Grant Agreement Number: 731993

Project acronym: AUTOPILOT

Project full title: AUTOmated driving Progressed by Internet Of Things

D. 1.5

Initial Open IoT Vehicle Platform Specification

Due delivery date: 30.09.2017

Actual delivery date: 03.10.2017

Organization name of lead participant for this deliverable: CRF

Project co-funded by the European Commission within Horizon 2020 and managed by the European GNSS Agency (GSA)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the GSA)	
RE	Restricted to a group specified by the consortium (including the GSA)	
CO	Confidential, only for members of the consortium (including the GSA)	

Project funded by the European Union's Horizon 2020 Research and Innovation Programme (2014 – 2020)

AUTOPILOT

Document Control Sheet

Deliverable number:	D 1.5
Deliverable responsible:	Visintainer Filippo – CRF
Workpackage:	WP 1
Editor:	Visintainer Filippo, Luciano Altomare – CRF

Author(s) – in alphabetical order		
Name	**Organisation**	**E-mail**
Alén Gonzalez, Silvia	CTAG	silvia.alen@ctag.com
Alesiani, Francesco	NEC	Francesco.Alesiani@neclab.eu
Almeida, Miguel	NEC	miguel.almeida@neclab.eu
Altomare, Luciano	CRF	luciano.altomare@crf.it
Arat, Mustafa Ali	NEVS	mustafa.ali.arat@nevs.com
Brevi, Daniele	ISMB	brevi@ismb.it
Bosi, Ilaria	ISMB	bosi@ismb.it
Di Massa, Vincenzo	THA	vincenzo.dimassa@thalesgroup.com
Ferrera, Enrico	ISMB	ferrera@ismb.it
Galli, Mauro	CRF	mauro.galli@crf.it
Marcasuzaa, Herve	VCDA (VALEO)	herve.marcasuzaa@valeo.com
Martinez, Jose Manuel	CTAG	josemanuel.martinez@ctag.com
Nicoud, Anne-Charlotte	VEDECOM	Anne-charlotte.nicoud@vedecom.fr
den Ouden, Jos	TUEIN	j.h.v.d.ouden@tue.nl
Pasquier, Eric	VEDECOM	eric.pasquier@vedecom.fr
Petrescu, Alexandre	CEA	alexandre.petrescu@cea.fr
Scholliers, Johan	VTT	Johan.Scholliers@vtt.fi
Schreiner, Floriane	VEDECOM	floriane.schreiner@vedecom.fr
Sousa Schwartz, Ramon	TNO	ramon.desouzaschwartz@tno.nl
Van Eert, Marc	TECH	marc.van.eert@technolution.nl
Van Venrooy, Roland	TT	roland.vanvenrooy@tomtom.com
Visintainer, Filippo	CRF	filippo.visintainer@crf.it
Yeung, Michel	CONTI	Michel.Yeung@continental-corporation.com

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V0.1	04/09/2017	ToC definition and tasks assignment	CRF
V0.2	16/09/2017	Provided 1st contents, template for requirements + instructions how to contribute in chapter 2 + actions to start filling in chapter 2.	CRF
V1.3	26/09/2017	Input by partners	T1.3 partners
V1.6	05/09/2017	Input by partners	T1.3 partners
V1.7	08/09/2017	Advanced draft before peer review, not yet final, integrated input from partners	CRF
V1.8	11/09/2017	Version for peer review, integrated input from partners	T1.3 partners
V1.9	28/09/2017	After first peer review, input from partners	T1.3 partners
V2.1	02/10/2017	Final version	CRF
V3.0	02/10/2017	Final check for submission	ERTICO
V3.1	03/10/2017	Minor amendments	CRF

Abstract

This document describes the vehicle IoT platform providing details on functional needs, architecture, requirements and initial specifications, as defined in AUTOPILOT EU project.

Main objective is the specification of an Open IoT Vehicle platform suited to selected Autonomous Driving functions and Use Cases, together with the definition of the actual data to be exchanged from in-vehicle proprietary network and IoT platform components.

The work is the preliminary result of Task 1.3.

Legal Disclaimer

Abbreviations and Acronyms

Acronym	Definition
ACC	Advanced Cruise Control
AD	Autonomous Driving
ADAS	Advanced Driving Assistance System
API	Application Programming Interface
AVP	Automated Valet Parking
CAD	Connected and Automated Driving
CAN	Controller Area Network
C-ITS	Cooperative Intelligent Transportation Systems
DGNSS	Differential Global Navigation Satellite System
EC	European Commission
GA	Grant Agreement
GNSS	Global Navigation Satellite System
HMI	Human Machine Interaction
ICT	Information and Communication Technology
IoT	Internet of Things
LiDAR	Light Detection And Ranging
LTE	Long Term Evolution
OEM	Original Equipment Manufacturer
PF	Platform
PO	Project officer
RSU	Road Side Unit
RTK	Real Time Kinematics
TS	Test Site
UI	User Interface
V2I	Vehicle-to-Infrastructure (communication)
V2V	Vehicle-to-Vehicle (communication)
V2X	Vehicle-to-Everything (communication)
VIN	Vehicle Identification Number
VIP	Vehicle IoT Platform
VRU	Vulnerable Road User(s)
WM	World Model
WP	Work Package

Table of Contents

List of Figures

List of Tables

Executive Summary

Autonomous Driving is expected to progress substantially in the near future towards growing levels of automation, but also higher percentage of circulating vehicles with automated functions. The AUTOPILOT project brings together relevant knowledge and technology from the automotive and the IoT value chains in order to develop IoT-architectures and platforms which will bring Automated Driving towards a new dimension. The AUTOPILOT approach is twofold: utilize the IoT potential for automated driving and make data available to the Internet-of-Things. AUTOPILOT IoT enabled automated driving cars will be tested, in real conditions, at large scale pilot sites.

Within the system definition of AUTOPILOT WP1, task T1.3 is devoted to the in-vehicle IoT platform definition, yielding as outcome initial requirements and specifications. The IoT vehicle is the aggregation point for sensors and actuators which extend vehicle functionalities provided by OEM equipment. It coordinates the connectivity of these devices to each other, to OEM sub-systems and to an external network. The IoT vehicle platform can be considered as the automotive counterpart of the IoT Gateway component of an IoT infrastructure.

This deliverable (D1.5 Initial Open IoT Vehicle Platform Specification) is the outcome of the first definition phase, in which the different stakeholders of the in-vehicle system IoT technologies and devices, OEMs, automotive software and system suppliers, and research entities have been working together to identify the in-vehicle IoT platform based on the general architecture defined in T1.2. The strategic approach aimed at a harmonized system in terms of main IoT capabilities, and standardized interface towards the outside world (ETSI ITS G5, oneM2M) whilst leaving freedom for the internal implementation, on condition to introduce and specify the software framework, components and use of the “open IoT part”.

The operational approach was to start from the use cases and the in-vehicle systems envisaged in the test sites and, based on this, to extract IoT platform features. In particular, “prototype” leaders were identified, for the definition of their own In-vehicle system and its components, both concerning the IoT platform and the legacy components. Prototype leaders worked together with technology providers to outline pragmatically the main components that should build up their prototypes, with a focus on the IoT platform itself. In parallel, core IoT technology experts have been identified, to define the basic functionalities of the In-vehicle IoT platform, the functional architecture and the so-called “parent” requirements (needs). Then, the team identified the detailed requirements, both functional and non-functional, and their degree of applicability. In the last part of this phase, the team addressed the main software components of each implementation of the in-vehicle IoT platform. Beyond in-vehicle IoT specifications, some of the test sites have already specified its physical integration, as annex to the main work.

1 Introduction

1.1 Purpose of Document

This deliverable has the purpose to define initial specifications of the Open IoT Vehicle Platform within the AUTOPILOT project. It lays the groundwork mainly for the development of the in-vehicle IoT platform that will be developed in Task T2.1. It also provides input for the adaptation of autonomous driving functions of T2.2.

Chapter 2 introduces IoT and autonomous driving, recapitulates the use cases and provides an overview of IoT-enabled prototype concepts in the different sites. Chapter 3 defines the in-vehicle IoT platform functionalities and requirements. Chapter 3 provides a list of the software frameworks, tools and standards selected for the vehicle IoT platform, and specifies which of these components are addressed in the different test sites. In addition, Annex I reports technical details of the French prototype vehicle.

1.2 Intended audience

This document is public. Internally in the consortium, the outcomes of D1.5 will be mainly used by the partners active in task T2.1 notably involved in the vehicle IoT platform development.

2 In-vehicle system overview

This chapter reports an overview of the in-vehicle system in the AUTOPILOT project. It briefly introduces the IoT technology applied to Autonomous Driving, referring to the use cases in AUTOPILOT. Then it outlines the general vehicle concept, showing the in-vehicle IoT platform and its interfaces with the other components. The final part of Chapter 2 illustrates at high level the prototypes deployed in the AUTOPILOT test sites, to give a general idea of the differences in the implementation.

2.1 Introduction: from Connected Automated driving to the IoT

IoT extends the principle of internet services to include all possible types of devices thus bringing the advance in accessibility of information from the virtual world to the physical world. For instance, cities are becoming more and more digitalized and connected as numerous sensors have been widely deployed for various purposes (e.g. vibration sensors, cameras, etc.). Those connected devices form a large scale of IoT systems with geographically distributed endpoints and they generate a huge volume of data streams over time. Potentially, the generated data can help us increase the efficiency in various domains such as transportation, safety, and environment.

Autonomous Driving (AD) is an emerging field of transportation which will be of utmost importance. IoT can enable new capabilities, enhanced security, higher efficiency, and much more in AD for various layers. These layers include in-vehicle, cooperation, and also smart city zones.

An in-vehicle IoT platform can manage various sensors inside the vehicle and provide knowledge exchanges between different in-vehicle components. Furthermore, it can enable communication and dynamic collaborative operations for the cooperation zone and connect the autonomous vehicles to the cloud IoT platform through the Internet for the smart city zone (see Figure 1).

Figure 1 – Different layers of IoT: 1) Car zone, 2) Cooperation zone, 3) Smart city zone

At the state of the art, the interoperable standard ETSI ITS G5 (V2X) as well as cloud-connected services, extend and complement the vehicle sensors to build a dynamic map of the vehicle surroundings. Vehicular networking and connected cloud services are thus major enablers for the future mobility and Autonomous Driving. Also in AUTOPILOT, V2X will be one of the links used to collect information from different sources [1] considering the connectivity roadmap of C-ITS, and the several initiatives going on in the European sites. But ETSI ITS G5 connectivity (or cellular V2X) is not sufficient for IoT, since the exchanged data have to be managed according to the "IoT World", in order to benefit from its characteristics (abstraction, scalability, etc.).

The in-vehicle IoT platform is the logical component allowing a connected vehicle to become part of

the “IoT World”. To achieve this, several possible software & hardware solutions are possible and could be applied based on the specific implementation choices. For the initial specifications of the in-vehicle IoT Platform, a subset of this solutions (§ 4.1) are being considered in the different AUTOPILOT Test Sites, in a federated approach.

Connected vehicles with no in-vehicle IoT platform can also benefit from the IoT, either through the roadside units (e.g. ETSI G5/ETSI OneM2M “adapter” software) or through cloud back-end services (e.g. e-Horizon connected to IoT in the cloud). These indirect approaches however are out of the scope of D1.5.

Another key aspect for Autonomous Driving is data fusion. At the state-of-art, solutions rely on in-vehicle sensors such as cameras, radars, LiDARs, etc., and on data fusion engines that allow the vehicle to remove duplicated information and consolidate its local dynamic map (LDM). The LDM stores objects describing the current situation of a vehicle: its position, obstacles, neighboring vehicles, points of interest and road events. Overall, the fusion of information from as many reliable sources as possible (in-vehicle sensors, V2X, cloud services, and other connected devices) will substantially increase the perception of the environment, giving more reliable information to the autonomous decisions and guaranteeing the needed redundancy for higher SAE levels of automation [3]. The IoT approach helps to have a scalable management of all these new sensors.

Initial AUTOPILOT IoT architecture is defined and described in detail in [6] as output of the activities in Task 1.2. This document provides a functional view on the overall IoT architecture which starts from the “things” (including AD cars), continues with network layer, IoT layer, and ends with AUTOPILOT applications. The network layer covers all possible communication functionalities including vehicle-to-vehicle, vehicle-to-RSU, and vehicle-to-cloud communications. The IoT layer can be physically located in the cloud or in the edge (e.g. RSU). It covers functionalities such as context management, analytics, IoT device management, semantics, processing and service optimization and so on. AUTOPILOT applications are specific for the defined use cases.

While AUTOPILOT D1.3 [6] provides an overall view, it mainly focuses on specification of the IoT layer. Vehicles are included in the overall view as one of the key elements of the “things” layer. The components and the IoT platform inside vehicle is the aim of present deliverable D1.5. In this document, we specify the components of AD vehicles including sensors, in-vehicle IoT platform and define the initial in-vehicle IoT platform architecture.

2.2 AUTOPILOT Use Cases

This section reviews the use cases and services previously defined in document D1.1 of Autopilot [2]. The goal is to provide a brief summary of each use case and further elaborate on the specific challenges and requirements of the overall system integration between vehicle and external IoT cloud components.

2.2.1 Automated Valet Parking

Automated Valet Parking (AVP) has two main scenarios:

- Autonomously parking of the vehicle, after the driver has left the car at the *drop-off point*, which may be located near the entrance of a parking lot
- Autonomous collection of the vehicle. When the driver wants to leave the site, he/she will request the vehicle to return itself to the *collect* point, using (for example) a smartphone app

To navigate safely around the parking lot to (and from) its parking place, the automated vehicle uses driving functions based on knowledge about the environment around the vehicle. An example would

be a navigation functionality based on a digital map, positions of the automated vehicle and vacant parking spots. The vehicle can use its own functions and sensors to accomplish this task, but it can also benefit from accessing IoT platforms which can provide data and functions based on IoT enabled sensors like parking cameras. Parking cameras can also warn the vehicle about other vehicles and pedestrians in the parking lot. Furthermore, IoT platforms may offer booking and payment services.

The IoT platform can identify empty parking places, and hence inform the car or its destination. Besides navigation, also functionality on the tactical decision level may be shifted to the IoT-platform so that less functionality is required on the vehicle itself. Through the use of IoT, the IoT platform can monitor and/or coordinate traffic on the parking lot and do efficient route planning based on real time available traffic. Hence, the IoT platform will exchange information on the dynamic and static obstacles on the parking lot and/or the route to be followed to the vehicle.

Interaction between the vehicle and the outside world/IoT platform is needed for:

- Determination of the destination point (parking place, collect point)
- Identification when the vehicle is ready to move unmanned to the destination (parking place or collect point), e.g. when the driver has moved out of the proximity of the vehicle or has locked the doors. The IoT platform informs the vehicle when to start moving to the destination. In case of motion to the collect point, the vehicle should be woken up
- Synchronization of the vehicle's world model and the model at the IoT backend, including more detailed layout of the parking place and the location of dynamic objects
- Navigation to the destination, following a route either determined by the vehicle or the IoT platform, while avoiding obstacles detected by either the vehicle sensors or the IoT platform
- Remote connection between the vehicle and a control centre during unmanned driving. The operator of the control centre becomes the "driver" of the vehicle (in Finland and the Netherlands the vehicle driver does not have to be in the vehicle). Remote connection includes uninterrupted observation from the vehicle, e.g. through traffic cameras. The vehicle should respond to control commands of the control centre, e.g. performing emergency stop, or take-off
- Transmission of observation data from IoT sensors or of events detected by IoT sensors (e.g. pedestrians or other objects on the parking place) to the vehicle
- When the vehicle has arrived at the parking place, the vehicle goes to a low power consumption mode, with acknowledgement of the control centre or the IoT platform.
- If the driver requests the vehicle to the control point, the IoT platform has to validate the request, and if the request is valid, start the collecting process
- At the collect point, the vehicle has to validate that access to the vehicle is provided to the authorized driver

2.2.2 Highway Pilot

In the Highway Pilot use case, a cloud service merges the sensors measurements from different IoT devices (in particular from vehicles and roadside cameras) in order to locate and characterize road hazards (potholes, bumps, fallen objects, etc). The goal is then to provide incoming vehicles with meaningful warnings and adequate driving recommendations (taken into account by the Autonomous/Assisted Driving functions) to manage the hazards in a safer or more pleasant way. Built upon collective learning of IoTs, this 6th Sense Anticipation mechanism aims at replicating people driving experience and road awareness into autonomous vehicles.

A second mechanism that skips the cloud service learning part is triggered whenever one observed anomaly can confidently be characterized as an immediate and critical danger right away at vehicle

level. Such a thing may occur with an obstacle in the middle of the road (like a broken car, rock). The cloud collective learning process may resume immediately after that to reinforce the knowledge about the hazard (persistence or removal).

The vehicle needs an on-board IoT-like platform to handle the various sources of data (IoT sensors like Lidar, Camera, IMUs, etc) with the various local detection services. Also this platform will handle the maps/alerts data processed and displayed within the vehicle's screens (IoT displays).

In addition, an interaction between the vehicle and the outside world/IoT platform is needed for:

- The transfer from the Vehicle to the Cloud of all likely anomalies detected by IoT sensors within the Vehicle
- The retrieval into the vehicle of High Definition (HD) Maps provided by the Maps provider
- The retrieval into the vehicle of Live Maps provided by the Maps provider
- The retrieval into the vehicle of Road Hazards alerts emitted by the Cloud service
- The retrieval into the vehicle of ADAS adaptation instructions published by the Control Center
- The broadcast to nearby vehicles and road infrastructure equipment of alerts for hazards qualifying as obstacles with immediate danger
- The reception of messages from nearby vehicles and road infrastructure equipment

2.2.3 Platooning

The platooning use case consists of vehicles following another preceding vehicle at relatively close distance. This brings benefits in terms of traffic throughout and homogeneity, enhancement of traffic safety due to small speed variations and relatively low impact velocities in collisions, and reduction of fuel consumption and emissions due to lowering the air drag.

A few variants of platooning will be deployed and evaluated in AUTOPILOT:

- An urban variant to enable car rebalancing of a group of driverless vehicles (up to 4), involving one driver in the leading vehicle driving at a maximum speed of 30 km/h. As the first deployment of highly automated driving will likely be on dedicated lanes, the scenario to be implemented in Versailles will start from a parking where the driverless vehicles will join the leading vehicle to form a platoon. The platoon will then move to another location, where driverless vehicles will also use automated parking
- A highway variant at Brainport, where one or more highly automated vehicles follow a leading vehicle on the highway. Also in this variant, the usage of a dedicated lane is considered, i.e. the electronic allowance of the emergency lane is explored. The scenario will start from a platooning appointment that has been made and will consider the forming of the platoon. An approaching lead vehicle will pick up the following vehicle, which has just arrived from automated parking. Dynamic pick up of the vehicle will be explored, where platoon forming is done while driving. After the platoon is formed, it will drive from the city of Helmond to the city of Eindhoven. On their way, other vehicles may join or leave the platoon dynamically

Driving in a platoon requires vehicles to use inter-vehicle communications to anticipate maneuvers of other vehicles in the platoon. The following vehicles have automated steering and distance control to the vehicle ahead and the control is supported by advanced V2V communication. IoT enables the interaction between platooning vehicles with infrastructure, traffic management and services, thereby bringing more efficiency in terms of platoon maneuvering, such as platoon forming. In addition, IoT can effectively extend the range of awareness from what it is currently provided by in-vehicle or road-side sensors, as well as improve confidence of detected dynamic

objects and traffic information by providing redundant sensor information.

The interaction between vehicle and external entities via IoT platform is required for exchanging coordination parameters/commands to improve efficiency in terms of platoon maneuvering capabilities, such as platoon forming and vehicles organization:

- Vehicles can send requests to the platooning service based on input from the driver, to make themselves available as vehicle platoon leaders
- Position, velocity and planned route including intents for exiting motorway via off ramps or entrance via on-ramps is sent to platooning and car sharing services for overall traffic management and platooning coordination
- Traffic light status and speed recommendations are sent from the platooning service to the vehicles in order to keep the platoon formed in intersection scenarios
- After entering the highway, vehicles in the platoon send request to the platooning service for confirmation of permission to use priority lanes, e.g., emergency lane
- Platoon vehicles receive speed advice from the platooning service based on current traffic conditions
- If the platoon is not disengaged as determined by the platooning service (e.g., after leaving designated allowed area), the platooning service may send a command to move and stop platoon vehicles in a nearby emergency lane
- Vehicles which are not part of a platoon also send their position, velocity and planned route to IoT services, for instance traffic management, to improve overall traffic coordination
- World model data, generated by the vehicle based on internal sensors' data, is shared with IoT cloud services. These services can improve global awareness on the road by processing and fusing world model data from different road users (vehicles and VRUs) and sending it back to each vehicle to effectively extend and increase reliability of each vehicle's local world model
- Information aiming for (redundant) localization can be exchanged between vehicle and IoT services, e.g., real-time HD maps and GPS error corrections with RTK DGPS services

2.2.4 Urban driving

Urban Driving assisted by the IoT has the main objective to support connected and automated driving (CAD) functions through the extension of the electronic horizon in automated vehicles. This means that the vehicle can process data from external sources complementing those provided by its own sensors (cameras, LIDAR, radars, etc.).

In the framework of AUTOPILOT, the automated urban driving use case focuses on the interaction with traffic lights and legacy traffic, on the robustness of the AD functions of the vehicle, safety when dealing with vulnerable road users, and positioning. Indeed, in order to enhance the performances of AD functions in an urban environment, the vehicle needs to extract relevant information from:

- Traffic lights at intersections
- Hazard warnings
- Infrastructure cameras (detecting events such as pedestrians, bicycles, obstacles)
- Other vehicles sharing their own sensors' data
- VRU sharing data by means of connected devices (e.g. smartphones, smart glasses, etc.)

This additional information is transmitted wirelessly to an on-board IoT platform and permits the CAD system to adapt its behaviour accordingly. The on-board IoT platform will have to act as a new source for the perception module of the CAD system but also for the other vehicles.

Moreover, by the integration of an IoT platform into the in-vehicle architecture, the electronic horizon extension is taken to a new dimension. With C-ITS, communication range is limited to the traffic lights of approaching intersections. IoT integration will set the basis for enabling the access to a larger volume of data (different traffic lights, routing, pedestrians, hazard warnings, priority to automated vehicles).

Integrating the IoT platform in the CAD system introduces the challenge of fusing all this new information with the existing on-board data (provided by the vehicle sensors). As mentioned above, a large amount of data being more or less informative (the exchanged data may range from raw sensor data to hazard warnings) can be available. Therefore, it is required that the data obtained through IoT is described using standard description.

As an example, VRU detection for safe driving has to be completed in AUTOPILOT and requires that all the dynamic obstacles located in the surrounding environment of the CAD system are expressed in terms of position, speed and acceleration. Therefore, the formatting, encoding (syntactic) and meaning (semantics) of the different structures is basic for the use case, more details about this syntactic and semantic interoperation are available in section 4.1.7.

Following an IoT architecture, the information and the interaction between the vehicle and the outside world/IoT platform are needed for:

- Traffic light status and time to change
- Hazard warnings
- VRU detection by the infrastructure
- VRU detection by the vehicle
- VRU detection by connected objects such as smartphones, smart glasses, smart watches, etc.

2.2.5 Real-time car sharing

A car sharing service is intended as a tool to enable different customers to make use of a fleet of cars (either self-driving or not) shared amongst them. Car sharing can be interpreted as a service that finds the closest available car and assigns it to a single customer, or drive the closest available car to the interested customer. Car sharing can also be intended as ride sharing, where multiple customers that possibly have different origins and destinations share a part of the ride on a common car (either manually or self-driven). Finally, car sharing services can also be considered as services that allow customers to specify pick-up and drop-off time-windows to increase flexibility and planning.

The car sharing service will match vehicles with customers' requests of origin and destination locations and several other requests (ride alone or with somebody, possibly time-windows.
The car sharing will utilize a new authentication service using mobile App both for web and car access. The authentication will follow privacy by design approach, user will be provided with anonymous or semi-anonymous credentials while preserving a possibility of investigation of incidents by authorized entities.

Figure 2 shows the target architecture for the car sharing use case. The focus here is on the interaction between the various car sharing actors and components and the Open IoT platform common services as a whole, represented as one box.

Users should book cars and manage (modify, cancel, etc.) their bookings using the central car sharing service through a mobile or desktop application, referred to as the client app.

The proposed architecture requires that shared vehicles should be equipped with the necessary

hardware and software to: (1) communicate their probe data (GPS location, speed, etc.) to the open IoT platform common services and the car sharing service, and (2) compute optimal routes and their costs (distance, energy consumption, etc.) given an assigned destination. These may be fully implemented inside the vehicle itself, or may be delegated to external web services.

IoT-enabled devices and vehicles of the IoT ecosystem should publish relevant events (traffic, accidents, weather, parking spot availability, etc.) on the open IoT platform. In order for the car sharing service and shared cars to be notified about events that may affect their planned trips, they should subscribe to the open IoT platform for relevant events.

The open IoT platform should be responsible for collecting data from the various IoT devices, storing them, and communicating the relevant pieces of data (events) to the subscribers.

