Universidad Autónoma Chapingo

Departamento de Mecánica Agrícola

Ingeniería Mecatrónica Agrícola

Proyecto Final: Robot tipo SCARA de 2GDL

Asignatura:

DINÁMICA Y CONTROL DE ROBOTS

Nombre del profesor:

Luis Arturo Soriano Avendaño

Alumno:

Jym Emmanuel Cocotle Lara [1710451-3]

GRADO:	GRUPO:
7°	7

Chapingo, Texcoco Edo. México

Fecha de entrega: 29/12/2021

Índice

Introducción

De acuerdo con Leopoldo (Sánchez, L. A. 2005) un robot es cualquier estructura mecánica que opera con un cierto grado de autonomía, bajo el control de un computador, para la realización de una tarea, y que dispone de un sistema sensorial más o menos evolucionado para obtener información de su entorno.

Un robot está compuesto por una serie de elementos hardware, como son: una estructura mecánica, un sistema de actuación, un sistema sensorial interno, un sistema sensorial externo y un ordenador en el que se encuentra un software que gestiona el sistema sensorial y mueva la estructura mecánica para la realización de una determinada tarea.

En este informe se muestra el prototipo de un robot tipo SCARA de 2 grados de libertad, esto con ayuda de diferentes componentes tanto de hardware como de software, entre los cuales de utilizo, servomotores para permitir el movimiento, una tarjeta de desarrollo Arduino el cual indico a los servomotores los ángulos correctos dependiendo el caso, dichos ángulos fueron obtenidos a partir de un programa realizado en el software de Python, el cual contiene diferentes casos para la determinación de los ángulos.

La primera parte del programa de Python permite la obtención de los ángulos a partir de coordenadas en los ejes X y Y (cinemática inversa), de igual forma permite almacenar varias coordenadas para posteriormente ser ejecutadas de manera secuencial y muestra las diferentes posiciones a través de una gráfica en 3 dimensiones.

La segunda parte del programa permite el movimiento del robot a través de ángulos proporcionados por el usuario de forma manual (cinemática directa), y contiene de igual forma un sistema de almacenamiento de ángulos y las gráficas de cada combinación de ángulos.

Por último, con ayuda de visión artificial se obtienen las coordenadas de un objeto con un color especifico (rojo,verde y azul), con lo cual, con ayuda de botones poder seleccionar que clasificación se desea.

Los diferentes resultados de las diferentes partes del programa se muestran en el siguiente video:

https://www.youtube.com/watch?v=tqwliawR5pk

Desarrollo

En este informe se comenzará hablando del prototipo y de los elementos que lo conforman, para posteriormente mostrar el programa empleado en el robot y los resultados obtenidos del mismo.

Prototipo

Materiales:

- Piezas de madera
- 1x Arduino Uno
- 1x Módulo PCA9685
- 3x servomotor MG995
- Jumpers
- 1x Protoboard
- 1x Clavo
- Cinta de aislar
- Alambre magneto
- Cámara de celular
- 1x Laptop
- 1x Relé

Construcción

Con respecto a la construcción, primeramente, se armó el robot tipo SCARA con ayuda de las piezas de madera las cuales fueron obtenidas por “Mercado Libre”, por lo cual, las medidas ya estaban definidas, se unieron las piezas y se colocaron los servomotores.

Para el caso del actuador empleado, se fabricó un electroimán con ayuda de el alambre magneto y el clavo, con lo cual con ayuda del Relé se pudo encender y apagar de manera eficiente.

Por último, se conectaron los servomotores al módulo PCA9685 y el módulo se conectó al Arduino Uno, dando como resultado el prototipo mostrado en la figura 1.

Fig 1.- Prototipo de robot tipo SCARA

Cabe mencionar que la protoboard se empleo ya que se necesitaban varios puertos de voltaje y tierra del Arduino Uno.

Programas empleados

Programa de Arduino

```
#include <string.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

#define RELE 2

Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);

unsigned int pos0=172; // ancho de pulso en cuentas para posición 0°
unsigned int pos180=565; // ancho de pulso en cuentas para la posición
180°

void setup() {
  Serial.begin(9600);
  delay(30);
  pinMode(RELE, OUTPUT);
  digitalWrite(RELE,0);
  servos.begin();
  servos.setPWMFreq(60); //Frecuecia PWM de 60Hz o T=16,66ms
}
void setServo(uint8_t n_servo, int angulo) {
  int duty;
  duty=map(angulo,0,180,pos0, pos180);
  servos.setPWM(n_servo, 0, duty);
}
int poserv1=0;
int poserv2=0;
int poserv3=0;
int pos,pos2,pos3;
int relei;
String cad,Sserv1,Sserv2,Sserv3,reles;

void loop() {
  if(Serial.available()){
    cad = Serial.readString();
```

```
    pos = cad.indexOf(',');
    pos2 = cad.indexOf(',',pos+1);
    pos3 = cad.indexOf(',',pos2+1);
    Sserv1= cad.substring(0,pos);
    Sserv2= cad.substring(pos+1,pos2);
    Sserv3= cad.substring(pos2+1,pos3);
    reles= cad.substring(pos3+1);

    if(poserv1 != Sserv1.toInt()){
      poserv1 = Sserv1.toInt();
      setServo(0,poserv1);
      }

    if(poserv2 != Sserv2.toInt()){
      poserv2 = Sserv2.toInt();
      setServo(1,poserv2);
      }

    if(poserv3 != Sserv3.toInt()){
      poserv3 = Sserv3.toInt();
      setServo(2,poserv3);
      }

    if(relei != reles.toInt()){
      relei = reles.toInt();
      digitalWrite(RELE, relei);
      }

    }

}
```

