Smart Traffic Control Systems

1st Igor Risteski
Group 5
Fachhochschule Dortmund
Dortmund, Germany
igor.risteski001@stud.fh-dortmund.de

2nd Gökcer Sönmezocak
Group 5
Fachhochschuke Dortmund
Dortmund, Germany
goekcer.soenmezocak001@stud.fh-dortmund.de

3rd Ilker Kurtulan
Group 5
Fachhochschule Dortmund
Dortmund, Germany
ilker.kurtulan002@stud.fh-dortmund.de

4th Furkan Iskender
Group 5
Fachhochschule Dortmund
Dortmund, Germany
furkan.iskender001@stud.fh-dortmund.de

***Abstract*— Envisioning the future of urban transportation, this abstract delves into an advanced traffic control system designed for autonomous vehicles, with features such as optimizing traffic flow with intelligent vehicle routing and dynamic traffic lights. It prioritizes safety by granting emergency vehicles swift passage and ensures secure pedestrian crossings. This system represents a transformative leap, blending technology and thoughtful design for an efficient and harmonious future in smart city transportation.**

Keywords—traffic control system, monitoring, autonomous vehicle, pedestrian, priority, safety.

I. MOTIVATION

In the realm of autonomous vehicles, an advanced traffic control system would represent a breakthrough, and reshape urban transportation. Such a system should be designed for efficiency, and employ intelligent vehicle routing system algorithms to optimize traffic flow. It should also include an improved traffic light system that dynamically adjusts its signals based on real-time data, fostering well-coordinated vehicle movement through intersections. This should also include a vehicle priority system, which should prioritize emergency vehicles for swift passage during critical situations. Additionally, it should contain a pedestrian crossing system that ensures safety for the pedestrians about to cross the road in an environment that includes autonomous vehicles as entities. Our interpretation of such a system and its design is further discussed in this paper.

II. MODEL

A. Requirements

As per every new project, one of the very first steps consists of determining what the system as a whole is supposed to do, what should the system be able to achieve. In this particular paper, we will be looking at the requirements for a Traffic Control System for Autonomous vehicles. The best way to show this, is by using a Requirements Diagram, which has the purpose of giving a graphical representation of the entire system. Such a diagram is shown on Figure 1.1 below:

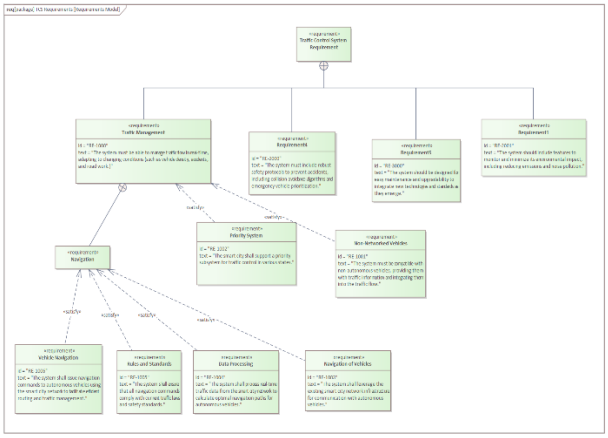

Figure 1.1 – Requirements Diagram

Each requirement is written in a box, and consists of a name, a unique identifier (Unique ID) and a textual description. How these requirements are connected to one another, is shown by the type of relations that are connecting them. These can be of three types: Containment (⊗), Refinement (keyword “refine”), and Derivation (keyword “deriveReqt”). Containment makes it possible to deconstruct a composite requirement into several single requirements, which are easier to trace. Refinement, consists of adding additional precision (ex. quantitative data), and lastly, Derivation, which consists of connecting requirements from different levels of the system (ex. connecting system requirements to requirements at the subsystem level).

The requirements for our particular system can be divided into General, Communication and Behavioral Requirements. As part of our general requirements, we have the “Traffic Management”, with a unique id “RE-1000” and a textual description “The system must be able to manage traffic flow in real-time, adapting to changing conditions (such as vehicle density, accidents and road work.”. This requirement is further divided into the following requirements: “Navigation”, “Priority System” and “Non-networked Vehicles”. The requirement “Navigation” can be further decomposed into other, more specific requirements, such as: “Vehicle Navigation Commands”, “Rules and Standards”, “Data Processing” and “Continuous Navigation of Vehicles”. The other branches of requirements consist of the following: a “Safety system”, which must ensure safety protocols to prevent accidents, “Upgrade & Maintenance” which should ensure that the system’s design is further upgradable and can

adapt to changes, as well as, ensure easy maintenance when needed, and lastly, a "Monitoring" requirement, which should include monitor, and therefore, minimize the environmental impact on the system.

Subsequently, it is possible to add further properties on top of these requirements. These properties can be of the following type: priority (high, medium, low), source (ex. customer, marketing, legislation etc.), risk (high, medium, low), current status (ex. suggested, validated, implemented, tested, delivered etc.), and verification method (analysis, demonstration, test).

B. Use-cases and Use-Case Diagrams

Use cases are a technique for capturing, modeling and specifying the requirements of a system. The best way to graphically show this, is by the use of Use-Case Diagrams.

These diagrams have four main purposes:

1) To specify the context of the system
2) To capture the requirements of the system
3) To validate the system architecture
4) To drive implementation and generate test cases

A Use-Case Diagram summarizes some of the relationships between the use cases, actors and systems. However, these diagrams do not show the order in which steps to achieve the goals of each use case are taken. The use cases represent only the functional requirements of the system. To clarify, a use case corresponds to a set of behaviors that the system may perform in interaction with its actors, which would lead to an observable result.

Each Use-Case Diagram may include the following:

Actors – entities who interact with the system, and in most cases, the ones that create a use case.

Use-Case – a system function or a process. The name is constructed of a combination of a verb and a noun, or a phrase. Each actor must be associated with a use case, while some use cases might not be associated with actors. Use cases determine the expected behavior, or in simpler words, "what" should happen, but they do not describe "how" to make this happen.

System Boundaries – for large and complex systems, each module may be the boundary of the system, but potentially it might be the whole system.

Types of relationships:

1) Association – indicate that the actor and the use case communicate with each other through messages.
2) Extension – indicate that the latter use case will happen in some, but not in all cases where the former is called upon.
3) Inclusion – indicate a dependency between the use cases it connects. Every time the first use case is executed, the second use case will be executed as well.
4) Generalization/Inheritance – indicate a parent-child relationship, where the child use case is an extension of its parent use case. A child use case inherits the behavior and the semantic component from the parent use case. The child use case, may add or override the behavior of its parent.

For our scenario of a Traffic Control System, we have defined four different Use-Case Diagrams, one for each of the subsystems it contains. Each of them will be further explained in detail.

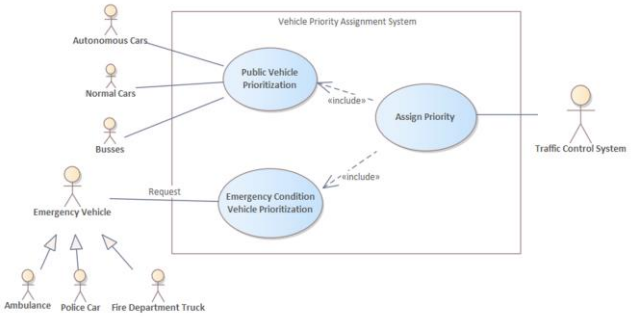

Figure 2.1 – Use case Diagram for Priority Assignment

In Figure 2.1 above, the Use-Case Diagram demonstrates the "Vehicle Priority Assignment System", which consists of the following Actors: the Public Vehicles (Autonomous Cars, Non-autonomous Cars, Busses), which are associated with the "Public Vehicle Prioritization" use case, and the Emergency Vehicles, which inherits the following actors: Ambulance, Police Car, and Fire Department Truck, which are associated with the "Emergency Vehicle Prioritization" use case, though the message "Request". Both of these use cases are associated with the use case "Assign Priority" with a include relationship. The "Assign Priority" use case is associated with the Traffic Control System, which acts as an actor in the system.

