Politecnico di Torino
Academic Year 2018/2019

Master's Thesis

Master's Degree in Mechatronic Engineering

Hardware-in-the-Loop Simulation for V2X Use Cases

Supervisor:
Prof. Claudio Ettore Casetti
Supervisor (Magneti Marelli):
Dr. Maria Carmela De Gennaro

Candidate:
Giulia Andriolo
S240135

Hardware-in-the-Loop Simulation for V2X Use Cases

Giulia Andriolo

Supervised by:

Prof. Claudio Ettore Casetti

Politecnico di Torino

Dr. Maria Carmela De Gennaro

Magneti Marelli

Abstract

The main purpose of this thesis work is to create a simulation environment for the testing of the V2X use cases. We are talking about a simulation environment and not only single software because does not exist any self-supporting simulation tool in order to simulate a V2X operations. However it was possible to integrate many of different software, that when operate together allow to analyse benefits of V2X communication.

The Hardware-in-the-Loop simulation is built for the Connectivity Boards prepared by Magneti Marelli for V2X communication through DSRC (Dedicated Short Range Communication). So, the simulation environment aims to support the boards testing on bench in order to avoid the direct in-vehicle driving test that are often expensive, time-consuming and not reproducible too.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context

Nowadays we depend heavily on transport in our everyday lives. Yet ever increasing road traffic generates serious problems in terms of congestion, safety and environmental impact. Fortunately, information and communication technologies offer new advanced solutions to today's transport problems.
Intelligent Transport System (ITS) embrace a wide variety of communication-related applications intended to increase travel safety, minimize environmental impact, improve traffic management and maximize the benefits of transportation to both commercial uses and general public [4].
Modern vehicle equipped with driver assistance can "feel" (e.g. by sensors), can "see" (e.g. by cameras) and in the future thanks to the new technology of Cooperative Intelligent Transport System and Service (C-ITS) vehicles will be able to "speak" among each other (V2V) and with Traffic Infrastructure (V2I). In general these communication processes are called Vehicle-to-Everything or V2X.
The technology behind V2V communication allows vehicles to broadcast and receive omni-directional messages creating a 360-degrees "awareness" of other vehicles in proximity within a range about 300 meters. Thanks to a wireless Dedicated Short Range Communication, DSRC, each vehicles equipped with appropriate software can use the messages from surrounding vehicles to determine potential crash threats as they develop [8]. The National Highway Traffic Safety Administration of United States estimates that Vehicle-To-Everything "are capable of eliminating 94 percent of fatal crashes involving human error"[8].
The communication networks that allows information transfer among the vehicles as well as vehicles and infrastructure is called VANETs (Vehicular Ad-hoc NETwork). The buildings block of this communication network are the On Board Units (OBU), a radio interfaces inside the car that can access the network created by other OBUs or Road Side Units (RSU), fixed access point along the road such as, for example, intelligent traffic lights.

Currently VANETs rely on dedicated short range communication technologies, such as, IEEE 802.11p, but other access method are developing by research institutions, universities and automobiles manufactures. The most promising path for new communication technologies is represented by 5G, which would be a more reliable and efficient technology for the realization of Cellular V2X. C-V2X is the evolution of V2X based on 802.11p and exploits the cellular network to extend the range of awareness of road entities up to a mile[3].

1.2 The Company

Magneti Marelli is an international company founded in Italy in 1919, committed to the design and production of hi-tech systems and components for the automotive sector in the following business areas: Electronic System, Automotive Lighting, Powertrain, Suspension System, Exhaust Systems, Motorsport, Plastic Components and Modules and After Market Parts and Services[6].
The plant in Venaria Reale, Italy, also has several groups that work in the innovation field "Technology Innovation", one of them is the so called "Innovation Connectivity" group that focuses its attention of all the aspects concerning connectivity. In particular the team is focused on the V2X DSRC technology and its implementation with some of the most popular use cases like for example the Control Loss Warning (CLW), Emergency Electronic Brake Light (EEBL), Forward Collision Avoidance (FCW), Left Turn Assist (LTA), Intersection Movement Assist (IMA), Stationary Vehicle Warning (SV) for what concern the V2V communication; while the Green Light Optimized Speed Advise (GLOSA) and In Vehicle Road Sign (IVRS) for what concern the V2I communication.
Moreover the Innovation Connectivity team is also investigating the new connectivity technologies like the C-V2X in order to be prepared with the launching of the fifth generation of connectivity 5G.

1.3 Goal of the thesis

As I said above, implementation of Vehicle-to-everything (V2X) communication technologies, for traffic management, has been envisioned to have a plethora of far-reaching and useful consequences[1].
With this background the goal of this thesis is the development of a simulation environment that allows to test the specific V2X use case, listed above, without the imperative to go inside the car. The "Innovation & Connectivity" team is equipped with two cars for the tests, whose Electronic Control Units have been conveniently modified to integrate with the V2X Communication board made by Magneti Marelli. Until now the procedure to test a specific use case consists to drive the two cars in order to recreate the traffic scenario of interest and records the specific results in the

log files. Naturally this procedure is not the best one from the repeatability point of view and represents also a waste of time energy and money too.
So my job was to find out the way to simplify this procedure in order to create a bench test that allows to go inside the car as the last step of the chain.
I'm talking about a simulation environment and not only single software because does not exist any self-supporting simulation tool in order to simulate a V2X operations. However it is possible to integrate many of different software, that when operate together allow to analyse the effects of V2X technologies we want to achieve.
So in the next chapters will be described the technology used and the specific role covered by every single software. It will be also described the interaction between these software and the relative created chain. Moreover, attaching the V2X communication board, it has been possible to do Hardware-in-the-Loop for the V2X use cases implemented by Magneti Marelli.

Chapter 2

Technologies Involved

In the following chapter are detailed the most relevant features of the software that have been used in order to implement the simulation environment and a brief technical description of the Magneti Marelli DSRC connectivity boards that have been used in order to perform the Hardware-in-The Loop simulation.

2.1 SUMO

"Simulation of Urban Mobility", or "SUMO" for short, is an open source, microscopic, multi-modal traffic simulation. It allows to simulate how a given traffic demand which consists of single vehicles moves through a given road network. The simulation allows to address a large set of traffic management topics. It is purely microscopic in the sense that each vehicle is modelled explicit, has an own route, and moves individually through the network [11].
The German Aerospace Center (DLR) started the development of the open source traffic simulation package SUMO back in 2001. Since then SUMO has evolved into a full features suite of traffic modelling utilities including a road network capable to read different source formats demand generation and routing utilities from various input source, a high performance simulation usable for single junctions as well as whole cities including a "remote control" interface (TraCI) to adapt the simulation online [11].

The Traffic Simulation can be realized in two different ways as shows the diagram, through command line or through the SUMO-GUI in order to have a graphic feedback.

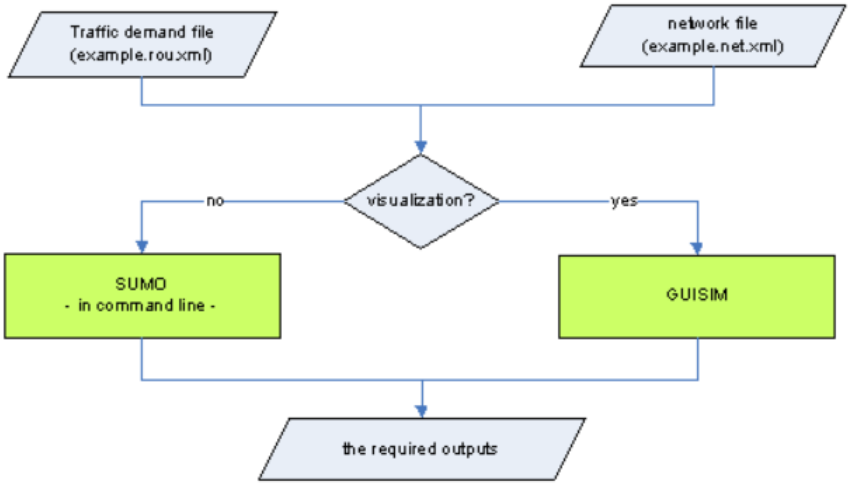

Figure 2.1: Traffic simulation process

A SUMO network file describes the traffic-related part of a map, the roads and intersections the simulated vehicles run along or across. At a coarse scale, a SUMO network is a directed graph. Nodes, usually named "junctions" in SUMO-context, represent intersections, and "edges" roads or streets. Specifically, the SUMO network contains the following information:

- Every street (edge) as a collection of lanes, including the position, shape and speed limit of every lane.
- Traffic light logics referenced by junctions.
- Junctions, including their right of way regulation.
- Connections between lanes at junctions (nodes).

In order to create the relative .net.xml file it is possible to use SUMO XML description files, together with NETCONVERT (command that allows to import digital road networks from different sources and generates road networks that can be used by other tools from the package). It is also possible convert an existing map from various formats using NETCONVERT or generate geometrically simple, abstract road maps with NETGENERATE (command that generates abstract road network that may be used by other SUMO-applications) or NETEDIT (is a graphical network editor for SUMO) for building own road networks or for reworking the ones obtained from NETCOVERT or NETGENERATE [11].

After having generate a network, it is possible take a look at it using SUMO-GUI, but no cars would be driving around. One still needs some kind of description

about the vehicle. This is called the traffic demand. In this context there is a very important distinction that has to be done: a trip is a vehicle movement from one place to another defined by the starting edge (street), the destination edge and the departure time. A route is an expanded trip, that means, that a route definition contains not only the first and the last edge but all edge the vehicle will pass. SUMO and SUMO-GUI need routes as input for vehicle movements [11].
There are several ways to generates routes for SUMO. The choice depends on your available input data:

- Using trip definitions: as described above, each trip consists at least of the starting and the ending edge and the departure time. This is useful for when you want to create demand by hand or when writing your own scripts to import custom data. It is possible to use DUAROUTER (command that imports different demand definitions, computes vehicle routes that may be used by SUMO using shortest path computation) to turn your trips into routes, or it can be loaded the trips directly into SUMO.

- Using flow definitions: this is mostly the same approach as using trip definitions, but one may join vehicles having the same and arrival edge using this method.

- Using Randomization: this is a quick way to get some traffic if you do not have access to any measurements but the results are highly unrealistic.

- Using OD-matrices: origin-Destination-Matrices (or OD matrices) are often available from traffic authorities.

- Flow definitions and turning ratios: one may also leave out the destination edges for flows and use flows and use turning ratios at junctions instead.

- Using detector data (observation points): induction loops and similar devices are commonly used by authorities to measure traffic.

- By hand: you can of course generate route XML-files by hand.

- Using populations statics: the program ACTIVITYGEN (reads the definition of a population matching an also given network. It computes and mobility wishes for this population) can be used to turn population statics into traffic demand.

- Using data from other sources.

2.1.1 TraCI

TraCI is the short term for "Traffic Control Interface". Giving access to a running road traffic simulation, it allows to retrieve of simulated object and to manipulate their behavior "on-line".
Traci uses a TCP based client/server architecture to provide access to SUMO. Thereby, SUMO acts as server that is started with additional command line option: –remote-port <INT> where <INT> is the port SUMO will listen on for incoming connections [11].
After starting SUMO, clients connect to SUMO by setting up a TCP connection to the appointed SUMO port.

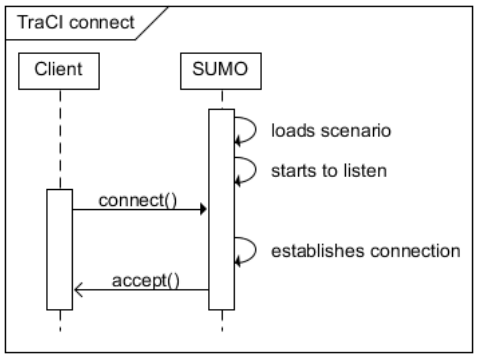

Figure 2.2: TraCI: establishing a connection to SUMO

The client application sends commands to SUMO to control the simulation run, to influence single vehicle's behavior or to ask environmental details. The client is also responsible for shutting down the connection using the close command.

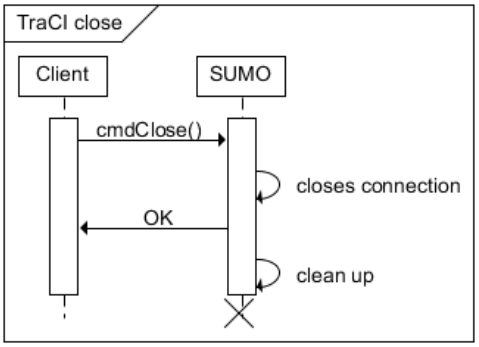

Figure 2.3: TraCI: closing a connection to SUMO

2.1.2 TraCI4Matlab

TraCI4Matlab is an API (Application Programming Interface) developed in Matlab that allows the communication between any application developed in this language and the urban traffic simulator SUMO.
The functions that comprise TraCI4Matlab implement the TraCI application level protocol, which is built on top of the TCP/IP stack, so the application developed in Matlab, which is the client, can access and modify the simulation environment provided by the server (SUMO).
In general SUMO objects are grouped in thirteen domains: gui, lane, poi, simulation, traffic lights, vehicletype, edge, iductionloop, junction, multientryexit, polygon, route and vehicle.
Of course the attributes of the SUMO's simulation objects can be accessed and modified with this command: traci.<domain>.<get/set_wrapper()> where domain can take any of the domains listed previously and get/set_wrapper() are the functions to access the value (get) or modify (set) the attributes of the object of interest [12].

2.2 Matlab&Simulink

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python. Simulink is a block diagram environment for multidomain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis[7].

2.3 CANoe

CANoe is the comprehensive software tool for development, test and analysis of individual ECUs and entire ECU networks. It supports network designers, development and test engineers throughout the entire development process from planning to system level test [13] .
CANoe users can analyse the multi-bus communication of ECUs and entire systems at their desk as well as in the vehicle. These smart windows can support the analysis:

- Trace Window: for listing all bus activities such as messages or error frames.

For each messages there is the possibility of displaying the individual signal values.

- Graphical Window: for graphical online display for values transmitted in messages and diagnostic requests, such as rpm or temperature values, over a time axis.
- Statistics Window: for displaying useful network and node statistics, e.g. bus load on node and frame levels, burst counter/duration, counter/rate for frames and errors, controller states.
- Data Window: for displaying preselected data, e.g. numeric or bar graph data.
- State Tracker: for displaying states and bit signals.

Moreover further analysis windows and blocks are available like:

- Measurement Setup: for graphical display and parametrization of function blocks and evaluation functions.
- Scope Window: for offline display of bit level measurements record with the option scope.
- Interactive generator: for stimulating the buses and for easy sending of modifies signals.
- Signal generator: for generating signal waveforms (sine, ramp, pulse, value list, etc.).
- Logging/Replay: for logging and later analysis or replay of measurments.
- Write Window: for system messages and user-specific outputs from CAPL programs.

A simulation can be generated manually or automatically from the underlying communication database. This remaing bus simulation of communication behavior of complete networks or individual ECUs is the basis for the subsequent analysis and testing phases. The developer may implement different test scenario along the development process:

- SIL (Software-in-the-Loop).
- MIL (Model-in-the-Loop) - Integration of Matlab/Simulink models.
- HIL (Hardware-in-the-Loop) - Execute simulation with the real time hardware.

2.3.1 CANoe/Matlab Interface

The purpose of this interface is to extend CANoe's mode modelling capability by adding the strength of the MATLAB/Simulink environment. It allows execution of Simulink models inside the CANoe network simulation environment.
This interface consists of a block set for MATLAB/Simulink and the Simulink coder. It provides data exchange with CANoe for simulations running inside Simulink and assures time sinchronization between both tools [10].
Data exchange between CANoe and MATLAB/Simulink is provided by a library for Simulink. It contains blocks for the following CANoe elements:

- Signals.
- Environment variables.
- System variabes.
- CAPL functions.

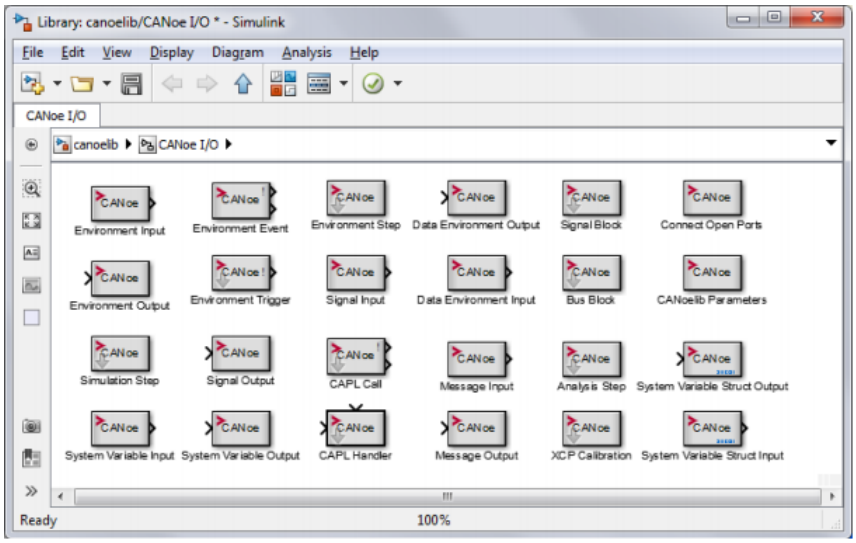

Figure 2.4: CANoe block set library for Simulink

The blocks are used as sources and sinks for Simulink models. They provide the current value of the according CANoe element, the signal input block for example returns the current value of the last CAN message containing the signal. The block set is a very easy way to extend MATLAB/Simulink models with the ability to communicate with real bus devices.

Simulations utilizing the CANoe/MATLAB interface can be operated in three different modes.

Hardware-in-the-Loop (HIL) Mode:
In this mode the simulation is run in the CANoe execution environment. With the Simulink Coder you can target CANoe and create a Windows DLL which can be loaded in CANoe's simulation environment. With this approach it is possible to test and verify a design with real hardware in a networked environment [10].

Offline Mode:
In this mode the simulation is run in the MATLAB/Simulink environment. CANoe is operated in slave mode whereas MATLAB/Simulink is the simulation master. This is the mode for non-real-time simulation. The simulation is controlled from the MATLAB/Simulink environment. No real-time simulation is provided with this kind of simulation. The simulation is run as fast as possible. Therefore no real hardware is needed and development cycles are short [10].

Figure 2.5: Synchronization in offline mode

Synchronized mode:
In this mode the simulation is run in the MATLAB/Simulink environment. In contrast to the Offline Mode the CANoe time base is used in MATLAB/Simulink. Therefore the simulation is executed in almost real-time (typical resolution is about 1ms depending on the model). It is necessary that MATLAB/Simulink can compute the model faster than real-time in order to use this mode. CANoe can either run in simulated mode or in real-time mode providing real hardware access. This mode can be used for interaction with real (CAN) hardware devices [10].

Figure 2.6: Synchronization in synchronized mode

2.4 CAN network

In order to understand how MATLAB and CANoe interact each other it is important to focus on what CAN protocol is and and how the CAN network works.
Controller Area Network (CAN network) is a vehicle bus standard that allows microcontrollers and devices to communicate with each other within a vehicle. CAN is a message-based protocol originally designed for automotive applications, but it is also used in other areas such as industrial automation.