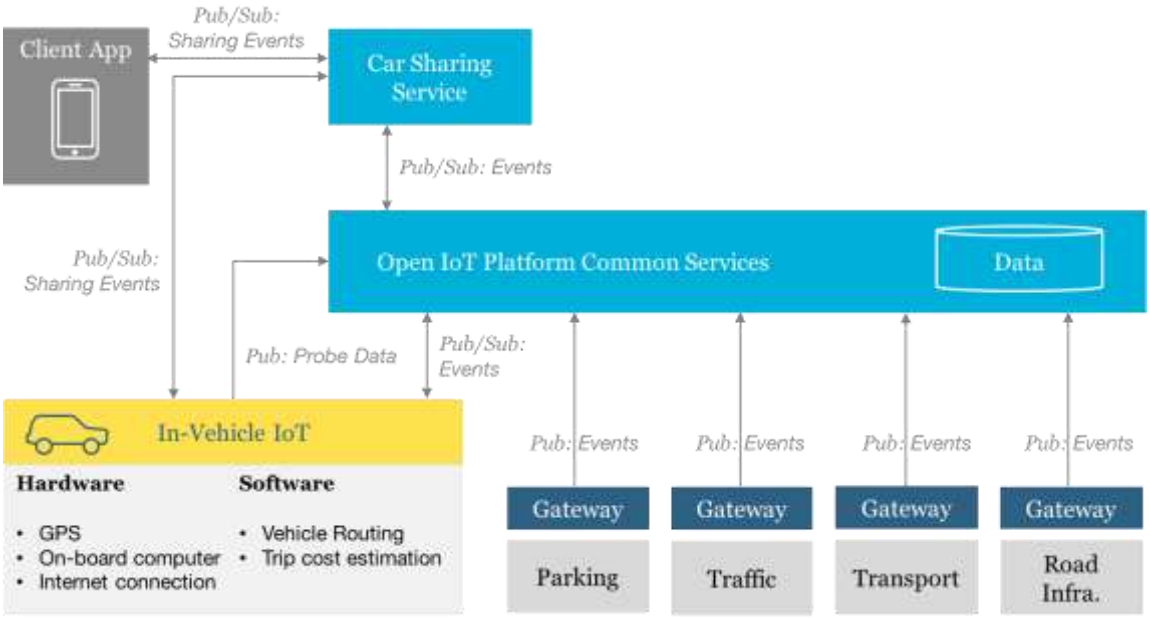

Figure 2 – Car sharing use case architecture

Interaction between the vehicle and the outside world/IoT platform is therefore needed for:

- Determination of the destination point (parking place, collect point)
- Identification when the vehicle is ready (parking place or collect point)
- Navigation to the destination, following a route either determined by the vehicle or the IoT platform, while avoiding obstacles detected by either the vehicle sensors or the IoT platform
- Transmission of observation data from IoT sensors or of events detected by IoT sensors (e.g. pedestrians or other objects on the parking place) to the vehicle
- If the driver requests the vehicle to the control point, the IoT platform has to validate the request, and if the request is valid, start the collecting process
- At the collect point, the vehicle has to validate that access to the vehicle is provided to the authorized driver

2.3 AUTOPILOT vehicle concept

This section illustrates the logical placement of the in-vehicle IoT platform within the on-board system. Figure 3 shows how the in-vehicle IoT platform has to be set up in order to receive information from in-vehicle components and to connect with the cloud IoT system. It also shows how the existing connectivity to third party cloud services (e.g. OEM specific connectivity services) and to vehicular ad hoc networks (e.g. ETSI ITS G5) have to be taken into account when defining the

in-vehicle IoT platform.

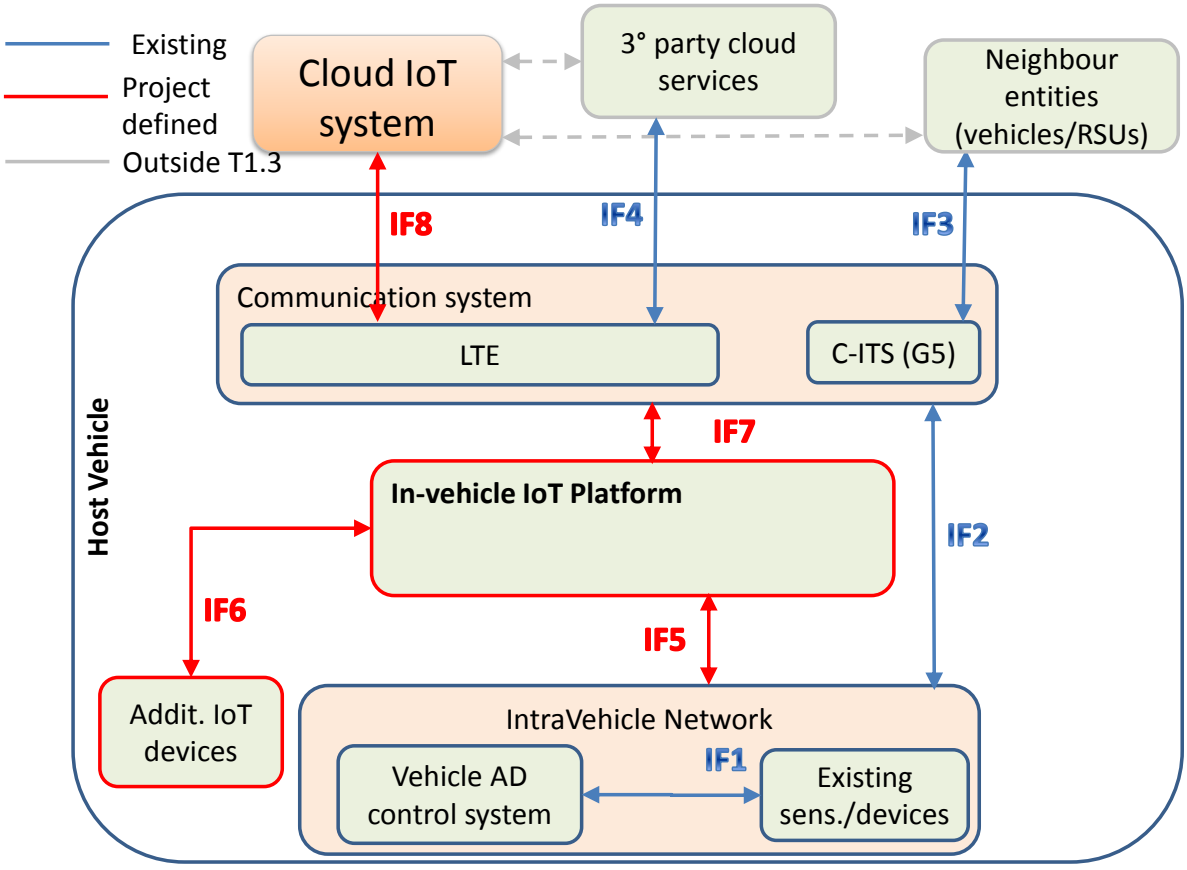

Figure 3 – In-vehicle IoT platform (red box) with the vehicle concept scheme

Table 1 outlines at high level the interfaces among components, i.e. what kind of data are exchanged, in order to show which interfaces have to be defined by AUTOPILOT.

Table 1 – Interface between main components - high level description

ID	Existing / New	Component 1	Component 2	Data / short desc.: 1 to 2	Data / short desc.: 2 to 1
IF1	Existing	Existing Sens./Dev.	Control	OEM defined and proprietary/ *enable existing AD functions*	OEM defined and proprietary */ enabling sensor functionalities (if needed)*
IF2	Existing	Intra vehicle network	ITS G5/ Cellular unit	In-vehicle data (as in ETSI G5)/ *enable C-ITS use cases*	Dispatching internally received ITS-G5 data, generated by other entities
IF3	Existing	ITS G5	Other entities	ETSI ITS G5: CAM, DENM, SPaT, MAP */enable C-ITS use cases*	ETSI ITS-G5 data generated by other entities
IF4	Existing	Cellular / (smartphone / embedded)	3rd party cloud service	Car data to cloud services */ Infotainment, connected apps, connected sensors*	Cloud data to car applications */ data provided by remote services*
IF5	New	Intra vehicle network	Vehicle IoT platform	Car data for IoT */ IoT platform gets vehicle parameters, which define vehicle as moving object*	IoT data for Intra-vehicle network / *Intra vehicle network should include IoT data into data fusion internal process*

ID	Existing / New	Component 1	Component 2	Data / short desc.: 1 to 2	Data / short desc.: 2 to 1
IF6	New	Additional IoT sensors	Vehicle IoT platform	Sensor data to IoT application in vehicle / *information from IoT objects within the vehicle*	In-vehicle IoT PF data to additionalsensors / *enabling sensor functionalities (if needed)*
IF7	New	Vehicle IoT platfform	Communication system	Data from vehicle IoT to be dispatched outside	Data from external IoT entities, dispatched internally
IF8	New	Vehicle communication system	Cloud IoT system	AUTOPILOT specific messages (could be the same as IF3-IF4) / *vehicle as moving object in the general IoT*	AUTOPILOT specific messages / *related with AUTOPILOT only*

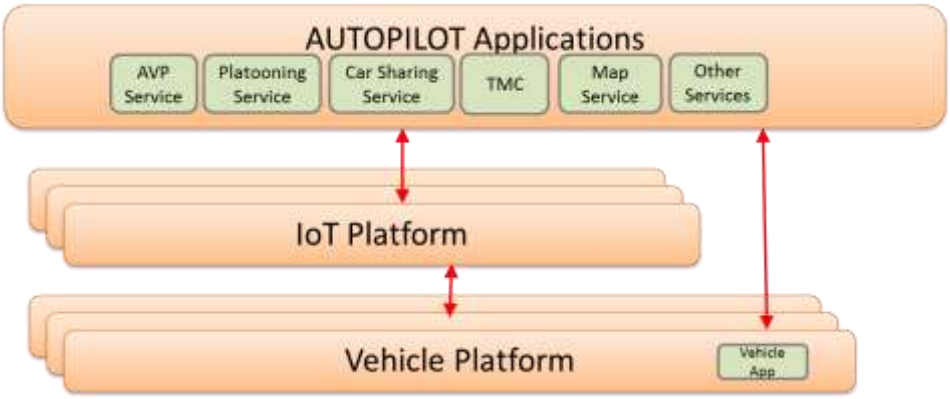

Figure 4 – IoT High View Architecture: conceptual separation in AUTOPILOT

From the IoT prospective, the vehicle is a IoT device, but also an edge computing unit and gateway for other IoT devices. The vehicle is actually a mobile node in the view of the IoT platform that publishes and subscribes (receives) content from and to the IoT Platform. The IoT platform itself is composed of central units (cloud) and distributed edge computation nodes (edge). It creates a unifying view of the IoT entities. IoT enabled applications interface the IoT Platform to interact with IoT entities, but they can also run in the vehicle platform. Here we distinguish between the IoT Platfom as the set of functions that manages the IoT devices and entities, while we define Vehicle Platform the complex entity that includes all the software and hardware components deployed in the vehicle. IoT Application can also be deployed (partially) into the vehicle. Let consider for example a parking application that either run in the vehicle application container or in some smartphone connected to the vehicle. These applications have the possibility to access services and information directly with the IoT Platform that runs in the vehicle. Some information indeed are only available and used locally. The in-vehicle application is typically then connected to the cloud counterpart to exchange service related information. A closer look into the vehicle platform is presented in Chapter 2, where the vehicle functional architecture of AUTOPILOT is presented.

2.4 Prototype descriptions

In this chapter the prototypes are described at high level, to show the operational context of the vehicles that will be involved in AUTOPILOT testing. Each partner having an AUTOPILOT prototype in plan has provided a description of the system, along with a high level logical scheme where the IoT platform is interfaced with other in-vehicle components.

2.4.1 Test site Finland prototypes

VTT will provide two automated vehicles for the AUTOPILOT:

Marilyn 2.0 is a Citroen C4 (Figure 5), which has been updated for automated driving, by installing electric actuators for control of throttle, steering wheel and brake. Marilyn 2.0 is the first passenger car in Finland, which received permission for autonomous driving in real traffic. The car is equipped with advanced sensor technology, software solutions and automated driving functions. The vehicle has been modified for automated driving.

Figure 5 – Sensors installed in the Finnish automated vehicle prototype

- Martti is a modified Volkswagen Touareg (Figure 6**Error! Reference source not found.**). On the bumper of the vehicle a sensor rack is installed, which contains similar sensors as Marilyn 2.0. Martti also has V2X communications and advanced HMI. Martti will be upgraded for automated driving by Autumn 2017, and will include a similar sensor set as Marilyn 2.0

Figure 6 – Sensor kit installed in VTT's pilot vehicle Martti

The following use cases will be demonstrated in Finland TS in Tampere:

- Urban driving: intersection support. Traffic signal data will be in real time collected from the traffic signal control system. Potential conflicts, observed by traffic cameras, will be transmitted to the vehicles
- Automated valet parking: traffic cameras installed at the parking lot will assist the vehicle in

the parking maneuver

Figure 7 shows the draft in-vehicle architecture of the prototype
The vehicle scheme is organized in the following component groups:

- Communication devices: these devices of all communication of the vehicle, including:
  - ITS-G5 interface for communication to road side units and other vehicles in the neighbourhood, using V2X messages, complying with ETSI (CAM [7], DENM [8])
  - Cellular interface for communication to IoT clouds
- Sensing devices:
  - GNSS sensor and additional sensors, such as IMU
  - Environmental perception sensors, including radar, LIDAR and cameras, as described above and shown in
  - Figure 7
  - Vehicle CAN-bus interface
- On board Units:
  - AD unit: set of devices responsible for the real-time functions, including fusion of the data of the both the different positioning sensors and of the environmental sensors for detecting and classifying objects, for threat assessment, trajectory planning and control of the automated vehicle
  - IoT in-vehicle platform, managing the IoT services:
    - Urban driving support:
      - Traffic light support (in case the message is not delivered in SPaT format)
      - Traffic information on potential disruptions
    - Automated valet parking
      - Control of the vehicle: setting destination (parking place), allowing to start, support during parking (e.g. improved location)
- AD/Units outputs:
  - Actuators: components that act on the commands determined by the AD Unit to control basic vehicle functions such as steering, braking and acceleration
  - HMI: user interface for the driver

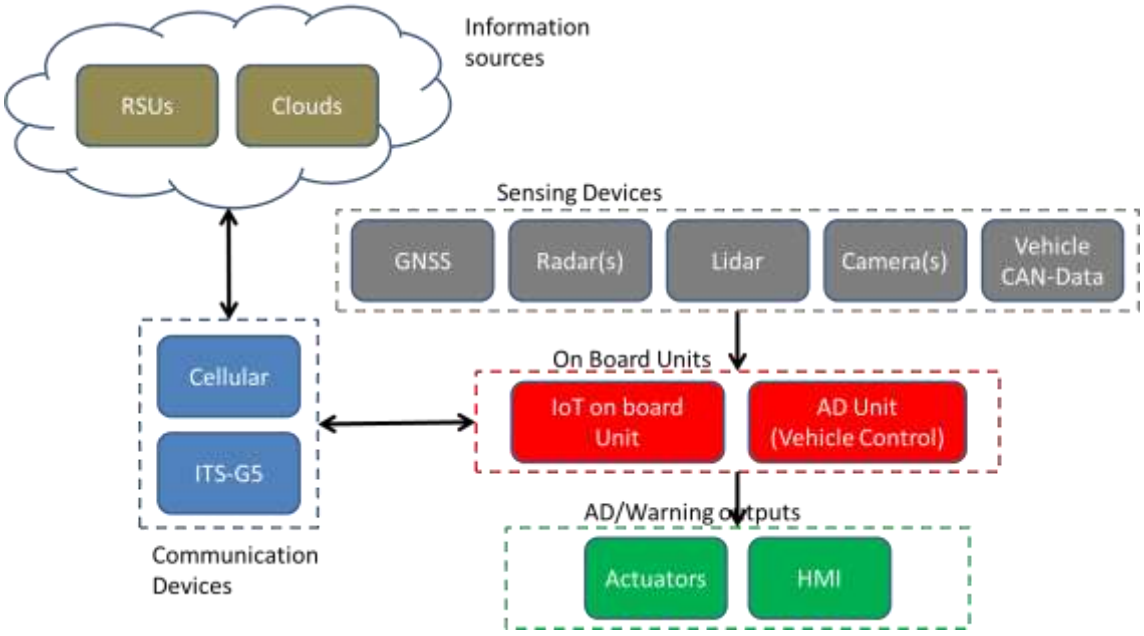

Figure 7 – In-vehicle architecture of the Finnish prototypes

2.4.2 Test site France prototypes

VEDECOM is going to provide a fleet of 10 to 15 vehicles (VFLEX prototypes) for the French pilot site. A VFLEX is a L4 prototype made from a Renault Twizy which has been modified in order to perform autonomous driving uses cases.

Figure 8 – Image of a French prototype

First, thanks to VEDECOM's collaboration with Renault, the Twizy has been transformed in an open robotic platform (with an open access to the CAN bus of the vehicle).

On top of that, the VFLEX is equipped with:

- Two CONTINENTAL radar ARS 408 (one at the front and one at the back)
- One perception camera provided by TU Eindhoven
- One VALEO ultrasonic belt
- One VLP 16 Velodyne lidar on the roof
- One SBG inertial sensor

The in-vehicle IoT platform (hardware and software) is guaranteed by CEA.

Figure 9 – Overview of VEDECOM prototype sensors

The basic functions such as steering, breaking, throttling as well as the automation system (with lidar/radar perception), the handling of vulnerable road users as well as the generation of the trajectory are managed by VEDECOM.
The software of the perception camera (obstacles detection and SLAM) is going to be provided by TU Eindhoven.

The VFLEX fleet will be able to perform the following uses cases on the French pilot site:

- Connected urban driving with PoI notifications
- Full autonomous driving in a controlled environment with:
  - VRU detection
  - PoI notifications
- Urban automated fleet rebalancing (on public roads):
  - Platooning
  - Traffic light support
  - VRU detection
  - Parking maneuvers

In terms of software architecture, the following partners are involved on the VLEX platform:

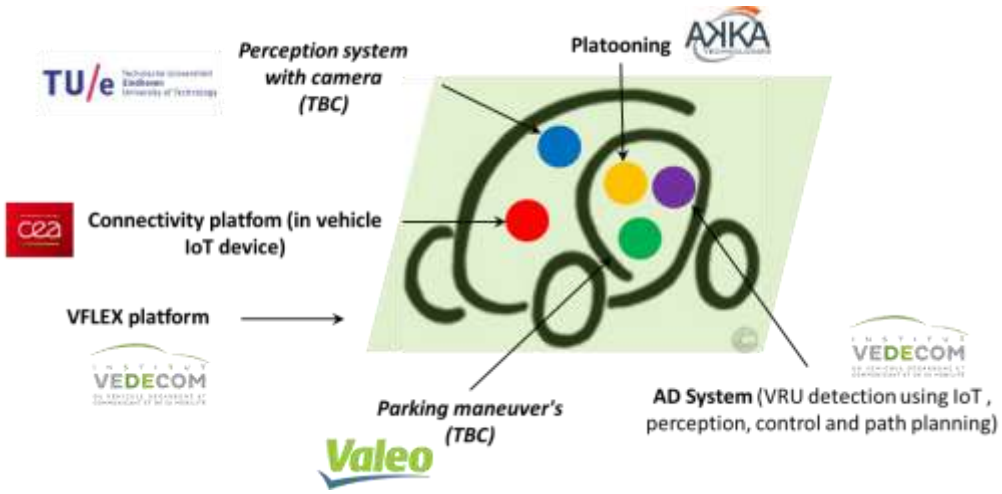

Figure 10 – Overview of synergy for French prototypes

Functional architecture

The main functions selected for the French prototypes in autonomous driving mode are:

- Follow a predefined itinerary
- Track and keep the driving lane
- Respect the road rules (stop at red light, stay below maximum speed, stop at stop sign, etc.)
- Avoid obstacles (adapt speed, stop or change lane)

The main functions in platooning mode are:

- Change direction by following the vehicle in front
- Stay at close distance to the vehicle in front

The main functions in parking maneuvers mode are:

- Exit a parking spot
- Position behind a predefined vehicle
- Park in a specific parking spot

In order to achieve these functions, the system shall have the following technical functions:

- Detect, qualify and filter obstacles (using direct or collaborative perception)
- Identify road rules (speed limit, traffic light state, stop signs, etc.)
- See road surface markings
- Locate the vehicle position
- Save a reference itinerary
- Start the vehicle (manually or wirelessly)
- Switch between the 4 driving modes: manual, autonomous, parking maneuvers, platooning
- Change the vehicle direction
- Start the vehicle movement
- Accelerate the vehicle
- Regulate the vehicle movement at a constant speed
- Decelerate the vehicle (braking)
- Stop the vehicle
- Shut down the vehicle and activate the parking brake
- Give light signals (braking light, turn signals, horn, etc.)
- Acquire driver's commands and inform the driver (HMI)
- Communicate with the environment (with traffic lights, road users, etc.)

- Communicate with other vehicles

A global diagram of the French prototypes is presented in Figure 11.

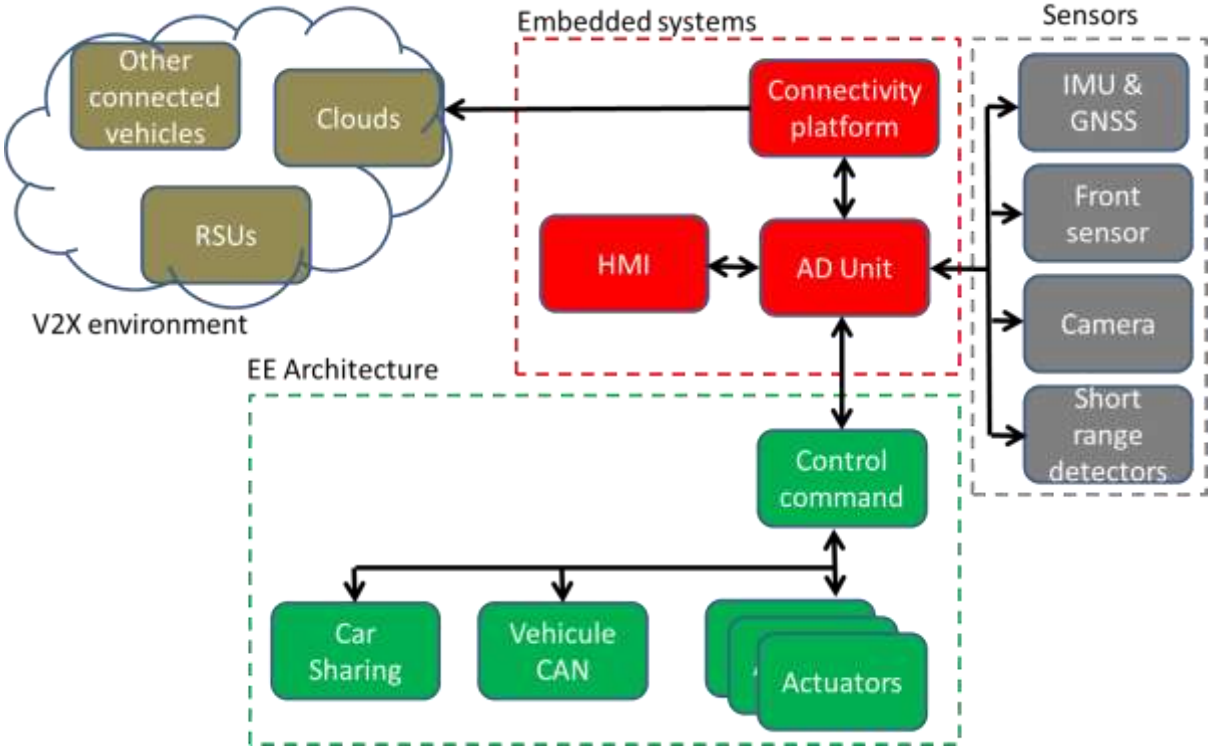

Figure 11 – High level scheme of French prototypes

2.4.3 Test site Italy prototypes

In the Italian test site will be present two kinds of prototypes:

- Connected prototype (provided by CRF/FCA and AVR):
  It is equipped with a communication platform with both ETSI ITS G5 [9] and LTE connectivity. It is also equipped with sensors to get information from the environment and with on board unit that elaborates the collected information to give warning notification to the driver or to broadcast aggregated information to the other road users.

- AD & Connected prototype (provided by CRF/FCA):
  It is equipped with the same kind of device of the Connected prototype with in addition some modules to enable autonomous driving functionalities. The outputs of these devices permit the control of the actuators (breaking and steering system, adaptive cruise control) managing the vehicle behavior.

Figure 12 – Jeep Renegade, vehicle model used by CRF in Italian Test Site (source Jeep official site)

A general scheme of the functional components that can be installed in a prototype for the Italian Test site it is shown in Figure 13:

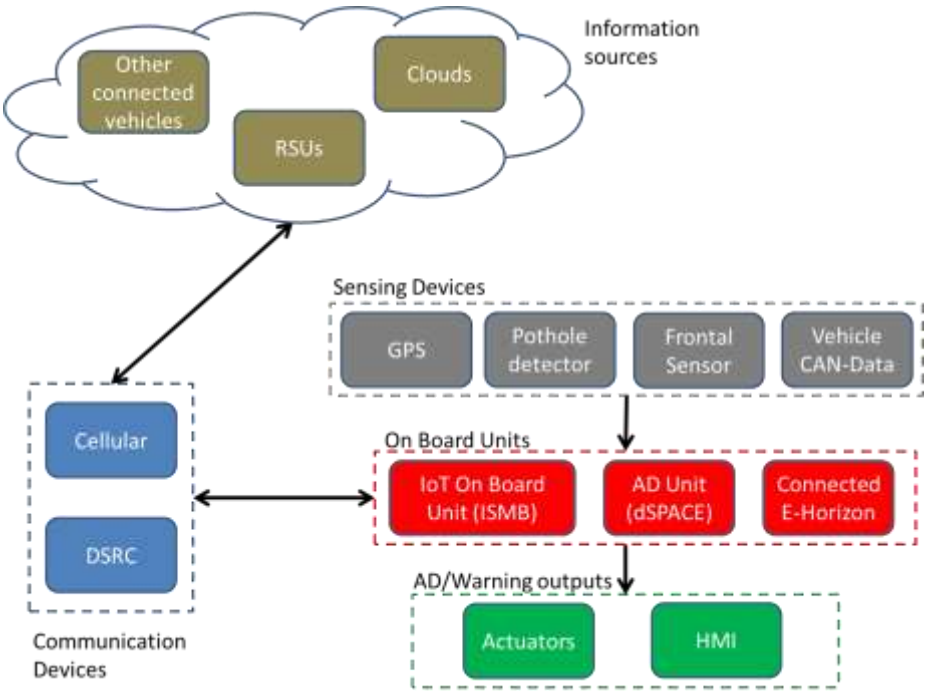

Figure 13 – General scheme of Italian Test Site vehicles

- Communication devices
  - Cellular: Cellular LTE interface to Tx/Rx information from the cloud
  - DSRC: Wireless communication module based on ITS-G5 technology to communicate with vehicles in the neighborhood and with the road infrastructure
- Sensing devices
  - GPS: It estimates the position of the vehicle (latitude, longitude, heading, speed etc.)
  - Pothole detector: it detects the presence of pothole in the street during the vehicle motion using a dedicated sensor and possibly with additional information coming from the in-vehicle native sensors
  - Frontal sensor: It is a radar and/or camera able to detect obstacle placed in front of the vehicle

- Vehicle CAN data: it provides data from sensing devices installed in the normal production vehicle as odometers, accelerometers, information on the vehicle state, etc

- On board Units
  - IoT On Board Unit: It manages the bidirectional DSRC communication. It encodes/decodes V2X messages (CAM, DENM, SPAT, MAP). It receives information from the cloud, from connected vehicles/road infrastructure and from sensing devices. It aggregates this IoT information and shares it
  - E-Horizon: It matches the vehicle position to an "HD Map" (High Definition Map) and it estimates the most probable path as well as alternatives paths that the vehicle will follow. It manages dynamic events as the variable speed limits
  - AD Unit It is the main component of the Autonomous Driving system. The logic for the decision making runs inside this device and its output controls the prototype actuators
- AD outputs
  - Actuators: these components transform digital output of the board in action. In particular, the steering system, the braking system and the adaptive cruise control (ACC) system permits the automated vehicle control
  - HMI: it receives data from on board units and shows to the driver warning notification in case of dangerous scenarios

Prototypes fleet

As a whole, a fleet of 7 prototypes is planned for Test Site Italy: 2 CRF connected vehicles, 3 AVR connected vehicles, 2 CRF AD & connected vehicles.

