Politecnico di Torino
Academic Year 2018/2019

Master Thesis

Master in Computer Engineering (Networks)

Green Light Optimal Speed Advisory for Connected Vehicles

Supervisor:
Prof. Gianpiero Cabodi
Supervisor Magneti Marelli:
Ing. Fabio Tosetto

Candidate:
Giuseppe Sturniolo
S232561

Turin, April 2019

Green light optimal speed advisory for connected vehicles

Giuseppe Sturniolo

Supervised by:

Prof. Gianpiero Cabodi

Politecnico di Torino

Ing. Fabio Tosetto

Magneti Marelli

Abstract

The main purpose of this thesis is to develop an embedded software with the aim to suggest to the driver, when it is possible, the optimal speed in order to overcome a traffic light during the green phase.
This software, called **G.L.O.S.A.** (an acronym for Green Light Optimal Speed Advisory), is part of the Magneti Marelli Connectivity Framework , and represents one of the use cases developed by the team to improve driving comfort and safety.
The **G.L.O.S.A.** algorithm takes information provided by traffic lights, other vehicles and the road infrastructure (according to the IEEE WAVE standard for the V2I and V2V communication) and gives an appropriate advice to the driver .

Dedication

I would like to express my gratitude to my family, in particular to my mother, my brother and my father.
They always supported me in my choices and gave me the possibility to achieve this goal.

Contents

1 Introduction **11**
- 1.1 Vehicle-to-Everything Communication . . . 11
- 1.2 Business organisation . . . 12
- 1.3 Goals of the thesis . . . 13

2 Background **15**
- 2.1 Inter-Vehicular Communication Applications . . . 15
- 2.2 The SAEJ2735 . . . 16
  - 2.2.1 The MAP Message . . . 16
  - 2.2.2 The Signal Phase and Timing message (SPAT) . . . 17
  - 2.2.3 The Traveler Information Message (TIM) . . . 17
  - 2.2.4 The Basic Safety Message (BSM) . . . 18
- 2.3 The Magneti Marelli Connectivity Framework . . . 18
  - 2.3.1 The Middleware layer . . . 19
  - 2.3.2 The Facilities layer . . . 20
  - 2.3.3 The Applications layer . . . 21

3 Implemented Background **23**
- 3.1 Software Instruments . . . 23
  - 3.1.1 The C++ language . . . 23
  - 3.1.2 Makefile/cmake . . . 23
  - 3.1.3 Cross-Compiling . . . 24
  - 3.1.4 Apache Subversion (SVN) . . . 24
  - 3.1.5 Matlab . . . 25
  - 3.1.6 Simulink . . . 25
  - 3.1.7 Enterprise Architect . . . 25
  - 3.1.8 SUMO (Simulation of Urban Mobility) . . . 26
- 3.2 The MM Connectivity hardware . . . 27
  - 3.2.1 The Magneti Marelli Step 03 board . . . 27
  - 3.2.2 MK5-Cohda Wireless OBU . . . 28
  - 3.2.3 Road Side Unit . . . 30

4 State of art **32**
4.1 GLOSA existing versions . . . 32
4.1.1 Single segment GLOSA . . . 32
4.1.2 Multi segments GLOSA . . . 34

5 Software : Use Case GLOSA **38**
5.1 GLOSA overview . . . 38
5.2 Use Case GLOSA one vehicle version . . . 40
5.2.1 Algorithm . . . 41
5.2.2 Simulation on Simulink . . . 57
5.2.3 C++ implementation . . . 62
5.3 Use Case GLOSA more vehicles version . . . 66
5.3.1 Algorithm . . . 67
5.3.2 Simulation on Simulink . . . 70
5.3.3 C++ implementation . . . 74

6 Testing and Validation **78**
6.1 Testing e Validation . . . 78
6.1.1 Bench tests . . . 78
6.1.2 On road tests . . . 84

7 Results **90**
7.1 Results . . . 90

8 Future improvements **92**
8.1 Future improvements . . . 92

List of Figures

1.1 V2X communication 12

2.1 Comparison between IEEE WAVE (left) and ETSI ITS station (right) stacks 16
2.2 Map Data Structure according SAEJ2735 17
2.3 Spat Data Structure according SAEJ2735 18
2.4 Bsm Data Structure according SAEJ2735 19
2.5 Framework V2X Magneti Marelli 20
2.6 Middleware layer 20
2.7 Facilities layer 20
2.8 Applications layer 22

3.1 A Subversion example schema. 25
3.2 The step 03 board 27
3.3 MK5-Cohda Wireless OBU 29
3.4 RSU example 30
3.5 RSU schema 31

4.1 Glosa algorithm of the University of Surrey Guildford 33
4.2 Distance Model to control updating of suggested speed for human drivers 34
4.3 Decision tree to determine the message generated to advice the human driver on his speed decision based on the traffic light status 35
4.4 fuel consumption model for a single intersection 35
4.5 total fuel consumption model for N intersections 36
4.6 B&B algorithm representation 36

5.1 Logical Model of one vehicle version 40
5.2 Flowchart of first step of GLOSA 43
5.3 Flowchart of the general case of green phase 44
5.4 Flowchart of the first case of green phase 45
5.5 Flowchart of the second case of green phase 46
5.6 Flowchart of the third case of green phase 47

5.7 Flowchart of yellow phase . . . 48
5.8 Flowchart of red phase . . . 49
5.9 Flowchart of the first case during red phase . . . 50
5.10 Flowchart of the second case during red phase . . . 51
5.11 Flowchart of the third case during red phase . . . 52
5.12 Flowchart of the fourth case during red phase . . . 53
5.13 Flowchart of the fifth case during red phase . . . 54
5.14 Flowchart of the sixth case during red phase . . . 55
5.15 Representation of the 4 possible strips for hysteresis . . . 56
5.16 Simulink model overview . . . 57
5.17 Focus on input simulator block . . . 58
5.18 Focus on GLOSA block . . . 59
5.19 Focus on UcManager simulator block . . . 59
5.20 Focus on driver response simulator block . . . 60
5.21 Focus on UDP_SOCKET_SEND_and_RECEIVE block . . . 61
5.22 Focus on main plot block . . . 61
5.23 Focus on settings block . . . 62
5.24 Glosa class and its main parts . . . 63
5.25 BPMN of the two threads of one vehicle version of GLOSA . . . 64
5.26 Vehicles distance . . . 66
5.27 Logical model of UC-GLOSA more vehicles version . . . 69
5.28 Simulink model of UC-GLOSA more vehicles version . . . 70
5.29 OTHER_VEHICLE_DYNAMIC Simulink block . . . 71
5.30 FCW_BLOCK Simulink block . . . 71
5.31 SETTINGS_BLOCK Simulink block . . . 72
5.32 EGO_VEHICLE_SUBSYSTEM Simulink block . . . 73
5.33 OTHER_FCW_PLOT Simulink block . . . 73
5.34 GLOSA class more vehicles version . . . 74
5.35 BPMN more vehicles version . . . 76

6.1 Example of output on a Simulink scope . . . 79
6.2 Legend of a Simulink scope . . . 79
6.3 Example of a test case using only Simulink . . . 80
6.4 Schema of the bench tests using both Simulink and C++ . . . 81
6.5 TEST case 2 Simulink and C++ . . . 82
6.6 C++ output on Simulink example . . . 83
6.7 Simulink output example . . . 83
6.8 Map configuration for tests . . . 84
6.9 Map areas in Viale Venaria . . . 85
6.10 Hardware schema for on road test . . . 88
6.11 UC-GLOSA log example . . . 89

7.1 Glosa output on tablet in case of green window 91
7.2 Glosa output on tablet in case of critical stop 91

List of Tables

5.1 Table of GLOSA input 41
5.2 Table of GLOSA output 41
5.3 Table of GLOSA input (more vehicles) 67
5.4 Table of GLOSA output (more vehicles) 67

List of Acronyms

V2X *Vehicle-To-Everything*

V2V *Vehicle-To-Vehicle*

V2I *Vehicle-To-Infrastructure*

V2N *Vehicle-To-Network*

V2D *Vehicle-To-Device*

V2P *Vehicle-To-Pedestrians*

V2G *Vehicle-To-Grid*

LTE-V2X *Long-Term Evolution Vehicle-to-Everything*

C-V2X *Cellular Vehicle-To-Everything*

DSRC *Dedicated short-range communications*

IVC *Inter-Vehicular Communication*

RSU *Road Side Unit*

OBU *On-Board Unit*

C-ITS *CooperativeIntelligent Transport Systems*

ETSI *European Telecommunications Standards Institute*

FCC *Federal Communi-cations Commission*

WAVE *Wireless Access in Vehicular Envi-ronments*

CAN *Controller Area Network*

API *Application Programming Interface*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

GPS *Global Positioning System*

LDM *Local Dynamic Map*

POTI *Positioning Timing*

VDP *Vehicle Data Provider*

SPAT *Signal Phase and Time*

HMI *Human Machine Interface*

BSM *Basic Safety Message*

TIM *Traveler Information Message*

WSA *WAVE Service Advertisement*

CAM *Cooperative Aware Message*

DENM *Decentralized Enviromental Notification Message*

SV *Stationary Vehicle*

FCW *Forward Collision Warning*

EEBL *Electronic Emergency Brake Ligh*

GLOSA *Green Light Optimised Speed Advise*

IMA *Intersection Movement Assist*

CLW *Control Loss Warning*

LTA *Left Turn Assist*

BSW *Blind Spot Warning*

IVRS *In-Vehicle Road Sign*

Chapter 1

Introduction

1.1 Vehicle-to-Everything Communication

In the scenario of modern technological innovation, vehicular networks assume an increasingly important significance as regards the development of transport. In fact, nowadays, mobility plays a central role in the social system and is continuously growing worldwide.
However,travel could cause problems related to traffic efficiency, safety and pollution.
According to the World Health Organization (WHO), the World Health Organization, road accidents each year cause around 1.2 million deaths and around 50 million injured worldwide. If preventive measures are not taken, road death is destined to become the third leading cause of death in 2020 from ninth place in 1990 [1].

At the same time, this sector is responsible for about 33% of final energy consumption, and therefore represents an increasingly central element in European policies to fight climate change and reduce pollution in urban areas. European statistics referring to 28 member countries show that as much as 30.4% of greenhouse gases and 30.5% of carbon dioxide emissions, as well as a considerable part of urban air and noise pollution, are attributable to transport. These values for Italy increase to around 34%[2].
Vehicular Networks could represent a solution to fix these issues thanks to the creation of smart vehicles that are able to communicate with other vehicles and also with the infrastructure.All this features are possible through the exchange of message between all the actors involved in a travelling environment, in order to discover what is happening around a vehicle.
The vehicular communication involves the exchange of many messages that contain information on a large number of details concerning the studied scenario.

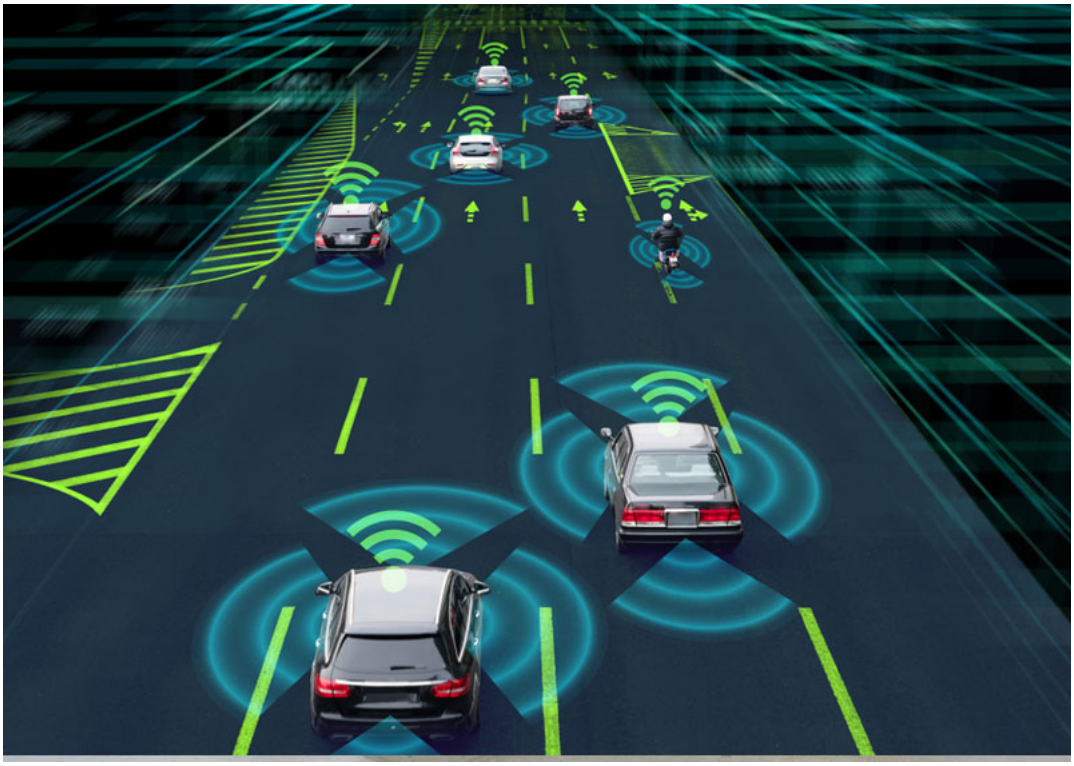


Figure 1.1: V2X communication

Currently, two standards are available to provide communications in support of transportation efficiency, safety and sustainability: European ETSI and United States IEEE WAVE. These two standards are similar in some ways, but very different from the point of view of the multi-channel management.
The framework developed by the Magneti Marelli Innovation Connectivity team takes information by all these messages and use them for various use cases, in order to help the driver in some different situations.

1.2 Business organisation

Magneti Marelli is an italian multinational, founded in **1919** as result of a joint-venture between FIAT and Ercole Marelli [3]. It is specialized in the supply of high technology products and systems for the automotive industry.

It began as a producer of magnets for both aviation and automotive and motorcycle engines, but it gradually extended its activities in the field of embedded systems , developing and producing embedded platforms for several purposes, in particular as regards the automotive world.

In Magneti Marelli there are also several Research & Development divisions such as the **Innovation Connectivity** and the Autonomous Driving ones. The Innovation Connectivity team is working to develop a solution for enabling Vehicle-to-Everything (V2X) communication.

1.3 Goals of the thesis

One of the use cases developed by the team is the "TRAFFIC LIGHT":
This use case was developed only to show to the driver the current phase of a traffic light and the respective countdown to the next phase.
The aim of this thesis is to build a software, integrated in the Magneti Marelli Connectivity Framework, that takes information from other connected actors (traffic lights, other vehicles and infrastructures) in order to suggest a range of speeds that permit to overcome a traffic light during the green phase, if it is possible.
This software is called **G.L.O.S.A.**, and its building consisted of several steps:
The first one has been to choice an appropriate algorithm that takes into account as much information as possible in order to make the result more accurate.
Then, there was the implementation of this algorithm in a testing environment, to check its validity and to correct any errors.
The third step has been the implementation of the C++ version in order to be deployed on the custom **ARM**-based Magneti Marelli Connectivity Board(nicknamed **Step 03**).
This process has been followed by the validation of the C ++ version of the software using bench-based inputs. Finally, there are road tests to verify if the result produces a pleasant driving experience.

Chapter 2

Background

2.1 Inter-Vehicular Communication Applications

Inter-Vehicular Communication (IVC [4]) represents an important component of the ITS (Intelligent Trasportation System) architecture.
Its main feature is to ensure communication between the driver, or his vehicle, with the other drivers, or their vehicles, which are outside the LOS (line of sight).
The architecture of an IVC can be classified into three large types:

- The first one is the cellular or WLAN network in which the gateways of the the mobile network or access points of the WLAN to connect to the Internet.However, this typology can not be implemented on a large scale because it has high costs and there are geographical limitations.

- The second one is the ad-hoc network without infrastructure in which communication V2V is built using DSRC communication, Dedicated Short-Range Communication [5], and each vehicle is equipped with a wireless network device.

- The third one is hybrid because it has both the characteristics of a cellular network WLAN / WLAN that those of an ad-hoc network. Vehicles use the Infrastructure Unit to access dynamic information exchanged outside its own range and share information via a V2V communication without infrastructure.

Different kinds of applications for IVC are identified by The Vehicle Safety Communication (VSC) consortium: The most relevant concerns the Road Safety and involves the implementation of a lot of use cases: Warning, Forward Collision,Intersection Violation Warning, Blind Spot Warning, Icy Road Warning, and Longitudinal Collision Risk Warning. The main purposes is increasing the transportation efficiency and the road safety in order to reduce the impact of transportation on the environment. To implement all these applications it is mandatory to have

two categories of messages, called periodic and the event-driven messages exchanged in the fastest possible way. The periodic messages are sent with a certain frequency, instead the event-driven ones are messages activated from particular events, like an emergency event , and it must be transmitted immediately on the channel.
IVC systems are based on two standards : **IEEE** for the American Standard [6] and **ETSI** for the European Standard [7].
These two standards implement the same concept but with different architectures. In particular,IEEE defines a system architecture for IVC called wireless access in vehicular environments(**WAVE**) architecture.

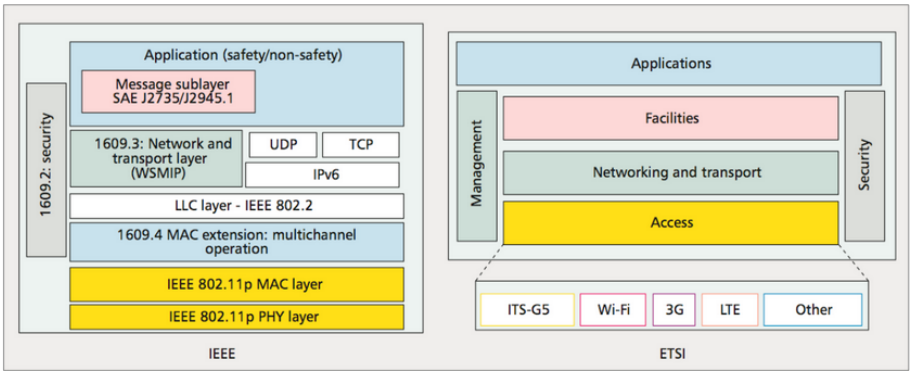


Figure 2.1: Comparison between IEEE WAVE (left) and ETSI ITS station (right) stacks

This thesis is focused on the application layer, which consist of applications able to elaborate and use messages exchanged between nodes and described in the SAEJ2735 (as regards the WAVE architecture). Thanks to the SAEJ2735 is possible to develop a lot of use cases for the safety of the driver.

2.2 The SAEJ2735

The main purpose of the SAEJ2735 Standard is to support interoperability among DSRC applications through the use of a standardized message set, and its data frames and data elements. The most important messages defined in this standard are the Basic Safety Message (BSM), Signal Phase and Timing (SPAT), Map Data (MAP) and Traveler Information Message(TIM).

2.2.1 The MAP Message

The **MAP** message contains information about the topology of an intersection and it is sent every 5 second. The structure of this message is the following:

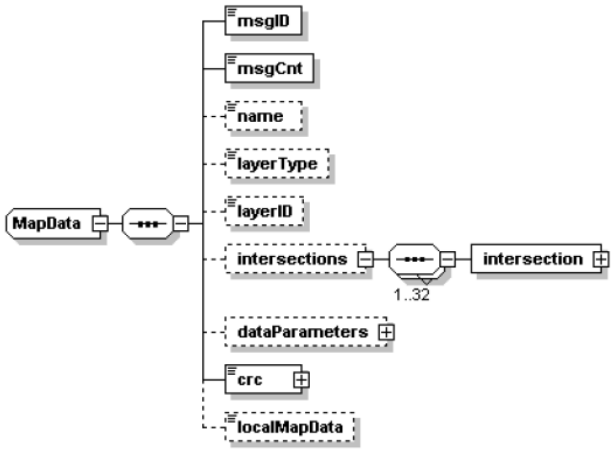


Figure 2.2: Map Data Structure according SAEJ2735

The rectangles of the figure are the different fields of the packet.Some fields must be always present in the message, others are not mandatory.
A single MAP message can transmit descriptions of one or more intersections or geographical areas. The content of this message is often used by other modules to generate new messages.For instance, the SPAT message needs some of the MAP information to define in which lane a certain semaphore is located.

2.2.2 The Signal Phase and Timing message (SPAT)

The **SPAT** message is used to transmit information about the current state of one or more intersections.If combined with a MAP type message, the receiver can determine the signal status and the relative timing.

The SPAT message sends the current status of each active phase of the traffic light and the countdown to the next phase. Usually, information on inactive phases is not transmitted. The overall use of the SPAT message is to reflect the current state of all lanes in all approaches in single intersection. Any preemption or priority then follows in a structure for the whole intersection.

2.2.3 The Traveler Information Message (TIM)

The Traveler information Message provides information gives information about the traffic status or road signs. It is mostly used to communicate the street condition. For example, it is useful to know the current legal speed limit of the road near an intersection.

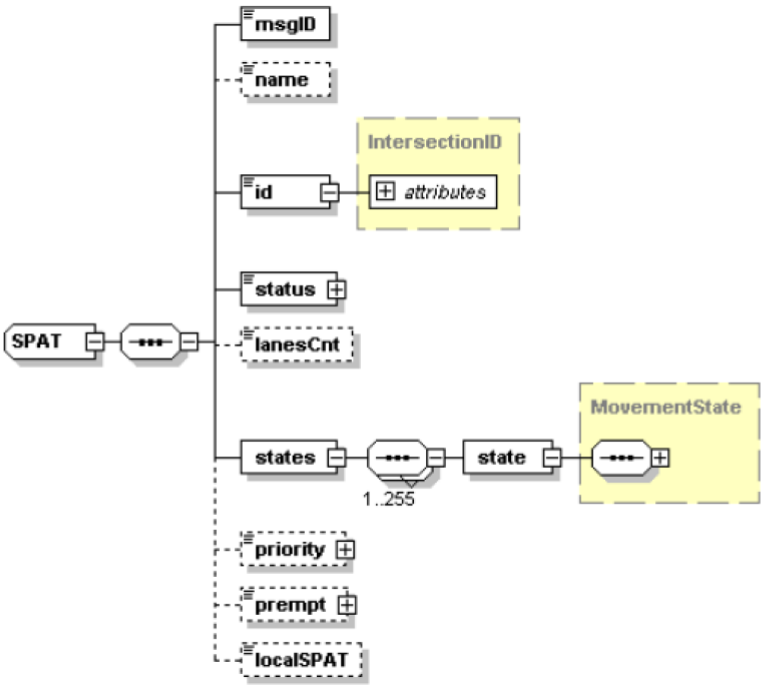


Figure 2.3: Spat Data Structure according SAEJ2735

2.2.4 The Basic Safety Message (BSM)

The Basic Safety Message (figure 2.4) belongs to the American standard and it is used in a variety of applications to exchange information related to both safety and vehicle status. It is a message that is sent with an high frequency, up to ten times per second.
In the following figure is showed its structure:
It is divided into two parts:

- The first part contains data about the vehicle, retrieved from GPS and CAN network. The most important fields are the Message Sequence Number identifying a packet sent from a vehicle and the Temporary ID identifying the Vehicle.
- The second part is composed by four modules that contain additional information.

2.3 The Magneti Marelli Connectivity Framework

The Magneti Marelli Connectivity Framework was designed with the aim to unify the the American standard (WAVE) with the European standard (ETSI) for vehicular

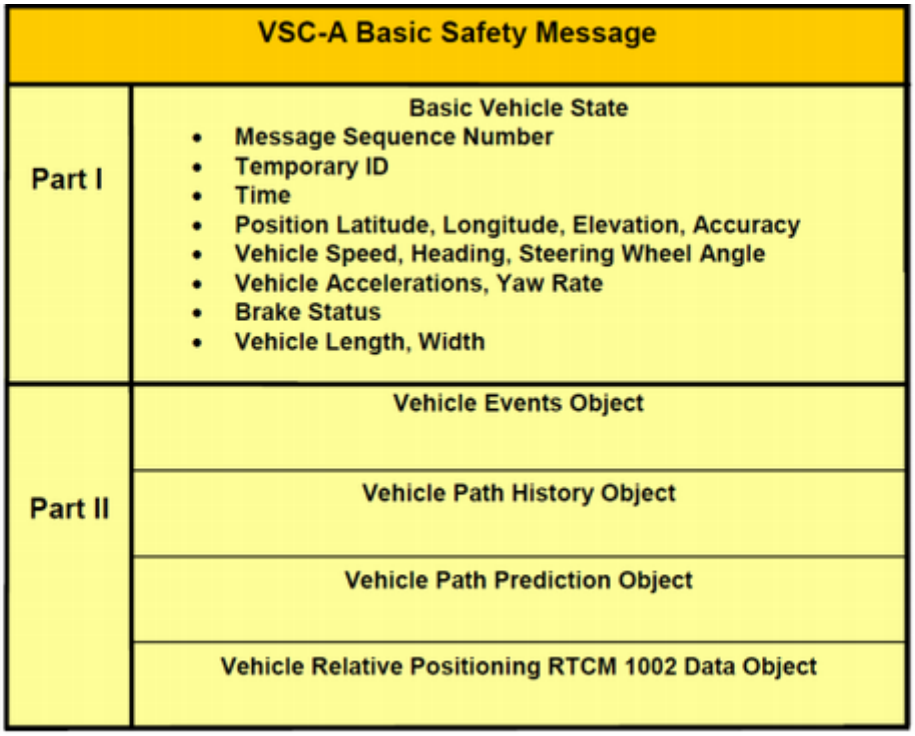


Figure 2.4: Bsm Data Structure according SAEJ2735

communication. Three software groups can be identified:

- Those that refer only to the European standard, such as DENM and CAM, decoded according to the standard EN 302 637-2 [8] and EN 302 637-3 [9].
- Those that belong to the American standard (eg BSM, TIM).
- Those that are in common with the two standards (eg LDM, MAP, HMI, SPAT).

