Networked Traffic Control For Autonomous Cars

Ghazaleh Hadian Ghahfarokhi
Computer Science Department
Fachhochschule Dortmund
Dortmund, Germany
ghazaleh.hadianghahfarokhi001
@stud.fh-dortmund.de

Behnoosh Hashemi Hendoukoshi
Computer Science Department
Fachhochschule Dortmund
Dortmund, Germany
behnoosh.hashemihendoukoshi001
@stud.fh-dortmund.de

Shirin Babaeikouros
Computer Science Department
Fachhochschule Dortmund
Dortmund, Germany
shirin.babaeikouros001
@stud.fh-dortmund.de

Kanika Sehghal
Computer Science Department
Fachhochschule Dortmund
Dortmund, Germany
kanika.sehgal001
@stud.fh-dortmund.de

***Abstract*— Smart Intersection is expected to be an important future intelligence transport system. Decreasing energy consumption, and traffic congestion, as well as increasing road safety, are the main benefits that have brought much research to this area. In this system, sensors collect information about traffic patterns, vehicle density, and pedestrian activity at intersections in real time. This paper specifies details on various requirement diagrams and implements part of the behavior based on real situations using SysML for certain use cases like passing the intersection, managing priority, and traffic analysis. Also, it deals with the design and implementation of real-time scheduling, simulation of use cases using Tinkercad, and verification & validation using the Cunit framework. Also, the results from these tools are shown.**

Keywords—Smart Intersection, Traffic Control system, Scheduling, Simulation.

I. Introduction

In the ever-evolving landscape of smart cities, the integration of autonomous vehicles into urban mobility systems has emerged as a groundbreaking solution. Autonomous vehicles, with the potential to operate without human intervention, promise to revolutionize transportation, enhance safety, and streamline urban mobility. The urgency for such innovation stems from the alarming global toll of lives lost annually due to traffic accidents, as well as the imperative to mitigate economic losses and environmental impact associated with increasing traffic congestion [1].

In the quest for smart city applications, the need for effective traffic management becomes paramount. The population explosion and the subsequent increase in vehicles demand innovative solutions to alleviate traffic congestion and its associated economic and environmental costs. A significant stride in this direction is the design and implementation of the Smart Traffic Signal Control System (STSC). This system, equipped with multi-modal applications, including emergency vehicle signal pre-emption, transit signal priority, adaptive traffic signal control, and more, plays a pivotal role in shaping intelligent transportation systems within smart cities [2].

II. Motivation

The integration of autonomous vehicles and networked traffic control within smart city systems is motivated by a collective pursuit of safer, more efficient, and environmentally conscious urban mobility. This integration is driven by the urgent need to address the alarming toll of global traffic accidents and reduce economic losses. The aspiration is to create urban environments that are resilient and adaptable to the evolving needs of their inhabitants. The primary motivation is to enhance safety by minimizing human errors inherent in traditional traffic systems. Autonomous vehicles, equipped with advanced sensors and AI capabilities, communicate with each other and respond to real-time traffic conditions, thereby reducing the risk of accidents [1].

Efforts are directed toward improving overall transportation efficiency by optimizing traffic flow, reducing congestion, and dynamically adjusting traffic signals and routing through a centralized control system. This dynamic approach ensures a smoother flow of traffic, addressing inefficiencies present in conventional traffic management. The integration also seeks to mitigate economic losses associated with traffic accidents, thereby enhancing overall productivity and contributing to the creation of economically resilient urban environments. Additionally, the move towards autonomous and electric vehicles aligns with environmental goals, aiming to minimize air pollution and reduce the carbon footprint of urban transportation [3].

Considering the continuous growth in urban population and vehicle numbers, innovative solutions are required. The Smart Traffic Signal Control System (STSC) plays a crucial role by adapting traffic signals in real-time based on data from autonomous vehicles, addressing challenges posed by the ever-growing urban landscape. Technologically, the integration leverages advancements such as artificial intelligence, machine learning, and the Internet of Things (IoT) to create a cohesive and interconnected urban mobility ecosystem. This not only enhances transportation efficiency but also positions cities at the forefront of technological innovation.[4]

The motivation extends beyond immediate gains; it involves building resilient and adaptive urban landscapes. Smart city systems, with integrated autonomous vehicles and traffic control, can evolve to meet the changing needs of their inhabitants, contributing to the long-term sustainability of urban environments. In essence, these advancements underscore a shared commitment to shaping cities that are not only technologically advanced but also resilient, sustainable, and responsive to the evolving.

III. Analysis

A. Interaction and Communication

The effective operation of the intelligent intersection lighting system relies significantly on the interaction and communication between its components. Below are potential scenarios of interaction and communication, derived from the system's characteristics and underlying assumptions:

1) Real-time Traffic Monitoring and Adaptive Signal Control Interaction: The traffic sensors monitor the real-time traffic flow, and the data is transmitted to the signal control module. The signal module, with the help of intelligent algorithms, dynamically adjusts the signal timings based on the traffic flow to minimize congestion and optimize traffic flow.

2) Smart Lighting and Visibility Enhancement and Traffic Control Interaction: The smart lighting module constantly monitors the ambient light level and weather conditions. If visibility conditions become poor, the smart lighting system can adjust the light intensity and color temperature to improve visibility for drivers and pedestrians. The traffic control module can also work with the lighting module to optimize traffic flow based on lighting conditions.

3) Communication with Connected Vehicles and Traffic Control Interaction: The smart intersection lighting system can communicate with connected vehicles through wireless communication protocols such as Dedicated Short-Range Communication (DSRC) or Cellular Vehicle-to-Everything (C-V2X). The system can provide real-time traffic flow information and personalized alerts and instructions to connected vehicles. The traffic control module can also use the information to optimize traffic flow based on the number and location of connected vehicles.

4) Integration with Traffic Management Systems: The smart intersection lighting system integrates with existing traffic management systems to share real-time data and enable centralized traffic management. The traffic management system can monitor traffic flow across multiple intersections, gather data for analysis, and remotely manage the lights by overriding control if necessary.

