Autonomous Intersection Management System

Electronic Engineering A Lab

1st Ahmed Hemly
dept. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany

2nd Ammar Imran Khan
dept. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany

2nd Kashif Raza
dept. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany

***Abstract*—Autonomous vehicles (AVs) are set to revolutionize transportation with promises of improved safety, efficiency, and convenience. However, managing AVs at intersections remains a major challenge due to the need for precise coordination and control to ensure smooth traffic flow and prevent accidents. This paper presents a new solution for an autonomous intersection controller aimed at addressing these challenges. Our system uses advanced algorithms, real-time data, and vehicle-to-infrastructure (V2I) communication to optimize traffic movement and reduce delays. By guiding AVs through intersections, our controller ensures safe and efficient passage. We validate the effectiveness of our solution through simulations and practical tests, demonstrating its potential to enhance intersection management and support the broader adoption of AVs in cities.**

***Index Terms*—FreeRTOS, VHDL,Uppaall, Intersection management, Intelligent intersection controller, V2I, autonomous vehicles.**

I. INTRODUCTION

The rise of autonomous vehicles (AVs) is poised to significantly reshape the future of transportation. These self-driving cars promise not only to enhance road safety and reduce traffic congestion but also to provide greater convenience and mobility for users. However, the integration of AVs into existing traffic systems presents a host of challenges, particularly at intersections where coordination between multiple vehicles is crucial. Intersections are complex environments where traffic from different directions converges, necessitating precise timing and control to avoid collisions and minimize delays. Traditional traffic management systems, which rely heavily on static signals and human decision-making, are not equipped to handle the dynamic and rapid decision-making capabilities of AVs. Therefore, there is a critical need for an advanced intersection control system that can seamlessly manage the interactions of autonomous vehicles.

In this work, we propose an autonomous traffic intersection controller utilizing Vehicle-to-Infrastructure (V2I) communication. We selected V2I over Vehicle-to-Vehicle (V2V) communication due to the high data transmission volume in V2V systems, which can result in delays and noise. V2I communication provides greater noise resistance and reliability, making it more suitable for our system.

This paper is structured into four main sections. The first section outlines the concept with UML and SysML diagrams. The second section focuses on the verification and validation of the proposed approach using UPPAAL. The third section of this paper delves into the practical implementation of our proposed autonomous traffic intersection controller. This implementation is divided into two distinct parts: FreeRTOS and VHDL, which are thoroughly explained in sections four and five. Finally, the paper concludes with a summary of findings and implications drawn from our research.

II. SYSML MODELLING

In this section,we are presenting different models to illustrate the core functionalities of V2I communication in our autonomous traffic intersection controller ,we start with a use case diagram first, then we outline state transitions to show how the system manages different operational states. Next, we specify the essential requirements guiding the development of V2I communication. Finally, an activity diagram details the operational flow within the server or interface, providing a clear depiction of how V2I communication functions in practice. These models succinctly capture our approach to improving intersection management through autonomous technologies.

A. Use-Case Diagram

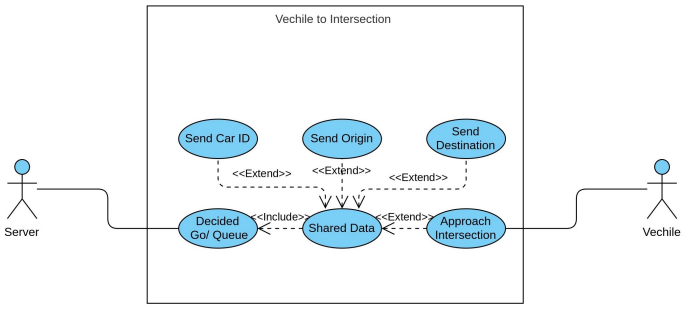


Fig. 1. Use Case Diagram

Figure. 1 describes the process of how the vehicle is communicates with an intersection management unit, as it is shown in the figure once any vehicle is approaching the intersection, it will share critical data, such as its unique Car ID, origin, and destination. And then according to all

this information that sent to the server in addition to the proximity of each vehicle, the intersection management unit will decide whether the vehicle can proceed and cross or it will be queued. So, we can guarantee that the procedure of crossing the intersection is working in a smooth flow and the traffic is managed efficiently which will help to decrease the probability of accidents.

