



Traffic light simulator

CSA0884 – Python Programming to Analyse Data – Slot A

Guided By,

Dr A G Ramchandran (Course Faculty)
Python Programming
SSE, SIMATS

Project by,

K Prasanth (192211920)
CH Ajay (192210473)

Computer Science & Engineering
SSE, SIMATS

ABSTRACT

The Traffic Light Simulator is a Python-based program designed to replicate the functionality of a traffic light system at a road intersection. The simulation is implemented using the Python programming language, employing basic threading to independently control traffic lights on different roads. Each traffic light undergoes a cycle comprising green, yellow, and red phases, with customizable durations to mimic real-world scenarios. The program allows for the simulation of multiple intersections, enabling the study of traffic dynamics and coordination between roads. The Traffic Light Simulator offers a starting point for educational purposes, providing a platform for users to explore and understand the fundamental principles of traffic management. The modular design allows for extensibility, making it suitable for integration with graphical libraries or frameworks to enhance user experience and facilitate more detailed simulations.

Keywords

Road safety, Green light Duration, Red light interval

INTRODUCTION

- The Traffic Light Simulator presented here is a software application developed to emulate the behavior of traffic lights at a road intersection.
- Traffic lights play a pivotal role in regulating this flow, orchestrating a synchronized dance of movement and halts.
- This simulation aims to provide a platform for users to explore and analyze the intricate dynamics of traffic control systems.
- The simulator is designed to serve as an educational tool, offering insights into the fundamental principles of traffic management while providing a foundation for further development and exploration.
- This introduction sets the stage for a comprehensive exploration of the simulator's features, capabilities, and potential applications in the realm of traffic engineering and urban planning.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirements:

Processor: 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz 2.59 GHz

Installed RAM: 16.0 GB (15.9 GB usable)

System type: 64-bit operating system, x64-based processor

Software requirements:

Pycharm

Pygame module

Class module

EXISTING SYSTEM

- It provides a wide range of features, including traffic light control, vehicle routing, and emissions modeling.
- It is widely used for modeling and simulating complex traffic scenarios, including intersections with traffic lights.
- The traffic signal controller is a key component responsible for managing the timing and operation of traffic lights at intersections. It determines when each signal (red, yellow, green) should be active based on predefined timing plans.

Disadvantages:

Idling and Fuel Consumption:

Vehicles waiting at red lights contribute to idling, leading to increased fuel consumption and emissions. This is particularly concerning in terms of environmental impact and air quality.

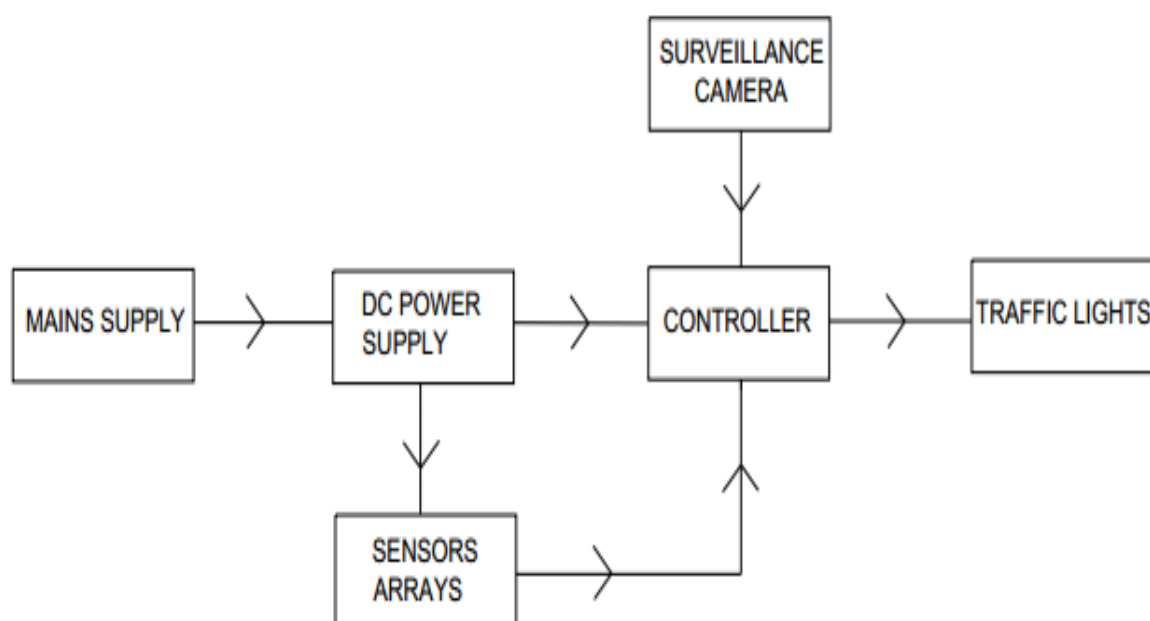
Dependency on Electricity:

