Grant Agreement Number: 731993

Project acronym: AUTOPILOT

Project full title: AUTOmated driving Progressed by Internet Of Things

D 2.1

Vehicle IoT Integration Report

Due delivery date: 30.06.18

Actual delivery date: 28.06.2018

Organization name of lead participant for this deliverable: CRF

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

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

AUTOPILOT

Document Control Sheet

Deliverable number:	D 2.1
Deliverable responsible:	Visintainer Filippo – CRF
Workpackage:	WP 2
Editor:	Altomare Luciano, Galli Mauro, Visintainer Filippo – CRF

Author(s) – in alphabetical order		
Name	**Organisation**	**E-mail**
Alén Gonzalez, Silvia	CTAG	Silvia.alen@ctag.com
Alesiani, Francesco	NEC	Francesco.Alesiani@neclab.eu
Altomare, Luciano	CRF	luciano.altomare@crf.it
Belz, Joerg	DLR	Joerg.Belz@dlr.de
Bosi, Ilaria	ISMB	bosi@ismb.it
Bosma, Jan	TECHN	jan.bosma@technolution.nl
Brevi, D.	ISMB	brevi@ismb.it
Daalderop, Gerardo	NXP	Gerardo.Daalderop@nxp.com
De Souza Schwarz, Ramon	TNO	ramon.desouzaschwartz@tno.nl
Den Ouden, Jos	TU/e	j.h.v.d.ouden@tue.nl
D'Orazio, Leandro	CRF	leandro.dorazio@crf.it
Galli, Mauro	CRF	mauro.galli@crf.it
Kaul, Robert	DLR	robert.kaul@dlr.de
Legaspi, Xurxo	CTAG	xurxo.legaspi@ctag.com
Marcasuzaa, Hervee	VALEO	herve.marcasuzaa@valeo.com
Markowski, Robert	DLR	robert.markoswki@dlr.de
Marimuthu, Balraj	NEVS	balraj.marimuthu@nevs.com
Petrescu, Alexandre	CEA	alexandre.petrescu@cea.fr
Simeon, Jean Francois	CONTINENTAL	Jean-Francois.Simeon@continental-corporation.com
Simonetto, Andrea	IBM Ireland	Andrea.Simonetto@ibm.com
Scholliers, Johan	VTT	Johan.Scholliers@vtt.fi
Souza Schwartz, Ramon	TNO	ramon.desouzaschwartz@tno.nl
Visintainer, Filippo	CRF	filippo.visintainer@crf.it
Yeung, Michel	CONTI	michel.yeung@continental-corporation.com

Document Revision History			
Version	**Date**	**Modifications Introduced**	
		Modification Reason	**Modified by**
V0.1	14/11/2017	ToC	L. Altomare
V0.2	30/11/2017	Inputs from ISMB on TS IT in-vehicle IoT platform description, from CTAG on ITS Spain in-vehicle IoT platform description, TNO on TS NL verification tables	Bosi, ISMB, Alen, C-TAG, Souza Schwartz, TNO Minor modification by F. Visintainer, CRF
V0.3	10/01/2018	Integrated contributions by NEC, IBM, TECHN, NXP, TNO, TU/e	CRF based on partners' input
V0.41	21/02/2018	Updated contribution for TUe and TECH	Jan Bosma, TECH
V0.5	21/02/2018	Integrated first contribution by TS France; restructured TNO prototype IoT verification	F. Visintainer CRF, A. Petrescu
V0.6	28/02/2018	Integrated partners contributions	F. Visintainer
V0.61	11/04/2018	Integrated partners contribution: Data logging and management, and Conti contribution	Visintainer F. Galli M. CRF, Scholliers J. VTT, Petrescu A. CEA, Simeon JF. Continental, Brevi D. ISMB, Marcasuzaa H. VALEO, Souza Schwartz R. TNO, Marimuthu B. NEVS
V0.7	19/04/2018	Formal revisions	F. Visintainer
V1.0	24/05/2018	Integrated chapter on data logging	F. Visintainer, M. Galli, L. D'Orazio, CRF, D. Brevi, ISMB
V1.1	04/06/2018	Full draft of chapter on data logging	F. Visintainer, L. D'Orazio, CRF
V1.2	06/06/2018	Includes TNO full revision of related prototype chapter (received 6/6), cross-check and refinement of logging chapter 3. Added references to T2.5	R. De Souza Schwarz, TNO, F. Visintainer, L. D'Orazio, M. Galli, CRF
V1.2	08/06/2018	Added DLR, TUEIN, CTAG, VTT input; completed missing parts	F. Visintainer, CRF, J. Belz, DLR, X. Legaspi, CTAG, J. Scholliers, VTT, J. Den Ouden, TUEIN
V1.3	11/06/2018	Edited peer review version	F. Visintainer, CRF
V1.4	14/06/2018	Modified after CERTH peer review	F. Visintainer, CRF
V1.5	28/06/2018	Modified after AKKA peer review	F. Visintainer, CRF
V1.6	28/06/2018	Integrated ISMB, DLR, NEC, TU/e, VTT, CEA latest updates	F. Visintainer, CRF

V2.0	28/06/18	Final check for submission	R. Bhandari, ERTICO

Abstract
This document reports the IoT integration into the vehicle within the AUTOPILOT project. The work is the result of Task 2.1. It describes how IoT is deployed on the vehicles of the different AUTOPILOT sites, which are the main software components, the connectivity within the general IoT platform and the interface with the in-vehicle system.

