Proyecto del 2º parcial

Manipuladores Robóticos

Universidad Autonoma de Aguascalientes
Ingenieria Robótica

Evelyn Lázaro Guerra
Universidad Autonoma de Aguascalientes
Ingenieria Robótica

Universidad Autonoma de Aguascalientes
Ingenieria Robótica

Universidad Autonoma de Aguascalientes
Ingenieria Robótica

***Resumen*— En este trabajo se presenta el diseño y construcción de un robot planar con dos grados de libertad (RR) que utiliza un plumón acoplado como efector final. El sistema permite controlar el plumón usando cinemática directa o inversa en dos posiciones (arriba y abajo), con el objetivo de dejar o no una marca sobre la superficie de trabajo. (*Abstract*)**

Keywords—Robot planar, Robot RR, grados de libertad, plumón, servomecanismo, control de posición, diseño de robots.

I. INTRODUCCION.

El desarrollo de robots con dos grados de libertad (RR) tiene diversas aplicaciones, entre ellas la realización de trazados en superficies planas. El presente trabajo aborda el diseño y construcción de un robot planar RR con un plumón acoplado como efector final, capaz de desplazarse en dos direcciones en un plano y de controlar la posición del plumón para dejar o no una marca. Este diseño utiliza un mecanismo simple para controlar la posición del plumón, facilitando aplicaciones en el ámbito de la robótica educativa y artística.

II. DESCRIPCIÓN DEL DISEÑO

A. Cinemática del Robot RR

El robot consta de dos eslabones rotacionales (R) acoplados en serie, proporcionando dos grados de libertad que permiten controlar la posición del efector final en un plano. La cinemática directa e inversa del sistema fue calculada para definir las posiciones y orientaciones posibles en el espacio de trabajo.

B. Seleccion del Efecto Final y Control de Dos Posiciones.

El efector final del robot es un plumón. Se diseñó un mecanismo de soporte que permite controlar el plumón en dos posiciones: arriba (sin contacto con la superficie) y abajo (en contacto con la superficie), activado mediante un motor a pasos. Este mecanismo permite realizar marcas en la superficie de trabajo de forma controlada y precisa.

C. Selección de Motores y Drivers

Para el control de los movimientos del robot, se consideraron diferentes tipos de motores: motores de CD con encoder de cuadratura, motores a pasos y servomotores. Tras evaluar cada opción en términos de precisión y facilidad de implementación, se seleccionaron **motores a pasos**. Los drives utilizados fueron el drv556 y dm6600 debido a su compatibilidad, precisión y facilidad de integración.

D. Diseño del sistema robótico

Se eligió el diseño que se muestra en la figura 1 por su estructura compacta, facilidad de montaje y desmontaje, rigidez, cableado ordenado y estética.

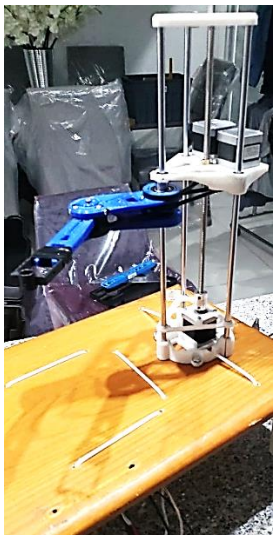

Figura 1. Diseño final

La estructura está impresa en 3D, acompañada de 3 motores NEMA 17, engranajes, bandas para engranajes, la punta del efector final tiene acoplado una herramienta para colocar el

plumón que realizará las marcas en el espacio de trabajo. El primer eslabón (L1) tiene una longitud de 90mm y el segundo (L2), que va desde el final del primero a la punta del plumón mide 160mm. Además, el robot está sujeto a una mesa pequeña hecha con una tabla de madera y debajo de ella se colocaron los 3 drivers (1 por motor), la fuente de alimentación y el microcontrolador ESP32, con el que le mandan las ordenes al robot.

