INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

REPORTE

BRAZO ROBOTICO

2AM2

INTEGRANTES:

CORDERO RICO ÁNGEL

GUEVARA GUZMAN ERNESTO

LUGO MATEHUALA LEE AYSHANE

INTRODUCCIÓN TEORICA

Este proyecto se centra en el modelado y simulación de un brazo robótico simplificado en dos dimensiones, compuesto por dos segmentos conectados por articulaciones ajustables. Su objetivo es desarrollar un modelo matemático basado en la cinemática directa para calcular las posiciones de las articulaciones y del extremo del brazo a partir de los ángulos y longitudes de los segmentos.

La cinemática directa permite determinar la posición y orientación de los extremos de un sistema robótico a partir de valores conocidos de entrada, como los ángulos de las articulaciones y las longitudes de los segmentos. Este enfoque es esencial para analizar cómo las configuraciones internas del brazo se traducen en movimientos y posiciones en el espacio, siendo clave en el diseño y control de robots.

Para implementar este modelo, se emplean matrices de rotación y traslación. Las matrices de rotación transforman las coordenadas de un punto en función de un ángulo, modelando las rotaciones en un plano 2D. Por su parte, las matrices de traslación desplazan puntos en el espacio, acumulando los efectos de las articulaciones. El uso combinado de estas herramientas matemáticas permite representar con precisión los movimientos del brazo, calculando las posiciones del hombro, codo y extremo de manera eficiente.

Un componente destacado es la visualización gráfica interactiva, que muestra el brazo en diferentes configuraciones. Los usuarios pueden modificar los ángulos de las articulaciones y observar en tiempo real los cambios en las posiciones del brazo. La representación gráfica utiliza colores distintivos para identificar las partes del sistema y facilita la comprensión de las configuraciones posibles.

El sistema permite la entrada de datos de dos formas: manualmente, a través de la consola, donde se ingresan las longitudes de los segmentos y los ángulos de las articulaciones, o mediante archivos CSV, lo que facilita la carga de configuraciones predefinidas para análisis o simulaciones repetitivas.

Este proyecto combina teoría y práctica de la robótica, ofreciendo una herramienta educativa y versátil para comprender los principios fundamentales del movimiento robótico. Al aplicar la cinemática directa y las matrices de rotación y traslación, se ilustra cómo las configuraciones internas del brazo influyen en su posición final. Estas técnicas tienen aplicaciones prácticas en la automatización, el diseño de robots y el control de sistemas mecánicos, sirviendo como una introducción al campo de la robótica moderna.

DESARROLLO DEL SISTEMA

Importación de librerías: Lo primero que se tuvo que realizar para implementar el programa fue importar las librerías necesarias, en este caso numpy, matplotlib, pandas y os.

- numpy (np): Para cálculos matemáticos avanzados (como senos, cosenos, y conversiones de grados a radianes).
- matplotlib.pyplot (plt): Para graficar el brazo robótico y sus posiciones.
- pandas (pd): Para manejar datos estructurados, exportarlos a CSV y trabajar con configuraciones.
- os: Sirve para interactuar con el sistema de archivos, como verificar si un archivo existe.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
```

Calculo de posiciones: Posteriormente se implementaron las funciones para calcular las coordenadas de los puntos (hombro, codo, mano) con base a las longitudes y los ángulos proporcionados por el usuario.

- Convierte los ángulos theta1 y theta2 de grados a radianes.
- Calcula las posiciones: Codo (x_codo, y_codo): Usa trigonometría con el primer segmento.
- Mano (x_mano, y_mano): Se calcula sumando las contribuciones de ambos segmentos.
- Devuelve las coordenadas del hombro, codo y mano.

```
def calcular_posiciones(l1, l2, theta1, theta2):
    theta1 = np.radians(theta1)
    theta2 = np.radians(theta2)
    x_codo = round(l1 * np.cos(theta1), 2)
    y_codo = round(l1 * np.sin(theta1), 2)
    x_mano = round(x_codo + l2 * np.cos(theta1 + theta2), 2)
    y_mano = round(y_codo + l2 * np.sin(theta1 + theta2), 2)
    return (0, 0), (x_codo, y_codo), (x_mano, y_mano)
```

Gráfica: Seguido de eso, encontramos las funciones para la generación de la grafica con los datos previamente calculados.

Entradas:

- posiciones: Coordenadas del hombro, codo y mano.
- l1, l2: Longitudes de los segmentos.
- theta1, theta2: Ángulos.

Visualización:

- Usa líneas para representar los segmentos del brazo.
- Marca las posiciones del hombro, codo y mano con puntos y etiquetas.
- Añade un sistema de ejes y define un rango adecuado basado en la longitud del brazo.

