Truck Platooning System

CHUN-KUAN, CHIH
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
chun-kuan.chih003@stud.fh-dortmund.de

RAMYA MANJUNATH
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
ramya.manjunath003@stud.fh-dortmund.de

SAUL GARCIA RODRIGUEZ
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
saul.garciarodriguez004@stud.fh-dortmund.de

Abstract — Platooning for trucks on highways has been increasingly investigated in lots of research projects. Its technologies have made significant advances in the last decade. In simple words, a platoon comprises of an array of consecutive automated vehicles travelling as a string. Platooning allows for increased road utilization and reduced fuel consumption due to short inter-vehicular distances.

While many projects mainly focus on proposing new strategies to ensure road safety, traffic congestion and pollution, little work is done on collecting the basic ideas, use cases or system models of the requirements. This paper specifies details on various requirement diagrams and implements part of the behaviour based on real situations using SysML for certain use cases like platoon initialization, joining, splitting and parking. It also focuses on truck platooning with respect to the traits of embedded system engineering by considering real time scheduling using simso tool, simulation of use cases using autodesk tinkercad, verification & validation using Junit framework. Also, the results from these tools are also shown.

Keywords — Truck platooning, C-ACC, platoon engagement/disengagement, gap adaptation, platoon parking, pedestrian detection, obstacle detection

I. Introduction

Truck platooning has been aware in recent years as a leading area and both automated and cooperative vehicle technologies largely apply this technique into this category. Because trucking costs a lot and has dangerous situations, potential financial savings, and road safety have already demonstrated proof of concept of truck platooning, makes trucking an area that is attractive to apply this skill. Several trucks form an automated convoy with the lead truck sending out acceleration, braking and steering signals so the trucks can react accordingly. The communication between the trucks happens through V2V wireless technology. The following vehicles autonomously measure distance, speed and direction and adjust to the vehicles in front. With reduced safety distance and more constant driving speed, air drag decreases for the following trucks. Therefore, lowering fuel consumption and CO2 emission by up to 15 percent. Shorter and more predictable journey makes the driver's work less stressful and more convenient.

II. Model based design with SysML

Specification of the analysis model in SysML

a. Below are some of the requirements derived for truck platooning system

1. Platoon Behaviour: The platoon must ensure every truck behaviour is coordinated.
2. Distance Management: The trucks must be able to maintain a uniform distance between each other or create a gap.
3. Manoeuvre Coordination: The system must be able to coordinate the platoon manoeuvres.
4. Speed Management: The trucks must be able to maintain a uniform speed.
5. Positional Control: The trucks must be able to be controlled laterally and longitudinally.
6. Gap Adaptation: The gap between the trucks must be able to adapt.
7. Uniform Distance: The trucks must be able to maintain a uniform distance between each other.
8. Emergency Braking: The trucks must have an emergency braking system.
9. Lateral Control: The trucks must be able to be controlled laterally.
10. Longitudinal Control: The trucks must be able to be controlled longitudinally.
11. Platoon Formation: The truck must be able to form a platform within a specific range.
12. Platoon Disengagement: The trucks must be able to disengage the platoon.
13. Communication Management: The trucks must be able to maintain a constant communication.
14. Road Recognition: The system must be able to identify road properties.
15. Fuel monitoring: The fuel must be monitored for each truck in the platoon.

The below diagram illustrates the requirements discussed in the above section.

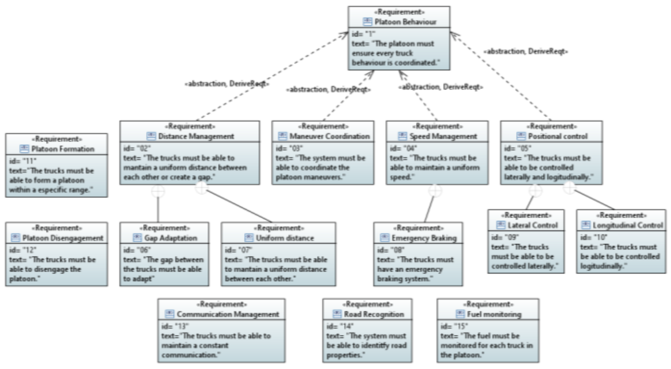

«Requirement» Platoon Behaviour
id= "1"
text= "The platoon must ensure every truck behaviour is coordinated."
«abstraction, DeriveReqt»
«abstraction, DeriveReqt»
«abstraction, DeriveReqt»
«abstraction, DeriveReqt»
«Requirement» Platoon Formation
id= "11"
text="The trucks must be able to form a platoon within a especific range."
«Requirement» Distance Management
id= "02"
text= "The trucks must be able to mantain a uniform distance between each other or create a gap."
«Requirement» Maneuver Coordination
id= "03"
text= "The system must be able to coordinate the platoon maneuvers."
«Requirement» Speed Management
id= "04"
text= "The trucks must be able to maintain a uniform speed."
«Requirement» Positional control
id= "05"
text= "The trucks must be able to be controlled laterally and logitudinally."
«Requirement» Platoon Disengagement
id= "12"
text= "The trucks must be able to disengage the platoon."
«Requirement» Gap Adaptation
id= "06"
text= "The gap between the trucks must be able to adapt"
«Requirement» Uniform distance
id= "07"
text= "The trucks must be able to mantain a uniform distance between each other."
«Requirement» Emergency Braking
id= "08"
text= "The trucks must have an emergency braking system."
«Requirement» Lateral Control
id= "09"
text= "The trucks must be able to be controlled laterally."
«Requirement» Longitudinal Control
id= "10"
text= "The trucks must be able to be controlled logitudinally."
«Requirement» Communication Management
id= "13"
text="The trucks must be able to maintain a constant communication."
«Requirement» Road Recognition
id= "14"
text= "The system must be able to identitfy road properties."
«Requirement» Fuel monitoring
id= "15"
text= "The fuel must be monitored for each truck in the platoon."

Fig. 1. Requirement diagram

b. Define the parametric constraints diagram of the system

The requirements from the previous section have been refined into three use cases which will be discussed in the later part of this paper. The below diagram shows the parametric constraints of the requirements that we have already seen above:

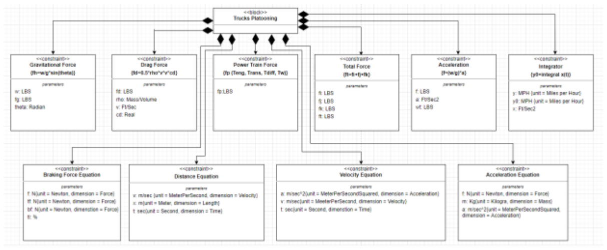


Fig. 2. Parametric constraints diagram

In the following sections, we have shown several sysML models for the major three use cases derived from the requirement diagram namely Formation of Platooning, Gap Adaptation Techniques, Emergency braking and Parking of platoon system.

