

# Informe Brazo Robotico

## Robotica Industrial

Grupo A1 o G1

Profesor: Edwin Nikolay Prieto Parrado

12 de junio de 2024

## Resumen

En este informe presentaremos detalladamente lo relacionado al brazo robótico de cuatro grados de libertad que diseñamos, ensamblamos y analizamos. Se describirán cada uno de los requerimientos planteados previamente con lujo de detalles y resultados lo más aproximados posibles a los reales.

## 1. Introducción

En la era de la automatización y la robótica, el desarrollo de brazos robóticos ha transformado sectores como la manufactura, la medicina y la exploración espacial. Este informe se enfoca en un brazo robótico de 4 grados de libertad (4DOF), analizando sus componentes, principios de funcionamiento y aplicaciones. Un brazo robótico de 4DOF es capaz de realizar movimientos en cuatro ejes distintos, lo que le confiere flexibilidad y versatilidad para llevar a cabo diversas tareas con alta precisión y repetibilidad.

Se abordarán los aspectos técnicos del brazo robótico, incluyendo su diseño mecánico, sistemas de actuadores y sensores, así como el control y la programación necesarios para su operación. Además, se explorarán las aplicaciones prácticas de este tipo de robot en diferentes industrias, destacando tanto sus capacidades como los desafíos asociados a su implementación. El objetivo es proporcionar una visión integral del funcionamiento y las aplicaciones de un brazo robótico de 4DOF, subrayando su impacto en la eficiencia y la innovación tecnológica.

## 2. Revisión de la literatura

### Antecedentes sobre Brazos Robóticos 4-DoF, Cinemática Inversa y ROS2

La investigación sobre brazos robóticos de 4 grados de libertad (4-DoF), cinemática inversa y ROS2 es un campo amplio y multifacético dentro de la robótica y la ingeniería.

En esta investigación se presenta una revisión detallada de los antecedentes y conceptos clave en estas áreas.

### **Brazos Robóticos de 4-DoF**

Los brazos robóticos de 4-DoF son dispositivos que permiten movimientos precisos y controlados en un espacio tridimensional. Su uso abarca aplicaciones en manufactura, medicina y educación.

1. Diseño y Construcción de un Brazo Robótico de 4 Grados de Libertad: Este estudio describe el diseño y construcción de un brazo robótico tipo SCARA (RRP) de 4-DoF, cuyo efecto final es un gripper. Está orientado a facilitar el aprendizaje en estudiantes de robótica a nivel universitario, proporcionando una plataforma práctica para el estudio de la robótica.
2. Diseño, Simulación y Construcción de un Robot Manipulador Serial de 4 Grados de Libertad: En este trabajo, se desarrolla un brazo robótico serial de 4-DoF utilizando algoritmos de cinemática inversa y manufactura aditiva. Se presenta un control basado en un programa que interpreta archivos de trayectoria, permitiendo la automatización de tareas específicas.
3. Sistema Teledirigido de un Brazo Robótico de 4 Grados de Libertad: Este estudio investiga un sistema de control para el movimiento teledirigido de un brazo robótico de 4-DoF, utilizando algoritmos de visión de máquina y teleoperación para lograr un control preciso y remoto.

Los brazos robóticos de 4-DoF realizan cuatro movimientos independientes: rotación de base, articulación del hombro, articulación del codo y articulación de la muñeca. Estas capacidades son esenciales en aplicaciones industriales como la soldadura, ensamblaje y manipulación de materiales, así como en entornos de investigación.

### **Cinemática Inversa**

La cinemática inversa es un problema fundamental en la robótica, que consiste en calcular las posiciones angulares de las articulaciones de un robot para alcanzar una posición y orientación deseadas del efecto final.

1. Cinemática Inversa y Análisis de Trayectoria: Este trabajo se centra en el análisis de trayectoria mediante cinemática directa e inversa para un manipulador de 4-DoF. Utiliza un análisis geométrico parcial y métodos iterativos para determinar los ángulos de las articulaciones necesarios para posicionar el efecto final.
2. Análisis de un Brazo Robótico con Gazebo y ROS para la Cinemática Inversa: Este estudio presenta un análisis detallado de un brazo robótico utilizando Gazebo y ROS para resolver la cinemática inversa. Se enfoca en la integración de servomotores y la interpretación geométrica de los algoritmos de cinemática inversa.

La resolución de la cinemática inversa es crucial para que un brazo robótico pueda realizar tareas precisas, como ensamblaje de piezas, cirugía robótica y manipulación de objetos en entornos no estructurados.

## **ROS2**

ROS2 (Robot Operating System 2) es una plataforma de software de código abierto utilizada para la programación y control de robots. Proporciona herramientas y bibliotecas para el desarrollo y ejecución de aplicaciones robóticas, basándose en una arquitectura de publicación-suscripción que facilita la comunicación eficiente entre los componentes del software. Algunos estudios relevantes son:

1. Diseño y Construcción de un Brazo Robótico de 4 Grados de Libertad orientado a la Educación Universitaria: Este trabajo utiliza ROS2 para el control del brazo robótico diseñado, implementando un control PID en la segunda articulación para mejorar la precisión y estabilidad del movimiento.
2. Análisis de un Brazo Robótico con Gazebo y ROS para la Cinemática Inversa: Este estudio utiliza ROS2 para integrar el brazo robótico con Gazebo y realizar análisis de cinemática inversa, demostrando la utilidad de ROS2 en la simulación y control de robots.