B. Assumptions

The following presumptions are taken into account for networked traffic management for autonomous vehicles. Firstly, each car and pedestrian is assumed to be equipped with smart devices to transmit intersection pass requests, receive authorization to proceed, and specify the starting point and destination. Secondly, the traffic control system is expected to regulate traffic flow and execute decision-making processes. Lastly, the priority hierarchy is established as:

Emergency Car > Pedestrian > Ordinary Vehicles

While numerous potential concurrent situations for passing the intersection exist, the document concentrates on key scenarios and presents the general operational logic.

C. System Operation

When approaching an intersection, it is common to encounter situations where different vehicles are simultaneously navigating the intersection from various directions. However, this study primarily concentrates on a select number of prominent scenarios and demonstrates the underlying logic that can be applied in a general context.

Furthermore, the proposed system possesses the capability to calculate the level of congestion present on roadways. The system's response to the level of congestion on the roads is crucial for efficient traffic management. By evaluating the congestion assessment, the system adapts its operation to ensure optimal performance. This adaptation is reflected through distinct states that the system can operate in. In the low congestion state, where traffic flows smoothly with minimal congestion, the system can prioritize maximizing traffic throughput. It may adjust signal timings to facilitate faster movement, minimize waiting times at intersections, and optimize the overall efficiency of traffic flow. On the other hand, in the high congestion state, where traffic volume is significantly higher, the system focuses on congestion mitigation strategies. Additionally, two additional scenarios may arise concurrently with the aforementioned conditions, namely the passage of pedestrians or the presence of emergency vehicles.

According to Figure 1, as long as both vehicles maintain lane discipline, they can safely pass each other without any collision. Additionally, it is feasible for a pedestrian to cross the lateral road without any issues. For example, there may be cars that intend to move from South to North, while others may want to travel in the opposite direction. These movements can occur concurrently, happening at the same time.

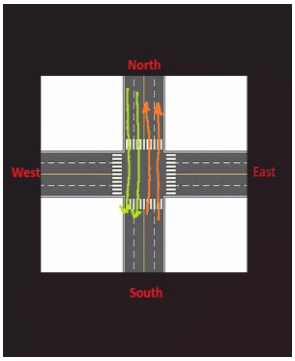

Figure 1: Vehicles entering an intersection with no interference

The system's design allows for the seamless movement of vehicles without the need to stop, as long as there is no interference with other cars. This feature is particularly applicable to situations where cars are making right turns at intersections, the system aims to improve traffic efficiency and reduce congestion. In Fig2, the scenario depicted showcases the implementation of this feature. When a car

intends to turn in the right direction, it can do so without needing to halt its movement. This is because the right-turning vehicles can navigate their turns without causing any disruption or interference with the flow of other vehicles passing through the intersection.

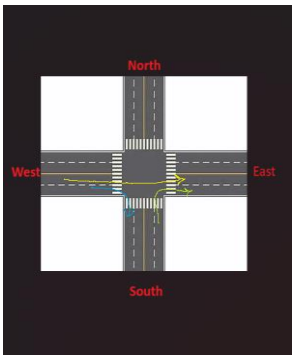

Figure 2: Vehicles entering an intersection with no interference

Figure 3 illustrates a situation where there is a significant risk of collision if proper coordination is not established among the intersecting cars. Due to the possibility of collision, our project acknowledges the need for effective management of the intersection and thus presents a priority scheduling algorithm specifically designed for this purpose. These factors include the speed of the vehicles, the distance from the entrance of the intersection, emergency vehicles, and the congestion levels of the respective roads.

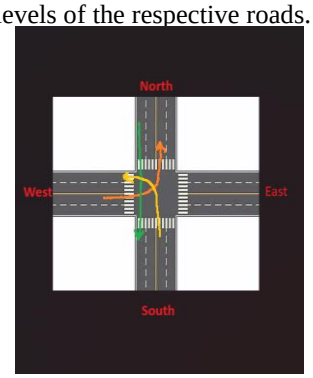

Figure 3: Vehicles entering an intersection with interference

D. Priority of scheduling

The priority of scheduling in the Smart Intersection Lighting System plays a crucial role in managing traffic flow, especially in the context of varying levels of traffic congestion. This system ensures that the allocation of right-of-way is determined in a manner that prioritizes safety and urgency. However, the scheduling algorithm also considers the prevailing traffic congestion conditions. During low congestion, the system ensures that emergency vehicles receive immediate priority, enabling them to navigate the intersection efficiently and without obstruction. Pedestrians are then given priority, allowing them sufficient time to cross safely. Autonomous vehicles are granted a lower priority in such scenarios. Conversely, during high congestion periods, the system dynamically adjusts the scheduling to allocate priority based on the congestion levels. This means that the road with higher congestion will be prioritized, ensuring smoother traffic flow, and reducing overall congestion. By considering both the priority order and traffic congestion, the smart intersection lighting system optimizes traffic management, enhances safety, and caters to the specific requirements of emergency vehicles, pedestrians, and autonomous vehicles within the intersection environment.

IV. MODEL-BASED DESIGN WITH SYSML

This section aims to conduct an in-depth analysis and design of the system through the utilization of SysML diagrams. The following paragraph presents various diagrams, including requirements, a fundamental use case, a block definition diagram, an allocation diagram, different activity diagrams, state machine diagrams, sequence diagrams, and a fault tree. These diagrams provide visual representations that aid in understanding and analyzing the system's different aspects and behaviors.

SYSTEM REQUIREMENTS

To access the predefined goals of the project and ensure the development of a proficient system, the initial stages of the project involved the systematic collection, analysis, specification, documentation, and validation of the requirements. In Fig 4, a requirement diagram is shown.