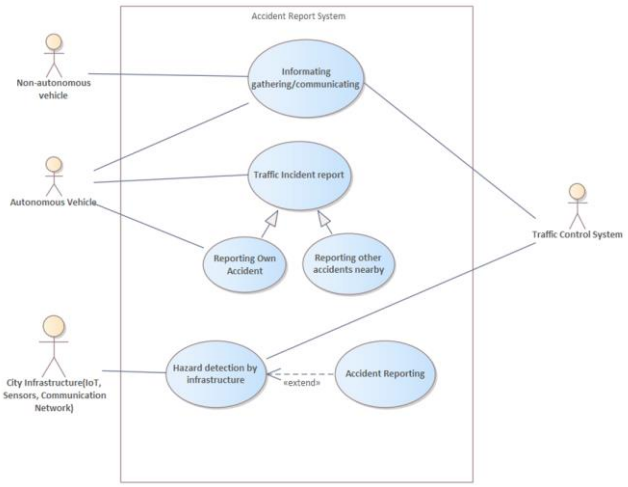

Figure 2.2 – Use case Diagram for Accident Reporting

The second use case diagram is shown in Figure 2.2. This diagram consists of the actors "Non-autonomous Cars" and "Autonomous vehicles", both of which are associated with the use case "Information gathering/communicating", which is also associated with the actor "Traffic Control System". The entity "Autonomous vehicles" however, is additionally associated with the "Traffic Incident Report" use case, which inherits the use cases "Reporting Own Accident" and

"Reporting other accidents nearby". The last actor included in this diagram, is the "City infrastructure (IoT, Sensors, Communication Network)", which is associated with the "Hazard detection by infrastructure" use case. This use case extends an additional use case called "Accident Reporting", and is additionally associated with the "Traffic Control System".

The next Use-Case Diagram is about the "Vehicle Rerouting System", which can be seen on Figure 2.3.

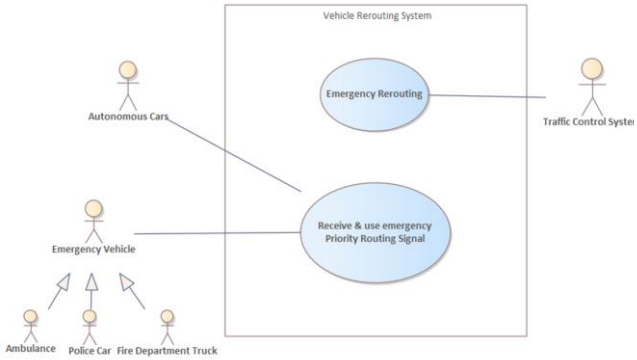

Figure 2.3 – Use case Diagram for the Rerouting System

In this diagram, as actors, we have the normal "Autonomous vehicles", and the "Emergency vehicles", which inherits the "Ambulance", "Police Car", and "Fire Department Truck". Both actors are associated with the "Receive and use emergency Priority Routing System", as both actors interact with such a system and receive instructions. Furthermore, we have the "Emergency Rerouting" use case, which interacts with the "Traffic Control System" and realizes the required rerouting of the vehicles.

Other type of actors that are a part of such a system and can interact with it, are the pedestrians. In almost every intersection, at a certain point in time, a human is bound to be present and interact in one way or another with the system. For such a case, yet another diagram is necessary. The diagram that includes people as actors of such a system, is shown in Figure 2.4 below.

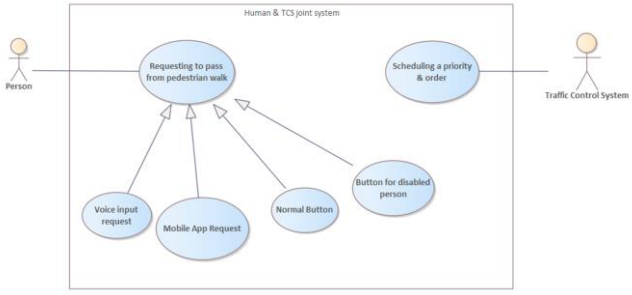

Figure 2.3 – Use case Diagram for Pedestrians requesting to pass an intersection

The entity that interacts with the system in this case, is the pedestrian. It is shown here as the actor "Person", who is associated with the use case "Requesting to pass from pedestrian walk". The pedestrian can interact with the system in multiple ways, shown as the following use cases: "Voice input request", "Mobile App Request", "Normal button" and "Button for disabled person". All of these use cases are inherited inside the request use case, and describe different ways a pedestrian might interact with the system, either by a voice request, through a mobile application, by pressing a button on the traffic light, or pressing a special kind of button for disabled people, which would ask for a prolonged action. The next step is demonstrated by the use case "Scheduling a priority and order", which is associated with the actor "Traffic Control System", in a way that it updates the schedule and executes a priority order in the system.

C. Sequence Diagrams

Sequence Diagrams are another type of interaction diagrams that detail how operations are performed. They capture the interaction between objects in the context of collaboration, and show how elements interact over time, and how they are organized with respect to objects and time.

The horizontal axis shows elements that participate in the interaction and can appear in any order. Objects participating in an operation are typically listed from left to right, according to when they participate in the message sequence.

The vertical axis shows the progressing time. The time in the sequence diagram is denoted in the context of the execution order, that is the sequence itself, and not the actual duration. For that purpose, a different type of diagrams are used, which are known as "timing diagrams".

The notation in the sequence diagrams consists of:

1) Actors – corresponds to entities that interact with the system
2) Lifeline – represents an individual participant in the interaction, and displays the passage of time
3) Activation & Deactivation line – represented by the start and the end of the thin rectangle on the lifeline, which indicates the period during which the element performs an operation
4) Message – represents multiple types of messages (call message, return message, self message) that represent a certain activity, depending on the type of message
5) Notes – gives the ability to attach various notes to elements
6) Sequence Fragments – represented as a box called a combined fragment, which contains a fragment operator (opt, alt, par, loop, break, critical, strict, ignore, ref, etc.) that indicates the type of fragment.

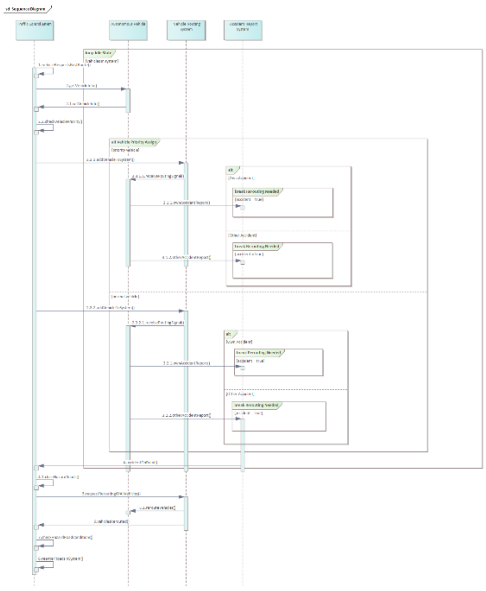

Figure 3.1 – Sequence Diagram for the Traffic Control System

For our particular system, a main sequence diagram and some subsystem sequence diagrams can be shown. From the diagram in Figure 3.1, we can demonstrate the execution order of the system. The Lifeline "Traffic Control System", continuously sends a Self-message containing a request for the best route, in order to constantly keep the vehicles that are a part of the system on the best possible route. Then, when it receives a request for a new car to join the system, it sends a call message to the Lifeline "Autonomous Car" calling a "getVehicleInfo()" method, to which, it receives a message containing the relevant information. Then the system sends another self-message, executing the "checkVehiclePriority()" method and adds the vehicle in the system, by sending a call message to the "Vehicle Routing System" lifeline. Now, the new vehicle is added to the system as either a normal or a priority vehicle, as the "addVehicleToSystem()" function is inside a alt sequence fragment, which means that it executes only one of the two call messages it sends. These cars are now in the system and are part of a loop sequence, which constantly updates the routing signals transmitted to them, depending on the current optimal route. When any of these cars that are inside this loop sequence are a part, or see an accident, they send a call message to the "Accident Report System" lifeline, in a alt sequence, depending whether they are a part of the accident themselves, or report an accident they encounter. In such a scenario, they enter a break loop which takes the system out of the idle state, sends a self-message to close the roads included in the hazard and start a rerouting process. Then this new route is transmitted to all of the vehicles, and they enter the idle loop again. When the hazardous roads are cleared up, the system reenters the roads in the routing. Now, inside the idle loop, the optimization of the routing will be done.