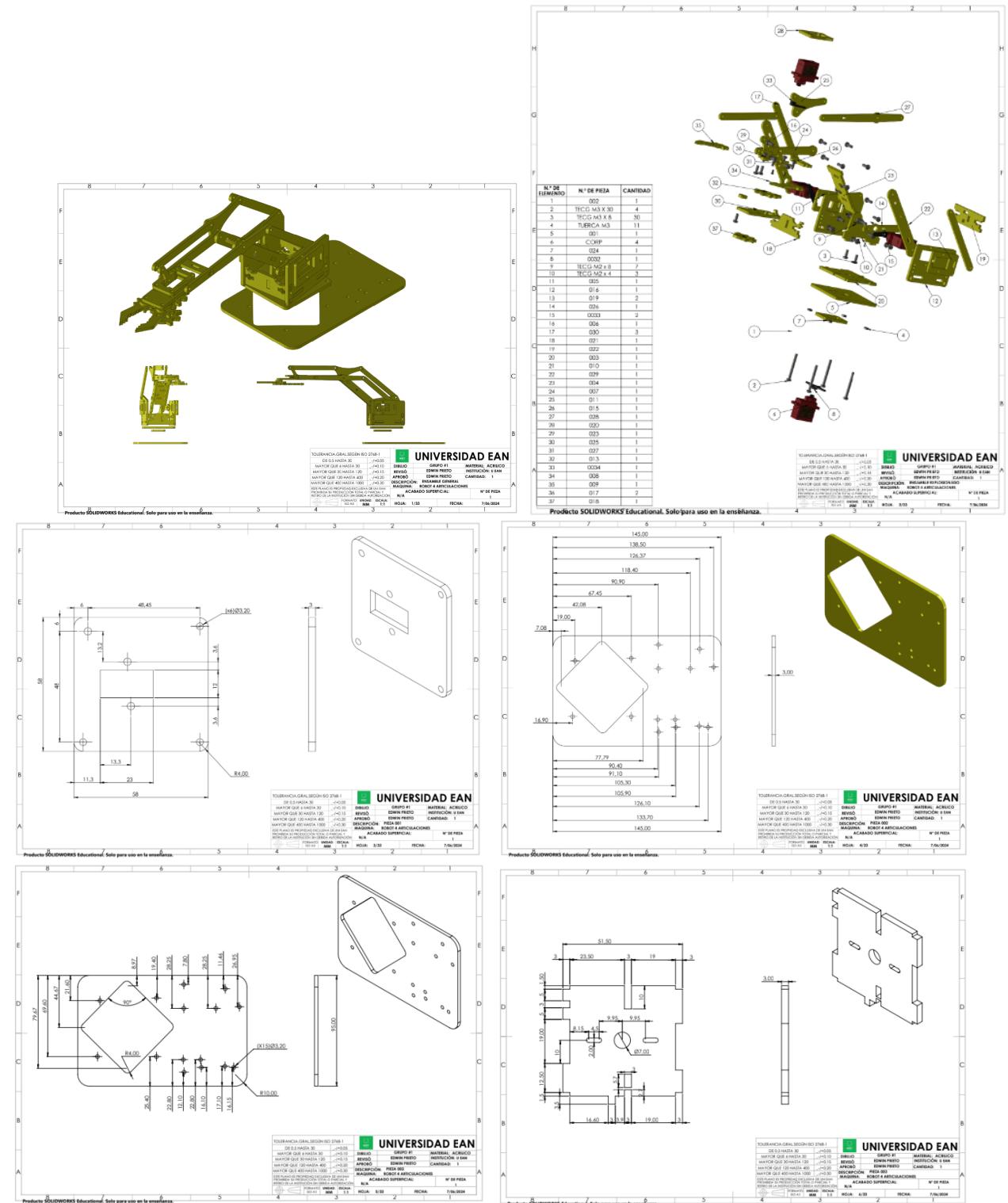
ROS2 es ampliamente utilizado en el desarrollo de brazos robóticos debido a su robusta infraestructura de comunicación y su flexibilidad para implementar algoritmos complejos e integrar diversos sensores y actuadores.