B. State Machine Diagram

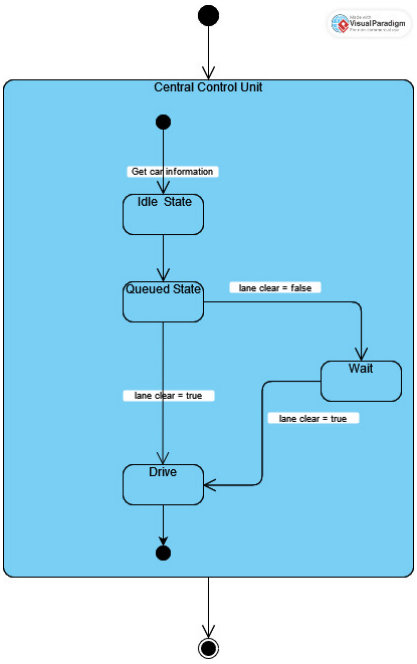


Fig. 2. State Machine Diagram

This diagram in Figure.2 outlines the various states that autonomous vehicles experience during the intersection crossing process. Initially, the vehicle is in an Idle state. Upon approaching an intersection, it identifies its location and communicates this along with its desired destination to the intersection interface. Subsequently, the vehicle's ID is added to the entry queue. If the lane requested by the vehicle is available, the intersection manager grants permission for it to proceed. Otherwise, the vehicle enters a waiting state until it receives sequential instructions for crossing the intersection. Once permitted, the vehicle proceeds through the intersection. Upon exiting the intersection zone, the cycle repeats for subsequent vehicles. This process ensures orderly and efficient traffic management at intersections for autonomous vehicles.

C. Requirement Diagram

The essential functionalities and capabilities expected from an autonomous intersection control system are outlined in Figures 3 and 4. These requirements are crucial for the successful implementation and operation of the system.

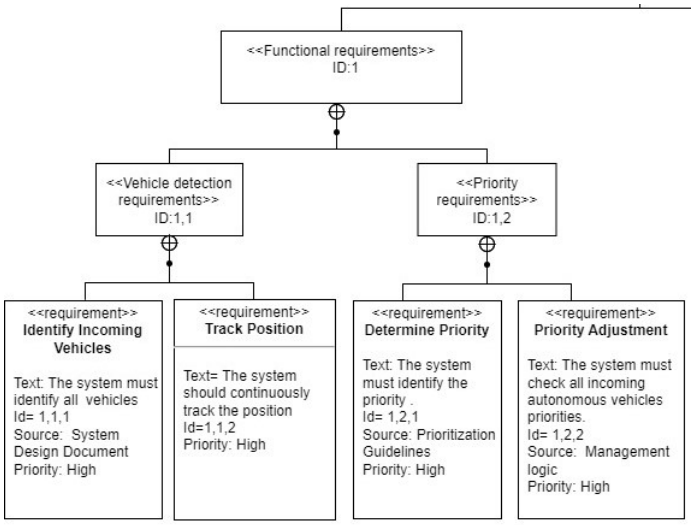


Fig. 3. Functional Requirement

The **Functional requirements** as shown in Figure.3 include the need for the system to identify all incoming vehicles and continuously track their positions, both of which are prioritized as high. Additionally, the system must determine and adjust the priority of each vehicle, ensuring efficient traffic management based on the established prioritization guidelines and management logic, again marked with high priority.

