Smart Cross-Traffic Management

Anguiga Hermann
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
hermann.anguiga@stud.hshl.de

Brian Kelein Ngwoh Visas
Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
brian-kelein-ngwoh.visas@stud.hshl.de

Enkeledi Mema
Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
enkeledi.mema@stud.hshl.de

Luis Cabezas Suarez
Electronic Engineering
Hochschule Hamm Lippstadt
Lippstadt, Germany
luis-david.cabezas-suarez@stud.hshl.de

***Abstract*—There is controversy in the Cross-traffic Management about when to recommend the practice of uncomplicated crossing. The aim of this paper is to assess the level of communication evidence on Autonomous Vehicles in an intersection to determine whether existing recommendations are appropriate. To carry out the research, a review of certain articles and research papers has been carried out consulting the behaviours of autonomous Vehicles as well as the functionality of a crossing intersection without any restriction, combining both ideas. No restrictions have been made on the type of study. Where necessary, this all research, has been revised, to finally consider all the information that included recommendations on creating the best solution for this Crossing Intersection Most of the recommendations made for this research, have been through finalizing the implementation of a V2I and V2V communication and a certain performance of a FIFO Queue while the vehicle complete its entering in the intersection. Studies on Queue application and this communication (V2I,V2V) have been able to determine that the recommendations made by articles and other resources concluded implementing a HW/SW code design has a solid management basis. In this case of Control side unit, communication with the vehicles, environment and Queue execution clarify which vehicles has the priority and therefore which it should be performing the crossing first avoiding collision with others in a most suitable way.**

***Index Terms*—Autonomous vehicle,traffic, crossing,**

I. Introduction

The fourth industrial revolution has enhanced drastic changes in the vehicular technology sector. Technological advancement in areas such as wireless communication, sensors, control strategies, and data management are the catalyzers of this transformation. Autonomous vehicle are equipped with capacities to communicate with other autonomous vehicles and with control units which are types of automated traffic agents that are able to regulate traffic at intersections where conventional traffic lights were removed. Studies that introduced autonomous intersection management have shown promising results while using this type of traffic management.
Traffic management at any intersection is a nightmare, especially at rush hours or during weekend and holidays. Many factors may be pointed out to be the cause. But the major culprits are human drivers. Autonomous vehicles can reduce some of these pains. Autonomous vehicle can accurately monitor and quickly react to its surroundings. They also follow tightly the traffic rules which minimizes the problems common with human driving such as large reaction time, lack of cooperation, and over-speed. Moreover, autonomous vehicles reduce energy and fuel consumption. Hence, several car manufactures have started to massively invest in the research and development in this field
Autonomous vehicles and infrastructures are equipped with sensors and others data collectors means to gather important data such as position, speed and destination of a vehicle. The gathered data are then exchange between vehicles through a vehicle-to-vehicle communication (V2V) or between vehicle and infrastructure through a vehicle-to-infrastructure communication.
The aim of this paper is to propose a collision-free and a very efficient scheme to manage the journey of autonomous vehicles through the intersection.
Rest of the paper is organized as follows. In “system overview” section, we discuss mainly about the type of queuing algorithm: the genetic algorithm, the type of scheduling algorithm: FIFO, and the types of communication systems: V2V and V2I, we used. In “implementation and simulations” section, we published results from this study and then the “conclusion” section concludes the paper

II. System overview

To begin, the system being designed here is an intelligent traffic system(STS). So we aim to develop an intelligent traffic system in a Hardware/Software Co-design system. This system has to work so that cars have to move between lanes and in a cross-traffic system without colliding with each other. We have three major actors, the car, the control unit, and the Roadside unit. We start with the cars; the first thing that happens when systems leave the idle state is that they establish a connection among themselves and the surrounding. So we begin with the communication between cars. First, Cars try to detect each other and other objects around with the help of Sensors. If Cars or objects are available, then the car stops, else it moves. When given the all-clear to move, since it is a traffic system, the cars have to Queue, and then an algorithm is put in place to prioritize cars.