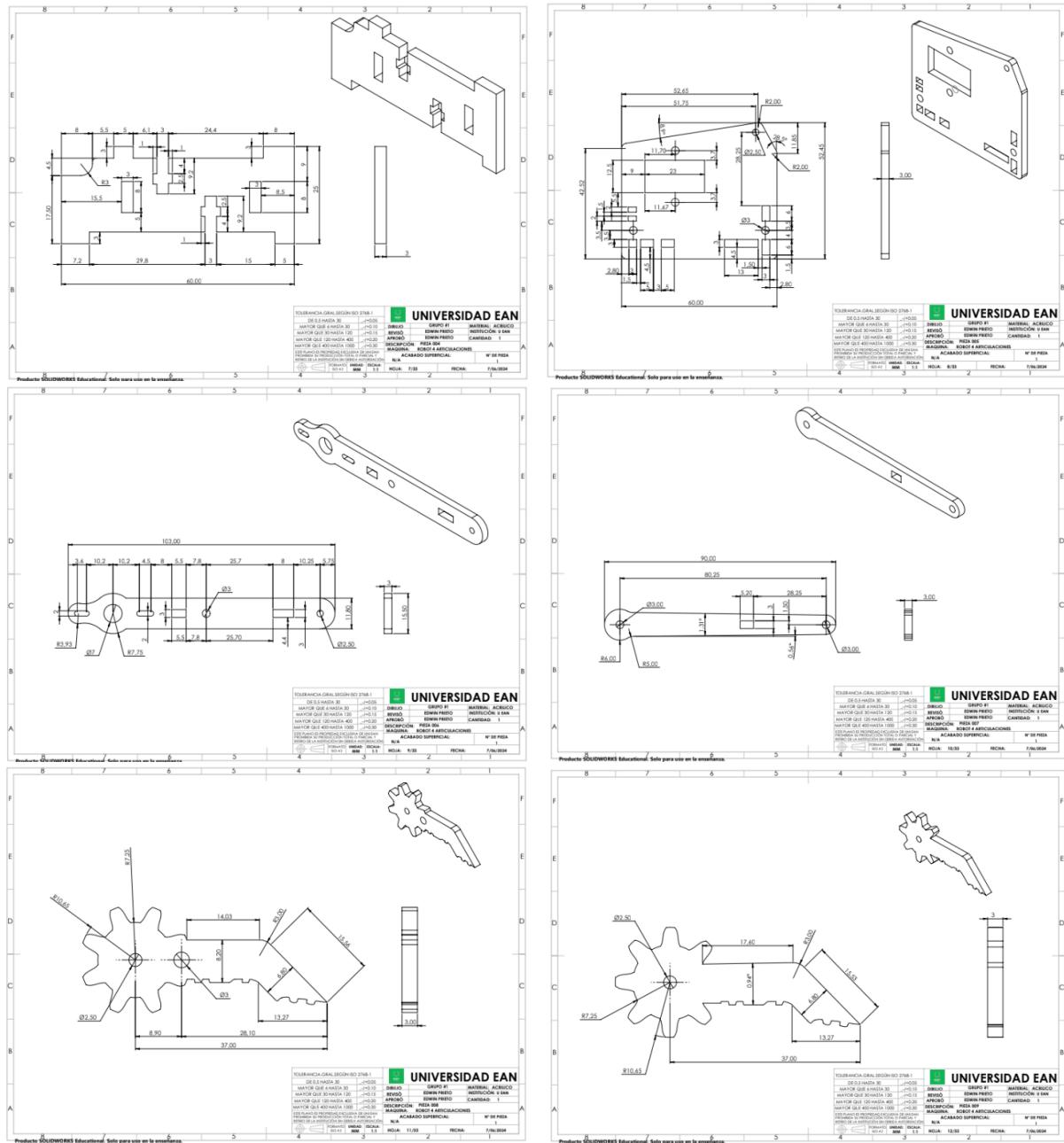
## **Conclusión**

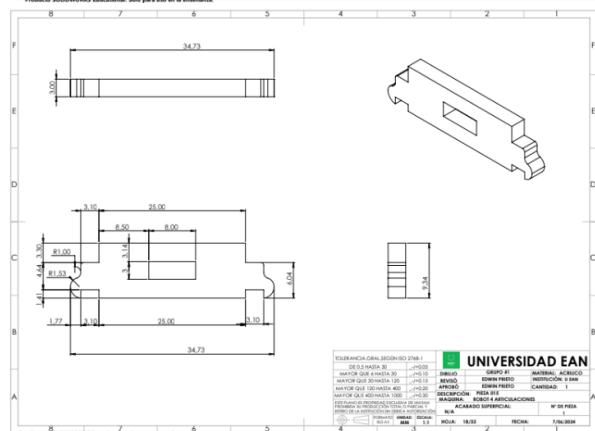
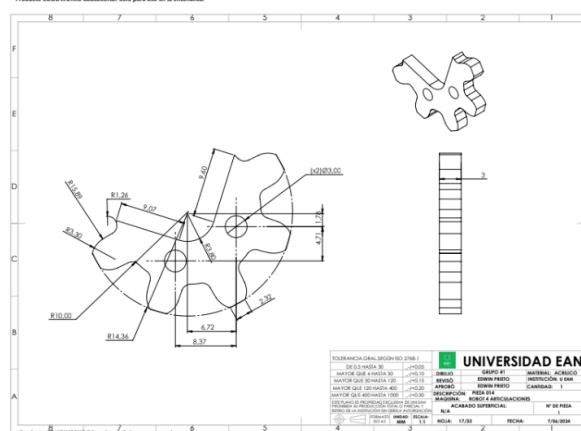
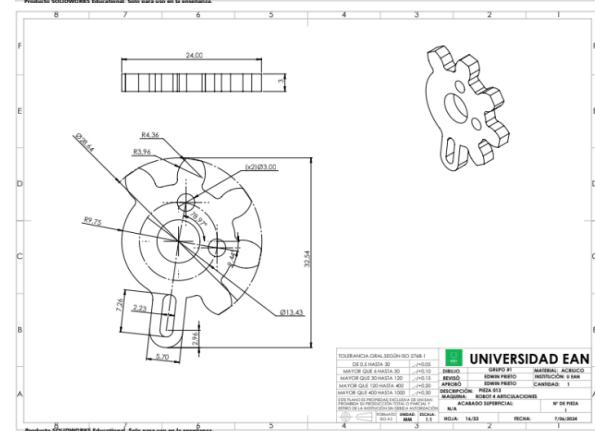
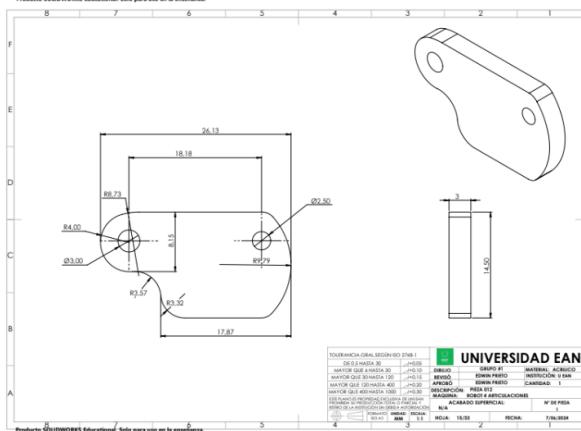
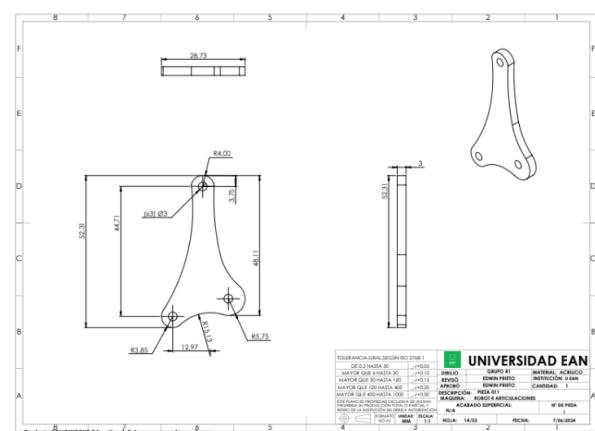
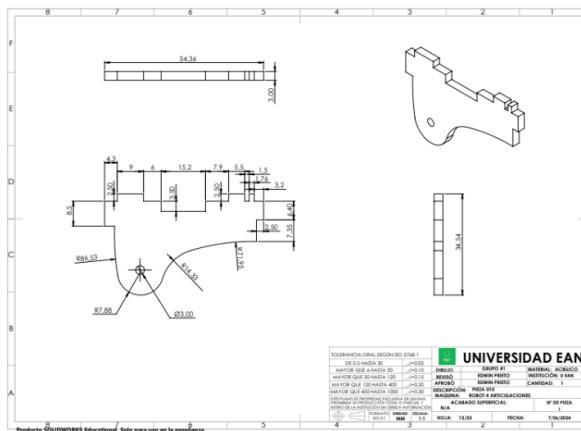
Básicamente, la investigación sobre brazos robóticos de 4-DoF, cinemática inversa y ROS2 abarca una amplia gama de estudios que abordan el diseño, construcción y control de brazos robóticos, así como el análisis de trayectorias y la integración de tecnologías avanzadas como Gazebo y ROS2. Estos antecedentes proporcionan una base sólida para el desarrollo de futuros proyectos en robótica y mecatrónica, orientados a mejorar el rendimiento, precisión y robustez de los sistemas robóticos.