As shown in the figure, it is divided into three large layers: middleware, facilities and applications.

2.3.1 The Middleware layer

The **middleware** (figure 2.6) is a layer that creates an interface in order to make the upper layers independent from the driver chosen at the lowest level.This make the software adaptable to any V2X stack that will be used (Cohda Wireless, Autotalks).

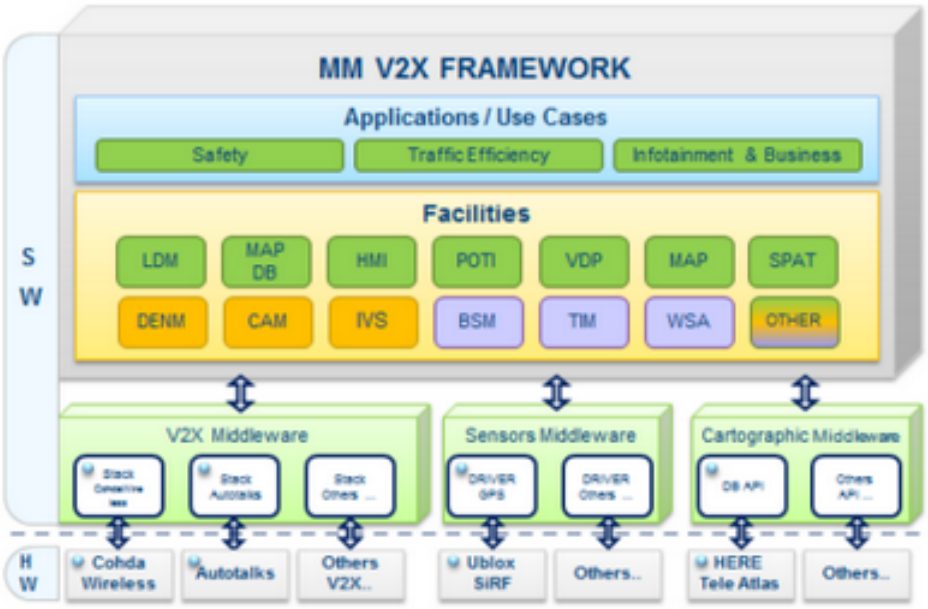


Figure 2.5: Framework V2X Magneti Marelli

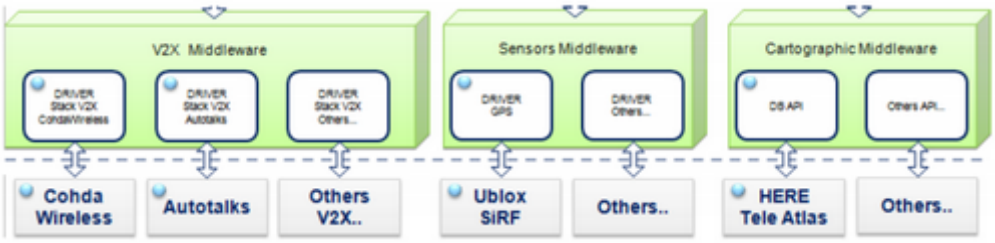


Figure 2.6: Middleware layer

2.3.2 The Facilities layer

The **Facilities layer** (figure 2.7) contains most of the software modules that compose the Framework.

Figure 2.7: Facilities layer

The most important modules of this layer are:

- **Local Dynamic Map (LDM)** is a software module that cooperates with intelligent transport systems. Its main function is to manage the information that influences or is influenced by road traffic, in particular the sources that generate data can be different as vehicles, external infrastructures, on-board sensors, signals. The LDM module is the hub of the framework, from which the vast majority of information passes.

- **Positioning and Timing (POTI)** provides time information and 3D position information (latitude, longitude, altitude). POTI processes and merges data received from sensors, such as GPS and other data from different vehicle sensors in order to obtain location information on the mobile station. It also allows to synchronize the timing of the system thanks to time information taken from the satellites.

- **Vehicle Data Provider (VDP)**is a module connected to the entire vehicle network and provides information on the status of the same (speed,longitudinal acceleration,braking status, active arrows).

- **The Human Machine Interface (HMI)** is a link between the driver and all the possible messages that the vehicle can communicate to it.The HMI can exchange information both in a monodirectional and bidirectional way, according to the application needs. It can be a dynamic (following a priority) or static information handler , such as the emergency level of information such as reporting of collision risk, traffic status information.

- **The MAP-BS** is the module responsible to process MAP in the MM V2X Framework.

- **The SPAT-BS** is the module responsible to process SPAT in the MM V2X Framework.It also merges some information from MAP messages(taken from MAP-BS) and from SPAT messages, generating a new message type, the **SPATMAP**.This is the message actually used by UC-GLOSA.

- **The BS-Basic Service (BS-BS)** is the module responsible to process BSM in the MM V2X Framework.

2.3.3 The Applications layer

The **Applications layer** (figure 2.8) consists of several applications that represent the use cases, that are a practical example of all the possible features of the framework.
There are three different types of use cases:

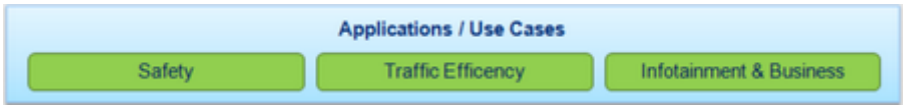


Figure 2.8: Applications layer

The first one is **Safety** and concerns road safety. In fact, when an obstacle is present on the roadway or when an accident has occurred, vehicles may receive alarm messages from other vehicles, even in automatic mode.

The second is **Traffic Efficiency** and concerns traffic management. The introduction of real intelligent traffic lights would reduce the long waits at an intersection.As a matter of fact, these traffic lights could take information about traffic near the intersection and so regulate their red-green cycles.The **UC-GLOSA** is part of this category of applications.

The third is **Infotainment & Business** and provides information of different kinds: tourist (like restaurants, cinemas, museums) or services, such as hospitals, car parks in the area crossed by the vehicle.

Chapter 3

Implemented Background

3.1 Software Instruments

3.1.1 The C++ language

The **C++** is an object-oriented programming language. It has been developed as a **C** language improvement through the introduction of the object-oriented programming paradigm, operator overloading, virtual functions,template, multiple inheritance and exception handling.
This makes possible to develop software following the most modern design patterns, thanks to many ready and reusable libraries that can be integrated with the projects.
C++ compilers directly to a machine's native code, allowing it to be one of the fastest languages in the world, if optimized. There are C++ compilers for all platforms and operating systems,so it is possible to recompile a software and running it in different application contexts. This is due to the fact that C ++ is defined as standard [10].

3.1.2 Makefile/cmake

Makefiles are special format files that together with the **make** utility help to build and manage the project.
The utility is mainly used for compiling source code in object code, joining and then linking the object code in executable programs or in libraries.
Makefiles are useful to determine the dependencies graph for a particular output the scripts necessary for the compilation to be passed to the shell.

3.1.3 Cross-Compiling

C++ is a portable language, so it can be compiled for different architecture. A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. The fundamental use of a cross compiler is to separate the build environment from target environment. It is necessary to compile for multiple platforms from one machine. A platform could be infeasible for a compiler to run on, such as for the microcontroller of an embedded system because those systems contain no operating system. GCC, a free software collection of compilers, can be set up to cross compile. It supports many platforms and languages, in particular G++ is one of the version of GCC to compile C++. In this thesis embedded system with ARM processor and Laptop X86 processor has been used, so Cross-compiling is necessary to execute the same programs on both platforms [11].

3.1.4 Apache Subversion (SVN)

Apache Subversion (often abbreviated SVN, after its command name svn) is a software versioning and revision control system distributed as open source under the Apache License.
Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly compatible successor to the widely used Concurrent Versions System (CVS).
The open source community has used Subversion widely: for example in projects such as Apache Software Foundation, Free Pascal, FreeBSD, GCC and SourceForge. CodePlex used to offer access to Subversion as well as to other types of clients.
The main features of SVN are:

- **Folder version**: the content of each directory is also tracked and therefore the movement of a file is considered a modification, therefore traceable and reversible.

- **Atomic commits**: a series of changes is applied only in bulk and if even one of them creates errors, none of the changes sent are applied. This avoids the creation of incomplete versions that should then be corrected by hand.

- **Metadata version**: it is possible to assign custom properties and files to directories and files and to keep the history of changes to these properties.

This tool has been used very a lot on this thesis to work with the Magneti Marelli Venaria Team, in fact thanks to SVN, sharing the code has been very easy [12].

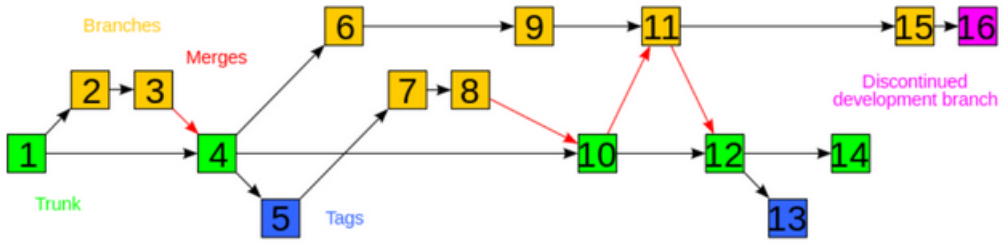


Figure 3.1: A Subversion example schema.

3.1.5 Matlab

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.
Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems [13].

3.1.6 Simulink

Simulink, developed by MathWorks, is a graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in automatic control and digital signal processing for multidomain simulation and Model-Based Design[14].

3.1.7 Enterprise Architect

Sparx Systems Enterprise Architect is a visual modeling and design tool based on the OMG UML. The platform supports: the design and construction of software systems, modeling business processes and modeling industry based domains. It is used by businesses and organizations to not only model the architecture of their systems, but to process the implementation of these models across the full application development life-cycle. Systems modeling using UML provides a basis for modeling all aspects of organizational architecture, along with the ability to provide a foundation for designing and implementing new systems or changing existing

systems. The aspects that can be covered by this type of modeling range from laying out organizational or systems architectures, business process re-engineering, business analysis, and service-oriented architectures and web modeling, through to application and database design and re-engineering, and development of embedded systems. Along with system modeling, Enterprise Architect covers the core aspects of the application development life-cycle, from requirements management through to design,construction, testing and maintenance phases, with support for traceability, project management and change control of these processes, as well as, facilities for model driven development of application code using an internal integrated-development platform [15].

3.1.8 SUMO (Simulation of Urban Mobility)

SUMO (Simulation of Urban MObility) is a free, open, microscopic and continuous road traffic simulation suite designed to handle large road networks. It allows modelling of intermodal traffic systems including road vehicles, public transport and pedestrians. SUMO includes a wealth of supporting tools, which handle tasks such as route finding, visualisation, network import and emission calculation. SUMO can be enhanced with custom models and provides various APIs to remotely control the simulation.

SUMO offers various features such as:

- Microscopic simulation.
- Online interaction.
- Multimodal simulation incl. vehicles, public transport, pedestrians.
- Automatic generation of time schedules of traffic lights.
- No limitations in network size and number of simulated vehicles.
- Evaluation of eco-aware routing based on pollutant emission and investigations of autonomous route choice on the overall network.

SUMO supports the import formats OpenStreetMap, VISUM, VISSIM and NavTeq. SUMO is implemented in C++ and uses only portable libraries. The SUMO package contains a number of different applications [16].

3.2 The MM Connectivity hardware

3.2.1 The Magneti Marelli Step 03 board

The Step 03 board (figure 3.2) is a custom board developed in the Magneti Marelli Electronic Systems division.It was originally developed for vehicle infotainment purposes,but it has been modified for V2X functions.
This board has the following specifications:

Figure 3.2: The step 03 board

- 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

It is based on the NXP-Freescale Smart Application Blueprint for Rapid Engineering (**SABRE**) **i.MX 6QuadPlus** reference design and for a limitation of its System-on-Chip, the Step 03 supports only two Flexible Controller Area Network (**FlexCAN**) buses. FlexCAN modules provide a full implementation of the CAN protocol specification. In order to communicate with other on-vehicle CAN buses, the PEAK-System PCAN-USB interface is used.

3.2.2 MK5-Cohda Wireless OBU

The **MK5-OBU**(a Cohda Wireless' 5^{th} generation On-Board Unit (OBU))(figure 3.3) are products ready to be used in large scale field trials, aftermarket deployments, or serve as a reference design for automotive production and Smart City deployments.
It incorporates dual IEEE 802.11 radios, a GNSS positioning system providing lane-level accuracy,a powerful processor running V2X software stacks and applications, , and V2X security with hardware acceleration and tamper-proof key storage.
The MK5 chip-set , based upon the automotive-grade RoadLink^{TM} developed by NXP/Cohda, has unmatched radio performance in mobile environments, harsh outdoor, particularly in critical safety use-case scenarios.[17]

Some features of this board are the following :

- Embedded processor with optional SDK
- Embedded GNSS with dead reckoning (optional)
- IEEE 802.11p Access Layer
- IEEE 1609 Network Layer software
- ETSI TC-ITS Network Layer software
- V2X Facilities Layer software
- V2X Applications Layer software
- Dual or single antenna operation
- Dual or single radio operation
- Outstanding performance under outdoor, mobile conditions
- Security co-processor with tamper-proof key storage (optional)
- USB 2.0 host and OTG interfaces
- CAN bus interfaces
- Ethernet interface
- Audio output
- MicroSD card slot

- 12V & 24V operation
- Expansion Card (optional) 2nd CAN, serial, SPI, I2C, HDMI, WiFi, Bluetooth, Cellular
- Available in NEMA2 OBU enclosure

Figure 3.3: MK5-Cohda Wireless OBU

As regards the specifications of the MK5 :

- **Standard Conformance** : IEEE 802.11p - 2010 IEEE 1609 - 2010 IEEE 802.11an -2012 ARIB STD - T109 - 2012 ETSI ES 202 663 SAE J2735 - 2009
- **Bandwidth** : 10 MHz
- **Data Rates** : 3 - 27 Mbps
- **Operating System**: Linux 3.10.17
- **Antenna Diversity**
- CDD Transmit Diversity - MRC Receive Diversity
- **Receiver Sensitivity**: -100 dBm @ 3 Mbps
- **Environmental Operating Ranges**: -40ÂřC to +85ÂřC
- **Frequency Band**: 5 GHz

- **Max Tx Power**: +24 dBm (ETSI Mask C)
- **GNSS**: 1.5 m Best-In-Class Accuracy
- **Mobility & Multipath Tolerance**
- **Doppler Spread: 800 km/hr Delay Spread: 1500ns**
- **Dimensions**: 130 x 120 x 35 mm
- **Power Suppl**: 12/24V

3.2.3 Road Side Unit

Road Site Unit (RSU) are infrastructure mounted on one side of the street which provide information about the street and communicates with vehicles equipped with an On Board Unit (OBU).

Figure 3.4: RSU example

They are usually situated close to the traffic lights, typically on high supports in order to have a wider transmission range.
Thanks to Cohda Wireless there is the possibility to run a custom RSU application on an MK5 board to broadcast SAEJ2735 packets related to one or more intersections.
The Cohda Wireless board has been inserted in a box created with the external

ports to plug in the antennas, power and Ethernet.
The below schema represents the internal structure of an RSU:

Figure 3.5: RSU schema

The RSU can be configured in order to generate custom SPAT and MAP messages by using scripts and configuration files running on it.

Chapter 4

State of art

4.1 GLOSA existing versions

The first part of this thesis concerned the search for the existing **G.L.O.S.A** versions to evaluate the state of art regarding this type of application.
Below are some of these solution, divided according the considered number of approaches.

4.1.1 Single segment GLOSA

Here there are two articles regarding some possible implementation of the GLOSA algorithm as concern a single road segment.

The first one is **"Application of Vehicular Communications for Improving the Efficiency of Traffic in Urban Areas"**,as result of a research of the University of Surrey Guildford in 2011. This study uses an integrated simulation platform.
Here there is a brief description of this algorithm (figure 4.1):
When the on-board unit receives a cooperative awareness message (CAM, from the European standard), the algorithm checks if it was sent by a traffic light or not. Taking into account the position information of the message and the vehicle's own position and heading, it calculates if this traffic light is relevant (on its route) or not (line 1). The application can then calculate the distance from the traffic light and with the current speed and acceleration, the time that it would take to reach it (time-to-traffic-light TTL) (line 2). Next, it checks the traffic light phase at that time (TTL) (line 3). If the traffic light is green when the vehicle reaches it, then the vehicle continues its trip trying to reach the maximum speed limit of the road **Umax**(lines 4-6). If it is red, it calculates the speed that it should have to reach it in the next green phase (lines 7-9). If it is yellow, depending on the remaining yellow time and the acceleration capabilities of the vehicle, it could advice to accelerate or

decelerate again within the permitted range (lines 10-13). Finally, the driver gets an advise with the speed limited within the permitted range [V min, V max] (line 15). This algorithm runs every second, which makes it more robust against external interference, such as other vehicles, that do not follow the same advisory speed or are non-equipped [18].

Algorithm 1 *GLOSA* Algorithm

1: Find the most relative traffic light
2: Calculate Distance and Time to traffic light T_{TL}
3: Check phase at T_{TL}
4: **if** GREEN **then**
5: Continue Trip
6: Target Speed $(U_t) = U_{max}$
7: **else if** RED **then**
8: Calculate remaining Red Time (T_{red})
9: Calculate target speed for $T_{red} + T_{TL} : U_t$
10: **else if** YELLOW **then**
11: Calculate remaining Yellow Time (T_{yellow})
12: Check for possible acceleration
13: Calculate target speed for $T_{yellow} + T_{red} + T_{TL} : U_t$
14: **end if**
15: Advisory speed = MAX (U_t, U_{min}) & MIN (U_t, U_{max})

Figure 4.1: Glosa algorithm of the University of Surrey Guildford

The second article is **"A Cooperative ITS study on green light optimization using an integrated Traffic, Driving, and Communication Simulator"**. In the algorithm developed during this study are defined seven types of advisory message that can be used to instruct the human drivers:

1. Please wait for next green light.
2. Maintain speed at [suggested speed] km per hour to pass next green light.
3. Speed up to [suggested speed] km per hour to pass next green light.
4. Slow down to [suggested speed] km per hour for safe driving.

5. Maintain speed at [suggested speed] km per hour for safe driving.

6. Speed up to [suggested speed] km per hour for safe driving.

7. Slow down to [suggested speed] km per hour for safe driving.

The suggested speed is not updated each time step, but in three specific points (figure 4.2):

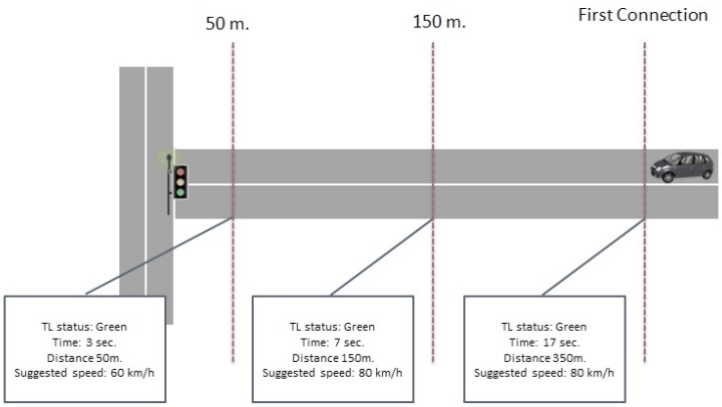


Figure 4.2: Distance Model to control updating of suggested speed for human drivers

Here is represented the decision tree to choice which is the best advice to give to the driver (figure 4.3) [19]:

4.1.2 Multi segments GLOSA

The following article regards a scenario composed by contiguous intersections:

Here is showed **"B&B Algorithm Based Green Light Optimal Speed Advisory Applying to Contiguous Intersections Biao XU, Fang ZHANG, Jianqiang WANG, Keqiang LI "**.This research is funded by the Chinese National Program for High Technology Research and Development , which is also supported by the joint research project of Tsinghua University and NISSAN Motor.
It starts with the definition of a **kinematic model**:
According to this model, when approaching an intersection, the vehicle will adjust its velocity, and then pass the intersection with constant speed.The adjustment process the vehicle is running with constant acceleration or deceleration.So the vehicle adapts its velocity to target velocity **Vi** and keeps the pace until passing each intersection during the green phase.
The second step consists in the study of a fuel consumption model, that shows the

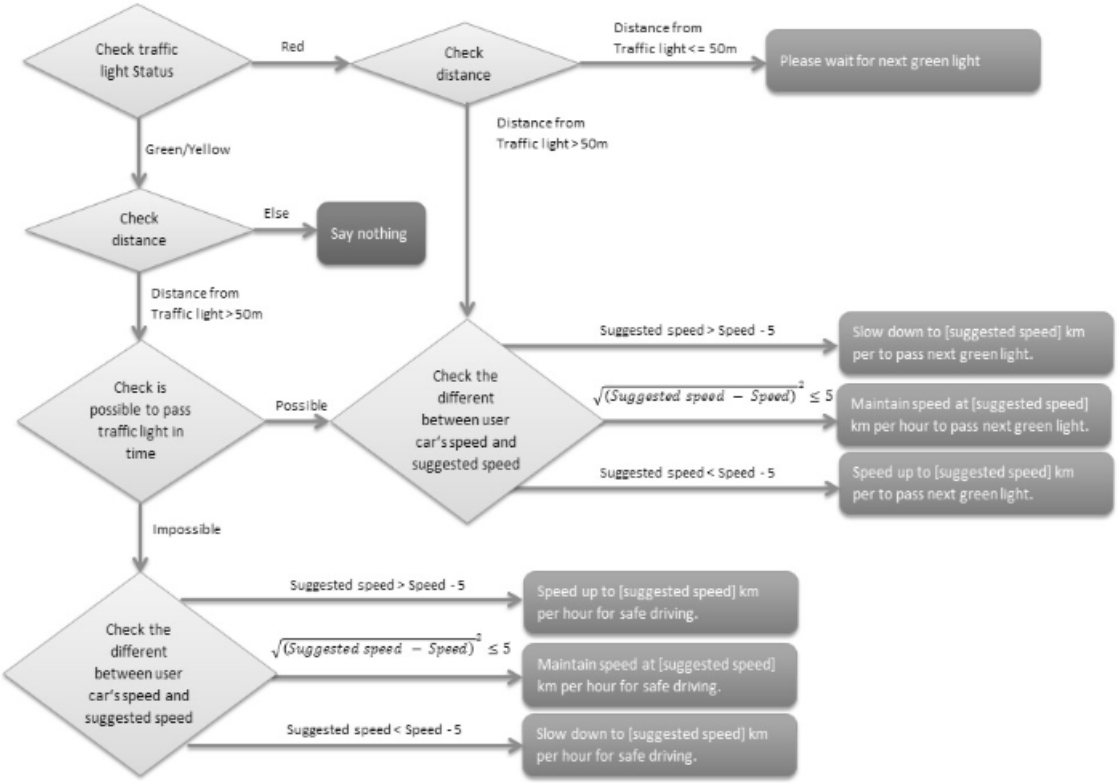


Figure 4.3: Decision tree to determine the message generated to advice the human driver on his speed decision based on the traffic light status

$$F_c = k_0 + \left(k_1 + k_2 \frac{dv}{dt}\right)v + \left(k_3 + k_4 \frac{dv}{dt}\right)v^2 + k_5 v^3 + k_6 v^4 \, , T_e > 0$$

Figure 4.4: fuel consumption model for a single intersection

relation among the fuel consumption, velocity and acceleration is built as follows according to equation (figure 4.4):

Here the coefficients, k0 k6, need to be fitted based on the fuel consumption data and **V** is the target speed in a specific intersection.
Given this equation , it is possible to determine the total fuel consumption to pass N intersections (figure 4.5) :

This objective function (in which a_i and v_i are the acceleration and the speed in each road segment) will be minimized.
The issue is that it is difficult to find the analytical solution so the **B&B algorithm** is used to obtain the approximate optimal solution.

$$F(v_i) = \sum_{i=1}^{N} \int_{t_i^1}^{t_i^2} \left(k_0 + (k_1 + k_2 a_i)v + (k_3 + k_4 a_i)v^2 + k_5 v^3 + k_6 v^4\right) dt$$
$$+ \sum_{i=1}^{N} \int_{t_i^1}^{t_i^2} \left(k_0 + k_1 v_i + k_3 v_i^2 + k_5 v_i^3 + k_6 v_i^4\right) dt \quad (t_0^2 = 0)$$

Figure 4.5: total fuel consumption model for N intersections

This algorithm is applied in several steps (figure 4.6):

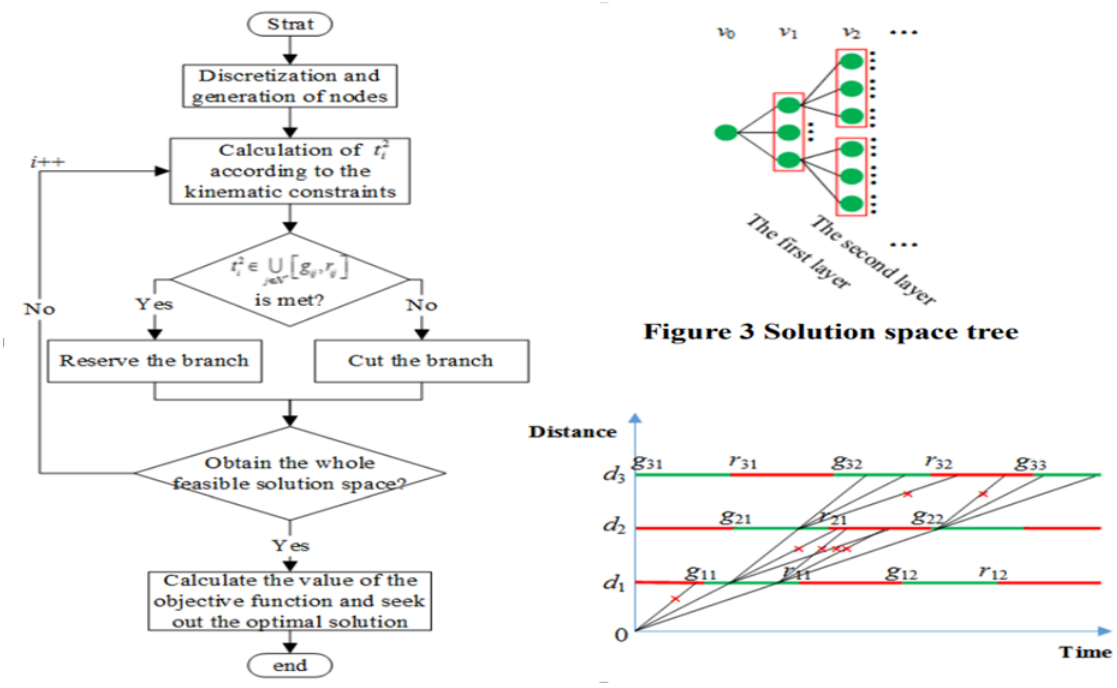


Figure 4.6: B&B algorithm representation

- The first step of the B&B algorithm is to discretize the target velocity in each segment according to the velocity limit constraints.
- The discretization values of the target velocity in each segment named nodes constitute a solution space tree with plenty of branches,
- Then the corresponding trip time of each node in the first layer is calculated using the kinematic constraints. When the trip time fails to reach the green light constraint at the first intersection, the corresponding branch should be cut, Otherwise, go to the next step.
- Next, the corresponding trip time of each node in the i-th layer is calculated using the kinematic constraints. When the trip time fails to reach the green

light constraint at the i^{th} intersection, the corresponding branch should be cut. Otherwise, reserve the branch.

