```
import RPi.GPIO as GPIO
import cv2
import numpy as np
import time

# Define GPIO pins
connected to L298N motor driver
motor_pins = {
    'left': {
        'IN1': 17,

'IN2': 27,
        'Enable': 22
    },
    'right': {
        'IN1': 23,
        'IN2':
24,
        'Enable': 25
    }
}

# Initialize
GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
for motor in motor_pins.values():

GPIO.setup(motor['IN1'], GPIO.OUT)
    GPIO.setup(motor['IN2'], GPIO.OUT)

GPIO.setup(motor['Enable'], GPIO.OUT)
    motor['PWM'] = GPIO.PWM(motor['Enable'], 1000)

#
Start PWM with 0% duty cycle
for motor in motor_pins.values():
    motor['PWM'].start(0)

#
Define motor control functions
def move_forward(speed=0.5):
    print("Moving
forward")
    for motor in motor_pins.values():
        GPIO.output(motor['IN1'],
GPIO.HIGH)
        GPIO.output(motor['IN2'], GPIO.LOW)

motor['PWM'].ChangeDutyCycle(speed * 100)

def rotate_left(speed=0.5):

print("Rotating left")
    GPIO.output(motor_pins['left']['IN1'], GPIO.LOW)

GPIO.output(motor_pins['left']['IN2'], GPIO.HIGH)

motor_pins['left']['PWM'].ChangeDutyCycle(speed * 100)

def rotate_right(speed=0.5):

print("Rotating right")
    GPIO.output(motor_pins['right']['IN1'], GPIO.LOW)

GPIO.output(motor_pins['right']['IN2'], GPIO.HIGH)

motor_pins['right']['PWM'].ChangeDutyCycle(speed * 100)

def stop_motors():

print("Stopping motors")
    for motor in motor_pins.values():
```

```
GPIO.output(motor['IN1'], GPIO.LOW)
        GPIO.output(motor['IN2'], GPIO.LOW)

motor['PWM'].ChangeDutyCycle(0)

def detect_ball_position():
    cap =
cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        if not ret:

          print("Error: Unable to capture video.")
            break

        hsv
= cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        lower_color = np.array([23, 50, 50])

  upper_color = np.array([43, 150, 150])

        mask = cv2.inRange(hsv, lower_color,
upper_color)

        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

        if contours:
            largest_contour = max(contours,
key=cv2.contourArea)
            M = cv2.moments(largest_contour)
            if
M["m00"] != 0:
                cX = int(M["m10"] / M["m00"])

             cY = int(M["m01"] / M["m00"])

                if cX <
frame.shape[1] // 3:
                    position = "left"

print("Ball detected on the left")
                elif cX > 2 * frame.shape[1]
// 3:
                    position = "right"
                    print("Ball
detected on the right")
                else:
                    position =
"centre"
                    print("Ball detected in the centre")


                break
                cv2.putText(frame, position, (cX, cY),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        masked_frame = cv2.bitwise_and(frame,
frame, mask=mask)

        combined_frame = np.hstack((frame, masked_frame))

cv2.imshow("Original vs Masked", combined_frame)

        if cv2.waitKey(1) &
0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

return position if position in ["left", "centre", "right"] else
"centre"
```

```
def main():
    count = 0
    while count < 100:

ball_position = detect_ball_position()
        print("ball",ball_position)

if ball_position == "left":
            rotate_left()
        elif ball_position ==
"centre":
            move_forward()
        elif ball_position ==
"right":
            rotate_right()
        else:
            print("Unknown
position", flush=True)

        count += 1
        time.sleep(1)  # Delay for 10
seconds

if __name__ == "__main__":
    try:
        main()
    except
KeyboardInterrupt:
        print("Program terminated by user", flush=True)

finally:
        stop_motors()
        GPIO.cleanup()
```