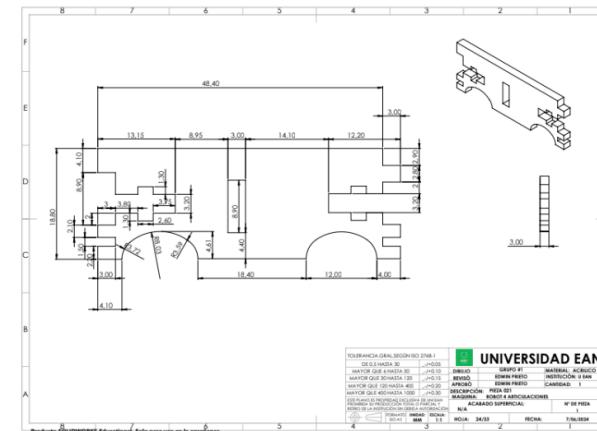
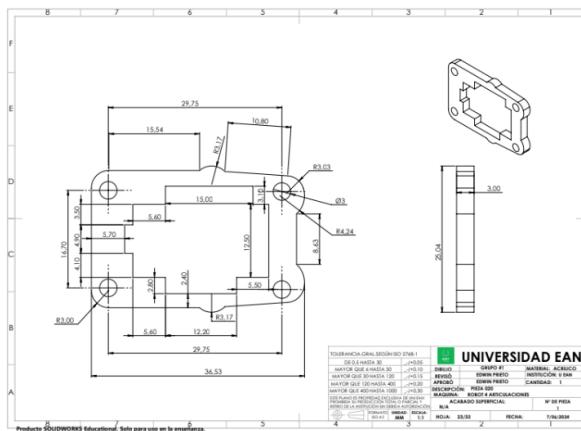
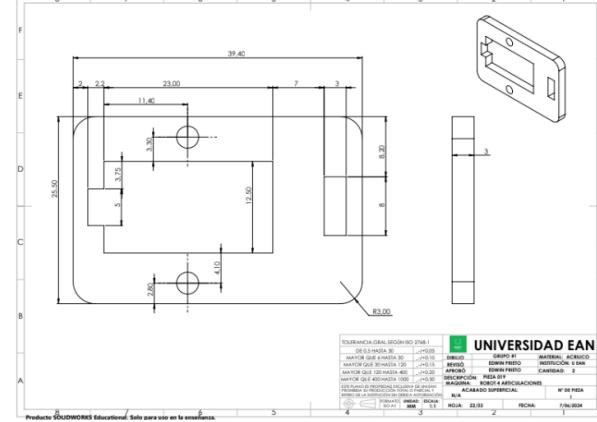
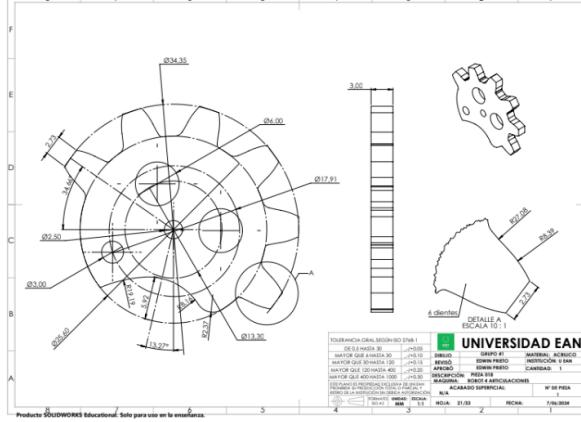
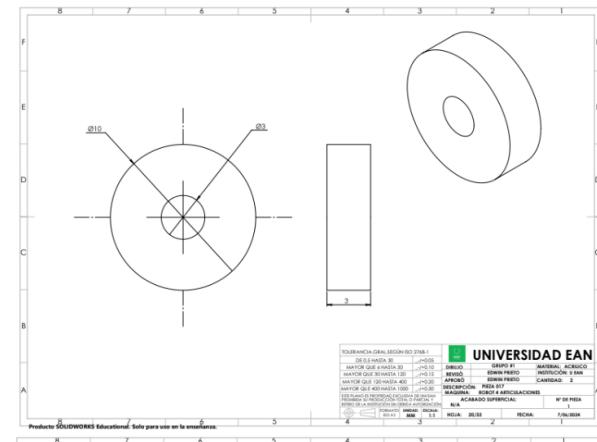
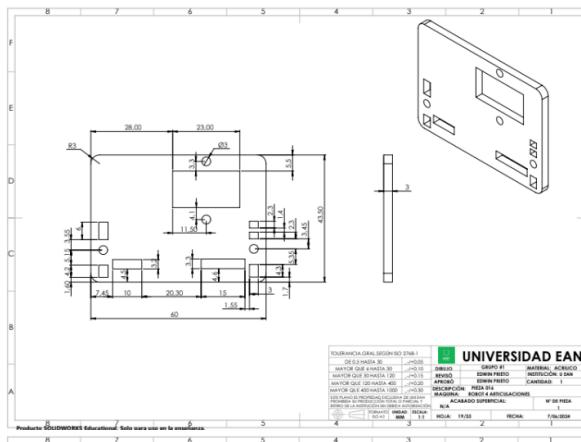
La continua investigación en estos campos es esencial para el avance de la robótica, con aplicaciones potenciales que abarcan desde la manufactura avanzada hasta la medicina y la exploración espacial.

