Grant Agreement Number: 731993

Project acronym: AUTOPILOT

Project full title: AUTOmated driving Progressed by Internet Of Things

D.1.6

FINAL OPEN IOT VEHICLE PLATFORM SPECIFICATION

Due delivery date: 30.06.2019

Actual delivery date: 30.09.2019

Organization name of lead participant for this deliverable: CRF

Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential , only for members of the consortium (including the Commission Services)	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731993.

AUTOPILOT

Document Control Sheet

Deliverable number:	D1.6
Deliverable responsible:	Visintainer Filippo , CRF
Workpackage:	WP 1
Editor:	Visintainer Filippo, CRF

Author(s) – in alphabetical order		
Name	**Organisation**	**E-mail**
Visintainer, Filippo	CRF	filippo.visintainer@crf.it
Galli, Mauro	CRF	mauro.galli@crf.it
Bosi, Ilaria	LINKS	ilaria.bosi@linksfoundation.com
Brevi, Daniele	LINKS	daniele.brevi@linksfoundation.com
den Ouden, Jos	TUE	j.h.v.d.ouden@tue.nl
Sousa Schwartz, Ramon	TNO	ramon.desouzaschwartz@tno.nl
Schreiner, Floriane	VEDECOM	floriane.schreiner@vedecom.fr
Marcasuzaa, Hervé	VALEO	Herve.marcasuzaa@valeo.com
Scholliers, Johan	VTT	Johan.Scholliers@vtt.fi
Petrescu, Alexandre	CEA	alexandre.petrescu@cea.fr
van den Brand , Jan Willem	TT	JanWillem.vandenBrand@tomtom.com
Balraj, Marimuthu	NEVS	balraj.marimuthu@nevs.com

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V0.1	28/01/216	Basis (D1.6) Task assigment	CRF
V0.2		LINKS: IoT standards used; TNO: minor modifications on description; VEDECOM: use case car rebalancing refined description; prototypes nr; specs of TS France prototype; VALEO: updated prototype specs; CEA: IoT updates in French TS; VTT: updates TS Finland; TT: updates on TT prototypes; NEVS: updated Chapter 1.4.4.2 and table 15; CRF update prototype.	LINKS, TNO; VEDECOM, VALEO, CEA, VTT, TT,NEVS, CRF
V1.2		Modif. CEA, VEDECOM, TU\e	CEA, VEDECOM, CRF
V1.4		Version for peer review	CRF
V1.5	20/09/2019	Amended version after peer review	CRF, addressing comments and integrating modifications suggested by three peer reviewers: Daniele Brevi (Links), Mariano Falcitelli (CNIT), Sadeq Zougari (AKKA)
V2.0	30/09/2019	Final formatting for submission	ERTICO

Abstract

This document reports the final vehicle IoT platform, as defined and demonstrated in the AUTOPILOT EU project. The Open IoT Vehicle platform has been deployed in car prototypes, enabling Autonomous Driving functions and Use Cases. D1.6 is an upgrade of the preliminary specifications (D1.5) based on the implementation, as final 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
MQTT	Message Queuing Telemetry Transport
OEM	Original Equipment Manufacturer
PF	Platform
PS	Pilot Site
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
WSN	Wireless Sensor Network

Table of Contents

List of Figures

List of Tables

1 Executive Summary

AUTOPILOT Task T1.3 was aimed at identifying the specifications of the Open IoT vehicle platform. This includes the definition of the format to be used for exchanging data between the in-vehicle proprietary network and the IoT platform components embedded into the vehicle itself, enabling the possibility to develop different OEM-specific gateways thanks to the openness of the resulting architecture. The functionalities of the IoT platform were also specified taking into account AD functions and use-cases, considering the adaptation required for their implementation as well. Moreover, the possibility to introduce additional sensors and components into the existing vehicle architecture was analysed, with a particular emphasis on the new generation connectivity technologies which are not yet available on the market.

The AUTOPILOT has developed IoT-architectures and platforms which will bring Automated Driving towards a new dimension. AUTOPILOT IoT enabled automated driving cars have been tested in real conditions in the pilot sites Tampere, Versailles, Livorno, Brainport, Vigo as well as the Korean pilot site. The latter is not treated here, as from initial discussions, it did not have a specific in-vehicle IoT platform, but rather V2X connected with IoT infrastructure.

Task T1.3, devoted to the in-vehicle IoT platform definition, delivered initial requirements and of the IoT in-vehicle platform (D1.5 [1]) in the first project phase. In this last phase, after integrating the IoT in vehicle (D2.1 [3]), testing and evaluating the prototypes (WP3, WP4) task 1.3 has updated the specifications, keeping however the main aspects of the in vehicle IoT platform as defined in D1.5.

A vehicle IoT platform can be seen as an aggregation point for sensors and actuators which extends the vehicle functionalities provided by OEM equipment. It coordinates the connectivity of these devices to each other, to OEM sub-systems and to the external networks. From a logical perspective, it acts as the IoT Gateway component of an IoT infrastructure.

The work within WP1 and then in WP2 progressed as planned: the IoT platform prototypes were integrated in the vehicle; demonstrators have been used in the Test Sites for the intended use cases. The differences from the initial specifications are related to specific test site implementation and are generally due to practical reasons, keeping the IoT architecture defined in AUTOPILOT in line with the original plan. Therefore, the original D1.5 structure has been maintained, and reflects the workflow followed in the task: starting from target use cases, outlining test site implementation, focusing on car prototypes, giving the general representation of the IoT platform and providing the specifications. The main differences are included and highlighted in the sections describing the prototypes for the different Test Sites and the specifications.

Chapter 1 presents target use cases that have been tested in the pilots, their technological maturity and, where applicable, the future plans.

Chapter 2 reports the basic functionalities and requirements of the Vehicle IoT platform.

Chapter 3 reports the final specifications of the In-Vehicle IoT platform for each pilot site.

2 Overview of the In-Vehicle IoT Platform

2.1 Vehicles become part of the IoT

Nowadays, smart cities, roads and highways 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 IoT system with geographically distributed endpoints, generating a huge volume of data streams over time. Potentially, big data can help us increase efficiency in various domains such as transportation, safety, and environment. In the Autonomous Driving (AD) area, IoT gives new capability to the in-vehicle system especially in environment perception, enhances the existing AD functions such as valet parking, highway pilot, platooning and empowers current sensing systems such as pedestrian detection in a city. It also enables efficient car sharing services, by providing up-to-date information. In order for the vehicle to be part of an IoT eco-system, it needs to be equipped with an In-Vehicle IoT platform, a hardware/software gateway that connects the existing in-vehicle system with other entities on board (car zone), in the vicinity (cooperation zone) or in a larger area (Smart City).

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

Automotive IoT differs from C-ITS, in that Vehicle-to-Everything communication (V2X) is "given" as a link to collect information from different sources, while the research focus is on the representation and management of exchanged data at application level within an "IoT World", in order to benefit from its characteristics (abstraction, scalability, etc.). A major benefit coming from IoT is the wider availability of information, as it enables a data fusion between the vehicle own sensors (cameras, radars, LiDARs, etc.) and external information, to consolidate the so-called local dynamic map (LDM). The LDM stores objects describing the current situation of a vehicle, position, obstacle, neighbouring vehicles, points of interest and road events. This up-to-date environment representation gives the needed information redundancy for higher SAE levels of automation [6]. The IoT approach helps to have scalable management of all these new sensors enriching this LDM.

The overall AUTOPILOT IoT architecture, defined in Task 1.2, starts from the "things" (including AD cars), continues with the network layer, IoT layer, and ends with AUTOPILOT applications. Specifically, the IoT layer can be physically located in the cloud or on the edge (e.g. RSU). It covers functionalities such as context management, analytics, IoT device management, semantics, processing and service optimization and so on. In this architecture, vehicles are part of the "things" layer.

The in-vehicle IoT platform allows the vehicle to become a "Thing" part of the "IoT World". To achieve this, several possible software & hardware solutions are possible and can be applied based on the specific implementation choices. Vehicles also receive IoT information through specific

devices that do not represent a vehicle platform (e.g. IoT connected electronic horizon, V2X radio transceivers, telematic on board units) but these devices alone are out of the scope of this D1.6.

2.2 IoT applied to Autonomous Driving and Mobility

Hereafter, we provide a brief summary of use cases enabled by the integration of a vehicle with the IoT World [5].

2.2.1 Use Case: 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/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 in 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 a 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 by the vehicle, and if the request is valid, start the collecting process.
- At the collect point, the vehicle must validate that access to the vehicle is provided to the authorized driver.

2.2.2 Use Case: 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 (puddles, roadworks, 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 Driving anticipation" mechanism aims at replicating people driving experience and road awareness into autonomous vehicles.
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 to:

- transfer, from the Vehicle to the Cloud, all likely anomalies detected by IoT sensors within the Vehicle.
- transfer, from cloud to vehicle, Live Maps information provided by the Maps provider.
- transfer from cloud to vehicle, Road Hazards alerts emitted by the Cloud service.

The retrieval into the vehicle of ADAS adaptation instructions published by the Control Centre.

2.2.3 Use Case: platooning

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

A few variants of platooning are 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 20 km/h. The scenario to be implemented in Versailles will start from one of the AUTOPILOT car sharing station where the driverless vehicles will join the leading vehicle to form a platoon. The platoon will then move to the other car sharing station, where automated parking will also be used to position the vehicles on the foreseen parking slot.
- 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 timely on

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 the confidence of detected dynamic objects and traffic information by providing redundant sensor information.

The interaction between vehicle and external entities via the 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, decision to cross an intersection and speed recommendations are sent from the platooning service (Traffic Light Assist Service) to the vehicles in order to keep the platoon formed in intersection scenarios.
- After entering the highway, vehicles in the platoon send a 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 a member of a platoon also send their position, velocity and planned route to IoT services, e g., 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 the reliability of each vehicle's local world model.
- Information aiming for (redundant) localization can be exchanged between the vehicle and IoT services, e.g., real-time HD maps and GPS error corrections with RTK DGPS services.

2.2.4 Use Case 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 of the AD cars with traffic lights and real-life traffic,
- the robustness of the AD functions of the vehicle,
- the safety issues when vulnerable road users are involved,
- the positioning enhancement.

Indeed, in order to enhance the performances of AD functions in the 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.
- data shared from VRUs 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, the 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 (faraway 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 accomplished 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 are a fundamental requirement of this use case, more details about this syntactic and semantic interoperation are available in section 4.1.6.

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, smartwatches, etc.

2.2.5 Service: 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. It 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. It can also be intended as ride sharing, where multiple customers possibly having different origins and destinations, share a part of the ride on a common car. 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 matches vehicles with customers' requests of origin and destination locations and several other requests (ride alone or with somebody, possibly time-windows.

It utilizes 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.

The focus in AUTOPILOT 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 in Figure 2.

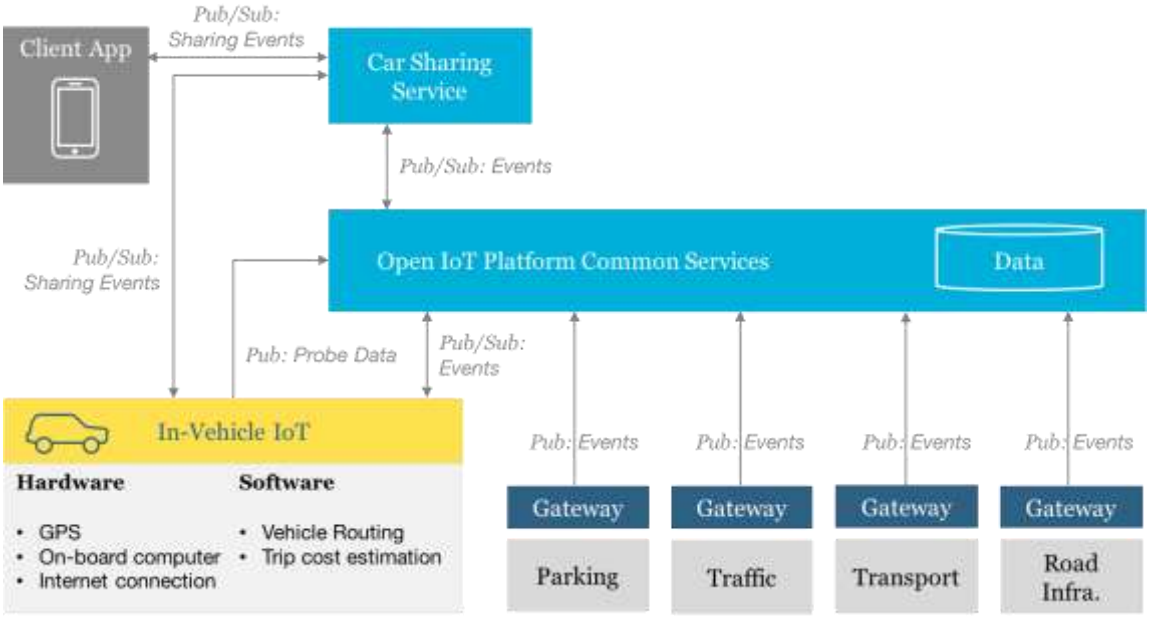

Figure 2 – Car sharing use case architecture

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.

2.3 In-vehicle IoT platform within the on-board system

The vehicle itself can be considered in two ways: (1) as an IoT device, but also (2) as an edge computing unit and gateway for other IoT devices. It is a mobile node within the whole eco-system, producing and consuming content from and to the general IoT Platform.

The IoT Platform is composed of central units (cloud) and distributed edge computation nodes (edge) and creates a unifying view of the IoT entities. IoT enabled applications use the interface to the IoT Platform in order to interact with IoT entities; this represents the aforementioned option (1), i.e. of applications running outside the vehicle, the latter being a IoT entity. But applications can also run in the in-vehicle platform, this case corresponding to option (2). The “IoT Platform” is the set of functions that manages the IoT devices and entities, while we define “In-Vehicle IoT Platform” the complex entity that includes all the software and hardware components deployed in the vehicle. In

general, and IoT Application is deployed partially in the vehicle, as some information is only available and used locally (edge), and has a cloud counterpart to exchange service related information. A closer look into the vehicle platform is presented in Chapter 3 , where the vehicle functional architecture of AUTOPILOT is presented.

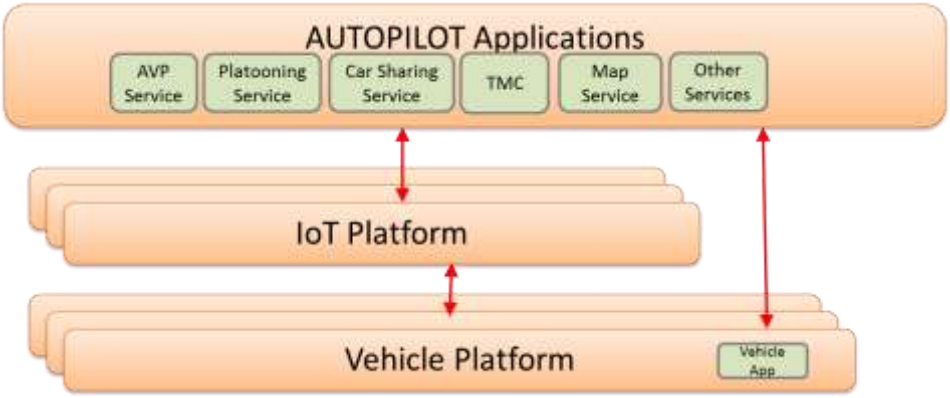

Figure 3 – IoT High View Architecture: conceptual separation in AUTOPILOT

The scheme of Figure 4 shows the logical placement of the In-Vehicle IoT platform within the vehicle components, to receive information from existing (Sensors) and next generation (C-ITS) components and to connect with the cloud IoT system but also to legacy cloud services.