Traffic lights require a stable supply of electricity. Power outages or disruptions can lead to non-functioning traffic lights, potentially causing chaos at intersections.

PROPOSED SYSTEM

- Implement adaptive signal control algorithms that can dynamically adjust signal timings based on real-time traffic conditions.
- Establish a robust communication network to connect traffic signal controllers at different intersections.
- Implement machine learning algorithms to analyze historical and real-time data for optimizing signal timings.
- Develop a user interface accessible to traffic management authorities for real-time monitoring and adjustment of signal timings.
- Incorporate energy-efficient LED lights for traffic signals.

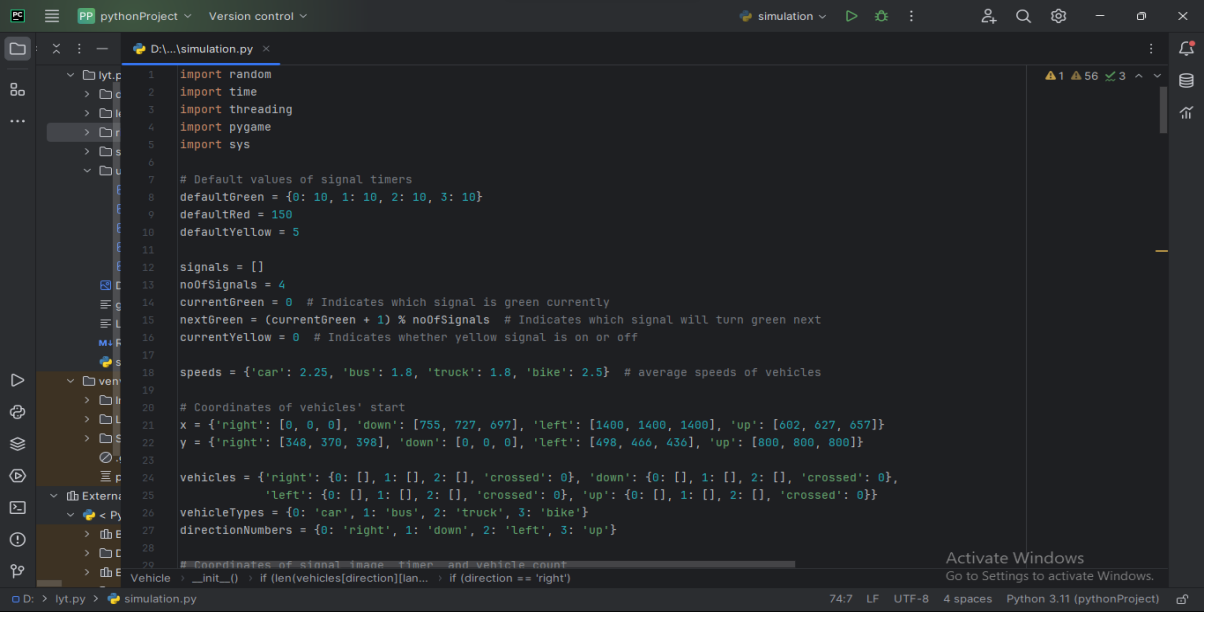
ARCHITECTURE



DESIGN

- Traditionally at the top, indicating stop. Use a bright, easily distinguishable red color.
- In the middle, indicating prepare to stop. Use a distinct yellow color.
- At the bottom, indicating go. Use a bright, easily distinguishable green color.
- Use energy-efficient LED lights for each signal. LEDs offer long life, low power consumption, and better visibility.
- Implement a control system that manages the timing of each signal. The controller should follow a predefined sequence (e.g., red, green, yellow) and be programmable for different traffic patterns.