The following diagram, shown in Figure 3.2, demonstrates the Traffic Lights Sequence.

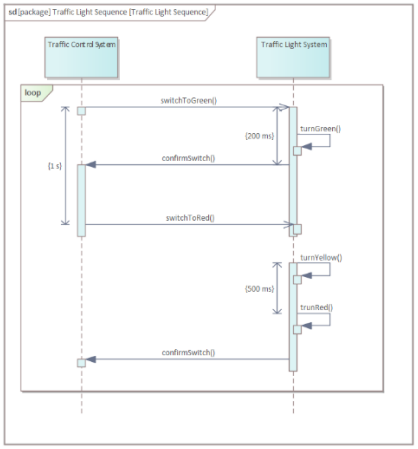

Figure 3.2 – Sequence Diagram for the Traffic Lights

This diagram is simple, and contains only two lifelines, the "Traffic Control System" and the "Traffic Light System". The way that they interact with each other, is that they are inside a loop fragment, in which the "Traffic Control System" sends a message to switch the lights to green, the "Traffic Light System" sends a self-message with the command to turn the lights to green, and sends a return message containing the switching of the light. After some time, the "Traffic Control System" sends a message to turn the lights Red, to which the "Traffic Lights System" sends itself a self-message turning them yellow, and after a set amount of time, sends itself a second self-message turning the light to red. After the actions are executed, a return message is sent back to the "Traffic Control System" confirming the action. The system keeps doing this in a loop, as stated by the loop fragment.

Another Sequence Diagram is shown in Figure 3.3.

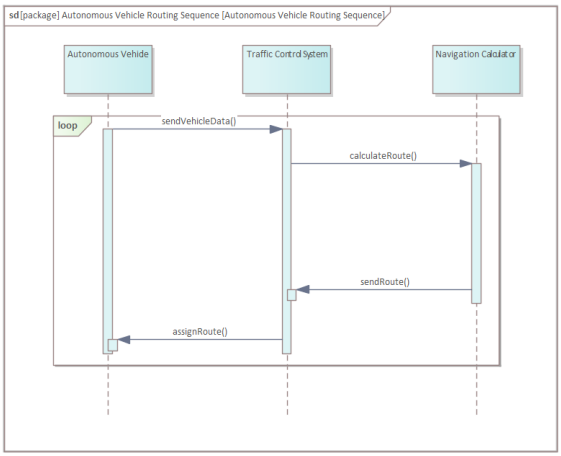

Figure 3.3 – Sequence Diagram for Routing Sequence

This diagram shows the sequence for the Routing system. The lifelines that are a part of this sequence are the "Autonomous Vehicles", the "Traffic Control System" and the "Navigation Calculator". The actions are executed inside a loop fragment. The "Autonomous Vehicles" sends a call message containing the "sendVehicleData()" function to the "Traffic Control System", which then sends a message to the "Navigation Calculator" executing the "calculateRoute()" function. Then, after the operation is executed, the "Navigation Calculator" sends back a call message "sendRoute()" to the Traffic Control System. After this is executed, the system sends a call

message with the "assignRoute()" method inside to the "Autonomous Vehicle". By then, all of the operations are executed and the loop is restarted.

Activity diagrams play an important role in a project design phase. It helps to visualize process and ongoing activities. During the design phase, four activity diagram are constructed for a *Smart City System*. Those include "Smart Traffic Lighting for Pedestrians", "Accident Reporting System", "Vehicle Routing System" and "Traffic Lighting Priority System".

- Traffic Lighting Priority System: A subsystem that assigns priority to emergency vehicles (ambulance, police car, fire truck) and autonomous vehicles. Assigned priorities will change the traffic lights. This will help vehicles by going their own way quicker.
- Accident Report System: A subsystem that reports nearby accidents via sensors network, camera systems and autonomous sensor reports.
- Vehicle Routing System: A subsystem that finds the shortest, best route for vehicles.
- Smart Traffic Lighting System for Pedestrian: A subsystem that changes pedestrian lighting for disabled and non-disabled people.

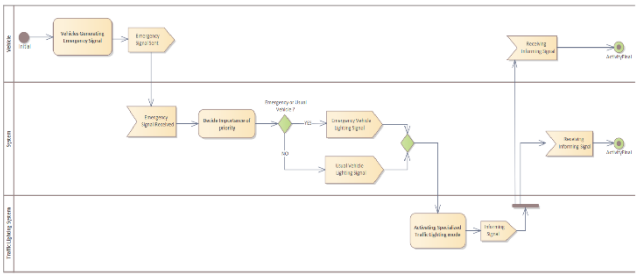

Fig.2.7 Traffic Lighting Priority System

As shown in Fig.2.7, the activity diagram "Traffic Lighting Priority System" has 3 partitions. Vehicles start off by generating emergency signal. Emergency signal could come from many vehicles such as ambulance, police car or could be from usual autonomous cars. This signal is sent to the system in order to rank it's priority. After this step, vehicle classification is made on the system. Then, system decides if the sent signal is coming from the emergency vehicle such as ambulance, police car or fire department truck. Depending on this classification, seperate traffic lighting modes created in order to be sent to the traffic lights. Received traffic mode, will be used by traffic lights for different scenarios. As a last step system and prioritized vehicle will be informed about the current lighting mode.

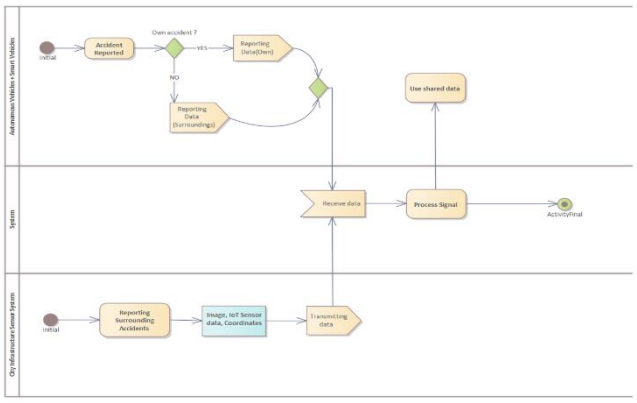

Fig. 2.8. Accident Report System

Above diagram Fig. 2.8 "Accident Report System" initially can start from either the city infrastructure sensor system or autonomous vehicles. City infrastructure will be reporting accidents with nearby sensors, camera systems and IoT systems. These reports from sensor, cameras and IoT devices will include smoke sensors data for proximate incidents, multimedia records of the accidents with video/image data and various collision detection sensor technologies that will be used on the roads. Autonomous cars will be reporting nearby accidents also with computer vision systems along with the estimated location of the incident. Accidents will be sent to the system as a combined report with location, multimedia datas and combined sensor datas. All these datas will be processed and autonomous car drivers will be informed about location of accidents.