E. Cinemática directa..

La cinemática directa se calculó usando el método de Denavit Hartenberg, para lo cual, primero se obtuvo la Tabla 1.

i	θ_i	di	ai	αi
1	θ_1^*	0	L_1	0
2	θ_2^*	0	L_2	0

Tabla 1. Tabla de los parámetros de eslabón θi,di,ai y αi

La multiplicación de matrices fue realizada en MATLAB se llegó a que la matriz de transformación homogénea $T_{0/2}$ es:

$$\begin{pmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & L_2\cos(\theta_1+\theta_2)+L_1\cos(\theta_1) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & L_2\sin(\theta_1+\theta_2)+L_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Como es sabido, la cinemática directa para un robot SCARA se define por:

1. $X = L1 \cdot cos(\theta 1)+L2 \cdot cos(\theta 1+\theta 2)$ – Ec.1
2. $y = L1 \cdot sin(\theta 1)+L2 \cdot sin(\theta 1+\theta 2)$ – Ec. 2

donde:

- $L1$ y $L2$ son las longitudes de los dos enlaces.
- $\theta 1$ y $\theta 2$ son los ángulos de cada articulación.

Con lo anterior como base, se realizó un código en el que ingresando los ángulos theta1 y theta2 se permita conocer las posiciones x y y del robot.

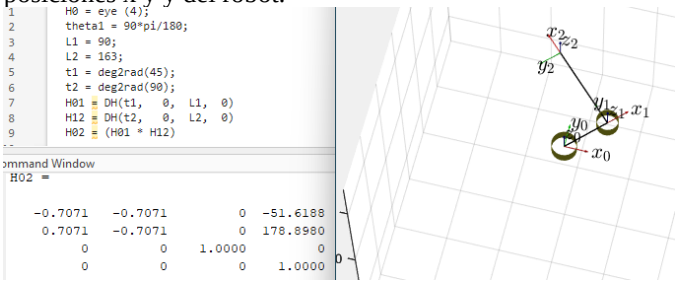

Figura 2. Simulación en MATLAB de la cinemática directa

F. Cinemática inversa.

Para un róbot SCARA dados los puntos deseados (x,y), los ángulos θ1 y θ2 se calculan como:

1. D= $\frac{x^2+y^2-L_1^2-L_2^2}{2 \cdot L_1 \cdot L_2}$
2. θ2=atan2($\pm\sqrt{1-D^2}$,D)
3. θ1=atan2(y,x)−atan2(L2·sin(θ2),L1+L2·cos(θ2))

Para las configuraciones codo arriba y codo abajo simplemente se cambia el signo que multiplica a la raíz, siendo positivo para codo abajo y negativo para codo arriba.

Para compobar los resultados fisicos también se ingresaron estas ecuaciones en MATLAB

```
function [q1, q2] = fcn_inversa(x, y, L1, L2, codo)

% codo = 1; % 0 es codo arriba
D= (x^2+y^2-L1^2-L2^2)/(2*L1*L2);
if codo==0 % codo abajo
q2 = atan2 (+sqrt (1-D^2), D);
else % codo arriba
q2 = atan2 (-sqrt (1-D^2), D);
end
q1 = atan2 (y, x) -atan2 (L2*sin (q2), L1+L2*cos (q2));
```

Figura 3. Código de MATLAB para la cinemática inversa

En el anexo E se muestra el código explicado que se usó en Matlab para comprobar la cinemática inversa con el origen del espacio de trabajo y en el Anexo G este código integrado a Python.

Finalmente, las ecuaciones de ambos tipos de cinemáticas fueron ingresadas en el código Python junto con la interfaz. Cabe aclarar que todos los cálculos para ambos tipos de cinemáticas se realizaron en el lenguaje Python y no en Arduino por practicidad, en el Arduino solo se controla el movimiento de los motores dependiendo de los datos que reciba en la interfaz.

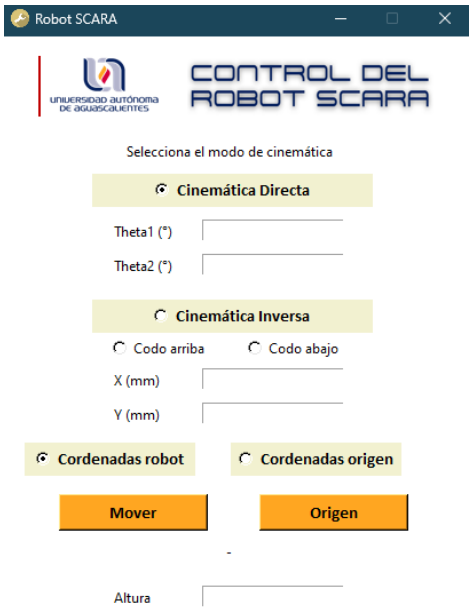

Figura 4. Interfaz

G. *Algunos errores comunes*

- El error que se tiende mayoritariamente a tomar en cuenta es la mecanica, es parte escencial a la hora de estar diseñando el robot y a la hora de ensamblarlo llega a presentar fallas y eso hace que pierda exactitud que se pudo haber tomado en cuenta anteriormente.