1) Intersection traffic behavior: The system needs to make sure every vehicle and pedestrian passes safely.
2) Collision Avoidance: Each vehicle should utilize algorithms to predict potential collisions.
3) Traffic Management: The system should be able to monitor and manage traffic flow.
4) Optimize Waiting Time: It must be able to enhance traffic flow and reduce delays.
5) Warning Signal: It must be able to possess the capability to notify relevant individuals or entities promptly and effectively.
6) Communication with Autonomous Vehicles: Smart streetlights should be able to communicate with autonomous vehicles, exchanging information about traffic conditions, upcoming signal changes, and potential hazards.
7) Emergency Vehicle Priority: It should be able to navigate through traffic with minimal disruption and ensure rapid response times during emergencies.
8) Calculating Traffic Flow: It should be to calculate traffic congestion to reduce travel time and enhance overall traffic efficiency.
9) Communication Behavior: It should be able to exchange information and signals among various components.
10) Sensor Integration: The system should be able to integrate various sensors, such as motion sensors, ambient light sensors, and weather sensors, to gather data on traffic, pedestrian activity, and environmental conditions.
11) V2X Data: It should be able to facilitate real-time communication.
12) Failure and Hazard strategies: They should be able to robust and reliable to potential failure.
13) Communication failure: The system should be able to implement robust backup mechanisms.
14) Accident signals: They should be able to identify the occurrence of an accident and initiate a coordinated response.

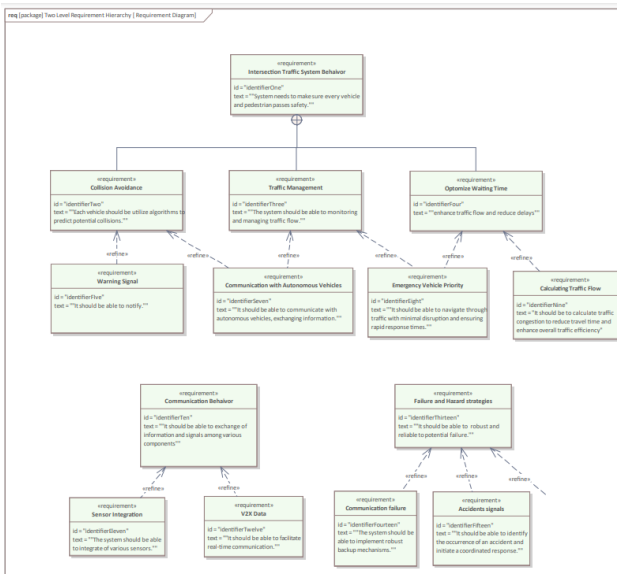

Figure 4: Requirement diagram

USE CASE DIAGRAM

In the context of a smart intersection lighting system, the actors can be the traffic controller, pedestrians, autonomous vehicles, emergency vehicles, and sensor module. The use case for the system is shown in Figure 5.

The Traffic Controller is responsible for managing and controlling the traffic flow at the intersection. They can adjust the timing of the traffic lights based on the traffic conditions. Pedestrians can also interact with the smart intersection lighting system by pressing a button to activate a pedestrian crossing signal.

The system itself can detect the presence of vehicles at the intersection through various sensors. Based on the detected presence of vehicles and the pedestrian crossing signal, the system can change the state of the traffic lights to control the flow of vehicles and ensure safe crossing for pedestrians. Additionally, the system can monitor the traffic flow at the intersection and provide data for analysis and optimization purposes. The use case diagram shows the relationships between the actors and the use cases they interact with.

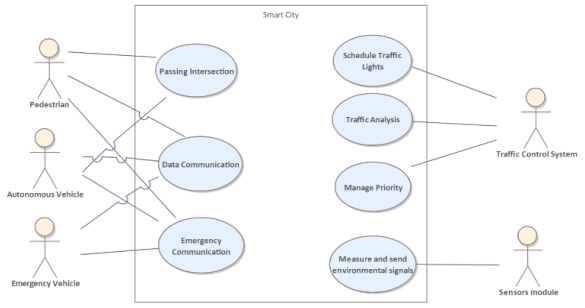

Figure 5: Use case diagram

PEDESTRIAN STATE MACHINE

The pedestrian state machine depicts the sequence of states pedestrians experience when approaching a crosswalk within our integrated traffic system. As outlined in the diagram, the pedestrian begins in the "Waiting" state, transitions to the "Crosswalk" state upon signal activation, and moves into a "Countdown" phase. During this countdown, the possibility of encountering an obstacle may prompt an immediate shift to an "Emergency Stop" state, prioritizing safety measures. This seamless integration with the traffic control system ensures efficient traffic flow and safety for all road users.

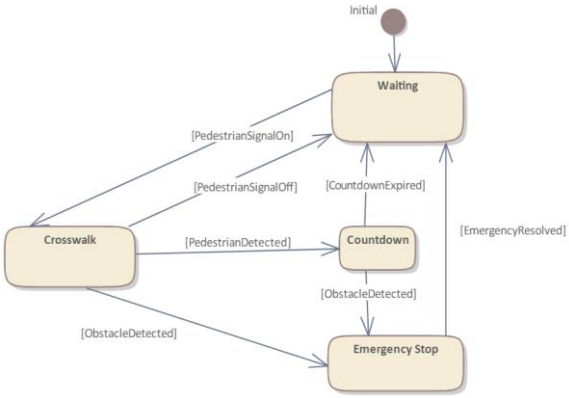

Figure 6: Pedestrian State Machine diagram

In planning for future advancements, the pedestrian state machine is designed to adapt as autonomous cars and emergency vehicles become more integral to traffic management. This future-proofing ensures our system's continual adaptability beyond its current scope.

TRAFFIC FLOW OPTIMIZATION STATE MACHINE

In the traffic flow optimization state machine, the system's capability to evolve through various states in response to traffic data is highlighted. The system's core, the "Normal" state, dynamically adjusts to traffic conditions and further refines into "Low Congestion" and "High Congestion" states. Notably, the integration of "Pedestrian Mode" and "Emergency Mode" states demonstrates the system's adaptability to diverse traffic scenarios while prioritizing pedestrian safety and effectively responding to emergency situations.

The traffic flow control system, as epitomized by its meticulously crafted state machine, is engineered to enhance urban mobility by efficiently adapting to and optimizing traffic flow. From transitioning through states to accommodate varying traffic conditions, pedestrians, and emergency scenarios, the system stands as a testament to our dedication to efficient, safe, and adaptable traffic management.

