Smart City : Networked Control of Autonomous Vehicles

Akash Cuntur Shrinivasmurthy
Embedded Software Engineering
FH Dortmund
Dortmund, Germany
akash.cunturshrinivasmurthy001@stud.fh-dortmund.de

Akhil Narayanaswamy
Embedded Software Engineering
FH Dortmund
Dortmund, Germany
akhil.narayanaswamy001@stud.fh-dortmund.de

Krithika Premkumar
Embedded Software Engineering
FH Dortmund
Dortmund, Germany
krithika.premkumar001@stud.fh-dortmund.de

Madhukar Devendrappa
Embedded Software Engineering
FH Dortmund
Dortmund, Germany
madhukar.devendrappa002@stud.fh-dortmund.de

***Abstract*—This paper presents a comprehensive analysis, design and development of a system within a smart city framework. Our approach was based on three use cases with the primary focus on networked traffic control of autonomous cars. Using SysML, we intricately modeled the system's architecture, behavior and interactions. Furthermore, we executed a segment of the systems state machine and the appropriate scheduling algorithm is used to demonstrate the practical feasibility of our approach thus paving the way for a more efficient and smart transport system.**

Keywords—Smart City, SysML, Scheduling Algorithms, Traffic control.

I. INTRODUCTION

Smart City technologies have substantial potential to improve the quality of urban life. One of the main factors in contributing to the Smart City development is urban mobility. Mobility solutions to orchestrate the movement of vehicles, autonomous vehicles, emergency vehicles, pedestrians, etc in a seamless, safe and reliable manner is key to revolutionizing urban transportation. Our motivation stems from the recognition of the enormous potential of autonomous driving technology in urban mobility. By developing robust networked control mechanisms, we aspire to contribute to the realisation of safe, more efficient and sustainable transport systems, thereby enhancing the quality of life for urban dwellers while addressing important challenges such as congestion, road safety, etc.

Our primary objective is two-fold. Firstly, to identify three use cases in the context of networked traffic control in smart cities and secondly, to analyse, model, design and develop a system meeting our requirements in the identified use cases.

This paper contains many parts of our work as a guide. Further documentation is provided via the GitHub repository.

II. APPROACH

In this section, we outline the steps and the approach followed to engineer our system.

A. Use Case Identification

We begin by selecting three use cases relevant to networked traffic control of autonomous vehicles in smart cities.

B. System Requirement Analysis

Based on the identified use cases, we define the functional and non-functional requirements for our system using the SysML Requirements diagram. These requirements form the foundation for the system design and implementation.

C. SysML Modelling

Utilizing SysML, we set the context by developing a context diagram and then define the system architecture, behavior, constraints and interaction. This involves identifying the key components of our system, modelling their interactions with other components, and defining their behavior using state machines, sequence and activity diagrams.

D. Fault Analysis

In order to ensure that the system is capable of handling failures and is more robust, we identify the possible hazards and sketch a fault tree analysis. We limit the hazards identified to two and conducted a study on the possible root causes and how to handle them in the design.

E. Implementation

A prototype of a part of our system is implemented and integrated in a simulation environment to validate the desired functionality.

F. Evaluation and Testing

To validate and evaluate the implementation, we employ unit testing and peer to peer inspection methods. The defects or improvements identified are updated in the design/implementation.

III. CONCEPT

A. Use Cases

In order to simplify our system, we consider a two-phase signal. This means in the 1st phase, vehicles can move in the direction North to South (aka NS Phase) or South to North and in the second phase, vehicles can move in the East to West direction or West to East direction (aka EW Phase) as shown in Fig. 1. We then identified three use cases pertaining to the smart city context. They are as follows:

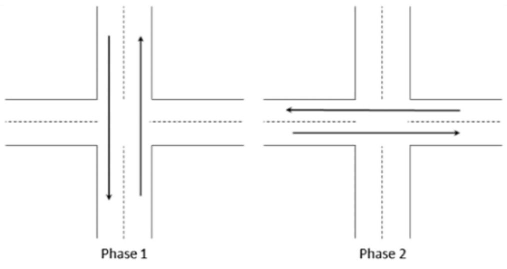


Fig. 1. Two phase intersection

1) *Dynamic Traffic Management*

Traffic congestion is one of the major problems in urban cities today and one way to ensure the smooth flow of traffic and avoiding a pile up is to employ a dynamic traffic signal timing based on the traffic density detected at an intersection. This would not only reduce congestion; it will also lead to better fuel efficiency.

The system continuously monitors the intersection to detect the traffic volume and accordingly adjusts the signal timing until a nominal traffic volume is reached.

2) *Emergency Vehicle Response*

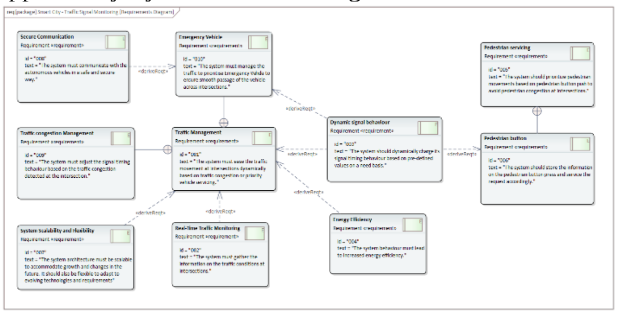
To accommodate emergency vehicles and ensure timely care for the patients, it is vital to ensure rapid and safe movement of vehicles across intersections. The system utilizes GPS and V2X Communication to request priority and communicate the position of the vehicle. This would lead to a pre-emption of the on-going task and service the need of the emergency vehicle. The autonomous vehicles also are indicated about this need by a display of a special SOS Signal in the traffic light display (combination of red and blue color on the traffic display).

3) *Pedestrian Handling*

It is important to consider the pedestrians while designing the traffic signal timing to facilitate ease of their movement and reduce incidents of accidents at intersections. The system detects the presence of pedestrians upon the press of a button switch allocating sufficient time for pedestrian movement. The servicing of pedestrian request by pressing the button must happen at the intersection only after the elapse of a certain time to avoid repeated prioritization of the pedestrian and thereby increasing the traffic at the intersection.

B. *Requirements*

The requirements towards the system, both functional and non-functional, are derived based on the use cases identified. The requirements are listed in the requirements diagram supported by SysML as seen in Fig. 2.



Secure Communication
Requirement «requirement»
id = "008"
text = "The system must communicate with the autonomous vehicles in a safe and secure way."

Emergency Vehicle
Requirement «requirement»
id = "010"
text = "The system must manage the traffic to prioritize Emergency Vehicle to ensure smooth passage of the vehicle across intersections."

Pedestrian servicing
Requirement «requirement»
id = "005"
text = "The system should prioritize pedestrian movements based on pedestrian button push to avoid pedestrian congestion at intersections."

Traffic congestion Management
Requirement «requirement»
id = "009"
text = "The system must adjust the signal timing behaviour based on the traffic congestion detected at the intersection."

Traffic Management
Requirement «requirement»
id = "001"
text = "The system must ease the traffic movement at intersections dynamically based on traffic congestion or priority vehicle servicing."

Dynamic signal behaviour
Requirement «requirement»
id = "003"
text = "The system should dynamically change its signal timing behaviour based on pre-defined values on a need basis."

Pedestrian button
Requirement «requirement»
id = "006"
text = "The system should store the information on the pedestrian button press and service the request accordingly."

System Scalability and Flexibility
Requirement «requirement»
id = "007"
text = "The system architecture must be scalable to accommodate growth and changes in the future. It should also be flexible to adapt to evolving technologies and requirements"

Real-Time Traffic Monitoring
Requirement «requirement»
id = "002"
text = "The system must gather the information on the traffic conditions at intersections."

Energy Efficiency
Requirement «requirement»
id = "004"
text = "The system behaviour must lead to increased energy efficiency."

Fig. 2. Requirements Diagram

The requirements are broken down into finer requirements using the "containment" property of SysML.
Some of the functional requirements identified are:

- The system must ease the traffic movement at intersections dynamically based on the traffic Requirements Diagram
  congestion.
- The system must manage the traffic to prioritize the Emergency Vehicle to ensure smooth passage of the vehicle across intersections.
- The system should prioritize pedestrian movements based on pedestrian button push.

Some of the non-functional requirements identified are:

- The system architecture must be scalable to accommodate further growth and changes in the future.
- The system must ensure increased energy efficiency.

These requirements then form the base for the design and modelling of the system.

C. *Structure Diagram*

In this section, we present the architecture of the system highlighting the key components, their structure and functionality using Context Diagrams, Block Diagrams and Internal Block Diagrams and Parametric Constraint Diagrams.

1) *Context Diagram*

The Context Diagram is illustrated in Fig. 3. In this diagram we present a high-level view of the system boundary and its interfaces with other external elements within the smart city context. Some of the key external entities are Weather, Road condition, Emergency Vehicle, pedestrians, etc.

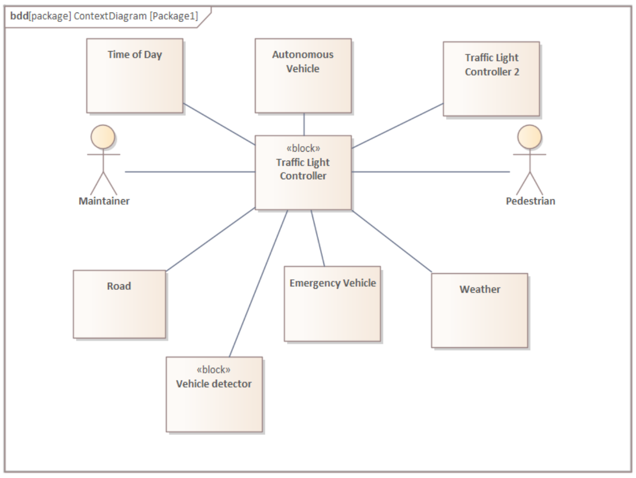


Fig. 3. Context Diagram

2) *Block Diagram*

The block diagram shows all the components involved in the system and the dependencies between the components. The properties, operations and flow properties for each block is mentioned as illustrated in Fig. 4. Some of the main blocks are listed below:

- Block Traffic Signal Controller: Some of the key features of this block is to dynamically update the signal timings based on the traffic density. It also plays an important role in servicing the emergency vehicle priority request and the pedestrian request for passing.
- Block Autonomous Vehicle: Some of the key features of this block is to recognize the traffic light displayed and accordingly accelerate or brake. This block can also request for the time left to change to red from the Traffic Signal Controller block to make a decision to either accelerate or brake.
- Block Emergency Vehicle: Some of the key attributes of this block is to request from block Traffic Signal Controller for priority. This would require the properties such as position of the vehicle in GPS to be communicated to the Traffic Signal Controller block.
- Block Intersection Monitoring: This block monitors the intersection and calculates the density of traffic near the intersection and provides this information to the block Traffic Signal Controller to adjust its signal timing accordingly.