Formation of Platooning

To enable the trucks to be part of a platoon there must be at the beginning at least 2 trucks available and with the desire to from a platoon. The diagrams presented in this section allow the reader to understand the behaviour of the system when the platoon formation is taking place.

A use case diagram corresponding to the forming platoon activity was created considering two different actors. Both actors must enable the connectivity option and there is the possibility that the connection of an actor is not discovered. To connect to a leader truck a truck must receive data about the trajectory of the leader truck and after that it must enter a specific PIN.

c. *Define the use case diagram of the system*

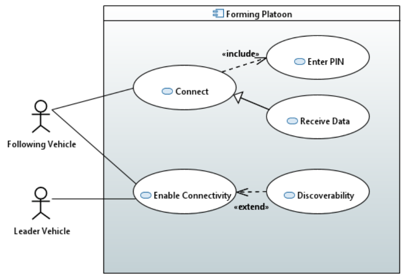


Fig. 3. Use case diagram for formation of platoon

d. *Define the activity diagram of the system*

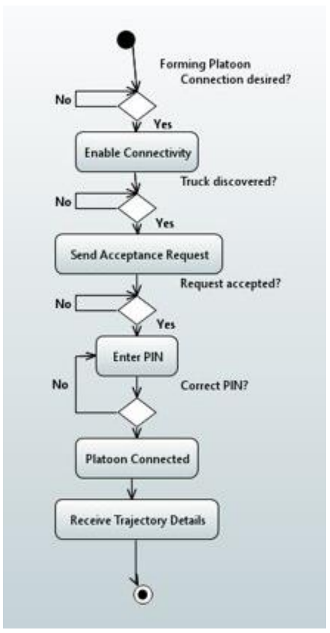
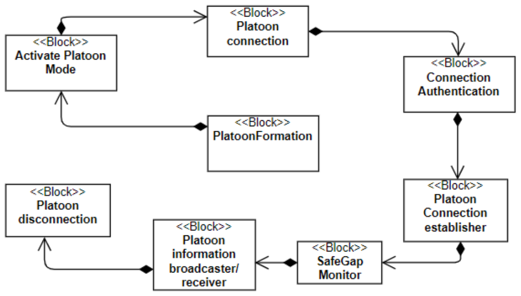
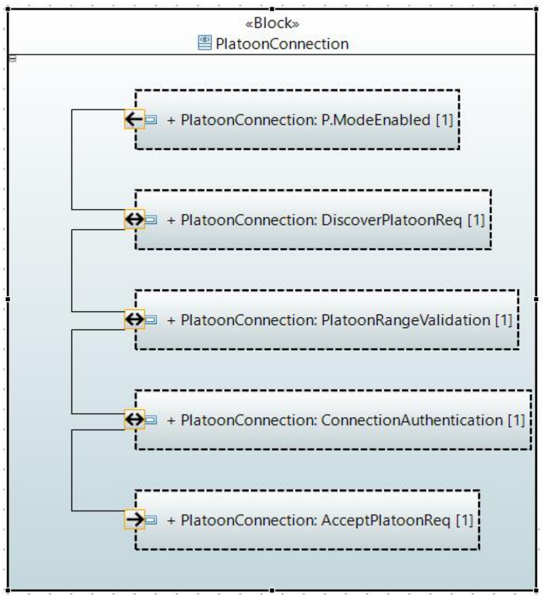


Fig. 4. Activity diagram for formation of platoon

e. *Define the block diagram of the system*


<<Block>> Platoon connection
<<Block>> Activate Platoon Mode
<<Block>> Connection Authentication
<<Block>> PlatoonFormation
<<Block>> Platoon disconnection
<<Block>> Platoon Connection establisher
<<Block>> Platoon information broadcaster/ receiver
<<Block>> SafeGap Monitor
Fig. 5. Block diagram for platoon formation.

f. *Define the internal block diagrams of the system*


«Block»
PlatoonConnection
+ PlatoonConnection: P.ModeEnabled [1]
+ PlatoonConnection: DiscoverPlatoonReq [1]
+ PlatoonConnection: PlatoonRangeValidation [1]
+ PlatoonConnection: ConnectionAuthentication [1]
+ PlatoonConnection: AcceptPlatoonReq [1]
Fig. 6. Internal block diagram for platoon connection

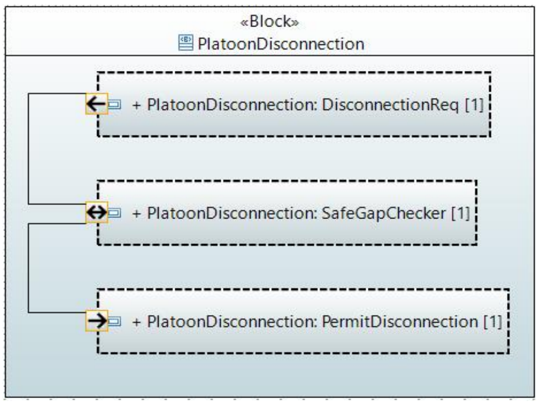

«Block»
PlatoonDisconnection
+ PlatoonDisconnection: DisconnectionReq [1]
+ PlatoonDisconnection: SafeGapChecker [1]
+ PlatoonDisconnection: PermitDisconnection [1]

Fig. 7. Internal block diagram for platoon disconnection

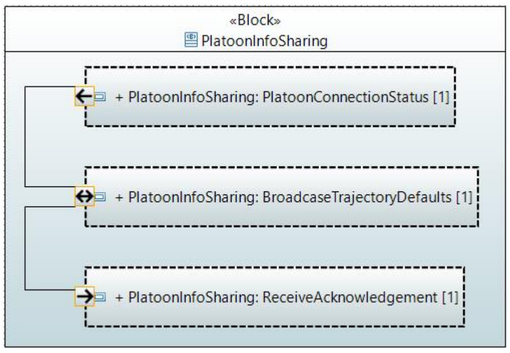

«Block»
PlatoonInfoSharing
+ PlatoonInfoSharing: PlatoonConnectionStatus [1]
+ PlatoonInfoSharing: BroadcastTrajectoryDefaults [1]
+ PlatoonInfoSharing: ReceiveAcknowledgement [1]