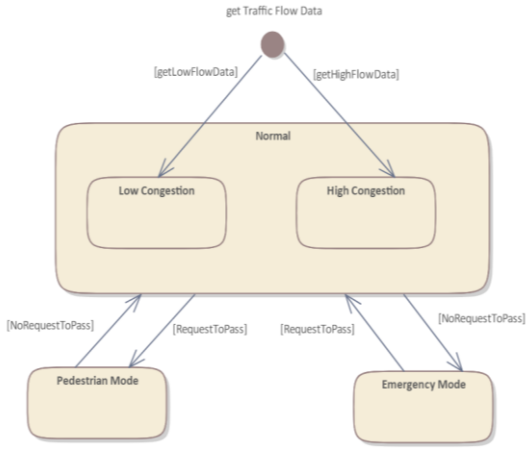

Figure 7: Traffic Flow Optimization State Machine

Operational Flow State Machine

The operational flow state machine outlines the six key states guiding the system's operational sequence, from "Idle" readiness to "Planning," "Navigating," and "Stopped," as well as the transitions involved in task execution and emergency response. This adaptive design ensures seamless progression through various operational states, efficiently transitioning between static and dynamic states based on real-time environmental cues. The state machine diagram serves as a comprehensive framework for efficient and intelligent task management, emphasizing adaptability and responsiveness in the face of diverse operational challenges, thereby underscoring the system's sophisticated operational flow.

Communication Failure Handling

The activity diagram for handling critical situations showcases the system's rapid response to communication failures, accidents, and extreme weather conditions. It presents a comprehensive approach to ensuring safety and system resilience during challenging scenarios.

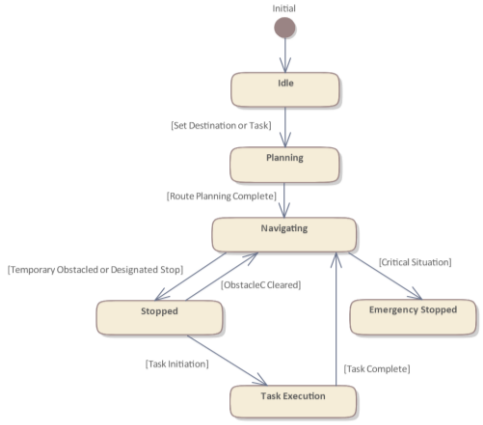

Figure 8: Operational Flow State Machine Diagram

Figure 9: Activity Diagram for Communication Failure Handling

The activity diagram begins with a critical decision point the "Communication Failure Signal." The system promptly evaluates whether a communication failure has occurred, thereby triggering a series of decisive actions. Upon ruling out communication failure, the system proceeds to check for "Accident Detection," swiftly handling any identified accidents and notifying emergency services for a coordinated response.

In cases where no accidents are detected, the system moves on to evaluate "Extreme Weather Conditions," adjusting traffic lights and notifying emergency services to address potential safety risks associated with extreme weather. Following the handling of critical situations, the system engages in proactive "System Maintenance and Recovery" actions to ensure sustained functionality and preparedness for future challenges.

Traffic Flow Prioritization

Priority Decision Nodes

The activity diagram initiates crucial decision points, determining the prioritization of emergency vehicles, pedestrians, and traffic flow management in potentially congested and challenging conditions.

Pedestrian Priority

After prioritizing emergency vehicles, the system evaluates the need to "Prioritize Pedestrians," entering the "Pedestrian State" when necessary to ensure safe passage for pedestrians.

Traffic Flow Management

Subsequently, the system assesses "High Congestion Detection," dynamically adjusting to high traffic scenarios and updating street priorities and traffic flow permissions based on the detected congestion levels and traffic conditions.

Extreme Weather Condition Handling

Finally, the diagram accounts for "Extreme Weather Conditions," demonstrating the system's adaptability by effectively managing extreme weather scenarios to maintain

an optimal balance between safety and efficiency. In conclusion, the activity diagram illustrates the dynamic and adaptive nature of the system, showcasing its ability to prioritize emergency vehicles and pedestrians, efficiently manage traffic flow, and adapt to extreme weather conditions to ensure the safety and efficiency of the overall traffic management system.

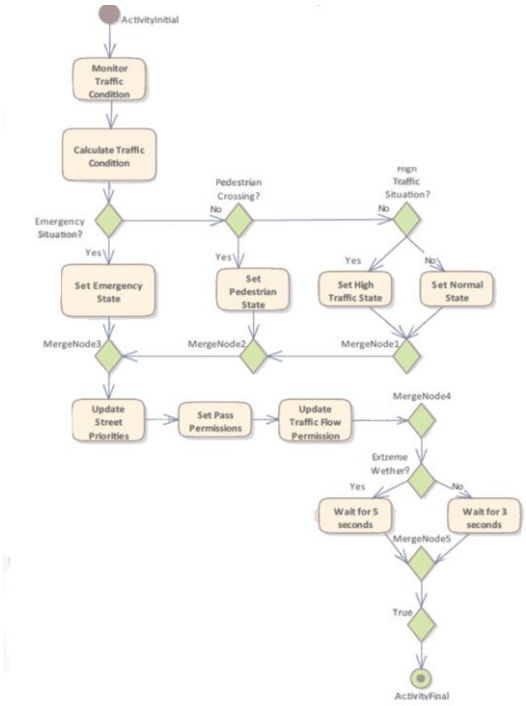

Figure 10: Activity Diagram for Traffic Flow Prioritization

SEQUENCE FLOW

A sequence diagram is a visual representation that illustrates the interactions and flow of messages between different components or actors within a system over time. In the context of a smart traffic control system at an intersection with priorities for emergency vehicles and pedestrians, the sequence diagram would capture the dynamic behaviour during specific scenarios

Participant/actors

1) Intersection Controller: This represents the central control system managing traffic signals for pedestrians.
2) Emergency Vehicle: Represents a vehicle with emergency status (e.g., ambulance, fire truck).
3) Pedestrian Crossing System: This represents the system managing pedestrian signals and crossings.
4) Traffic Monitoring System: Represents the system responsible for monitoring traffic conditions.
5) Autonomous Vehicles: This represents the autonomous vehicles equipped with vehicle-to-vehicle communication.

