Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

on

"Smart Traffic Controller"

[Code No : COMP 202]

(For partial fulfillment of II/I Year/Semester in Computer Science/Engineering)

Submitted by

Prapti Dhamala (13)

Submitted to

Mr. Sagar Acharya

Department of Computer Science and Engineering

Submission Date:

18/02/2026

Declaration

I hereby declare that the work presented in this report titled “SmartCity Traffic Controller” is my own original work carried out independently. I have not copied from any source except where explicitly cited. All sources and references used have been acknowledged.

Name: Prapti Dhamala

Roll Number: 13

Date: 18/02/2026

Acknowledgement

I would like to express my gratitude to Kathmandu University for providing the resources and environment to carry out this project. I would also like to acknowledge our teacher Mr. Sagar Acharya for his time and effort invested while teaching us DSA. I also thank my family and friends for their constant support and encouragement throughout the development of this project.

Finally, I acknowledge the use of online resources, documentation, and tutorials that helped me understand the concepts of DSA clearly and implement the SmartCity Traffic Controller.

Abstract

The urban traffic congestion has been an alarming challenge affecting time, fuel and environment as a whole. Traditional traffic signal systems often rely on fixed timing schedules, which cannot dynamically adapt to real time traffic conditions.
This project introduces Smart Traffic Controller, an adaptive traffic management system which is made by integrating C- based simulation engine to model traffic at intersection with vehicle queues and priorities and a python based visualization interface using Pygame. The system dynamically selects green light roads, tracks emergency vehicles priorities, and provides real time analytics. External situations like rain and traffic light override manually is implemented.
Through simulation, this system demonstrates the ability to reduce congestion, prioritize critical vehicles, and provide actionable data for smart city traffic management.

Table of Contents

Chapter 1 Introduction

1.1 Background

With the increasing number of vehicles, there is an increase in traffic congestion in urban areas . The urban traffic congestion has been an alarming challenge affecting time, fuel and environment as a whole. Traditionally traffic light signal systems operate in fixed timers, which are unable to adapt with traffic density which results in unnecessary delays and traffic jams.

Modern cities are shifting towards smart traffic controlling systems which adjust traffic signals based on real time road situations. Systems like this help us to improve traffic flow, reduce waiting times, and prioritize certain types of vehicle.

With the increasing amount of traffic on the road, we found a gap between manual traffic signal systems and smart simulation systems, this is where this system is conceived.

1.2 Objectives

The specific objective of this project is :

- To simulate real time traffic flow.
- To implement adaptive traffic signal control.
- To prioritize emergency vehicles.
- To develop an interactive visualization system.
- To enable manual intervention.
- To show critical data, including worst-case delays, vehicle counts, and recommended count.

1.3 Motivation and Significance

The motivation behind this project is to design a simulation based traffic control system that:

- Responds to real time traffic queues.
- Gives priority to emergency vehicles.
- Provides data driven insights to operators.

Chapter 2 Related Works

There are many relevant works done in this sector using DSA concept and they are as follow :

2.1 Traffic Simulation Using DSA

Many researchers have proposed this idea where we can use graph based algorithms to model road networks and optimize routing and flow.

- Network graph modelling: Roads and intersections are modeled as nodes and edges.
- Shortest Path Algorithms: Algorithms like Dijkstra are used to find optimal routes for vehicles.
- Queue Data Structures: Queues manage vehicles at intersections.

2.2 Adaptive Traffic signal control

A class of related work focuses on real-time traffic light decisions based on current traffic conditions, often using:

- Priority Queues : This is used to rank intersections or vehicles needing green lights.
- Sensor Data Integration: It updates data structures dynamically.

Chapter 3 Design and Implementation

This system was developed using a modular approach, separating the traffic simulation engine (C backend) and the visualization part was handled by Pygame/Python. This separation ensures clarity in implementation, visualization, and maintainability.