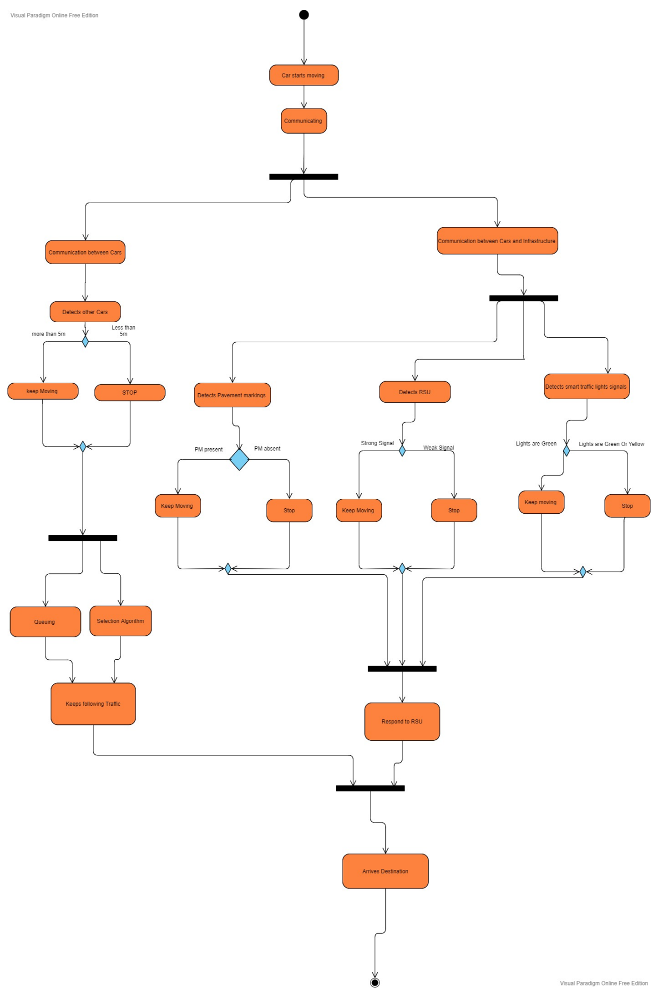


Fig. 1. Activity Diagram

Furthermore, when the car goes through the queue and is finally selected by an Algorithm, the next stage is for the car to communicate with other parts of its immediate environment. The next move is to communicate with its surrounding, which is the infrastructures. Here the infrastructures include the roadside unit, the pavement markings and to some extent, intelligent traffic lights. Furthermore, the communication between the car and these infrastructures could communicate through short-range frequencies with the help of the VANET(Vehicle Adhoc network explained later in the paper). If the message received is positive from these structures, then the car keeps moving; else, it stops. All these actions take place in traffic or at traffic crossroads, and after that, the car generally drives to its destination.

A. V2V communications

1) requirements analysis : We separated the system in subsystems. The separation was performed regarding the requirements. One of the requirements was the V2V communication. To understand better the requirements, we start analysing them. During the analyses phase we determine user expectations for final product. The effort is directed towards ensuring that each requirement full fill the final system needs. During the requirement analysation for the V2V communication, we generate the necessary components that we need to realise this subsystem. Analysation is the first step in the design process.

2) first approach : After the analysis phase was done for the V2V subsystem. It was time to define the subsystem and to develop a first raw approach. The develop the raw approach we decided initially to create some scenarios and user case. The primary goal of the use cases is to support us to understand the overall subsystem by getting in consideration small pieces.The first scenario that was taken in consideration was exchanging information for their directions, speed and trajectory so the driving maneuvers can be done smoother.The communication can be realized in two ways.In the firs way the cars communicate to each other's directly with their on board Wi-Fi communication .To preform this type of communication the car should be in a specific range . This range enable them to communicate. The use case in this scenario consists in one actor that are vehicles and different cases like breaking, accelerating, turning left or right, stopping and other cases related to the driving. The second way of communication is performed by a cloud control unit. The vehicles can transmit and receive information to the cloud. This communication is beneficial in case a log range and if the communication directly between vehicles is unstable or it is impossible. In this scenario the use case includes one actor and one external system. The actor is a vehicle, and the external system is the cloud-based control unit.

3) definition: The following step after the first approach is to define our subsystem. To provide a more thorough overview of the V2V subsystem, we decided to use different UML and SysML diagrams. The first diagram that is used is a block definition diagram. In the BDD are represented

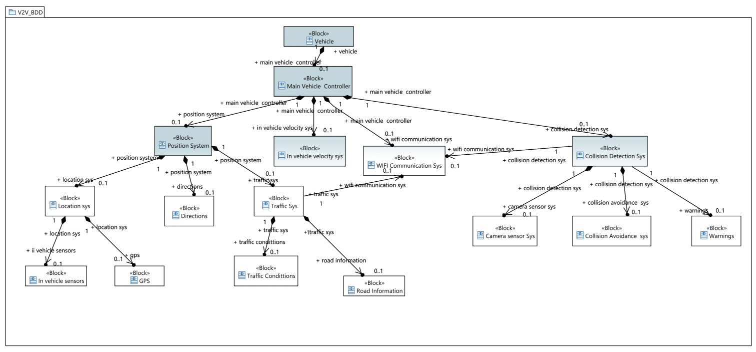

V2V_BDD
«Block» Vehicle
«Block» Main Vehicle Controller
«Block» Position System
«Block» In vehicle velocity sys
«Block» WiFi Communication Sys
«Block» Collision Detection Sys
«Block» Location sys
«Block» Directions
«Block» Traffic Sys
«Block» Camera sensor Sys
«Block» Collision Avoidance sys
«Block» Warnings
«Block» In vehicle sensors
«Block» GPS
«Block» Traffic Conditions
«Block» Road Information

Fig. 2. BDD

four parts which are position, velocity, communication, and collision. All these four components will be used to design and perform the V2V communication. To contain a more thorough overview regarding the activities that are running in the subsystems we designed an activity diagram. The activity diagram was based in the collision avoidance system. It