Three kinds of vehicles are represented, equipped with subsets of the functional components above, as the pictures hereafter show.

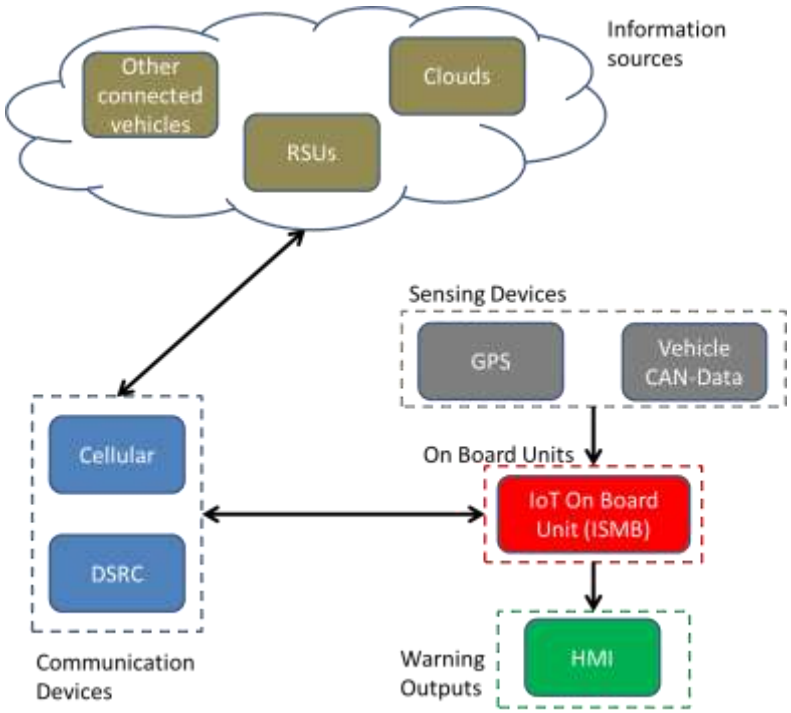

Figure 14 – Scheme of the CRF connected vehicles

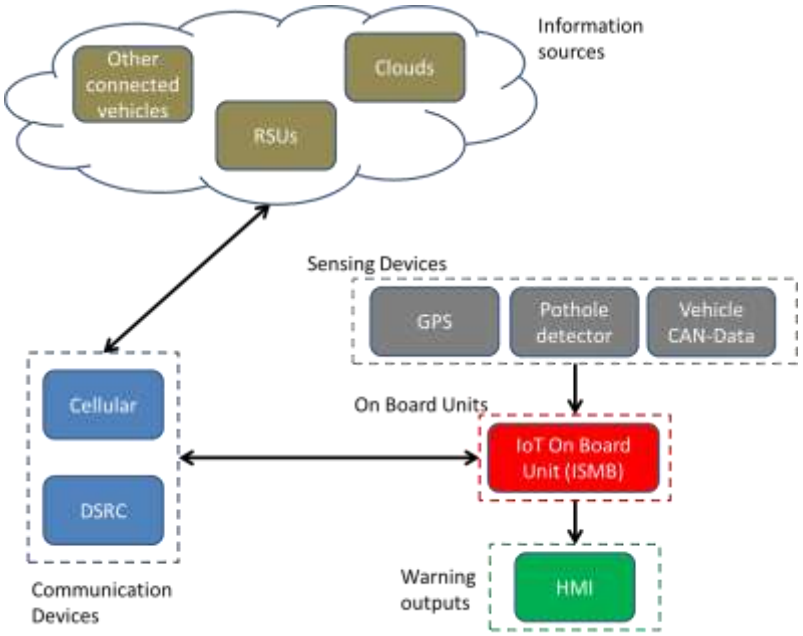

Figure 15 – Scheme of the AVR connected vehicles

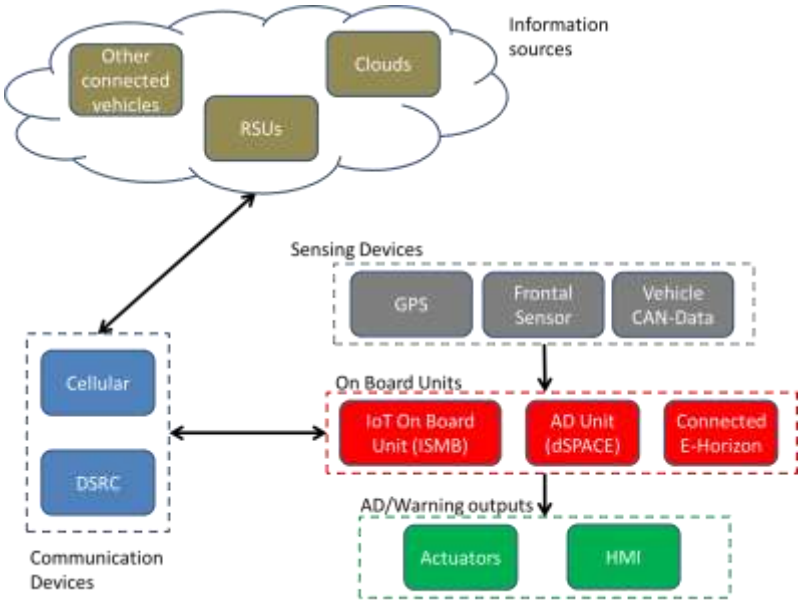

Figure 16 – Scheme of the 2 AD & Connected vehicles

Communication interfaces of AD & connected vehicles in the Italian Test site are shown in Figure 17.

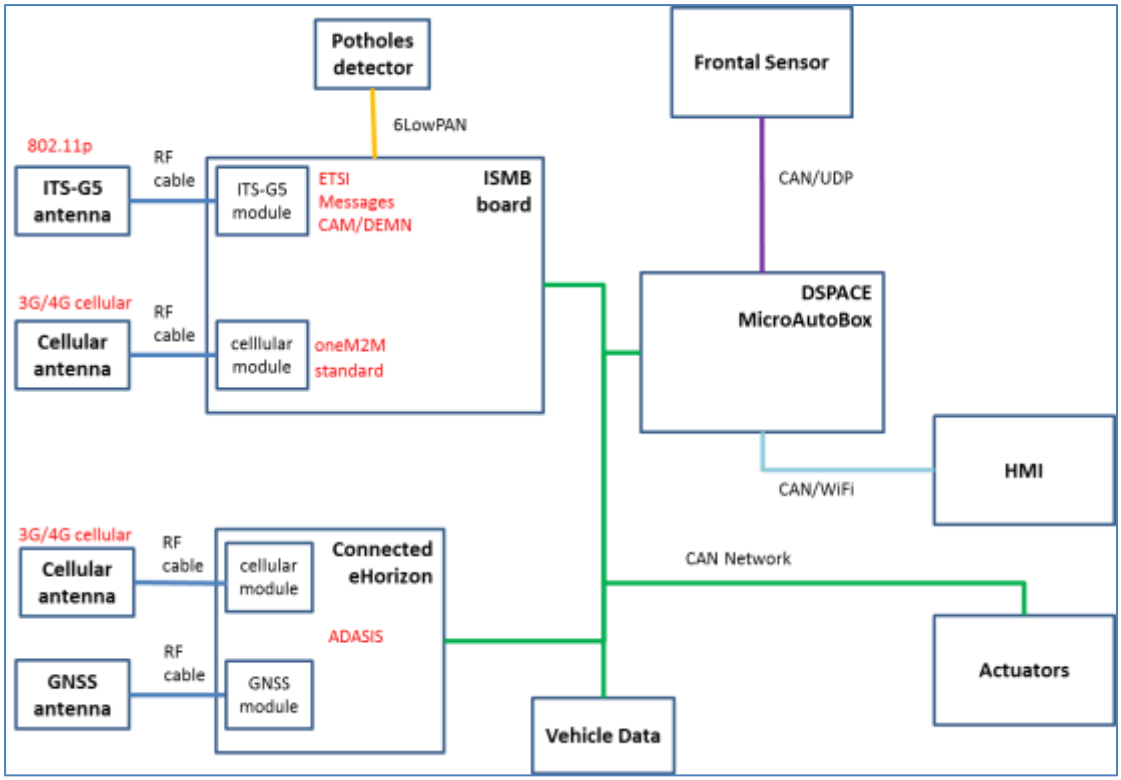

Figure 17 – Components and interfaces of Italian Test Site AD & connected prototypes.

2.4.4 Test site Netherlands prototypes

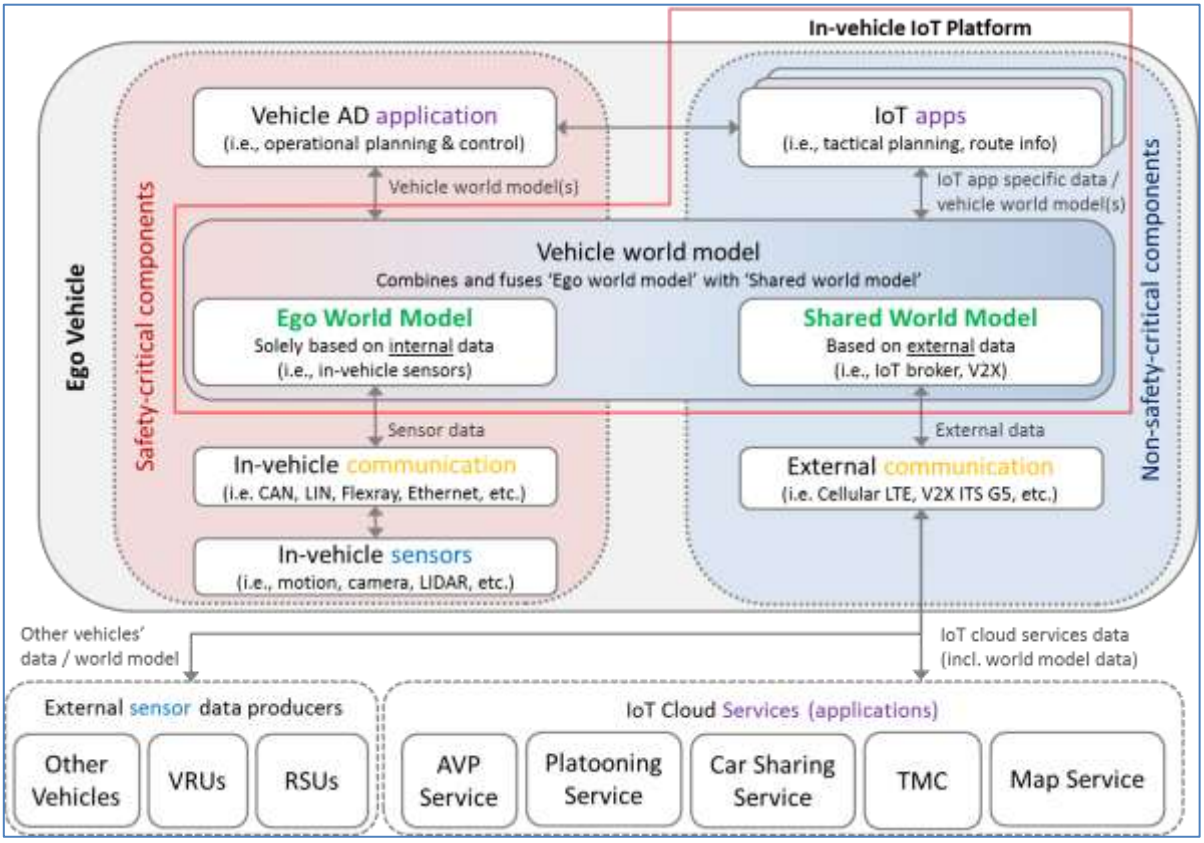

Figure 18 – In-vehicle high-level architecture for Brainport pilot site

Figure 18 shows the high-level architecture for the Brainport test site in The Netherlands. It includes concepts and components that are common to the prototypes in the test site. The architecture conceptually separates safety-critical and non-safety-critical components to allow critical components in the vehicle to work at all times even in the event of communication loss with external entities (e.g., IoT services or V2X communication).

Safety-critical components address functionalities in the operational level of the vehicle, thereby requiring high reliability and timely updates. Vehicle AD (Autonomous Driving) application contains the operational vehicle planning and control that will be specific to each vehicle provider and use case. In-vehicle sensors are connected to the system via in-vehicle communication (e.g., CAN bus, Ethernet) and will also be vehicle provider specific. Examples of internal sensors are cameras, radars, motion sensors, etc.

Non-safety-critical components address functionalities in the tactical and strategic levels of the vehicle, thereby posing less strict time and reliability requirements. IoT apps will consume and process data coming from external entities such as IoT Cloud services, other vehicles in the surroundings (V2X), Roadside units (RSUs) and Vulnerable Road Users (VRUs) via external communication links such as Cellular LTE or V2X ITS G5 communication.

The 'Vehicle world model' component creates a high-level view of the surroundings that can be used by either the Vehicle AD application or IoT apps in both safety-critical and non-safety critical levels. This component will combine and fuse data coming both from: (i) in-vehicle sensors to create the 'Ego world model', and (ii) external entities such as IoT cloud services via the IoT broker, roadside units or other vehicles (V2X) to create the 'Shared world model'. Depending on applications' requirements, the output is one or more application specific vehicle world models. Each world

model provides high-level description of objects (e.g., shape of cars, pedestrians), road/lane (e.g., road shape) and optional semantics information (e.g., classification of objects). This allows, for example, operational path planning algorithms to make the best decision at a certain point in time based on an extended view of the environment that is currently relevant to the ego vehicle.

In the event of communication loss with the external entities, the vehicle world model component will rely solely on the in-vehicle sensor data to create the world model that would correspond in this case to the 'ego world model'. In this manner, the overall system benefits from external data when available to extend its range of awareness and yet it remains robust against communication failure.

In the project, the 'in-vehicle IoT platform' comprises the 'Vehicle world model' and IoT apps which together build the bridge from the in-vehicle system to the external IoT world. The high-level architecture above intentionally hides components that might be vehicle provider specific, such as high- and low-level controllers that can be specified in different ways within the 'Vehicle AD application'. Also, the use of a standardized IoT broker such as the oneM2M platform is conceptually grouped into the 'Shared world model' sub-component that is gathering data from external entities.

TomTom contributes in Autopilot to the development of a localization sub-system and a streaming service for map delivery. The intention is to integrate those development in the test vehicles being built up for the Brainport platooning- and highway pilots. This service will be used in the different brain-port use-cases to access the map.
TomTom will also build up its own car for validation purposes as shown in Figure 19 below. This car will be based on a Mobile Mapping VAN of TomTom with high cost sensors used to create ground, extended with a localization sub-system that connects to the HD-map delivery service and to different sensors such as Lidar, Camera and Radar sensors.
This car will also integrate a sub-system demonstrating lane-navigation functionality based on the localization output also considering dynamic information received from the IoT system via cellular or V2X connectivity.

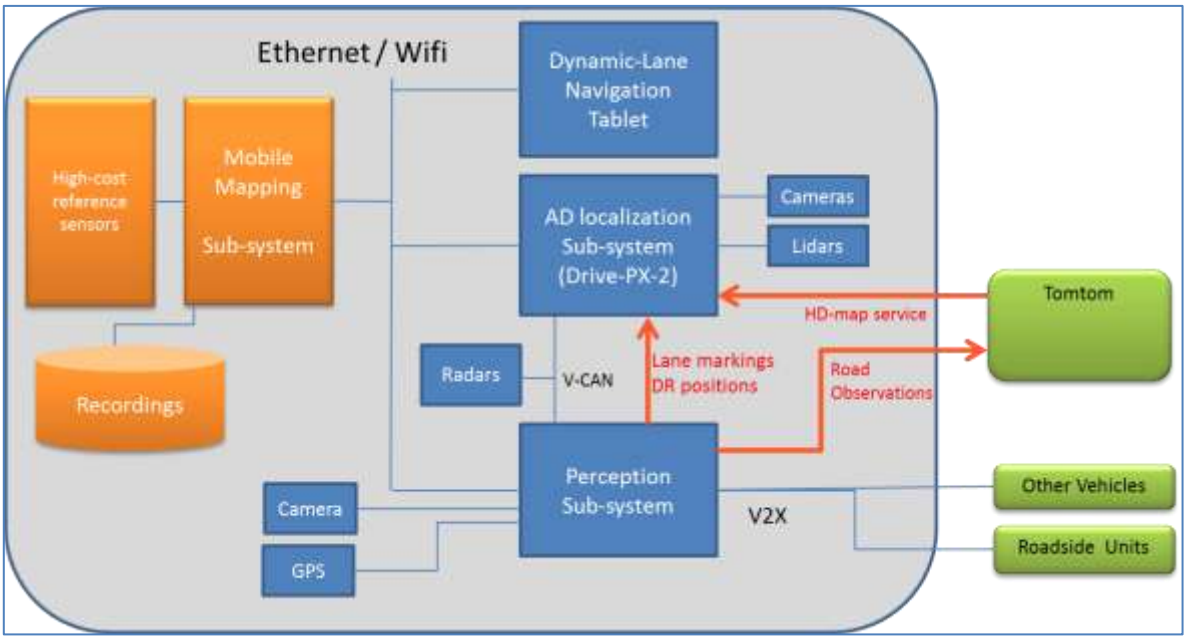

Figure 19 - TomTom prototype for validation purpose, high-level architecture.

2.4.4.1 TNO prototype

Figure 20 – TNO prototype vehicle (source: TNO website [10])

TNO will use a modified Toyota Prius equipped with additional sensors to support automated driving functions (shown in Figure 20). The general vehicle scheme of hardware components is shown in Figure 21.

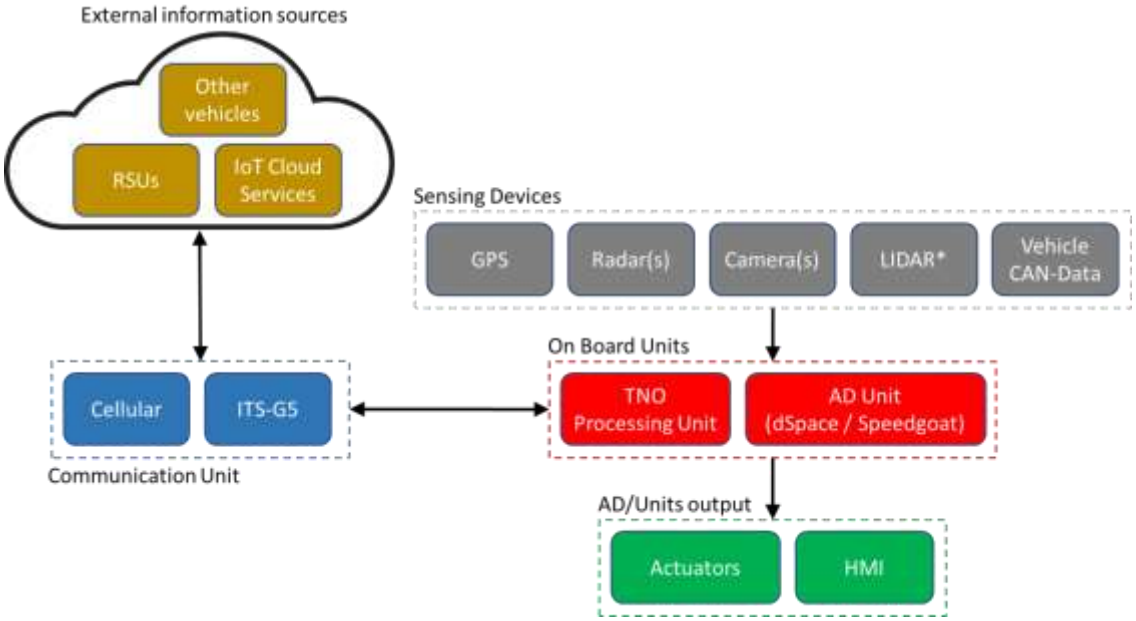

Figure 21 – TNO vehicle scheme

The vehicle scheme is organized in the following component groups:

- Communication unit: device that processes network data coming from two interfaces:
  - Cellular: Cellular interface LTE(-V2X) to Tx/Rx information from IoT cloud services
  - ITS-G5: Wireless communication module based on ITS-G5 technology to communicate with vehicles in the neighborhood and with the road infrastructure (roadside units)
- Sensing devices:
  - GPS module: it estimates the position of the vehicle (latitude, longitude, heading, speed). RTK-GPS might be used to enhance the positioning precision
  - Radar(s): Radars generate point cloud data of the environment that can be used for tasks such as localization and object detection
  - LIDAR*: LIDAR will be optionally used to generate precise point cloud data of the environment that can be used for tasks such as localization
  - Camera(s): (Stereo-)camera(s) are used to get images of the environment that can be used for tasks such as object classification and scene understanding
  - Vehicle CAN data: it provides data from sensing devices installed in the normal

production vehicle as odometers, accelerometers, information on the vehicle state, etc

- On board Units:
  - TNO processing unit: it processes and fuses data coming from sensors and from the communication unit to create 'world model' data that it is needed internally by vehicle components such as path planner. Also, it shares world model data with external entities via the communication unit
  - AD Unit: it is the component responsible for control related functions that will send commands to actuators in the vehicle
- AD/Units outputs:
  - Actuators: components that act on the commands determined by the AD Unit to control basic vehicle functions such as steering, braking and acceleration
  - HMI: shows data currently being processed in the on board units

2.4.4.2 NEVS prototype

The NEVS prototype is an electric vehicle (EV) platform based on a passenger car (D-class) chassis. The vehicle provides control interfaces to the steering, propulsion and brake mechanisms to allow realization of various AD functionalities. The specific use-cases within the project scope that will involve the NEVS platform are defined as the AVP, Platooning and Car-Sharing. Due to the variety of the possible requirements of these use-cases the control interfaces are left accessible through a prototyping environment (i.e. dSpace MABX). Also, although set for current development efforts, the specifications of the acceptable control inputs are in a flexible range for adapting to the aforementioned use-cases. The platform can provide access to interior sensor readings regarding vehicle dynamic states, e.g. rotational or translational accelerations, steering angle, brake pressures, wheel speeds, etc.

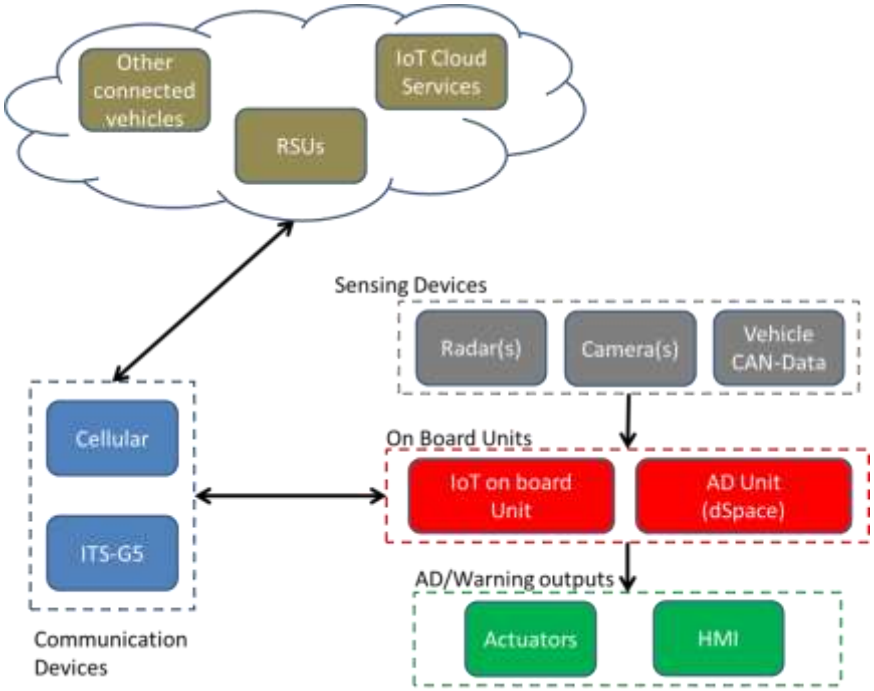

Figure 22 – NEVS vehicle scheme

In addition, a set of environmental sensors has been used for NEVS's internal development. It is possible to use an identical set of sensors in this vehicle based on the use-case requirements and partner agreement.

The list of sensors includes a front-looking multifunctional mono-camera and dual-scanner radar units. All units are capable of communicating over the vehicle's CAN bus, the radar and multifunctional camera are sending processed feedback (i.e. object information). It is also possible to equip the vehicle with surround-view cameras that can be utilized for image processing applications and be implemented in a specific use-case scenario.