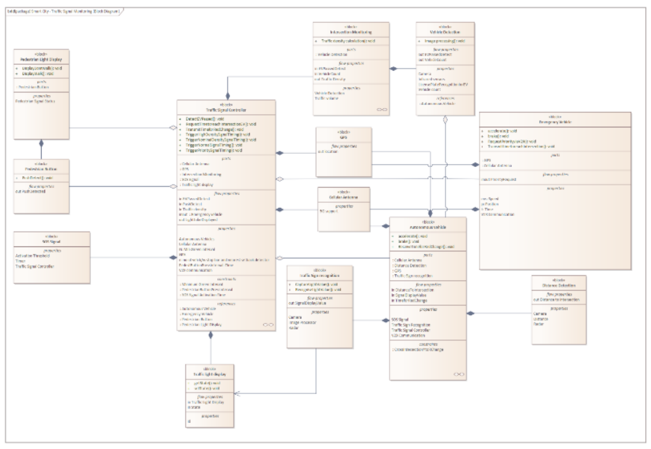
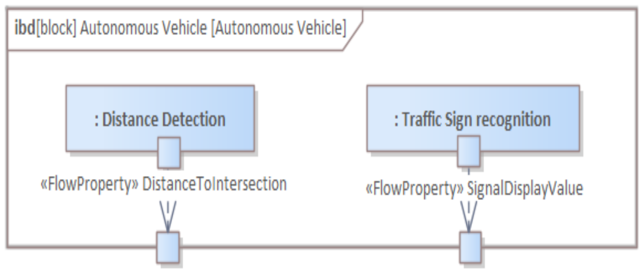


Fig. 4. Block Diagram

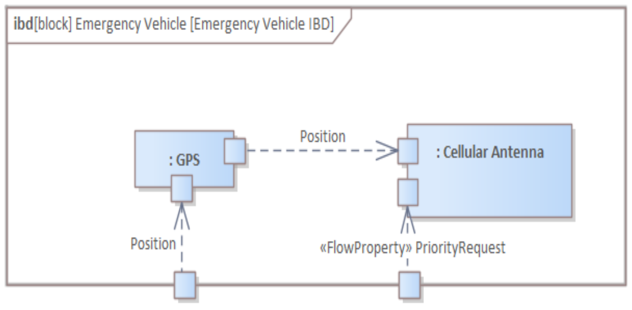
3) *Internal Block Diagram*

An Internal Block Diagram (IBD) is a visual representation within the system that showcases the internal structure of a specific block or component, illustrating how different internal parts collaborate to achieve a block's functionality. It provides a detailed view of the interconnections and interactions between various internal elements, offering insights into the system's behavior and communication patterns.

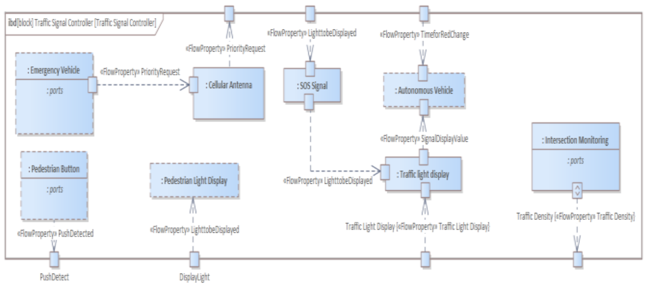
In the Autonomous Vehicle block, interactions with the Distance Detection block enable the reception of crucial information about the vehicle's proximity to the intersection, while collaboration with the Traffic Sign Recognition block facilitates the recognition of displayed signs, contributing to informed decision-making as shown in Fig. 5.


ibd[block] Autonomous Vehicle [Autonomous Vehicle]
: Distance Detection
: Traffic Sign recognition
«FlowProperty» DistanceToIntersection
«FlowProperty» SignalDisplayValue
Fig. 5. IBD of Autonomous Vehicle

The Emergency Vehicle block interfaces with the GPU block to pinpoint the exact location of the traffic signal junction and communicates with the Cellular Antenna block to announce its arrival, ensuring prioritized passage through the junction. It is shown in Fig. 6.


ibd[block] Emergency Vehicle [Emergency Vehicle IBD]
Position
: GPS
: Cellular Antenna
Position
«FlowProperty» PriorityRequest
Fig. 6. IBD of Emergency Vehicle

Additionally, the Traffic Signal Controller block engages with various components, such as Pedestrian Button, Cellular Antenna, Pedestrian Light Display, SOS Signal, Autonomous Vehicle, Traffic Light Display, and Intersection Monitoring, orchestrating a seamless coordination of signals, emergency vehicle prioritization, and real-time traffic data as depicted in Fig. 7.


ibd[block] Traffic Signal Controller [Traffic Signal Controller]
«FlowProperty» LighttobeDisplayed
«FlowProperty» TimeforRedChange
«FlowProperty» PriorityRequest
: Emergency Vehicle
:ports
«FlowProperty» PriorityRequest
: Cellular Antenna
: SOS Signal
: Autonomous Vehicle
«FlowProperty» SignalDisplayValue
: Intersection Monitoring
:ports
: Pedestrian Button
:ports
: Pedestrian Light Display
«FlowProperty» LighttobeDisplayed
: Traffic light display
«FlowProperty» PushDetected
«FlowProperty» LighttobeDisplayed
Traffic Light Display [«FlowProperty» Traffic Light Display]
Traffic Density [«FlowProperty» Traffic Density]
PushDetect
DisplayLight
Fig. 7. IBD of Traffic Signal Controller

These internal block diagrams elucidate the intricate connections among system elements, laying the foundation for efficient traffic signal control and emergency vehicle management in the smart city infrastructure.

4) *Parametric Constraint Diagram*

Parametric Constraint Diagrams play a pivotal role in capturing and visualizing essential system constraints, ensuring the precise functionality and performance of critical components. These diagrams serve as a structured representation of key parameters that influence the behaviour of blocks within the system. Parametric Constraint Diagrams are shown in Fig. 8 and Fig. 9.

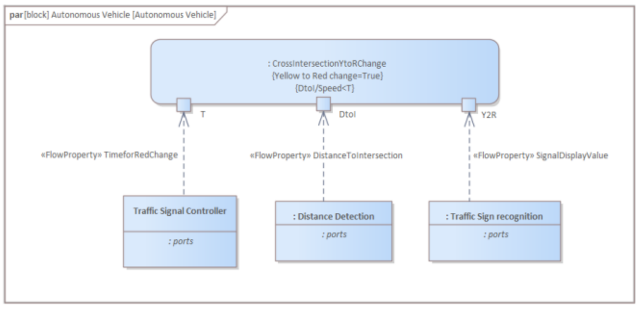

par[block] Autonomous Vehicle [Autonomous Vehicle]
: CrossIntersectionYtoRChange
{Yellow to Red change=True}
{Dtol/Speed<T}
T
Dtol
Y2R
«FlowProperty» TimeforRedChange
«FlowProperty» DistanceToIntersection
«FlowProperty» SignalDisplayValue
Traffic Signal Controller
: ports
: Distance Detection
: ports
: Traffic Sign recognition
: ports

Fig. 8. Parametric Constraints of Autonomous Vehicle

The CrossIntersectionYtoRChange constraint governs the decision the block Autonomous Vehicle has to take inorder to cross an intersection when the current traffic signal state is yellow and the next state would be red. For this, it would require the input of the time left for change to the Red state from the Traffic Signal Controller block and the distance to the intersection from the Distance Detection block.The 'Traffic Signal Controller' block, on the other hand, adheres to constraint like Minimum Green Duration (G), ensuring a minimum value of Green to be designed to ensure enough passage of vehicles to avoid a huge pile up and to accommodate reaction time of vehicles. The minimum Green signal duration based on the number of vehicles in the intersection as given by equation (1).

$$G = 3 + 2n \tag{1}$$

The SOS Signal Activation Time constraint sets a 30-second duration, allowing enough time for the vehicles to pull up to the right lane and the Emergency vehicle to traverse the junction either in the NS or EW phase smoothly.

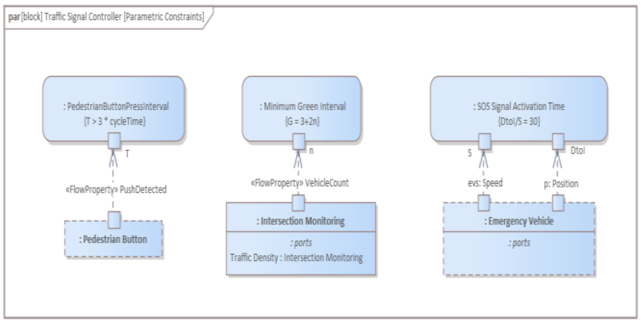

par[block] Traffic Signal Controller [Parametric Constraints]
: PedestrianButtonPressInterval
{T > 3 * cycleTime}
T
«FlowProperty» PushDetected
: Pedestrian Button
: Minimum Green Interval
{G = 3+2n}
n
«FlowProperty» VehicleCount
: Intersection Monitoring
: ports
Traffic Density : Intersection Monitoring
: SOS Signal Activation Time
{Dtol/S = 30}
S
Dtol
evs: Speed
p: Position
: Emergency Vehicle
: ports

Fig. 9. Parametric Constraints of Traffic Signal Controller

Meanwhile, the Traffic Signal Controller block imposes constraints on prioritizing the pedestrian signal only once every three cycles of normal traffic signal operation, striking a balance between pedestrian safety and overall traffic efficiency. These parametric constraints collectively contribute to the robust and reliable operation of the traffic signal monitoring system, aligning with the project's goals of enhancing urban mobility and emergency vehicle passage efficiency.