Legal Disclaimer

Abbreviations and Acronyms

Acronym	Definition
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks (IoT comm. technology)
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
CAM	Cooperative Awareness Message (ITS G5 message type)
CAN	Controller Area Network
C-ITS	Cooperative Intelligent Transportation Systems
CTS	Central Test Server
DDS	Data Distribution Service
DENM	Decentralized Notification Message (ITS G5 message type)
DGNSS	Differential Global Navigation Satellite System
EC	European Commission
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
GA	Grant Agreement
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMI	Human Machine Interaction
ICT	Information and Communication Technology
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
ITS G5	Wireless Short range comm. in the ETSI standard
JSON	JavaScript Object Notation
LDM	Local Dynamic Map
LiDAR	Light Detection And Ranging
LTE	Long Term Evolution
MAP	Map message (ITS G5 message type: local topology)
OBU	On Board Unit
OEM	Original Equipment Manufacturer
PF	Platform
PO	Project officer
ROS	Run-time environment
RSU	Road Side Unit
PS	Pilot Site
RTK	Real Time Kinematics
SPAT	Signal Phase and Timing (ITS G5 message type)
TS	Test Site
UI	User Interface
UWB	Ultra-Wide Band
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
XML	Extensible Markup Language

Table of Contents

List of Figures

List of Tables

Executive Summary

Work-package 2 (WP2) of the AUTOPILOT project focusses on the integration of the IoT platform in the vehicles and other devices, to enable AUTOPILOT AD enhancement through the IoT, in the use cases of Automated Valet Parking, Urban Driving, Highway Driving, Highway Pilot and Car Rebalancing for Shared Vehicles.

Within WP2, two specific tasks are dedicated to the in-vehicle systems: T2.1, which aims at integrating vehicles into the IoT world, and T2.2 which aims at the development and adaptation of autonomous driving functions for the intended use cases. This deliverable refers to task T2.1, whose scope is to allow vehicles to access and use IoT devices and capabilities, thus enabling AD functions enhancement of T2.2 and contribute to IoT applications such as car rebalancing.

Based on the design carried out in T1.3 (D1.5 Initial Open IoT Vehicle Platform Specification) task T2.1 is dedicated to hw/sw development for the AUTOPILOT connected prototype. In particular, it delivers the in-vehicle IoT platform and related components, enabling the vehicle to interact within the AUTOPILOT Internet-of-Things. The in-vehicle components dedicated to IoT developed in the different AUTOPILOT sites, as baseline act as gateways between the internal network and functions (including Autonomous Driving) of the several vehicles, thus making the vehicle as “IoT” device within the whole eco-system; but in most cases AUTOPILOT in-vehicle IoT also includes the capability of “edge” functionalities and applications, such as environment perception; this motivates the naming and approach of a “vehicle IoT platform” addressed in T1.3 and continued in T2.1.

Already in WP1, “prototype” leaders had been identified, for the definition of their own In-Vehicle system and its components, both concerning the IoT platform and the legacy components. In T2.1 the same prototype leaders, jointly with the partners supplying components, are integrating the IoT platform on board the vehicle to contribute to the use cases which are defined at pilot site level. These components have been preliminarily tested within T2.1, following a basic plan as proposed by prototype leaders, and intended to verify the basic functionalities of the units. This will enable the more systematic readiness verification of T2.5 and then the piloting of IoT systems in WP3.

Since T2.1 role is devoted to the in-vehicle integration, this task also contributed to data recording and management within the AUTOPILOT project, focussing on the vehicle data and defining a format for data recording based on WP4 requirements which is aligned with the other data recording work (V2X logging, application logging). A certain degree of freedom has been left to prototype implementation, with the condition that management procedures are aligned with T3.4 and that the data format is met at the end of the data transfer chain in the pilots.

Chapter 1 introduces the concept of Internet of Things integration on the vehicles, as adopted by AUTOPILOT.

Chapter 2 reports, per pilot site and prototype, the IoT implementation, giving a short description of the system architecture (coherently with D1.5), the individual software components constituting the in-vehicle IoT system, their interfaces and functional verification.

Chapter 3 gives an overview of the recording and management of data coming from the vehicles, and focusses on the data format for evaluation, as defined by T2.1.

