```python
# ============================================================
# Team Members:
# 1. Akansha Shrestha(as4329)
# 2. Dikshanya Lashmi Ramaswamy(dr655)
# Project: Hershey’s Kisses Placement Robot
# Date: 2024-12-15
# Section: Tuesday, Group 7
# ============================================================

import time
import RPi.GPIO as GPIO
import pygame
import pigame
import math
import sys
import os
from pygame.locals import *

# -----------  Setup environment for the framebuffer  ---------------
os.putenv('SDL_VIDEODRV', 'fbcon')
os.putenv('SDL_FBDEV', '/dev/fb0')
os.putenv('SDL_MOUSEDRV', 'dummy')
os.putenv('SDL_MOUSEDEV', '/dev/null')
os.putenv('DISPLAY', '')

# ----------------------  GPIO Setup  -----------------------------
GPIO.setmode(GPIO.BCM)
# Button setup
GPIOBtns = [17, 22, 23, 27]
for button in GPIOBtns:
    GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# ----------------------  Servo Setup  ----------------------------
servo_pin = 24
GPIO.setup(servo_pin, GPIO.OUT)
freq = 50
dc = 2.5
servo_dc = [2.5, 4.6, 6.3, 8.4, 10.5]
servo_driver1 = GPIO.PWM(servo_pin, freq)

# ----------------------  Line Follower ---------------------------
ENC_CENTER = 25
```

```
ENC_LEFT = 12
ENC_RIGHT = 20
GPIO.setup(ENC_CENTER, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(ENC_LEFT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(ENC_RIGHT, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# ----------------------  Motor Setup ----------------------------
AI_1_pin, AI_2_pin, PWM_Apin = 5, 6, 26
BI_1_pin, BI_2_pin, PWM_Bpin = 19, 13, 16
motor_pins = [AI_1_pin, AI_2_pin, PWM_Apin, BI_1_pin, BI_2_pin, PWM_Bpin]
for pin in motor_pins:
    GPIO.setup(pin, GPIO.OUT)

pwm_driver1 = GPIO.PWM(PWM_Apin, freq)
pwm_driver2 = GPIO.PWM(PWM_Bpin, freq)
pwm_driver1.start(0)
pwm_driver2.start(0)

# ----------------------  Pygame Setup  ------------------------
pygame.init()
pitft = pigame.PiTft()
screen = pygame.display.set_mode((320, 240))
pygame.display.set_caption("Robot Cone Drop Simulation")

# Colors and Fonts
WHITE, BLACK, RED, GREEN = (255, 255, 255), (0, 0, 0), (255, 0, 0), (0,
255, 0)
font_small = pygame.font.Font(None, 20)

# Button setup
btn_start_rect = pygame.Rect(20, 60, 60, 40)
btn_stop_rect = pygame.Rect(20, 120, 60, 40)
btn_quit_rect = pygame.Rect(240, 90, 60, 40)
btn_manual_rect = pygame.Rect(130, 200, 60, 40)

# ---------------------- Variables ----------------------------
cone_count, servo_index, rotation = 0, 0, False
last_drop_time = time.time()
drop_interval = 2

# PID variables
Kp, Kd, Ki = 20, 10, 1
prev_error, integral = 0, 0
```

```
# ----------------------  Move Servo ----------------------------
def move_servo(index):
    servo_driver1.ChangeDutyCycle(servo_dc[index])
    time.sleep(0.5)

# ----------------------  Drop Cone  ----------------------------
def drop_cone():
    global servo_index, rotation
    if not rotation:
        servo_index = 0
        rotation = True
    else:
        servo_index = (servo_index + 1) % 5
    move_servo(servo_index)
    print(f"Dropped cone at {time.strftime('%H:%M:%S')} - Servo position:
{servo_index + 1}")

# ----------------------  Draw Disk  ----------------------------
def draw_disk(surface):
    center, outer_radius, hole_radius = (160, 120), 80, 10
    hole_offset, num_holes, initial_offset = outer_radius - 15, 5, 270
    pygame.draw.circle(surface, BLACK, center, outer_radius, 2)
    for i in range(num_holes):
