CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT

OF

SMART TRAFFIC LIGHT CONTROL

Program Name: BCA

Subject Name: Computing Aptitude

Submitted by:

Name: Varun Kumar
UID: 23BCA10437
Section: BCA – 1 'B'

Submitted to:

Name: Ms. Disha Sachdeva
Designation: Asst Professor

2. Abstract

This project implements a C++ simulation of an intelligent traffic light controller that adaptively allocates green-light time to each approach (North, East, South, West) based on current traffic density. The system demonstrates proportional allocation of green durations within specified min/max bounds, dynamic phase changes, and basic fairness checks to avoid starvation. The simulation supports manual or random density input, logs outputs, and is suitable for classroom demonstration of adaptive control concepts used in smart cities.

3. Introduction

Traffic congestion causes delays, fuel waste, and pollution. Traditional fixed-time traffic lights cannot react to changing traffic patterns. Adaptive traffic control systems adjust signal timing in real-time using vehicle-detection data. This project simulates such an adaptive controller to demonstrate improved utilization of intersection green-time allocation according to measured densities.

Objectives

- Design a controller that reads traffic densities and assigns green times adaptively.
- Ensure fairness (no direction starves) and bounds on green times.
- Provide an easy simulation to visualize phases and test strategies.

4. Problem Statement / Objectives

Problem: Fixed-time lights are inefficient under fluctuating traffic.

Goal: Create an adaptive controller to minimize waiting by giving longer green to heavier flows while preserving minimum service to all directions.

Functional Requirements

- Accept traffic density input for 4 directions.
- Compute green-time allocation proportionally but constrained by min/max per-phase.
- Run simulation cycles and show state transitions and waiting times.
- Allow both manual input and random/simulated sensor input.

Non-functional

- Modular C++ design
- Easy to extend for more complex strategies
- Clear logs for testing

5. System Design / Approach

High-level architecture

- **Sensor**: provides traffic counts (vehicles) per direction.
- **Controller**: computes green-allocation each cycle based on densities.
- **TrafficLight/Phase**: represents state (GREEN/RED) and timing.

7. Results / Testing

How to compile & run

- Compile:

```
g++ -std=c++17 smart_traffic.cpp -o smart_traffic
```

- Run:

```
./smart_traffic
```

Choose mode:

- 1: random runs (automated per-cycle random densities)
- 2: interactive input (enter counts each cycle or -1 to randomize)
- 3: example scenario (predefined scenario)

Sample run (example scenario output excerpt)

```
Smart Traffic Light Simulation

Choose mode: 3

Enter cycles to run (default 8): 8

=== Cycle 1 ===

Current counts (N E S W): 18 4 20 3

Allocated green seconds (N E S W): 30 6 22 2

N GREEN for 30s, served ~18 vehicles; remaining 0

E GREEN for 6s, served ~4 vehicles; remaining 0

S GREEN for 22s, served ~20 vehicles; remaining 0

W GREEN for 2s, served ~2 vehicles; remaining 1
```

Cumulative served (N E S W): 18 4 20 2

...

Simulation finished.

Testing Process

- **Unit tests** (manual): verify proportional allocation for different count patterns:
    - All zeros -> equal splits
    - One heavy direction (e.g., N=50 others=1) -> N gets very large but capped by maxGreen
    - Starvation scenario: repeatedly produce zero for a direction to check fairness boost
- **Stress**: run with random counts up to 100 vehicles and different cycleTime, minGreen, maxGreen to ensure allocations stay within constraints.

8. Conclusion

The project demonstrates a simple but effective adaptive traffic signal controller. It allocates green times proportionally, enforces minimum service, caps maximum time, and applies a simple starvation avoidance. The simulation shows that heavy flows receive more green time, reducing queue sizes. The architecture is modular and ready for enhancements such as queue dynamics, multi-lane modeling, vehicle arrival processes, and optimization-based controllers.

9. Future Scope

- Replace proportional allocation with optimization (linear programming) to minimize total delay.

- Use queuing models (Poisson arrivals) and simulate vehicle-by-vehicle movement.
- Add pedestrian phases and emergency vehicle preemption.
- Integrate reinforcement learning-based traffic signal control (e.g., deep RL).
- Real-time integration with sensors (e.g., camera or loop detectors) and hardware signal controllers.

10. References

- D. C. Gazis, Traffic Theory — classic papers on traffic flow.
- S. P. Hoogendoorn & P. H. L. Bovy, "State-of-the-art of vehicular traffic flow modelling."
- Online tutorials for traffic signal control (simulator examples) and general C++ reference (cppreference.com).

Testing Scenarios (suggested)

1. Balanced: {10,10,10,10} -> equal green times
2. Heavy North: {50,2,3,1} -> North should receive capped but much larger green
3. Starvation test: run cycles where West always 0 until boosted by starve threshold