- After the feasible solution space is found, the value of the objective function is computed and the optimal solution is sought out. [20]

.

Chapter 5

Software : Use Case GLOSA

5.1 GLOSA overview

UC-GLOSA is one of the use cases developed by the Magneti Marelli Innovation Connectivity team for the Magneti Marelli Connectivity Framework, which is a solution to enable a Vehicle-to-Everything (V2X) communication.
This use case takes information from other road's actors (traffic lights, other vehicles and other infrastructures) with the aim to suggest to the driver an appropriate range of speeds(when it is possible) to overcome a traffic light during the green phase.
All the advice provided by this software are given paying particular attention to driver's safety.

The UC-GLOSA starts to work only when the vehicle enters the area defined by MAP messages provided by the RSU (Road Site Unit).The map message content includes such items as complex intersection descriptions, road segment descriptions, high speed curve outlines (used in curve safety messages), and segments of roadway (used in some safety and for platoon applications).
These information, combined with those coming from the SPAT messages (current phase and time to next phase of a traffic light),the VDP mediator, the BSMs and GPS data (in the second version of the algorithm) , allow to have an appropriate advice about the range of speeds to overcome the traffic light.

The initial idea (**3rd revision**) was to find an exact speed value that was the goal to be reached to pass during the green.
This recommended speed **(Vr)** was an average speed given by the ratio between distance from stop and time.

$$Vr = \frac{DistanceFromStop}{TimeToChange}$$

According to the current phase, Vr was set to the lower value of the scale (RED case) or to the upper value (GREEN case), because of the granularity of the speedometer.
This implementation proved to be ineffective given that the algorithm's output was a different speed every second, so the driver did not have a comprehensible feedback.

The second step was the **4th revision**, that introduced some news respect to the previous one:

- Average speed was calculated once only when the first SPATMAP was received.
- Corrections made only if the current speed exceeds the recommended average speed (red light) or is lower than it (green light).This correction consisted of a new recommended speed calculated according to the uniformly accelerated motion's laws.

The issue of this second solution was that,in some cases,the output of the algorithm was a negative recommended speed (different from error messages).This problem was an effect deriving from the physics' model used in this version.

$$Vr = V_0 + a * t$$

As a matter of fact, assuming a uniformly accelerated motion, the path will be a parabola,therefore in some points the arrival speed will be negative because the derivative of speed (acceleration) in that specific point could be negative too.
So the following step was trying to find a curve that allows to get to the point of arrival while maintaining positive speed and minimizing energy consumption (In case of going above or below the recommended speed initially, depending on the color of the light).
This idea was abandoned because of the difficulty in finding this new function and in favor of a new and simpler model, closer to the driver's necessities.In fact, one of the main issues in these models was that for a driver it's hard to follow a specific speed that could change each second:
All these evaluations brought to the definition of a new model, based on a range of possible speeds, also called **"green window"**.

5.2 Use Case GLOSA one vehicle version

This section introduces the final version of this thesis' work.At the basis of this last version there is the idea to give a range of speeds that permit to overcome a traffic light during the green phase.
This range, sometimes called **"green window"**, is included between a maximum speed $\mathbf{V_{max}}$ and a minimum value $\mathbf{V_{min}}$.
In all cases, the recommended speed must not overcome the legal speed limit of the road and must not be lower than an acceptable minimum speed.
When these two limits are not respected by both ends of the range (too higher or too lower), it is not possible to cross the current approach in a safe way, so some advice are given to the driver.

Logical Model

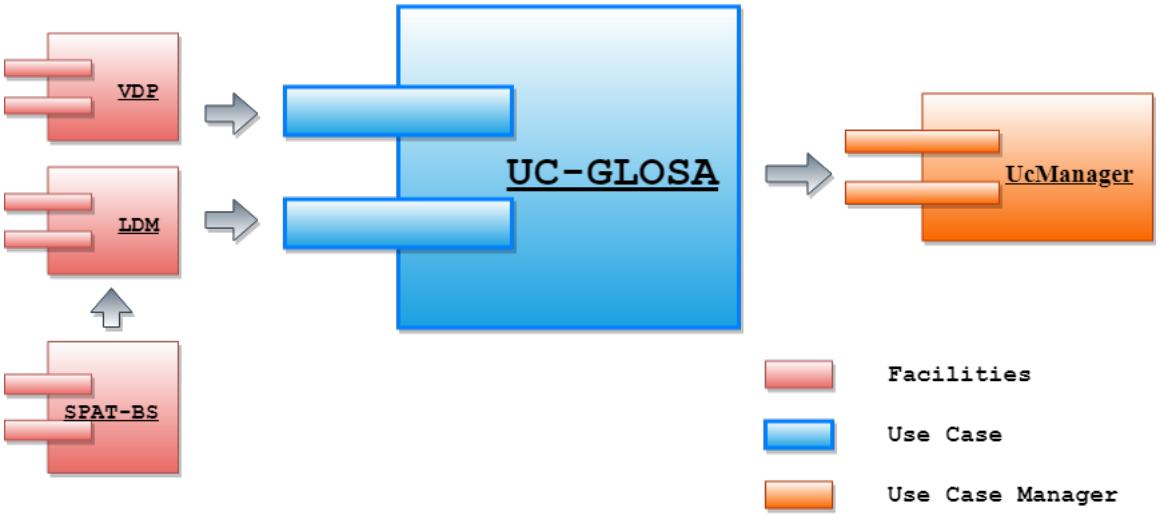


Figure 5.1: Logical Model of one vehicle version

This logical model describes in general how the module UC-GLOSA (one vehicle version) works and which is the data flow according to the framework architecture. Inputs comes from the **"Facilities"** sub-layer and output is sent to the **UseCase Manager** and then propagated to the **HMI** if there are no data from Use Cases with higher priority.
In particular, this UC-GLOSA version takes **SPATMAP** messages from **LDM Database** (coming from an RSU) and data related to the vehicle from the **CAN Network** (ex. speed) through the **VDP** (Vehicle Data Provider).

5.2.1 Algorithm

Here is described, through the use of diagrams and flowcharts, the algorithm at the basis of the **UC-GLOSA**:
The following flowcharts have been made using the tool **"Enterprise Architect"**, which allows to build these models in a simple way.

The tables shown below contain an accurate description of all input and output of the UC-GLOSA algorithm, paying particular attention to the unit of measure of each parameter.
As regards the output speeds, although they are expressed in Km/h, their unit is changed by the UcManager according to the current one of the country in which they are used.

INPUT

NAME	SOURCE	UNIT	DESCRIPTION
Distance from stop	spatmap/Simulink	m	Distance from traffic light
Time to change	spatmap/Simulink	s	Time left to the next color
TL color	spatmap/Simulink	/	Current traffic light color
Egospeed	VDP/Simulink	m/s	Current vehicle speed
Legal speed	Config file or TIM	m/s	Maxi speed allowed by law
Minspeedlimit	Config file or TIM	m/s	Min reasonable speed

Table 5.1: Table of GLOSA input

OUTPUT

NAME	UNIT	DESCRIPTION
V_MAX	Km/h	Upper limit of green window
V_MIN	Km/h	Lower limit of green window
Popup	/	Advice to the driver

Table 5.2: Table of GLOSA output

Possible advice to the driver

As regards this first version of **UC-GLOSA** algoritm, there are four possible types of popup which can be given to the driver, with the aim to suggest the right behaviour to follow:

- **popup = 0=> "maintain speed"** . The speed of the vehicle is the right one to reach the green phase when crossing the intersection.
- **popup = 1 => "info warning"** . The speed of the vehicle is higher than V_MAX, so a small deceleration is need or , in some cases, it is too lower than MINSPEEDLIMIT so it is better to stop the vehicle just before the stop.
- **popup = 2 => "Critical warning"** . The speed is too high respect to the "green window" or the "green window" is not feasible, so the vehicle must stop.
- **popup = 3 => "Accelerate"** . In case of feasible "green window", if Egospeed is lower than V_MIN, it is possible to accelerate in a totally safe way.

V_MIN and V_MAX shown on HMI

Sometimes the "green window" could cover a very extended range of speeds, so it could cause an unpleasant view on the HMI.
To correct this behaviour, when Egospeed is between V_MIN and V_MAX, the window is truncated on the top, on the bottom or in both sides respect to Egospeed if these limits exceed it at least 10 km/h .

GLOSA-general

The algorithm starts with the analysis of the SPATMAP message coming from the LDM database once per second. According to the traffic light color in that specific moment, three different cases are possible (figure 5.2).

- TLcolor = 1 => GLOSA_GREEN_LIGHT-general
- TLcolor = 2 => GLOSA_YELLOW_LIGHT-general
- TLcolor = 4 => GLOSA_RED_LIGHT-general

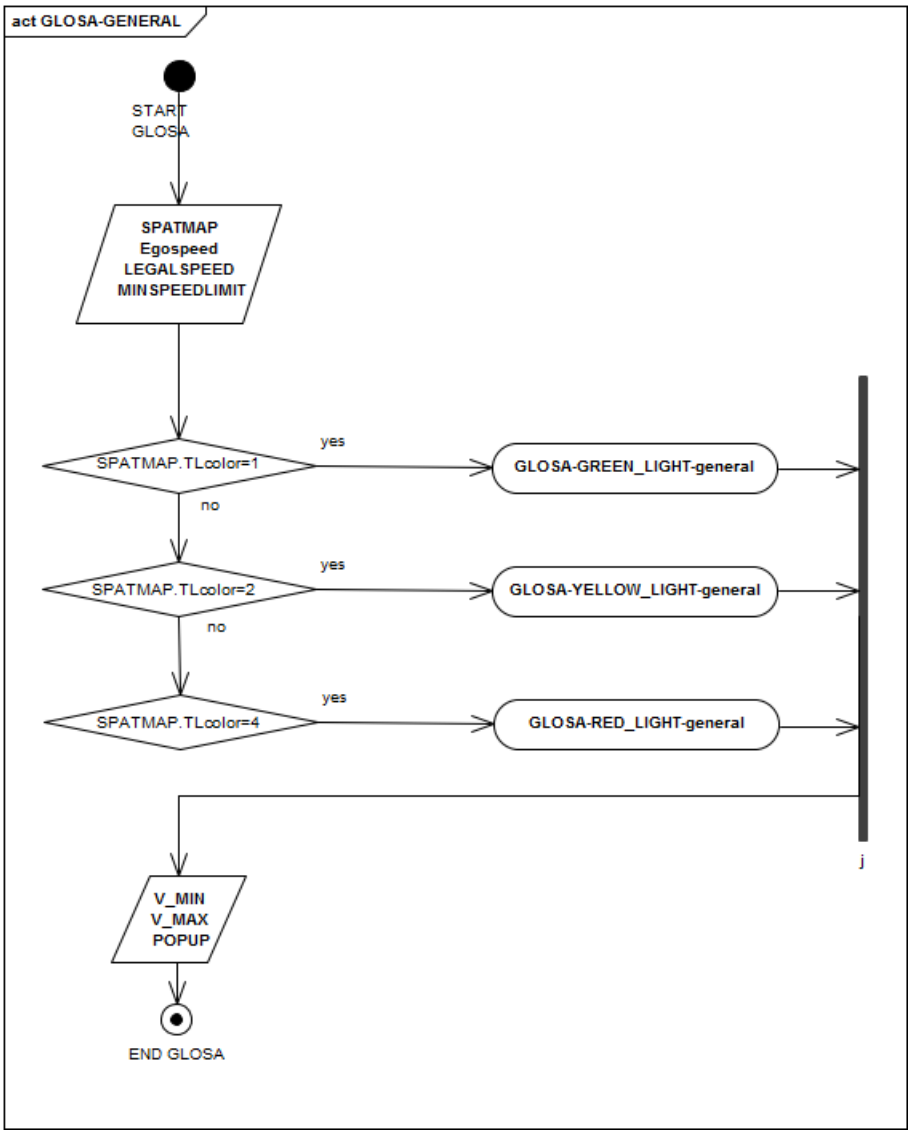


Figure 5.2: Flowchart of first step of GLOSA

GLOSA_GREEN_LIGHT-general

Because of the current color of the traffic light is green, the first step is to evaluate if the driver is able to reach the intersection before the next phase changing. For this reason V_MAX is always set to the Legal speed of that specific road, while V_MIN is obtained using the uniformly accelerated motion formula (the addition of 2 seconds is in order to allow the crossing of the intersection also during the first seconds of the yellow phase).

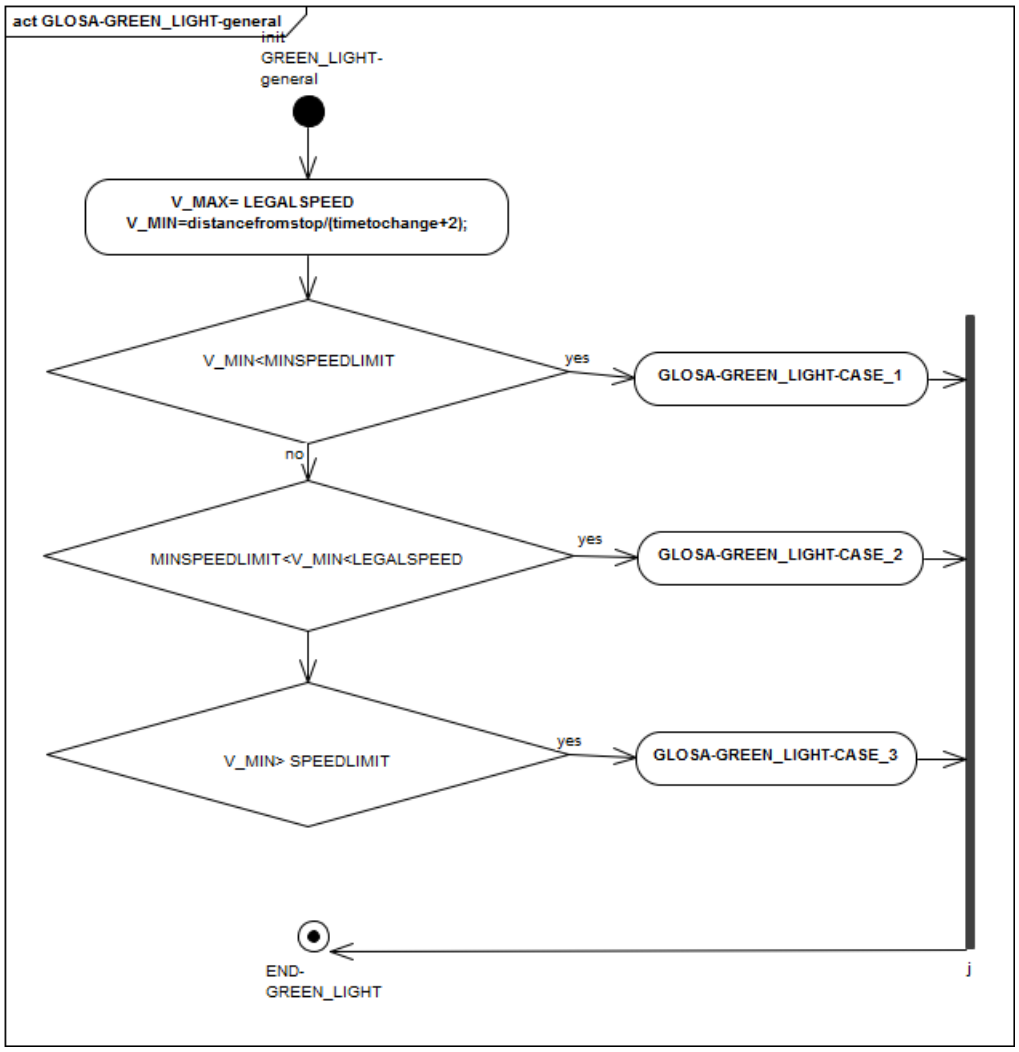


Figure 5.3: Flowchart of the general case of green phase

The value of V_MIN discriminates the next step of the algorithm.

GLOSA_GREEN_LIGHT_case_1

In this case V_MIN is lower than the chosen MINSPEEDLIMIT, so it is set to this specific last value.The "green window" is the largest possible, from the Legal speed to the MINSPEEDLIMIT.
From the current speed of the car (**Egospeed**) derives the specific advice given to the driver, in order to enter this green window or to remain in it.

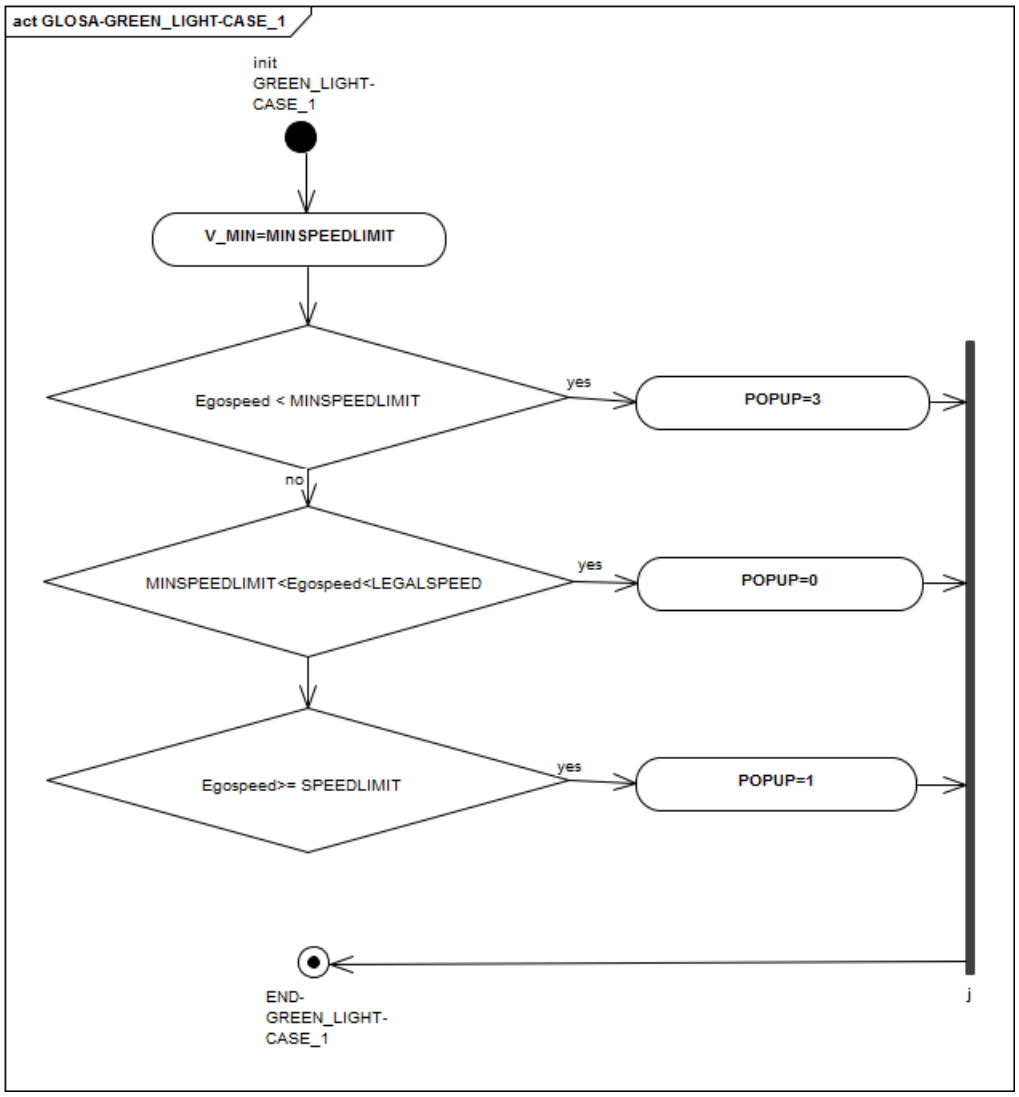


Figure 5.4: Flowchart of the first case of green phase

GLOSA_GREEN_LIGHT_case_2

The advice given to the driver depends not only by the relative position of the Egospeed respect to the V_MIN or V_MAX, but also from the comparison between the distance from the traffic light and the current safety space to stop the vehicle (which depends from the Egospeed):
This evaluation is visible in the second case of this flowchart.
Another evaluation is given by comparing the difference between V_MIN and Egospeed: because the V_MIN has the tendency to become higher second after second, if Egospeed is still lower than it and the difference is higher than 4 Km/h (1 m/s),it could be hard for the driver to reach the green window.