```
def graficar_brazo(posiciones, l1, l2, theta1, theta2):
    origen, codo, mano = posiciones

    plt.figure(figsize=(8, 6))
    plt.plot([origen[0], codo[0]], [origen[1], codo[1]], color='#FFB3BA', linewidth=3, label="Hombro a Codo")
    plt.plot([codo[0], mano[0]], [codo[1], mano[1]], color='#BFFCC6', linewidth=3, label="Codo a Mano")

    plt.scatter(*origen, color='#FF8C42', label="Hombro", s=100, zorder=5)
    plt.scatter(*codo, color='#BAE1FF', label="Codo", s=100, zorder=5)
    plt.scatter(*mano, color='#FFD700', label="Mano", s=100, zorder=5)

    plt.text(origen[0], origen[1], f'Hombro ({round(origen[0], 2)}, {round(origen[1], 2)})', fontsize=10, ha='right', color='#FF8C42')
    plt.text(codo[0], codo[1], f'Codo ({round(codo[0], 2)}, {round(codo[1], 2)})', fontsize=10, ha='left', color='#BAE1FF')
    plt.text(mano[0], mano[1], f'Mano ({round(mano[0], 2)}, {round(mano[1], 2)})', fontsize=10, ha='center', color='#FFD700')

    plt.annotate(f"θ1 = {round(theta1, 2)}°", (origen[0], origen[1] + 1), fontsize=10, color='#FFB3BA')
    plt.annotate(f"θ2 = {round(theta2, 2)}°", (codo[0], codo[1] + 1), fontsize=10, color='#BFFCC6')

    max_longitud = l1 + l2 + 2
    plt.xlim(-max_longitud, max_longitud)
    plt.ylim(-max_longitud, max_longitud)

    plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
    plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
    plt.grid(True)

    plt.title("Visualización del Brazo Robótico", fontsize=14)
    plt.xlabel("Eje X")
    plt.ylabel("Eje Y")
    plt.legend()
    plt.tight_layout()
    plt.show()
```

Exportar a CSV: Posterior a eso, tenemos las funciones que permiten guardar los datos en un archivo CSV.

- Organiza los datos de configuraciones en un formato tabular.
- Usa pandas para exportar los datos a un archivo CSV.

```python
def exportar_a_csv(l1, l2, configuraciones):
    data = {
        "Longitud Segmento 1": [round(l1, 2)] * len(configuraciones),
        "Longitud Segmento 2": [round(l2, 2)] * len(configuraciones),
        "Theta 1 (°)": [round(conf["theta1"], 2) for conf in configuraciones],
        "Theta 2 (°)": [round(conf["theta2"], 2) for conf in configuraciones],
        "Hombro (x, y)": ["(0.00, 0.00)"] * len(configuraciones),
        "Codo (x, y)": [f"({round(conf['codo'][0], 2)}, {round(conf['codo'][1], 2)})" for conf in configuraciones],
        "Mano (x, y)": [f"({round(conf['mano'][0], 2)}, {round(conf['mano'][1], 2)})" for conf in configuraciones]
    }
    df = pd.DataFrame(data)
    df.to_csv("configuraciones_brazo_robotico.csv", index=False)
    print("Datos exportados correctamente a 'configuraciones_brazo_robotico.csv'.")
```

Leer desde CSV: El sistema permite leer archivos con formato CSV para dar la gráfica de los datos dados.

- Lee configuraciones desde un archivo CSV.
- Valida que las columnas requeridas estén presentes.
- Calcula las posiciones y grafica el brazo para cada configuración.

