Traffic Light Control System

PROJECT REPORT

NAME: - Jitesh Kumar
BRANCH: - CSE(AIML)
SECTION: - B
ROLL NO.: - 202401100400103

Traffic Light Control System

1. Introduction

Traffic Light Control System is a Python and Matplotlib library-based simulation. The project simulates graphically the way a three-light-phase traffic light works: Red (Stop), Yellow (Wait), and Green (Go). The system allocates the correct time among the phases, providing an interactive and dynamic simulation of actual traffic lights.

2. Objectives

- To simulate a real-world traffic light system.
- To provide a dynamic framework where the total time can be adjusted.
- To display a live countdown for each signal phase.
- To provide user intervention to dynamically change the cycle time.

3. Implementation and Features

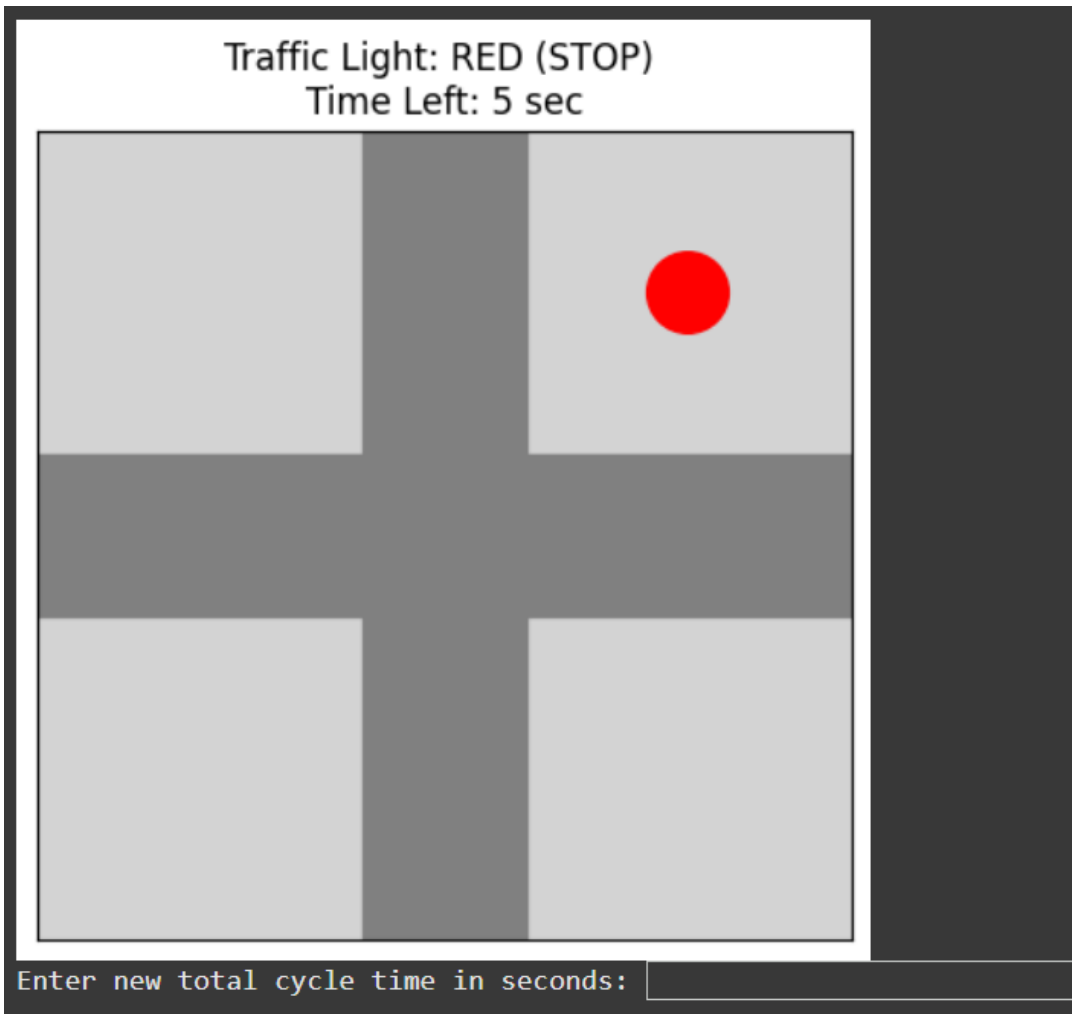
- Graphical Representation: Employs Matplotlib to present the traffic light system.
- Dynamic Timing Allocation: The program starts at a default of 15 seconds and allocates time in a 7:3:7 ratio (Red:Yellow:Green).
- User Input Handling: User can alter total cycle time during interruption.
- Real-Time Countdown: Displays the time remaining for every light phase.
- Keyboard Interrupt Handling: Allows the user to break the simulation and enter a new cycle time manually.

