CHALMERS
UNIVERSITY OF TECHNOLOGY

Chalmers University of Technology

Robotized Test Setup for Autonomous Driving - Virtual Objects Injection

Master's thesis in Electrical Engineering

Preethi Bodduluri
Shiva Ajay Jagadeepan

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:NN

Robotized Test Setup for Autonomous Driving - Virtual Objects Injection

Building a test setup for the injection of virtual objects to the real car for the better verification of the active safety functions

Preethi Bodduluri
Shiva Ajay Jagadeepan

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems, Control and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2018

Robotized Test Setup for Autonomous Driving - Virtual Objects Injection
Building a test setup for the injection of virtual objects to the real car for the better verification of the active safety functions

Preethi Bodduluri
Shiva Ajay Jagadeepan

Supervisor: Siddhant Gupta and Francesco Costagliola, Volvo Car Corporation
Examiner: Knut Åkesson, Electrical Engineering, Chalmers

Master's Thesis 2018:NN
Department of Electrical Engineering
Division of Systems, Control and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Visualization of real object tracking by RADAR and Camera sensors of an Autonomous car. In this thesis we bypass the RACam part and inject the virtual objects directly into the ECU of the car.
Printed by: Chalmers Reproservice/Department of Electrical Engineering
Gothenburg, Sweden 2018

Robotized Test Setup for Autonomous Driving-Virtual Objects Injection
Building a test setup for the injection of virtual objects to the real car for the better verification of the active safety functions

Preethi Bodduluri, Shiva Ajay Jagadeepan
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This master thesis focuses on injecting virtual objects to active safety ECU of the test car, to verify the active safety and autonomous driving functions of the car. Virtual objects like a pedestrians, cyclists, cars and other stationary and dynamic objects are placed in front of the car to ensure that the car activates its active safety and autonomous functions by itself. These objects are used instead of the real obstacles to avoid causing damage to the real actors. Another advantage is that, damage cost can be reduced. High risk scenarios which can't be tested in real time can be implemented with ease by injecting virtual objects.

To perform the injection, vehicle-in-the-loop methodology is implemented by creating a virtual object in the simulation environment and sending it to the car. The information of the created object in the desktop simulation environment, that is the position, height, speed etc. are acquired in a SIMULINK model. This SIMULINK model then sends the data using a CAN-box which acts as a bridge between the ECU of the car and the desktop simulator. This hence, efficiently replaces the CAN-bus message that would have been sent by the physical sensors to the ECU of the car when a real actor is detected in the same physical environment. The car then performs the required active safety function according to the scenario.

Finally, to evaluate the methodology, different scenarios are implemented. This is done to check if the active safety functions like autonomous emergency braking, collision forward warning on the physical car are working according to the expected behaviour.

Keywords: Autonomous vehicles, Active Safety Function, ECU, Testing Hardware-in-the-loop, Software-in-the-loop, Vehicle-in-the-loop

Acknowledgements

We would like to thank our supervisor at Chalmers, Knut Åkesson for guiding and driving us in the right direction through entire course of the project.

Our special thanks to the supervisors at Volvo Car Corporation, Siddhant Gupta and Francesco Costagliola for providing us with right tools and valuable inputs to successfully complete the project. Their knowledge and expertise in the topic has helped us a lot.

Additionally, we would like to thank Goksan Isil and Junhua Chang for clearing our doubts during the project. Finally, we would like to thank our friends and family for their continuous encouragement and support.

Preethi Bodduluri, Shiva Ajay Jagadeepan, Gothenburg, 2018

Contents

List of Figures

List of Tables

1
Introduction

Autonomous cars make the mobility safer, sustainable and more convenient with the help of the equipped sensors that read the surroundings and adapt to the changing traffic conditions, it navigates itself without any human intervention. It ensures safer journey for everyone, helps save fuel, provides a smoother ride, allows everyone to relax on board and spend time efficiently. Volvo cars have been doing many research projects for the development of this fully autonomous driving cars.

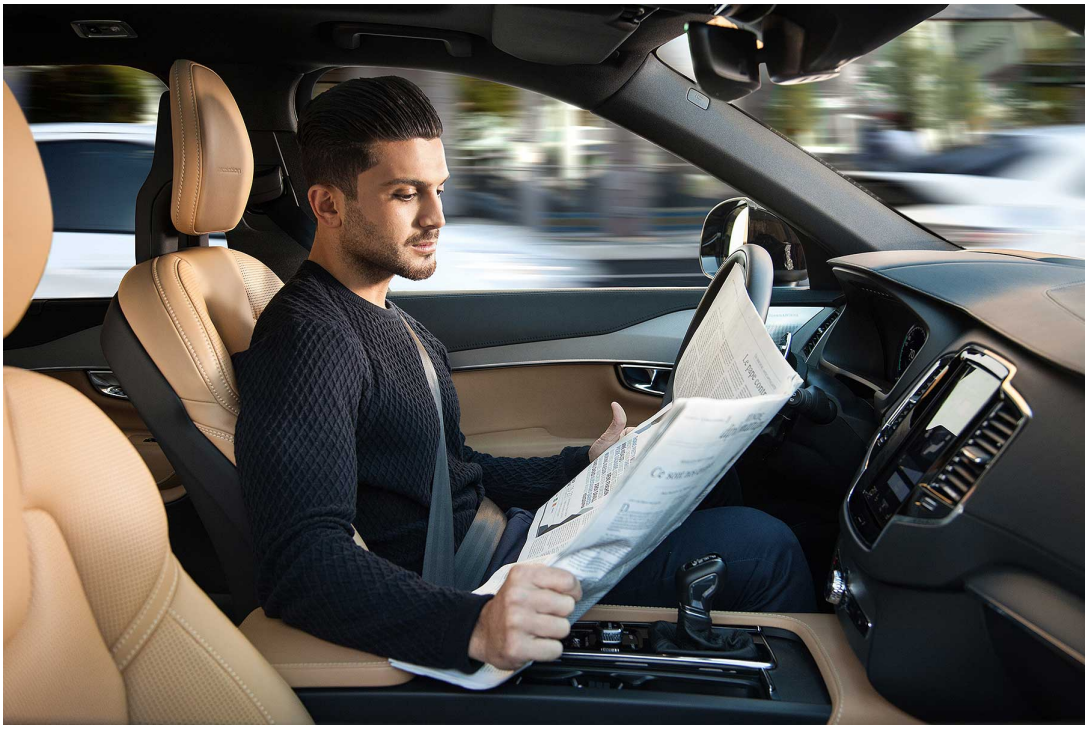

Figure 1.1: The figure shows a person relaxing on board and no longer requires constant attention on the road (Source : Drive Me Project - Volvo Cars).

The development of the Autonomous cars raises the major concern to ensure safety. The motive of Volvo Cars is to reduce the fatality rate by using the Active Safety and Autonomous Driving (AD) functions. In order to ensure the working of these functions in the real time, it is necessary that extensive testing is done. During the development of these functions there is usually a large parameter space that has to be investigated and some scenarios have to be tested using the computer simulations and on real vehicles.

Though there are several tests that can be performed on the closed test track using vehicle under tests and targets, the efficiency of such automated testing systems are not good. These tests bring the necessity that all the parameters should be handled with high accuracy and the replication of the test scenarios should be fast in the real time.

Volvo cars along with project partners has developed a concept called Steer by Server with an aim of increasing the efficiency of testing the future AD functions. This concept of novel fully-automated test system involves testing on a mix of real cars and radio-controlled test dummies in real time, and virtual objects. The signals from the robotized targets are collected by the server and it controls the trajectories. This is done to ensure that specified scenarios are followed by avoiding any conflicts. A closed loop communication has to be developed between the server and the test targets.

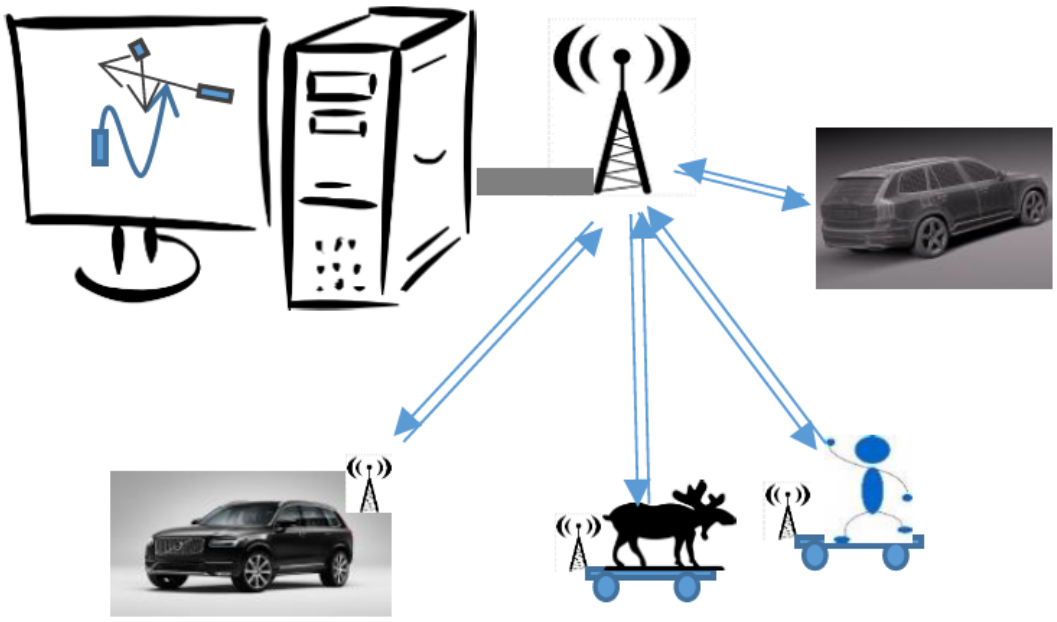

Figure 1.2: The figure shows different actors of the testing environment like a real car, a virtual car, a pedestrian, a moose and a server controlling the actors with a simulator connected to it.

1.1 Purpose

The purpose of the thesis work is to verify the capabilities of active safety functions in autonomous driving vehicles by reproducing the potential high risk scenarios in a virtual environment with a physical vehicle-in-the-loop methodology, this can be observed in papers [4] [5].

The main challenges with the verification of the autonomous drive are to make sure that the generated virtual environment for testing the test vehicle, is an exact replication of the real world. The whole purpose of replication is to send the virtual objects to the real car at the same location and same time stamp where the objects have been placed in the virtual environment. This is done to ensure that the vehicle under test triggers the safety functions corresponding to the injected virtual obstacles. Modeling and predicting uncertainties can also be a problem while performing rigorous testing for high risk scenarios. Hence, verifying the autonomous driving functions is important.

The current thesis is a part of a project, creating a robotized test setup for testing the safety and autonomous driving functions of autonomous cars. Where the first part of the project involves the generation of dynamic trajectory that is achieved using a path planning algorithm which finds the best path to travel from a fixed place to another, the second part involves taking the vehicle dynamics into account and modeling a control algorithm for the control of car, also includes synchronizing with the created path planning algorithm and making sure that the car follows the proposed trajectory.

This thesis work involves injection of the virtual objects to the steer by server environment to ensure efficient and reproducible testing of Autonomous driving functions through a simulator. Scenario components such as cars, roads, guard rails, pedestrians or cyclists and all other objects other than the test vehicle are injected as virtual objects. High risk scenarios can be created and analyzed in order to ensure the working of the safety functions in the Autonomous test vehicle. The implementation of this is tested using both computer simulations and practically on a real car. This can be done in the simulation environment and can later be interfaced to the real test vehicle for testing it on the test track. This thesis will contribute to the challenges currently faced by the vehicle manufacturers for testing their autonomous vehicles, without the loss of life and property.