```python
def leer_desde_csv(nombre_archivo):
    try:
        df = pd.read_csv(nombre_archivo)
        columnas_necesarias = ["Longitud Segmento 1", "Longitud Segmento 2", "Theta 1 (°)", "Theta 2 (°)"]
        for columna in columnas_necesarias:
            if columna not in df.columns:
                raise ValueError(f"La columna '{columna}' no está en el archivo CSV.")
        configuraciones = []
        for _, row in df.iterrows():
            l1 = row["Longitud Segmento 1"]
            l2 = row["Longitud Segmento 2"]
            theta1 = row["Theta 1 (°)"]
            theta2 = row["Theta 2 (°)"]
            posiciones = calcular_posiciones(l1, l2, theta1, theta2)
            configuraciones.append({
                "l1": l1,
                "l2": l2,
                "theta1": theta1,
                "theta2": theta2,
                "codo": posiciones[1],
                "mano": posiciones[2],
            })
            graficar_brazo(posiciones, l1, l2, theta1, theta2)
        return configuraciones
    except FileNotFoundError:
        print(f"El archivo '{nombre_archivo}' no existe.")
        return []
    except ValueError as e:
        print(f"Error en el formato del archivo CSV: {e}")
        return []
    except Exception as e:
        print(f"Error inesperado: {e}")
        return []
```

Menú interactivo: Por último, el sistema proporciona una interfaz para interactuar con el sistema.

Implementa un ciclo while que muestra un menú con opciones:

- Ingresar datos manualmente.
- Exportar datos a CSV.
- Leer configuraciones desde CSV.
- Salir del programa.

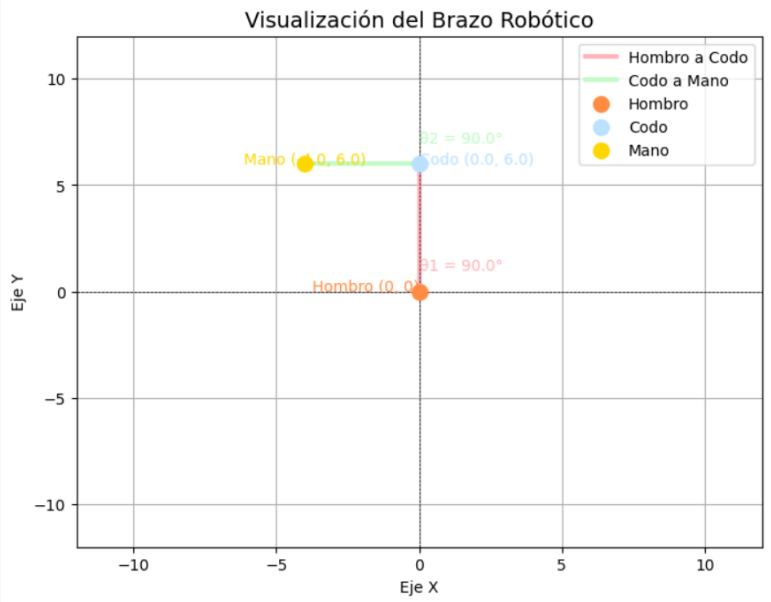
Permite interacción dinámica con el usuario, integrando todas las funciones previas.

```
def menu_interactivo():
    print("Bienvenido al simulador del brazo robótico")
    configuraciones = []
    while True:
        print("\nMenú de opciones:")
        print("1. Ingresar manualmente longitudes y ángulos ")
        print("2. Exportar datos a CSV")
        print("3. Leer configuraciones desde CSV")
        print("4. Salir")
        opcion = input("Elige una opción: ")
        if opcion == "1":
            l1 = float(input("Longitud del segmento 1 (hombro a codo): "))
            l2 = float(input("Longitud del segmento 2 (codo a mano): "))
            theta1 = float(input("Ángulo del hombro (θ1, en grados): "))
            theta2 = float(input("Ángulo del codo (θ2, en grados): "))
            posiciones = calcular_posiciones(l1, l2, theta1, theta2)
            print(f"Posiciones calculadas: Hombro: {posiciones[0]}, Codo: {posiciones[1]}, Mano: {posiciones[2]}")
            configuraciones.append({
                "l1": l1, "l2": l2, "theta1": theta1, "theta2": theta2,
                "codo": posiciones[1], "mano": posiciones[2]
            })
            graficar_brazo(posiciones, l1, l2, theta1, theta2)
        elif opcion == "2":
            if configuraciones:
                exportar_a_csv(configuraciones[0]["l1"], configuraciones[0]["l2"], configuraciones)
            else:
                print("No hay configuraciones para exportar.")
        elif opcion == "3":
            nombre_archivo = input("Ingresa el nombre del archivo CSV: ")
            if not os.path.exists(nombre_archivo):
                print(f"El archivo '{nombre_archivo}' no existe en el directorio actual: {os.getcwd()}")
            else:
                configuraciones = leer_desde_csv(nombre_archivo)
        elif opcion == "4":
            print("Saliendo del programa...")
            break
        else:
            print("Opción no válida. Inténtalo de nuevo.")

