D5.6 – TeamMate Car Demonstrator after 3rd Cycle

Project Number:	690705
Classification **Deliverable No.:**	Public D5.6
Work Package(s): **Document Version:** **Issue Date:** **Document Timescale:** Start of the Document: Final version due:	WP5 Vs. 0.1 (draft) Month 36 Project Start Date: September 1, 2016 Month 33 Month 36
Deliverable Overview:	**Main document:** D5.6
Compiled by:	CRF
Authors:	Fabio Tango, CRF
Technical Approval:	Fabio Tango, CRF
Issue Authorisation:	Andreas Lüdtke, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	AutoMate Consortium	all AutoMate Partners

RECORD OF REVISION		
Date	Status Description	Author
19/07/2019	Structure proposal	F. Tango
02/08/2019	First partners contributions	F. Tango, M. Fossanetti (CRF); E. Yousfi (VED); M. Graf (ULM); A. Castellano, E. Landini (REL).
09/09/2019	Other partners contributions	M.C. Rahal, J. Pichen
13/09/2019	Deliverable final version (ready for submission)	F. Tango
24/09/2019	Deliverable final version (ready for submission)	F. Tango

[1] Copy types: E=Email, C=Controlled copy (paper), D=electronic copy on Disk or other medium, T=Team site

Table of Contents

List of Figures

List of Tables

List of Acronyms

SW = Software

HW = Hardware

TM = TeamMate

AD = Autonomous Driving

AV = Autonomous Vehicle

PP = Path Planning

TP = Trajectory Planning

AR = Augmented Reality

HMI = Human Machine Interaction

HUD = Head-Up Display

DIR = Driver Intention Recognition

DMS = Driver Monitoring System

ADF = Autonomous Driving Function

ORA = Online Risk Assessment

CAN = Controller Area Network

GPS = Global Positioning System

ADAS = Advanced Driving Assistance System

1 Executive Summary

This deliverable describes the integration of the SW/HW components for the TeamMate technologies, as derived by WP2/3/4, into the six project demonstrators in the current project cycle 3, which is the final one in the project. These demonstrators are represented by the three driving simulators (from REL, VED and ULM partners) and by the three prototype-vehicles (from VED, ULM and CRF partners). We have also described the set-up tests between the components and the subsystem, in order to ensure proper functionality before demonstrators are ready for the evaluation (in WP6). In addition, we described the final realization and implementation of each demo (including all enablers and components now fully integrated), within the selected system architecture solutions.

2 Introduction

This document describes the integration of the SW and HW components for the TeamMate technologies from WP2/3/4 into the six project demonstrators, in the final project cycle 3, within the current system architecture (defined in D5.1 and D5.3). In addition, this deliverable illustrates also the set-up tests between the components and the subsystems, in order to ensure proper functionality before demonstrators are ready for the evaluation (in WP6). This document includes also both the description of the simulator and vehicle demonstrators.

We remind here that all demonstrators use the final common architecture. A detailed description can be found in D5.1 [1] and D5.3 [2]. For sake of comprehension, Figure 1 shows the currently used approach to integrate the automate enablers together with a given platform, i.e., vehicle or simulator:

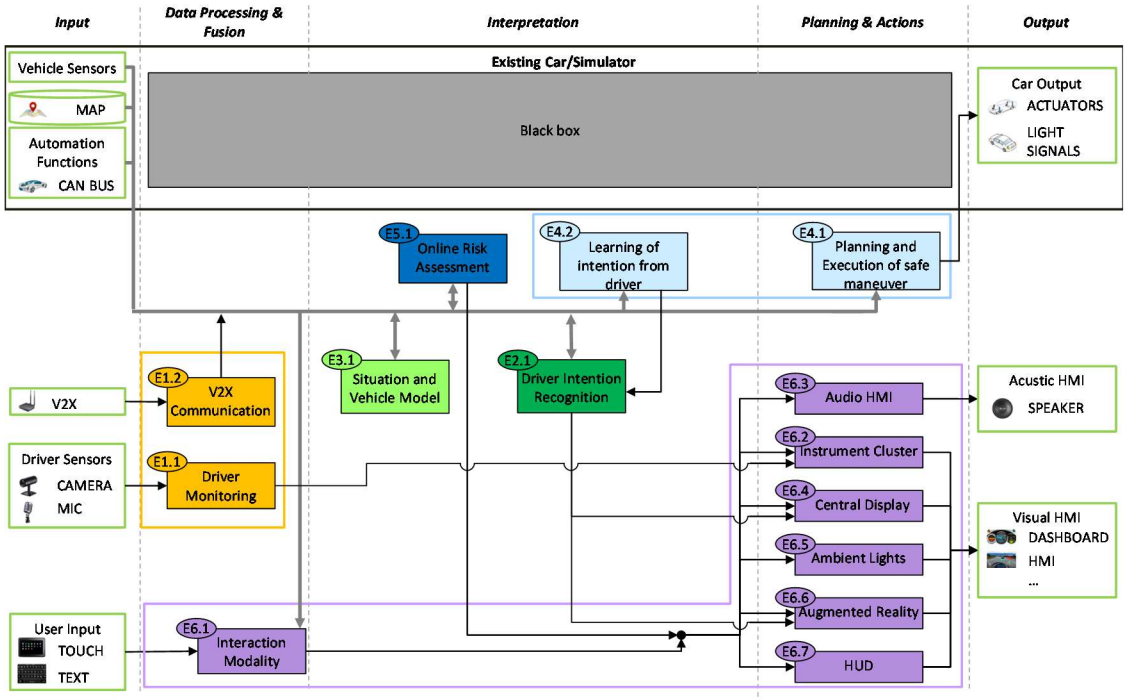

Figure 1: Current Common Architecture.

The enablers support different functional steps: "data processing & fusion", "Interpretation", and "planning & actions". The existing platform might already have modules that perform one or more of these steps. The TeamMate system does not need to know how these internal modules work or interact (they are regarded as a black box). However, the TeamMate system requires the simulator, or the vehicle, to provide interfaces for certain input and output data.

In the next sections, the demonstrators and the related implementations of the three scenarios (*Eva*, *Martha* and *Peter*[2]) are described and detailed, considering both the driving simulators (REL, VED and ULM partners) and the real-prototype vehicles (CRF, VED and ULM partners). Currently at the end of the project, all demonstrators have the same level of development, described here. In addition, in this document, we reply also to some comments provided by PO and reviewers during the mid-term review of AutoMate project. Specifically, we address the following comments:

- "ULM and VED are currently implementing two different prototypes for E4.1. This is a waste of resources and it cannot be justified by making sure that something is delivered in case integration fails. The two enablers do not seem to share the same rationale/goals".

[2] As reminder, the EVA scenario addresses extra-urban roads with roundabouts and is implemented by CRF; the scenario Martha addresses motorways and is implemented by VED; finally, the scenario Peter addresses rural roads and is implemented by ULM.

- "CRF has its own trajectory planning system; ULM and VED have their own. Specifics of integration among these different systems need to be described in the forthcoming deliverables and implemented".

In the second half of the project, ULM, VED and CRF worked on a common concept for Enabler 4.1 *Trajectory Planning*, in line with the abovementioned comments. We elaborated common principles and solutions. This concept has been adapted to the different demonstrators and scenarios. Because the requirements for the scenarios and also the demonstrators themselves are partly very different[3], we decided to produce two instantiations for this Enabler: the ULM planner and the CRF planner. The Enabler from ULM has been integrated also into the VED vehicle. These common principles and solutions are described in Section 3.

Overall, in this third cycle, we completed the integration of the enablers both into the simulators and the vehicles.

[3] Because of different requirements for urban scenarios and extra-urban scenario, as well as different requirements coming from the real-car hardware: e.g. PC versus D-Space embedded platform.

3 Trajectory Planning common Concept and Principles

Path/Trajectory planning has been a subject of study for the last decades, especially in mobile robotics. Most of the authors divide the problem into global and local planning, as the following figure shows:

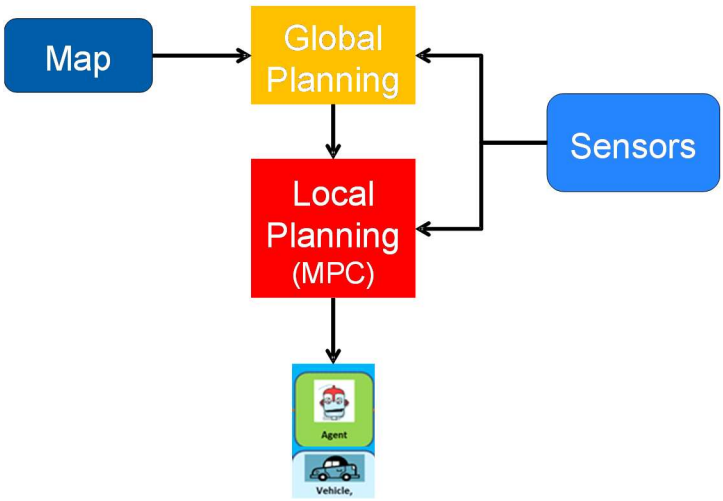

Figure 2: Sketch of the main blocks in the two-level architecture for trajectory planning.

A review of the different approaches and concept definitions (as global, local or reactive motion planning) can be found in [3–5]. In fact, a great amount of navigation techniques has been taken by this domain and then modified to face the challenges of road networks and driving rules [6–8].

In particular, trajectory planning is a basic requirement for autonomous driving and must be present in every automatically controlled vehicle. Consequently much effort has been put into this field and several techniques have been developed already (see also [9-12] for more details). In fact, many situations in daily traffic, such as following a leading vehicle or stopping behind it, require knowledge about how the scene may evolve. In recent years, much effort has

been put into developing driver models to predict traffic scenes, as realistic as possible according to human behaviour.

The decision making in contemporary autonomous driving systems is typically hierarchically structured into route planning, behavioural decision-making, local motion planning and feedback control (see also the European project ROBUST-SENSE, https://www.robustsense.eu/).

The general scheme for a Trajectory Planning module is sketched in the following figure:

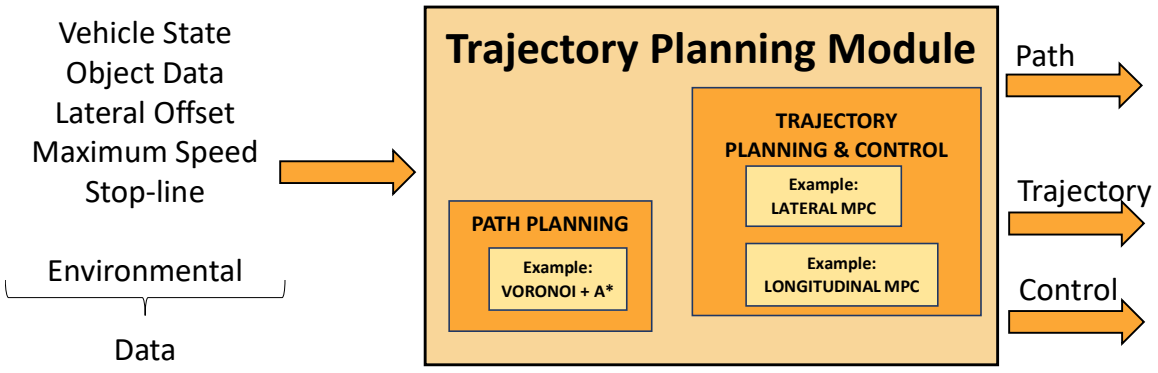

Figure 3: trajectory planning – sketch of architectural scheme, including inputs and outputs.

In the AutoMate project, we followed this conceptual scheme and particularly focused on the aspects of decision-making, motion planning and control for self-driving cars, in particular for systems falling into the SAE automation level 3 (L3) and above.

The components developed for planning, execution and learning of adaptive driving manoeuvres take as inputs the state and intentions of the driver, as well as the current situation state from the environment (as sketched in figure 3).

Based on these data, potential strategic manoeuvres are identified and planned up to a concrete action sequence on an operational level. The resulting plans include a suitable task distribution plan between driver and automation.

Given this general framework, inside the AutoMate project, we have developed two instances of the trajectory planning, in order to deal with two main aspects:

- The different scenarios (Martha, Peter and Eva) to be implemented in each demo-car (from VED, ULM and CRF partners, respectively).
- The different available HW (with related constraints) and characteristics of the sub-systems already present in every demonstrator.

An example is illustrated by the Eva scenario in the CRF vehicle: here, the trajectory algorithms run on "dSpace" platform requiring SIMULINK and MATLAB codes.

4 REL Simulator Demonstrator

REL driving simulator is based on SCANeR Cabin, a vehicle mock-up including real commands and different input modalities. All the components installed and connected to the pc communicates via UDP with the simulation software. The messages communicated with the simulator are the following:

- Acceleration
- Braking
- Steering
- Hand-braking
- Ignition
- Indicators
- Gear

These elements represent the baseline vehicle. The following enablers were integrated in the simulator in the previous project cycles:

- E1.1 – Driver Monitoring System (DMS)
- E6.2 – TeamMate multimodal HMI

The enablers integrated for the 3rd project cycle and the final instantiation of the system architecture will be described in the next section.

4.1 Enablers and system Architecture

Overall, four enablers have been integrated into the REL driving simulator throughout the project (Table 1):

ID	Enabler	Demonstrator
E1.1	Driver monitoring system with driver state model for distraction and drowsiness	REL simulator
E3.1	Driver Intention Recognition	REL simulator
E6.1	Interaction Modality	REL simulator
E6.2	TeamMate Multimodal HMI	REL simulator

Table 1: list of Enablers for REL driving simulator.

As reported in Table 1, in order to test the EVA use case in REL simulator, "E1.1 Driver Monitoring System (DMS)" and "E6.2 TeamMate multimodal HMI" have been integrated during the previous project cycles.

These were complemented in the final project cycle with the integration of enablers for "E3.1 Driver Intention Recognition" and "E6.2 Interaction Modality".

In Figure 4, the functional schema of the REL simulator adapted to Automate is presented:

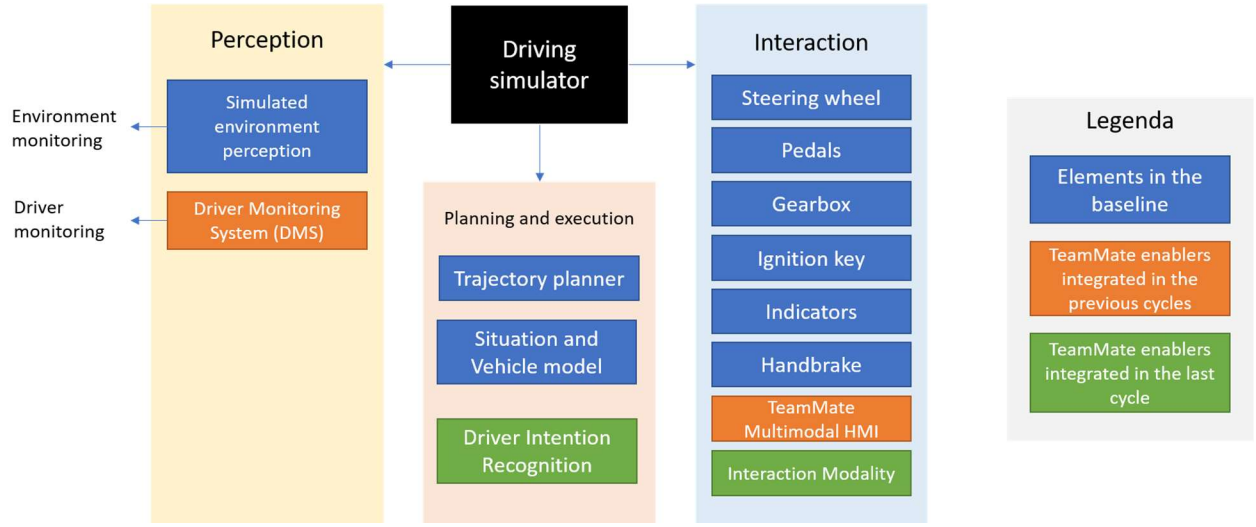

Figure 4: functional scheme of the Automate REL driving simulator.

This architecture is an instantiation of the common scheme provided by HMT and described in chapter 2, Figure 1.

4.2 Results of Set-up Tests

This section includes check list for functional tests and data communication (as described in T5.5) that show that the integration has been completed, that is, the enablers communicate with demonstrators in the intended way.

4.2.1 HMI Integration

The HMI software developed in QT (E6.2 TeamMate multimodal HMI), was installed on the simulator PC in the previous cycles. A custom communication protocol was employed to allow the sharing of information between the HMI and the driving simulator.

The communication was implemented through a client-server system; it was tested with a range of data simulated inside the server in order to verify the correct data packet flow.

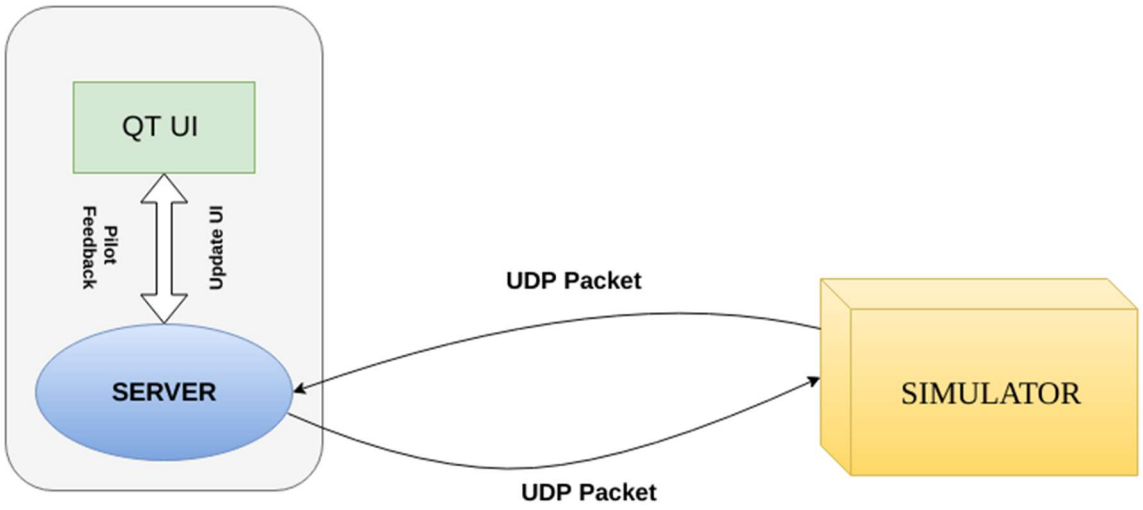

Figure 5: scheme of the communication protocol between the HMI and the simulator.

In this last cycle, an upgraded version of this enabler has been integrated. This version consisted in a new layout of the graphical component, deriving from the feedbacks collected in the third validation cycle, and new components to enrich the multimodality: in particular, the haptic seat and the HMI on the mobile application have been integrated. The mobile application simulates the behaviour of the same module installed on the real vehicle, using a slightly different architecture. While the app installed in the real car (developed in iOS) uses the vehicle data collected on the CAN network by the "Gino" device and the TeamMate SDK (see D5.7 for more details), the app integrated in the driving simulator (developed in Android) uses the software of the HMI as server for the simulator's data: the application collects data about the scenario

(e.g. the Take Over Request) and the other modules (e.g. the distraction and the Area of Interest from the DMS) from the simulator.

Figure 6 shows the different architectures used for the integration.

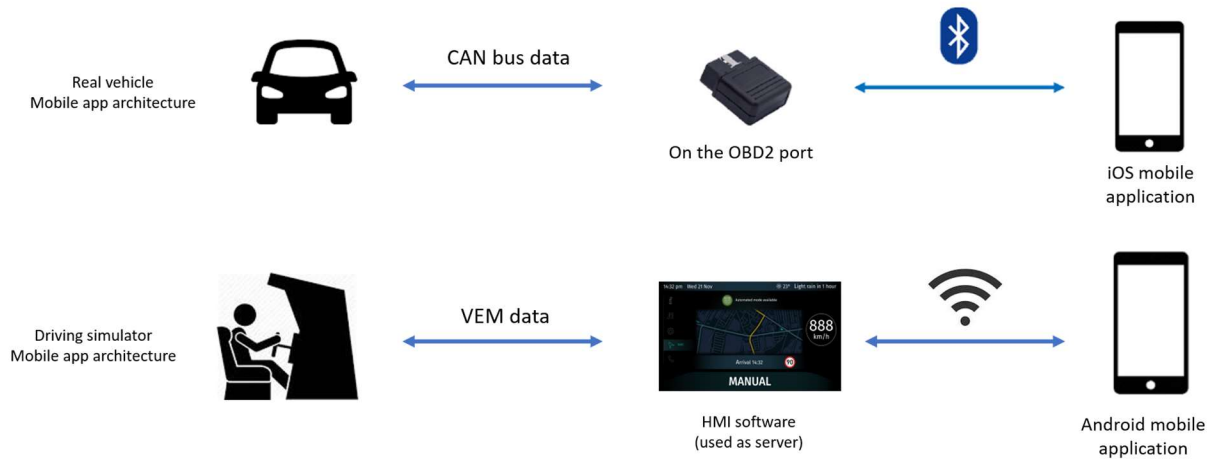

Figure 6 - Mobile app integration architecture on real car and driving simulator

As in the real car, the warning in the application is triggered when the vehicle is no longer able to keep the automated driving and the DMS detects that the driver is looking at the specific Area of Interest in which the smartphone is placed.

Moreover, in the last cycle also the haptic seat has been integrated in the driving simulator. Details on the rationale and the implementation of the haptic seat, as part of the TeamMate multimodal HMI, are reported in deliverable D4.6.

The haptic seat is composed by a sensorized seat cover and a Phidgets control board. The seat cover is connected to the power socket, while the board is connected to the seat cover and to the simulator controller PC via USB cable (both for data collection and supply). Figure 7 describes the architecture of the haptic seat integration.

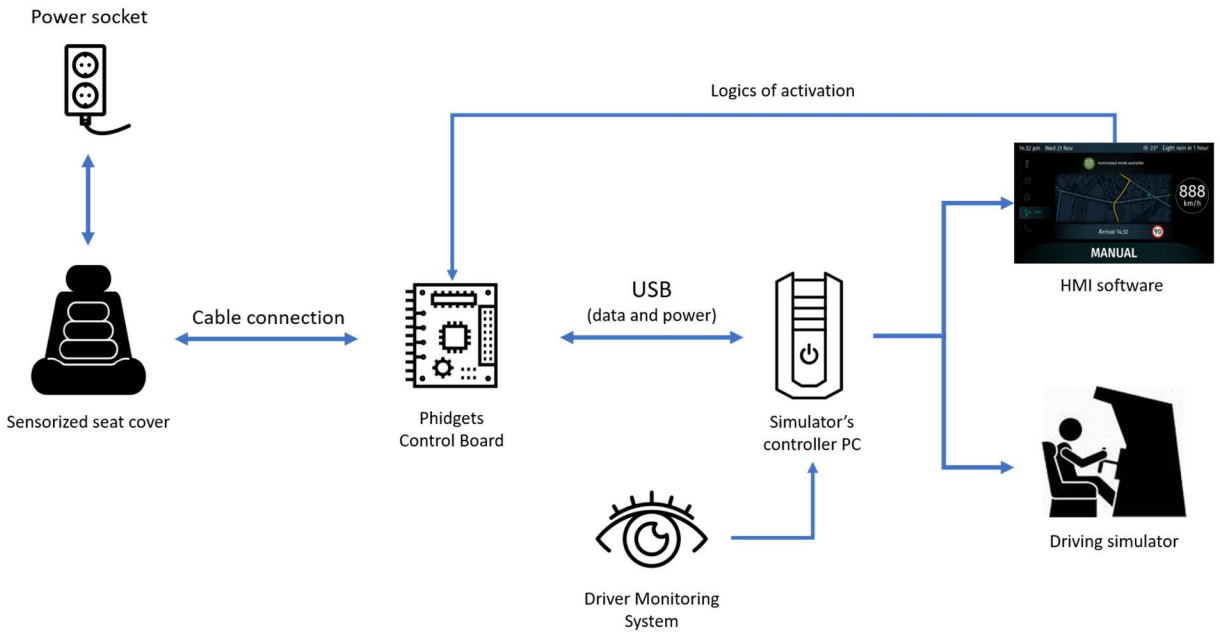

Figure 7: Haptic seat integration architecture.

Several functional tests have been performed to check the successful integration of these modules. The table describes the procedure and the set-up tests for the HMI integration:

ID	Step and test	Accomplished
1	Check the UPD communication with the new version of the HMI	Yes
2	Check wi-fi communication between the HMI and the mobile app	Yes
3	Check the activation of the conditions for the warning activation in the mobile app	Yes
4	Check the activation of the warning through the scenario, for the haptic seat	Yes
5	Check and calibrate the intensity and the direction of the vibration in the haptic seat	Yes

Table 2: Set up tests for HMI integration.

4.2.2 Driver Intention Recognition Integration

The Driver Intention Recognition (DIR) module, developed by HMT, has been developed and integrated in order to allow a human-like driving in the roundabout. While in baseline, in fact, the automation behavior has been kept without any modifications, in the TeamMate modality the intention has been used to trigger the entering in the roundabout.

The DIR module, based on Bayesian Neural Networks, has been trained through manual driving data in roundabouts, collected in REL and OFF driving simulators. The road and traffic characteristics have been taken into account in order to create a realistic representation of the intention.

The model has been integrated in REL driving simulator through a specific communication module part of TeamMate component SDK developed by HMT (for more details see D5.7).

The communication is based on a UDP socket and a custom payload, which is a very simple customized struct - called Payload - that encapsulates all the needed info.

The figure below shows a Payload structure example:

```
1. Struct Payload {
2. Date/time timestamp;
3. int64 vehicle Id;
4. int64 Road Id;
5. int64 Lane Id;
6. double Road abscissa;
7. double CoG position/X;
8. double CoG position/Y;
9. double CoG position/Yaw;
10.  double Tangential speed;
11.  double Gas pedal;
```

```
12.  double Brake pedal force;
13.  double Road angle;
14.  }
```

Figure 8: example of a Payload structure for DIR integration in REL driving simulator.

Figure 9 shows the schematic representation of the DIR integration in REL simulator.

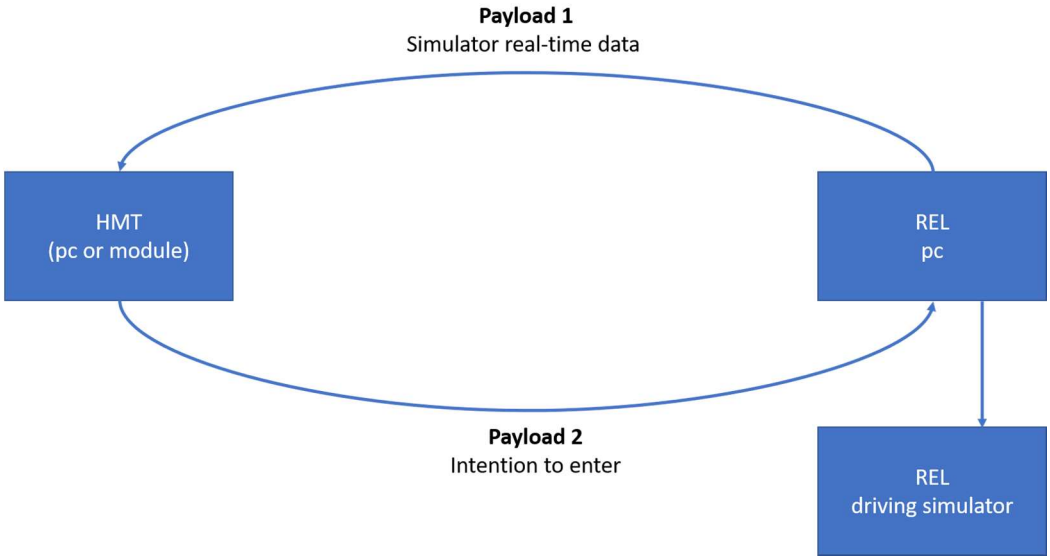

Figure 9: Driver Intention Recognition integration architecture in REL driving simulator

ID	Step and test	Accomplished
1	Check the data collected to train the model, including labelling and synchronization	Yes
2	Check the communication of the module with Wireshark	Yes
3	Check the length of the payload received (120) and sent (12) by the module	Yes

4	Refine the module "AutoMate_Bridge" used to send data from the simulator on a UDP port, including the order the datastructure in the payload	Yes
5	Check the value of the "intention" in the scenario	Yes

Table 3: Set up tests for DIR integration

4.3 Demonstrator physical Integration

Besides the baseline, the TeamMate demonstrator includes a 15.6" screen located behind the steering wheel, i.e. in the typical instrument cluster position. The screen is connected to the simulator PC through a DPort cable. The software with the HMI is installed in the simulator PC; the information of the simulation software is sent to the HMI software and displayed; the simulator script manages the HMI modes (i.e. the "states" of the state machine).

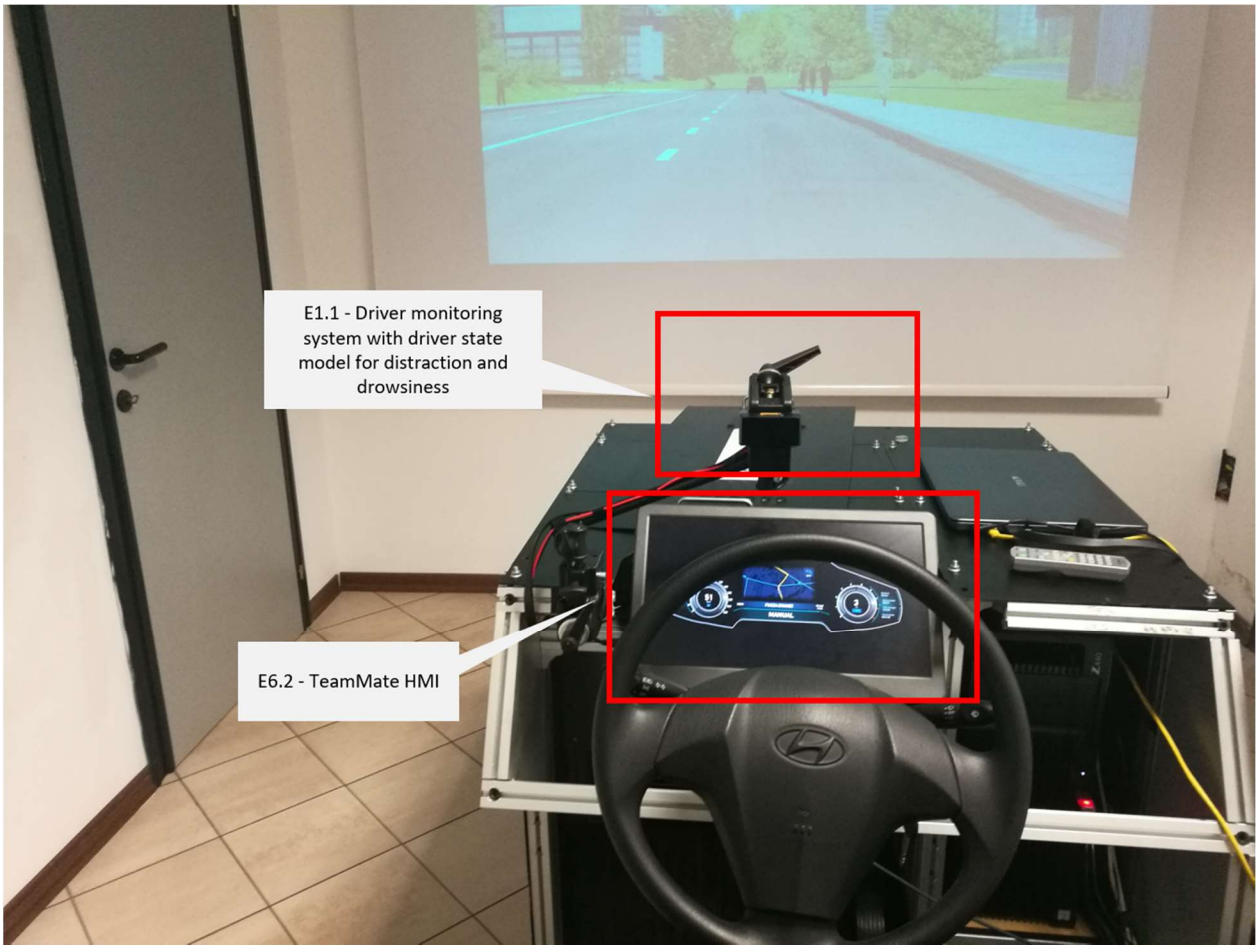

Figure 10 – the AutoMate REL driving simulator

The DMS is installed behind the instrument cluster in order to avoid the occlusion of the HMI. Since the scenario of the simulation is shown to the driver through a projector, the camera doesn't obstruct the view of the driver. The camera has been mounted with a mechanical support; it is handled by its own PC, which is connected through an "Ethernet/UDP" link to the simulator local network. The haptic seat cover has been applied on the driver seat; the smartphone is placed with a special support in a position corresponding to the central tunnel.

Finally, the Driver Intention Recognition (DIR) has been installed with a local host network on the simulator pc controller.

5 VED Simulator Demonstrator

In this section, the VED driving simulator demonstrator is described, for the 3rd project cycle.

5.1 Enablers and system Architecture

Three TeamMate enablers have been integrated into the VED simulator, namely, E1.1 Driver monitoring system (DMS), E6.1 Interaction Modality and E6.2 & E6.3 TeamMate HMI & Central stack display. The functionalities of the remaining enablers were either simulated or were already available in the simulator.

ID	Enabler	Demonstrator
E1.1	Driver monitoring system with driver state model for distraction and drowsiness	DMS : Yes, • Drowsiness : Yes • Distraction : Yes This module has been successfully installed and tested by CAF in VED simulator.
E1.2	V2X communication	Simulated

E3.1	Situation and vehicle model	From simulator (idealistic perception and ego-pose/state)
E4.1	Planning and execution of safe manoeuvre	From simulator (Scanner autonomous driving module)
E6.1	Interaction modality	Different interaction modalities implemented and tested
E6.2 - 3 -4	TeamMate HMI (Cluster + audio, Central stack display)	Cluster : Yes, provided by REL, installed by VED. Audio : messages recorded in French provided by VDC

Table 4: list of Enablers for VED driving simulator.

5.2 Results of Set-up Tests

This section includes functional tests and data communication that show that the integration has been completed (i.e. enablers communicate with demonstrators).

5.2.1 Driver Monitoring System

The enabler E1.1 software provided by CAF has been installed with a SmartEye eye tracker system. It takes the outputs of the SmartEye system and produces the driver state as an output.

As the VED driving simulator set-up was upgraded after the 2nd evaluation cycle, an updated version of the 3D world was generated. Based on this 3D world, two versions of the world coordinate system were tested, and the one providing the most accurate detection of the distraction was selected.

The UDP protocol and the local network connecting the DMS app and the simulator created for the 2nd cycle evaluation were used for the 3rd cycle.

The calibration and testing of the enabler E1.1 on VED simulator followed the same schema of integration as the REL-simulator since REL and VED has the same driving simulation software and a smartEye system for eye tracking.

ID	Step and test	Accomplished
1	Place the camera in the most suitable position	Yes
2	Create the driving simulator's 3D world	Yes
3	Calibrate the Camera	Yes
4	Integrate the simulator world in the world coordinate system	Yes
5	Connect with a local network the DMS app and the simulator PC	Yes
6	Create and validate the UDP protocol for data communication	Yes

7	Integrate the classified data into the simulation software	Yes

Table 5: Set-up tests for DMS integration in VED simulator.

5.2.2 Human-Machine Interaction

The definition of the different HMI states that were needed for each use-case was made. An executable of the HMI has been coded by REL in order to test if the HMI included all needed states.

All the displayed messages were translated in French by VED and integrated into the code of the HMI provided by REL. For the vocal messages, VED decided to remove them as they were in English and needed to be translated in French in order to be included in the HMI. The needed vocal messages were recorded by VED and included in the scenarios developed for the 3rd cycle evaluation.

Many tests of the exchange of information between the HMI compiled code and the driving simulator were made by running different scenarios in the driving simulator and the check of the update of the state of the HMI and the variables displayed in the HMI.

5.3 Demonstrator

VED simulator platform is a static driving simulator with 2 seats (one driver seat and one front passenger seat). It allows to display the driving environment on 3 panels of 140 *240 cm offering a total visual field of 230°. Rear view was displayed on three separate screens. Lateral mirrors are 2 screens of 8*5 cm and central mirror is a 10*7 cm screen. A 10'' screen set behind the steering wheel is used as a dashboard. This baseline platform is also equipped with:

- 3 pedals (with a possibility to only use two pedals)
- Gearbox 7 or Automatic Option
- Peugeot Steering Wheel & Commodo Functions
- Force feedback system
- Sound system with 5 speakers & subwoofer
- 4 infrared Cameras
- An iPad mini mounted on the central console

This simulator runs on Oktal SCANeR studio 1.8 driving simulation software.

Figure 11: VED Simulator.

6 ULM Simulator Demonstrator

In this section, the ULM simulator demonstrator is described.

6.1 Enablers and system Architecture

List of enablers that have been integrated into the demonstrator:

ID	Enabler	Demonstrator
E1.1	Driver monitoring system with driver state model for distraction and drowsiness	DMS : Yes, • Drowsiness : Yes • Distraction : Yes This module has been successfully installed by CAF and tested in the ULM simulator.
E2.1	Driver intention recognition	Yes
E3.1	Situation and vehicle model	Yes
E3.2	Driving task model - DriveGOMS	Yes, first experiment was conducted to analyze the PETER scenario

E4.1	Planning and execution of safe manoeuvre	From simulator
E4.2	Learning of intention from the driver	Yes
E5.1	Online risk assessment	Yes
E6.1	Interaction modality	Yes
E6.2	HMI	Yes
E6.3	Augmented reality	Yes

Table 6: list of Enablers for ULM driving simulator.

6.2 Results of Set-up Tests

The implemented Enablers were tested separately and will be integrated and tested all together in the next cycle.

6.2.1 Driver Monitoring System

The Driver Monitoring System (DMS) application (E1.1) was installed on the dedicated eye-tracking computer in the ULM driving simulator. The SmartEye application provides the needed eye-tracking data for the application. The individual ULM world coordinate system of the different car components was converted and implemented into the application. The DMS application provides the driver state as an output through the local network, therefore it can be used in the simulation software in real-time. The validation of the functionality

of the application was similar to the one conducted in the REL and VED simulators:

ID	Step and test	Accomplished
1	Place the camera in the most suitable position	Yes
2	Create the driving simulator's 3D world	Yes
3	Calibrate the Camera	Yes
4	Integrate the simulator world in the world coordinate system	Yes
5	Connect with a local network the DMS app and the simulator PC	Yes
6	Create and validate the UDP protocol for data communication	Yes
7	Integrate the classified data into the simulation software	Yes

Table 7: Set-up tests for DMS integration in ULM simulator.

6.2.2 Human-Machine Interaction

The Cluster HMI (E6.2) was integrated into the driving simulator. Different modes are triggered and updated in real-time according to the situation automatically.

6.2.3 Augmented reality Display

The augmented reality display was implemented as a software package for the SILAB simulator software. This addition draws AR elements into the simulated environment on the front screen. A first experiment was conducted where different versions of the AR display were tested against each other. The tests

were successful and the AR displays were shown correctly. The AR display changed itself automatically according to the traffic situation (E3.1) with the input of the driver intention (E2.1) and the online risk assessment (E5.1).

6.2.4 Communication test between enablers

After the integrations of the enablers (see Enablers and system Architecture) the communication (via a network interface) and the correct technical functionality was tested in exemplary scenarios with experts. All systems worked together properly and were evaluated in a final evaluation of the Peter scenario (results see D6.3).

6.3 Demonstrator physical Integration

ULM driving simulator is equipped with the SILAB driving simulation engine. The simulator is a mock-up that represents a real car (as shown in Figure 9) with a driver and a passenger seat. Additionally, there are several features in the driving simulator (see D1.3):

- steering wheel (force-feedback)
- pedals
- indicators
- central touch panel
- displayed rear mirrors (central, left, right)
- Smart-eye camera (static eye tracking system)

Figure 12: ULM driving simulator inside projection screen.

The driving simulator also includes three high definition beamers that project the simulated environment onto a projection screen in front of the driver to create an immersive driving environment (see again D1.3).

7 CRF Vehicle Demonstrator

In this section, the CRF vehicle demonstrator is described, after the project 3rd cycle.

7.1 Enablers and system Architecture

The table below shows the list of enablers, required to implement the Eva scenario and thus integrated into the CRF demonstrator:

ID	Enabler
E1.1	Driver monitoring system with driver state model for distraction and drowsiness
E3.1	Situation and vehicle model
E4.1	Planning and execution of safe maneuver
E6.2 - E6.3 - E6.4	TeamMate HMI (Cluster + audio, Central stack display, HUD)

Table 8: list of final Enablers for CRF demonstrator car.

With reference to the general architectural schema presented in Section 1, the following figure shows which components have been integrated in the third project cycle for the CRF vehicle:

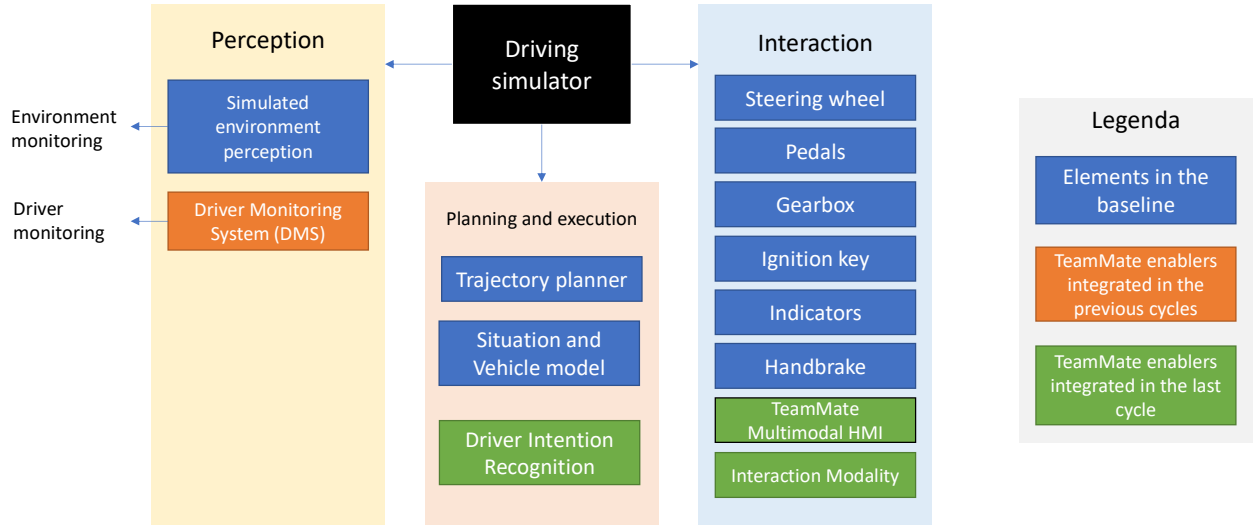

Figure 13: schema of the main components for the CRF vehicle, on which the Eva scenario is implemented. NP-ADAS means "Normal Production ADAS", that is the ADAS applications available for all customers when this vehicle model is purchased.

In this schema, we have highlighted the three main parts that is "Perception", "Planning & execution and Interaction". In addition, we have illustrated also which enablers have been integrated in the third cycle (the HMI and the parts developed by CRF), which ones have been updated (the DMS interfaces, to solve and optimise some communication problems) and which enablers were already present for the baseline configuration (see also deliverable D5.2 [15]).

7.2 Communication between Enablers

This section describes how the communication between the enablers of the CRF vehicle is implemented.

The general scheme is reported in the following figure:

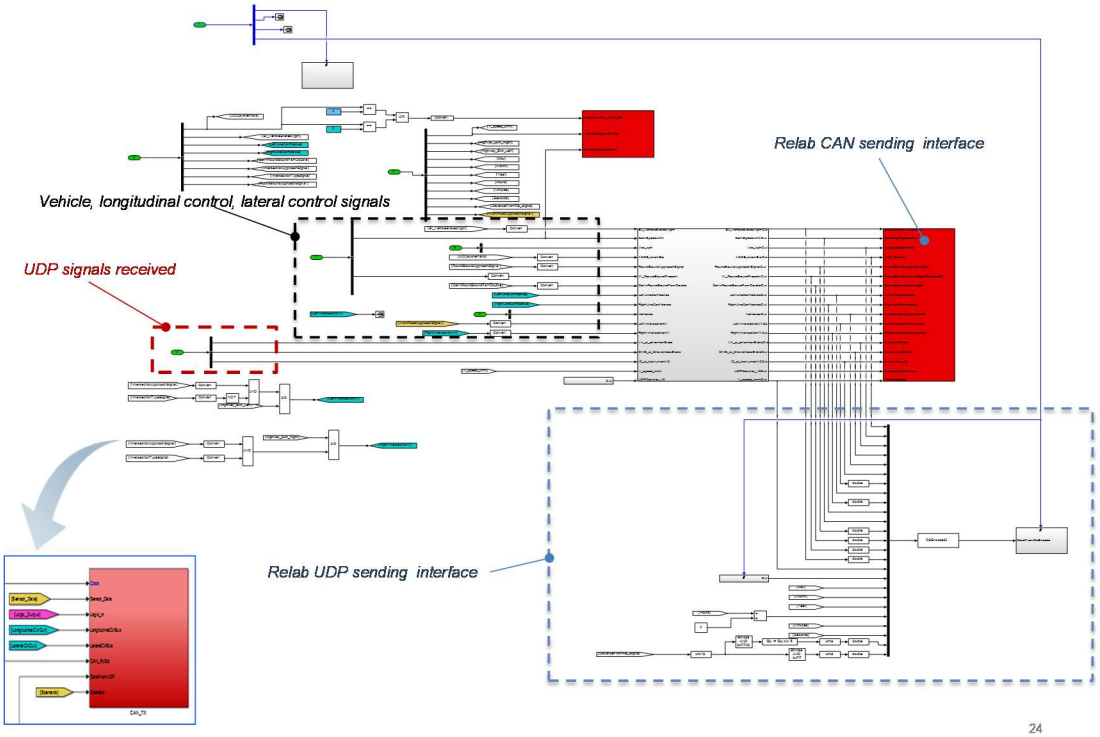

Figure 14: CAN and UDP interfaces for enablers in CRF vehicle.

In details, the following figure shows the list of the signals exchanged with the REL modules for the related HMI:

UDP_RelabData - DSPACE SEND TO HMI					
Signal Name	**Unit**	**Meaning**	**Type**	**N. Byte**	**UDP Streaming Byte Index**
ACC_Status	Enum	ACC status. 0: Idle; 1:Cruise; 2: Track; 3:Stop; 4:Override	Uint8	1	0
AccSpeedSetPoint	km/h	Acc speed setpoint	Uint8	1	1
AttentionState	Enum	Driver Attention level. 0: Unvailable; 1: Attentive; 2:Mid attention;3:Low attention; 4:Distracted	Uint8	1	2
DrowsnessState	Enum	Drowsnes State. 0:Unvailable; 1:Alert; 2:Slighty Drowsy, 3:Drowsy, 4:Sleepy	Uint8	1	3
HandOnSteeringWheelSt	Boolean	Status Hands on Steering wheels. 0: Not present;1: Present	Uint8	1	4
Instrument_ID	Enum	ID internal vehicle instument. 0=UNAVAILABLE; 1 = UNKNOWN; 2 = WSL; 3 = WSR; 4 = LR; 5 = RR; 6 = CR; 7 = CD; 8 = IC	Uint8	1	5
LeftIntersectionWarning	Boolean	Left intersection warning. 1: Present	Uint8	1	6
LeftLineConfidence	Boolean	Left line confidence level. 0: Not good; 1: Good	Uint8	1	7
RightIntersectionWarning	Boolean	Right intersection warning. 1: Present	Uint8	1	8
RightLineConfidence	Boolean	Right line confidence level. 0: Not good; 1: Good	Uint8	1	9
RoadSignSpeedLimit	km/h	Road Sign Speed limit warning	Uint8	1	10
RoundBoundApproachingSt	Boolean	Closer Round Bound approaching warning detected.0: Not present; 1: Round Bound approaching	Uint8	1	11
RoundBoundCrossingSt	Boolean	Round Bound in crossing status. 0: Not present; 1: Round Bound in crossing	Uint8	1	12
RoundBoundWarningSignReachedSt	Boolean	Round Bound warning signal reached by the car. 0: Not present; 1: Round Bound reached	Uint8	1	13
StatusFSM_Automate	Enum	State of Automate Finite State Machine.0: Manual Mode; 1: Automatic Mode; 2: Control Sharing Mode; 3: Safety Stop Mode (V	Uint8	1	14
VehicleSpeed	km/h	Vehicle speed	Uint8	1	15
DspaceUdpCounter	Enum	Free running counter : value from 0 to 15	Uint8	1	16
Day	Enum	Day from gps	Uint8	1	17
Month	Enum	Month from gps	Uint8	1	18
Year	Enum	Year from gps	Uint8	1	19
Hours	Enum	Hours from gps	Uint8	1	20
Minutes	Enum	Minutes from gps	Uint8	1	21
Seconds	Enum	Seconds from gps	Uint8	1	22
DistanceFromRB_HighByte	Enum	Distance from Round Bound High Byte	Uint8	1	23
DistanceFromRB_LowByte	Enum	Distance from Round Bound Low Byte	Uint8	1	24

Table 9: UDP interfaces signals mapping for HMI modules and components.

As sketched by above figure and table, we use two main communication types: UDP and CAN. The first is used by the DMS node to provide its outputs to the "dSpace" ECU, which represent the core of the TeamMate car system. Then these messages are sent to the multimodal HMI, using two channels:

- The UDP protocol for the main part, implemented in the on-board tablet, which contains all the logics and strategies.
- The CAN protocol for the secondary part, implemented as an APP on the smartphone, to provide info to the driver when s/he is distracted by using the phone.

Actually, there is also a third protocol, via USB, which is used by REL partner, to control the vibrating seat (vibration has been investigated to take the distracted driver back into the control loop, when necessary).

Next sections describe both the HW and SW architectures as well as the actual implementations on the vehicle.

7.3 Demonstrator physical Integration

The CRF demonstrator is based on a Jeep Renegade 1.4 MultiAir 140HP DDCT, as illustrated in the figure:

Figure 15: CRF demonstrator vehicle, during the final event of the project, held together with the IV2019 Symposium, in June, in Satory (FR).

In the following figures, the location of the vehicle sensors is sketched:

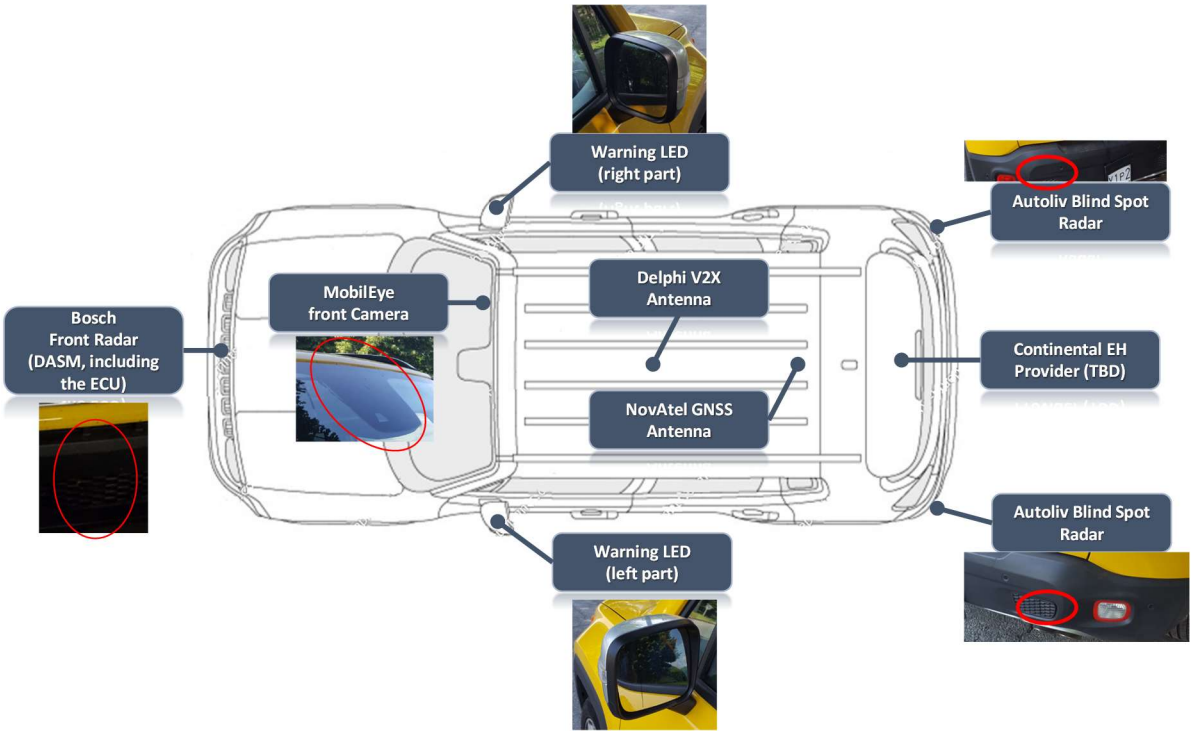

Figure 16: location of sensors in the final version (after 3rd cycle) of CRF prototype vehicle.

The frontal part is covered by two sensors: a medium range radar and a mono-camera, which is able to provide several features, such as object detection and classification, lane recognition, traffic-sign detection (speed limits, roundabout, etc.) and so on. The "eHorizon" and "GPS system" are used for the vehicle localization on the road (respect to rounbout). The "Blind Spot Radars" (for rear obstacle detection) and the related "Warning LEDs" (for HMI to the driver) are used for the ADAS application named *Blind Spot* function (during the baseline, see deliverable D6.3 for more details).

This sensorial platform is used to feed the enablers E3.1, E4.1 and E5.1.

The deployment diagram, for SW modules in specific HW components, is the same as the one presented in D5.3; here, the Automate vehicle configuration is presented:

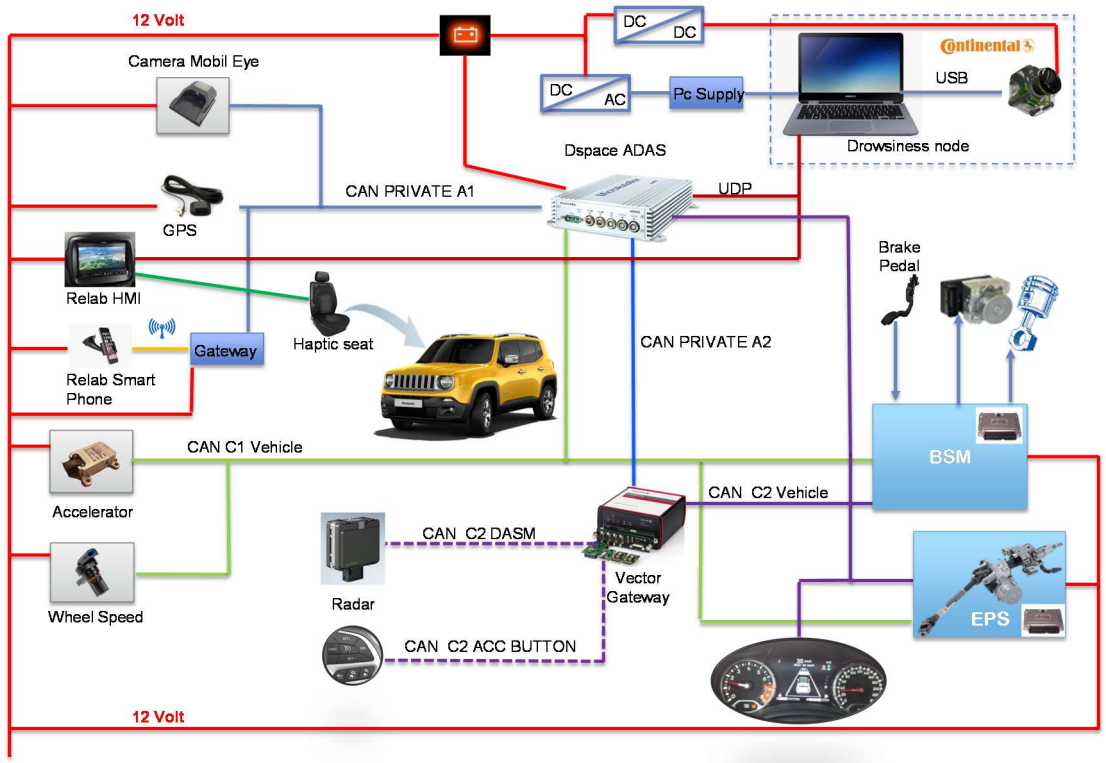

Figure 17: Automate HW configuration in CRF prototype-vehicle.

The most relevant enablers (namely, the corresponding SW modules) are implemented in the *dSpace MicroAutoBox*, which is the system central unit. The PC is used for the image processing and the monitoring of driver's status. In particular, the enablers have been installed in the car as follows:

- The enablers E3.1, E4.1 and E5.1 are embedded in the *MicroAutoBox* Central Unit.
- The enabler E1.1 is embedded on a dedicated hardware (PC connected with the IR camera from CAF partner).

- All the enablers related to HMI are installed on a dedicated tablet, which is also a gateway to convert data from UDP to CAN protocols.

The final SW architecture is described as follows:

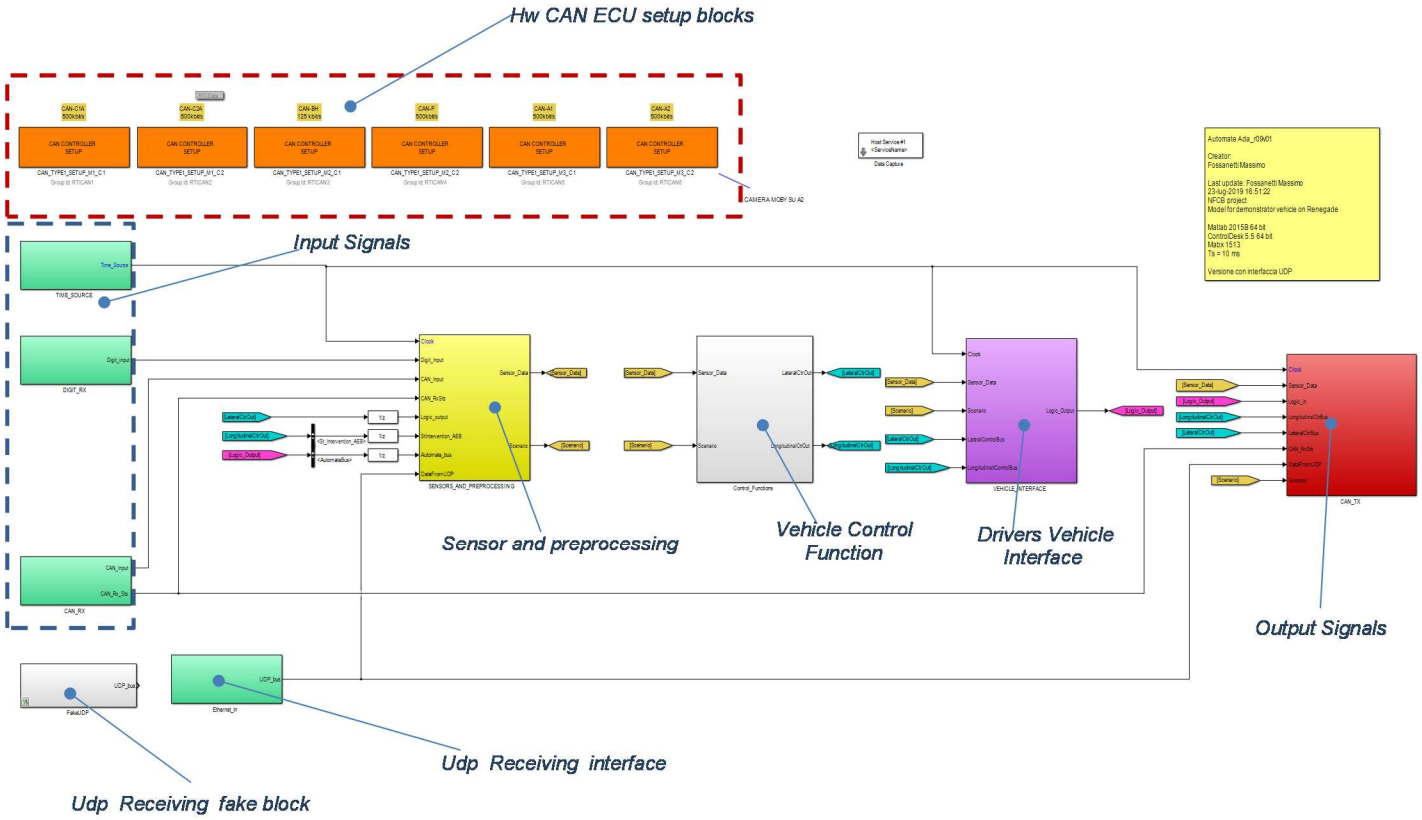

Figure 18: AutoMate general SW architecture.

As the figure shows, there are three main blocks. The first one is the "*Sensor and Pre-processing*", which receives inputs from the sensors. The perception platform was already present before AutoMate and not developed for this project, both for the baseline and TeamMate configurations (only the necessary adaptations have been done). The second block is represented by the "*Vehicle Control Function*", where the functions to control both the lateral and longitudinal aspects of the vehicle are implemented. Then, there is third

block, "*Driver Vehicle Interface*", which commands the basic HMI of the vehicle and provide the necessary outputs to the multimodal HMI of REL partner. All these modules are implemented using MATLAB/SIMULINK in the dSpace ECU.

7.4 Implementation of the TeamMate Car on CRF Demonstrator

The TeamMate car (TM car), as implemented in CRF vehicle, is able to provide full automated control and shared control, depending on the external and internal conditions (state of human and state of machine). Where possible, the CRF TM car is capable to provide highly autonomous functionalities:

- Lane-keeping (driving in the middle of the right lane).
- Keep target speed (defined by the speed limits on that particular road segment or by the user's will).
- Keep safety distance from obstacles; the vehicle can also stop until the end and then re-start when the obstacle ahead does like that (such as – for example – at the traffic light).

The vehicle cannot perform lateral control (lane-keeping) if the lines of the lane are not present. This is why in Eva scenario (?) a shared control is requested during the approach of a roundabout (in Turin and its surroundings, roundabouts do not contain lanes).

The following paragraphs detail the implementation solutions for this TM car.

7.4.1. Pre-processing Phase

In the pre-processing phase, two aspects have been considered: road path and object positioning; traffic sign recognition. The following two figures show the related block diagrams:

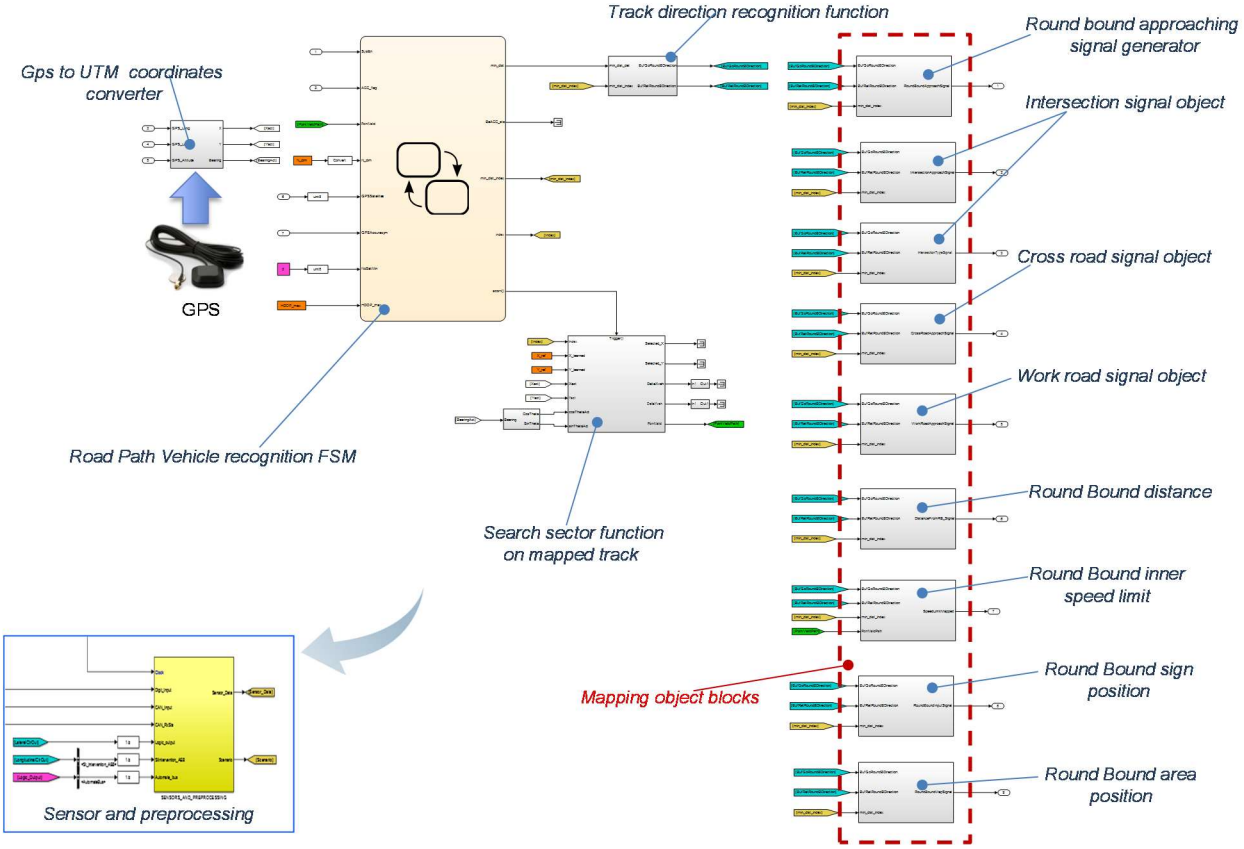

Figure 19: pre-processing phase for road path and object positioning.

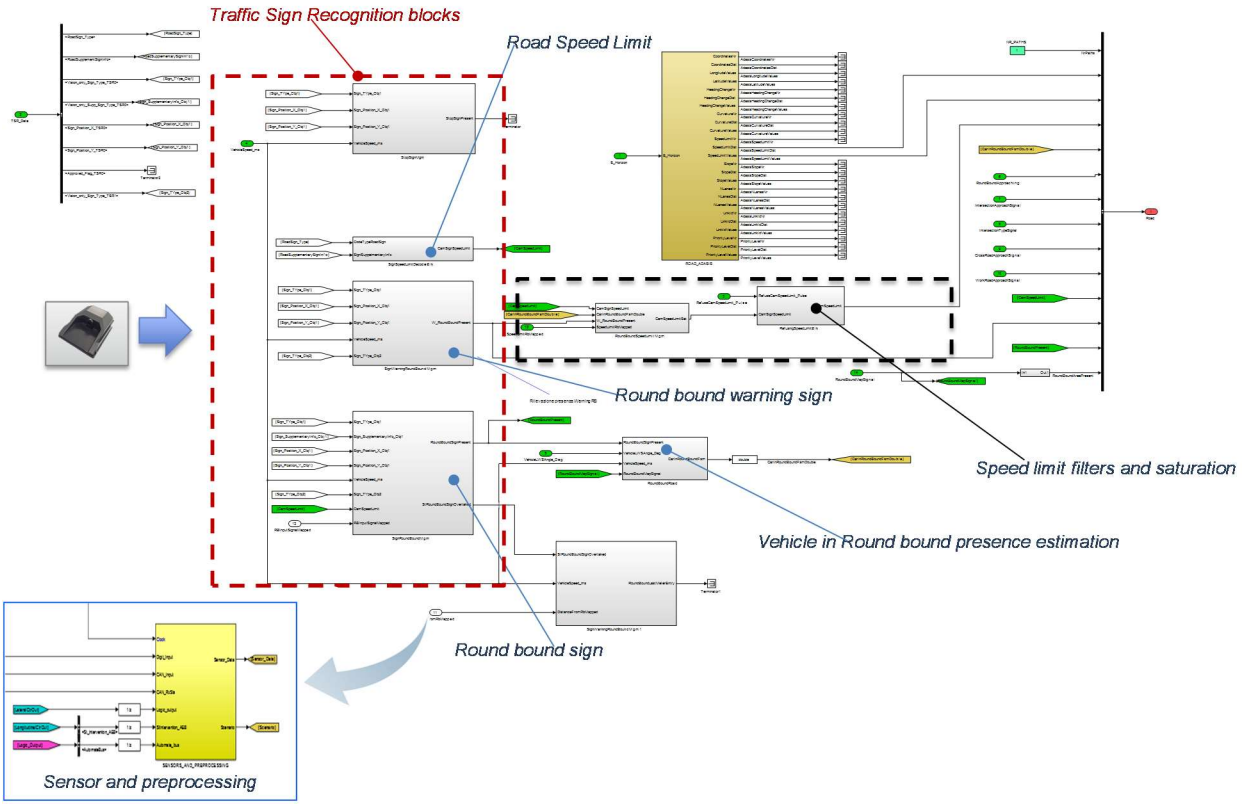

Figure 20: pre-processing phase for traffic sign recognition from camera.

These two diagrams illustrate the adaptation activities from the sensorial inputs performed by CRF. The first one, Figure 19, is about the road path selection and the object positioning on it. This is used to identify the path for the vehicle travelling and select the appropriate dynamic objects (that is, the one of interest, e.g. on the TM car trajectory). The second diagram, Figure 20, is related to the location and positioning of the traffic sign to be used by the system. In particular, we consider two types: speed limits and approach to /presence of a roundabout.

7.4.2. Vehicle Control Functions

This is the core of the TM Car implementation. It is divided in two parts: longitudinal and lateral controls.

7.4.2.1 Longitudinal Control

The SW architectural scheme is sketched below:

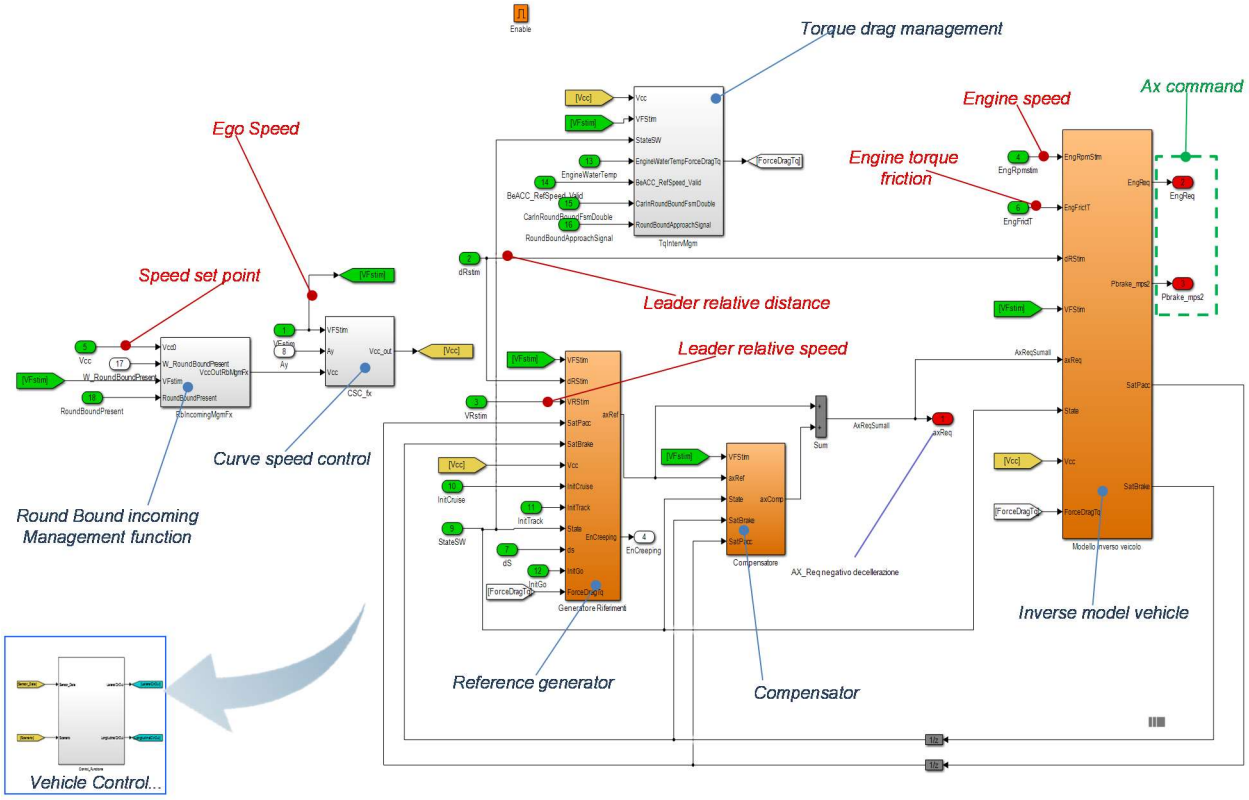

Figure 21: longitudinal control SW architecture.

The longitudinal control is implemented as an **ACC** with **Stop & Go** function. The logical scheme is reported in the following figure:

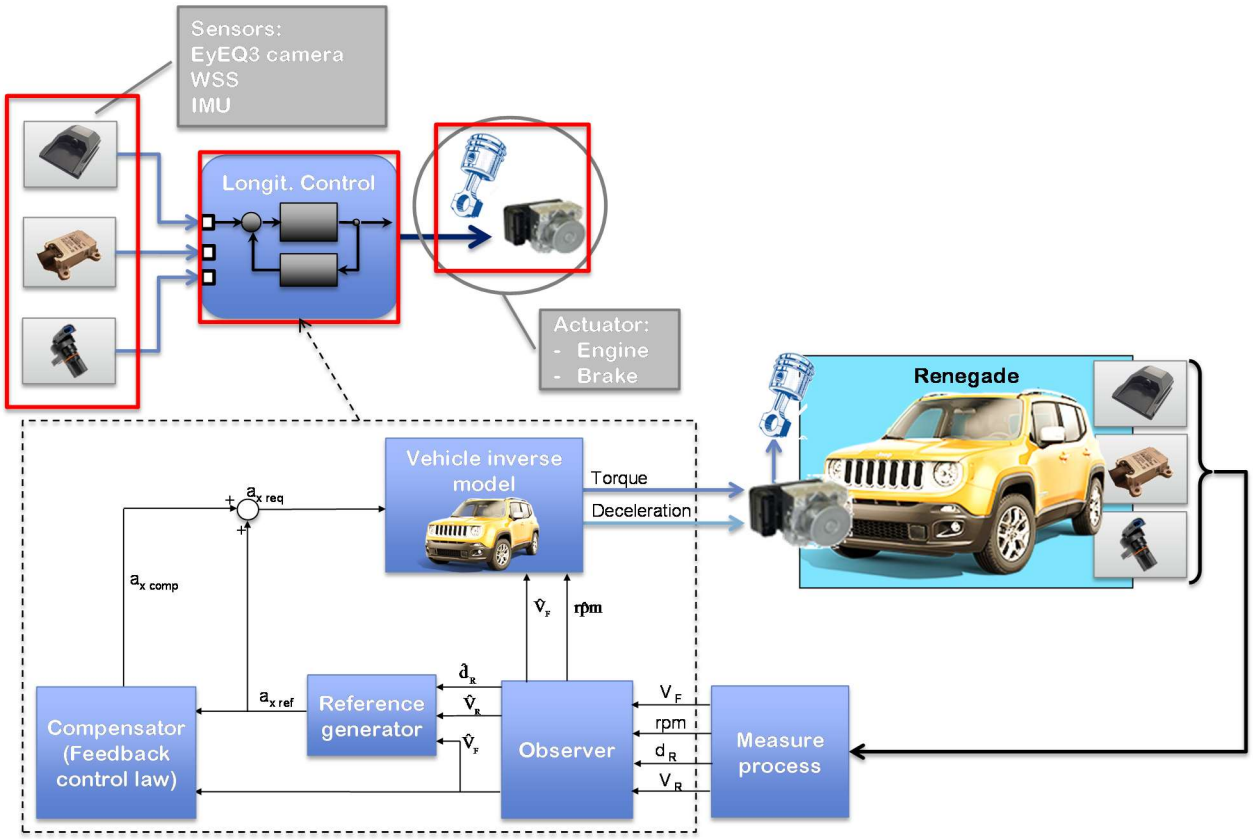

Figure 22: Ax control logical scheme.

This scheme describes the flow of data and info for the different blocks, as detailed below. The aim of longitudinal control is to follow longitudinal speed reference defined by the driver or by road camera information available (e.g. speed limits). In particular, two references are generated: one for the speed control and one for the distance control.

There are three main components: the "**compensator**", the "**observer**" and the "**vehicle inverse model**".

The goal of the first one is to compensate the overall disturbs/interferences:

- Road slope variation
- Wind presence

- Inverse vehicle model inaccuracy

We used a first order compensator:

$$a_{x\,comp} = K_{comp} \cdot \int (a_{x\,ref} - a_F)\, dt \quad \text{Eq. (1)}$$

Where K_{Comp} is defined according to the stability control requirements and is related to the vehicle longitudinal dynamics.

For the "observer", we used a first order low pass filter on vehicle speed signal (VF) and engine speed signal (RPM).

The last part of longitudinal control is the "vehicle inverse model", whose block is designed to manage system actuators transforming the requested acceleration into an engine torque or a brake deceleration demand.

The algorithm is based on the calculation of the requested torque applied to the wheels:

$$m_{at} = m + \frac{J_t + \tau_P^2 \cdot \tau_C^2 \cdot J_m}{R^2}$$

$$T_{req} = \frac{R}{\eta_t \cdot \tau_P \cdot \tau_C} \cdot \left(m_{at} \cdot a_{x\,req} + F_{res} \right)$$

Eq. (2)

Where:

F_{res}	Rolling and drag resistance	m	vehicle weight
R	wheel radius	τ_P, τ_C	automatic transmission gear ratios
η_t	transmission efficiency	J_m, J_t	engine and transmission inertia

Next section describes the lateral control.

7.4.2.2 Lateral Control

The SW scheme for lateral control is sketched in the following figure:

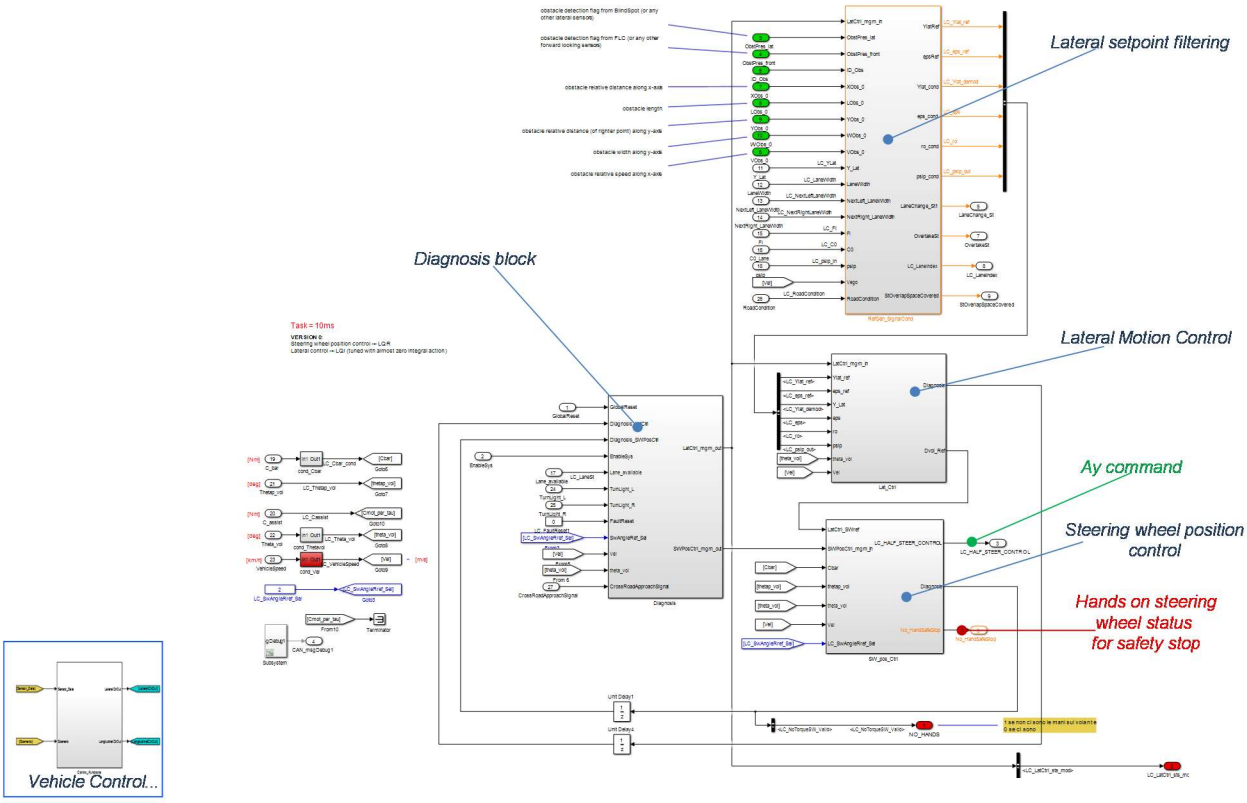

Figure 23: Ay control SW architecture scheme.

The lateral control is implemented, essentially, as a **lane centering**. Its aim is to minimize lateral deviation (Y_{lat}) and heading angle (ϵ) errors, ensuring in the mean time, vehicle stability. The logic architecture is as follows:

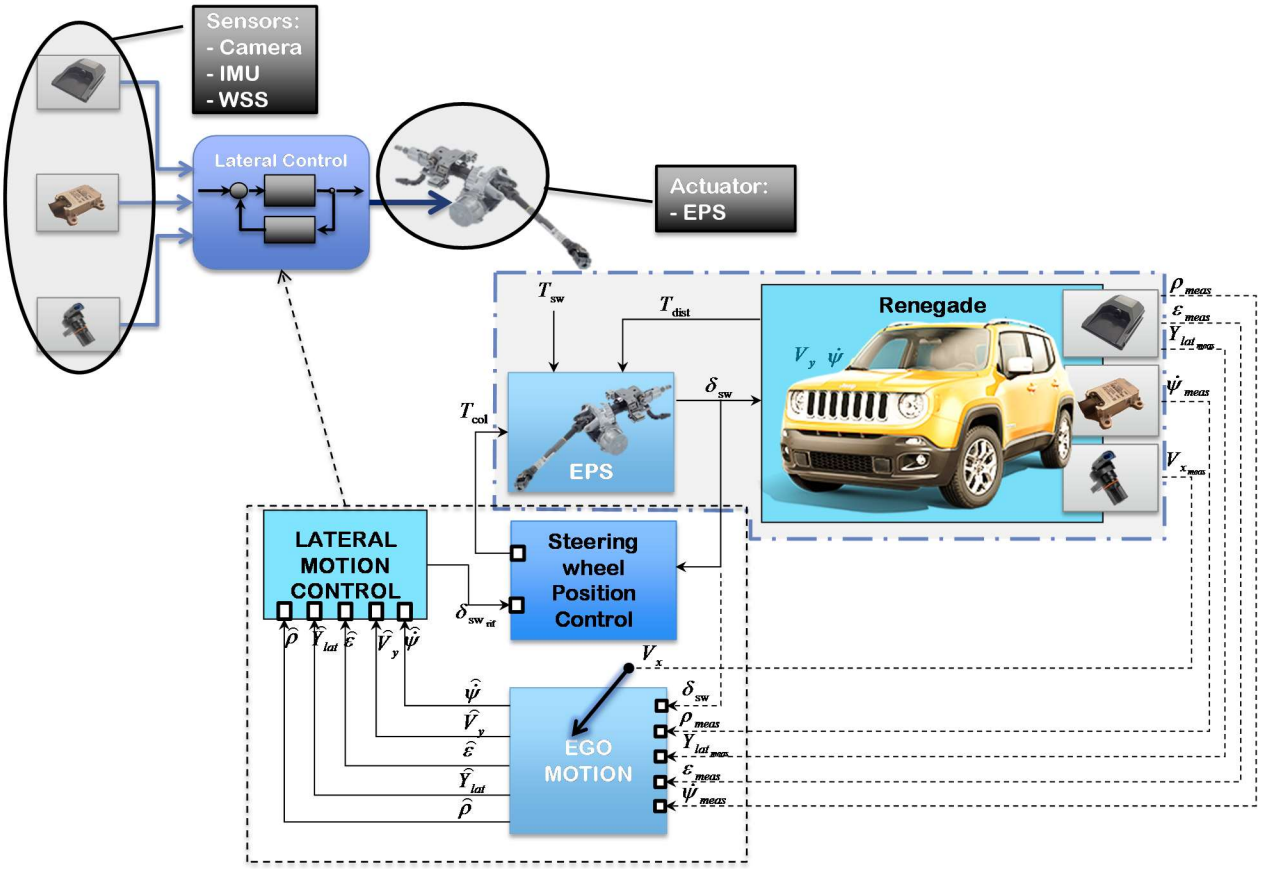

Figure 24: general logic architecture for Ay control.

The main ideas of this approach are described here [13] and sketched in the following figure, where the lateral control is divided in two parts: Steering wheel position control and lateral motion control.

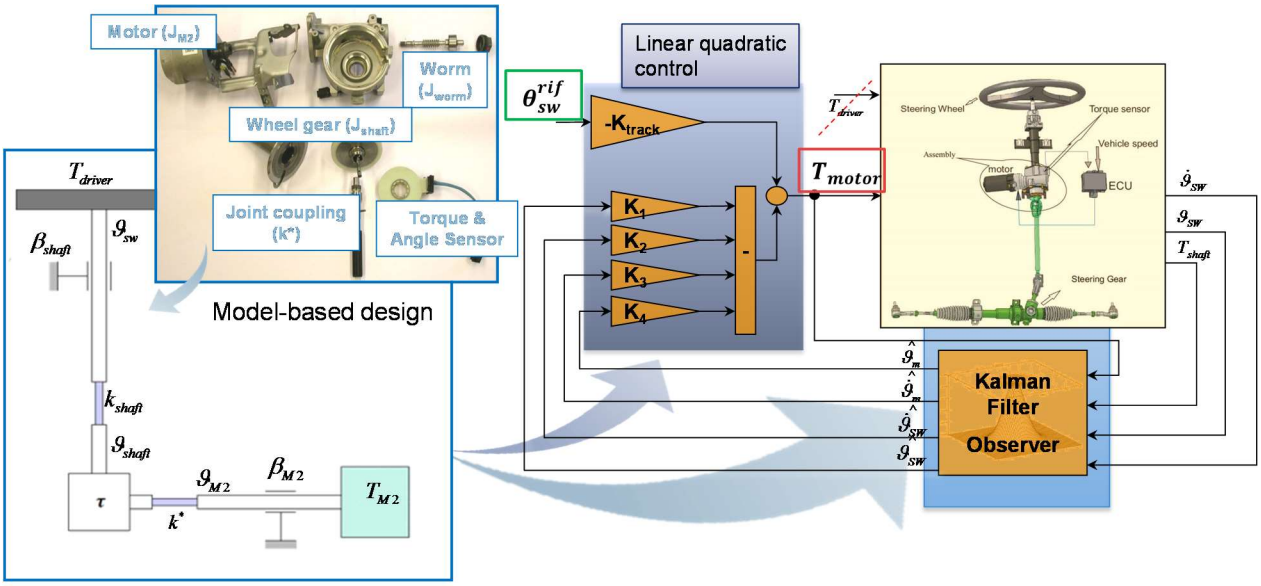

Figure 25: steering wheel position control for Ay control.

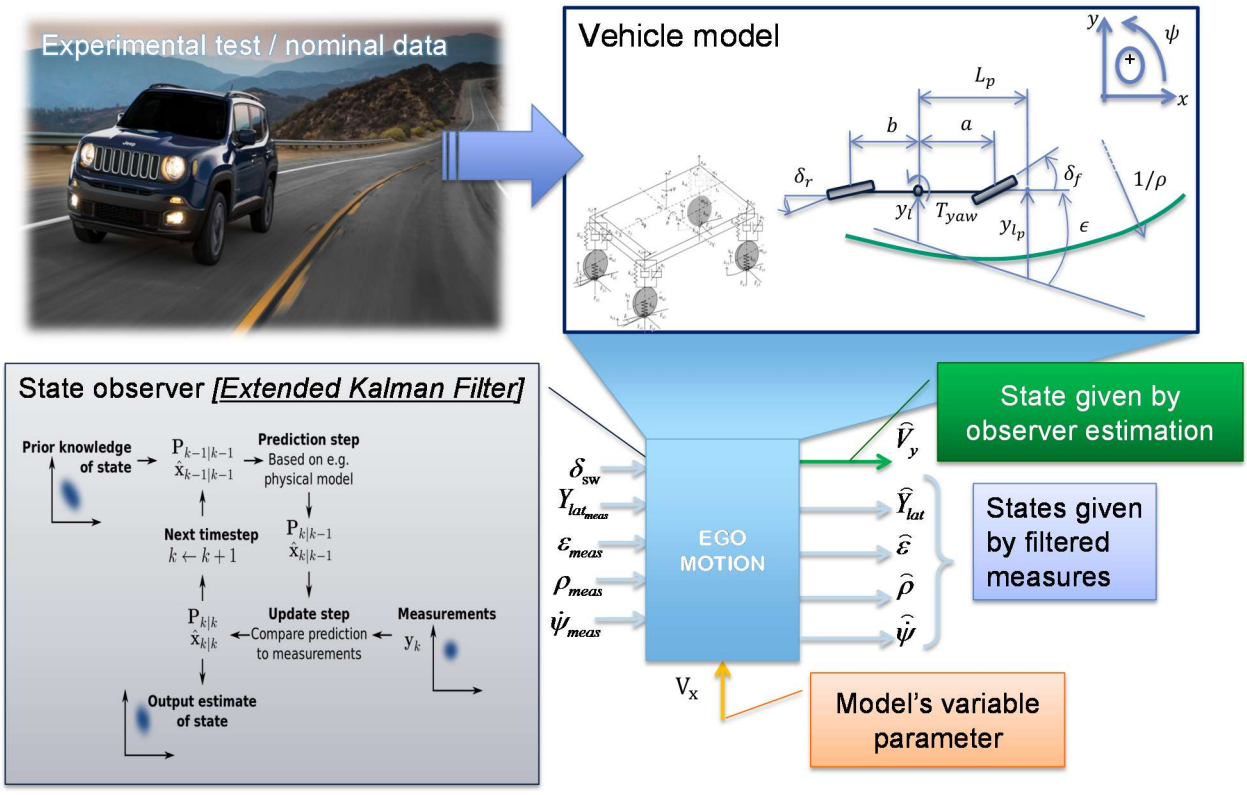

Figure 26: vehicle states estimation for Ay control.

State observer design is one of the key technologies in research for autonomous vehicles, specifically the unmanned control of the steering wheel. Currently, estimation algorithms design is one of the most important challenges facing researchers in the field of intelligent transportation systems. The mathematical model and dynamic response identification of electric power steering column are based on:

- least square identification experiments;
- observability analysis of identified models;
- model simplification via mechanical approach and singular perturbation model reduction;
- two reduced order steering Kalman filter syntheses for estimation of steering column states and disturbances.

This approach has been implemented on CRF TM Car for vehicle path tracking in outdoor experiments. More details are available in [13].