        angle = (initial_offset - i * (180 / (num_holes - 1))) % 360
        x = center[0] + hole_offset * math.cos(math.radians(angle))
        y = center[1] - hole_offset * math.sin(math.radians(angle))
        color = RED if i == servo_index else WHITE
        pygame.draw.circle(surface, color, (int(x), int(y)), hole_radius)
        pygame.draw.circle(surface, BLACK, (int(x), int(y)), hole_radius,
2)
        hole_number_text = font_small.render(str(i + 1), True, BLACK)
        surface.blit(hole_number_text,
hole_number_text.get_rect(center=(int(x), int(y))))
    pygame.draw.circle(surface, RED, center, 5)

# ----------------------  Draw Buttons   ----------------------------
def draw_buttons():
    for btn, text in zip([btn_start_rect, btn_stop_rect, btn_quit_rect,
btn_manual_rect], ['START', 'STOP', 'QUIT', 'Manual Drop']):
        pygame.draw.rect(screen, WHITE, btn)
        text_surf = font_small.render(text, True, BLACK)
```

```
        screen.blit(text_surf, text_surf.get_rect(center=btn.center))

# ----------------------  Move Robot  ----------------------------
def move_robot(timeout, drop_interval):
    global prev_error, integral, servo_index, rotation, motor_running,
last_drop_time
    start_time = time.time()
    last_drop_time = time.time()
    while time.time() - start_time < timeout:
        if not GPIO.input(GPIOBtns[3]):
            print("Quitting Program")
            code_running = False
            time.sleep(0.2)
        screen.fill(WHITE)
        draw_disk(screen)
        draw_buttons()
        pitft.update()
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                x, y = event.pos
                if btn_stop_rect.collidepoint(x, y):
                    motor_running = False
                    stop_motor()
                    last_drop_time = time.time()  # Reset the timer
                elif btn_start_rect.collidepoint(x, y):
                    motor_running = True
                elif btn_manual_rect.collidepoint(x, y):
                    drop_cone()
                elif btn_quit_rect.collidepoint(x, y):
                    return False

        if motor_running:
            if not GPIO.input(GPIOBtns[3]):
                print("Quitting Program")
                return False
                time.sleep(0.2)
            # Line following logic (unchanged)
            if GPIO.input(ENC_CENTER) == GPIO.HIGH and
GPIO.input(ENC_RIGHT) == GPIO.LOW and GPIO.input(ENC_LEFT) == GPIO.LOW:
                print("Black line detected")
                error = 0
                pwm_driver1.ChangeDutyCycle(50)
                pwm_driver2.ChangeDutyCycle(50)
```

```
                GPIO.output(AI_1_pin, GPIO.HIGH)
                GPIO.output(AI_2_pin, GPIO.LOW)
                GPIO.output(BI_1_pin, GPIO.HIGH)
                GPIO.output(BI_2_pin, GPIO.LOW)
                prev_error = error
            elif GPIO.input(ENC_CENTER) == GPIO.LOW and
GPIO.input(ENC_LEFT) == GPIO.HIGH and GPIO.input(ENC_RIGHT) == GPIO.LOW:
                print("Moving RIGHT")
                error = -1
                integral = max(min(integral + error, 100), -100)
                pid = Kp * error + Kd * (error - prev_error) + Ki *
integral
                dc_A = max(0, min(100, 25 + pid))
                dc_B = max(0, min(100, 25 - pid))
                pwm_driver1.ChangeDutyCycle(dc_A)
                pwm_driver2.ChangeDutyCycle(dc_B)
                GPIO.output(AI_1_pin, GPIO.HIGH)
                GPIO.output(AI_2_pin, GPIO.LOW)