CAN Benefits:

Low-Cost, Lightweight Network
CAN provides an inexpensive, durable network that helps multiple CAN devices communicate with one another. An advantage to this is that electronic control units (ECUs) can have a single CAN interface rather than analog and digital inputs to every device in the system. This decreases overall cost and weight in automobiles.

Broadcast Communication
All devices on the network see all transmitted messages. Each device can decide if a message is relevant or if it should be filtered. This structure allows modifications to CAN networks with minimal impact. Additional non-transmitting nodes can be added without modification to the network.

Priority
Every message has a priority, so if two nodes try to send messages simultaneously, the one with the higher priority gets transmitted and the one with the lower priority gets postponed. This arbitration is non-destructive and results in non-interrupted transmission of the highest priority message. This also allows networks to meet deterministic timing constraints.

Error Capabilities
The CAN specification includes a Cyclic Redundancy Code (CRC) to perform error checking on each frame's contents. Frames with errors are disregarded by all nodes, and an error frame can be transmitted to signal the error to the network. Global and local errors are differentiated by the controller, and if too many errors are detected, individual nodes can stop transmitting errors or disconnect itself from the network completely. [9]
CAN networks are implemented with two wires: CAN-H and CAN-L, terminated with 120ohm resistance and allow communication at transfer rates up to 1Mbit/s.

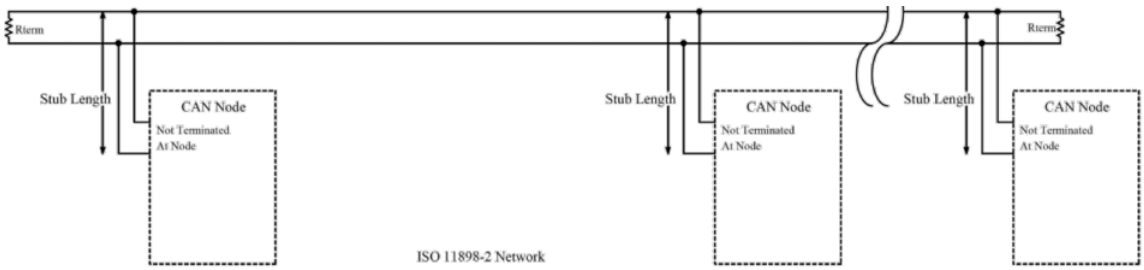

Figure 2.7: Representation of the Physical layer in the CAN protocol

There are different types of CAN message called *frame*:
Data frame: used to send data over the network.
Remote frame: used by a node to request data to another node in the network.
Error frame: sent after a frame is received with wrong format.
Overload frame: used to avoid overloading the CAN network.

Each data frame, as shows the figure has an ID, data and overhead. In general based on the length of the ID there is a distinction for the *Standard frame* and the *Extended frame*:
Standard frame of 44 bits to 108 bits, ID of 11bits, data from 0 to 64 bits.
Extended frame of 62 bits to 126 bits, ID of 29 bits, data from 0 to 64 bits.

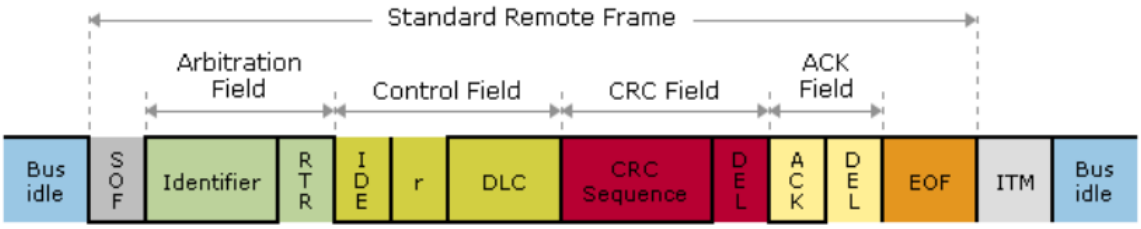

Figure 2.8: Structure of a Data frame

CAN is a peer-to-peer network, this means that there is no master that controls when individual nodes have access to read and write data on the CAN bus. When a CAN node is ready to transmit data, it checks to see if the bus is busy and then simply writes a CAN frame onto the network. The CAN frames that are transmitted do not contain addresses of either the transmitting node or any of the intended receiving node(s). Instead, an arbitration ID that is unique throughout the network labels the frame. All nodes on the CAN network receive the CAN frame, and, depending on the arbitration ID of that transmitted frame, each CAN node on the network decides whether to accept the frame.
If multiple nodes try to transmit a message onto the CAN bus at the same time,

the node with the highest priority (lowest arbitration ID) automatically gets bus access. Lower-priority nodes must wait until the bus becomes available before trying to transmit again. In this way, you can implement CAN networks to ensure deterministic communication among CAN nodes [9].

Using CAN has different meanings depending on where you are in typical V-shape development flow:

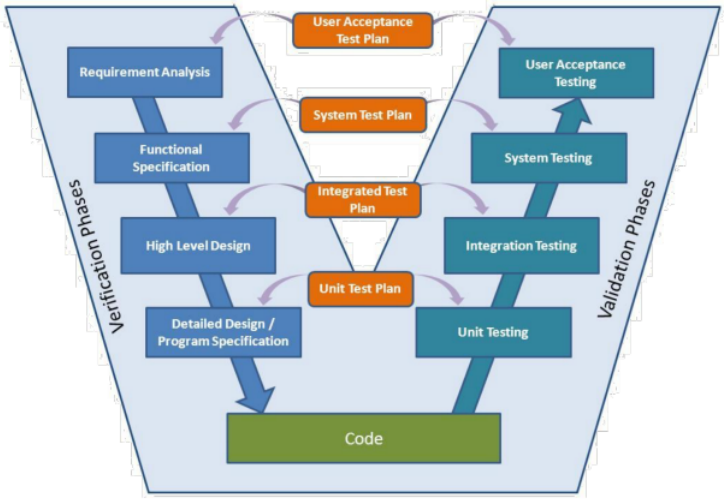

Figure 2.9: V-shape development flow

In a **Network design** it is possible to define the number of CAN nodes, the configuration of CAN common parameters or for example CAN message in terms of ID and payload.
In a **Coding** phase it is possible to define how CAN transmission is managed: message-based or signal-based; and how CAN reception is managed: Polling vs Interrupt, Message-based vs signal-based; unfiltered vs filtered.
While in **Validation** phase it is possible to test if each node to verify it communicates correctly or the network to verify that all nodes communicate correctly.

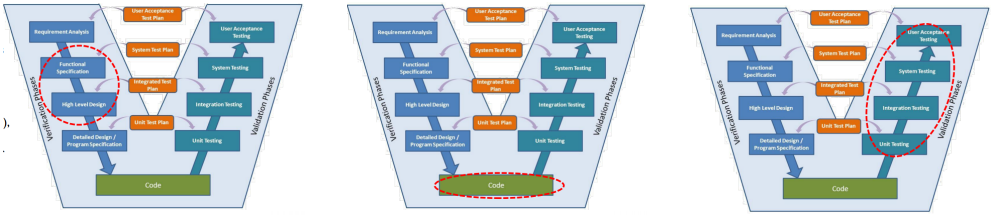

Figure 2.10: representation of different phases on the V-shape development flow

When addressing the CAN network design it is important to specify the number of network nodes, the network topology, the network speed and the **CAN network database**.
CAN database files are text files that contain scaling information for CAN Frames and Signal definitions. Information of a DBC:

- Channel name
- Location (start bit) and size (number of bits) of the channel within a given message
- Byte order
- Data Length
- Data type (signed, unsigned, and IEEE float)
- Scaling and units string
- Range (Minimum value and Maximum value)
- Default value
- Comment

These informations can be used to easily convert the raw frame information (usually bytes) to physical readable data. There are a lot of supporting tools that allow to edit a CAN database like for example *Vector CANdb++* that semplify the definition of signals and messages.

2.5 The MM connectivity framework and hardware

In order to understand how it has been implemented the simulation environment it is important to understand how the framework inside the MM connectivity hardware works. The MM Connectivity Framework has a layered structure, characterised by three main layers: the **middleware**, the **facilities** and the **use cases**. The Framework architecture is reported in figure (2.12).
V2X messages have different specifications and protocols, according to the country in which the technology is deployed. In particular, for the United States and Europe, the defined standards are the following:

- Wireless Access in Vehicular Environments (**WAVE**): is a standard developed by the IEEE (Institute of Electrical and Electronics Engineers), based on the IEEE 802.11p communication standard.

- European Telecommunications Standards Institute Intelligent Transport Systems (**ETSI ITS-G5**):

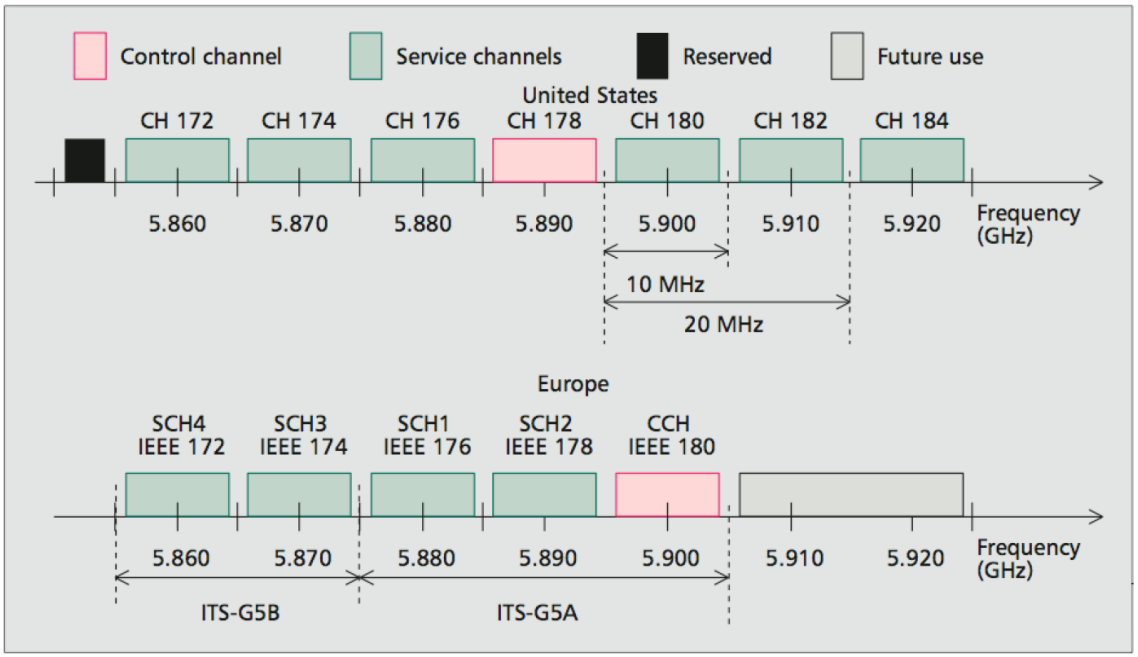

Figure 2.11: The ETSI ITS-G5 and WAVE frequency allocation [5].

2.5.1 Connectivity protocol stacks

This layer provides support to the different connectivity technologies used by V2X applications, such as 802.11p or the various C-V2X flavours. Drivers and libraries for transferring informations to/from the devices used by the Framework are placed in this layer.

2.5.2 Facilities

This layer contains a set of software modules and data definitions (such as messages) that provide the required support to the use cases. Each entity can be **common** to the communication standards ETSI and WAVE, or **specific** to one of those. In the following subsections some of those architectural elements are descripted.

Common facilities

The following facilities are **common** between the ETSI and WAVE standards:

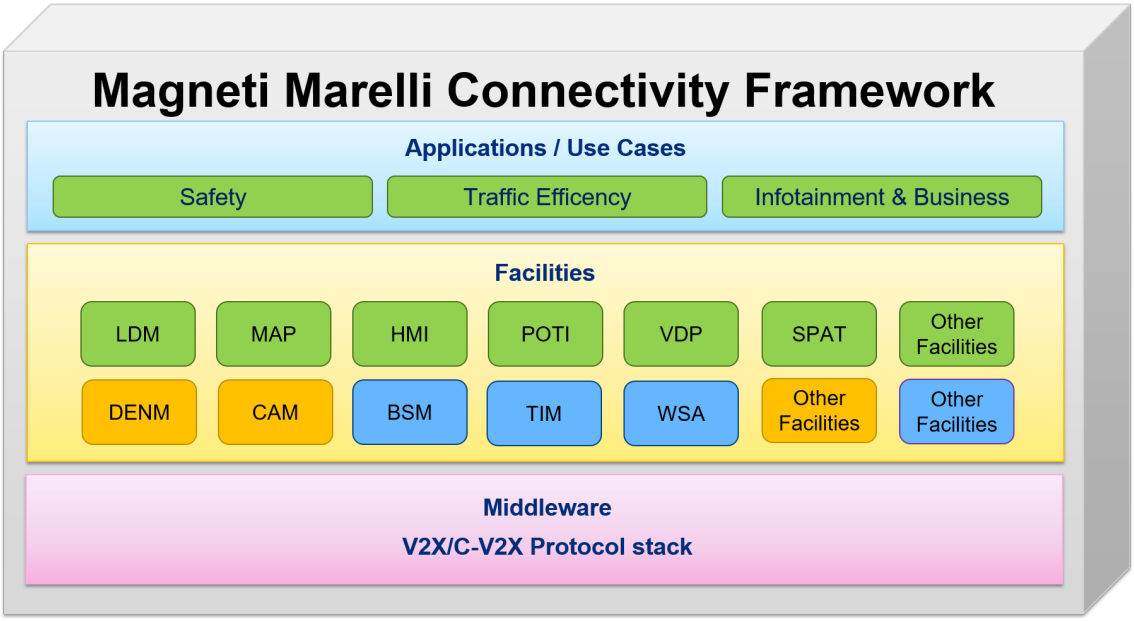

Figure 2.12: The MM V2X Framework architecture. In green the facilities common to ETSI and WAVE standards, in orange the ETSI-specific components, while in blue the WAVE-specific components.

- Local Dynamic Map (**LDM**): this entity works as a database that stores relevant data received from vehicles, infrastructures, etc.
- Topology Message (**MAP**): a component that manages a specific kind of message with the same name. This message contains detailed infos about the current road.
- Human Machine Interface (**HMI**): it has the responsibility of communicating the informations such as warnings, hazards, etc. to the user (for example by showing them on the vehicle instrument panel display).
- POsitioning TIming (**POTI**): this component provides vehicle location informations such as latitude, longitude and altitude.
- Vehicle Data Provider (**VDP**): coupled with the POTI, supplies to the upper layer components other vehicle informations such as its dynamic (coming from the CAN network).

WAVE-specific elements

The following entities are specific to the **WAVE** part of the Framework:

- Basic Safety Message (**BSM**): a component that handles a kind of message that is periodically sent by the vehicles. The BSM message contains informations about the position, the speed, the dynamics, etc. of the vehicle.

- Traveler Information Message (**TIM**): this component manages messages about traffic, street signs, speed limits, etc.

- Wave Service Advertisement (**WSA**): this message type represents the announcement of an ITS.

ETSI-specific elements

The following entities are only present in the **ETSI** part of the Framework:

- Decentralized Enviromental Notification Message (**DENM**): it is a message that is used for warning users of a street hazardous situation.

- Cooperative Aware Message (**CAM**): this message is similar to the BSM (WAVE). It is sent periodically by the ITS.

2.5.3 Magneti Marelli connectivity Hardware

One of the most important component of the developed work it was MM DSRC connectivity board ("Step03" board). The Step 03 board is a Magneti Marelli custom board, developed in Electronic System division. It is based on the NXP-Freescale Smart Application Blueprint for Rapid Engineering (**SABRE**) **i.MX 6QuadPlus** reference design, and it was originally developed for the vehicle infotainment purpose. The board has been modifed for V2C functions and has the following specifications:

- Arm Cortex-A9-based i.MX 6QuadPlus processor

- 1 GB DDR3L RAM

- 512 MB NAND

- 32 GB eMMC

- CAN High Speed & Low Speed

- GPS, Wi-Fi, Bluetooth, 4G modem

- 802.11p transceiver

Figure 2.13: MM DSRC connectivity board

Chapter 3

Implementation

This chapter will introduce the simulation environment that has been developed analyzing the strengths of every single component. In this chapter it will be also described which every single components focus on and at the end the importance of the Hardware-in-the-Loop technique.

3.1 Simulation tools

The three major components, from the different task point of view, that have been led to the birth of the simulation environment are:

- Traffic Simulator
- Application Simulator
- CAN network simulator

3.1.1 Traffic Simulator: SUMO

As I said in the previous chapter, SUMO has been chosen as traffic simulator because it allows to create accurate urban mobility models and investigate real-world traffic models.
In particular for this project in order to represent the real scenario that has been used for the drive test, it has been used a particular Sumo feature that allows to customize your scenario importing a map from OpenStreetMap. This feature is represented by the command **NETCONVERT** that allows to import digital road networks (in this case the Magneti Marelli drive test scenario) and to convert the file derived from OpenStreetMap (.osm) into a Sumo network file (.net.xml).