1 Introduction

1.1 Purpose of Document

Task T2.1 aims to develop and verify integration of the IoT in-vehicle platform components. This deliverable reports the integration of the In-Vehicle IoT platform components planned for AUTOPILOT prototypes.

1.2 AUTOPILOT IoT and the in-vehicle system

In the task T3.1 the in-vehicle IoT platform initial specifications were preliminarily provided. Partners agreed on the logical representation of the in-vehicle system, identifying the role and placement of the in-vehicles IoT platform and the main in-vehicle components interacting with it. Based on this, its main functionalities were identified, and are reported thoroughly in D1.5 [1]. To facilitate this document, it is useful to briefly recall the introductory picture (Figure 1). In particular, it shows the needed integration to receive information from in-vehicle components, connect with the cloud IoT system, but also to third party cloud services and to vehicular ad hoc networks (ETSI ITS G5).

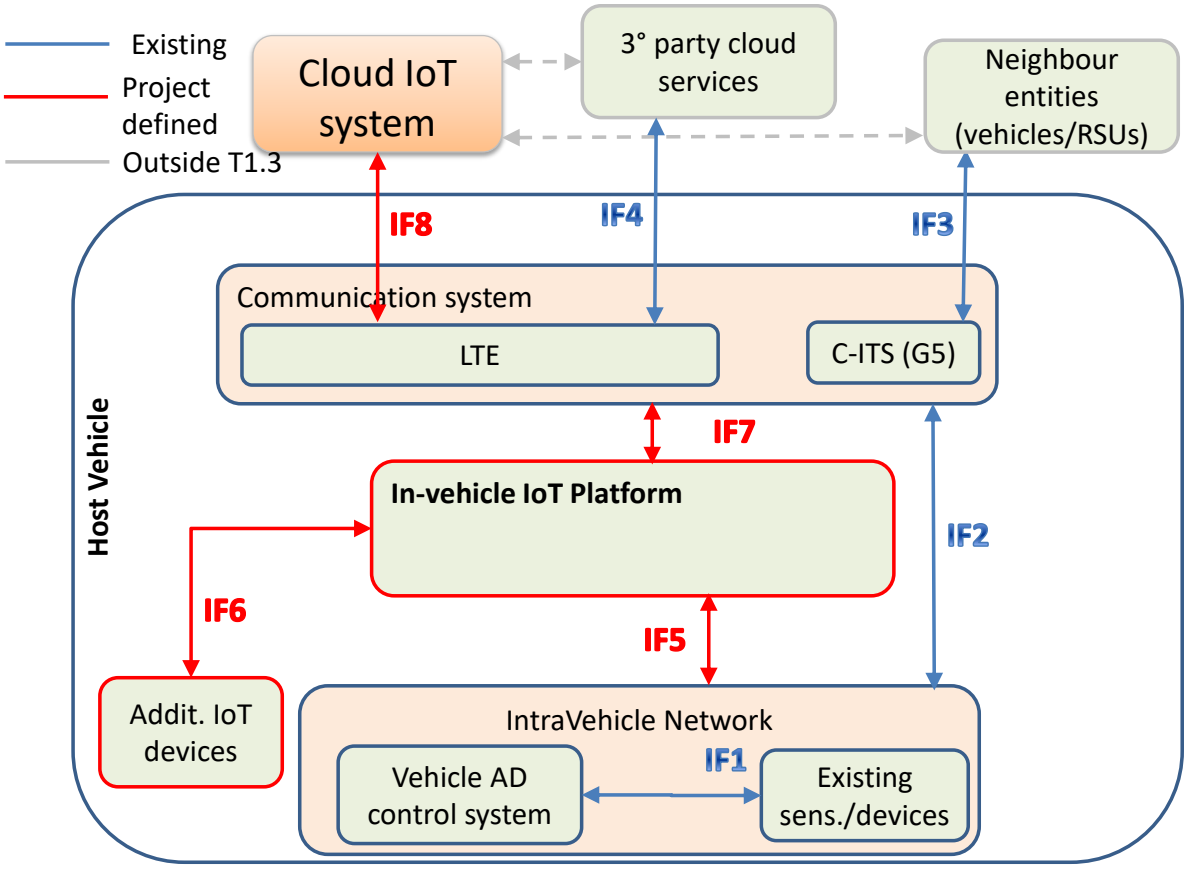

Figure 1: In-Vehicle IoT platform (red box) with the vehicle concept scheme; source D1.5

Another useful picture from D1.5 is Figure 2, showing the vehicle within the general IoT architecture: the vehicle is an IoT device, an edge computing unit and a gateway for other devices (including the vehicle network and its peripherals). It is a mobile node publishing and receiving contents from and to the IoT Platform, thanks to its connectivity capabilities (ITS G5, LTE).