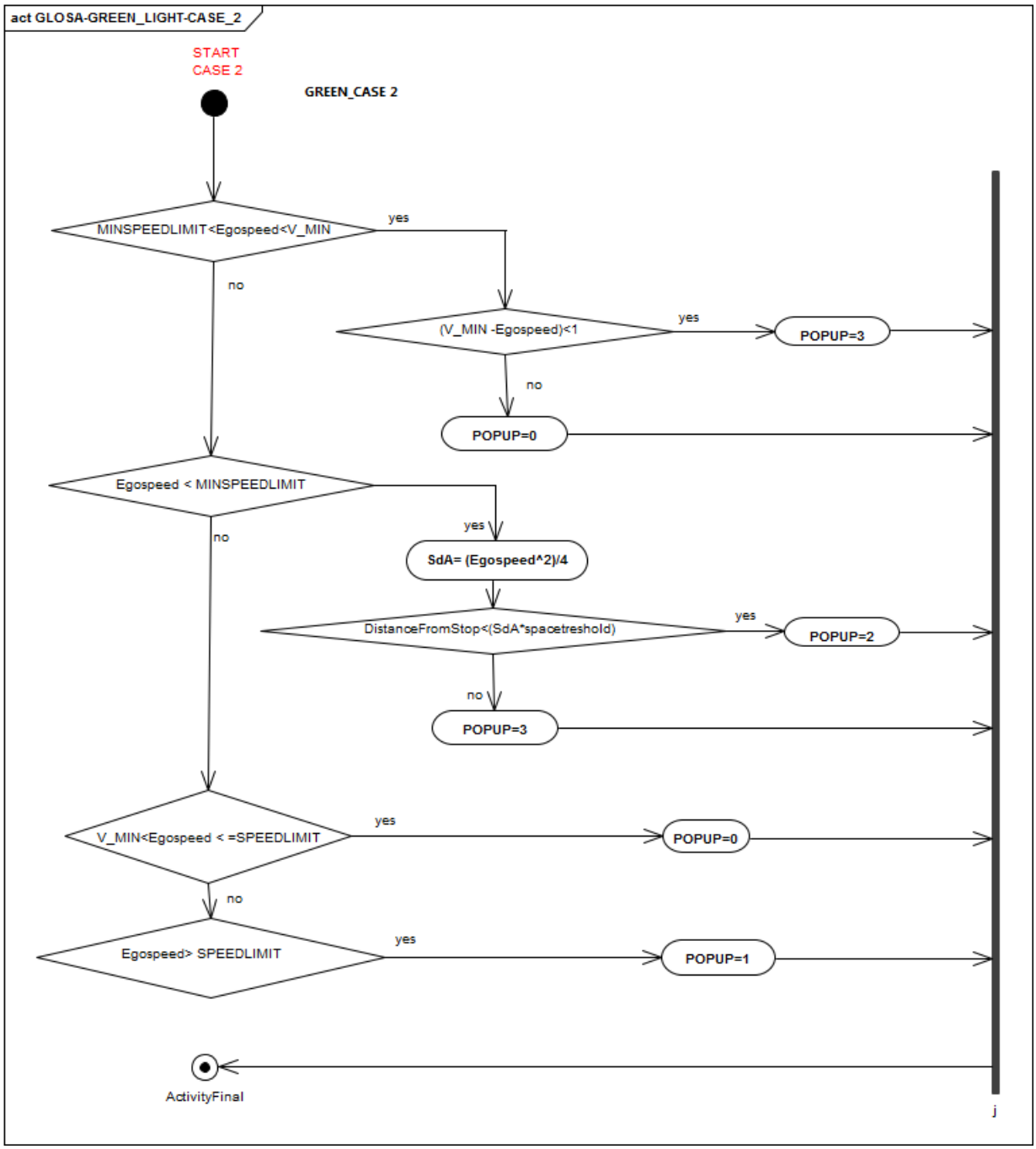


Figure 5.5: Flowchart of the second case of green phase

GLOSA_GREEN_LIGHT_case_3

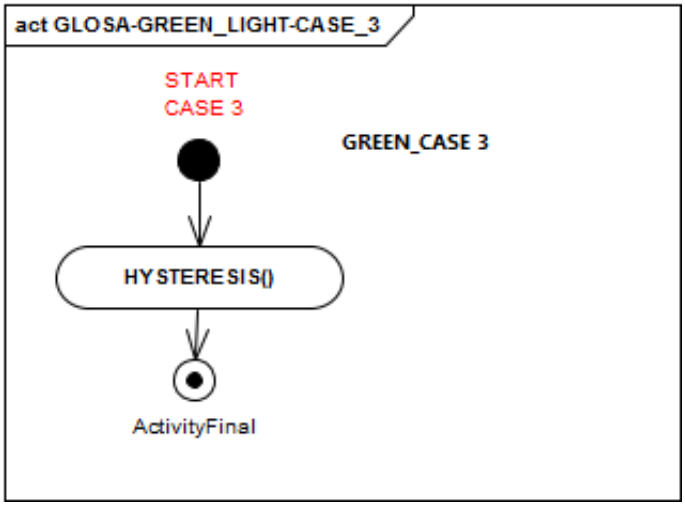


Figure 5.6: Flowchart of the third case of green phase

This can be considered as a particular situation, because during the several tests made using Simulink,a not appropriate sequence of popups was shown (ex. "Critical Stop" followed by "Maintain speed" and then by "Critical stop").
This wrong result was avoided introducing an hysteresis,which permits to give the same advice in consecutive moments,so to go far from that edge situation.

GLOSA_YELLOW_LIGHT

During the yellow phase there are some checks about the distance from stop and the Egospeed:
The first one is to use the hysteresis in order to give the same advice (as regards the popup) respect to the previous one.
After this check, if the distance from the traffic light is higher than 10 meters, the advice is "maintain speed".

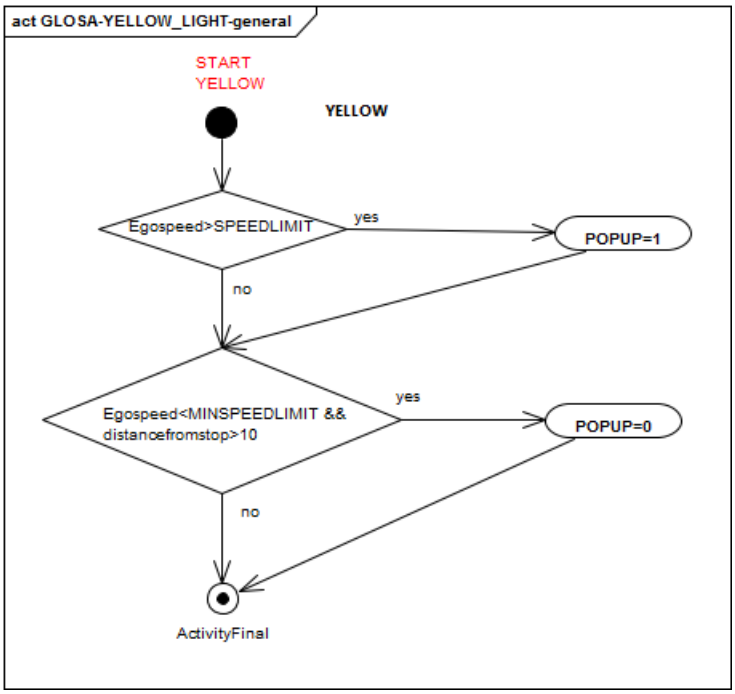


Figure 5.7: Flowchart of yellow phase

GLOSA_RED_LIGHT-general

This is the first step of the GLOSA algorithm in case of red light:
In this situation the "green window" is calculated as follows:

- V_MAX is given as the rate between the current distance from the traffic light and the time to the next phase switch ("time to green").

- V_MIN is calculated in the same way but, in this case, the time is the sum of "time to green" and a "TverdeMin" that can be considered as a reasonable minimum duration of the green phase.
  This last value can be changed from configuration file.

In this case there are 6 possible "green windows", depending from the relative position of V_MAX and V_MIN respect to the legal speed of that specific road and the minimum speed limit set in configuration file.

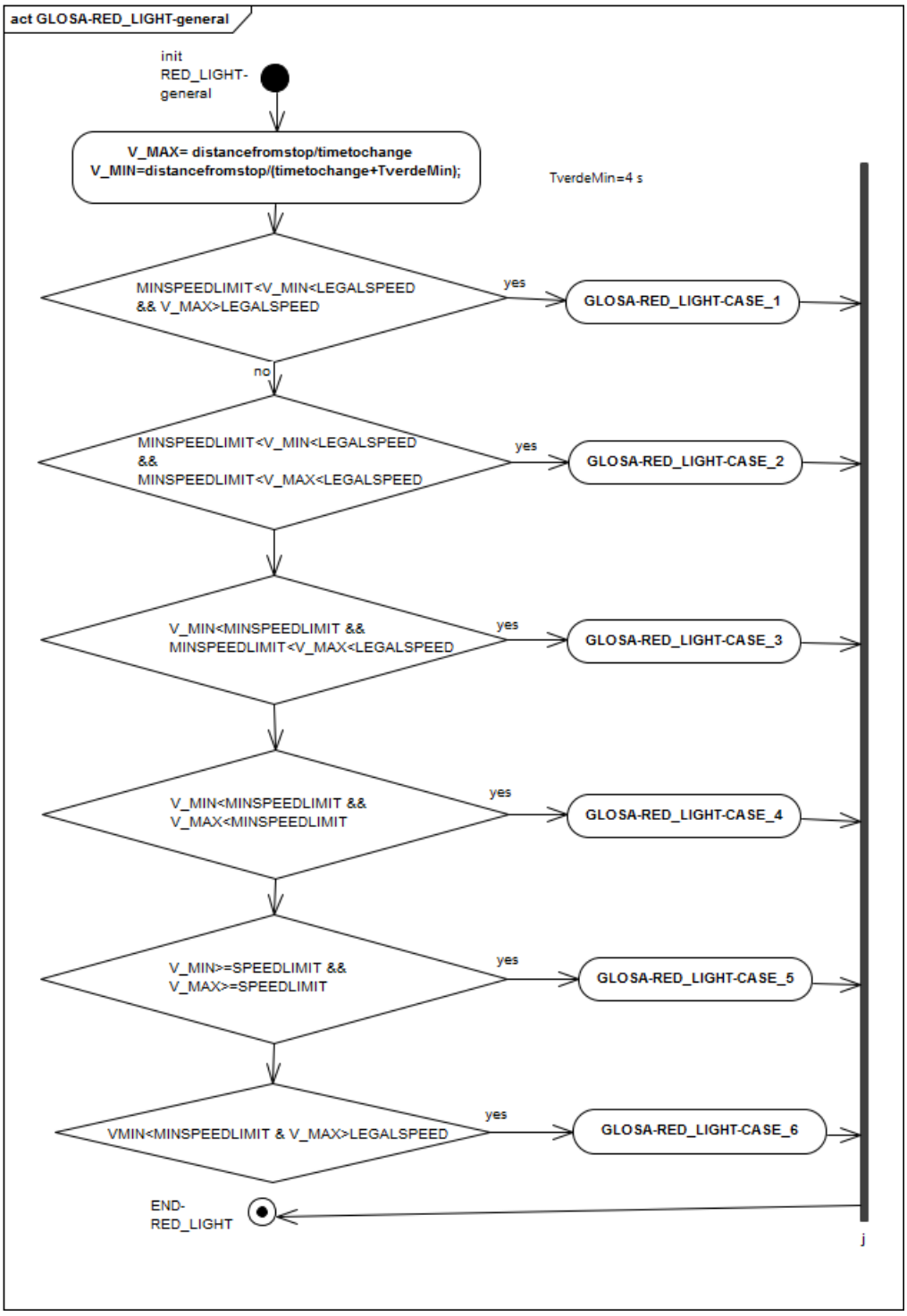


Figure 5.8: Flowchart of red phase

GLOSA_RED_LIGHT-CASE_1

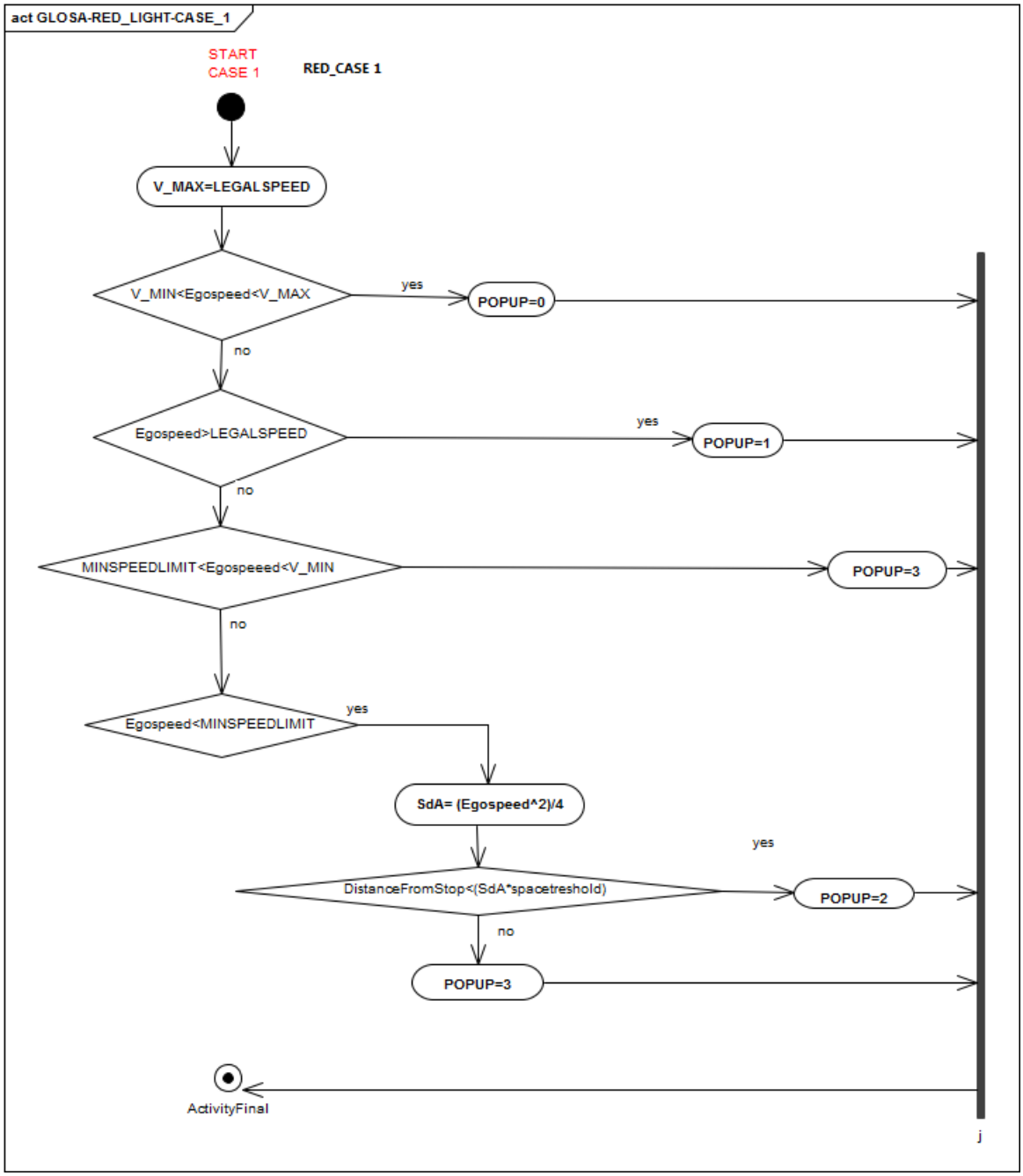


Figure 5.9: Flowchart of the first case during red phase

In this situation V_MAX is higher than the legal speed limit of the road, so it is set to this maximum allowed speed.

The most relevant thing of this "case" is the check on the current distance from approach:
If this this distance is higher than the current space of stop of the car (which depends from the egospeed) adjusted with a space threshold,the driver can accelerate to reach the green window in complete safe.

GLOSA_RED_LIGHT-CASE_2

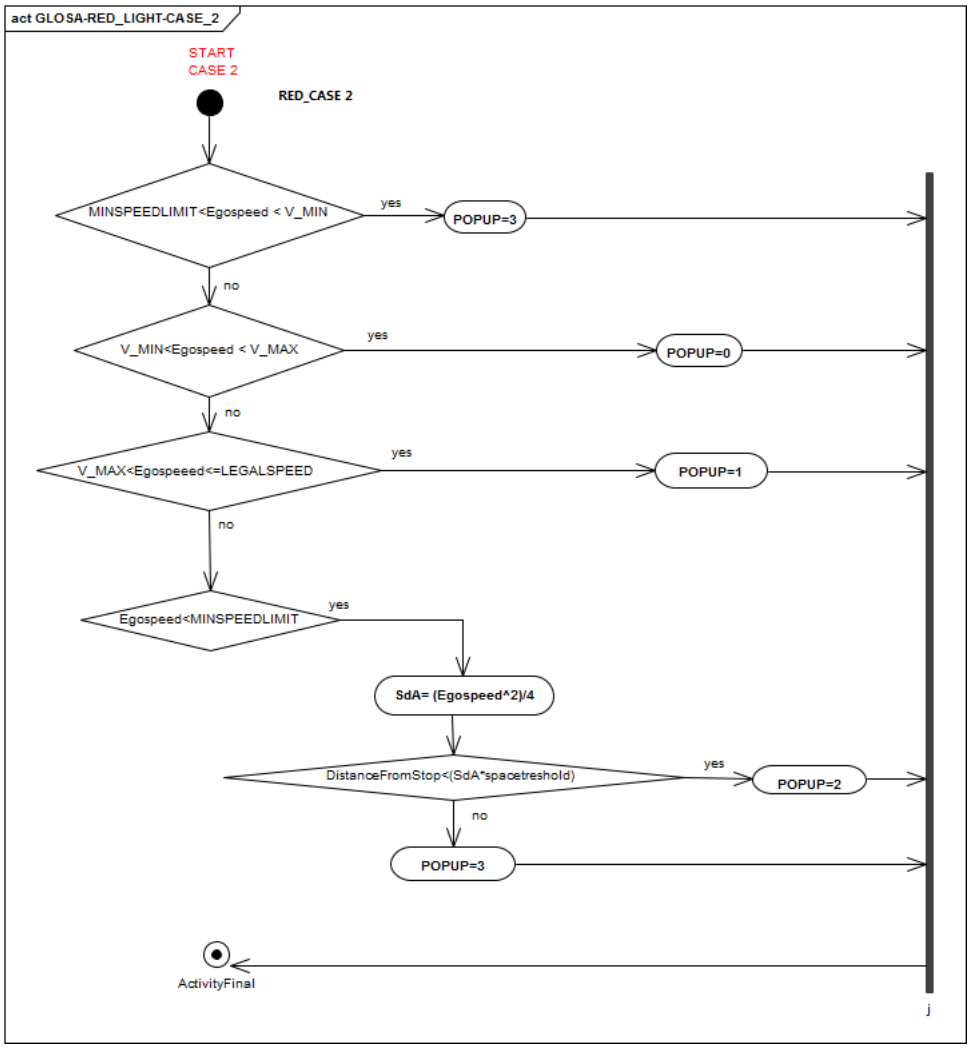


Figure 5.10: Flowchart of the second case during red phase

Here V_MAX and V_MIN are both included between the legal speed and minimum speed limit.
There is the same check regarding the current distance from stop as in the previous case (when Egospeed is lower than MINSPEEDLIMIT).

GLOSA_RED_LIGHT-CASE_3

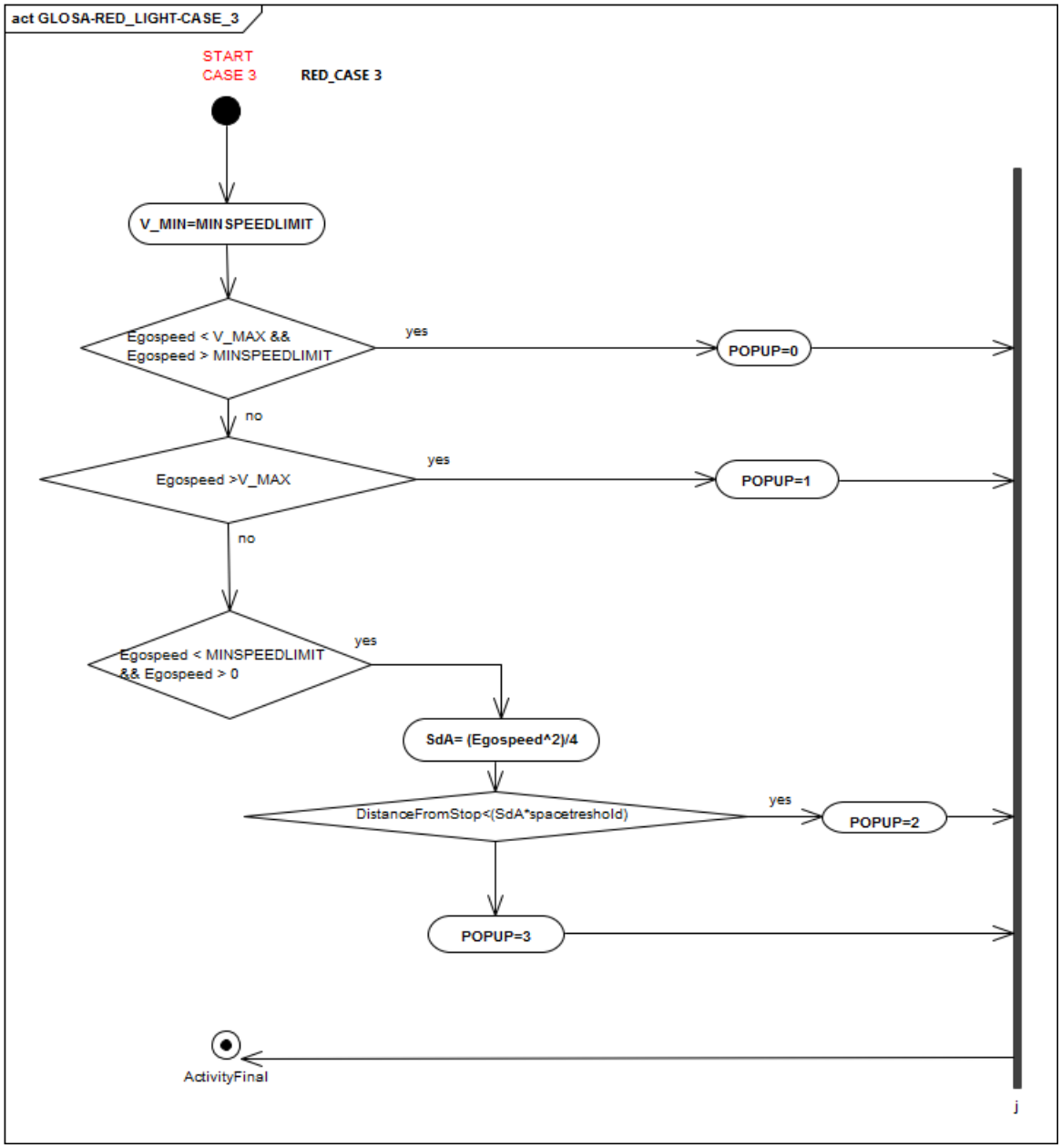


Figure 5.11: Flowchart of the third case during red phase

This case is similar to previous except for V_MIN, which is lower than MIN-SPEEDLIMIT, so it must be set to this last value, reducing the extension of the green window.

GLOSA_RED_LIGHT-CASE_4

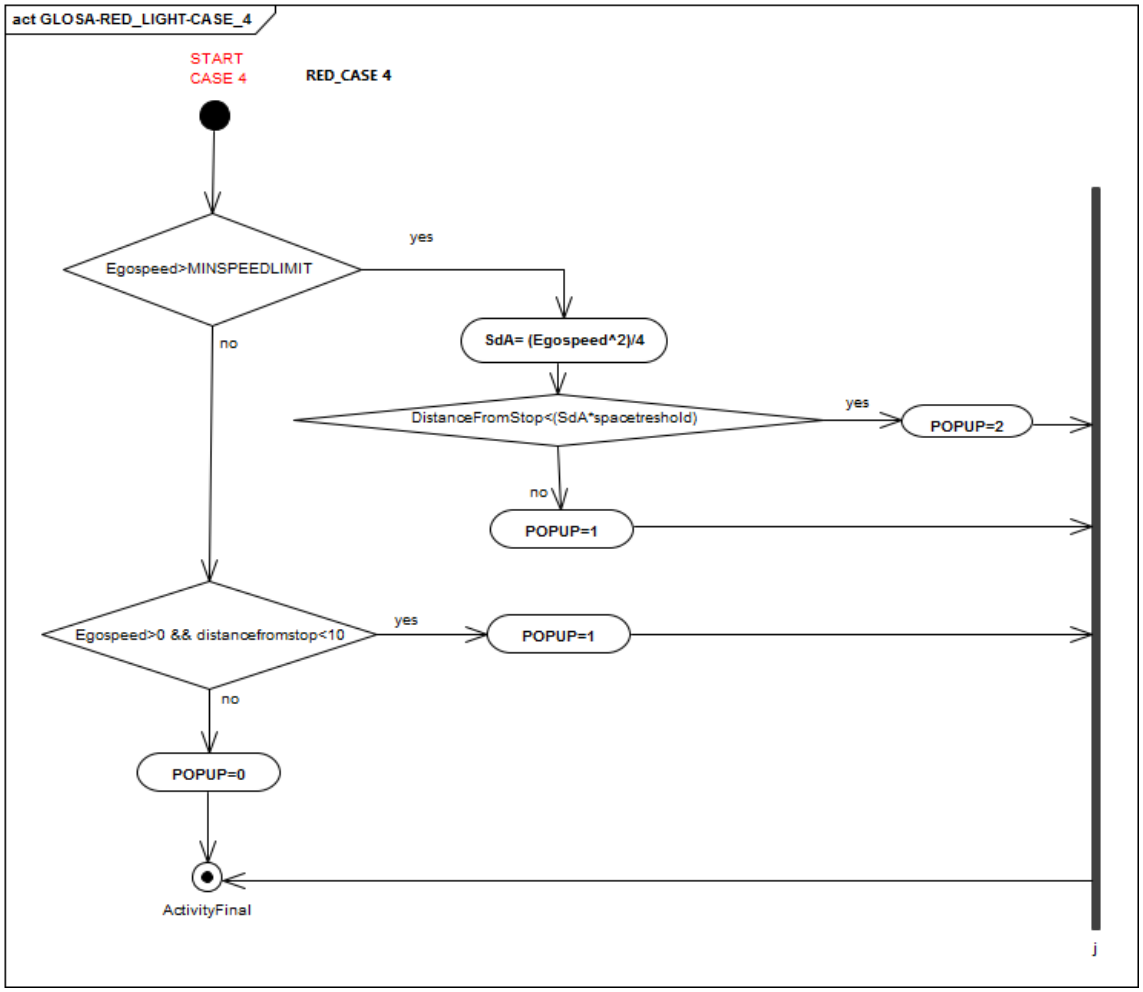


Figure 5.12: Flowchart of the fourth case during red phase

The fourth case of the RED case checks if Egospeed is lower than MINSPEEDLIMIT. In affirmative case, if the current speed of the vehicle is not zero and the distance from the approach is lower than 10 metres a not critical stop popup is given to the driver,in order to make him stop before the end of the lane.