1.2 Scientific Challenges

The aim of the thesis work is to verify the capability of autonomous driving vehicles by reproducing the potential high risk scenarios in a virtual environment with a physical vehicle-in-the-loop. Following research questions will be addressed in the thesis work:

1. Assuming that the Autonomous driving functions are implemented in the physical car, evaluate how accurate is the performance of the functions using the vehicle-in-the-loop method? - This question focuses on understanding the accuracy of a testing method called vehicle-in-the-loop where the implementation involves merging the virtual environment with the real environment.

2. What are the potential and challenging scenarios that can be tested with the given setup for the verification of the Autonomous Driving Vehicles? - This question directs towards understanding the recreation of various dangerous situations or scenarios that can happen in the real world, in a virtual environment.

1.3 Outline of the thesis

The document is structured to explain all the aspects of the thesis. The chapter 1 includes introduction explaining in brief the background of the thesis and related challenges. Then, the chapter 2 gives an overview of the methods followed to implement the project. In Chapter 3 the complete system architecture of the methods are explained. Chapter 4 includes the evaluation and finally conclusions including the future work are discussed in chapter 5 and 6 respectively.

2
Testing Methods

The testing and verification of the active safety functions is necessary to ensure safe riding of cars. The functionality of ensuring safety of autonomous vehicles can be tested using various testing methodologies. The Forward Collision Warning(FCW) and Autonomous Emergency Braking(AEB), are considered to be some of the important safety functions that have to be tested before releasing the car to the day-to-day usage. Among the several methods, simulation method is one of the testing methods where the actual vehicle behavior is replicated with sensors and vehicle model in a simulation environment. The developed autonomous driving and active safety function modules can be tested using SIL (Software-in-the-loop), HIL (Hardware-in-the-loop), VEHIL (Vehicles-in-the-loop) or mixed simulation methodologies as mentioned in [6]. Hardware-in-The-Loop (HIL) and Software-in-The-Loop (SIL) simulations have long been used to test electronic control units (ECUs) and software.

Software-in-The-Loop testing or SIL testing is used to describe a test methodology where executable code such as algorithms (or even an entire controller strategy) usually written for a particular mechatronic system is tested within a modelling environment, that can help prove or test the software.

Hardware-in-The-Loop or HIL testing is a test methodology that can be used throughout the development of real-time embedded controllers to reduce development time and improve the effectiveness of testing.

Vehicle-in-The-Loop or VEHIL testing is almost same as HIL testing methodology which includes a real test vehicle in the place of embedded controllers and through this test method the real vehicle behavior is validated.

	Vehicle Environment	Test Setup
SIL	Created by traffic Simulator	All ECUs and the CAN bus are simulated by Vector CANoe.
HIL	Created by traffic Simulator	Only one physical ECU is present and all the other ECUs and CAN bus are simulated.
VEHIL	Created by traffic Simulator	It is a real test setup in which all ECUs and CAN buses are physically present.

Table 2.1: This table represents the vehicle environments used for different methods of testing.

The detailed behaviour and components involved in these test methods are explained in the following subsections.

2.1 Software-in-The-Loop Testing

In this project, while performing SIL testing, the model-in-the-loop part is skipped since the model of the controller unit that is used to perform active safety functions is already present. The entire controller strategy part is made of executable programs and algorithms. In this the traffic simulator with all simulated ECUs are created in Vector CANoe platform. The traffic simulator is nothing but module that creates traffic scenarios and gives the object information like the object's relative position, velocity, width, height corresponding to the test car, this can be found in the paper [7].

Advantages of using SIL testing, according to [8] is that SIL testing and simulation does not require real-time execution, as it uses the program codes, algorithms and functions to perform various tasks. It requires special drivers or hardware to run in real-time. Sometimes the software runs much faster than real-time, due to the code's simplicity and high processing speed of the system in which the code is executed. But sometimes it runs slowly due to the low processing speed and complexity. If realistic timing information is required, then this can either be simulated numerically, or in some cases pseudo real-time blocks can be used in the modelling environment to pace the model as required. Additionally, SIL test scenario doesn't require hardware for its implementation if it is not run in real time, this eliminates the issues that a hardware can cause while execution.

Thus, Software-in-loop testing methodology can be used for it's low cost and flexibility. According to [9] it can also be understood that this is a useful technique for software proving at earlier stages as well as in the testing phase of the design.

2.2 Hardware-in-The-Loop Testing

As the complexity of electronic control units (ECUs) increases, the number of combinations of tests required to ensure correct functionality and response increases exponentially. Older testing methodology tends to wait for the controller design to be completed and integrated into the final system before system issues could be identified [10].

Hardware-In-the-Loop testing provides a way of simulating sensors, actuators and mechanical components in a way that connects all the inputs and outputs of the ECU being tested, long before the final system is integrated, this can observed in [11]. It does this by using representative real-time responses, electrical stimuli and functional use cases. In this method, the part of the simulation that represents the environment, sensors and associated hardware is called the plant model which is the model of the car and the parts of the simulation that represent other controllers are called controller models. Hence, according to the [12] HIL is advantageous because it uses both real sensors and virtual sensors, this helps in extreme verification of a functionality. It is also observed from the paper [13], that HIL testing can be used to implement the real world logic.

The Hardware-In-the-Loop testing is carried out by replacing the controller models for real controllers and uses further input and output to test those components. All the controllers within the scope of the test system are connected to their respective inputs and outputs and wiring harnesses, a detailed representation of this can be found in [14].

For a user who wants to perform testing with a HIL system will require a modelling environment, such as Simulink, CANoe, Automotive Simulation Models(Model desk, Control desk and Motion desk) or any other simulators, which can be used to create the car model and may include a model of the controller strategies for components not available for the test system. This environment is usually run on a workstation or a laptop and is known as the host computer.

2.3 Vehicle-in-The-Loop Testing

Vehicle-in-the-loop (VIL) methodology is advantageous over other methods for several factors. As explained in [15], Vehicle-in-the-loop methodology is used to create a safe, easily reproducible and resource saving testing environment. The ideology of this method is to synchronize the test vehicle with that of the test environment created on the simulator. This brings out combined advantages of using a real and simulated environment. In this setup, the car can be either real or simulated with vehicle dynamics acting on it and tested with some traffic scenarios. This testing methodology can be adopted for aggressive testing of the active safety systems or autonomous driving functions. By using this method the real actors are not affected by any failure of the system.

The main reason for choosing Vehicle-in-the-loop (VIL) method over the others for this project is that, it is a resource saving system. A further understanding of the system can be obtained by looking into the figure 2.1, it gives an overview of whole setup.

Figure 2.1 shows that two kinds of environments, the simulation environment and real environment are involved in the setup. Simulation environment is developed on a desktop based simulator. The data of the objects created on the desktop simulator is sent to the real car by using a suitable communication protocol, discussion regarding the choice of the protocol is done in the upcoming chapters. The ECU of the real car is the main brain to which the required information is sent from the simulator. The reactions of the car vary from it applying a brake to it controlling the speed. The reaction of the car to the information from the simulator gives an extra clarification that the active safety functions of the car are working good and no real actors are harmed in the process of verification.

The only thing which cannot be examined is the placement of the virtual objects in the real environment. This can be overcome by rendering the simulation environment where the virtual object are created, into a 3-D head-mounted virtual reality tool, which is been included in the future scope of this thesis work.

While the Figure 2.1 gives a brief explanation of the VIL setup, the benefits of the method is understood by implementing the Software-in-the-loop, Hardware-in-the-loop and Vehicle-in-the-loop methodology. Further details regarding the implementation can be found in the subsequent chapter.

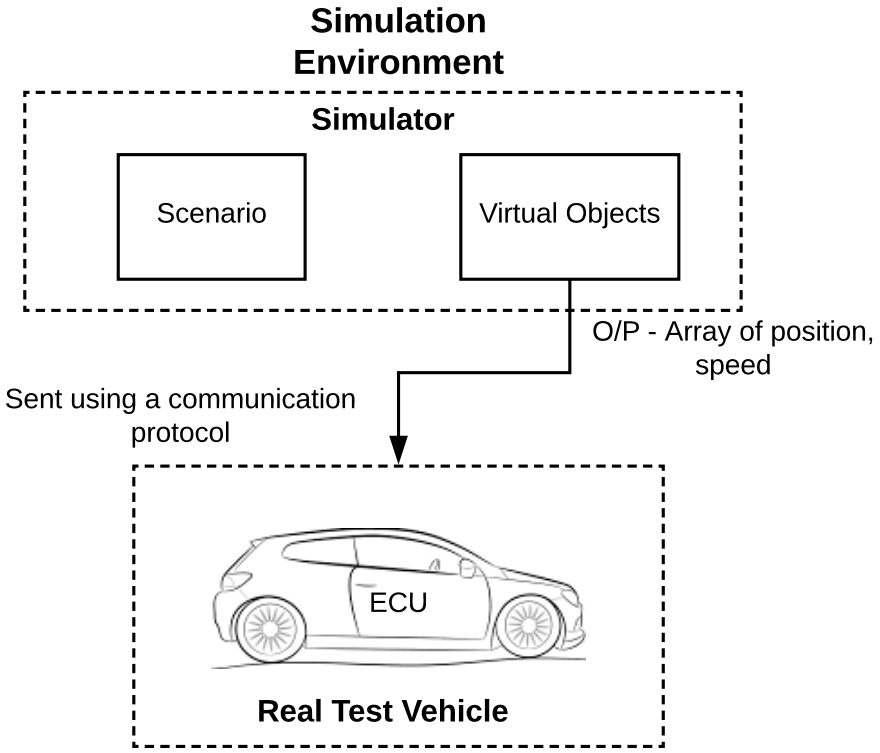

Figure 2.1: Represents the Vehicle-in-the-loop setup in brief. The flow of data such as the array of position, speed of an virtual object, from a desktop simulator to that of the real test vehicle is indicated in the setup.

3
System Architecture

This chapter focuses on providing a detailed description for the methodologies explained in chapter 3, that is, the implementation of Software-in-the-loop, Hardware-in-the-loop and Vehicle-in-the-loop methodologies are represented with their respective architectures. Basic block diagram, figure 3.1 explains the similarities and differences between the approaches. The commonality that is observed from figure 3.1 is that all the methods use a common traffic simulator to create an object list which consists of an array of position, speed, etc. This object list data is then sent to the respective ECU using a communication platform via a CAN bus. The data is received by different ECUs in the 3 methods, that is, in Software-in-the-loop approach simulated ECUs are used, in Hardware-in-the-loop approach a physical ECU and some simulated ECUs are used, in Vehicle-in-the-loop approach all the ECUs used are physical. The tools used in the block diagram and the methodologies are explained further in the below sub sections.

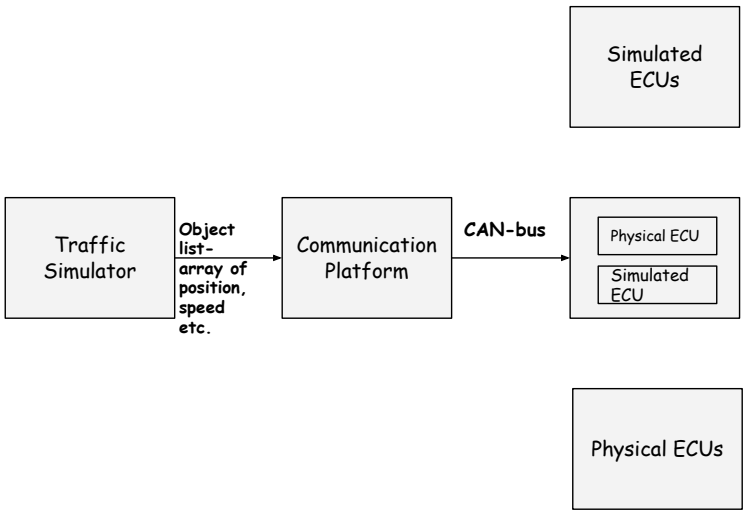

Figure 3.1: Block diagram indicating the common tools and features used in three testing methodologies, SIL, HIL and VIL.

