto inicial	Punto final
Trayectoria 1	$X_1, Y_1$	$X_2, Y_1$
Trayectoria 2	$X_2, Y_1$	$X_2, Y_2$
Trayectoria 3	$X_2, Y_2$	$X_1, Y_2$
Trayectoria 4	$X_1, Y_2$	$X_1, Y_1$

La gráfica realizada por el robot se la puede visualizar en la Figura 11.

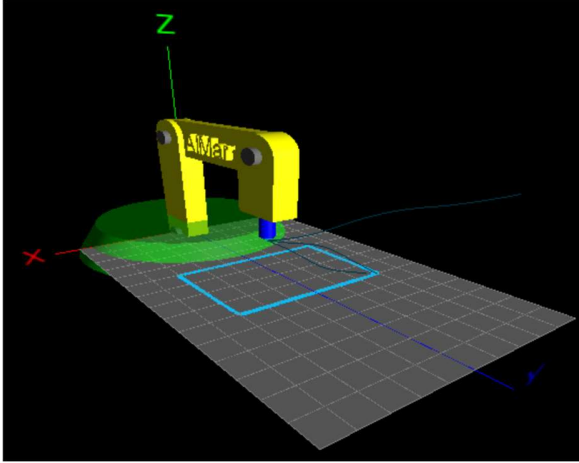


Figura 11. Trayectoria rectangular

Los datos ingresados en la figura anterior son:  
 XY: -60, 180  
 XY: 60, 100

#### X. REALIZACIÓN DE TRAYECTORIA CIRCULAR

Para el cálculo de una trayectoria circular, se solicita al usuario el punto central en XY del círculo y el radio a dibujar, para la trayectoria se ha generado en Matlab la siguiente función:

```
function [] = FnCircleDr(X_c,Y_c,Z_des,R_c)
```

Donde: X<sub>c</sub>: Coordenada en X del centro  
 Y<sub>c</sub>: Coordenada en Y del centro  
 Z<sub>des</sub>: Coordenada para graficar (-2)  
 R<sub>c</sub>: Radio del círculo

En este caso, al igual que en la trayectoria rectilínea, se calcula el perímetro del círculo y se la divide para el step que se haya definido, de esta manera se obtiene n pasos para la generación de la trayectoria, posterior se obtiene el ángulo entre paso y paso con la siguiente formula:

$$\text{Angle}_{\text{step}} = \frac{2*\pi}{n} [\text{rad}] \quad (19)$$

Para la generación de los puntos se utilizan las siguientes formulas por cada punto a generar

$$\sum_{i=1}^n \text{Angle} = (i-1) * \text{Angle}_{\text{step}} \quad (20)$$

$$\sum_{i=1}^n x(i) = X_c + R_c * \cos(\text{Angle}) \quad (21)$$

$$\sum_{i=1}^n y(i) = Y_c + R_c * \sin(\text{Angle}) \quad (22)$$

Por cada punto generado, se envía las coordenadas al robot para la ejecución del movimiento, como en la Figura 12, a través de la función MoveRobot.

En este caso el movimiento del robot responde al círculo dado los puntos:

X : 0  
 Y : 150  
 R : 50

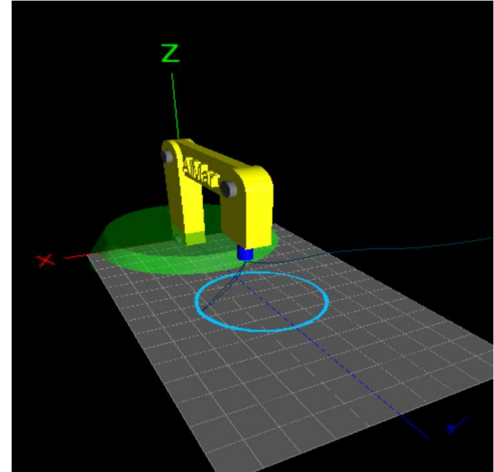


Figura 12. Trayectoria circular

#### XI. CONCLUSIONES

El proyecto desarrollado se centró en el trazado de trayectorias utilizando un robot de 4 grados de libertad simulado, abordando aspectos críticos como conversión de unidades de ángulo, control de servos y cinemática directa e inversa.

El proceso de conversión entre ángulos de robot y servomotores se basó en la relación entre los ángulos de montaje y la posición del servo. Esta relación es fundamental para garantizar la precisión del movimiento del robot y su adaptación a los requisitos del sistema. La fórmula generada para esta conversión permitió ajustar los valores de ángulos de robot a valores de servo necesarios para el control de las articulaciones.

El control de servos implica la generación de señales PWM para cada servomotor, permitiendo movimientos precisos en cada articulación. El uso de funciones especializadas para esta conversión facilita el manejo de los ángulos de robot y su traducción a señales PWM, permitiendo controlar el robot con exactitud y mantener la posición deseada durante el trazado de las trayectorias.

La cinemática directa permitió obtener las coordenadas XYZ de la pluma a partir de los ángulos de robot y las longitudes de los eslabones. Este proceso fue crucial para determinar las posiciones del robot y validar su capacidad para moverse de

manera controlada y precisa.

Por otro lado, la cinemática inversa fue utilizada para calcular los ángulos necesarios para que la pluma alcanzara puntos específicos en el tablero. Esta técnica fue esencial para el diseño de trayectorias, permitiendo al robot moverse de un punto a otro.

El diseño y la ejecución de la trayectoria rectilínea fueron logrados mediante cálculos de longitud y pasos, con una programación que asegura el movimiento suave y preciso. La evaluación de precisión y velocidad demostró que el sistema puede mantener una precisión aceptable mientras se optimiza la velocidad de ejecución.

En cuanto a las trayectorias rectangulares y circulares, se diseñaron funciones que permitieron trazar estas formas con precisión y consistencia, aplicando cálculos geométricos y trigonométricos para generar puntos de control. Esto permitió al robot trazar figuras complejas manteniendo la pluma en la posición adecuada.

En el proyecto se logró desarrollar un programa para el trazado de trayectorias con un robot de 4 grados de libertad, demostrando la eficacia de técnicas de control de servos, cinemática directa e inversa, y diseño de trayectorias. Los resultados obtenidos indican que el sistema es capaz de cumplir con los requisitos de precisión y velocidad, proporcionando una base sólida para futuras mejoras y aplicaciones en robótica y automatización.

## XII. REFERENCIAS

- [1] Universidad de Oviedo, "Documentación para el uso del Simulador AlMarDrawingRobot" Departamento de Ingeniería eléctrica, electrónica, de computadores y sistemas. Oviedo, España.
- [2] F. Ramos, Geometría.: Teoría Y Practica, AlfaOmega, 2012.
- [3] J. J. Craig, Introduction to Robotics: Mechanics and Control, United States of America: Pearson, 2005.