3.1 System Design Approach

The development of this system followed the following designing procedure as follow:

1. Problem Analysis
    - Identified limitations of fixed timer traffic signals.
    - Defined the need for adaptive traffic signals.
    - Determined the need for emergency vehicle prioritization.
2. System Architecture Planning
    - Choose C for simulation using the DSA concept.
    - Choose Python with pygame for the visualization interface.
3. Data Structure Design
    - Designed Queue structures for each road.
    - Designed Hash table for tracking vehicle waiting lines.
    - Designed JSON structure for inter-process communication.
4. Algorithm design
    - Developed vehicle spawning logic.
    - Implemented queue based green light selection algorithm.
    - Designed priority based enqueue logic for emergency vehicles.
5. Implementation
    - Implemented backend simulation in C.
    - Implemented frontend visualization in Python.
    - Integrated both modules using subprocess.
6. Testing and Validation
    - Tested rush hour logic.

- Verified emergency vehicle prioritization.
- Validated dashboard analytics accuracy.
- Tested manual override functionality.

3.2 Software Architecture

This is how the project's system architecture looks like our C simulation works as the brain and python simulation works as our visualization eye which is best described in the following figure.

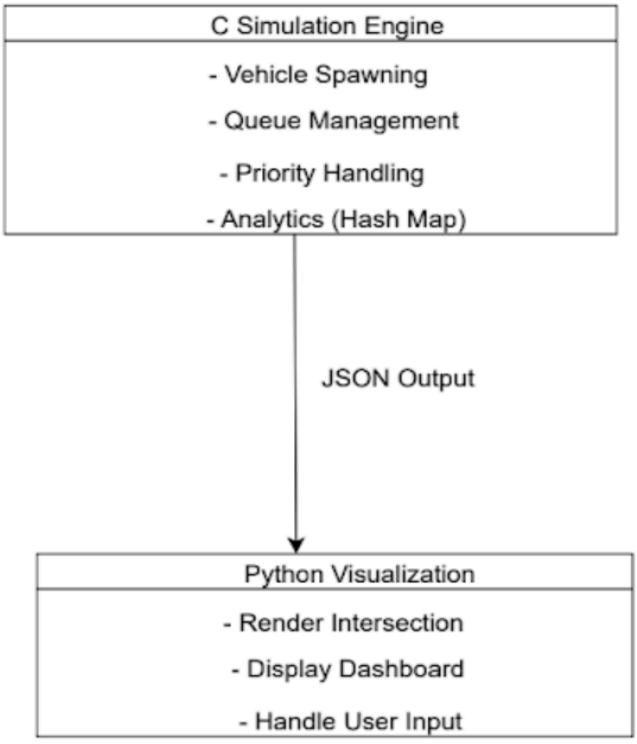

Fig 3.2.1 : Software Architecture of Smart Traffic Controller

3.3 Algorithm Design

3.3.1 Vehicle Spawning Algorithm

1. Check present virtual hour.
2. If hour is between 17 and 19 → increase spawn probability.
3. Generate vehicle with:
    - Random ID
    - Random priority (emergency or normal)
4. Insert into respective road queue.

3.3.2 Traffic Signal Selection Algorithm

Objective: Select the road with maximum congestion.

Steps:

1. For each road: Calculate queue size.
2. Compare queue sizes.
3. Select road with maximum size.
4. Set selected road to Green.
5. Allow limited vehicles to pass.
6. Update total passed count.

3.3.3 Emergency Priority Handling

When enqueueing a vehicle:

- If priority = 1: Insert ahead of normal vehicles.
- Else: Insert at rear.

This ensures emergency vehicles are cleared faster.