3.1 Traffic Simulator

Traffic / Driving simulators are used to reproduce the environment of the real world. The replication of the environment or the surroundings is done in the form of real

time models. These models mimic the behaviour of combustion engines, vehicle dynamics, electric components, and traffic environment. ASM (Automotive Simulation Model), SPAS, VIRES - Virtual Test Drive are examples of some of the simulators. These simulators contain information of the environment in the form of the roads, traffic signs and dummy actors or pedestrians. This data from the traffic environment is retrieved for the implementation of the methodologies in the project.

To implement the project, Automotive Simulation Model (ASM) is the chosen simulator. The simulator consists of a model desk, this tool is used to set up initial parameters for the model. These parameters are used to start the simulation environment. The simulation environment is visualized in ASM with the help of a tool called motion desk, this tool is used to play the scenario created in model desk. From the different models of the simulator, ASM traffic information is the only data used in the project. Flexibility of ASM traffic makes it possible to create any type of traffic scenario for testing of active safety functions. Highly complex road networks and maneuvers on the roads can be recreated. Hence, high risk scenarios like the ones mentioned below are used in the project:

- Obstacle crossing in front of the test vehicle (Autonomous car) with a closer proximity. Figure 3.2 represents the start of the simulation of the said scenario.
- Collision avoidance when the test vehicle's speed is increased.
- Test vehicle turning in an cross section with many virtual objects as obstacles.

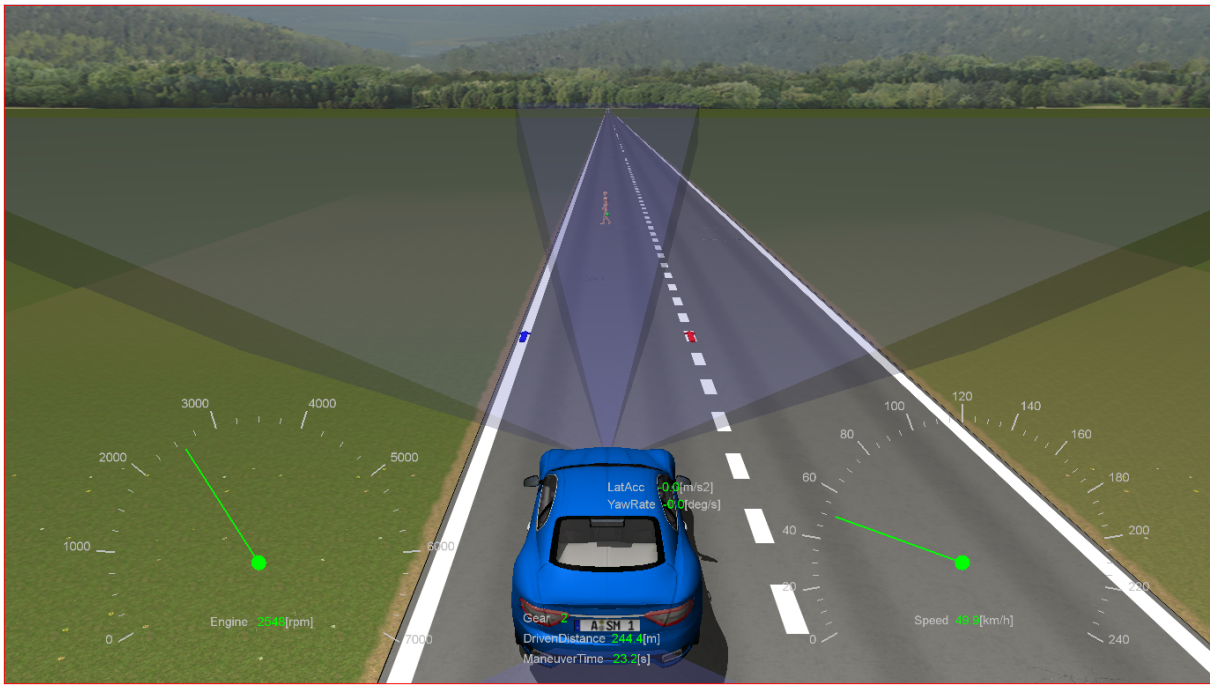

Figure 3.2: This figure depicts the visualization of the objects used in ASM traffic simulator for an example scenario.

The information from these complex scenarios are then transferred via the Simulink model using a CAN-bus to the ECU of the car.

3.2 Communication Platform - Vehicular Network Toolbox

Simulink contains an inbuilt tool box that is used to send CAN messages to the ECU of the car. In order to send messages over this interface, initially the parameters are set according to the requirements by using a constant block. Then, a delay block is introduced to ensure that the data is output after a fixed delay, so that real time interface with that of the hardware can be maintained. A rate transition block is introduced so that different working rates of the blocks don't cause an issue. Before the messages are packed into a CAN message with the required identifier and data bytes, the parameters are sent through a data type conversion block to ensure uniformity. This packed message is further sent to the CAN transmit block which sends the message to the selected CAN device. Required actions are handled by the ECU after receiving the messages. The implementation can be observed from the figure 3.3.

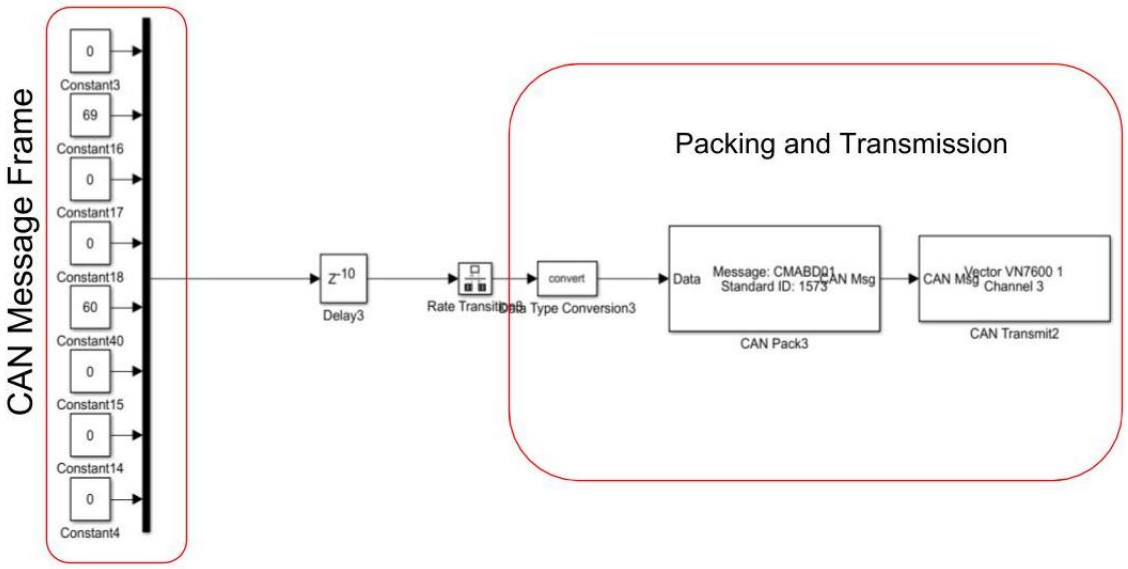

Figure 3.3: The figure indicates the conversion of the CAN messages data into required format and the transmission of the data to the ECU via a CAN transmit block.

3.3 ECU

The active Safety ECU of the car is responsible for the control of the active safety functions, this module contains all the active safety functions that get triggered when an object or an unusual scenario is detected.

3.3.1 Sending CAN messages to ECU

To initiate the injection of the object list, a CAN message is sent via the CAN channel to the active safety ECU. The ECU then enters into the data Injection Mode. Data Injection Mode is activated or deactivated by sending XCP user defined command. It is done prior to sending the object data. It is represented as shown in the table 3.1. An identifier indicating the injection mode is used before the byte information.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0xF1	0xB1	0x00	0x00	0x00	0x00	0x00	0x00

Table 3.1: This command is used to start or stop the Data Injection mode

Then, the ECU replies by sending an response acknowledgement. It gives back the response signal stating that the injection mode is enabled. Figure 3.2 represents the notation.

Byte 0	Byte 1	Byte 2	Byte3	Byte4	Byte 5	Byte 6	Byte 7
0xFF	0xB1	0x01(enable) or 0x00(disable)	0x00	0x00	0x00	0x00	0x00

Table 3.2: This command indicates the response received from the ECU.

Once the Injection Mode is activated, the fused Object parameters like the latitudinal position, longitudinal position, velocity, height and width, are sent through instrumented CAN to the ECU. The transmitted and the received signal to and from ECU are examined through the CANoe/CANalyzer, a Vector CAN communication instrument.

3.3.2 CANoe

Development, analysis and test of the ECU networks and individual ECUs can be performed using the CANoe tool. The development phase in CANoe is explained as a three phase model.

3.3.2.1 Phase 1

In Phase 1, firstly, the complete functionality of the system is distributed among different network nodes and the design is refined. Then, the messages are defined and the baud rate of the bus is selected. Later, each individual network node is specified with a particular bus behaviour either in terms of incorporating a complex protocol or by specifying the times at which the bus messages has to be sent. Once the setup is established, simulation can be performed to estimate and understand the bus load and latency times that can be expected for the specified baud rate. Event driven messages can also be specified on the network node to ensure proper transmission and reception of the messages.

3.3.2.2 Phase 2

Next phase in the model is to finish the simulation of the remainder of the bus and to establish the implementation of the components with that of the simulation. This creates a developed network node. CANoe tool then requires a real bus for the interface, this enables the simulation in real time.

3.3.2.3 Phase 3

The final phase is to ensure the integration of the whole system. All the created real network nodes are connected to the bus, this makes CANoe perform as an intelligent analysis tool that can monitor the message of the real network nodes. This monitored messages can be compared with that of the intended behaviour of the system. The whole system can thus provide a fair conclusion after analysis and testing.

The basic system requirements for each methodology have been explained in above sections, the next sections will focus on explaining the working of each approach in detail.

3.4 Software-in-the-loop Testing

In this methodology, no real hardware is used. The programming tool, CANoe is used to simulate the behaviour of the car with active safety functions. The active safety ECU (Electronic Circuit Unit) handles the active safety functions of the car. The functions of the ECUs are replicated by using include files that contain the unit level replication of the real behaviour of ECU hardware. All other ECUs are interconnected via a backbone of the system in CANoe. Backbone is the complete architecture of the car. The messages between the ECUs on the backbone is exchanged via a virtual CAN (Controller Area Network) bus.

To inject the objects to the ECU, as shown in figure 3.4, firstly, the objects are created in a simulation environment. ASM (Automotive Simulation Model) is the simulator used in the project, further information regarding the simulators has already been discussed in section 3.1. It generates an object list with the parameters representing the placement of the object in proximity of the car. The object list contains an array of the longitudinal position, lateral position, velocity, etc. These parameters are then sent to CANoe (a tool used to communicate with the ECU of the car) via a Simulink model. The Simulink model configures a CAN channel, this configured CAN channel is used by CANoe to send the data. The data is now converted into CAN format for the virtual ECU to understand.

The output signals from the virtual ECU are then monitored over the virtual FlexRay channel. Flexray is a high-speed communication bus used in automotive network communication protocols. The Flexray channel helps in monitoring the active safety functional signals. These signals are used to check if the active safety functionality of the car is causing the virtual ECU to perform the braking function

or not. The main signals monitored are:

- Frontal collision warning signal that indicates if the object is placed in front of it.
- Active Emergency Braking and the deceleration request signal indicates if the braking is achieved.

Hence, Software-in-the-loop testing methodology explained above provides the understanding of the real vehicle performance in a simulation environment but it lacks to replicate the troubles a real hardware causes during the implementation in the real environment. This issue calls for exploring other testing methodologies.