III. Desarrollo y construccion.

A. *Recoleccion de informacion.*

En base a que fue un proyecto donde Nuestro principal objetivo sería el conocimiento, empezamos por ir investigando sobre como armarlo, viendo diseños y eso.

B. *Montaje Mecanico.*

Fuimos armando el brazo robótico en base a todo lo que investigamos y fuimos dándonos cuenta de lo que implicaria armar el brazo y más.

También en cada detalle fueron surgiendo los errores de la mecanica los cuales se corrigieron en su mayoria.

Y los resultados nos fueron llevando a cada vez ir corrigiendo los errores y llegar a los resultados finales.

C. *Resultados*

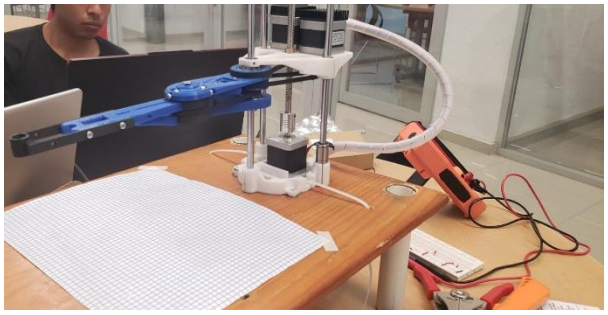

Figura 5. Resultado final

Logramos terminar el robot de dos grados de libertad con la única limitante de que el alance del los eslabones no cubren el área de una hoja tamaño carta. Por demás todo funcionó correctamente y de acuerdo a lo esperado, tanto la cinemática directa como la inversa.

CONCLUSIONES.

El diseño y construcción del robot planar RR con dos grados de libertad y un plumón como efector final demuestra la viabilidad de un sistema sencillo y económico para aplicaciones de trazado en superficies planas. Los resultados experimentales mostraron precisión adecuada para tareas de dibujo y marcado, con una estructura mecánica estable y un sistema de control eficiente.

AGRADECIMIENTOS.

Agradecemos profundamente el apoyo y la motivación de nuestros amigos, quienes estuvieron presentes en cada etapa del proyecto y nos impulsaron a continuar en los momentos difíciles. Queremos extender un agradecimiento especial a César Alejandro Chávez Olivares, cuya experiencia y consejos técnicos fueron invaluables para el avance y éxito de esta investigación.

ANEXOS.

Anexo A.

```
//Cinemática directa
H0 = eye (4);
theta1 = 90*pi/180;
L1 = 90;
L2 = 163;
t1 = deg2rad(45);
t2 = deg2rad(90);
H01 = DH(t1,   0,  L1,  0)
H12 = DH(t2,   0,  L2,  0)
H02 = (H01 * H12)
```

```
% ------------------------- D I B U J A R ----------------------
-----

figure(2)
clf
hold on

[origen0] = Sistema3D (H0, 45, 'x_0','y_0', 'z_0',16);
[origen1] = Sistema3D (H01, 45, 'x_1','y_1', 'z_1',16);
[origen2] = Sistema3D (H02, 45, 'x_2','y_2', 'z_2',16);

eslabon(origen0, origen1, 'k', 1)
eslabon(origen1, origen2, 'k', 1)

cilindro(H0, -23, 23, -12.5) % cilindro(K,alto,radio,offset
negativo con respecto al origen)
cilindro(H01, -23, 20, -12.5)

grid on
axis equal
axis([-1 1 -1 1 -1 1]*300) % Establece los límites de los
ejes

%view ([93,2.7])
cameratoolbar
camlight('left')
camproj('perspective')
grid on
title('Robot 2 GDL')
xlabel('X')
ylabel('Y')
zlabel('Z')
```

Anexo B.