menu_interactivo()
```

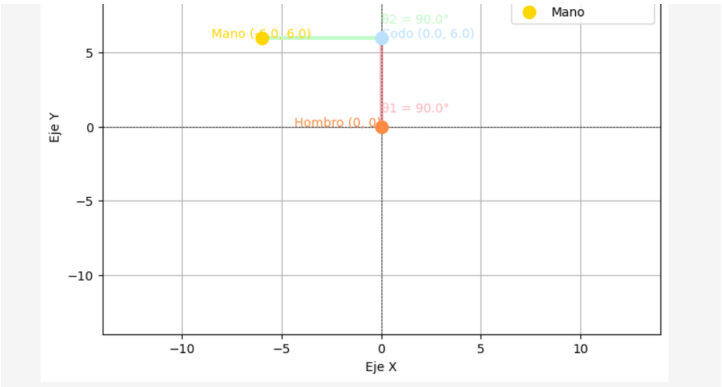
PRUEBAS REALIZADAS

Primera prueba: la primera prueba consto de ingresar los datos de manera manual, una vez ingresados los datos obtenemos la gráfica final con los cálculos realizados por el programa.

```
Bienvenido al simulador del brazo robótico

Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
Posiciones calculadas: Hombro: (0, 0), Codo: (np.float64(0.0), np.float64(6.0)), Mano: (np.float64(-4.0), np.float64(6.0))
```



Segunda prueba: la segunda prueba consistió en que después de ingresar los datos y obtener la gráfica, el sistema guardara los datos en un archivo CSV.