D. *Behaviour Diagram*

In order to model the behavior of the system we have used the following diagrams.

1) *Activity Diagram*

The activity diagrams modeled for the three use cases are shown in Fig. 10, Fig. 11 and Fig. 12.

In the case of Fig 11, the traffic density value is received from the Intersection Monitoring block. If the density is greater than the threshold value, the extended green timing is used to clear the traffic at the intersection; otherwise the normal signal timing is maintained.

In the case of Fig 10, once the pedestrian presses the button, it is checked, if sufficient time is elapsed before the previous press of the button to avoid reacting on redundant button press as this might cause to continuously stop the traffic flow to service pedestrian.

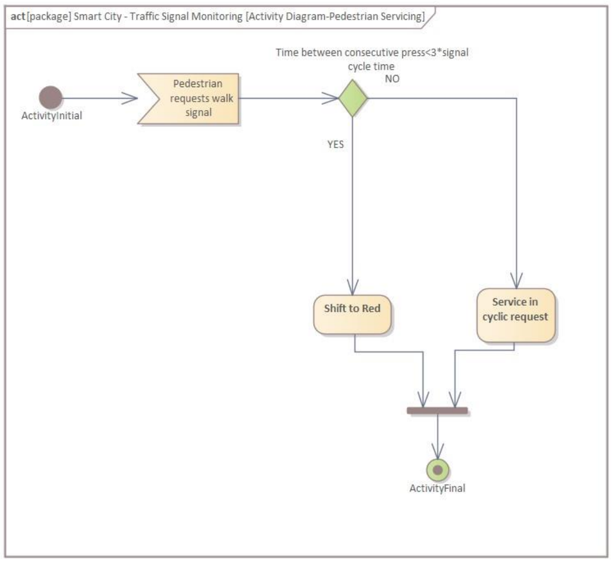

act[package] Smart City - Traffic Signal Monitoring [Activity Diagram-Pedestrian Servicing]
ActivityInitial
Pedestrian requests walk signal
Time between consecutive press<3*signal cycle time
NO
YES
Shift to Red
Service in cyclic request
ActivityFinal

Fig. 10. Activity Diagram for Pedestrian Servicing

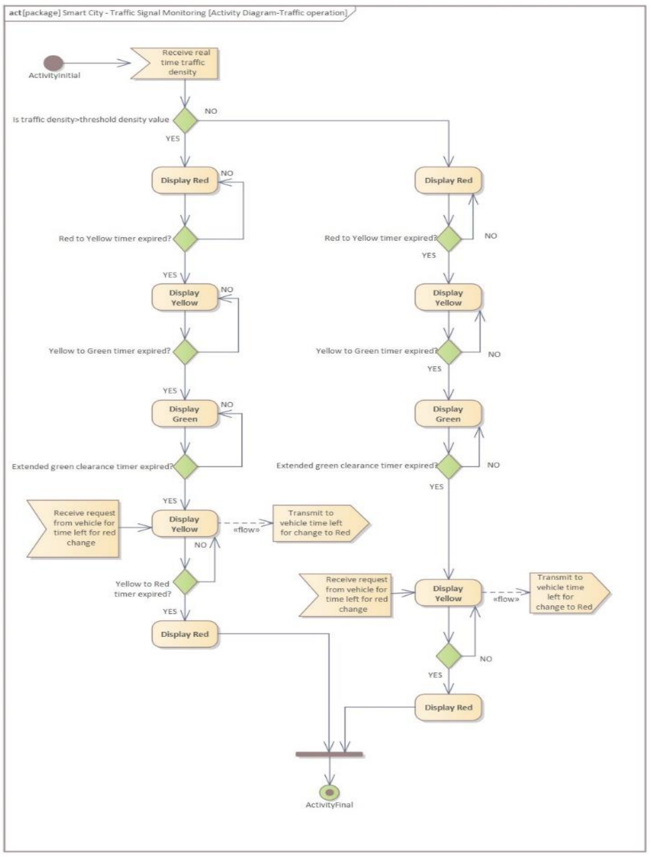

act[package] Smart City - Traffic Signal Monitoring [Activity Diagram-Traffic operation]
ActivityInitial
Receive real time traffic density
Is traffic density>threshold density value
NO
YES
Display Red
Red to Yellow timer expired?
Display Yellow
Yellow to Green timer expired?
Display Green
Extended green clearance timer expired?
Receive request from vehicle for time left for red change
«flow»
Transmit to vehicle time left for change to Red
Yellow to Red timer expired?
Display Red
Display Red
Red to Yellow timer expired?
Display Yellow
Yellow to Green timer expired?
Display Green
Extended green clearance timer expired?
Receive request from vehicle for time left for red change
Display Yellow
«flow»
Transmit to vehicle time left for change to Red
Display Red
ActivityFinal

Fig. 11. Activity Diagram for Traffic Operation

In the case of Fig 12, the Request for Emergency operation is received, and once the other phase has communicated that it is safe to move to Priority operation, a SOS Signal (combination of Red and blue) is displayed. This would direct all the cars towards the right lane, providing an ease of travel for the emergency vehicle. The signal then proceeds to normal operation once the Emergency Vehicle has been detected as cleared from the intersection.

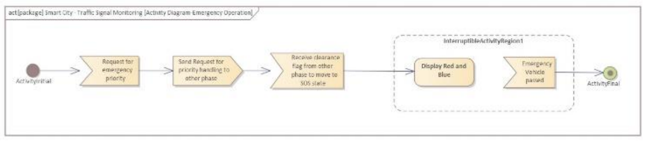

act[package] Smart City - Traffic Signal Monitoring [Activity Diagram Emergency Operation]
ActivityInitial
Request for emergency priority
Send Request for priority handling to other phase
Receive clearance flag from other phase to move to SOS state
InterruptibleActivityRegion1
Display Red and Blue
Emergency Vehicle passed
ActivityFinal

Fig. 12. Activity Diagram for Emergency Operation

2) *Sequence Diagram*

Sequence Diagrams serve as visual representations, illustrating the dynamic interactions and communications among key components such as autonomous vehicles and traffic controllers. These diagrams help to understand the order of events and how everything works together to make smart decisions in a system.

a) Pedestrian servicing

In the context of our Smart City embedded system project, the "Pedestrian Servicing" sequence diagram captures the dynamic interactions among key components - namely, the 'Pedestrian Button, 'Traffic Controller' and 'Pedestrian Sign' This diagram models the intricate process of managing pedestrian signals at junctions as shown in Fig. 13.

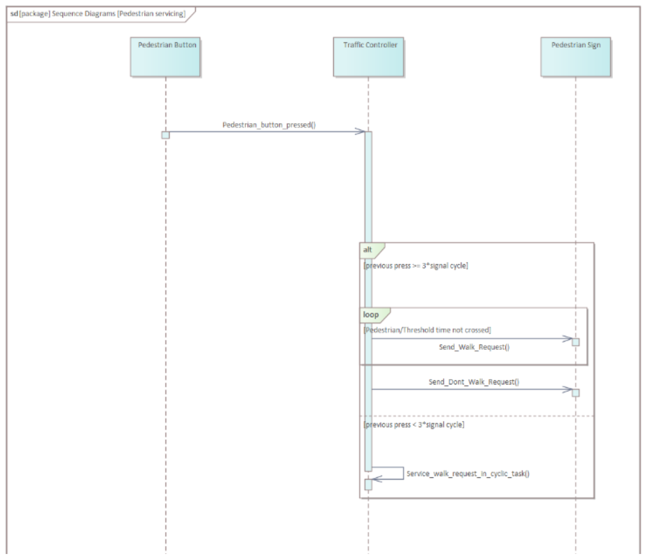

sd [package] Sequence Diagrams [Pedestrian servicing]
Pedestrian Button
Traffic Controller
Pedestrian Sign
Pedestrian_button_pressed()
alt
[previous press >= 3*signal cycle]
loop
[Pedestrian/Threshold time not crossed]
Send_Walk_Request()
Send_Dont_Walk_Request()
[previous press < 3*signal cycle]
Service_walk_request_in_cyclic_task()

Fig. 13. Sequence diagram for Pedestrian servicing

The scenario unfolds as follows:

Upon detecting a pedestrian presence, the 'Pedestrian Button' triggers an event Pedestrian_button_pressed() in the system. The 'Traffic Controller' then evaluates the situation, considering the history of pedestrian signals. If the previous press of the pedestrian button is greater than or equal to three times the signal cycle time, the 'Traffic Controller' initiates a 'Send_Walk_Request()' to the 'Pedestrian Sign'. Subsequently, the 'Pedestrian Sign' responds by displaying the 'Walk' signal. This "Walk" sign continues till the threshold time configured for pedestrians to cross has been reached. Subsequently, the 'Traffic Controller' sends a 'Send_Dont_Walk_Request()' to the 'Pedestrian Sign', indicating that it is not safe for pedestrians to cross.

In the event that the previous press is less than three times the signal cycle, the 'Traffic Controller' facilitates a cyclic walk operation, allowing normal signal cycles to proceed for three cycles.

This sequence of interactions reflects a sophisticated and context-aware pedestrian signal control mechanism, ensuring both the safety of pedestrians and the efficiency of traffic flow. The design not only considers real-time events but also incorporates historical information to make informed decisions. The 'Pedestrian Servicing' sequence diagram encapsulates the complexity of this process, showcasing the seamless integration and coordination of various embedded components within the Smart City infrastructure.

b) Preferential Treatment

The "Preferential Treatment" sequence diagram encapsulates a critical aspect of the Smart City's traffic management system, focusing on the prioritized handling of emergency vehicles. In this orchestrated sequence, the primary actors include the 'Emergency Vehicle', 'Traffic Controller NS(North-South) Phase', 'Traffic Controller EW(East-West) Phase' and 'Autonomous Vehicle'. It is shown in Fig. 14.