SEQUENCE DIAGRAM DESCRIPTION

The sequence diagram begins with the initialization of the smart traffic control system at the intersection.

1) Normal Traffic Flow (Autonomous Vehicles): Autonomous vehicles communicate with each other for efficient traffic flow without the need for physical traffic lights.
2) Emergency Vehicle Activation: When an emergency vehicle approaches the intersection, it sends a request message to the Intersection Controller, indicating its status and the urgency of the situation.
3) Highest Priority Handling (Emergency Vehicle): The Intersection Controller, upon receiving the emergency vehicle request, prioritizes the path for the emergency vehicle. Autonomous vehicles adjust their routes based on the emergency vehicle's priority.
4) Pedestrian Priority Handling (Physical Traffic Light): Simultaneously, the Pedestrian Crossing System is notified of the emergency vehicle's approach. The Intersection Controller activates the traffic light for pedestrians, indicating a stop signal.
5) Normal Traffic Resumption (Autonomous Vehicles): Once the emergency vehicle has cleared the intersection, autonomous vehicles resume normal traffic flow without the need for physical traffic lights.
6) Congestion Priority Handling (Autonomous Vehicles): The intersection controller communicates with autonomous vehicles to optimize traffic flow in congested areas and the Traffic Monitoring System detects increased traffic density among autonomous vehicles.
7) End of Scenario: The sequence diagram concludes, representing the successful handling of emergency vehicle priority, pedestrian safety with a physical traffic light, and congestion management for autonomous vehicles.

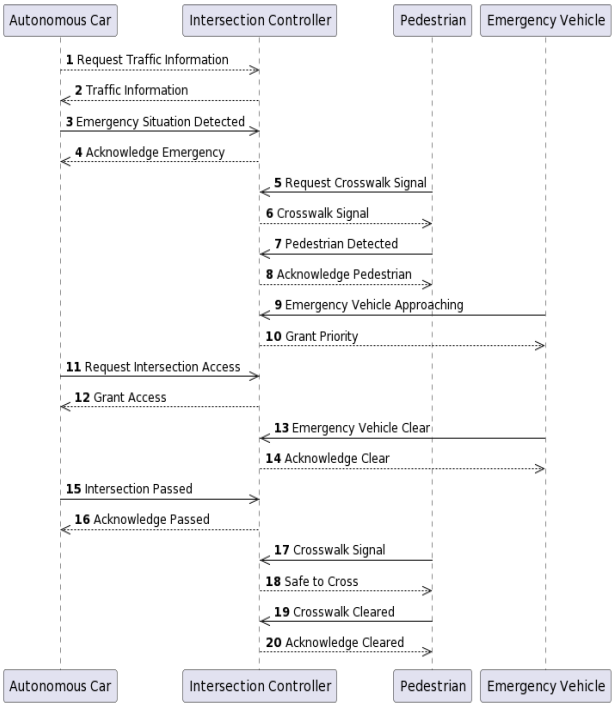

Figure 11: Sequence Diagram for Traffic Flow Prioritization

CONTEXT DIAGRAM

A context diagram provides an overview of a system, its external entities, and the interactions between them.

External Entities:

1) Pedestrian Crossing Sensors: Represented as an external entity providing input signals related to the presence of pedestrians at the crossing.
2) Weather Sensors: Represented as an external entity providing input signals related to weather conditions.
3) Camera and Ultrasonic Sensors: Represented as external entities providing input signals related to traffic conditions, including vehicle positions and congestion.

System Components:

1) Intersection Controller: Positioned at the center of the diagram, the Intersection Controller serves as the central component managing the smart traffic control system. It receives input signals from various sources and makes decisions to control traffic and prioritize vehicles.
2) Traffic Data Analysis: Represented as an external entity that receives output signals from the Intersection Controller. It processes the traffic data for analysis and future planning.
3) Permission to Autonomous Vehicles: Represented as an external entity receiving output signals from the Intersection Controller. It communicates with autonomous vehicles to grant permission for movement based on traffic conditions.
4) Permission to Emergency Vehicles: Represented as an external entity receiving output signals from the Intersection Controller. It communicates with emergency vehicles to grant permission and provide a clear path through the intersection.

Interactions:

Input Signals to Intersection Controller:

1) Position, speed, and intention signals from autonomous vehicles.
2) Status, urgency, and path signals from emergency vehicles.
3) Data from pedestrian crossing sensors indicating pedestrian presence.
4) Weather information from weather sensors.
5) Traffic conditions, including vehicle positions, from camera and ultrasonic sensors.

Output Signals from Intersection Controller:

1) Signals granting permission to autonomous vehicles.
2) Signals granting permission to emergency vehicles.
3) Traffic control signals for pedestrians (e.g., physical traffic lights).

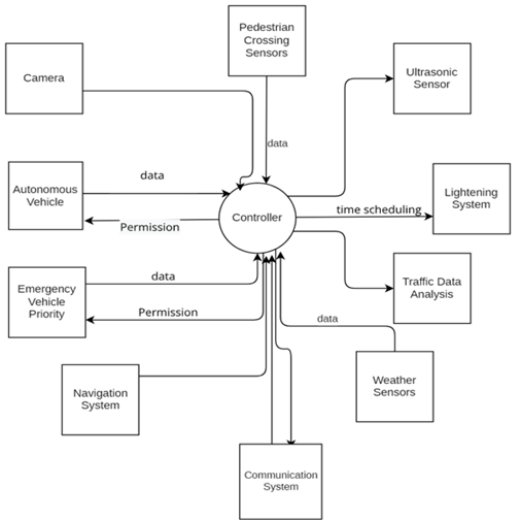

Figure 12: Context Diagram for Traffic Flow Management

ALLOCATION DIAGRAM

Allocation Diagram showcases a seamless integration of key components. Traffic sensors, strategically positioned throughout the network continuously collect real-time data on vehicle movement and road conditions. This info flows into the data processing and analysis component, where a sophisticated algorithm interprets the data enabling quick decision-making. The Security System ensures the integrity of the processed data, safeguarding against potential threats. Together these elements form a robust system that enhances traffic flow, safety, and overall urban mobility.