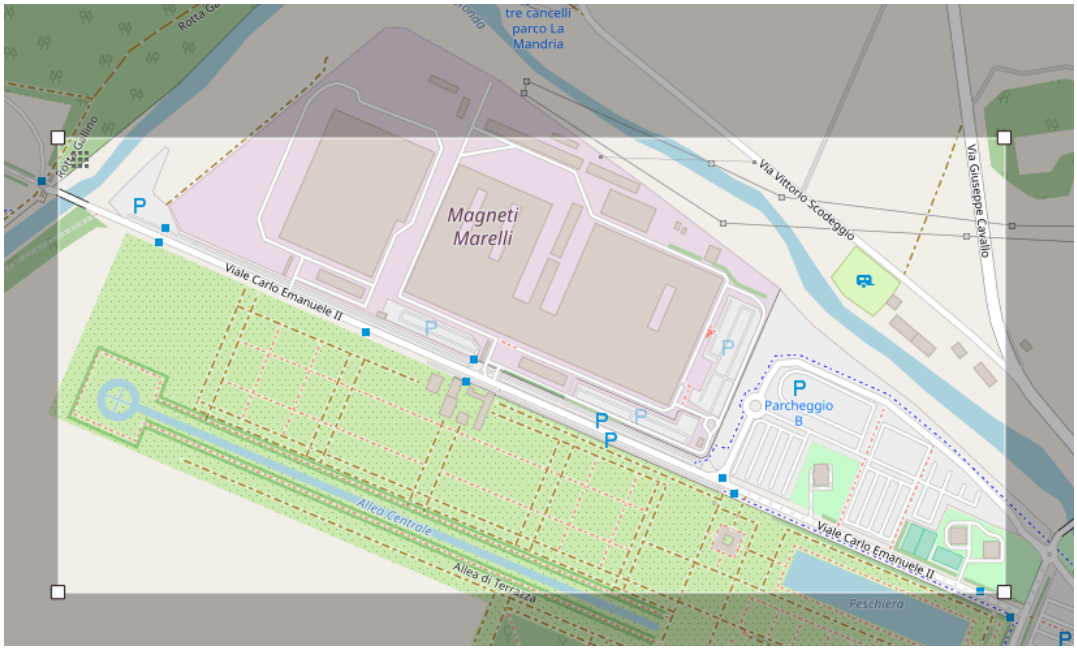

Figure 3.1: MM Venaria plant visualized in OpenStreetMap

```
netconvert -osm-files <file_name>.osm -o <file_name>.net.xml
-output.street-names true -output.original-names true
```

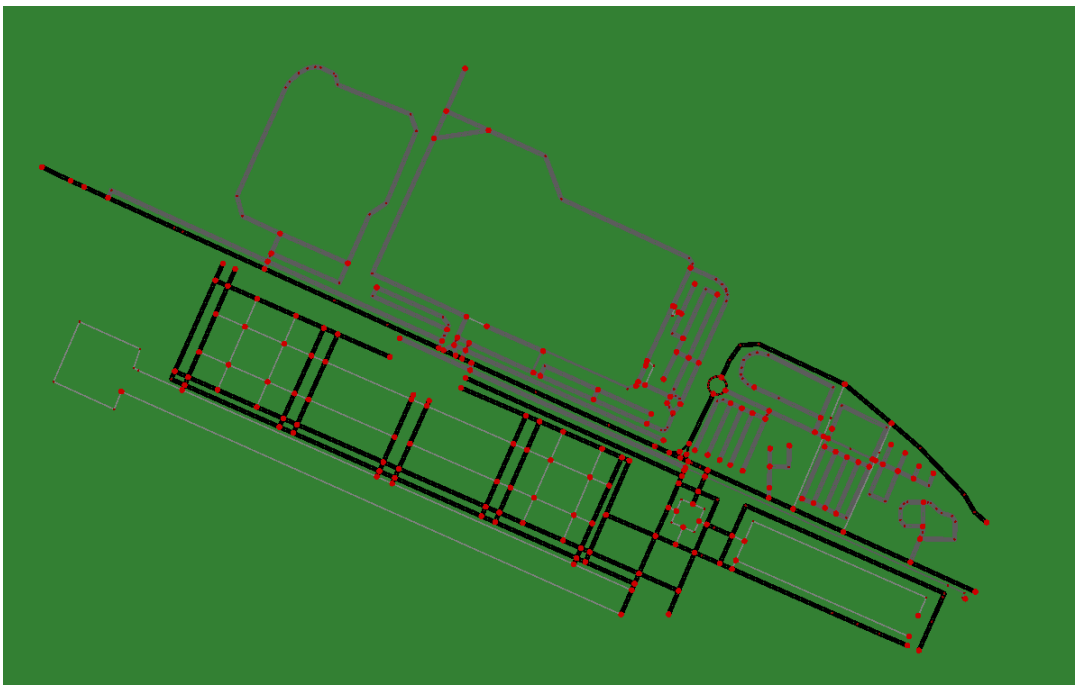

Figure 3.2: MM Venaria plant converted into Sumo network

Once created the Traffic scenario, it has been created the Traffic demand with .rou.xml file created by hand that consists in two different vehicles that for the V2X application point of view represent respectively the *Ego vehicle* (the V2X event "generator") and the *Host vehicle* (the vehicle that receives the event and shows the alert in the dashboard), that moves in the same direction, with same vehicle dynamic (uniformly accelerated motion) but different acceleration.

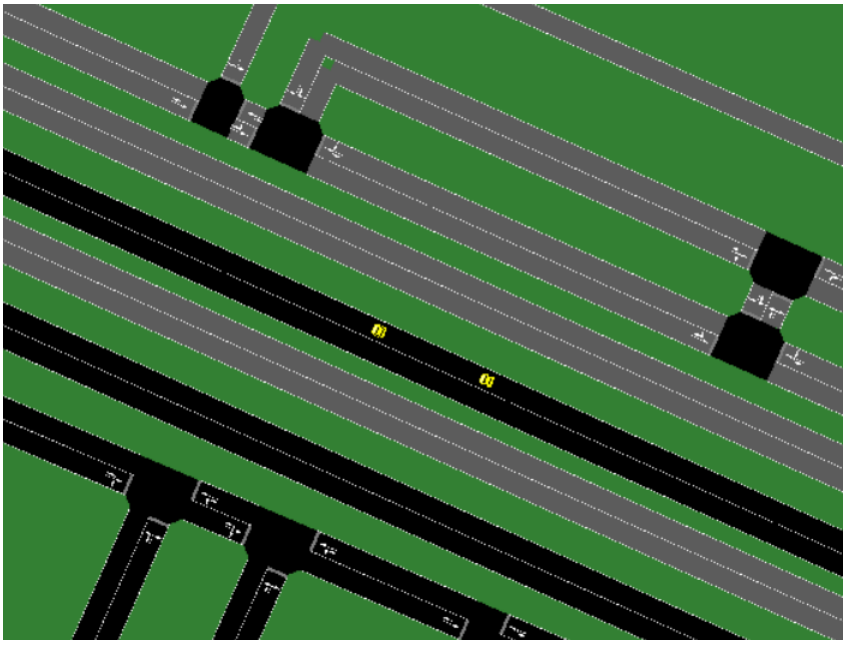

Figure 3.3: Traffic scenario and Traffic demand in SUMO

An important role it has been covered by the "TraCI4Matlab" application program interface that allows to access to the SUMO simulation while running.
TraCi4Matlab has been the first step for the development of the entire simulation environment because it allows to can interface two different software like SUMO and Matlab. In this way if SUMO can provide the data we needed, Matlab can elaborate these data in such a way to manipulate them according to the desired preferences.

3.1.2 Application Simulator: Matlab&Simulink

The reason why it has been chosen Matlab as Application Simulator is because it is one of the most versatile software for the application point of view and it presents a great variety of API and toolbox that extend its capability too.
In particular the two interfaces that allowed to the realization of the entire simulation environment were **TraCI4Matlab** in order to can get the data from SUMO during the Simulation and the **CANoe/Matlab Interface** to can deliver these data to the real hardware and can stimulate the framework inside it through the CAN interface.
It has been realized a Simulink model that allowed the communication between the Traffic Simulator SUMO and the CAN Simulator CANoe:

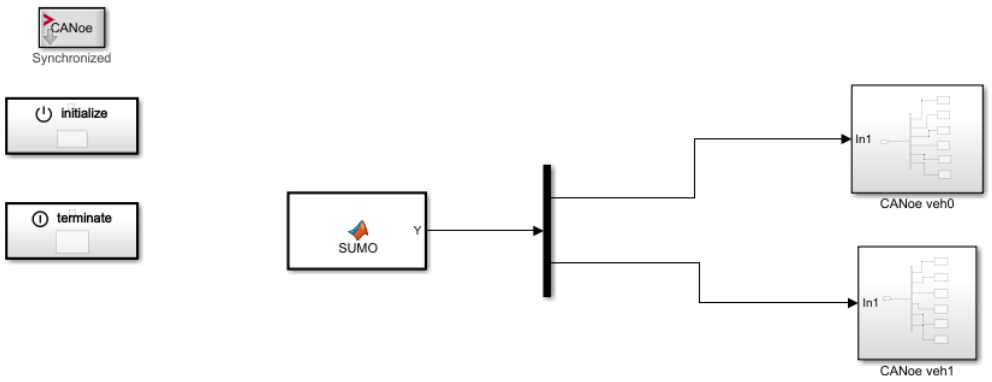

Figure 3.4: Simulink model

In this model every single block has a well defined role. The *"Initialize"* block executes the initialize event that is the connection with SUMO, in fact inside this block there is a "Matlab Function" block that establishes a socket connection with the SUMO server:

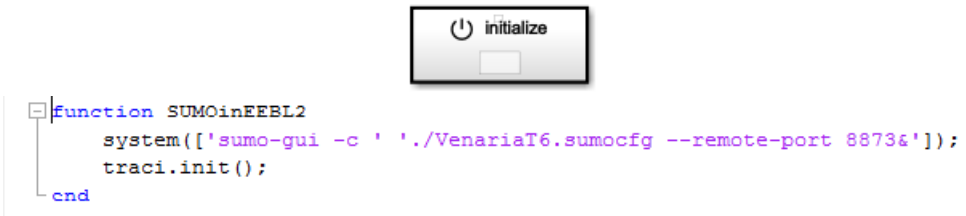

Figure 3.5: Initialize block

In the same way, at the end of the simulation the *"Terminate"* block closes the connection with SUMO:

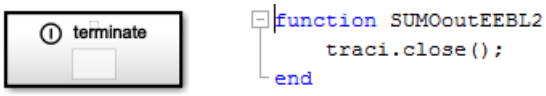

Figure 3.6: Terminate block.

The main block is the "Matlab Function" called SUMO (fig: 3.4), in this function at each step of the simulation time the relative data for the specific V2X use case are extrapolated from the SUMO scenario and organized in a array in a Matlab environment. After that, this array has been exploded in such a way to can associate each term of the array to the corresponding CANoe block to allow the communication with the CAN simulator.

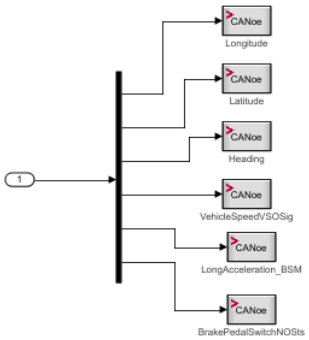

Figure 3.7: Communication between Simulink and CANoe.

3.1.3 CAN network Simulator: CANoe

In order to can access to the MM DSRC connectivity hardware through CAN interface it has been necessary to use CANoe software.
It is important focus on the configuration of the network chosen in CANoe and what were the reasons that led to this choice. In this project it has been used a particular network in which every single node was associated to a different network (fig:3.8).
This choice is referred to the fact that in this configuration each CANoe "Node" corresponds to a particular vehicle, so each "Network" block in CANoe will be associated to a different MM DSRC connectivity hardware.

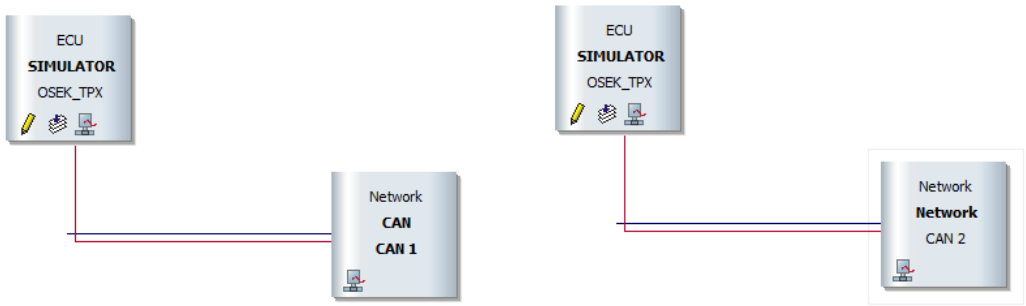

Figure 3.8: Network implemented in CANoe

In order to match data coming from Sumo with the corresponding CAN signals it has been used the *CANoe/Matlab Interface* and in particular the *"Signal Output"* Simulink block.
So once chosen the network, it has been necessary to create a specific database (.dbc) in order to match each Sumo data with a specific signal inside the database. This new "ad hoc database" consists in two ECUs the *Simulator* that represents transmitted node and the *Connectivity* that represented the receiver node.
In a created network (fig:3.8) the two nodes represented the two vehicles have been both associated to *Simulator* ECU because these nodes have both to send to the real hardware.

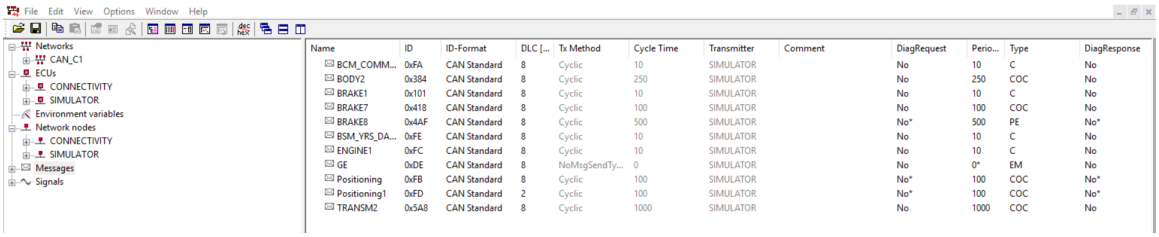

Figure 3.9: Structure of the new database

Inside the database it has been created a list with all the available messages and for each message a list with all the signals that have been managed according to the requirements of the specific use case.
So the configuration in CANoe consists of two networks, in order to can interact with two MM DSRC connectivity hardware, for each network there is a node and for each node there is the same "ad hoc database" in order to can match the data coming from Simulink with the real signals that will be delevered to a real hardware.

Another note regarding the "ad hoc database": one of the most important created message is the *Positioning message* that includes the *Latitude, Longitude* and

Heading signals. It is an important message because it has been done some modifications in the software in order to can get the GPS signals directly from the CAN. These modifications will be explained later on.

3.2 Time Synchronization

At this point it is important to talk about the simulation time of these software and the synchronization between them.

Regarding SUMO it is important to remember the the traffic simulator acts as server so, through TraCI, Matlab (the client), is responsible to decide the step of the simulation time. In fact in the main "Matlab function" of the Simulink model the first command is *Traci.simulationStep(STEP)* that performs a simulation step in the SUMO server with a time step equal to the number defined in STEP.

Regarding Simulink it has been important to choose the right solver selection. In simulink a solver applies a numerical method to solve the set of ordinary differential equations that represent the model. For the model representing in fig3.4 it has been chosen the Fixed-Step Solver that solves the model at step sizes from the beginning to the end of the simulation.

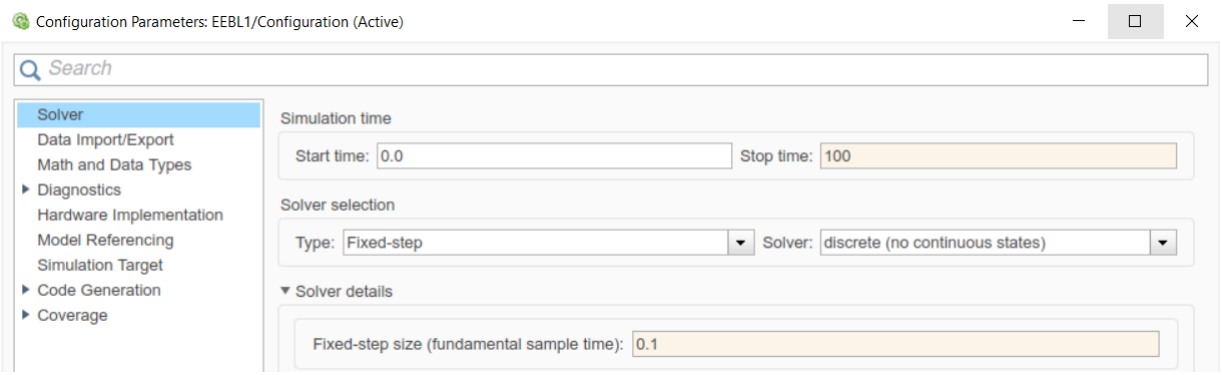

Figure 3.10: Solver selection in Simulink

From the CANoe point of view, it has been chosen the "Synchronized mode" in which the simulation is run in the MATLAB/Simulink environment and in contrast to the Offline Mode the CANoe time base is used in MATLAB/Simulink.
A Simulink model in fig3.4 that contains blocks of the CANoe block set library and executed in synchronized mode must use exactly one *Simulation step* block. Just like other Simulink blocks it can be dragged from the Simulink library browser

into the model. By default this block sets the execution mode to offline mode or synchronized mode. The Simulation Step block also manages the data exchange when the simulation is run in Simulink[10].

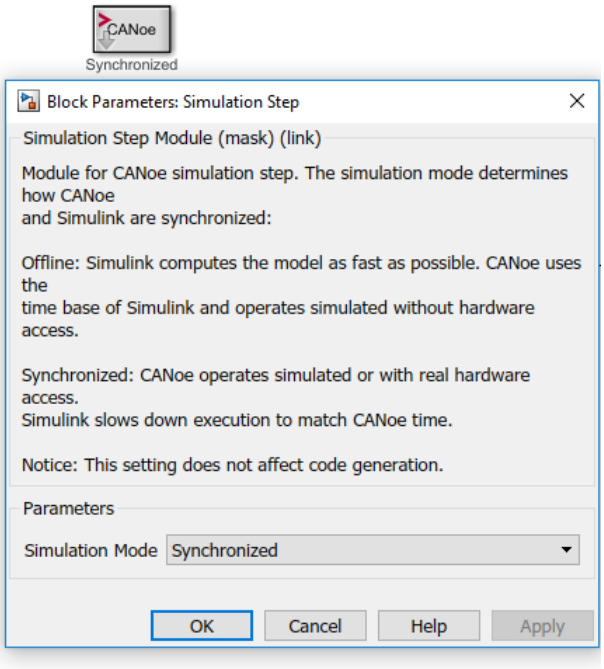

Figure 3.11: Simulation step CANoe

3.3 Hardware-in-the-Loop

Hardware-in-the-loop (HIL) simulation, is a technique that is used in the development and test of complex real-time embedded systems.
There are several areas in which HIL simulation offers cost savings over validation testing. For example in a context of automotive applications Hardware-in-the-loop simulation systems provide such a virtual vehicle for systems validation and verification. Since in-vehicle driving tests for evaluating performance and diagnostic functionalities of Engine Management Systems are often time-consuming, expensive and not reproducible, HIL simulators allow developers to validate new hardware and software automotive solutions, respecting quality requirements and time-to-market restrictions. HIL simulation also offers a high degree of repeatability during testing phase.
Hardware-in-the-Loop is one of the most important phases of the V-shape development flow described in a Model Based Design approach:

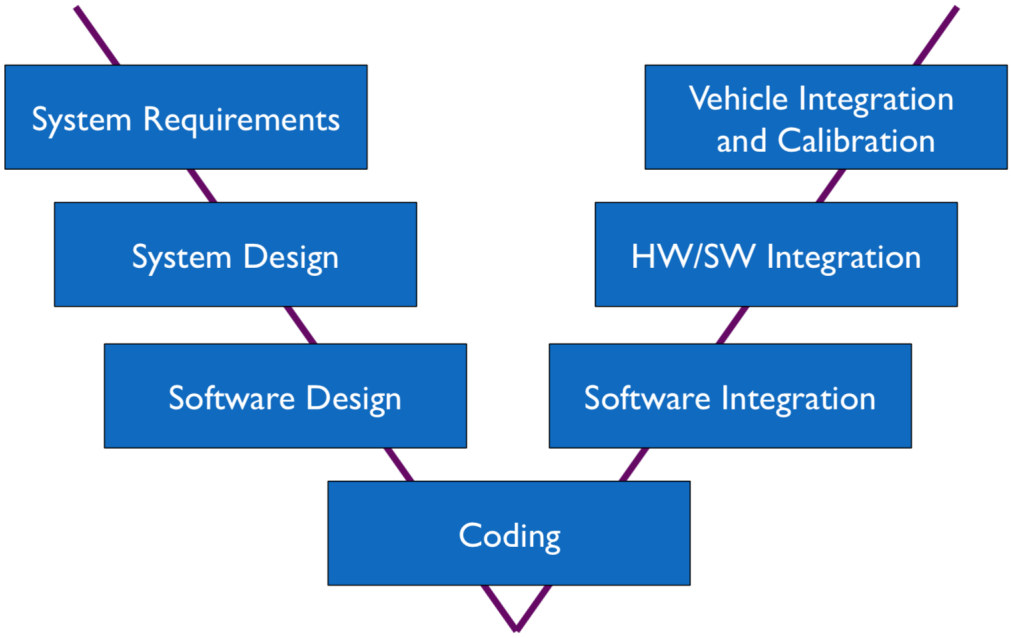

Figure 3.12: V-shape development flow

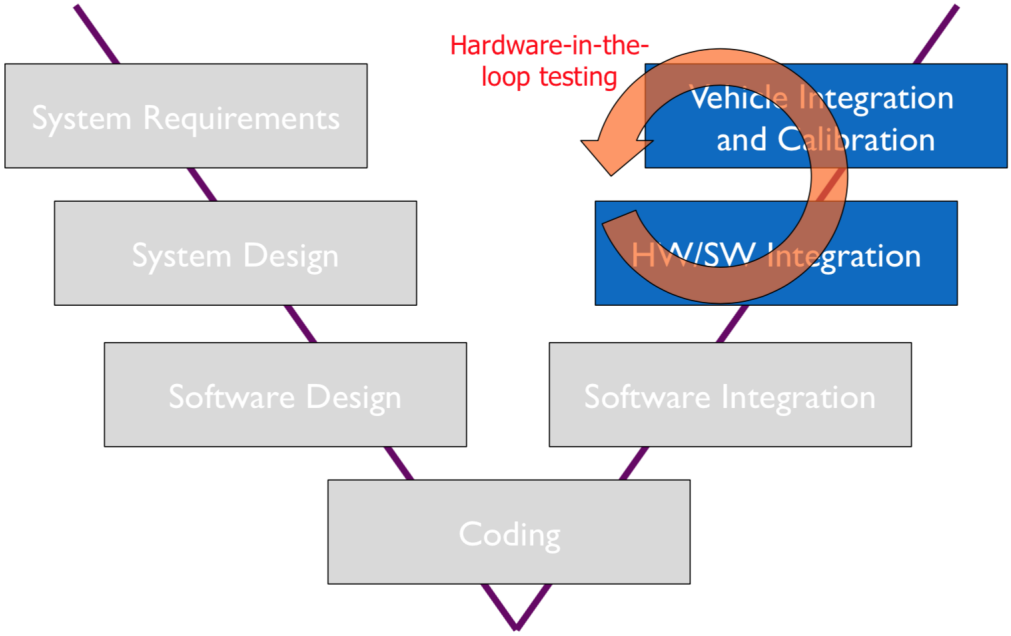

Figure 3.13: Hardware-in-the-Loop in the V-shape development flow

3.4 The Simulation Environment

It has been chosen to apply the Hardware-in-the-loop simulation technique to validate the entire model. So if in principle the MM Connectivity boards were essentially used to execute the in-vehicle driving test, now it has been decided to create a simulation bench test but with the requirements to can interact with the real hardware in order to can test with a simulated scenario the real use cases implemented. Therefore the purpose of implementing the Hardware-in-the-Loop in which the real world (the MM DSRC connectivity boards) can communicate and get data from a simulated bench test (implemented with the communication chain SUMO-MATLAB-CANoe).

The idea that was implemented consists in stimulating the specific use case in the framework under testing avoiding to bring the MM DSRC connectivity boards inside the real vehicle. In order to do this it has been created a bench test as depicted in figure 3.14 in which the scenario will be implemented in SUMO, the specific data regarding the dynamic vehicle will be get by MATLAB and delivered them to CANoe in order to can have the access with the real hardware and stimulate the specific use case directly inside the MM connectivity framework.

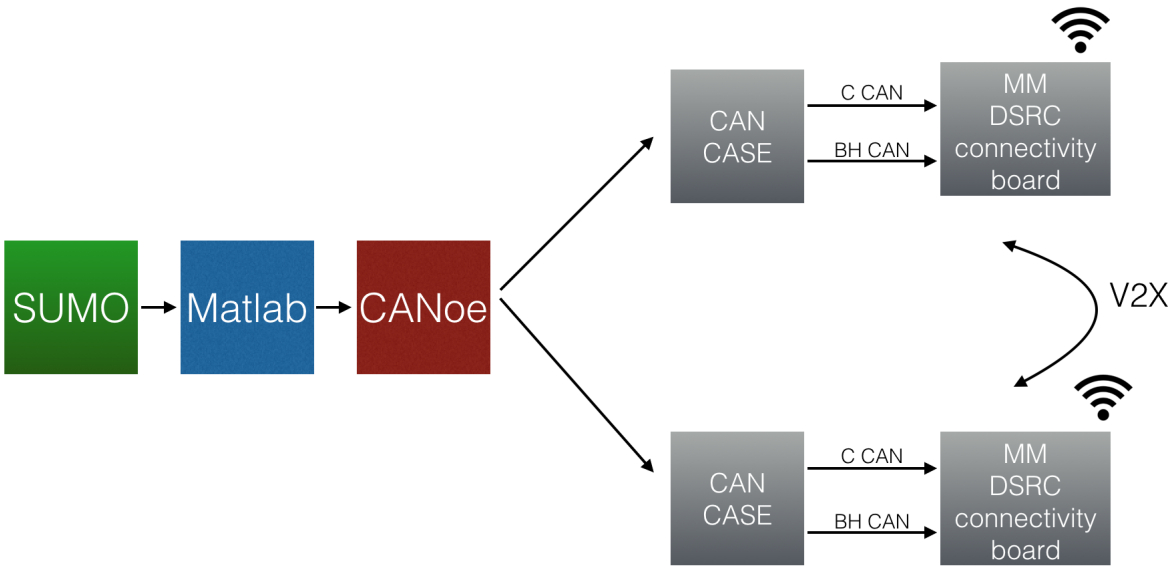

Figure 3.14: Block scheme of the simulation environment

Figure 3.15: Picture of the hardware settings.

The picture shows a photo of the two MM DSRC connectivity boards, each of one is connected to a single "Can Case" that via USB are connected to a PC.
As the picture shows, it has been using another powerful instrument of the MM connectivity team that is the "V2X HMI App", in order to visualize on the tablet the V2X event alerts that the vehicle should be shows in own dashboard.

In order to implement this configuration, the team supported me with an important modification of the software regarding the acquisition of the GNSS data (Latitude, Longitude, Heading).
Inside the MM connectivity framework there are two main components: the *"CANdataprovider"* and the *"GPSdataProvider"*, the first one is responsible for the acquisition of the messages delivered through CAN network, while the second one provides the GPS coordinates of the vehicle. This configuration represented an obstacle for my purpose, because in my simulation all the data regarding the vehicle, Latitude and Longitude too, were delivered to the MM DSRC connectivity hardware only through CAN network.
In order to reach this goal the team modified the *"GPSdataprovider"* in order to be able to get the GNSS data directly from the CAN through *"CANdataprovider"*.

Practically as shows the figure 3.16 it has been delivered to the CANdataprovided the new XML files, in which together with the previous messages it has been added a messages regarding the *Positioning* with the Latitude, Longitude and Heading signals. Of course there was a correspondence between this new XML file with the ad-hoc database created in CANoe, even better this new XML file has been derived from the CANoe database thanks to a particular MM editor. The second file XML is the so called *GENIVI* file. In this file there is a higher level matching in which there is an association between the message with the relative signals and an engineer value.
With these modifications the CANdataProvider can communicate with VDP Mediator component (that in general supplies to the upper layer components other vehicle informations such as its dynamic), and this structure substitute the role of the POTI mediator and the GPSdataProvider.

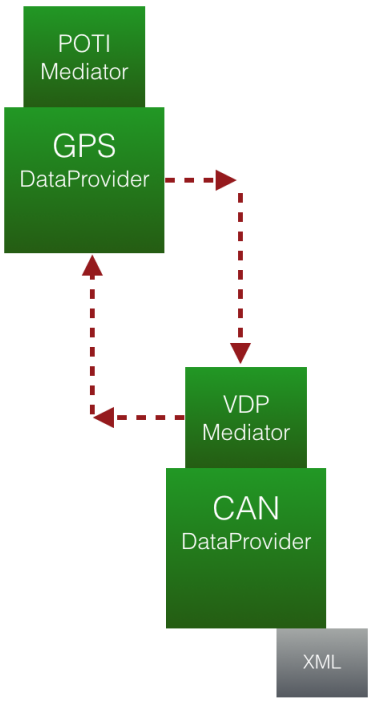

Figure 3.16: modification of the CANdataProvider

Chapter 4

Use Cases implemented

4.1 V2X Use Cases

As I said in the Introduction chapter the currently V2X use cases implemented by the team in the MM connectivity framework are the Control Loss Warning (CLW), Emergency Electronic Brake Light (EEBL), Forward Collision Avoidance (FCW), Left Turn Assist (LTA), Intersection Movement Assist (IMA), Stationary Vehicle Warning (SV) for what concerns the V2V communication and the Green Light Optimized Speed Advise (GLOSA) and In Vehicle Road Sign (IVRS) for what concerns the V2I communication.
After a brief description of each of these uses cases, it will be described how the simulation environment implemented it has been applied for one the V2V use case.

4.1.1 Control Loss Warning CLW

Control Loss Warning is an event generated and broadcast by a vehicle whose sensors detect loss of traction. Upon receiving such information, the surrounding vehicles evaluate the relevance of the event and, if appropriate, provide a warning to the driver[2].

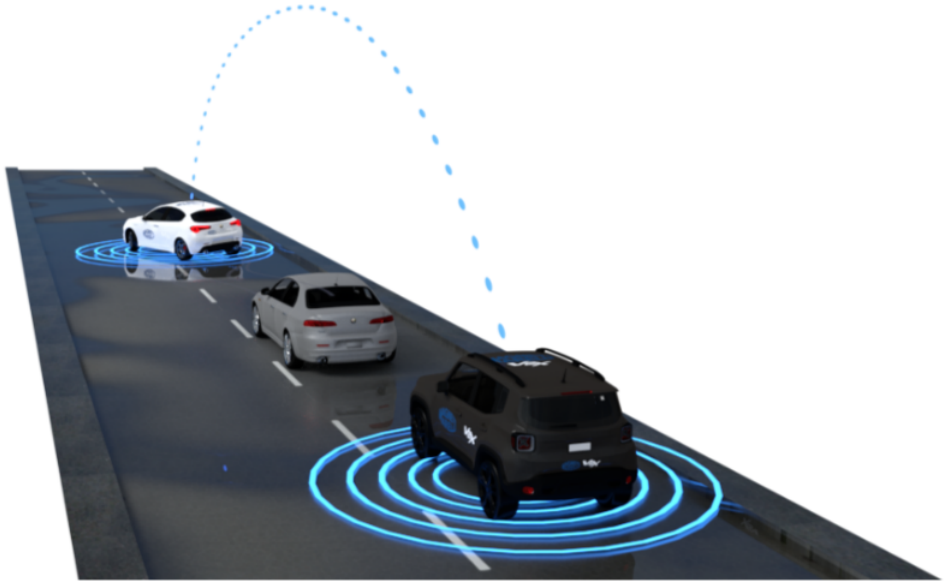

Figure 4.1: Control Loss Warning event

4.1.2 Emergency Electronic Brake Light EEBL

Emergency Electronic Brake Light is an event generated and broadcast by a vehicle performing an emergency brake. Upon receiving such information, the surrounding vehicles evaluate the relevance of the event and, if appropriate, provide a warning to the driver[2].

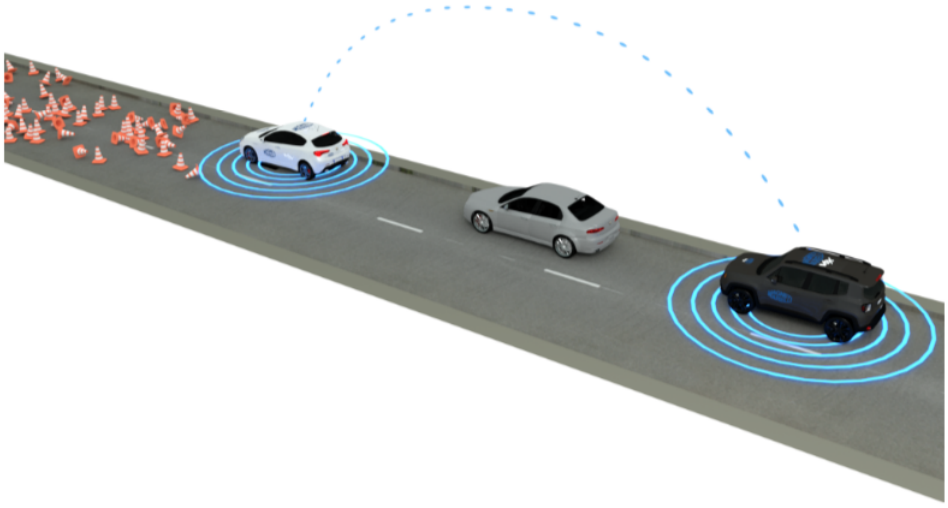

Figure 4.2: Emergency Electronic Brake Light event.

4.1.3 Forward Collision Avoidance FCW

Forward Collision Warning is an event generated by a vehicle whose sensors detect an impending rear-end collision with the vehicle ahead in traffic, in the same lane and direction of travel.

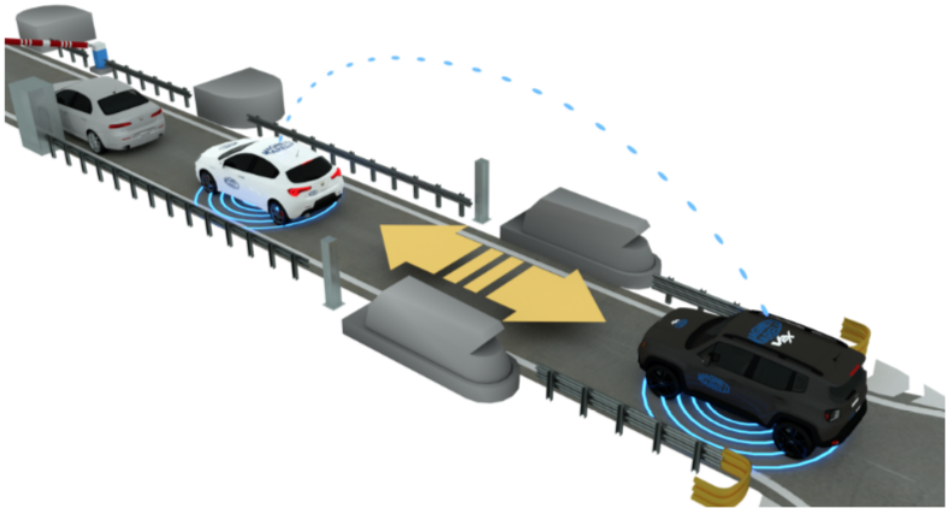

Figure 4.3: Forward Collision Avoidance event

4.1.4 Left Turn Assist LTA

This use case is generated by a vehicle attempting to turn left at an intersection in order to warn its driver an oncoming vehicle that may be out-of-sight[**usecase**].

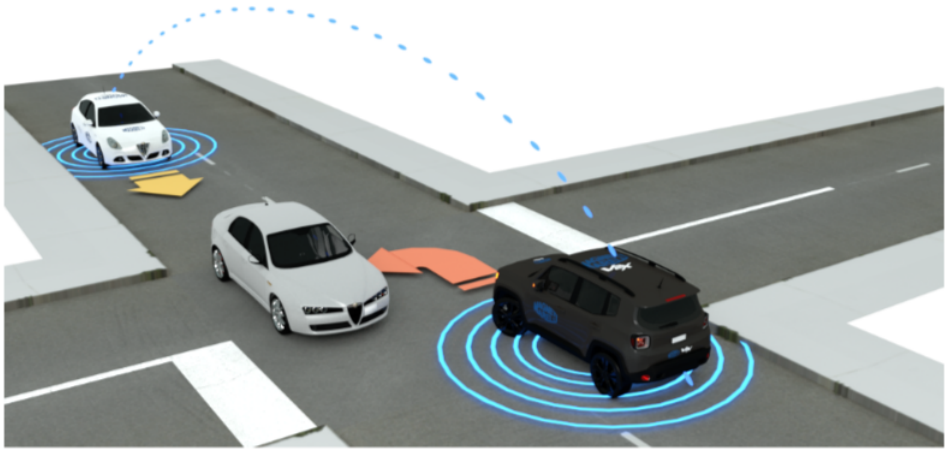

Figure 4.4: Left Turn Assist event

4.1.5 Intersection Movement Assist IMA

Intersection Movement Assist is an event generated by a vehicle approaching an inter- section, which detects a crossing vehicle. The goal is to warn the driver about the high collision probability [**usecase**].

Figure 4.5: Intersection Movement Assist event

4.1.6 Stationary Vehicle SV

Stationary Vehicle is an event generated and broadcast by a vehicle having flashing emergency lights on. The goal is to warn other drivers and have them reducing their speed and possibly making a lane change.

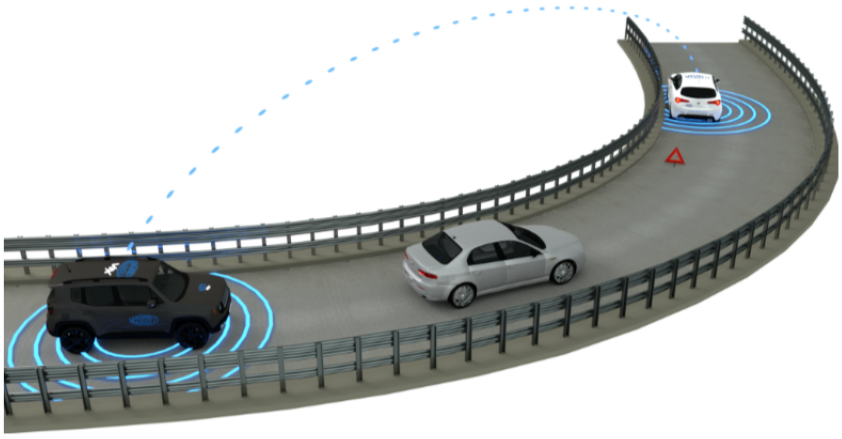

Figure 4.6: Stationary Vehicle event

4.2 Emergency Electronic Brake Light UC-EEBL

The **EEBL** component aims to manage the EEBL events in bidirectional way: either as sender, when the Ego vehicle makes a hard braking and it has to alert the vehicles nearby about such event, or as receiver when the Ego vehicle has to be aware of a hard braking of the vehicle(s) ahead.
In order to apply the created simulation environment for this use case, first of all it has been created the network file and a traffic demand file in SUMO. The **Network file** *.net.xml* consists in a map of the MM plant in Venaria downloaded form OpenStreetMap and converted in SUMO; while the **Traffic demand file** *.rou.xml* file consists of a definition of two vehicles (the Ego vehicle and the Host one) that moves in the same street and in the same direction with the same dynamic vehicle too, but different acceleration.