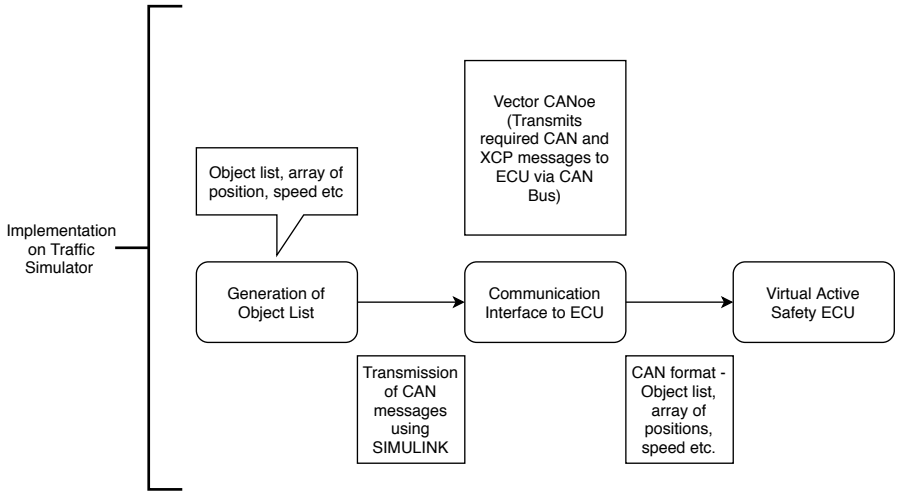

Figure 3.4: Diagram represents the architecture of the system using complete virtual testing. The blocks represent the flow of the data from one module of the system to the other using different interfaces as explained in the text.

3.5 Hardware-in-the-loop Testing

HIL testing methodology is performed in a rig. This setup is an extension to the SIL testing method, where the virtual active safety ECU is replaced with an instrumental ECU and sensors are emulated to enhance the verification process. The hardware used in the setup are connected using Flexray, Ethernet and CAN cables to ease the flow of information among them. The setup of the HIL rig for performing the verification of the active safety functions usually contains a camera attached to the ECU along with a traffic simulator. In this methodology, firstly the traffic simulator (VIRES) is used to create scenarios replicating the real world traffic situations. These scenarios are then played on the screen, that is placed in front of the camera. Camera continuously captures the images of the screen playing the scenario and sends the information as a camera input to the ECU. The simulator also contains a RADAR model which mimics the working of a real RADAR sensor. The Radar model is used to send the data in the form of detections. These detections indicate that an object is placed in front of the ego car in the simulator. If the model detects

that the object is in the range of the sensor, only then it sends the data to ECU via the Ethernet cable. This RADAR detection data along with the frames captured by the camera are fused together to form the RaCam fused data, this improves the accuracy of the placement of the created object. The setup of the rig can be observed from the block diagram 3.5. This fusion data is then used by the ECU to activate the active safety functions.

In this project, HIL rig is modified to enhance the verification process. In the modified version, a desktop simulator (ASM) is used to re-create the real world scenario. The VIRES simulator connected to the rig, sends the host vehicle details to the ECU and runs the scenario continuously. The host vehicle data is the same in both the simulators. A fused object list from ASM is then sent via a different CAN channel connected to the ECU. This list bypasses the RaCam sensor data that is already being processed in the ECU of the rig. This enables the ECUs ability to react to the list of data sent by ASM instead of just RaCam fused data. The object list data sent to the ECU is visualized through a software tool called CANalyzer. All the information handled by the ECU, is additionally visualized through a VDR (Video Data Recorder). The sent object list triggers the active safety functions of the ECU and causes the car to break, it is visualized on the simulator the rig uses.

Major problem to be tackled during the implementation of this setup is synchronization of the VIRES simulator time with that of the ASM simulator. It is also important to maintain the synchronization of time between the simulator and ECU. To solve the problem, time stamp information from the VIRES simulator is retrieved and used by ASM and also sent to the ECU. This ensures that no data is lost during injection of objects.

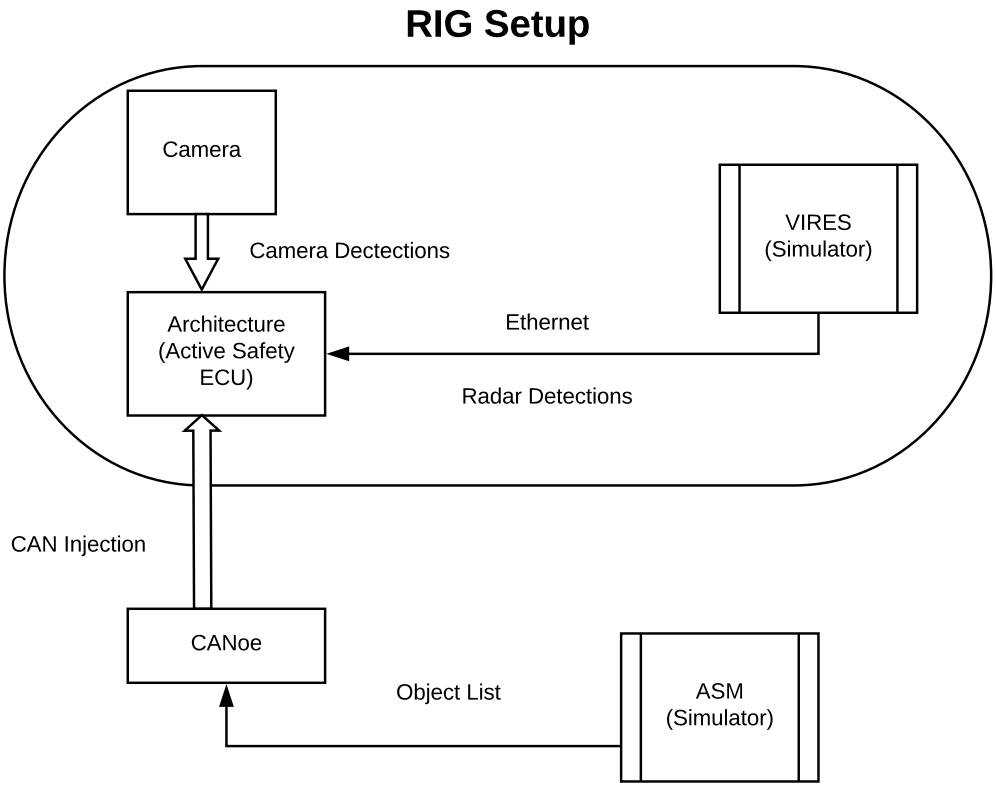

Figure 3.5: Diagram represents the architecture of the system using Hardware-in-the-loop setup.

3.6 Vehicle-in-the-loop Testing

Vehicle-in-the-loop method is implemented using a test vehicle. All ECUs in the car are involved for the purpose of testing the active safety functions. These ECUs continuously send and receive data among various nodes and themselves. As shown in figure 3.6, this methodology also uses a desktop simulator similar to other approaches, to send the required object data to the active safety ECU. The scenario created on the desktop simulator replicates the real time behaviour. CAN channel is used to send object list to the ECU of the car. If the correct object information is sent to the car, the safety functions like Forward Collision Warning, Active Emergency Braking gets activated and the car decelerates and brakes to indicate that an obstacle is in it's way.

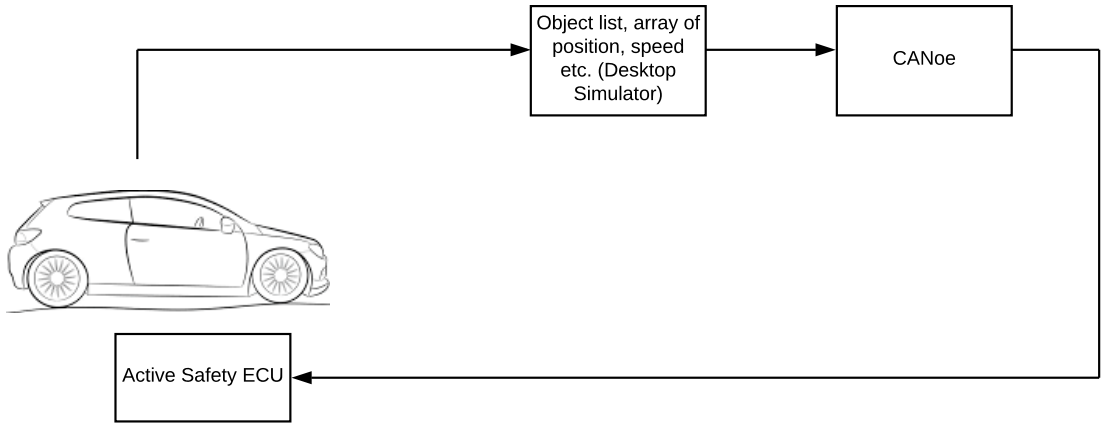

Figure 3.6: Diagram represents the architecture of the system using a real car. The blocks represent the flow of the data from the desktop simulator to the CAN bus via CANoe. The information is then sent via the CAN bus to the ECU, this triggers the active safety functions.

Hence, the three approaches give an insight about verifying the active safety functions of a car by not involving the real actors in the verification process. While Vehicle in the loop approach is more real time testing approach as the real car is tested in the process.

4
Evaluation

Once the setup is established, the next step is to test if the vehicle-in-the-loop methodology is a good testing method. The question to answer while testing the methodology is to understand how accurate do the autonomous driving functions work. In order to explain this, a set of scenarios are created and analyzed by logging the data received by the ECU. The Log contains the information of the signals that indicate the working of active safety functions. Currently two active safety functions are validated and the results are shown in the subsequent sections.

The high risk scenarios used for the testing are listed below:

- **Autonomous Emergency Braking:** In this scenario the autonomous test vehicle is driven at a moderate speed and there is a stationary virtual target before this vehicle. The autonomous car should decide either to steer before the virtual target approaches or brake and wait.

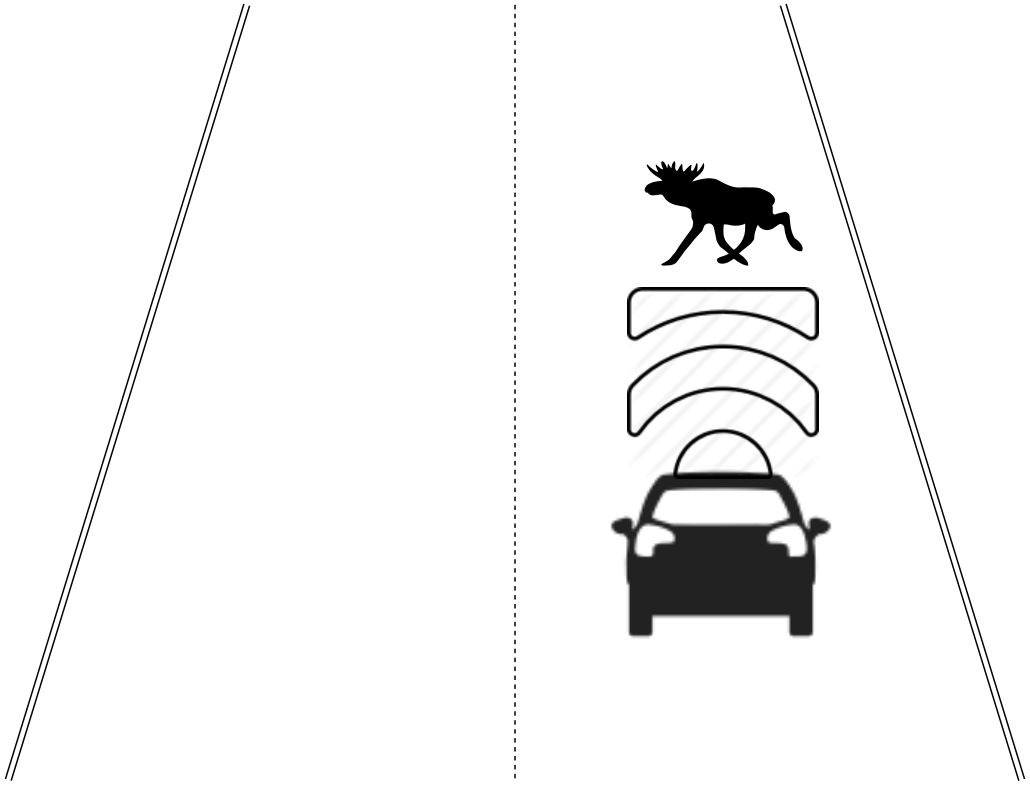

Figure 4.1: Autonomous Emergency Braking Scenario.

