SIMULATE TRAFFIC LIGHT SYSTEM

MINI PROJECT REPORT

By

APPAJI SREE DHARMA SHASTA RAO[RA2211026010162]
GUNTOORI MRUNAL VARMA[RA2211026010156]
BAKKAREDDY MANIDEEPAK REDDY[RA2211026010166]

Under the guidance of

Dr. KANIPRIYA M

In partial fulfillment for the Course

of

21CSS201T - Computer Organization and Architecture

in CINTEL - **Computer Science and Engineering (AIML)**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSS201T Computer Organization and Architecture** entitled in " **SIMULATE TRAFFIC LIGHT SYSTEM**" is the bonafide work of **APPAJI SREE DHARMA SHASTA RAO(RA2211026010162), GUNTOORI MRUNAL VARMA(RA2211026010156). and BAKKAREDDY MANIDEEPAK REDDY (RA2211026010166)** who carried out the work under my supervision.

SIGNATURE

Dr Kanipriya M
COA– Course Faculty
Assistant Professor
Department of Computational Intelligence

SRM Institute of Science and Technology
Kattankulathur

SIGNATURE

Dr Annie Uthra R
Head of the Department
Professor
Department of Computational Intelligence
SRM Institute of Science and Technology Kattankulathur

ABSTRACT

The objective of this project is to show an automated traffic light control system that effectively manages and simulate traffic flow in busy junctions. Traffic congestion and inefficiency at intersections have become a growing concern in urban areas. This project addresses these challenges by designing a code-based solution that optimizes the control of traffic lights and their timings. The system utilizes a software-based approach to show automated operation of traffic lights, ensuring smooth traffic flow, minimizing congestion, and enhancing safety for commuters. By analyzing real-time traffic data, the code dynamically shows traffic light timings to respond to changing traffic conditions, which prevents gridlock and promoting efficient vehicle movement.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. C. Malathy,** Professor, Department of Networking and Communication and Course Coordinator **Dr. Kanipriya M,** Assistant Professor, Department of Computational Intelligence for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. Kanipriya M,** Assistant Professor, Department of Computational Intelligence for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **Head of the Department, Dr. R. Annie Uthra, Professor, Department of Computational Intelligence** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

SIMULATE TRAFFIC LIGHT SYSTEM

1. INTRODUCTION

- 1.1-**OBJECTIVE**

To display Traffic management system that moderates traffic lights with different timings through a GUI. The gives a rough idea on working and managing Traffic lights.

1.2-INTRODUCTION

In today's rapidly growing urban landscapes, efficient traffic management is a critical necessity to ensure the smooth flow of vehicles and pedestrians while minimizing congestion and accidents. One of the key components of any traffic management system is the intelligent control of traffic lights. These traffic lights are strategically placed at intersections and crossings, playing a vital role in regulating the movement of vehicles and pedestrians, ensuring order and safety on the roads.

To address the challenges of modern traffic management, we propose the development of a Traffic Management System with a Graphical User Interface (GUI). This system aims to provide a visual representation of how traffic lights are controlled and managed, allowing users to understand the principles behind traffic light sequencing and timing.

The system will simulate different traffic scenarios, complete with intersections and traffic lights. Users will be able to observe how the traffic lights change in response to traffic conditions. The system will allow users to adjust the timing of traffic lights at various intersections. This feature is essential for demonstrating how different timings can affect traffic flow and congestion levels.

Users will be able to observe and analyze the impact of different traffic light timings on the overall traffic flow. This feature will help users understand the importance of well-timed traffic lights in reducing congestion and improving efficiency. The GUI will serve as an educational tool for students, traffic engineers, and anyone interested in understanding the fundamentals of traffic management. It will provide insights into the science and art of traffic light control.

2.HARDWARE/SOFTWARE REQUIREMENTS:

SOFTWARE REQUIREMENTS:

- VS-CODE
- JDK-JAVA COMPILER
- SWING MODULE FOR GUI

HARDWARE REQUIREMENTS:

- MINIMUM PROCESSOR OF I3 IS REQUIRED
- DISPLAY
- MOUSE AND KEYBOARD

3.CONCEPTS/WORKING PRINCIPLE

This project is a Java traffic light system project, the primary focus is on simulating the behavior of traffic lights through a graphical user interface (GUI). This typically involves using a GUI library like Swing to create a visual representation of traffic lights. Each traffic light is modeled as an object with states such as "Green," "Yellow," and "Red."

The core logic revolves around implementing methods within the traffic light class to manage state transitions. This includes defining time intervals for each state, creating a timer mechanism (using Java's Timer or ScheduledExecutorService) to control these transitions, and handling user interactions such as button clicks.