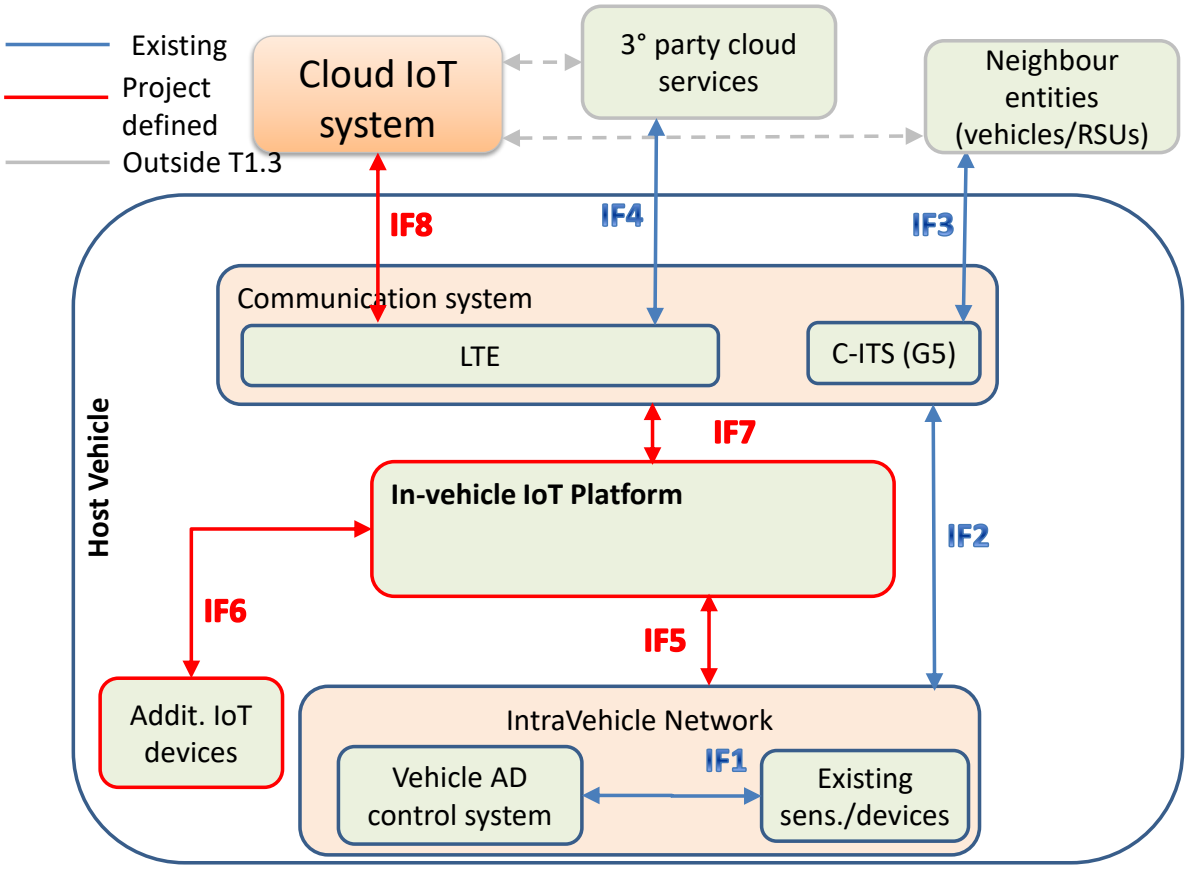

Figure 4 – In-Vehicle IoT platform (red box) with the vehicle concept scheme

The next table outlines the interfaces among components, i.e. what kind of data are exchanged in order to show which are the new interfaces to be considered for the IoT connectivity.

Table 1 – Interface between main components - high level description

ID	Existing/ New	Component 1	Component 2	Data / short description: 1 to 2	Data / short description: 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*
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 additional sensors / *enabling sensor functionalities (if needed)*
IF7	New	Vehicle IoT platform	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*

A closer look into the vehicle platform is presented in Chapter 3, where the vehicle functional architecture of AUTOPILOT is presented.

2.4 Proof-of-concept: AUTOPILOT IoT and AD vehicle prototypes

AUTOPILOT has deployed several prototypes to show how the in-vehicle IoT platform can be interfaced to in-vehicle components and connected to the IoT eco-system, to enable the pilot use cases.

2.4.1 AD vehicle prototypes in Tampere Pilot Site

The main differences from the initial plan, regarding the IoT platform, is that the vehicle does not use ITS-G5 in the use cases, and ITS-G5 was hence not integrated into the vehicle platform. The communication interfaces to the IoT platform uses available LTE channels.

This research vehicle prototype has been demonstrated in controlled environments, and has been tested on public roads. The software is still in prototype level but compiling to a licensed package has been started. The Technology Readiness Level (TRL) is 7 “system prototype demonstration in operational environment” [2].

VTT provided one automated vehicle for AUTOPILOT project, 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

The following use cases were demonstrated in Finland TS in Tampere:

- Urban driving: intersection support. Traffic signal data are collected in real time from the traffic signal server of the city of Tampere Potential conflicts, observed by traffic cameras, will be transmitted to the vehicles.
- Automated valet parking: a traffic camera installed at the parking lot assists the vehicle in the parking manoeuvre.

Figure 6 shows the in-vehicle architecture of the prototype. The vehicle scheme is organized in the following component groups:

- Communication devices: communication between the vehicle and backend systems is

through cellular (3G/4G) communications.
    - Cellular interface for communication to IoT clouds.
- Sensing devices:
    - RTK-GPS sensor and additional sensors, such as odometer and IMU.
    - Environmental perception sensors, including radar, LIDAR and cameras, as described above and shown in Figure 5.
    - Vehicle CAN-bus interface.
- On board Units:
    - AD unit: set of devices responsible for the real-time functions, including fusion of the data of 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.
            - Detection of VRUs at pedestrian crossings.
        - Automated valet parking
            - Control of the vehicle: destination (parking place, pick-up point), and route, allowing to start manoeuvre.
- 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.

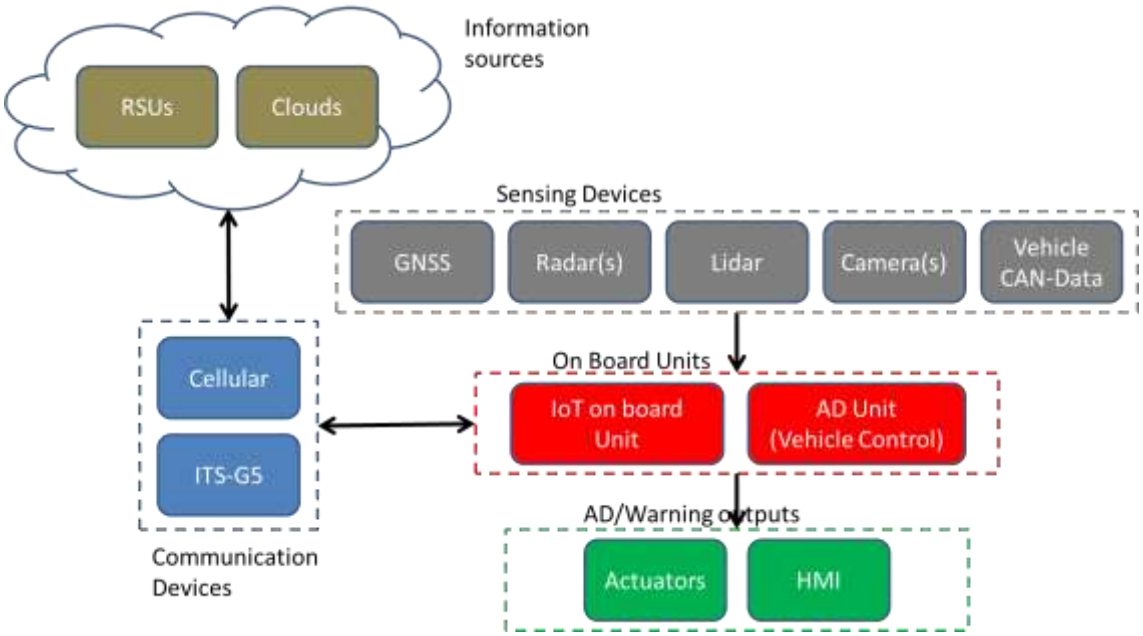

Figure 6 – In-vehicle architecture of the Finnish prototypes

2.4.2 AD vehicle prototypes in Versailles Pilot Site

VEDECOM achieved a fleet of 3 vehicles (VFLEX prototypes) for the French pilot site. VFLEX is a SAE L4 prototype made from a Renault Twizy which has been modified in order to perform autonomous driving uses cases.

Figure 7 – Image of a French prototype

First, thanks to VEDECOM's collaboration with Renault, the Twizy has been transformed into an open robotic platform (with 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.

The prototype system as a whole has achieved a TRL of 6 and will be integrated into the roadmap of future activities of connected and autonomous vehicles. The VFLEX can be part of a global mobility service managed by a central control center, including different types of vehicles such as robot taxis, shuttles and urban pods.

The embedded IoT platform (IP-OBU) delivered by CEA (the component used for communication aspects) is compliant to TRL 7: system prototype demonstration in operational environment.

The whole system has been tested and assessed in Versailles City during Urban Driving and Platooning use cases piloting sessions, in real-world traffic, without dedicated or protected lanes.

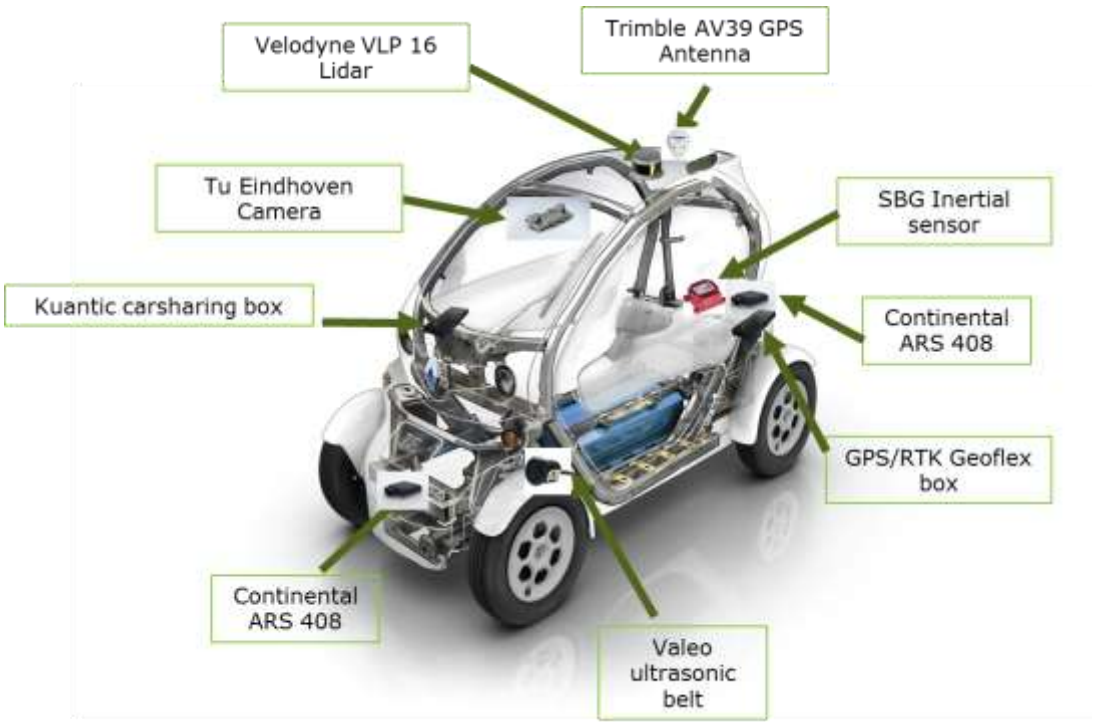

Figure 8 – Overview of VEDECOM prototype sensors

The basic driving 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) has been provided by TU Eindhoven. Figure 9 shows how partners are involved on the VLEX platform.

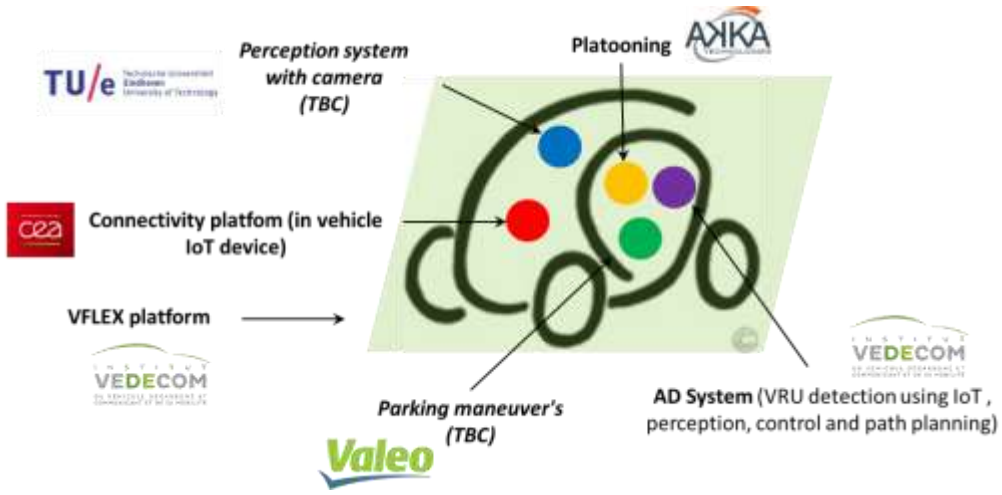

Figure 9 – Overview of synergy for French prototype

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, …).

- 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…).
- 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…).
- Acquire driver's commands and inform the driver (HMI).
- Communicate with the environment (with traffic lights, road users…).
- Communicate with other vehicles.

A global diagram of the French prototypes is presented in Figure 10.

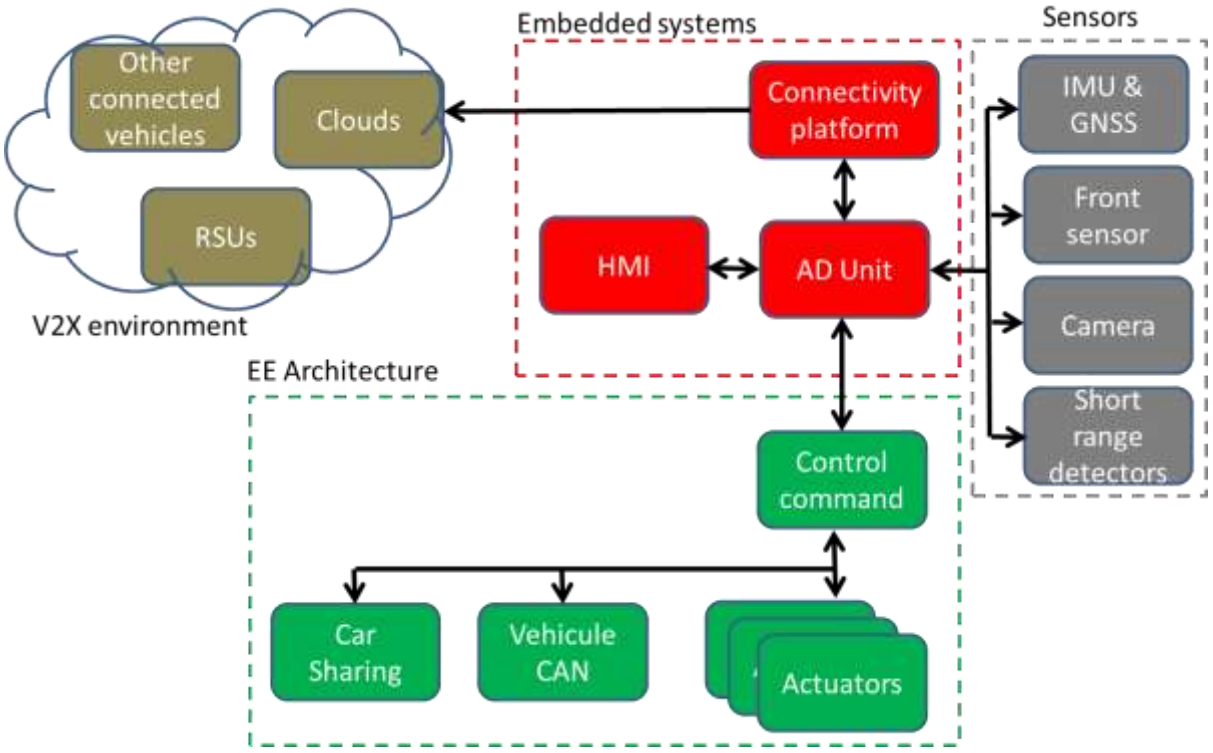

Figure 10 – In-vehicle architecture for French prototype

"IoT Platform" and network in the car

The following description corresponds to the current state of the embedded IoT platform deployed in Versailles PS.

The description of the IoT Platform is different from the description given in the deliverable D1.5. In fact, formerly a YoGoKo OBU was considered to handle the in-vehicle communication. However, since then, the hardware has been changed with a smaller device, more easy to be integrated in the VFLEX.

In each VLFEX, the "IoT Platform" includes a Gateworks branded single board computer (SBC) of the Ventana SBC Family. It provides a multi-interfaces board with:

- Ethernet connection to handle the IPv4/IPv6 configuration of the PC-AD;
- WiFi (802.11a/g/n) connection to handle WiFi hotspot;
- 4G (LTE) connection to handle the connection to the Internet and to the cloud services;
- IEEE 802.11-OCB connection to handle Vehicle-to-Vehicle and Vehicle-to-Infrastructure and Vehicle-VRU (Vulnerable Road Users) communication.

Respect to the YoGoKo OBU, the number of 802.11-OCB interfaces is still three (3). But, the cellular modem (4G) is now integrated inside the Ventana board instead of being integrated by means of an external dedicated module.

The in-car internal network links together the various computers with an IP network (see Figure 11). In the Figure 12 the application-specific boxes and their means of communications, such as Ethernet cables and antennas, are shown.