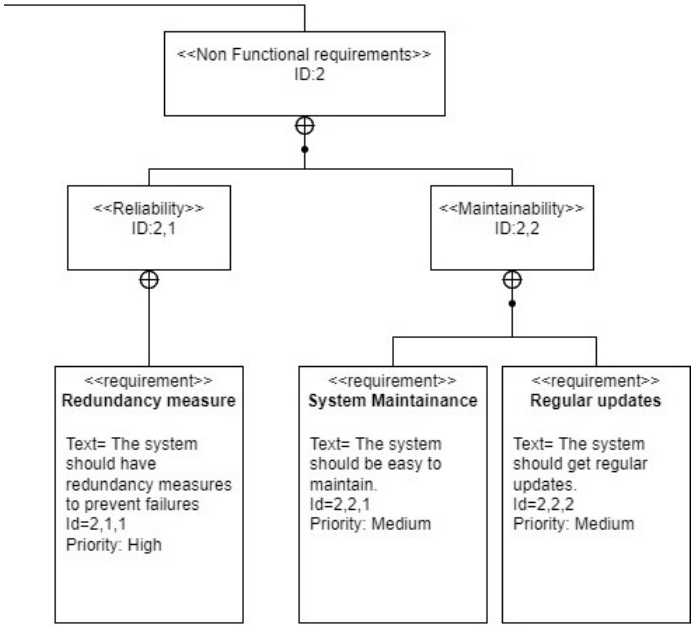


Fig. 4. Non-Functional Requirement

In terms of **Non-Functional requirements** as shown in Figure.4 , the system must be reliable, incorporating redundancy measures to prevent failures, which is a high-priority requirement. Maintainability is also crucial, with the system needing to be easy to maintain and receive regular updates,

both of which are considered medium priorities. These requirements ensure that the system remains robust and up-to-date, facilitating smooth and efficient traffic flow at intersections managed by autonomous vehicles.

D. Activity Diagram

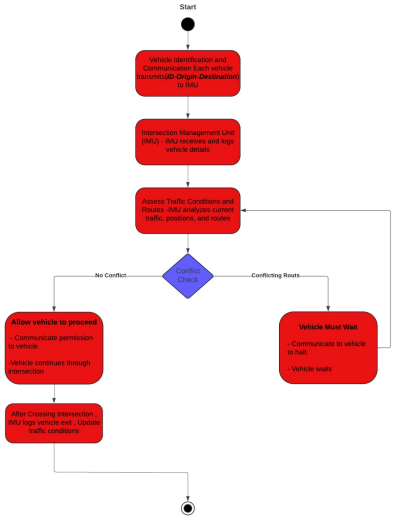


Fig. 5. Activity Diagram

The Activity Diagram in Figure.5 illustrates the process flow for an Autonomous Intersection Traffic Management system. It begins with the *Vehicle Identification and Communication* phase, where each vehicle transmits its ID, origin, and destination to the Intersection Management Unit (IMU). The IMU receives and logs these vehicle details. Next, the *IMU assesses traffic conditions and routes* by analyzing current traffic, vehicle positions, and routes. Following this, a *Conflict Check* is performed to determine if there are any conflicting routes. If no conflict is detected, the system proceeds to *allow the vehicle to proceed* by communicating permission to the vehicle, enabling it to continue through the intersection. After the vehicle crosses the intersection, the IMU logs the vehicle's exit and updates traffic conditions. If conflicting routes are detected, the system communicates to the vehicle to halt, and the vehicle must wait until it can proceed safely. This ensures orderly and efficient management of traffic at intersections.

III. UPPAAL IMPLEMENTATION

In this section, we utilized Uppaal to model our proposed solution, aiming to verify the system's intended behavior and validate it through verification queries. We employed five automata templates in our model. Four of these templates represent cars approaching the intersection from different directions (Car1, Car2, Car3, and Car4), while the fifth automaton is dedicated to the central intersection unit. All cars simultaneously request their desired destinations (Left, Right, or Straight) from their respective entries. The central unit, acting as the server, decides which cars are placed on the waiting list and which are allowed to proceed, implementing a First-In-First-Out (FIFO) queuing system. The automata templates use synchronization signals to manage these interactions. When a car enters the intersection zone, it sends an "Arrivingfrom(N, E,S,W)!" signal to inform the server of its entry. It then requests crossing permission via a "request!" signal. The server responds by sending either a crossing approval signal ("approved!") or a rejection signal ("wait!") after evaluating potential collision scenarios. Based on the received signal, the car either proceeds through the intersection and sends a "movingout!" signal upon exit or remains stopped until it receives permission to cross. This model ensures orderly and safe management of vehicle movements at the intersection.