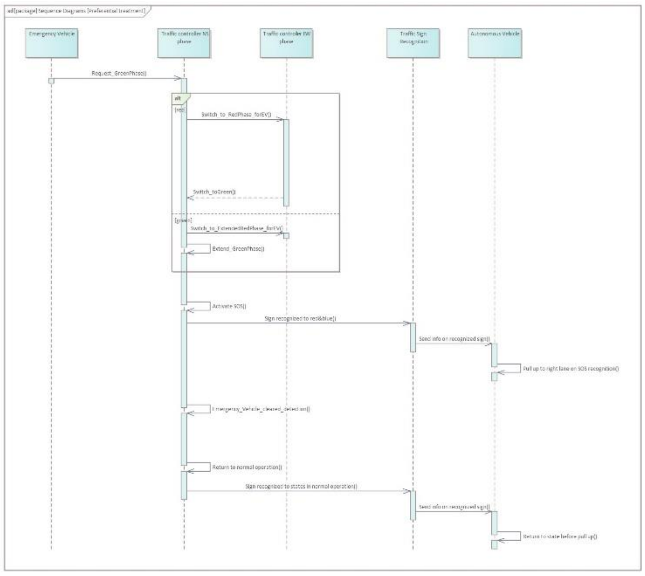

sd [package] Sequence Diagrams [Preferential treatment]
Emergency Vehicle
Traffic controller NS phase
Traffic controller EW phase
Traffic Sign Recognition
Autonomous Vehicle
Request_GreenPhase()
alt
[red]
Switch_to_RedPhase_forEV()
Switch_toGreen()
[green]
Switch_to_ExtendedRedPhase_forEV()
Extend_GreenPhase()
Activate SOS()
Sign recognized to red&blue()
Send info on recognized sign()
Pull up to right lane on SOS recognition()
Emergency_Vehicle_cleared_detection()
Return to normal operation()
Sign recognized to states in normal operation()
Send info on recognized sign()
Return to state before pull up()

Fig. 14. Sequence diagram for Preferential Treatment

The scenario unfolds with the 'Emergency Vehicle' initiating a 'Request_GreenPhase()' to the 'Traffic Controller NS Phase'. If the 'Traffic Controller NS Phase' finds itself in a Red state, it dynamically communicates with the 'Traffic Controller EW Phase' to execute a 'Switch_to_RedPhase_for_EV()' operation, prompting a reciprocal 'Switch_to_Green()' response from the latter. Importantly, if the 'Traffic Controller NS Phase' is already in a Green state, it maintains the status quo by sending 'Switch_to_ExtendedRedPhase_for_EV()' to 'Traffic Controller EW Phase'. Simultaneously, a SOS Signal activation (combination of red and blue) on the NS Traffic display. This is recognized by the traffic sign recognition block and the Autonomous Vehicle then pulls up to the right, facilitating the seamless passage of the emergency vehicle. Following the occurrence of the 'Emergency_vehicle_cleared_detection()' event, the 'Traffic Controller NS Phase' triggers the return to the normal operation which is then recognized by the traffic sign recognition and is then communicated to the 'Autonomous Vehicle', ensuring a smooth transition back to normal traffic operations. Notably, a vice versa process occurs when an emergency vehicle appears at the 'Traffic Controller EW Phase', demonstrating the bidirectional effectiveness of the system in handling emergency scenarios. This well-orchestrated sequence exemplifies the preferential treatment accorded to emergency vehicles, showcasing the adaptability and responsiveness of the system to ensure efficient traffic flow while prioritizing emergency services in critical situations.

c) Vehicle to Traffic Controller

The "Vehicle to Traffic Controller" sequence diagram intricately details the communication dynamics among key components, namely the 'Autonomous Vehicle at NS Signal', 'Traffic Controller NS Phase' and 'Traffic Controller EW Phase' as shown in Fig. 15.

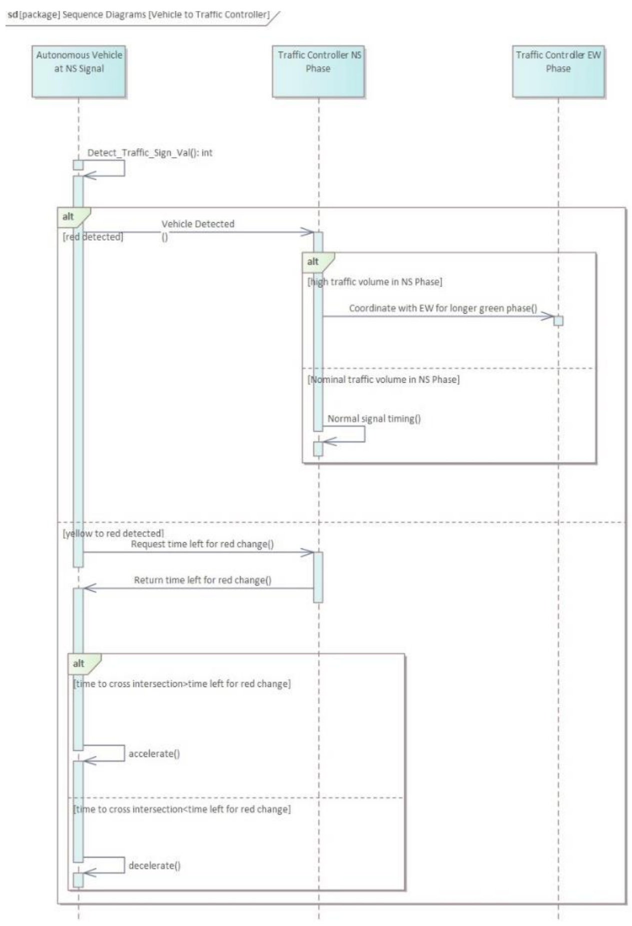

sd [package] Sequence Diagrams [Vehicle to Traffic Controller]
Autonomous Vehicle at NS Signal
Traffic Controller NS Phase
Traffic Controller EW Phase
Detect_Traffic_Sign_Val(): int
alt
[red detected]
Vehicle Detected ()
alt
[high traffic volume in NS Phase]
Coordinate with EW for longer green phase()
[Nominal traffic volume in NS Phase]
Normal signal timing()
[yellow to red detected]
Request time left for red change()
Return time left for red change()
alt
[time to cross intersection>time left for red change]
accelerate()
[time to cross intersection<time left for red change]
decelerate()

Fig. 15. Sequence Diagram for Vehicle to Traffic Controller

The sequence unfolds as the 'Vehicle to Traffic Controller' initiates a self-triggered event, 'Detect_Traffic_Sign_Value()' aimed at evaluating the current traffic sign status. If a red signal is detected, the 'Autonomous Vehicle at NS Signal' is detected by the vehicle detection at the NS phase assessing the traffic density. Concurrently, if the 'Traffic Controller NS Phase' identifies high traffic volume in the NS phase, it engages in collaborative decision-making dispatching 'Coordinate_with_EW_for_longerGreenPhase()' command to the 'Traffic Controller EW Phase'. This collaborative approach extends the green phase to alleviate traffic congestion. In cases of normal traffic volume in the NS phase, the 'Traffic Controller NS Phase' seamlessly proceeds with regular signal operations. A notable interaction occurs during the transition from the Yellow to Red state, where the 'Autonomous Vehicle at NS Signal' actively engages with the 'Traffic Controller NS Phase' by sending a 'Request_time_left_for_Red_change()'. Following the response, the autonomous vehicle dynamically adjusts its speed based on the time left to cross the intersection. If the remaining time exceeds the time left for the red signal change, the vehicle accelerates, otherwise, it decelerates, showcasing an adaptive traffic interaction model. This sequence diagram exemplifies the sophisticated communication and coordination mechanisms employed by autonomous vehicles and traffic controllers, ensuring both traffic efficiency and vehicular safety. A vice versa process unfolds when an autonomous vehicle is stationed at the EW signal, mirroring the adaptability of the system across different traffic scenarios.

3) State Machine Diagram

State machines are modelling techniques used in system design to represent the dynamic behaviour of a system. In the context of traffic control, our state machine diagrams specifically illustrate the detailed states and transitions involved in governing the actions of autonomous vehicles and traffic controllers. These diagrams delve into the sequences of states during normal operation, emergency scenarios triggered by SOS signals, and the adaptive response of the system.

a) Autonomous Vehicle State Machine

The state machine of the autonomous vehicle at a traffic signal junction consists of three states during normal operation: stop, accelerate, and decelerate as shown in Fig. 16. During normal operation, the vehicle enters the stop state upon detecting a red signal. It transitions from the stop state to accelerate if it detects a yellow to green transition, it will be in the acceleration state if green is detected and during the green to yellow transition, if the distance to be covered by the autonomous vehicle to pass the junction is less than the threshold distance, it stays in the acceleration state. However, if the distance is greater than the threshold distance, it transitions to the decelerating state as shown in the Fig. 14.

In the SOS condition, regardless of its state, the autonomous vehicle pulls up to the right lane. When the SOS signal is turned off, it resumes normal operation as shown in the Fig. 10.

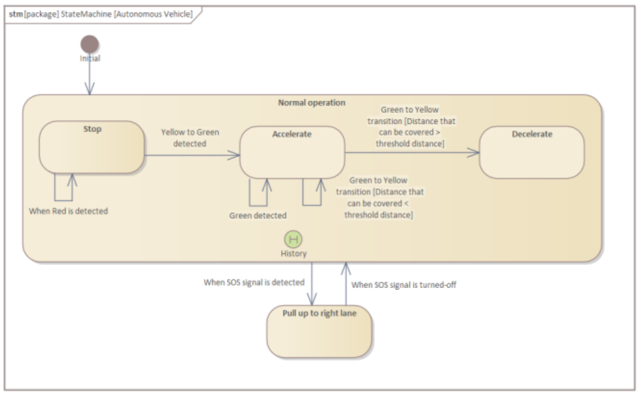

stm [package] StateMachine [Autonomous Vehicle]
Initial
Normal operation
Stop
Yellow to Green detected
Accelerate
Green to Yellow transition [Distance that can be covered > threshold distance]
Decelerate
When Red is detected
Green detected
Green to Yellow transition [Distance that can be covered < threshold distance]
History
When SOS signal is detected
When SOS signal is turned-off
Pull up to right lane

Fig. 16. State Machine Diagram of Autonomous Vehicle

b) Traffic Controller State Machines

For both state machines (NS and EW phases), the normal operation involves transitions from red to yellow, green to yellow, and yellow to red cycling based on a timer.