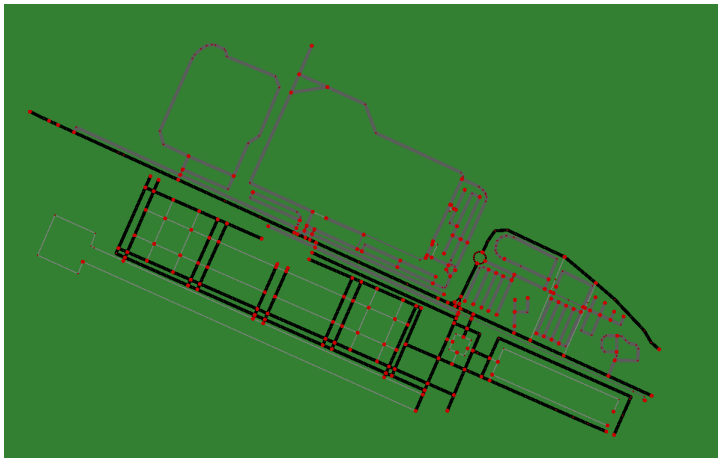

Figure 4.7: Network file

```
<routes>
    <vType id="Car" length="2.0" maxSpeed="100.0" sigma="0.0" />
    <route id="route0" edges="-43284483#8 -43284483#7 -43284483#6 -43284483#5 -43284483#4 -432844
    <vehicle depart="0" id="veh0" departSpeed="10" Speed="50" route="route0" type="Car" />
     <vehicle depart="0" id="veh1"  departSpeed="10" Speed="10"  route="route0" type="Car" />
</routes>
```

Figure 4.8: Traffic Demand File

Once created the scenario in SUMO, it has been imported in the MATLAB environment with a "MATLAB Function" Simulink block called *SUMO*(fig:3.4). Inside this block through **TraCI** it has been possible to get the input data for the EEBL while running the simulation, in particular these data concern only a dynamic of a vehicle so it has been used only the TraCI domain *vehicle*.
So, since the input for this use case were:

- Latitude
- Longitude
- Heading
- Speed
- Longitudinal Acceleration
- Brake Pedal Status

and the attributes of the SUMO's simulation objects can be accessed with this command:
traci.<domain>.<get/set_wrapper()>, inside the "SUMO" Matlab Function we can find:

```
   pos = traci.vehicle.getPosition('veh0')
(long,lat) = traci.simulation.convertGeo(pos(1),pos(2))
heading = traci.vehicle.getAngle('veh0')
speed = traci.vehicle.getSpeed('veh0')
accel = traci.vehicle.getAccel('veh0')
BrakePedal = boolean(traci.vehicle.getSignals('veh0'))
```

A note about the position because the command *traci.vehicle.getPosition()* returns the position as X,Y coordinates so it is necessary to convert this data to GPS coordinates with the command *traci.simulation.convertGeo()*.
This procedure it has been implemented for the two different vehicles and it has been collected these data in two different arrays Y0 and Y1 corresponding to the first and the second vehicle. After this step in Simulink each array it has been delivered to a subsystems called *veh0* and *veh1* (fig: 3.4) in which the previous array was exploded in order to can associate each element of the array to a corresponding *Signal output* Vector block to can switch to CANoe environment.
As the picture 4.9 shows every "signal output" block corresponds to a real signal of the "ad hoc database" and it is possible to select the signal of interest through the *SignalOutputBlock* window (fig: 4.9).

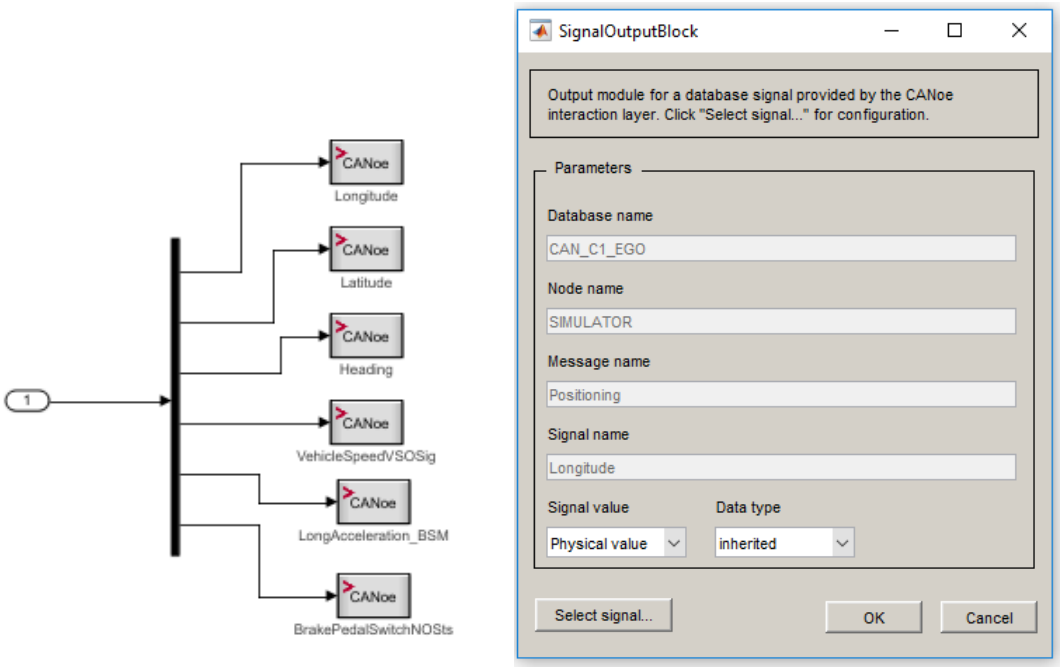

Figure 4.9

In parallel in the CANoe environment it has been developed the two different networks. Each network has a node associated corresponding to a vehicles and each node has the same database associated called *CAN_ C1_ EGO* and *CAN_ C1_ HOST*.

Once completed this configuration, it has been proceeded with the test analysis. The input for the EEBL were the position in terms of **Latitude**, **Logitude** and **Heading**; the **Speed** of the vehicle; the **Longitudinal Acceleration** of the vehicle and the **Status of the Brake Pedal**; once get this information, if the Ego vehicle notify a deceleration equal to 4 m/s$\hat{2}$ and the status of the brake pedal is equal to one, the EEBL event was delivered through *BSM (Basic Safety Message)* via DSRC.
BSM stands for *Basic Safety Message*. The Basic Safety Message (BSM) is a standardized communication packet that is sent every tenth of a second between connected vehicles using Dedicated Short Range Communication (DSRC). BSMs contain data about the sending vehicle's state, such as speed, location, and the status of the turn signal. Applications use this data to issue warnings and alerts to drivers regarding nearby vehicles.

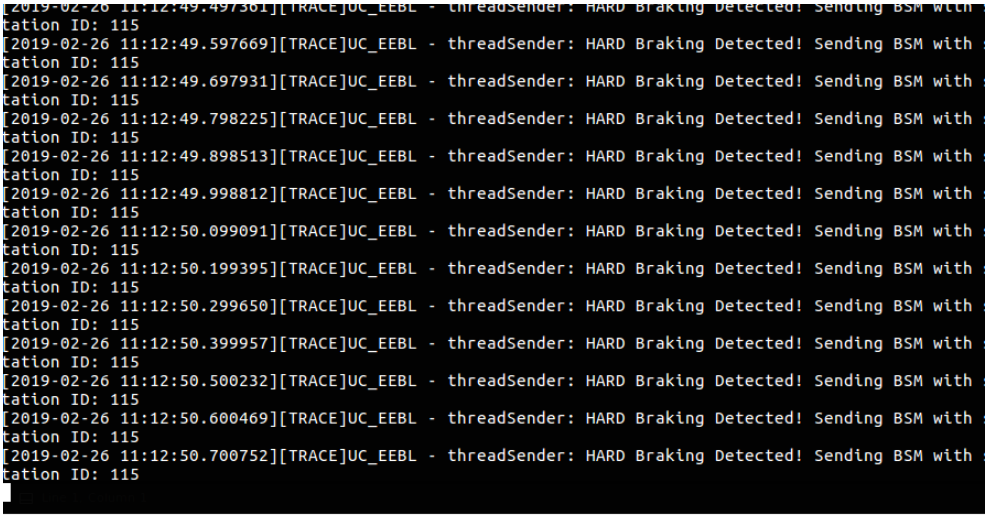

Figure 4.10: Ego vehicle

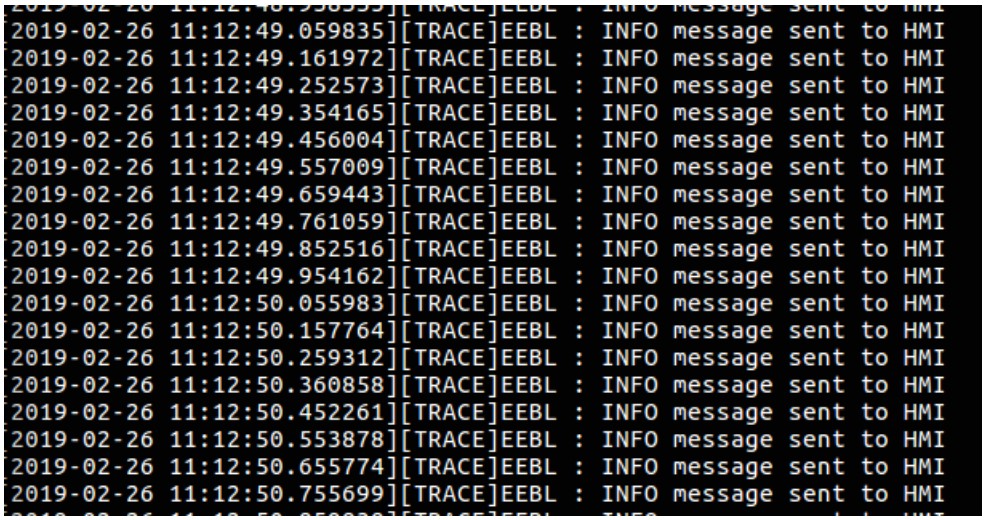

Figure 4.11: Host vehicle

The picture shows the behaviour of the two MM DSRC connectivity hardware related to the Ego vehicle and the Host vehicle while running the simulation with the environment described above.
In the first picture is shown the behaviour of the Ego vehicle, as a matter of fact there is a message *HARD BRAKING DETECTED*, this means that the requirements for the EEBL use case have been satisfied and the EEBL event has been notified; another important message that appears in the same line is *SENDING BSM*.
In the second one there is the "response" of the Host vehicle *INFO message sent to*

HMI. Once received the BSM with the information of the hard braking the vehicle sends this information to the "Use Case Manager" and consequently to the "HMI Manager Dispatcher" in order to can display on the dashboard and on the V2X HMI app the relative warning message.

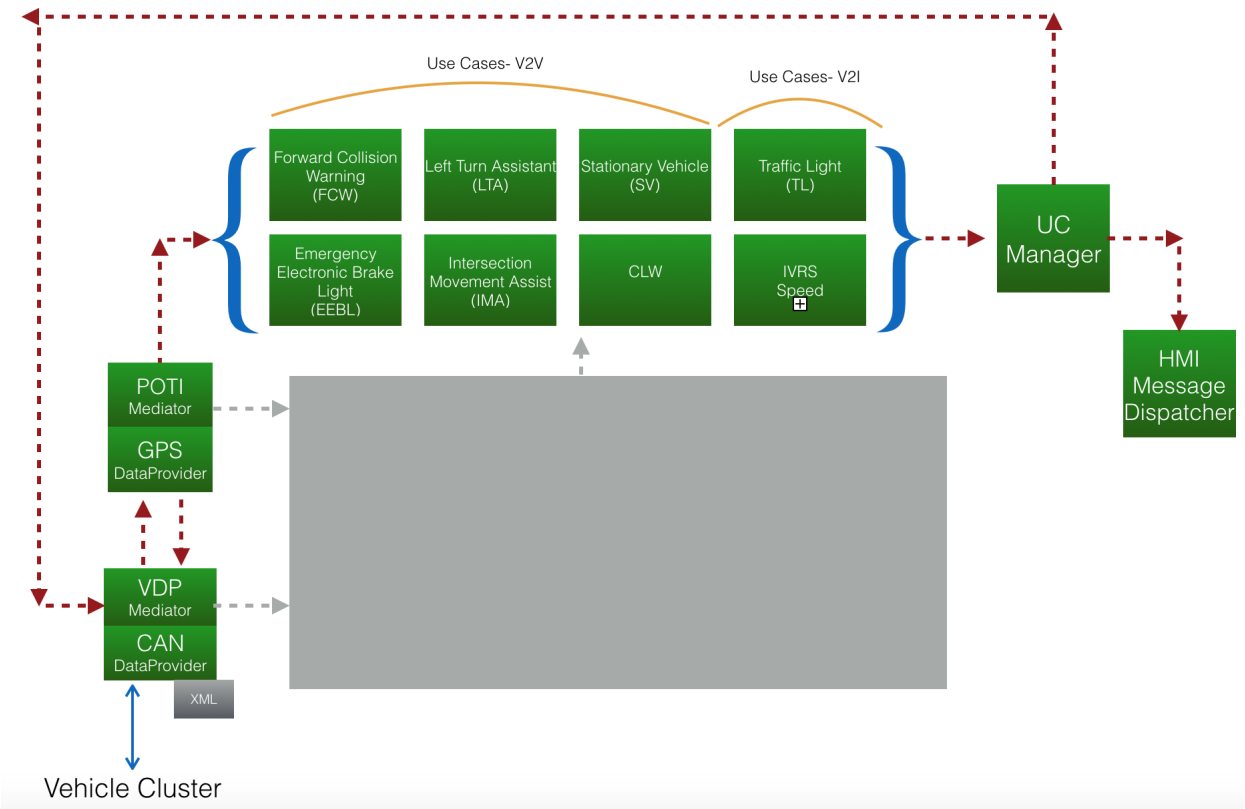

Figure 4.12: Full Architecture

It is important to underline that there is a specific procedure to evaluate when it is opportune to warn the driver of the Host vehicle about a received event that can pose a danger for the vehicle itself.
Taking into account some parameters like the position, velocity and acceleration of the Host vehicle, the uncertainty of the position of the Host vehicle and of the possible Event the MM connectivity team developed an algorithm in order to define the so called **"Area of Interest"**. The Area of Interest of the Host vehicle is the set of points that the Host vehicle can reach in order to notify to the driver the relative warning associate to a use case event.

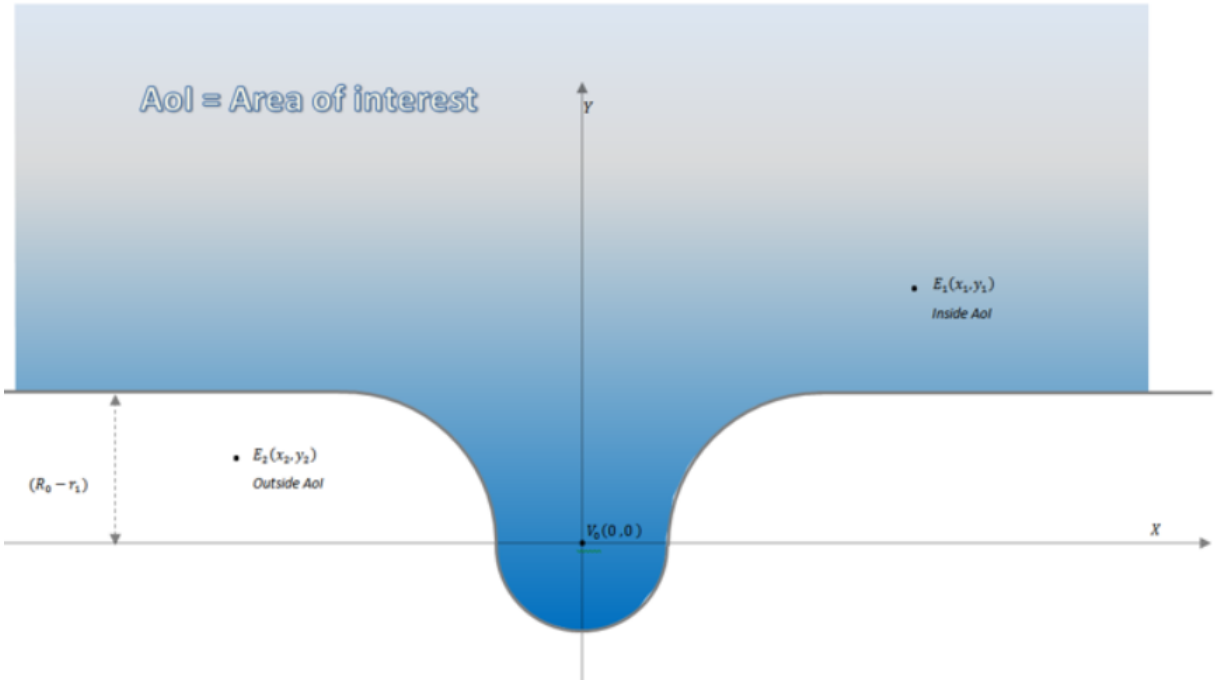

Figure 4.13: Definition of the Area of Interest.