Programa de Python

```
# Universidad Autónoma Chapingo
# Departamento de Ingeniería Mecánica Agrícola
# Ingeniería Mecatrónica Agrícola
# Jym Emmanuel Cocotle Lara
# 7° 7
# ------Proyecto Final------
# ------Robot RR-----------------

# Librerías utilizadas
import tkinter as tk
import matplotlib.pyplot as plt
import math
import numpy as np
import serial, time
import cv2
from mpl_toolkits import mplot3d
from matplotlib import cm
from matplotlib.widgets import Slider
from tkinter import *
from tkinter import Tk, Label, Button, Frame,  messagebox, filedialog,
ttk
from tkinter import scrolledtext as st

robot_ser = serial.Serial('COM5',baudrate=9600)
```

```
# FUNCIONES DE MATRICES DE TRASLACIÓN
# X
def matriz_traslacion_x(x): # Definimos la matriz de traslación en x
        traslacion_x = np.array([[1,0,0,x], # Primera fila de la matriz
de traslación
                                 [0,1,0,0], # Segunda
fila de la matriz de traslación
                                 [0,0,1,0], # Tercera
fila de la matriz de traslación
                                 [0,0,0,1]]) # Cuarta
fila de la matriz de traslación
        return traslacion_x # Devolvemos la matriz de traslación en x

# Y
def matriz_traslacion_y(y):  # Definimos la matriz de traslación en y
        traslacion_y = np.array([[1,0,0,0], # Primera fila de la matriz
de traslación
                                 [0,1,0,y], # Segunda
fila de la matriz de traslación
                                 [0,0,1,0], # Tercera
fila de la matriz de traslación
                                 [0,0,0,1]]) # Cuarta
fila de la matriz de traslación
        return traslacion_y # Devolvemos la matriz de traslación en y

# Z
def matriz_traslacion_z(z):  # Definimos la matriz de traslación en z
        traslacion_z = np.array([[1,0,0,0], # Primera fila de la matriz
de traslación
                                 [0,1,0,0], # Segunda
fila de la matriz de traslación
                                 [0,0,1,z], # Tercera
fila de la matriz de traslación
                                 [0,0,0,1]]) # Cuarta
fila de la matriz de traslación
        return traslacion_z # Devolvemos la matriz de traslación en z

# FUNCIONES DE MATRICES DE ROTACIÓN
# X
def matriz_rotacion_x(grados_x): #Definimos la función de rotación en
x
        rad = grados_x/180*np.pi # Conversión a grados
        matriz_rotacion_x = np.array([[1, 0, 0,0], #Matriz de rotación
primera fila
                                      [0,  np.cos(rad), -
np.sin(rad),0], #Matriz de rotación segunda fila

[0,np.sin(rad),np.cos(rad),0], #Matriz de rotación terca fila
                                      [0,0,0,1]]) # Cuarta
fila de la matriz

        return matriz_rotacion_x #Devuelvo la matriz de rotación en x

# Y
def matriz_rotacion_y(grados_y): #Definimos la función de rotación en
y
        rad = grados_y/180*np.pi # Conversión a grados
```

```
        matriz_rotacion_y = np.array([[np.cos(rad), 0, -np.sin(rad),0], #Matriz de rotación primera fila
                                      [0,  1, 0,0], #Matriz de rotación segunda fila
                                      [np.sin(rad), 0, np.cos(rad),0], #Matriz de rotación terca fila
                                      [0,0,0,1]]) # Cuarta fila de la matriz

        return matriz_rotacion_y #Devuelvo la matriz de rotación en y

# Z
def matriz_rotacion_z(grados_z): #Definimos la función de rotación en z
        rad = grados_z/180*np.pi # Conversión a grados
        rotacion_z = np.array([[np.cos(rad),-np.sin(rad),0,0],  #Matriz de rotación primera fila

        [np.sin(rad),np.cos(rad),0,0], #Matriz de rotación segunda fila

        [0,0,1,0], #Matriz de rotación terca fila

        [0,0,0,1]]) # Cuarta fila de la matriz
        return rotacion_z # devolvemos la matriz de rotación en

# Cinemática inversa
def cinematica_inversa(x,y,a1,a2):
        # Le damos posición y obtenemos ángulo
    theta_2 = math.acos((x**2+y**2-a1**2-a2**2)/(2*a1*a2))
    theta_1 = math.atan2(y,x)-math.atan2((a2*math.sin(theta_2)),(a1+a2*math.cos(theta_2)))

    theta_1 = round((theta_1*180/np.pi),1)
    theta_2 = round((theta_2*180/np.pi),1)
    theta_2 = theta_2
    print('\u03B8\u2081: ',theta_1)
    print('\u03B8\u2082: ',theta_2)
    print()
    return theta_1,theta_2

# Configutación de la gráfica
def configuracion_grafica(): # función de configuración de la gráfica
        plt.title("Robot 2 G.D.L. RR",x=3, y=12) # Título de la gráfica
        ax.set_xlim(-17,17) # Límites en el eje x
        ax.set_ylim(0,17) # Límites en el eje y
        ax.set_zlim(-0,5) # Límites en el eje z

        ax.set_xlabel("x(t)") # Nombre del eje x
        ax.set_ylabel("y(t)") # Nombre del eje y
        ax.set_zlabel("z(t)") # Nombre del eje z
        ax.view_init(elev = 83, azim = -90) # # Tipo de vista de la gráfica

def sistema_coordenadas(a,b,c,a_f,b_f,c_f):
        x = [a,a_f]
        y = [b,b_f]
        z = [c,c_f]
```

```
        ax.plot3D(x,[b,b],[c,c],color="red") # X
        ax.plot3D([a,a],y,[c,c],color="blue") # Y
        ax.plot3D([a,a],[b,b],z,color="green") # Z

# Sistema de coordenadas móvil para la matriz de rotación
def sistema_coordenadas_movil(matriz_rotacion): # definimos la matriz
        r_11 = matriz_rotacion[0,0] # Columna 0, Fila 0
        r_12 = matriz_rotacion[1,0] # Columna 1, Fila 0
        r_13 = matriz_rotacion[2,0] # Columna 2, Fila 0
        r_21 = matriz_rotacion[0,1] # Columna 0, Fila 1
        r_22 = matriz_rotacion[1,1] # Columna 1, Fila 1
        r_23 = matriz_rotacion[2,1] # Columna 2, Fila 1
        r_31 = matriz_rotacion[0,2] # Columna 0, Fila 2
        r_32 = matriz_rotacion[1,2] # Columna 1, Fila 2
        r_33 = matriz_rotacion[2,2] # Columna 2, Fila 2

        dx = matriz_rotacion[0,3] # Columna 0, Fila 3
        dy = matriz_rotacion[1,3] # Columna 1, Fila 3
        dz = matriz_rotacion[2,3] # Columna 2, Fila 3

        ax.plot3D([dx,dx+r_11],[dy,dy+r_12],[dz,dz+r_13], color="m") #
X
        ax.plot3D([dx,dx+r_21],[dy,dy+r_22],[dz,dz+r_23], color="c") #
Y
        ax.plot3D([dx,dx+r_31],[dy,dy+r_32],[dz,dz+r_33], color="k") #
Z

def denavit_hatemberg(theta_i,d_i,a_i,alpha_i):
        MT =
matriz_rotacion_z(theta_i)@matriz_traslacion_z(d_i)@matriz_traslacion_
x(a_i)@matriz_rotacion_x(alpha_i)
        return MT

def robot_RR(theta_1,d1,a1,alpha_1,theta_2,d2,a2,alpha_2):
        A0 = np.eye(4) # Matriz identidad de 4x4
        A_0_1 = denavit_hatemberg(theta_1,d1,a1,alpha_1)
        A_1_2 = denavit_hatemberg(theta_2,d2,a2,alpha_2)
        A_0_2 = A_0_1 @ A_1_2

        sistema_coordenadas_movil(A0) # sistema móvil de la base
        sistema_coordenadas_movil(A_0_1) # sistema móvil del eslabón 1
        sistema_coordenadas_movil(A_0_2) # sistema móvil del eslabón 2

        ax.plot3D([A0[0,3],A_0_1[0,3]],[A0[1,3],A_0_1[1,3]],[A0[2,3],A_
0_1[2,3]], color="blue") # Eslabón 1
        ax.plot3D([A_0_1[0,3],A_0_2[0,3]],[A_0_1[1,3],A_0_2[1,3]],[A_0_