During an emergency condition, assuming the NS phase receives an Emergency Vehicle Passing Request, it stores the normal operation history and transitions into the emergency operation mode, turning red state. It then prompts other phases to accept the Emergency Vehicle Operation Mode. The traffic controller for the EW phase, upon receiving this request, follows a similar process, storing history and entering the emergency operation mode with a red signal state. Subsequently, the NS phase traffic controller enters the SOS ON state. Upon the emergency vehicle passing the junction, both NS and EW phases revert to the normal operation mode using the previously stored history as shown in the Fig. 17 & Fig. 18.

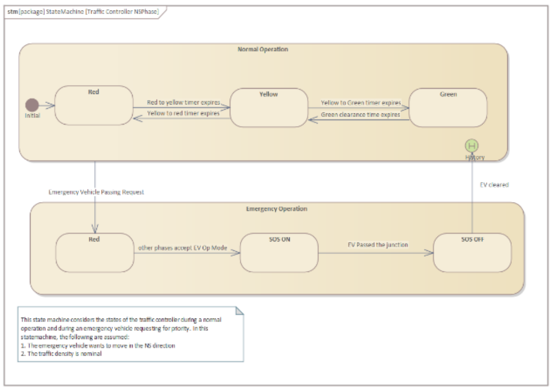

stm [package] StateMachine [Traffic Controller NSPhase]
Normal Operation
Initial
Red
Red to yellow timer expires
Yellow to red timer expires
Yellow
Yellow to Green timer expires
Green clearance time expires
Green
History
Emergency Vehicle Passing Request
EV cleared
Emergency Operation
Red
other phases accept EV Op Mode
SOS ON
EV Passed the junction
SOS OFF
This state machine considers the states of the traffic controller during a normal operation and during an emergency vehicle requesting for priority. In this statemachine, the following are assumed:
1. The emergency vehicle wants to move in the NS direction
2. The traffic density is nominal

Fig. 17. State Machine Diagram of NS Phase Traffic Controller

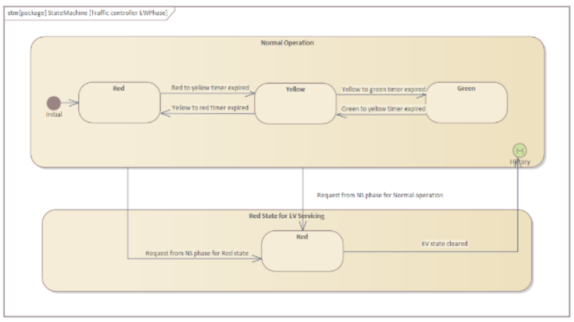

stm [package] StateMachine [Traffic controller EWPhase]
Normal Operation
Initial
Red
Red to yellow timer expired
Yellow to red timer expired
Yellow
Yellow to green timer expired
Green to yellow timer expired
Green
History
Request from NS phase for Normal operation
Red State for EV Servicing
Red
Request from NS phase for Red state
EV state cleared

Fig. 18. State Machine Diagram of EW Phase Traffic Controller

These state machine diagrams provide a detailed understanding of the intricate dynamics involved in managing traffic signals, autonomous vehicles, and emergency scenarios within the Traffic Signal System.

E. Cross-Cutting Diagram

1) Allocation Diagram

Allocation Diagram is a critical tool in system engineering that visually represents the allocation of various activities or functions to specific system components or blocks. It provides a clear mapping of how different tasks are assigned to individual elements within a system architecture, facilitating a comprehensive understanding of resource allocation and responsibility. The allocation of activities to Signal Monitoring is shown in Fig. 19.

The activity 'Pedestrian requests walk signal' finds its allocation to the 'Pedestrian Button' block, signifying that this block is responsible for processing pedestrian requests for walk signals. Furthermore, activities such as 'Shift to Red' and 'Service in cyclic request' are appropriately allocated to the 'Pedestrian Light Display' block, encapsulating the functionalities related to managing pedestrian signals. Similarly, the 'SOS Signal' block takes charge of activities like 'Display Red and Blue' and 'Emergency vehicle passed', aligning with its role in signaling emergency scenarios. The allocation of 'Receive request from vehicle for time left for red change' to both the 'Traffic Signal Controller' and 'Intersection Monitoring' blocks highlights their shared responsibilities in managing requests related to signal timing. Lastly, the 'GPS' block is designated for handling the activity 'Receive real time traffic density', showcasing its role in gathering pertinent information for optimizing traffic control strategies.

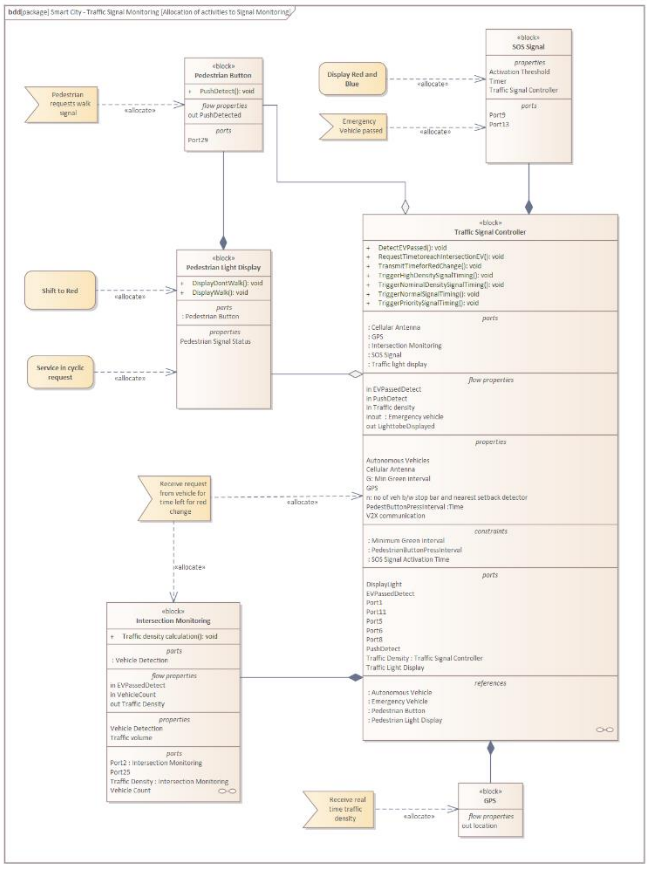

bdd [package] Smart City - Traffic Signal Monitoring [Allocation of activities to Signal Monitoring]
Pedestrian requests walk signal
«allocate»
«block» Pedestrian Button
+ PushDetect(): void
flow properties
out PushDetected
ports
Port29
Display Red and Blue
Emergency Vehicle passed
«block» SOS Signal
properties
Activation Threshold
Timer
Traffic Signal Controller
ports
Port9
Port13
Shift to Red
Service in cyclic request
«block» Pedestrian Light Display
+ DisplayDontWalk(): void
+ DisplayWalk(): void
ports
: Pedestrian Button
properties
Pedestrian Signal Status
«block» Traffic Signal Controller
+ DetectEVPassed(): void
+ RequestTimetoreachIntersectionEV(): void
+ TransmitTimeforRedChange(): void
+ TriggerHighDensitySignalTiming(): void
+ TriggerNominalDensitySignalTiming(): void
+ TriggerNormalSignalTiming(): void
+ TriggerPrioritySignalTiming(): void
ports
: Cellular Antenna
: GPS
: Intersection Monitoring
: SOS Signal
: Traffic light display
flow properties
in EVPassedDetect
in PushDetect
in Traffic density
inout : Emergency vehicle
out LighttobeDisplayed
properties
Autonomous Vehicles
Cellular Antenna
G: Min Green Interval
GPS
n: no of veh b/w stop bar and nearest setback detector
PedestButtonPressInterval :Time
V2X communication
constraints
: Minimum Green Interval
: PedestrianButtonPressInterval
: SOS Signal Activation Time
ports
DisplayLight
EVPassedDetect
Port1
Port11
Port5
Port6
Port8
PushDetect
Traffic Density : Traffic Signal Controller
Traffic Light Display
references
: Autonomous Vehicle
: Emergency Vehicle
: Pedestrian Button
: Pedestrian Light Display
Receive request from vehicle for time left for red change
«block» Intersection Monitoring
+ Traffic density calculation(): void
ports
: Vehicle Detection
flow properties
in EVPassedDetect
in VehicleCount
out Traffic Density
properties
Vehicle Detection
Traffic volume
ports
Port12 : Intersection Monitoring
Port25
Traffic Density : Intersection Monitoring
Vehicle Count
Receive real time traffic density
«block» GPS
flow properties
out location

Fig. 19. Allocation Diagram

IV. IMPLEMENTATION

In response to the evolving dynamics of modern cities, an traffic signal system has been meticulously designed and implemented. With a focus on optimizing traffic flow and ensuring pedestrian safety, the Traffic Signal System represents a paradigm shift in urban planning and traffic control. The report navigates through the various operational modes, emergency conditions, and the synergy between vehicular and pedestrian signals. The implementation of this system, driven by the principles of efficiency, safety, and adaptability, underscores its pivotal role in the future of urban traffic management.

A. Operational Modes

For the implementation, a two-phase traffic signal is considered. To achieve robust and organized control over traffic signals, an object-oriented approach was adopted. The system employs a TrafficSignal class to facilitate seamless management of NS and EW traffic signals. The states of the traffic signals are defined through the TrafficSignalState enumeration. The tasks for NS and EW signals are implemented using FreeRTOS. The traffic control system operates with well-defined states for each traffic signal, with individual task functions (TaskFunction1 and TaskFunction2). These tasks manage normal signal operations, responding to both external user inputs and internal timer. The integration of pedestrian crossing and emergency SOS functionalities through other two tasks enhance the overall robustness and responsiveness of the traffic control system and ensures that the real-time monitoring and feedback on the current state of NS and EW traffic signals are accomplished. Fig. 20 and Fig. 21 illustrate the nomenclature considered for Vehicle and Pedestrian movement respectively.