4. Working Mechanism

The program starts with a default cycle duration of 15 seconds.
Time is divided in the ratio of 7:3:7 respectively.
The time is divided as follows:
- **Red (STOP):** 7/17 of all time.
- **Yellow (WAIT):** 3/17 of the entire time.
- **Green (GO**): 7/17 of total time.

The simulation is iteratively executed, showing the countdown timer and updating the graphical output.

The run can be halted manually by the user using the Stop button (Google Colab) or Ctrl + C (Terminal), upon which they are prompted to input a fresh total cycle time.
The simulation is repeated later with the new cycle time input by the user.



5. Code Overview

The implementation consists of:
- Traffic Light Representation: Generates a graphical traffic system including roads and lights.
- Time-Based Simulation: Uses a loop to constantly cycle through the traffic lights.
- User Input and Exception Handling: Facilitating smooth user interaction and simulation control.

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import time
from IPython.display import display, clear_output

# Function to get the label for the traffic light
def get_light_label(color):
    labels = {
        'red': 'RED (STOP)',
        'yellow': 'YELLOW (WAIT)',
        'green': 'GREEN (GO)'
    }
    return labels.get(color, '').upper()

# Function to draw the traffic system with the active light
def draw_traffic_system(light_color, remaining_time):
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.set_xlim(0, 10)
    ax.set_ylim(0, 10)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_facecolor('lightgray')

    # Draw roads
    ax.add_patch(patches.Rectangle((4, 0), 2, 10, color='gray'))  # Vertical road
    ax.add_patch(patches.Rectangle((0, 4), 10, 2, color='gray'))  # Horizontal road
```

```
    # Draw traffic light signal
    traffic_light = patches.Circle((8, 8), 0.5, color=light_color)
    ax.add_patch(traffic_light)
    # Set title with light label and countdown
    ax.set_title(f'Traffic Light: {get_light_label(light_color)} \nTime Left: {remaining_time} sec')
    clear_output(wait=True)
    display(fig)
    plt.close(fig)
# Function to run the traffic light simulation
def traffic_simulation(total_time):
    red_time = int(total_time * 7 / 17)
    yellow_time = int(total_time * 3 / 17)
    green_time = total_time - (red_time + yellow_time)

    sequence = [('red', red_time), ('yellow', yellow_time), ('green', green_time)]

    while True:
        for light, duration in sequence:
            for remaining_time in range(duration, 0, -1):
                print(f'{get_light_label(light)} - {remaining_time} sec left')
                draw_traffic_system(light, remaining_time)
                time.sleep(1)
# Run the traffic light simulation with a default cycle time
if __name__ == "__main__":
    try:
        default_time = 15
        print(f"Starting with default cycle time of {default_time} seconds...")
```

6. Future Improvements

- Employing AI-based traffic management through the identification of current car density.
- Adding machine learning to dynamically regulate traffic light cycles.

7. Conclusion

This project can simulate a traffic light control system with real-time visualization and adjustable time control. It provides an interactive way of learning traffic control mechanisms and can be further developed with AI and real-world integration.