3.4 Flowchart

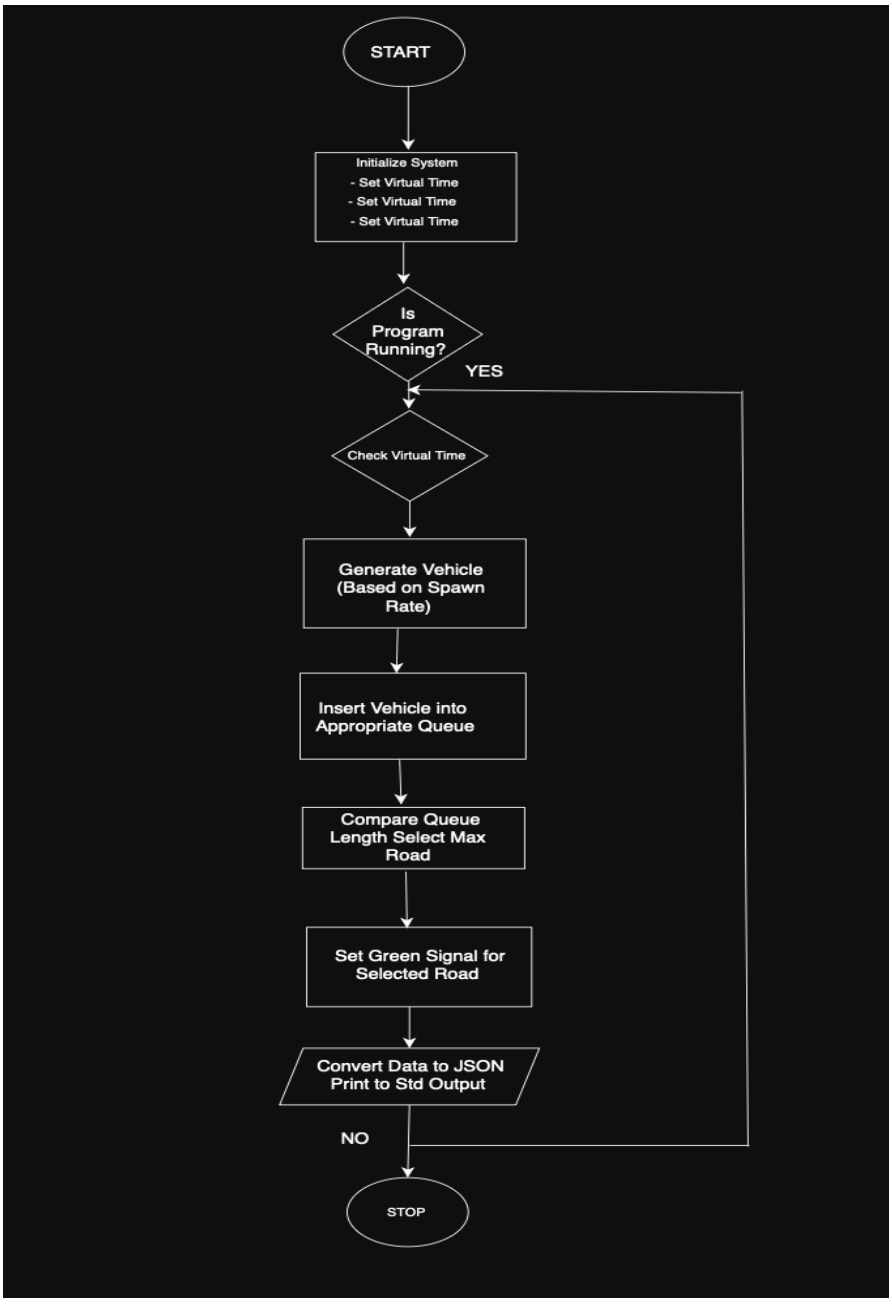

Fig 3.4.1 : Flowchart of C simulation of Smart Traffic Controller

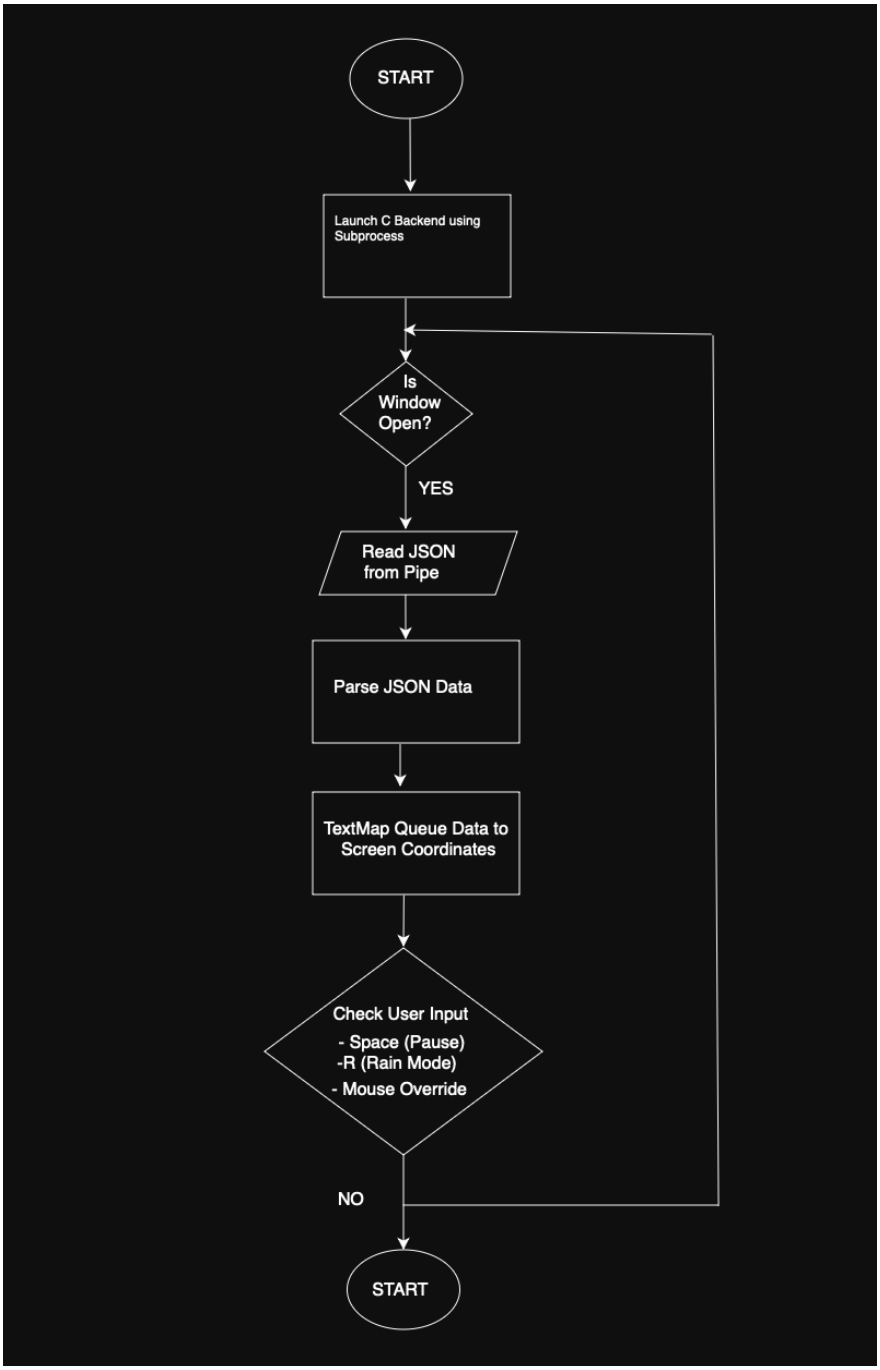

Fig 3.4.2 : Flowchart of Python visualization of Smart Traffic Controller

3.5 Software Specifications

3.5.1 Functional Requirements

The functional requirements of the system are :

- Simulates four traffic roads with independent queues.
- Track Vehicle waiting time
- Dynamically generate vehicles based on virtual time
- Prioritize emergency vehicles.
- Provide dashboard analytics
- Allow manual override of traffic light.
- Allow pause and weather toggle option.

3.5.2 Non-Functional Requirements

The non- functional requirements of the system are :

- Performance: System should update in every 0.3 seconds without lag.
- Reliability: JSON communication must remain stable.
- Usability: Interface should be interactive and easy to understand.
- Maintainability: Code should be modular and well-structured.