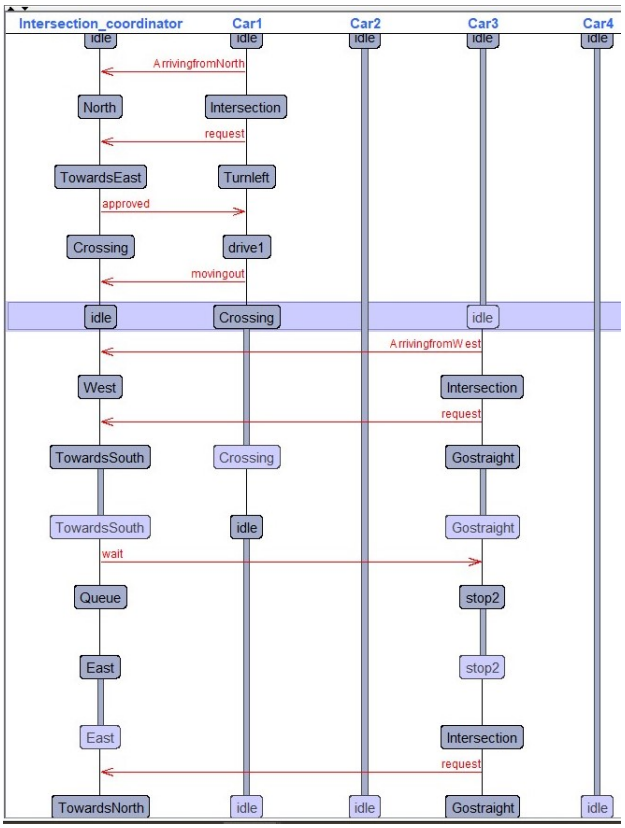


Fig. 6. Sequence Diagram

In Figure 6, the simulation of the system can be effectively demonstrated using a sequence diagram in Uppaal. This diagram illustrates the interactions and behavior of autonomous vehicles as they navigate an intersection. The sequence diagram captures the series of events, beginning with the arrival of vehicles at the intersection, the communication of their intended routes, and the subsequent decisions made by the central intersection unit. It visually represents how vehicles send signals to request entry, how the server processes these requests, and how it manages the flow of traffic by approving or denying crossing permissions.

Additionally, we conducted a deadlock check within the Uppaal model to ensure that our system operates without any potential deadlock scenarios. Deadlocks are critical issues in

concurrent systems, where two or more processes are unable to proceed because each is waiting for the other to release resources. In the context of our autonomous intersection controller, a deadlock could occur if vehicles are indefinitely waiting for permission to cross, causing a halt in the flow of traffic. In the following Figure.7 we verified a deadlock-free system means our system guaranteed that all the different parts will be able to make progress and no lock will be realized.

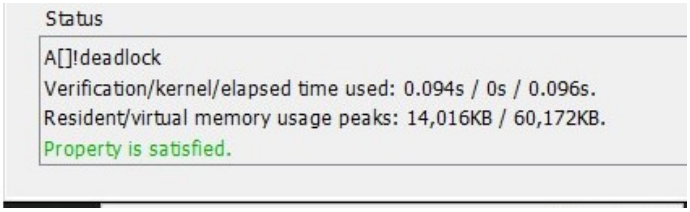


Fig. 7. Deadlock Verification

IV. PROPOSED SOLUTION OVERVIEW

The project's objective is to manage the traffic of autonomous vehicles (AVs) at an intersection using advanced algorithms, real-time data, and Vehicle-to-Infrastructure (V2I) communication. Vehicles can approach the intersection from four directions (North, East, South, West) and may go straight, turn left, or turn right. Each direction has its own queue to maintain the order of vehicles. The traffic management system operates by having each vehicle communicate its ID, origin, and destination to a central Intersection Management Unit (IMU) upon approaching the intersection. The IMU logs these details and evaluates current traffic conditions, assessing whether there are any conflicting routes. If no conflicts are detected, the IMU grants permission for the vehicle to proceed, sending a signal to the vehicle to continue through the intersection. The vehicle then proceeds, and the IMU updates the traffic conditions and logs the vehicle's exit from the intersection. In the event of conflicting routes, the IMU instructs the vehicle to wait until it is safe to proceed. This ensures a smooth and efficient flow of traffic, minimizing delays and preventing accidents. The proposed solution leverages the reliability of V2I communication over Vehicle-to-Vehicle (V2V) communication to handle the high data transmission volumes with greater noise resistance and reliability. The system is validated through simulations using UPPAAL, ensuring that it operates without deadlocks and that vehicles can make progress through the intersection without indefinite waiting periods. The integration of this autonomous intersection controller aims to enhance traffic flow, reduce delays, and support the broader adoption of autonomous vehicles in urban environments.