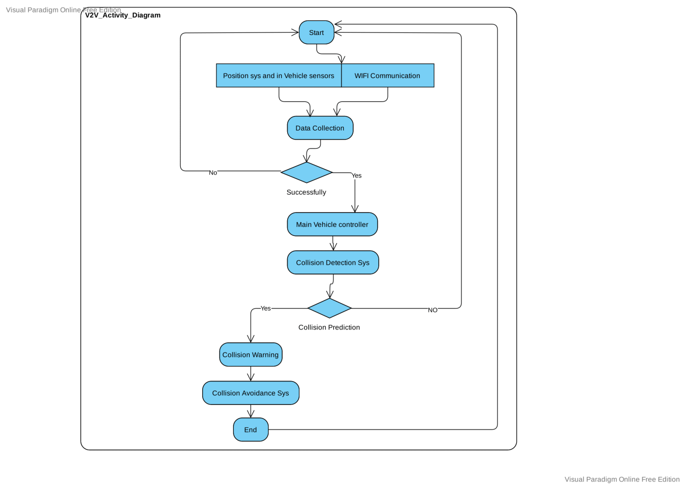


Fig. 3. Activity Diagram

indicates all activities from gathering information from sensors till in collision avoidance system. Another diagram was the state machine diagram. It represents the diverse states that a performed during a V2V communication. In the state machine diagram transmitting and receiving date are performed in a parallel. There are some main states that we switch. For example, we initialize the communication. Than we perform the communication and in the contiguous state we finish the communication.After that we leap to an idle state or maybe we disconnect. Next diagram is sequence diagram. Based in the

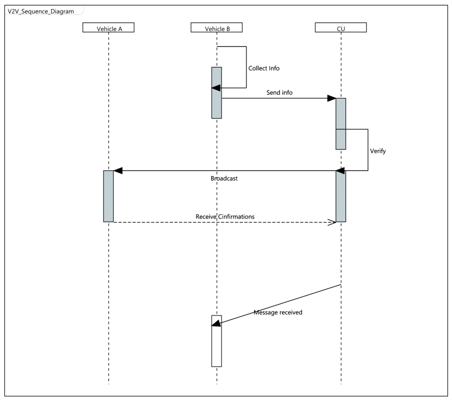
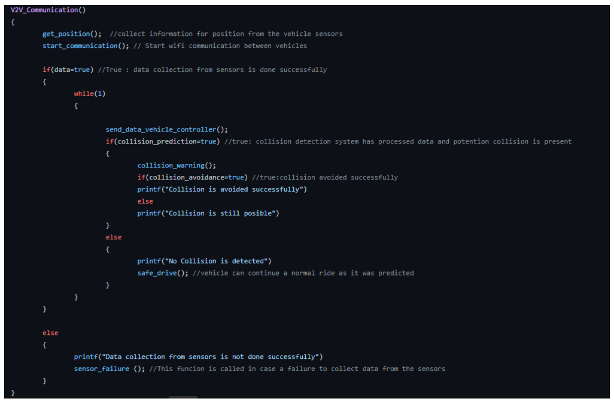


Fig. 4. Sequence Diagram

scenario when two vehicles must communicate between them using control unit, we created this sequence chart. We can see the vehicle B collects information first. On that occasion vehicle B forward this information the control unit(cloud). After the information is received, then the control unit verifies it. After the information is processed now control unit can broadcast the information to the vehicle A or multiple vehicles at the same time. After vehicle A receives the information that previously was asked, it sends a revived confirmation to the control unit.

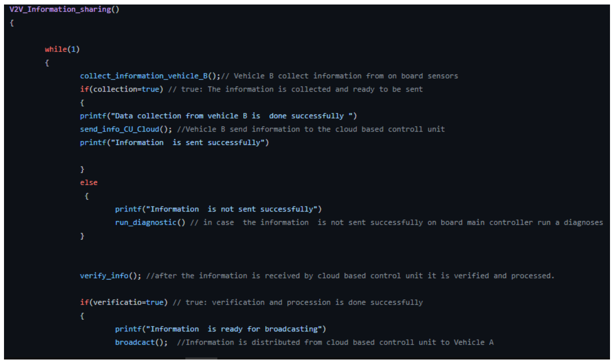
4) development: Development The next step is to generate the pseudo-codes and the algorithm. These pseudo-codes are used later in the implementations. The pseudo-code is generated by checking closely the UML an SysML diagrams. For this purpose, the activity diagram, State machine diagram, sequence diagram and class diagram are analyzed and taken into consideration.V2V communication algorithm (V2V-Communication) was generated by carefully checking the V2V activity diagram. The activities are translated to the functions.



```
V2V_Communication()
{
        get_position();  //collect information for position from the vehicle sensors
        start_communication(); // Start wifi communication between vehicles

        if(data=true) //True : data collection from sensors is done successfully
        {
                while(1)
                {

                        send_data_vehicle_controller();
                        if(collision_prediction=true) //true: collision detection system has processed data and potention collision is present
                        {
                                collision_warning();
                                if(collision_avoidance=true) //true:collision avoided successfully
                                printf("Collision is avoided successfully")
                                else
                                printf("Collision is still posible")
                        }
                        else
                        {
                                printf("No Collision is detected")
                                safe_drive(); //vehicle can continue a normal ride as it was predicted
                        }
                }
        }

        else
        {
                printf("Data collection from sensors is not done successfully")
                sensor_failure (); //This funcion is called in case a failure to collect data from the sensors
        }
}
```