- **Adaptive Cruise Control:** In this adaptive cruise control scenario a virtual target running at a lower speed than the test vehicle is made to switch from the next lane to the lane where the vehicle under test is running. when the test vehicle has the target before that it has to slow down and follow the target with a constant distance between them.

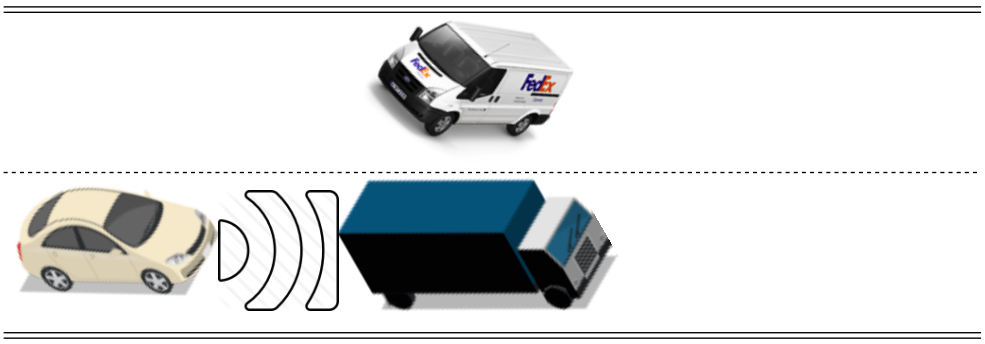

Figure 4.2: Adaptive Cruise Control Scenario.

Once the scenarios are created, the log data is recorded and analyzed as explained in the subsequent section.

Autonomous Emergency Braking

An Autonomous Emergency Braking (AEB) scenario is generated with one virtual target in front of the test vehicle. To obtain the required signals for evaluation the object data is sent via the vector CAN instrument to the ECU and a CANalyzer/-CANoe is used to retrieve the information from the ECU.

The sample log for the AEB scenario from the CANalyzer is visualized in the figures 4.3 and 4.4. In this figure four important signals to indicate the working of active safety functions can be observed. The first signal to monitor is the cllsnThreat signal it goes high if there is a threat in front of the car, the threat could be a pedestrian, a moose and other kinds of object. This data is sent from the Simulink model to the ECU via the vector CAN instrument. The next signal to monitor is the cllsnFwdWarnReq which goes high indicating that the car is about to brake now. Some important signals to monitor are the AsySftyDecelReq and the AsyBrkGainReq which indicate that the car is slowing down and is about to come to an immediate stop. The results shown in the figures 4.3 and 4.4 are obtained from testing the scenario in a rig. The figures 4.3 and 4.4 prove that the test is successful.

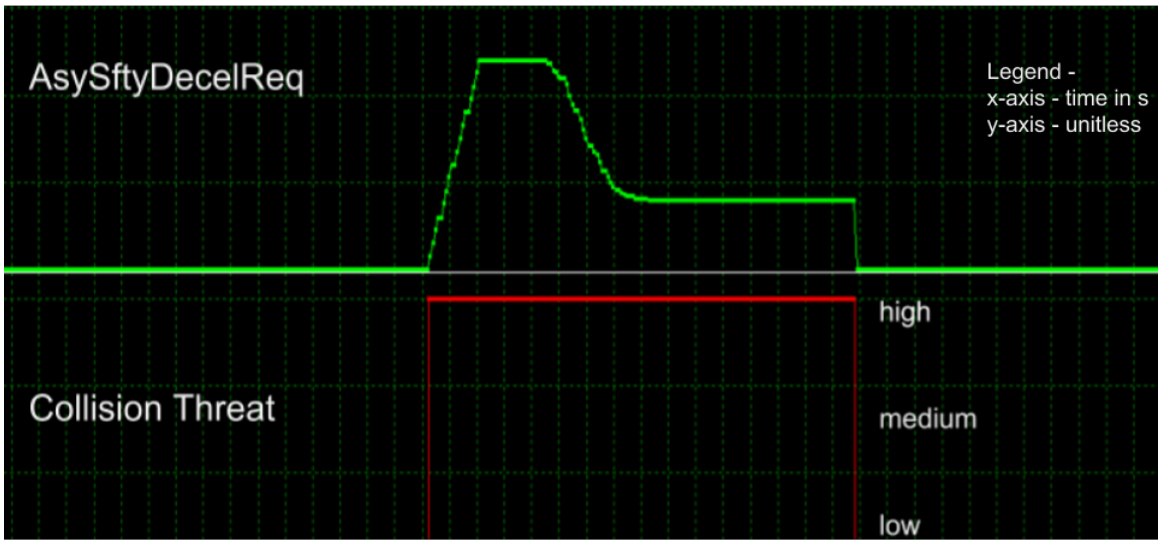

Figure 4.3: From the figure two signals can be observed, AsySftyDecelReq signal shows that the car is slowing down to apply brake signal at certain fixed point, and final signal is the cllsnThreat, this indicates the presence of a object in front of the car.

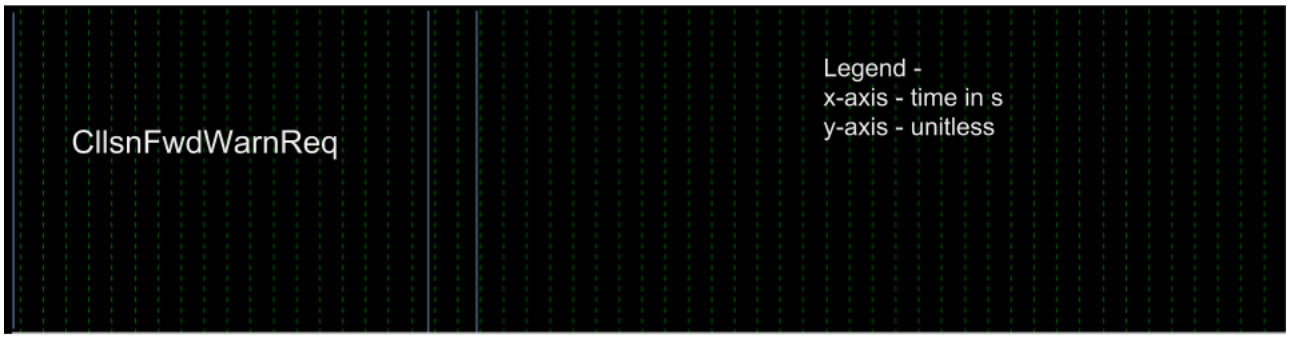

Figure 4.4: The graph indicates CllsnFwdWarnReq signal,it is the request sent by the active safety ECU to indicate that it has to start braking.

When the same setup is carried out on the real testing vehicle, it can be observed that similar results are obtained from the ECU of the test vehicle. Figures 4.5, 4.6, 4.7 when compared with that of the figure **??** it can be observed that the CllsnFwd-WarnReq signal, cllsnThreat signal, AsySftyDecelReq signal gave out the results as intended.

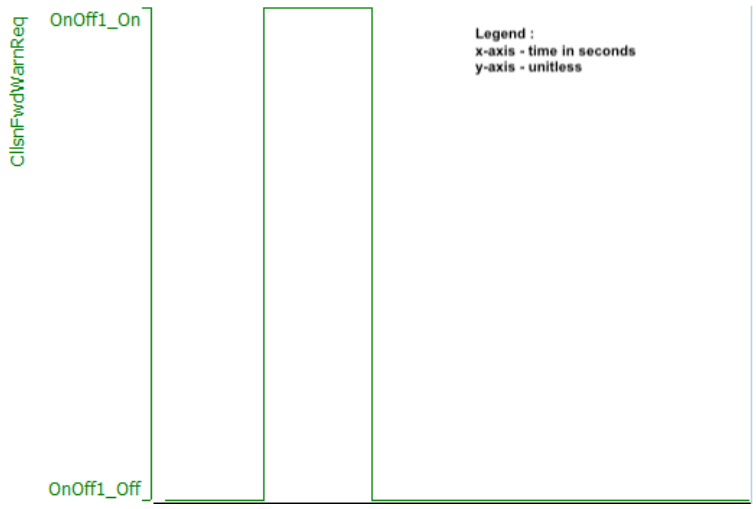

Figure 4.5: This graph indicates a Collision forward warning request signal that gets activated when an object is placed in front of it. It goes high for an object in front of it and goes low when it disappears.

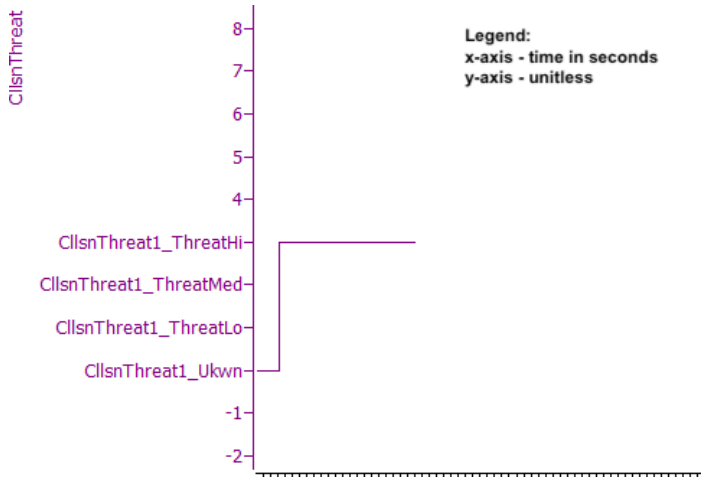

Figure 4.6: This graph indicates a collision threat signal, it goes high on an oncoming threat to the car. The range where the threat is closing towards the car is indicated with the different level notations on the signal, such as the collision threat low, high and medium.

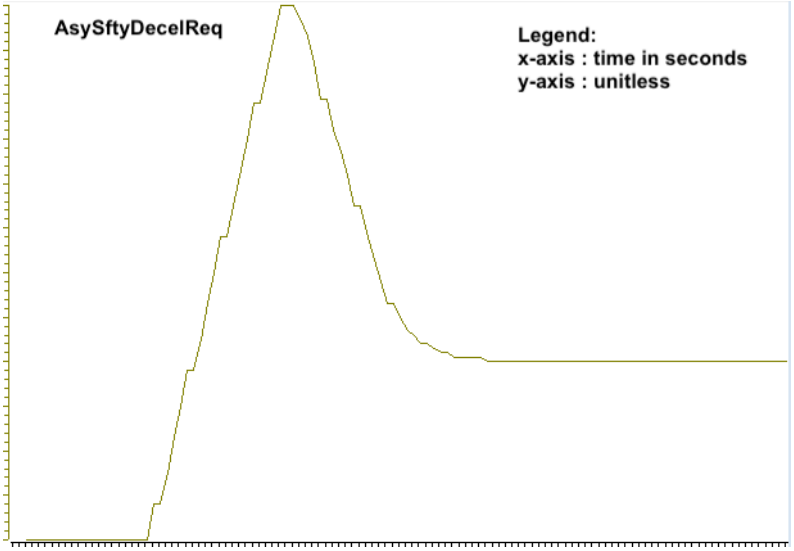

Figure 4.7: This graph indicates the deceleration signal output of the car from the ECU when it starts decelerating to avoid possible collision of the to the car.

The figures 4.8, 4.9, 4.10 represent the graphs which are obtained when a stream of data from the ASM simulator is sent to the ECU of the car. The data signals that are retrieved from ASM for this purpose are the relative distance between the ego vehicle and the object, relative velocity, width and height of the object. These data signals along with some preset signals are sent to the car. From the figures it can be observed that there are several spikes of requests this is because the simulator sends similar data for several samples. Eliminating the spikes from the signal while retrieving the data from the ASM in real time provides accurate signal representation.