Fig. 8. Internal block diagram for platoon sharing

g. Define the sequence diagram of the system

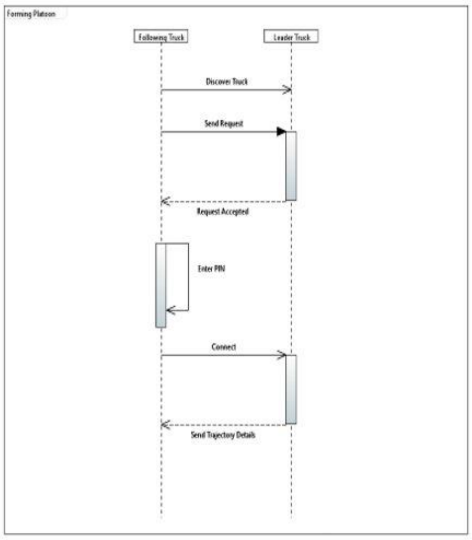


Fig. 9. Sequence diagram for formation of platoon

h. Define the state machine diagram of the system

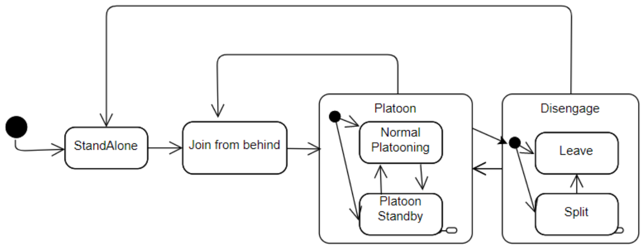


Fig. 10. State machine diagram for formation of platoon

Gap Adaptation using Object Detection in the Platoon Systems

Once the platoon connection is established it is substantial to react to the real time road scenarios. The platoons maintain distance between them based on the default trajectory settings which is generally 10m. We also receive the information about the constant speed and braking information from the leading vehicle through V2V communication. There is also some information which could be transmitted from the following vehicle to the leading vehicle, for instance if there is a vehicle on the adjacent lane that wants to change its lane and wants to overtake the platoon, there should be a mechanism to support this feature. In our paper we propose a solution which is feasible to achieve in the real time scenario. The platoon trucks will inherit some features from the radars and cameras installed within them. We consider at-least 4-6 cameras with wide-angle lenses integrated in the body panel of the trucks. Usually, these cameras are placed on the front grille, under the rear-view mirrors on either side, and on the tail. The cameras are located at spots that enable the surveillance of the entire perimeter of the vehicle. Therefore, when a vehicle is detected adjacent to the platoon wants to overtake to change the lane, the following truck senses the information like vehicle speed which is assumed to be greater than the platoon speed, indicator status and sends that information to the leading truck vehicle. On receiving this alert, the leading vehicle sends an open gap request to the corresponding truck. The non-platoon vehicle could now enter the platoon lane, however, now the following trucks have to adapt to the speed of the vehicle in the platoon lane. This can be achieved by using Automatic Cruise Control. The NP vehicle checks for the desirable conditions to change the lane and moves out of the platoon lane. Once the NP vehicle moves out, the following truck again sends information to the leading vehicle and upon the close gap request, it gets back to its original position by maintaining a distance of 10m between the platoons.

The behaviour can be easily understood from the below SysML models

a. Define the use casse diagram of the system

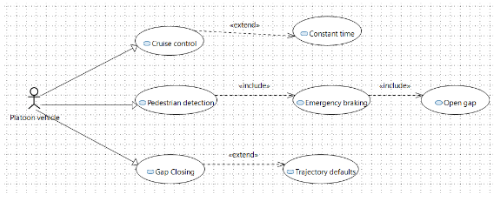


Fig. 11. Use case diagram for gap adaptation using object detection.

b. Define the use case diagram of the system

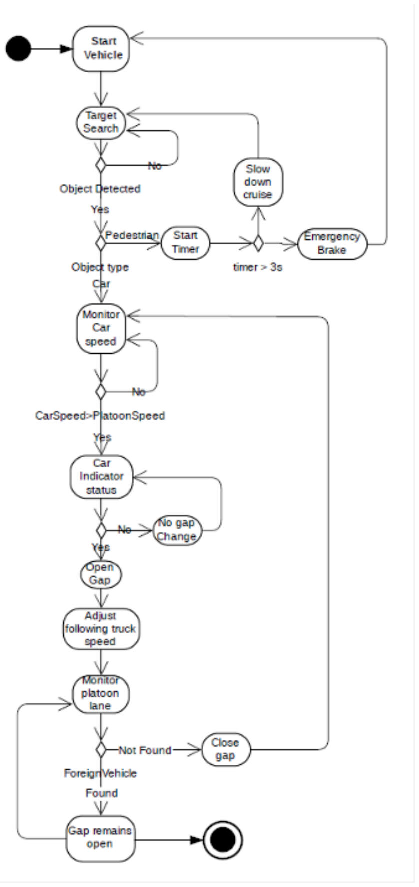


Fig. 12. Activity diagram for gap adaptation using object

c. Define the allocation diagram of the system

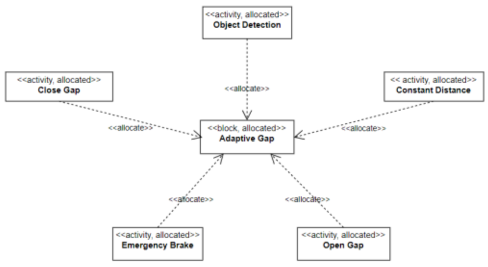


Fig. 13. Allocation for gap adaptation using object detection

Emergency Braking using Object Detection

As mentioned earlier, each truck is provided with at-least 4-6 cameras which includes front grille camera. This can be both narrow angle and wide-angle camera. Using narrow angle camera, the truck can detect objects from a far distance. If there are any objects like a pedestrian in the platoon lane, the leading truck senses this and sends information to the following trucks to either slow down or apply emergency brake based on certain conditions. In our scenario, if the pedestrian stays in the platoon lane for more than 3 seconds an emergency brake is applied, otherwise, the leading truck sends a message to slow down the speed. Once the platoon lane is clear from obstacles, the platoon picks up the original speed.

During the course of operation, it is also necessary to monitor the platoon lane. Since a series of trucks drive in a single lane, it is important to stay in the same lane throughout the journey. To accomplish this, we again use multifunctional camera sensors to monitor the lane attributes. This information is processed using the lane keeping assist algorithm and the steering wheel angle will be adjusted based on the sensor output.