V. IMPLEMENTATION ON FREERTOS

FreeRTOS is a prevalent real-time operating system designed for embedded devices. It provides services such as preemptive multitasking, inter-task communication and synchronization, and memory management. The design of FreeRTOS is minimalist, making it suitable for microcontrollers. Implementing FreeRTOS in managing traffic intersections, particularly for autonomous vehicles, offers numerous advantages that enhance the overall efficiency and reliability of the system. One of the primary benefits is deterministic task scheduling, which allows for the prioritization of critical tasks such as emergency vehicle detection and collision avoidance. FreeRTOS ensures that these high-priority tasks are executed promptly and predictably, maintaining the timely flow of traffic.

This intersection control system uses the Arduino Mega due to its better hardware compared to other Arduino boards. The Mega is powered by the ATmega2560 powerful microcontroller with more memory and input/output pins than the standard Arduino Uno, particularly useful for large projects like ours involving many tasks and peripherals at the same time. The Arduino Mega offers 54 digital I/O pins and 16 analog inputs, hence giving a high degree of connectivity and flexibility. Moreover, its higher SRAM and flash memory capacity ensure the FreeRTOS, along with the associated tasks, run efficiently without running into memory constraints. These very characteristics make the Arduino Mega an excellent board for the development and test of a complex real-time system since it provides enough resources and performance to support the project's demands. Intersection Control System: FreeRTOS on Arduino Mega emulates an Autonomous traffic management example where cars are arriving from four directions: North, South, East, and West, and are controlled with regards to their closeness to the intersection. This project uses the Mega board of Arduino, owing to its enhanced hardware. This board uses the powerful microcontroller ATmega2560, which provides it with more memory and input/output pins than most of the other boards available from Arduino, thus making it appropriate for running several tasks and peripherals simultaneously, hence ensuring efficient real-time operations. An Arduino Mega is used, which has 54 digital I/O pins and 16 analog inputs with increased SRAM and flash memory in order to handle the additional overhead of FreeRTOS and associated tasks without memory concerns. The major components involved are: car structure, FreeRTOS tasks, queue, and mutex. Every car is treated as a structure containing a unique ID, arrival direction, destination direction, and proximity to the intersection. There exist two major FreeRTOS tasks used in this system:

Car Arrival Task: This task models cars arriving from some direction; there will be one such task per direction. All of these tasks generate cars continuously, in turn, and randomly assign a destination and proximity value to the car before putting it into a common queue. The tasks can also model variable inter-arrival times for additional realism in modeling real-life traffic.

Task Intersection: This task controls the intersection. It periodically polls the queue for the arrival of cars. The latter allows passage to the car closest to the intersection, but not more than one in parallel, due to mutual exclusion via a mutex. It also resolves conflicts, sets priorities by proximity, and prints diagnostic messages to the serial monitor.

The setup function configures the serial communication, and then it initializes the car queue and mutex. After that, it creates the tasks for managing car arrivals and the intersection. The hook functions handle stack overflow and failures in memory allocation, producing messages in the serial monitor.

```
Car from WEST with ID 4 wants to go to NORTH with proximity 3
Car from EAST with ID 5 wants to go to WEST with proximity 8
Car with ID 2 has passed through the intersection.
Car from SOUTH with ID 6 wants to go to NORTH with proximity 3
Priority given to car from WEST with ID 4 to go to NORTH due to higher proximity.
Car from NORTH with ID 7 wants to go to SOUTH with proximity 10
Car from WEST with ID 4 is passing through to NORTH.
Car from SOUTH with ID 8 wants to go to NORTH with proximity 4
Car from NORTH with ID 9 wants to go to EAST with proximity 7
Car from WEST with ID 10 wants to go to SOUTH with proximity 7
Car from EAST with ID 11 wants to go to WEST with proximity 1
Car with ID 4 has passed through the intersection.
Priority given to car from EAST with ID 11 to go to WEST due to higher proximity.
```

