# Virtual Traffic Signal System – Software Documentation

|  |  |
| --- | --- |
| Document Version | 1.0 |
| Author | Shivaraj Kumar, Sohan Tikare, Rakshita |
| Reviewer | Nisha N, Nireeksha |
| Status | In Progress |

## 1. Introduction

### 1.1 Purpose

This document describes the design and implementation of Traffic Signal System simulation software, written in C++. The system controls signal lights for a two-way intersection (South-North and East-West), incorporating normal cycling, pedestrian requests, and emergency vehicle priority.

### 1.2 Scope

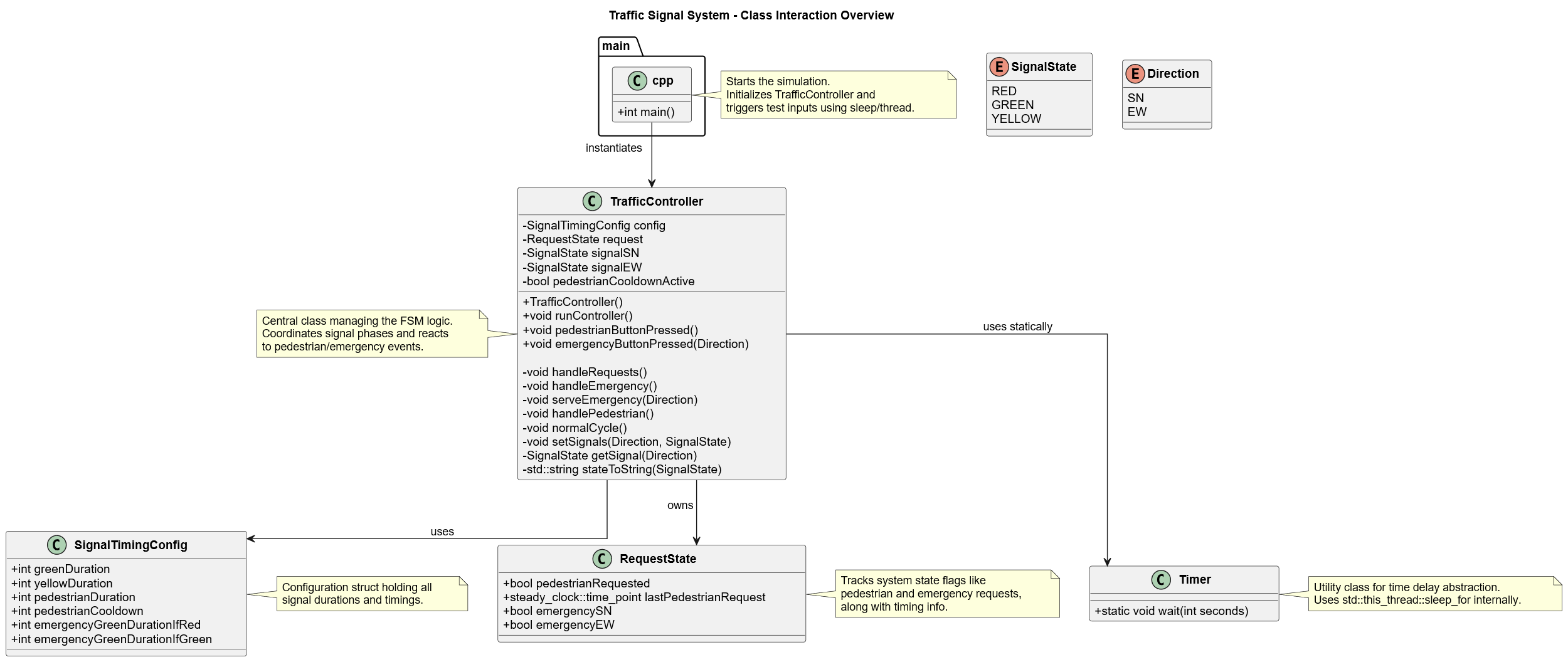
The system emulates a real-time embedded controller managing traffic lights using state machines and configurable timing. It supports external trigger events (pedestrian/emergency requests) and operates continuously in a multithreaded environment.

### 1.3 Terminology

|  |  |
| --- | --- |
| **Term** | **Description** |
| SN | South-North traffic direction |
| EW | East-West traffic direction |
| FSM | Finite State Machine |
| Cooldown | Wait time before accepting next pedestrian request |
| Emergency Mode | Priority logic for incoming emergency vehicle |
| Pedestrian Mode | Both SN and EW set to RED for safe pedestrian crossing |

## 2. System Overview

### 2.1 Architecture Diagram



*Fig 1: Software Architecture*

### 2.2 Module Structure

|  |  |
| --- | --- |
| **Module** | **Description** |
| main.cpp | Initializes and simulates system operation. |
| Timer | Utility to block/wait for N seconds. |
| RequestState | Tracks pedestrian/emergency request states and cooldowns. |
| TrafficController | Core FSM handling normal/emergency/pedestrian cycles. |

## 3. Module Descriptions

### 3.1 Timer Module (Timer.h / Timer.cpp)

Purpose:  
Provide utility functions to simulate delays using std::this\_thread::sleep\_for.

### 3.2 RequestState Module (RequestState.h)

Purpose:  
Maintain pedestrian and emergency request flags and timestamps.

### 3.3 TrafficController Module (TrafficController.h / TrafficController.cpp)

Purpose:  
Implements FSM for managing signal light states, priority overrides, and request handling.

Key Responsibilities:

* State transition for SN/EW signals (RED, GREEN, YELLOW)
* Emergency override logic
* Pedestrian request management with cooldown
* Configurable timing parameters
* Console status outputs

Public Methods:

* void runController();
* void pedestrianButtonPressed();
* void emergencyButtonPressed(Direction dir);

### 5.4 Main Module (main.cpp)

Purpose:  
Initializes TrafficController and simulates real-time button press events.

Actions:

* Launches runController() in a thread
* Simulates user input (pedestrian/emergency) after time intervals

## 4. Signal State Machine Logic

### 4.1Normal Cycle

EW: GREEN → YELLOW → RED → GREEN

SN: GREEN → YELLOW → RED → GREEN

Repeats continuously

### 4.2 Pedestrian Mode

Activated by button press (if cooldown expired)

SN & EW → RED

Pedestrian lights simulated as GREEN

Duration: config.pedestrianDuration

### 4.3 Emergency Mode

Activated by emergency request from SN or EW

If signal already GREEN → Extend green for emergencyGreenDurationIfGreen

If RED → Preempt normal cycle and turn GREEN for emergencyGreenDurationIfRed

Revert back to normal cycle afterward

## 5. Limitations

* No real pedestrian traffic light visualization
* Emergency requests do not interrupt ongoing pedestrian cycle
* No persistent logging

## 6. Future Enhancements

* GUI or WebSocket simulation interface
* Test suite using GoogleTest or Catch2
* Logging subsystem
* Configurable JSON/XML file-based input