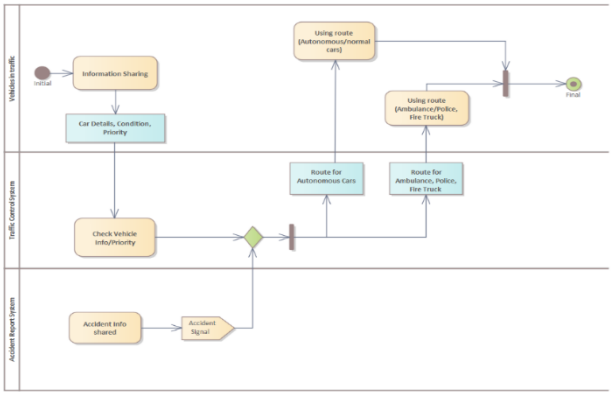

Fig. 2.9. Vehicle Routing System

"Vehicle Routing System" and "Traffic Lighting Priority System" work mutually to form a good coordinated traffic management system for both Emergency vehicles and usual autonomous vehicles. As shown in the Fig. 2.9. "Vehicle Routing System" starts off by vehicle, it shares it's own information such as vehicle details and current condition in the traffic. Condition here refers to the importance of vehicle's purpose. If a driver had an accident or other kind of accident happened nearby. Besides autonomous cars, smart emergency vehicles can also send their information and priority condition. For example ambulance drives can share how critical the situatiaon of a patient is or police car share a report based on scenario to the system. Sent information and condition report are processed and different routes will be created for both autonomous cars and emergency vehicles. Created routes sent to the vehicles in order to be used.

.

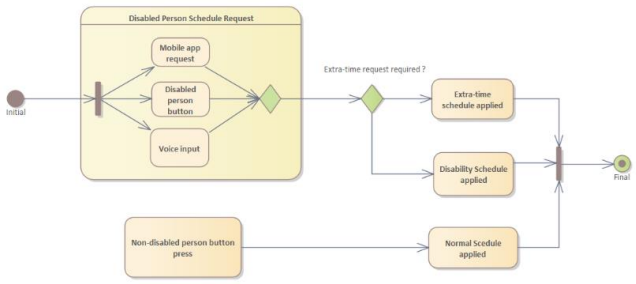

Fig. 2.10. Smart Traffic Lighting for Pedestrians

Smart traffic lighting sub system's main purpose is to focus on people passing pedestrian cross with different schedules based on disability of a person. Main focus here is for disabled people. As shown on Fig. 2.10 in order to start a request, disabled person can use mobil app, manual button or voice input that will be located on sidewalk. Requests from the mobile app will be checked by the system and based on person's disability, extra time for passing or assist will be provided to the person. Beside mobile app request, voice input device on the sidewalk can be used by people with blindness. Lastly, manual buttons on the sidewalk can be used by both disabled and non-disabled people.

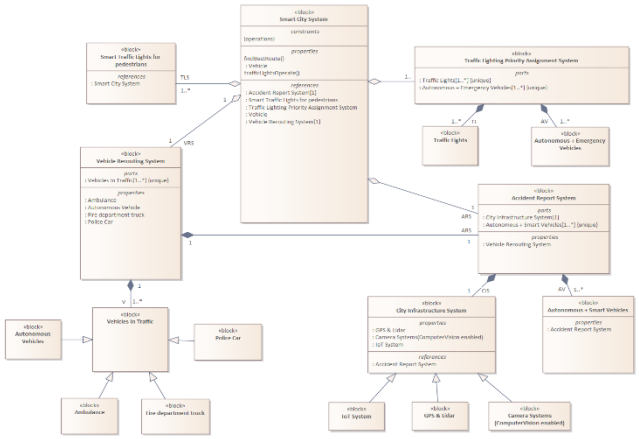

Fig. 2.11. Block (Context) Diagram

As demonstrated in Fig. 2.11, diagram consist of main smart city system block having aggregation and composition behavior relationships with other sub-systes. Sub-systems have their sub-blocks. These sub blocks have generalization relationship with the sub-blocks. This block diagram showing us general structure of smart city system.

Internal block diagrams has divided into four sections. Main reason for dividing these internal diagrams into four section is to show them seperately with better internal parts compare to a single diagram. Purpose of this Internal Block Diagram is to show internal structure of a subsystem and it's relation with it's parts. These parts have specific data flows and information sharing. Each section representing our subsystems.

- Internal block diagram of Accident Report System
- Internal block diagram of Traffic Lighting Priority System
- Internal block diagram of Vehicle Routing System

Internal block diagram of Smart Traffic Lighting for Pedestrians

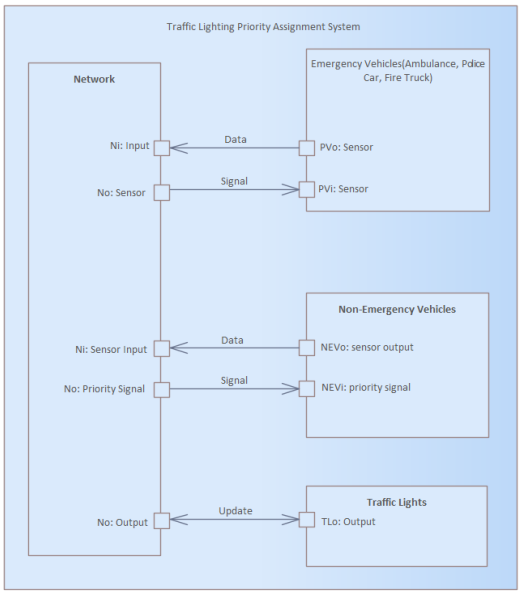

Fig.2.12. Internal Block Diagram - Traffic Lighting Priority System

First sub-system is demonstrates internal structure of "Traffic Lighting Priority System". As it can be seen from the above diagram Fig. 2.12, "Emergency Vehicles" such as ambulance, police car, fire truck also "Non-Emergency Vehicles" referring to the autonomous cars will be sending their emergency signals. When signals received by the system, prioritization has to be made. During extreme case scenarios, certain emergency vehicles can get higher priority based on how urgent the situation is. For example, police car could be chasing a criminal, fire trucks rushing to extinguish a forest fire or ambulance could be trying to reach a scene where a patient needs immediate treatment. On the other side, "Non-Emergency" meaning autonomous cars could also encounter urgent situations, such as a person need to be rushed to the hospital. All the mentioned scenarios and their priority levels will be handled by system and a traffic lighting mode will be activated for specific vehicle. System in the diagram refers to the overall system. This includes "Vehicle Routing System". When a "Vehicle Routing System" provides a vehicle the shortest route, the "Traffic Lighting Priority Assignment System" will activate the special lighting mode. This means some side roads on the shortest path will be blocked by red light until vehicle passes through.

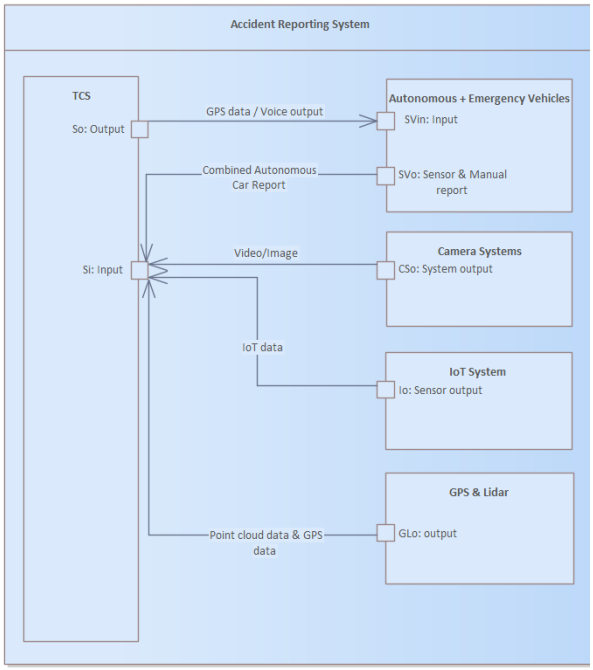

Fig.2.13. Internal Block Diagram – Accident Reporting System

Above figure Fig. 2.13 is explaning the internal structure of "Accident Report System". Combined autonomous car report, emergency vehicle report, video stream, images, IoT data, gps and lidar datas are sent to the traffic control system. Spotted incident's exact location will be provided to the emergency vehicles and autonomous vehicles.