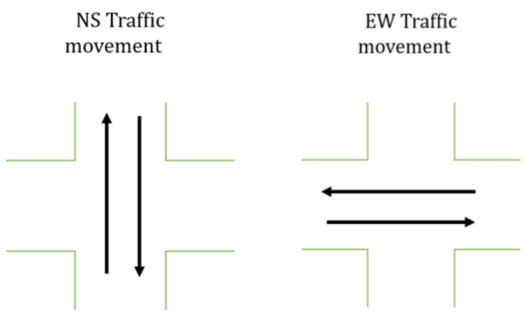


Fig. 20. Nomenclature considered for Vehicle movement

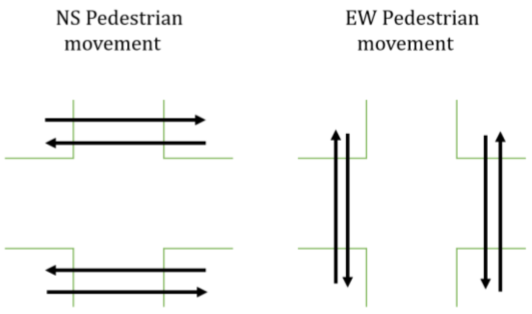


Fig. 21. Nomenclature considered for Pedestrian movement

1) Normal Traffic Signal Operation

In the normal operation of the traffic signal system, both North-South (NS) and East-West (EW) traffic lights follow a synchronized cycle. The green signal remains active for 26 seconds, facilitating smooth vehicular movement. This is followed by a 2-second yellow signal, signalling the impending transition. The red signal is then displayed for 31 seconds, ensuring the safety of all junction users. An additional 1-second yellow signal separates the end of the red signal and the beginning of the next green phase. Pedestrian signals complement the vehicular flow, with a green signal lasting for 27 seconds, followed by a red signal as shown in the Fig. 22.

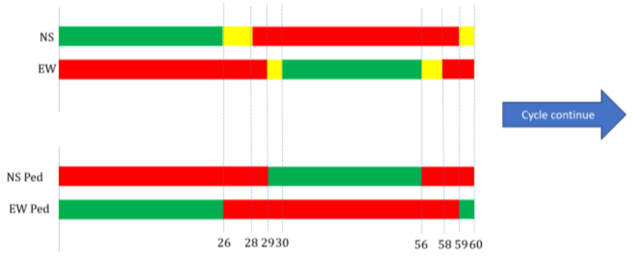


Fig. 22. Timing diagram of States during Normal Operation

2) Emergency Conditions

The system is equipped to handle emergency conditions triggered by the activation of the SOS button. If the SOS button is pressed at NS, the blue LED turns on, indicating an emergency. The NS traffic signal turns red for both vehicle movement and pedestrian crossing, prioritizing the emergency vehicle's passage. Simultaneously, the EW traffic signal transitions to a specific emergency mode, but pedestrian crossing will be green.

Similarly, if the SOS button is pressed at EW, the blue LED signals the emergency, turning the EW traffic signal red for both vehicle movement and pedestrian crossing. The NS traffic signal adjusts to facilitate the emergency vehicle's movement, but pedestrian crossing being green.

If SOS is pressed at both NS and EW simultaneously, the system prioritizes the button pressed first, implementing emergency measures accordingly. In the event of an emergency alert, the current state of the traffic system is stored as history, and the SOS condition remains activated until the emergency vehicle passes, or the emergency alert is turned off. Once resolved, the traffic system retrieves the stored history, seamlessly continuing from the point it left as shown in the Fig. 23.

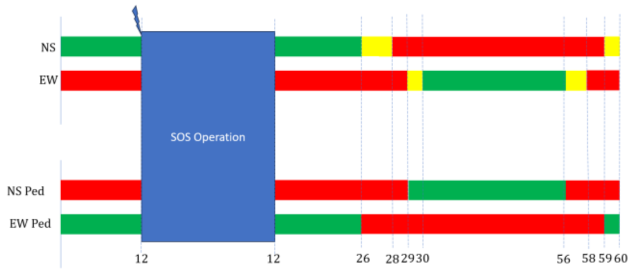


Fig. 23. Timing diagram of States during Emergency Conditions

3) Pedestrian Signal Operation

The pedestrian signal operation is intricately designed to prioritize pedestrian safety and facilitate an efficient crossing.

- Buffer Time Before Activation
  The pedestrian signal delays turning green for 1 second after the vehicle signal turns red. This brief buffer time allows vehicles to clear the junction before pedestrians start crossing.
- Buffer Time After Activation
  After the pedestrian signal is activated, it remains green for a maximum of 24 seconds, providing a 3-second buffer time. This ensures pedestrians complete their crossing before the vehicle signal turns yellow.
- Pedestrian Button Interaction

Pedestrian buttons empower pedestrians to request an early crossing. When the pedestrian button is pressed, the green signal for vehicles is shortened by 5 seconds in normal operation, advancing the pedestrian green signal by 5 seconds in that cycle, resulting in a 5-second reduction in that cycle, as shown in Fig 24. If the pedestrian button is pressed at both NS and EW phases in the same cycle, a cumulative reduction of 10 seconds occurs in that cycle, as shown in Fig. 24. The pedestrian button remains active for the initial 10 seconds of the vehicular traffic signal turning green, granting pedestrians 5-second early access to the green signal for crossing the road. Subsequently, the pedestrian button deactivates for the next three normal traffic light cycles, reactivating after the third cycle. This feature significantly improves both safety and traffic flow, optimizing conditions for pedestrian crossing.

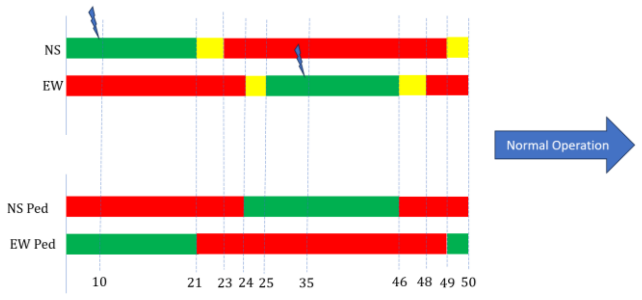


Fig. 24. Timing diagram of States if the Pedestrian Button is pressed at both NS and EW Phases

B. *Code Implementation*

The code for the Traffic Control System is developed in Arduin o C++ using the FreeRTOS library. The code includes tasks for NS and EW traffic signal operation, pedestrian signal operation, and emergency SOS handling. The SOS button takes precedence over other operations, ensuring a swift response to emergencies.

The implemented tasks include:

- NS and EW traffic signal tasks with normal and emergency phases.
- Pedestrian signal task with buffer times for pedestrian safety.
- Emergency SOS task handling

Additionally, the code embodies modularity and scalability, allowing for easy integration of future functionalities. The use of coding best practices enhances maintainability and facilitates potential updates or expansions. Real-time monitoring provide insights into system performance, aiding in debugging and optimization efforts.

C. *Simulation*

The system's code has been implemented and simulated using a platform called Wokwi, providing a virtual environment to test and validate the system's functionalities accurately.

The system utilizes Arduino and FreeRTOS for task scheduling. The hardware components include LEDs for traffic signals, pedestrian signals, and emergency indication, as well as buttons for pedestrian crossing and SOS activation as shown in the Fig. 25.

Clicking on Fig. 25 enables the simulation of the traffic light system. Within the simulation environment, activation of the pedestrian button functionality is achieved by pressing a push button. A simple press on this button initiates the activation, facilitating the simulation of pedestrian crossings. Furthermore, the SOS activation is simulated through a sliding switch. Interacting with this switch triggers emergency conditions, allowing observation of the system's response in prioritizing emergency vehicles.

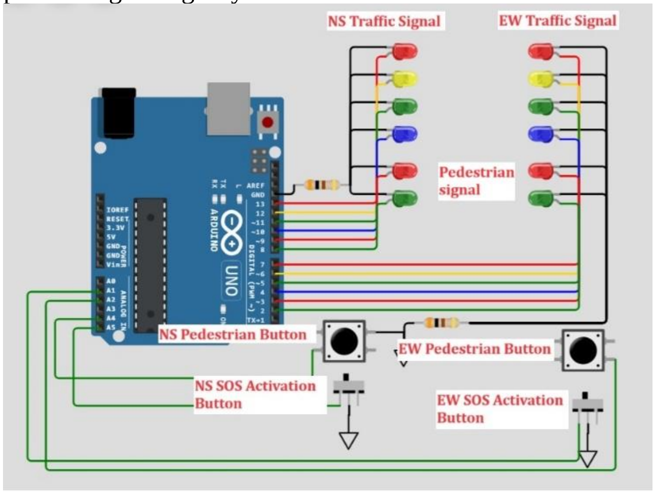


Fig. 25. Traffic Signal implemented on Wokwi

D. *Future Scope*

The implemented system, focused on a two-phase traffic signal model, lays the groundwork for potential advancements. There is scope for extending a similar traffic signal system to accommodate four-phase traffic signals, particularly in bustling junctions with complex traffic dynamics. The adaptability of the current system serves as a solid foundation, facilitating scalability to handle the intricacies of multi-phase traffic scenarios. In addition to the potential extension to four-phase traffic signals, the Traffic Signal System can evolve to incorporate advanced functionalities catering to dynamic traffic scenarios. Implementation possibilities include a dynamic traffic light system where busy lanes or sides receive a higher proportion of green signals based on real-time traffic conditions, optimizing overall traffic flow. Furthermore, the system could explore a night mode feature, adjusting the duty cycle of normal signal operations during the night. This adaptive approach aims to minimize waiting times during low-traffic periods at night, enhancing the efficiency of traffic signal management. These envisioned functionalities align with the system's adaptive architecture, showcasing its capacity for continual enhancement and customization to meet the evolving demands of urban traffic control.

E. *Practical Deployment*

The main actors in our model are the Traffic Light Display, Traffic Signal controller, Pedestrians, Autonomous vehicles, and Emergency vehicles. Focusing on the practical deployment of the model we went with a decentralized system architecture. This design comprises a main controller board, complemented by a separate controller responsible for displaying the state and processing inputs. This decentralization significantly diverges from traditional traffic control systems that rely on a single microcontroller for both display and input processing tasks. Moreover, this approach significantly enhances the overall system reliability. In a decentralized architecture, the failure of one component does not lead to a complete system shutdown. Components can be serviced or upgraded individually, minimizing the impact on the entire traffic control system. For this implementation, we will be detailing the setup and functionality of the separate controller, designated as Traffic Signal Controller. This component is pivotal for sending the state to be displayed to the Traffic Light Display block and processing various inputs, a cornerstone of our decentralized approach to traffic control systems.