Fig. 8. FreeRTOS Implementation

The Figure.8 illustrate the output of an intersection control system showing the cars arrival and are being processed, considering their IDs, directions, and proximity to the intersection. First of all, there will be a car from the WEST with ID 4, having a proximity of 3 and aiming to head NORTH. After that, another car from the EAST, with an ID of 5 and having a proximity of 8, will intend to head WEST. It logs the passage of a car whose ID is 2 through the intersection. A car from the SOUTH, with ID 6, proximity 3, coming to go to the NORTH, arrives. In line with the higher proximity, priority is given to the car from the WEST with ID 4 to pass through to the NORTH. Subsequently, the following vehicles approach vehicle from the NORTH with ID 7 proximity 10 to the SOUTH, one from the SOUTH with ID 8 proximity 4 to the NORTH, one from the NORTH with ID 9 proximity 7 to the EAST, one from the WEST with ID 10 proximity 7 to the SOUTH, and finally one from the EAST with ID 11 proximity 1 to the WEST. As vehicle 4 crosses the intersection, the system addresses the vehicle coming from the EAST with ID 11, since it is much closer, and moves it forward. This example shows how the system controls the intersection by controlling the cars' movement on the basis of proximity to ensure orderly and on-time passage through the intersection.

VI. SOFTWARE/HARDWARE CO-DESIGN IMPLEMENTATION ON VHDL

The VHDL implementation aimed to integrate the functionality of a decision-making mechanism within a digital system. The primary objective of the VHDL component is to facilitate the management of vehicle movement at an intersection. This is achieved by evaluating various car attributes and determining whether a car is allowed to proceed based on its priority level.

The **entity declaration** defines the inputs and outputs for the Car entity. The clk and reset are input signals for clock and reset. The car-id, from, destination, proximity, and priority are all input signals related to the car's attributes. The allow is the output signal indicating whether the car is allowed to proceed.

The **architecture behavioral** snippet contains the logic for controlling the allow signal based on the priority. If reset is active ('1'), allow is set to '0'. On the rising edge of the clock, if the priority is greater than 5, allow is set to '1'; otherwise, it remains '0'.

```
68          -- Simulate car arrival
69          car_id <= 1;
70          from <= 0;  -- From NORTH
71          Destination <= 1;    -- To SOUTH
72          proximity <= 5;
73          wait for 20 ns;
74
75          car_id <= 2;
76          from <= 1;  -- From SOUTH
77          Destination <= 2;    -- To EAST
78          proximity <= 3;
79          wait for 20 ns;
80
81          car_id <= 3;
82          from <= 2;  -- From EAST
83          Destination <= 3;    -- To WEST
84          proximity <= 1;
85          wait for 20 ns;
86
87          car_id <= 4;
88          from <= 3;  -- From WEST
89          Destination <= 0;    -- To NORTH
90          proximity <= 7;
91          wait for 20 ns;
```

Fig. 9. Testbench VHDL

The **test bench code** in Figure.9 describes the simulation of each car arriving at the intersection from different directions. When each car arrives from a specific origin and heads to a specific destination, with a predefined proximity value, the simulation involves setting the car-id, from, destination, and proximity signals for each car and then waiting for 20 ns before moving to the next car. This approach ensures that the system accurately processes cars arriving from different directions to different destinations and proximity values. The wait periods simulate the time intervals between the arrivals of successive cars, allowing for a thorough evaluation of the car system's response to dynamic car traffic, thereby reducing the possibility of accidents.

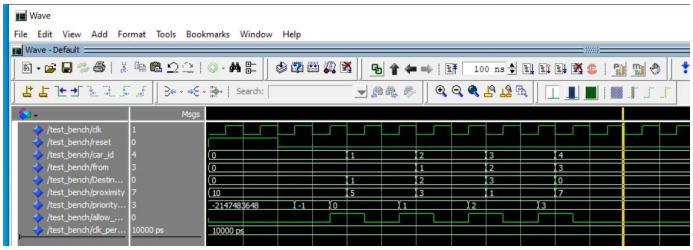


Fig. 10. Waveform VHDL

The **output waveform** in Figure.10 illustrates the simulation results of the Car entity in VHDL and the test bench. As shown in the waveform, the allow signal is correctly determined by the priority value of each car. The reset mechanism functions as expected, setting allow to '0' initially. The clk signal drives the synchronous behavior, ensuring that changes occur at the correct times. This confirms that the logic for controlling the allow signal based on the priority is working as intended,

with the reset mechanism and clock signal facilitating proper synchronization.