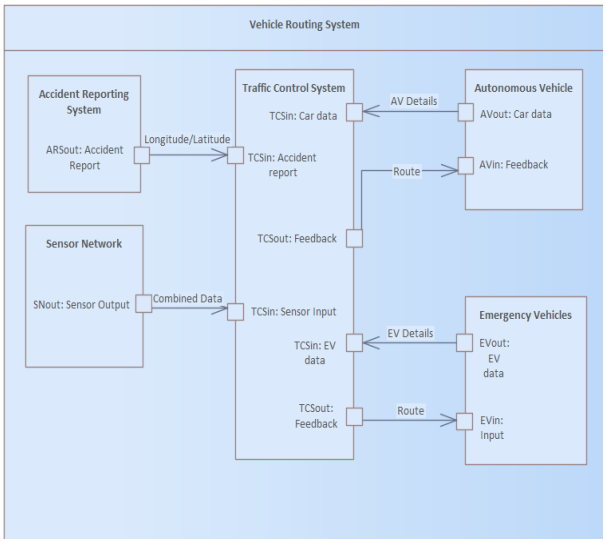

Fig.2.14. Internal Block Diagram – Vehicle Routing System

Here on Fig 2.14 showing internal structure of "Vehicle Routing System". On the one side, sensor network, "Accident Reporting System" sharing longtitude/lattitude and sensor data, on the other side autonomous vehicles and emergency vehicles sharing their internal reports with the traffic control system. As a result, traffic control system gives them best route possible.

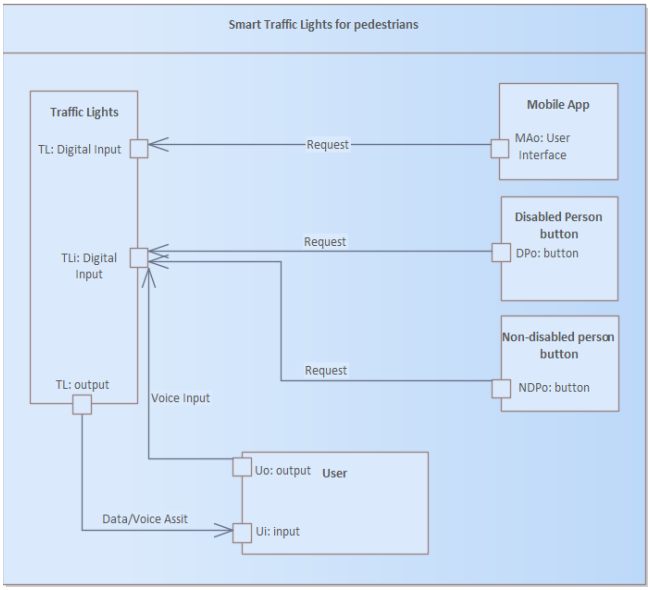

Fig. 2.15. Internal Block Diagram – Smart Traffic Lights for Pedestrians

Above diagram Fig.2.15 shows structure of "Smart Traffic Lights for Pedestrians". Both disabled and non-disabled users can request to pass pedestrian walk using button, mobile app or voice input. Depending on user's disability, extra time will be added to the schedule.

D. Constraint Block Parametrics Diagrams

The system that has been designed was investigated by creating constraint block parametric diagrams as they offer a robust limitation in the context of a candidate implementation phase. The main purpose was considered as the interrelation of the use cases under the smart networked traffic control system to detect fragile parts while improving the overall system quality in terms of the level of satisfaction of the pre-determined requirements. This step of the design enabled us to realize overall structure that is composed of the each independent modules. The lack of constraint block parametrics diagram analysis would cause loose interrelation of the modules as well as unrealizable use cases for the candidate implementation, which may result in increased time of re-work to adjust the design.

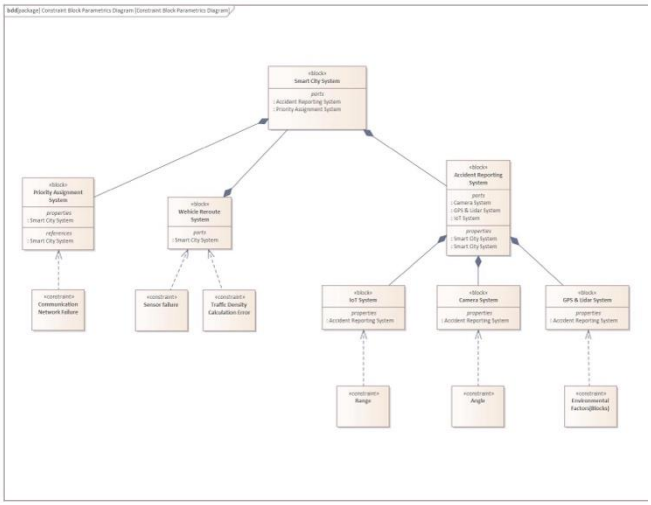

Figure.2.16. Smart City System Constraint Block Diagram

There are three main block relationships which are namely as priority assignment along with vehicle rerouting and accident reporting modules that constructs the overall dependencies. Priority assignment system depends on communication network failure as constraint. Moreover, sensor failure and traffic density calculation error constitute the prerequisite constraint for the vehicle rerouting system. On the other hand, accident reporting system is aggregated from several systems namely as IoT system, camera system and GPS & Lidar System. Each system depends on their corresponding specific data constraint. IoT system depends on the range while camera system depends on view angle. GPS & Lidar system depends on environmental factors.

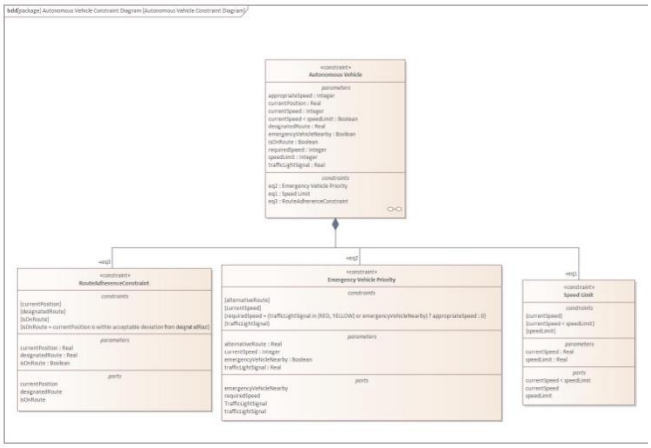

Figure.2.17. Autonomous Emergency Vehicle Priority Constraint Block Diagram

Autonomous emergency vehicle priority system constraints are based on the autonomous vehicle itself. The parameters for the autonomous vehicle are its data for information related with its journey state and parameters such as its position and speed. The constraints for the vehicle are namely as emergency vehicle priority, route adherence constraint and the speed limit. The route adherence constraint checks for the availability of the designated route compared with the current route of the vehicle. The emergency vehicle priority constraints are the alternative route along with the current speed of the vehicle compared with the traffic light state to decide on the required speed for the vehicle. The speed limit constraint are the current speed of the vehicle compared with the speed limit to check the obedience of the traffic rules. The priority assignment module is strained with the transfer time of the signal to response in an optimized duration to ensure stability.

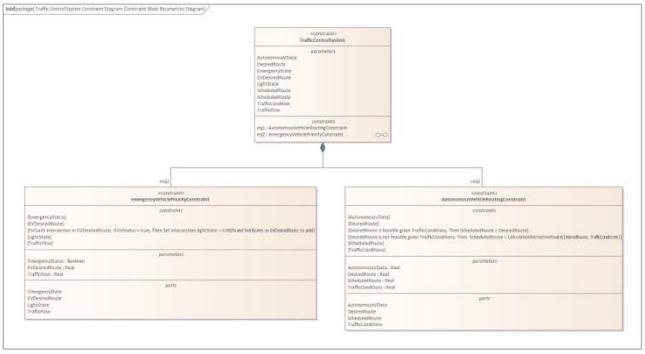

Figure.2.18. Traffic Control System Emergency Rerouting Constraint Block Diagram

The traffic control system emergency rerouting depends on both emergency vehicle priority and autonomous vehicle rerouting constraints. Autonomous vehicle routing constraint includes real time conditions such as current crowdedness of the traffic which is not under control of the system itself. In such a case the priority assignment range is considered as 1km in the context of in city traffic. Beyond that distance the system is not responsible for the priority assignment of any emergency vehicles as a constraint to narrow down the complexity of the problem.