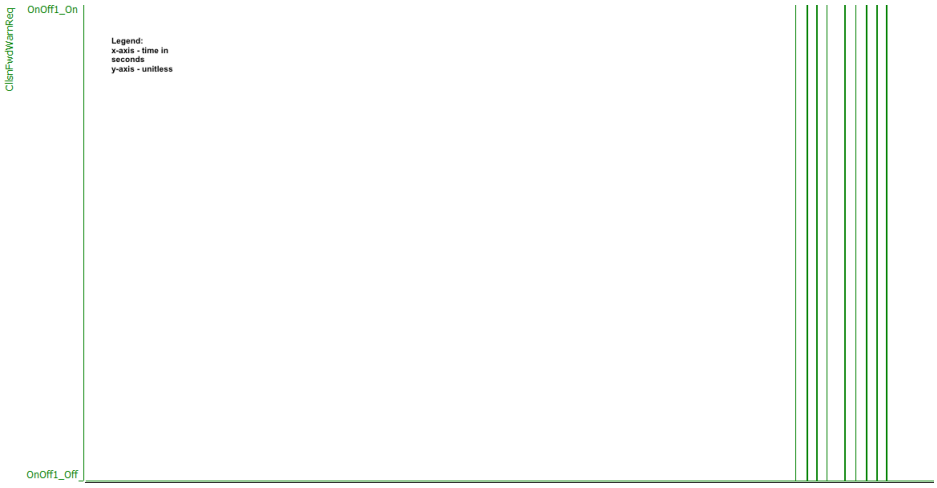

Figure 4.8: This graph indicates a Collision forward warning request signal that gets activated when an object is placed in front of it. It goes high for an object in front of it and goes low when it disappears.

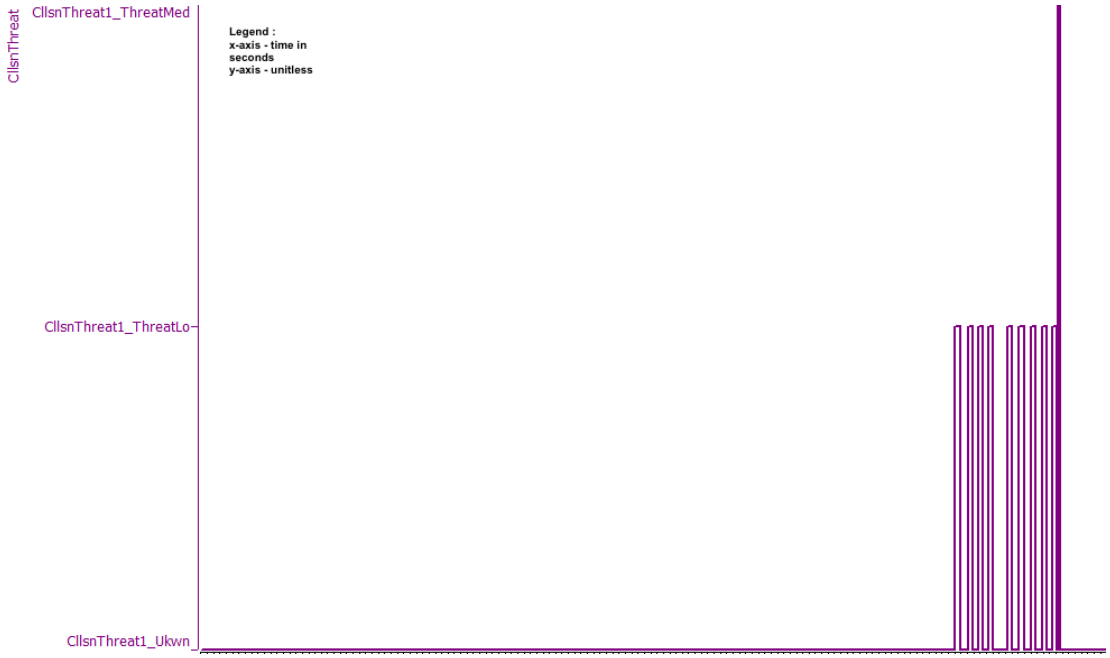

Figure 4.9: This graph indicates a Collision forward warning request signal that gets activated when an object is placed in front of it. It goes high for an object in front of it and goes low when it disappears.

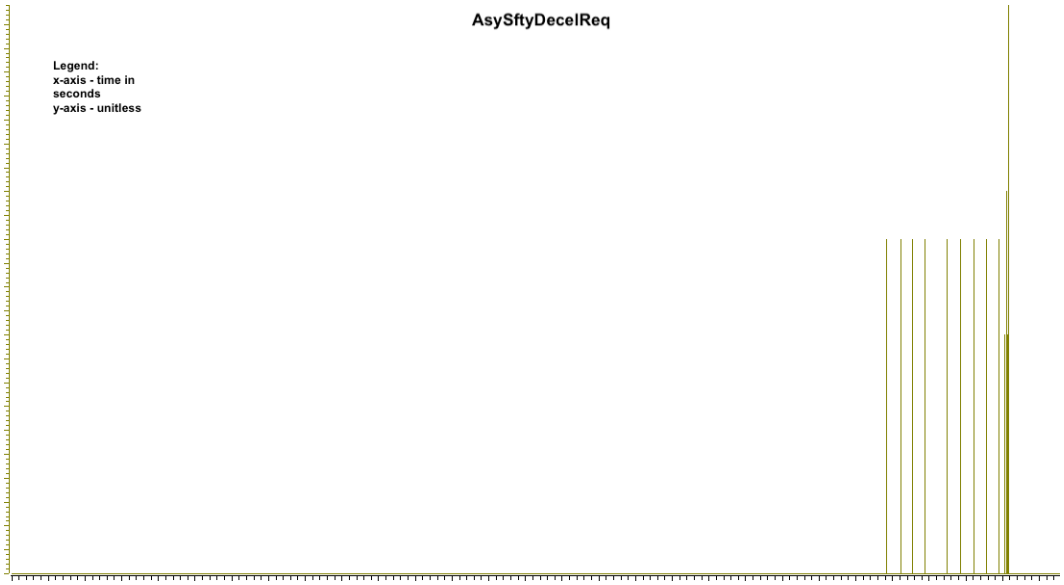

Figure 4.10: This graph indicates a Collision forward warning request signal that gets activated when an object is placed in front of it. It goes high for an object in front of it and goes low when it disappears.

5
Conclusion

Thus, the main goal of this master thesis work is to build a test platform to verify and validate the capabilities of the active safety and the autonomous driving functions by injecting fused radar and camera information of the virtual obstacles, directly into the active safety ECU of the real test car, by bypassing the sensor fusion part. This has been achieved and tested with a set of potential high risk scenarios in both simulation and real physical environment which are shown in the Evaluation section.

The scientific research questions such as the comparative accuracy of the performance to the simulation environment and the real physical world doesn't have much difference in their performances but it becomes slow sometimes due to the processing speed of the computer in which the simulator is running and the validation of the potential and challenging high risk scenarios are also done by testing with different scenarios like stationary/moving car on the road, lane change with cars before the test vehicle, pedestrian crossing the road and Adaptive Cruise Control (following the car before with the same speed). Further improvements in this master thesis work and the future works that can be made are explained in the next section.

6
Future Work

Though this project satisfies the required working of a vehicle-in-the-loop methodology, the data sent from the ASM in this project isn't entirely accurate, a real time working of the simulator to send the required data has to be implemented. The real time implementation also leads to the possibility of allowing the synchronization of the position of the car with that of the car in the simulation environment.

Another expansion to the existing project is to test the methodology with various other scenarios apart from the ones mentioned in the current implementation. These scenarios can help in enhancing the accuracy of testing the active safety functions. These scenarios are:

- **Highway Ramp Merge:** Autonomous vehicle merges from a highway ramp, there can be real and virtual targets on the lane to be merged, to make the scenario complex. The Autonomous vehicle should adapt its speed and change its maneuver accordingly. This can be observed from figure 6.1.

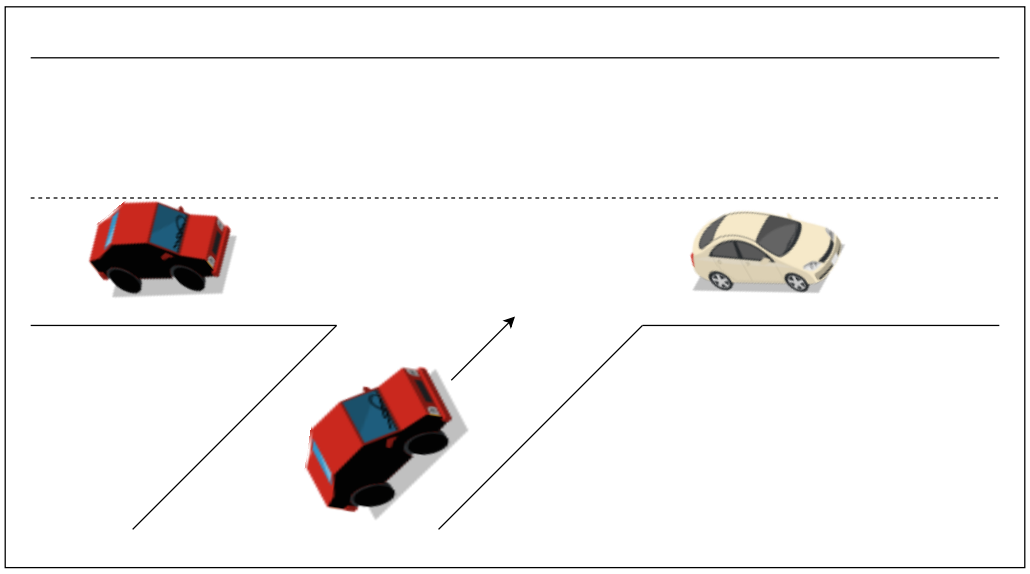

Figure 6.1: Highway Ramp Merge

- **Collision Avoidance By Steering:** In this scenario, a vehicle with advanced driver assistance system driven by a driver with Virtual reality headset tries to change lane to avoid the incoming high speed virtual target from behind

and triggers the Active safety function corresponding to the slow real target. This can be observed from figure 6.2.

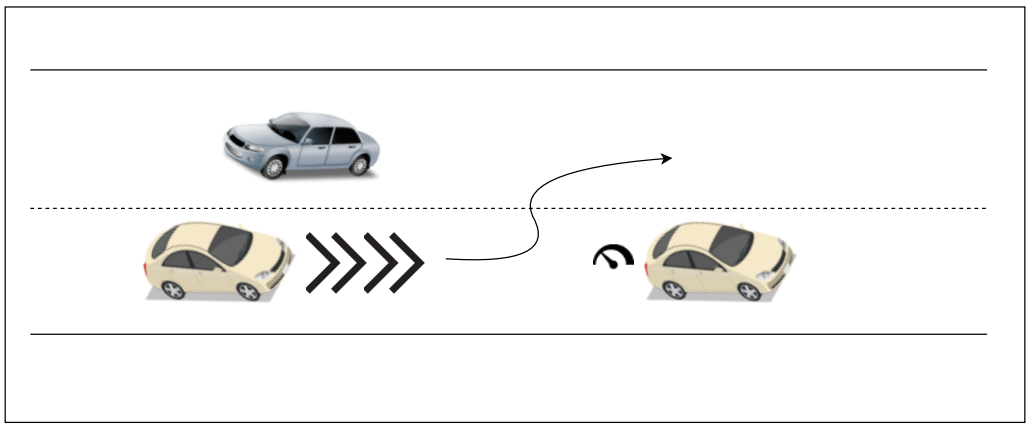

Figure 6.2: Collision Avoidance By Steering

- **Car Crash Ahead:** In this a virtual vehicle in the opposite lane tries to overtake the slow virtual vehicle before and crashes with the oncoming virtual vehicle on the next lane ahead of the autonomous vehicle under test. This can be observed from figure 6.3.