d. Define the block diagram of the system

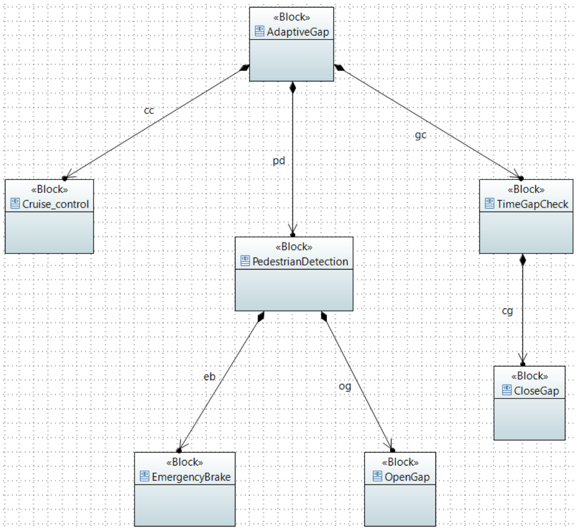


Fig. 14. Block diagram of pedestrian detection

e. Define the internal block diagram of the system

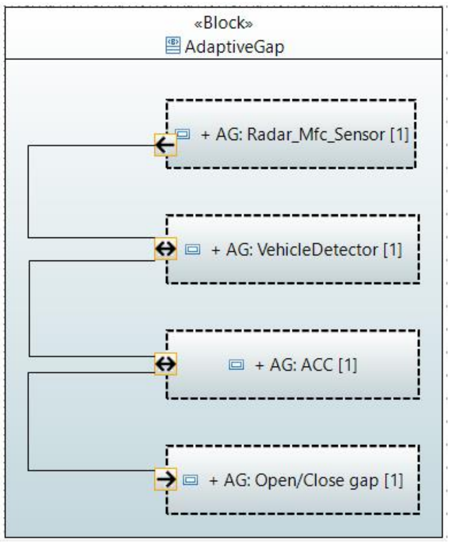
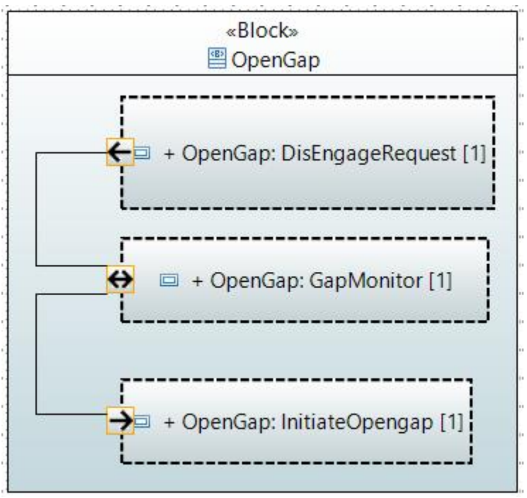
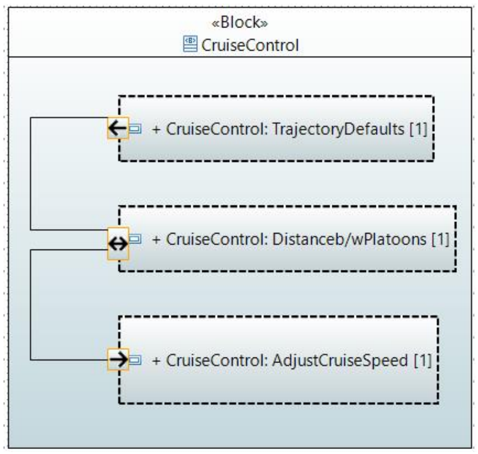


Fig. 15. Internal block diagram for adaptive gap


«Block»
OpenGap
+ OpenGap: DisEngageRequest [1]
+ OpenGap: GapMonitor [1]
+ OpenGap: InitiateOpengap [1]
Fig. 16. Internal block diagram for Open gap


«Block»
CruiseControl
+ CruiseControl: TrajectoryDefaults [1]
+ CruiseControl: Distanceb/wPlatoons [1]
+ CruiseControl: AdjustCruiseSpeed [1]
Fig. 17. Internal block diagram for Cruise control

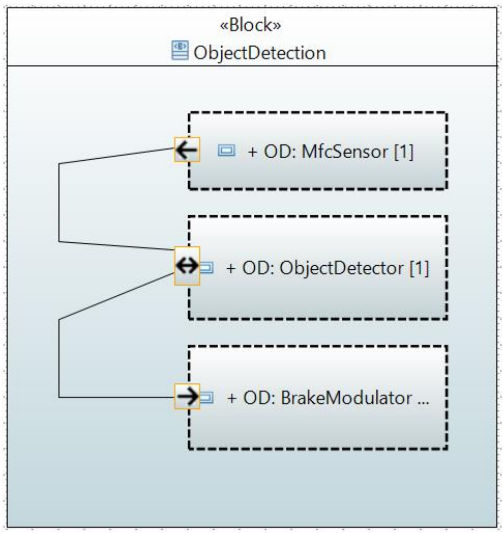


Fig. 18. Internal block diagram for object detection

f. Define the sequence diagram of the system

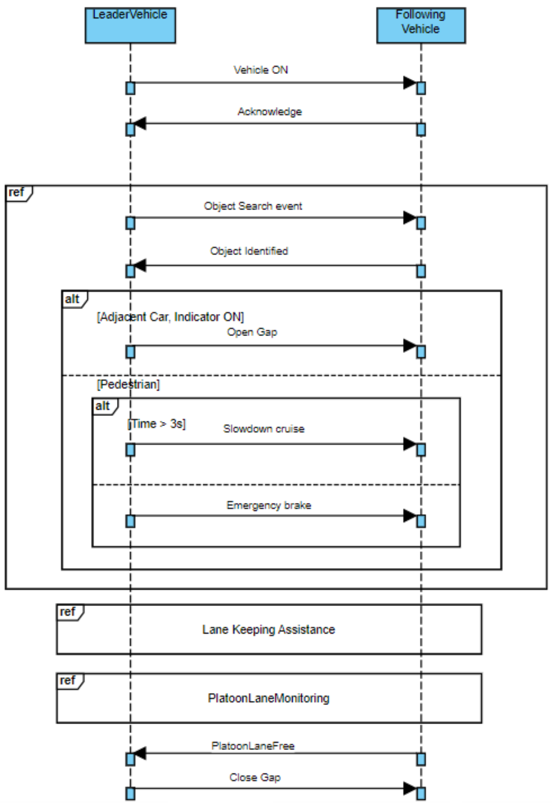


Fig. 19. Sequence diagram for gap adaptation

g. Define the state machine diagram of the system

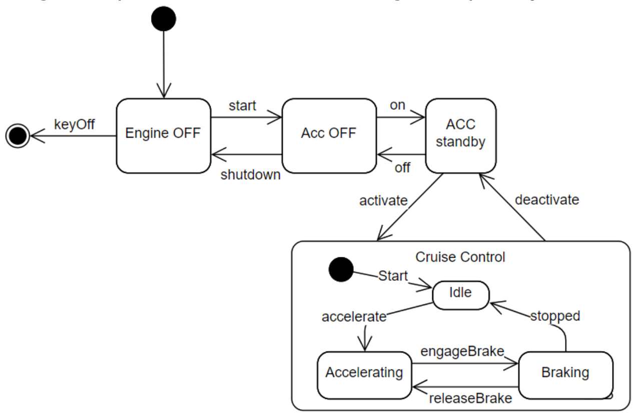


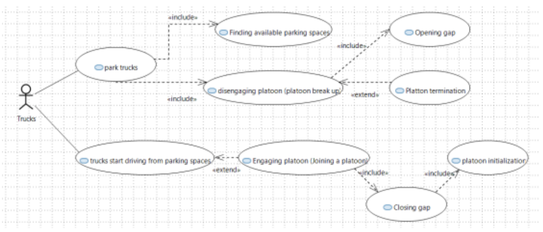
Fig. 20. State machine diagram depicting ACC