Concurrency may come into play if the project aims to simulate multiple traffic lights at different intersections. Threads or concurrency mechanisms can be employed to manage the independent behavior of each traffic light.

User interface components are crucial for user interaction. This involves adding buttons or controls to start, stop, and reset the traffic light system. The current state of the traffic light should be visually displayed on the GUI.

GUI Setup:

- Use a GUI library like Swing or JavaFX to create the graphical interface.
- Represent each traffic light with colored circles or rectangles.

Traffic Light Logic:

- Create a class for the traffic light, which includes states such as "Green," "Yellow," and "Red."
- Implement methods to transition between these states based on a predefined time sequence.

Timer Functionality:

- Utilize Java's Timer or ScheduledExecutorService to manage the timing of state transitions.
- Define time intervals for each state (e.g., Green for 30 seconds, Yellow for 5 seconds, Red for 30 seconds).

Event Handling:

- Implement event handling to respond to button clicks or other user interactions.
- Allow the user to control the traffic light manually or let it run automatically based on the predefined time intervals.

Concurrency (Optional):

- If you want to simulate multiple traffic lights at different intersections, consider using threads or concurrency to manage their independent behavior.

User Interface Interaction:

- Add buttons or controls to start, stop, and reset the traffic light system.
- Display the current state of the traffic light.

Error Handling:

- Implement error handling for invalid state transitions or other

To enhance the project's robustness, error handling mechanisms should be implemented. This includes addressing invalid state transitions or unexpected scenarios that may arise during the simulation.

Finally, thorough documentation and comments in the code are essential. This ensures clarity for anyone reviewing or modifying the code, explaining the purpose of each class, method, and significant code block. By adhering to these principles, a well-structured and functional Java traffic light system project can be developed.

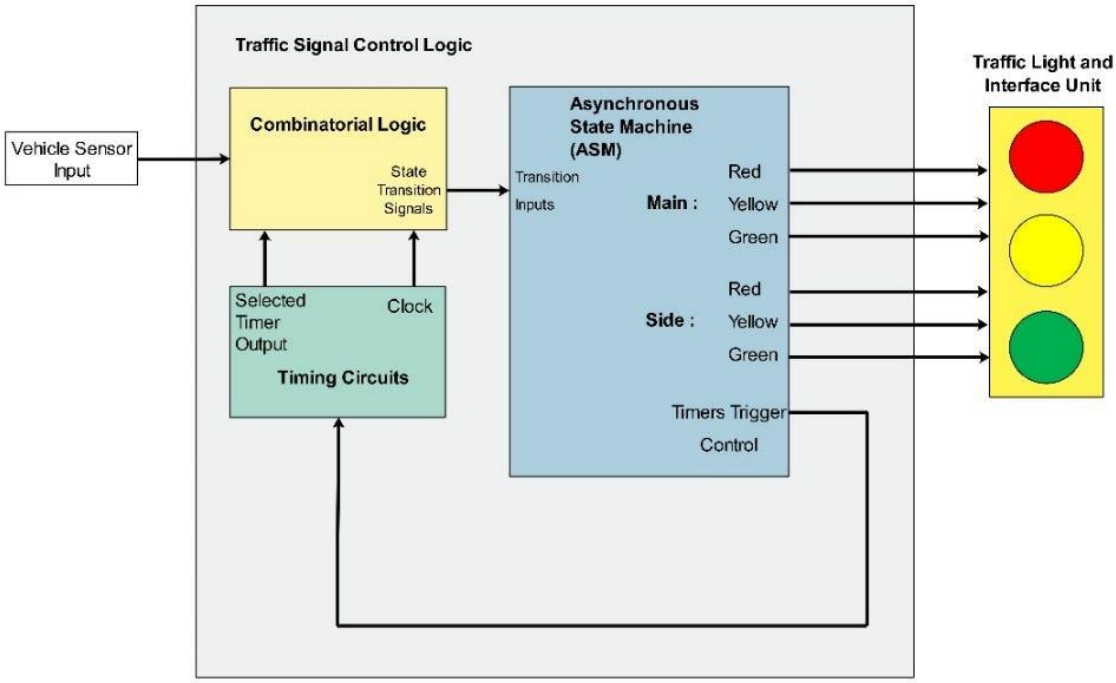
4.APPROACH/METHODOLOGY/PROGRAM:

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

public class TrafficLight extends JFrame implements ActionListener {
    JButton buttonRed, buttonYellow, buttonGreen;
    Signal green = new Signal(Color.green);
    Signal yellow = new Signal(Color.yellow);
    Signal red = new Signal(Color.red);