1[2,3],A_0_2[2,3]], color="green") # Eslabón 2
        return A_0_2


def actualizacion_juntas(val):

        ax.cla() # Limpia la gráfica
        configuracion_grafica()
        theta_1 = sld_angulo_1.val
        theta_2 = sld_angulo_2.val
```

```
        Matriz_TH = robot_RR(theta_1,0,10,0,theta_2-90,0,8.5,0)

def actualizacion_juntas2(val):

    ax.cla() # Limpia la gráfica
    configuracion_grafica()
    x = sld_angulo_1.val
    y = sld_angulo_2.val
    theta_1,theta_2 = cinematica_inversa(x,y,10,8.5)
    Matriz_TH = robot_RR(theta_1,0,10,0,theta_2,0,8.5,0)
    sld_angulo_2.eventson = False
    sld_angulo_2.set_val(Matriz_TH[1,3])
    sld_angulo_2.eventson = True

def actualizacion_juntas3(val):

    ax.cla() # Limpia la gráfica
    configuracion_grafica()
    x = sld_angulo_1.val
    y = sld_angulo_2.val
    theta_1,theta_2 = cinematica_inversa(x,y,10,8.5)
    Matriz_TH = robot_RR(theta_1,0,10,0,theta_2,0,8.5,0)

    sld_angulo_1.eventson = False
    sld_angulo_1.set_val(Matriz_TH[0,3])
    sld_angulo_1.eventson = TRUE

ventana = Tk()
ventana.minsize(width=950, height=700)
ventana.title('Dinamica de robots')

# Funcion para la deteccion del color rojo
def vision_r():
    global x_r_3,y_r_3,cad,x_r_4,act_r
    video_r = cv2.VideoCapture(1) # Captura el video de la cámara del
celular
    azul_bajo = np.array([0,104,58]) # RGB del color azul con tono
bajo
    azul_alto = np.array([34,255,111]) # RGB del color azul con tono
alto

    #15,130,255

    while True:
        # Capturamos el video
        ret,frame = video_r.read()
        # Si el video se captura correctamente, entonces
        if ret:
            # Convertimos lo capturado a HSV (Hue, Saturation, Value)
            frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
            # Aplicamos una máscara pasando los valores bajos y altos
del color
            mascara = cv2.inRange(frameHSV,azul_bajo,azul_alto)
                      # Encontramos los contornos del objeto
            contornos, __ =
cv2.findContours(mascara,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
            # Dibujamos los contornos
```

```
            cv2.drawContours(frame,contornos,-1,(255,0,0),3)


            for c in contornos:
                area = cv2.contourArea(c)
                if area > 600: # área del objeto
                    # Encontramos los momentos del objeto
                    M = cv2.moments(c)
                    if M["m00"] == 0:
                        M["m00"] = 1 # Igualamos a 1 ya que si
dividimos entre 0 sería una indefinición

                    # Obtenemos coordenadas del objeto
                    x = int(M["m10"]/M["m00"])
                    y = int(M["m01"]/M["m00"])
                    x_2 = x/20
                    y_2 = y/20

                    y_r_3 = x_2+8
                    x_r_3 = y_2-10

                    x_r_4 = x_r_3
                    y_r_4 = y_r_3
                    z_r_4 = 2

                    th_r_1,th_r_2 =
cinematica_inversa(x_r_4,y_r_4,10,8.5)
                    th_r_2_1 = th_r_2+90
                    th_r_3 = 0

                    actr="0"

                    cad =
str(th_r_1)+","+str(th_r_2_1)+","+str(th_r_3)+","+str(actr)
                    robot_ser.write(cad.encode('ascii'))

                    print(f"X: {x_r_3}")
                    print(f"Y: {y_r_3}")
                    # Dibujamos el círculo del centro del objeto y
escribimos las coordenadas de este
                    cv2.circle(frame,(x,y),7,(0,0,255),-1) # Rellena
el círculo
                    # Escribimos las coordenadas en pantalla

cv2.putText(frame,"{},{}".format(x_r_3,y_r_3),(x+10,y),cv2.FONT_HERSHE
Y_SIMPLEX,1.2,(0,0,255),2,cv2.LINE_AA)
                    nuevo_contorno = cv2.convexHull(c) # cierra los
objetos para completar una figura

cv2.drawContours(frame,[nuevo_contorno],0,(255,0,0),3) # Dibuja los
nuevos contornos

                    time.sleep(2)
                    cad = str(th_r_1)+","+str(th_r_2_1)+",180"+",1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = str(th_r_1)+","+str(th_r_2_1)+",0"+",1"
```

```
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "150,160,0,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "150,160,180,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "150,160,180,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "150,160,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "53,172,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)

                    indica_x['text']=x_r_4
                    indica_y['text']=y_r_4
                    indica_z['text']=z_r_4

                    indica_t_1['text']=th_r_1
                    indica_t_2['text']=th_r_2
                    indica_t_3['text']=th_r_3


            cv2.imshow("Video",frame) # Mostramos el video
            if cv2.waitKey(1) & 0xFF == 27: # si presiona escape se
sale del programa
                break

# Funcion para la deteccion del color verde
def vision_g():
    global x_g_3,y_g_3,cad,actg
    video_g = cv2.VideoCapture(1) # Captura el video de la cámara del
celular
    azul_bajo = np.array([22,130,0]) # RGB del color azul con tono
bajo
    azul_alto = np.array([110,255,255]) # RGB del color azul con tono
alto

    while True:
        # Capturamos el video
        ret,frame = video_g.read()
        # Si el video se captura correctamente, entonces
        if ret:
            # Convertimos lo capturado a HSV (Hue, Saturation, Value)
            frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
            # Aplicamos una máscara pasando los valores bajos y altos
del color
            mascara = cv2.inRange(frameHSV,azul_bajo,azul_alto)
                      # Encontramos los contornos del objeto
            contornos, __ =
cv2.findContours(mascara,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
            # Dibujamos los contornos
            cv2.drawContours(frame,contornos,-1,(0,255,0),3)
```

```
            for c in contornos:
                area = cv2.contourArea(c)
                if area > 600: # área del objeto
                    # Encontramos los momentos del objeto
                    M = cv2.moments(c)
                    if M["m00"] == 0:
                        M["m00"] = 1 # Igualamos a 1 ya que si
dividimos entre 0 sería una indefinición

                    # Obtenemos coordenadas del objeto
                    x = int(M["m10"]/M["m00"])
                    y = int(M["m01"]/M["m00"])
                    x_g_2 = x/20
                    y_g_2 = y/20

                    y_g_3 = x_g_2+8
                    x_g_3 = y_g_2-10

                    x_g_4 = x_g_3
                    y_g_4 = y_g_3
                    z_g_4 = 2

                    th_g_1,th_g_2 =
cinematica_inversa(x_g_4,y_g_4,10,8.5)
                    th_g_2_1 = th_g_2+90
                    th_g_3 = 0
                    actg="0"

                    cad =
str(th_g_1)+","+str(th_g_2_1)+","+str(th_g_3)+","+str(actg)
