Universidad Nacional Autónoma de México

Facultad de ingeniería

ROBÓTICA

Proyecto Final

Profesor: Erick Peña Medina

Alumno:Raya Cruz José Luis

Reyes González Diego Iván

Romero Bernal Rocio Fabiola

Fecha de entrega:03/12/25

Contents

1 Introducción **2**
- 1.1 Modelo cinemático del robot SCARA 2
- 1.2 Definición de la trayectoria Bang–Bang 3
- 1.3 Cinemática de velocidades y aceleraciones 3
- 1.4 Índice de manipulabilidad 3
- 1.5 Modelo dinámico y cálculo de pares 3

2 Objetivo del proyecto **5**

3 Hipótesis **5**

4 Metas **5**

5 Desarrollo **5**
- 5.1 Síntesis del desarrollo 5
- 5.2 Nodo en Python para reproducción de trayectoria en ROS 2 11
- 5.3 Modelo URDF del robot SCARA 13
- 5.4 Archivo `launch` para visualización en ROS 2 15
- 5.5 Archivo de configuración de RViz2 16

6 Resultados **16**

7 Análisis de resultados **18**

8 Conclusiones individuales **18**

Proyecto Final

9 de diciembre de 2025

1. Introducción

La planificación de trayectorias en manipuladores industriales es un elemento clave para garantizar movimientos precisos, repetibles y seguros en tareas de manufactura y automatización. En particular, los robots tipo SCARA son adecuados para operaciones de ensamblado y manipulación en el plano, por lo que su análisis cinemático y dinámico permite explotar de forma eficiente su espacio de trabajo y los límites de operación de sus articulaciones.

En este proyecto se aborda el diseño y evaluación de una trayectoria de tipo Bang–Bang Parabolic Blend para un robot SCARA, con el objetivo de desplazar el efector final entre dos puntos definidos en el espacio de trabajo en un tiempo finito. Este tipo de trayectoria combina segmentos de aceleración y desaceleración máximas con transiciones parabólicas suaves, buscando un compromiso entre minimizar el tiempo de movimiento y reducir cambios bruscos en la aceleración que puedan inducir vibraciones o desgaste en los actuadores.

1.1. Modelo cinemático del robot SCARA

El manipulador considerado es un SCARA planar de tres grados de libertad, con dos articulaciones rotacionales coplanarias y una articulación rotacional asociada a la orientación del efector final. Los parámetros geométricos se fijan como $L_1 = 0.5$ m, $L_2 = 0.5$ m y $L_3 = 0.3$ m, que representan las longitudes efectivas de cada eslabón en el plano XY.

La postura del efector final se expresa mediante la posición (x_P, y_P) y el ángulo θ_P respecto al eje X. [file:31] Para una configuración articular $(\theta_1, \theta_2, \theta_3)$, la cinemática directa en el plano es:

$$x_P = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3)$$

$$y_P = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

$$\theta_P = \theta_1 + \theta_2 + \theta_3$$

Estas expresiones se utilizan al final del programa para reconstruir la trayectoria del efector en el espacio de trabajo y verificar que une correctamente los puntos P_1 y P_2.

Para resolver la cinemática inversa se parte de la posición deseada del efector (x_P, y_P, θ_P) y se calcula primero la posición del extremo del segundo eslabón (x_3, y_3) restando la contribución de L_3:

$$x_3 = x_P - L_3 \cos\theta_P, \qquad y_3 = y_P - L_3 \sin\theta_P$$

$$R = \sqrt{x_3^2 + y_3^2}$$

Usando el triángulo formado por L_1, L_2 y R, el ángulo θ_2 se obtiene con la ley del coseno:

$$\theta_2 = \pi - \arccos\left(\frac{R^2 - L_1^2 - L_2^2}{2L_1L_2}\right)$$

mientras que θ_1 se calcula a partir de los ángulos auxiliares α y ϕ:

$$\alpha = \arccos\left(\frac{R^2 + L_1^2 - L_2^2}{2L_1R}\right), \qquad \phi = \operatorname{atan2}(y_3, x_3), \qquad \theta_1 = \alpha - \phi$$

Finalmente, la tercera articulación se obtiene como:

$$\theta_3 = \theta_P - \theta_1 - \theta_2$$

Estas ecuaciones se aplican punto a punto sobre la trayectoria deseada $(x_P(t), y_P(t), \theta_P(t))$, generando las trayectorias articulares $\theta_i(t)$ que se almacenan en los vectores `theta1`, `theta2` y `theta3`.

1.2. Definición de la trayectoria Bang–Bang

La tarea consiste en mover el efector desde un punto inicial $P_1 = (x_{in}, y_{in}, \theta_{P,in})$ hasta un punto final $P_2 = (x_{fin}, y_{fin}, \theta_{P,fin})$ en un tiempo total $t_{\text{total}} = 20$ s. [file:31] Para ello se adopta una trayectoria de tipo Bang–Bang Parabolic Blend en el espacio de trabajo, donde la primera mitad del movimiento corresponde a una fase de aceleración máxima y la segunda mitad a una fase de desaceleración simétrica.

En el intervalo $0 \leq t \leq t_{\text{total}}/2$ la posición del efector se interpola mediante:

$$x_P(t) = x_{in} + \frac{2t^2}{t_{\text{total}}^2}(x_{fin} - x_{in}), \quad y_P(t) = y_{in} + \frac{2t^2}{t_{\text{total}}^2}(y_{fin} - y_{in}), \quad \theta_P(t) = \theta_{P,in} + \frac{2t^2}{t_{\text{total}}^2}(\theta_{P,fin} - \theta_{P,in})$$

mientras que para $t_{\text{total}}/2 < t \leq t_{\text{total}}$ se emplea una expresión equivalente que invierte el signo de la aceleración y garantiza que el efector llegue exactamente a P_2 con velocidad nula. Esta construcción genera perfiles de velocidad con forma trapezoidal y aceleraciones que cambian de signo en el punto medio, característica típica de los controles Bang–Bang mínimos en tiempo.

La trayectoria se evalúa con un periodo de muestreo $t_{in} = 0.1$ s, obteniendo un total de $n = 201$ muestras que se usan en todo el análisis cinemático y dinámico.