CODING



```
1 import random
2 import time
3 import threading
4 import pygame
5 import sys
6
7 # Default values of signal timers
8 defaultGreen = {0: 10, 1: 10, 2: 10, 3: 10}
9 defaultRed = 150
10 defaultYellow = 5
11
12 signals = []
13 noOfSignals = 4
14 currentGreen = 0 # Indicates which signal is green currently
15 nextGreen = (currentGreen + 1) % noOfSignals # Indicates which signal will turn green next
16 currentYellow = 0 # Indicates whether yellow signal is on or off
17
18 speeds = {'car': 2.25, 'bus': 1.8, 'truck': 1.8, 'bike': 2.5} # average speeds of vehicles
19
20 # Coordinates of vehicles' start
21 x = {'right': [0, 0, 0], 'down': [755, 727, 697], 'left': [1400, 1400, 1400], 'up': [602, 627, 657]}
22 y = {'right': [348, 370, 398], 'down': [0, 0, 0], 'left': [498, 466, 436], 'up': [800, 800, 800]}
23
24 vehicles = {'right': {0: [], 1: [], 2: [], 'crossed': 0}, 'down': {0: [], 1: [], 2: [], 'crossed': 0},
25             'left': {0: [], 1: [], 2: [], 'crossed': 0}, 'up': {0: [], 1: [], 2: [], 'crossed': 0}}
26 vehicleTypes = {0: 'car', 1: 'bus', 2: 'truck', 3: 'bike'}
27 directionNumbers = {0: 'right', 1: 'down', 2: 'left', 3: 'up'}
28
29 # Coordinates of signal image, timer, and vehicle count
30 Vehicle = __init__() if (len(vehicles[direction])>0) if (direction == 'right')
```

TESTING

- Physically inspect and manually operate each component of the traffic light system, including signal heads, pedestrian signals, and control panel.

- Use simulation software to model and test traffic scenarios under different conditions. This can help identify potential issues with signal timings and coordination.
- Verify that each LED in the signal heads is functioning correctly and is visible from appropriate distances.
- Test the functionality of inductive loop sensors to ensure they detect the presence of vehicles.

OUTPUT



IMPLEMENTATION

- Choose or design signal heads with LED lights for each color (red, yellow, green).
- Ensure the signal heads are durable, weather-resistant, and visible from a distance.
- Implement vehicle and pedestrian detection sensors, such as inductive loop sensors, cameras, or other sensor technologies.
- Connect sensors to the traffic signal controller to provide input for signal timing adjustments.

CONCLUSION

In conclusion, traffic lights play a crucial role in regulating and managing traffic flow, ensuring the safety of pedestrians and drivers alike. These simple yet effective devices help maintain order on roads, preventing accidents and promoting efficient transportation systems. The standardized color-coding of red, yellow, and green has become a universal language that communicates right of way and signals when to stop, proceed with caution, or move forward. As technology advances, there is potential for further innovations in traffic light systems, such as smart signals that adapt to real-time traffic conditions.

FUTURE SCOPE

Connected and Autonomous Vehicles (CAVs):

- Traffic lights may communicate with connected vehicles to optimize traffic patterns and enhance safety.
- Integration with autonomous vehicles to provide them with real-time information about signal changes, facilitating smoother traffic transitions.

Smart Traffic Management Systems:

- Integration with smart city initiatives could lead to traffic lights becoming part of a broader intelligent transportation system.

REFERENCE

- <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6326903>

R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction" in A Bradford Book, Cambridge MA: The MIT press, 1998.

- G.J. Tesauro, "Temporal difference learning and TD-Gammon", *Communications of the ACM*, vol. 38, pp. 58-68, 1995.
- Robert H. Crites and Andrew G. Barto, "Improving elevator performance using reinforcement learning" in *Advances in Neural Information Processing Systems*, The MIT Press, vol. 8, pp. 1017-1023, 1996.