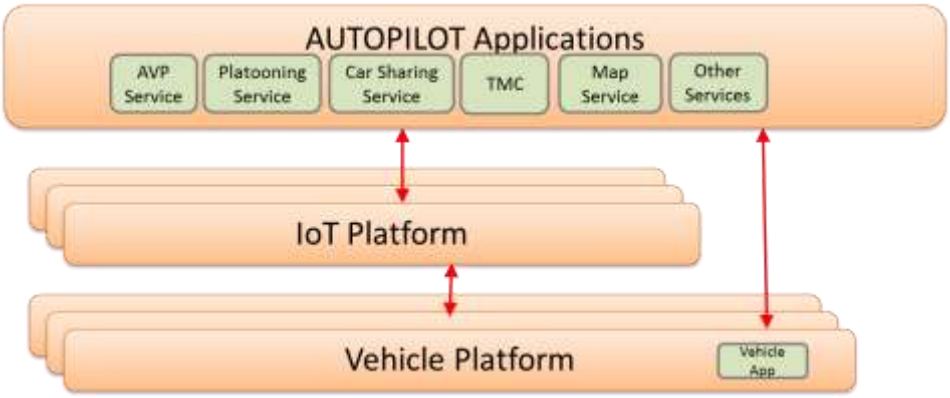

Figure 2: IoT High View Architecture: conceptual separation in AUTOPILOT

We can make a distinction between the IoT Platform as the set of functions that manages the IoT devices and entities, and the "Vehicle IoT Platform" as the complex entity that includes all the software and hardware components deployed in the vehicle. As a whole, IoT Applications may have a different level of integration within the vehicle: for example, a parking application can either run in the vehicle application container or in some smartphone connected to the vehicle. Such applications have typically a local processing and are then connected to the cloud counterpart to exchange service related information. But in general, applications that need information from a single vehicle, can access it through the Vehicle IoT Platform. That is why one of the key elements in T2.1 is the interface to the private vehicle network, and for the scope of AUTOPILOT on-site evaluation, one of the objectives of T2.1 is to contribute to data format definition.

2 In-vehicle IoT integration

This section provides a synthesis of approaches adopted in different Test Sites, with focus on software components designed to obtain the on board units and components for IoT that are integrated in the vehicles.

This chapter is structured in sections, according to the IoT prototypes implemented in the different pilot sites. In the Brainport section, sub-chapters refer to the different on board IoT implementations. Livorno section is unitary, as the on-board IoT platform is the same for all prototype vehicles.

Each prototype section starts with a summary, including the use cases where prototype are involved, how the IoT is used in the prototype, how it is implemented and the partners involved in the development of its IoT functionalities. Typically the implementation in a specific vehicle reflects the general integration scheme of T1.3 (Figure 1), is carried out by one partner, supported by partners providing specific components.

Then, the software architecture scheme of implemented IoT platform is provided, highlighting the software components that provide core functionalities and relationship between components.

After, brief descriptions of the main components that constitute the vehicle IoT platform follow: their role within the IoT, relationships with other components and main interfaces.

Finally, verification is reported, aimed at delivering checked on-board IoT functionalities to T2.5 [2] and to the pilots (WP3). Therefore, verification tests are structured in tables where specific items are checked. Tests should regard at least basic hardware/software functionalities of the on board IoT platform and related components developed within T2.1; the IoT connectivity with other components outside T2.1, especially the in-vehicle network and other IoT systems (e.g. OneM2M platform); data contents with respect to the designed data model. In addition, early verifications of key IoT functionalities on-board the vehicle is recommended.

2.1 Pilot Site Finland

2.1.1 Short summary

The VTT prototype consists of hardware devices (i.e. environmental perception sensors, inertial sensors and electronically controlled actuators) which are connected to Linux machines which process and integrate the information. To establish connectivity with IoT services and with other vehicles and roadside units (ITS-G5), two separate communication system units are deployed in the vehicle. Figure 3 shows the software architecture scheme defined for the following use cases:

- **Automated Valet Parking (AVP)**: connects the vehicle to the AVP service which provides parking lot availability and allocation as well as information about other road objects in the vicinity.
- **Urban Driving**: focuses on the interaction with traffic lights and legacy traffic, and on safety when dealing with vulnerable road users.

Figure 3. Hardware integrated in the VTT prototype vehicle

2.1.2 Software components

Figure 4 shows the software components of the in-vehicle IoT platform of the VTT prototype vehicle and how they connect to the IoT cloud platform. Communication between the in-vehicle components is based on Data Distribution Service (DDS).