// **Cinemática inversa**

```
L1=90;
L2=163;
ql = deg2rad(0);
q2 = deg2rad(30);
TK = fcn_dibujar_scara(ql, q2, L1, L2, 1)
X=TK(1,4);
Y=TK(2, 4);

[qli, q2i] = fcn_inversa(X, Y, L1, L2, 1); % 0 es codo
abajo, 1 codo arriba
TKi = fcn_dibujar_scara(qli, q2i, L1, L2,1)
Xi=TKi(1, 4);
Yi=TKi(2,4);
```

Anexo C.

//fcn_dibujar_scara.m

```
function [H02] = fcn_dibujar_scara(ql, q2, L1, L2, n)
H0 = eye (4);
H01 = rotz(ql) * trans(L1,0,0);
H12 = rotz(q2) * trans(L2,0,0);
H02 = H01 * H12;
figure (n)
clf
hold on

[origen0] = Sistema3D(H0, 15, 'x_0','y_0', 'z_0',16);
[origenl] = Sistema3D(H01, 15, 'x_1', 'y_1', 'z_1',16);
[origen2] = Sistema3D(H02, 15, 'x_2', 'y_2', 'z_2', 16);

eslabon(origen0, origenl, 'k', 1)
eslabon(origenl, origen2, 'k', 1)

cilindro(H0, -23, 23, -12.5) % cilindro(K,alto,radio,offset
negativo con respecto al origen)
cilindro(H01, -23, 20, -12.5)

grid on
axis equal
axis([-1 1 -1 1 -1 1]*300) % Establece los límites de los ejes

%view ([93,2.7])
cameratoolbar
camlight('left')
camproj('perspective')
grid on
title('Robot 2 GDL')
xlabel('X')
ylabel('Y')
zlabel('Z')
```

Anexo D.

// **fcn_inversa.m**

```
function [ql, q2] = fcn_inversa(x, y, L1, L2, codo)

% x = 50;
% y = 100;
% codo = 1; % 0 es codo arriba
D= (x^2+y^2-L1^2-L2^2)/(2*L1*L2);
if codo==0 % codo abajo
q2 = atan2 (+sqrt (1-D^2), D);
else % codo arriba
q2 = atan2 (-sqrt (1-D^2), D);
end
ql = atan2 (y, x) -atan2 (L2*sin (q2), L1+L2*cos (q2));
% t1 = rad2deg(ql)
% t2 = rad2deg(q2)
```

Anexo E.

```
% Longitudes de los brazos
L1 = 85;
L2 = 160;
```

```
% Definir el origen del espacio de trabajo en las coordenadas del robot
x_offset = 10;
y_offset = 10;

% Posición deseada en el sistema de coordenadas del espacio de trabajo
x_workspace = 7;
y_workspace = 3;

% Convertir a las coordenadas del sistema del robot
x = x_workspace + x_offset;
y = y_workspace + y_offset;

% Verificar si el punto está dentro del alcance del robot
if sqrt(x^2 + y^2) > (L1 + L2)
   error('El punto deseado está fuera del alcance del robot.');
end

% Configuración Codo Arriba
cos_theta2 = (x^2 + y^2 - L1^2 - L2^2) / (2 * L1 * L2);

% Asegurarse de que cos_theta2 esté en el rango [-1, 1]
cos_theta2 = max(min(cos_theta2, 1), -1);

theta2_up = acos(cos_theta2); % En radianes
theta1_up = atan2(y, x) - atan2(L2 * sin(theta2_up), L1 + L2 * cos(theta2_up));

% Configuración Codo Abajo
theta2_down = -acos(cos_theta2); % En radianes
theta1_down = atan2(y, x) - atan2(L2 * sin(theta2_down), L1 + L2 * cos(theta2_down));

% Convertir los ángulos a grados (opcional)
theta1_up_deg = rad2deg(theta1_up);
theta2_up_deg = rad2deg(theta2_up);
theta1_down_deg = rad2deg(theta1_down);
theta2_down_deg = rad2deg(theta2_down);

% Mostrar los resultados
fprintf('Codo Arriba:\n');
fprintf('Theta1: %.2f\n', theta1_up_deg);
fprintf('Theta2: %.2f\n\n', theta2_up_deg);

fprintf('Codo Abajo:\n');
fprintf('Theta1: %.2f\n', theta1_down_deg);
fprintf('Theta2: %.2f\n\n', theta2_down_deg);
```

Anexo F.