7.4.3. Vehicle states mode management and interfaces

The following figure describes the general scheme for the automate vehicle state mode management:

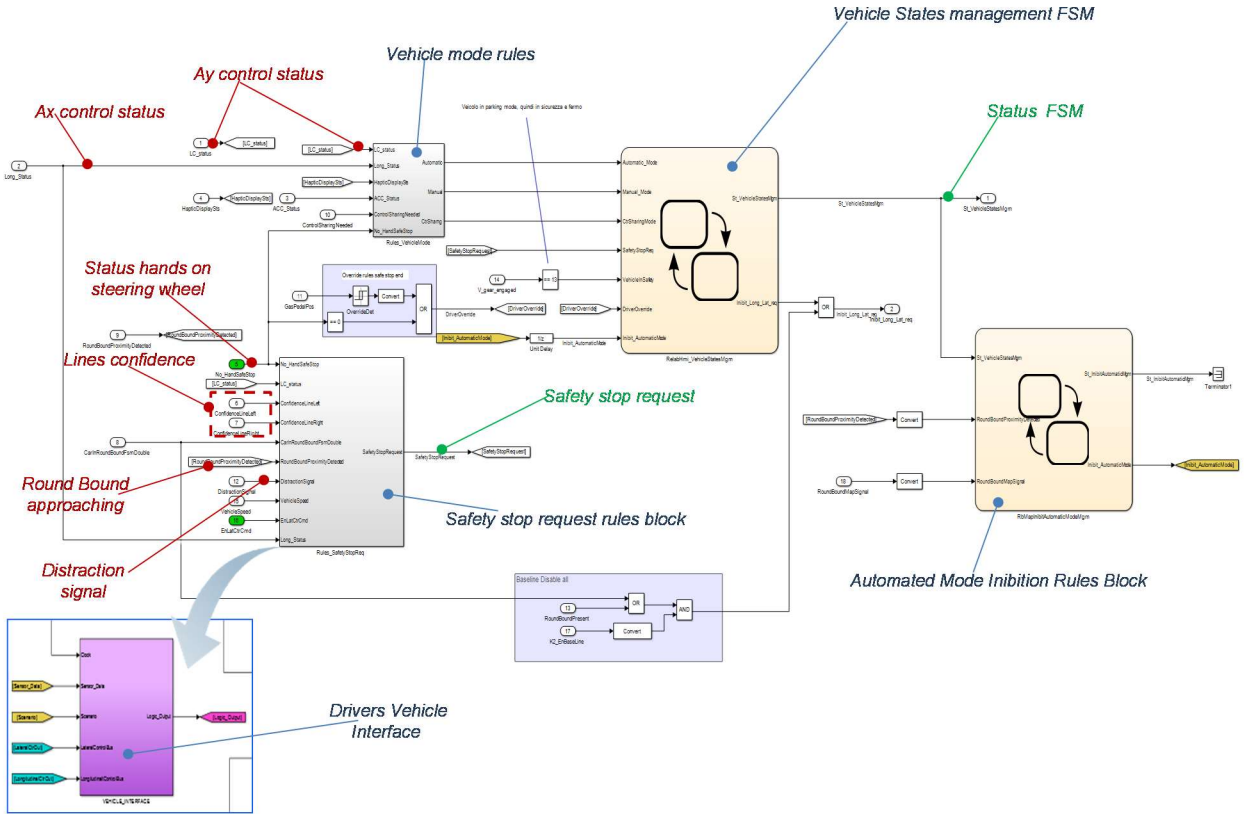

Figure 27: vehicle states mode management for automate and shared modalities.

The transitions from one state to another are sketched in the following figure:

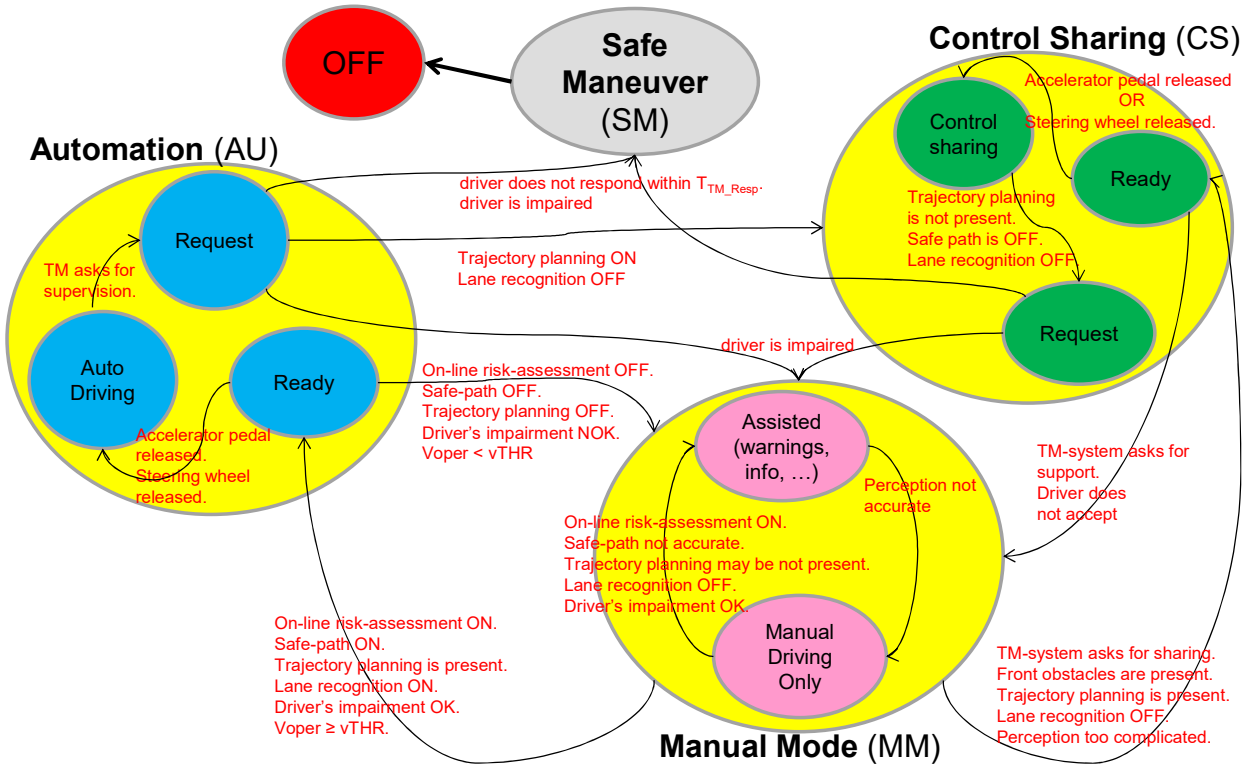

Figure 28: state diagram for the TeamMate car in CRF vehicle.

There are three main blocks (manual, automation and control sharing), described as following (with the related sub-states):

- **Manual Mode** (MM)
  - Manual driving (like a traditional pre-ADAS car)
  - Assisted modality, where the system supports the driver with warning, information and advice, such as nowadays ADAS application.
- **Team-Mate Mode** (TM)
  - After all the conditions are met (*Ready*) the TM car is able to drive autonomously (*Auto Driving*).

- When the TM system needs the driver to supervise (or even intervene), asks for that (*Request*).

- **Control Sharing Mode** (CSM)
  - Several cooperation levels are possible with the system (Control Sharing)
  - Cooperation mode in action means that both agents (human and machine) are sharing the intervention task (e.g. driver is responsible for the lateral aspect, while the TM system of the longitudinal one)
  - Cooperation mode in planning means that the system delegates the final decision task to the human (e.g.: due to limited information)

For the CSM, Control Sharing Mode, in the TM car of CRF, two different cooperation modes have been implemented: *in perception* and *in action*.

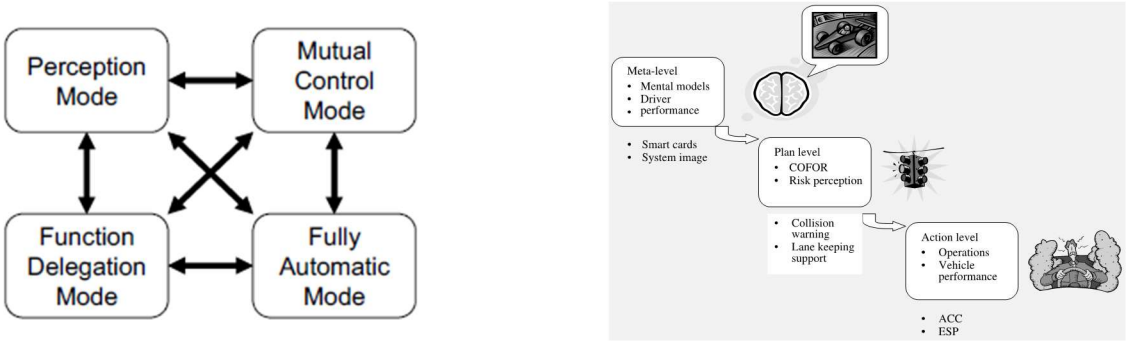

Figure 29: Driver-Vehicle Cooperation Framework according to Hoc.

Cooperation mode in perception means that the system can ask for, or can give, support in the environment perception (e.g. because of limitation in the perception aspects). *Cooperation mode in action* means that the system can

ask for, or can give, support in executing specific actions. This can happen in different forms:

1. by acting on the same actuator at the same time (e.g. both machine-agent and human-agent use the steering wheel);
2. by acting on different actuators at the same time (e.g. machine-agent uses brake and accelerator pedals, while human-agent uses steering wheel);
3. by acting on different actuators at different times (not relevant for this project).

More details can be found in [14].

In our case, we focused especially on condition 2) when approaching a roundabout, the system asks for this type of shared control: "I will take the longitudinal control, while you take the lateral".

In addition, we have another block: the **Safe Manoeuvre Mode** (SMM). When the system needs a support from the driver and thus asks for a cooperation, if the driver is not responding after a given time, or it is necessary an emergency shutdown of TM functionalities, the system enters in the SM. This means that it tries to perform a minimum risk manoeuvre, in order to preserve the safety of the vehicle and its passengers (that is, the vehicle is stopped in an automatic manner).

Finally, all the information related to the system, vehicle and driver states are transmitted to the HMI, following this scheme:

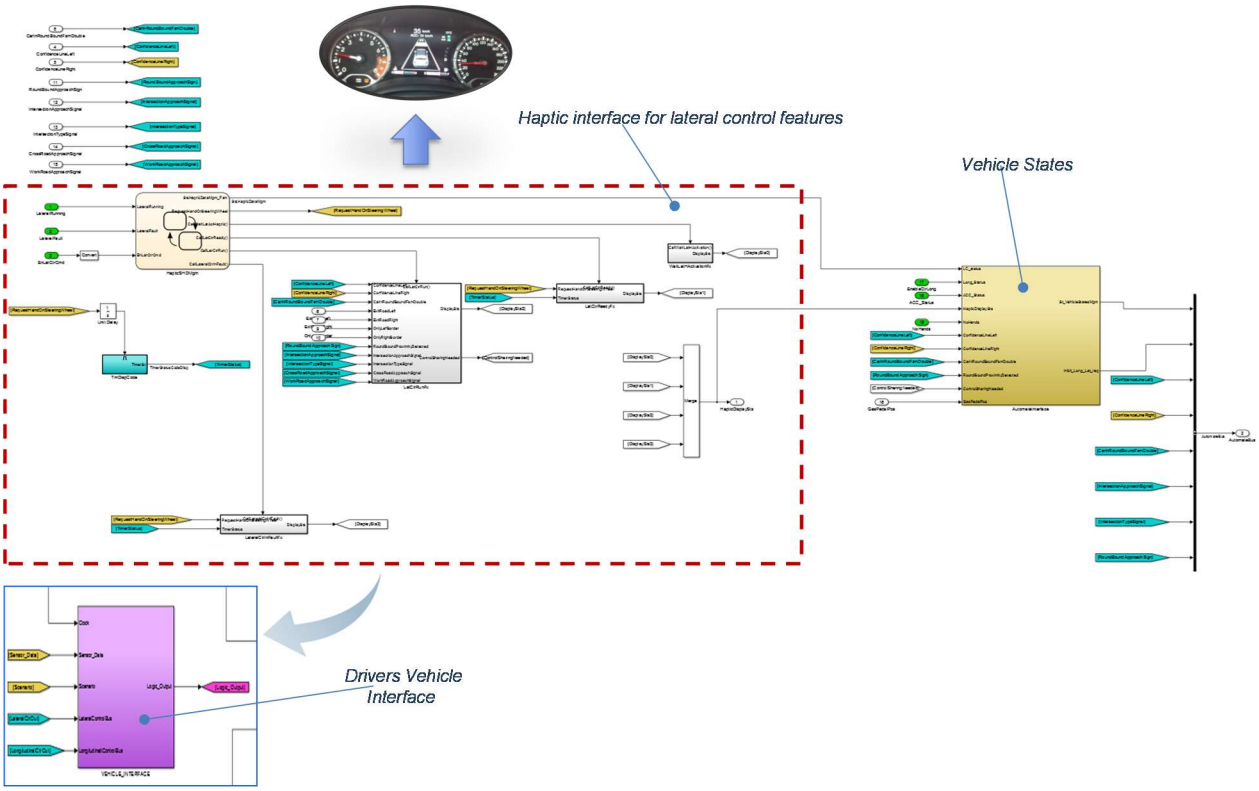

Figure 30: Interface, System and Vehicle States for Automate.

By means of the multimodal HMI, also the request of cooperation is provided to the driver.

In case s/he is distracted, first the system tries to call the driver back into the control loop, by means of the vibrating seats. As mentioned, if the user does not respond within a certain time, a Safe Manoeuvre is actuated (as previously described) and the driver informed. More details on the interaction modalities will be given in deliverable D6.3, together with the results of the evaluation phase.

8 VED Vehicle Demonstrator

In this section, the VED vehicle demonstrator is described, in terms of embedded hardware (HW) components – sensors, actuators, HMI – and in terms of software (SW) solutions (both baseline and TeamMate).

8.1 Enablers and system Architecture

The table below shows the list of enablers integrated in the VED demonstration car. All these enablers have been tested and demonstrated at the Intelligent Vehicle event at Satory:

ID	Enabler	Demonstrator
E1.1	Driver monitoring system with driver state model for distraction and drowsiness	DMS : Yes, • Drowsiness : Yes • Distraction : Yes This module has been successfully installed by CAF and tested in the VED simulator.
E1.2	V2X communication	Yes
E3.1	Situation and vehicle model	From simulator

E3.2	Driving task model - DriveGOMS	Yes
E4.1	Planning and execution of safe manoeuvre	Yes
E6.1	Interaction modality	Different interaction modalities implemented and tested
E6.5	Augmented reality	Yes

Table 10: list of Enablers for VED demonstrator car.

For the final event demonstrator, all enablers have been integrated, tested and demonstrated in the VED car.

8.2 Demonstrator physical Integration

This section describes the status of the VED demonstrator. We will describe the different hardware installed to accommodate different enablers and facilitate the integration process.

In addition, this session addresses the VED demonstrator description in terms of software solution.

8.2.1. Hardware Part

Let us remember (as described in the D5.1) the different sensors and PCs embed in the demonstrator.

VED's demonstrator is an automated vehicle, from a Citroen C4 Picasso platform. This vehicle has two driving modes:

- Standard manual driving and
- Level 4 + self-driving mode.

In manual driving mode, the performance of the vehicle remains compatible with a standard vehicle, with additional access to the connectivity services of Cooperative ITSs (standardized V2X messaging). Access to these services is guaranteed by the on-board platform, which includes several communication media, such as 802.11n standard WiFi, 802.11p and 3G/4G cellular network, compatible with the Martha scenario.

The sensor platform integrated in the Vedecom vehicle (see **Errore. L'origine riferimento non è stata trovata.**) allows a 360° Field of View around the vehicle and provides required information about the vehicle surrounding to support the following 2 main features:

- safely drive in autonomous mode;
- get TeamMate enablers (situation and vehicle model, online risk assessment and path planning) correctly feed with required information.

The sensor platform combines the following sensors:

- 6 Valeo Scala LIDARs ensuring 360° of coverage, with a minimization of blind spots.
- 2 Radars Continental ARS 408, one at the front and one in the rear of the vehicle.
- 2 cameras at the front and 2 in the rear allowing both mono and stereo vision for ground detection and lane marking detection.
- A Velodyne VLP 16 for hd Maps building.
- Atlans IMU for accurate positioning.
- Spetentrio RTK (real time kinematics) GPS.

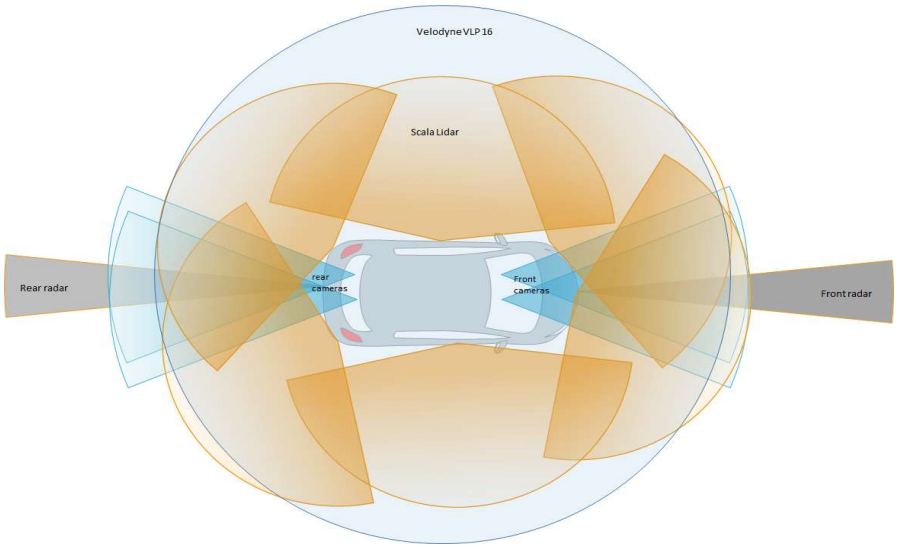

Figure 31 : overview of the Vedecom sensor platform.

The next figure (figure 32) provides the existing relationship between installed HW and enablers defined in the TeamMATE car which rely on them.