Where R0 and r1 are respectively the uncertainty refers to the Host vehicle and the Event.
The Area of Interest is function of the velocity of the Host vehicle too. Specifically as increase the velocity as the Area of Interest decrease with a particular behaviour. Let's assume:

- R0 = 5m
- r1 = 5m
- a = 3m/s$\hat{2}$ (acceleration of the Host vehicle)

Changing the value of the velocity from 40Km/h to 120Km/h the Area of Interest change in this way:

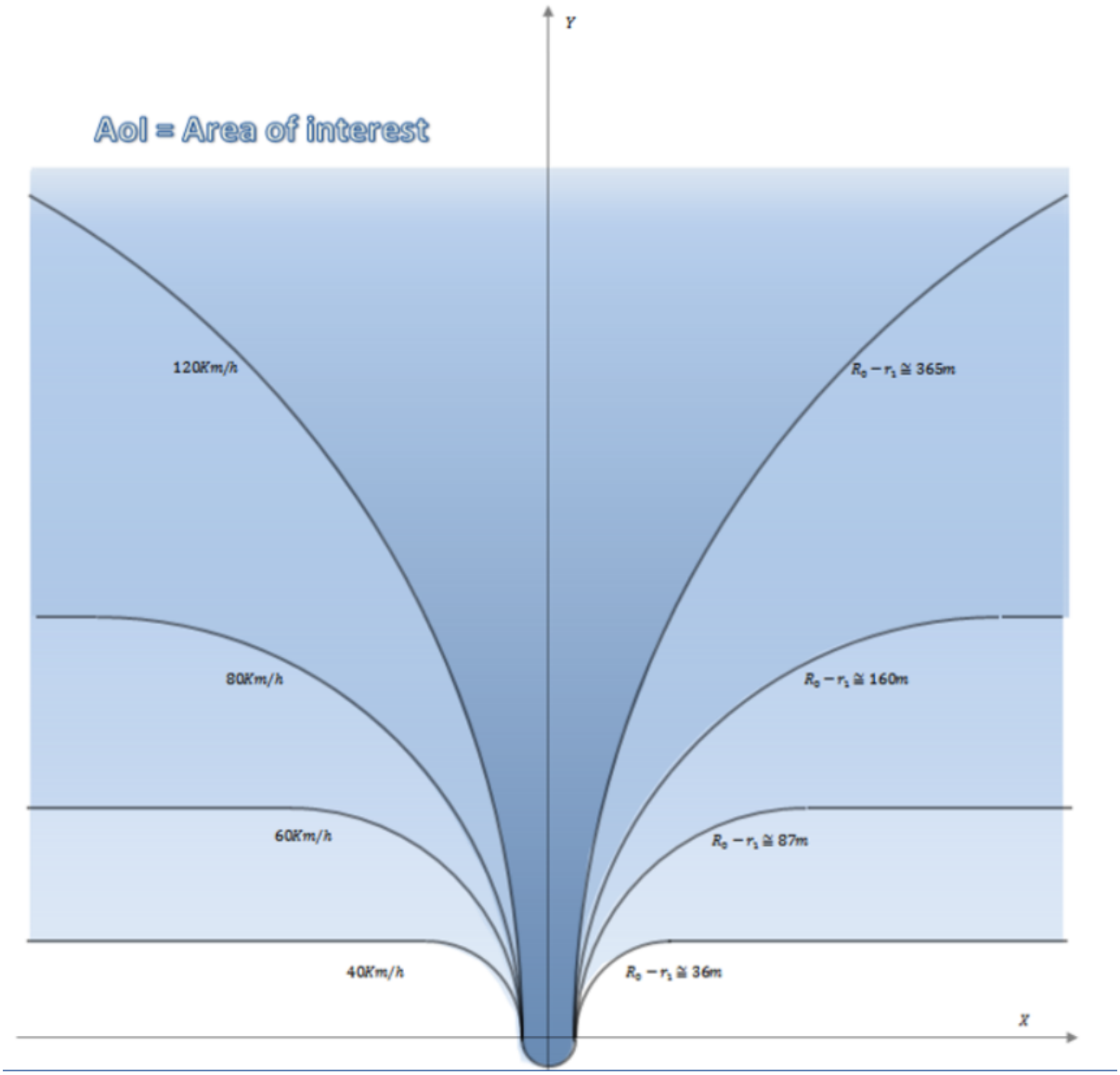

Figure 4.14: Definition of the Area of Interest.

4.2.1 Test Analysis

In order to test the correctness of the simulation environment implemented, for the EEBL use case it has been repeated all the executed in-vehicle driving tests :

- **Test 1**: EV travels at medium speed (e.g. 50Km/h), HV follows at the same speed, in the same lane, with a medium distance (e.g. 20m) when the EV performs the harsh braking.
  Expected Result: HV DOES issue warning to HMI.

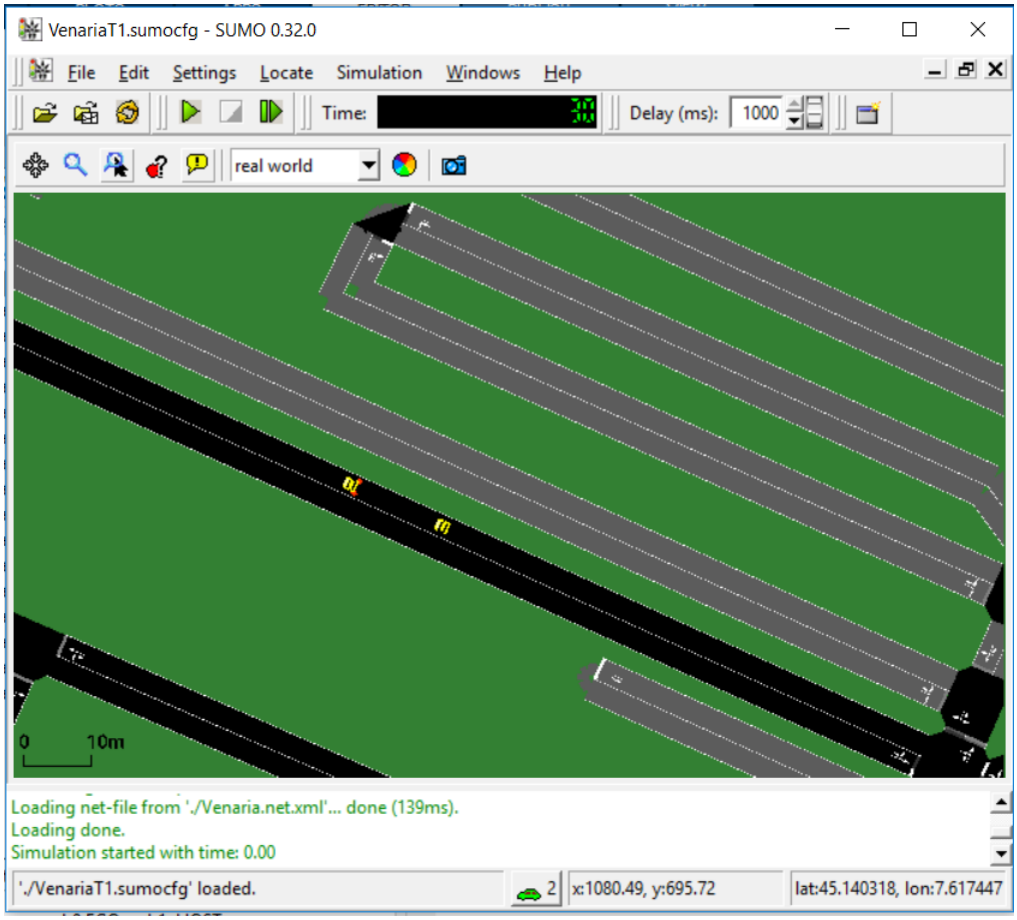

Figure 4.15: Test1 in Sumo

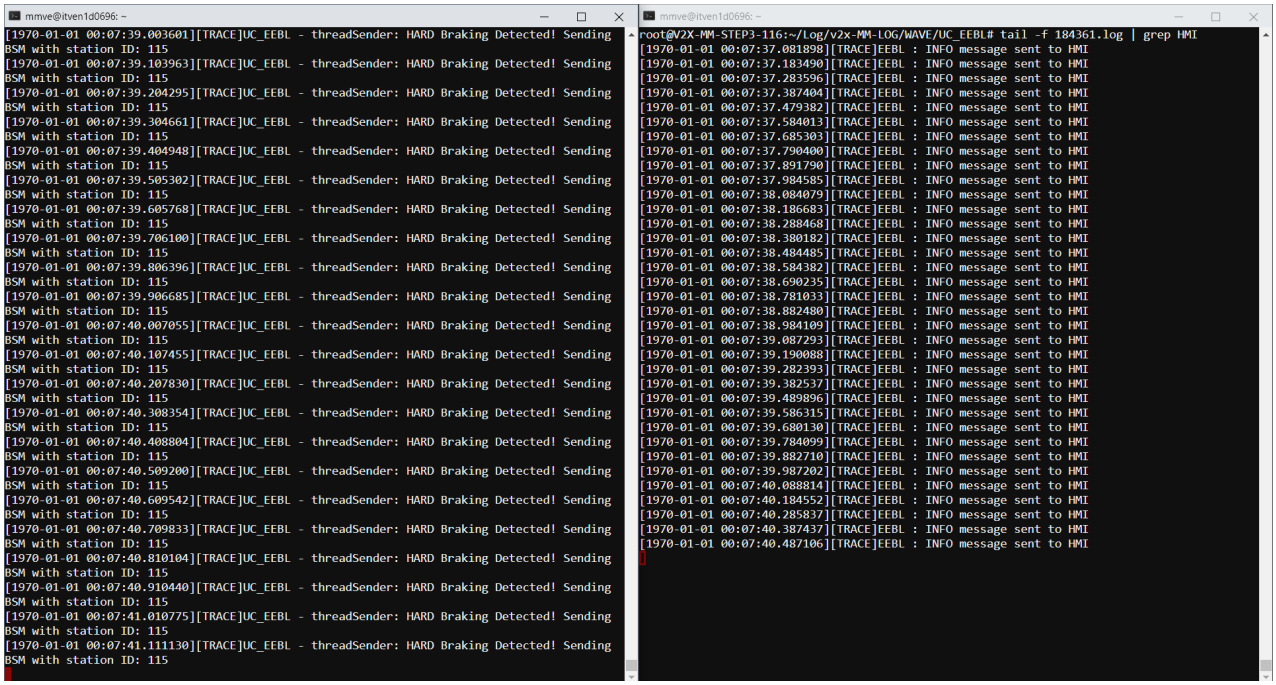

Figure 4.16: Test1

With the created simulation environment, the implementation of this Test was very simple because it has been just modified the Traffic Demand File setting the new velocity for the two vehicles equal to 50Km/h and the distance between them equal to 20m. As the second figure shows the Ego vehicle sends the BSM with the event of the EEBL and the Host vehicle, being in the Area of Interest sent the warning to the HMI.
The expected result is the obtained result.

- **Test 2**: EV travels at medium speed (e.g 50Km/h), HV travels ahead, at the same speed, in the same lane, with a medium distance (e.g. 20m) when the EV performs the harsh braking.
  Expected Result: HV DOES NOT issue warning to HMI.

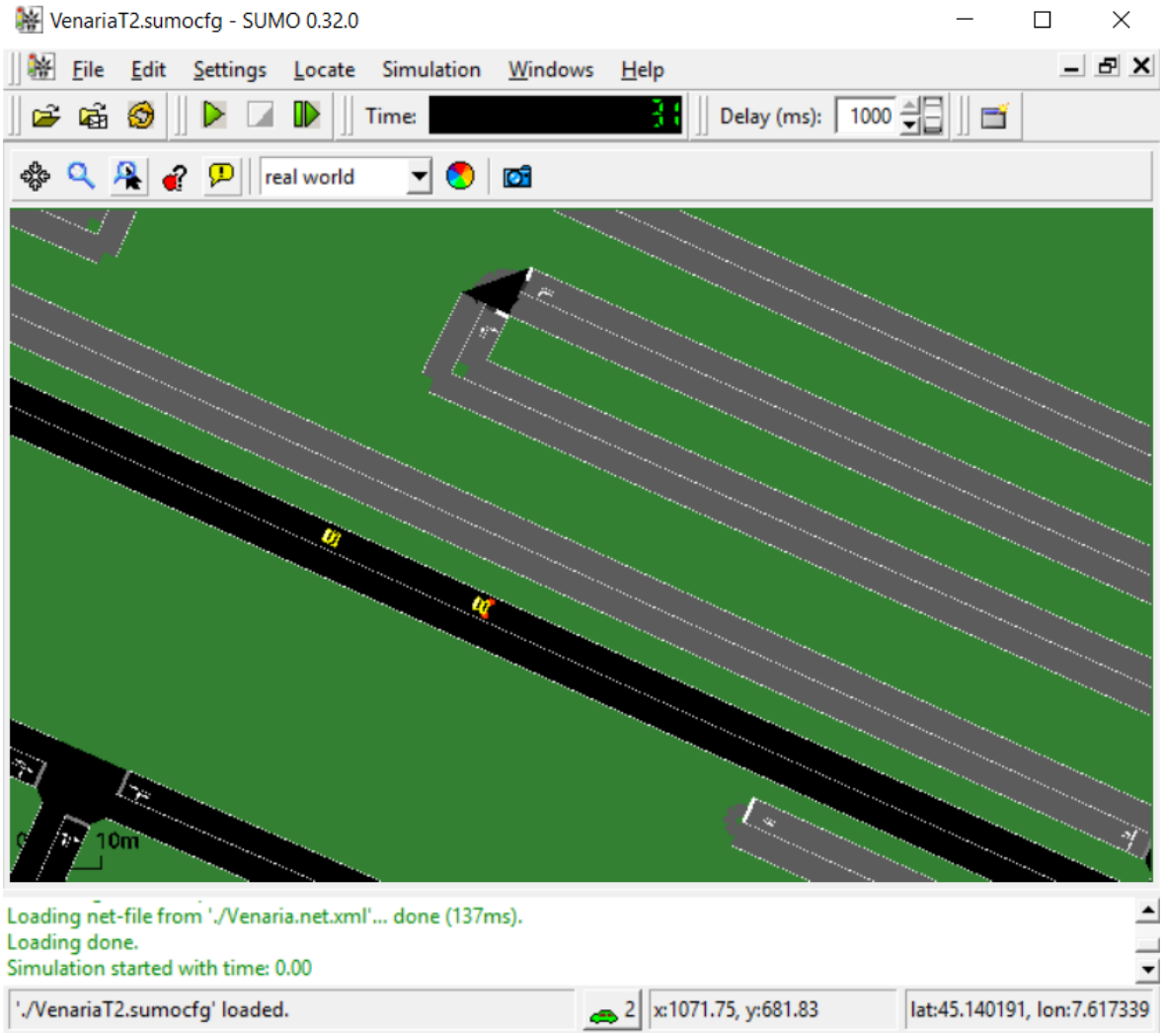

Figure 4.17: Test2 in Sumo

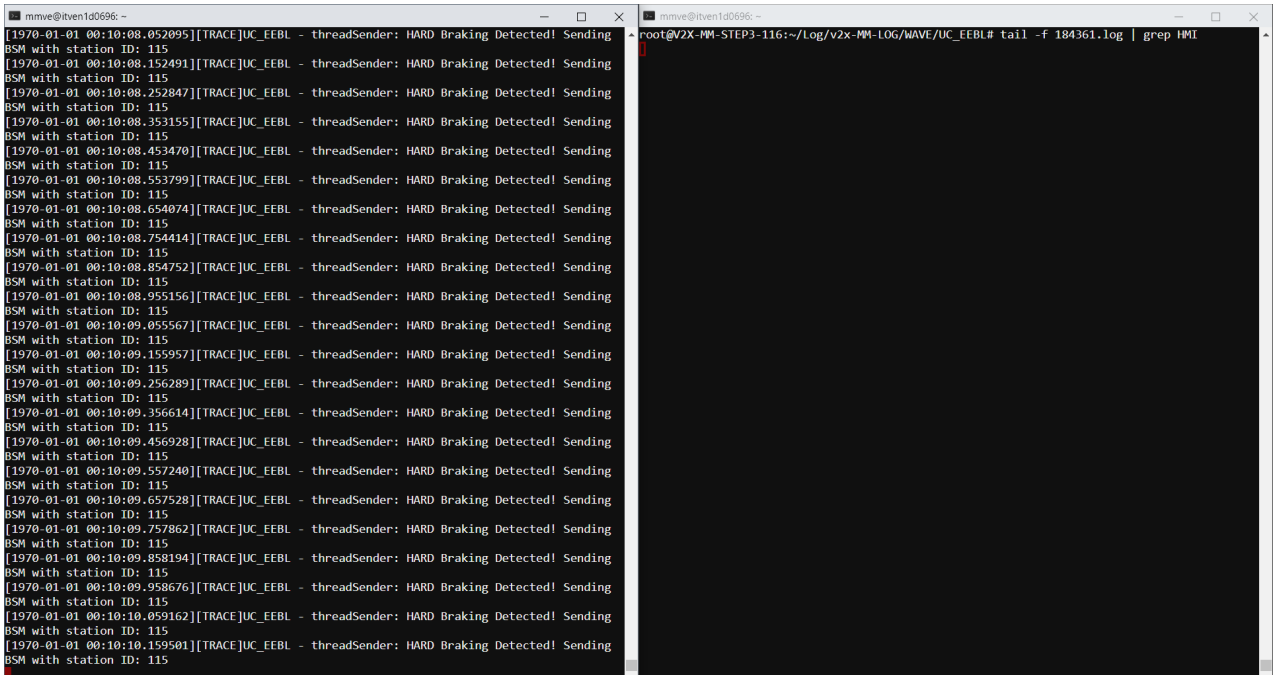

Figure 4.18: Test2

In this test case even if the dynamic and the speed of the two vehicles is the same and even if the distance between them is the same, the Ego vehicle stays behind the Host vehicle so the EEBL event sent by the Ego vehicle does not affect the Area of Interest of the Host vehicle so no warning has been transmitted to the HMI.
The expected result is the obtained result.

- **Test 3**: EV travels at medium speed (e.g. 50Km/h). HV travels at the same speed, in the opposite direction when the EV performs the harsh braking. **Expected Result**: HV DOES NOT issue warning to HMI.