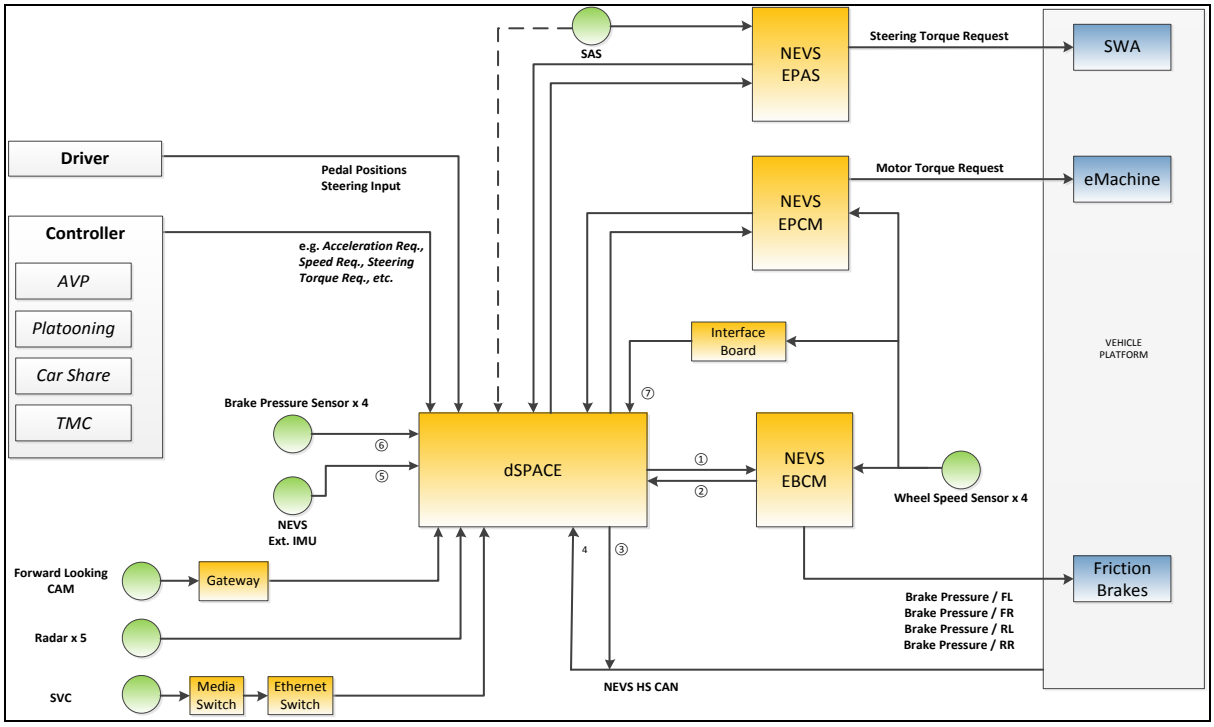

Figure 23 – NEVS vehicle scheme

2.4.4.3 TUEIN prototype

TU/e will start with using two different vehicles: a Toyota Prius and a VEDECOM vehicle for implementation and execution in the Brainport TU/e Car Rebalancing (Urban Driving) use case.
Both vehicles have a basic Automated Driving functionality which can be controlled from a real time target (AD unit), but differ partially in sensory input.

For the final use case pilot testing, TU/e plans to equip 3 VEDECOM vehicles in total to demonstrate the rebalancing variant of the platooning use case on TU/e campus.

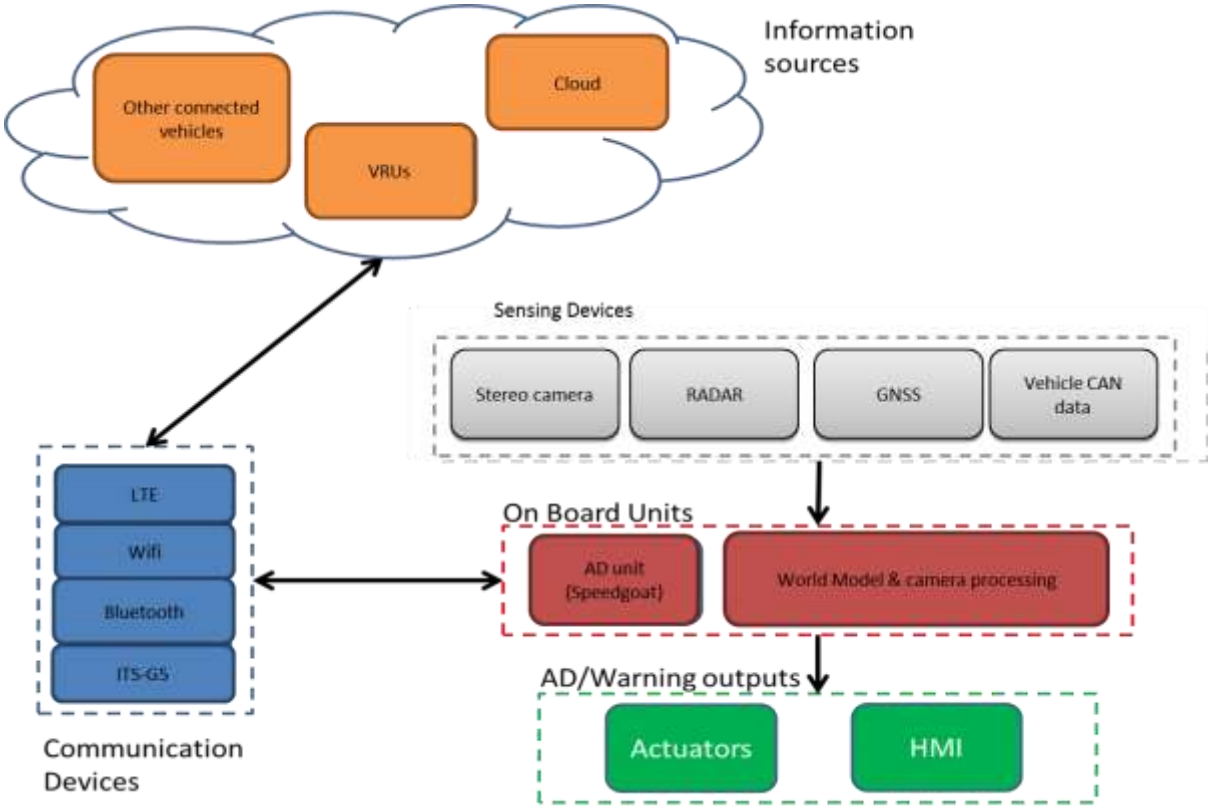

Figure 24 – Scheme of the 1 TU/e AD vehicle: Prius

The overall vehicle scheme is organized in the following component groups:

- Communication unit: device that processes network data coming from two interfaces:
  - Cellular: Cellular interface LTE(-V2X) to Tx/Rx information from IoT cloud services and VRU detection application
  - ITS-G5: Wireless communication module based on ITS-G5 technology to communicate with vehicles in the neighborhood and with the road infrastructure (roadside units)
- Sensing devices:
  - GPS module: It estimates the position of the vehicle (latitude, longitude, heading, speed). RTK-GPS might be used to enhance the positioning precision
  - Radar: Off-factory installed front facing radar generating x- and y-position including range, rates of up to 8 objects that can be used for tasks such as localization and object detection
  - Camera(s): (Stereo-)camera(s) are used to get images of the environment that can be used for tasks such as object classification and scene understanding and localization
  - Vehicle CAN data: It provides data from sensing devices installed in the normal production vehicle as odometers, accelerometers, information on the vehicle state, etc.
- On board Units:
  - World model & camera processing: It processes and fuses data coming from sensors and from the communication unit to create 'world model' data that it is needed internally by vehicle components such as path planner. Also, it shares world model data with external entities via the communication unit
  - AD Unit: It is the component responsible for control related functions that will send commands to actuators in the vehicle
- AD/Units outputs:

- Actuators: components that act on the commands determined by the AD Unit to control basic vehicle functions such as steering, braking and acceleration
- HMI: shows data currently being processed in the on board units

For the first implementation phase the Prius is used, similar to the Prius platform used by TNO (see 1.3.4.1).

The current hardware architecture implementation for the TU/e prototype Prius is shown in Figure 26.

Figure 25 – TU/e prototype vehicle

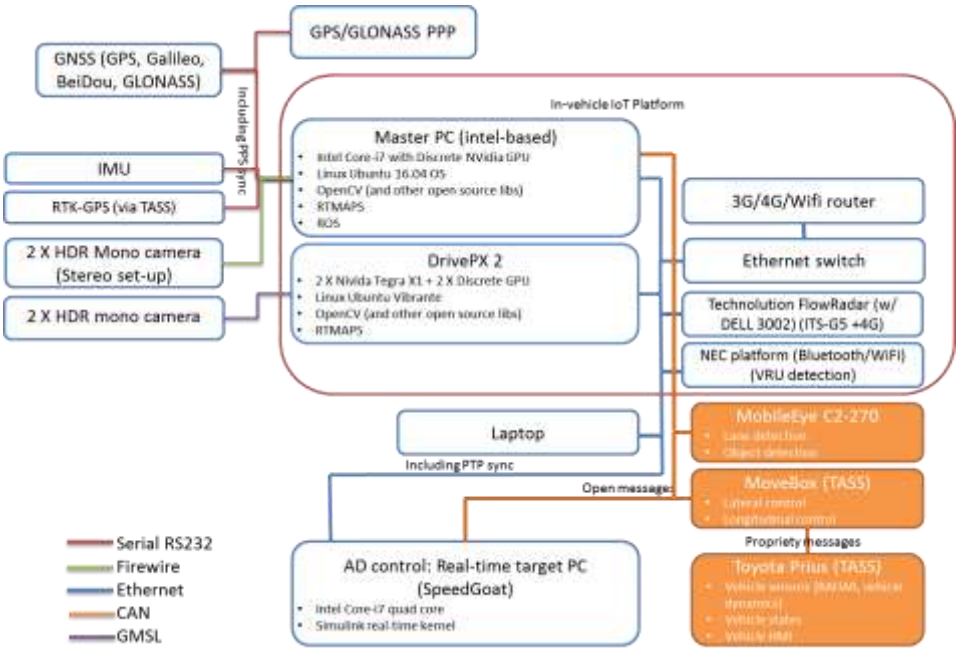

Figure 26 – Current planned TU/e prototype vehicle architecture, possibly extra multiple cameras to be added for VRU detection & mapping.

For the second implementation phase the VEDECOM vehicle is used as described in paragraph 1.3.2.

2.4.4.4 VALEO prototype

The Valeo prototype, to be used at the Brainport pilot site, will be a Volkswagen Tiguan 2.

Figure 27 – Valeo prototype vehicle

This car comes equipped with an Adaptative Cruise Control (ACC) and a Side Assist for Traffic Jam [11]. This specific model was selected because of its generic MQB platform [12] developed by Volkswagen, which Valeo has experience in upgrading to AD level.

This Tiguan will be prepared with multiple and diverse additional sensors and software units needed to conduct the obstacles detection and the road surface scanning that power the use case.

The simplified functional highview below illustrates the main components of the setup. Sensors data capture will be handled only through the Valeo Smart Platform, which behaves as an In-vehicle IoT Platform. Similarly all additional components (software or hardware) will be plugged into the Valeo Smart Platform and communicate between each other through it.

It is worth pinpointing that the ACC/AD control system of the car preserves its own functions and its own direct links (not represented here) with sensors and actuators, without critically relying on the features enabled via the Valeo Smart Platform.

Hence the developments supporting the use case can be considered as an add-on module, whose information will serve as complementary aid to the functionality of the ACC/AD control system. This aid will be mediated by a secure agent that will pass on suggested new speed limit, increased safety distance, etc.

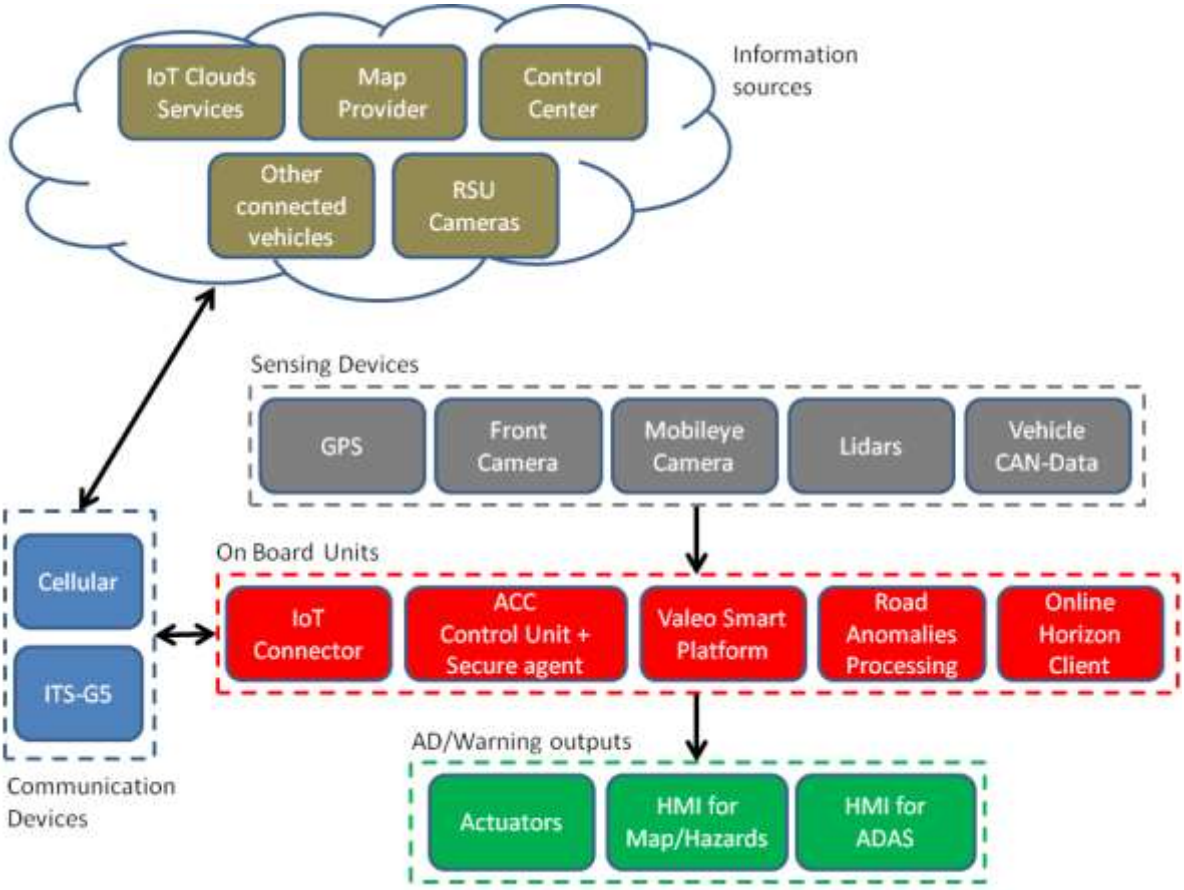

Figure 28 – Valeo prototype, functional view.

The functional blocks can be further described:

- On Board Units
  - ACC Control Unit
    - VW system featuring Lane detection, Trajectory, Pedestrian detection, Mission planning, Object detection, Traffic signs detection
    - An additional secure agent will mediate extra external information to these
  - IoT Connector: as provided by the project partners, a solution to enable compatibility with OneM2M platform
  - Road anomalies processing
    - Integration of 3rd party video and lidar processing algorithms tuned for our use case
    - Also real-time module that will perform the road scanning and run algorithms for the discovery of pattern anomalies in other sensors data
  - Online Horizon Client:
    - Retrieving and caching solution for HDmaps and Live maps that will carry Road Hazards location, description as well as ADAS instructions
  - Reporting module (not illustrated)
    - A service that will package relevant sensors data for upload whenever peculiar events are observed (which may be later be analyzed and confirmed as actual hazards)
    - A service that will manage broadcasting alerts in case of critical/serious obstacles observation
  - Valeo Smart Platform:
    - A MQTT broker that handles subscriptions and messages across all the

components
    - Note that actual video and lidar streams will be controlled via messages through this broker but carried through direct links not appearing on the diagram
- Sensing Devices:
    - Those listed are picked from Valeo's portfolio
    - More sensors may be added during the Project
- Communication Devices
    - The telematics module, also manufactured by Valeo, will handle both ITS-G5 communication (for time critical messages) and 4G cellular communication (most of the time, in particular for data upload)
- AD/Warning outputs
    - HMI for Map/Hazards: displaying SD and HD navigation, along with relevant hazards locations and descriptions
    - HMI for ADAS: displaying specifically the cars intents upon processing the AD strategy results after reception of ADAS instructions
    - Actuators: components controlling steering, speed, braking

2.4.5 Test site Spain prototypes

Spanish Test Site will provide 3 automated vehicles for AUTOPILOT: PSA will contribute with 2 vehicles and CTAG will contribute with 1 vehicle (PSA branded).

All the vehicles will be delivered with an IoT in-vehicle platform and the necessary automatic driving functions to support urban driving and valet parking use cases as described in sections 2.2.4 and 2.2.1.

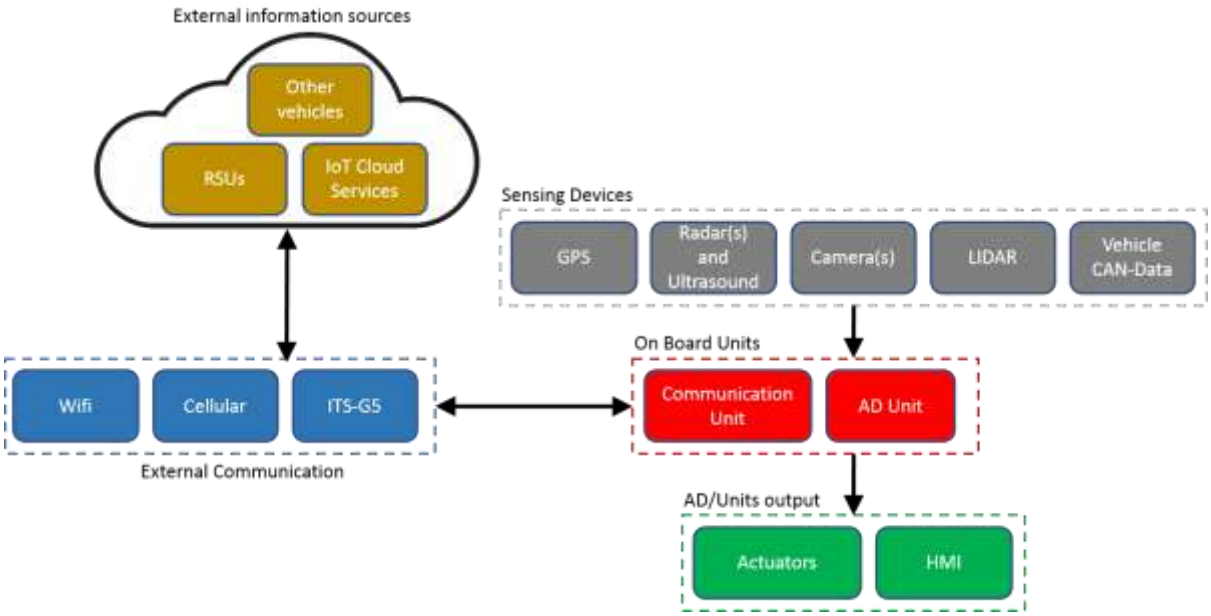

Figure 29 - Spain test site prototype architecture

As an example, the Spanish architecture is described in Figure 29. The main components and interfaces are described in Figure 30**Error! Reference source not found.** where the proposed components are described as follows:

- External communication:
    - Cellular: Cellular LTE interface to Tx/Rx information from cloud

  - ITS-G5: Wireless communication module based on ITS-G5 technology to communicate with vehicles in the neighborhood and with the road infrastructure
  - Wi-Fi: Wi-Fi interface to Tx/Rx information from the cloud
- Sensing devices
  - GPS: Estimates the position of the vehicle (latitude, longitude, heading, speed etc.) RTK-GPS might be used to enhance the positioning precision
  - Radar and ultrasound: Generate data of the environment that can be used for tasks such as localization (parking maneuver) and object detection
  - LIDAR: Will be used to generate precise point cloud data of the environment that can be used for both positioning and object detection. This sensor is essential for the indoor parking positioning when there is no GPS signal
  - Camera(s): Are used to get images of the environment that can be used for tasks such as VRU or obstacle detection. They are also used for the positioning in Urban Driving, providing images of lane markings
  - Vehicle CAN data: It provides data from sensing devices installed in the normal production vehicle as odometers, accelerometers, information on the vehicle state, etc
- On board Units
  - Communication unit: Is the component responsible for the V2X communication using ETSI ITS G5 [9] bidirectional communication, cellular communication or any other that could be necessary during the project. This unit is also responsible to transform the vehicle in a "Thing" within the IoT environment with the capability of being a data provider or a data consumer. Using the communication unit, the vehicle can communicate directly to other "Things" in its environment, as it does using V2X communication
  - AD Unit: It is the component responsible for control related functions that will send commands to actuators in the vehicle. This platform includes the different perception algorithms (positioning, road description and object fusion), and the function algorithms (route planning, state machine, motion planning and control)
- AD outputs
  - Actuators: Are comprised of engine, break and steering wheel
  - HMI: Receives data from different units and displays useful information to the driver in different places like head display, instrument cluster or head unit.

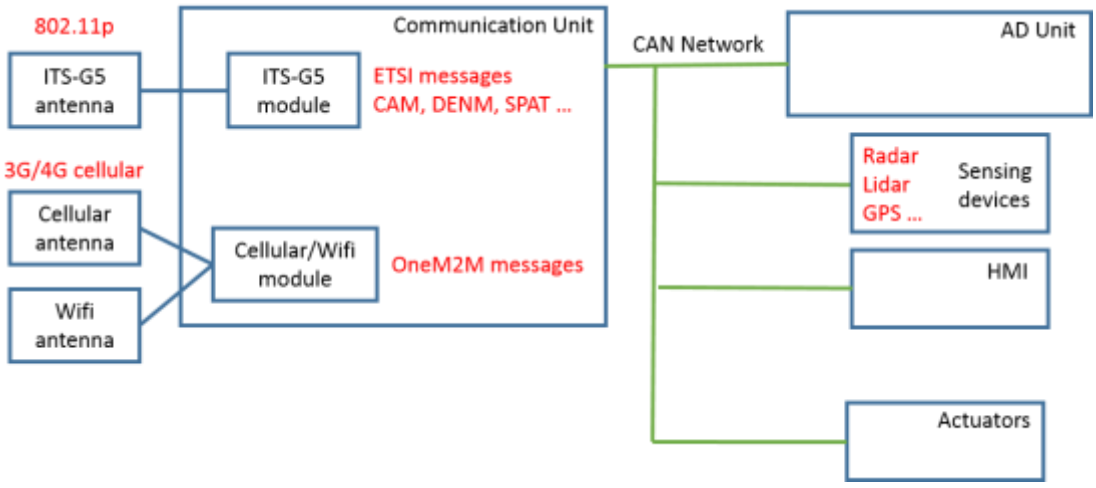

Figure 30 – Pilot Spain main components interfaces

3 Needs, Functional architecture and Requirements

The IoT vehicle platform behaves as the aggregation point for sensors and actuators which extend vehicle functionalities provided by OEM equipment. It coordinates the connectivity of these devices to each other, to OEM sub-systems and to an external network. The IoT vehicle platform can be considered as the automotive counterpart of the IoT Gateway component of an IoT infrastructure.

3.1 Main functionalities and needs (Top Level Requirements)

In-vehicle IoT platform acts as a gateway which interfaces physical devices (e.g. sensors, actuators, devices) and OEM systems, communicating with a heterogeneous approach and in a protocols-agnostic way, with the global cloud-based IoT platform. The core functionalities required by IoT In-vehicle platform can be summarized in the followings:

- Interoperability: The In-vehicle IoT platform should work with heterogeneous devices, technologies, applications, without additional effort from the application or service developer. Heterogeneous components need to be abstracted and must be able to exchange data and services. Interoperability can be seen from network, syntactic, and semantic perspectives.
    - Communication interoperability should allow the platform to transfer information seamlessly among sensors and actuators networks, physical devices and sub-systems which use different transport protocols.
    - Syntactic interoperation should allow the harmonization of formatting and encoding structures of any exchanged information or service.
    - Semantic interoperability refers to the meaning of information or services, and should enable mutual understanding of interchanged information among the set of devices and services connected to the platform.
- Service-based: The In-vehicle IoT platform should be service-based to offer high flexibility when new and advanced functions need to be added. All these and other advanced services can be designed, implemented, and integrated in an application container, or run-time environment, which is a service-based framework (e.g. Java-OSGi, Python, ROS, Node.js) able to provide a flexible and easy environment for application development.
- Context-awareness: Context-awareness is a key requirement in building adaptive applications and services and in annotating values from sensed data. The in-vehicle IoT platform needs to be aware of the context through a sort of “world model” (e.g. LDM), using this for development of effective services.
- Data management: Data refer to sensed data or any information of interest to IoT applications. An in-vehicle IoT platform needs to provide data management services to applications, including data acquisition, data processing, data fusion at the “edge” and data local storage capabilities to deal with network latency and reliability. Data Management also deals with the collection of information from external elements to the vehicle (i.e. cloud / RSU / other vehicles and infrastructures), exploiting data in order to create services such as planning and control application related to AD system. The In-vehicle IoT should handle events typically with not real-time constraints. The platform could have analytics capabilities and should be connected to the UI to interact with the driver.
- Remote management: the ability to remotely provision, configure, update, monitor, startup/shutdown the In-vehicle IoT platform as well as software components (i.e. services and applications) running on the platform itself.
- Security and privacy: security needs to be implemented for both devices and applications. Features such as authentication, encryption, and authorization need to be part of each component of the architecture. Furthermore, every block of in-vehicle IoT platform which uses personal information, needs to preserve the owner’s privacy.

- Based on open standards: communication between the stacks should be based on open standards to ensure interoperability.
- Defined APIs: Providing an API for application developers is an important functionality. APIs allow easy integration with existing applications and integration with other IoT solutions. The programming paradigm (e.g., publish/subscribe, REST) deals with the model for developing or programming the applications or services.
- Event Management, Analytics & UI: The In-vehicle IoT should handle events typically with not real-time constraints. The platform could have analytics capabilities and should be connected to the UI to interact with the driver.

The listed core functionalities and needs can be summarized in the high-level functional architecture shown in Figure 31.