1.3. Cinemática de velocidades y aceleraciones

Una vez obtenidas las trayectorias articulares $\theta_i(t)$, las velocidades $\dot{\theta}_i(t)$ y aceleraciones $\ddot{\theta}_i(t)$ se calculan mediante derivación numérica usando diferencias centrales para los puntos interiores y diferencias hacia adelante/atrás en los extremos. [file:31] Este esquema aproxima:

$$\dot{\theta}_i(t_k) \approx \frac{\theta_i(t_{k+1}) - \theta_i(t_{k-1})}{2\Delta t}, \qquad \ddot{\theta}_i(t_k) \approx \frac{\dot{\theta}_i(t_{k+1}) - \dot{\theta}_i(t_{k-1})}{2\Delta t}$$

donde $\Delta t = t_{in}$ es el paso de muestreo.

Estos perfiles de velocidad y aceleración permiten verificar que la trayectoria Bang–Bang efectivamente acelera en la primera mitad del ciclo y desacelera en la segunda, así como alimentar el modelo dinámico que calcula los pares requeridos en cada articulación.

1.4. Índice de manipulabilidad

Para evaluar la capacidad del robot de generar velocidades en el efector a lo largo de la trayectoria se utiliza el índice de manipulabilidad propuesto por Yoshikawa, que en el caso del SCARA planar se reduce a:

$$w(t) = L_1 L_2 \left|\sin\left(\theta_2(t)\right)\right|$$

Este índice es proporcional al área del paralelogramo formado por los vectores de los eslabones y se anula cuando $\theta_2 = 0$ o $\theta_2 = \pi$, es decir, cuando el robot se encuentra completamente extendido o plegado y entra en una configuración singular. En el programa se normaliza de forma cualitativa para comparar su evolución con el rango recomendado $0.7 \leq w \leq 1.0$, lo que permite juzgar si la trayectoria aprovecha zonas de buena destreza cinemática.

1.5. Modelo dinámico y cálculo de pares

La dinámica del SCARA se modela mediante la formulación de Euler–Lagrange en el espacio articular, considerando masas concentradas en cada eslabón y momentos de inercia aproximados como varillas delgadas:

$$I_i = \frac{1}{12} m_i L_i^2, \qquad i = 1, 2, 3$$

A partir de estas hipótesis se construye la matriz de inercia $M(q)$ con términos acoplados entre las dos primeras articulaciones:

$$M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$

donde, por ejemplo,

$$M_{11} = I_1 + I_2 + I_3 + m_2 L_1^2 + m_3(L_1^2 + L_2^2) + 2m_3 L_1 L_2 \cos\theta_2$$

$$M_{12} = I_2 + I_3 + m_3 L_2^2 + m_3 L_1 L_2 \cos\theta_2, \quad M_{22} = I_2 + I_3 + m_3 L_2^2,$$

mientras que los elementos restantes se completan por simetría e incluyendo la contribución del tercer eslabón.

Los efectos de Coriolis y centrífuga se agrupan en la matriz $C(q, \dot{q})$, cuyo término clave depende de

$$h = -m_3 L_1 L_2 \sin\theta_2$$

y genera acoplamientos proporcionales a los productos $\dot{\theta}_1\dot{\theta}_2$ y $\dot{\theta}_2^2$ entre las dos primeras articulaciones. Despreciando la gravedad (movimiento en plano horizontal), la ecuación de movimiento queda:

$$\tau = M(q)\,\ddot{q} + C(q, \dot{q})\,\dot{q}$$

donde $q = [\theta_1, \theta_2, \theta_3]^T$, $\dot{q}$ y $\ddot{q}$ son los vectores de velocidades y aceleraciones articulares calculados previamente.

En cada instante de tiempo se evalúan $M(q)$ y $C(q, \dot{q})$ y se obtiene el vector de pares $\tau = [\tau_1, \tau_2, \tau_3]^T$, que representa el esfuerzo que deben entregar los motores para seguir la trayectoria Bang–Bang especificada. [file:31] A partir de estos pares y de las velocidades articulares se calcula la potencia mecánica instantánea de cada motor:

$$P_i(t) = |\tau_i(t)\,\dot{\theta}_i(t)|, \qquad P_{\text{total}}(t) = P_1(t) + P_2(t) + P_3(t)$$

lo que permite cuantificar tanto el pico de potencia requerido como la energía total consumida durante la ejecución de la trayectoria.

En resumen, a partir del modelo cinemático directo e inverso del robot, se analiza el índice de manipulabilidad a lo largo de la trayectoria propuesta, con el fin de evaluar la destreza del manipulador y evitar configuraciones cercanas a singularidades. Además, se determinan las características cinemáticas y dinámicas de los actuadores —velocidad, par y potencia mecánica requerida— para verificar si el SCARA puede ejecutar la tarea de seguimiento de trayectoria sin exceder sus límites de diseño.

Finalmente, la implementación numérica y la simulación de la trayectoria se realizan en un entorno de software que permite visualizar el movimiento del robot y registrar las variables relevantes para el análisis. [file:31] De esta manera, el proyecto integra conceptos de modelado, simulación y control de sistemas mecatrónicos, y contribuye al desarrollo de competencias profesionales como el pensamiento crítico, la solución de problemas de ingeniería y el uso de herramientas de cómputo científico.

Perfiles profesionales involucrados

El desarrollo de este proyecto integra actividades de modelado matemático, simulación numérica, programación y validación experimental en un entorno robótico, por lo que se relaciona con varios perfiles profesionales típicos de la ingeniería mecatrónica y la automatización industrial.

Ingeniero de control y robótica

Es el responsable de formular los modelos cinemáticos y dinámicos del manipulador, diseñar la trayectoria Bang–Bang en el espacio de trabajo y evaluar el índice de manipulabilidad, las velocidades articulares, los pares y la potencia requerida en cada motor. Este perfil también define los criterios de desempeño del robot, verifica la ausencia de configuraciones singulares y propone especificaciones iniciales para los actuadores y la electrónica de potencia.

Ingeniero de automatización y sistemas embebidos

Se encarga de la implementación práctica de las estrategias de movimiento en plataformas de cómputo y control, integrando los resultados de MATLAB con controladores industriales o tarjetas embebidas. En un proyecto como el presente, este perfil puede adaptar los perfiles de par y velocidad obtenidos a controladores de motores reales, así como diseñar rutinas de arranque, paro y supervisión segura del SCARA en una celda de trabajo.