Fig. 5. V2V communication pseudo-code

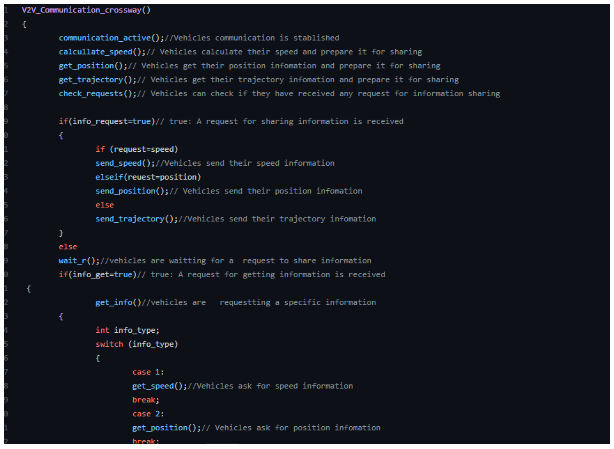
These functions we return specific values or messages. For the collision detection system we first get the position of the car by calling function(get-position) and we star communication by calling(start-communication) function and if there is any collision we can predict and avoid it. Based in our sequence diagram and states machine diagram we generated the V2V-information-sharing pseudo-code . As we described in the use



```
V2V_Information_sharing()
{

        while(1)
        {
                collect_information_vehicle_B();// Vehicle B collect information from on board sensors
                if(collection=true) // true: The information is collected and ready to be sent
                {
                printf("Data collection from vehicle B is  done successfully ")
                send_info_CU_Cloud(); //Vehicle B send information to the cloud based controll unit
                printf("Information  is sent successfully")

                }
                else
                 {
                        printf("Information  is not sent successfully")
                        run_diagnostic() // in case  the information  is not sent successfully on board main controller run a diagnoses
                }


                verify_info(); //after the information is received by cloud based control unit it is verified and processed.

                if(verificatio=true) // true: verification and procession is done successfully
                {
                        printf("Information  is ready for broadcasting")
                        broadcact();  //Information is distributed from cloud based controll unit to Vehicle A
```

Fig. 6. V2V communication pseudo-code

case and later in sequence diagram the vehicles communicate between them using the cloud control unit when they can receive and transmit specific information. As it is sown in the pseudo-code vehicle B collect information from on board sensors. In the next step vehicle B send information to the cloud-based control unit. in case the information is not sent successfully on-board main controller run a diagnosis. After the information is received by cloud-based control unit it is verified and processed. In the next step Information is distributed from cloud-based control unit to Vehicle A or to more vehicles. If the information is received successfully by vehicle A than vehicle A send back a confirmation that information is received successfully. The last algorithm that we generated in pseudo-code is V2V-Communication-cross-Section .



```
V2V_Communication_crossway()
{
        communication_active();//Vehicles communication is stablished
        calcullate_speed();// Vehicles calculate their speed and prepare it for sharing
        get_position();// Vehicles get their position infomation and prepare it for sharing
        get_trajectory();// Vehicles get their trajectory infomation and prepare it for sharing
        check_requests();// Vehicles can check if they have received any request for information sharing

        if(info_request=true)// true: A request for sharing information is received
        {
                if (request=speed)
                send_speed();//Vehicles send their speed information
                elseif(reuest=position)
                send_position();// Vehicles send their position infomation
                else
                send_trajectory();//Vehicles send their trajectory infomation
        }
        else
        wait_r();//vehicles are waitting for a  request to share information
        if(info_get=true)// true: A request for getting information is received
 {
                get_info()//vehicles are   requestting a specific information
        {
                int info_type;
                switch (info_type)
                {
                        case 1:
                        get_speed();//Vehicles ask for speed information
                        break;
                        case 2:
                        get_position();// Vehicles ask for position infomation
                        break;
```

Fig. 7. V2V communication pseudo-code

Our focus is to analyze and perform the communication in the cross-sections. The first part of algorithm requests some information from the vehicle. After this request a specific vehicle or a group of vehicles calculate their speed and prepare it for sharing. Also, vehicles get their position information and prepare it for sharing. In the same way vehicles get their trajectory information and prepare it for sharing. Vehicles can check if they have received any request for information sharing. If this is true, they can transmit the collected information.

