PRÁCTICA #X

TEMA: Dibujo de Figuras Básicas con el Dobot Magician utilizando Python

1. Objetivos

- Familiarización con el control del Dobot Magician a través de Python.
- Programar el Dobot Magician para dibujar figuras geométricas básicas (cuadrado, triángulo, círculo) en una superficie plana.
- Comprender y optimizar el uso de posiciones y velocidades para lograr un trazo continuo y preciso.

2. Marco teórico

Definición

El **Dobot Magician** es un brazo robótico versátil capaz de realizar diversas tareas de manipulación y dibujo. Para programar el dibujo de figuras, se utilizan coordenadas cartesianas que determinan la posición del efector en el espacio de trabajo. Mediante la API de Python, se pueden definir trayectorias precisas para que el robot dibuje sobre una superficie usando un plumón u otra herramienta de escritura.

En el contexto de control robótico, los **movimientos lineales** y **circulares** permiten realizar figuras geométricas al mover el plumón en el eje XY, mientras se mantiene una altura constante en el eje Z para asegurar el contacto con la superficie. La velocidad de movimiento y el tiempo de espera entre pasos son parámetros clave para obtener trazos suaves y reducir el tiempo de ejecución.

3. Descripción

En esta práctica, se busca que el Dobot Magician dibuje figuras geométricas básicas (cuadrado, triángulo y círculo) sobre una hoja colocada en su área de trabajo. La posición de la hoja en el espacio de trabajo debe definirse previamente, y el plumón debe estar instalado en el efector del robot.

El control del dibujo se realiza mediante coordenadas cartesianas en Python, donde cada figura se representa mediante una serie de puntos en el plano XY. El robot ejecuta movimientos lineales entre estos puntos, con una altura fija en Z para mantener contacto con la hoja.

Configuración Inicial en Python

- Instala la biblioteca de control (pydobot) y conecta el Dobot Magician mediante el puerto COM correspondiente.

```python
import time
from pydobot import Dobot

# Conectar el Dobot al puerto COM
port = 'COM3'  # Cambia al puerto verificado en el administrador de dispositivos
device = Dobot(port)
```

Definición de Posiciones para el Dibujo

- Define las posiciones de inicio y de dibujo para cada figura geométrica básica en el plano XY. La altura se mantiene constante para asegurar el contacto del plumón con la hoja.

```python
# Definir posiciones iniciales y altura de dibujo
drawing_height = -5     # Altura para mantener el contacto del plumón con la hoja
safe_height = 20        # Altura segura cuando no está dibujando

# Coordenadas de inicio para cada figura (pueden ajustarse según la posición de la hoja)
start_position = (200, 0, drawing_height)
square_side = 30        # Tamaño de los lados del cuadrado
triangle_side = 30      # Tamaño de los lados del triángulo
circle_radius = 15      # Radio del círculo
```

Programación de las Figuras

Dibujo de un Cuadrado:

- Define las coordenadas de los cuatro vértices del cuadrado en el espacio de trabajo.
- Usa la función move_to() para mover el plumón entre los vértices y completar la figura.

```
def draw_square():
    device.move_to(200, 0, drawing_height, 0)
    time.sleep(0.5)
    device.move_to(200 + square_side, 0, drawing_height, 0)
    device.move_to(200 + square_side, square_side, drawing_height, 0)
    device.move_to(200, square_side, drawing_height, 0)
    device.move_to(200, 0, drawing_height, 0)
```

Dibujo de un Triángulo:

- Define las coordenadas de los tres vértices del triángulo.
- Realiza movimientos entre estos puntos para formar un triángulo equilátero.

```
def draw_triangle():
    device.move_to(200, 0, drawing_height, 0)
    time.sleep(0.5)
    device.move_to(200 + triangle_side, 0, drawing_height, 0)
    device.move_to(200 + triangle_side / 2, triangle_side * 0.866, drawing_height, 0)
    device.move_to(200, 0, drawing_height, 0)
```

Desafíos

- **Dibujar un Círculo**: Implementa un algoritmo para aproximar un círculo mediante pequeños movimientos lineales alrededor de un punto central.
- **Escalabilidad de Figuras**: Modifica el código para aceptar diferentes tamaños de figuras en cada ejecución.
- **Velocidad y Tiempo de Ciclo**: Ajusta las velocidades de movimiento y los tiempos de espera para optimizar el tiempo de dibujo sin perder precisión.