Desarrollador de software para robótica (ROS/URDF)

Tiene como función construir los modelos URDF del robot, configurar los nodos de ROS 2 para reproducir la trayectoria calculada y preparar archivos `launch` y configuraciones de RViz para la simulación visual. Además, este perfil se ocupa de la comunicación entre los datos generados en MATLAB y el entorno ROS, garantizando que la reproducción de la trayectoria en el simulador refleje fielmente los resultados teóricos.

Ingeniero de integración y pruebas

Coordina la validación del sistema completo, comparando las gráficas y resultados numéricos (manipulabilidad, pares y potencia) con el comportamiento observado en la simulación y, en su caso, en el prototipo físico. Este perfil documenta los casos de prueba, identifica posibles mejoras en la trayectoria o en la selección de motores y asegura que el proyecto cumpla con los objetivos académicos y con el perfil de egreso requerido.

2. Objetivo del proyecto

Evaluar las capacidades cinemáticas de un robot SCARA mediante el índice de manipulabilidad y determinar las características cinemáticas, dinámicas y de potencia de sus actuadores para el seguimiento de una trayectoria Bang–Bang Parabolic Blend entre dos puntos de su espacio de trabajo.

3. Hipótesis

Es posible evaluar cuantitativamente las capacidades cinemáticas de un robot SCARA mediante el índice de manipulabilidad y, con base en este análisis, determinar si el manipulador es capaz de realizar de forma segura y precisa el seguimiento de una trayectoria simple definida en su espacio de trabajo. Si el índice de manipulabilidad se mantiene en valores adecuados durante la ejecución de la trayectoria, entonces el robot podrá completar el movimiento respetando sus restricciones de posición, velocidad y par articular.

4. Metas

- Proponer el lugar geométrico de una trayectoria Bang–Bang Parabolic Blend que lleve al efector final del robot desde un punto inicial P_1 hasta un punto final P_2 dentro de su espacio de trabajo.
- Evaluar cualitativamente las capacidades cinemáticas del SCARA mediante el cálculo del índice de manipulabilidad a lo largo de la trayectoria propuesta.
- Determinar los perfiles de velocidad y par que deben cubrir los motores de cada articulación para cumplir con el seguimiento de la trayectoria en el tiempo total de simulación definido.
- Estimar la potencia mecánica requerida por cada motor y la potencia total demandada por el robot durante el movimiento.

5. Desarrollo

En esta sección se describen las principales ecuaciones empleadas para modelar la cinemática, la dinámica y la evaluación de desempeño del robot SCARA sujeto a una trayectoria tipo Bang–Bang con mezcla parabólica en el espacio de trabajo. El objetivo es vincular cada bloque del código con los fundamentos teóricos correspondientes para justificar los resultados de posiciones, manipulabilidad, pares y potencia obtenidos en la simulación.

5.1. Síntesis del desarrollo

El proyecto consiste en tomar un robot SCARA “ideal” y analizar si realmente puede seguir una trayectoria exigente entre dos puntos, no solo desde la parte geométrica, sino también observando qué tan forzados trabajan sus motores. Para ello se construye una cadena completa que va desde el modelo matemático en MATLAB hasta la simulación visual en ROS 2 con un modelo URDF del robot.

Primero, en MATLAB se definen las longitudes, masas e inercias de los tres eslabones del SCARA y se plantean las poses inicial y final del efector final en el plano XY. Con esos datos se genera una trayectoria tipo Bang–Bang Parabolic Blend en el espacio de trabajo, que acelera en la primera mitad del tiempo y desacelera en la segunda, logrando un movimiento rápido pero con un perfil de velocidad bien definido. Para cada punto de esa trayectoria se resuelve la cinemática inversa y se obtienen los ángulos articulares $\theta_1, \theta_2, \theta_3$, a partir de los cuales se aproximan numéricamente las velocidades y aceleraciones de cada junta.

Con las trayectorias articulares ya calculadas, el código aplica un modelo dinámico basado en Euler–Lagrange para formar la matriz de inercia y los términos de Coriolis del SCARA, y con ello calcula los pares que deberían entregar los motores en cada instante. A partir de esos pares y de las velocidades se obtiene la potencia mecánica por motor y la potencia total del robot, y al mismo tiempo se evalúa el índice de manipulabilidad $w = L_1 L_2 |\sin(\theta_2)|$ para verificar si la trayectoria evita configuraciones singulares. Las gráficas finales permiten observar cómo cambian posiciones, velocidades, aceleraciones, pares, potencia y manipulabilidad durante todo el movimiento y si los valores se mantienen dentro de rangos razonables.

La segunda parte del proyecto traslada esa trayectoria del entorno de MATLAB a un entorno robótico. El script guarda en un archivo CSV el tiempo, los ángulos de cada articulación, la manipulabilidad y la potencia total, y posteriormente un nodo en Python para ROS 2 lee dicho archivo y publica mensajes `JointTrajectory` a un controlador de juntas. En paralelo, se define un modelo URDF del SCARA con tres eslabones rotacionales y se lanza RViz con un archivo de configuración básico para visualizar la grilla y el robot. Al ejecutar el archivo `launch`, el modelo URDF se anima siguiendo exactamente la trayectoria calculada en MATLAB, de modo que se puede comparar lo que muestran las gráficas con el movimiento observado en la simulación.

En conjunto, el funcionamiento del proyecto ilustra el flujo típico de un problema de robótica: partir de un modelo matemático, diseñar una trayectoria que respete las restricciones del robot, evaluar cinemática, manipulabilidad y dinámica, y finalmente validar el resultado en una simulación tridimensional utilizando herramientas modernas como MATLAB, ROS 2 y RViz.