B. Queuing

1) requirements analysis : In this idea. We need a process in where these vehicles can enter in a form of waiting time or zone to perform the entry into the intersection with no collision. Many ideas come in mind and the first one is the implementation of a Queue where the server or control unit usually selects the vehicle according to the priority. However, developing this idea, that the vehicle arrive must be placed in a way and wait their turn to be processed, the way to empty the queue is to take the package that has been in the algorithm. Nevertheless, is we explain this control unit and how it will take part in this queue is the heart of the traffic management system. As such, it must be able to incorporate functionalities related to the management of vehicles not only during the crossing of the intersection, but also before. monitors any vehicle entering the facilities. It tags each vehicle with a specific tracking number and maintains also a constant digital communication. Each vehicle approaching the critical queuing zone should reduce its speed to the authorized speed. The control unit then queues vehicles according to the genetic algorithm

2) Approach :

a) first approach: Queuing in an automated traffic crossing is of high importance and under no circumstance it has be taken lightly. It is full of pain that we regularly see how many times people lose their lives at a traffic intersection because of minor mistakes such as a driver who engaged himself into the crossing section while the light was already red or when he has waited for minutes and he lost his patience
To deal with this situation, in this paper, we have chosen a system that queues vehicles in a “waiting” zone. A vehicle will only cross if it receives instructions from the control unit
.

b) second approach: Another approach we decided to use was the Genetic Algorithm.In This Algorithm, as its name suggests, we would be using the genetic structure to solve computational problems in a very structured manner. This how Biological genetics work concerning the Algorithm. First, an individual contains Chromosomes. These Chromosomes contain Genes. Also, we have what is called the Fitness value or function. This Fitness value/function is based on the performance of the Chromosomes in everyone. Hence the fittest individuals are those individuals with the best or highest-performing Chromosomes. These individuals are selected and then undergo Mutation. Then, the Genes from these individuals are applied to the next generation to achieve better results going forward. Applying this system to schedule our Smart Traffic System, we do it in the following way. The Cars here are the Genes; these cars make up a car sequence in a particular lane. This Car sequence can relate directly to the Chromosomes. Furthermore, multiple Chromosomes are generated for better assessment. The various sequences have different scheduling sequences, arrival times, length of the queue, and many more. In addition, the sequence (Chromosome) with the best fitness value is then selected based on its performance. Furthermore, the Chromosome with the best fitness is selected, and then Mutation occurs so this sequence can be passed onto the next generation. The above process of selecting a fitness value, Mutation, and gene exchange will repeatedly occur until the Algorithm is terminated.

c) third approach: Also, we thought of another way of implementing the queuing: using the cloud to allocate time slots to cars, and then they use the FIFO queuing method. So first, the cars request a slot from the cloud, then the cloud checks for available time slots, and then allocate one for the car. After this, they move into the queue. When it is their time slot, they now have permission to move into the intersection and then further to the other side of the road.

3) definition: To execute the crossing with maximum efficiency, vehicles waiting in the queuing area have to be

schedule in a way that they will leave the area as fast as possible without causing any accident or incident. For obvious reason, we will use the First in First out (FIFO) algorithm. For queuing,we will use the genetic algorithm. Genetic algorithm performs well in environment where many vehicles from the same lane or different lanes are queued before being scheduled in the crossing area, at the same time. But in this project, we may start schedule one vehicle at the time. FIFO algorithm is then the one we chose.
The queuing process is entirely dependent on the control unit. The control unit use-case in the appendix C shows all the functionalities present within the control unit.
First of all, the control unit establishes a communication contact with the incoming vehicle. It then monitors the vehicle parameters such speed and priority to cross. If for example a car does not slow down enough when approaching the intersection, it will be tagged with a special tag number and specific actions against this ambiguous behavior will be taken.

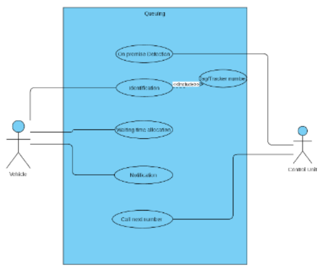


Fig. 8. Queuing use-case

C. V2I communications

1) requirements analysis : In the system, we need cars to communicate with the infrastructures. How does this happen ?. We need a communication protocol that can help link up the Car with the infrastructures, including road signs, lane markings, traffic signals and a lot more. The technology works wirelessly to communicate captured and permanently stored data with the environment of the vehicle. Various wireless technologies can be used, e.g. Wi-Fi, Bluetooth, ZigBee or dedicated short-range communication networks (DSRC networks) under IEEE 802.11p. Communication with satellites (e.g. for GPS) and reception of broadcast signals (e.g. TCM for traffic information) can also be considered part of V2I communication. Furthermore, various mobile radio systems can send and receive data via the mobile Internet while on the move. The vehicle itself must therefore have the appropriate technical equipment in order to benefit from the V2I. Modern vehicles often have an extensive infotainment system with integrated hardware. Some functions can also be used via smartphone.