E. State Machine Diagrams

After having the understanding of the module constraint and their interrelation in between each other, the need for a visualized system behavior comes up to the stage. Therefore, state machine diagrams corresponding to the use cases that we consider to implement are created.

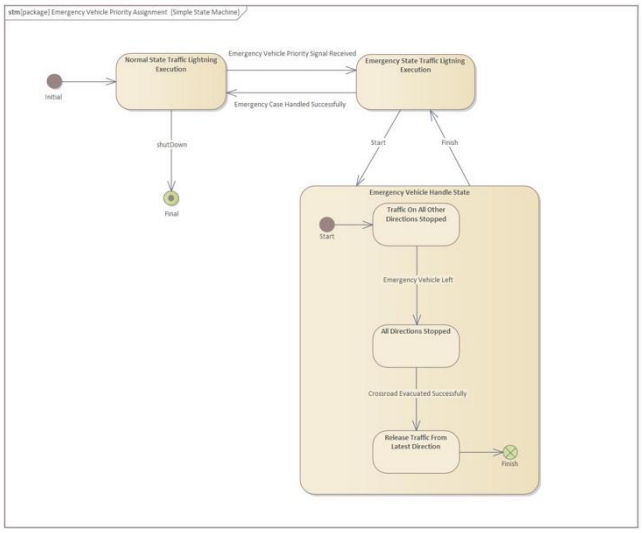

Figure.2.19. Emergency Vehicle Priority Assignment SMD

Emergency vehicle priority assignment state machine diagram is composed of two states along with one sub-state

to detail the behavior of emergency handling. The system is considered as in the normal state execution of traffic lights as initially. Upon the reception of the emergency vehicle priority assignment signal to the system, the state transits into the emergency state traffic lighting execution. To give insight about the handling procedure a sub-state under the emergency state has been created as emergency vehicle handle state. It firsts enter the state of traffic halt from all other directions based the received emergency vehicle priority signal. After the emergency vehicle left the junction, the system enters the state of junction safe evacuation by halting all direction in the junction to enable the ongoing emergency vehicle an empty bubble in the road. This state also makes sure that the junction is safely evacuated by the other cars that may be went after the emergency vehicle. After the junction is successfully evacuated, the system is ready to turn back to its initial state by releasing the traffic from the latest direction that it halted. Finally, the lighting timing and execution order turns back to its default state as normal. Unless the system is forced down to the shutdown, it continuously executes itself.

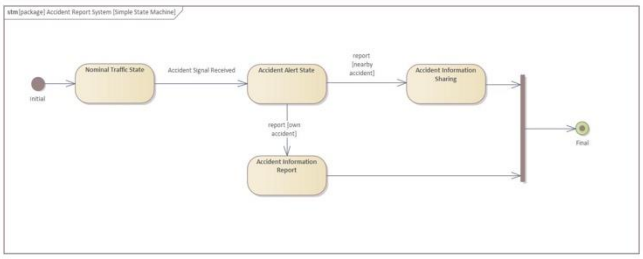

Figure.2.20. Automated Accident Report SMD

Accident reporting system initially perceive the traffic as nominal without any disorder in the traffic flow. Upon the reception of the accident alert signal the system changed its state into accident alert state and investigate more about the source of the signal to decide whether or not the signal is coming from the car that is involved into the traffic accident or it is just an information sharing from a vehicle that witnessed another traffic accident. Therefore, if the signal comes from the accident involved car, the reporting system perceives the case as information report. On the other hand, if the report signal is coming from another accident nearby the system goes into accident information state. Since the information sharing has temporary importance during a traffic flow, it handles and goes into final as system. If the signal continuous to keep coming from the flow, information report or sharing is kept as in a time instance.

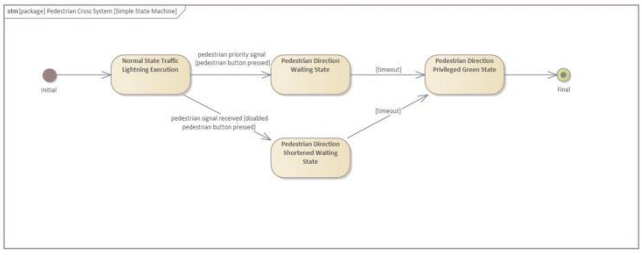

Figure.2.21. Pedestrian Cross SMD

The pedestrian crosswalk system has been designed to have two different waiting states for the two types of pedestrians namely as pedestrians and disabled pedestrians. Upon the corresponding button is pressed in the system, the state goes into the corresponding waiting state. For the disabled pedestrians the waiting time adjusted to be shortened so that the upon the timeout the state of privileged green light state is entered faster. After the green light execution state the system goes into its final as it is the end of the use case.

F. Fault Tree Diagrams and Analysis

The fault tree diagrams are beneficial tools to visualize and foresee possible risks in the system which may result in errors during the execution. The fault trees enabled us to investigate fragile parts of the design of our pre-determined use cases. In each fault tree diagram the combination of hazards in the system combined with the hazards in an autonomous vehicle result in fatal accident scenario.

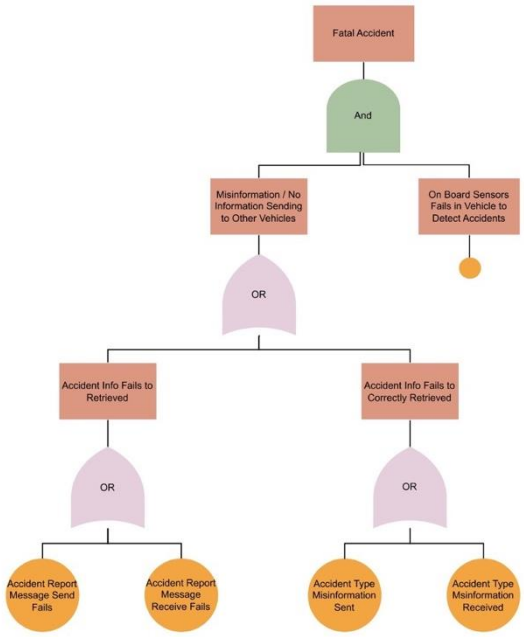

Figure.2.22. Automated Accident Report Fault Tree Diagram

The fault tree analysis of the automated accident reporting is based on the two actors namely as the system itself and the autonomous vehicle. There are two fundamental branches that may cause erroneous state of the subsystem. Each branch is rooted from basic failure event. For the case of accident information retrieval failure, either a system may not receive, or the autonomous vehicle cannot send the message. Moreover, the correctness of the message is also important as it directly affects the behavioral reaction of the system as it has been stated in the state machine diagram. Therefore, misinformation of the message content either received by the system or sent by the autonomous vehicle cause to that failure. These two failure branches are combined as one of them is enough to propagate to the misinformation or no-information sending the other vehicles in the traffic flow. In the case of a failure in an autonomous vehicle in terms of sensors and their capability to detect

accidents or obstacles on the road combined with the system failure result in fatal accident as collusion is inevitable.

The analysis of this fault tree diagram shows that the conditional failure can emerged both from hardware and software related basic failures. For the hardware caused fragility, sensor redundancy in our system can be adopted to have a backup hardware all the time to limit the probability of failure as it has been stated in the article of “Failure Prediction for Autonomous
Systems” by Christopher B. Kuhn [1]. Moreover, external computer vision implementation as a back system for the messaging can be implemented to double check the retrieved information validity.