Listing 1: Cálculo de trayectoria Bang–Bang, manipulabilidad y dinámica del robot SCARA.

```
clear all; close all; clc;

L1 = 0.5;
L2 = 0.5;
L3 = 0.3;

x_in = 0.4;
y_in = -0.1;
theta_P_in = pi/2;

x_fin = 0.0;
y_fin = -1.3;
theta_P_fin = -pi/2;

t_total = 20;
t_in = 0.1;
t_sim = 0:t_in:t_total;
n = length(t_sim);

m1 = 0.8;
m2 = 0.6;
m3 = 0.4;

I1 = (1/12)*m1*L1^2;
I2 = (1/12)*m2*L2^2;
I3 = (1/12)*m3*L3^2;

xp = zeros(1, n);
yp = zeros(1, n);
theta_P = zeros(1, n);

theta1 = zeros(1, n);
```

```
theta2 = zeros(1, n);
theta3 = zeros(1, n);

theta1_v = zeros(1, n);
theta2_v = zeros(1, n);
theta3_v = zeros(1, n);

theta1_a = zeros(1, n);
theta2_a = zeros(1, n);
theta3_a = zeros(1, n);

w = zeros(1, n);
tao1 = zeros(1, n);
tao2 = zeros(1, n);
tao3 = zeros(1, n);
pot1 = zeros(1, n);
pot2 = zeros(1, n);
pot3 = zeros(1, n);
pot_total = zeros(1, n);

for i = 1:n
    t = t_sim(i);

    if t <= t_total / 2
        xp(i) = x_in + (2 * t^2 / t_total^2) * (x_fin - x_in);
        yp(i) = y_in + (2 * t^2 / t_total^2) * (y_fin - y_in);
        theta_P(i) = theta_P_in + (2 * t^2 / t_total^2) * (theta_P_fin - theta_P_in);
    else
        xp(i) = x_fin + ((4 * t / t_total - 2 * t^2 / t_total^2) - 2) * (x_fin - x_in)
            ;
        yp(i) = y_fin + ((4 * t / t_total - 2 * t^2 / t_total^2) - 2) * (y_fin - y_in)
            ;
        theta_P(i) = theta_P_fin + ((4 * t / t_total - 2 * t^2 / t_total^2) - 2) * (
            theta_P_fin - theta_P_in);
    end
end

for i = 1:n
    x3 = xp(i) - L3 * cos(theta_P(i));
    y3 = yp(i) - L3 * sin(theta_P(i));
    R = sqrt(x3^2 + y3^2);

    arg = (R^2 - L1^2 - L2^2) / (2 * L1 * L2);
    if arg > 1
        arg = 1;
    elseif arg < -1
        arg = -1;
    end
    theta2(i) = pi - acos(arg);

    alfa = acos((R^2 + L1^2 - L2^2) / (2 * L1 * R));
    phi = atan2(y3, x3);
    theta1(i) = alfa - phi;

    theta3(i) = theta_P(i) - theta1(i) - theta2(i);

    w(i) = abs(sin(theta2(i)));
end

for i = 2:n-1
```

```
    theta1_v(i) = (theta1(i+1) - theta1(i-1)) / (2*t_in);
    theta2_v(i) = (theta2(i+1) - theta2(i-1)) / (2*t_in);
    theta3_v(i) = (theta3(i+1) - theta3(i-1)) / (2*t_in);
end

theta1_v(1) = (theta1(2) - theta1(1)) / t_in;
theta2_v(1) = (theta2(2) - theta2(1)) / t_in;
theta3_v(1) = (theta3(2) - theta3(1)) / t_in;
theta1_v(end) = (theta1(end) - theta1(end-1)) / t_in;
theta2_v(end) = (theta2(end) - theta2(end-1)) / t_in;
theta3_v(end) = (theta3(end) - theta3(end-1)) / t_in;

for i = 2:n-1
    theta1_a(i) = (theta1_v(i+1) - theta1_v(i-1)) / (2*t_in);
    theta2_a(i) = (theta2_v(i+1) - theta2_v(i-1)) / (2*t_in);
    theta3_a(i) = (theta3_v(i+1) - theta3_v(i-1)) / (2*t_in);
end

theta1_a(1) = (theta1_v(2) - theta1_v(1)) / t_in;
theta2_a(1) = (theta2_v(2) - theta2_v(1)) / t_in;
theta3_a(1) = (theta3_v(2) - theta3_v(1)) / t_in;
theta1_a(end) = (theta1_v(end) - theta1_v(end-1)) / t_in;
theta2_a(end) = (theta2_v(end) - theta2_v(end-1)) / t_in;
theta3_a(end) = (theta3_v(end) - theta3_v(end-1)) / t_in;

for i = 1:n
    M11 = I1 + I2 + I3 + m2*L1^2 + m3*(L1^2 + L2^2) + 2*m3*L1*L2*cos(theta2(i));
    M12 = I2 + I3 + m3*L2^2 + m3*L1*L2*cos(theta2(i));
    M13 = I3;
    M21 = M12;
    M22 = I2 + I3 + m3*L2^2;
    M23 = I3;
    M31 = M13;
    M32 = M23;
    M33 = I3;
    M = [M11 M12 M13; M21 M22 M23; M31 M32 M33];

    h = -m3*L1*L2*sin(theta2(i));
    C11 = h*theta2_v(i);
    C12 = h*(theta1_v(i) + theta2_v(i));
    C13 = 0;
    C21 = -h*theta1_v(i);
    C22 = 0;
    C23 = 0;
    C31 = 0;
    C32 = 0;
    C33 = 0;
    C = [C11 C12 C13; C21 C22 C23; C31 C32 C33];

    q_v = [theta1_v(i); theta2_v(i); theta3_v(i)];
    q_a = [theta1_a(i); theta2_a(i); theta3_a(i)];

    tao = M*q_a + C*q_v;

    tao1(i) = tao(1);
    tao2(i) = tao(2);
    tao3(i) = tao(3);

    pot1(i) = abs(tao1(i) * theta1_v(i));
    pot2(i) = abs(tao2(i) * theta2_v(i));
```

```
    pot3(i) = abs(tao3(i) * theta3_v(i));
    pot_total(i) = pot1(i) + pot2(i) + pot3(i);
end

figure('Position', [50 50 1400 900]);

subplot(3, 4, 1);
plot(xp, yp, 'b-', 'LineWidth', 2); hold on;
plot(x_in, y_in, 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
plot(x_fin, y_fin, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
xlabel('x [m]'); ylabel('y [m]');
title('Trayectoria en Espacio de Trabajo');
legend('Trayectoria', 'P1', 'P2', 'Location', 'best');
grid on; axis equal;

subplot(3, 4, 2);
plot(t_sim, theta1, 'r-', 'LineWidth', 1.5); hold on;
plot(t_sim, theta2, 'g-', 'LineWidth', 1.5);
plot(t_sim, theta3, 'b-', 'LineWidth', 1.5);
xlabel('t [s]'); ylabel('\theta [rad]');
title('Posiciones Articulares');
legend('\theta_1', '\theta_2', '\theta_3', 'Location', 'best');
grid on;

subplot(3, 4, 3);
plot(t_sim, theta1_v, 'r-', 'LineWidth', 1.5); hold on;
plot(t_sim, theta2_v, 'g-', 'LineWidth', 1.5);
plot(t_sim, theta3_v, 'b-', 'LineWidth', 1.5);
xlabel('t [s]'); ylabel('\omega [rad/s]');
title('Velocidades Articulares');