2) first approach : In principle, we first decided to use the cloud and WIFI; as a form of communication. So it had to function so that the car sends information to the cloud, and then from the cloud, the infrastructure can send and collect

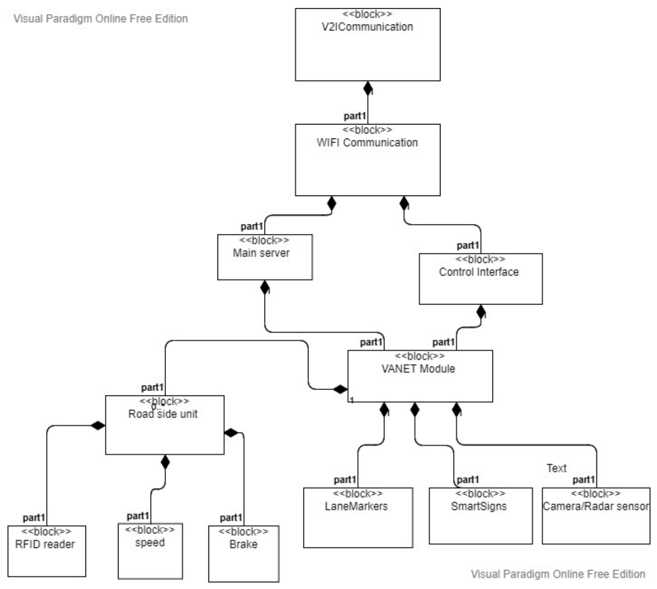


Fig. 9. Block Diagram V2I communication system

information from the cloud. However, we later discovered that this mode of communication would be problematic because of timing. It would be very slow to receive and give out information, causing the system not to be fully productive and very slow, which is not ideal for the kind of system we want to realize. Also, another first idea we had was communication with intelligent traffic lights. It was sending and receiving messages on when to stop and when to move. Nevertheless, later on, we discarded this information because we wanted to realize a fully autonomous system.

3) definition: Here we will discuss the general functioning of the V2I system. We begin from the idle state, where the car is doing nothing. After activation, the car now moves to the roads towards the traffic lanes. Before then, it has to check its surroundings for any objects, with the sensors' help, and possibly a camera. Furthermore, when the car confirms there are no objects around, it keeps moving. Then it communicates with the roadside unit to request traffic information. The roadside unit sends the car the status of traffic. Then the car keeps moving and then receives other signals from other Infrastructures. Then it gets into the queue, follows the protocol, gets chosen, communicates with the cross-section, and negotiates a bend after receiving a signal from a roadside unit. Then finally drives to its destination.All this could be seen in the form of a state machine diagram.

4) development: After all, is said and done, we finally decided to implement the system abstractly, in the form of Hardware/Software Co-design. In the Vehicle to Infrastructure communication, we came up with the following solution. At the top of our communication module, we have WIFI, which is like the network linked to all parties. Furthermore, we have a Main Control Unit and a Control Interface with a WIFI connection. Then present are all the Infrastructures: the Roadside unit (RFID reader), Lanemarkers, Smart signs, Camera/Radar

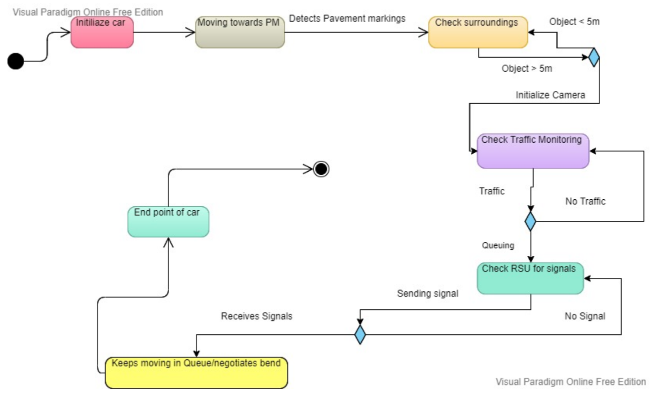


Fig. 10. State Machine diagram for V2I

sensors. All infrastructures first are connected to the WIFI, when in range. In addition, they communicate through dedicated short-range communication networks (DSRC networks) under IEEE 802.11p(Vehicle ad-hoc networks(VANETS)) with the car.

Nevertheless, we decided to design this both in Hardware and Software. First, we had to abstract the functions(communication) of the Roadside unit into an FPGA(Field Programmable Gate Array), having just one hardware to carry out some functionalities of the system. Then the communication between different parts of the vehicle to infrastructure communication would be implemented in software.

III. Implementation and Simulations

Implementing and simulating a system like this may be a very huge task and will require some time. We decided then to focus only on specific parts.

A. Hardware

After we developed the algorithm. We discussed what was better to be implemented in the software and what in the hardware. We decided to implement in the hardware the grid. The grid consists in different cells. These cells generate information if a vehicle is in that cell. The information is collected and process by a roadside control unit. In this case we decided to use the modelsim software for simulation.