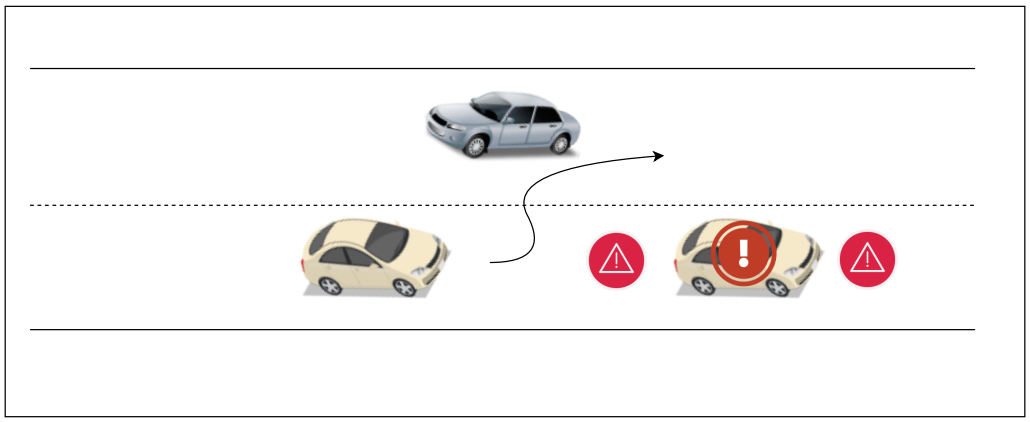

Figure 6.3: Car Crash Ahead

- **Oncoming Vehicle In Wrong Direction:** In this scenario an Autonomous car is driving on a empty road and the virtual or real target vehicle in the next lane drifts into the vehicle under test's lane and the autonomous car should decide what to do to avoid collision. This can be observed from figure 6.4.

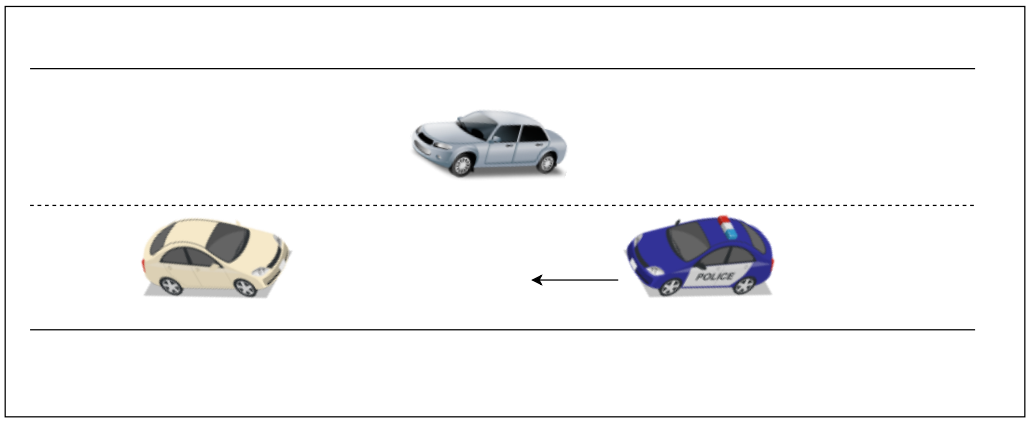

Figure 6.4: Oncoming Vehicle In Wrong Direction

- **Cross Roads Junction:** In this scenario an Autonomous test vehicle should turn left at the intersection of the cross roads at low speed and multiple targets like pedestrians and other target vehicles are crossing at the same time. This scenario helps in testing the behavior of the vehicle under test from the sensor signals and logic. This can be observed from figure 6.5.

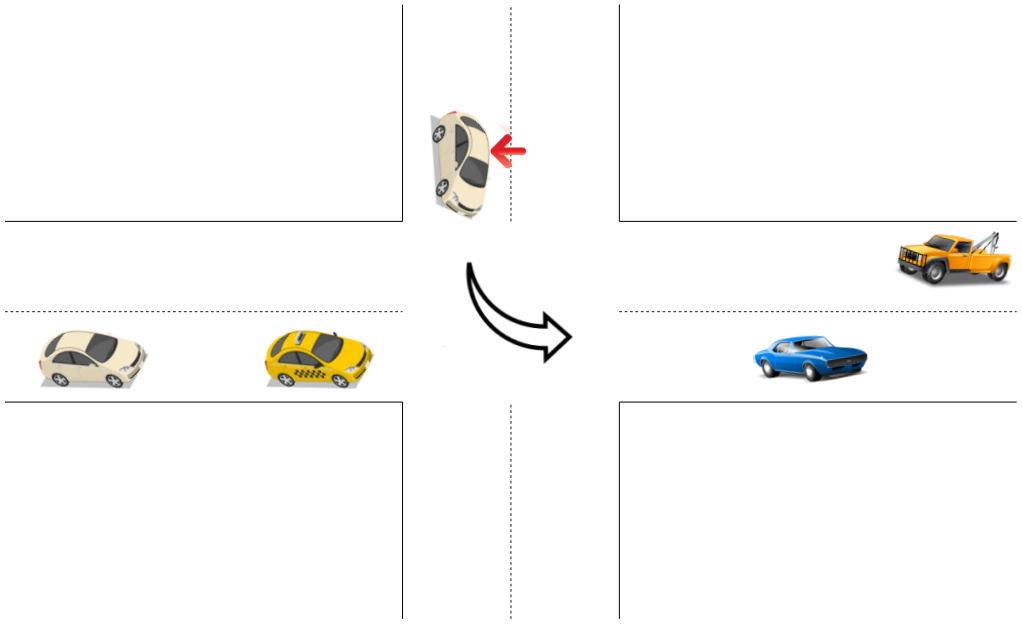

Figure 6.5: Cross Roads Junction

- **Swarm Traffic:** Autonomous car driving in a crowded highway with real and virtual targets. This scenario can be used to test lane change and platooning. This can be observed from figure 6.6.

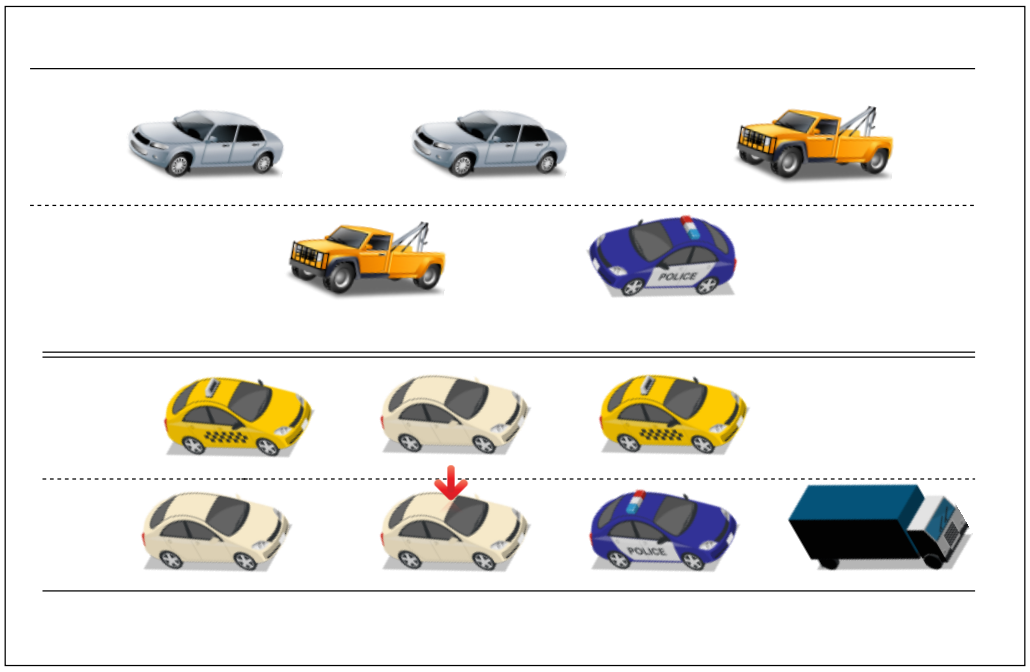

Figure 6.6: Cross Roads Junction

The further scope of the project could be to render the simulated environment in the traffic simulator and visualize it in the VR headset with the driver sitting in the vehicle and driving it. This helps for the driver to understand why the car is braking at certain instance even when the object is not exactly in the field of view of the car. Another possibility of expansion could be to try the current implementation methodology using different ECU's.

Currently, the virtual object from the simulator is being sent from a desktop simulator, this can be further modified by implementing wireless injection of the objects by placing the simulator in the server.

By achieving the above suggested changes, the project could significantly contribute to improvise the exiting technologies.

A

Appendix

The main aspect to be considered while sending the required information to the brain (ECU) of a car is to establish proper communication. This is crucial to ensure that no data is lost while transmission. There are several methods to communicate with the ECU of the car, some of the protocols that can be used are Diagnostic Protocol Data Unit Application Programming Interface (D-PDU API), XCP- Measurement and Calibration protocol.

A.1 D-PDU API

Communication access to the Vector network interfaces is established using this API. Vector network refers to a hardware network interface which provides the communication via the hardware CAN channels and also converts the data to required CAN message structure which is used to communicate with the ECU.

To establish the communication with the ECU, firstly a Open Diagnostic data eXchange (ODX) scheme is developed to understand the description of the ECU. This scheme supports the structured files, which are used for the ECU diagnosis and validation. The D-PDU API application accesses a MVCI (Modular vehicle communication interface) D-server. The D-server gets all the information about the ECU using the ODX scheme. Then, the application request for the ECU is converted into a byte stream by the D-Server, this is called as a diagnostic protocol data unit(D-PDU). Modular vehicle communication interface protocol module then transmits this D-PDU to the ECU of the vehicle. The MVCI module also receives the response from the ECU of the car and sends it back to the D-Server. This two way communication helps in understanding and transmitting necessary data streams. The picture A.1 depicts the working of the D-PDU API.

In order to use the D-PDU API protocol in the project, firstly, a client and server applications are setup that communicate via TCP/IP communication protocol. Once the client and server communicate with each other, data is sent out from the client to the server. Data used here is a structure of 8 information fields containing details about the object type, acceleration, long distance, latitude distance, vehicle speed and other such fields. Server receives this data and sends it to the ECU. Data to be sent is constructed as a fused RaCam object message and is sent over CAN bus.

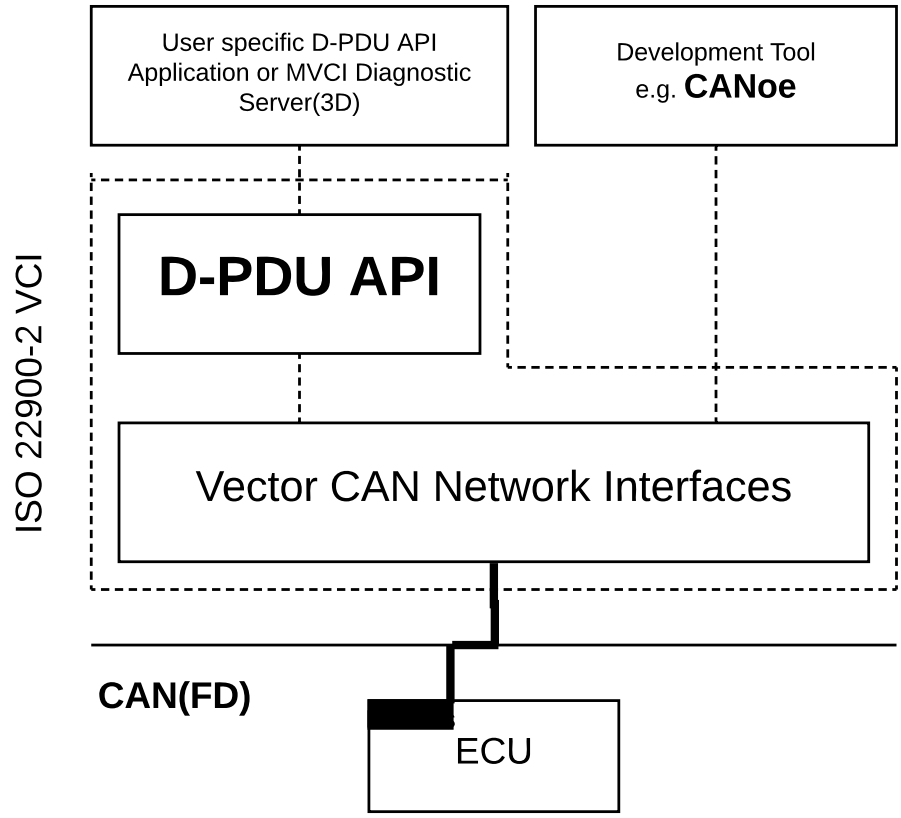

Figure A.1: Sample Application architecture using D-PDU API. This architecture represents the flow of the data from the user specific application to the ECU by opening the Server to convert the data into streams [1].