                GPIO.output(BI_1_pin, GPIO.HIGH)
                GPIO.output(BI_2_pin, GPIO.LOW)
                prev_error = error
            elif GPIO.input(ENC_CENTER) == GPIO.LOW and
GPIO.input(ENC_RIGHT) == GPIO.HIGH and GPIO.input(ENC_LEFT) == GPIO.LOW:
                print("Moving LEFT")
                error = 1
                integral = max(min(integral + error, 100), -100)
                pid = Kp * error + Kd * (error - prev_error) + Ki *
integral
                dc_A = max(0, min(100, 25 + pid))
                dc_B = max(0, min(100, 25 - pid))
                pwm_driver1.ChangeDutyCycle(dc_A)
                pwm_driver2.ChangeDutyCycle(dc_B)
                GPIO.output(AI_1_pin, GPIO.HIGH)
                GPIO.output(AI_2_pin, GPIO.LOW)
                GPIO.output(BI_1_pin, GPIO.HIGH)
                GPIO.output(BI_2_pin, GPIO.LOW)
                prev_error = error
            elif GPIO.input(ENC_CENTER) == GPIO.LOW and
GPIO.input(ENC_RIGHT) == GPIO.LOW and GPIO.input(ENC_LEFT) == GPIO.LOW:
                print("Moving BACKWARD")
                pwm_driver1.ChangeDutyCycle(40)
                pwm_driver2.ChangeDutyCycle(40)
                GPIO.output(AI_1_pin, GPIO.LOW)
```

```
                GPIO.output(AI_2_pin, GPIO.HIGH)
                GPIO.output(BI_1_pin, GPIO.LOW)
                GPIO.output(BI_2_pin, GPIO.HIGH)

            # Drop cone logic
            if time.time() - last_drop_time >= drop_interval:
                last_drop_time = time.time()
                pwm_driver1.ChangeDutyCycle(0)
                pwm_driver2.ChangeDutyCycle(0)
                drop_cone()
                time.sleep(1)
        pygame.display.flip()
        time.sleep(0.1)

    return True


def stop_motor():
    pwm_driver1.ChangeDutyCycle(0)
    pwm_driver2.ChangeDutyCycle(0)
    GPIO.output(AI_1_pin, GPIO.LOW)  # Stop motor A
    GPIO.output(AI_2_pin, GPIO.LOW)
    GPIO.output(BI_1_pin, GPIO.LOW)  # Stop motor B
    GPIO.output(BI_2_pin, GPIO.LOW)
    print("Motors stopped.")

# ---------------------- MAIN FUNCTION
-------------------------------------------------
print("Starting the simulation... Press Ctrl+C to quit.")
code_running = True
motor_running = False
servo_driver1.start(dc)


while code_running:
    screen.fill(WHITE)
    draw_disk(screen)
    draw_buttons()
    pitft.update()

    if not GPIO.input(GPIOBtns[3]):
        print("Quitting Program")
        code_running = False
```

```
        time.sleep(0.2)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            raise KeyboardInterrupt
        elif event.type == pygame.MOUSEBUTTONDOWN:
            x, y = event.pos
            if btn_start_rect.collidepoint(x, y):
                print("Simulation started. Moving and dropping cones.")
                motor_running = True
                code_running = move_robot(timeout=600, drop_interval=5)  #
10 minutes timeout, drop every 5 seconds
            elif btn_stop_rect.collidepoint(x, y):
                motor_running = False
                stop_motor()
            elif btn_quit_rect.collidepoint(x, y):
                code_running = False
            elif btn_manual_rect.collidepoint(x, y):
                drop_cone()

    pygame.display.flip()

# ---------------------- CLEANUP ----------------------------
print("Exiting the simulation. Goodbye!")
servo_driver1.stop()
pwm_driver1.stop()
pwm_driver2.stop()
GPIO.cleanup()
pygame.quit()
del pitft
sys.exit()
```