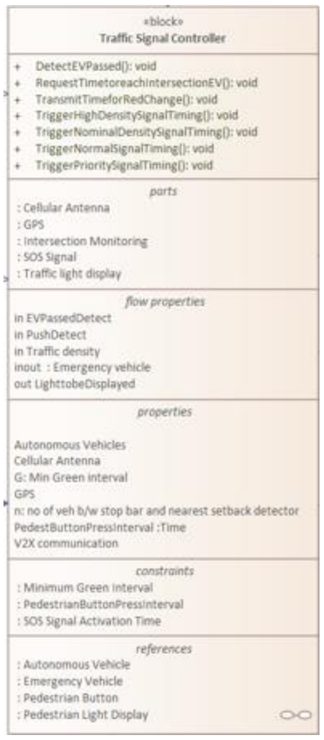

«block»
Traffic Signal Controller
+ DetectEVPassed(): void
+ RequestTimetoreachIntersectionEV(): void
+ TransmitTimeforRedChange(): void
+ TriggerHighDensitySignalTiming(): void
+ TriggerNominalDensitySignalTiming(): void
+ TriggerNormalSignalTiming(): void
+ TriggerPrioritySignalTiming(): void
parts
: Cellular Antenna
: GPS
: Intersection Monitoring
: SOS Signal
: Traffic light display
flow properties
in EVPassedDetect
in PushDetect
in Traffic density
inout : Emergency vehicle
out LighttobeDisplayed
properties
Autonomous Vehicles
Cellular Antenna
G: Min Green interval
GPS
n: no of veh b/w stop bar and nearest setback detector
PedestButtonPressInterval :Time
V2X communication
constraints
: Minimum Green Interval
: PedestrianButtonPressInterval
: SOS Signal Activation Time
references
: Autonomous Vehicle
: Emergency Vehicle
: Pedestrian Button
: Pedestrian Light Display

Fig. 26. Traffic Signal Controller Block

1) *Calculation*

The Arduino Uno R3 is selected for implementation of the Traffic Signal Controller shown in Fig. 26, this choice is motivated by the Arduino's versatility and capability to handle real-time operations effectively, especially when coupled with FreeRTOS, a real-time operating system for micro controllers. The primary responsibility of the Traffic Signal controller is,

- Signal Update Monitoring: Its first task is to continuously monitor for updates from the main microcontroller. This task ensures that the traffic signal displays are in sync with the traffic control system. To maintain timely updates without overburdening the system, Task1 is set to run once every second (1000ms). This frequency strikes a balance between real-time responsiveness and system efficiency.
- Pedestrian Button Press Detection: The second task involves listening for pedestrian button presses. Given the critical nature of pedestrian safety, this task is assigned a higher priority. It is designed to run 10 times per second (every 100ms), ensuring that pedestrian requests are promptly addressed, thereby enhancing the safety and user experience at pedestrian crossings.
- Emergency Signal Listening: The third task is to listen for emergency signals from vehicles such as ambulances and fire trucks. Similar to Task2, this task is also of high priority due to its implications for public safety and emergency response efficiency. It runs at the same frequency as Task2, i.e., every 100ms, to ensure that emergency vehicles are given immediate right-of-way, reducing response times in critical situations.

In deploying these tasks within the Traffic Signal controller, we have meticulously tailored their operational frequencies to align with the system's requirements. To calculate the WCET for these tasks, I have adopted a method that estimates the number of cycles required to execute each task and then divides it by the Arduino Uno's clock frequency of 16MHz.

$$WCET = \frac{Number\ of\ Cycles}{Clock\ Frequency} \tag{3}$$

This process involves considering various factors like GPIO delays, context switching, interrupt handling, and task management overheads. For instance, I assume that each FreeRTOS task consumes approximately 320,000 instructions for its initiation and termination, and each line of code incurs around 1,000 instructions. This method, while crude, provides a practical approach to estimating the execution time for each task, serving as a crucial step in ensuring the Traffic Signal's efficient operation within the larger traffic control system. Continuing with the WCET calculation for each task, the detailed analysis of the code for the Traffic Signal controller reveals the following metrics

- Task 1: With a maximum of 23 lines of code, this task accumulates approximately 23,000 instructions (considering 1,000 instructions per line).
- Task 2: This task, comprising 7 lines, translates to 7,000 instructions.
- Task 3: Similarly, with 6 lines of code, this task involves 6,000 instructions.

Using the Equation (3) we divide these instruction counts by the Arduino Uno's clock frequency of 16MHz, we derive the WCET for each task. Task 1 takes 21.4ms, Task 2 takes 20.4ms and finally, Task 3 takes up 20.3ms to execute in the worst-case scenario. These calculations form the basis for understanding each task's time requirements under worst-case scenarios. The constraint table, TABLE I. outlines the period, WCET, and the deadline for each task, with the deadline being considered equal to the period of the task. This approach allows for a clear and concise representation of the task parameters, facilitating a better understanding of the system's timing constraints and operational efficiency.

TABLE I. SCHEDULING CONSTRAINTS

Tasks	Period (ms)	WCET (ms)	Deadline (ms)
Task1	1000	21.4	1000
Task2	100	20.4	100
Task3	100	20.3	100

2) *Simulation*

The simulation setup for the Traffic Signal controller has been meticulously designed on the Wokwi online simulator, utilizing the Arduino Uno R3 as seen in Fig. 27. This setup includes six LEDs: three for the primary traffic signals representing the 'stop', 'ready', and 'go' states—and an additional blue LED for the SOS state. The remaining two LEDs, one red and one green, serve to control and indicate the stop and go states for pedestrian traffic, respectively. User interaction is facilitated through a push button acting as the pedestrian signal and a toggle switch that represents the emergency signal input. This hardware configuration offers a comprehensive representation of an actual traffic signal controller, allowing for a realistic simulation of its functionality and behavior in a controlled virtual environment.

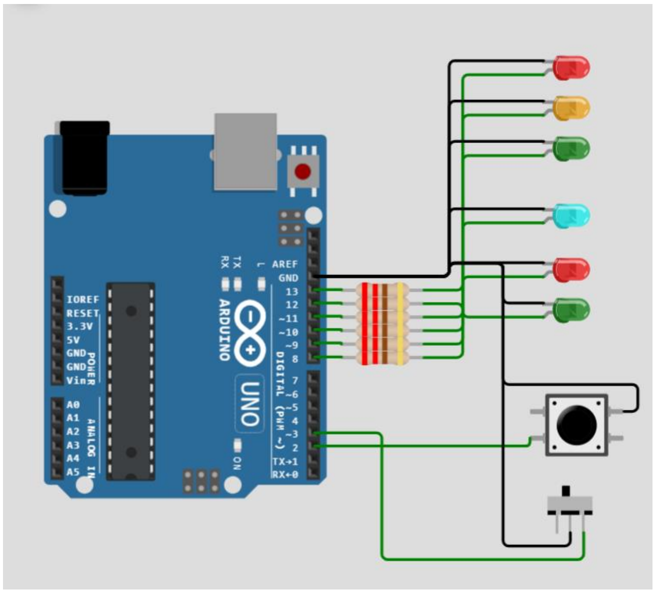


Fig. 27. Traffic Signal block implemented on Wokwi.

xTaskGetTickCount function from FreeRTOS within the Wokwi simulator to ascertain the actual tick count. However, due to the nature of the simulation environment, the tick rate was consistently returned as zero. This limitation prompted the use of the SimSo tool for schedulability analysis.

The constraint diagram, which will be illustrated in TABLE I. served as the basis for the simulation. The subsequent analysis confirmed that the tasks are schedulable under both Earliest Deadline First (EDF) and Rate Monotonic Scheduling (RMS) policies. In case of the Rate monotonic scheduling the utilization rate is 0.6210 for our model. Since the utilization rate is less than 0.6930 our implemented model is schedulable in RMS policy. The Fig. 28 depicts the timing diagram of the model being scheduled in RMS by SimSo simulator.

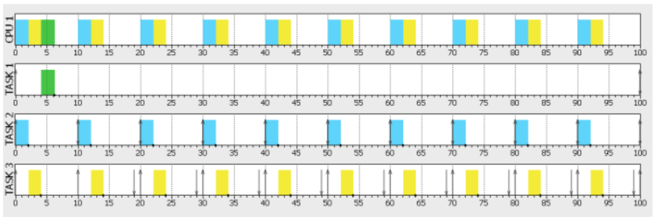


Fig. 28. Monotonic Scheduling of tasks by SimSo.

For Earliest Deadline First Scheduling policy the utilization rate from the simulation was also 0.6210 which is less than 1. Hence the tasks are also schedulable considering that their deadline is same as their period. The Fig. 29depicts the timing diagram of the model being scheduled in EDF by SimSo simulator.

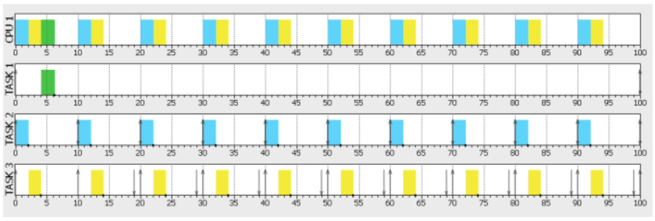


Fig. 29. Earliest Deadline First Scheduling of Tasks by SimSo

V. FAULT ANALYSIS

Fault tree analysis is a top-down approach for deriving failure nodes that in combination with other failures can result in the occurrence of the top event.

We identified two hazards in the system and derived the fault tree analysis of the possible failures that could lead to the events identified.

A. *Vehicle crossing a red light*

This hazard occurs when a vehicle fails to adhere to the traffic signal indications and crosses an intersection during a red light. This could lead to the risk of collisions and accidents at intersections endangering the safety of the vehicle and other road users and can cause property damage, injuries or fatalities.

The potential root causes could be the distance to cross the intersection is underestimated because of sensor failures or communication issues in receiving the correct value for the time from yellow to red change.