    Timer lightTimer;
    int lightDuration = 5000; // 5 seconds

    public TrafficLight() {
        super("Java Traffic Light Program");
        getContentPane().setLayout(new GridLayout(2, 1));

        buttonRed = new JButton("Red");
        buttonYellow = new JButton("Yellow");
        buttonGreen = new JButton("Green");

        buttonRed.addActionListener(this);
        buttonYellow.addActionListener(this);
```

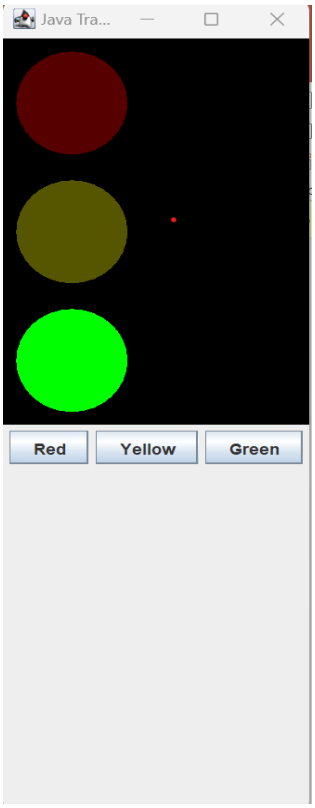
```
        buttonGreen.addActionListener(this);

        green.turnOn(false);
        yellow.turnOn(false);
        red.turnOn(true);

        JPanel trafficPanel = new JPanel(new GridLayout(3, 1));
        trafficPanel.add(red);
        trafficPanel.add(yellow);
        trafficPanel.add(green);

        JPanel lightPanel = new JPanel(new FlowLayout());
        lightPanel.add(buttonRed);
        lightPanel.add(buttonYellow);
        lightPanel.add(buttonGreen);

        getContentPane().add(trafficPanel);
        getContentPane().add(lightPanel);

        pack();

        // Set up the timer to change lights every 5 seconds
        lightTimer = new Timer(lightDuration, this);
        lightTimer.start();
    }

    public static void main(String[] args) {
        TrafficLight trafficLight = new TrafficLight();
        trafficLight.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == lightTimer) {
            // Change lights after 5 seconds
            changeLights();
        } else {
            // Button click event
            changeLights();
        }
    }

    private void changeLights() {
        if (green.isOn()) {
            green.turnOn(false);
            yellow.turnOn(true);
            red.turnOn(false);
        } else if (yellow.isOn()) {
            yellow.turnOn(false);
            red.turnOn(true);
            green.turnOn(false);
```

```
        } else if (red.isOn()) {
            red.turnOn(false);
            green.turnOn(true);
            yellow.turnOn(false);
        }
    }

    class Signal extends JPanel {
        Color on;
        int radius = 40;
        int border = 10;
        boolean change;

        Signal(Color color) {
            on = color;
            change = true;
        }

        public void turnOn(boolean a) {
            change = a;
            repaint();
        }

        public boolean isOn() {
            return change;
        }

        public Dimension getPreferredSize() {
            int size = (radius + border) * 2;
            return new Dimension(size, size);
        }

        public void paintComponent(Graphics graphics) {
            graphics.setColor(Color.black);
            graphics.fillRect(0, 0, getWidth(), getHeight());

            if (change) {
                graphics.setColor(on);
            } else {
                graphics.setColor(on.darker().darker().darker());
            }
            graphics.fillOval(border, border, 2 * radius, 2 * radius);
        }
    }
}
```

5.FLOWCHART:


Traffic Signal Control Logic
Vehicle Sensor Input
Combinatorial Logic
State Transition Signals
Asynchronous State Machine (ASM)
Transition Inputs
Main :
Red
Yellow
Green
Side :
Red
Yellow
Green
Timers Trigger Control
Selected Timer Output
Clock
Timing Circuits
Traffic Light and Interface Unit

6.EXPERIMENT RESULTS & ANALYSIS:



7.CONCLUSIONS:

In conclusion, we have developed a traffic light system that is basic GUI. A Java traffic light system project involves creating a graphical simulation of traffic lights using a GUI library like Swing or JavaFX. The project's foundation lies in modeling traffic lights as objects with distinct states and implementing logic for state transitions. Time intervals, controlled by a timer mechanism, dictate the flow between "Green," "Yellow," and "Red" states.

8.REFERENCES:

online resources for Java programming, GUI libraries (Swing or JavaFX), and concurrent programming in Java. Websites like Oracle's Java Documentation (https://docs.oracle.com/en/java/) and tutorials on platforms like Stack Overflow, GeeksforGeeks, or JavaWorld can be valuable resources for in-depth learning and refer

1