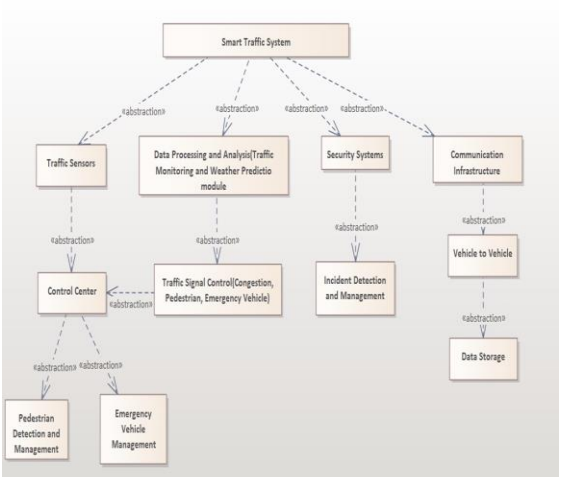

Figure 13: Allocation Diagram for Smart Traffic Systems

INTERNAL BLOCK DIAGRAM

This block diagram represented below has a centralized traffic management system that serves the central unit for smart intersections. It receives input from various sources to make decisions and manages the overall traffic flow.

The vehicle represents the vehicle approaching the intersection, equipped with V2V communication to interact with traffic management. It sends data such as speed, location, and vehicle status to the centralized system and, upon receiving a signal from the controller takes action. There is a block for environmental sensors that monitors the conditions and provides data such as weather, visibility, and

road conditions. It helps the Centralized System make informed decisions during bad weather.

A timer controls the duration of the signal, which includes a countdown timer for the pedestrian and synchronizes with the Traffic Management System. Vehicles and pedestrians interact with the system by sending and receiving data, ensuring safe and efficient traffic flow.

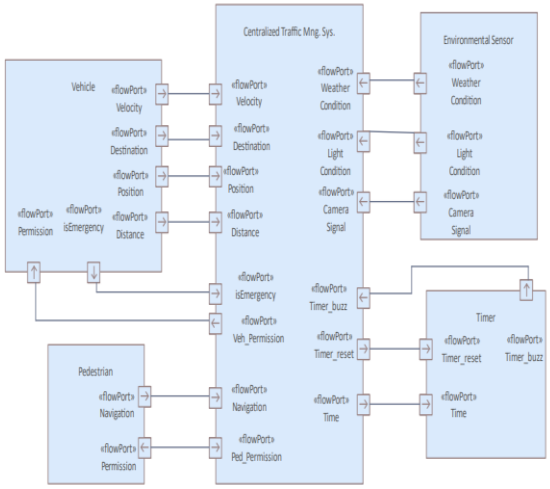

Figure 14: Inner Block Diagram for Traffic Flow Management

V. IMPLEMENTATION

The implementation of the project is divided into two parts of software and hardware developments. Coding is developed using C++ programming language to illustrate the output of the desired intersection system. Hardware part is simulated in Tinkercad using microcontrollers being programmed by the software codes, as well.

A. Coding

In software side, there are five main classes: Pedestrian, Car, Street, Intersection, and Controller. These classes hierarchically are inherited from each other, for instance, the street has the features of cars and pedestrians from the corresponded child classes, respectively. The attributes and methods in the classes in a large extent demonstrate the I/O signals and responsive behaviors of each object interacting with the whole system. For example, the class Controller is responsible for managing and scheduling the whole traffic flow at an intersection. It gets the cars' information in 500 m radius from the intersection through cloud, as well as weather condition data form sensor module. The vehicle data contains the id, whether it's emergency or not, the distance from the intersection, the speed of the vehicle, and the destination to follow. To count the congestion, the number of cars entering a street is calculated. The sensors also are there to detect the number of cars entering and leaving the intersection. These information help to decide on which level of congestion the intersection is at the moment and by that controller can run the suitable algorithm to prioritize the streets and in turn the vehicles to pass. At the same time, pedestrians might send request to get permission to pass or the sensors inform the system when they are crossing the street. Last, this controller class is also responsible for sending the messages to the vehicles. The message shows the permission that a car is allowed to pass or the waiting time it must stop at the intersection point and wait for other vehicles to get pass first. As in previous section mentioned, in this project, the priority order of the streets are set first to street with an emergency car, next with pedestrian, then high congestion street and last low congestion street including normal vehicles. Pedestrian, Car, Street and Intersections classes are shown at Figure 15, Figure 16, Figure 17, Figure 18, respectively

```
class Pedestrian
{
public:
    int id;
    Direction destination;
    bool isWaitingResponse;
    bool isPermitted;

    Pedestrian(int id, Direction destination);
    void setPermission(bool newPermission);
};
```

Figure 15: Pedestrian Class

```
class Car
{
private:
    int id;
    double velocity;
    bool isEmergency;
    std::pair<int, int> location;
    Direction origin;
    Direction destination;
    double distance;
    bool isWaitingResponse;
    bool isPermitted;
```

Figure 16: Car class

```
class Street
{
private:
    int id;
    Direction direction;
    State state;
    std::vector<Car> cars;
    std::vector<Pedestrian> pedestrians;

public:
    Street(int id, Direction direction, State state);  // Constructor
    bool calculateTraffic(std::vector<Car> cars) const;
    void setCarPermission(int carId, bool permission);
    void setPedestrianPermission(int pedestrianId, bool permission);
    void sortStreetCarsByDistance(const Street &street);
```

Figure 17: Street class

```
class Intersection
{
public:
    int id;
    std::vector<Street> streets;
    State state = NORMAL;
    bool isExtremeWeather;
    bool isDay;

    Intersection(int id, bool isExtremeWeather);
    void addStreet(const Street &street);
    void addCarsToStreet(int streetId, const Car &car);
    void addPedestrianToStreet(int streetId, const Pedestrian pedestrian);
    bool isThereEmergencyStreet();
    bool isThereHighCongestionStreet(std::vector<Street> streets);
    bool isTherePedestrianStreet(std::vector<Street> streets);
    const std::vector<Street> getStreets();

    void sortAllIntersectionCarsByDistance(std::vector<Street> streets);
};
```