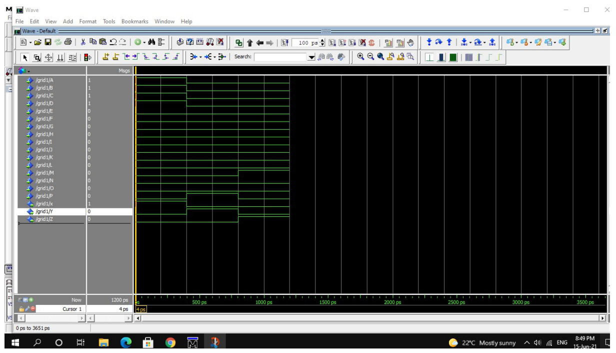
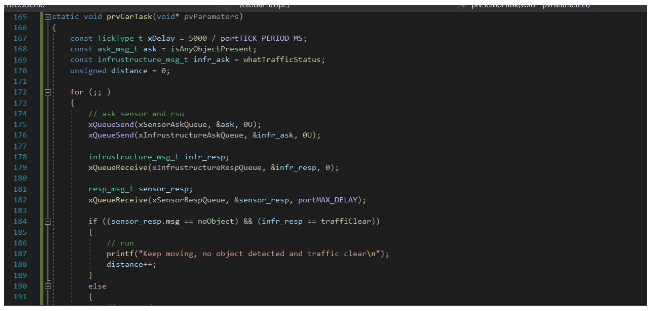


Fig. 11. VHDL Simulation

And we decided to use a hardware description language for the simulation. We decided to use VHDL. Our grid predicted the cases when there is a conflict. With the simulation we are able to show the whole cases and we can demonstrate that there are no conflicts and therefor no collection will occur.The simulation is done first by using FIFO and we suppose that the there is one car passing through the cross sections.

B. Software

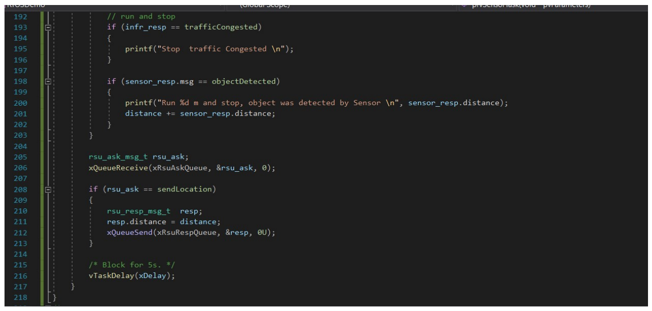
- V2V communication:
  To implement communication between vehicles (V2V communication) we used a specific real time operating system (RTOS) called freeRTOS and an intergrated development environment called visual studio. We used two processes to represent two vehicles. After successfully establishing a communication channel between them, we then sent messages from process A to process B. Process B received all messages from process A. (see appendix D)
- V2x communication:
  It is called V2x communication because we tried to implement some form of message passing between cars and the sensors and then between the Car and the control unit. We do all these things with the help of FreeRTOS. Here tasks are created, Car, Roadside unit, sensor and also for other infrastructures. These tasks request and receive information from each other. With FreeRTOS, we can give tasks some priority and also schedule these tasks. The FreeRTOS contains a scheduler which we call with a command, vtaskStartScheduler. Without this command, the FreeRTOS library will not activate the scheduler to move into functions and schedule processes, which is one of the main functions of this operating system. Also, there is the possibility to delay a particular task. The Car is first to communicate by requesting information about surrounding objects. Then the sensor reacts by checking and sending information back to the Car. In the presence of an object, do not move else; keep moving.



```
static void prvCarTask(void* pvParameters)
{
    const TickType_t xDelay = 5000 / portTICK_PERIOD_MS;
    const ask_msg_t ask = isAnyObjectPresent;
    const infrustructure_msg_t infr_ask = whatTrafficStatus;
    unsigned distance = 0;

    for (;; )
    {
        // ask sensor and rsu
        xQueueSend(xSensorAskQueue, &ask, 0U);
        xQueueSend(xInfrustructureAskQueue, &infr_ask, 0U);

        infrustructure_msg_t infr_resp;
        xQueueReceive(xInfrustructureRespQueue, &infr_resp, 0);

        resp_msg_t sensor_resp;
        xQueueReceive(xSensorRespQueue, &sensor_resp, portMAX_DELAY);

        if ((sensor_resp.msg == noObject) && (infr_resp == trafficClear))
        {
            // run
            printf("Keep moving, no object detected and traffic clear\n");
            distance++;
        }
        else
        {
```

Fig. 12. Requesting a message from a task

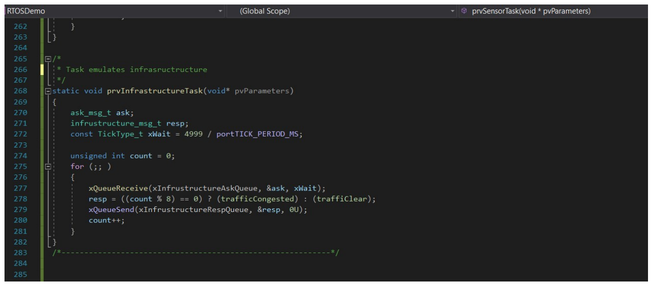
Furthermore, we have communication between the Infrastructure and the Car. The Car requests traffic information, and the Infrastructure checks for traffic situations and returns a message. In addition, we have the Roadside unit, which requests the cars exact location at a time, and



```
            // run and stop
            if (infr_resp == trafficCongested)
            {
                printf("Stop  traffic Congested \n");
            }

            if (sensor_resp.msg == objectDetected)
            {
                printf("Run %d m and stop, object was detected by Sensor \n", sensor_resp.distance);
                distance += sensor_resp.distance;
            }
        }

        rsu_ask_msg_t rsu_ask;
        xQueueReceive(xRsuAskQueue, &rsu_ask, 0);

        if (rsu_ask == sendLocation)
        {
            rsu_resp_msg_t  resp;
            resp.distance = distance;
            xQueueSend(xRsuRespQueue, &resp, 0U);
        }

        /* Block for 5s. */
        vTaskDelay(xDelay);
    }
}
```