legend('\omega_1', '\omega_2', '\omega_3', 'Location', 'best');
grid on;

subplot(3, 4, 4);
plot(t_sim, theta1_a, 'r-', 'LineWidth', 1.5); hold on;
plot(t_sim, theta2_a, 'g-', 'LineWidth', 1.5);
plot(t_sim, theta3_a, 'b-', 'LineWidth', 1.5);
xlabel('t [s]'); ylabel('\alpha [rad/s ]');
title('Aceleraciones Articulares');
legend('\alpha_1', '\alpha_2', '\alpha_3', 'Location', 'best');
grid on;

subplot(3, 4, 5);
plot(t_sim, w, 'm-', 'LineWidth', 2);
xlabel('t [s]'); ylabel('w');
title(' ndice  de Manipulabilidad');
yline(0.7, '--r', 'LineWidth', 1);
yline(1.0, '--g', 'LineWidth', 1);
grid on; ylim([0 1.1]);

subplot(3, 4, 6);
plot(t_sim, tao1, 'r-', 'LineWidth', 1.5); hold on;
plot(t_sim, tao2, 'g-', 'LineWidth', 1.5);
plot(t_sim, tao3, 'b-', 'LineWidth', 1.5);
xlabel('t [s]'); ylabel('\tau [N m]');
title('Pares Articulares');
legend('\tau_1', '\tau_2', '\tau_3', 'Location', 'best');
grid on;

subplot(3, 4, 7);
```

```
plot(t_sim, pot1, 'r-', 'LineWidth', 1.5); hold on;
plot(t_sim, pot2, 'g-', 'LineWidth', 1.5);
plot(t_sim, pot3, 'b-', 'LineWidth', 1.5);
xlabel('t [s]'); ylabel('P [W]');
title('Potencia por Motor');
legend('P_1', 'P_2', 'P_3', 'Location', 'best');
grid on;

subplot(3, 4, 8);
plot(t_sim, pot_total, 'k-', 'LineWidth', 2);
xlabel('t [s]'); ylabel('P_{total} [W]');
title('Potencia Total');
grid on;

subplot(3, 4, 9);
bar([max(abs(tao1)), max(abs(tao2)), max(abs(tao3))]);
set(gca, 'XTickLabel', {'Motor 1', 'Motor 2', 'Motor 3'});
ylabel('\tau_{max} [N m]');
title('Par M ximo por Motor');
grid on;

subplot(3, 4, 10);
bar([max(pot1), max(pot2), max(pot3)]);
set(gca, 'XTickLabel', {'Motor 1', 'Motor 2', 'Motor 3'});
ylabel('P_{max} [W]');
title('Potencia M xima por Motor');
grid on;

subplot(3, 4, 11);
plot(t_sim, theta2, 'b-', 'LineWidth', 2);
xlabel('t [s]'); ylabel('\theta_2 [rad]');
title(' ngulo  \theta_2');
grid on;

subplot(3, 4, 12);
pie([sum(pot1), sum(pot2), sum(pot3)]);
title('Distribuci n de Energ a');
legend({'Motor 1', 'Motor 2', 'Motor 3'}, 'Location', 'best');

fprintf('=====================================================\n');
fprintf('EVALUACI N DE CAPACIDADES DEL ROBOT SCARA\n');
fprintf('=====================================================\n\n');

fprintf('1.  NDICE  DE MANIPULABILIDAD:\n');
fprintf('   M nimo: %.4f\n', min(w));
fprintf('   M ximo: %.4f\n', max(w));
fprintf('   Promedio: %.4f\n', mean(w));
porcentaje_w = 100*sum(w>0.7)/n;
fprintf('   Tiempo w > 0.7: %.1f%%\n', porcentaje_w);
fprintf('\n');

fprintf('2. ESPECIFICACIONES DE MOTORES:\n');
fprintf('   Motor 1 - Par m ximo: %.4f N m\n', max(abs(tao1)));
fprintf('             Velocidad m xima: %.4f rad/s\n', max(abs(theta1_v)));
fprintf('             Potencia m xima: %.4f W\n', max(pot1));
fprintf('\n');
fprintf('   Motor 2 - Par m ximo: %.4f N m\n', max(abs(tao2)));
fprintf('             Velocidad m xima: %.4f rad/s\n', max(abs(theta2_v)));
fprintf('             Potencia m xima: %.4f W\n', max(pot2));
fprintf('\n');
```

```
fprintf('   Motor 3 - Par m ximo: %.4f N m\n', max(abs(tao3)));
fprintf('           Velocidad m xima: %.4f rad/s\n', max(abs(theta3_v)));
fprintf('           Potencia m xima: %.4f W\n', max(pot3));
fprintf('\n');

fprintf('3. POTENCIA TOTAL:\n');
fprintf('   M xima: %.4f W\n', max(pot_total));
fprintf('   Promedio: %.4f W\n', mean(pot_total));
energia_total = sum(pot_total)*t_in;
fprintf('   Energ a total: %.4f J\n', energia_total);
fprintf('\n');

fprintf('4. EVALUACI N CUALITATIVA:\n');
if min(w) > 0.3
    fprintf('   Robot mantiene buena manipulabilidad\n');
else
    fprintf('   Robot presenta configuraciones singulares\n');
end

if max(abs(tao1)) < 10 && max(abs(tao2)) < 10 && max(abs(tao3)) < 10
    fprintf('   Pares dentro de rango razonable\n');
else
    fprintf('   Se requieren motores de alto torque\n');
end

if max(pot_total) < 100
    fprintf('   Consumo de potencia aceptable\n');
else
    fprintf('   Alto consumo de potencia\n');
end

fprintf('=====================================================\n');
if porcentaje_w > 80
    fprintf('CONCLUSI N: El robot SCARA S  ES capaz de ejecutar la tarea\n');
else
    fprintf('CONCLUSI N: El robot SCARA NO ES capaz de ejecutar la tarea\n');
end
fprintf('=====================================================\n');

data_table = table(t_sim', theta1', theta2', theta3', ...
                   theta1_v', theta2_v', theta3_v', ...
                   tao1', tao2', tao3', ...
                   w', pot_total', ...
                   'VariableNames', {'t', 'theta1', 'theta2', 'theta3', ...
                                     'omega1', 'omega2', 'omega3', ...
                                     'tau1', 'tau2', 'tau3', ...
                                     'w', 'pot_total'});

writetable(data_table, 'scara_trajectory_data.csv');
saveas(gcf, 'scara_evaluation_results.png');
saveas(gcf, 'scara_evaluation_results.fig');

fprintf('\nDatos guardados en archivos CSV y PNG\n');
```