```
Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
Datos exportados correctamente a 'configuraciones_brazo_robotico.csv'.
```

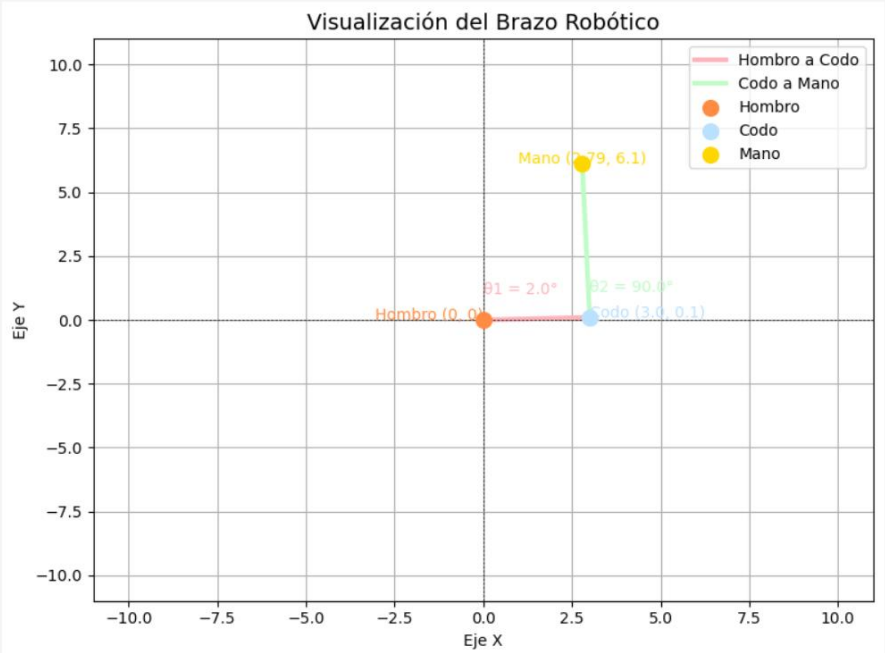
BRAZO ROBOTICO.ipynb ● configuraciones_brazo_robotico.csv ×

configuraciones_brazo_robotico.csv > data

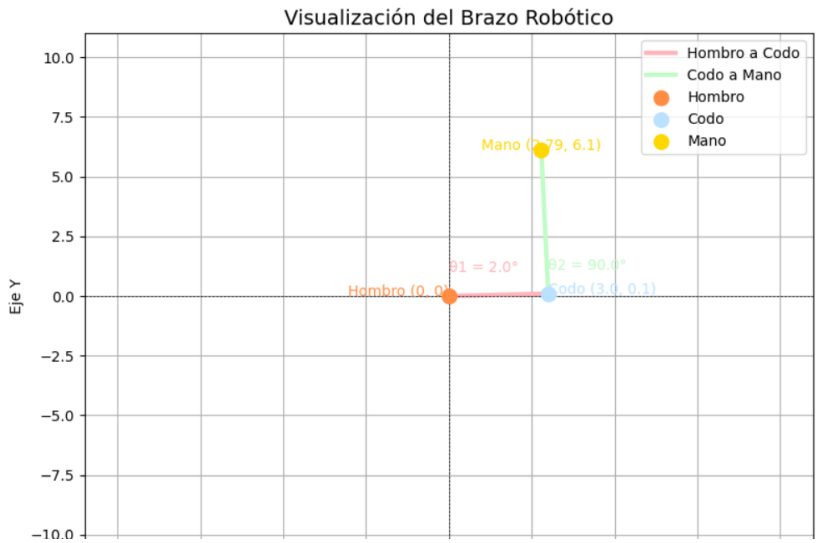
```
1  Longitud Segmento 1,Longitud Segmento 2,Theta 1 (°),Theta 2 (°),"Hombro (x, y)","Codo (x, y)","Mano (x, y)"
2  6.0,6.0,90.0,90.0,"(0.00, 0.00)","(0.0, 6.0)","(-6.0, 6.0)"
3
```

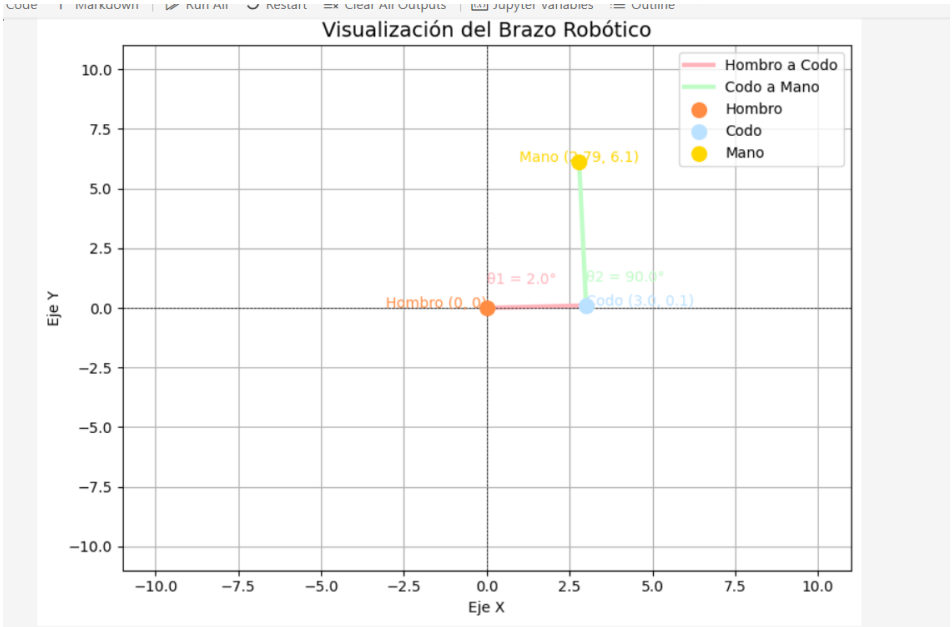
Tercera prueba: Para la tercera y ultima prueba lo que se hizo fue ingresar los datos de manera manual, posteriormente exportar los datos a un archivo CSV, leer ese mismo archivo desde CSV y posteriormente salir, lo que nos dio como salida, la gráfica.

```
Bienvenido al simulador del brazo robótico

Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
Posiciones calculadas: Hombro: (0, 0), Codo: (np.float64(3.0), np.float64(0.1)), Mano: (np.float64(2.79), np.float64(6.1))
```



```
Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
Datos exportados correctamente a 'configuraciones_brazo_robotico.csv'.

Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
```






```
Menú de opciones:
1. Ingresar manualmente longitudes y ángulos
2. Exportar datos a CSV
3. Leer configuraciones desde CSV
4. Salir
Saliendo del programa...
```

CONCLUSIONES

El desarrollo de este proyecto no solo permite explorar los fundamentos de la cinemática directa y el uso de matrices de rotación y traslación, sino que también ofrece una implementación práctica mediante un código funcional que cumple con los objetivos planteados. La herramienta creada permite calcular las posiciones de un brazo robótico simplificado con precisión, mostrar su configuración en una visualización gráfica interactiva y gestionar entradas y salidas de datos de manera eficiente.

En conclusión, el código implementado cumple eficazmente con los requerimientos del proyecto, proporcionando una solución funcional que combina precisión matemática, facilidad de uso e interactividad. Esto no solo lo convierte en una herramienta educativa ideal para comprender los fundamentos de la robótica, sino también en un ejemplo práctico de cómo las matemáticas y la programación pueden unirse para resolver problemas complejos en sistemas reales.