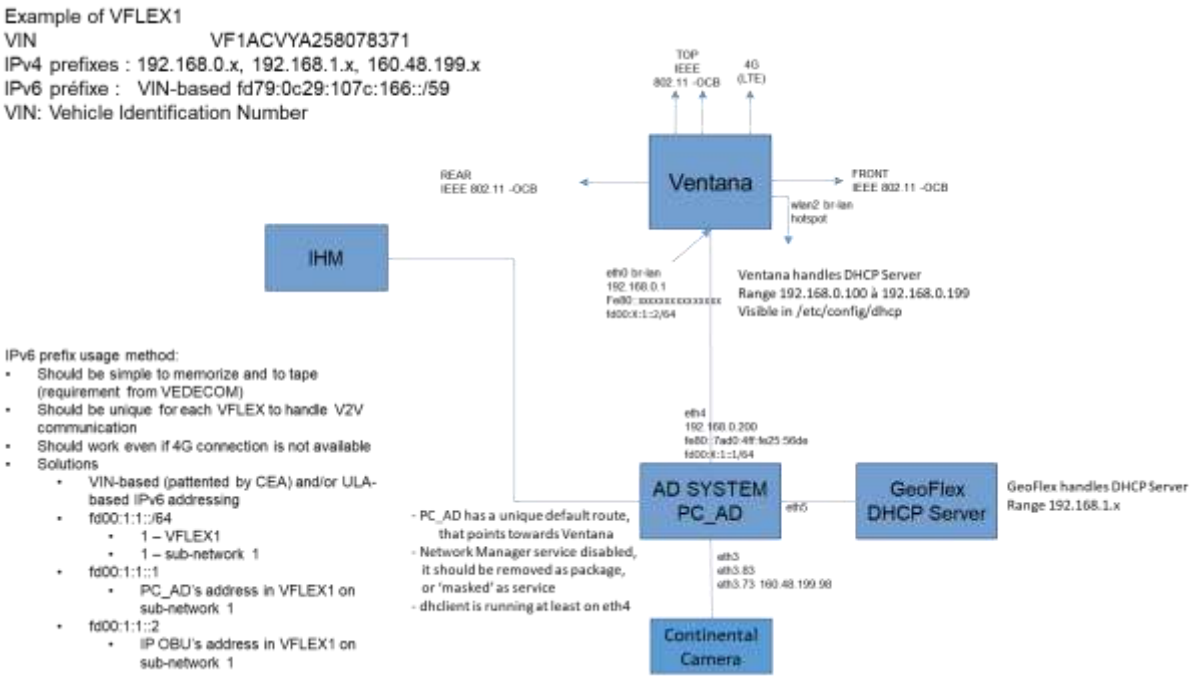

Figure 11 – Illustration of the car Network set inside the VFLEX, with IoT Platform and others computers

The Ventana board is represented hereafter with its features. We exploited three mini-PCIe sockets to handle the IEEE 802.11-OCB communication where we put the ad-hoc modified WiFi mini-PCIe modules. One mini-PCIe socket is used to support the 4G modem, in order to handle cellular communication. Unfortunately, depending on the modem used, sometime the temperature inside the box during the operations can exceed the maximum allowed. This problem will be fixed in the next revision of the system. The storage memory is extended by a mSATA disk drive of 16 Gb to enable long duration data logging.

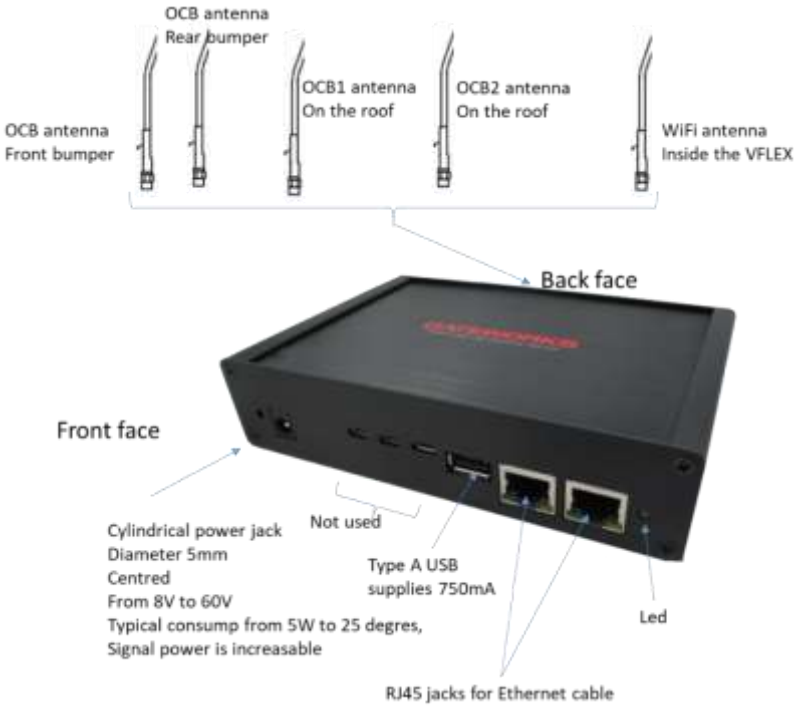

Figure 12 – Gateworks Ventana SBC inside its enclosure

Features

- Freescale™ i.MX6 1GHz Quad Core Cortex™-A9 SoC Processor
- 1Gbyte DDR3-106 SDRAM Memory
- 256Mbytes Flash System Memory
- Micro SD™ Flash Expansion Socket
- Six High-Power Gen 2.0 Mini-PCIe Sockets
  - One Mini-PCIe Socket Supports an Optional GSM11-11 SIM Socket
  - One Mini-PCIe Socket Supports an Optional mSATA Disk Drive
  - Three Mini-PCIe Sockets Support Both PCIe and USB
- Two GbE Ethernet Ports
- USB 2.0 Host and OTG Ports up to 480Mbps
- HDMI 1.4 Input and Output Ports with Audio Support
- LVDS Camera and Display Port
- Three Video Input and Output Channels Supports CVBS/Composite
- Analog Stereo Line In/Out or Stereo Headphone/Mic Audio
- CAN 2.0B 1Mbps Serial Port
- RS232 Serial Port
- Optional RS485 Serial Port
- General Purpose Digital I/O and I2C Expansion Port
- Real Time Clock with Battery Backup
- Voltage and Temperature Monitor
- Serial Configuration EEPROM
- Programmable Watchdog Timer
- Reverse Voltage and Transient Protection
- Power Through Ethernet or Barrel Jack
- Ethernet Power Supports Passive or 802.3af Compatible PoE
- Optional GPS Receiver and Digital 3-axis Accelerometer
- Optional PCIe Expansion Port for Additional Mini-PCIe Sockets
- 8 to 60VDC Input Voltage Range
- 5W Typical Operating Power
- 30W Shared Between Mini-PCIe Sockets
- -40°C to +85°C Operating Temperature
- OpenWrt, Android, OpenEmbedded Linux and Windows Embedded Compact 7 (WEC7) Board Support Packages
- 1 Year Warranty
- (1) Please see this page for more information on 802.3af/802.3at support

Figure 13 – Features of Gateworks Ventana SBC, GW5400[1]

The experience during the previous piloting indicates several limits:

- Throughput limits: during the pre-piloting tests and due to RTMaps application messages frequencies matter, we were able to face the bandwidth limits of the IEEE 802.11-OCB used to handle the V2V communication. In fact, a signal was sent with a frequency of 10 kHz instead of 20 Hz and that caused the usage of the whole available bandwidth even more the dysfunction of the whole platooning system, as consequence of a very high latency. Even though the current need is not to send at such a high frequency the messages between the vehicles, a similar problem may occur when it would come to transfer sensor data between cars. That is why in the future, it might be a good thing
  - to improve the capabilities of the embedded communication device, in terms of latency and bandwidth. For that, we would like to implement IEEE 802.11bd standard that is under development; we would use an open source ath10k driver (Atheros for 802.11ac) rather than IEEE 802.11-OCB on top of ath9k (Atheros for 802.11n) as currently in use.
  - to implement IEEE 802.11 QoS data Headers rather that IEEE 802.11 Data Headers as currently in use. That would allow to apply different data transfer policies according to the priority of the messages.
- Evolution of cellular modems: the tests also revealed some limits of the cellular modem usage in some area of Versailles PS in terms of network coverage, as well as in terms of overheating issue that we would like to address in the next months. A possible solution for

[1] http://www.gateworks.com/imx6-single-board-computers-gateworks-ventana-family/item/ventana-gw5400-network-processor

the network coverage problems would be to change the network operator that provides better coverage if available. For the temperature, an alternative solution would be to study the usage of other cellular modems that support higher temperature values.

- Evolution of the communication device in general:
    - We would like to make the on-board IoT platform (including the CEA IP-OBU) to run IPv6-only in the future, to align with the evolution of IoT in general. Currently, there are two OCB interfaces that are working in IPv6-only and the Ethernet interface that could run that way if needed (currently the Ethernet interface is working in both IPv4 and IPv6). Our wish is to also use IPv6-only in the LTE interface which is currently working in IPv4-only. For that, we might need either to perform some modifications in the kernel or to change LTE modem hardware.
    - It would be interesting to add GNSS Galileo module in order to complete the features available in the communication device.
    - We are thinking also about the possible usage of pre-5G modem in order to switch from 4G-V2X to 5G-V2X.

The results of the works done in AUTOPILOT for the IoT embedded platform would be exploited in three directions:

- The results concerning the limits of the different subsystems of the communication device (IP-OBU) and software[2] would be used to find ways to improve technically the solution. That would help in particular for problems encountered in LTE communication interface (coverage, latency and temperature) and IEEE 802.11-OCB interfaces (bandwidth improvement).
- The results of the project in general are promising and are used to create added-value in terms of IPR. At CEA, we submitted a patent application request to the French Bureau of Industrial Property (Institut National de la Propriété Industrielle - INPI) thanks to the work that we did in V2V communication protocol for Platooning Use Case.
- Thus the results are also exploited to contributed to the advancement of technology. We contributed to Internet Drafts, as well as to a ETSI technical report related to IPv6-based V2X communications [ref. ETSI/IPv6-based V2X (ISG IP6), .

2.4.3 AD vehicle prototypes in Livorno Pilot Site

Regarding the in-vehicle IoT platform, with respect to the initial version of the node architecture, some changes have been made, with the purpose of improving the solution, as further explained in the following. For pothole detection, the initial concept was to use the 6LoWPAN sensor in the OBU architecture to detect the vehicle vibrations. But later on, given the availability of other connected vehicles, but without the CAN interface (Figure 17) an IMU (Inertial Measurement Unit) was introduced to allow not only to measure vertical accelerations for pothole detection, but also to

[2] In particular we are referring to OCB driver, CFIO/CFOI, V2V. In particular CFIO are CAM From Inside to Outside: CAM - Cooperative Awareness Message generated (in XER format) by the embedded computer (PC-AD) of each car in order to be sent (in UPER format) outside of the car throught OCB interface; CFIO are CAM From Outside to Inside (received (in UPER format) from external ITS-stations such as roadside units, connected bykes, throught an OCB interface and that have to be transmitted to the PC-AD); CFIO/CFOI software package handles CAM conversion and transmission from the vehicle's inside to the vehicle's outside (XER over Ethernet to UPER over IEEE 802.11-OCB) and vice versa (UPER over IEEE 802.11-OCB to XER over Ethernet)

record most of the vehicle data about its dynamics, such as lateral, longitudinal accelerations and yaw-rate.

In fact, some internal components of LINKS OBU work in a more stable way when dealing with values of dynamics from IMU. We expect these changes to ease the post processing the logs from both DSRC/ETSI ITS G5 and IoT messages and to improve the overall OBU as a product.

The experimentation has permitted to successfully trial the OBU in relevant environments, such as Highways and urban-suburban areas, with many runs in hours and kilometers of successful testing, including the demonstration programmed at Livorno in October 2018. This has matured the OBU in a TRL6, namely “technology demonstrated in relevant environment” [2].

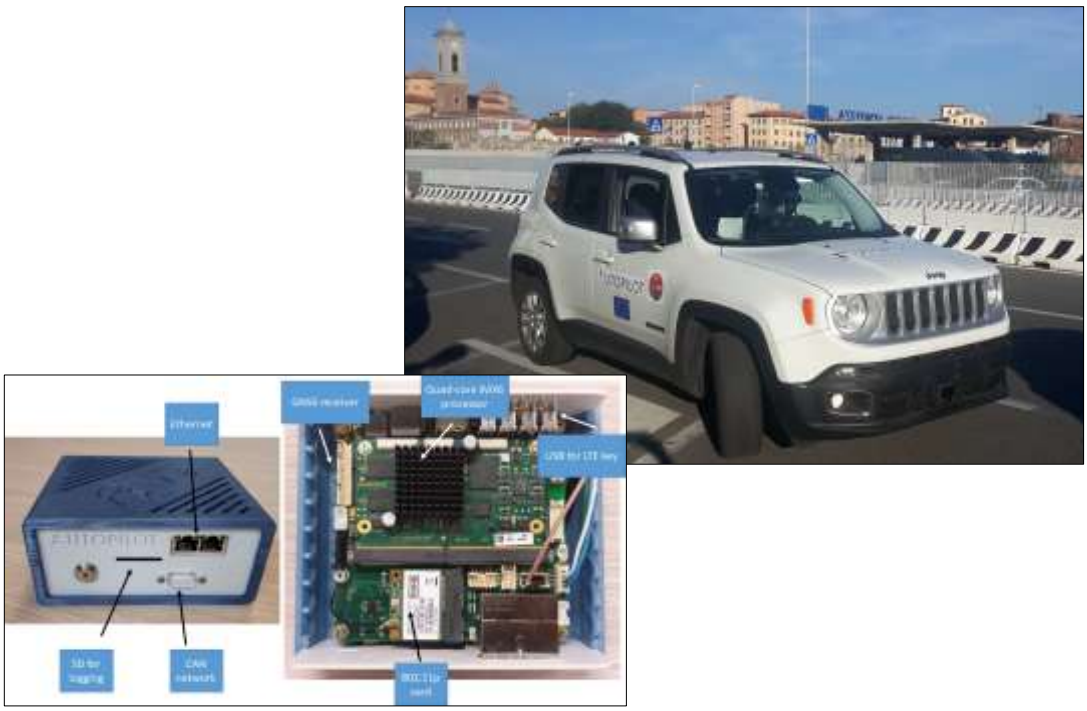

Figure 14 – CRF vehicle prototype in Livorno (upper right) and Links in-vehicle IoT platform (lower left)

Hereafter the final description of the vehicle prototypes used in Livorno Pilot Site is provided:

- IoT connected prototypes (by CRF and AVR), equipped with: a communication platform with both ETSI ITS and LTE connectivity; sensors to get information from the environment; 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 (by CRF), 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 behaviour.

A general scheme is shown in Figure 15.

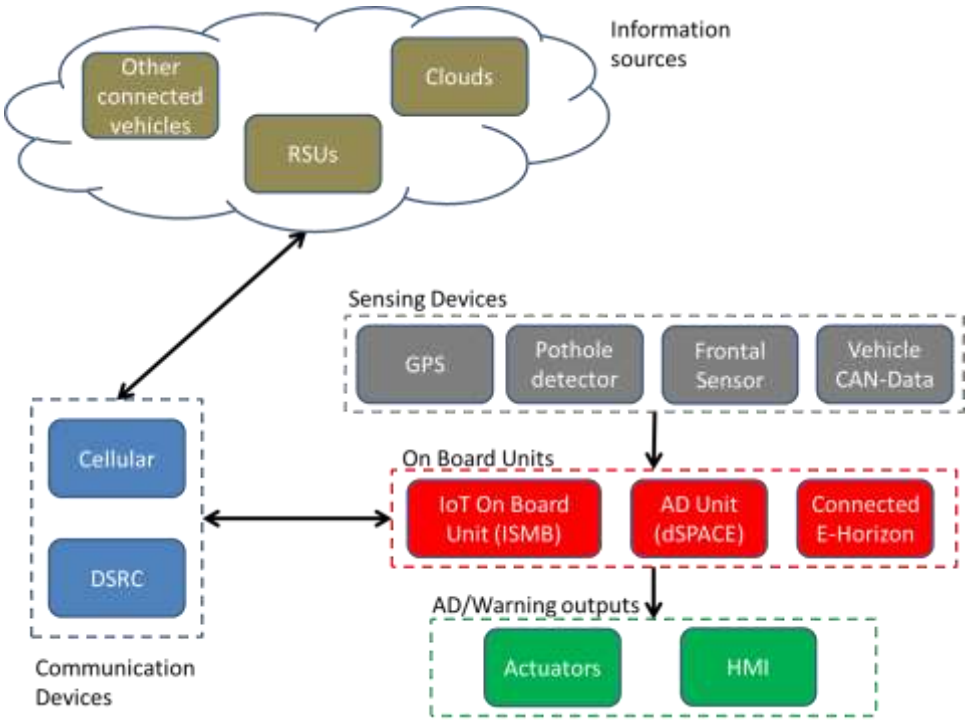

Figure 15 – General scheme of Italian Test Site vehicles

- 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.
  - 6LoWPAN: Wireless communication based on IEEE802.15.4 radio to communicate between Wireless Sensor Networks on-board and the IoT vehicular platform.
- 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.
  - 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 including the WSNs. 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.
- 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.