                    robot_ser.write(cad.encode('ascii'))

                    print(f"X: {x_g_3}")
                    print(f"Y: {y_g_3}")
                    # Dibujamos el círculo del centro del objeto y
escribimos las coordenadas de este
                    cv2.circle(frame,(x,y),7,(0,0,255),-1) # Rellena
el círculo
                    # Escribimos las coordenadas en pantalla

cv2.putText(frame,"{},{}".format(x_g_3,y_g_3),(x+10,y),cv2.FONT_HERSHE
Y_SIMPLEX,1.2,(0,0,255),2,cv2.LINE_AA)
                    nuevo_contorno = cv2.convexHull(c) # cierra los
objetos para completar una figura

cv2.drawContours(frame,[nuevo_contorno],0,(255,0,0),3) # Dibuja los
nuevos contornos

                    time.sleep(2)
                    cad = str(th_g_1)+","+str(th_g_2_1)+",180"+",1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = str(th_g_1)+","+str(th_g_2_1)+",0"+",1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
```

```
                    cad = "130,150,0,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "130,150,180,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "130,150,180,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "130,150,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "53,172,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)

                    indica_x['text']=x_g_4
                    indica_y['text']=y_g_4
                    indica_z['text']=z_g_4

                    indica_t_1['text']=th_g_1
                    indica_t_2['text']=th_g_2
                    indica_t_3['text']=th_g_3


            cv2.imshow("Video",frame) # Mostramos el video
            if cv2.waitKey(1) & 0xFF == 27: # si presiona escape se
sale del programa
                #robot_ser.close()
                break

# Funcion para la deteccion del color azul
def vision_b():
    global x_3,y_3,cad,x_4,y_4,z_4,actb
    video = cv2.VideoCapture(1) # Captura el video de la cámara del
celular
    azul_bajo = np.array([90,100,20]) # RGB del color azul con tono
bajo
    azul_alto = np.array([120,255,255]) # RGB del color azul con tono
alto

    while True:
        # Capturamos el video
        ret,frame = video.read()
        # Si el video se captura correctamente, entonces
        if ret:
            # Convertimos lo capturado a HSV (Hue, Saturation, Value)
            frameHSV = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
            # Aplicamos una máscara pasando los valores bajos y altos
del color
            mascara = cv2.inRange(frameHSV,azul_bajo,azul_alto)
                # Encontramos los contornos del objeto
            contornos, __ =
cv2.findContours(mascara,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
            # Dibujamos los contornos
            cv2.drawContours(frame,contornos,-1,(0,0,255),3)
```

```
            for c in contornos:
                area = cv2.contourArea(c)
                if area > 600: # área del objeto
                    # Encontramos los momentos del objeto
                    M = cv2.moments(c)
                    if M["m00"] == 0:
                        M["m00"] = 1 # Igualamos a 1 ya que si
dividimos entre 0 sería una indefinición

                    # Obtenemos coordenadas del objeto
                    x = int(M["m10"]/M["m00"])
                    y = int(M["m01"]/M["m00"])
                    x_2 = x/20
                    y_2 = y/20

                    y_3 = x_2+8
                    x_3 = y_2-10

                    x_4 = x_3
                    y_4 = y_3
                    z_4 = 2

                    th_1,th_2 = cinematica_inversa(x_4,y_4,10,8.5)
                    th_2_1 = th_2+90
                    th_3 = 0
                    actb="0"

                    cad =
str(th_1)+","+str(th_2_1)+","+str(th_3)+","+str(actb)
                    robot_ser.write(cad.encode('ascii'))

                    print(f"X: {x_3}")
                    print(f"Y: {y_3}")
                    # Dibujamos el círculo del centro del objeto y
escribimos las coordenadas de este
                    cv2.circle(frame,(x,y),7,(0,0,255),-1) # Rellena
el círculo
                    # Escribimos las coordenadas en pantalla

cv2.putText(frame,"{},{}".format(x_3,y_3),(x+10,y),cv2.FONT_HERSHEY_SI
MPLEX,1.2,(0,0,255),2,cv2.LINE_AA)
                    nuevo_contorno = cv2.convexHull(c) # cierra los
objetos para completar una figura

cv2.drawContours(frame,[nuevo_contorno],0,(255,0,0),3) # Dibuja los
nuevos contornos

                    time.sleep(2)
                    cad = str(th_1)+","+str(th_2_1)+",180"+",1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = str(th_1)+","+str(th_2_1)+",0"+",1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "10,90,0,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
```

```
                    cad = "10,90,180,1"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "10,90,180,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "10,90,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)
                    cad = "53,172,0,0"
                    robot_ser.write(cad.encode('ascii'))
                    time.sleep(2)

            indica_x['text']=x_4
            indica_y['text']=y_4
            indica_z['text']=z_4

            indica_t_1['text']=th_1
            indica_t_2['text']=th_2
            indica_t_3['text']=th_3

            cv2.imshow("Video",frame) # Mostramos el video
            if cv2.waitKey(1) & 0xFF == 27: # si presiona escape se
sale del programa
                #robot_ser.close()
                break

# Limpiar tablas de guardado
def limpiar():
    res_X.delete('1.0', tk.END)
    res_Y.delete('1.0', tk.END)
    res_Z.delete('1.0', tk.END)
    actuador.delete('1.0', tk.END)

def limpiar_2():
    res_the_1.delete('1.0', tk.END)
    res_the_2.delete('1.0', tk.END)
    res_the_3.delete('1.0', tk.END)
    actuador2.delete('1.0', tk.END)


# Operaciones con respecto a X,Y,Z
def oper():

    #plt.clf()
    global
X_3,Y_3,Z_3,sld_angulo_1,sld_angulo_2,fig,ax,thet_1,thet_2,thet_3,cad,
act

    fig, ax = plt.subplots() # Creamos una figura y sus ejes
    ax = plt.axes(projection = "3d") # Establecemos los ejes como 3
dimensiones


    X_3 = float(entrada1.get()) # X
    Y_3 = float(entrada2.get()) # Y
    Z_3 = float(entrada3.get()) # Z
```

```
    act = int(actue.get())

    ax1 = plt.axes([0.2,0.15,0.65,0.03])
    ax2 = plt.axes([0.2,0.1,0.65,0.03])
    #ax3 = plt.axes([0.63,0.3,0.07,0.03])

    sld_angulo_1 = Slider(ax1,r"$X$",-10,10,valinit=X_3)
    sld_angulo_2 = Slider(ax2,r"$Y$",0,14,valinit=Y_3)

    thet_1,thet_2 = cinematica_inversa(X_3,Y_3,10,8.5)
    configuracion_grafica()
    Matriz_TH = robot_RR(thet_1,0,10,0,thet_2,0,8.5,0)

    thet_3 = Z_3*60

    thet_1 = thet_1
    thet_2 = thet_2+90
    thet_3 = thet_3

    sld_angulo_1.on_changed(actualizacion_juntas2)
    sld_angulo_2.on_changed(actualizacion_juntas3)

    plt.show()

    cad = str(thet_1)+","+str(thet_2)+","+str(thet_3)+","+str(act)