Parking system

Parking system is very important in trucks platooning. Therefore, it is crucial to handle the string of trucks during the need of parking. Firstly, we must ensure that there is enough space in the parking-lot to accommodate all the trucks of the platoon. To handle this, we propose that the trucks should be installed with parking condition receptors that monitors the parking status in the surrounding environment. When the parking space is found and the driver decides to park the truck, the leader truck sends platoon breakup request to other trucks in platoon along with the parking coordinates. After the request is accepted, the trucks open the gap and will be parked at the respective parking coordinates received from the leading platoon vehicle.

When the truck launches from the park spaces, it can choose to either join the platoon or drive on its own. If the truck chooses to join the previously connected platoon, it has to send re-join request to the platoon. After the request is accepted by the leading truck, it sends the trajectory messages to the following truck through constant communication channel (V2V). The following vehicle now adjusts its gap in the platoon lane and maintains the pre-set gap with the other trucks.

a. Define the use cases diagram of the system



Trucks
park trucks
Finding available parking spaces
Opening gap
disengaging platoon (platoon break up)
Platton termination
trucks start driving from parking spaces
Engaging platoon (Joining a platoon)
platoon initialization
Closing gap
«include»
«extend»

Fig. 21. Use cases for parking system

b. Define the activity diagram of the system

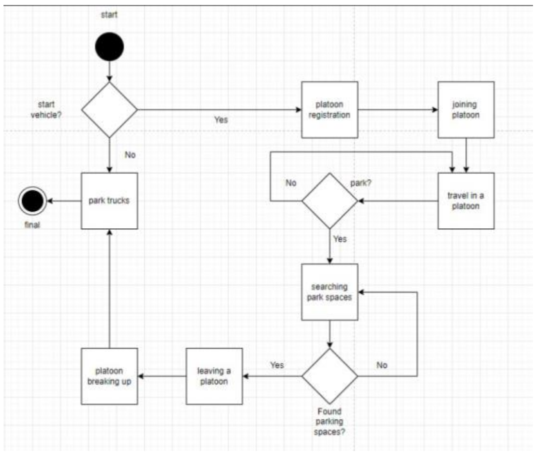
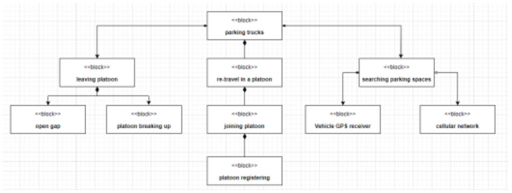


Fig. 22. Activity diagram for parking system

c. Define the block diagram of the system



<<block>> parking trucks
<<block>> leaving platoon
<<block>> re-travel in a platoon
<<block>> searching parking spaces
<<block>> open gap
<<block>> platoon breaking up
<<block>> joining platoon
<<block>> Vehicle GPS receiver
<<block>> cellular network
<<block>> platoon registering

Fig. 23. Block diagram for parking system

d. Define the internal block diagram of the system

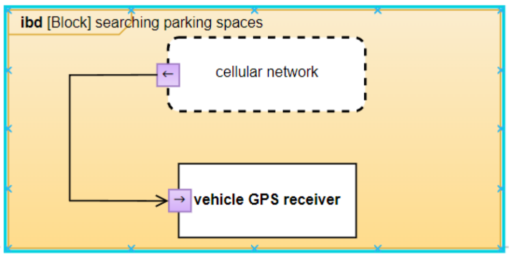


Fig. 24. ibd for searching parking space

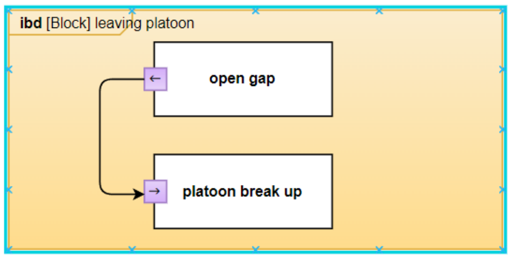


Fig. 25. Internal block diagram for leaving platoon

e. Define the sequence diagram of the system

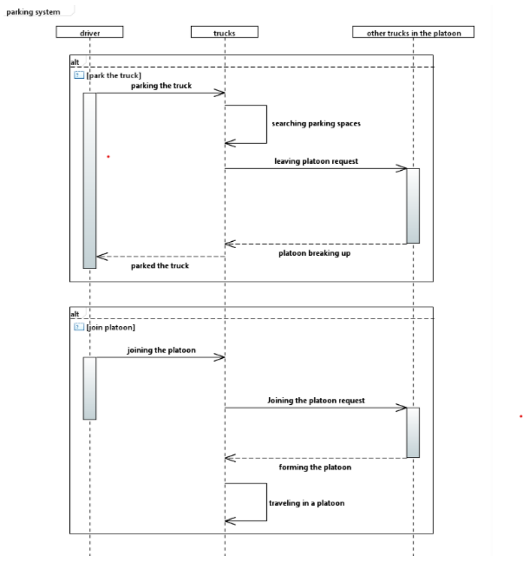


Fig. 26. Sequence diagram for parking system

f. Define the sequence diagram of the system

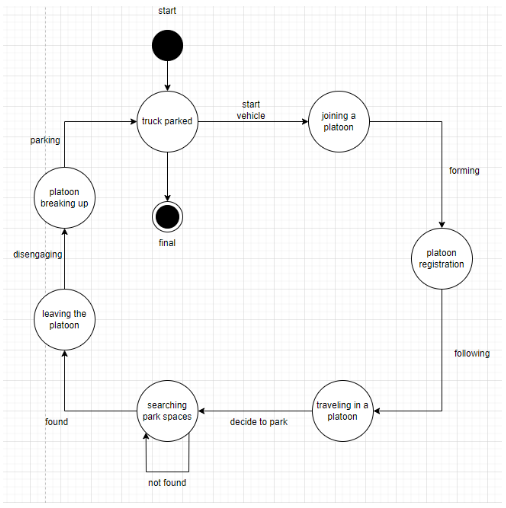


Fig. 27. State machine for parking system

III. *HARDWARE IMPLEMENTATION*

For the hardware implementation the following components were considered: an Arduino Uno board to process obtained signals, ultrasonic sensors HCSR04 to detect nearby objects, cameras OV5642 for pedestrian detection and lights detection and a screen to display messages. A simulation of the system was accomplished using the platform Tinkercad. Due to platform-related constraints, a camera could not be simulated and for this its simulation is achieved by entering specific values in the serial monitor.