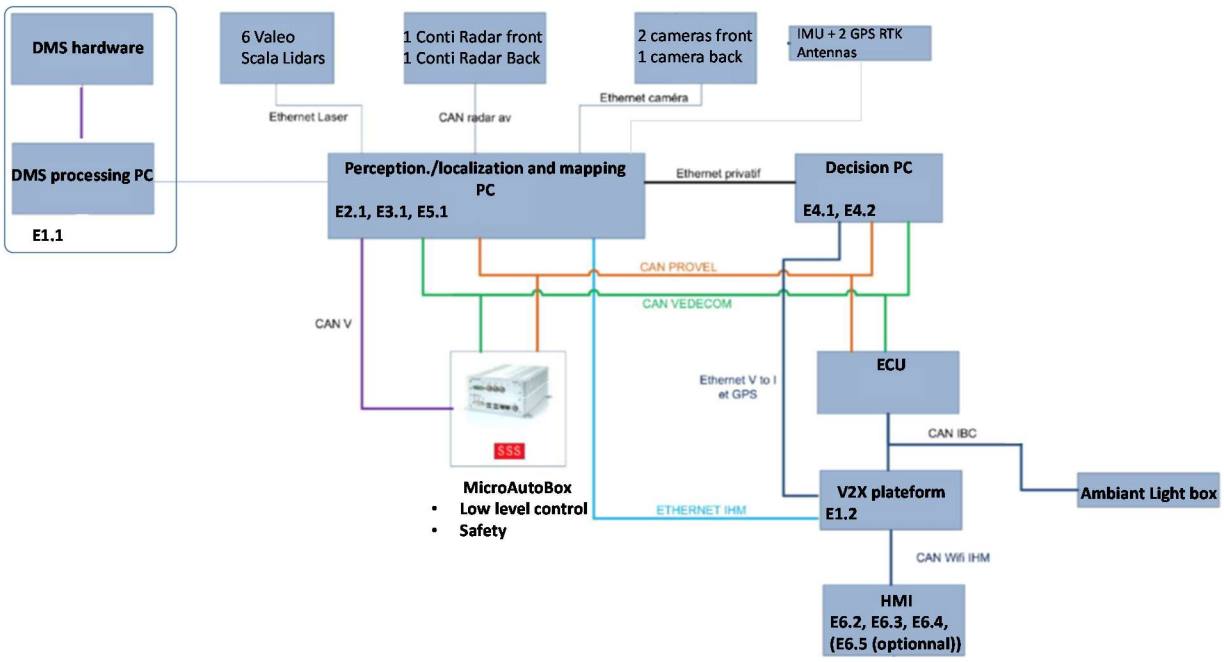

Figure 32: HW architecture and deployment diagram of VED-Demo car.

The following pictures show the physical sensor integration in the Vedecom demonstrator:

Figure 33: Global overview

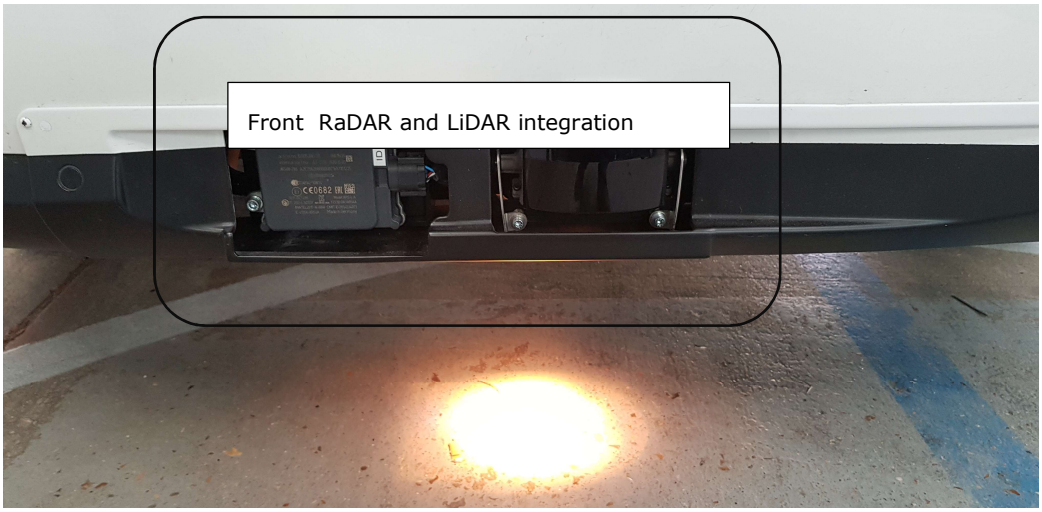

Figure 34: RaDAR, LiDAR and odemeter (light on the ground) installation.

In particular, the figure shows an example of the integration of the front/lateral LiDAR and a front RaDAR in addition to the optic odometer (light on the ground). The LiDAR and the Radar sensors are used for front obstacles detection and feed the enablers E.5.1 and E.3.1. The odometer provides accurate odometer data to the IMU in order to refine the position of the ego-vehicle, which is a very important input to the enablers E3.1, E.4.1 and E.5.1.

Figure 35: Lateral LiDAR integration.

The installations on the roof of the car are shown in the following figure:

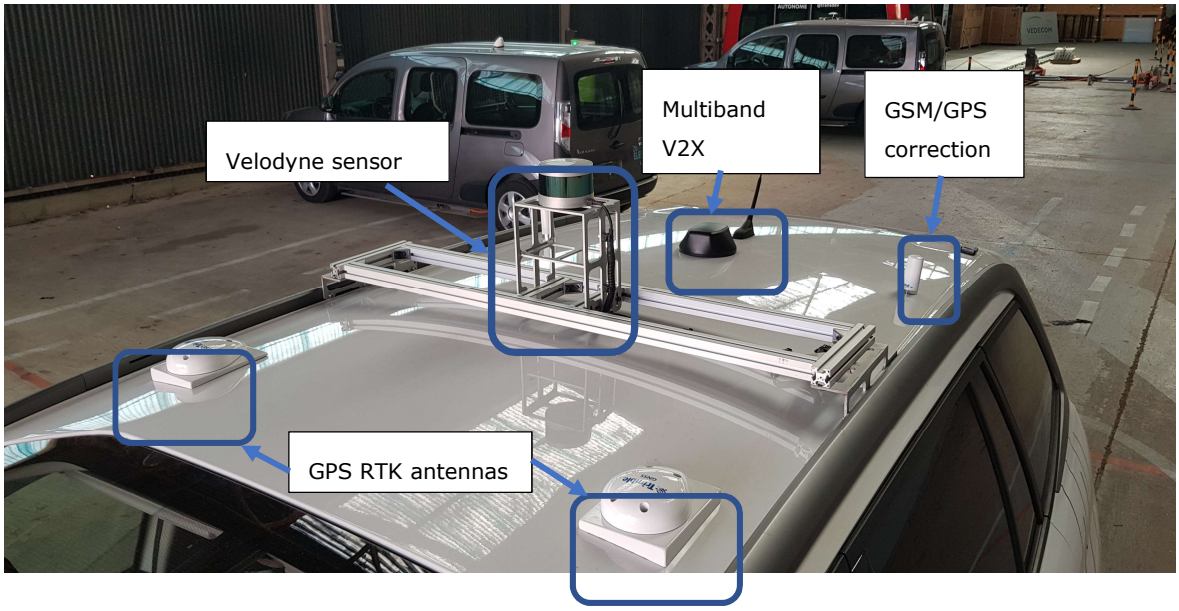

Figure 36 : Integrations on the rough of the VED demo car.

We can see on this figure the following sensors and antennas:

- 2 RTK GPS antennas for the positioning
- 1 multi-band V2X antennas, this platform includes several communication media: 802.11n standard WiFi, 802.11p and 3G/4G cellular network, which is compatible with the Martha scenario, this part will be used by the enabler E1.2
- A Velodyne VLP 16 for hd Maps building, ground and obstacles detection.
- GSM (2G/3G) antenna which can connect to a correction server in order to filter out and decrease the uncertainty of the position received by the GPS.

The electrical system is stored in the car trunk; the next figure gives a panorama of the different PCs switches that are installed. In the car trunk, we can see that we have access to the vehicle data and to the actuators via the

primary and the secondary CAN networks, these variables enrich the inputs of the enablers E.3.1, E.4.1 and E.5.1.

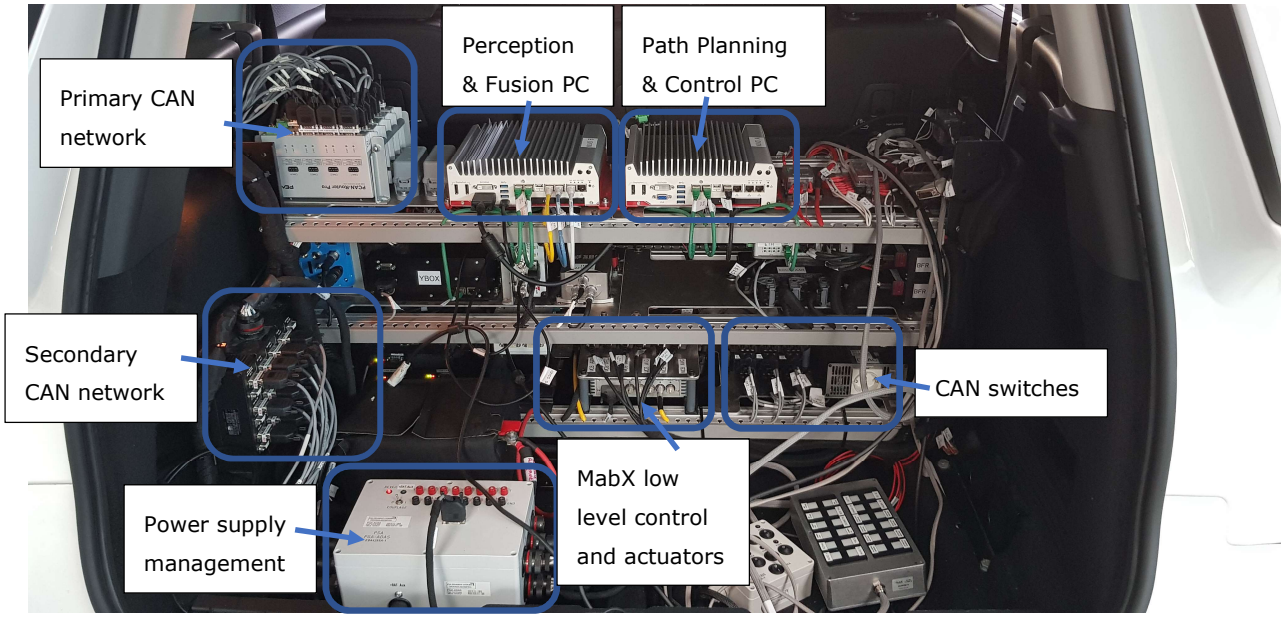

Figure 37: The car trunk and the hardware and electronic part of the vehicle

The installation of enablers in the car is as follows:

- The enablers E2.1, E3.1 and E4.2 and E5.1 are embedded in the Perception and fusion PC.
- The enabler E4.1 is installed in the path planning and control PC.
- Since the enabler E1.1 is a hardware sensor, it is connected to a dedicated mini-PC which is used as a gateway to send the data to the Perception PC.
- The enabler 1.2 is also composed by two parts, the antenna is on the roof and the on-board unit is installed in the car trunk.

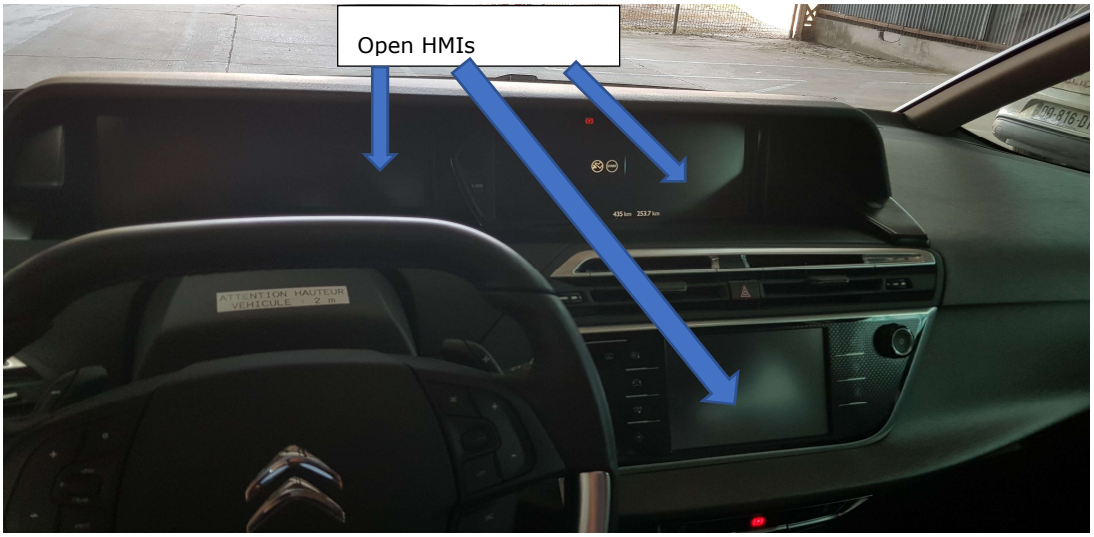

Figure 38: HMI emplacements

Since the car is also equipped with an open dashboard and cluster, there is possibility to install the HMI enablers (E.6.1, E6.2, E6.3 and E6.4) in the different surfaces, as shown in the figure 37.

8.2.2. Software Part

The previous section described the VED demonstrator in terms of material, while this section addresses the VED demonstrator description in terms of software solutions.

The VED demonstrator relies on a base system existing previously to Automate. It provides a base SW environment required for an autonomous vehicle and allows the integration of the Automate enablers.

The VED car base-system is currently managed by a prototyping software solution, called RtMAPS. It allows to define a system in terms of "modules" and is responsible of their executions and data exchanges among them.

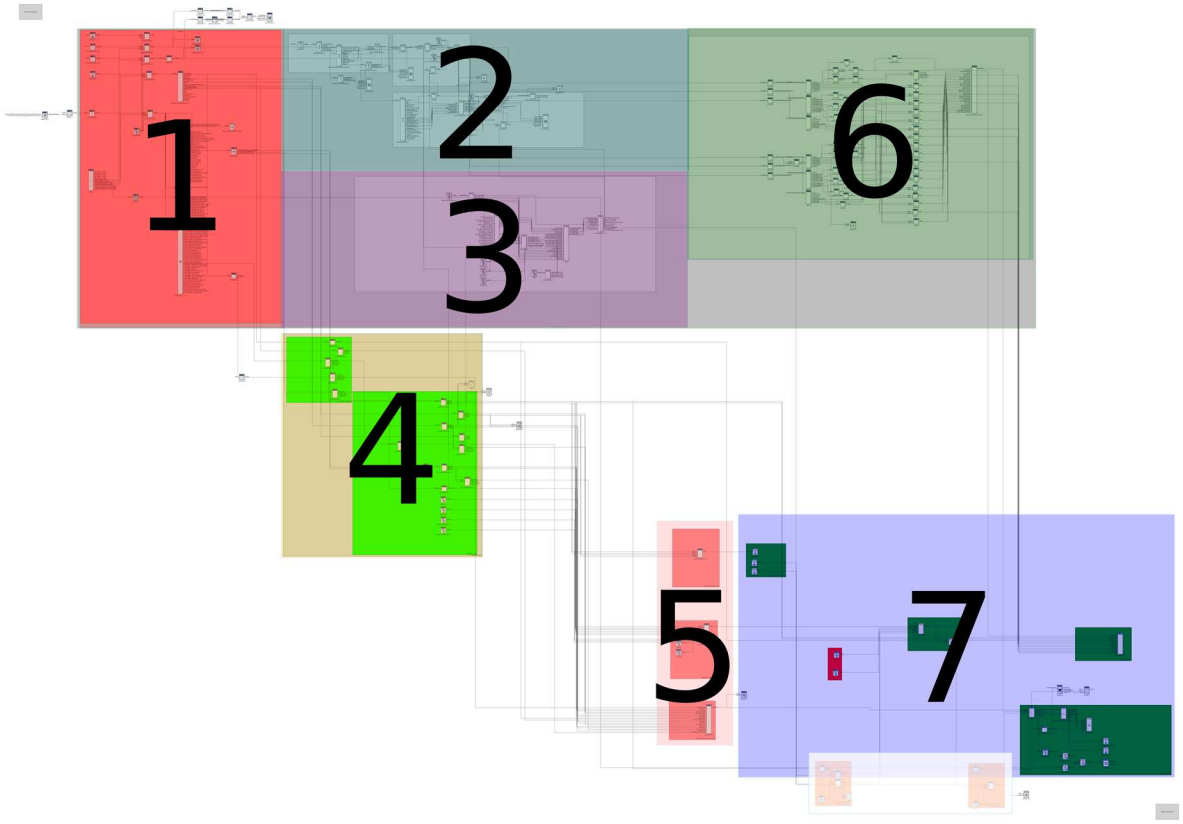

Figure 39: VED Demonstrator – RtMAPS diagrams

The **Errore. L'origine riferimento non è stata trovata.** presents the global SW diagram running in the vehicle. It has been split in different parts (i.e. super blocks), each responsible of a specific aspect of an autonomous vehicle architecture.

The first "super-block" (SB-1) is in charge of sensor/actuator communications. It retrieves data from sensors and send commands to the actuators. It is directly linked (with several links) to the "super-blocks" (SB-2) and (SB-3). (SB-2) is in charge of perception, it gets LIDAR's raw data and generates models of the environment (namely, obstacles) which can be used by the others included functions. (SB-3) is in charge of control. It generates

commands that are sent to the vehicle to operate manoeuvre. (SB-1), (SB-2) and (SB-3) are part of the base system of the VED car.

For the AutoMate project, several others "super-blocks" have been necessary to operate communication with the TeamMate enablers.

It is important to note that the openAPI of the AutoMate project (see T5.1) provides a solution to use this formalism easily by providing a Google ProtoBuf definition of all messages, and the associated source code for different development languages.

The AutoMate API also specify the transport layer, which relies on the standard DDS – Data Distribution Service – and it is compliant with the open source objectives of European projects.

However, the VED demonstrator relies on a third party application (RtMAPS) which already includes helpers to exchange information among modules. It provides solution for primitive arrays, strings and other kinds of simple messages.

To keep VED Demonstrator compliant with the AutoMate API, a set of new messages, respecting the formalism of AutoMate have been defined to guarantee the compatibility of the demonstrator with the AutoMate framework, but which are not directly linked to the Google ProtoBuf interface provided in the AutoMate OpenAPI.

Moreover, as the employed third-party application also provides mechanisms to exchange data, the VED demonstrator does not rely on DDS but on a proprietary solution, which will not interfere with the philosophy of an AutoMate demonstrator as all Enablers will be integrated in the global VED software solution.

As explained in the reminder, the VED car already uses a fully integrated solution to inter-operate several functions inside a global framework. Moreover, the data format exchanged in the base system are not all compliant with AutoMate specification. This is the role of the SB-4 and SB-6, to respectively standardize ego-data and environment-data, SB-5 generates the compliant messages to address the different integrated AutoMate enablers, located in SB-6.