// **CODIGO DE ARDUINO**

```
#include <math.h>

// driver 2
int PUL2 = 34;
int DIR2 = 35;
int ENA2 = 32;
// driver 1
int PUL1 = 33;
int DIR1 = 25;
int ENA1 = 26;
// driver 3
int ENA3 = 27;
int DIR3 = 14;
int PUL3 = 12;

double pasos1, pasos2;
int angulo1, angulo2, velocidad = 400, angulo1R = 0, angulo2R = 0, posicion1 = 0, posicion2 = 0, t1 = 0, t2 = 0;

void setup() {
   Serial.begin(115200);
   pinMode(PUL1, OUTPUT);
   pinMode(DIR1, OUTPUT);
   pinMode(ENA1, OUTPUT);
   digitalWrite(ENA1, LOW); // Activa el driver
   pinMode(PUL2, OUTPUT);
   pinMode(DIR2, OUTPUT);
   pinMode(ENA2, OUTPUT);
   digitalWrite(ENA2, LOW); // Activa el driver
   Serial.println("Ingrese los ángulos theta1 y theta2 en grados (separados por espacio):");
}

void loop() {
   if (Serial.available() > 0) {
      Serial.println("Ingrese los ángulos theta1 y theta2 en grados:");

      // Leer la línea completa de entrada
      String input = Serial.readStringUntil('\n');
      int separatorIndex = input.indexOf(' '); // Encuentra el espacio

      if (separatorIndex > 0) { // Asegúrate de que haya un espacio
         angulo1R = input.substring(0, separatorIndex).toInt();
         angulo2R = input.substring(separatorIndex + 1).toInt();
         }

         t1 = angulo1R - posicion1;
         t2 = angulo2R - posicion2;

         angulo1 = abs(t1);
         angulo2 = abs(t2);
         // Calcular pasos para cada motor
         pasos1 = ((angulo1 * 6400) / 360) * 2.4;
         pasos2 = ((angulo2 * 6400) / 360) * 3.92;

         // Mover los motores en paralelo
         unsigned long maxPasos = pasos1 + pasos2;
```

```
        for (unsigned long i = 0; i < maxPasos; i++) {
          if (i < pasos1) { // Ejecuta un paso para el motor 1 si no ha terminado

            digitalWrite(DIR1, t1 >= 0 ? LOW : HIGH);
            digitalWrite(DIR2, t1 >= 0 ? LOW : HIGH);

              digitalWrite(PUL1, HIGH);
              delayMicroseconds(velocidad);
              digitalWrite(PUL1, LOW);
              delayMicroseconds(velocidad);

              // Ejecuta un paso para el motor 2 si no ha terminado
              digitalWrite(PUL2, HIGH);
              delayMicroseconds(velocidad);
              digitalWrite(PUL2, LOW);
              delayMicroseconds(velocidad);
            }
            if (i > pasos1){
            digitalWrite(DIR2, t2 >= 0 ? LOW : HIGH); // avanza el 2
              digitalWrite(PUL2, HIGH);
              delayMicroseconds(velocidad);
              digitalWrite(PUL2, LOW);
              delayMicroseconds(velocidad);
            }
          }
          posicion1 = angulo1R;
          posicion2 = angulo2R;
    }
}
```

Anexo G.