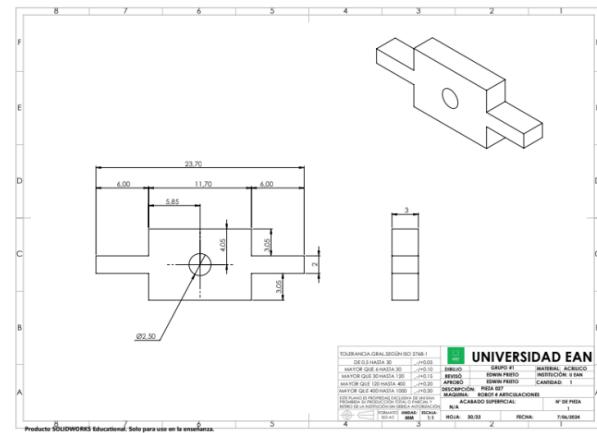
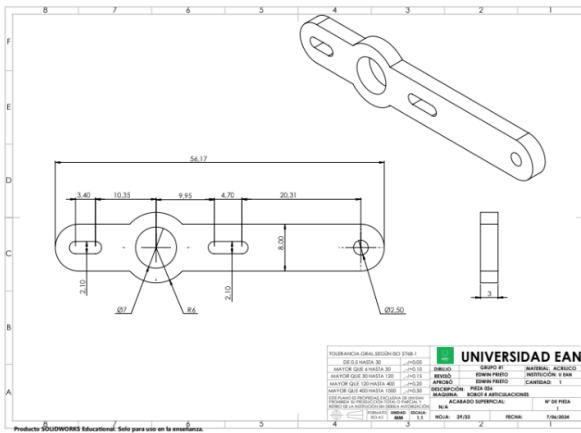
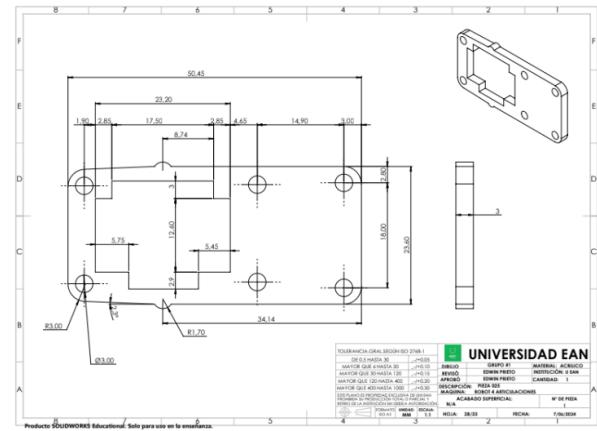
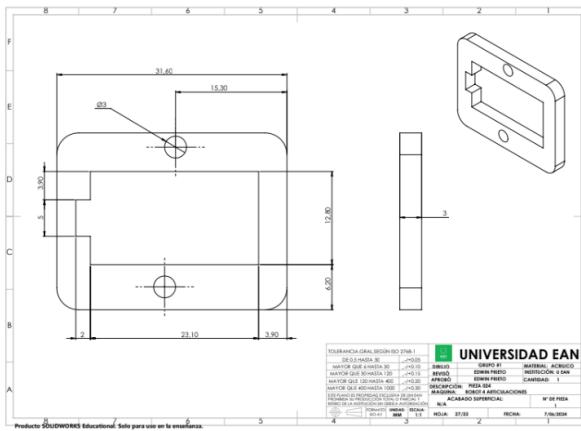
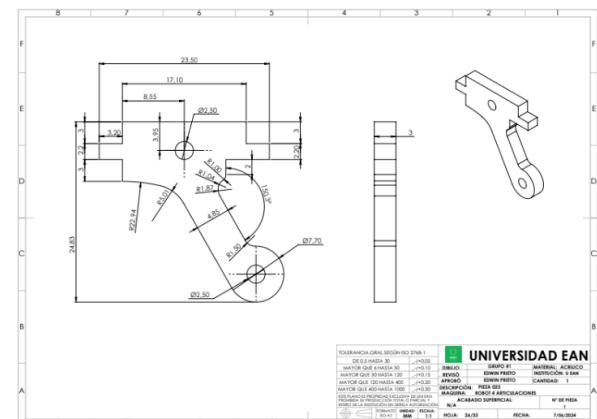
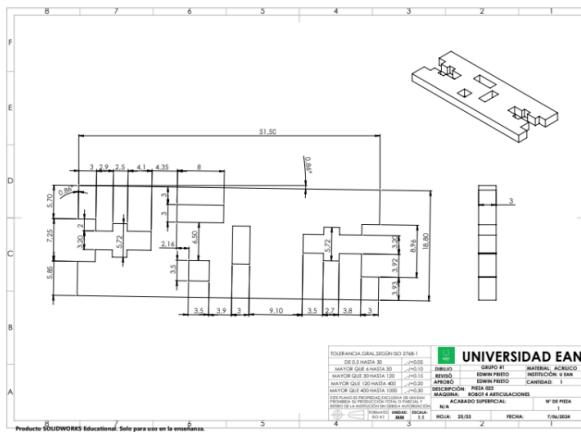
### 3. Diseño del sistema

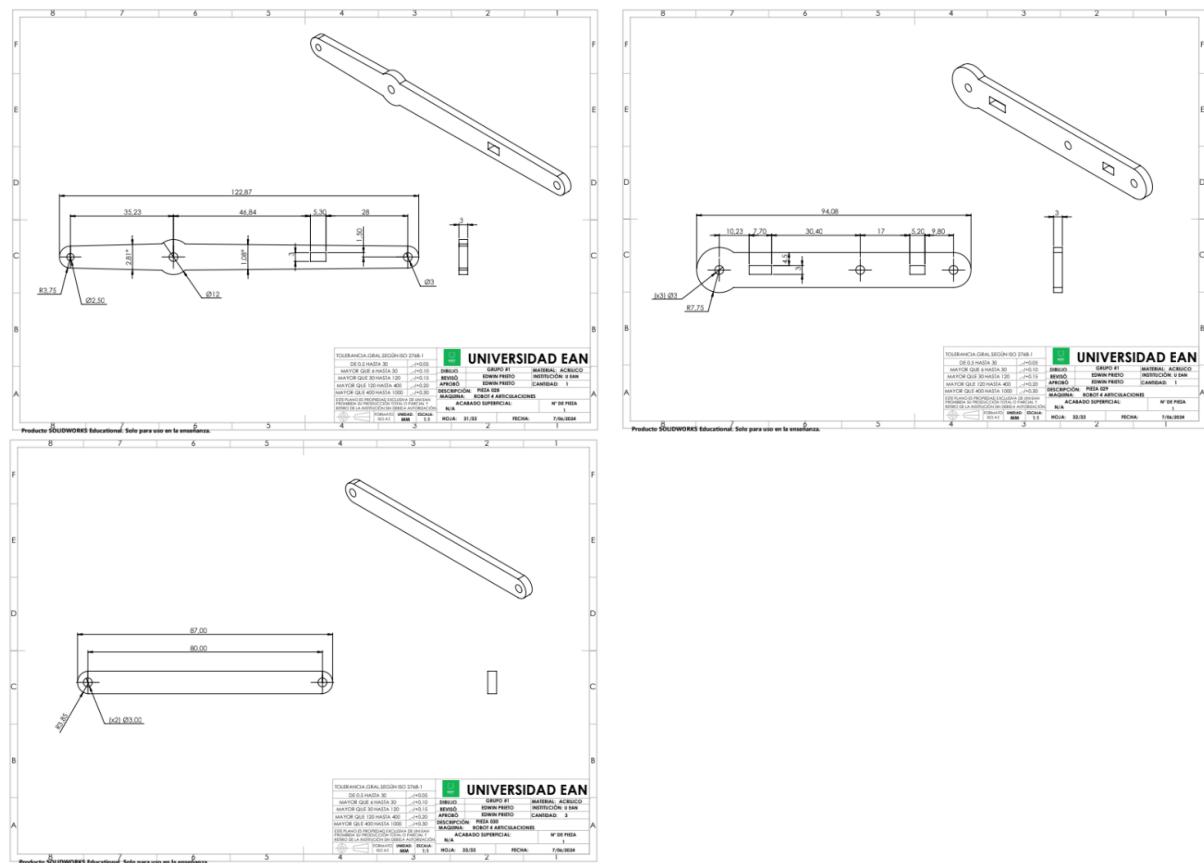






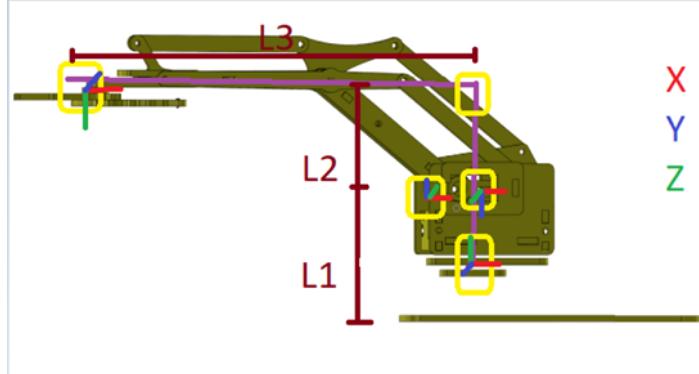






#### 4. Análisis cinematico

# Cinemática Directa Denavit-Hartenberg



Articulacion	Z		X	
	theta	d1	a1	alfa1
1	Theta1	L1	0	90
2	theta2	0	0	180
3	theta3	L2	0	0
4	theta4	0	L3	-90