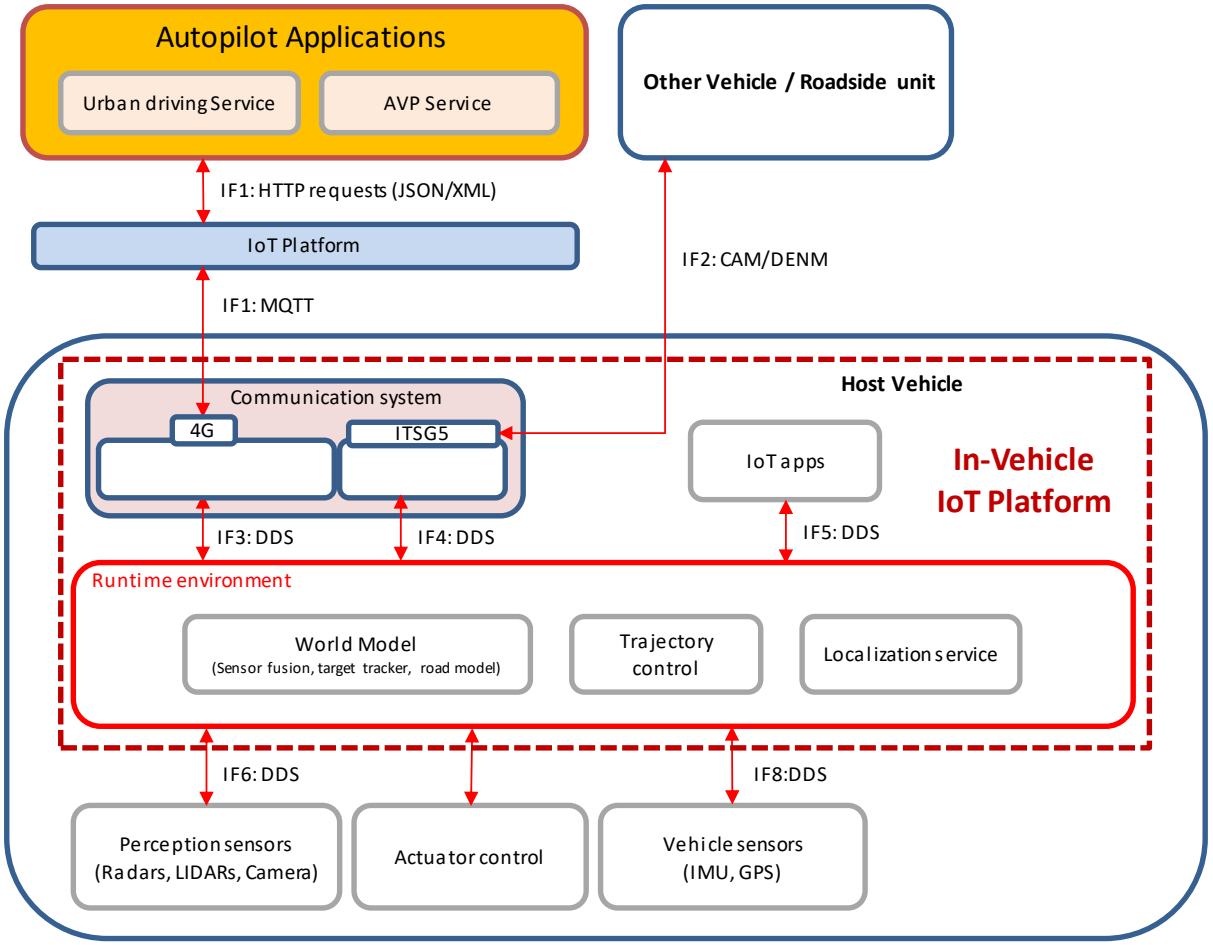

Figure 4: Vehicle scheme of software components, with In-Vehicle IoT platform

The main components of the in-vehicle IoT platform are:

- **Communication system** contains two communication system units:
  - Mobile communication unit (4G)
  - ITS-G5 unit from Dynniq. The ITS-G5 radio is used to broadcast/receive standard C-ITS messages (CAM, DENM, SPAT, MAP).
- **Run-time environment** runs on the Linux machine that aggregates and processes sensor and IoT data to be used by IoT apps and the AD unit:
  - World model: performs functions such as sensor fusion, target tracking and road model computation. It essentially aggregates data coming from multiple sensors, V2V communication and the IoT platform to build a road object level description of the world.
  - Trajectory planning: software component, that performs situational analysis, performs reactive obstacle avoidance and calculates the target heading and velocity as input for the vehicle actuators.
  - Localization service: correction of GPS position using IMU. Potentially also improvement of location using HERE's UWB approach.
- **IoT apps** generate and/or consume vehicle status and application-specific data that are exchanged with the IoT platform.
- **Intra-vehicle network** contains all other components outside the in-vehicle IoT platform:
  - Perception sensors: Radars, LIDARs and camera to be used for object and

environmental perception.
- AD unit: real-time platform running control-related algorithms.
- Vehicle sensors: other internal sensors such as IMU, and GPS.

2.1.3 Verification

The verification of VTT in-vehicle IoT platform envisages the following tests:

- FI-VTT-1: ITS-G5 connectivity
- FI-VTT-2: 4G IoT connectivity
- FI-VTT-3 IoT connectivity
- FI-VTT-4: Data Model – Traffic Camera object
- FI-VTT-5: Data Model - Traffic light status
- FI-VTT-6: Trip planning integration

Tests specifications are reported in the following tables.