// **CODIGO DE PYHTON**

```
import math
import serial
import time
import tkinter as tk
from tkinter import *
from tkinter import messagebox
from PIL import Image, ImageTk

# Configuración del puerto serial y la velocidad de baudios
arduino = serial.Serial('COM5', 115200)
time.sleep(2)  # Espera para que el Arduino inicie la comunicación serial

L1 = 85.0  # Longitud del primer eslabón
L2 = 160.0  # Longitud del segundo eslabón
# Variables para almacenar el origen de la herramienta
x_origen = 0.0
y_origen = 0.0
theta1_origen = 0.0
theta2_origen = 0.0

# ---------------------------------------------- F U N C I O N E S ----------------------------------------------

def enviar_angulo(angulo1, angulo2, altura):
    # Convertir ángulos a cadena y enviarlos al Arduino
    comando = f"{angulo1} {angulo2} {altura}\n"  # Formato: "angulo1 angulo2 altura\n"
    arduino.write(comando.encode())
    print(f"Ángulos enviados: {angulo1}°, {angulo2}°, altura: {altura}")
    time.sleep(0.1)  # Pequeña pausa para evitar desbordamiento en Arduino

def asignar_origen():
    modo = modo_cinematica.get()
    global x_origen, y_origen, theta1_origen, theta2_origen
    if modo == "directa":
        theta1_origen = float(entry_theta1.get())
        theta2_origen = float(entry_theta2.get())
                                x_origen, y_origen, altura = cinemática_directa(theta1_origen, theta2_origen)

    elif modo == "inversa":
        x_origen = float(entry_x.get())
        y_origen = float(entry_y.get())
                                        theta1_origen,theta2_origen, altura=cinemática_inversa(x_origen, y_origen)

# Cinemática directa
def cinemática_directa(theta1, theta2):
    a = alt.get()
    theta1_rad = math.radians(theta1)
    theta2_rad = math.radians(theta2)
    x = L1 * math.cos(theta1_rad) + L2 * math.cos(theta1_rad + theta2_rad)
    y = L1 * math.sin(theta1_rad) + L2 * math.sin(theta1_rad + theta2_rad)
    return x, y, a

# Cinemática inversa
def cinemática_inversa(x, y):
    a = alt.get()
    solucion=codo.get()
     # Ajustar las coordenadas (x, y) para tener en cuenta el origen de herramienta
    cos_theta2 = (x**2 + y**2 - L1**2 - L2**2) / (2 * L1 * L2)
    if solucion =="arriba":
        sin_theta2 = math.sqrt(1 - cos_theta2**2)
    elif solucion =="abajo":
        sin_theta2 = -math.sqrt(1 - cos_theta2**2)
    else:
         messagebox.showerror("Error", "Selecciona el tipo de codo.")

    theta2 = math.atan2(sin_theta2, cos_theta2)

    k1 = L1 + L2 * cos_theta2
```

```
    k2 = L2 * sin_theta2
    theta1 = math.atan2(y, x) - math.atan2(k2, k1)

    theta1_deg = math.degrees(theta1)
    theta2_deg = math.degrees(theta2)

    return theta1_deg, theta2_deg, a

# Seleccionar y ejecutar la cinemática
def mover_robot():
    modo = modo_cinematica.get()
    coor=Coordenadas.get()

    if modo == "directa":
        theta1 = entry_theta1.get()
        theta2 = entry_theta2.get()

        if theta1 and theta2:
            theta1 = float(theta1)
            theta2 = float(theta2)
            if coor=="robot":
                x, y,a = cinemática_directa(theta1, theta2)
                enviar_angulo(theta1, theta2, a)
            elif coor=="origen":
                    x, y,a = cinemática_directa(theta1+theta1_origen,
theta2+theta2_origen)
                                enviar_angulo(theta1+theta1_origen,
theta2+theta2_origen, a)
             lbl_resultado.config(text=f"Posición: X = {x:.2f} mm,
Y = {y:.2f} mm")
        else:
            messagebox.showerror("Error", "Debes ingresar ambos
ángulos para la cinemática directa.")

    elif modo == "inversa":
        x = entry_x.get()
        y = entry_y.get()

        if x and y:
            x = float(x)
            y = float(y)
            if coor=="robot":
                theta1, theta2, altura = cinemática_inversa(x, y)
                        lbl_resultado.config(text=f"Ángulos: θ1 =
{theta1:.2f}°, θ2 = {theta2:.2f}°")
                enviar_angulo(theta1, theta2, altura)
            elif coor=="origen":
                x+=x_origen
                y+=y_origen
                theta1, theta2, altura= cinemática_inversa(x, y)
                        lbl_resultado.config(text=f"Ángulos: θ1 =
{theta1:.2f}°, θ2 = {theta2:.2f}°")