L1 = 7 cm
L2 = 8cm
L3= 20 cm

## Matrices Homogeneas

$$T = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & a * \cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & a * \sin(\theta) \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Articulación 1:

$$T_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & L_1 * \cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & L_1 * \sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Articulación 2:

$$T_2 = \begin{bmatrix} -\cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Articulación 3:

$$T_3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L_2 * \cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & L_2 * \sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Articulación 4:

$$T_4 = \begin{bmatrix} -\sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & 1 & 0 & 0 \\ \cos(\theta_4) & 0 & \sin(\theta_4) & L_3 * \cos(\theta_4) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Matriz de transformación

$$T = T_4 * T_3 * T_2 * T_1$$

$$T = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_3 + \theta_4) & -\sin(\theta_1 + \theta_2 + \theta_3 + \theta_4) & 0 \\ \sin(\theta_1 + \theta_2 + \theta_3 + \theta_4) & \cos(\theta_1 + \theta_2 + \theta_3 + \theta_4) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \\ 0 \\ 1 \end{bmatrix}$$

## Cinemática inversa

```

1 import numpy as np
2 from roboticstoolbox import DHRobot, RevoluteDH
3 from spatialmath import SE3
4
5 # Definimos los parámetros del robot
6 L1 = 0.07
7 L2 = 0.08
8 L3 = 0.2
9
10 # Definimos los enlaces utilizando los parámetros DH
11 links = [
12     RevoluteDH(d=0, a=L1, alpha=np.pi/2),
13     RevoluteDH(d=0, a=0, alpha=np.pi, offset=np.pi),
14     RevoluteDH(d=0, a=L2, alpha=0),
15     RevoluteDH(d=0, a=L3, alpha=-np.pi/2),
16 ]
17
18 # Creamos el objeto Robot
19 robot = DHRobot(links, name="SimpleRobot")
20
21 # Definimos la posición deseada
22 pos_deseada = SE3(0.1, 0.2, 0.3)
23
24 # Calculamos la cinemática inversa
25 ik_sol = robot.ikine_LM(pos_deseada)
26
27 # Imprimimos las soluciones
28 if ik_sol.success:
29     print("Soluciones:")
30     print(ik_sol.q)
31 else:
32     print("No se encontraron soluciones")

```

Este código es la manera de cómo utilizamos la biblioteca roboticstoolbox para calcular la cinemática inversa de un robot teniendo en cuenta la tabla de Denavit-Hartenberg (DH) y la biblioteca spatialmath para manipulaciones espaciales como la creación de matrices de transformación homogénea.

- ✓ `import numpy as np`: Importa la biblioteca NumPy bajo el nombre np, es comúnmente utilizada para operaciones matriciales y numéricas en Python.

- ✓ from roboticstoolbox import DHRobot, RevoluteDH: Importa la clase DHRobot y RevoluteDH desde la biblioteca roboticstoolbox. Esta biblioteca proporciona herramientas para trabajar con robots manipuladores, incluyendo cálculos de cinemática directa, inversa, y simulaciones.
- ✓ from spatialmath import SE3: Importa la clase SE3 desde la biblioteca spatialmath, que se utiliza para representar transformaciones espaciales.
- ✓ Se definen los parámetros del robot: L1, L2 y L3, que representan las longitudes de los eslabones del robot manipulador.
- ✓ Se define la estructura del robot utilizando los parámetros DH. Cada enlace se define como un objeto RevoluteDH, que especifica los parámetros Denavit-Hartenberg de cada eslabón del robot
- ✓ Se crea el objeto del robot robot utilizando la clase DHRobot, pasando la lista de enlaces creada anteriormente y asignándole un nombre.
- ✓ Se define la posición deseada del efecto final del robot utilizando una matriz de transformación homogénea (SE3). En este caso, se establece una posición específica (0.1, 0.2, 0.3) en el espacio tridimensional.
- ✓ Se calcula la cinemática inversa del robot para alcanzar la posición deseada utilizando el método ikine-LM() del objeto robot. Este método implementa un algoritmo de mínimos cuadrados para resolver la cinemática inversa.
- ✓ Se verifica si se encontraron soluciones para la cinemática inversa. Si ik-sol.success es True, significa que se encontraron soluciones y se imprimen; de lo contrario, se imprime un mensaje indicando que no se encontraron soluciones.

Básicamente este código muestra cómo definir la estructura de un robot manipulador utilizando parámetros Denavit-Hartenberg y calcular su cinemática inversa para alcanzar una posición deseada en el espacio tridimensional.

## 5. Estrategia de control

El objetivo de controlar este brazo robótico es a través de un código que nos permite conectar Visual Studio con Arduino para manejar el brazo mediante una cámara que detecta el movimiento de nuestros gestos. El elemento principal encargado de esta orquestación es el Arduino Uno, que con sus múltiples capacidades de procesamiento nos permite lograrlo.