Fig. 13. Requesting a message from a task 2

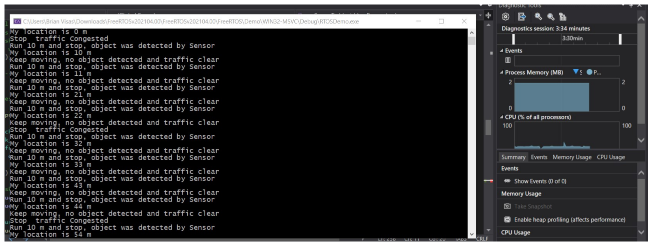
then the Car returns a message with its exact location, possibly with the help of a GPS. Furthermore, there is a possibility of including protocols with the FreeRTOS library and using these protocols to communicate between tasks. Also, the possibility of allocating priorities to each task during the scheduling process.



RTOSDemo (Global Scope) prvSensorTask(void * pvParameters)

```
    }
}

/*
 * Task emulates infrasructructure
 */
static void prvInfrastructureTask(void* pvParameters)
{
    ask_msg_t ask;
    infrustructure_msg_t resp;
    const TickType_t xWait = 4999 / portTICK_PERIOD_MS;

    unsigned int count = 0;
    for (;; )
    {
        xQueueReceive(xInfrustructureAskQueue, &ask, xWait);
        resp = ((count % 8) == 0) ? (trafficCongested) : (trafficlear);
        xQueueSend(xInfrustructureRespQueue, &resp, 0U);
        count++;
    }
}
/*-----------------------------------------------------------*/
```

Fig. 14. Receiving a message from a task



C:\Users\Brian Visas\Downloads\FreeRTOSv202104.00\FreeRTOSv202104.00\FreeRTOS\Demo\WIN32-MSVC\Debug\RTOSDemo.exe

```
My location is 0 m
Stop  traffic Congested
Run 10 m and stop, object was detected by Sensor
My location is 10 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 11 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 21 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 22 m
Keep moving, no object detected and traffic clear
Stop  traffic Congested
Run 10 m and stop, object was detected by Sensor
My location is 32 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 33 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 43 m
Keep moving, no object detected and traffic clear
Run 10 m and stop, object was detected by Sensor
My location is 44 m
Keep moving, no object detected and traffic clear
Stop  traffic Congested
Run 10 m and stop, object was detected by Sensor
My location is 54 m
```

Diagnostics session: 3:34 minutes
Events
Process Memory (MB)
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (0 of 0)
Memory Usage
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage

Fig. 15. Simulation result and display of RTOS

IV. Conclusion

In this paper is showed the implementation of an efficient cross-traffic management. The goal of this project was to provide the most ideal scenario to reduce the traffic congestion in an intersection. To achieve this goal was decided to separate the overall system in three sub-systems. The first subsystem is the V2V communication. The primary goal of this subsystem is to optimise the movement of the vehicles by sharing information regarding their speed, position, turning information and all other information regarding the driving process. The V2V communication is developed in two ways. The first way is directly through on board communication system. The second way is through the cloud where all vehicles can transmit and receive information for the driving maneuvers. We are more interested to maintain the V2V communication in the cross section. It is a effective tool to improve the cross-section stability by prevent the collision. The second is V2I communication. This communication plays an important role in our project.V2I communication is the key point for implementing a smart and autonomous system. In the cross section we have a control unit that is responsible for coordinating all the activities. The main function of the V2I is collect information from the sensors in the cross section, processes this information in the roadside control unit and distributes the information to the vehicles. The main functionalities of the control side unit were implemented in the Hardware. In this way we are faster. The third subsystem is queuing. Queuing is the best approach to reduce the traffic congestion in an intersection. The queuing process is realised with the help of an algorithm. The leading principle of the algorithm is to reduce as much as possible the waiting time in cross-section for each vehicle also to provide some priorities in case of a need. The algorithm is design to optimise the waiting time and as well to avoid any starvation scenarios. For each subsystem that we implemented some simulations preformed. Simulations for the traffic intersection are performed using C-code (core logic) , freeRTOS and for the hardware implementation,ModelSimm is used.

References

[1] https://www.freertos.org/index.html. visited on the 14/06/2021
[2] https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.html.visited on the 14/06/2021