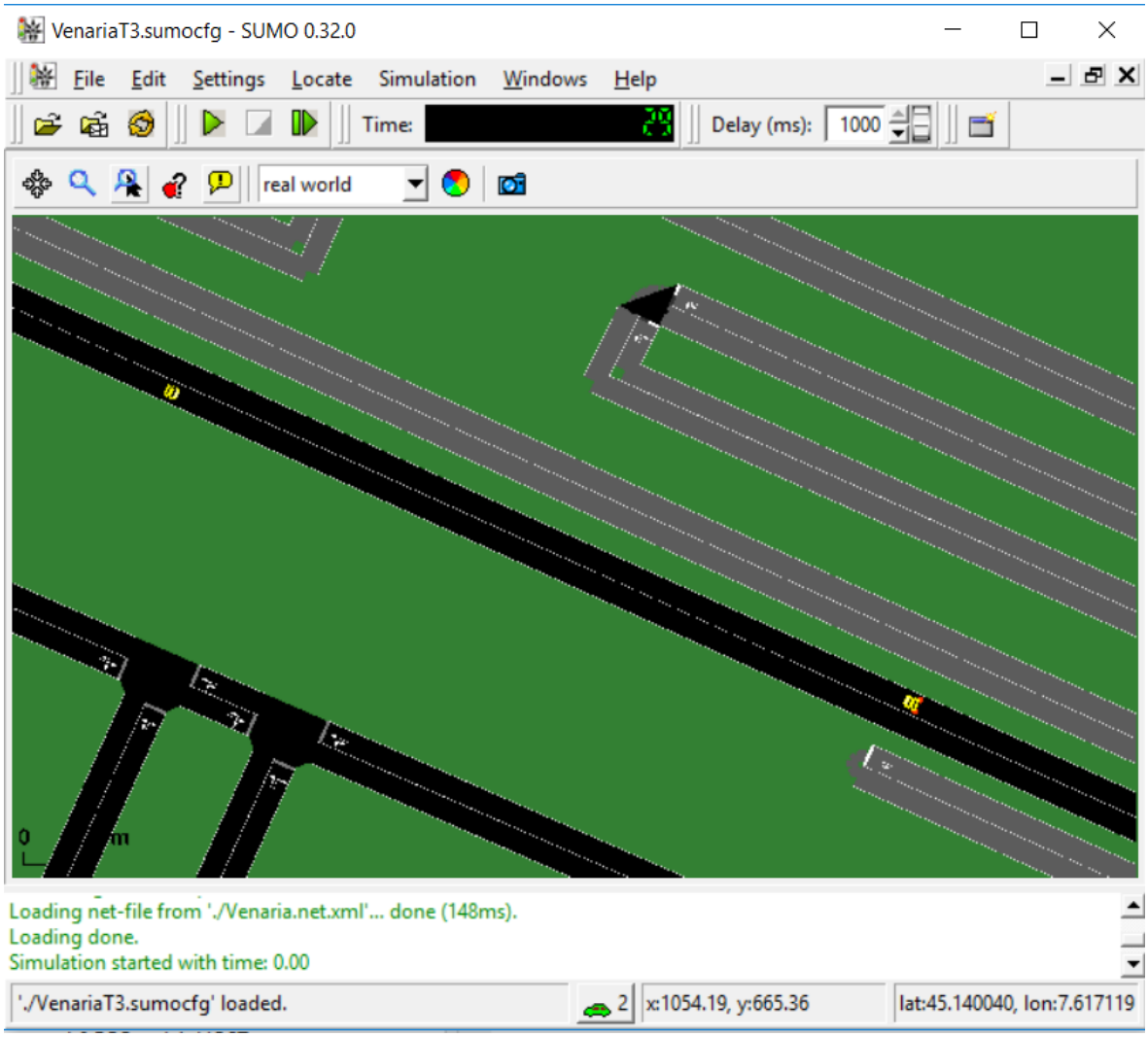

Figure 4.19: Test3 in Sumo

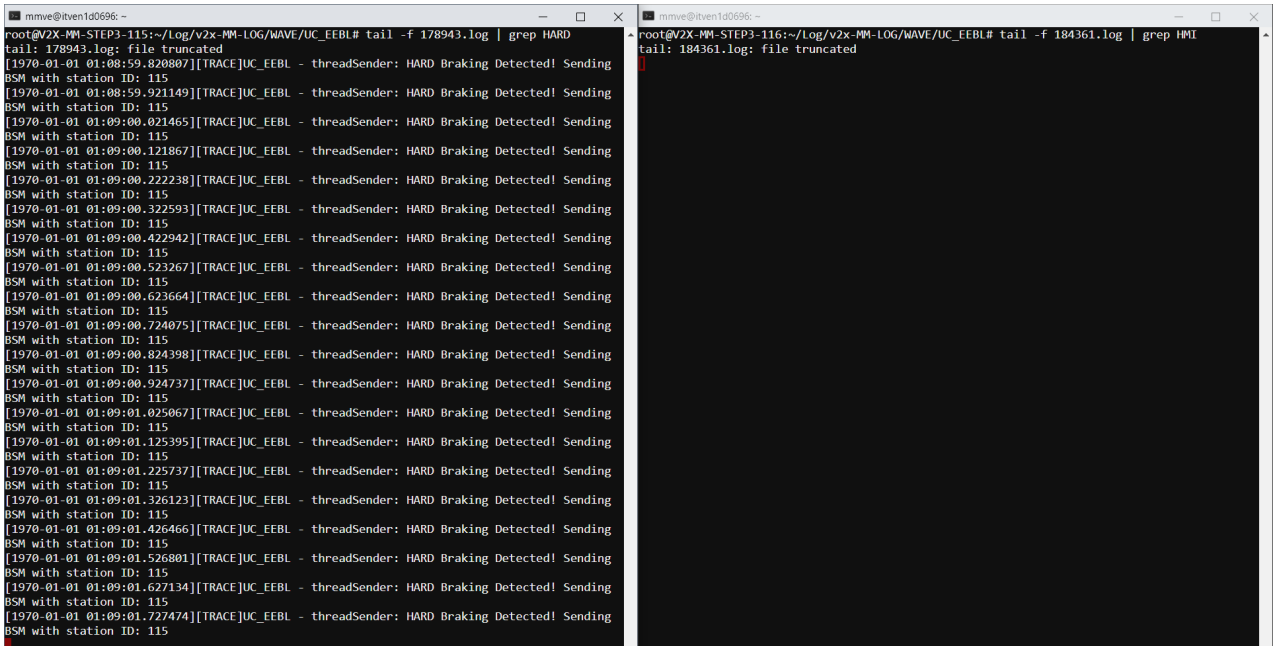

Figure 4.20: Test3

Also in this case the Host vehicle does not send warning to the HMI because it is travelling with an opposite direction with respect to the Ego one; the algorithm of the Area of Interest takes care of this factor so, as expected, the Host vehicle does not issue any warning.

For this test case, this kind of simulation allows to create a critical scenario, that during the test with the real vehicles it had not been possible to create.
This critical scenario is the scenario in which the position of the Host vehicle, when the Ego vehicle performs the hard braking, is almost aligned with respect to the Ego one. The scenario in SUMO in reported in figure 4.21; this configuration brought to an unexpected result because, the Host vehicle notified a warning as if it was behind the Ego vehicle. The algorithm of the "Area of Interest", for the unknown reason, did not take into account the opposite direction.

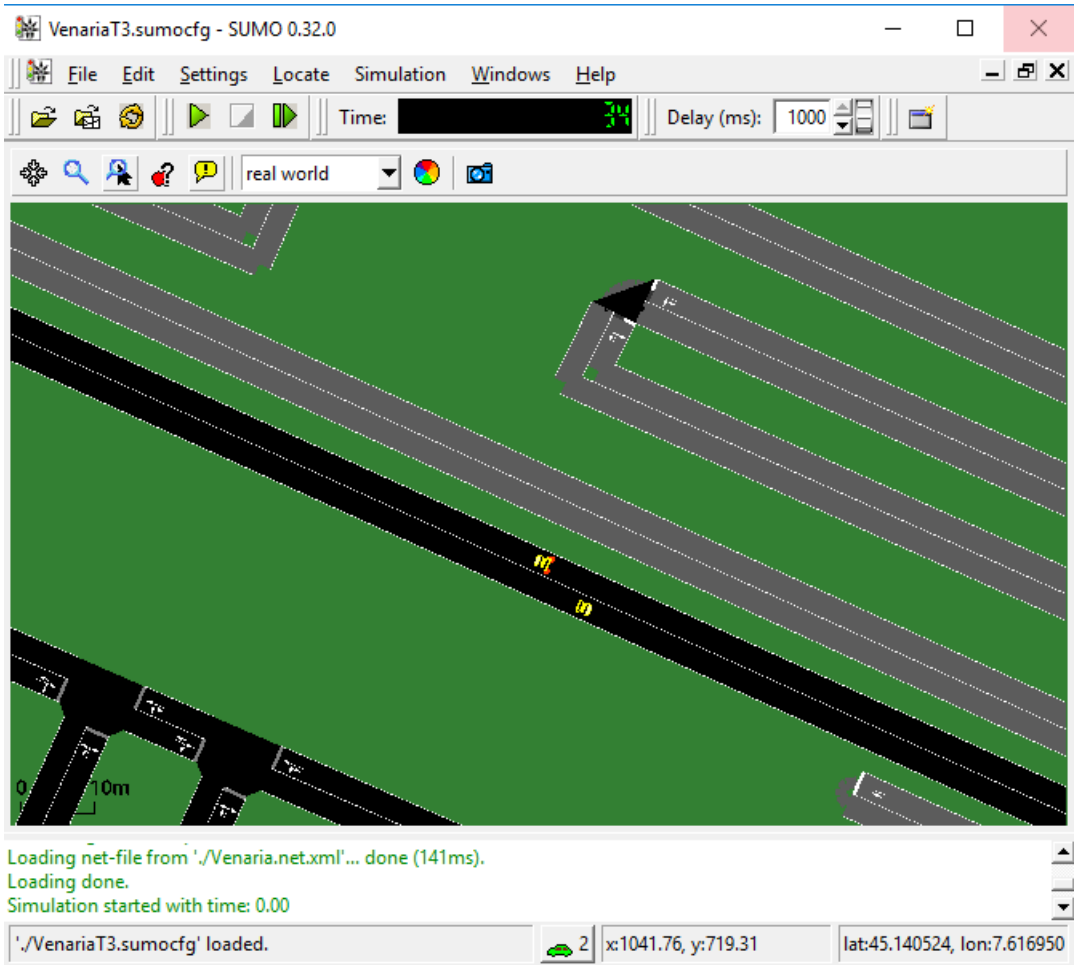

Figure 4.21: Test3, Critical scenario

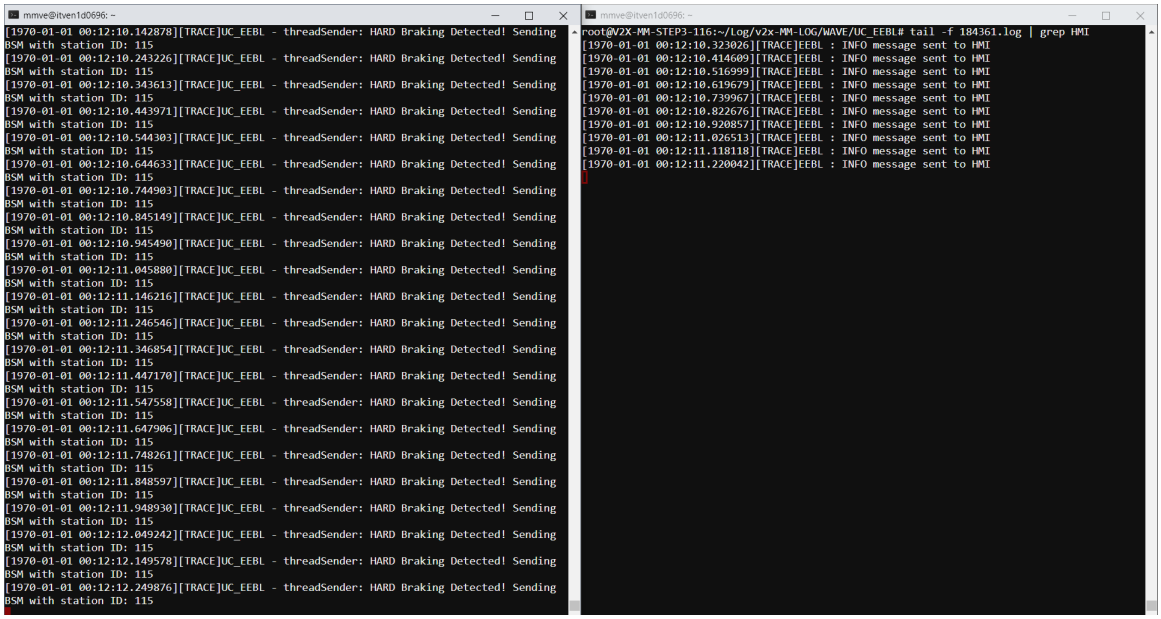

Figure 4.22: Test3

The problem in this scenario is the relative position of the two vehicles, it is a critical scenario because the algorithm "Area of Interest", due to the relative position of the two vehicles considers them as in the same direction. It was interesting to notice the problem in this case because with the tests in the real vehicles there had never been the possibility to fix this bug in the code.

- **Test 4**: EV travelling at medium speed (e.g. 30Km/h), HV is following at the same speed, in the same lane. When the distance among vehicles is upper than 30m, the EV performs the harsh braking. A WARNING MESSAGE APPEARS to the HMI oh HV.
  Expected Result: POPUP with YELLOW background and BRAKE message.

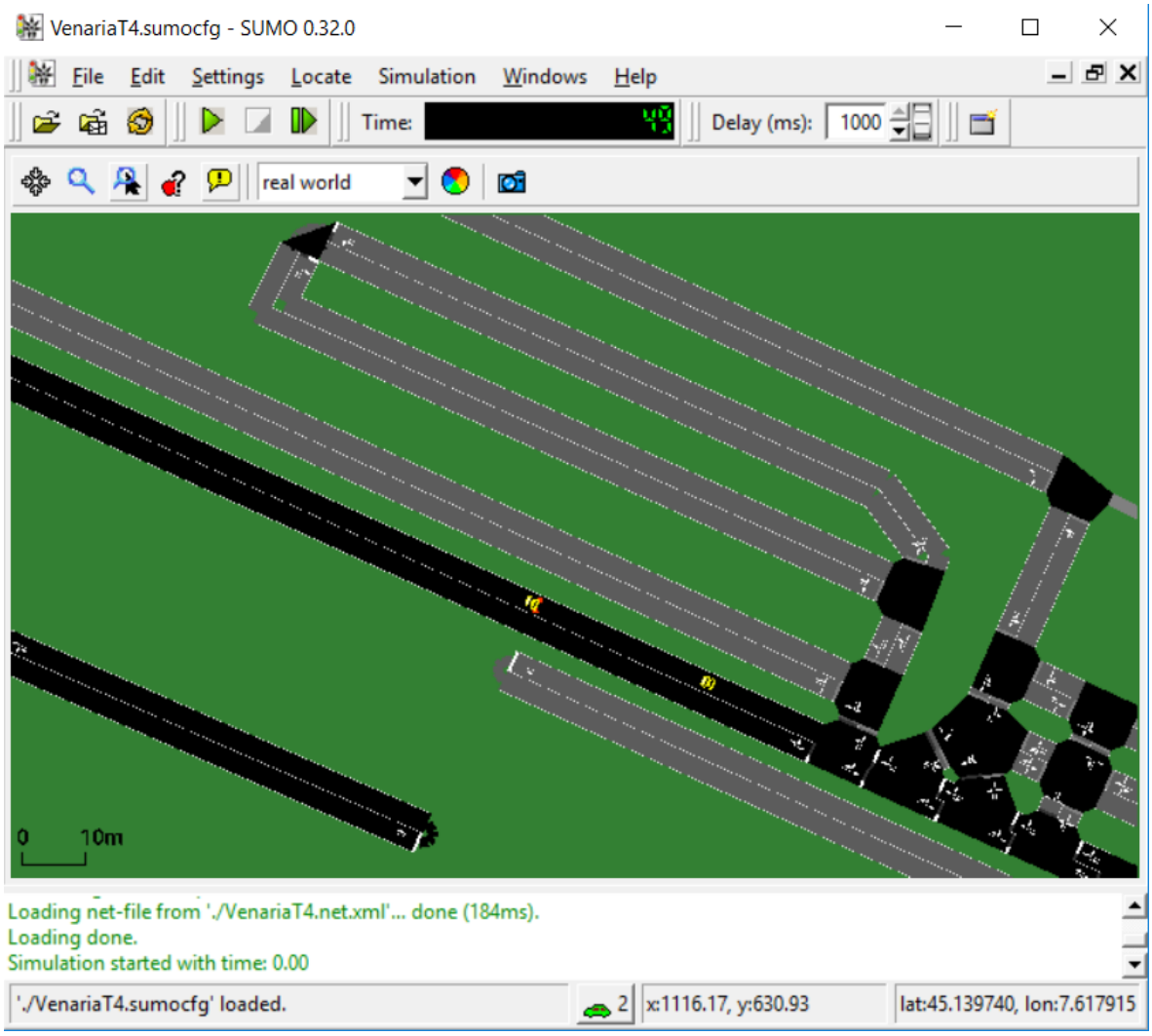

Figure 4.23: Test4 in Sumo

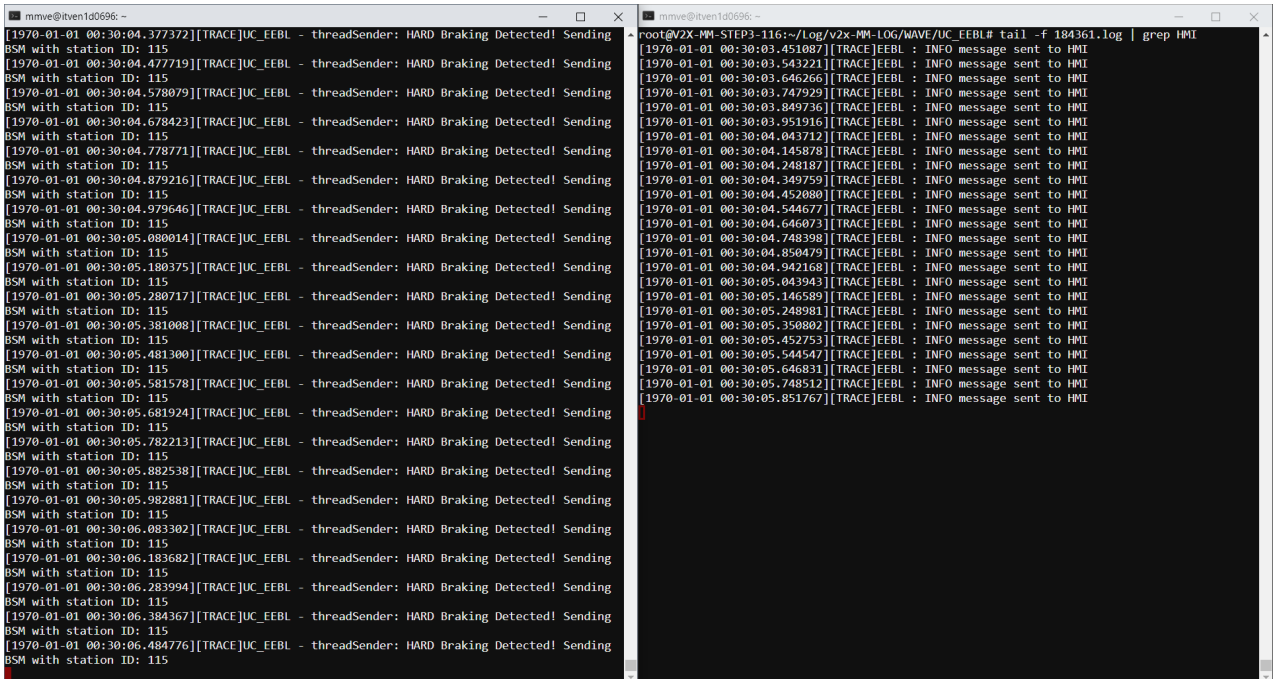

Figure 4.24: Test4

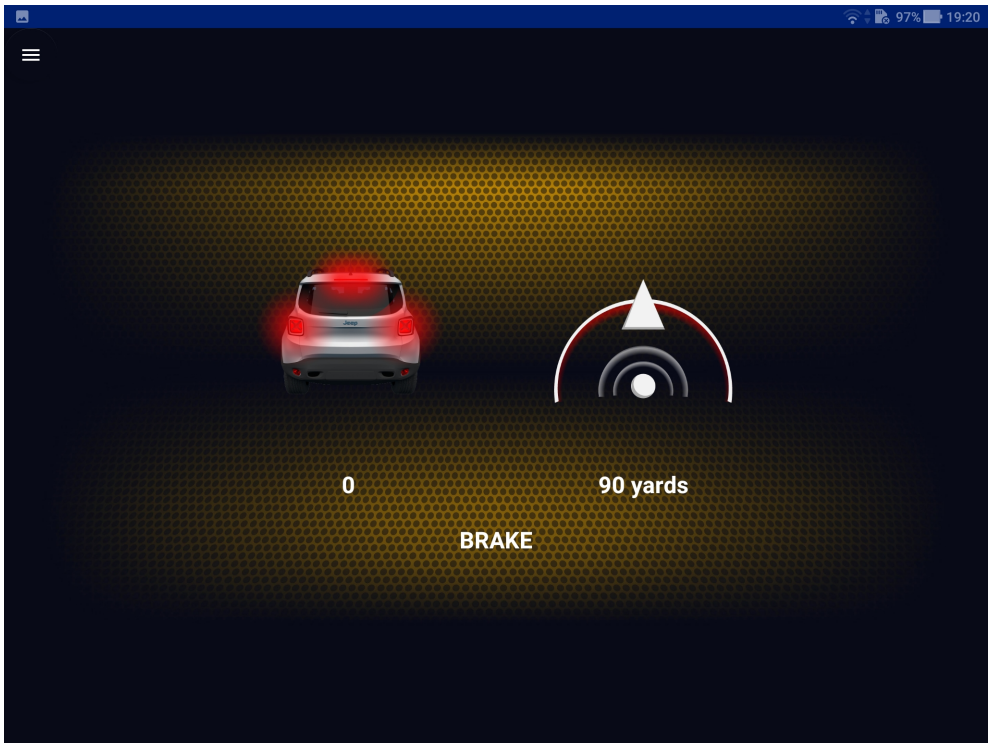

Figure 4.25: Warning message in the V2X HMI App

For this test it has been changed the speed of the vehicles through a modification in the Traffic Demand file, and according to the requirements the Ego vehicle performs the hard braking when the distance among vehicles is upper than 30m. In this case the EEBL event involves the Area of Interest of the Host vehicle, so the warning is notified to the HMI and a confirmation of that is the yellow Popup of the V2X HMI App reported on the tablet.