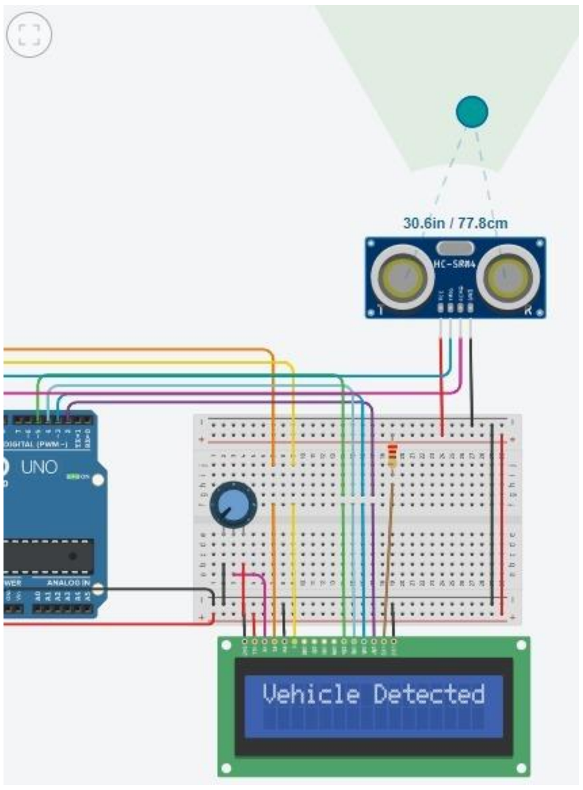


Fig. 28. Hardware implementation using Tinkercad simulator

The simulation considers the following cases:

1. If a pedestrian is detected the message “Slowing Down…” must be displayed.
2. If a pedestrian is detected after two seconds of being detected for the first time the message “Braking…” must be displayed.
3. If the sensor does not detect any pedestrian the message “Happy Journey!” must be displayed.
4. If a vehicle is detected next to the truck with its lights off the message “Vehicle Detected” must be displayed.
5. If a vehicle is detected next to the truck with its lights on the message “Opening Gap...” must be displayed and after 2 seconds the message “Opened Gap”.
6. If a vehicle is detected next to the truck with its lights off when the truck has an opened gap in front of it the message “Closing Gap...” must be displayed and after 2 seconds the message “Vehicle detected”.
7. If the sensor stops detecting a vehicle next to the truck when the truck has an opened gap in front of it the message “Closing Gap...” must be displayed and after 2 seconds the message “Happy Journey!”.

The following state machine shows the messages which will be displayed on the screen and the events that need to happen to display a different message.

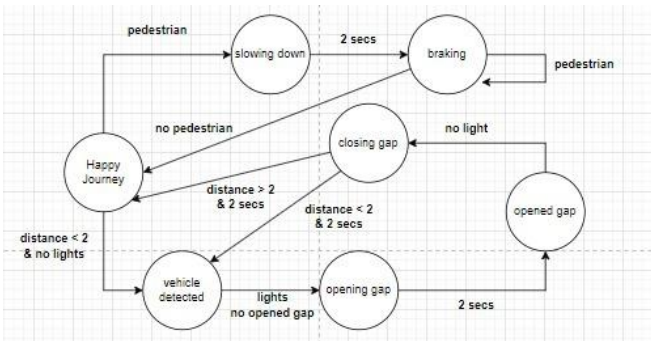


Fig. 29. State machine diagram for hardware implementation

The behaviour of the implemented code is shown in the following diagram.