                enviar_angulo(theta1, theta2, altura)
        else:
            messagebox.showerror("Error", "Debes ingresar ambas
coordenadas para la cinemática inversa.")
```

```
# ---------------------------------------- I N T E R F A Z -------
---------------------------------------

app = tk.Tk()
app.title("Robot SCARA")
app.geometry("410x550")
app.resizable(False,False)
app.iconbitmap("../pytonn/imagenes/icono.ico")
app.config(bg = "white") # color de fondo

# imagenes

logo_img = Image.open("../pytonn/imagenes/uaa.png")
logo_img = logo_img.resize((100, 50)) # cambiar el tamaño de
la imagen
logo_img = ImageTk.PhotoImage(logo_img)
logo_label = Label(app, image=logo_img, bg="white")
logo_label.image = logo_img  # Mantener la referencia
logo_label.place(x=35, y=15)

linea = Image.open("../pytonn/imagenes/linea.png")
linea = linea.resize((2, 55)) # cambiar el tamaño de la imagen
linea = ImageTk.PhotoImage(linea)
linea_label = Label(app, image=linea, bg="white")
linea_label.image = linea  # Mantener la referencia
linea_label.place(x=30, y=15)

astron = Image.open("../pytonn/imagenes/astron.png")
astron = astron.resize((215, 50)) # cambiar el tamaño de la
imagen
astron = ImageTk.PhotoImage(astron)
astron_label = Label(app, image=astron, bg="white")
astron_label.image = astron  # Mantener la referencia
astron_label.place(x=160, y=17)

# Modo de cinemática (Directa o Inversa)
modo_cinematica = tk.StringVar(value="directa")
Coordenadas = tk.StringVar(value="robot")
codo = tk.StringVar(value=" ")

#tk.Label(app, text="Control de Robot SCARA", width=56,
justify="center", bg="white").place(x=0, y=90)
tk.Label(app, text="Selecciona el modo de cinemática",
width=56, justify="center", bg="white").place(x=0, y=90)
tk.Radiobutton(app, text="Cinemática Directa",
variable=modo_cinematica, value="directa", width=27,
justify="center", bg="#f2f1d0", font=("Calibri", 11,
"bold")).place(x=79, y=120)

# Entrada para ángulos de cinemática directa
tk.Label(app, text="Theta1 (°)", bg="white").place(x=95,
y=160)
entry_theta1 = tk.Entry(app)
entry_theta1.place(x=175, y=160)
tk.Label(app, text="Theta2 (°)", bg="white").place(x=95,
y=190)
entry_theta2 = tk.Entry(app)
```

```
entry_theta2.place(x=175, y=190)

tk.Radiobutton(app, text="Cinemática Inversa", variable=modo_cinematica, value="inversa", width=27, justify="center", bg="#f2f1d0", font=("Calibri", 11, "bold")).place(x=79, y=230)
tk.Radiobutton(app, text="Codo arriba", variable=codo, value="arriba", bg="white").place(x=92, y=260)
tk.Radiobutton(app, text="Codo abajo", variable=codo, value="abajo", bg="white").place(x=210, y=260)

# Entrada para coordenadas de cinemática inversa
tk.Label(app, text="X (mm)", bg="white").place(x=95, y=290)
entry_x = tk.Entry(app)
entry_x.place(x=175, y=290)
tk.Label(app, text="Y (mm)", bg="white").place(x=95, y=320)
entry_y = tk.Entry(app)
entry_y.place(x=175, y=320)

# Botón para mover el robot
tk.Radiobutton(app, text="Cordenadas robot", variable=Coordenadas, value="robot", width=15, justify="center", bg="#f2f1d0", font=("Calibri", 11, "bold")).place(x=20, y=355)
tk.Button(app, text="Mover", command=mover_robot, width=15, bg="#ffa621", font=("Calibri", 11, "bold")).place(x=50, y=400)
# Botón para cero
tk.Radiobutton(app, text="Cordenadas origen", variable=Coordenadas, value="origen", width=15, justify="center", bg="#f2f1d0", font=("Calibri", 11, "bold")).place(x=200, y=355)
tk.Button(app, text="Origen", command=asignar_origen, width=15, bg="#ffa621", font=("Calibri", 11, "bold")).place(x=225, y=400)

# Mostrar el resultado (posición o ángulos calculados)
lbl_resultado = tk.Label(app, text="-", width=56, justify="center", bg="white")
lbl_resultado.place(x=0, y=440)

# Altura
tk.Label(app, text="Altura ", bg="white").place(x=95, y=480)
alt = tk.Entry(app)
alt.place(x=175, y=480)

# Ejecutar la aplicación
app.mainloop()
```

REFERENCIAS.

[1] Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer.

[2] González, J., González, R., & Garcia, J. (2014). "Design and implementation of a low-cost robot manipulator for educational purposes." *IEEE Latin America Transactions*.

[3] Bojczuk, K., & Oliva, J. (2015). "Design of a two-link robotic arm with 2 DOF for drawing applications." *International Journal of Engineering Research and Applications*.