GLOSA_RED_LIGHT-CASE_5

The fifth case regards a situation of a not feasible "green window", because V_MAX and V_MIN are both higher than legal speed.
If the comparison between the distance from stop and space of stop shows that the car is in the proximity of the traffic light, a critical stop popup appears because the current Egospeed will never be enough to get the green light.

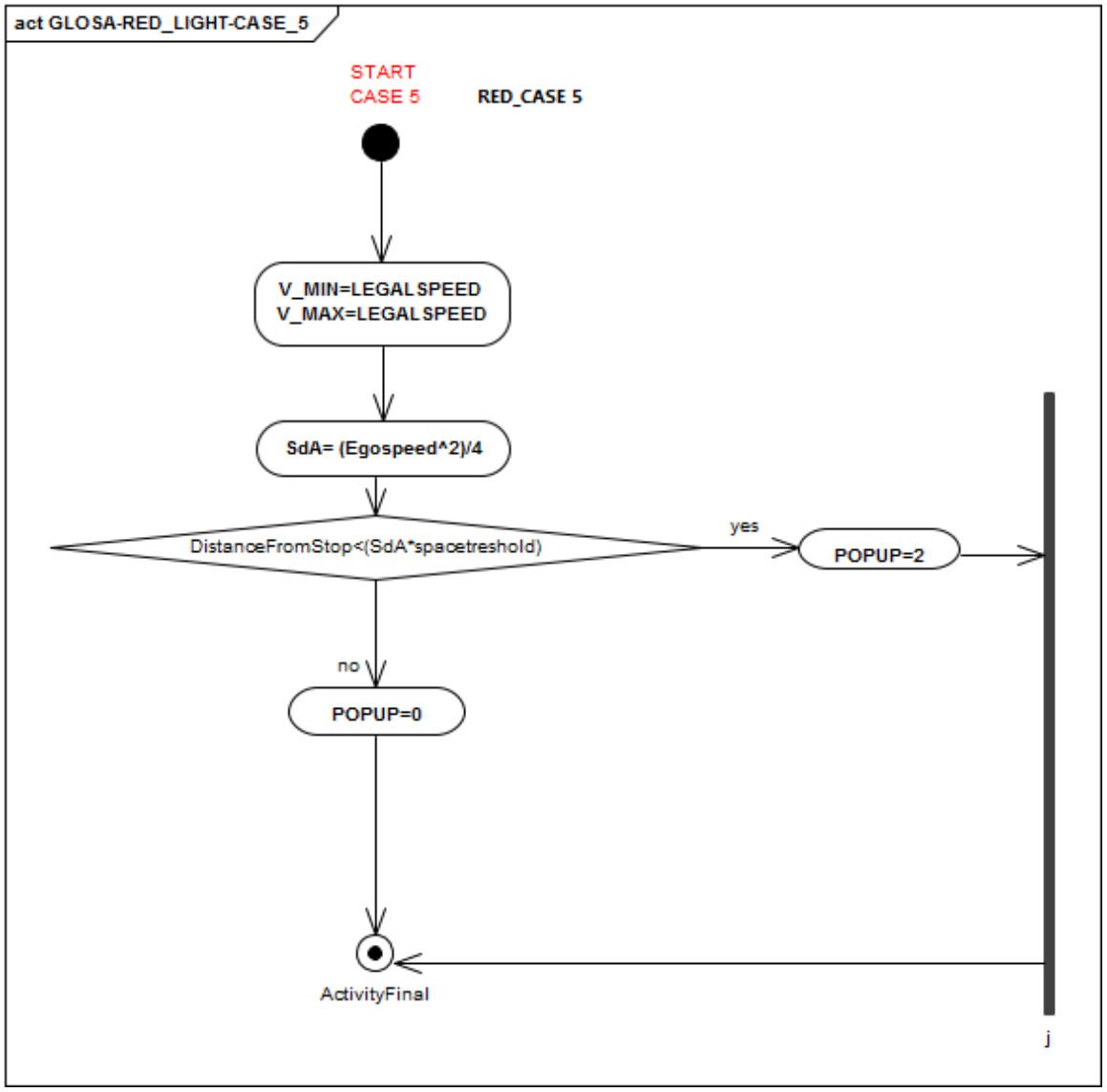


Figure 5.13: Flowchart of the fifth case during red phase

GLOSA_RED_LIGHT-CASE_6

In this sixth case the "green window" has the largest possible size (from legal speed to MINSPEEDLIMIT) so, according to Egospeed, the advice given to the driver could be "maintain speed","info warning" or "accelerate".

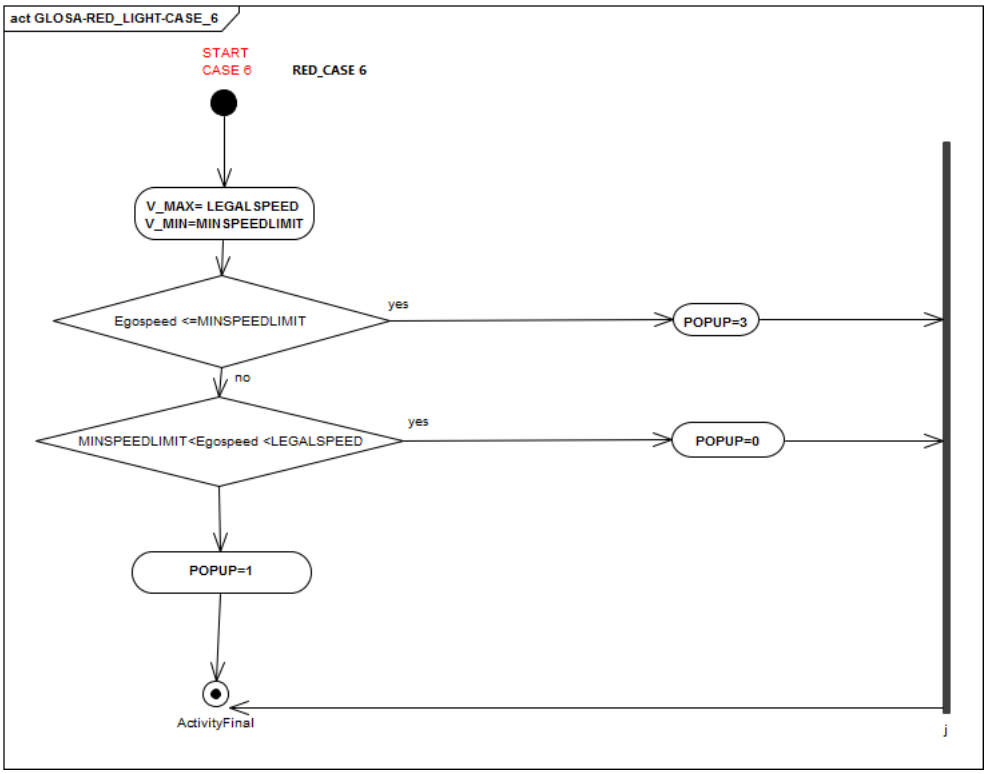


Figure 5.14: Flowchart of the sixth case during red phase

HYSTERESIS

It was introduced to avoid the alternation between different popups in consecutive time-steps when the difference between distance from stop and (space of stop *threshold) is among 0. In that cases difference can be alternately <0 or >0 in consecutive seconds and the result was very unpleasant to the driver. In order to fix that, 4 logical strips were considered, each one with its ID (figure 5.15):

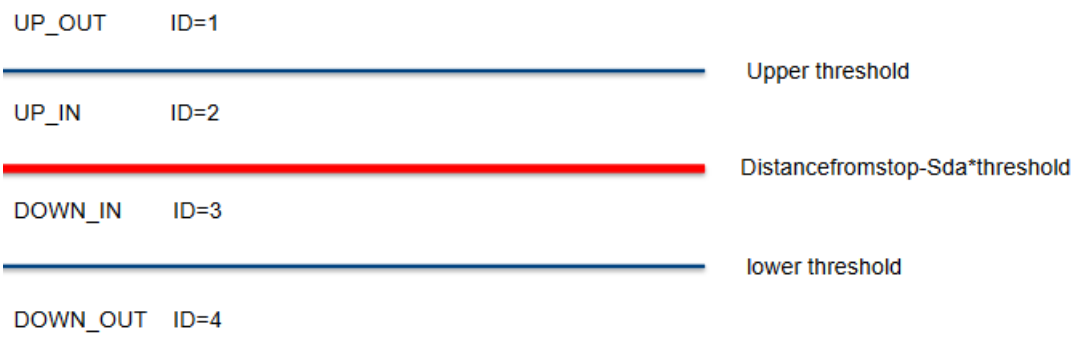


Figure 5.15: Representation of the 4 possible strips for hysteresis

Strip is calculated at each second, taking into account the formula:

$$Difference = distance\_from\_stop - (Space\_of\_stop * space\_threshold))$$

According to "difference", it is possible to determine the exact "strip ID".

1) Difference >strip_threshold ==> **UP_OUT ID=1**

2) 0<Difference <= strip_threshold ==> **UP_IN ID=2**

3) strip_threshold <Difference <= 0 ==> **DOWN_IN ID=3**

4) Difference <= strip_threshold ==> **DOWN_OUT ID=4**

The amount of threshold is configurable.
In the **"OUT"** strips, in that cases the popup is **fixed**, in particular "0"(no popup or maintain speed) when ID=0, "2"(Critical stop) when ID=4.
The situation changes as regards the **"IN"** strips because the popup value depends from the previous one,in order to give not conflicting information.
This new feature was applied in case of conflict between CRITIC_STOP popup and NO_POPUP. It remains to be considered whether to apply the same criterion in other cases of alternation .

5.2.2 Simulation on Simulink

The next step in this thesis work was the development of a simulation environment, in order to test and verify the validity of the algorithm.For this purpose the chosen software was Matlab with its extension Simulink.
In particular, the created Simulink model is composed by some blocks, each with the aim to simulate a particular function, from the input to the algorithm to a possible driver reaction to the output.
Each simulation block contains a matlab function written in Matlab programming language, a C-like language.

Here is shown an overview of the complete model, and then this there will be the description of the single blocks(figure 5.16):

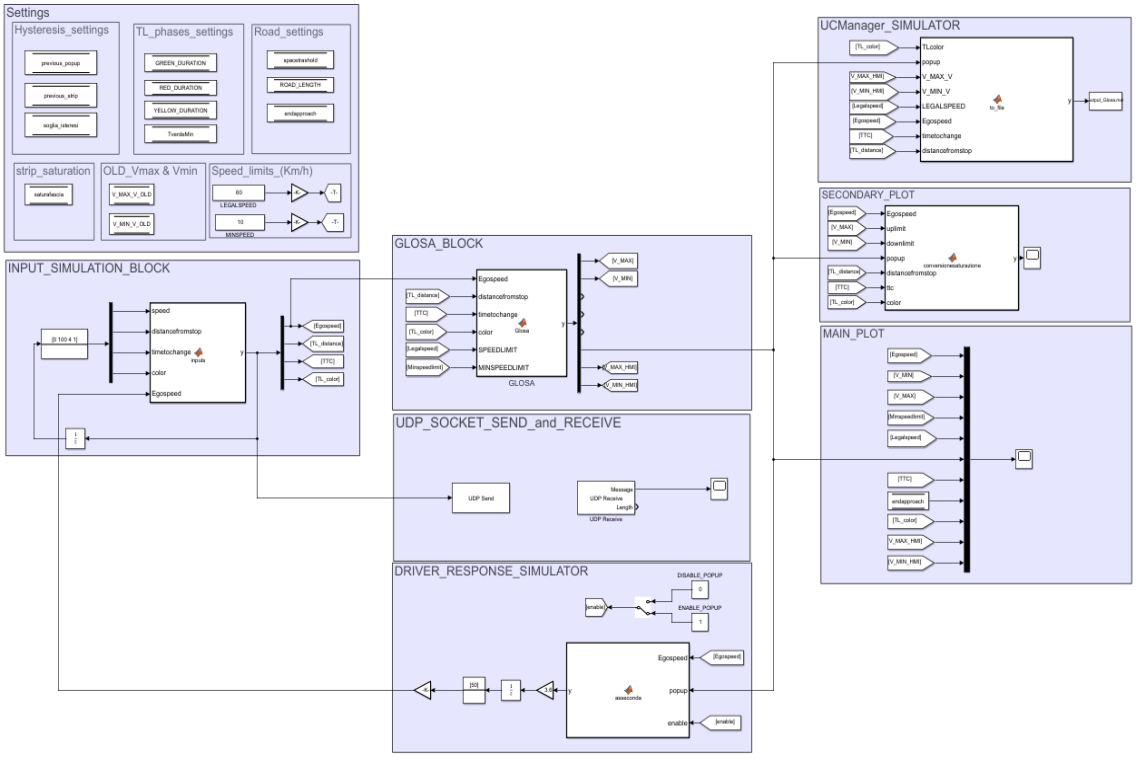


Figure 5.16: Simulink model overview

The main blocks of the developed model are described below:

INPUT_SIMULATION_BLOCK

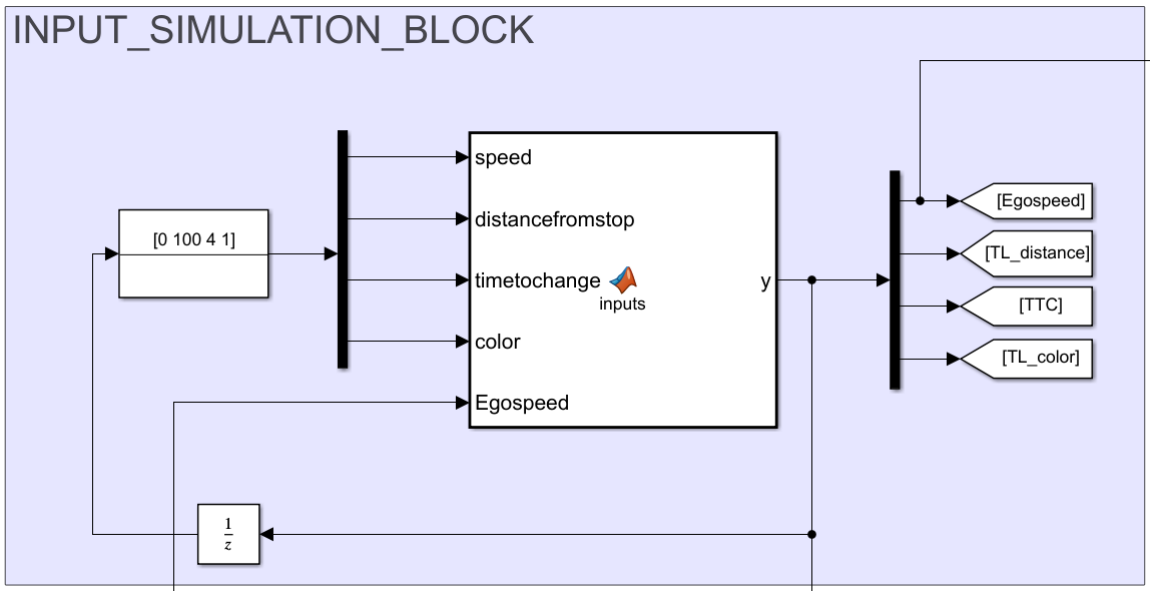


Figure 5.17: Focus on input simulator block

This is the starting function of the model.Through the array on the left of this this block it is possible to set the initial conditions (in this example a Map Area of 100 m,a time to change of 4s and green as starting color).
The "inputs" function receives as input the current vehicle speed, distance to approach, the time to the next phase and the color and decreases their value according the simulation step (1s).
These input can be considered as a mix between the SPATMAP message and the data given by VDP.
The output is sent to the following step, the GLOSA_BLOCK.

GLOSA_BLOCK

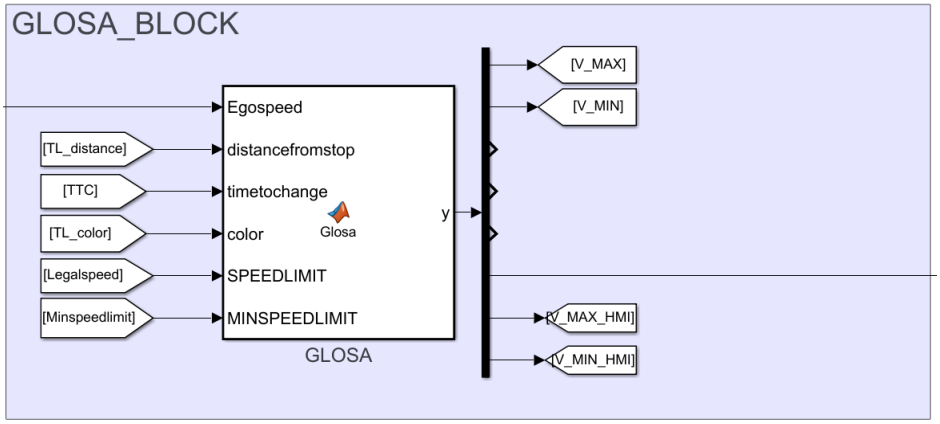


Figure 5.18: Focus on GLOSA block

This block receives the information coming from the INPUT_SIMULATION_BLOCK and the chosen legal speed and MINSPEEDLIMIT (they are not hardcoded in order to make simple change them from settings_block).
It contains the implementation in Matlab language of the complete algorithm described in the flowcharts, so the output consists of 3 parameters, that are V_MAX_HMI, V_MIN_HMI and popup.
V_MAX and V_MIN are the maximum and the minimum speed not truncated and so they are not sent to the HMI, but they are useful in the debugging process.

UcManager_SIMULATOR

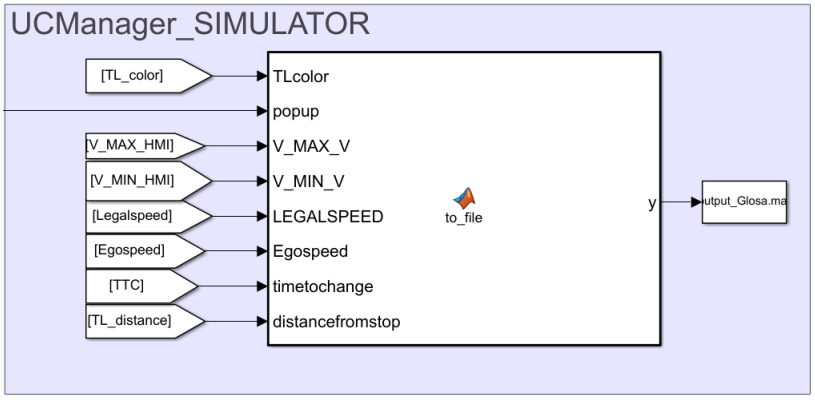


Figure 5.19: Focus on UcManager simulator block

This part of the model has the function to simulate the behaviour of the UcManager.
According to the input coming from the GLOSA_BLOCK it builds a matrix containing all the data for the HMI dispatcher and for the tablet, and write it on a ".mat" file.
This simulator is very important to test the algorithm because the generated matrix can be written on a file and then used for a simulation on the android tablet.

DRIVER_RESPONSE_SIMULATOR

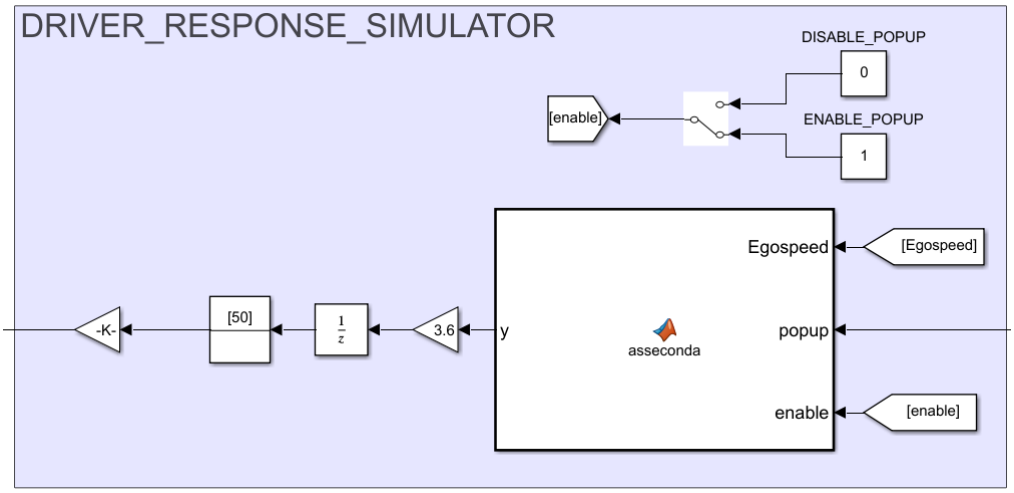


Figure 5.20: Focus on driver response simulator block

The "DRIVER_RESPONSE_SIMULATOR" tries to reproduce a possible reaction of the driver according to the popup that appears on the HMI.
In particular, the input speed is modified adding or subtracting an acceleration (this operation is possible because the simulation step is 1 second) which could be reasonable from the physic point of view.
For instance, if popup = "accelerate", the acceleration is set to 1 m/s^2, or if popup = "critical stop", acceleration is -2 m/s^2.
The output Egospeed is the one used in the next simulation step.

UDP_SOCKET_SEND_and_RECEIVE

This block allows to communicate with the framework running on the Step 03 board.
This data exchange is realized through two UDP sockets, in order to send and receive data from the C++ version of UC-GLOSA during the testing phase.

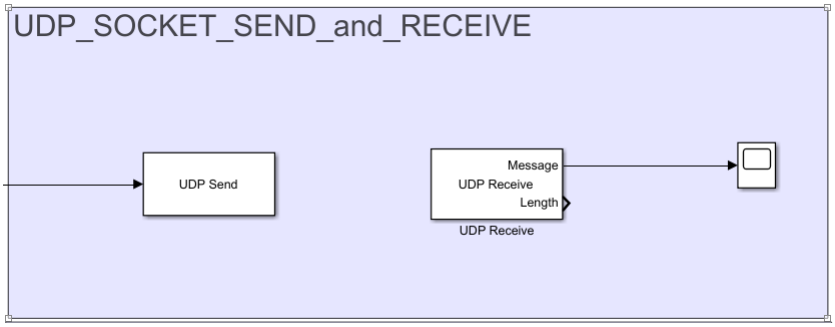


Figure 5.21: Focus on UDP_SOCKET_SEND_and_RECEIVE block

Data coming from the Framework are so plotted to verify their correctness.

MAIN_PLOT

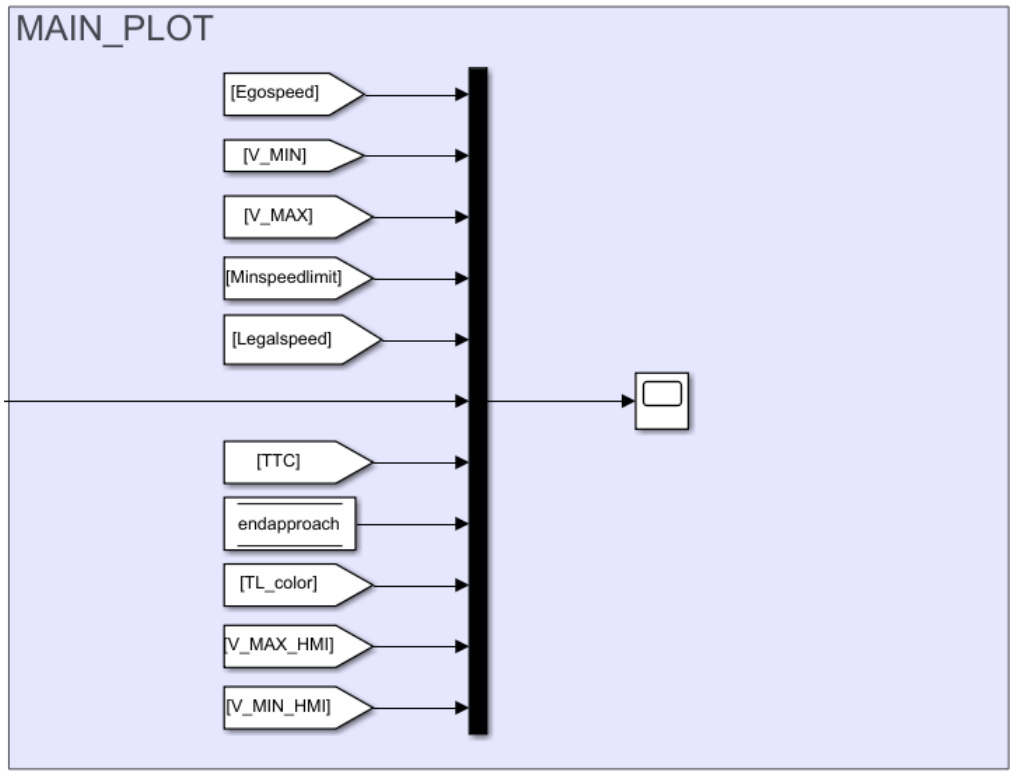


Figure 5.22: Focus on main plot block

This is the block which plots all variables coming from GLOSA_BLOCK and INPUT_SIMULATION_BLOCK. It allows to test the output in order to avoid errors.