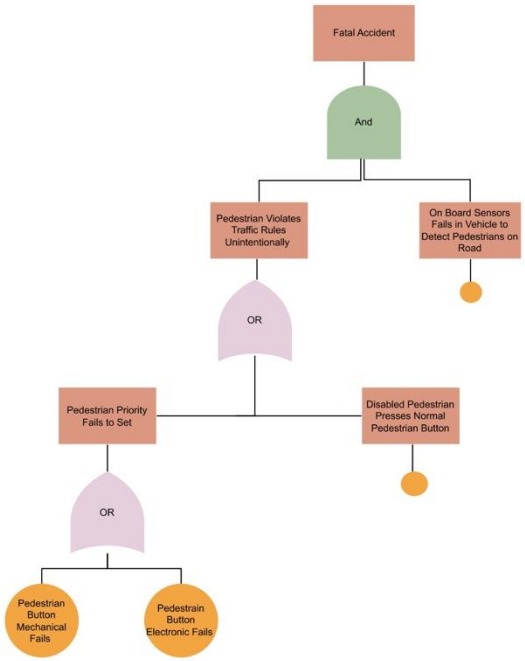

Figure.2.23. Pedestrian Cross Fault Tree Diagram

Pedestrian cross fault tree diagram rooted from the human input related hazard along with hardware failures. The mechanical or electronic system of the button may fail to set the priority signal even if the pedestrian takes action to set it. The human related hazard of the system emerges from the case when the disabled person presses a normal pedestrian button where the allocated the cross time may not be enough for the person to cross the street. The only one of those two stages are enough to put the pedestrian in a case where it violates the traffic rules unintentionally. While the pedestrian are still on the road to cross when the traffic lighting system sets the right to pass for the drivers, an autonomous driving vehicle may hit the pedestrian if an on-board sensor failure may happen at the same time.

The analysis of the diagram mainly focused on the user input failure case as the misinformation input can cause fatal accident. Therefore, user input should be eliminated at first by proposing computer vision implementation to detect a pedestrian that is intended to cross the street. Moreover, the system should be designed to take into consideration of the disabled pedestrians to let them cross the street in an appropriate time. An advanced computer vision algorithm may even assess the time of cross for that specific pedestrian to optimize the traffic flow while prioritizing the safety of the pedestrian.

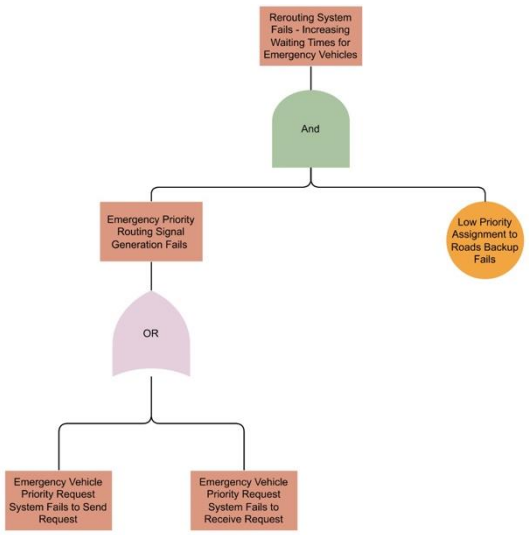

Figure2.24. Emergency Vehicle Priority Fault Tree Diagram

The fault tree diagram of the emergency vehicle priority assignment rooted from the component failure event where the message is either cannot be received by the system or sent by the vehicle. Any of these failures can be caused from both hardware or software related issues. Moreover, the environmental factors such as weather conditions and electromagnetic noise around are also important where the medium for the message transmission is considered as wireless yet not with a specific protocol. Therefore possibility to predict for the case of such a failure is seemed to be unmeasurable in terms of the environmental factors but rather they may be regarded as exceptional scenarios. Therefore, a backup system is thought to be introduced in the case the priority signal may not be received by the system correctly. A low priority assignment backup system works as a reciprocal system to validate the high priority assignment system. However, even this reciprocal system may fail, which causes an unoptimized behavior for the traffic routing system resulting in an increased waiting times for the emergency vehicles.

III. DESIGN

As Furkan a member of this group projects in Embedded Systems Engineering my main contributions to the project were the architecture of the implementation, the development of the simulation, scheduling concepts and also testing. The approach to the simulation and our expectations from it were that it was to simulate and display 3 main use cases we supported in the networked traffic in the smart city system.

These main use cases include the navigation of the autonomous vehicle by the central system, traffic light control directly from the main control system and lastly the emergency vehicle priority handling.

To explain them in a more nuanced way and why we have chosen to migrate the control over crucial things to the central unit which is our Traffic Control System (TCS), was that these use cases had intertwined scenarios such as emergency vehicle priorities changing the light signals and rerouting the vehicles out of the emergency road. With the central architecture the control system could react and handle these situations with an acceptable overhead while also maintaining a simple structure for the ease of implementation.

Also, we had future plans to include light synchronized together with autonomous vehicle routing to minimize the wait time on red lights to reduce emissions and provide a better travel for car passengers as the journey would be continuous but the complexity of the implementation of this concept had a negative effect on final agreement in the inclusion of it.

A. Concept

In this section of the design choices in the implementation side we are going to be discussing the main architecture elements we have decided on such as different components of the system and how they are connected together while also explaining the scheduler requirements that are needed from our central controller unit.

,

1) I2c Bus as Network

For communication purposes we have selected an I2c serial communication bus that will connect all of the components of the system together. This approach allowed every component in the bus to be able to communicate with each other, even though our simulation does not utilize this feature extensively and only communicates between the central component and a secondary component, this design choice was much superior on simplicity perspective compared to the UART serial communication alternative that we have identified from both a wiring and an implementation perspective.

2) The TCS and Scheduling

The heart of our system at the center, the Traffic Control System had several requirements and supported our main cases directly or indirectly. It needed to support message types that are specific to the component that it is in communication with, it had to be able to control the traffic lights directly, also to support a navigation calculation capabilities, a state machine design that will allow it to switch its state to change the behavior of the system and lastly it had to support an emergency priority features for emergency scenarios.

Scheduler of the tasks by the traffic control system is decided on to be the earliest deadline first since some of our tasks are sporadic in nature, namely the emergency task. By comparing EDF scheduling algorithm with rate monotonic scheduling, you can clearly see that with the rate monotonic scheduling the emergency task would fail the real time requirements since its period is the longest out of all.

Through constraining the arrival of emergency requests to 1 at most in every 3 seconds, we have managed to constrain the system to be able to test it on theory since non specified constraint on that part would mean the emergency request could be received 5 times a second, which would already push the utilization factor to 1 and make the system unschedulable

	T1 TrafficLight1Task	T2 TrafficLight2Task	T3 AutonomousVehicleRouting1Task	T4 AutonomousVehicleRouting1Task	T5 EmergencyRequestTask
Ci	100	100	100	100	200
Ti	1000	1000	1000	1000	3000
Di	1000	1000	1000	1000	500

Fight 3.1 Task Table

3) Traffic Lights

Traffic lights for our system had to be able to support features such a 3 color display for red yellow and green for demonstration purposes and also to be able to connect to the network also had to have I2c setup. For our current design we have just implemented these traffic lights units that take in requests for light switches from the traffic control system, and execute them, they are set to receive instructions that are enumerated and execute light switches accordingly.

4) Autonomous Vehicles

The design choices for autonomous vehicles was that they would have the capabilities of route navigation requests to Traffic Control System and receive back an optimal navigation from the central controller due to this design, compared to the other components having a more active communication between the TCS and itself. Also for demonstrative purposes we needed a display that will show the current navigation of the vehicle itself.

5) Emergency Vehicles

One of the use cases, which included the need for an emergency vehicle on our network that had the ability to request higher priority on a certain route from the traffic control system had to be implemented. The sporadic nature of this event and for the demonstration to be more interactive, we have decided to utilize an input such as a button to trigger this emergency scenario, with the press of this button the vehicle would request a priority from the TCS on the specified route.