The following pictures represent CRF and AVR connected vehicles (Figure 16,Figure 17) and CRF automated vehicles (Figure 18).

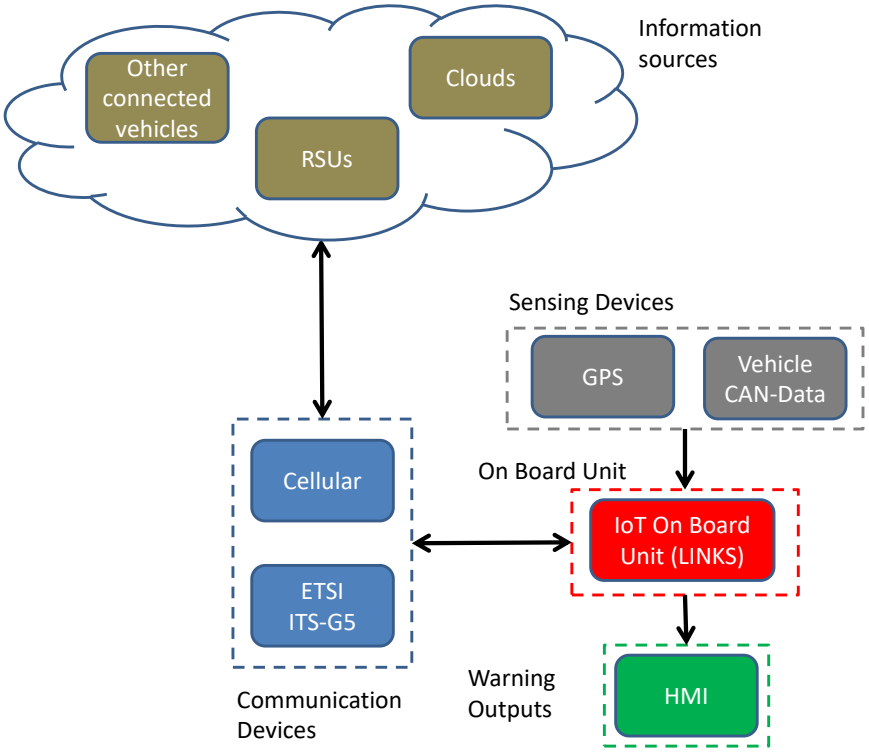

Figure 16 – CRF connected vehicles

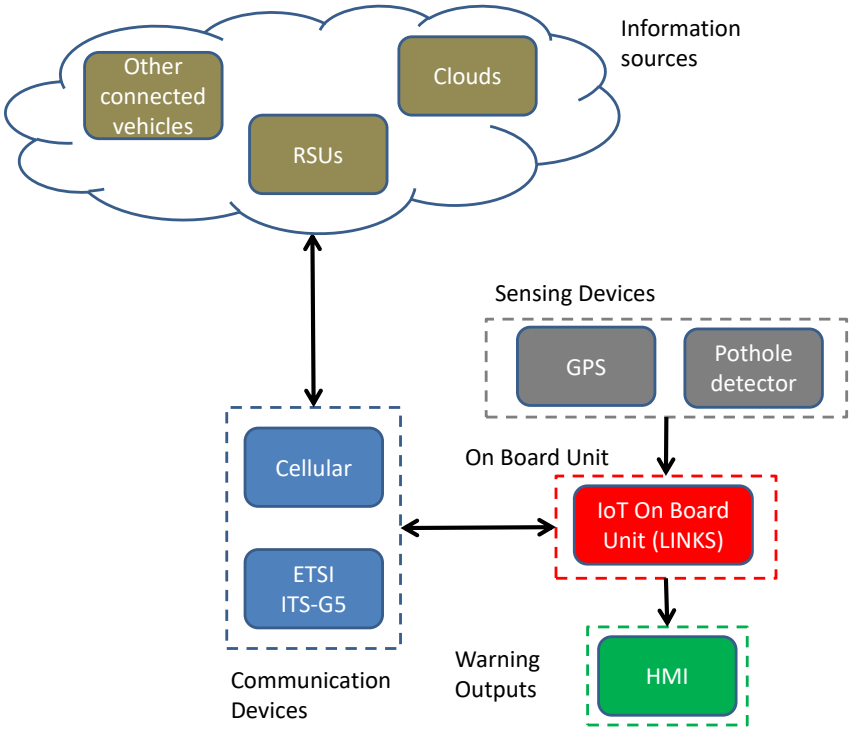

Figure 17 –AVR connected vehicles

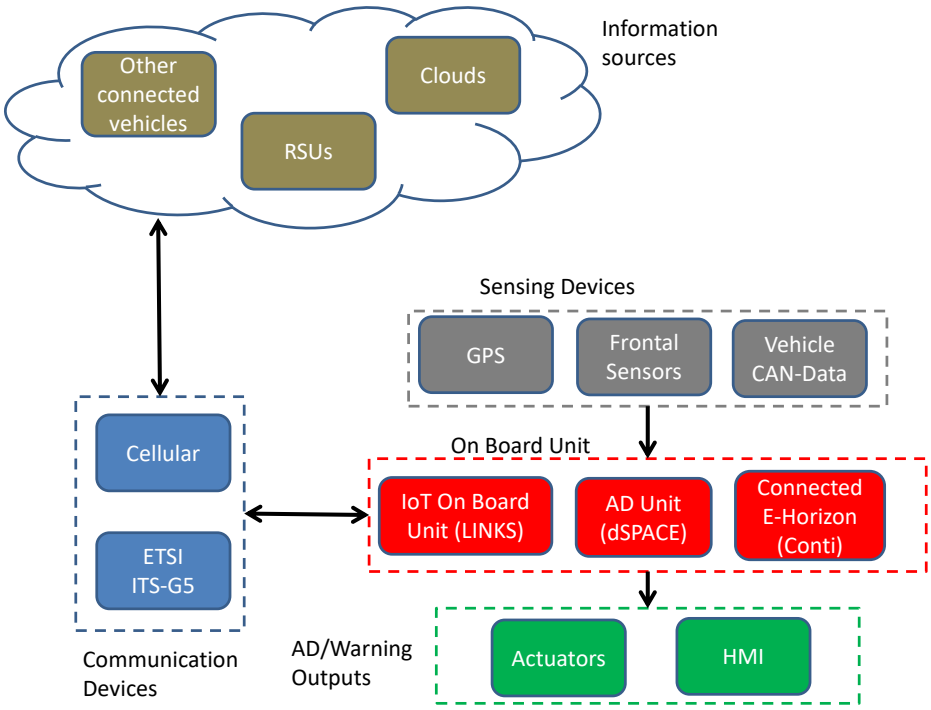

Figure 18 – CRF AD & Connected vehicles

The communication interfaces of AD & connected vehicles in Livorno PS are shown in Figure 19.

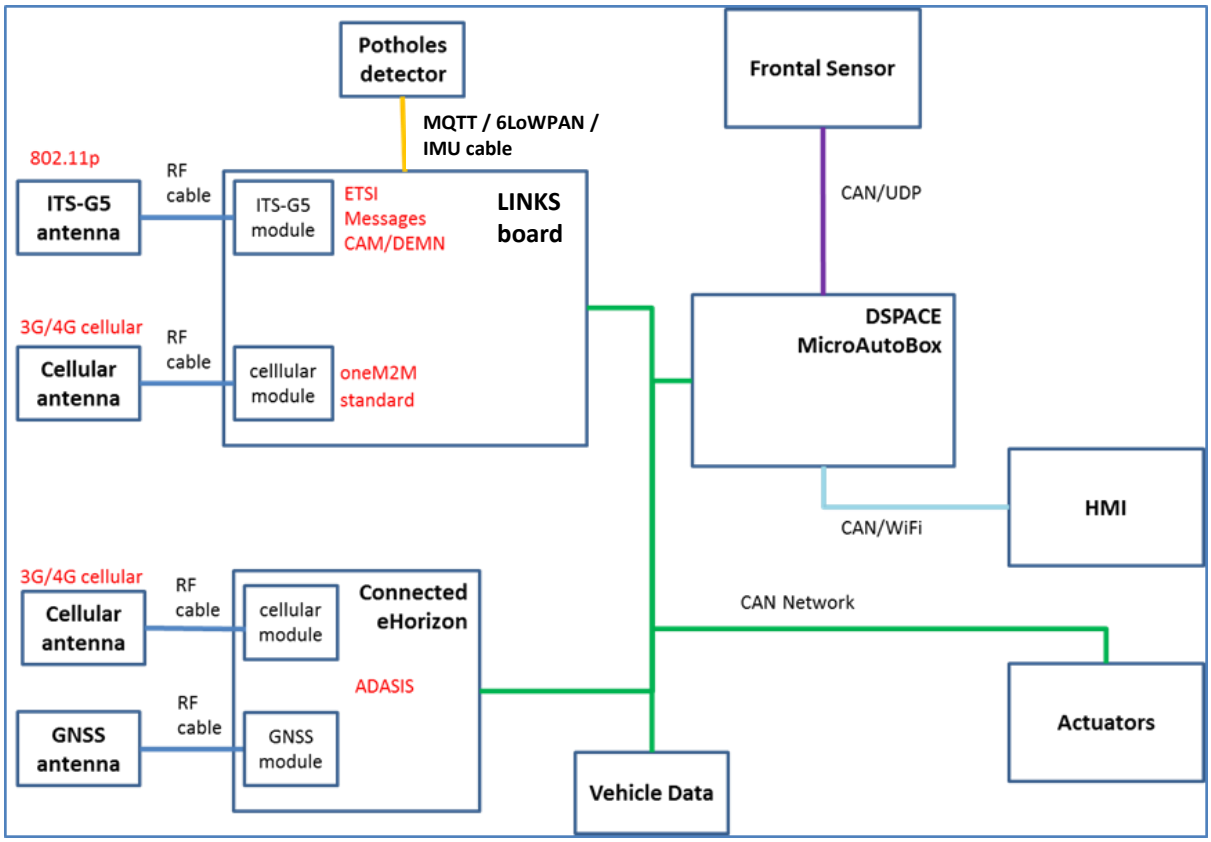

Figure 19 – Components and interfaces of Italian Test Site AD & connected prototypes.

2.4.4 AD vehicle prototypes in Brainport Pilot Site

Figure 20 shows the in-vehicle IoT platform high-level architecture for the AD vehicle prototipes used in Brainport Pilot Site. It includes concepts and components that are common to all the prototypes involved in the Use Cases tested in Brainport. 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).

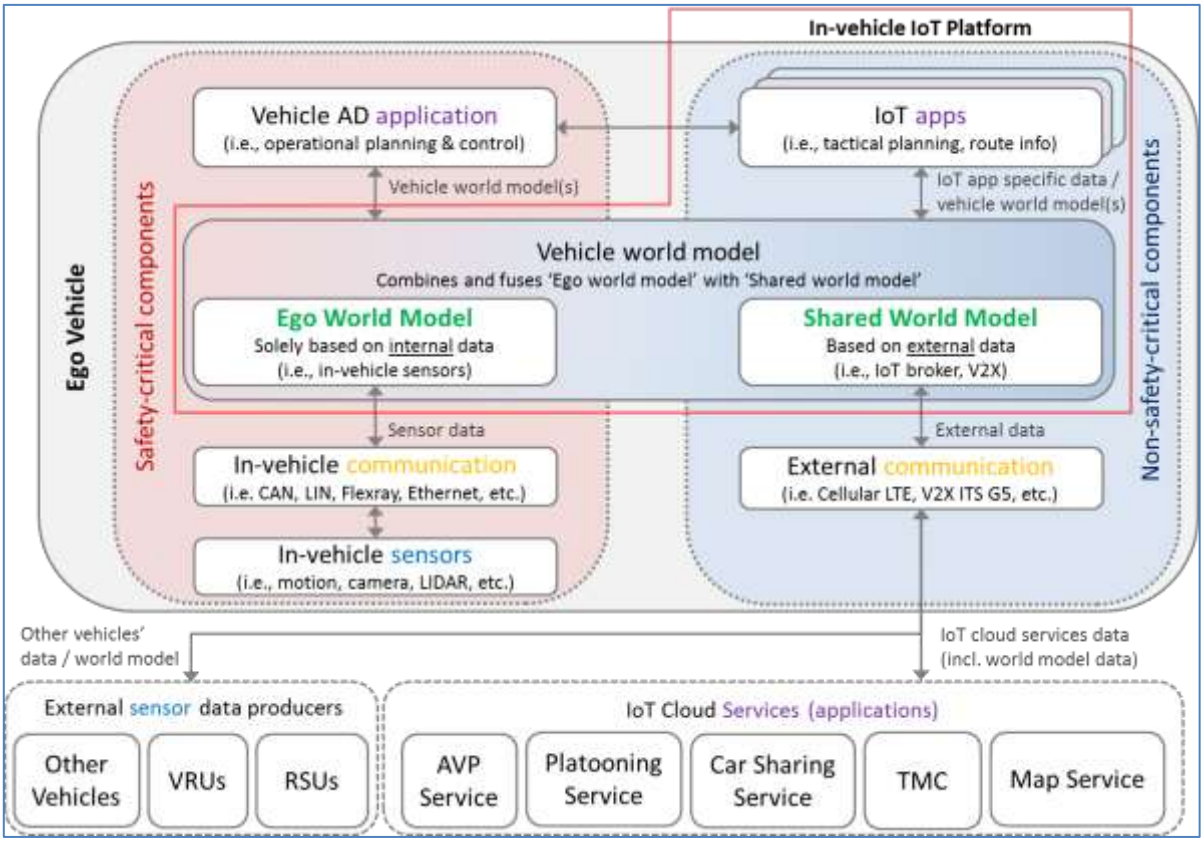

Figure 20 – In-vehicle high-level architecture for Brainport pilot site

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 was to integrate those developments in the test vehicles for the Brainport platooning- and highway pilots. This service would have been used in the different brain-port use-cases to access the map. However, Localization in the HD-Map did not fit in the available vehicles. TomTom has then set-up its own car for technology validation purposes as shown in Figure 21. This car is based on a survey vehicle with high cost sensors used to create ground truth, extended with a localization sub-system that connects to the map delivery service and to different automotive grade sensors such as Lidar, Camera and Radar (see Figure 21).

Lane accurate localization in combination with HD Map data enables lane accurate navigation. The online horizon has been replaced with Autostream for HD-Map. Live information such as hazards and traffic jams are downloaded alongside the HD-Map via location based dynamic objects.

Figure 21–TomTom survey vehicle used for adding automotive grade sensors for localization

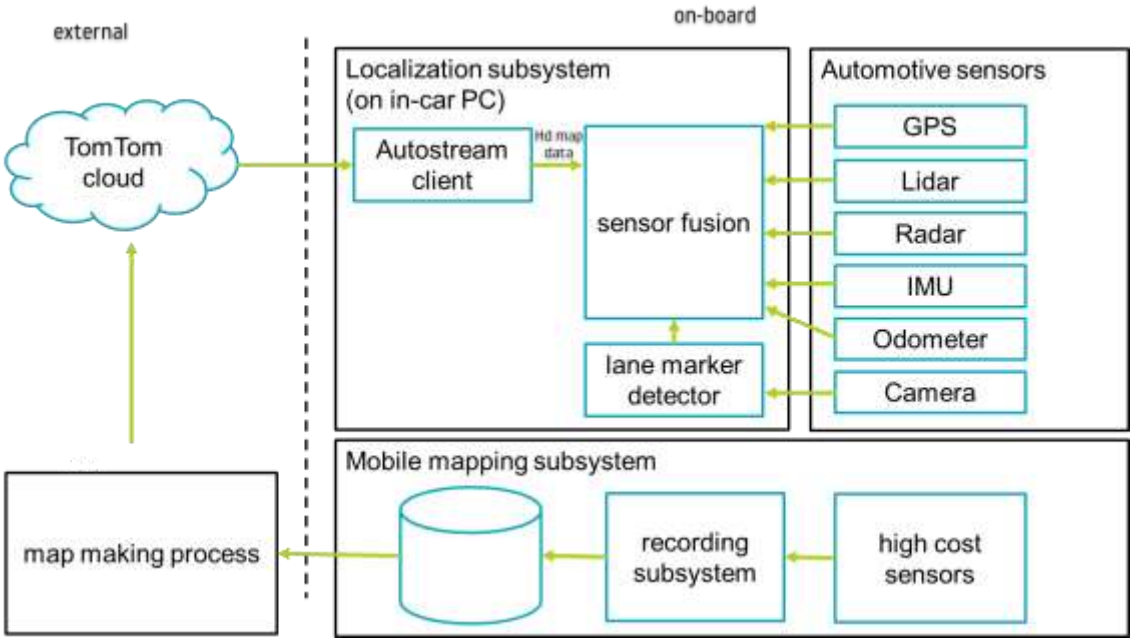

Figure 22 - TomTom prototype for technology validation

The maturity of TomTom prototype is according to the component, shown in Table 2.