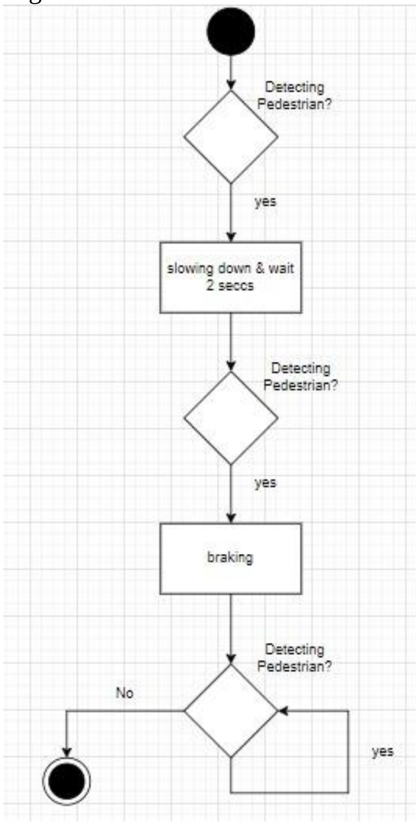


Fig. 30. Activity diagram for pedestrian detection

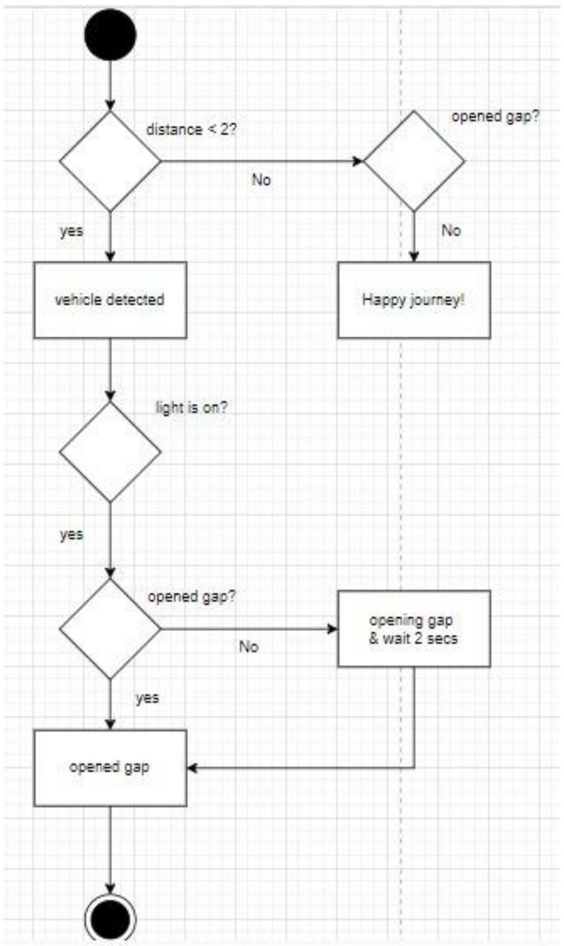


Fig. 31. Activity diagram for vehicle and lights detection

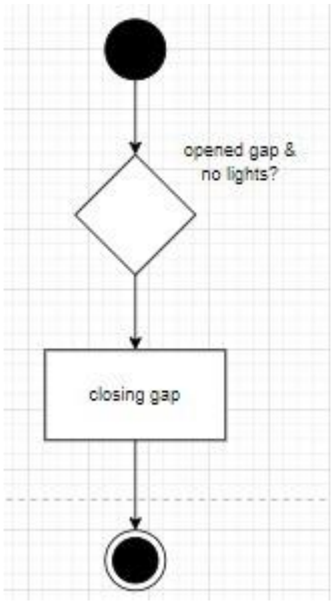


Fig. 32. Activity diagram for gap detection

IV. Scheduling

A. Specified requirements to the level of scheduling

Truck platoon system is naturally a real time operating system; therefore, it has to perform tasks based on the real time input from the sensors. These tasks are related to the control of certain events reacting to them. Failure to perceive the real time data could lead to fatal accidents, therefore this system can be considered as a hard-time real time system.

In real-time systems, the scheduler is considered as the most important component which is typically a short-term task scheduler. The main focus of this scheduler is to reduce the response time associated with each of the associated processes instead of handling the deadline.

Static real time system – In static real-time systems, where the task set is fixed and known a priori, all task activations can be precalculated off line, and the entire schedule can be stored in a table that contains all guaranteed tasks arranged in the proper order. Then, at runtime, a dispatcher simply removes the next task from the table and puts it in the running state. The main advantage of the static approach is that the runtime overhead does not depend on the complexity of the scheduling algorithm. This allows very sophisticated algorithms to be used to solve complex problems or find optimal scheduling sequences. On the other hand, however, the resulting system is quite inflexible to environmental changes; thus, predictability strongly relies on the observance of the hypotheses made on the environment.

Aperiodic task scheduling: In general, scheduling problems where pre-emption is allowed typically are simpler than those where it is not allowed. In cases where tasks are not synchronous but may arrive at arbitrary times, pre-emption becomes important. In a non-pre-emptive scheduling algorithm, the scheduler must ensure that a newly arriving task will never need to interrupt a currently executing task in order to meet its own deadline. This guarantee requires a considerable amount of searching. If pre-emption is allowed, however, this searching is unnecessary, since a task can be interrupted if a more important task arrives. The problem of scheduling a set of n independent tasks on a uniprocessor system, when tasks may have dynamic arrivals and pre-emption is allowed (1 | preem | Lmax).

The algorithm, called Earliest Deadline First (EDF), can be expressed by the following theorem [Hor74]:

Theorem 3.2 (Horn) Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.

Parameters of real time task:

Ai -> Arrival time is the time at which the task arrives in the ready queue

Ci -> Computation time is the time required for the execution of task without interruption also known as worst case execution time (WCET)

Di -> Deadline is the time before which the task has to complete its execution

Si -> Start time is the time at which the task starts its execution

Fi -> Finish time is the time at which the task finishes its execution

Ti -> The period of a task is the time between successive executions

With scheduling periodic processes that have deadlines equal to their periods, EDF has a utilization bound of 100%. Thus, the schedulable test for EDF is:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1,$$

EDF is also an optimal scheduling algorithm on non-pre-emptive uniprocessors, but only among the class of scheduling algorithms that do not allow inserted idle time. When scheduling periodic processes that have deadlines equal to their periods, a sufficient (but not necessary) schedulable test for EDF becomes:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 - p,$$

Where p represents the penalty for non-pre-emption, given by max {Ci}/min {Ti}. If this factor can be kept small, non-pre-emptive EDF can be beneficial as it has low implementation overhead. [Giorgio C. Buttazzo, 2011]

Task	WCET (Ci)	Deadline (Di)	Period (Ti)	Ci/Ti	Response Time
Braking	1	3	4	0.25	3.6
Gap Adaptation	3	5	20	0.15	8
Object Detection	1	3	5	0.2	3.75
Cruise Control	1	5	6	0.16	5.75
Lane Keeping	2	4	20	0.1	5

Fig. 33. Scheduling constraints

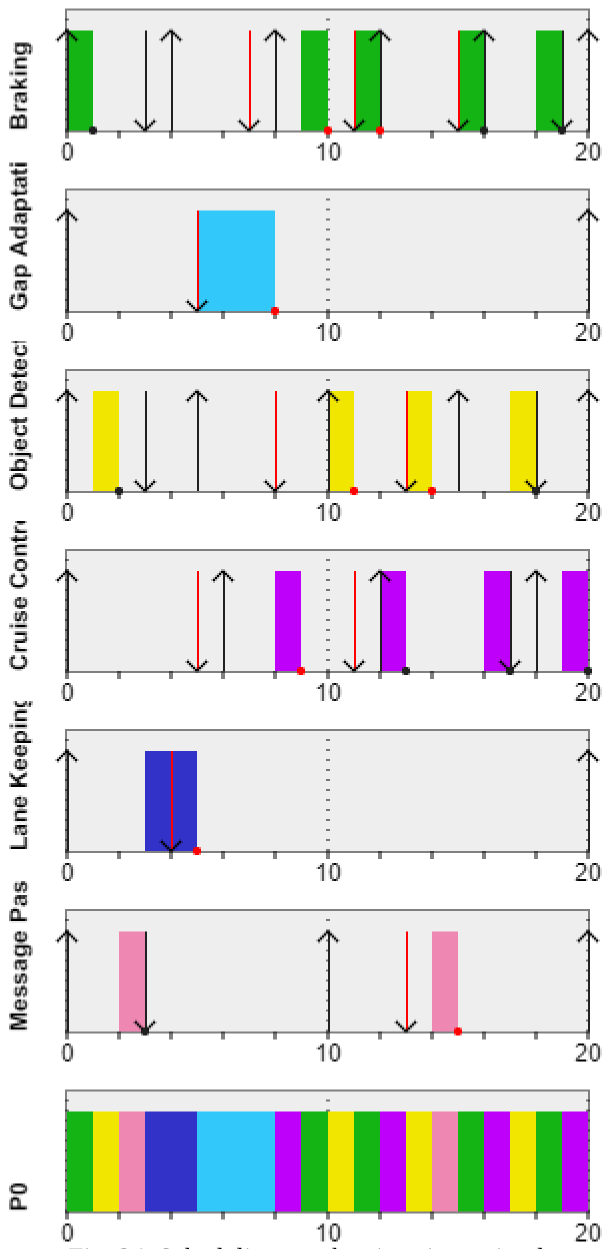
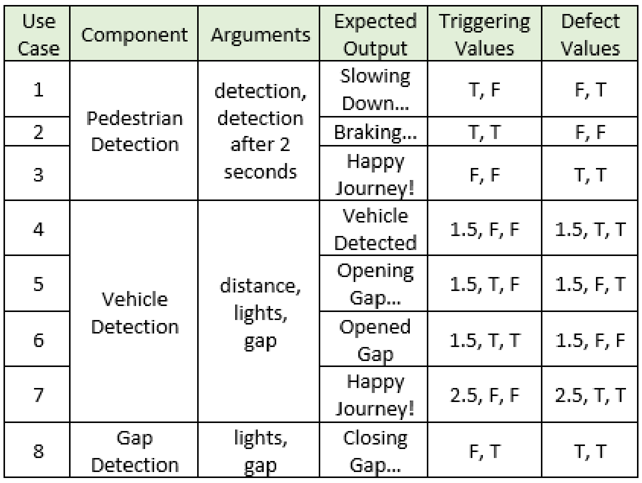