Figure 18: Intersection class

In Figure 19, there is a function which specifying the state of each street. There are four different states of EMERGENCY, PEDESTRIAN, HIGH CONGESTION, and NORMAL.

```
std::vector<Street> streets = intersection.getStreets();
for (auto &street : streets)
{
    if (street.hasEmergencyCar(street.cars))
    {
        street.state = State::EMERGENCY;
    }
    else if (street.hasPedestrian(streets.pedestrian))
    {
        street.state = State::PEDESTRIAN;
    }
    else if (street.isTrafficHigh(street.cars))
    {
        street.state = State::HIGHTRAFFIC;
    }
    else
    {
        street.state = State::NORMAL;
    }
}
```

Figure 19: A method for street prioritization

In Figure 20, some parts of code is shown to check and set permission to an emergency car.

```
if (street.state == EMERGENCY)
{
    std::vector<Car> cars = street.getCars();
    std::vector<Pedestrian> pedestrians = street.getPedestrians();
    for (auto &car : cars)
    {
        if (car.isEmergency)
        {
            street.setCarPermission(car.id, true);
            if (intersection.isExtremeWeather)
            {
                street.setCarVelocity(car.id, extreme_weather_speed_limit);
            }
            else
            {
                street.setCarVelocity(car.id, speed_limit);
            }
        }
        else
        {
            street.setCarPermission(car.id, false);
            street.setCarVelocity(car.id, 0);
        }
    }
    for (auto &pedestrian : pedestrians)
    {
        street.setPedestrianPermission(pedestrian.id, false);
    }
```

Figure 20: A method for emergency permission

One sample dataset is used to run and check the software. Four streets, three normal cars, one emergency car and one pedestrian were added to an intersection id 1, see Figure 21.

```
Street street1(0, SOUTH, State::NORMAL);
Car car1(0, 35, false, {20, 30}, SOUTH, EAST, 50, false, false);
Pedestrian pedestrian1(1, SOUTH);
Street street2(1, WEST, State::NORMAL);
Car car2(1, 72, true, {60, 10}, WEST, EAST, 10, false, false);
Street street3(2, NORTH, State::NORMAL);
Car car3(2, 43, false, {205, 300}, NORTH, SOUTH, 300, false, false);
Street street4(3, EAST, State::NORMAL);
Car car4(3, 31, false, {100, 100}, EAST, NORTH, 100, false, false);
Intersection intersection(1, true, false);
intersection.addStreet(street1);
intersection.addStreet(street2);
intersection.addStreet(street3);
intersection.addStreet(street4);
intersection.addCarsToStreet(0, car1);
intersection.addCarsToStreet(1, car2);
intersection.addCarsToStreet(2, car3);
intersection.addCarsToStreet(3, car4);
intersection.addPedestrianToStreet(0, pedestrian1);

Controller controller(1, intersection);
```

Figure 21: Sample dataset for simulation

By running the program, as the result shows in Figure 22, the emergency car with id 102 in West street got permitted to cross the intersection and the others must stop and wait for permission. More details about the software implementation can be seen at this GitHub repository.

```
Car Id: 101 in Street: SOUTH with Priority: PEDESTRIAN,  Velocity: 0, waiting for permission.
Car Id: 102 in Street: WEST with Priority: EMERGENCY,  Velocity: 50, is passing.
Car Id: 103 in Street: NORTH with Priority: NORMAL,  Velocity: 0, waiting for permission.
Car Id: 104 in Street: EAST with Priority: NORMAL,  Velocity: 0, waiting for permission.
Pedestrian Id: 201 in Street: SOUTH with Priority: NORMAL, waiting for permission.
```

Figure 22: Result of simulation dataset

B. *Testing*

Software testing is a critical phase in the development cycle of any application. It involves the systematic evaluation of software components to verify that they meet specified requirements and function correctly. Testing helps identify defects and ensures that the software behaves as expected under different conditions, enhancing its quality and reliability.

Software testing encompasses various types of testing, each serving a distinct purpose in ensuring the quality and reliability of applications. End-to-end testing evaluates the entire application flow from start to finish, simulating real-world user scenarios to validate the system's behavior and functionality. Integration testing focuses on testing the interactions between different modules or components of the software to ensure that they work together seamlessly. Unit testing, on the other hand, targets individual units or components in isolation, verifying their correctness and functionality independent of other parts of the system. It aims to validate the functionality of each unit and detects any defects early in the development process. Unit tests are typically automated and run frequently during development, providing quick feedback to developers and helping maintain code integrity and stability. In this project, two unit tests are developed to evaluate the functionality of two methods “SetCarPriority” and “CalculateTrafficCongestion”, as shown in Figure 23 and Figure 24.

```
TEST(SetStreetPriorityTest, EmergencySituation)
{
    // Arrange
    Intersection intersection(1, true);
    Street emergencyStreet(1, SOUTH, EMERGENCY);
    intersection.addStreet(emergencyStreet);
    Controller controller(1, intersection);
    // Add an emergency car to the street
    Car emergencyCar(1, 50, true, {0, 0}, SOUTH, EAST, 0, false, false);
    intersection.addCarsToStreet(1, emergencyCar);

    // Act
    controller.setStreetPriority(intersection);

    // Assert
    EXPECT_EQ(emergencyStreet.getState(), EMERGENCY);
};
```

Figure 23: Unit test to set priority

```
TEST_F(IntersectionTest, TrafficCongestionTest)
{
    // Assuming the implementation of calculateTrafficCongestion is in your Intersection class
    bool congestion = controller.calculateTrafficCongestion(intersection);

    // Assert that the method returns the expected result
    ASSERT_FALSE(congestion);
}
```

Figure 24: Unit test to check traffic flow

The results of the test cases are presented in Figure 25. The successful passed message confirms that the functions written above are working properly.

```
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 2 tests from SmartTrafficMngStstem
[ RUN      ] SmartTrafficMngStstem.PositiveNos
[       OK ] SmartTrafficMngStstem.PositiveNos (0 ms)
[ RUN      ] SmartTrafficMngStstem.NegativeNos
[       OK ] SmartTrafficMngStstem.NegativeNos (0 ms)
[----------] 2 tests from SmartTrafficMngStstem (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 2 tests.
```

Figure 25: Results of unit tests

C. *Scheduling*

Scheduling algorithms play a vital role in computer systems by overseeing resource allocation and task coordination. Particularly in real-time systems, where meeting deadlines is imperative, these algorithms are essential for timely task completion and efficient resource utilization. They are broadly categorized into Dynamic and Static types. In this project, two different scheduling algorithms of Earliest Deadline First (EDF) from dynamic category and Rate Monotonic Scheduling (RMs) from static one is applied. To do so, the utilization of the CPU should be calculated according to the tasks are running on the system. The important parameters are execution time(C), deadline(D), arrival time(A), and task period(T).