Table 2 – Maturity and relevance of TomTom prototype

Component	Readiness	Exploitation
Vehicle platform	TRL 6	Used for map recording, technology validation and partner collaborations
HD-map delivery service	TRL 7	Commercial map product for automotive customers
Localization software	TRL 4	used for technology validation (HD-map and its delivery) and partner collaborations
Visualization software	TRL 6	used as visualization tool of HD-map products
Platform for Hazard service	TRL 3	used as tool for partner collaborations

2.4.4.1 TNO prototype

Overall TNO has made only minor changes to the vehicle architecture specified in D1.5. These changes concern the decision to use a single processing unit instead of two, as initially planned.

Another change regards the motion planning functionality that had to be simplified due to lack of resources, especially for the Automated Valet Parking use case. The initial intention was to incorporate IoT data about other road objects (pedestrian, obstacles, etc.) to be fused in the vehicle world model and thereby be taken into account by the path planning algorithm in the vehicle. This remains an interesting topic for future work to understand how the fusion of data from multiple sources can be effectively used by automated driving vehicles.

As per the technological maturity achieved, distinction should be made between Platooning and Automated Valet Parking (AVP). Moreover, certain functional parts could be validated and piloted better than other parts.

In case of platooning the goal was TRL level 7 for the whole system (vehicle plus cloud) and was

almost reached. Specific issues which prevented from convincingly reaching that goal are: 1) platooning does not work well under all weather conditions; 2) the GLOSA functionality developed does not work well in combination with smart Traffic Light Control systems; 3) the control algorithm which acts on externally fed World Model targets could not be validated sufficiently and still has some performance issues.

In case of AVP, the TRL level is 6 for the basic automated driving behavior (only manual gear change for reverse) assuming a predefined external environment and also TRL 7 for the interaction between the vehicle and the Parking service (cloud). AVP lacked however sophisticated agile motion planning (only static routes possible) in combination with situational awareness (no real world model applied so no detection of obstacles). These ambitions could not be realized with the resources available.

The following section describes TNO prototype vehicle.

Figure 23 – TNO prototype vehicle (source: TNO website)

TNO uses a modified Toyota Prius equipped with additional sensors to support automated driving functions (shown in Figure 23). The general vehicle scheme of hardware components is shown in Figure 24.

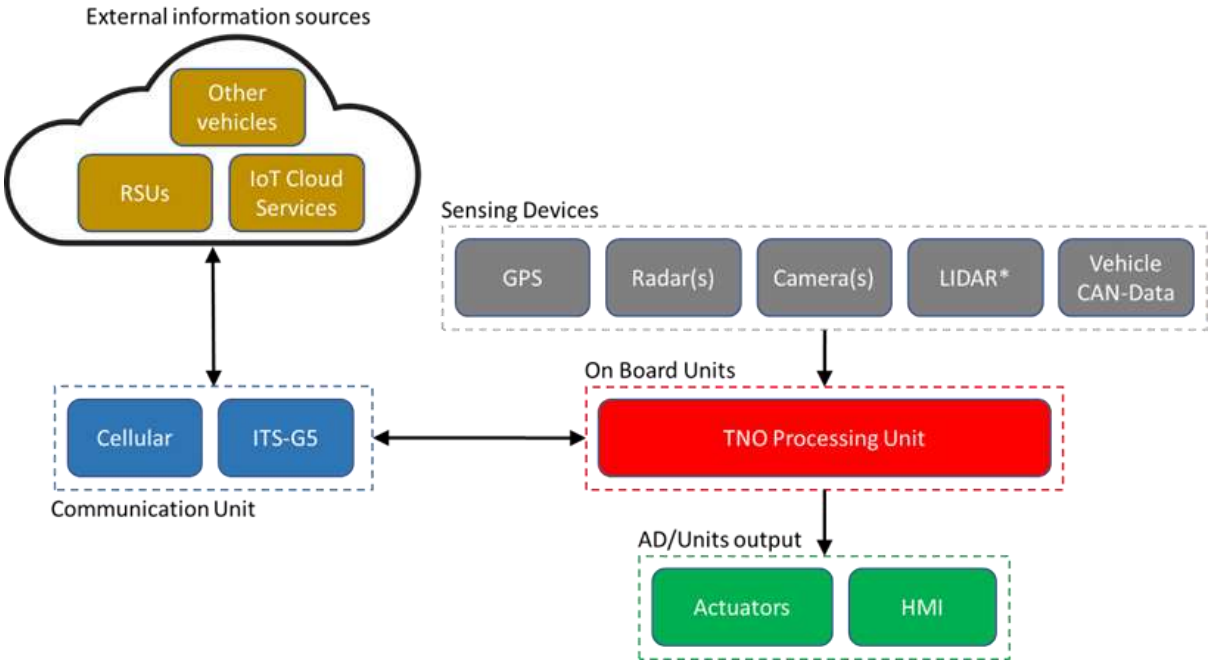

Figure 24 – TNO vehicle scheme

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 neighbourhood and with the road infrastructure

(roadside units).

- Sensing devices:
  - GPS module: it estimates the position of the vehicle (latitude, longitude, heading, speed). RTK-GPS is 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 specific tasks such as localization.
  - Camera(s): camera(s) are used to get images of the environment that can be used for specific tasks such as object classification.
  - 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. Finally, this processing unit also includes 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 is defined as AVP. Due to the variety of the possible requirements of this use-case the control interfaces are left accessible through a prototyping environment (i.e. dSpace MABX). 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.

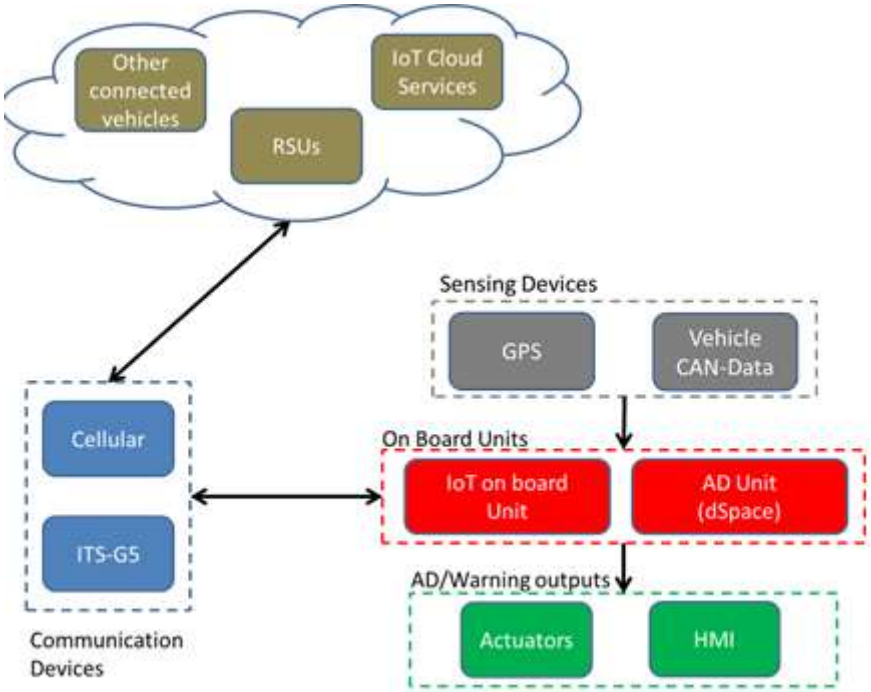

Figure 25 – NEVS vehicle scheme

In addition, GPS is used to know the exact location of the vehicle and appropriate commands to manoeveur the vehicle is given by In-vehicle IoT platform connected to MABX dSpace unit.

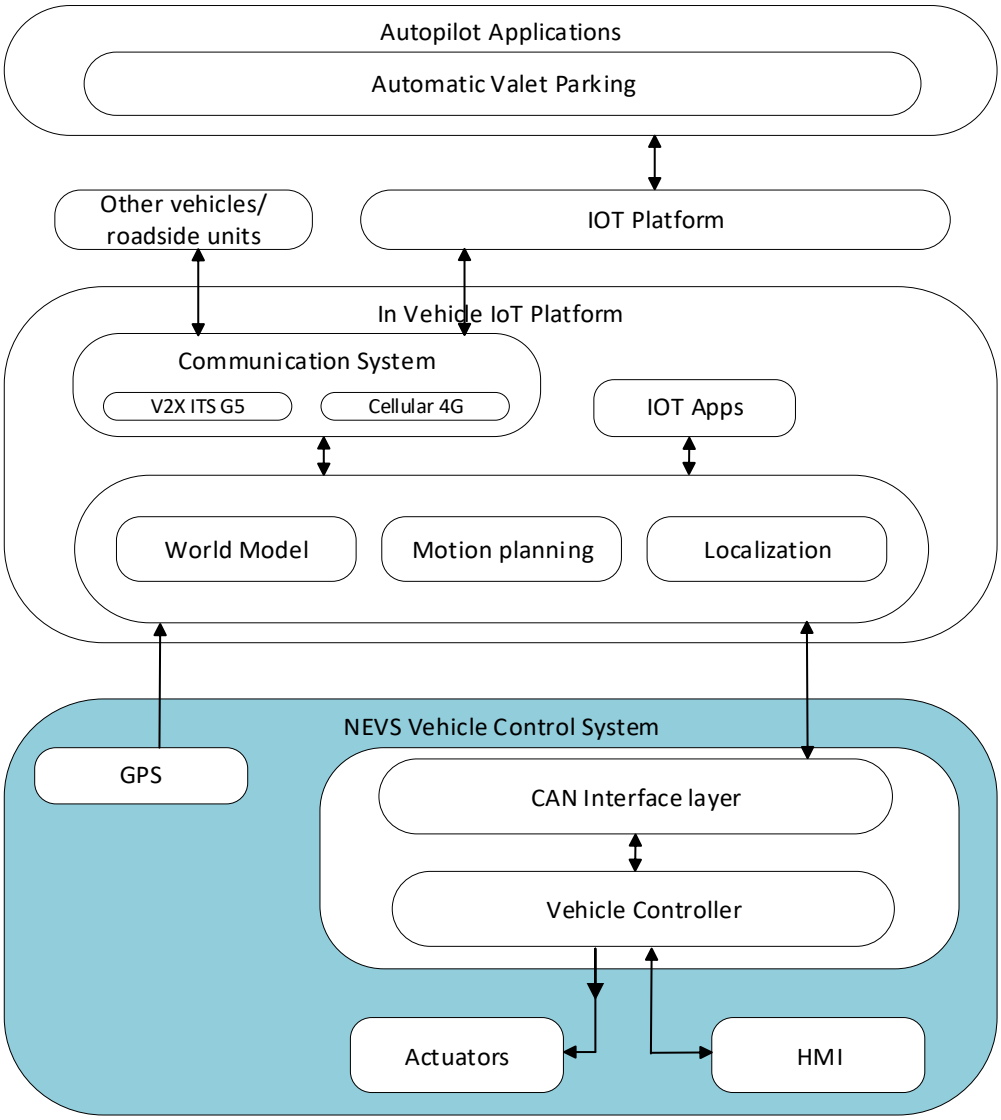

Figure 26 – NEVS vehicle scheme

2.4.4.3 TUEIN prototype

TU/e used a modified Toyota Prius for implementation and execution in the Brainport TU/e Car Rebalancing (Urban Driving) use case.

It has a basic Automated Driving functionality which can be controlled from a real time target (AD unit).

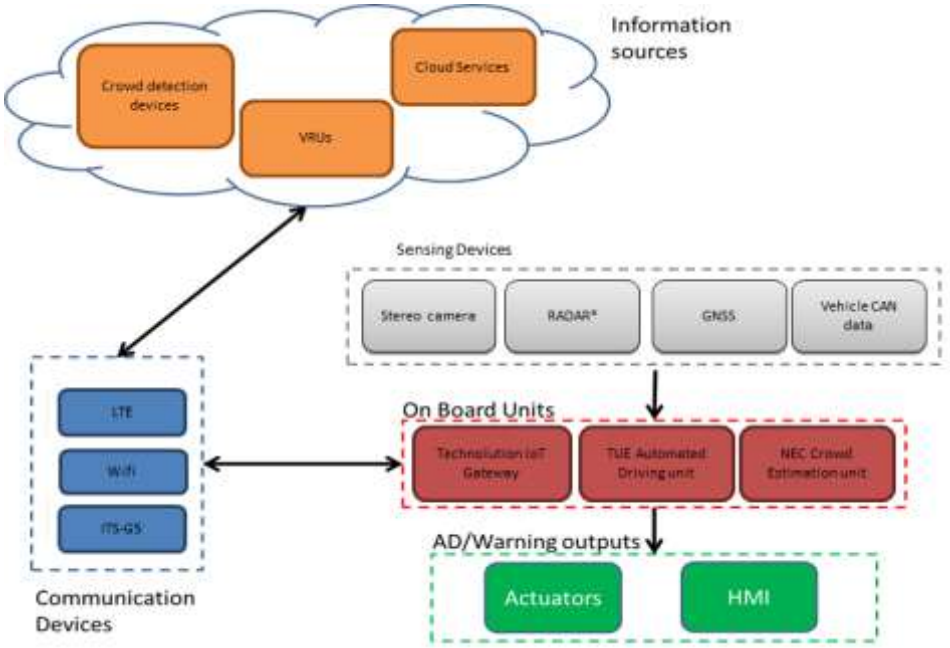

Figure 27 – Scheme of the TU/e AD vehicle: Prius

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 other roadusers (ie. VRU, vehicle) 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*: is available in the vehicle, but was not used in this project specifically.
  - 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:
  - Technolution IoT Gateway: Used as a gateway between the AD unit and the IoT cloud and ITS-G5 enabled devices. It can also be used as a processing unit for datalogging.
  - TUE Automated Driving Unit: It is the component responsible for control related functions that will send commands to actuators in the vehicle.
  - NEC Crowd Estimation unit: detects crowd/persons around the vehicle based on WiFi signals. The unit is directly communicating with the NEC FIWARE cloud and information is being send to OneM2M platform where the vehicles can get the information to use in rerouting applications.
- 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 implementation phase, starting from the same Prius model used also by TNO (see 2.4.4.1 and Figure 28), specific adaptations have been introduced.

The current architecture of the hardware implemented in the Prius TU/e prototype is shown in Figure 29.

Figure 28 – TU/e prototype vehicle

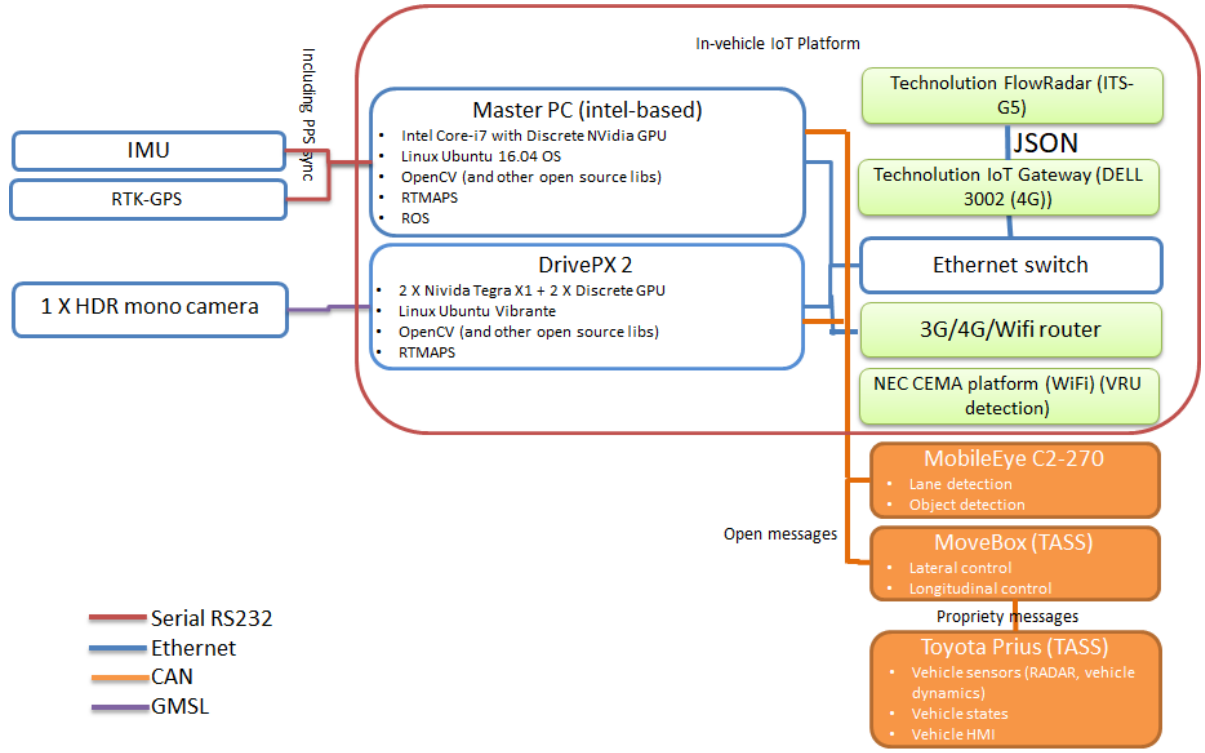

Figure 29 – Final implemented TU/e prototype vehicle architecture, as used in pilot tests.