Fig. 34. Scheduling graph using simso simulator

V. *V&V (TESTING)*

Testing was performed using Eclipse IDE and Junit to validate the code implemented in the Arduino board, thus three different components were created to simulate the Arduino code with specific use cases for each component considering wrong values for each use case as well. The first component was employed for pedestrian detection validation considering three use cases, the second one to validate the detection of a vehicle with or without lights contemplating four different scenarios and the last component was used to validate the algorithm for detection of a gap when lights are not detected.

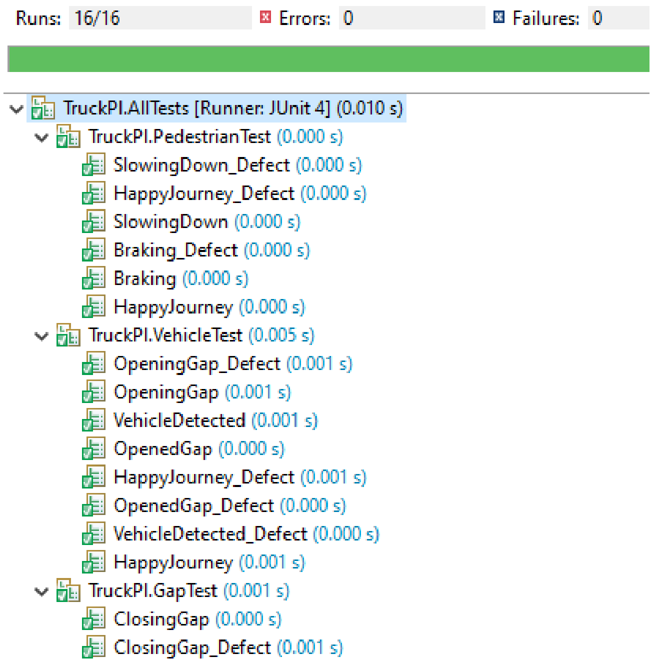
The following table shows the components for the testing process using Junit. Depending upon the arguments received by each component it is expected a specific returned value. To validate the performance of the system, not only triggering values were simulated for each use case, but also defect values. We must be sure that the system will have the expected behaviour only with the triggering values shown below.



Use Case	Component	Arguments	Expected Output	Triggering Values	Defect Values
1	Pedestrian Detection	detection, detection after 2 seconds	Slowing Down...	T, F	F, T
2			Braking...	T, T	F, F
3			Happy Journey!	F, F	T, T
4	Vehicle Detection	distance, lights, gap	Vehicle Detected	1.5, F, F	1.5, T, T
5			Opening Gap...	1.5, T, F	1.5, F, T
6			Opened Gap	1.5, T, T	1.5, F, F
7			Happy Journey!	2.5, F, F	2.5, T, T
8	Gap Detection	lights, gap	Closing Gap...	F, T	T, T

Fig. 35. Values given for testing system using Junit

As expected, every use case was validated considering also the defect cases. The next figure shows the 16 tests including those with defect values.



```
Runs: 16/16        Errors: 0        Failures: 0

TruckPl.AllTests [Runner: JUnit 4] (0.010 s)
  TruckPl.PedestrianTest (0.000 s)
    SlowingDown_Defect (0.000 s)
    HappyJourney_Defect (0.000 s)
    SlowingDown (0.000 s)
    Braking_Defect (0.000 s)
    Braking (0.000 s)
    HappyJourney (0.000 s)
  TruckPl.VehicleTest (0.005 s)
    OpeningGap_Defect (0.001 s)
    OpeningGap (0.001 s)
    VehicleDetected (0.001 s)
    OpenedGap (0.000 s)
    HappyJourney_Defect (0.001 s)
    OpenedGap_Defect (0.000 s)
    VehicleDetected_Defect (0.000 s)
    HappyJourney (0.001 s)
  TruckPl.GapTest (0.001 s)
    ClosingGap (0.000 s)
    ClosingGap_Defect (0.001 s)
```

Fig. 36. Unit test results in Junit

VI. CONCLUSION

In this article, we have discussed about the truck platoon system as a form of embedded system. We have refined some of the use cases from the requirement diagram and illustrated them using the sysML models. In addition to that, we have created separate tasks for each use case, investigated and researched on the suitable scheduling algorithm using event-based control strategy for the platoon system subject to V2V communication and time constraints. The scheduling algorithm used is Earliest Deadline First, and the runtime is chosen in such a way that it meets the schedulablity test. The scheduling graphs were generated using simso simulator.

Based on the use cases and diagrams, we have successfully proved that the algorithm and simulation satisfied our assumptions. Also, verification & validation by using Junit framework shows that the system works functionally as expected.

Furthermore, the simulation on the tinkercad provided us some amazing results in order to speculate our assumptions. In the future research, the real arduino board and cameras can be used to obtain accurate results.

VII. REFERENCES

[1] Flores, Carlos & Merdrignac, Pierre & Charette, Raoul & Navas, Francisco & Milanes, Vicente & Nashashibi, Fawzi. (2018). A Cooperative Car-Following/Emergency Braking System With Prediction-Based Pedestrian Avoidance Capabilities. IEEE Transactions on Intelligent Transportation Systems. PP. 1-10. 10.1109/TITS.2018.2841644.

[2] L. Konstantinopoulou, A. Coda, e.a. 2018. Functional specification for white-label truck, D2.4 of H2020 project ENSEMBLE, (www.platooningENSEMBLE.eu)

[3] L. Wu, L. Zhang and Q. Zhou, "Event-Based Control and Scheduling of a Platoon of Vehicles in VANETs," in IEEE Access, vol. 9, pp. 166223-166233, 2021, doi: 10.1109/ACCESS.2021.3135439.

[4] Zhenhai, Gao & Wei, Yan. (2016). A Headway Control Algorithm for ACC Vehicles with the Compensation of the Preceding Vehicle Acceleration. Procedia Engineering. 137. 669-679. 10.1016/j.proeng.2016.01.304.