    robot_ser.write(cad.encode('ascii'))

    return X_3,Y_3,Z_3,sld_angulo_1,sld_angulo_2,fig,ax,thet_1,thet_2

# Operaciones con base a theta
def oper_2():

    global
sld_angulo_1,sld_angulo_2,fig,ax,theta_1_2,theta_2_2,theta_3_2,cad,act
_2

    fig, ax = plt.subplots() # Creamos una figura y sus ejes
    plt.subplots_adjust(left = 0, bottom = 0.3, right =0.74, top = 1)
# Ubicamos la figura
    ax = plt.axes(projection = "3d") # Establecemos los ejes como 3
dimensiones

    theta_1_2 = float(entrada4.get()) # X
    theta_2_2 = float(entrada5.get()) # Y
    theta_3_2 = float(entrada6.get()) # Z
    act_2 = int(actue2.get())

    ax1 = plt.axes([0.2,0.15,0.65,0.03])
    ax2 = plt.axes([0.2,0.1,0.65,0.03])
    #ax3 = plt.axes([0.63,0.3,0.07,0.03])

    sld_angulo_1 = Slider(ax1,r"$Theta 1$",0,180,valinit=theta_1_2)
    sld_angulo_2 = Slider(ax2,r"$Theta 2$",0,180,valinit=theta_2_2)

    configuracion_grafica()
    Matriz_TH = robot_RR(theta_1_2,0,10,0,theta_2_2-90,0,8.5,0)
```

```
    sld_angulo_1.on_changed(actualizacion_juntas)
    sld_angulo_2.on_changed(actualizacion_juntas)

    cad =
str(theta_1_2)+","+str(theta_2_2)+","+str(theta_3_2)+","+str(act_2)
    robot_ser.write(cad.encode('ascii'))

    plt.show()

    return
sld_angulo_1,sld_angulo_2,fig,ax,theta_1_2,theta_2_2,theta_3_2,act_2

# Graficacion de los valores mostrados en los cuadros de texto de la
ventana 1
def B1():
    global res_X,res_Y,res_Z,actuador,resu_X,resu_Y,resu_Z,resu_actu

    X_3 = float(entrada1.get()) # X
    Y_3 = float(entrada2.get()) # Y
    Z_3 = float(entrada3.get()) # Z
    actua = int(actue.get()) # Z

    res_X.insert(tk.INSERT,f"{X_3:.2f}\n")
    res_Y.insert(tk.INSERT,f"{Y_3:.2f}\n")
    res_Z.insert(tk.INSERT,f"{Z_3:.2f}\n")
    actuador.insert(tk.INSERT,f"{actua}\n")

    re_X = res_X.get('1.0', 'end-1c')
    re_X = re_X.split()
    re_X = list(map(float, re_X))
    resu_X = np.array(re_X)

    re_Y = res_Y.get('1.0', 'end-1c')
    re_Y = re_Y.split()
    re_Y = list(map(float, re_Y))
    resu_Y = np.array(re_Y)

    re_Z = res_Z.get('1.0', 'end-1c')
    re_Z = re_Z.split()
    re_Z = list(map(float, re_Z))
    resu_Z = np.array(re_Z)

    ac_1 = actuador.get('1.0', 'end-1c')
    ac_1 = ac_1.split()
    ac_1 = list(map(int, ac_1))
    resu_actu = np.array(ac_1)

    return res_X,res_Y,res_Z,actuador,resu_X,resu_Y,resu_Z,resu_actu

# Graficacion de los valores mostrados en los cuadros de texto de la
ventana 2
def B2():
    global
res_the_1,res_the_2,res_the_3,actuador2,resu_the_1,resu_the_2,resu_the
_3,resu_actu2

    the_1_1 = float(entrada4.get()) # X
```

```
    the_2_1 = float(entrada5.get()) # Y
    the_3_1 = float(entrada6.get()) # Z
    actua2 = int(actue2.get()) # Z

    res_the_1.insert(tk.INSERT,f"{the_1_1:.2f}\n")
    res_the_2.insert(tk.INSERT,f"{the_2_1:.2f}\n")
    res_the_3.insert(tk.INSERT,f"{the_3_1:.2f}\n")
    actuador2.insert(tk.INSERT,f"{actua2}\n")

    re_the_1 = res_the_1.get('1.0', 'end-1c')
    re_the_1 = re_the_1.split()
    re_the_1 = list(map(float, re_the_1))
    resu_the_1 = np.array(re_the_1)

    re_the_2 = res_the_2.get('1.0', 'end-1c')
    re_the_2 = re_the_2.split()
    re_the_2 = list(map(float, re_the_2))
    resu_the_2 = np.array(re_the_2)

    re_the_3 = res_the_3.get('1.0', 'end-1c')
    re_the_3 = re_the_3.split()
    re_the_3 = list(map(float, re_the_3))
    resu_the_3 = np.array(re_the_3)

    ac_2 = actuador.get('1.0', 'end-1c')
    ac_2 = ac_2.split()
    ac_2 = list(map(int, ac_2))
    resu_actu2 = np.array(ac_2)

    return
res_the_1,res_the_2,res_the_3,actuador2,resu_the_1,resu_the_2,resu_the
_3,resu_actu2

# Graficacion del conjunto de valores mostrados en los cuadros de
texto de la ventana 1
def B3():

    global
val_x,val_y,val_z,val_act,sld_angulo_1,sld_angulo_2,fig,ax,thet_1,thet
_2,thet_2_1,cad

    total = len(resu_X)
    i=0
    for i in range (total):
        val_x = resu_X[i]
        val_y = resu_Y[i]
        val_z = resu_Z[i]
        val_act = resu_actu[i]

        fig, ax = plt.subplots() # Creamos una figura y sus ejes
        ax = plt.axes(projection = "3d") # Establecemos los ejes como
3 dimensiones

        ax1 = plt.axes([0.2,0.15,0.65,0.03])
        ax2 = plt.axes([0.2,0.1,0.65,0.03])
        #ax3 = plt.axes([0.63,0.3,0.07,0.03])
```

```
        sld_angulo_1 = Slider(ax1,r"$X$",-10,10,valinit=val_x)
        sld_angulo_2 = Slider(ax2,r"$Y$",0,14,valinit=val_y)

        thet_1,thet_2 = cinematica_inversa(val_x,val_y,10,8.5)
        thet_2_1 = thet_2+90

        thet_3 = val_z*60
        configuracion_grafica()
        Matriz_TH = robot_RR(thet_1,0,10,0,thet_2,0,8.5,0)

        sld_angulo_1.on_changed(actualizacion_juntas2)
        sld_angulo_2.on_changed(actualizacion_juntas3)


        plt.show()

        cad =
str(thet_1)+","+str(thet_2_1)+","+str(thet_3)+","+str(val_act)
        robot_ser.write(cad.encode('ascii'))

        plt.pause(3)

# Graficacion del conjunto de valores mostrados en los cuadros de
texto de la ventana 1
def B4():

    global
val_t_1,val_t_2,val_t_3,val_act2,sld_angulo_1,sld_angulo_2,fig,ax,cad

    total = len(resu_the_1)
    i=0
    for i in range (total):
        val_t_1 = resu_the_1[i]
        val_t_2 = resu_the_2[i]
        val_t_3 = resu_the_3[i]
        val_act2 = resu_actu2[i]

        fig, ax = plt.subplots() # Creamos una figura y sus ejes
        ax = plt.axes(projection = "3d") # Establecemos los ejes como
3 dimensiones


        ax1 = plt.axes([0.2,0.15,0.65,0.03])
        ax2 = plt.axes([0.2,0.1,0.65,0.03])

        sld_angulo_1 = Slider(ax1,r"$Theta 1$",0,180,valinit=val_t_1)

        sld_angulo_2 = Slider(ax2,r"$Theta 2$",0,180,valinit=val_t_2)

        configuracion_grafica()
        Matriz_TH = robot_RR(val_t_1,0,10,0,val_t_2-90,0,8.5,0)

        sld_angulo_1.on_changed(actualizacion_juntas)
        sld_angulo_2.on_changed(actualizacion_juntas)

        cad =
str(val_t_1)+","+str(val_t_2)+","+str(val_t_3)+","+str(val_act2)
```

```
        robot_ser.write(cad.encode('ascii'))

        plt.show()
        plt.pause(3)

def salir():
    ventana.quit()
    ventana.destroy()
    robot_ser.close()
s = ttk.Style()
s.configure('TFrame', background='#6DBFD3')

# Diseño de las ventanas
tab_control = ttk.Notebook(ventana)

tab1 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Cinematica inversa')
tab_control.pack(expand=1, fill='both')

tab2 = ttk.Frame(tab_control)
tab_control.add(tab2, text='Cinematica directa')
tab_control.pack(expand=1, fill='both')


tab3 = ttk.Frame(tab_control)
tab_control.add(tab3, text='Coordenadas por vision')
tab_control.pack(expand=1, fill='both')

# Almacenamiento valores de X,Y,Z,actuador
res_X = st.ScrolledText(tab1,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_X.place(x = 20,y = 300)

res_Y = st.ScrolledText(tab1,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_Y.place(x = 250,y = 300)

res_Z = st.ScrolledText(tab1,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_Z.place(x = 480,y = 300)

actuador = st.ScrolledText(tab1,
wrap = tk.WORD,
width = 15,
height = 10,
```

```
font = ("Sylfaen",
15))
actuador.place(x = 710,y = 300)

# Almacenamiento valores de theta
res_the_1 = st.ScrolledText(tab2,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_the_1.place(x = 20,y = 300)

res_the_2 = st.ScrolledText(tab2,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_the_2.place(x = 250,y = 300)

res_the_3 = st.ScrolledText(tab2,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
res_the_3.place(x = 480,y = 300)

actuador2 = st.ScrolledText(tab2,
wrap = tk.WORD,
width = 15,
height = 10,
font = ("Sylfaen",
15))
actuador2.place(x = 710,y = 300)

# Labels y botones ventana 1
e1=tk.Label(tab1,text="Coordenada en X: ",bg="#2DE0AE",fg="#2F3257")
e1.place(x = 10,y = 5,width=200, height=40)
entrada1=tk.Entry(tab1)
entrada1.place(x = 220,y = 5,width=200, height=40)

e2=tk.Label(tab1,text="Coordenada en Y: ",bg="#2DE0AE",fg="#2F3257")
e2.place(x = 10,y = 60,width=200, height=40)
entrada2=tk.Entry(tab1)
entrada2.place(x = 220,y = 60,width=200, height=40)

e3=tk.Label(tab1,text="Coordenada en Z: ",bg="#2DE0AE",fg="#2F3257")
e3.place(x = 10,y = 115,width=200, height=40)
entrada3=tk.Entry(tab1)
entrada3.place(x = 220,y = 115,width=200, height=40)

actu=tk.Label(tab1,text="Actuador ",bg="#2DE0AE",fg="#2F3257")
actu.place(x = 460,y = 20,width=200, height=40)
actue=tk.Entry(tab1)
actue.place(x = 460,y = 70,width=200, height=40)
```

```
e11=tk.Label(tab1,text="---X---",bg="#2058EC",fg="#FFFFFF")
e11.place(x = 30,y = 250,width=200, height=40)

e12=tk.Label(tab1,text="---Y---",bg="#2058EC",fg="#FFFFFF")
e12.place(x = 265,y = 250,width=200, height=40)

e13=tk.Label(tab1,text="---Z---",bg="#2058EC",fg="#FFFFFF")
e13.place(x = 490,y = 250,width=200, height=40)

e14=tk.Label(tab1,text="---Actuador---",bg="#2058EC",fg="#FFFFFF")
e14.place(x = 715,y = 250,width=200, height=40)

calc=tk.Button(tab1,text="Calcular",fg="#2F3257",command=oper)
calc.place(x = 10,y = 165,width=200, height=40)

guardar=tk.Button(tab1,text="Guardar",fg="#2F3257",command=B1)
guardar.place(x = 220,y = 165,width=200, height=40)

secuencia=tk.Button(tab1,text="Secuencia",fg="#2F3257",command=B3,widt
h=10, height=1)
secuencia.place(x = 750,y = 50)

limpiar=tk.Button(tab1,text="Limpiar
tablas",fg="#2F3257",command=limpiar,width=15, height=1)
limpiar.place(x = 750,y = 100)

salir1=tk.Button(tab1,text="Salir",fg="#2F3257",bg =
'#EC5757',command=salir,width=10, height=1)
salir1.place(x = 750,y = 150)

# Labels y botones ventana 2
e4=tk.Label(tab2,text="Theta 1: ",bg="#2DE0AE",fg="#2F3257")
e4.place(x = 10,y = 5,width=200, height=40)
entrada4=tk.Entry(tab2)
entrada4.place(x = 220,y = 5,width=200, height=40)

e5=tk.Label(tab2,text="Theta 2: ",bg="#2DE0AE",fg="#2F3257")
e5.place(x = 10,y = 60,width=200, height=40)
entrada5=tk.Entry(tab2)
entrada5.place(x = 220,y = 60,width=200, height=40)

e6=tk.Label(tab2,text="Theta 3: ",bg="#2DE0AE",fg="#2F3257")
e6.place(x = 10,y = 115,width=200, height=40)
entrada6=tk.Entry(tab2)
entrada6.place(x = 220,y = 115,width=200, height=40)

actu2=tk.Label(tab2,text="Actuador ",bg="#2DE0AE",fg="#2F3257")
actu2.place(x = 460,y = 20,width=200, height=40)
actue2=tk.Entry(tab2)
actue2.place(x = 460,y = 70,width=200, height=40)

e21=tk.Label(tab2,text="---\u03B8\u2081---",bg="#2058EC",fg="#FFFFFF")
e21.place(x = 30,y = 250,width=200, height=40)

e22=tk.Label(tab2,text="---\u03B8\u2082--",bg="#2058EC",fg="#FFFFFF")
e22.place(x = 265,y = 250,width=200, height=40)
```

```
e23=tk.Label(tab2,text="---\u03B8\u2083---",bg="#2058EC",fg="#FFFFFF")
e23.place(x = 490,y = 250,width=200, height=40)

e24=tk.Label(tab2,text="---Actuador---",bg="#2058EC",fg="#FFFFFF")
e24.place(x = 715,y = 250,width=200, height=40)

calc_2=tk.Button(tab2,text="Calcular",fg="#2F3257",command=oper_2)
calc_2.place(x = 10,y = 165,width=200, height=40)

guardar_2=tk.Button(tab2,text="Guardar",fg="#2F3257",command=B2)
guardar_2.place(x = 220,y = 165,width=200, height=40)

secuencia_2=tk.Button(tab2,text="Secuencia",fg="#2F3257",command=B4,wi
dth=10, height=1)
secuencia_2.place(x = 750,y = 50)

limpiar_2=tk.Button(tab2,text="Limpiar
tablas",fg="#2F3257",command=limpiar_2,width=15, height=1)
limpiar_2.place(x = 750,y = 100)

salir2=tk.Button(tab2,text="Salir",fg="#2F3257",bg =
'#EC5757',command=salir,width=10, height=1)
salir2.place(x = 750,y = 150)

# Labels ventana 3
indica_xl = Label(tab3, fg= 'white', bg='gray26', text= 'Coordenada en
el eje X', font= ('Sylfaen',10,'bold') )
indica_xl.place(x = 20,y = 15,width=200, height=40)
indica_x = Label(tab3, fg= 'white', bg='gray26', text= '-----', font=
('Sylfaen',10,'bold') )
indica_x.place(x = 300,y = 15,width=200, height=40)

indica_yl = Label(tab3, fg= 'white', bg='gray26', text= 'Coordenada en
el eje Y', font= ('Sylfaen',10,'bold') )
indica_yl.place(x = 20,y = 65,width=200, height=40)
indica_y = Label(tab3, fg= 'white', bg='gray26', text= '-----', font=
('Sylfaen',10,'bold') )
indica_y.place(x = 300,y = 65,width=200, height=40)

indica_zl = Label(tab3, fg= 'white', bg='gray26', text= 'Coordenada en
el eje Z', font= ('Sylfaen',10,'bold') )
indica_zl.place(x = 20,y = 115,width=200, height=40)
indica_z = Label(tab3, fg= 'white', bg='gray26', text= '-----', font=
('Sylfaen',10,'bold') )
indica_z.place(x = 300,y = 115,width=200, height=40)

indica_t_1l = Label(tab3, fg= 'white', bg='gray26', text=
'\u03B8\u2081', font= ('Sylfaen',10,'bold') )
indica_t_1l.place(x = 20,y = 180,width=200, height=40)
indica_t_1 = Label(tab3, fg= 'white', bg='gray26', text= '-----',
font= ('Sylfaen',10,'bold') )
indica_t_1.place(x = 300,y = 180,width=200, height=40)

indica_t_2l = Label(tab3, fg= 'white', bg='gray26', text=
'\u03B8\u2082', font= ('Sylfaen',10,'bold') )
indica_t_2l.place(x = 20,y = 230,width=200, height=40)
```

```
indica_t_2 = Label(tab3, fg= 'white', bg='gray26', text= '-----',
font= ('Sylfaen',10,'bold') )
indica_t_2.place(x = 300,y = 230,width=200, height=40)

indica_t_3l = Label(tab3, fg= 'white', bg='gray26', text=
'\u03B8\u2083', font= ('Sylfaen',10,'bold') )
indica_t_3l.place(x = 20,y = 280,width=200, height=40)
indica_t_3 = Label(tab3, fg= 'white', bg='gray26', text= '-----',
font= ('Sylfaen',10,'bold') )
indica_t_3.place(x = 300,y = 280,width=200, height=40)

# Botones ventana 3
vision_r=tk.Button(tab3,text="Clasificacion
rojos",fg="#2F3257",command=vision_r,width=50, height=1)
vision_r.place(x = 50,y = 350)

vision_g=tk.Button(tab3,text="Clasificacion
verdes",fg="#2F3257",command=vision_g,width=50, height=1)
vision_g.place(x = 50,y = 400)

vision_b=tk.Button(tab3,text="Clasificacion
azules",fg="#2F3257",command=vision_b,width=50, height=1)
vision_b.place(x = 50,y = 450)

salir3=tk.Button(tab3,text="Salir",fg="#2F3257",bg =
'#EC5757',command=salir,width=10, height=1)
salir3.place(x = 800,y = 620)

ventana.mainloop()

robot_ser.close()
```