2.4.4.4 VALEO prototype

Valeo provides two vehicles for the needs of the Highway Pilot pothole detection use case: a Volkswagen Tiguan is used for the Detection phase, and a Jaguar F-Pace is used for the Driving Adaptation phase.

Actually, both functionalities might have been embedded into a single vehicle, as in the initial plan.

However it was soon discovered that it was more manageable (for development and piloting) and understandable (for explanations and visitors) to split the roles.

Figure 30 – Valeo prototype vehicle: VW Tiguan

Figure 31 – Valeo prototype vehicle: Jaguar F-Pace

The VW Tiguan (Figure 30) was selected because of its generic MQB platform [13], which Valeo has much experience with. The car was prepared with multiple and diverse additional sensors and software units needed for road hazards detection. The Jaguar F-Pace (Figure 31) was selected because previously equipped with computing and customizable HMIs.

Both vehicles solutions are architected around a common Valeo Smart Platform (Let's Do), which behaves as an In-Vehicle IoT Platform. Most modules exchanges pass through it.

Initially, the use case intended to cover a large variety of road hazards, from highly critical ones (ex: a life-threatening car standing still on a highway) to almost insignificants ones (ex: a shallow pothole). The industry focuses its safety efforts on the former, including in the standardization of emergency signals in V2V for instance. However, VALEO chose to focus on the later ones which are largely ignored and that can take better advantage of an IoT approach, where the number and diversity of observations is more important than instant signalling. For this reason, cellular only was favoured over dual Cellular and ITS-G5 in this implementation.

Actually, the communication link between vehicles and cloud services did not connect to an IoT open platform for the trials, and furthermore, the in-vehicles systems themselves achieved only Technical Readiness Level 3 (TRL 3 – experimental proof of concept) as they are equipped only pre-commercial platforms. On the other hand Valeo could prove a concept close to an IoT service, namely anomaly detections by car and hazard generation by cloud, meant for Highway Pilot; and at the same time, Valeo could build on technical choices (Valeo Smart Platform, RT maps) that are in its roadmap and can be further evolved to IoT, to facilitate the integration of other vehicles or objects.

The prototype solution can be illustrated on a single functional view as shown in Figure 32. It is also important to mention that the vehicular native systems (native sensors, ACC control system, etc.) are preserved, the modifications and modules are only supporting the use case, not affecting native car functionalities.

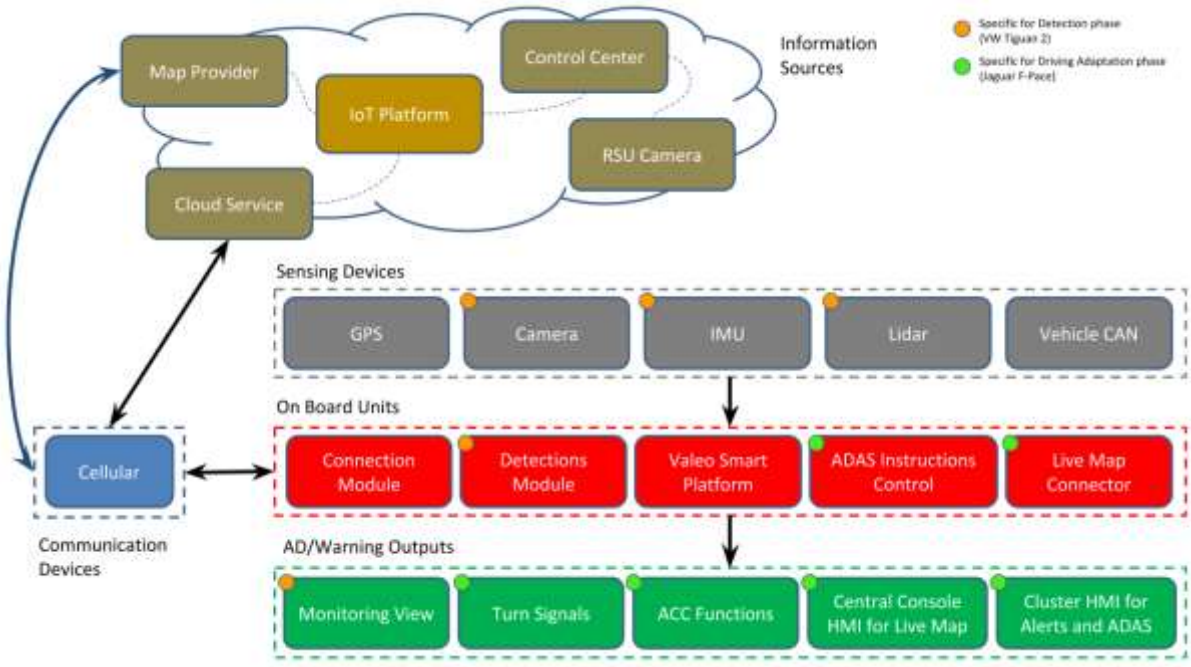

Figure 32 – Valeo prototype vehicles: common functional view

The functional blocks can be further described:

- On Board Units
  - Connection Module
    - Reports anomalies and associated images to the Cloud
  - Detections Module
    - Runs IMU, Camera and Lidar processing algorithms tuned for our use case
  - Valeo Smart Platform
    - Handles subscriptions and messages across all the components (composed of MQTT Broker and libraries).
  - ADAS Instructions Control
    - Processes received external ADAS Instructions and applies commands on ACC
  - Live Map Connector
    - Retrieves live map objects (Anomalies, Hazards, ADAS instructions) from Map Provider
- Sensing Devices:
  - Most are picked from Valeo's own portfolio.
- Communication Devices
  - Handles 4G cellular connectivity
- AD/Warning Outputs
  - Monitoring View
    - Displays running road hazards detection
  - Turn Signals
    - Provides native control of lights
  - ACC Functions
    - Provides native control of ACC
  - Central Console HMI for Live Map
    - Displays local area map, test track, traffic signs, anomalies, hazards and ADAS Instructions
  - Cluster HMI for Alerts and ADAS

- Displays warnings, instructions or vehicle intents

2.4.5 AD vehicle prototypes in Vigo Pilot Site

The Spanish Test Site has provided 3 automated vehicles for AUTOPILOT: PSA has contributed with 2 vehicles and CTAG with 1 vehicle (PSA branded).
All the vehicles are equipped with an IoT in-vehicle platform and the necessary automatic driving functions to support urban driving and valet parking use cases.

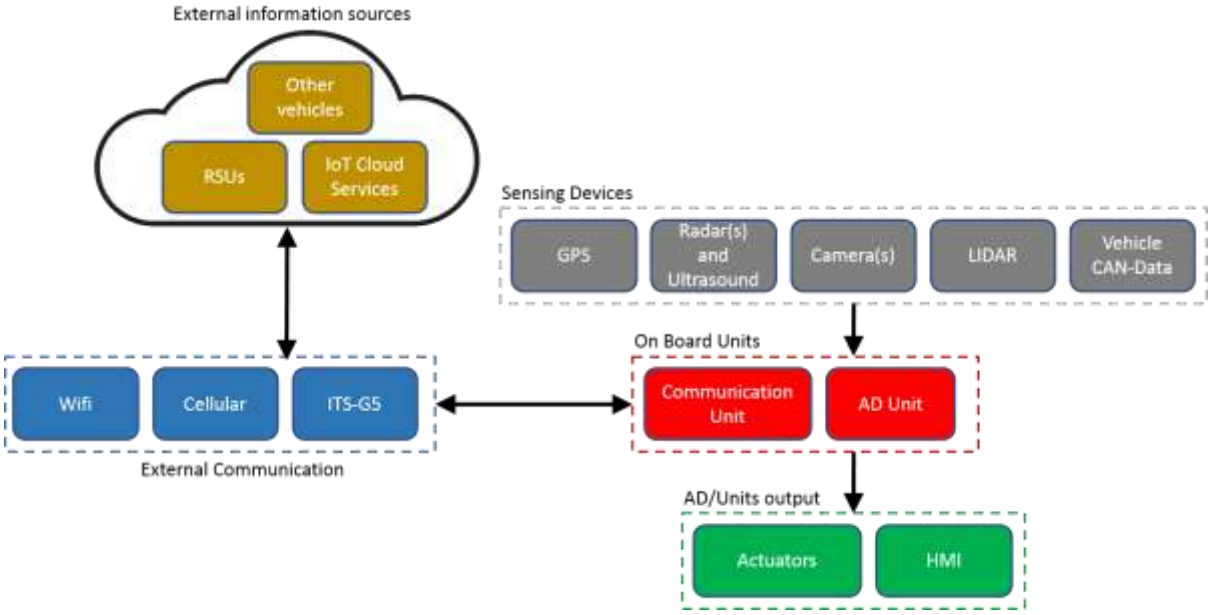

Figure 33 - Architecture of the AD vehicle prototypes used in Vigo Pilot Site

As an example, the architecture of the AD vehicles used in Vigo Pilot Site is described in Figure 33. The main components and interfaces are described in Figure 34, as explained below:

- 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 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.

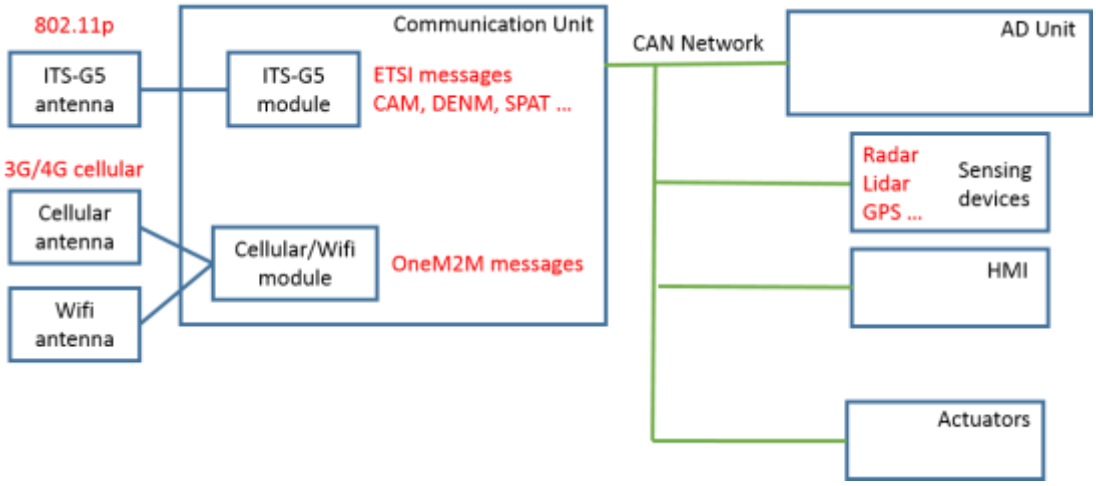

Figure 34 – Interfaces between the main components at Vigo Pilot Site

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 the external networks. The IoT vehicle platform can be considered as the automotive counterpart of the IoT Gateway component of an IoT infrastructure.

3.1 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 35.

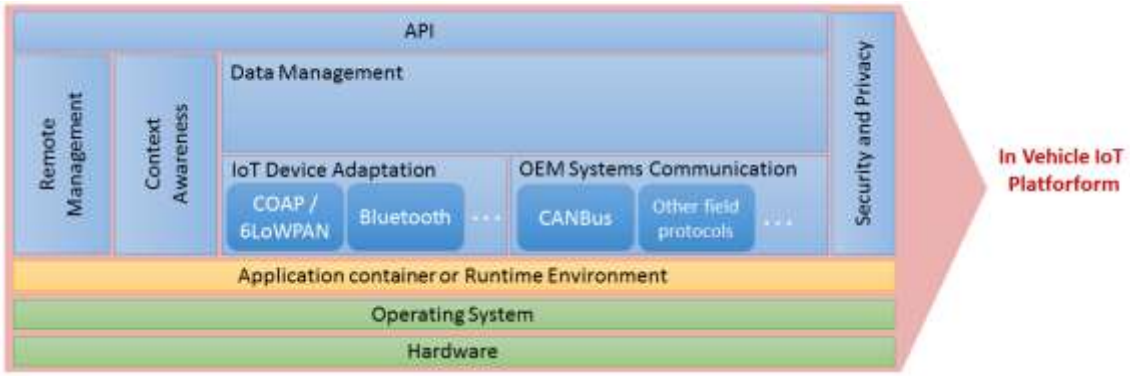

Figure 35 - High-level functional architecture

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 architecture in Figure 35 shows that AUTOPILOT applications interact with the vehicle either directly or via the IoT Platform (Figure 3). While, the architecture depicted in Figure 36 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.

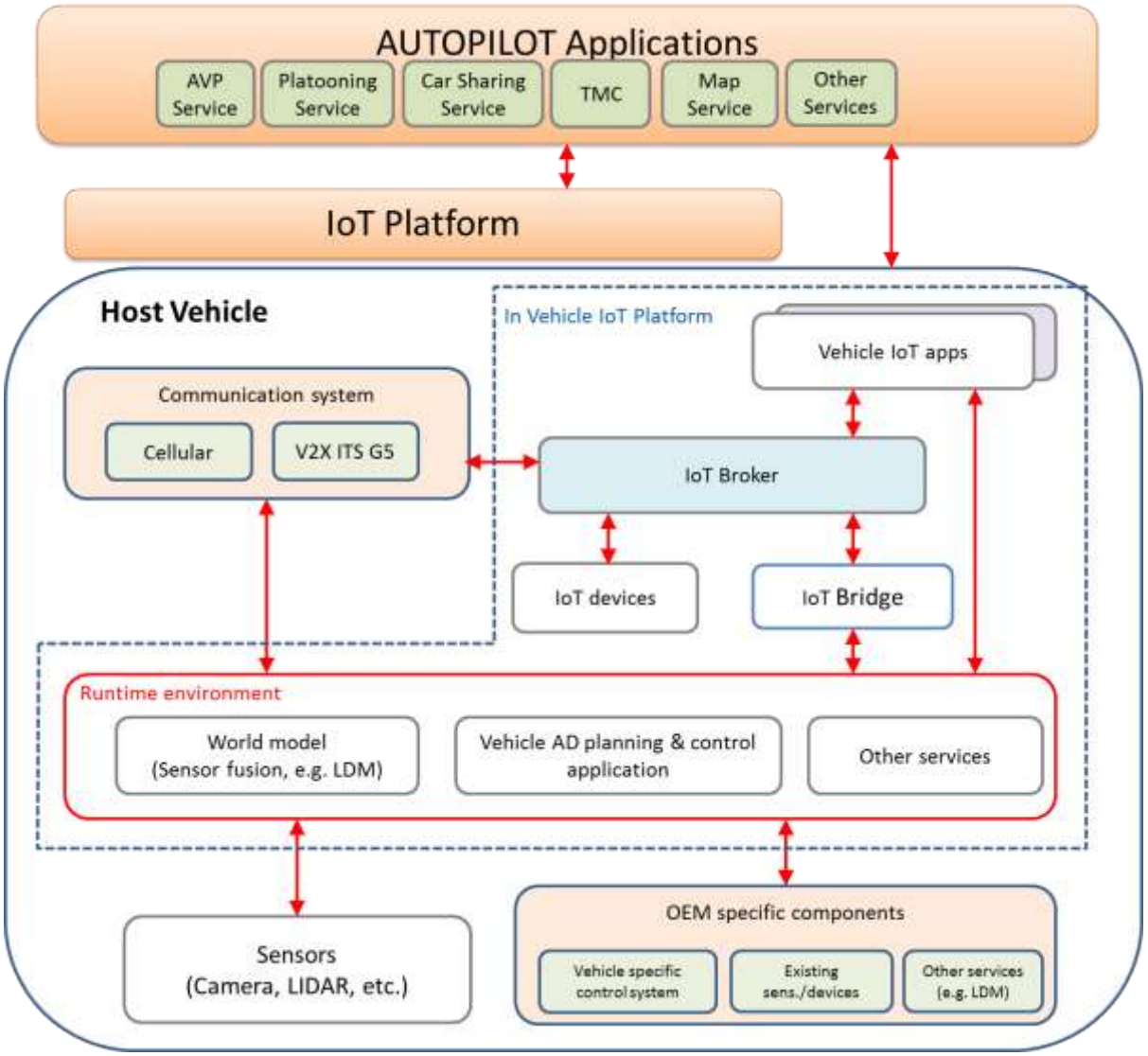

Figure 36 – In Vehicle Architecture

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 IoT devices in the 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 from 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 component 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 3 shows how the functional architecture components satisfy the needs and functionalities of the Vehicle IoT Platform.

Table 3 – 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 the In-Vehicle IoT Platform.

In relation with the defined primary functions, a list of 56 functional and 53 Non-functional requirements was produced (Table 4, Table 5). The table reports a Requirement Identifier (ID#), the link to Parent requirement (P# Link), Usecase, Title and Description.