Some of the ways to mitigate this failure could be redundancy or driver intervention. Having redundant communication channels ensures, in case of failure in reception of a critical message, the same information can be obtained from the secondary channel. In case of system unavailability due to sensor failures, the same must be displayed in the Human Machine Interface and driver intervention must be requested. It is shown in Fig. 30

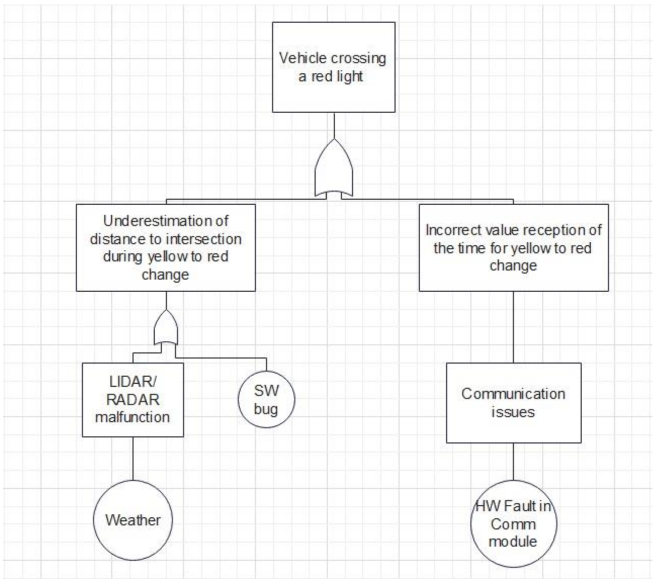

Vehicle crossing a red light
Underestimation of distance to intersection during yellow to red change
Incorrect value reception of the time for yellow to red change
LIDAR/ RADAR malfunction
SW bug
Communication issues
Weather
HW Fault in Comm module

Fig. 30. Fault tree Analysis for Hazard Vehicle crossing red light

B. *Traffic Light Misidentification*

This hazard occurs when a vehicle incorrectly mis identifies the light of the traffic signal which can lead to the risk of collisions, fatalities or property damage.

The potential root causes could be issues with the hardware such as camera quality issues caused by the weather or distorted lens or software issues like software defects.

Some of the ways to mitigate this failure could be in case of unclear weather conditions, the system can request for the traffic light information from the traffic signal controller using V2I Communication. In case of function unavailability such as traffic sign recognition due to camera issues, the same must be displayed on the Human Machine Interface and driver intervention must be requested as depicted in Fig. 31.

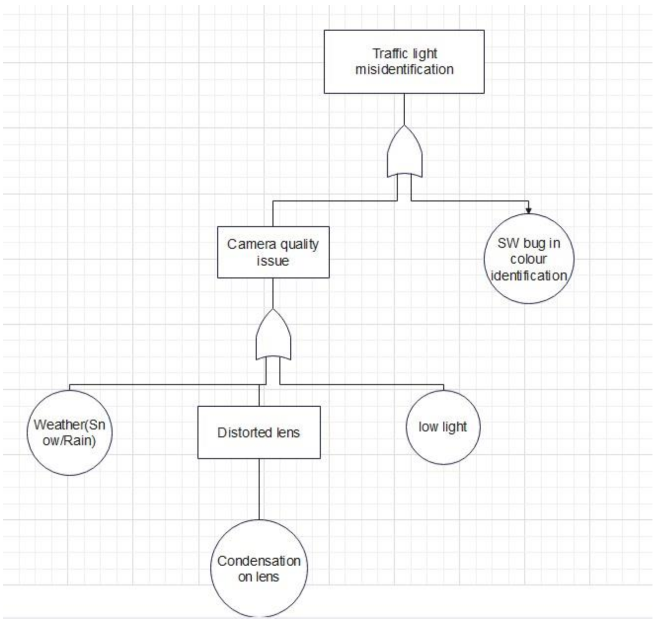


Fig. 31. Fault tree analysis for the hazard traffic light misidentification

VI. EVALUATION AND TESTING

A. *Unit Testing*

Testing acts as a quality assurance measure, ensuring that each component functions as intended. In our traffic signal control system, unit testing is especially crucial, as it verifies the reliability and accuracy of each discrete functionality within the broader application. To implement this, we utilized DocTest, a C++ testing framework that allows for the seamless integration of test cases with the code they are testing.

In our approach, we crafted a comprehensive suite of test cases that covered a wide array of scenarios. The Traffic Signal Test Suite focused on the various states of the traffic signals, rigorously checking for both standard and edge-case behaviours to ensure reliability under all possible real-world conditions. Meanwhile, the Scheduler Test Suite examined the timing aspects, ensuring that signals changed at the correct intervals, which is vital for maintaining traffic flow and pedestrian safety. The Emergency Test Suite tested the system's responsiveness to urgent scenarios, validating that emergency protocols were triggered correctly and promptly.

Through meticulous testing, we identified several areas for improvement. These included defects related to signal identification, scheduling logic, and state transitions, each critical to the system's consistent operation. Addressing these issues head-on, we employed a test-driven approach to refine the Traffic Signal block, resulting in a more robust and reliable system.

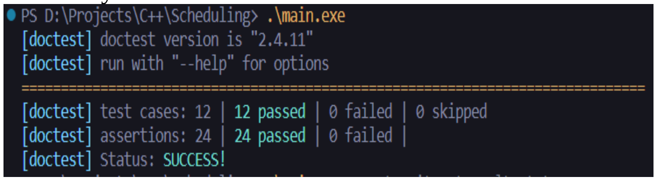

```
PS D:\Projects\C++\Scheduling> .\main.exe
[doctest] doctest version is "2.4.11"
[doctest] run with "--help" for options
===============================================================================
[doctest] test cases: 12 | 12 passed | 0 failed | 0 skipped
[doctest] assertions: 24 | 24 passed | 0 failed |
[doctest] Status: SUCCESS!
```

Fig. 32. DocTest result

The successful conclusion of the testing phase is visually evidenced by the DocTest output in Fig. 32, which will be included in the documentation. This image will serve as a testament to the thoroughness of our testing process, with all 12 test cases passing, underscoring the system's readiness for deployment. The results from DocTest not only validate the functional soundness of our implementation but also reflect our rigorous commitment to quality and performance in traffic signal control systems.

B. *Inspection*

Peer to peer inspection was done to identify defects and improve the overall system quality. Clear objectives and roles and responsibilities were defined for the participants. An inspection document was created so that the inspectors and the authors can record their findings or address the findings respectively.

Each team member selected the document/content to be inspected and documented their findings in the inspection document. The defects and improvements identified were then addressed by the author and was tracked either by a update in the document or an explanation for the rationale behind the finding.

1) Inspection of the Internal Block Diagram and Code

The internal block diagram and the code were inspected and the findings recorded. Some of the findings were:

- Some of the blocks which are encapsulated by the block Traffic Signal Controller was not mentioned in the internal block diagram.
- Sufficient time must be allocated for the pedestrians to consider different walking patterns.
- Buffer time must be allocated once the pedestrian signal goes from green to red and the corresponding traffic signal goes from Red to Yellow to Green.

2) Inspection of the State Machine Diagram and Simulation

The state machine diagram and the simulation were inspected and the findings are recorded. The findings were:

- Some functionalities which are implemented in simulation was not included in the state machine diagrams.
- Maintaining the history to record the previous state of traffic signal after emergency vehicle passing is updated in state machine diagram.
- Pedestrian interrupt before the threshold time altering the green signal duration is inspected.

3) Inspection of the Activity Diagram and Sequence Diagram

The Activity diagram and the Sequence Diagram were inspected, and the findings were recorded. Some notable observations include:

- Considering both the initiation time limit and the previous press cycle are essential for the activation of the pedestrian Button.
- The use of red and blue signals to alert emergency vehicle passing in a specific lane. This, in turn, notifies autonomous vehicle to pullup to right lane.
- During the passage of an emergency vehicle in a specific phase, the signal for pedestrians in another phase could be green. This consideration is particularly relevant in the context of the implemented two-phase signal system.

4) Inspection of Fault Tree Analysis

The Fault Tree Analysis identified two primary hazards in the system.

- It examined potential causes and solutions for vehicles crossing red lights, including sensor failures and communication issues.
- The analysis also explored causes and mitigation for traffic light misidentification, such as hardware issues and software defects.
- A detailed study of root causes for the identified hazards was conducted.
- Effective mitigation strategies were proposed to enhance system safety and reliability.

VII. SUMMARY

The Traffic Control System presented in this paper demonstrates effective traffic management of two-phase traffic signal model through task scheduling and logic. By incorporating features such as emergency response, pedestrian prioritization and their safety the system contributes to safer and more efficient urban traffic flow. As cities continue to grow, embracing smart solutions becomes imperative, and the Traffic Signal System emerges as a beacon for the future of urban mobility.

A traffic signal controller system in a smart city context was analysed and the main use cases were identified and the requirements were derived for the same. These requirements were used as a base for further developing our system. SysML Modelling techniques were used through out the analysis and design phase. The behavioural and structural diagrams helped visualise the architecture better paving the way for seamless improvements and integration of changes. Through-out the software engineering life cycle, peer inspection was performed and continuous improvement and optimisation of the software architecture and requirements were done.

The possible failures that could occur in the system was analysed and design changes to mitigate such failures were proposed. The system was partially implemented and scheduling algorithms were used to show case some of the main features of an embedded system being met i.e., reliability and dependability.

VIII. REFERENCES

[1] National Cooperative Highway Research Program (NCHRP), "Signal Timing Manual," NCHRP Report 812, Transportation Research Washington, D.C., 2015.

[2] Y. Kavitha, P. Satyanarayana, and S.S. Mirza, "Sensor based traffic signal pre-emption for emergency vehicles using efficient short-range communication network," *Measur. Sens.*, vol. 28, Article 100830, 2023. DOI: 10.1016/j.measen.2023.100830

[3]Sparx Systems, “SysML-Systems Modeling Language”, Enterprise Architect User Guide, Version 16.0, URL: [https://sparxsystems.com/enterprise_architect_user_guide/16.0/modeling_languages/sysml.html]

IX. AFFIDAVIT

We, Akash Cuntur Shrinivasmurthy, Akhil Narayanaswamy, Krithika Premkumar and Madhukar Devendrappa herewith declare that we have composed the present paper and work ourself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance.