Forma, Flecha

Descripción generada automáticamente

Universidad Autónoma Chapingo

Departamento de Irrigación

****

**GRADO: GRUPO:**

**7° 7**

**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]**

**Fecha: 10-Sep-2019**

**Fecha de entrega: 29/12/2021**

Chapingo, Texcoco Edo. México

Índice

[Introducción 2](#_Toc91639152)

[Desarrollo 3](#_Toc91639153)

[Prototipo 3](#_Toc91639154)

[Materiales: 3](#_Toc91639155)

[Construcción 3](#_Toc91639156)

[Programas empleados 4](#_Toc91639157)

[Programa de Arduino 4](#_Toc91639158)

[Programa de Python 5](#_Toc91639159)

[Resultados 24](#_Toc91639160)

[Primera parte del programa 24](#_Toc91639161)

[Segunda parte del programa 25](#_Toc91639162)

[Tercera parte del programa 26](#_Toc91639163)

[Conclusión 27](#_Toc91639164)

[Bibliografía 27](#_Toc91639165)

# 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.

Imagen que contiene cuarto, computadora, tabla, vivo

Descripción generada automáticamente

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\_HERSHEY\_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\_HERSHEY\_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\_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\_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,width=**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,width=**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

Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 2.- Interfaz del programa, primera sección.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 3.- Posición a partir de coordenadas.

Interfaz de usuario gráfica, Gráfico, Aplicación

Descripción generada automáticamente

Fig 4.- Secuencia a partir de datos almacenados.

### Segunda parte del programa

Interfaz de usuario gráfica

Descripción generada automáticamente

Fig 5.- Interfaz del programa, segunda sección.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fig 6.- Posición a partir de ángulos.

Gráfico

Descripción generada automáticamente

Fig 7.- Secuencia a partir de datos almacenados.

### Tercera parte del programa

Interfaz de usuario gráfica

Descripción generada automáticamente

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.