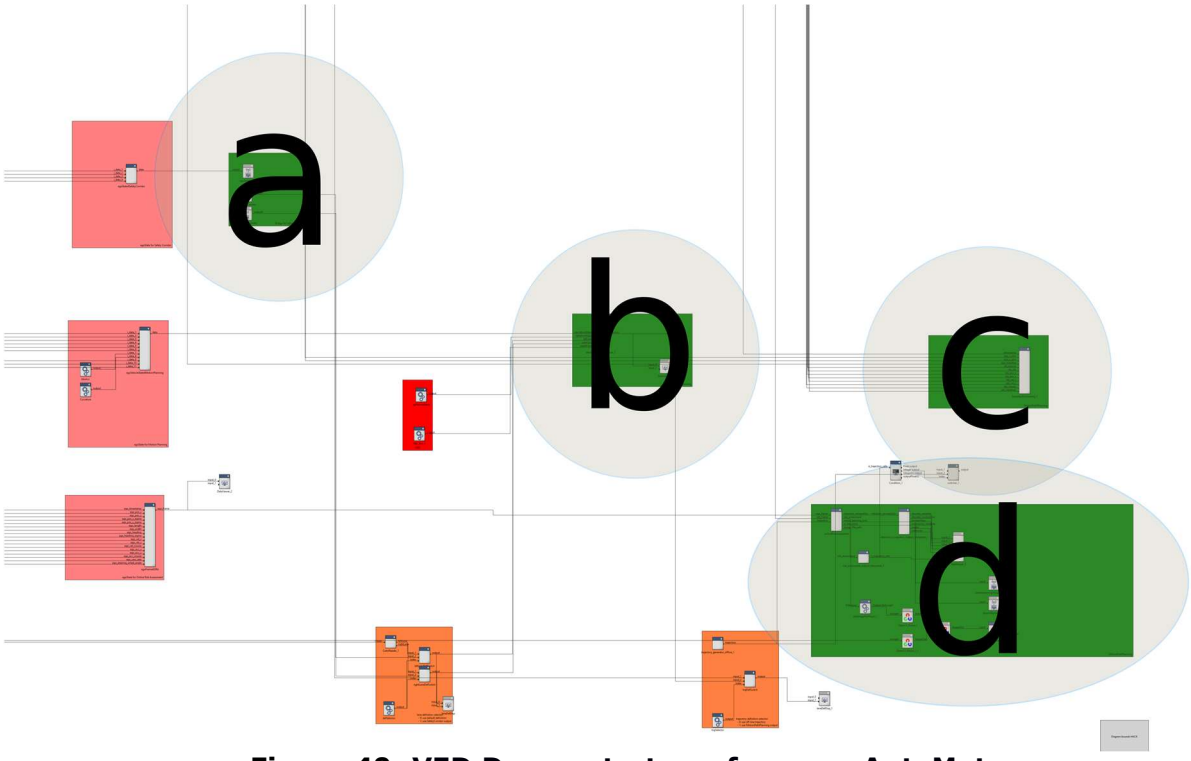

Figure 40: VED Demonstrator – focus on AutoMate

Errore. L'origine riferimento non è stata trovata. presents the SW architecture focussing on the AutoMate enablers. This architecture can be divided in te following main components:

- The left-most components are in charge of adapting the messages coming from the Autonomous Vehicle compliant with the ones expected by the AutoMate enablers according to the defined API.

- Component (a) is related with the SafetyCorridor enable. As this module has been provided as 32 Bits components, a second instance of RtMAPS embeds the enablers and an ethernet communication allows 32 bits components and 64 bits components to communicate in order to get the SafetyCorridor available in the global TeamMate diagrams,
- Component (b) runs the MotionPathPlanning enabler,
- Component (c) runs the SemanticEnrichment,

Finally, component (d) is the master-bloc component, it embeds the 'Online Risk-Assessment', the 'Driver Intention Recognition' and the interface with the Augmented Reality glasses.

9 ULM Vehicle Demonstrator

In this section, the ULM vehicle demonstrator is described for the 3rd cycle of the AutoMate project.

9.1 Enablers and system Architecture

List of enablers that have been integrated into the demonstrator:

ID	Enabler	Demonstrator
E3.1	Situation and vehicle model	For Trajectory planning
E4.1	Planning and execution of safe manoeuvre	To make the driver able to send commands to the vehicle
E6.1	Interaction modality	To make the vehicle able to inform the driver about the vehicle state
E6.2 - 3 - 4	TeamMate HMI (Cluster + audio, Central stack display, HUD)	For Trajectory planning

Table 11: list of Enablers for ULM demonstrator car

9.2 Results of Set-up Tests

Ulm has two computers integrated in the car. There is a perception and an application PC. The communication takes place by using common protocols.

The sensor setup is connected to the vehicle via CAN bus and Ethernet. The tests were successful and the communication is running already. To make sure that the communication works, watchdog signals are used to detect timeouts.

9.3 Demonstrator

The Ulm demonstrator vehicle is based on a Mercedes-Benz E-class T-model and depicted in the following figure:

Figure 41: ULM demonstrator vehicle.

The demonstrator contains two PCs, with Ubuntu 14.04 installed. The first one is the perception PC with an intel i7 processor, 32GB RAM and a Nvidida Gforce gtx1070 graphics card. The second one is the application PC with an intel i7 processor octacore processor, 32GB RAM and a Nvidia Gforce gtx1070 graphics card. Also a dSpace microautobox is built in, where the security system and the vehicle controller are running. The following figure shows the cars luggage space where the PCs are built in:

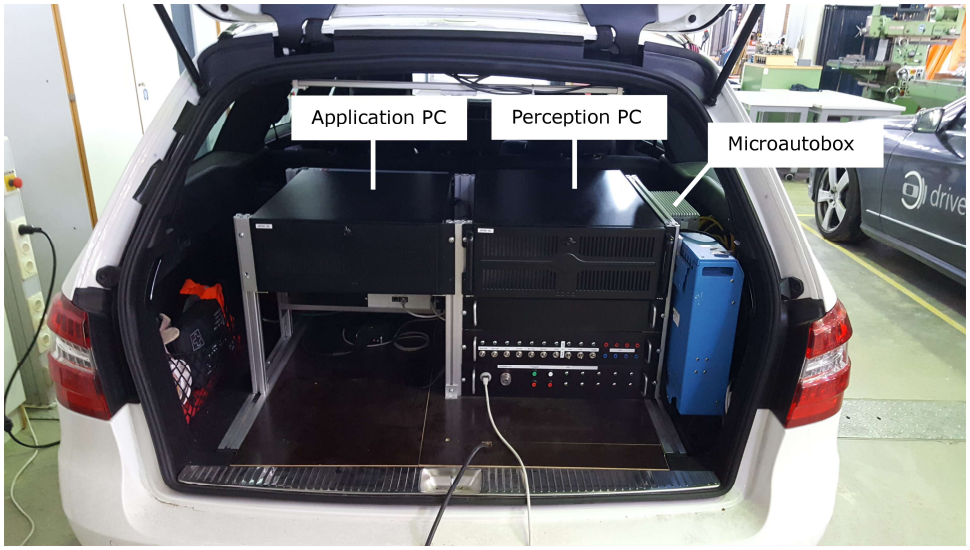

Figure 42: cars luggage space with the illustrated PCs and ECUs.

To perceive the environment, the demonstrator uses a Lidar, a long-range radar and 4 short range radars as to see in the following figure:

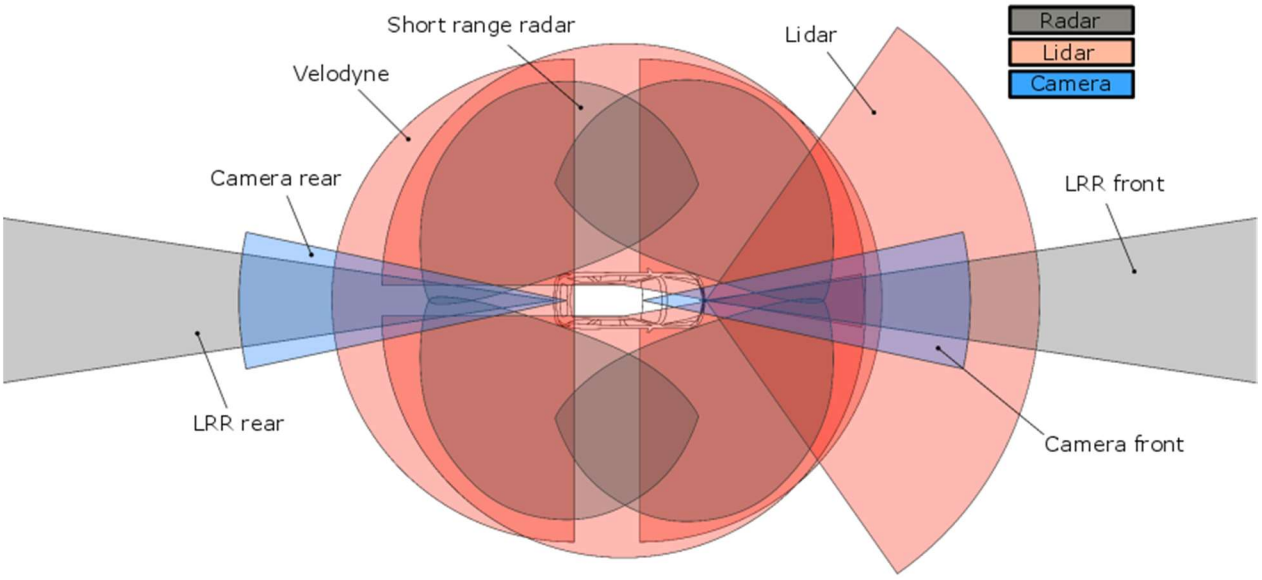

Figure 43: sensorial platform and related field of view.

The status of software component integration is as follows. To enable autonomous driving, a planning and execution of safe manoeuvre (E4.1) has already been integrated. This enabler uses the environmental model that was already integrated on the Ulm vehicle. The following figure visualizes the planned architecture.

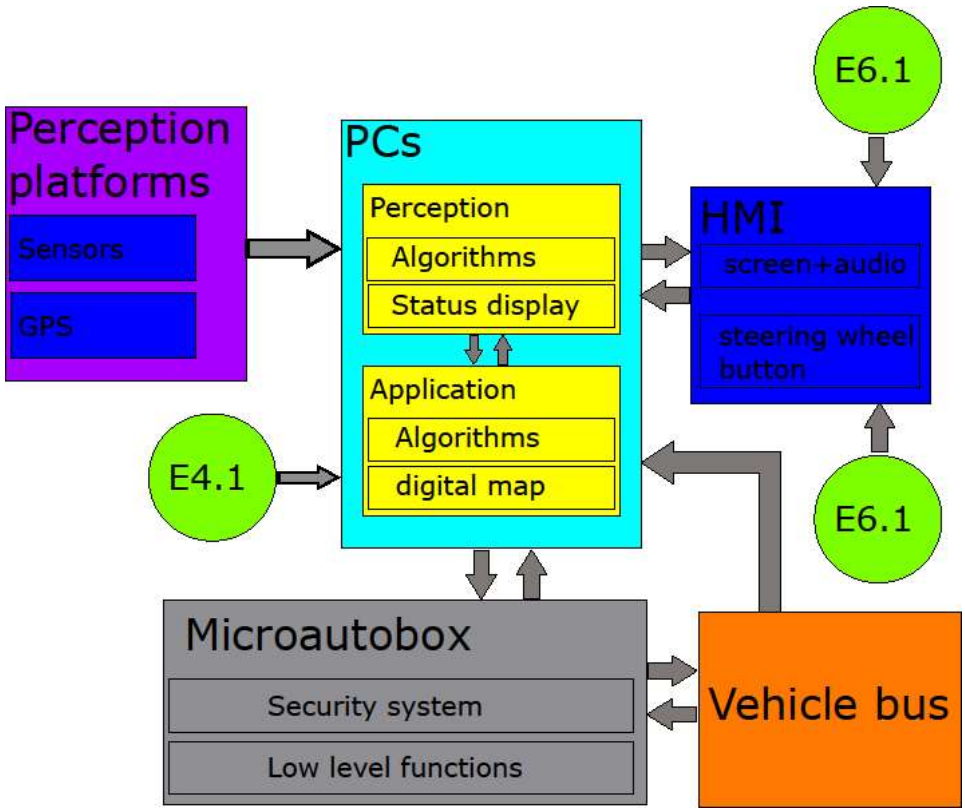

Figure 44: system architecture and deployment diagram in ULM demonstrator vehicle.

To enable permanent interaction with the driver, the teammate HMI (E6.2) together with interaction modalities (E6.1) are integrated within the cars cockpit. Via this system, the driver is able to supply the vehicle with information and the automation can assist the driver. The steering wheel button serves as an indicator to the vehicle, e.g. the driver can release an

overtaking manoeuvre via pressing the button highlighted in the following figure:

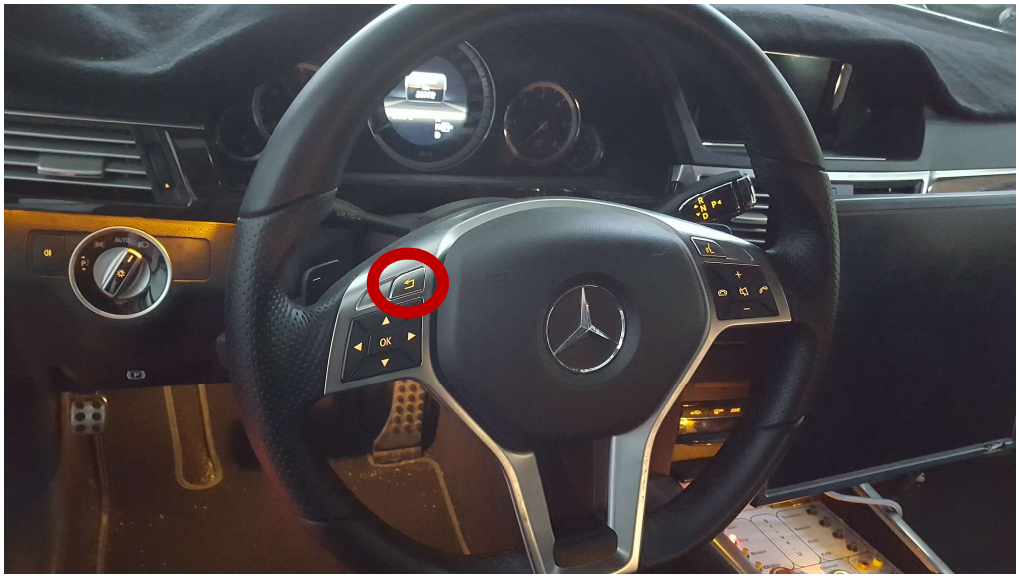

Figure 45: steering wheel button to inform the vehicle about manoeuvre decisions.

10 Conclusions

This deliverable described the integration of the SW/HW components for the TeamMate technologies into the six project demonstrators in the final project cycle (that is, cycle 3), showing how the specifications of the TeamMate system architecture have been implemented and integrated in the TeamMate SW components (from WP2, WP3 and WP4). In details, we started from the conceptual framework from WP1, which has been refined and formalized to specify the system architecture. Within this, all software components have been integrated to build up three demonstrators on driving simulators (from REL, VED and ULM partners) and three on real vehicles (from VED, ULM and CRF partners), to cover all scenarios defined in WP1 (T1.2).

It is worth noting here that input data from automation functions, from maps and from vehicle sensors are provided by the existing vehicle or simulator.

Each enabler is represented by a software component dependent on its own concern. A message bus oriented data exchange between the components is implied to support a communication via one or more channels. The TeamMate system delivers its outputs via acoustic and visual human-machine interfaces to the driver, as well as specific signals/commands to the actuators of the demonstrators.

The demonstrators, illustrated in this deliverable, represent the main output of WP5, thus contributing to milestone M6, by delivering a final version of the demonstrators, ready for the evaluation in WP6 (see deliverable D6.3 for details).

The AutoMate project laid the foundation of an ongoing cooperation between the human-agent and the machine-agent, where the driving control can be in charge of one specific agent, but also can be shared, depending on the specific

task to perform and on the available resources (of driver and of system). This approach opened new view and perspectives, beyond the "frozen" concept of level of automation (LOA), which can be difficult to interpret and understand from end-users point of view. In order to achieve that, the project developed enabler with common interfaces for the software components, which can be used and evolved beyond the AutoMate conclusion. The TeamMate concept was proven to be useful in terms of safety, efficiency, comfort, trust and acceptance (see D6.3). Therefore, the concept is a promising approach into the future design and development of autonomous vehicles and should be extended in next studies (investigation of different types of cooperative control, more use-cases, extended scenarios and so forth). This research is not only restricted to the partners within the AutoMate consortium, but could be shared and worked on by different research institutes and industrial partners using, for example, the results and open accessible data collected throughout the project.

11 References

[1] Rahal M.C. et al. Deliverable D5.1 "TeamMate System Architecture incl. open API for 2nd Cycle". AutoMate project, month 14.

[2] Tango F. et al. Deliverable D5.3 "TeamMate Car Demonstrator after 2nd Cycle". AutoMate project, month 23.

[3] Han S, Choi BS, Lee JM (2008). A precise curved motion planning for a differential driving mobile robot. Mechatronics 18(9):486–494.

[4] Kunchev V, Jain L, Ivancevic V, Finn A (2006). Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. Knowledge-Based Intelligent Information and Engineering Systems (Springer), pp 537–544.

[5] Hwang YK, Ahuja N (1992). Gross motion planning---a survey. ACM Comput Surv 24(3):219–291.

[6] Elbanhawi M, Simic M (2014). Sampling-Based Robot Motion Planning: A Review. IEEE Access 2:56–77.

[7] Marchese FM (2006). Multiple mobile robots path-planning with MCA. 2006 International Conference on Autonomic and Autonomous Systems, ICAS'06 doi:10.1109/ICAS.2006.38.

[8] Gonzalez D, Perez J, Milanes V, Nashashibi F (2015). A Review of Motion Planning Techniques for Automated Vehicles. Intell Transp Syst IEEE Trans PP(99):1–11.

[9] C. Katrakazas et al., "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," Transportation Research Part C: Emerging Technologies, vol. 60, pp. 416 – 442, 2015. [Online]. Available:

http://www.sciencedirect.com/science/article/pii/S0968090X15003447

[10] B. Paden et al., "A survey of motion planning and control techniques for self-driving urban vehicles," IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33–55, March 2016.

[11] J. Ziegler et al., "Trajectory planning for bertha-a local, continuous method," in Intelligent Vehicles Symposium Proceedings, 2014 IEEE. IEEE, 2014, pp. 450–457.

[12] J. Ziegler et al., "Making bertha drive-an autonomous journey on a historic route," IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, pp. 8–20, 2014.

[13] E. Raffone, "A Reduced Order Steering State Observer for Automated Steering Control Functions", in Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics, 426-432, 2016, Lisbon, Portugal

[14] Hoc, J.M., Amalberti, R.: 'Cognitive control dynamics for reaching a satisficing performance in complex dynamic situations', J. Cogn. Eng. Decision.Mak., 2007, 1, (1), pp. 22–55.

[15] F. Tango et al. Project Deliverable D5.2 "Simulated Baseline Cars", month 20, AutoMate project.