5.2. Nodo en Python para reproducción de trayectoria en ROS 2

Listing 2: Nodo ROS 2 que lee la trayectoria generada en MATLAB y la ejecuta en el controlador de juntas del SCARA.

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from trajectory_msgs.msg import JointTrajectory, JointTrajectoryPoint
from builtin_interfaces.msg import Duration
import csv

class ScaraTrajectoryPlayer(Node):
    def __init__(self):
        super().__init__("scara_trajectory_player")

        # Publicador para control de articulaciones
        self.publisher_ = self.create_publisher(
            JointTrajectory,
            "/joint_trajectory_controller/joint_trajectory",
            10
        )

        # Nombres de las articulaciones (deben coincidir con el URDF)
        self.joint_names = ["link_1_joint", "link_2_joint", "link_3_joint"]

        # Cargar datos del CSV generado por MATLAB
        self.trajectory_data = []
        with open('scara_trajectory_data.csv', 'r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                self.trajectory_data.append({
                    'time': float(row['t']),
                    'theta1': float(row['theta1']),
                    'theta2': float(row['theta2']),
                    'theta3': float(row['theta3']),
                    'w': float(row['w'])
                })

        self.get_logger().info(
            f'Cargados {len(self.trajectory_data)} puntos de trayectoria'
        )

        # Timer para enviar puntos (100 ms, igual que en MATLAB)
        self.timer_period = 0.1
        self.current_index = 0
        self.timer = self.create_timer(
            self.timer_period, self.publish_joint_states
        )

    def publish_joint_states(self):
        if self.current_index < len(self.trajectory_data):
            data = self.trajectory_data[self.current_index]

            # Crear mensaje de trayectoria
            msg = JointTrajectory()
            msg.joint_names = self.joint_names

            point = JointTrajectoryPoint()
            point.positions = [
                data['theta1'],
                data['theta2'],
                data['theta3']
            ]
            point.time_from_start = Duration(
```

```
                sec=int(data['time']),
                nanosec=0
            )

            msg.points.append(point)
            self.publisher_.publish(msg)

            # Mensaje de depuraci n cada 10 muestras
            if self.current_index % 10 == 0:
                self.get_logger().info(
                    f"Punto {self.current_index}: "
                    f" 1 ={data['theta1']:.3f}, "
                    f" 2 ={data['theta2']:.3f}, "
                    f"w={data['w']:.3f}"
                )

            self.current_index += 1
        else:
            self.get_logger().info('Trayectoria completada')
            self.timer.cancel()

def main(args=None):
    rclpy.init(args=args)
    node = ScaraTrajectoryPlayer()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

5.3. Modelo URDF del robot SCARA

Listing 3: Descripción URDF del robot SCARA utilizado en la simulación.

```
<?xml version="1.0"?>
<robot name="scara_robot">

  <!-- MATERIALES (colores) -->
  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0"/>
  </material>
  <material name="red">
    <color rgba="0.8 0.0 0.0 1.0"/>
  </material>
  <material name="green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>
  <material name="yellow">
    <color rgba="0.8 0.8 0.0 1.0"/>
  </material>
  <material name="black">
    <color rgba="0.1 0.1 0.1 1.0"/>
  </material>

  <!-- BASE FIJA -->
  <link name="base_link">
    <visual>
      <geometry>
        <box size="0.3 0.3 0.05"/>
```

```
      </geometry>
      <material name="black"/>
    </visual>
  </link>

  <!-- ESLAB N 1 -->
  <link name="link_1">
    <visual>
      <geometry>
        <box size="0.5 0.05 0.05"/>
      </geometry>
      <origin xyz="0.25 0 0" rpy="0 0 0"/>
      <material name="red"/>
    </visual>
  </link>

  <!-- ARTICULACI N 1 -->
  <joint name="link_1_joint" type="revolute">
    <parent link="base_link"/>
    <child link="link_1"/>
    <origin xyz="0 0 0.025" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-3.14159" upper="3.14159" effort="10.0" velocity="2.0"/>
  </joint>

  <!-- ESLAB N 2 -->
  <link name="link_2">
    <visual>
      <geometry>
        <box size="0.5 0.05 0.05"/>
      </geometry>
      <origin xyz="0.25 0 0" rpy="0 0 0"/>
      <material name="green"/>
    </visual>
  </link>

  <!-- ARTICULACI N 2 -->
  <joint name="link_2_joint" type="revolute">
    <parent link="link_1"/>
    <child link="link_2"/>
    <origin xyz="0.5 0 0.001" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-3.14159" upper="3.14159" effort="10.0" velocity="2.0"/>
  </joint>

  <!-- ESLAB N 3 -->
  <link name="link_3">
    <visual>
      <geometry>
        <box size="0.3 0.05 0.05"/>
      </geometry>
      <origin xyz="0.15 0 0" rpy="0 0 0"/>
      <material name="yellow"/>
    </visual>
  </link>

  <!-- ARTICULACI N 3 -->
  <joint name="link_3_joint" type="revolute">
    <parent link="link_2"/>
    <child link="link_3"/>
```

```
    <origin xyz="0.5 0 0.002" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-3.14159" upper="3.14159" effort="5.0" velocity="2.0"/>
  </joint>

  <!-- EFECTOR FINAL -->
  <link name="end_effector">
    <visual>
      <geometry>
        <sphere radius="0.02"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

  <joint name="end_effector_joint" type="fixed">
    <parent link="link_3"/>
    <child link="end_effector"/>
    <origin xyz="0.3 0 0" rpy="0 0 0"/>
  </joint>

</robot>
```

5.4. Archivo `launch` para visualización en ROS 2

Listing 4: Launch file para publicar el URDF del SCARA y abrir RViz2.

```python
from launch import LaunchDescription
from launch_ros.actions import Node
import os

def generate_launch_description():

    urdf_path = os.path.join(
        os.path.dirname(__file__),
        '..', 'urdf', 'scara.urdf'
    )

    with open(urdf_path, 'r') as f:
        robot_desc = f.read()

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            parameters=[{'robot_description': robot_desc}]
        ),

        Node(
            package='scara_robot',
            executable='move_scara.py',
            name='scara_mover',
            output='screen'
        ),

        Node(
            package='rviz2',
            executable='rviz2',
```

```
            name='rviz2',
            arguments=[
                '-d',
                os.path.join(
                    os.path.dirname(__file__),
                    '..', 'rviz', 'scara.rviz'
                )
            ]
        ),

        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
            name='joint_state_publisher_gui'
        )
    ])
```