Resultados

Primera parte del programa

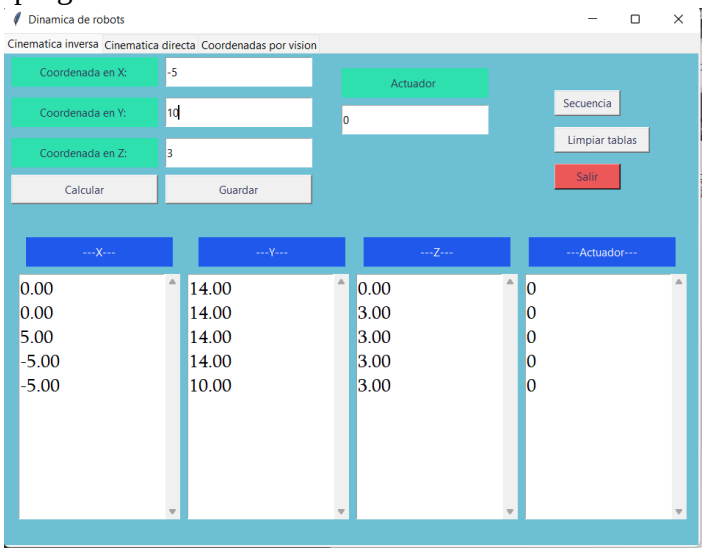

Fig 2.- Interfaz del programa, primera sección.

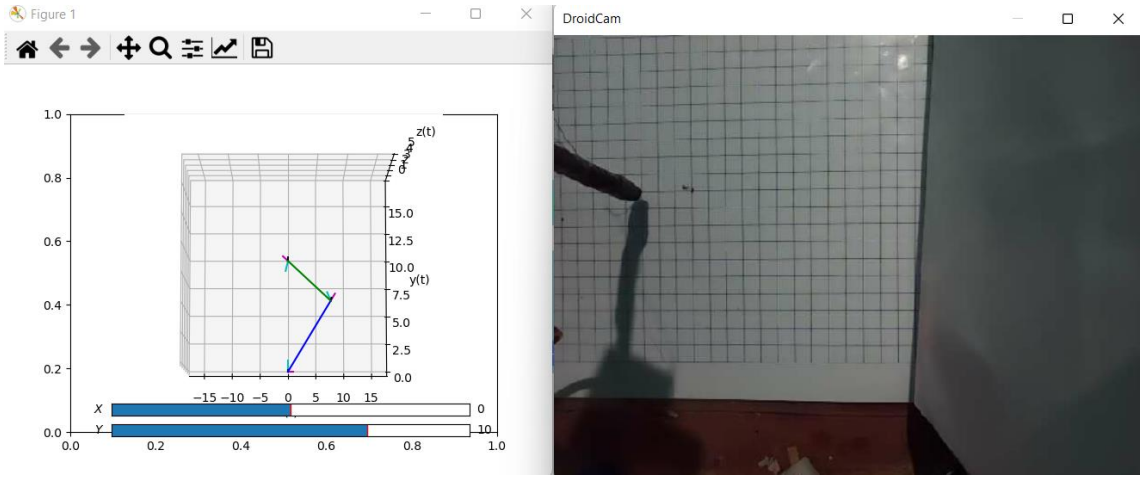

Fig 3.- Posición a partir de coordenadas.

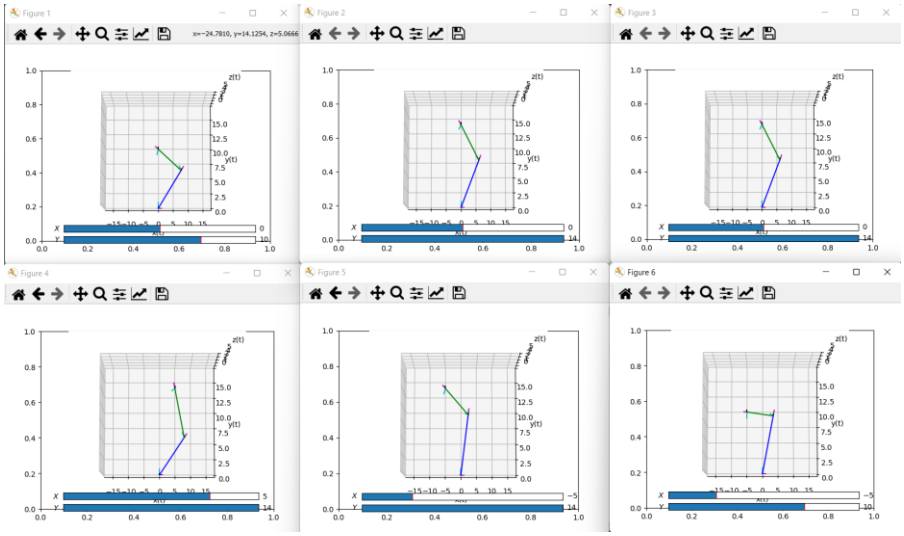

Fig 4.- Secuencia a partir de datos almacenados.