Table 4 – Functional requirements

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

ID#	P# Link	Use case	Keyword/Title	Description
				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.
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 be able to integrate heterogeneous IoT devices into the prototype, mantaining 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 realted with "opened doors" and "seat occupance") 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

ID#	P# Link	Use case	Keyword/Title	Description
				up the vehicle to move to the collect point
FR16	P3	AVP	Unmanned mode	Driver validation: the vehicle grants access to authorised drivers
FR17	P2	Highway pilot	Hazard detection	The In-Vehicle PF must be able to receive hazard warning information from connected IoT Infrastructure PF (alterntive to DENM from ITS-G5 channel, from cloud connection)
FR18	P4	Highway pilot	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	Highway pilot	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	Highway pilot	Lane Following	The In-Vehicle IoT PF can be informed when AD vehicle is activating Lane Following functionalities (lateral control)
FR21	P1	Highway pilot	Lane Change	The In-Vehicle PF can be informed when AD vehicle is activating Lane Change functionalities (lateral control)
FR22	P2	Urban driving	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).
FR23	P4	Urban driving	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.

ID#	P# Link	Use case	Keyword/Title	Description
FR24	P2	Urban driving	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.

ID#	P# Link	Use case	Keyword/Title	Description
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.
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,

ID#	P# Link	Use case	Keyword/Title	Description
				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 informations, 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, desciption 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

ID#	P# Link	Use case	Keyword/Title	Description
				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.
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, desciption 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

ID#	P# Link	Use case	Keyword/Title	Description
				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 (funct. rebalancing)	Probabilistic world model	The In-vehicle PF can be able to receive TU/e lecturing scheduling informaiton from the TU/e websever (WiFi connection).
FR56	P2	Urban driving (func. rebalancing)	Probabilistic world model	The In-vehicle PF is able to receive weather information from internet.

Table 5 – Non-Functional requirements

ID#	P# Link	Use case	Keyword/Title	Description
NFR1	FR1	All	Vehicle Reference Position	In-Vehicle dynamic data resolution: Refererence 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]

ID#	P# Link	Use case	Keyword/Title	Description
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 acceleratind 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 dynammmic 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]
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]

ID#	P# Link	Use case	Keyword/Title	Description
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	Highway pilot	Lane Keeping status	In-Vehicle AD function status: Lane Keeping [Idle, torque overlay activated, Not active, …
NFR29	FR4	Highway pilot	ACC status	In-Vehicle AD function status: ACC [Idle, target speed activated, Not active, …]
NFR30	FR5	All	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
NFR31	FR5	Urban driving	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

ID#	P# Link	Use case	Keyword/Title	Description
				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 related information: [absolute position lat;lon - 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	Age of measurements in seconds.

ID#	P# Link	Use case	Keyword/Title	Description
			information: age	Resolution: 0.001 s
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 is 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, that are used by the AUTOPILOT test sites, as specified in the second section

4.1 Survey of 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,[14] e.g. used in the In-vehicle IoT platform of Livorno Pilot Site, 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 to install, uninstall and update all modules without restarting or even stopping application for these operations. At once it is possible to think 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 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 registering 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. Here we give the example of the context awareness in TUEIN/TomTom vehicle in the Dutch pilot.

Online Horizon system

Vehicle-based applications, like 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.[3] 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.

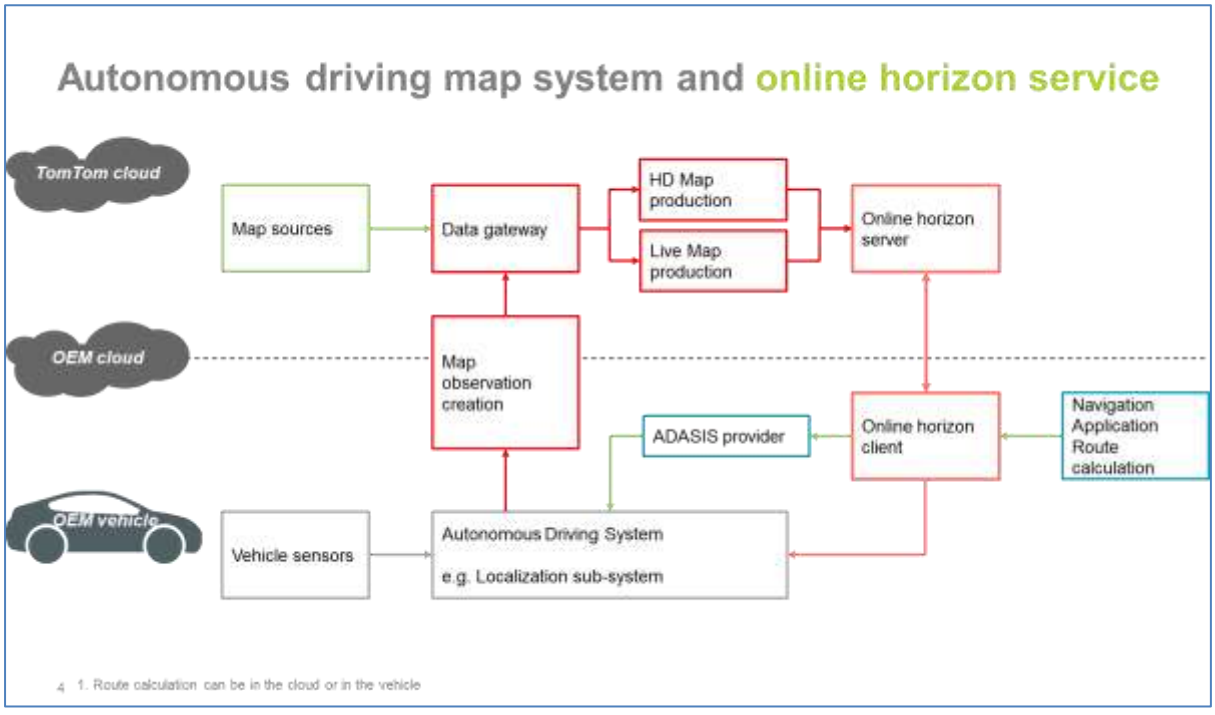

Figure 37 – Online horizon service, example from TomTom

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

[3] 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.

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.

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.

Is also the ability to filter, analyze, and correlate vehicle sensor data and take action on the large amount of data generated. I.e. 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 a 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 combination 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. Hereafter, examples of protocols that can be used for communication interoperability are provided. Each prototype leader has made its choices, for testing purposes, addressing interoperability within the vehicle components and between the vehicle and the IoT infrastructure.

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 v[23].

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) [17] 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 fanout 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.

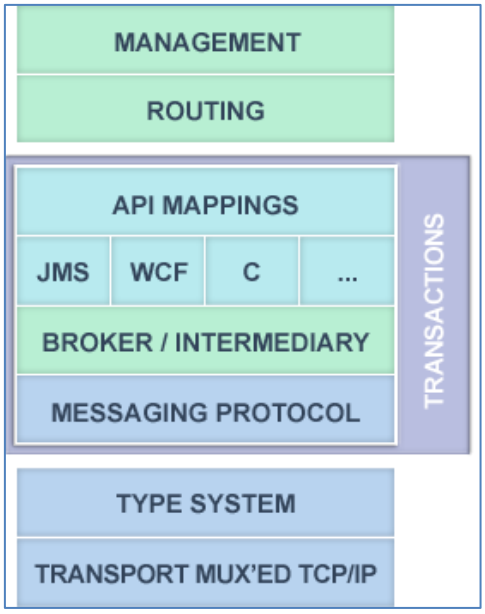

Figure 38 – AMQP Architecture

The AMQP Network Protocol

The AMQP Network protocol **[18]** 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 39 – 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 (used by Links)

To start with this specification, we need to introduce the consortium OASIS which is the Open Advancing Standard for the Information Society **[19]**. 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 **[20]**) has been developed by the company, Chargerlink, Inc. This protocol uses Message Queue Telemetry Transport (MQTT **[21]**) from IBM and Protocol Buffers (ProtoBuf **[22]**) 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.

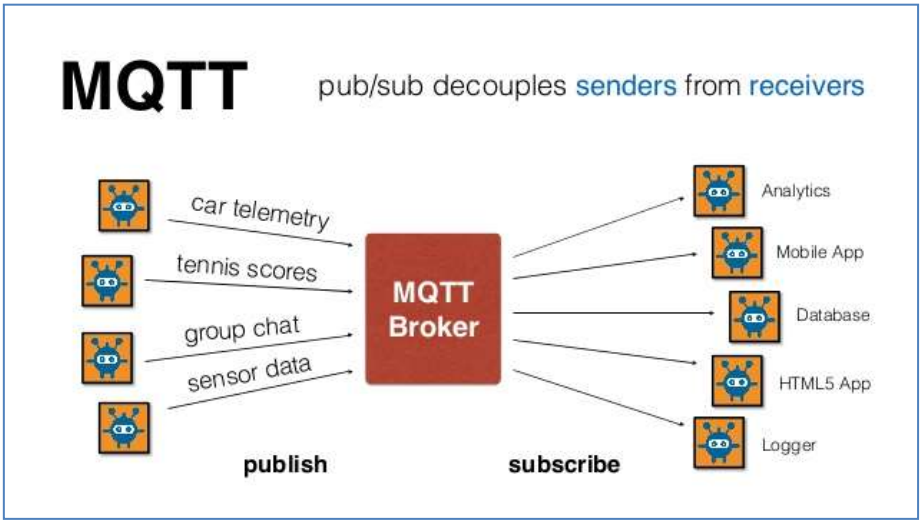

Figure 40 – MQTT description scheme

Data Distribution Service (DDS)

The Data Distribution Service (DDS™ [23]) used in Tampere and Vigo Pilo Sites, 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.

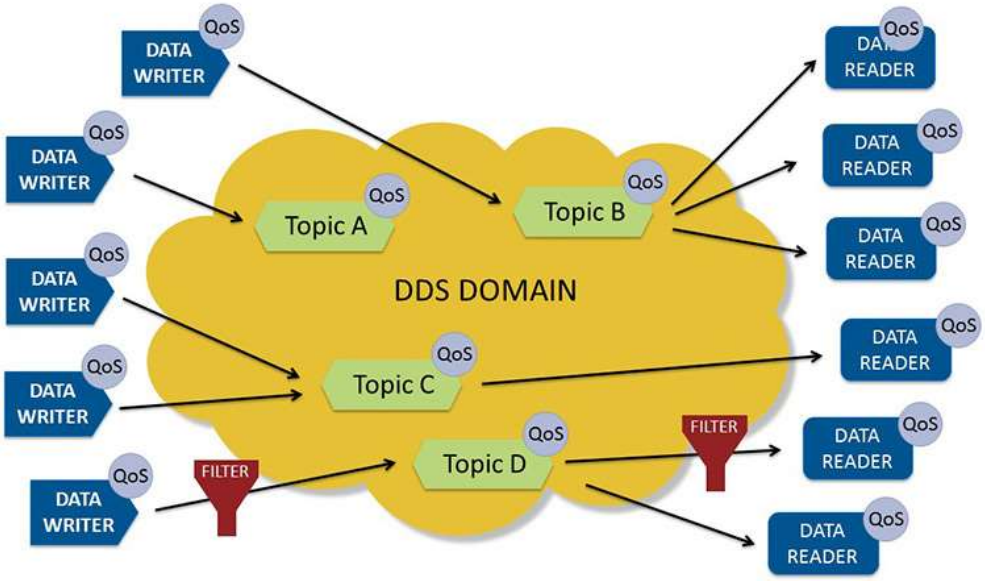

Figure 41 – DDS Architecture

Protocol Buffers (used by Links)

Protocol Buffers [22], used by Links in the Italian Test Site, 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 (used by LINKS)

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.

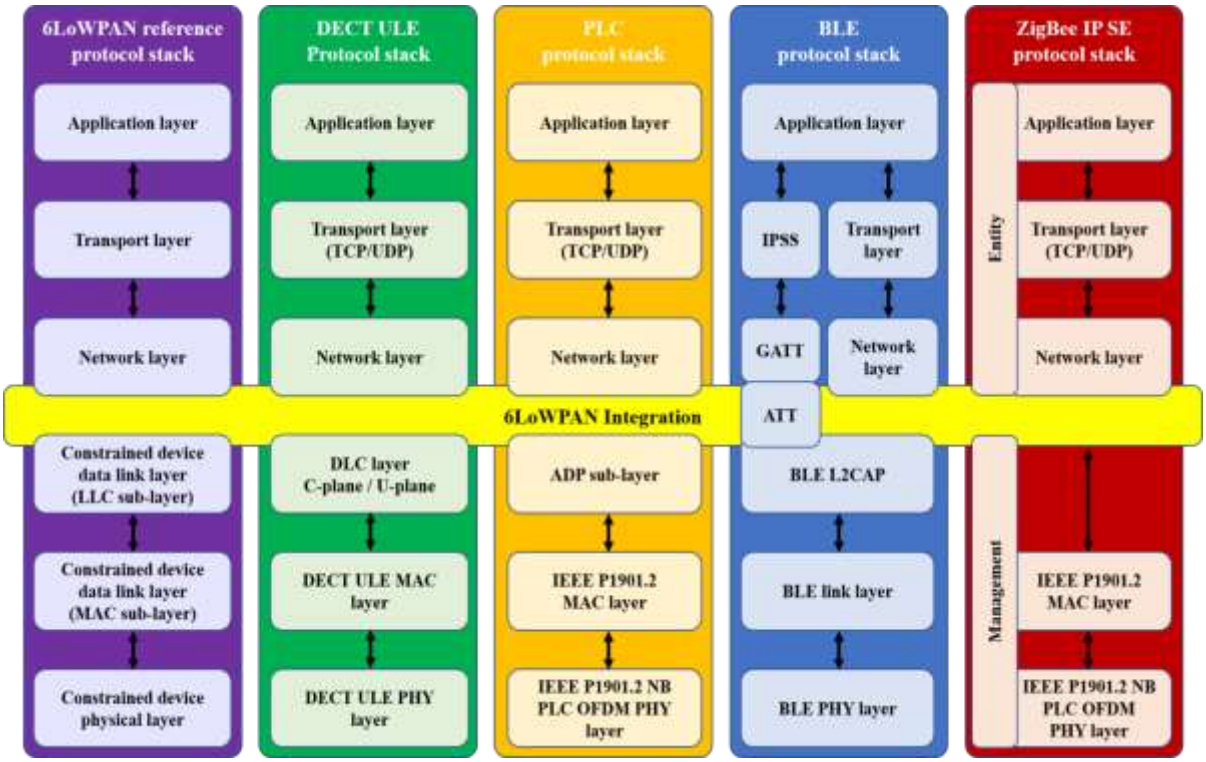

Figure 42 – 6LoWPAN integration

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 **[24]**.

Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) **[25]** 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 [7].

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 [26].

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 [27]

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 [28]; these reference points are also included in D1.3).

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 D1.3. 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 which has the ability to dynamically discover semantically annotated information in the oneM2M system [29].

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 43) 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.

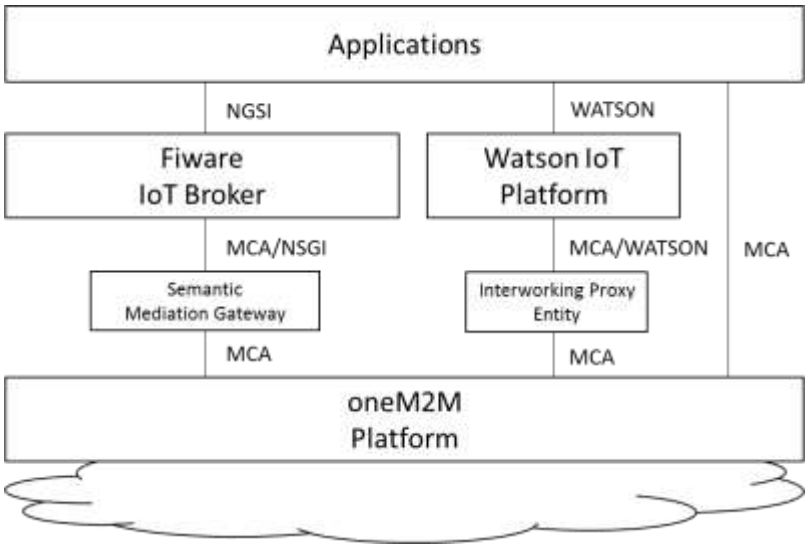

Figure 43 – Interworking through SMG and IPE

4.1.7 Application container or runtime environment

Java OSGI (Used by LINKS)

The OSGi Alliance [30] 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 [31].

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.

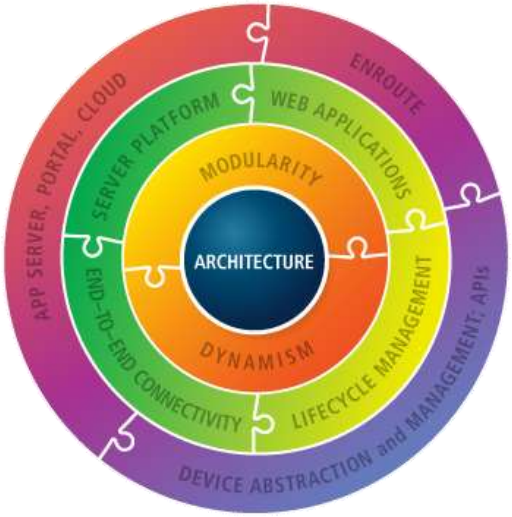

Figure 44 – OSGi and IoT similarities (source: OSGi Alliance)

Python iPOPO (Used by LINKS)

iPOPO [32] 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 [33][33] . 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 Specification of In-Vehicle IoT platform in the different Pilot Sites

For every Pilot 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 chapter § 3.1.