- **Test 5**: EV travelling at medium speed (e.g. 30Km/h), HV is following at the same speed, in the same lane. When the distance among vehicles is lower than 25m, the EV performs the harsh braking. A CRITICAL MESSAGE APPEARS to the HMI of HV.
  Expected Result: POPUP RED and STOP message, with AUDIO also.

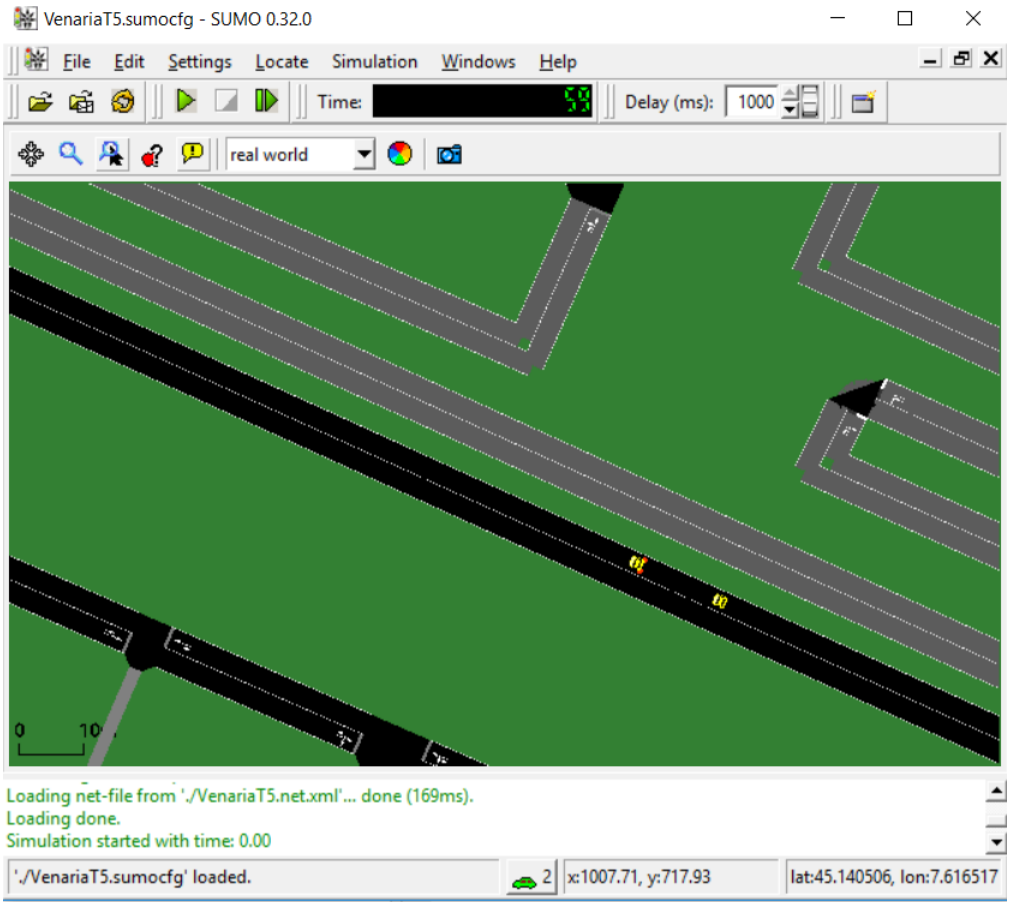

Figure 4.26: Test5 in Sumo

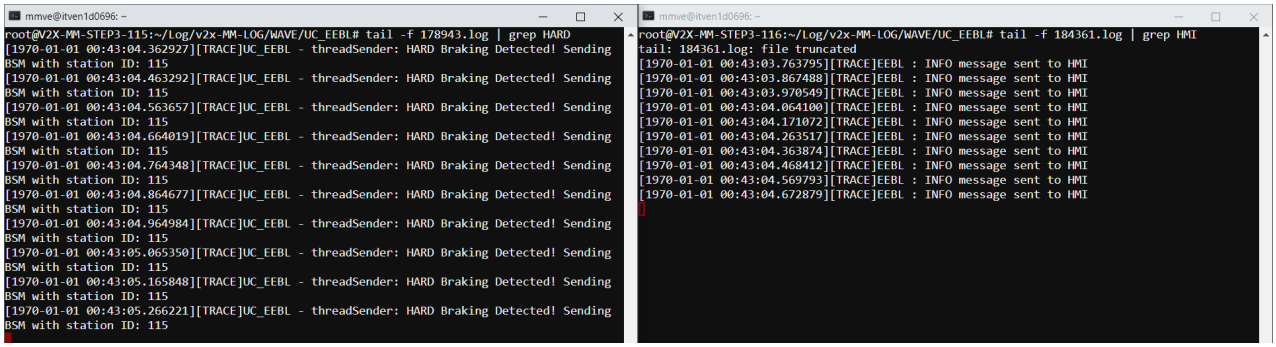

Figure 4.27: Test5

Figure 4.28: Stop message in the V2X HMI App

This case is a critical case in which at the moment of the hard braking the distance among vehicles is very small, so naturally the EEBL event involves the Area of Interest of the Host vehicle and the warning message sent to the HMI is a STOP message highlighted also by the red POPUP in the tablet.

- **Test 6**: EV travelling at medium speed (e.g. 50Km/h), HV is following at the same speed, in the same lane. When the distance among vehicles is upper than 120m, the EV performs the harsh braking. A WARNING MESSAGE APPEARS to the HMI of HV.
  Expected Result: POPUP with YELLOW background and BRAKE message.

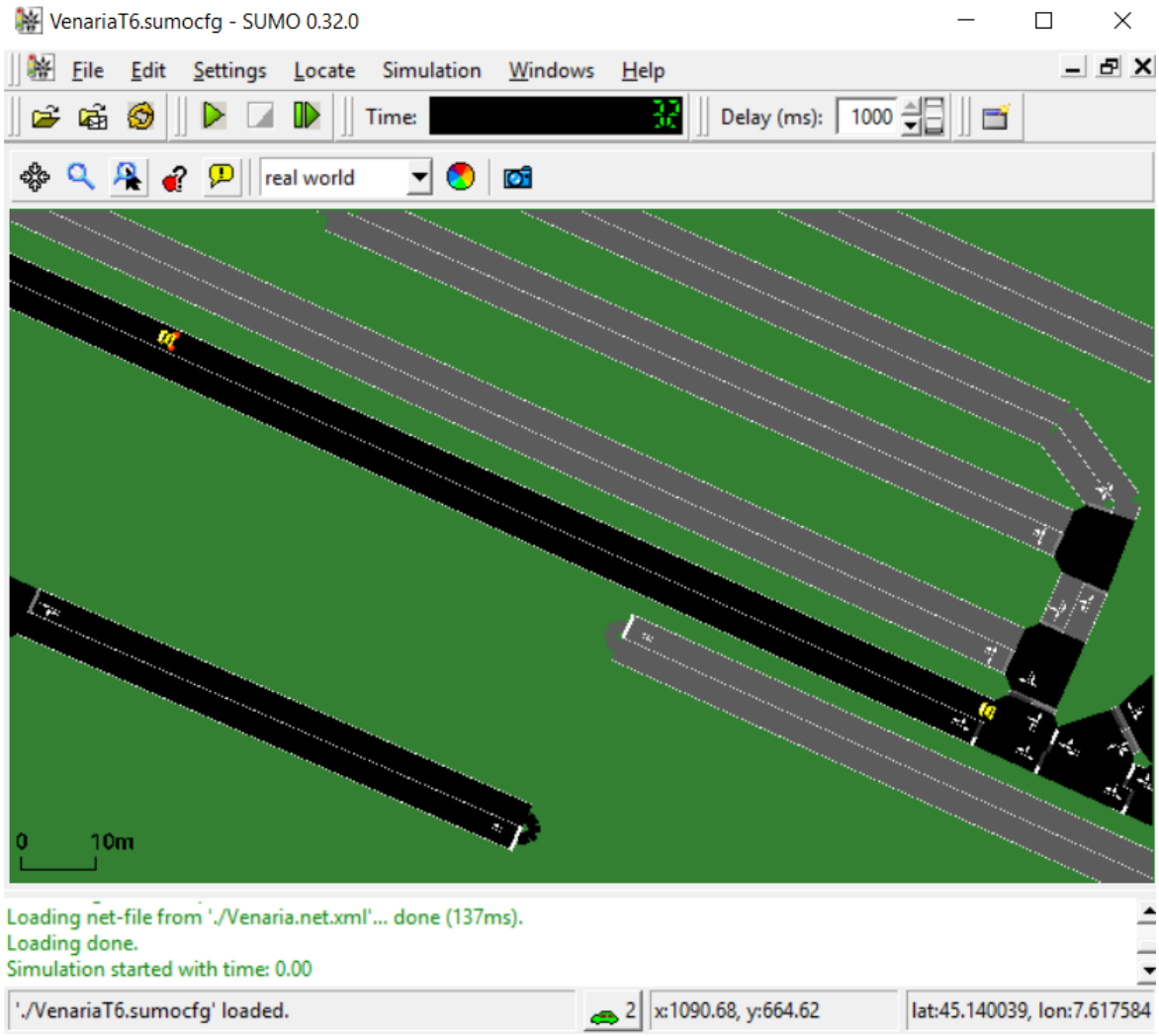

Figure 4.29: Test6 in Sumo

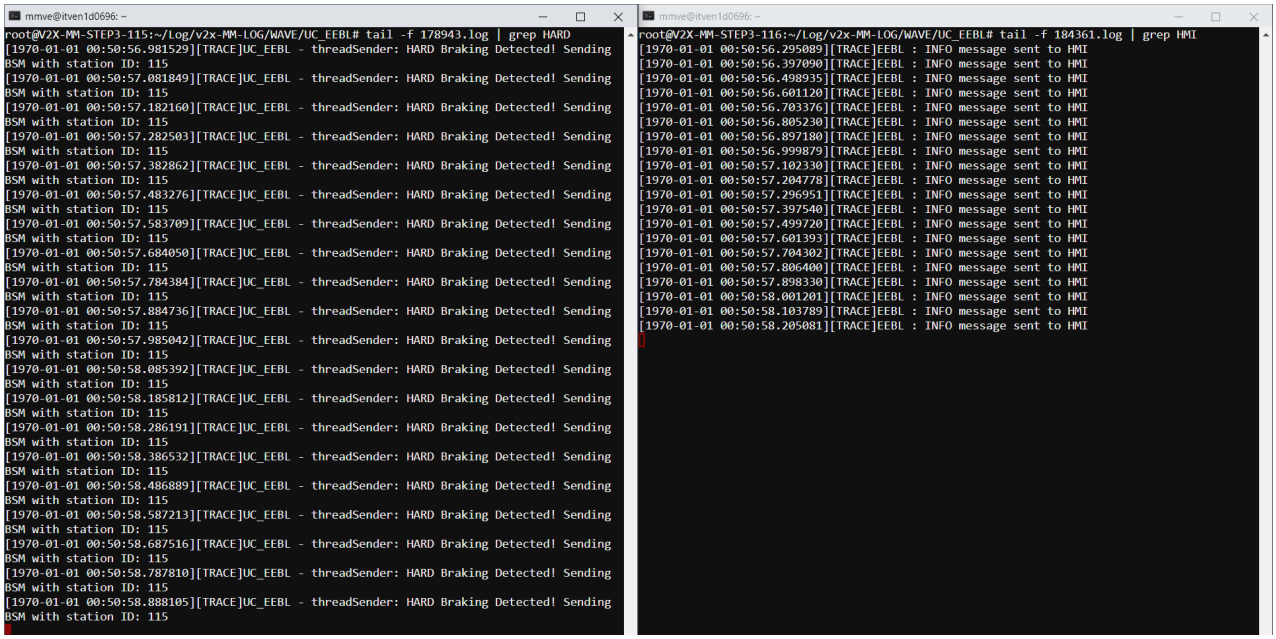

Figure 4.30: Test6

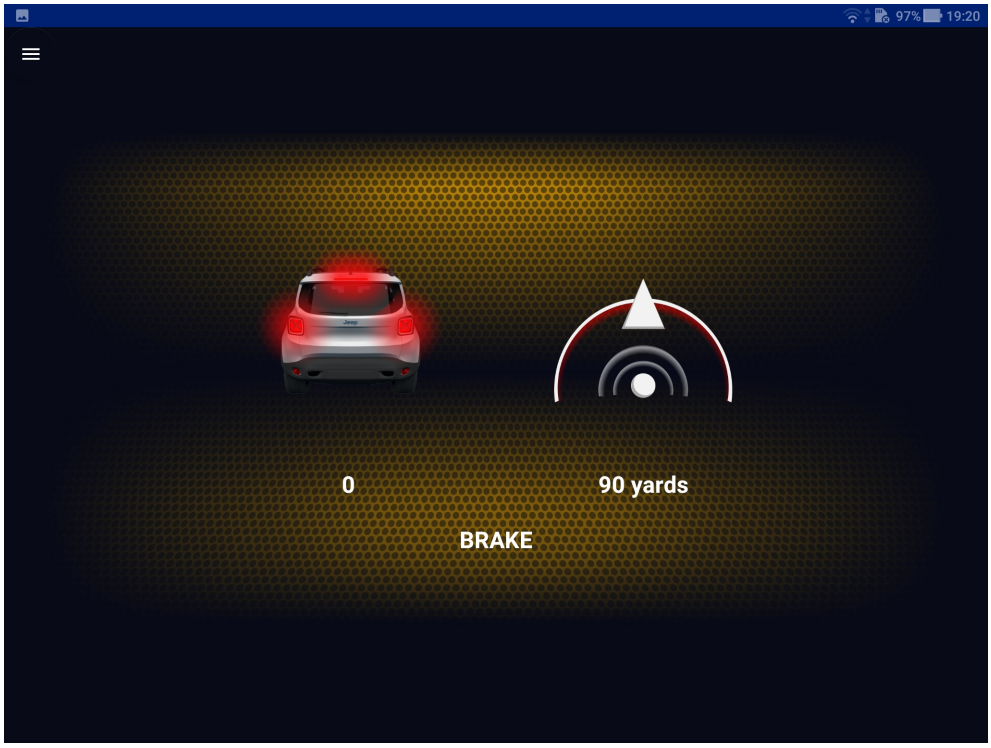

Figure 4.31: Warning message in the V2X HMI App

In this last case the velocity of both vehicles has been changed once again with

the Traffic Demand file and, according with the requirements, the Ego vehicle performs the hard braking when the distance between the two vehicles is upper than 120m. In this case the warning message is sent to the HMI because also in this case the EEBL event involves the Area of Interest of the Host vehicle but the distance between them is not so critical to send a stop message, so a yellow POPUP is shown in the V2X HI App.

Test Number	**Test Result**
1	Passed
2	Passed
3	Passed
4	Passed
5	Passed
6	Passed

Table 4.1: Test result

Chapter 5

Conclusion

In this chapter after a brief recap of the obtained result and of the advantages of the Hardware-in-the-Loop simulation implemented will be shown the further improvements and the possible future developments.

The goal of this thesis work was to create an environment simulation for the testing of the V2X Use Cases.
It has been chosen the Hardware-in-the-Loop simulation because the team have some connectivity boards that allow the V2X communication via DSRC, so the simulation environment aims to create a scenario and to deliver the necessary inputs data through CAN interface on these boards. Avoiding, with this procedure, the in-vehicle driving test that are often expensive, time-consuming and not reproducible. The simulation environment implemented fulfills the requirements of simplicity and repeatability and it lends itself for the creation of V2X scenarios. Moreover the tests realized with this simulation environment reached the expected result giving also the possibility to focus on some critical cases that with in-vehicle driving test could be unnoticed.

5.1 Further Improvements and Future Developments

This work represents the first step for the implementation of more structured simulation environment that could give the opportunity to test all the V2X use cases implemented by the MM connectivity group.
So first of all the work done for the EEBL use case should be done for the rest of the V2V use cases. In addition since Sumo can manage also the semaphores, also the implementation of the V2I use cases should not be a big problem.
The ideal case should be the implementation of one single scenario in SUMO in which there is the possibility to can activate, one by one, all the use cases analyzing the behaviour of the MM DSRC connectivity hardware.

As future developments it might be thought an implementation of the algorithm of the use cases in MATLAB. In this way if the algorithm of the single use case is imported in Matlab, the entire system will be more reliable. The team will have the possibility to have a double check of the expected output.

Bibliography

[1] T.Maszczyk A.Choudhruy. "An integrated simulation environment for testing V2X protocols and applications". In: 2016.

[2] Ahmed-Zaid et al. *Vehicle Safety Communications-Applications (VSC-A) Final Report. Tech. rep.* Tech. rep. NHTSA, Sept. 2011.

[3] *Electronic Design/DSRC vs. C-V2X: Looking to Impress the Regulators. 2017.* URL: http://www.electronicdesign.com/automotive/dsrc-vs-c-v2xlooking-impress-regulators.

[4] *ETSI/Automotive Intelligent Transport System. 2018.* URL: https://www.etsi.org/technologies/automotive-intelligent-transport.

[5] 1609_WG - Dedicated Short Range Communication Working Group. *IEEE Std 1609.1-2006 - Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE).* Tech. rep. VT - IEEE Vehicular Technology Society, 2006.

[6] *Magneti Marelli/ company.* URL: https://www.magnetimarelli.com/company.

[7] *MathWorks/Simulink Documentation.* URL: https://it.mathworks.com/simulink/.

[8] *NHTSA/Vehicle-to-vehicle communication.* URL: https://www.nhtsa.gov/technology-innovation/vehicle-vehicle-communication.

[9] *NI/Controller Area Network (CAN).* URL: http://www.ni.com/white-paper/2732/en/.

[10] M. Schwager J. Neuffer. *Using MATLAB with CANoe.*

[11] *SUMO/userdoc.* URL: https://sumo.dlr.de/userdoc.

[12] *TraCI4Matlab: User's Manual.*

[13] *Vector/CANoe.* URL: URL:www.vector.com/product-a-z/software/canoe.