6) Scenarios

First use case and the sequence of it in a written for emergency priority request handling can be seen in the sequence diagram, this approach basically is in three parts, emergency vehicle requests the priority, the traffic control systems calculates new navigational commands for the cars

on the route and reroutes them, then finally it turns the lights to green the let the emergency vehicles pass.

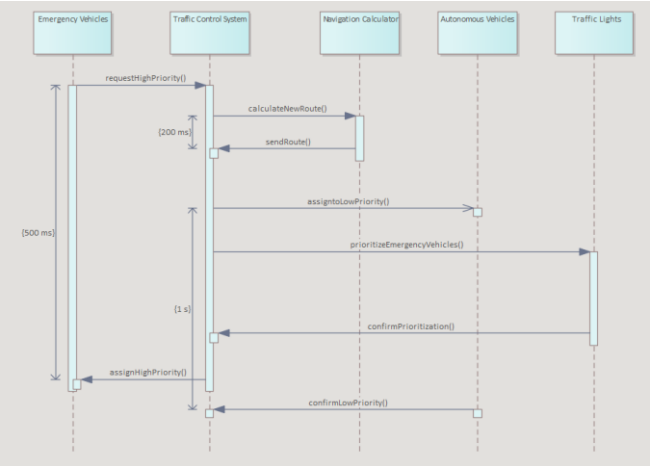

Fig 6.2 Sequence Diagram for Priority Request

Second use case includes the control of traffic lights over the network from the traffic controller system, cyclically this component is gonna schedule a task to switch the light of the traffic lights and ensure the simulation of an intersection.

Third use case for the simulation involves the autonomous vehicles cyclically sending a request for rerouting to a certain path and the traffic control system handling these requests with a scheduling approach.

IV. IMPLEMENTATION

In this section we are going to be taking a look into the realization of the proposed simulation designs, mainly with the messages going around with minor walkthroughs of the code's written.

1) Connectivity

As stated on the design, the key feature of this system and the actual part that makes the system truly smart is the I2c bus connecting each component together in a network. Also following the wiring from the schematics, the traffic lights and the 7 segment displays are visible for demonstration purposes, finally for the user interaction we have a button that will generate an emergency event and demonstrate the use cases working together as the simulation runs.

2) Implementation of Use Cases

The use cases for traffic light control are being handling cyclically from the traffic control system side, this component schedules a light switch for the both traffic lights on the intersection every second to control the traffic lights, the requests for autonomous vehicles use cases are scheduled are handled per request, the traffic control system schedules the rerouting as it is requested from the autonomous vehicles, and finally the emergency priority request is fully sporadic in nature since it doesn't happen cyclically and can be triggered in any given time, the arrival of the emergency priority request signal transitions the system to and emergency state to allow for an adaptive approach to handling said emergency scenario

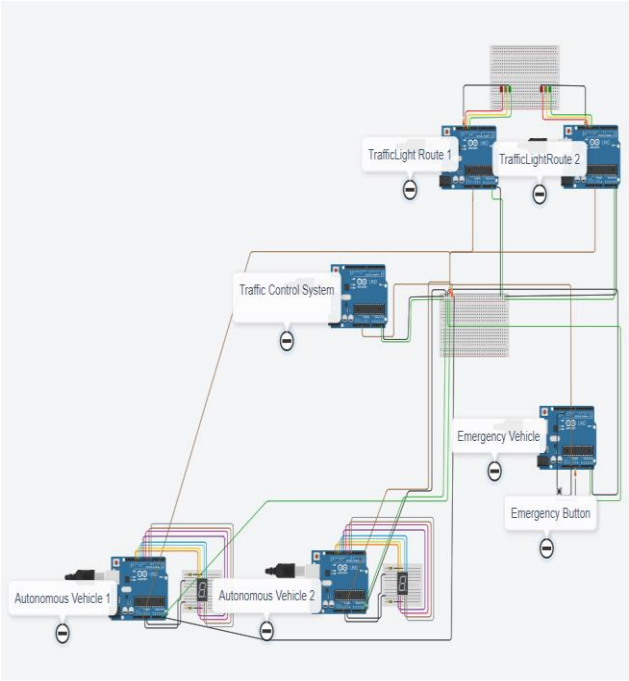

Fig 6.3 Connectivity Schema

3) Scheduling Design

To implement an EDF algorithm in our arduino's we have run into the problem of lacking materials and tools provided by TinkerCAD, this issue has been overcome by writing our own scheduler with a tasklist vector that supported ordering by the earliest deadline so they can be executed in correct order with correct priorities. However this approach limited us into a non preemptive solution since implementing a proper preemptive scheduler would call for an increased effort in writing a library for it, hence why we stuck with the non preemptive design, which would mean that our worst case response times being delayed by lower priority tasks that may have started executing earlier.

4) Scheduling Tests

To prove that our system was schedulable with our worst case execution times, we have come up with rudimentary values on a larger scale since the runtime of our tasks are quite small and are not of any challenge at the scale of our system in a scheduling context. These longer periods for worst case execution times are then used by us with and EDF scheduling algorithm and calculated to be schedulable with a utilization factor of 0.467 .

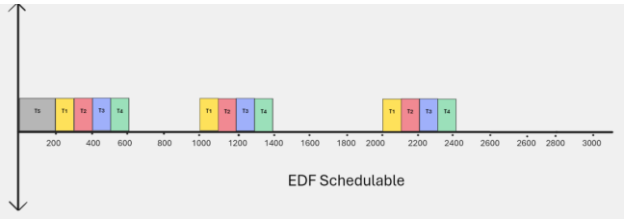

Fig 6.4 EDF Schedulability Test

5) Unit tests on ESP32

Through the aunit library in arduino, we have done unit testing for several functions inside the traffic control system which includes the tasks scheduling, state transitions and data parsing. The functions seemed stable for intended usages and passed basic tests on expected behavior.

6) Last release

The final version of the software is tagged as v1.0 in our github collaboration, the codes for all of the components and also the uni testing suite could be found within the files. Lastly for the actual simulation, a link to the TinkerCAD simulation has been added the the readme.md on the implementation branch and also the v1.0 tagged release.

V. REFERENCES

[1] Kuhn, B. (2022, November 18). Failure Prediction for Autonomous Systems. München; Technische Universität München.

VI. SUMMARY AND OUTLOOK

This paper consists of an interpretation of a Smart City System that contains an advanced Traffic Control System with the role of assigning driving routes and controlling live traffic-flow for autonomous vehicles, assigning priorities of such vehicles, adding a traffic light system and additionally includes possible human interaction in the form of pedestrians as part of the system. The paper covers the design process from motivation, to modelling, designing and implementation of such a system. A simulation of the traffic light system was done using TinkerCAD and can be seen there. The complexity of this system is considerable and can be further improved.

VII. APPENDIX

Contribution Percentage by members:
Furkan Iskender: 30%
Ilker Kurtulan: 27%
Gökcer Sönmezocak: 23%
Igor Risteski: 20%

Furkan Iskender: Focused on implementation of project in TinkerCAD(Simulation, Basic Design, Coding), Testing. Estimated work done was around 40 hours.

Ilker Kurtulan: Focused on Activity Diagrams, Block Diagrams, Internal Block Diagrams and some parts of Use Case diagrams. Estimated work was around 30 hours.

Gökcer Sönmezocak: Focused on Constraint Diagrams, Fault trees and State Machine diagrams. Estimated work was around 25 hours.

Igor Risteski: Focused on Use Cases, Requirement Diagrams and Sequence Diagrams. Estimated work was around 25 hours.

https://github.com/ilkerkurtulan97/ESE_Term_Project

VIII. AFFIDAVIT

We (Igor Risteski, Gökçer Sönmezocak, Ilker Kurtulan, Furkan Iskender) 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.