4.2.1 In Vehicle IoT Platform of Tampere Pilot Site

Table 6 – In Vehicle IoT Platform of Tampere Pilot Site

Functionality	Tampere Pilot Site 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.
Data Management	DDS
IoT Device Adaptation	DDS & MQTT
OEM Systems Communication	DDS + CANbus connection
IoT In-vehicle components	no additional in-vehicle sensors were added, In-vehicle HMI
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.

The following Table 7 reports a list of signals/parameters related to IF5 interface, introduced in §2.3, that the Intra-Vehicle Network makes available through DDS to the in-vehicle IoT platform.

Table 7 – Vehicle data (IF5 interface) implemented in Tampere Pilot Site

Signal/Parameter	Source
Vehicle Reference Position	RTK-GPS receiver + IMU
Vehicle Heading	RTK-GPS receiver + IMU
Vehicle TimeStamp	NTP
Vehicle Speed	RTK-GPS 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 In Vehicle IoT Platform of Versailles Pilot Site

Table 8 – In Vehicle IoT Platform in Versailles Pilot Site

Functionality	Test Site France implementation
Remote Management	Intempora RTMaps framework
Context-awareness	Vehicle environment (road object, lane markings, etc.) as outcome of data fusion from multiple sensors
Syntactic and Semantic Interoperability	oneM2M
Data Management	Intempora RTMaps framework
IoT Device Adaptation	Intempora RTMaps bridges (oneM2M, MQTT)
OEM Systems Communication	Intempora RTMaps, UDP, TCP
IoT In-vehicle components	Intempora RTMaps, UDP, TCP
OEM In-vehicle components	Intempora RTMaps, UDP, TCP
Application container or Runtime Environment	Linux Ubuntu Intempora RTMaps

Table 9 – Vehicle data (IF5 interface) implemented in Versailles Pilot Site

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	CAN bus
Vehicle TimeStamp	CAN bus
Vehicle Speed	CAN bus
Vehicle Longitudinal & Lateral Accelerations	IMU
Vehicle YawRate	CAN bus
Vehicle Vertical Acceleration	CAN bus
Vehicle Battery Life	CAN bus
Object Relative Localization	Vision System

4.2.3 In Vehicle IoT Platform of Livorno Pilot Site

Table 10 – In Vehicle IoT Platform of Livorno Pilot Site

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	- MQTT - CANbus - Extendable with other relevant protocols
IoT In-vehicle components	- Sensors for pothole detection (accelerometer, smartphone, IMU)
OEM In-vehicle components	- CONTI E-Horizon (IoT connected)
Application container or Runtime Environment	OSGi framework: - Felix iPOJO - Pelix iPOPO

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, different IoT communication protocols with the devices are supported, 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).

In brief, the in-vehicle IoT platform of Livorno Pilot Site is a modular software deployed on the On Board Unit (OBU). It is an IoT gateway, that allows to communicate with the outside through IoT or ITS (LTE communication, ITS G5) standards, which can send or receive critical information produced by the IoT system through V2V / V2X standard messages.

This IoT platform can be easily deployed on OBU based on different hardware components through a docker based approach.

Hereafter, a list of signals/parameters related to IF5 interface, introduced in §2.3 that In-Vehicle IoT Platform will get from Intra-Vehicle Network.

Table 11 – Vehicle data (IF5 interface) implemented in Livorno PIlot Site

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
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 12 – Additional IoT devices data (IF6 interface); Livorno Pilot Site vehicles

Signal/Parameter	Source
Acceleration Status	CAN or 6LoWPAN Inertial Sensors
Motion Detection Status	Smartphone

4.2.4 In Vehicle IoT Platform of Brainport Pilot Site

4.2.4.1 TNO prototype

Table 13 – In Vehicle IoT Platform of TNO vehicle

Functionality	TNO 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 (ETSI ITS G5). 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.

Hereafter the list of signals/parameters related to IF5 interface, introduced in §2.3, that the In-Vehicle IoT Platform will get from Intra-Vehicle Network.

Table 14 – 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
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

Table 15 – In Vehicle IoT Platform of NEVS vehicle

Functionality	NEVS implementation
Remote Management	ROS framework (*In coordination with Brainport*)
Context-awareness	*In coordination with Brainport*
Syntactic and Semantic Interoperability	oneM2M (*In coordination with Brainport*)
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 16 – Vehicle data (IF5 interface); NEVS prototype

Signal/Parameter	Source
Vehicle Speed	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
Steering Wheel Angle	CAN bus
Vehicle Yaw Rate	CAN bus
Wheel Speeds	CAN bus
Gear	CAN bus
Accelerator Pedal Position	CAN bus
Brake Pedal Status	CAN bus
Steering Torque	CAN bus

4.2.5 TUEIN prototype

The following description for In-Vehicle IoT Platform implementation can be considered for TUEIN prototype.

Table 17 – In Vehicle IoT Platform of TUEIN vehicle

Functionality	TUEIN implementation
Remote Management	-
Context-awareness	Probabilistic world model using camera input + IoT connectivity - WIRE - Data annotation with GPS timestamp for G5 connection
Syntactic and Semantic Interoperability	- OneM2M (MQTT) - HUAWEI EAI platform (HTTPS) - ROS framework
Data Management	- ROS framework bridges (oneM2M) - Technolution IoT GateWay (for LTE & ITS-G5 datalogging)
IoT Device Adaptation	- ROS Framework
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 16.04 RTMaps 4.6.0 ROS Kinetic Docker

Within AUTOPILOT the main challenge was to implement the vehicle as part of the Internet of Things. Two main routes are available to make a vehicle an IoT connected device. One route is to use a ETSI ITS 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 45 shows the general gateway layer in IoT devices to connect to applications through an 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.

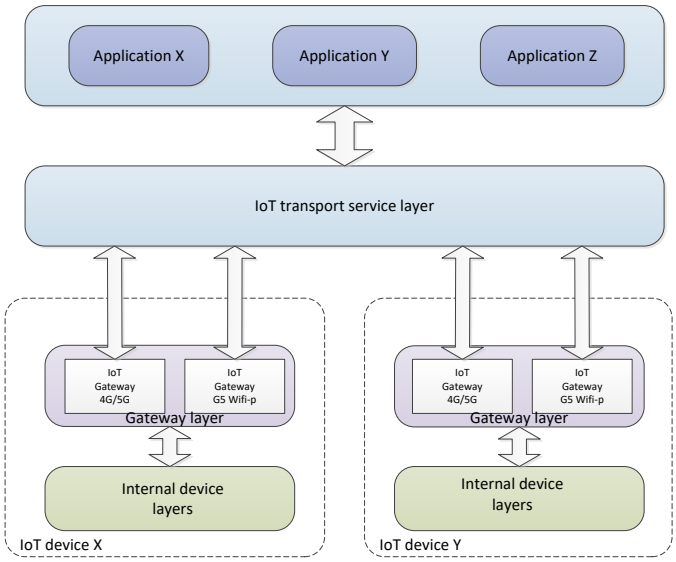

Figure 45 – General gateway layer in IoT devices

Flowradar

The Technolution on board Flowradar unit provides the ETSI ITS 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.

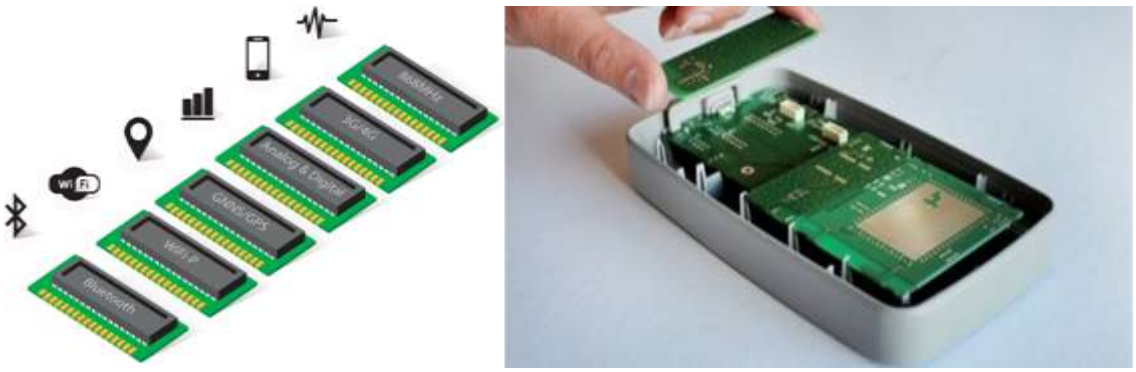

Figure 46 – Modular principle of the Flowradar G5 gateway

Basic functionality of the Flowradar is the ETSI ITS 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.

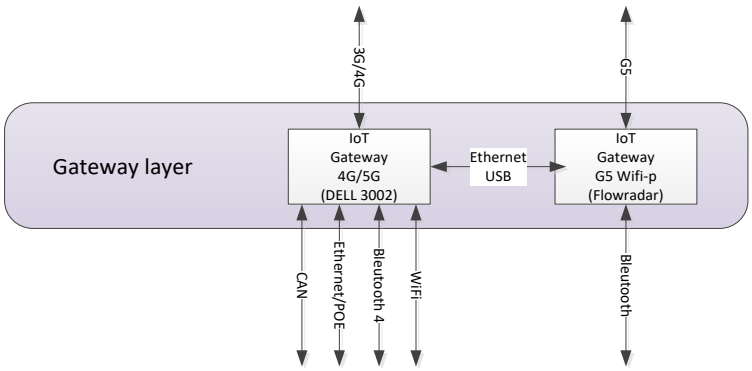

Figure 47 – Gateway layer functionality filled in with two interconnected units, one for the ETSI ITS G5 access and one for 4G.

In car internal connectivity of the gateway layer

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

In car external connectivity of the gateway layer

- ETSI ITS G5 / IEEE 802.11p
- 3G/4G LTE

Gateway protocols

- ETSI ITS G5 gateway protocols: CAM messages according to the ETSI standards
- 4G/5G gateway protocols: OneM2M standards

Gateway additional functions

- GPS (gps time used for time synchronization of ETSI ITS 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 (adapted for the pilotting for use with VRU) messages.

Connection with OneM2M MQTT and protobuf

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

The following Table 18 report a list of signals/parameters related to IF5 interface, introduced in §2.3, that In-Vehicle IoT Platform will get from Intra-Vehicle Network.

Table 18 – 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

4.2.5.1 VALEO prototype

Table 19 – In Vehicle IoT Platform of VALEO vehicle

Functionality	VALEO 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

The RTMaps middleware synchronizes the arrival of low-level signals from various sensors (cameras,

lidar, CAN data). Several RTMaps components handle the signal processing. The additional Valeo Smart Platform relies on MQTT to enable data exchanges between in-vehicle components.

The following table reports a list of signals/parameters related to IF5 interface that In-Vehicle IoT Platform will get from Intra-Vehicle Network

Table 20 – Vehicle data (IF5 interface); VALEO prototype

Signal/Parameter	Source
Vehicle Reference Position	GNSS receiver
Vehicle Heading	GNSS receiver
Vehicle Speed	CAN bus
Steering Wheel Angle	CAN bus
Vehicle Longitudinal & Lateral Accelerations	CAN bus
ESP Yaw Rate	CAN bus
Car suspensions	CAN bus
Wheel speed	CAN bus
Brake Pressure	CAN bus
Acceleration	IMU System
Pitch Angular velocity	IMU System
Car positioning	Vision System
Object classification	Vision system
Object relative localization	Vision system

4.2.6 In Vehicle IoT Platform of Vigo Pilot Site

Table 21 – In Vehicle IoT Platform of Vigo Pilot Site

Functionality	Test Site Spain 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.
Data Management	DDS
IoT Device Adaptation	DDS & MQTT
OEM Systems Communication	DDS + CANbus connection
IoT In-vehicle components	no additional in-vehicle sensors were added, In-vehicle HMI
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

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 section 3.1.1 and 3.1.7.

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 DDS 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 22 – Vehicle data (IF5 interface) implemented in Vigo Pilot Site

Signal/Parameter	Source
Vehicle Reference Position	RTK-GPS receiver + IMU
Vehicle Heading	RTK-GPS receiver + IMU
Vehicle TimeStamp	NTP
Vehicle Speed	RTK-GPS 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

5 Conclusions and outlook

At the end of the project, most of the envisaged prototype concepts have been implemented. Deviations from the original plan mainly concern technical details at prototype level, while at Pilot Site level the IoT implementation initially planned in WP1 has been followed.

The final specifications for an open in-vehicle IoT platform have been defined, and are included in D1.6 as requirements and high level view (updating the D1.5), while the detailed IoT integration specifications for each prototype are provided in D2.1.

In-vehicle IoT platform openness has been addressed in terms of minimum common dataset that thas to be made available from the vehicle to the IoT (IF5, interface between the in-vehicle network and the in-vehicle IoT platform, defined in sections 1,2 and specified for the prototypes in section 3. Regarding in-vehicle IoT platform interoperability (see also D2.1 conclusions) the focus has been to prove IoT concepts and use cases, rather than having a common implementation. As result, the technical solutions on vehicle prototypes show different instantiations of IoT, based on a common architectural view (chapter 2). Each IoT platform prototype demonstrates interoperability within vehicle components and with the IoT infrastructure of the pilot site.

In most cases in-vehicle IoT platform has achieved Technology Readiness Level of 6 (technology demonstrated in relevant environment) thanks to the activities performed in all the test sites. Some cases achieve a TRL 7 (system prototype demonstration in operational environment). Some prototypes are TRL 3 and do not have an open IoT yet, but they have been kept in the deliverable as they are an useful proof of concept of services that could be part of an open IoT in the future.

Interviewing the prototype owners, most of the lessons learned after implementation and piloting had to do with integration: in the Tampere Pilot Site a more modular approach would improve reusability of the developed components and allow the integration of more IoT nodes. In Brainport, the motion planning functionality used for Platooning and Automated Valet Parking should be further enhanced to include data about other road objects in the vehicle world model. The Highway Pilot integration in Brainport was a trade-off between exploiting existing platforms and the IoT, therefore is not a complete solution, which would enable the connection to more IoT objects. In the Urban Driving use case in Brainport, the choice for using multiple IoT platforms (OneM2M & HUAWEI EAI) and using different communication protocols (MQTT & HTTP) proved very useful to create a redundant system. Since during piloting some IoT clouds systems were not performing as expected, Brainport partners were able to partially continue tests anyway with the other working system.

In Versailles PS specifically, the lessons learned involved the limits and improvement margins for communication technology, both cellular and 802.11OCB based. The feedback from Livorno Pilot Site was mainly on the improvements of data processing function by the on board units.

We have also to consider that piloted solutions are using existing connectivity, while the incoming 5G deployment is expected to be highly beneficial to the Internet-of-Things. It can be reasonably stated that the forthcoming 5G based platforms used for connecting the vehicle to the IoT can reach higher level of maturity and be fully validated in the incoming 2-3 next years. Exploitation will involve highly automated passenger cars, shuttles, commercial as well as industrial vehicle automation, off-road vehicles, such as forestry, harbor and mining equipment, where the vehicle has to be integrated with operational management systems.

Another aspect is the use of the IoT backend system by new ITS service providers. For instance in case of Platooning, results are exploited by upgrading platooning functionality in freight vehicles as part of B2B contracts, particularly with respect to remote platoon guidance via the cloud-based platoon service. In both the Platooning and Automated Valet Parking application the software suite developed in AUTOPILOT will be ported to a new car laboratory which natively supports AD on a

higher SAE level.

Further pilot outcomes will be addressed by the evaluation WP (T4.2) while exploitation perspectives will be more extensively treated in WP5 (T5.4).

6 References

[1] AUTOPILOT Deliverable D1.5, “Initial Open IoT Vehicle Platform Specification”

[2] Technology readiness levels, in General Annexes, Horizon 2020, Work Programme 2018-2020

[3] AUTOPILOT Deliverable D2.1, “Vehicle IoT Integration Report”

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[21] 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

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

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

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

[25] Bluetooth Low Energy (BLE): http://www.bluetooth.com/Pages/Low-Energy.aspx/

[26] "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
[27] “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
[28] "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.
[29] Kovacs, Erno, et al. "Standards-Based Worldwide Semantic Interoperability for IoT." IEEE Communications Magazine 54.12 (2016): 40-46.
[30] OSGi Alliance: https://www.osgi.org/
[31] OSGi architecture: https://www.osgi.org/developer/architecture/
[32] Python iPOPO: https://ipopo.readthedocs.io/en/latest/
[33] ROS: http://www.ros.org/about-ros/
[34] H2020 Reference for Technology Readiness Level from H2020 Work Program 2018-2020