SETTINGS

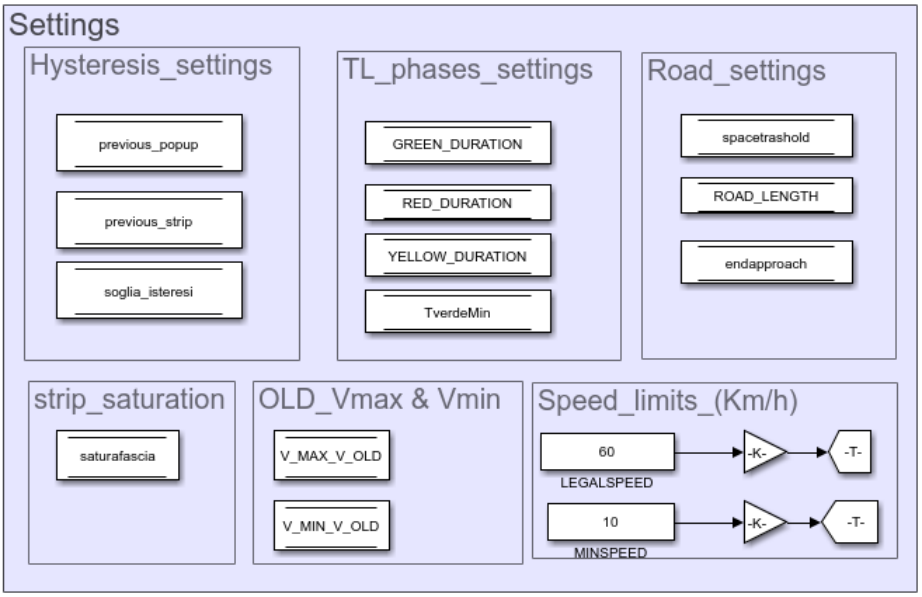


Figure 5.23: Focus on settings block

Through this block it is possible to change all the settings as regards the road (length , threshold for the space of stop), the traffic light (duration of each phase), legal speed and MINSPEEDLIMIT and parameters about the hysteresis.
There are also some global variables common to all blocks.

5.2.3 C++ implementation

For the C++ implementation of this Use Case the Kdevelop editor was used.
The Glosa class, as most of the other framework modules, derives from the class **MMv2x:Template**.
This choice has been done for uniformity reasons between the modules, because some methods like "threadProc()" (which starts all threads), "loadModuleConfigFile" (for the load of configuration files) or terminateThreads() (which allows to terminate all the threads of the framework in a common way through the handling of system signals), are managed in the same way.

GLOSA CLASS

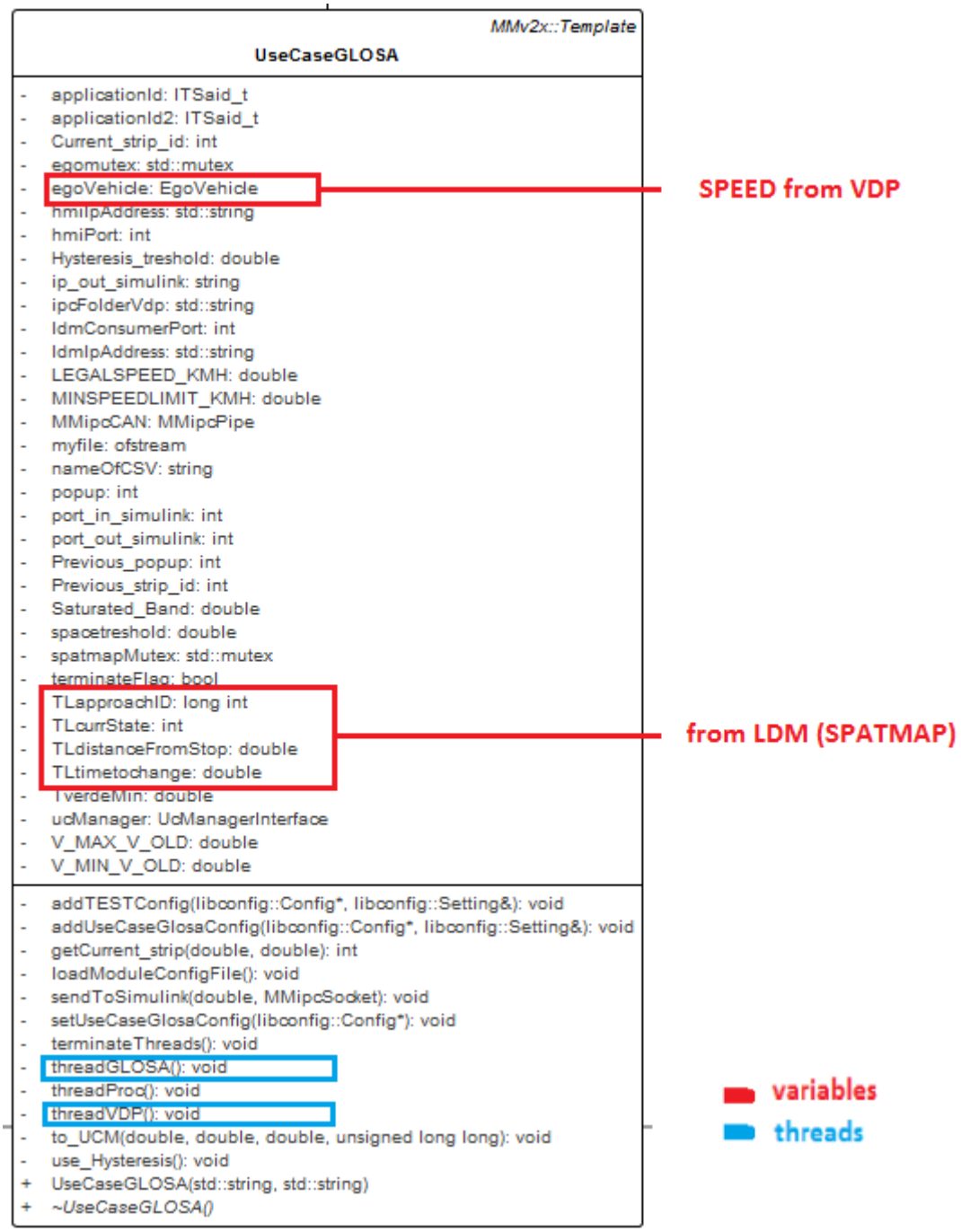


Figure 5.24: Glosa class and its main parts

The "one vehicle" version is basically composed by two threads:

- **threadGLOSA()**, which reads SPATMAP messages from the LDM and implements the Glosa algorithm.
- **threadVDP()**, which takes information from the VDP.

The following BPMN diagram shows how these two threads work:

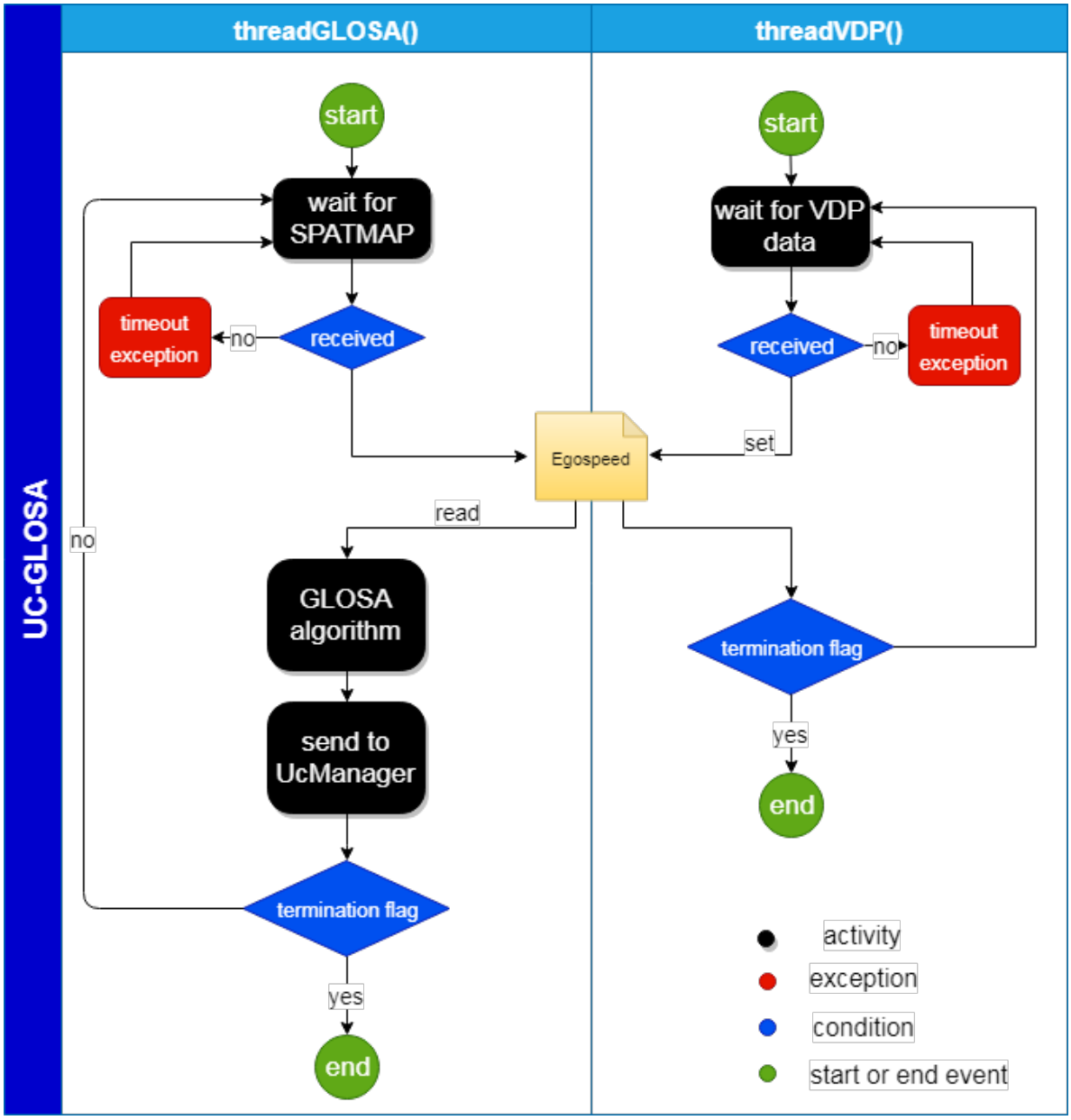


Figure 5.25: BPMN of the two threads of one vehicle version of GLOSA

INPUT AND OUTPUT

It is possible to compile this software with different options in order to change the input sources or the output direction.

Input can be generated from:

- **the framework**, through VDP (data from CAN) and LDM (SPATMAP).
- **Simulink**, through an UDP socket.

Output instead can be sent to:

- **UcManager**, which forwards it to HMI and tablet.
- **Simulink**, through another UDP socket.

5.3 Use Case GLOSA more vehicles version

Old Glosa versions are only based on the assumption that there are no vehicles between the Egovehicle and the traffic light.
This in most cases does not represents a real scenario, especially in an urban environment, in which with an higher probability there are one or more vehicles in front of Egovehicle.
As a matter of fact in a similar context the old Glosa algorithm loses part of its validity, in particular when the Use Case suggest to accelerate, but this action could lead to overcome the safety distance respect to the preceding vehicle.
So a new revision is required, in order to try to maintain the old functions but also paying attention to not give to the driver not safe information and,sometimes, generate warnings by other use cases like UC-FCW.

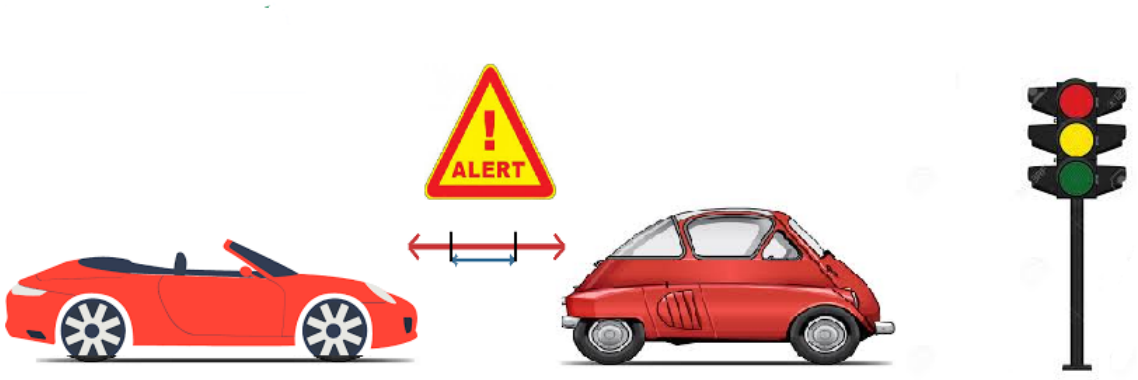


Figure 5.26: Vehicles distance

This new revision introduces another vehicle preceding the Egovehicle which has a simple dynamic:

- A constant speed (otherspeed)
- A starting distance from Egovehicle
- A zero acceleration

The aim is try to overcome the TL during the green phase paying attention to safety, that means to avoid collisions with other vehicles.

5.3.1 Algorithm

As in the section dedicated to the old algorithm, two tables containing a description of all input and output are shown below.

INPUT

NAME	SOURCE	UNIT	DESCRIPTION
Distance from stop	spatmap/Simulink	m	Distance from traffic light
Time to change	spatmap/Simulink	s	Time left to the next color
TL color	spatmap/Simulink	/	Current traffic light color
Egospeed	VDP/Simulink	m/s	Current vehicle speed
Legal speed	Config file or TIM	m/s	Maxi speed allowed by law
Minspeedlimit	Config file or TIM	m/s	Min reasonable speed
Ego_latitude	Poti mediator	°	Latitude of egovehicle
Ego_longitude	Poti mediator	°	Longitude of egovehicle
Other_latitude	BSM	°	Latitude of other vehicle
Other_longitude	BSM	°	Longitude of other vehicle

Table 5.3: Table of GLOSA input (more vehicles)

OUTPUT

NAME	UNIT	DESCRIPTION
V_MAX	Km/h	Upper limit of green window
V_MIN	Km/h	Lower limit of green window
Popup	/	Advice to the driver

Table 5.4: Table of GLOSA output (more vehicles)

A new popup type has been introduced:

popup = 5 => "Vehicle very nearly" in case of too short distance from in front vehicle.

This new version could be intended as a restriction of the previous one, so Glosa can suggest a valid green window only if a **safety distance** (more relaxed than the one used by UC FCW) is respected.

The indicator that represents the limit for giving valid GLOSA info derives from the safety distance's formula:

$$Safety\_distance = (Egospeed) + \frac{(Egospeed * Egospeed)}{2 * ego\_safety\_deceleration}$$

Ego_safety_deceleration has been set to 3 m/s^2 in order to have an higher safety distance.

Replacing the safety distance with the current distance between the two vehicles can be obtained ,for each time step, the maximum speed (**V_MAX_TEORICA**) that Egoveicle should follow to make safety distance equals than current distance.In formula (a,b and c are the coefficients of the equation):

$$\boldsymbol{V\_MAX\_TEORICA = \frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a}}$$

Where:

$$a = \frac{1}{2 * ego\_safety\_deceleration} \tag{5.1}$$

$$b = 1 \tag{5.2}$$

$$c = -vehicles\_distance \tag{5.3}$$

V_MAX_TEORICA represents the upper limit that Egospeed must not overcome to maintain a reasonable distance from in front vehicle, with the possibility to receive feasible information from GLOSA.
It directly reflects all the variations related to the distance between the two vehicles, so it take into account both vehicles speed.
If Egospeed become greater than V_MAX_TEORICA a new popup appears on the HMI ,to advise the driver that no GLOSA info are available because of the short distance from in front vehicle.

HYSTERESIS

As in the previous Glosa versions, hysteresis is used to avoid unpleasant alternations of different popups with conflicting indications to the driver.
According to different settings of hysteresis upper bound and lower bound, is easier to fall into danger situations and it is more difficult to come out from them.
In fact an higher upper bound permits to get far from V_MAX_TEORICA, with an higher possibility to receive feasible info from GLOSA and maintaining also a safety distance.
This conservative approach also reduces the possibility to get more serious warnings from UC-FCW, that is one of the main aims of this revision.

LOGICAL MODEL

Taking into account all this new constraints and corrections to the old version in order to follow the changes in the road scenario, it is possible to show the new Logical Model of the UC-GLOSA more vehicle version (figure 5.27):

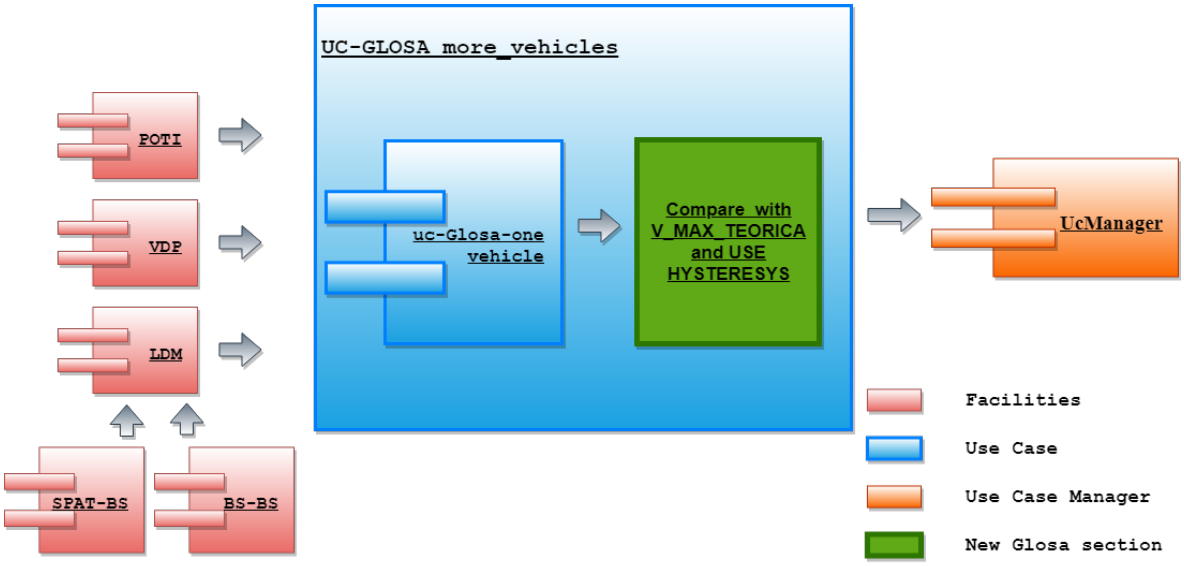


Figure 5.27: Logical model of UC-GLOSA more vehicles version

5.3.2 Simulation on Simulink

The building of the Simulink model associated to this version started from the existing "one vehicle" model with the addition of new features (figure 5.28):

- A simulation of another vehicle with a simplified behaviour (uniform motion).
- A simulation of the UC-FCW, in order to recognize when the distance between two vehicles is too small.

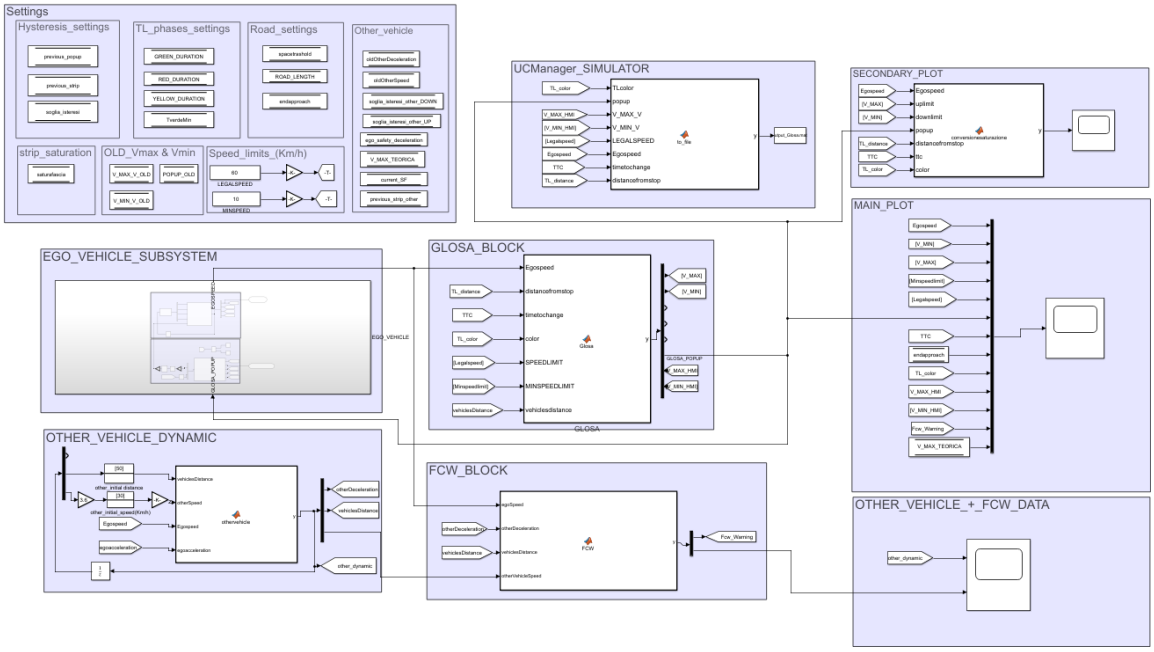


Figure 5.28: Simulink model of UC-GLOSA more vehicles version

The new blocks of the developed model are described below:

OTHER_VEHICLE_DYNAMIC

This block simulates another vehicle in front of the Egovehicle with a simplified uniform motion.
It is possible to set the uniform speed and the starting distance between the two vehicles.

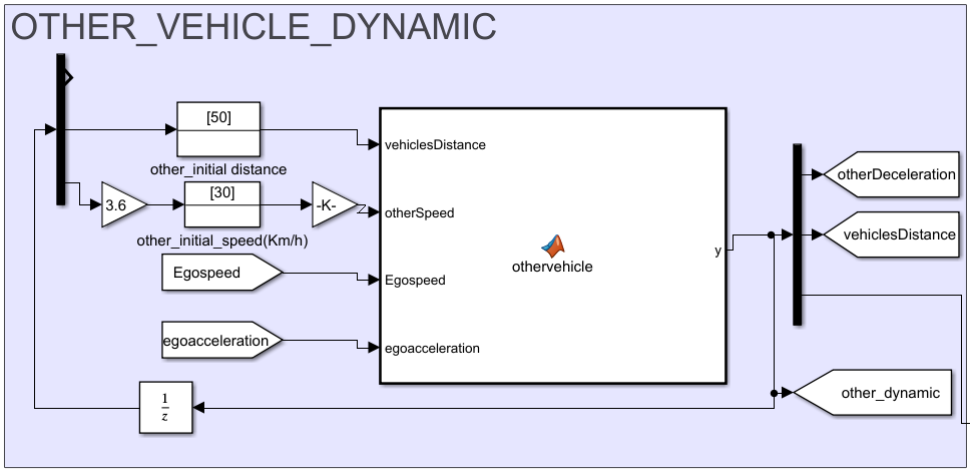


Figure 5.29: OTHER_VEHICLE_DYNAMIC Simulink block

The main function of "OTHER_VEHICLE_DYNAMIC" is to update the distance between the vehicles at each simulation step considering the Egospeed and the acceleration of the Egovehicle.

FCW_BLOCK

The FCW_BLOCK implements the logic of the UC-FCW, which gives a warning when the Egovehicle is too near to the preceding vehicle.

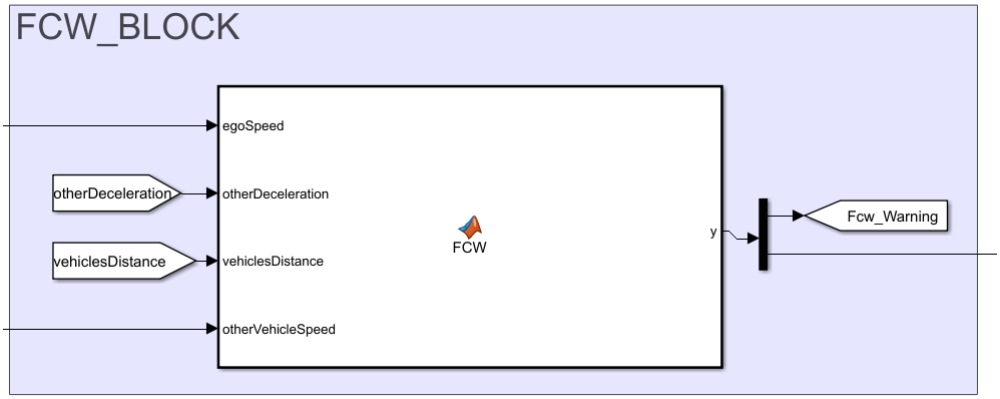


Figure 5.30: FCW_BLOCK Simulink block

SETTINGS_BLOCK

The SETTINGS_BLOCK has been improved to reflect the changes of the new version.