A.2 XCP Protocol

This is the universal measurement and calibration protocol used in the automotive industry for connecting calibration systems to the ECUs of the car and testing it. This protocol originated from ASAM (Association for Standardization of Automation and Measuring Systems) [16]. This calibration tool requires very less memory and short execution time based on the scaling of software components. It consists of different transport layer for CAN, FlexRay and Ethernet. The CANoe or CANape is used as the master where the ECU is the slave. The third party producer interface can also be done with small modifications to the XCP transportation layer. Once the master slave connection is configured the XCP slave (ECU) provides the XCP master (CANoe/CANape) with the access to measure signals and calibrate the parameters in the ECU with the help of an ECU description file in ASAM format (A2L).

This protocol has different functionalities such as:

- Generation of A2L file based on the configuration of the ECU.
- Entire datasets can be acquired or stimulated synchronous to events triggered

by timers or operating conditions.

- It enables read and write access to the variables and memory addresses of the ECU during development, testing and in vehicle calibration using the created A2L based on the ECU configuration.
- Initialization and switch-over of the memory area of the ECU for calibration data.
- Support of time stamps.
- Protection against unauthorized writing and reading in the ECU's memory.
- This protocol also supports programming of flash memory and EEPROM.
- It provides Block transfer communication mode.
- Transmission of service request packets.

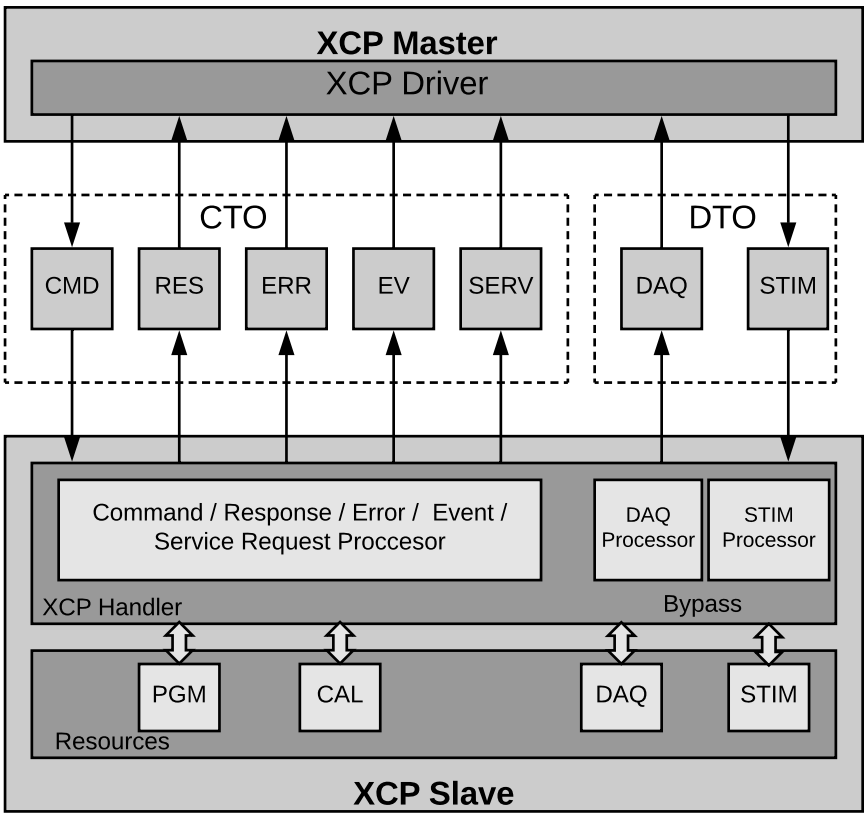

Figure A.2: The figure represents the communication interface between XCP Master (CANoe/CANape)and the XCP Slave (ECU)over a integrated XCP Driver [2]and the distinction between the Command Transfer Object (CTO) and the Data Transfer Object (DTO) through various layers.

A.3 XL driver Library

This API enables the development of own applications for CAN, CAN FD, LIN, MOST, Ethernet, FlexRay, digital/analog input/output (DAIO) or ARINC on supported Vector devices. This tools makes these applications independent of hardware and operating systems. It provides bus specific methods that can be easily used to operate bus interfaces from Vector, it gives access to channels and ports. The network node is configured to send and receive messages using the bus specific methods. More than one channel can also be used for each bus system.

XL driver uses certain principles to establish the bus interface. The main principles that are followed by the XL driver library are:

- Initializing the port of the driver to the required channels for a specified bus type.
- Configuring the initialized port and the respective channels.
- Defining tasks to transmit and receive messages.

Using the above mentioned principles as key aspects, XL driver library provides functions to perform various actions such a reading or writing hardware settings. An example of the application interface can be viewed in the figure A.3.

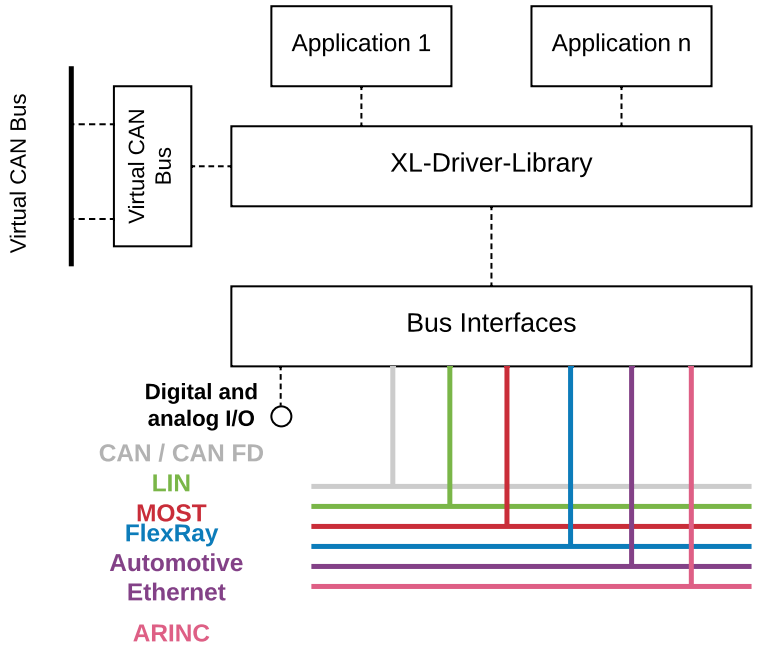

Figure A.3: The figure represents a sample application architecture using XL driver Library [3]. In this diagram an external application can communicate with the ECU of the car via a bus interface communication provided by the XL driver library.

A.4 CAN Communication

CAN communication is a message-based protocol which facilitates carrier-sense, multiple-access. It was originally designed for multiplex electrical wiring in automotive industry, it replaces complex wiring with a two-wire bus. This protocol also follows collision detection and arbitration on message priority. It means that on occurrence of collisions the priority of the message to be sent is resolved by the bit-wise arbitration, where the priority is decided by the identifier field of the message being sent.

The protocol used in the project is in accordance with the ISO 11898 (International Organization for Standardization). The specification of the standard contains 11 bit identifier which gives a signalling rate from 125 kbps to 1 Mbps. This is referred to as Standard CAN. Standard CAN protocol is used in the conversion of the data from the ASM simulator.

SOF	11-bit Identifier	RTR	IDE	r0	DLC	0...8 Bytes Data	CRC	ACK	EOF	IFS

Table A.1: Representation of Standard CAN 11-bit identifier.

The complete description of figure A.1 is as follows:

- SOF is the Start of the Frame bit, indicates the starting of a message. It is also used to synchronize the nodes of a bus.

- Identifier - The priority of the message is established using this 11-bit identifier.

- RTR is the single remote transmission request bit. This bit dominates when the information is required from another node. The request is received by all the nodes, but the required node is determined by the identifier.

- IDE represents the Identifier extension bit, a dominant bit in this field represents that a CAN identifier with no extension is being transmitted.

- r0 stands for reserved bit.

- DLC is the data length code, it is 4 bit long and indicates the number of bytes of data being transmitted.

- Data represents the field where 64 bits of application data can be transmitted.

- CRC is the cyclic redundancy check bit, this 16 bit contains the check sum of the previous application data. It is used for error detection.

- ACK, the integrity of data on each node is acknowledged using this field. A dominant bit on this field indicates that an error free message is sent.

- EOF is a 7-bit field which indicates the End of the frame. Furthermore it disables the bit stuffing. A dominant value on this field indicates a stuffing error.

- IFS is a 7-bit interframe space. Indicates the time required by the controller to move a received frame to its position in message buffer.

Bibliography

[1] Vector Informatik GmbH. D-pdu api. [Online].

[2] Vector Informatik GmbH. Xcp measurement and calibration protocol-fundamentals. [Online; Last modified: 2017-03-21].

[3] Vector Informatik GmbH. Xl-driver-library. [Online].

[4] D. J. Verburg, A. C. M. van der Knaap, and J. Ploeg. Vehil: developing and testing intelligent vehicles. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 537–544 vol.2, June 2002.

[5] Romain Rossi, Clement Galko, Hariharan Narasimman ´, and Xavier Savatier. Vehicle hardware-in-the-loop system for adas virtual testing. pages 251–267.

[6] P. Setlur, J. Wagner, D. Dawson, and L. Powers. A hardware-in-the-loop and virtual reality test environment for steer-by-wire system evaluations. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 3, pages 2584–2589 vol.3, June 2003.

[7] Add2. Software-in-the-loop testing applications. [Online - 2018].

[8] Andreas Krämer Eckard Bringmann. Model-based testing of automotive systems. *2008 International Conference on Software Testing, Verification, and Validation*, pages 485–493, 2008.

[9] S. Demers, P. Gopalakrishnan, and L. Kant. A generic solution to software-in-the-loop. In *MILCOM 2007 - IEEE Military Communications Conference*, pages 1–6, Oct 2007.

[10] Add2. Hardware-in-the-loop testing applications. [Online - 2018].

[11] B. A. Krishnan and A. S. Pillai. Digital sensor simulation frame work for hardware-in-the-loop testing. In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 813–817, July 2017.

[12] F. C. Nemtanu, I. M. Costea, D. Buretea, and L. G. Obreja. Hardware in the loop simulation platform for intelligent transport systems. In *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 247–250, Oct 2017.

[13] K. Fang, Y. Zhou, P. Ma, and M. Yang. Credibility evaluation of hardware-in-the-loop simulation systems. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 3794–3799, June 2018.

[14] dSPACE GmbH. Modular simulator concept with off-the-shelf components according to your needs. [Online - 2018].

[15] T. Bokc, M. Maurer, and G. Farber. Validation of the vehicle in the loop (vil); a milestone for the simulation of driver assistance systems. In *2007 IEEE Intelligent Vehicles Symposium*, pages 612–617, June 2007.

[16] Y. Zhang and T. R. Henderson. An implementation and experimental study of the explicit control protocol (xcp). In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1037–1048 vol. 2, March 2005.