Segunda parte del programa

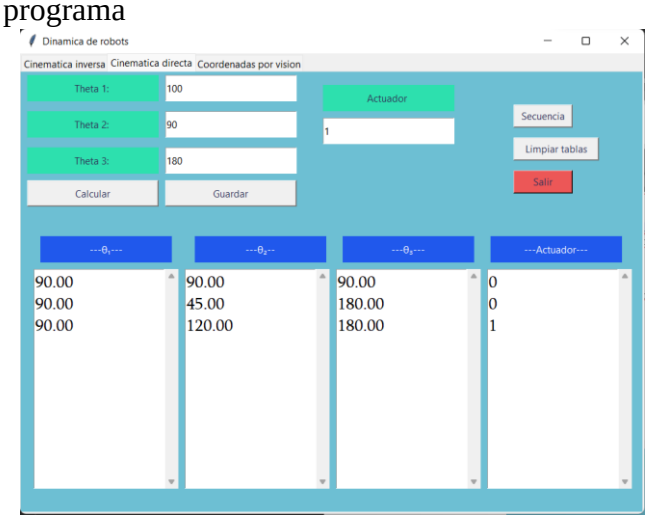

Fig 5.- Interfaz del programa, segunda sección.

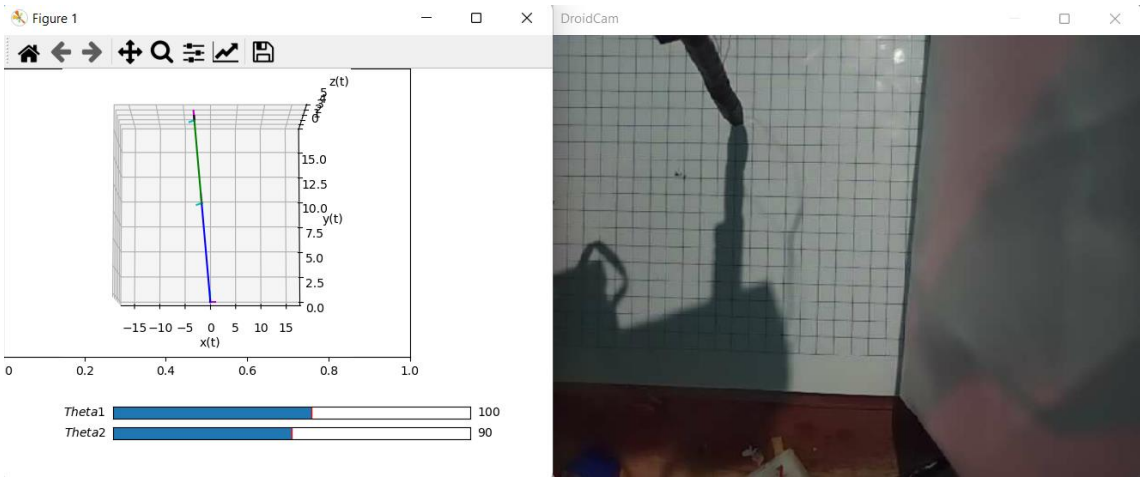

Fig 6.- Posición a partir de ángulos.

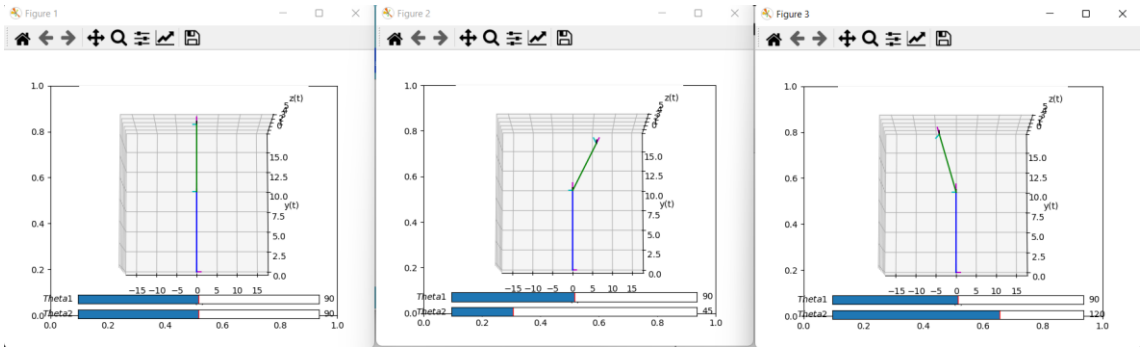

Fig 7.- Secuencia a partir de datos almacenados.

Tercera parte del programa

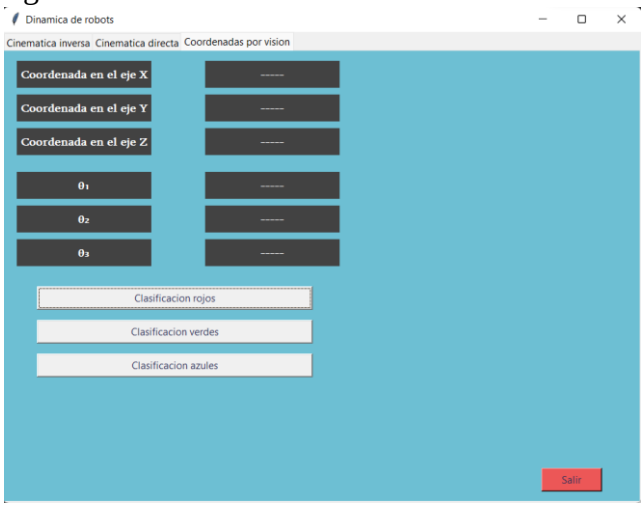

Fig 8.-Interfaz del programa, tercera sección.

Para poder observar mejor el funcionamiento de la tercera sección del programa es recomendable observar el video antes mencionado.

Conclusión

Con ayuda de la cinemática inversa y la visión artificial pude realizar un programa, el cual clasifica de manera eficiente objetos de diferentes colores (rojo, verde y azul), que para este prototipo los objetos tienen que ser metálicos debido al electroimán, sin embargo, el robot no solo puede ser operado por visión artificial, también se puede crear una secuencia de instrucciones a partir de coordenadas o ángulos y el robot lo ejecuta de manera eficiente.

La implementación de la visión artificial en los robots puede ayudar a mejorar la automatización, ya que, por medio de la obtención de las coordenadas por medio de los colores, se puede clasificar de mejor manera y no es necesario colocar los objetos en una posición en especifico para que el robot pueda agarrar el objeto y llevarlo a otro lugar.

Bibliografía

Sánchez, L. A., & Saavedra, M. S. (2005). Matemáticas y robótica. Curso Interuniversitario “Sociedad, Ciencia, Tecnología Y Matemáticas.