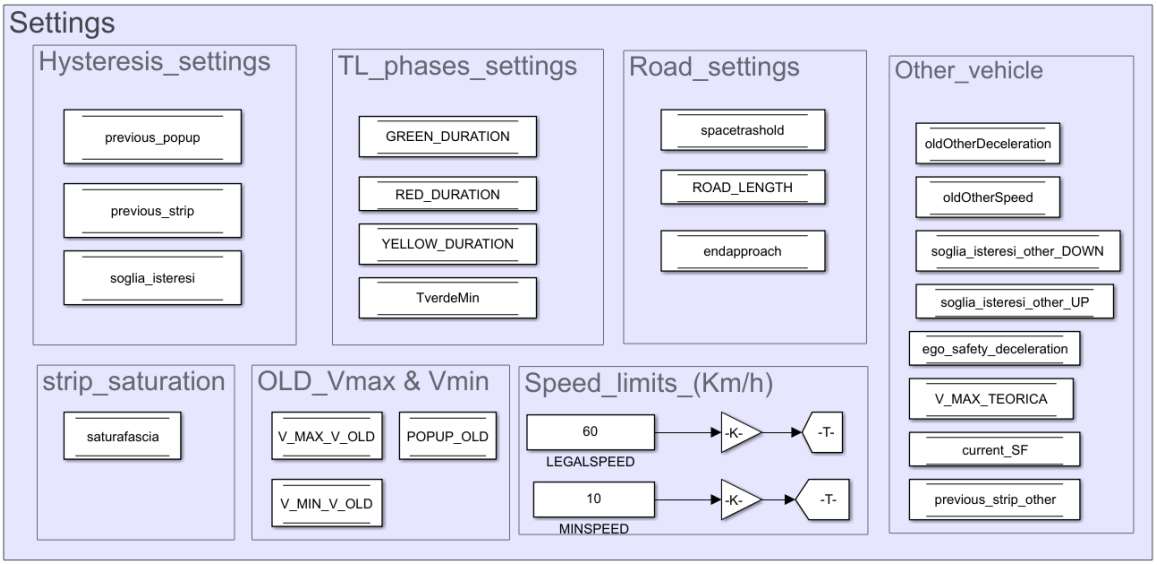


Figure 5.31: SETTINGS_BLOCK Simulink block

It is possible to see on the right of this block the section dedicated to the other vehicle and to the other parameters involved in the simulation(figure 5.31).

EGO_VEHICLE_SUBSYSTEM

This subsystem incorporates the two already existing blocks that concern the Egovehicle(figure 5.32):

- "EGO_INPUT_SIMULATION_BLOCK"
- "EGO_DRIVER_RESPONSE_SIMULATOR"

The only change respect to the previous version is that the Egovehicle is able to react to the new warning coming from FCW_BLOCK, which implies an high deceleration to get far from a danger situation.

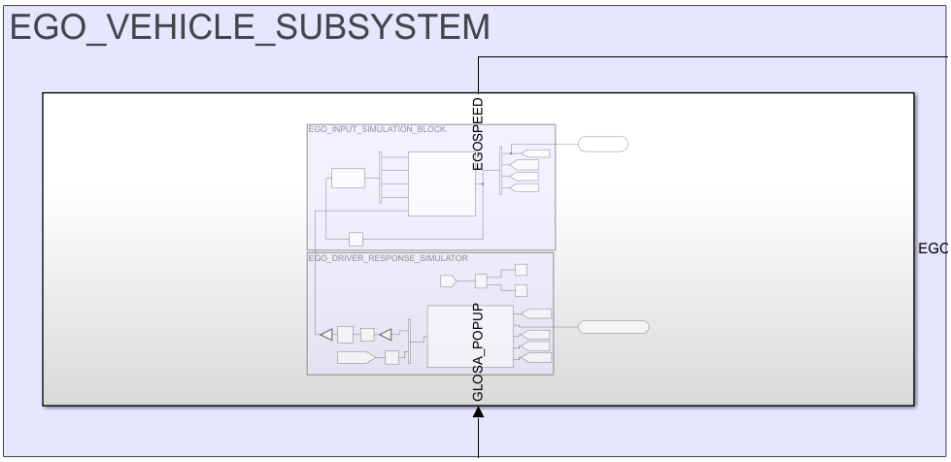


Figure 5.32: EGO_VEHICLE_SUBSYSTEM Simulink block

OTHER_FCW_PLOT

This block has the function to plot all data concerning the UC-FCW and the dynamic of the other vehicle.

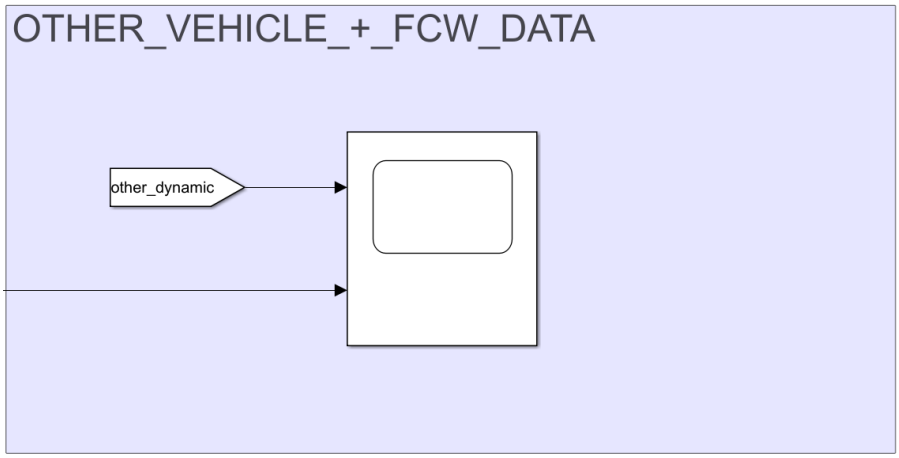


Figure 5.33: OTHER_FCW_PLOT Simulink block

5.3.3 C++ implementation

Here is the class diagram of the new version of UC-GLOSA, with the new threads and variables:

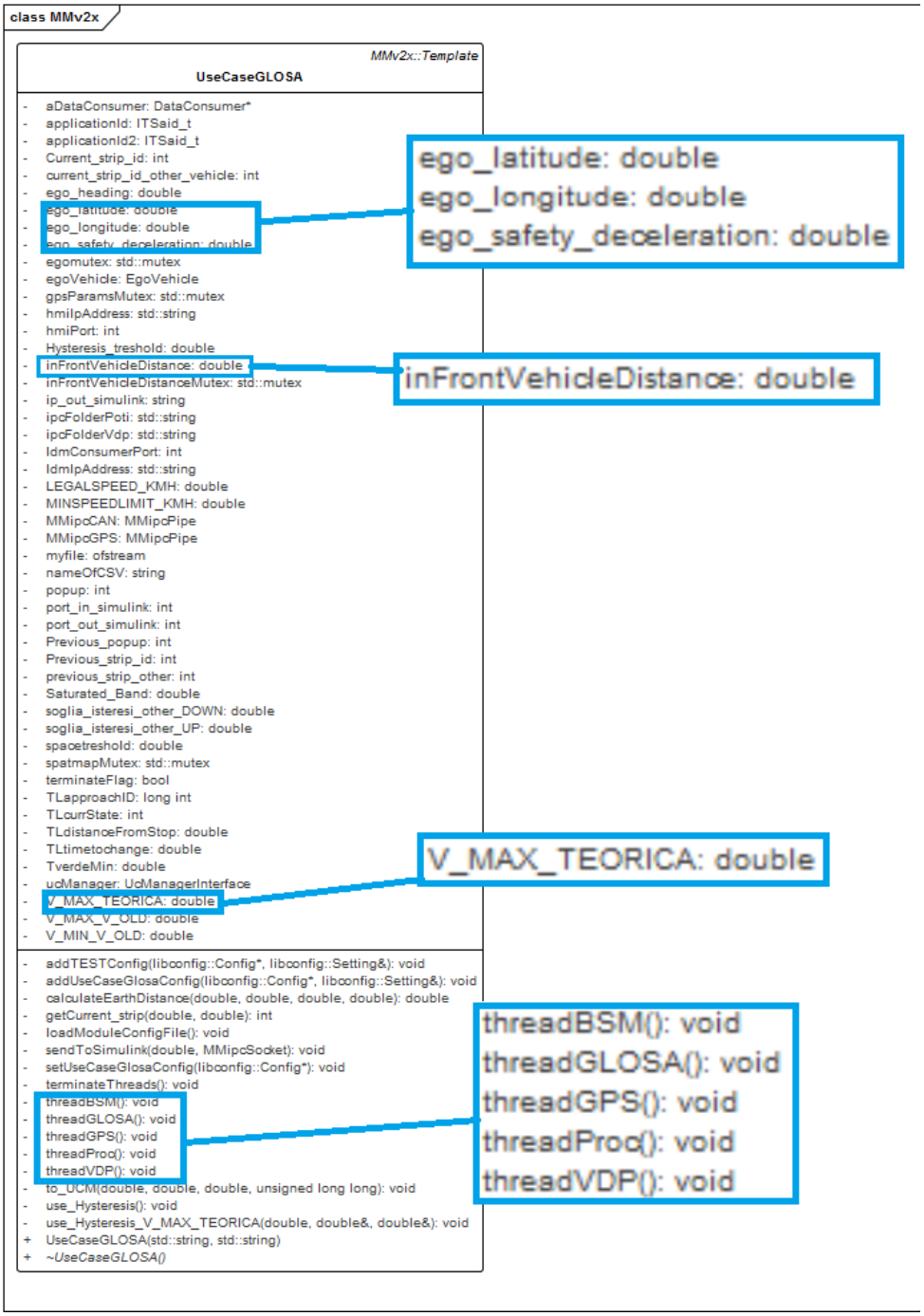


Figure 5.34: GLOSA class more vehicles version

THREADS

The "more vehicles" version is basically composed by 4 threads:

- **threadGLOSA()**:
  This is the main thread. It starts with the subscription to LMD in order to receive SPATMAP messages and thread is locked until a new SPATMAP arrives or the timeout expires.
  In case of a new SPATMAP the next step is to read the Egospeed from a shared variable (in mutual exclusion) and the application of the old Glosa version to obtain V_MAX, V_MIN and popup.
  Then the current distance between the two vehicles is read from another shared variable and it is used to calculate V_MAX_TEORICA. By using this upper speed limit it is possible to do a check by verifying if Egospeed is higher or lower than it and by changing the popup if needed (an hysteresis is applied in order to avoid alternating popups with different advice).

- **threadVDP()**: This thread starts with the subscription to CANDataprovider, with the aim to get , when it is available, the current speed of Egovehicle.
  When the data arrives the thread unlocks and write in a shared variable the obtained value. After that it starts waiting again.

- **threadBSM()**: It starts with the subscription to LMD in order to receive BSM messages and thread is locked until a new BSM arrives or the timeout expires.
  In case of a new BSM, latitude and longitude of the other vehicle are read. The next step is to get from two shared variables the latitude and the longitude of Egovehicle and through the **calculateEarthDistance()** method (which accept the two latitudes and longitudes) it calculates the distance between the two vehicles and writes it in a shared variable.

- **threadGPS()**: It takes information from the POTI Mediator in the already described way and write latitude and longitude of the Egovehicle in a shared variable.

The following BPMN shows in details how the threads of UC-GLOSA work,as regards the new version.

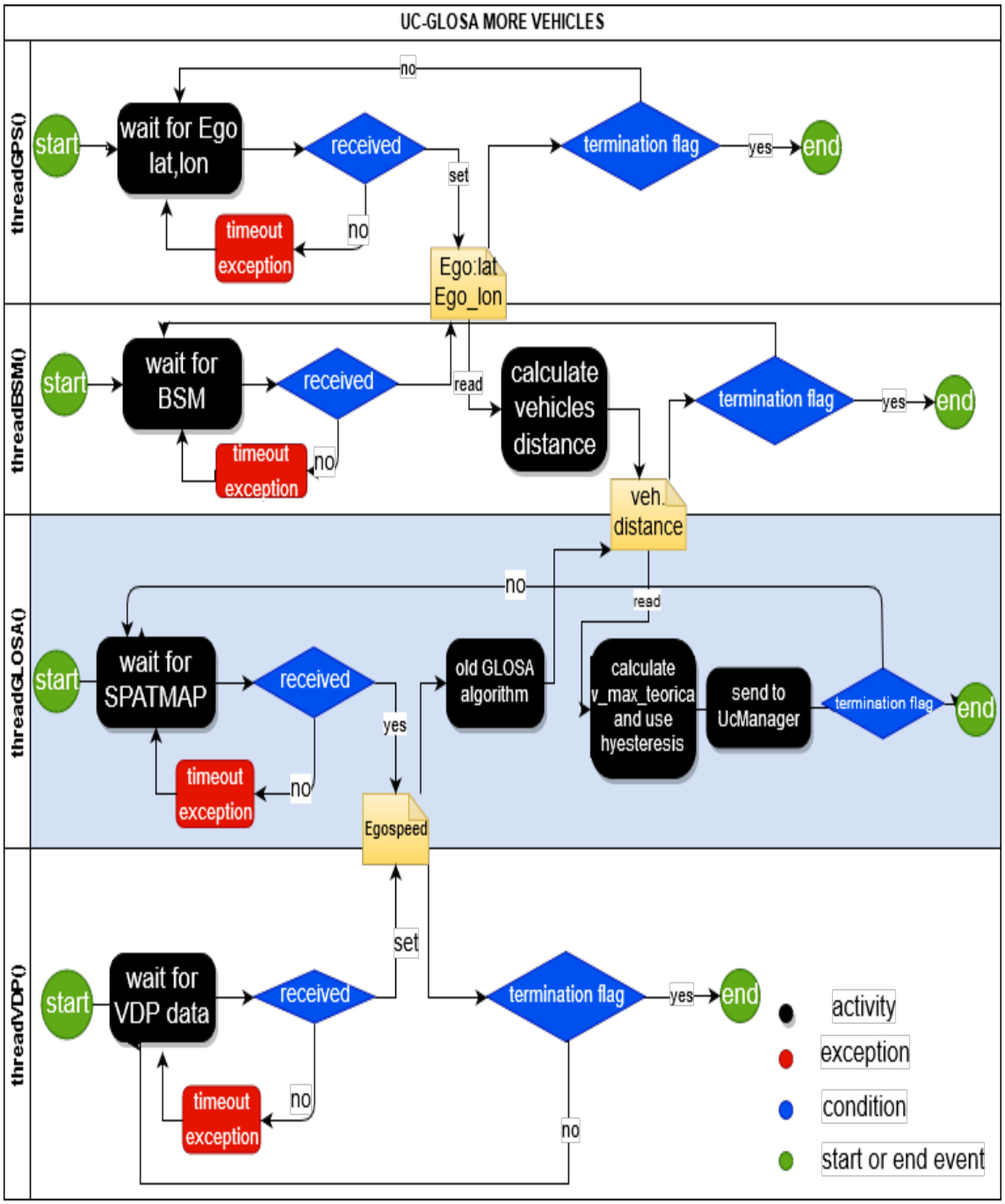


Figure 5.35: BPMN more vehicles version

As in the previous version, UC-GLOSA for more vehicles can be set in different ways. As a matter of fact it is possible to compile this software with different options in order to change the input sources or the output direction.

Input can be generated from:

- **the framework**, through VDP (data from CAN), LDM (SPATMAPs and BSMs) and POTI Mediator.
- **Simulink**, through an UDP socket.

Output instead can be sent to:

- **UcManager**, which forwards it to HMI and tablet.
- **Simulink**, through another UDP socket.

Chapter 6

Testing and Validation

6.1 Testing e Validation

6.1.1 Bench tests

Bench tests have the function to verify the correctness of the output given by UC-GLOSA in order to fix any bugs that could affect the software.
These kinds of test have been introduced because it was very uncomfortable to try the Use Case directly on road, because of the difficulty to recognize errors in real time.
Bench tests have been divided into two different phases:

- Tests done only using Simulink.
- A combination of the Simulink model and the C++ version running on the Step 03 board.

TESTS ON SIMULINK

The first phase of the bench tests consists of a set of simulations (TEST CASES) done each time by using different starting conditions.
These test cases are divided basing on the starting distance from the traffic light:

- **TEST CASE 1** : distance from stop 400 m
- **TEST CASE 2** : distance from stop 200 m
- **TEST CASE 3** : distance from stop 100 m

Each TEST CASE is composed by several simulations characterized by the same distance from the traffic light but different starting Egospeed (in a range from MIN-SPEEDLIMIT to legal speed), in order to try to stimulate the major number of cases of the algorithm.
Output is visualized on a Simulink "scope" and so it possible to verify if there are anomalies in the simulation. The image below shows an example of this output on a "simulink scope"(figure 6.1).

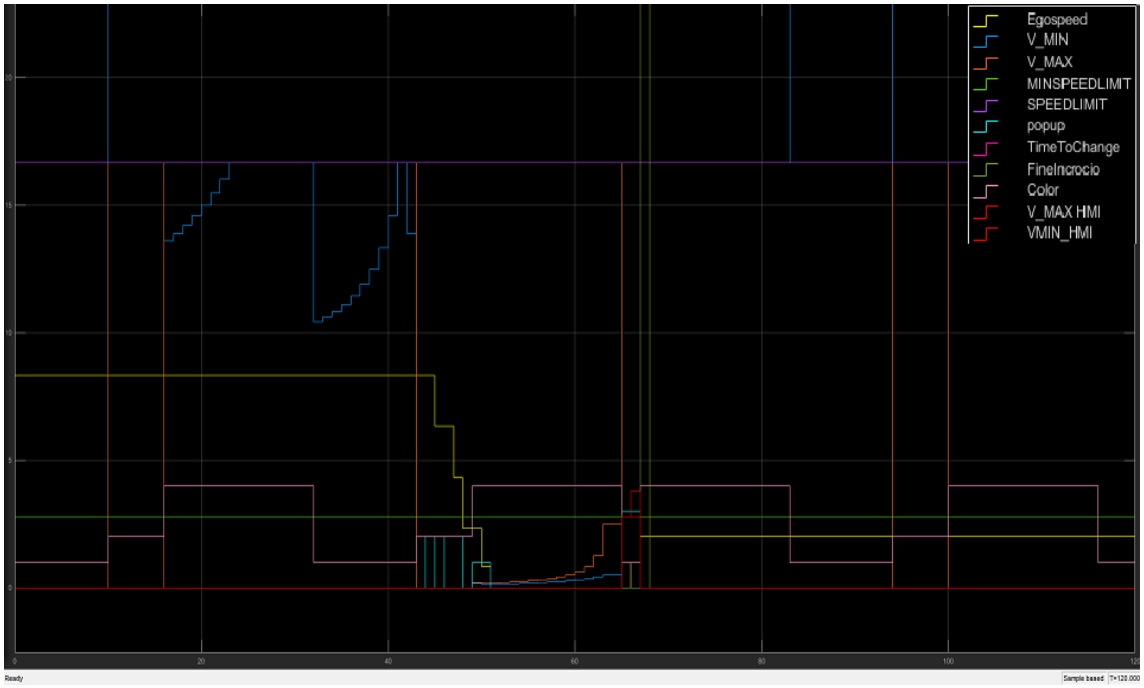


Figure 6.1: Example of output on a Simulink scope

Where:

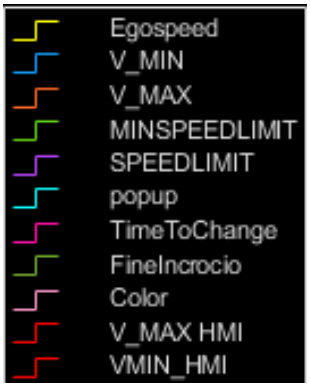


Figure 6.2: Legend of a Simulink scope

This is the output of the already described MAIN_PLOT of the Simulink model. In this particular scope the following variables (for each simulation step that is 1 second) are shown:

- Egospeed
- V_MIN : the real one directly deriving from the formula.
- V_MAX : the real one directly deriving from the formula.
- MINSPEEDLIMIT
- Legal speed
- Popup
- Time to change
- Fine incrocio : a variable that is always 0 and has a peak when distance from stop become null.
- Color : the current phase of the traffic light.
- V_MIN HMI : the lowest possible green speed visualized on the HMI.
- V_MAX HMI : the highest possible green speed visualized on the HMI.

The following table is an example of TESTCASE 1:

TESTCASE	Egospeed (km/h)	Distancefromstop (m)	abnormalities
1	10(MINSPEED)	400	NO
1	20	400	NO
1	30	400	NO
1	40	400	NO
1	50	400	NO
1	60	400	NO

Figure 6.3: Example of a test case using only Simulink

This test case concerns the situation in which there is a fixed distance from stop of 400 metres and an increasing Egospeed from MINSPEEDLIMIT to Legal speed. Two similar tests have been done, each time by halving the distance from the traffic light and with an similar Egospeed behaviour.

TESTS ON SIMULINK AND C++

The second step of the bench tests about UC-GLOSA consists of 3 test cases, with the aim to verify two output:

- Output directly from Simulink
- Output from the C++ UC-GLOSA running on the Step 03

To be able to compare in the right way these two output it is necessary to visualize the output of the "C++ version" in a "Simulink Scope" which is part of the simulink model.

Tests follow the below schema:

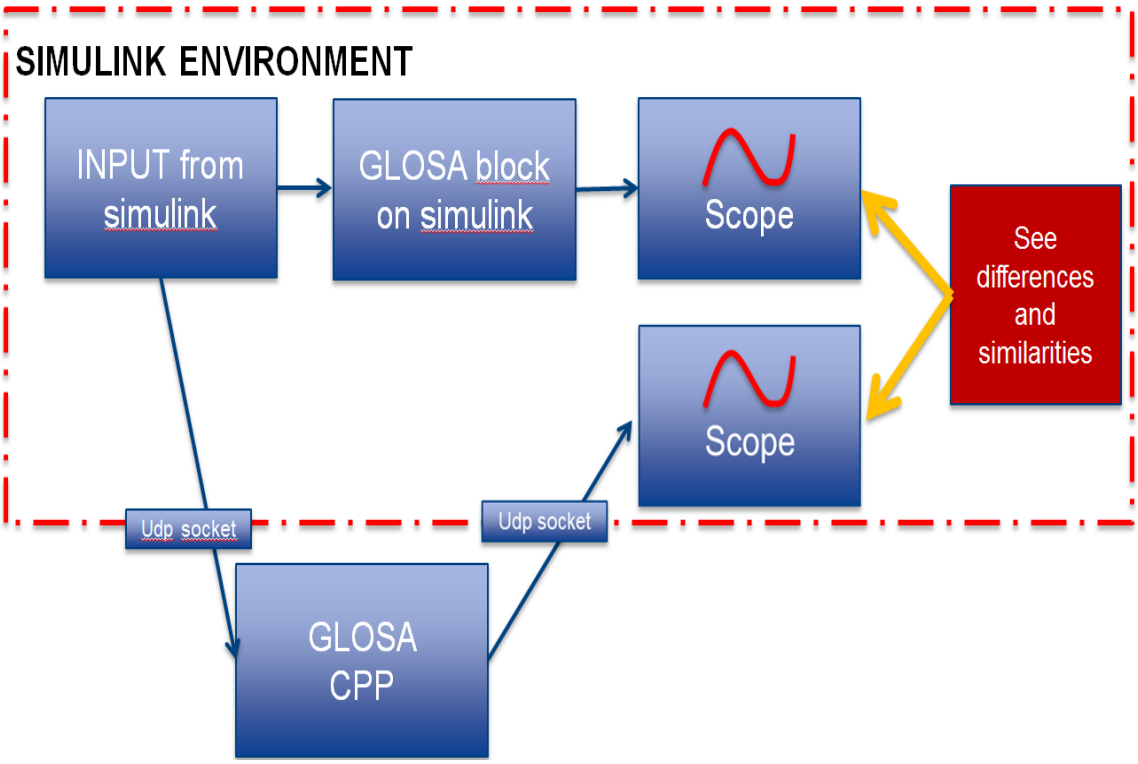


Figure 6.4: Schema of the bench tests using both Simulink and C++

As is possible to notice from this schema (figure 6.4), the Simulink environment must be able to communicate with the Step 03 board.
This is made possible by using two UDP sockets which transport data in and out from the Simulink model.
It is important that the PC on which Simulink is running and the Step 03 are in the same LAN, in order to make UDP protocol working in the right way.

As regards the UDP ports these are the settings:

- UDP SOCKET FROM SIMULINK port 8100

- UDP SOCKET TO SIMULINK port 8101

By comparing the two Simulink scopes it is possible to discover possible differences caused by a wrong C++ implementation.

The following images are an example of the output that shall be compared by setting a starting Egospeed and distance from stop:

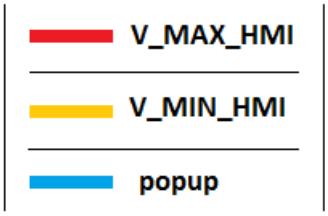
TESTCASE	Egospeed (km/h)	Distancefromstop (m)	abnormalities
2	10(MINSPEED)	200	NO
2	20	200	NO
2	30	200	NO
2	40	200	NO *(yellow)
2	50	200	NO
2	60	200	NO

Figure 6.5: TEST case 2 Simulink and C++

In this case we have a Distance from stop of 200 metres and an initial egospeed of 30 Km/h.

These are the corresponding output:

When "green window" exists, the red line is V_MAX on HMI, the yellow line is V_MIN on HMI while the blue line represents the popup.