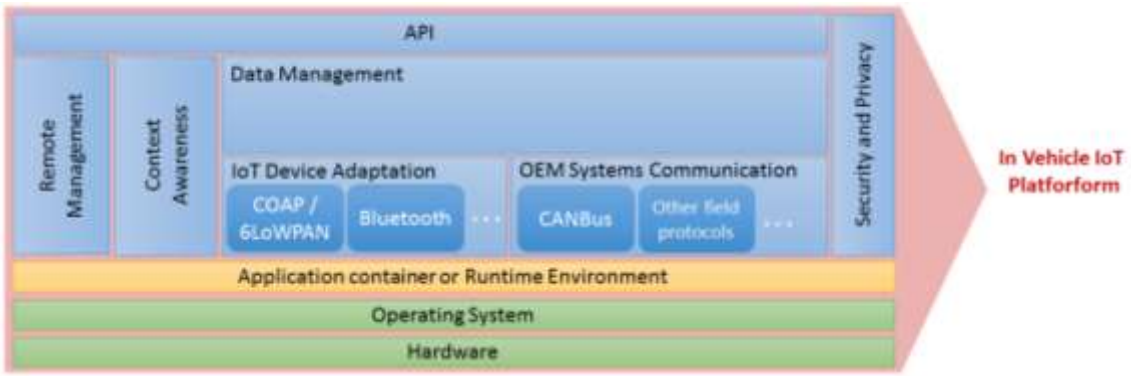

Figure 31 - High-level functional architecture

3.2 Functional Architecture

This section introduces the functional architecture of the Vehicle IoT Platform. This architecture is a high-level decomposition of the Vehicle IoT Platform into major components which aim to accomplish the general basic functionalities addressed in Section 3.1. The characterization of the interaction among the components is also considered.

The architecture in Figure 31 shows that AUTOPILOT applications interact with the vehicle either directly or via the IoT Platform (Figure 4). While, the architecture depicted in Figure 32 shows the Vehicle IoT Platform and how it interacts with the Vehicle On-board components and with the IoT platform. **Vehicle IoT Platfom** is the set of software and hardware elements that are related to IoT world.

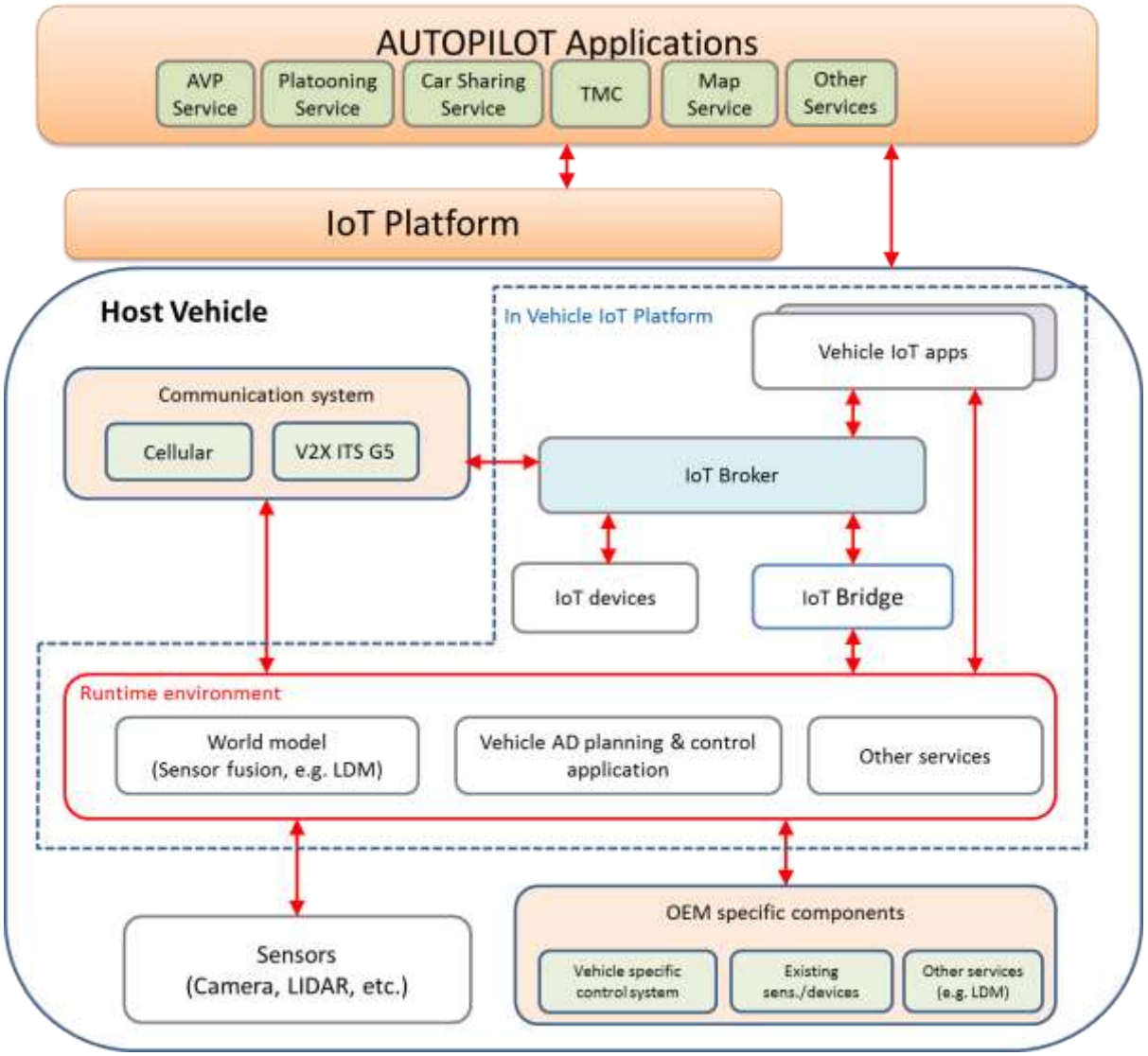

Figure 32 – In-vehicle Architecture

Architecture components can be classified in In-vehicles Components and external components.

In-vehicles Components:

- **In-vehicle IoT applications**: consume and process application specific data via IoT broker and can interact with other components in the system (e.g. world model). According to the type of application, they can cover Data Management functionalities and/or interface for the final user (UI)
- **IoT broker**: connects vehicle with other IoT brokers in the cloud, edge or in other vehicle/roadside units. Since it is directly related to the communication with external applications and other systems, the IoT Broker is the interface between the vehicle and the eternal world (cloud / edge / other vehicle)
- **IoT devices**: are other devices (in the edge/cloud) that are connected to the car. Software modules implement drivers to virtualize such physical IoT devices (sensors and actuators) into Vehicle IoT Platform, in such a way to satisfy the IoT Device Adaptation functionality
- **IoT Bridge**: connects the IoT Broker, and therefore the IoT world, to the development environment inside the car. It takes care of the exchange of data between IoT and non-IoT components. Considering its bridge position between the internal components and interface toward the external world, this component fulfils part of the APIs functionality, and also satisfies

the syntactic Interoperability functionality. In some cases the IoT Bridge and the IoT Broker can be merged into a single component

- **World model**: creates a high-level view of the surroundings that can be used by planning/control applications and IoT apps. The vehicle world model will combine and fuse data coming both from internal sensors and external entities such as the IoT cloud services via the IoT broker, the roadside units or the other vehicles (V2X). The vehicle world model component will include a high-level description of objects (e.g., shape of cars, pedestrians), road/lane (e.g., road shape) with optional semantics information (e.g., classification of objects). This will allow the high-level path planning and control to make the best decision at a certain point in time. Context-Awareness functionality is satisfied with such a type of architectural component
- **Vehicle path planning and control**: high-level planning and control that can leverage IoT data to improve its functionalities (e.g., global route or speed advice coming from the IoT apps). This will be complementary to the already present low-level control and actuators functions in the vehicle. Data Management functionality deals with the collection of information from external elements to the vehicle (i.e. cloud / RSU / other vehicles and infrastructures), exploiting data in order to create services such as planning and control application related to AD system
- **Other services**: are basically anything that can be used to support the car functionalities but that is not directly connected to the planning/control or WM (e.g. traffic light recognition, license plate identification, Vehicle Platform configuration and Remote Management, etc.)
- **Sensors**: refers to sensors that are not OEM specific (e.g. MAP, MAF, lambda, etc.) and whose purpose is usually associated with AD functions (stereo cameras, LIDAR, etc.). Software modules implementing drivers to adapt and virtualize such sensors are needed too
- **OEM specific components**: relates to components that are OEM specific such as actuators for power steering and brakes, inputs to gearbox, or vehicle sensors needed for the “normal” vehicle functions (MAP, MAF, ABS, etc.). Software modules implementing drivers to virtualize such OEM specific components into Vehicle IoT Platform are needed, in such a way to satisfy the OEM Systems Communication functionality
- **Communication system**: are the components that provide communications to the outside. Be it a simple IP based cellular network or V2X ITS G5. These communication media can provide information of the outside world or can send information to the outside world (e.g. V2X ITS G5)

External Components:

- **AUTOPILOT applications**: these applications interface the IoT Platform and implement AUTOPILOT function in the cloud. Each application communicates with the vehicle via the IoT Platform. An application can also comprise a component that runs in the Vehicle Platform. These components can be either an IoT application or an In-vehicle application, depending on the level of integration with the IoT platform.
- **IoT Platform**: this platform implements the IoT functions at the Cloud or Edge level. It comprises also other vehicles and roadside elements.

Table 2 summarises the functional architecture components that satisfy the needs and functionalities characterising a Vehicle IoT Platform as defined in Section 3.1.

Table 2 – Functional architecture components vs. functionality

Architecture Component	Functionality
In-vehicle IoT applications	- Data management - Event Management, Analytics & UI
IoT broker	- Defined APIs - Communication interoperability
IoT Bridge	- Communication interoperability - Syntactic and Semantic Interoperability
IoT devices	- IoT Device Adaptation
World model	- Data management
Sensors	- IoT Device Adaptation
OEM specific components	- OEM Systems Communication
Communication system	- Defined APIs
Vehicle path planning and control	- Data Management
Other services	- Remote Management

3.3 Requirements

Considering the various In-vehicle IoT Platforms and Use Cases that will be implemented in the AUTOIPILOT project, a set of detailed Functional Requirements is here collected. With the term "functional requirements" we mean those technical requirements to realize the In-vehicle IoT Platfrom in practice, some of these are general for all Test Sites, and some are more specific for single test sites. In some cases a functional requirement can be supported with the definition of a "non-functional requirement", that can provide descriptions of measurement details, required thresholds for expected performances and information related with possible physical constraints.

The AUTOPILOT Work Package 1 - Task 1.3 has provided a complete data of functional and non-functional requirements, with additional details and notes that will help next integration and implementation phases. In Task 1.3, partners that are directly responsible of an In-vehicle IoT Platform prototype or involved directly in Use Case for the Test Sites, provided input related to their implementations. The excel file that include all collected requirements is an internal AUTOPILOT document, and here is presented a summary of the contents.

We started our collection work, defining the "primary functions" as the main categories to which to associate a functional requirement:

- **P1 In-vehicle data**: this primary function is responsible for all task related with exchange of data with Intra-Vehicle Network (e.g. CAN bus), including vehicle positioning and timing, vehicle AD control system and existing sensors and devices (if needed).
- **P2 Communication**: this primary function is responsible for all task related with communication and connectivity with external services.

- **P3 IoT devices**: this primary function is responsible for all task related with exchange of data with additional IoT devices that can be installed directly inside prototype vehicles.
- **P4 In-vehicle IoT PF Services & Applications**: this primary function is related with all services and applications hosted by In-vehicle IoT Platform.

In relation with the defined primary functions, a list of 56 functional requirements was produced. Table 3 reports a summary of the content where the first five columns describe:

- ID# [Functional Requirement Identifier]
- P# Link (Associated Parent Link)
- Use Case
- Keyword/Title
- Description

Table 3 – Functional requirements (summary)

ID#	P# Link	Use case	Keyword/Title	Description
FR1	**P1**	All	Vehicle positioning and timing data	In-vehicle IoT Platform must be able to receive vehicle positioning and timing data (e.g. latitude, longitude, heading, timestamp) from same GNSS devices used by existing vehicle AD system
FR2	**P1**	All	Vehicle dynamic data	In-vehicle IoT Platform must be able to receive vehicle dynamic data (e.g. speed, yaw rate, accelerations, etc..) from intra-vehicle network
FR3	**P1**	All	Vehicle static data	In-vehicle IoT Platform must be able to receive vehicle static data (e.g. vehicle length and width) from intra-vehicle network
FR4	**P1**	Platooning	Vehicle AD functionalities related data	In-vehicle IoT Platform should be able to receive data from Autonomous Driving functionalities and related status (e.g. Lane Keeping function status, ACC function status, Lane Change function status, etc..)
FR5	**P2**	All	ITS-G5 received data: CAM/DENM	In-vehicle Iot Platform must be able to receive CAM/DENM decoded contents from received ITS-G5 messages
FR6	**P2**	All	ITS-G5 received data: SPaT/MAP	In-vehicle Iot Platform must be able to receive SPaT/MAP decoded contents from received ITS-G5 messages
FR7	**P2**	All	Cellular LTE received data for IoT App.	In-vehicle IoT Platform must be able to receive data from communication system, related with contents received from IoT external services.
FR8	**P2**	All	Cellular LTE generated data for IoT services	In-vehicle Iot Platform must be enabled to provide /communicate elaborated data to IoT external services, through communication system.
FR9	**P3**	All	IoT devices: output data	In-vehicle IoT Platform must be able to receive data from internal IoT devices, installed inside prototype vehicle.

ID#	P# Link	Use case	Keyword/Title	Description
FR10	**P3**	All	IoT devices: input data	In-vehicle IoT Platform must be able to provide data to IoT devices installed inside prototype vehicle
FR11	**P3**	All	IoT devices: plug&play	In-vehicle IoT Platform must should be able to integrate heterogeneous a new IoT devices into the prototype, maintaining a proper logical and functional separation between additional IoT sensors itself and intra-vehicle network (no interference).
FR12	**P1**	AVP	Unmanned mode	The In-vehicle IoT platform should be able to decide when the driver has left the vehicle (e.g. reading CAN bus data related with "opened doors" and "seat occupancies") and switch to **Unmanned Mode**
FR13	**P1**	AVP	Unmanned mode	The AD vehicle must respond to control commands transmitted by the control centre (emergency stop, take-off)
FR14	**P1**	AVP	Unmanned mode	The vehicle goes to a low power mode when arriving at the parking place
FR15	**P1**	AVP	Unmanned mode	The control centre wakes up the vehicle to move to the collect point
FR16	**P3**	AVP	Unmanned mode	Driver validation: the vehicle grants access to authorized drivers
FR17	**P2**	Hazard Avoidance	Hazard detection	The In-vehicle PF must be able to receive hazard warning information from connected IoT Infrastructure PF (alternative to DENM from ITS-G5 channel, from cloud connection)
FR18	**P4**	Pothole detection	Pothole analysis	The In-vehicle PF must be able to elaborate data received from (ego) vehicle and determine pothole presence on the road
FR19	**P1**	Speed Adaptation	Speed Limit adaptation	The In-vehicle IoT PF must be able to receive feedback from AD vehicle system, when vehicle reacting to speed limits modification.
FR20	**P1**	Lane Following	Lane Following	The In-vehicle IoT PF can be informed when AD vehicle is activating Lane Following functionalities (lateral control)
FR21	**P1**	Lane Change	Lane Change	The In-vehicle PF can be informed when AD vehicle is activating Lane Change functionalities (lateral control)
FR22	**P2**	Obstacle or VRU detection	VRU detection	The in-vehicle PF can be able to receive information related with VRU presence, generated by IoT infrastructure PF (alternative to CAM/DENM from ITS-G5 channel, for long range).

ID#	P# Link	Use case	Keyword/Title	Description
FR23	**P4**	Obstacle or VRU detection	VRU data elaboration	The in-vehicle PF must be able to elaborate VRU position compared with Ego-vehicle dynamic data, to estimate the related threats.
FR24	**P2**	Traffic Light handling	Traffic Light	The In-vehicle PF can be able to receive Signal Phase information, generated by IoT infrastructure PF (alternative to SPaT/MAP from ITS-G5 channel, for long range)
FR25	**P2**	AVP	Unmanned mode indoor	In-vehicle IoT Platform must be able to provide the vehicle identification to be authorized at the parking place
FR26	**P2**	AVP	Unmanned mode indoor	The In-vehicle IoT platform should be able to receive a detailed layout of the parking place and the location of dynamic objects
FR27	**P2**	AVP	Unmanned mode indoor	The In-vehicle IoT platform should be able to provide the Position related to a virtual parking map
FR28	**P2**	AVP	Unmanned mode indoor	The In-vehicle IoT platform should be able to receive a Pedestrian detection relative to a virtual parking map
FR29	**P2**	AVP	Sleep mode	The In-vehicle IoT platform should be able to inform when switch to **Sleep mode**.
FR30	**P2**	Highway Pilot	Speed Limit adaptation	The ACC/AD vehicle must acknowledge speed limitation command transmitted by the control centre.
FR31	**P1**	Highway Pilot	Speed Limit adaptation	The ACC/AD vehicle should execute the speed limitation command transmitted by the control centre.
FR32	**P2**	Highway Pilot	Speed Limit adaptation	The ACC/AD vehicle should acknowledge the execution of the speed limitation command transmitted by the control centre.
FR33	**P2**	Highway Pilot	Safety Distance adaptation	The ACC/AD vehicle must acknowledge safety distance command transmitted by the control centre.
FR34	**P1**	Highway Pilot	Safety Distance adaptation	The ACC/AD vehicle should execute the safety distance command transmitted by the control centre.
FR35	**P2**	Highway Pilot	Safety Distance adaptation	The ACC/AD vehicle should acknowledge the execution of the safety distance command transmitted by the control centre.
FR36	**P2**	Highway Pilot	Takeover order (Auto to Manual)	The ACC/AD vehicle must acknowledge the takeover command transmitted by the control centre.
FR37	**P1**	Highway Pilot	Takeover order (Auto to Manual)	The ACC/AD vehicle should execute the takeover command transmitted by the control centre.

ID#	P# Link	Use case	Keyword/Title	Description
FR38	**P2**	Highway Pilot	Takeover order (Auto to Manual)	The ACC/AD vehicle should acknowledge the execution of the takeover command transmitted by the control centre.
FR39	**P2**	Highway Pilot	Broadcast warning about road hazard	The vehicle must be able to broadcast over ITS G5 a warning message about a critical and relatively certain Road Hazard danger (ex: vehicle stopped on the lane)
FR40	**P4**	All	Service Based	The In-vehicle IoT PF should be service-based to offer high flexibility when new and advanced functions need to be added.
FR41	**P4**	All	Remote Management	The In-vehicle IoT PF, as well as software components (i.e. services and applications) running on the platform itself, should be able to be remotely provisioned, configured, updated, monitored, startup/shutdown.
FR42	**P4**	All	Interoperability	The In-vehicle IoT PF should allow applications to seamlessly work with heterogeneous devices, technologies and systems, without the need of extensive development effort.
FR43	**P4**	All	Semantic interoperability	The In-vehicle IoT PF should annotates raw data with semantic information, enabling mutual understanding of interchanged information among inner and external devices and services.
FR44	**P1**	All	World model	The IoT Vehicle platform must be able to receive world model related data (e.g., objects detected, road markings) coming from vehicle internal sensors, if available, with the following general measurement information required by sensor fusion algorithms: **confidence level, description of sensor type**, **timestamp.**
FR45	**P4**	All	World model	The sensor fusion algorithm must be able to process incoming data from internal vehicle sensors, IoT devices and IoT cloud, to build a coherent "World Model".
FR46	**P1**	Platooning	World model	The IoT platform must provide **tracking information (ID, speed, acceleration, position) of both front and rear vehicles** to the control system in order to meet requirements of the CACC system.
FR47	**P1**	Platooning	World model	The IoT platform must provide **road model information (road edge or lane markings)** to the control system in order to meet requirements of the CACC system.

ID#	P# Link	Use case	Keyword/Title	Description
FR48	**P2**	All	World model	The IoT platform must be able to receive and share world model related data (e.g., objects detected, road markings) coming from IoT cloud services with the following general measurement information required by sensor fusion algorithms: **confidence level, description of sensor type**, **timestamp**.
FR49	**P2**	All	World model	The IoT platform should be able to receive and share **shape description of objects** (e.g., bounding box dimensions).
FR50	**P2**	Platooning	CACC message	The IoT platform must be able to receive and share platooning management information such as **target acceleration**, **time gap to vehicle in front, controller type (manual, ac, acc, cacc)** to the control system in order to meet requirements of the CACC system.
FR51	**P4**	Platooning	Platooning IoT service data	The IoT platform must be able to receive and share platooning IoT service data such as **request to join, request to leave, location of formation.**
FR52	**P1**	All	Vehicle Safety	The safety and time critical information over the onboard IoT platform shall be clearly identified and separated over a dedicated channel.
FR53	**P1**	All	Vehicle Safety	The emergency stop button shall disconnect the prototype controllers / IoT platform, and provide the manual controls (back to the driver)
FR54	**P4**	All	UI connection	The IoT Platform should be able to inform and interact with the driver, concerning specific events notification or warnings
FR55	**P2**	Urban Driving (TU/e Rebalancing)	Probabilistic world model	The in-vehicle PF can be able to receive TU/e lecturing scheduling information from the TU/e webserver (WiFi connection).
FR56	**P2**	Urban Driving (TU/e Rebalancing)	Probabilistic world model	The in-vehicle PF is able to receive weather information from internet.

In relation with the functional requirements, a list of 53 non-functional requirements was collected. Table 4 reports a summary of the content where the first 5 columns describe:

- ID# [Non-Functional Requirement Identifier]
- FR# Link [Associated Functional Req.]
- Use Case
- Keyword/Title
- Description

Table 4 – Non-Functional requirements (summary)

ID#	FR# Link	Use case	Keyword/Title	Description
NFR1	**FR1**	All	Vehicle Reference Position	In-vehicle dynamic data resolution: Referecence Position latitude and longitude [deg] (0,0000001 deg; WGS84 co-ordinate system - see CAM req.)
NFR2	**FR1**	All	Vehicle Heading	In-vehicle dynamic data resolution: Reference Positiong Heading [deg] (0,1 degree; with regards to the WGS84 north - see CAM req.)
NFR3	**FR1**	All	Vehicle TimeStamp	In-vehicle dynamic data resolution: Reference timestamp as used to define GenerationDeltaTime inside CAM [ms]. Number of milliseconds since 2004-01-01T00:00:00.000Z, as specified in ISO 8601; +/- 1 ms
NFR4	**FR2**	All	Vehicle Speed	In-vehicle dynamic data resolution: speed [m/s] (0,01 m/s; range: [0; 163,82] m/s - see CAM req.)
NFR5	**FR2**	All	Reverse Gear	In-vehicle dynamic data resolution: reverse gear status range [0;1]
NFR6	**FR2**	All	Vehicle Longitudinal & Lateral Accelerations	In-vehicle dynamic data resolution: longitudinal and lateral accelerations [m/s^2] corresponds to the vehicle coordinate system as specified in ISO 8855 (0,1 m/s^2 ; negative values for longitudinal acc. indicates vehicle is braking; negative values for lateral acc. indicates the vehicle is accelerating right; range: [-16; +16] m/s^2 - see CAM req.)
NFR7	**FR2**	All	Vehicle YawRate	In-vehicle dynamic data resolution: yaw rate [deg/s] corresponds to the vehicle coordinate system as specified in ISO 8855 (0,01 deg/s; negative values indicates that the vehicle is rotating to the right; range:[-327,66; +327,66] - see CAM req.)
NFR8	**FR2**	All	Vehicle Vertical Acceleration	In-vehicle dynamic data resolution: vertical accelerations [m/s^2] corresponds to the vehicle coordinate system as specified in ISO 8855 (0,1 m/s^2; negative values indicates that vehicle is accelerated downwards; range: [-16;+16] m/s^2 - see CAM req.)
NFR9	**FR2**	All	Brake Pedal Status	In-vehicle dynamic data resolution: brake pedal status range [0;1]
NFR10	**FR2**	All	Gas Pedal Position	In-vehicle dynamic data resolution: gas pedal position [%]
NFR11	**FR2**	All	Emergency Brake (or ABS) Status	In-vehicle dynamic data resolution: Emergency Brake status range [0;1]
NFR12	**FR2**	All	Cruise Control Status	In-vehicle dynamic data resolution: Cruise Control Status range [0;1]

ID#	FR# Link	Use case	Keyword/Title	Description
NFR13	**FR2**	All	Adaptive Cruise Control Status	In-vehicle dynamic data resolution: Adaptive Cruise Control Status range [0;1]
NFR14	**FR2**	All	Speed Limiter Status	In-vehicle dynamic data resolution: Speed Limiter Status range [0;1]
NFR15	**FR2**	All	Steering Wheel Angle	In-vehicle dynamic data resolution: Steering Wheel Angle [deg] (1,5 deg; negative values indicates that vehicle is turning on right; range:[-766,5; +766,5] degree - see CAM req.)
NFR16	**FR2**	All	Low Beam Status	In-vehicle dynamic data resolution: Low Beam Lights Status range [0;1]
NFR17	**FR2**	All	High Beam Status	In-vehicle dynamic data resolution: High Beam Lights Status range [0;1]
NFR18	**FR2**	All	Left Turn Signal Status	In-vehicle dynamic data resolution: Left Turn Lights Status range [0;1]
NFR19	**FR2**	All	Right Turn Signal Status	In-vehicle dynamic data resolution: Right Turn Lights Status range [0;1]
NFR20	**FR2**	All	Day Time Running Lights Status	In-vehicle dynamic data resolution: Day Time Running Lights Status range [0;1]
NFR21	**FR2**	All	Reverse Lights Status	In-vehicle dynamic data resolution: Reverse Lights Status range [0;1]
NFR22	**FR2**	All	Fog Lights Status	In-vehicle dynamic data resolution: Fog Lights Status range [0;1]
NFR23	**FR2**	All	Park Lights Status	In-vehicle dynamic data resolution: Park Lights Status range [0;1]
NFR24	**FR2**	All	Path History	A path with a set of path points. It may contain up to 40 path points. (see CAM req.)
NFR25	**FR2**	All	Path Point	[In relation with Path Hystory] It defines a waypoint position within a path, and is composed by two parameters: (i) path position as a delta from a reference position point; (ii) path delta time as a travel time that separates the point from the reference point (see CAM req.)
NFR26	**FR3**	All	Vehicle Length	In-vehicle static data resolution: length [m] (0,1 m; range: [+1; +102,2] meters - see CAM req.)
NFR27	**FR3**	All	Vehicle Width	In-vehicle static data resolution: width [m] (0,1 m; range: [+1; +6,1] meters - see CAM req.)
NFR28	**FR4**		Lane Keeping status	In-vehicle AD function status: Lane Keeping [Idle, torque overlay activated, Not active, …
NFR29	**FR4**		ACC status	In-vehicle AD function status: ACC [Idle, target speed activated, Not active, …]
NFR30	**FR5**		CAM/DENM reception freq.	In-vehicle IoT PF: must be able to receive CAM/DENM contents generated by neighbours, at same freq. of data received from 802.11p interface