3.5.2 Software dependencies

The following are the software dependencies:

- C Compilers
- Python 3.x
- Pygame library
- JSON module
- Subprocess module

3.6 Hardware Specifications

The minimum hardware requirements are :

- Processor: Intel i5 or higher.
- RAM: 4GB minimum.
- Storage: 500 MB free space.
- Graphics: Basic GPU capable of running pygame.

Chapter 4 Discussion on the achievements

The primary objective of this project was to build a real time smart traffic control simulation system using Data Structures and Algorithms (DSA). And this project successfully demonstrates dynamic signal simulation based on vehicle density using queue and hash data structures. The implementation effectively integrated a C based backend and python based frontend ensuring smooth communication between frontend and back through JSON data exchange.

4.1 Challenges faced during development

In the development process of this system, there were plenty of challenges were encountered :

- Inter process communication : Establishing communication link between C backend and frontend required a lot of research and analysis and then we came across python frontend requiring handling of subprocess execution and JSON parsing.
- Real time synchronization : Maintaining real time updates of vehicles without freezing graphical interface requires loop control.

- Dynamic Signal allocation logic: designing an efficient algorithm to compare queue lengths and assigning green signals required optimization to avoid delay.
- User interface handling : Managing pause mode, raim simulation, and override without disturbing the main loop was quite difficult to figure out.

4.2 Deviation from initial objectives :

Initially, the system aimed to implement the features on graphing the route that the vehicle traveled but due to the time constraint, these features were not considered as of now. We also planned on giving AI based recommendations on how to optimize your time but we settled in what we can in the limited time.

4.3 Measurable Results

The measurable progress we made in this project is enlisted below :

- Reduced average waiting time during rush hour .
- Balanced queue distribution through dynamic signal switching.
- Accurate synchronization between frontend and backend modules.

4.4 Features:

The following are the key features that were implemented in the system :

1. Dynamic signal control : Green signal allocation based on maximum queue length.
2. Real time visualization: Live graphical representation of roads, vehicles and traffic lights.
3. Queue based vehicle management: Each road maintains a different queue to simulate traffic flow .
4. Rush hour simulation : Increased vehicle spawn rate during specific time intervals.
5. Manual override system : Users can manually control signals.
6. Pause and weather mode : Space key pause simulation and rain ode affects traffic behaviour.
7. Producer Consumer architecture: Backend generates traffic data, frontend consumes it.

Chapter 5 Conclusion and Recommendation

This project successfully archives its primary objective to implement DSA in smart control signals such as queue. The system effectively simulates real time traffic flow and dynamically generates the signals based on vehicle density. Integration between backend logic and frontend visualization was done seamlessly. Although advanced predictive techniques were not used , this system works perfectly to visualize dsa concept.

5.1 Limitations

Even though this project perfectly visualizes there are a lot of limitation that has to be overcomed and they are :

- This system does not use real world traffic data.
- No machine learning or predictive models were used
- Limited scalability for large multi-intersection networks.
- This system runs locally and does not support distributed architecture.

5.2 Future Enhancement

The following enhancements can improve the system:

- Implementation of emergency vehicle detection and priority control.
- Integration of AI/ML algorithms for traffic prediction.
- Support for multiple interconnected intersections.
- Real-time data integration from IoT traffic sensors.
- Cloud-based deployment for scalability.
- Performance analytics dashboard with statistical reports.
- Database integration for storing traffic history.

References

1.Pygame Community. (n.d.). *Pygame documentation*. https://www.pygame.org/docs/

2. Cppreference contributors. (n.d.). *C standard library reference*. https://en.cppreference.com/w/c

3. Ghazal, B., ElKhatib, K., Chahine, K., & Kherfan, M. (2016, April). Smart traffic light control system. In *2016 third international conference on electrical, electronics, computer engineering and their applications (EECEA)* (pp. 140-145). IEEE.