C++

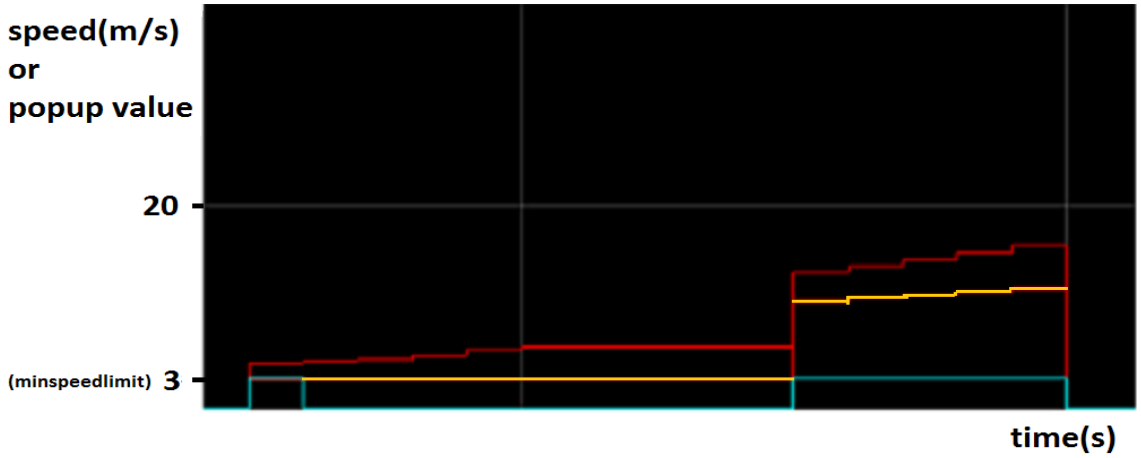


Figure 6.6: C++ output on Simulink example

SIMULINK

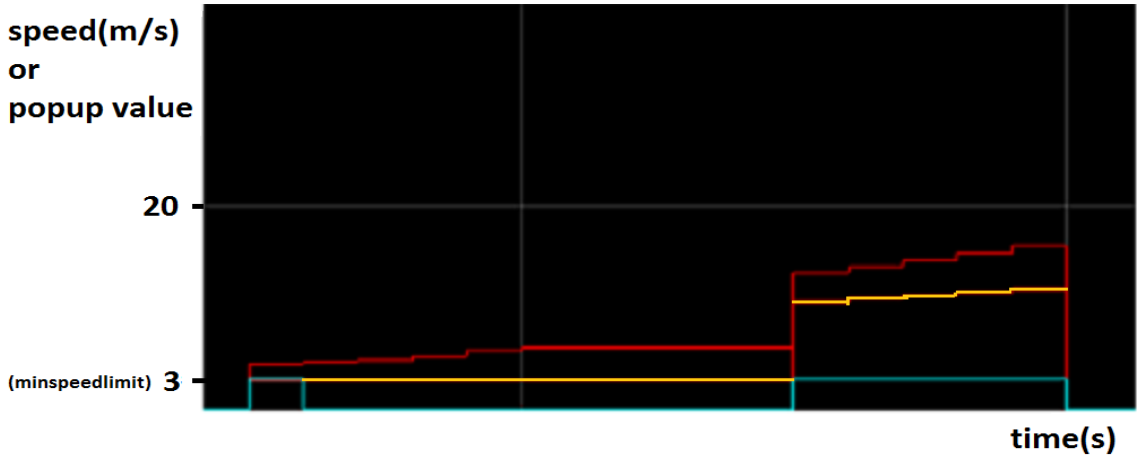


Figure 6.7: Simulink output example

6.1.2 On road tests

After the bench tests phase it is possible to do the on road tests and check if the use case works in the right way.
The necessary hardware to realize these tests is composed by:

- A **Step 03** board with the Magneti Marelli Connectivity Framework running on it
- A **RSU** (Road Side Unit) placed on the car
- A wi-fi **switch**
- A **PC-laptop** to monitoring the software
- An Android **tablet**

RSU configuration

Before on road tests it is important to configure the RSU to generate MAP and SPAT messages in order to simulate a real RSU placed on a traffic light.
This operation is possible by modifying the configuration files loaded on the MK5, in paticular:

- **"config_MM_WAVE_RSUbackend_MAPconfig_params.cfg"**
- **"config_MM_WAVE_RSUbackend_SPATconfig_params.cfg"**

MAP:
MAP has been defined by fixing a reference point and two map areas, one for each lane(figure 6.8).

POINT	LAT	LON
REF_POINT	45.140654	7.614455
TO_PARK_STOP	45.140584	7.614751
TO_PARK_START	45.140231	7.615881
FROM_PARK_STOP	45.140700	7.614235
FROM_PARK_START	45.141227	7.612509

Figure 6.8: Map configuration for tests

These points are located in the avenue adjacent to the Magneti Marelli headquarters of Venaria Reale(figure 6.9).:

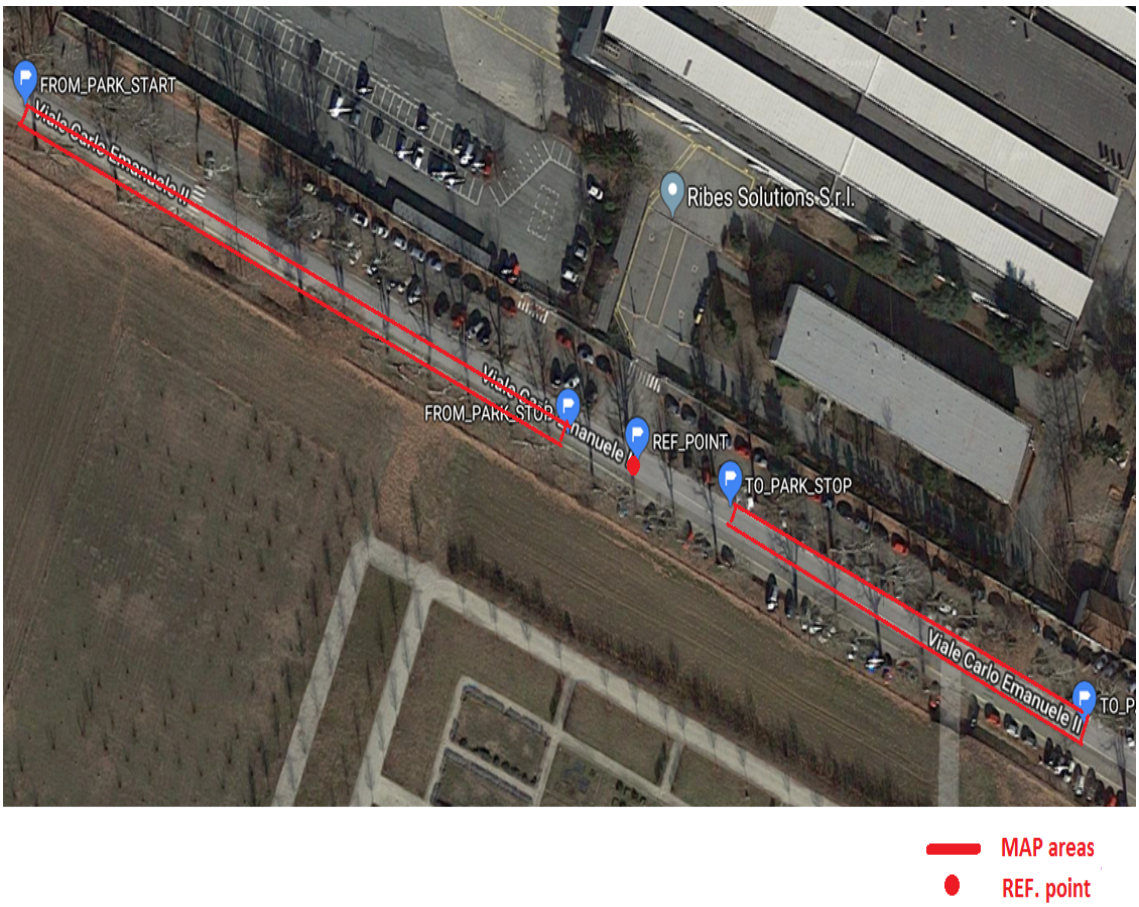


Figure 6.9: Map areas in Viale Venaria

These points have been inserted in the configuration file in the following order:

- REFERENCE POINT (Intersection ID,Latitude and Longitude)
- APPROACH NORTHBOUND (Intersection ID,Latitude and Longitude)
- APPROACH SOUTHBOUND (Intersection ID,Latitude and Longitude)

This is the configuration file on the RSU that defines MAP areas:

```
MAP_COMPOSER :
{
  INTERSECTION_ID = 3118;
  REF_POINT_LAT = 45.140654;
  REF_POINT_LON = 7.614455;
  APPROACH_NORTHBOUND :
  {
    ENABLED = 1;
    APPROACH_ID = 1;
    OFFSET_A_LAT = 45.140584;
    OFFSET_A_LON = 7.614751;
    OFFSET_B_LAT = 45.140231;
    OFFSET_B_LON = 7.615881;
  };
  APPROACH_SOUTHBOUND :
  {
    ENABLED = 1;
    APPROACH_ID = 2;
    OFFSET_A_LAT = 45.140700;
    OFFSET_A_LON = 7.614235;
    OFFSET_B_LAT = 45.141227;
    OFFSET_B_LON = 7.612509;
  };
  APPROACH_EASTBOUND :
  {
    ENABLED = 0;
    APPROACH_ID = 3;
    OFFSET_A_LAT = 45.0;
    OFFSET_A_LON = 8.0;
    OFFSET_B_LAT = 45.0;
    OFFSET_B_LON = 8.0;
  };
  APPROACH_WESTBOUND :
  {
    ENABLED = 0;
    APPROACH_ID = 4;
    OFFSET_A_LAT = 45.0;
    OFFSET_A_LON = 8.0;
    OFFSET_B_LAT = 45.0;
    OFFSET_B_LON = 8.0;
  };
};
```

SPAT:

According to the configuration file, SPAT messages have been set in the following way:

- **Frequency** = 2Hz , that means 2 SPATs per second
- **Red phase duration** = 60 s
- **Yellow phase duration** = 5 s
- **Green phase duration** = 40 s
- **Initial phase** = RED

This is the configuration file on the RSU that defines SPAT message:

```
SIM_SPAT :
{
ENABLED = true;
FREQUENCY_HZ = 2;
TIMETOCHANGE_GREEN_SEC = 40;
TIMETOCHANGE_YELLOW_SEC = 5;
TIMETOCHANGE_RED_SEC = 60;
INITIAL_PHASE = "RED";
};
```

ON BOARD CONFIGURATION

The needed components for these tests have been linked in a wired LAN through an ethernet switch. The Android tablet is connected to the LAN by using a wi-fi 802.11a/b/g/n switch, while SPATs and MAPs are sent using a DSRC (802.11p) communication(figure 6.10).

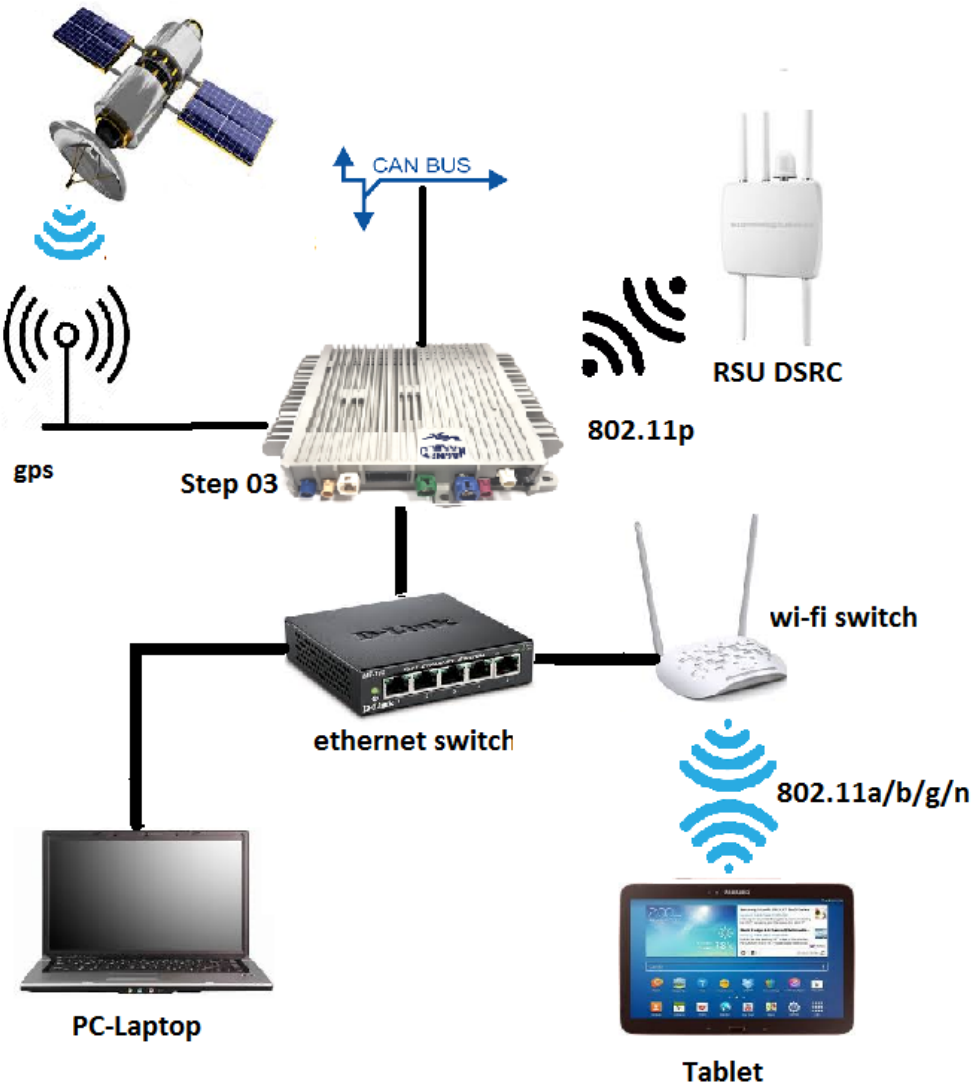


Figure 6.10: Hardware schema for on road test

LOG FILES

During on road test thanks to the PC has been possible to monitoring what was happening by observing the log file written by UC-GLOSA in real time.
In fact, for each time step (1 second), the following information were visualized:

- V_MAX (the one sent to HMI)
- V_MIN (the one sent to HMI)
- Time to the next phase
- Distance from traffic light
- Egospeed
- Traffic light current color
- Popup with relative advice
- Information about the distance from traffic light (to discriminate if the popup is for maintaining speed or for "not available advice"

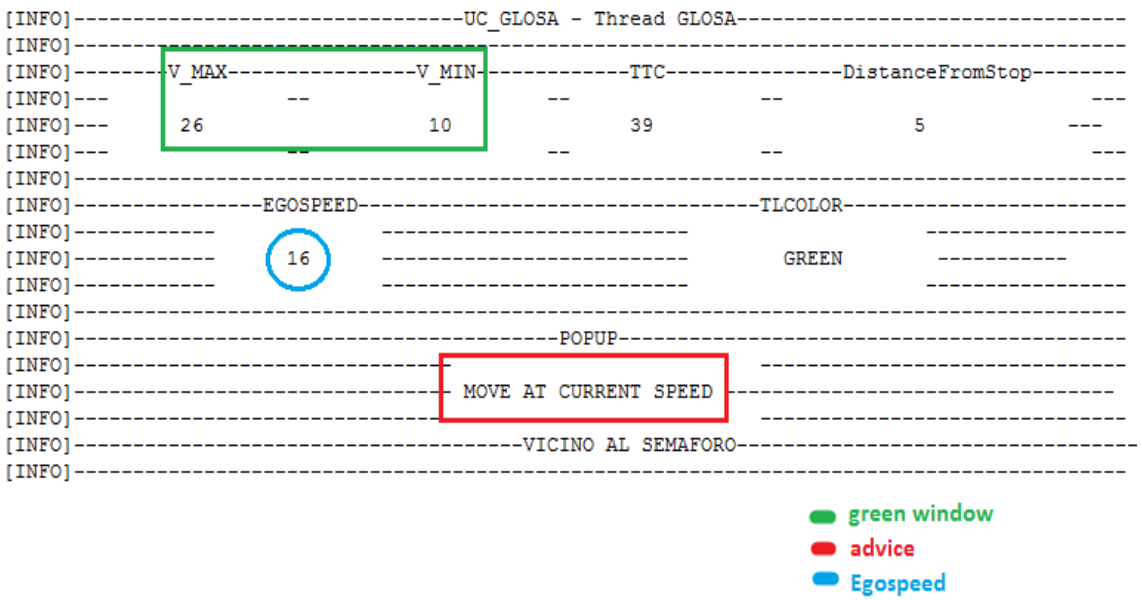


Figure 6.11: UC-GLOSA log example

Chapter 7

Results

7.1 Results

In this thesis a software architecture, named Use Case Glosa (UC-GLOSA), was designed and implemented.
This software is integrated in the Magneti Marelli Connectivity Framework, a solution for enabling Vehicle-to-Everything (V2X) communication, with the aim to try reducing problems related to traffic efficiency, safety and pollution.
The developed software is framed in the field of traffic efficiency and tries to resolve those problems related to the waiting time at intersections and consumption.
The UC-GLOSA software extended and improved the already existing UC-TRAFFIC-LIGHT, a simpler use case which shows the current phase of a traffic light and the time left to the next phase, by adding more useful information.

The thesis work lead to the development of this Use case in some steps:

- A research phase in order to know what already exists about these software types.

- The drafting of an algorithm which in the simplest possible way could find a feasible solution to these problems, taking into account both safety and driving comfort.

- The building of a simulation environment which makes no longer necessary to do frequent on road tests, in order to easily recognize issues in the algorithm and to avoid logistic problems.

- The implementation of a C++ software which is part of a complex embedded framework and which interacts with other modules.

The final result is an advice to the driver about what is the best action to do when he is approaching an intersection regulated by a traffic light and, if it is possible, a range of speeds that allows to overcome it during the green phase.
All this is done by paying particular attention to safety.
All the advice which are given as output,at this moment, are only sent to the v2x software installed on the tablet, while the car HMI shows only the already implemented UC-TRAFFIC LIGHT information.
The following images represent two possible final output on the tablet:

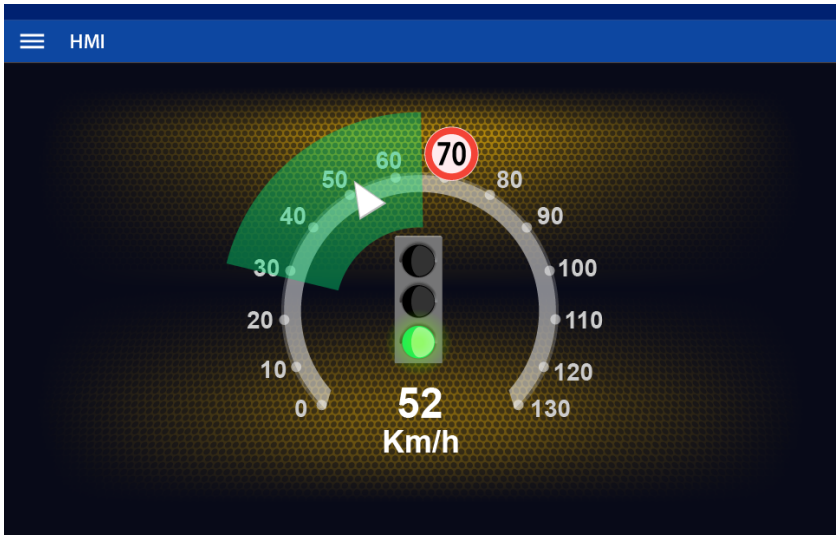


Figure 7.1: Glosa output on tablet in case of green window

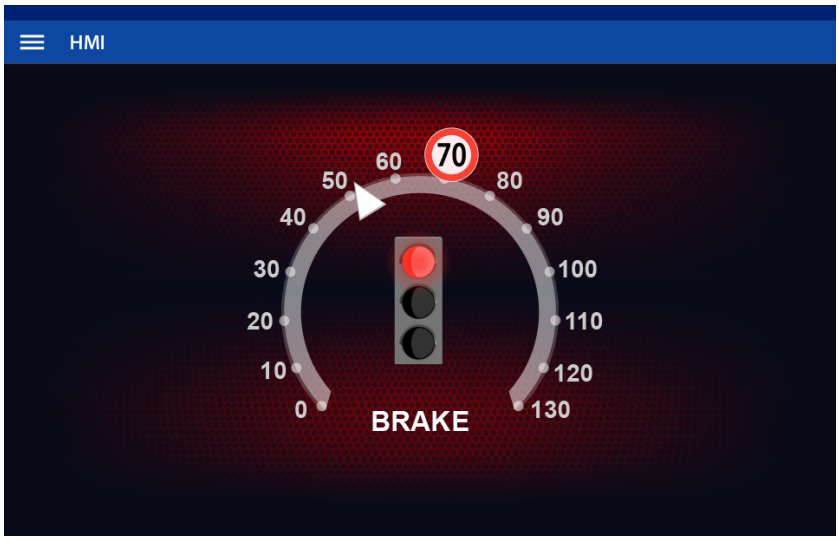


Figure 7.2: Glosa output on tablet in case of critical stop

Chapter 8

Future improvements

8.1 Future improvements

The work done for this thesis could be improved under different points of view:

- **Algorithm :**
  A possible change could be use the hysteresis not only to decide between no popup or critical stop, but also in other cases, for example when Egospeed is close to V_MAX or V_MIN, in order to maintain coherent and pleasant advice to the driver.

- **Simulink model :**
  As regards the Simulink model, an improvement could be to modify the vehicle reaction to the GLOSA output to make it more realistic. The same thing could be done for the other vehicle dynamic.
  Another possible change is to introduce the logic of other use cases to monitor possible different behaviour of the GLOSA One.

- **C++ software :**
  The C++ version could be improved by modifying the threadBSM() in order to filter the received BSMs and select only those coming from the in front vehicle.

Bibliography

[1] *World report on road traffic injury prevention*. URL: https://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/summary_en_rev.pdf (visited on 03/05/2019).

[2] *Creating the Clean Energy Economy - Analysis of the Electric Vehicle Industry*. URL: http://www.iedconline.org/clientuploads/Downloads/edrp/IEDC_Electric_Vehicle_Industry.pdf (visited on 03/05/2019).

[3] *Magneti Marelli website*. URL: https://www.magnetimarelli.com/it (visited on 01/02/2019).

[4] Imad Jawhar, Nader Mohamed, and Hafsa Usmani. "An Overview of Inter-Vehicular Communication Systems, Protocols and Middleware". In: *Journal of Networks* JOURNAL OF NETWORKS, VOL. 8, NO. 12, DECEMBER 2013 (Dec. 2013). DOI: 10.4304/jnw.8.12.2749-2761.

[5] John Kenney. "Dedicated Short-Range Communications (DSRC) Standards in the United States". In: *Proceedings of the IEEE* 99 (Aug. 2011), pp. 1162–1182. DOI: 10.1109/JPROC.2011.2132790.

[6] *IEEE organisation*. URL: https://www.ieee.org/ (visited on 01/02/2019).

[7] *ETSI organisation*. URL: https://www.etsi.org/ (visited on 01/02/2019).

[8] *ETSI EN 302 637-2 V1.3.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf (visited on 01/02/2019).

[9] *ETSI EN 302 637-3 V1.2.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.01_30/en_30263703v010201v.pdf (visited on 01/02/2019).

[10] *C++ description*. URL: http://www.cplusplus.com/info/description/ (visited on 01/02/2019).

[11] *Cross-compilation definition*. URL: https://en.wikipedia.org/wiki/Cross_compiler (visited on 01/02/2019).

[12] *Apache Subversion*. URL: https://en.wikipedia.org/wiki/Apache_Subversion (visited on 01/02/2019).

[13] *Matworks' Matlab*. URL: https://uk.mathworks.com/products/matlab.html (visited on 01/02/2019).

[14] *Matworks' Simulink*. URL: https://uk.mathworks.com/products/simulink.html (visited on 01/02/2019).

[15] *ENTERPRISE ARCHITECT*. URL: https://sparxsystems.com/products/ea/ (visited on 01/02/2019).

[16] *SUMO- Simulation of Urban MObility*. URL: http://sumo.dlr.de/index.html (visited on 01/02/2019).

[17] *Cohda Wireless' MK5*. URL: https://cohdawireless.com/solutions/hardware/mk5-obu/ (visited on 02/28/2019).

[18] Konstantinos Katsaros Ralf Kernchen Mehrdad Dianati David Rieck Charalampos Zinoviou. *Application of Vehicular Communications for Improving the Efficiency of Traffic in Urban Areas*. University of Surrey Guildford, 2011.

[19] Kugamoorthy Gajananan Sra Sontisirikit Jianyue Zhang Marc Miska Edward Chung Sumanta Guha Helmut Prendinger. *A Cooperative ITS study on green light optimisation using an integrated Traffic, Driving, and Communication Simulator*. National Institute of Informatics,Asian Institute of Technology,Smart Transport Research Centre, Queensland University of Technology, Brisbane, 2013.

[20] Biao XU Fang ZHANG Jianqiang WANG Keqiang LI. *B&B Algorithm Based Green Light Optimal Speed Advisory Applying to Contiguous Intersections1*. Chinese National Program for High Technology Research and Development, 2015.