VII. CONCLUSION

In summary, our project documentation offers a solution for an autonomous traffic intersection controller. We tackled the issues of traffic congestion, delays, and accidents at signalized intersections by proposing an intelligent intersection management system. This system leverages Vehicle-to-Infrastructure (V2I) communication to enable data exchange between self-driving vehicles and the intersection controller.

We provided a comprehensive explanation of the system's functionality using UML and SysML diagrams, including use case diagrams, activity diagrams, state machine diagrams, and requirement diagrams. These diagrams depict various aspects of the system such as traffic coordination, communication flow, decision-making processes, and system requirements.

Additionally, we implemented the system using FreeRTOS, a real-time operating system, and showcased its benefits in task management, preemptive scheduling, inter-task communication, synchronization, and memory management. We also presented the VHDL implementation for hardware-software co-design, integrating the decision-checking mechanism.

Overall, our project aims to enhance traffic management, improve safety, and optimize the movement of autonomous vehicles at intersections. By effectively coordinating and controlling the flow of vehicles, we expect a reduction in congestion, accidents, and fuel consumption, leading to a more efficient and sustainable transportation system.

REFERENCES

[1] U.S. Department of Transportation, "National Agenda for Intersection Safety", Federal Highway Administration, Washington, 2005.

[2] A. Parker and G. Nitschke, "How to best Automate Intersection Management," *2017 IEEE Congress on Evolutionary Computation (CEC)*, Donostia, Spain, 2017, pp. 1247-1254.

[3] M. Rouse, "intelligent transportation system", June 2023. [Online]. Available: https://whatis.techtarget.com/definition/intelligenttransportation-system.

[4] K. K. Daniel J. Fagnant, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations", *Transportation Research Part A*, vol. 77, pp. 167-181, 2015.

[5] J. C. S. and D. C. Chin, "Traffic-responsive signal timing for systemwide traffic control", *Transportation Research Part C: Emerging*, vol. 5, pp. 153–163, 1997.

[6] N. J. Goodall, "Machine learning in autonomous vehicles: Techniques and applications," *Transportation Research Part C: Emerging Technologies*, vol. 57, pp. 98-113, 2015.

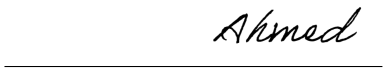
[7] D. B. Fambro, K. Fitzpatrick, and R. J. Koppa, "Determination of stopping sight distances," *Transportation Research Record*, vol. 157, pp. 77-89, 1997.

[8] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transportation Research Part C: Emerging Technologies*, vol. 10, pp. 303-321, 2013.

[9] Y. Li and M. McDonald, "Driving style and traffic safety: a simulator study," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 24, pp. 219-231, 2014.

VIII. DECLARATION OF ORIGINALITY

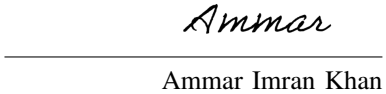
I Ahmed Hemly, declare that this paper is my own work, using only cited sources and aids. All direct quotes are marked, and all other references are fully detailed. This paper has not been submitted, published, or used for any other examination or course. I consent to plagiarism checks.



Ahmed Hemly
Lippstadt, 01.07.2024

IX. DECLARATION OF ORIGINALITY

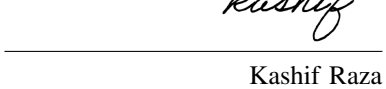
I Ammar Imran Khan, declare that this paper is my own work, using only cited sources and aids. All direct quotes are marked, and all other references are fully detailed. This paper has not been submitted, published, or used for any other examination or course. I consent to plagiarism checks.



Ammar Imran Khan
Lippstadt, 01.07.2024

X. DECLARATION OF ORIGINALITY

I Kashif Raza, declare that this paper is my own work, using only cited sources and aids. All direct quotes are marked, and all other references are fully detailed. This paper has not been submitted, published, or used for any other examination or course. I consent to plagiarism checks.



Kashif Raza
Lippstadt, 01.07.2024