ID#	FR# Link	Use case	Keyword/Title	Description
NFR31	**FR5**		SpaT/MAP reception freq.	In-vehicle IoT PF: must be able to receive Spat/MAP contents generated by infrastructures, at same freq. of data received from 802.11p interface
NFR32	**FR5, FR6**	All?	ITS-G5 security & privacy	In-vehicle IoT platform must/should ensure for ITS-G5 comm. a level of security compliant with last ETSI requirements on cybersecurity
NFR33	**FR7, FR8**	All?	Cellular LTE security & privacy	In-vehicle IoT platform must/should ensure for ITS-G5 comm. a level of security compliant with last ETSI requirements on cybersecurity
NFR34	**FR7, FR8**	AVP	Unmanned mode	During unmanned mode the vehicle has to have ***uninterrupted*** connection to a control centre
NFR35	**FR7**	Urban	Traffic light information	Traffic Light status/phase information: [traffic light ID; traffic light absolute position; interested road and direction; current status/phase, next status/phase, time to change status/phase, approximation trace]
NFR36	**FR7, FR8**	All?	VRU detection	VRU information: absolute position (latitude; longitude) with the same resolution as for vehicle positioning; type of VRU; estimated speed; estimated direction; estimated acceleration]
NFR37	**FR7, FR8**	Urban	traffic jam events	Traffic Jam information: [interested road/lane; interested direction; estimated starting point; estimated ending point]
NFR38	**FR7, FR8**	Urban	road work warning events	Road Works information: [interested road/lane; interested direction; estimated starting point; estimated ending point]
NFR39	**FR7, FR8**	Urban	Weather events	Weather information: [interested road/lane; interested direction; estimated starting point; estimated ending point] CTAG: propose to add "Weather type" (fog, wind, snow ….) and change start/end point by location and radius
NFR40	**FR42**	Platooning	Tracking information: position	Cartesian position (x and y in meters) with respect to the center point of the object. Resolution: 1 centimeter. The accuracy must be at least 0.5 meter as minimum requirement.
NFR41	**FR42**	Platooning	Tracking information: speed	Cartesian velocity (x and y in m/s). Resolution: 0.01 m/s
NFR42	**FR42**	Platooning	Tracking information: acceleration	Cartesian acceleration (x and y in m/s^2). Resolution: 0.01 m/s^2
NFR43	**FR42**	Platooning	Tracking information: age	Age of measurements in seconds. Resolution: 0.001 s

ID#	FR# Link	Use case	Keyword/Title	Description
NFR44	**FR42**	Platooning	Tracking information: identifier	Unique identifier of tracked object as integer [0, MAX_INTEGER]
NFR45	**FR43**	Platooning	Road model: lane polynomial	Lane markings of a road are represented with a third degree polynomial: y = c0 + c1 * x + c2 * x * x + c3 * x * x * x, where c0 and c1 are determined by the pose, c2 is the second derivative [1/m], c3 is the third derivative [1/m^2].
NFR46	**FR43**	Platooning	Road model: lane ID	Lane ID corresponding to the polynomial as defined above as integer [0, MAX_INTEGER]
NFR47	**FR46**	Platooning	CACC message: sending rate	CACC messages specific for platooning must be sent at 25 Hz
NFR48	**FR46**	Platooning	CACC message: controller type	Controller type: manual = 0, cc = 1, acc = 2, cacc = 3
NFR49	**FR46**	Platooning	CACC message: response delay	Response time constant. Resolution 0.01 seconds. Unavailable = 1001
NFR50	**FR46**	Platooning	CACC message: target longitudinal acceleration	Target longitudinal acceleration. Resolution: 0.01 m/s^2. Unavailable = 1001
NFR51	**FR46**	Platooning	CACC message: time gap	Time gap with respect to front vehicle. Resolution: 0.1 seconds. Unavailable = 361
NFR52	**FR46**	Platooning	CACC message: cruise speed	Cruise vehicle speed. Resolution: 1 cm/s. Unavailable = 5001
NFR53	**FR46**	Platooning	CACC message: rear axle location	Rear axle location of front vehicle. Resolution: 1 cm

4 Specifications

This chapter starts with a first section describing different IoT technologies, which can be intended as a glossary for the second section, in which every Test Site report the initial specifications for the In-vehicle IoT platform in plans or already implemented.

4.1 State of Art of different IoT technologies

The In-vehicle IoT Platfom is composed by a set of software and hardware elements that are related to IoT world, and into this "world" the available combinations are various. In the different Test Sites of the AUTOPILOT project, can be possible to follow different strategies to obtain an IoT Vehicle Platform. This section provides a description of State of Art of different technologies in relation with the core functionalities of an IoT Vehicle Platform:

- Remote Management
- Context Awareness
- Data Management
- Security and Privacy
- Communication Interoperability
- Syntactic and Semantic Interoperability
- Application container or Runtime Environment

4.1.1 Remote Management

Remote Management is the ability to remotely provision, configure, update, monitor, startup/shutdown the In-vehicle IoT platform as well as software components (i.e. services and applications) running on the platform itself.

Remote Management is also the ability to create simplified and optimized network driven by remote device and application lifecycle management (Application distribution and lifecycle management; Real-time device monitoring; Service management and diagnostics).

OSGi Remote Management Tools

OSGi Remote Management Tools [13] will introduce idea of remote monitoring and controlling of external application based on OSGi platform.

The crucial thing in OSGi is that it is a dynamic module system for Java. It allows installing, uninstalling and updating all modules without restarting or even stopping application for these operations. It seems apparent that OSGi is an ideal platform for each application server but it can find implementation in other environments (e.g., handheld devices, IT managed environments). In the Eclipse community OSGi is very important because the whole Eclipse platform is based on it.

Remote services are accessed in an entirely transparent way. All that a service provider framework has to do is to register a service for remote access. Subsequently, other peers can connect to the service provider peer and get access to the service. For every remote service, a local proxy bundle is generated that registers the same service. Local service clients can hence access the remote service in the same way and without regarding distribution.

Additionally, Remote Services for OSGi can interact with the EventAdmin service. Frameworks receive all events from connected peers that match one of their EventHandler subscriptions. Even though Remote Services for OSGi is a sophisticated middleware for OSGi frameworks, it uses a very efficient network protocol and has a small footprint. This makes it ideal for small and embedded

devices with limited memory and network bandwidth. The service runs on every OSGi-compliant environment.

Remote Services for OSGi has been tested with Eclipse Equinox, Knopflerfish, and Oscar / Apache Felix, as well as with our own lightweight OSGi implementation Concierge.

4.1.2 Context Awareness

The context awareness will be of great support to process and store the big data, and to make their interpretation easier.

Context-awareness is a key requirement in building adaptive applications and services and in annotating values from sensed data. The In-vehicle IoT platform needs to be aware of the context through a sort of "world model", using this for development of effective services.

Online Horizon system
Vehicle-based applications, such as driver assistance (ADAS) and automated driving (AD), need an up-to-date map for the road ahead and for its surroundings.

We can call this system the **Online Horizon System** to reflect the idea that the information is used to maintain a model of what is to be expected when the vehicle is moving forward.[1] Data in the online horizon typically includes:

1. HD map information, which consists of several map layers, such as lanes, dividers and RoadDNA, and traffic sign locations which are typically used for AD applications, or
2. Live map information, which includes more dynamic aspects, such as traffic jams, hazards and weather.

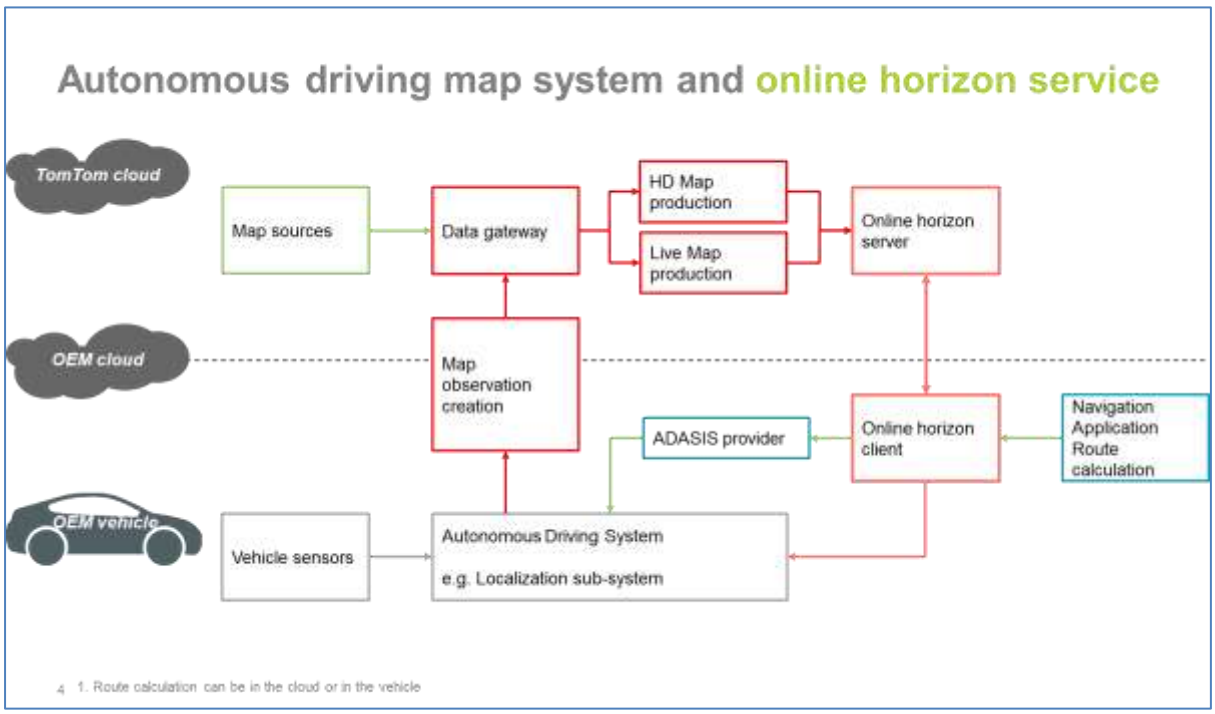

Figure 33 – Online horizon service, example from TomTom

[1] Note that the use of "horizon" in this sense is wider than that in ADASIS. This horizon is used to augment or validate the environmental model the vehicle creates with its own sensors.

The sensors in the car (camera, radar, LiDAR) are typically used for localization and at the same time the sensors also can generate data for keeping the map up-to-date. That is called map observation creation here. The local observations of the environment are collected in the cloud in a gateway, combined with other sources and then used in the production of both the HD Map as well as the Live Map. The Live Map contains temporary hazards like the tail of a traffic jam, temporarily closed lanes, accidents, etc.

World Information on Robot Environments (WIRE)
World Information on Robot Environments (WIRE) is a Robot Operating System (ROS) based probabilistic multiple hypothesis anchoring to create and maintain a semantically rich world model using probabilistic anchoring. Multiple hypothesis tracking-based data association is included to be able to deal with ambiguous scenarios. Multiple model tracking is included to be able to easily incorporate different kinds of prior knowledge.

Environment Description (ED)
Environment Description (ED) is a Robot Operating System (ROS) based 3D geometric, object-based world representation system for robots. In itself ED is database system that structures multi-modal sensor information and represents this in an object-based world representation that can be utilized for robot localization, navigation, manipulation and interaction functions and visualizes all this through a web-based GUI.

4.1.3 Data Management

Data is uploaded in real-time or at the conclusion of each trip. Data set examples include vehicle speed, GPS location, and exception events. The vehicle devices can communicate using MQTT, an industry-standard communications protocol. Each device must be mutually authenticated prior to exchanging data and must be associated with a specific VIN. Communication is two-way between the telemetry devices, because they both supply data and can accept messages.

It is also the ability to filter, analyze, and correlate vehicle sensor data and take action on the large amount of data generated, for instance: real-time situational awareness, faster decisions, and immediate actions locally at the machine level and the enterprise back end; agnostic of event sources, destination or underlying communication layer; tooling and event flow monitoring.

Data refer to sensed data or any information of interest to IoT applications. An in-vehicle IoT platform needs to provide data management services to applications, including data acquisition, data processing, data fusion at the “edge” and data local storage capabilities to deal with network latency and reliability. The in-vehicle IoT should handle events typically with not real-time constraints. The platform could have analytics capabilities and should be connected to the UI to interact with the driver.

Data Fusion Techniques
To its core, data fusion is the process of combining different data sources to generate better information to improve decision. In an automotive context, vehicles have embedded sensors that generate data to ADAS (Advanced Driver Assistance Systems). In Autopilot scope, where IoT data will enhanced and enable AD, data will be sent to and received from a back-end platform or an IoT platform.

Initially data fusion is needed to aggregate different data sources (from different sensors) in order to generate better information or to unify the same information, viewed by different sensors. With IoT data, vehicles need to communicate with a back-end platform/IoT platform. Data fusion will play an important role in such communications. Indeed, it is likely that bandwidth in-between vehicle and

cloud platform will be limited and as such, the vehicle will need to 'select' the data to send. In the idea, data should be fused and also reduced in volume.

There are numerous combinations of data fusion, e.g. GPS position data could be correlated with Road Sign Unit or a connected pedestrian. Each of these combinations is unique in terms of data format that could depend on the technologies, the use case or the vehicle.

In this perspective, providing a specific data fusion technique could be possible but it would be only usable in a specific case (i.e. for the specific data format/use case/vehicle). To tackle this issue, the idea behind this section is to provide some guidance to answer the question of the data fusion.

In a first step, a functional data classification should be done in order to evaluate what are the data available, the sources, format and the quality of the data. Based on this preliminary analysis, a common format or data structure should be chosen. Data transformation on each dataset should then be computed in order to transform them into the target format/data structure so that data fusion can be done. In this step of the process, data should be ready to be fused.

4.1.4 Security and Privacy

Security needs to be implemented for both devices and applications. Features such as authentication, encryption, and authorization need to be part of each component of the architecture. Furthermore, every block of in-vehicle IoT platform which uses personal information, needs to preserve the owner's privacy. Developing componentized applications according to supported security standards means that will be less likely to require new security measures as hardware platform changes. In-vehicle data and devices are segregated and protected from external threat sources.

All on board devices cooperate to protect sensitive data and access to key safety related devices.
Data is protected both during vehicle travel time and at rest, both on storage and in transit. Each in-vehicle device is classified according to standards in course of identification in D1.9 so that its security requirements are defined in accordance to the outcomes of the risk analysis contained into the same document.

DoS (Denial of Service) attacks against in-vehicle devices are promptly detected and countered by autonomous systems that have the primary purpose of protecting the vehicle, passengers and traffic safety.

In-vehicle devices also cooperate with infrastructure devices to protect user privacy in the cloud, for example by means of anonymized data and usage of pseudonyms.

A mechanism exists that allows critical vehicle components to be checked for genuineness / authenticity. Tamper proof techniques allow vehicle components to be secure even while not assembled into the vehicle.

4.1.5 Communication interoperability

Communication interoperability should allow the platform to transfer information seamlessly among sensors and actuators networks, physical devices and sub-systems which use different transport protocols.

LCM - Lightweight Communications and Marshalling
Lightweight Communications and Marshalling (LCM) is a set of libraries and tools that enable

message passing and data marshalling (i.e. encoding and decoding in efficient way, and rearrangement/assembling of message addressed for group of interested recipients), simplifying the development of low-latency messages and targeted to real-time robotics applications [14].

The LCM main components are:

- data type specification language
- message passing system
- logging/playback tools
- real time analysis tools

LCM implements an efficient broadcasting mechanism using UDP Multicast, with a “push”-based publish/subscribe model that offer low-latency performances, is supported by various platforms (e.g. GNU/Linux, OS X, Win, POSIX) and by many of the most used programming languages (e.g. C, C++, Java, Python, etc..). Thanks to the various API provided by LCM [15], it is possible to enable messages exchange, with a minimized effort for the configuration of involved systems or platforms and obtaining an efficient inter-process communication. The type specification language that can be used to create type definitions are independent of used platform and programming language. The LCM code generation tool is used to automatically generate language-specific bindings that provide representations of the message in a data structure.

LCM is distinctive from other approaches also for the offered debugging and analysis system, with tools for recording data (i.e. LCM-logger), for deep inspection of all messages captured from a network (i.e. LCM-spy) and the possibility to replay (i.e. LCM-logplayer) and analyze with graphical features, data captured from a test session.

ZeroMQ

ZeroMQ (or ØMQ) [16] is a messaging system, or "message-oriented middleware", used in environments as diverse as financial services, game development, embedded systems, academic research and aerospace.

ZeroMQ is developed by a large community of contributors, founded by iMatix, which holds the domain name and trademarks. There are third-party bindings for many popular programming languages.

ZeroMQ looks like an embeddable networking library but acts like a concurrency framework. It gives sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast. It is possible to connect sockets N-to-N with patterns like fan-out, pub-sub, task distribution, and request-reply. It's fast enough to be the fabric for clustered products. Its asynchronous I/O model gives scalable multicore applications, built as asynchronous message-processing tasks. It has a score of language APIs and runs on most operating systems.

Advanced Message Queuing Protocol (AMQP)

The enterprise-level Advanced Message Queuing Protocol (AMQP), developed by the OASIS open standards consortium, is an open standard application layer protocol for message-oriented middleware. It provides a platform-agnostic method for ensuring information is safely transported between applications, among organizations, within mobile infrastructures, and across the Cloud. AMQP is used in areas as varied as financial front office trading, ocean observation, transportation, smart grid, computer-generated animation, and online gaming. As the name implies, it provides a wide range of features related to messaging, including reliable queuing, topic-based publish-and-subscribe messaging, flexible routing, transactions, and security. AMQP exchanges route messages directly—in fan-out form, by topic, and also based on headers. There are Cloud-hosted offerings of

AMQP, and it is embedded in virtualization infrastructure.

To enable complete interoperability for messaging middleware requires that both the networking protocol and the semantics of the server-side services are sufficiently specified. AMQP, therefore, defines both the network protocol and the server-side services through:

- A defined set of messaging capabilities called the "Advanced Message Queuing Protocol Model" (AMQ model). The AMQ model consists of a set of components that route and store messages within the broker service, plus a set of rules for wiring these components together.
- A network wire-level protocol, AMQP, that lets client applications talk to the server and interact with the AMQ model it implements.

One can partially imply the semantics of the server from the AMQP protocol specifications but we believe that an explicit description of these semantics helps the understanding of the protocol.

There are three major pieces specified in the scope of AMQP 1.0. These define the networking protocol, a representation for message envelope data and the basic semantics of broker services.

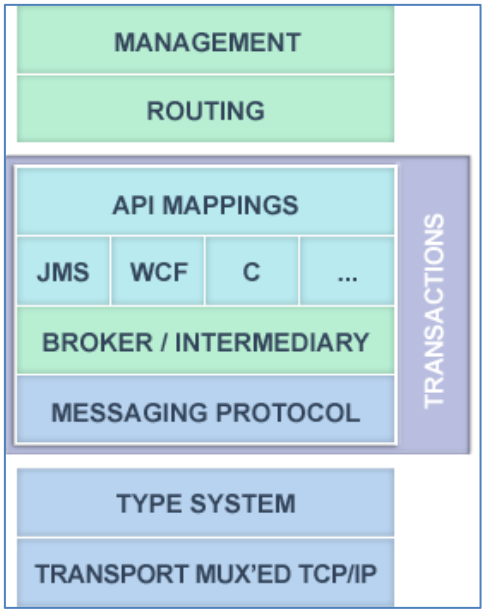

Figure 34 – AMQP Architecture

The AMQP Network Protocol

The AMQP Network protocol [17] defines:

- A peer to peer protocol; though normally in AMQP one peer is playing the role of a client application and the other peer is playing the role of trusted message routing and delivery service, or broker
- How to connect to services, including a method for failing over connections to alternative services
- A mechanism to enable peers to discover one another's capabilities
- Comprehensive security mechanisms, including SSL and Kerberos for seamless end-to-end confidentiality
- How to multiplex a TCP/IP connection in order that multiple conversations may happen over one TCP/IP connection (simplifies firewall management)
- How to address a source of messages with the network peer, and to specify which messages are of interest

- The lifecycle of a message through fetching, processing, and acknowledgement. AMQP makes it very clear when responsibility for a message is transferred from one peer to another thereby enhancing reliability
- How to enhance performance, if desired, by pre-fetching messages across the network ready for the client to process without delay
- A way of processing batches of messages within a transaction
- A mechanism to allow a complete message transfer from login to logout in one network packet for lightweight applications
- Very capable flow control, which enables consumers of messages to slow producers to a manageable speed, and which enable different workloads to proceed in parallel at different rates over one connection
- Mechanisms for resuming message transfers when connections are lost and re-established; for example in the event of service failover or intermittent connectivity

Message Representation

The applications based on the AMQP protocol do not exchange data speaking the framing "language", but rather it's the messaging layer built on top of it that provides messaging capabilities. This layer defines a well-known structure of the message composed of two main parts:

- Bare message: it's an immutable part from the sender to the receiver. No one intermediary can change its content.
- Annotated message: it consists of the previous bare message plus some annotations that can be used and changed by intermediaries between sender and receiver The bare message contains the body and two types of collections: the first one is for system properties that are standard and well-defined by the AMQP specification; the second one is for application specific properties (also named user properties) that can be added and changed by the application.

Figure 35 – AMQP Message representation

The AMQP 1.0 Type System and message encoding facilities provide a portable encoding for messages to meet this need.

Normally, this encoding is only used to add routing properties to the "envelope" of the message; the contents inside the envelope are transported untouched. Applications will likely use XML, JSON or similar encodings in their message content. Optionally, and application could choose to use AMQP encoding for message content too, but this is entirely optional.

Broker Services

The value of using message brokers is that a trusted intermediary designed for the purpose handles the complexities of message queuing, routing and delivery. That intermediary is the message broker.

AMQP defines the minimum set of requirements expected of a message broker, and where there are frequently used more advanced facilities; it specifies how those facilities are to be exposed to clients.

The goal of AMQP is to enable applications to send messages via the broker services; these lower level concepts are necessary but not the goal in themselves.

MQTT

To start with this specification, we need to introduce the consortium OASIS which is the Open Advancing Standard for the Information Society [18]. This consortium helps the community to reach an agreement on Standards as which we will use to define our in-vehicle IoT platform (VIP).

The Open Charge Point Protocol (OCPP [19]) has been developed by the company, Chargerlink, Inc. This protocol uses Message Queue Telemetry Transport (MQTT [20]) from IBM and Protocol Buffers (ProtoBuf [21]) from Google.

MQTT is a Client Server publish/subscribe messaging transport protocol. It is lightweight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.

The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bidirectional connections. Its features include:

- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
- A messaging transport that is agnostic to the content of the payload.
- Three qualities of service for message delivery:
    - "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
    - "At least once", where messages are assured to arrive but duplicates can occur.
    - "Exactly once", where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.
- A small transport overhead and protocol exchanges minimized to reduce network traffic.
- A mechanism to notify interested parties when an abnormal disconnection occurs.

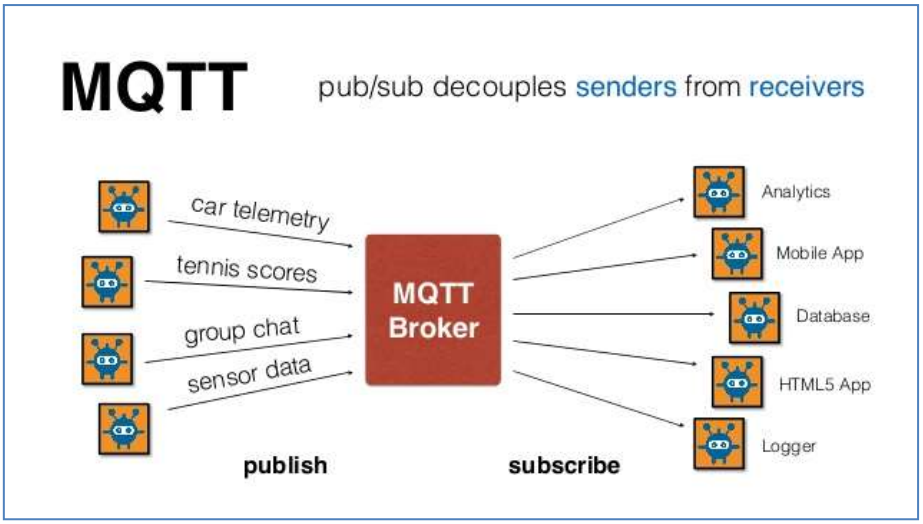

Figure 36 – MQTT description scheme

Data Distribution Service (DDS)

The Data Distribution Service (DDS™, [22]) is a middleware protocol and API standard for data-centric connectivity from the Object Management Group® (OMG®). It integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical Internet of Things (IoT) applications need. The standard is used in applications such as smartphone operating systems, transportation systems and vehicles, software-defined radio, and by healthcare providers.

DDS is uniquely data centric (which is ideal for the Internet of Things), so ensures that all messages include the contextual information an application needs to understand the data it receives.

Applications communicate by publishing and subscribing to Topics identified by their Topic name. Subscriptions can specify time and content filters and get only a subset of the data being published on the Topic. Different DDS Domains are completely independent from each other. There is no data-sharing across DDS domains.

The essence of data centricity is that DDS knows what data it stores and controls how to share that data.