Since measuring the execution time for all inputs and all hardware states is not feasible in practice. Generally, it needs approximation. In this project hardware simulation is implemented to measure a more reliable execution time for running tasks. The application in a specific situation when three objects normal vehicle, emergency vehicle and a pedestrian at the same time are requesting to pass and one controller for managing the requests is implemented on Tinkercad, as shown in Figure 26. Four microcontrollers Arduino Uno R3 with 1.2 MHz frequency are playing the roles of the objects in the system. The LEDs are displaying the communication signals including sending requests and getting permissions. In this case, there are two tasks of reading inputs and doing priority and permission calculations are defined. By running the simulation, it can be seen that first the permission for emergency vehicle is set, then the pedestrian and the last is normal car. to get the running time of the code. The calculated time for our tasks are printed in

Figure 27. More details about the hardware simulation can be found here.

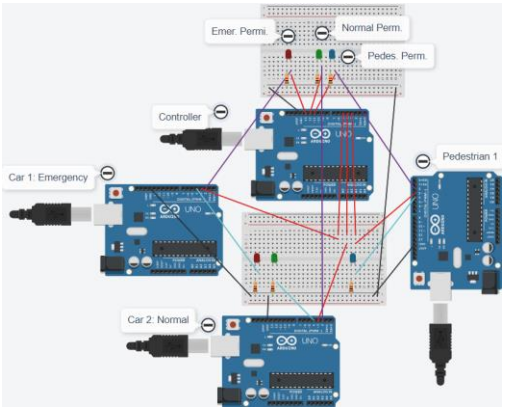

Figure 26: Simulation in Tinkercad

```
Emergency: 1
Pedestrian: 1
CAr: 1
Emergency condition
Set Permission Execution Time (us): 35372
Execution Time of Reading Inputs(us): 52
Emergency: 1
Pedestrian: 1
CAr: 1
Emergency condition
Set Permission Execution Time (us): 35380
Execution Time of Reading Inputs(us): 52
Emergency: 1
Pedestrian: 1
CAr: 1
Emergency condition
Set Permission Execution Time (us): 35376
Execution Time of Reading Inputs(us): 52
```

Figure 27: Execution time for two tasks

Table 1: Task Information

Execution time/ tasks	Reading Inputs	Set Permission
Ai	0	0
Ci	0.7 ms	40ms
Di	200 ms	200 ms
Ti	200 ms	200 ms

To ccalculate the utilities of EDF and RMS algorithms according the data on the Table 1, two following equations are used, respectively :

$$U_{EDF} = \sum_{i=1}^{n} \frac{C_i}{T_i} = \left(\frac{0.7}{200}\right) + \left(\frac{40}{200}\right) \sim 0.2 < 1$$

$$B(n) = n * \left(2^{\frac{1}{n}} - 1\right) = 2 * \left(2^{\frac{1}{2}} - 1\right) \sim 0.83 < 1$$

The calculation result shows, both EDF and RMS are feasible to be used in this project.

VI. CONCLUSION

Generally, a reliable real-time traffic flow management system for autonomous vehicles would bring about positive impacts on saving time and energy of the citizens, smoothing the traffic congestions on a smart city, subsequently protecting the environment. Since it needs an advanced infrastructure and technologies in terms of not only the autonomous vehicles but also well-equipped centralized controller systems, this system would be broadly used in future decades. However, through this small scaled project, many various aspects of a smart traffic control management system at an intersection like system requirements, system modeling, design and implementation technologies in both software and hardware perspectives, potential challenges are discussed and brainstormed, providing a more sensible overview of a real system.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Prof. Dr. Stefan Henkler, for their invaluable guidance and support in the completion of this project.

References

[1] V. Vishnevsky and D. Kozyrev (Eds.), "DCCN 2015, CCIS 601, pp. 1–10, 2016, Springer International Publishing Switzerland. DOI: 10.1007/978-3-319-30843-2 1 (2016).

[2] Mario Garzón, Anne Spalanzani. A hybrid simulation tool for autonomous cars in very high-traffic scenarios. ICARCV 2018 - 15th International Conference on Control, Automation, Robotics and Vision, Nov 2018, Singapore, Singapore. pp.1-6. ffhal-01872095f.

[3] T. Wietholt and J. Harding, "Influence of dynamic traffic control systems and autonomous driving on motorway traffic flow," Transportation Research Procedia, Available online 14 June 2016, Version of Record 14 June 2016. DOI: 10.1016/j.trpro.2016.06.015

[4] Wei-Hsun Lee and Chi-Yi Chiu, "Design and Implementation of a Smart Traffic Signal Control System for Smart City Applications," Sensors, vol. 20, no. 2, January 2020, Article 508. DOI: 10.3390/s20020508. (License: CC BY 4.0).

AFFIDATIV

We “Behnoosh Hashemi Hendoukoshi”, “Ghazaleh Hadian Ghahfarokhi”, “Shirin Babaeikouros”, “Kanika Sehghal” herewith declare that we have composed the present paper and work by ourselves 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.

Dortmund, 31.01.2024

Behnoosh Hashemi Hendoukoshi

Ghazaleh Hadian Ghahfarokhi

Shirin Babaeikouros

Kanika Sehghal