5.5. Archivo de configuración de RViz2

Listing 5: Configuración básica de RViz2 para visualizar el SCARA.

```
Panels:
  - Class: rviz_common/Displays
    Name: Displays

Visualization Manager:
  Displays:
    - Class: rviz_default_plugins/Grid
      Name: Grid
    - Class: rviz_default_plugins/RobotModel
      Name: RobotModel

  Global Options:
    Fixed Frame: base_link

  Views:
    Current:
      Class: rviz_default_plugins/Orbit
      Distance: 2.0
```

6. Resultados

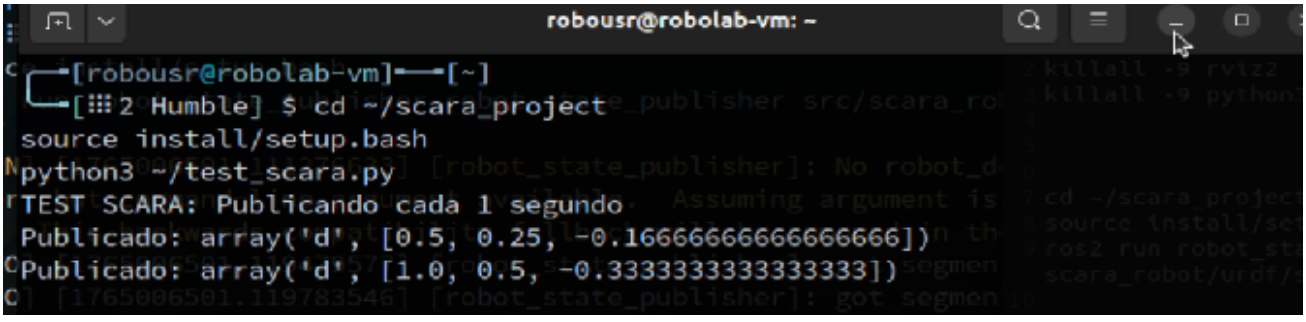

Figura 1: Nodo publicador

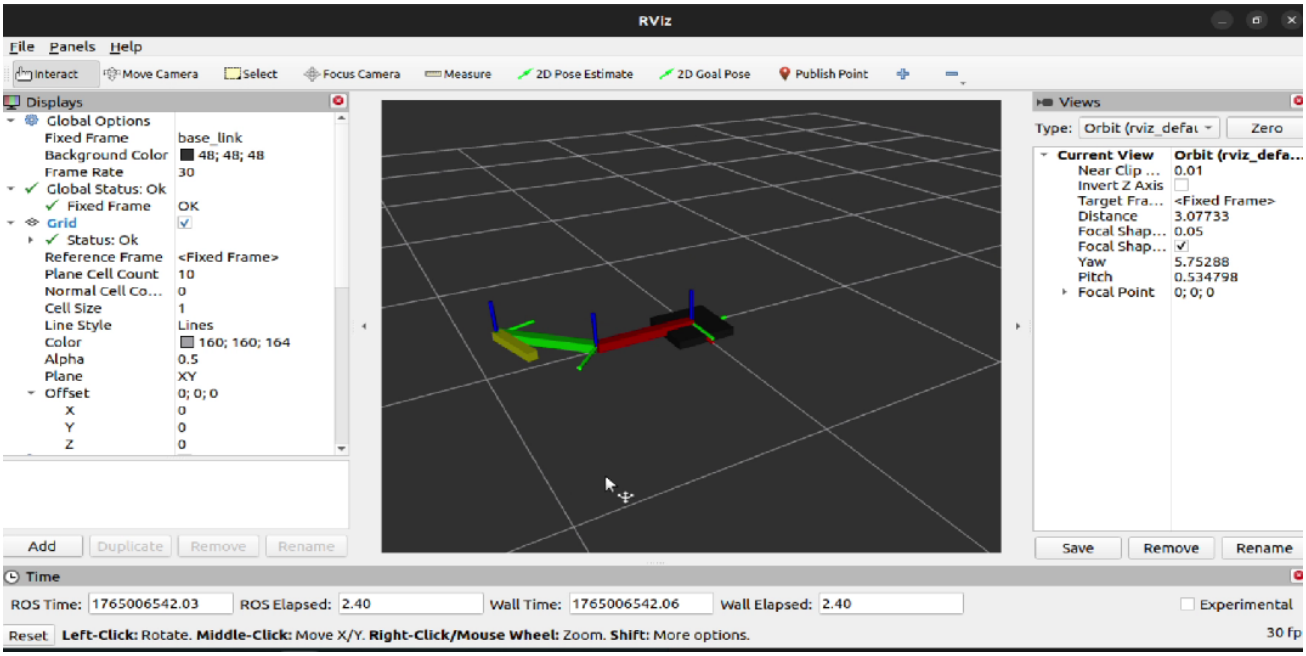

Figura 2: Rviz

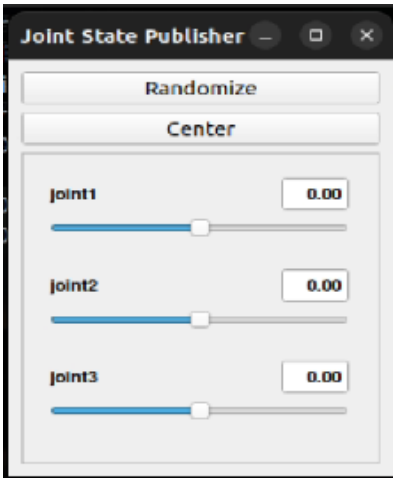

Figura 3: GUI

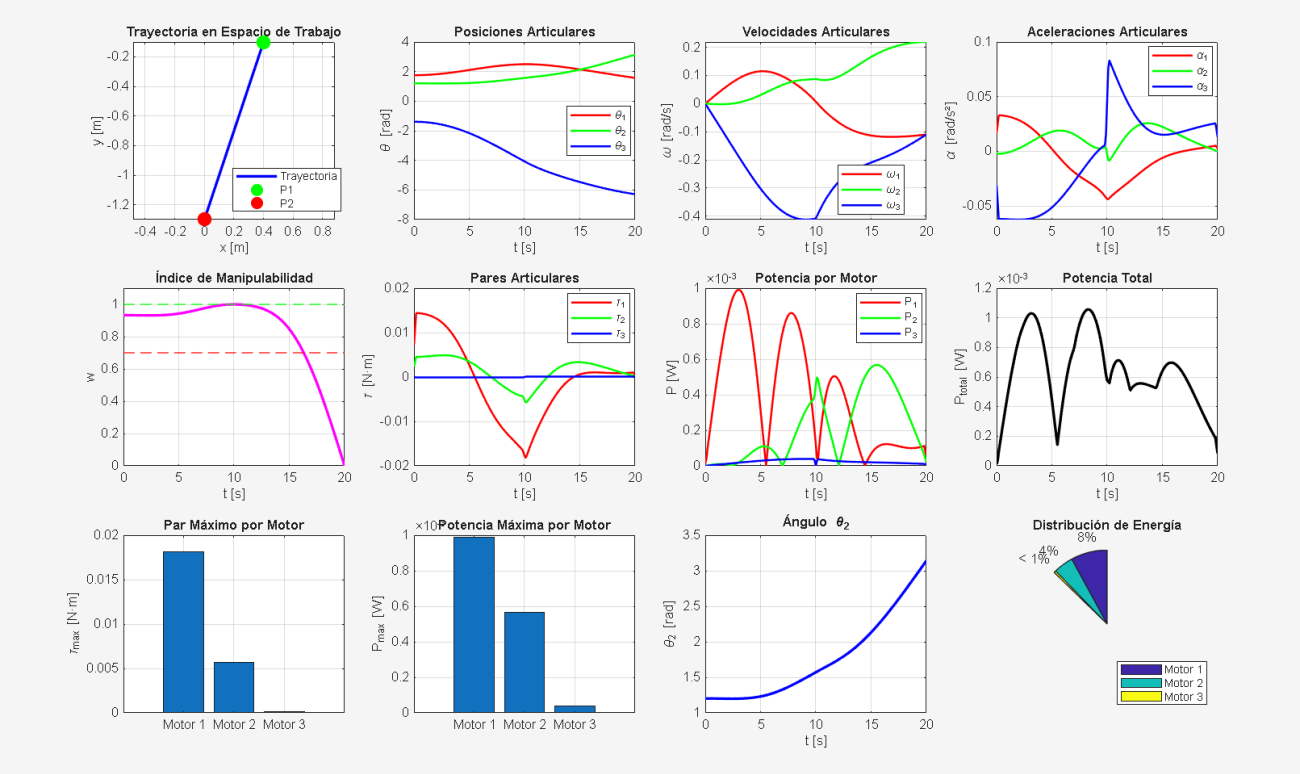

Figura 4: Graficas Matlab

7. Análisis de resultados

El análisis de la trayectoria Bang–Bang muestra que el robot SCARA logra desplazar el efector final desde P_1 hasta P_2 siguiendo el recorrido planeado en el plano XY sin desviaciones apreciables, lo que indica que la cinemática inversa y la interpolación de la trayectoria están bien planteadas. El índice de manipulabilidad w se mantiene en la mayor parte del trayecto dentro de un rango aceptable, evitando configuraciones singularmente extendidas o plegadas; esto sugiere que la trayectoria propuesta aprovecha regiones del espacio de trabajo donde el robot conserva buena capacidad de generación de velocidades.

En términos dinámicos, los perfiles de velocidad y aceleración articulares presentan el comportamiento esperado para una ley Bang–Bang con mezcla parabólica, con cambios de signo en el punto medio y sin picos numéricos anómalos. Los pares calculados mediante el modelo de Euler–Lagrange se mantienen por debajo de valores que podrían considerarse críticos para motores industriales de tamaño medio, y la potencia total presenta un máximo moderado, lo que apunta a que la tarea de seguimiento de trayectoria es factible tanto desde el punto de vista cinemático como energético. La simulación en ROS 2 con el URDF confirma visualmente estos resultados, ya que el movimiento reproduce la trayectoria planeada sin comportamientos erráticos, cerrando la coherencia entre el modelo matemático y la animación 3D.

8. Conclusiones individuales

Raya Cruz José Luis