## **6. Impresión y construcción 3D**

### **Proceso de Impresión 3D:**

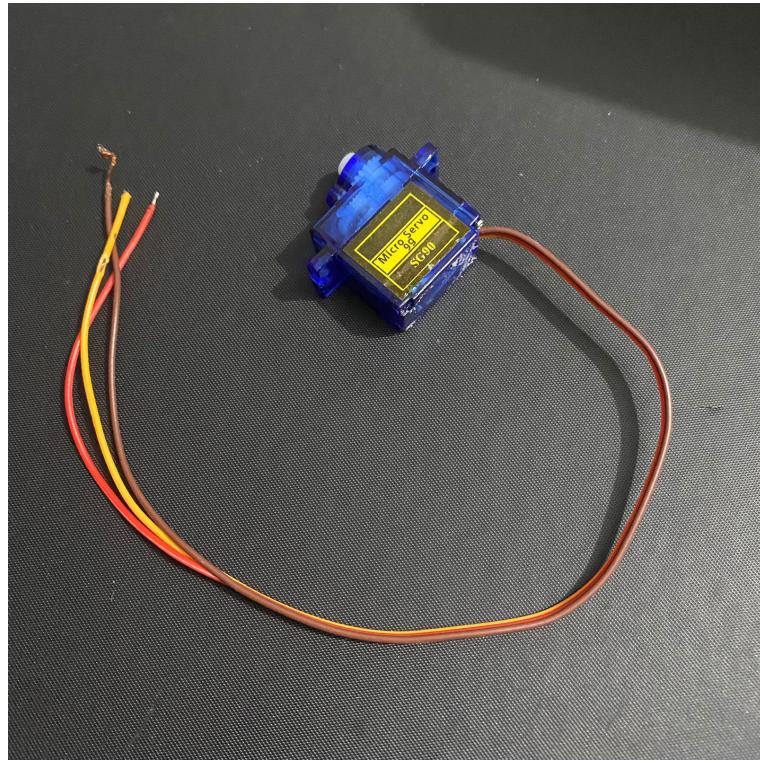
1. Diseño del Brazo Robótico: Utilizamos un software de diseño 3D como SolidWorks, para diseñar cada componente del brazo robótico. Aseguramos de diseñarlos de acuerdo con las especificaciones del proyecto, teniendo en cuenta la resistencia y la funcionalidad.
2. Preparación del Archivo para Impresión: Una vez que tuvimos los diseños listos, exporta cada componente como un archivo STL o compatible con la impresora 3D.
3. Post-Procesamiento: Una vez que la impresión estuvo completa, eliminamos soportes, lijamos las superficies para mejorar la estética, y aplicamos recubrimientos para mayor resistencia.

### **Instrucciones de Montaje:**

4. Ensamblaje de los Servomotores: Coloca cada servomotor en su posición correspondiente según el diseño de nuestro brazo robótico. Utilizamos tornillos M3 y tuercas M3 para fijar los servomotores en su lugar.
5. Pruebas y Ajustes: Una vez que el brazo robótico estuvo ensamblado, realizamos pruebas para asegurarnos de que todos los movimientos funcionen correctamente.

### **Materiales Utilizados:**

6. Material Acrílico: Utilizamos láminas de acrílico para imprimir los componentes principales del brazo robótico. El acrílico es duradero, ligero y fácil de trabajar.



7. Servomotores: Necesitamos 4 servomotores para controlar los diferentes movimientos del brazo robótico. Nos aseguramos de elegir servomotores que fueran compatibles con las necesidades de torque y velocidad.



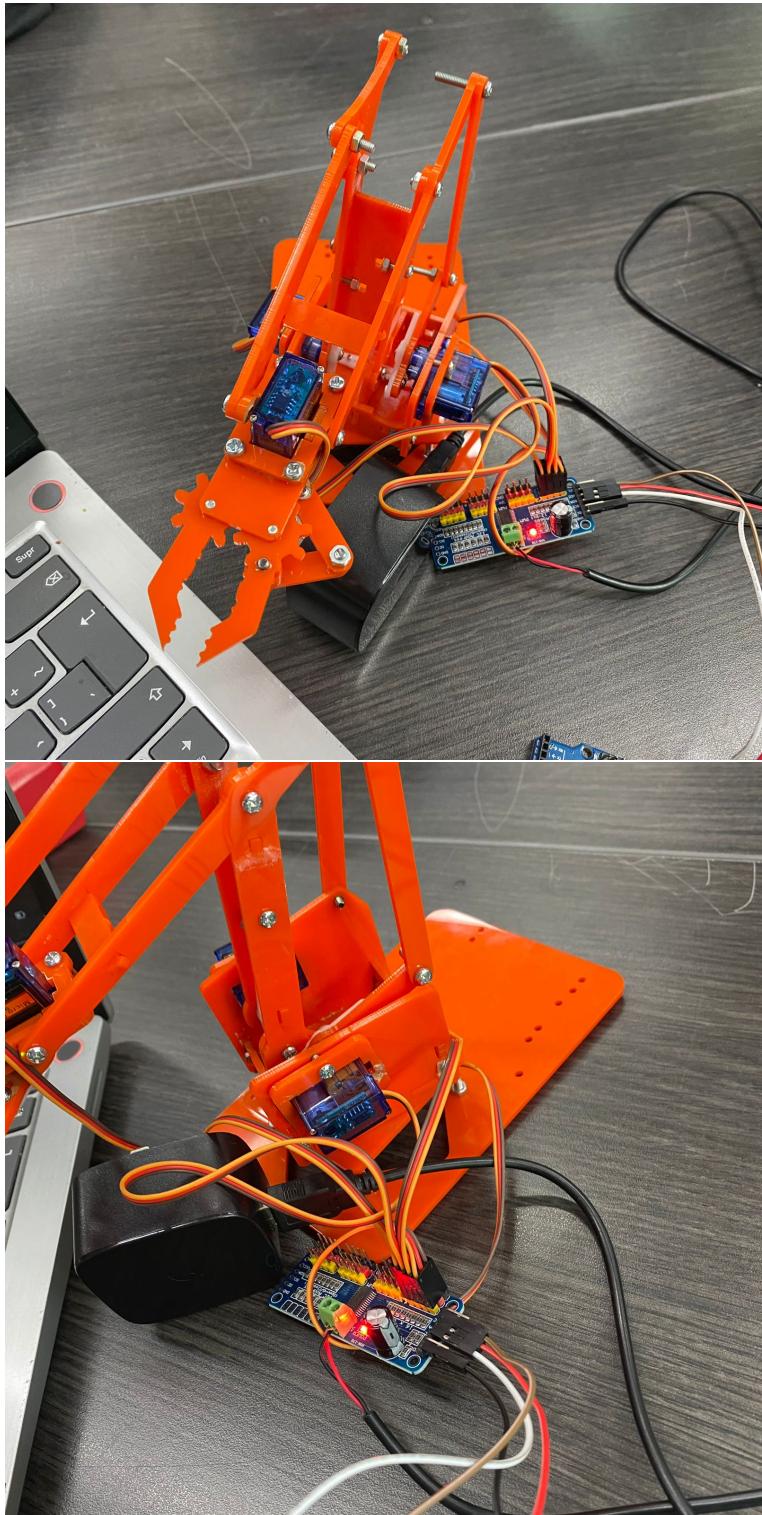
8. Tornillos M3 (3 mm): Utilizamos tornillos M3 de 3 mm de longitud para fijar los servomotores y otros componentes del brazo robótico.
9. Tornillos M3 (8 mm): Los tornillos M3 de 8 mm se utilizaron para unir las diferentes piezas de articulaciones impresas del brazo robótico entre sí.
10. Tuercas M3: Utilizamos tuercas M3 para asegurar los tornillos en su lugar y proporcionar una conexión segura entre los componentes del brazo robótico como por ejemplo en la base.