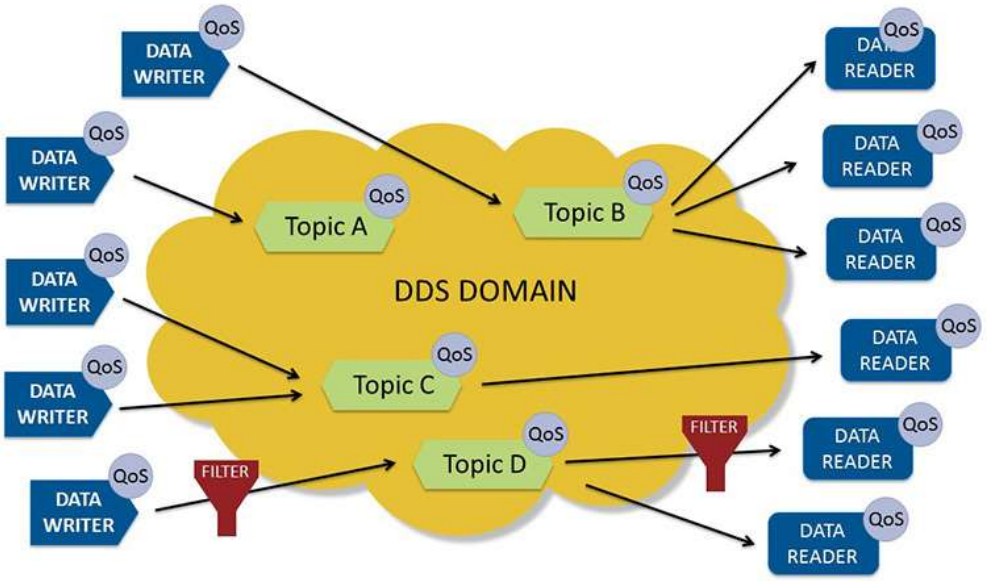

Figure 37 – DDS Architecture

Protocol Buffers

Protocol Buffers [21] is a method of serializing structured data. It is useful in developing programs to communicate. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data. The design goals for Protocol Buffers emphasized simplicity and performance. In particular, it was designed to be smaller and faster than XML.

Protocol Buffers is widely used at Google for storing and interchanging all kinds of structured information. The method serves as a basis for a custom remote procedure call (RPC) system that is used for nearly all inter-machine communication at Google.

A software developer defines data structures (called messages) and services in a proto definition file (.proto) and compiles it with protoc. This compilation generates code that can be invoked by a sender or recipient of these data structures.

CoAP / 6LoWPAN

The Constrained Application Protocol (**CoAP)** is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things.

The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

Like HTTP, CoAP is based on the wildly successful REST model: Servers make resources available under a URL, and clients access these resources using methods such as GET, PUT, POST, and DELETE. Since HTTP and CoAP share the REST model, they can easily be connected using application-agnostic cross-protocol proxies. A Web client may not even notice that it just accessed a sensor resource.

CoAP can also carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.

CoAP is designed to use minimal resources, both on the device and on the network. Instead of a

complex transport stack, it gets by with UDP on IP. A 4-byte fixed header and a compact encoding of options enables small messages that causes no or little fragmentation on the link layer. Many servers can operate in a completely stateless fashion.

The **6LoWPAN** standard (RFC 4944) has been defined by IETF to adapt IPv6 communication on top of IEEE 802.15.4 networks. 6LoWPAN refers to IPv6 over Low Power Wireless Personal Area Networks. It enables IPv6 packets communication over low power and low rate IEEE 802.15.4 links and assures interoperability with other IP devices. 6LowPAN devices can communicate directly with other IP-enabled devices.

IP for Smart Objects (IPSO) Alliance is promoting the use of 6LowPAN and embedded IP solutions in smart objects. 6LoWPAN provides an adaptation layer, new packet format, and address management to enable such devices to have all the benefits of IP communication and management. Since IPv6 packet sizes are much larger than the frame size of IEEE 802.15.4, the adaptation layer is introduced between MAC and the network layers to optimize IPv6 over IEEE 802.15.4. The adaptation layer provides mechanisms for IPv6 packet header compression, fragmentation and reassembly allowing IPv6 packets transmission over IEEE 802.15.4 links.

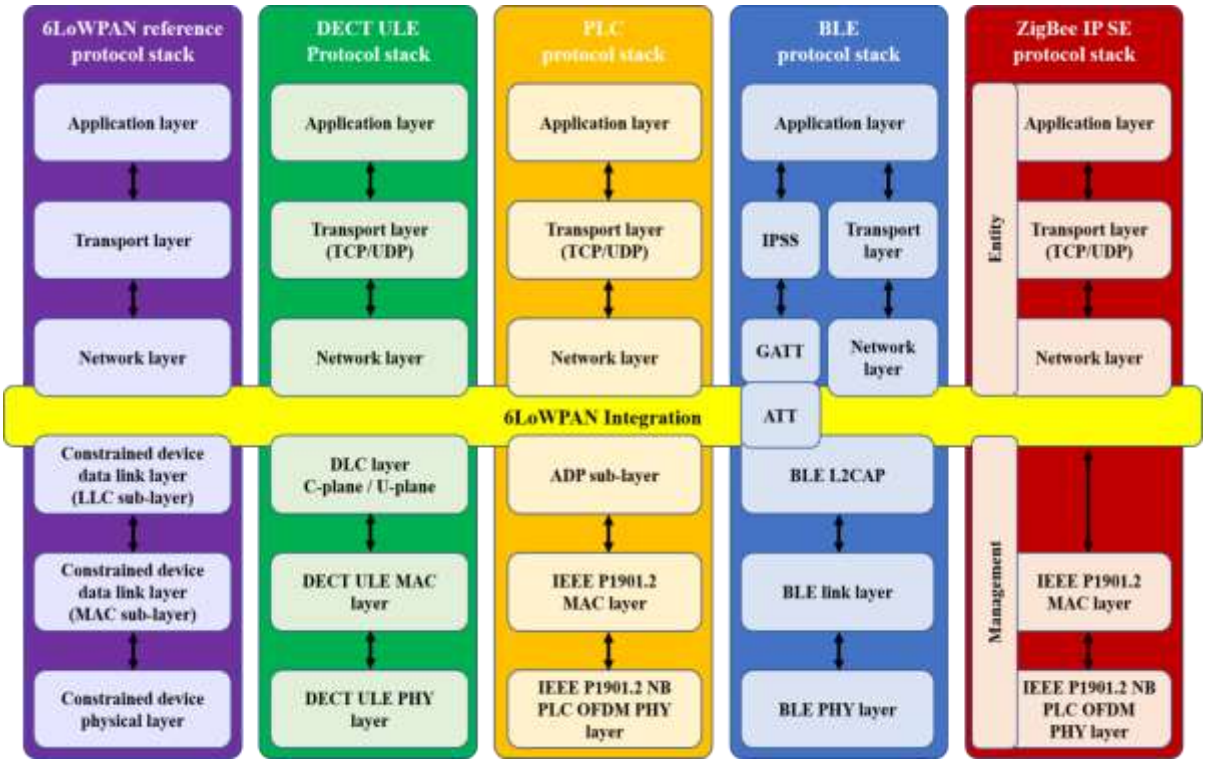

Figure 38 – 6LoWPAN integration

The fundamental difference between 6LowPAN and Zigbee is the IP interoperability of the first. 6LowPAN devices are capable of communication with other IP-enabled devices whereas Zigbee node needs an 802.15.4/IP gateway to interact with an IP network. The decision to select one standard versus another should be determined by the target application. For an application in which there is no need to interface with IP devices or the packet size is small, it is not necessary to implement 6LowPAN, which performs fragmentation [23].

Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) [24] is considered as an attractive technology for WSN applications demanding higher data rates, but short range. BLE technology enables new low-cost Bluetooth Smart devices to operate for months or years on tiny, coin-cell batteries. Potential markets for BLE-

based devices include healthcare, sports and fitness, security, and home entertainment. BLE operates in the same 2.45 GHz ISM band as classic Bluetooth, but uses a different set of channels. Instead of Bluetooth's 1-MHz wide 79 channels, BLE has 2-MHz wide 40 channels. As compared to classic Bluetooth, BLE is intended to provide considerably reduced power consumption and lower cost, with enhanced communication range. BLE allows 1 Mbps data rates with 200 m range and has two implementation alternatives; single-mode and dual-mode. Single-mode BLE devices support only new BLE connections, whereas dual-mode devices support both classic Bluetooth as well as new BLE connections and have backward-compatibility.

Zigbee
ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless M2M networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz.

The 802.15.4 specification upon which the ZigBee stack operates gained ratification by the Institute of Electrical and Electronics Engineers (IEEE) in 2003. The specification is a packet-based radio protocol intended for low-cost, battery-operated devices. The protocol allows devices to communicate in a variety of network topologies and can have battery life lasting several years.

The ZigBee protocol is designed to provide an easy-to-use wireless data solution characterized by secure, reliable wireless network architectures, also to communicate data through hostile RF environments that are common in commercial and industrial applications.

ZigBee enables broad-based deployment of wireless networks with low-cost, low-power solutions. It provides the ability to run for years on inexpensive batteries for a host of monitoring and control applications. Smart energy/smart grid, AMR (Automatic Meter Reading), lighting controls, building automation systems, tank monitoring, HVAC control, medical devices and fleet applications are just some of the many spaces where ZigBee technology is making significant advancements.

4.1.6 Syntactic and Semantic Interoperability

Syntactic interoperation should allow the harmonization of formatting and encoding structures of any exchanged information or service.

Semantic interoperability refers to the meaning of information or services, and should enable mutual understanding of interchanged information among the set of devices and services connected to the platform.

In this subsection, we summarize the interfaces and common data models of FIWARE, oneM2M, and the Watson IoT Platform. Moreover, we describe their interworking architecture. The interfaces and the data models can also be found in more detail in D1.3 [6].

FIWARE: FIWARE focuses on a common data model and powerful interfaces for searching and finding information in IoT. FIWARE IoT Platform is based on the OMA Next Generation Service Interface (NGSI) data model as the common information model of IoT-based systems and the protocol for communication. The two interfaces of NGSI data model, NGSI-9 and NGSI-10 are briefly described below. Both NGSI-9 and NGSI-10 support JSON and/or XML formats (HTTP-based).

NGSI9: it is used to manage the availability of context entity. A system component can register the availability status of context information, and later on the other system component can issue either discover or subscribe messages to find out the registered new context information. Detailed

specifications can be found in [25].

NGSI10: it is used to enable the context data transfer between data producers and data consumers. NGSI10 has query, update, subscribe and notify context operations for providing context values. A context broker is necessary for establishing data flow between different resources as well as consumers or providers. Detailed specifications can be found in [26].

oneM2M: In oneM2M, a reference point consists of one or more interfaces of any kind. The following reference points are supported by the Common Services Entity (CSE) (information included from oneM2M technical architecture document [27]). These reference points are also included in D1.3 [6].

Mca Reference Point: Communication flows between an Application Entity (AE) and a Common Services Entity (CSE) cross the Mca reference point. These flows enable the AE to use the services supported by the CSE, and for the CSE to communicate with the AE.

Mcc Reference Point: Communication flows between two Common Services Entities (CSEs) cross the Mcc reference point. These flows enable a CSE to use the services supported by another CSE.

Mcn Reference Point: Communication flows between a Common Services Entity (CSE) and the Network Services Entity (NSE) cross the Mcn reference point. These flows enable a CSE to use the supported services (other than transport and connectivity services) provided by the NSE.

Mcc' Reference Point: Communication flows between two Common Services Entities (CSEs) in Infrastructure Nodes (IN) that are oneM2M compliant and that resides in different M2M SP domains cross the Mcc' reference point. These flows enable a CSE of an IN residing in the Infrastructure Domain of an M2M Service Provider to communicate with a CSE of another IN residing in the Infrastructure Domain of another M2M Service Provider to use its supported services, and vice versa. Mcc' extends the reachability of services offered over the Mcc reference point, or a subset thereof. The trigger for these communication flows may be initiated elsewhere in the oneM2M network.

Watson IoT Platform: The information related to Watson IoT Platform interfaces and data models can also be found in Autopilot Deliverable D1.3 [6].

Watson IoT Platform is a pub/sub broker that supports the MQTT protocol for publishing and subscribing to device data.

In Watson IoT Platform, devices publish data using events. The device controls the content of the event and assigns a name for each event that is sent. When an event is received by the Watson IoT Platform from a device, the credentials of the connection on which the event was received are used to determine from which device the event was sent. This architecture prevents a device from impersonating another device.

Connecting Devices to Watson IoT Platform

Watson IoT Platform provides a HTTP API and an MQTT messaging interface. Typically, the HTTP API is used for registering and managing devices, publishing events and retrieving data. The MQTT interface allows devices to publish and subscribe to events.

A device must be registered with an organization before it can connect to Watson IoT Platform. Registered devices identify themselves to the Watson IoT Platform with a unique device identifier,

for example the MAC address, and an authentication token that is accepted for that device only.

MQTT is the primary protocol that devices and applications use to communicate with the IBM Watson IoT Platform.

oneM2M, FIWARE and Watson IoT interworking
Several interworking components such as the Semantic Mediation Gateway (SMG) and oneM2M connectors have been already defined to interface oneM2M, FIWARE and Watson IoT to facilitate system architecture extensibility, offer maximum flexibility to high level applications, and avoid vendor lock-in.

Semantic Mediation Gateway (SMG) is a component [28] which has the ability to dynamically discover semantically annotated information in the oneM2M system.

The semantic annotation may be attached to the oneM2M container resource that contains sensor readings as content instances. SMG subscribes to the sensor readings and whenever a new sensor reading becomes available, it uses its value and meta information together with the semantic annotation to create the NGSI data structure that is used to update a NGSI-based FIWARE Generic Enabler (GE), e.g. the Orion Context Broker or the Aeron IoT Broker.

Interworking Proxy Entity (IPE) is a component that can convert and integrate non-oneM2M devices and platforms (e.g., Watson IoT) to oneM2M standard. It enables seamless communication (bidirectional) between the interworking entities independently of the underlying technologies.

The following figure (Figure 39) illustrates the interworking between the three IoT Platform (oneM2M, FIWARE, Watson IoT Platform) with the two components SMG and IPE. Applications can operate on top of the FIWARE or Watson IoT Platform based on NGSI or Watson data models. Moreover, they can directly operate using MCA of the oneM2M platform.

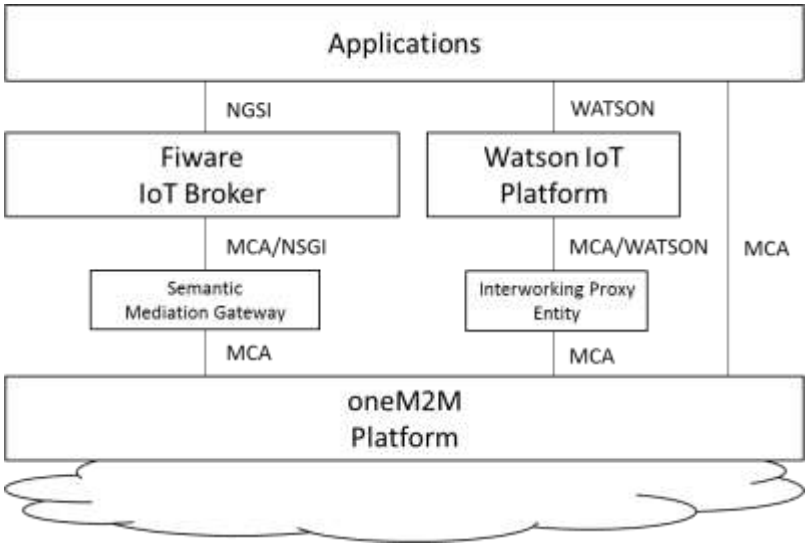

Figure 39 – Interworking through SMG and IPE

4.1.7 Application container or Runtime Environment

Java OSGI
The OSGi Alliance [29] is the promoting organization founded in 1999 by Ericsson, IBM and Oracle (afterwards extended with other members) that provided the first set of specifications to define a

framework – logic software architectures – to obtain:

- a service oriented software system
- a modular software system
- a dynamic system that allow to install, start, stop and uninstall modules at runtime

The OSGi technology is essentially designed for Java programming language, and give the possibility to build complex applications starting from basic modules (i.e. the bundles), designed keep in mind reusability, collaboration between components and flexibility. The so called *OSGi Framework* can be summarized in a modular framework designed to fulfill the increasing demand for extensible and cooperative execution of software *bundle* – the software module elements of the framework itself [30].

The OSGi framework approach and the IoT paradigm share many commons finalities. The OSGi ecosystem provides a large variety of interested stakeholders and a huge number of applications and developments tools. The affinity between OSGi and IoT emerge considering the OSGi programming model, that aim to build applications in components able to dynamically interact each other, and IoT world with the concept of *devices network* connected to Internet and each other.

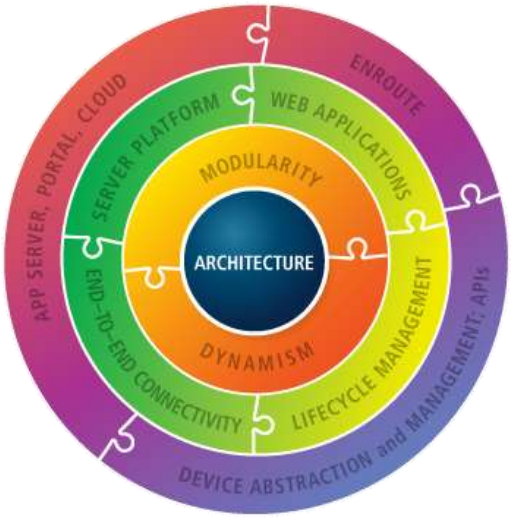

Figure 40 – OSGi and IoT similarities (source: OSGi Alliance)

Python iPOPO

iPOPO [31] is a Python-based Service-Oriented Component Model (SOCM) based on Pelix, a dynamic service platform. They are inspired on two popular Java technologies for the development of long-lived applications: the iPOJO component model and the OSGi Service Platform. iPOPO enables to conceive long-running and modular IT services in Python.

iPOPO aims to simplify service-oriented programming on OSGi frameworks in Python language; the name iPOPO is an abbreviation for injected POPO, where POPO would stand for Plain Old Python Object. The name is in fact a simple modification of the Apache iPOJO project, which stands for injected Plain Old Java Object.

The Service-Oriented Architecture (SOA) consists in linking objects through provided contracts (services) registered in a service registry.

The iPOPO framework allows developers to more clearly separate functional code (i.e. POPOs) from the non-functional code (i.e. dependency management, service provision, configuration, etc.). At run

time, iPOPO combines the functional and non-functional aspects.

ROS
ROS stands for Robotic Operating System and it is a very versatile framework with which software for robots can be developed [32]. The way ROS is built allows us to leverage the best tools developed by 3rd parties with our own tools. These can go from sensor integration, environment model and ADAS functions to vehicle control and HMI (Human machine interface). On top of this ROS also provides a fast and simple way for different processes to communicate with each other over the IP network.

4.2 Initial Specification of In-vehicle IoT platform in the different TS prototypes

For every Test Site, in this chapter the initial specifications are presented in table format. The reader has to consider the functionality entries into the tables, in relation with IoT technologies described in previous section (§ 4.1).

4.2.1 Test site Finland prototype specifications

Table 5 – Test Site Finland prototype specifications

Functionality	Test Site Finland implementation
Remote Management	DDS functionalities
Context-awareness	Vehicle world model: high-level view of the surroundings (road object, static and dynamic obstacles, etc.) as outcome of data fusion from multiple sensors
Syntactic and Semantic Interoperability	DDS is used for exchange of information in the vehicle. oneM2M will be assessed when implementing new devices in the vehicle and the cloud.
Data Management	oneM2M + DDS
IoT Device Adaptation	DDS
OEM Systems Communication	DDS + CANbus connection
IoT in-vehicle components	mobile phone as remote controller
OEM in-vehicle components	no in-vehicle OEM IoT components. Data are read from vehicle CAN-bus. Actuators are directly electronically controlled.
Application container or Runtime Environment	ROS

Communication between the different components in the vehicle is based on DDS. For each of the sensors and other inputs, including V2X input, topics are defined, making the information in real-time available to the other modules.

Table 6 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that the Intra-Vehicle Network makes available through DDS to the in-vehicle IoT platform.

Table 6 – Vehicle data (IF5 interface); Test Site Finland prototypes

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver + IMU
Vehicle Heading	GNSS receiver + IMU
Vehicle TimeStamp	NTP
Vehicle Speed	GNSS receiver
Vehicle Longitudinal & Lateral Accelerations	IMU
Vehicle YawRate	CAN bus (curvature)
Vehicle Vertical Acceleration	IMU
Gas Pedal Position	CAN bus
Steering Wheel Angle	CAN bus
Left Turn Signal Status	CAN bus
Right Turn Signal Status	CAN bus

4.2.2 Test site France prototype specifications

Preliminary specifications are reported in the annex, as they concern mostly the hardware platform. The software platform is in definition between TS France partners and their suppliers.

4.2.3 Test site Italy prototype specifications

Table 7 – Test Site Italy prototype specification

Functionality	Test Site Italy implementation
Remote Management	OSGi remote management tools
Context-awareness	Data annotation with GPS coordinates
Syntactic and Semantic Interoperability	oneM2M
Data Management	- pothole detection - other “edge” application/data fusion algorithm - Dedicated OSGi bundles implementing “edge” application/data fusion algorithm as a service (e.g. pothole detection algorithm, filtering and aggregation)
IoT Device Adaptation	- 6LoWPAN - MQTT - Extendable with other relevant protocols
OEM Systems Communication	- LCM - MQTT

Functionality	Test Site Italy implementation
	- CANbus - Extendable with other relevant protocols
IoT in-vehicle components	- Sensors for pothole detection (accelerometer, smartphone, etc)
OEM in-vehicle components	- CONTI E-Horizon (IoT connected)
Application container or Runtime Environment	OSGi framework: - Felix iPOJO - Pelix iPOPO

The functional block Application Containers are a lightweight approach to virtualization that developers can apply to rapidly develop, test, deploy, and update IoT applications at scale.

For a more detailed description of the IoT in-vehicle platform, the first choice is the use of OSGi framework. With the OSGi standard, natively, it is possible to fulfil some features (such as remote management of services). A management bundle provides different functions to organize an IoT system.

As far as the interoperability part is concerned, it should be considered that the in-vehicle IoT platform should work with heterogeneous devices, technologies, applications, without additional effort from the application or service developer.

Communication between the stacks should be based on open standards to ensure interoperability. The in-vehicle IoT platform is designed to be syntactically and semantically compliant with OneM2M standard. This has been made possible thanks to the interoperability at the communication level.

For the communication between the different OEM components, the introduction of protocols is still under definition. We are paying special attention to overhead lightness and communication speed.

Regarding the IoT device adaptation, it is planned to support different IoT communication protocols with the devices, such as CoAP/6LowPAN (which can also be used across multiple communications platforms) and MQTT (publish/subscribe-based lightweight messaging protocol for Machine to Machine (M2M) communication, on top of the TCP/IP protocol).

Regarding the specifications of the Italian Pilot Site, an important point concerns Data Management: the concept of “Virtual Sensor” is part of our IoT in-vehicle platform vision and is therefore bound to use cases. Since there is no sensor on the vehicle that "physically" detects the roadway potholes, it is thought to use a combination of sensors that can already be integrated into the OEM dispositive (e.g. accelerometers, gyroscopes, etc.) or use sensors from external devices to be placed on-board (e.g. smartphones, cameras, external accelerometers, etc.) to recover the same type of data (acceleration, orientation, etc.). In this way, data from different devices are fused together and processed: from this sensor fusion outputs the data that can be used to detect the road holes. The result of this fusion is therefore a “pothole detector”.

Also related to Data Management, it is important to consider filtering and aggregation of data before send them out from the platform.

Through the platform itself, this “virtual sensor” is exposed as a “physical sensor” outward (toward cloud based applications, road infrastructure or near vehicles).

Also Context-awareness is a key requirement in building adaptive applications and services and in annotating values from data. Based on the sensed information about the physical and environmental parameters, the sensor nodes gain knowledge about the surrounding context. The decisions that the sensor nodes take thereafter are context-aware. For our pilot site, we chose to track and annotate GPS coordinate. Therefore, it is possible to take advantage of these data from on-board sensors also to use them in the fusion sensor for pothole detection.

To conclude, the IoT in-vehicle platform is nothing more than a modular software: as far as the Italian Pilot Site is concerned, this is deployed on the On Board Unit (OBU). An IoT gateway is created that allows the possibility to communicate with the outside through IoT or ITS (LTE communication, ITS G5) standards, which can send critical information produced by the IoT system through V2V / V2X standard messages.

IoT platform can easily deployed on OBU based on different hardware platforms through a docker based approach.

Table 8 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that In-vehicle IoT Platform will get from Intra-Vehicle Network.

Table 8 – Vehicle data (IF5 interface); Test Site Italy prototypes

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	GNSS receiver
Vehicle TimeStamp	GNSS receiver
Vehicle Speed	CAN bus
Reverse Gear	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Vehicle YawRate	CAN bus
Vehicle Vertical Acceleration	CAN bus
Brake Pedal Status	CAN bus
Gas Pedal Position	CAN bus
Emergency Brake (or ABS) Status	CAN bus
Cruise Control Status	CAN bus
Adaptive Cruise Control Status	CAN bus
Speed Limiter Status	CAN bus
Steering Wheel Angle	CAN bus
Low Beam Status	CAN bus
High Beam Status	CAN bus

Signal/Parameter	Source
Left Turn Signal Status	CAN bus
Right Turn Signal Status	CAN bus
Day Time Running Lights Status	CAN bus
Reverse Lights Status	CAN bus
Fog Lights Status	CAN bus
Park Lights Status	CAN bus

Table 9 reports a list of signals/parameters related to IF6 interface, introduced in §2.2, that In-vehicle IoT Platform will get from additional IoT devices.

Table 9 –Additional IoT devices data (IF6 interface); Test Site Italy prototypes

Signal/Parameter	Source
Acceleration Status	Inertial Sensors
Motion Detection Status	Smartphone

4.2.4 Test site Netherlands prototypes specifications

4.2.4.1 TNO prototype

Table 10 – TNO prototype specifications

Functionality	Test Site Netherlands implementation
Remote Management	ROS framework
Context-awareness	Vehicle world model: high-level view of the surroundings (road object, lane markings, etc.) as outcome of data fusion from multiple sensors
Syntactic and Semantic Interoperability	oneM2M
Data Management	ROS framework
IoT Device Adaptation	ROS framework bridges (Simulink, oneM2M)
OEM Systems Communication	ROS, UDP
IoT in-vehicle components	N/A
OEM in-vehicle components	N/A
Application container or Runtime Environment	ROS framework

The TNO prototype will rely on the ROS framework as middleware for bridging vehicle control components (MATLAB/Simulink) with the IoT oneM2M platform residing in the cloud. Different ROS components (i.e., ROS nodes) will take care of handling the network interface with oneM2M (RESTful interface) and V2X communication (ITSG5). Other ROS nodes will manage the data coming from these different communication sources as well as the data coming from internal sensors to build the so-called world-model that includes object-level description of road objects, lane markings, etc. The world model data combined with application data from the IoT cloud services will be sent to control components via UDP/ROS to help in automated driving decisions. In the inverse path, control components will share internal vehicle data (e.g., acceleration, speed, etc.) to be shared with other IoT nodes via the communication interface with the oneM2M platform.