Desde mi perspectiva, este proyecto permitió entender de forma integrada cómo la elección de una trayectoria afecta directamente las exigencias dinámicas sobre los motores del robot SCARA. Implementar el modelo de Euler–Lagrange y comparar pares y potencias con la animación en ROS 2 me ayudó a relacionar las ecuaciones con un comportamiento físico concreto, reforzando mi criterio para seleccionar actuadores adecuados y para evitar configuraciones poco favorables en aplicaciones reales.

Romero Bernal Rocio Fabiola

Trabajar en la planeación de la trayectoria Bang–Bang y en el cálculo del índice de manipulabilidad me permitió apreciar la importancia de combinar cinemática y análisis numérico para garantizar movimientos eficientes y seguros. Además, la integración entre MATLAB, el nodo de Python y el modelo URDF me dio una visión más clara de cómo se conectan las etapas de modelado, simulación y validación en un flujo de trabajo profesional de robótica.

Reyes González Diego Iván

Mi principal aprendizaje fue la relevancia de la integración de software en proyectos mecatrónicos: pasar de un archivo CSV generado en MATLAB a un controlador de trayectorias en ROS 2 demostró la necesidad de escribir código limpio y bien documentado para que todas las herramientas se entiendan entre sí. Ver en RViz cómo el SCARA seguía la trayectoria calculada confirmó que los modelos teóricos eran coherentes y me motivó a seguir profundizando en el desarrollo de nodos, archivos `launch` y descripciones URDF para aplicaciones de automatización industrial.

Referencias

[1] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. MIT Press, 1990.

[2] B. Siciliano, L. Sciavicco, L. Villani y G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[3] M. W. Spong, S. Hutchinson y M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2006.

[4] R. M. Murray, Z. Li y S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[5] M. Quigley et al., "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, 2009.

[6] W. Meeussen et al., "URDF: Unified Robot Description Format," *ROS.org Documentation*, consultado 2025.

[7] S. A. Alshahrani et al., "Optimum trajectory function for minimum energy requirements of a spherical robot," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, 2002.