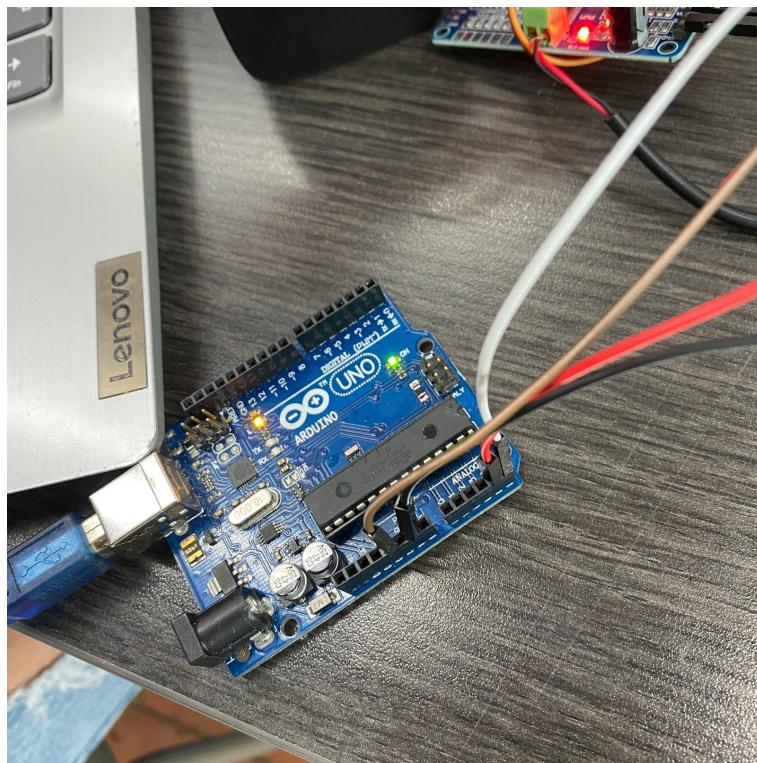


## 7. Resultados y discusión

### Brazo Robotico

El brazo es capaz de coger un objeto en una posición predeterminada desde el código y desplazarlo dentro de sus posibilidades a donde se le indique, esto mediante el procesamiento de instrucciones con la ayuda de los diferentes componentes, principalmente el arduino uno, quien es el que orquesta todo el funcionamiento y nos brinda una interacción mas amigable.





## Código

El código setea valores, como el valor 0 el cual sería la posición inicial, los valores donde agarrara el objeto y donde lo dejara. Esto es configurable, ya que se puede setear que tanto suve, que tanto baje o hasta donde se desplaza.

## Código en Arduino

ARDUINO:

```
#include <Servo.h>

Servo servo1; // Codo
Servo servo2; // Muñeca

void setup() {
    servo1.attach(9); // Pin al que está conectado el servo del codo
    servo2.attach(10); // Pin al que está conectado el servo de la muñeca
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        String data = Serial.readStringUntil('\n');
        int separator = data.indexOf(',');
        int angle1 = data.substring(0, separator).toInt();
        int angle2 = data.substring(separator + 1).toInt();

        servo1.write(angle1);
        servo2.write(angle2);
    }
}
```

## Código de visualstudio

```
import cv2
import mediapipe as mp
import numpy as np
import serial
import time

# Inicializar la conexión serial con Arduino
arduino = serial.Serial('COM3', 9600) # Reemplaza 'COM3' con el puerto correcto
time.sleep(2) # Esperar a que la conexión se establezca

# Inicializar Mediapipe Pose
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

def calculate_angle(a, b, c):
    a = np.array(a) # Primer punto
    b = np.array(b) # Segundo punto
    c = np.array(c) # Tercer punto

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return angle

# Inicializar la cámara
cap = cv2.VideoCapture(0)

with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Convertir la imagen a RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Realizar la detección
        results = pose.process(image)

        # Convertir la imagen de nuevo a BGR para OpenCV
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Dibujar las anotaciones en la imagen
        if results.pose_landmarks:
            mp_drawing.draw_landmarks(
                image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)

            landmarks = results.pose_landmarks.landmark

            # Coordenadas del hombro, codo y muñeca
            shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,
                        landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
            elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
                     landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
            wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
                     landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]

            # Calcular el ángulo del codo
            elbow_angle = calculate_angle(shoulder, elbow, wrist)

            # Enviar los ángulos a Arduino
            data = f"{int(elbow_angle)},90\n" # Asumiendo un ángulo fijo para la muñeca
            arduino.write(data.encode())

            # Mostrar la imagen con las anotaciones
            cv2.imshow('Brazo Robótico', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
arduino.close()
```

## 8. Conclusión

1. Eficiencia y Precisión Mejoradas: Los brazos robóticos de 4DOF aumentan la eficiencia y precisión en procesos industriales, reduciendo errores y costos operativos, y mejorando la calidad del producto.
2. Flexibilidad y Adaptabilidad: Estos robots ofrecen gran flexibilidad y pueden adaptarse a diversas tareas industriales, permitiendo una rápida respuesta a cambios en la producción y demanda del mercado.
3. Mejora en la Colaboración Humano-Robot: Los brazos robóticos de 4DOF facilitan la colaboración efectiva entre humanos y robots, optimizando procesos y aumentando la productividad y seguridad en el entorno industrial.

## Referencias

- [1] RobotShop. (s/f). Brazo Robótico Lynxmotion SES V2 - 4 DOF con Servos Inteligentes Instalados. Recuperado de <https://eu.robotshop.com/es/products/brazo-robotico-lynxmotion-ses-v2-4-dof-c-servos-inteligentes-instalados>
- [2] Bonga. (s/f). Título del recurso. Recuperado de <https://bonga.unisimon.edu.co/items/75d069f2-09df-4c66-84f3-260ae24a93ce>
- [3] Autor, A. (Año). Título del artículo. *Nombre de la revista, volumen*(número), páginas. Recuperado de [http://www.scielo.org.co/scielo.php?pid=S1794-12372015000200008&script=sci\\_arttext](http://www.scielo.org.co/scielo.php?pid=S1794-12372015000200008&script=sci_arttext)
- [4] Quanser. (s/f). Título del recurso. Recuperado de <https://www.quanser.com/>
- [5] Solectroshop. (s/f). Título del recurso. Recuperado de <https://solectroshop.com/es/>
- [6] Lynxmotion. (s/f). Título del recurso. Recuperado de <https://www.lynxmotion.com/>
- [7] igus. (s/f). Título del recurso. Recuperado de <https://www.igus.es/info/articulated-robot>