Table 11 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that In-vehicle IoT Platform will get from Intra-Vehicle Network.

Table 11 – Vehicle data (IF5 interface); TNO prototype

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	GNSS receiver
Vehicle TimeStamp	GNSS receiver
Vehicle Speed	CAN bus

Signal/Parameter	Source
Reverse Gear	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Vehicle YawRate	CAN bus
Vehicle Vertical Acceleration	CAN bus
Brake Pedal Status	CAN bus
Gas Pedal Position	CAN bus
Emergency Brake (or ABS) Status	CAN bus
Cruise Control Status	CAN bus
Adaptive Cruise Control Status	CAN bus
Speed Limiter Status	CAN bus
Steering Wheel Angle	CAN bus
Low Beam Status	CAN bus
High Beam Status	CAN bus
Left Turn Signal Status	CAN bus
Right Turn Signal Status	CAN bus
Day Time Running Lights Status	CAN bus
Reverse Lights Status	CAN bus
Fog Lights Status	CAN bus
Park Lights Status	CAN bus
Tracked front object time gap	Vehicle control system
Tracked object relative speed	CAN bus
Tracked object relative position	CAN bus

4.2.4.2 NEVS prototype

The NEVS prototype will be implemented in coordination with Brainport Test Site. When additional details will be available, specification will be updated.

Table 12 – NEVS prototype specifications

Functionality	Test Site Netherlands implementation
Remote Management	ROS framework (*In coordination with Brainport*)
Context-awareness	*In coordination with Brainport*
Syntactic and Semantic Interoperability	oneM2M (*In coordination with Brainport*)

Functionality	Test Site Netherlands implementation
Data Management	ROS framework (*In coordination with Brainport*)
IoT Device Adaptation	ROS framework (*In coordination with Brainport*)
OEM Systems Communication	dSpace, UDP
IoT in-vehicle components	N/A
OEM in-vehicle components	Adapted ECU software
Application container or Runtime Environment	ROS framework (*In coordination with Brainport*)

Table 13 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that In-vehicle IoT Platform will get from Intra-Vehicle Network.

Table 13 – Vehicle data (IF5 interface); NEVS prototype

Signal/Parameter	Source
Vehicle Speed	CAN bus
Reverse Gear	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Vehicle YawRate	CAN bus
Brake Pedal Status	CAN bus
Gas Pedal Position	CAN bus
ABS Status	CAN bus
Speed Limiter Status	Vehicle Control System
Steering Wheel Angle	CAN bus
Low Beam Status	CAN bus
High Beam Status	CAN bus
Left Turn Signal Status	CAN bus
Right Turn Signal Status	CAN bus
Reverse Lights Status	CAN bus
Fog Lights Status	CAN bus
Park Lights Status	CAN bus
Tracked front object time gap	CAN bus
Tracked object relative speed	CAN bus
Tracked object relative position	CAN bus

Signal/Parameter	Source
Distance to left road lane	CAN bus
Distance to right road lane	CAN bus
Lane/Road Curvature	CAN bus
Traffic Sign (Speed limit)	CAN bus

4.2.4.3 TUEIN prototype

The following description in Table 14 for IoT In-vehicle Platform implementation can be considered for TUEIN prototype and is also valid for TomTom vehicle (used as test vehicles) of the Nederland Test site.

Table 14 – TUEIN prototype specifications

Functionality	Test Site TU/e Netherlands implementation
Remote Management	-
Context-awareness	Probabilistic world model using camera input + IoT connectivity - WIRE - ED - Data annotation with GPS timestamp for G5 connection (CAM, DENM)
Syntactic and Semantic Interoperability	- OneM2M - ROS framework (incl. ED / WIRE)
Data Management	- ROS framework bridges (Simulink, oneM2M)
IoT Device Adaptation	- MQTT - Bluetooth - Extendable with other relevant protocols
OEM Systems Communication	- ITS-G5 (CAM, DENM) - CANbus - Ethernet/UDP
IoT in-vehicle components	n/a
OEM in-vehicle components	n/a
Application container or Runtime Environment	Linux Ubuntu RTMaps ROS Docker AD system: Speedgoat Real-Time Target (MATLAB Simulink)

Within AUTOPILOT the main challenge is to implement the vehicle as part of the Internet of Things. Two main routes are available to make a vehicle a IoT connected device. One route is to use a G5 connection. The other main route is to use a cellular 4G/5G connection. These connections can be made if gateway functionality is added to the vehicle. Figure 41 shows the general gateway layer in

IoT devices to connect to applications through a IoT transport layer. Applications can access data from several connected devices and use this information for their specific goal. The applications can reside somewhere on the web, on a mobile phone or even be part of applications in the vehicle itself.

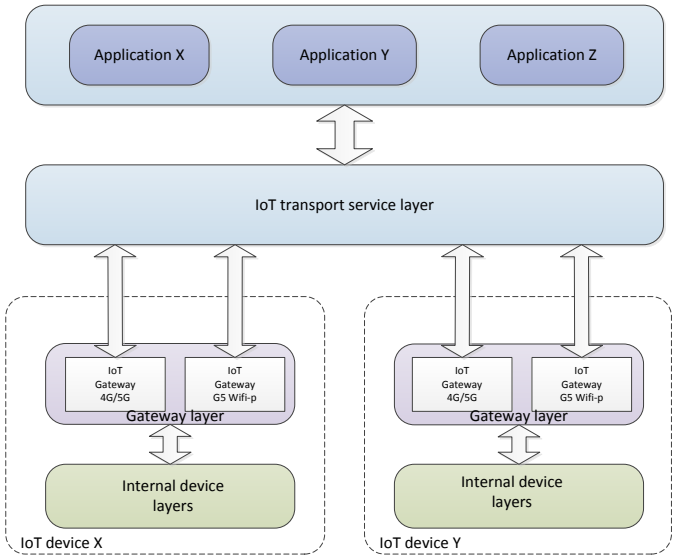

Figure 41 – General gateway layer in IoT devices

Flowradar

The Technolution on board Flowradar unit provides the G5 gateway functions in a vehicle. The system is modular by design and can also hold 3G/4G module to provide the IoT gateway to the cellular network. So the device should be able to provide even both gateway functions.

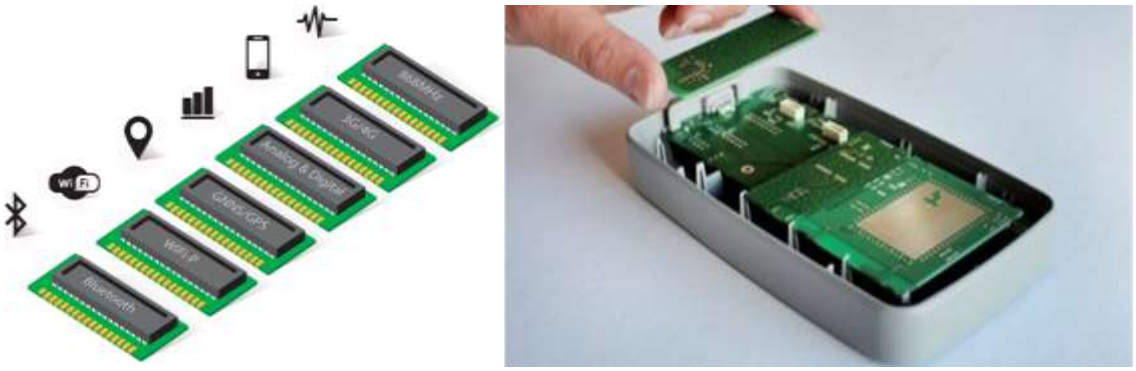

Figure 42 – Modular principle of the Flowradar G5 gateway

Basic functionality of the Flowradar is the G5 gateway for V2V and V2X communication in combination with GPS. A 4G/5G module is not yet available for the Flowradar. To be able to use the 4G gateway part in the vehicle we will use a DELL 3002 gateway with 4G functionality.

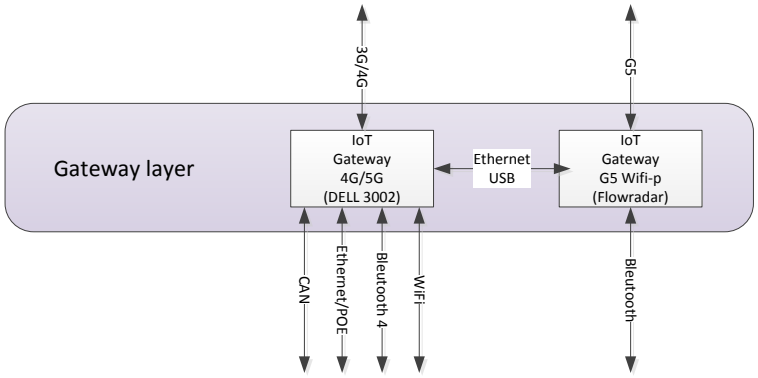

Figure 43 – Gateway layer functionality filled in with two interconnected units, one for the G5 access and one for 4G.

In car internal connectivity of the gateway layer

- Ethernet
- POE Ethernet
- Wifi
- Bluetooth
- USB
- CAN

In car external connectivity of the gateway layer

- G5 Wifi-p
- 3G/4G LTE

Gateway protocols

- G5 gateway protocols: CAM, DENM, SpaT and MAP messages according to the ETSI standards
- 4G/5G gateway protocols: OneM2M standards

Gateway additional functions

- GPS (gps time used for time synchronization of G5 messages)
- Log files/local storage
- Security

ETSI standards

For the interaction of Autonomous driving with the roadside, specific for "Vehicle to road side communication" it is needed that the V2I communication is standardized in an European format (ETSI). Information to and from the vehicle can be addressed through CAM and DENM messages; information from Traffic Lights can be addressed using SPaT and MAP messages.

Connection with OneM2M MQTT and protobuf

The gateway may also be positioned as a MQTT broker for the vehicle information. The vehicle then has its own information broker on board to share information between devices and applications with the publish subscribe mechanism.

Table 15 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that In-vehicle IoT Platform will get from Intra-Vehicle Network.

Table 15 – Vehicle data (IF5 interface); TUEIN prototype

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	GNSS receiver
Vehicle TimeStamp	GNSS receiver
Vehicle Speed	CAN bus
Reverse Gear	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Vehicle YawRate	CAN bus
Vehicle Vertical Acceleration	CAN bus
Brake Pedal Status	CAN bus
Gas Pedal Position	CAN bus
Cruise Control Status	CAN bus
Adaptive Cruise Control Status	CAN bus
Speed Limiter Status	CAN bus
Steering Wheel Angle	CAN bus
Low Beam Status	CAN bus
High Beam Status	CAN bus
Day Time Running Lights Status	CAN bus
Fog Lights Status	CAN bus
Tracked object relative speed	CAN bus
Tracked object relative position	CAN bus
Object classification	Vision system
Object relative localization	Vision system

4.2.4.4 VALEO prototype

Table 16 – VALEO prototype specifications

Functionality	Test Site Nederland implementation
Remote Management	Intempora RTMaps
Context-awareness	Vehicle world model: high-level view of the surroundings (road object, lane markings, etc.) as outcome of data fusion from multiple sensors
Syntactic and Semantic Interoperability	oneM2M
Data Management	Intempora RTMaps
IoT Device Adaptation	Intempora RTMaps + Valeo Smart Platform (HTTPS, MQTT, ProtoBuff)
OEM Systems Communication	Intempora RTMaps, UDP, CAN
IoT in-vehicle components	Misc sensors treated as IoT through Valeo Smart Platform
OEM in-vehicle components	N/A
Application container or Runtime Environment	Intempora RTMaps

4.2.5 Test site Spain prototype specifications

Table 17 – Test Site Spain prototype specifications

Functionality	Test Site Spain implementation
Remote Management	OSGi remote management tools
Context-awareness	Data annotation with GPS coordinates
Syntactic and Semantic Interoperability	oneM2M
Data Management	VRU detector Vehicle information (speed, location, …)
IoT Device Adaptation	HTTPS MQTT Extendable with other relevant protocols
OEM Systems Communication	Not considered/To be defined
IoT in-vehicle components	Sensors for VRU detection
OEM in-vehicle components	Not considered/To be defined
Application container or Runtime Environment	OSGi framework: - OSGI equinox

Application container and remote management

In the Spanish Pilot site the selected runtime environment is the OSGi framework Equinox. OSGi remote management tools are chosen to remotely provide, configure, update, monitor and start/stop the services and applications running in the IoT in-vehicle platform, see more information about OSGi in sections 4.1.1 and 4.1.7.

Context awareness and interoperability

To be aware of the context through a sort of "world model" the annotated values from sensed data are used.
The syntactic and semantic interoperation between devices and services connected to the in-vehicle IoT platform are provided using the oneM2M standard.

Data management and IoT device adaptation

The data from different devices as camera, mobile phones or its own vehicle sensors are fused together and processed working as a "virtual sensor", for instance, for VRU detection. Regarding the IoT device adaptation, the supported communication protocols are HTTPS and MQTT as basis, but other protocols could be supported to cover specific needs during the development phase.

OEM systems

The communication with OEM systems will be done through CAN and no OEM in-vehicle components will be provided.

Vehicle signals/parameters

Table 18 reports a list of signals/parameters related to IF5 interface, introduced in §2.2, that In-vehicle IoT Platform will get from Intra-Vehicle Network.

Table 18 – Vehicle data (IF5 interface); Spain prototype

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	GNSS receiver
Vehicle TimeStamp	GNSS receiver
Vehicle Speed	CAN bus
Reverse Gear	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Vehicle YawRate	CAN bus
Vehicle Vertical Acceleration	CAN bus
Brake Pedal Status	CAN bus
Gas Pedal Position	CAN bus

Signal/Parameter	Source
Emergency Brake (or ABS) Status	CAN bus
Cruise Control Status	CAN bus
Adaptive Cruise Control Status	CAN bus
Speed Limiter Status	CAN bus
Steering Wheel Angle	CAN bus
Low Beam Status	CAN bus
High Beam Status	CAN bus
Left Turn Signal Status	CAN bus
Right Turn Signal Status	CAN bus
Day Time Running Lights Status	CAN bus
Reverse Lights Status	CAN bus
Fog Lights Status	CAN bus
Park Lights Status	CAN bus
Tracked front object time gap	Vehicle control system
Tracked object relative speed	CAN bus
Tracked object relative position	CAN bus
Doors status	CAN bus
Seat occupation	CAN bus

5 Conclusion

In this document D1.5, planned in Month 9, we provided the in-vehicle IoT platform initial specifications. We agreed on the logical representation of the in-vehicle system, identifying what is the role and placement of the in-vehicles IoT platform and the main in-vehicle components interacting with it. We then defined the in-vehicle IoT platform, starting from its top level needs and functionalities, then outlining the functional architecture and identifying a list of requirements.

Already in the first steps, it became evident that AUTOPILOT is adopting a federated approach, i.e. prototype owners plan to use different sets of software solutions for the in-vehicle IoT platform. In order to give the reader an overview, we presented this selected set of IoT tools, frameworks and interfaces, organized by functionality. Then we asked each prototype partner to specify which of these are going to be implemented in its vehicle. For most of the prototypes, we also have initial specifications of the data, coming from the vehicle network and/or from on-board IoT components, which are made available to the in-vehicle IoT platform. In any case, we identified a basic rule for vehicle data: the dataset used for ITS Cooperative Awareness Message should also be available to the in-vehicle IoT platform. However, they should be properly managed to fulfill both syntactic and semantic interoperability within the IoT, and the context awareness needs (e.g. dynamic data characteristics and update rate of the world model, etc.).

This document D1.5 constitutes the basis for WP2, in particular for tasks 2.1 and T2.2 and T3.1. Some specifications are still unavailable due to pending verification; they will be completed within T2.1 development phase and in T3.1 pilot site specification. In any case, T1.3 will be following the definition and ensure that final specifications are reported in D1.6, planned in Month 30.

6 ANNEXES

6.1 Additional In-vehicle system specifications

6.1.1 VEDECOM prototype

"IoT Platform" and network in the car

The following description is preliminary and may change during the project, depending on constraints.

The description of the IoT Platform is new and different from any other previous OBUs used in VEDECOM cars. The main distinctions are in the size of the platform (smaller), the number of interfaces (more, e.g. 3 802.11-OCB interfaces instead of just 1), and the inclusion of a dedicated module for 4G that is a separated linux to ensure higher reliability.

The "IoT Platform" term is the equivalent of the term "On-Board Unit" used in other contexts, but:

- Whereas typical OBUs are made of a single computer (a single linux), for the VFLEX car the IoT Platform is made of at least two distinct computers: the YoGoKo box and the Sierra mangOh Red board.
- The Iot Platform (IoTP) coordinates the connectivity of devices to each other, to OEM sub-systems and to an external network. The IoT vehicle platform can be considered as the automotive counterpart of the IoT Gateway component of an IoT infrastructure.

The in-car internal network links together the various computers with an IP network. In the following figure we picture the application-specific boxes and their means of communications, such as Ethernet cables, USB and antennas.

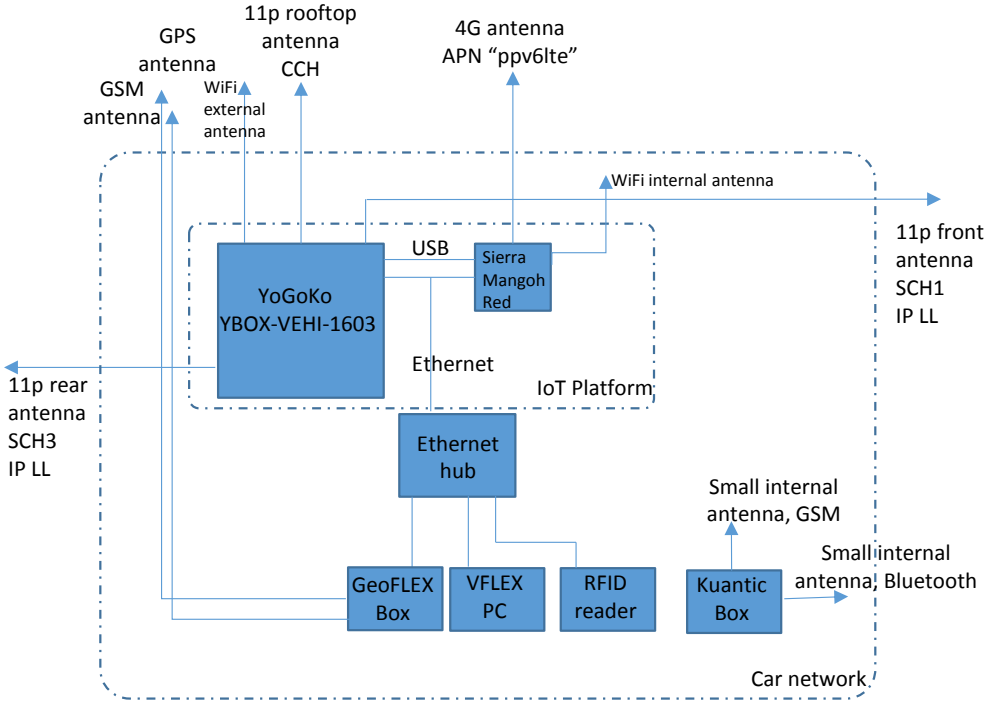

Figure 44 – Illustration of the car network to be set inside VFLEX, with IoT Platform and other computers

The way these devices are connected together may change.

The relevant IP-capable computers are the following:

- YoGoKo box for 802.11 OCB connectivity
- Sierra mangOh Red, for 4G connectivity
- Ethernet hub
- GeoFLEX box, for precise localisation
- VFLEX PC running collaborative perception algorithms
- RFID reader to read external RFID devices
- Kuantic box for access control to the car (open or lock car)

The current list is subject to change, by adding new computers for other applications, or removing some of the depicted computers.

The IoTP consists of:

- YoGoKo YBOX-VEHI
- Sierra mangOh Red

The picture of the YoGoKo YBOX-VEHI is the following:

Figure 45 – Photography of the YoGoKo YBOX-VEHI-1603

YBOX-VEHI-1603 may be used for operational deployments of V2X communications and Internet connectivity on vehicles, requiring low power consumption and state-of-the-art security features. This platform provides support for a diversity of access technologies (3G/4G, WiFi, Vehicular WiFi), enabling an ultra-connected vehicle. The list of the features of the YBOX is the following:

Product reference	YBOX-VEHI-1603
Platform	ARM (Freescale iMX.6, 1GB RAM)
Storage	16 GB
Type of 802.11p chipset	Autotalks
Number of 802.11p radios	2
3G/4G/4G+	3G/4G
External WiFi	802.11a/b/g/n
Internal WiFi	802.11a/b/g/n
Ethernet ports	2 ports 1Gbps
USB ports	1 port USB
CAN bus support	(option)
GNSS support	Yes
IO support	(option)
Hardware Security Module support	Yes
Automotive-grade enclosure	Yes
Dimensions	220 x 160 x 80 mm
Weight	1 kg
Power feed	12V/24V
Power consumption	~20W

Figure 46 – List of feature of the YoGoKo YBOX-VEHI-1603

The goal of the YoGoKo YBOX-VEHI is to offer 802.11-OCB connectivity to traffic lights, and to front

and rear cars. It can potentially receive GPS signal, and has some WiFi, Bluetooth, 802.15.4 and Lora features

The other part of the IoTP is the Sierra mangOH Red. The mangOH Red is an open source hardware platform designed for low-power IoT use-cases in a small form factor (69mm x 70mm).

The photography of the mangOH Red board is the following:

Figure 47 – Photography of the mangOH Red board

The characteristics of the Sierra mangOh Red module are the following:

- 4G module, with potential of evolution to high bandwidth Categories, future V2X, and more
- Ethernet module
- On-board WiFi interface as client or access point, 802.11ac
- Bluetooth interface
- GPS interface
- SIM card holder
- Linux based on yocto, and Legato platform.

The goal of the mangOH Red board is to offer IP connectivity to the car in a wide variety of geographical situations: wherever cellular connectivity is available, the mangOH red will connect the car to the Internet.

Both the YoGoKo YBOX-VEHI and the Sierra mangOH Red module support native IPv4, IPv6 and a combination of them.

Car antennas

The IoTP is essentially a mobile platform; as such it uses a number of antennas for various purposes. There are potentially 9 external and internal antennas for different applications:

- Dedicated 802.11-OCB 802.11 OCB antennas; one antenna for connectivity to the front vehicle, one for rear vehicle, and one for Road-Side Unit connectivity; this corresponds to three distinct IP interfaces in the YoGoKo YBOX-VEHI.
- 4G antenna or multiple 4G antennas for one interface, for MIMO
- Antennas for precise localisation offered by the GeoFLEX box
- In-car internal antennas for WiFi, Bluetooth and potentially NFC
- RFID reader antenna for reading the RFID deployed outside the car for POI or other application helping the driving automation.

7 References

[1] AUTOPILOT Deliverable D1.7, “Initial specification of Communication System for IoT enhanced AD”

[2] AUTOPILOT Deliverable D1.1, “Initial Specification of IoT-enabled Autonomous Driving use cases”

[3] Levels of Driving Automation defined by SAE © International standard J3016, 2014

[4] AdaptIVe European Project, www.adaptive-ip.eu, Deliverable D2.1 “System Classification and Glossary” available on website.

[5] Dreams4Cars European Project, www.dreams4cars.eu, Deliverable D1.1 “Application domain requirements” available on website

[6] AUTOPILOT Deliverable D1.3, “Initial IoT Self-organizing Platform for Self-driving Vehicles”

[7] ETSI TS 102 637-2, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service

[8] ETSI TS 102 637-3, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service.

[9] ETSI ITS G5 documents: http://www.etsi.org/technologies-clusters/technologies/intelligent-transport

[10] TNO website: https://www.tno.nl/nl/over-tno/nieuws/2016/4/geslaagde-demonstratie-automatisch-en-cooperatief-rijden-aan-eu-transportministers/

[11] VW Tiguan: https://www.volkswagen.co.uk/assets/common/pdf/brochures/tiguan-brochure.pdf

[12] MQB platform: https://en.wikipedia.org/wiki/Volkswagen_Group_MQB_platform

[13] OSGi Remote Management Tools: https://wiki.eclipse.org/OSGi_Remote_Management_Tool

[14] LCM: https://dspace.mit.edu/bitstream/handle/1721.1/46708/MIT-CSAIL-TR-2009-041.pdf

[15] API provided by LCM: https://lcm-proj.github.io/

[16] ZeroMQ: http://zeromq.org/whitepapers:architecture; http://www.aosabook.org/en/zeromq.html;

[17] AMQP: https://www.amqp.org/; https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol https://www.amqp.org/product/architecture

[18] OASIS: https://www.oasis-open.org/

[19] OCPP : https://www.oasis-open.org/committees/download.php/59590/

[20] MQTT: https://www.iso.org/standard/69466.html http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

[21] Protocol Buffers (ProtoBuf): https://developers.google.com/protocol-buffers/docs/overview

[22] DDS: http://portals.omg.org/dds/what-is-dds-3/

[23] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 packets over IEEE 802.15.4 networks, IETF RFC 494

[24] Bluetooth Low Energy (BLE): http://www.bluetooth.com/Pages/Low-Energy.aspx/
[25] "FI-WARE NGSI-9 Open RESTful API Specification", FIWARE Forge, 2017, to be retrieved via, https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-9_Open_RESTful_API_Specification
[26] “FI-WARE NGSI-10 Open RESTful API Specification", FIWARE Forge, 2017, to be retrieved via, https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI-10_Open_RESTful_API_Specification
[27] "Functional Architecture", OneM2M, TS-0001-V2.10.0, August 2016, to be retrieved via: http://www.onem2m.org/images/files/deliverables/Release2/TS-0001-%20Functional_Architecture-V2_10_0.pdf.
[28] Kovacs, Erno, et al. "Standards-Based Worldwide Semantic Interoperability for IoT." IEEE Communications Magazine 54.12 (2016): 40-46.
[29] OSGi Alliance: https://www.osgi.org/
[30] OSGi architecture: https://www.osgi.org/developer/architecture/
[31] Python iPOPO: https://ipopo.readthedocs.io/en/latest/
[32] ROS: http://www.ros.org/about-ros/