Verification test ID	FI-VTT-1 (M15)
Test Title	ITS-G5 connectivity
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Basic test regarding the connectivity between the mobile communications unit in the vehicle and road side unit using ITS-G5.
Involved software components	Communication system.
How the test is realized	The test is performed in the lab and is used to test if the messages are transmitted correctly transmitted between different ITS-G5 devices.
Pass test criteria	Messages are correctly sent and received from one communication unit to another.
Results [yes / no]	Planned M19

Verification test ID	FI-VTT-2 (M15)
Test Title	4G IoT connectivity: basic test in lab
Link to T2.5	Anticipates Vehicle_safety_valet_parking and Vehicle_safety_urban_driving type verification
High level objective of the test	Basic test regarding the connectivity between the mobile communications unit in the vehicle and the IoT platform to verify if the messages are exchanged correctly.
Involved software components	Communication system
How the test is realized	The test is performed in the lab and is used to test if the messages are transmitted correctly exchanged between the vehicle and the IoT system.
Pass test criteria	Messages are correctly sent and received from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	FI-VTT-3 (M16)
Test Title	IoT connectivity (cellular)
Link to T2.5	Anticipates Vehicle_safety_valet_parking and Vehicle_safety_urban_driving type verification.
High level objective of the test	Integration test with the IoT platform.
Involved software components	Communication system, world model, IoT apps.
How the test is realized	Messages are generated and travel from the mobile road side unit or IoT device, pass through the IoT platform and are received by the vehicle and made available to the in-vehicle DDS.
Pass test criteria	Messages are correctly sent and received by the vehicle from the IoT platform, and content is made available to in-vehicle applications.
Results [yes / no]	Yes

Verification test ID	FI-VTT-4 (M16)
Test Title	Data model - traffic camera object
Link to T2.5	Anticipates Vehicle_safety_valet_parking and Vehicle_safety_urban_driving type verification.
High level objective of the test	Verification of camera object data model.
Involved software components	World model, communication system, IoT apps.
How the test is realized	Object Messages are generated by the traffic camera, and travel from the mobile road side unit and travel through the IoT platform to the vehicle, and are made available to the in-vehicle trajectory control.
Pass test criteria	Messages are correctly sent and received from the camera to the in-vehicle applications.
Results [yes / no]	Yes

Verification test ID	FI-VTT-5 (M17)
Test Title	Data model - traffic light status
Link to T2.5	Anticipates Vehicle_safety_urban_driving type verification.
High level objective of the test	Verification of traffic light status data model.
Involved software components	World model, communication system, IoT apps.
How the test is realized	Traffic light messages are received an integrated into the world model and made available to the trajectory planning module.
Pass test criteria	Messages are correctly received from the traffic lights and made available to the trajectory planning module.
Results [yes / no]	Planned M19

Verification test ID	FI-VTT-6 (M18)
Test Title	Data model - trip planning integration
Link to T2.5	Anticipates Vehicle_safety_valet_parking and Vehicle_safety_urban_driving type verification.
High level objective of the test	Verification of the trip planning data model and integration in the vehicle.
Involved software components	World model, communication system, IoT apps, trajectory planning.
How the test is realized	A trip plan, is transmitted through the IoT vehicle platform, received by the vehicle, and taken into account for trajectory planning.
Pass test criteria	Externally defined trip plan is executed correctly by the automated vehicle.
Results [yes / no]	Yes

2.1.4 Data logging and management

The data needed for the car-sharing service are sent to the service and be available for evaluation. Data will be logged at the service side and stored in different formats depending on the nature of the data (typically routes, ride-assignments, GPS positions, etc.). The collected data will be available for evaluation for work package.

2.2 Pilot Site France

2.2.1 Short summary

The IoT platform to be deployed in vehicle comprises several components: the Communication System (OBU – On Board Unit), the AD System (AD – Autonomous Driving Unit), the IHM and a few other computers. An overall view of the vehicle scheme is depicted in the following figure:

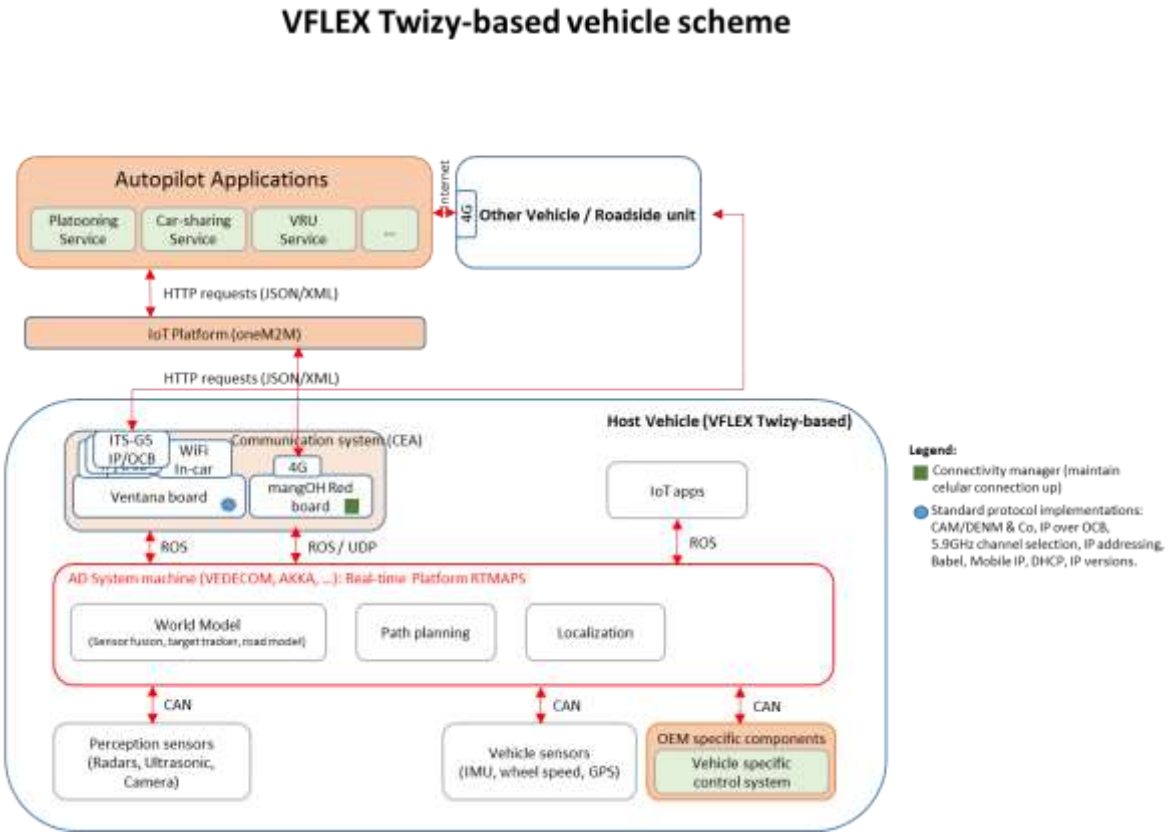

Figure 5: The software components for VFLEX Twizy-based in Versailles pilot site

The On-Board Unit (OBU) is made of the following hardware modules: a hardware module dedicated to the 802.11 OCB communications (titled 'Ventana' in the figure above) and a hardware module dedicated to LTE communications (titled 'mangOH Read'). The module dedicated to LTE communications contains several sensors that are qualified as 'Things'. Since the LTE module is connected to the Internet, this qualifies as an 'IoT'.

The LTE module is a Sierra Wireless mangOH Red board. This includes a Gyroscope/Accelerometer, a Pressure and Temperature sensor and a Light Sensor. The gyroscope/accelerometer sensor could be used for applications such as dead-reckoning with data fusion with GNSS receiver data such as to offer localization information in areas where GNSS data is unavailable (underground parking, tunnel, etc.) It could also be used for other applications such as shock-detection for airbag triggering, for acceleration confirmation and communication in platooning, and others. The Pressure sensor could be used for applications in need of altitude information, or as a micro-phone, or as theft intrusion detection, and others. The Temperature sensor could be used for slippery road warnings, and others. The Light sensor could be used for other applications. Each of the data units generated by these sensors can be communicated through the Internet.

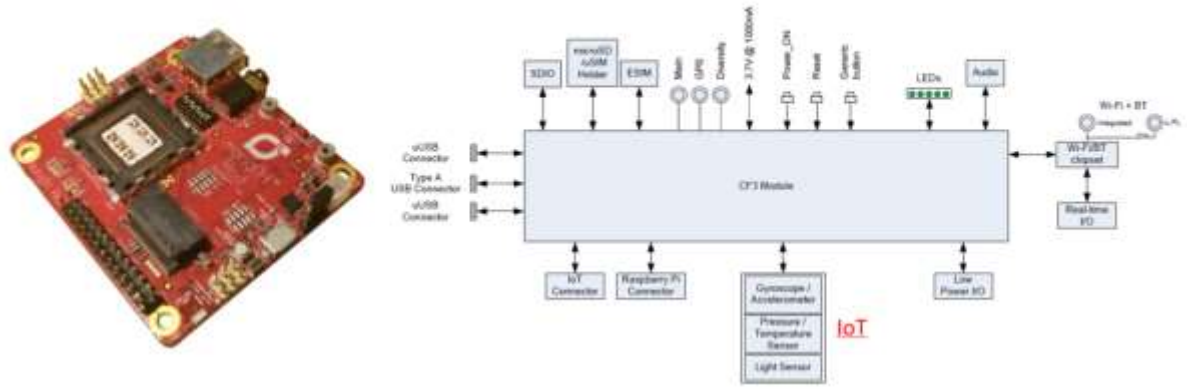

Figure 6: Sierra mangOH Red board and schematics of IoT sensors

2.2.2 Software components

The software components of the IoT platform to be integrated in the vehicle have three roles:

- Provide connectivity to the Internet for each computer in the vehicle that needs to be connected to the Internet
- Provide connectivity to the vehicles nearby
- Provide specific proprietary connectivity for legacy equipment such as existing, deployed Road-Side Units, with standard ETSI ITS-G5

The software components are the following:

- Internet Protocol version 4 and Internet Protocol version 6 addressing architectures for in-vehicle and between vehicles. Implementations of methods of forming IP addresses, starting from VIN numbers
- Static and potentially dynamic channel allocation software for an interference-free channel management in the 5.9GHz band
- Connectivity manager for maintaining the LTE connection while in movement
- Several protocol implementations for Internet Protocols: DHCPv6, DHCPv6 Prefix Delegation, DHCPv4, NAT, Babel, ICMPv6 Router Advertisement, Mobile IPv6, Mobile IPv4 with NEMO extensions (Network Mobility)
- Protocol implementations for potentially new protocols that may satisfy needs arising during project execution: CAM over LTE, CAM over IP, and potentially others
- Protocol implementations for backward compatibility with proprietary legacy systems: ETSI CAM, DENM and potentially GeoNetworking

The software components are divided in two main parts:

- Components derived from software available as open-source. Several protocol implementations are available as open source: Babel, DHCP, Mobile IP, NAT, NEMO, CAM/DENM, GeoNetworking
- New software written from scratch to implement extensions and improvements:
    - Connection keeper to maintain the LTE connection up during all mobility events
    - CAM implementation for backwards compatibility with legacy equipment
    - Static formation of IP addresses from VIN numbers
    - RA-based prefix exchanges for maintaining IP connectivity between cars in a platoon
    - Others

The software for implementation of protocols needed for backward compatibility to ETSI

CAM/DENM and GeoNetworking available as open source at github is the following:

- Vanetza implementation
- RendITS implementation
- GeoNetworking implementation derived from RendITS
- BTP SAP implementation
- DriveITS implementation

The analysis of the various characteristics of these software packages led to the establishment of a direction of development based on Vanetza. The Vanetza open-source platform uses the BOOST C++ library. This is very pertinent as it is included in the C++11 standard.

2.2.3 Verification

The verification of CEA in-vehicle IoT platform envisages the following tests:

- FR-CEA-1: Cellular IoT connectivity – Platooning
- FR-CEA-2: V2V connectivity
- FR-CEA-3: V2I connectivity

Verification test ID	FR-CEA-1
Test Title	Cellular IoT connectivity - Platooning
Link to T2.5	Anticipates IoT_platform and Vehicle_safety_platooning type verification.
High level objective of the test	End-to-end information exchange test between vehicle and oneM2M IoT platform for the platooning use case. Connectivity based on commercial cellular network.
Involved software components	OEM specific components, world model, IoT bridge, vehicle IoT apps, communication system, IoT platform (oneM2M)
How the test is realized	Messages are defined and generated for the platooning use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Basic IoT platform connectivity tested

Verification test ID	FR-CEA-2
Test Title	V2V connectivity
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	End-to-end information exchange test between AD System in one vehicle and AD System in a vehicle nearby.
Involved software components	Protocol implementations on Ventana; ping and RTMaps on AD System
How the test is realized	Ping first, then, eventually, messages from RTMAPS.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Planned
Verification test ID	FR-CEA-3

Test Title	V2I connectivity
Link to T2.5	Anticipates ITS-G5 type verification
High level objective of the test	End-to-end information exchange test between IHM (Tablet: Interface Homme-Machine, HMI: Human-Machine Interface) and Traffic Lights.
Involved software components	Protocol implementations on Ventana; ping the RSU, ping the Traffic Light Controller.
How the test is realized	Press button on IHM, turn the light Green on Traffic Lights.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes

2.2.4 Data logging and management

The data can be recorded in several ways.

The most straightforward mechanism to record data is the use of packet dump tools. The IP-OBU is composed of two distinct boards: the Ventana and the mangOH Red. Each of these boards runs an independent *linux* operating system. It is possible to run the *tshark* tool on Ventana, and the *tcpdump* tool on mangOH Red. Each of these tools can save the captures as "*.pcap*" files. Such a file contains each packet sent and received on the specified interface. The quantity of saved information (packet contents, time resolution, time specification, signal strength) can be specified as parameters in the command line.

It should be noted that memory space should be properly allocated (memory cards, SSD disk extensions) in order to accommodate the large quantity of data that could be recorded by the packet dump tools.

In addition to the packet dump tools, there exist a large number of software tools that can be used to record more data about the IP-OBU. The Linux Performance Observability Tools are illustrated in the following figure, with the reference to the source.

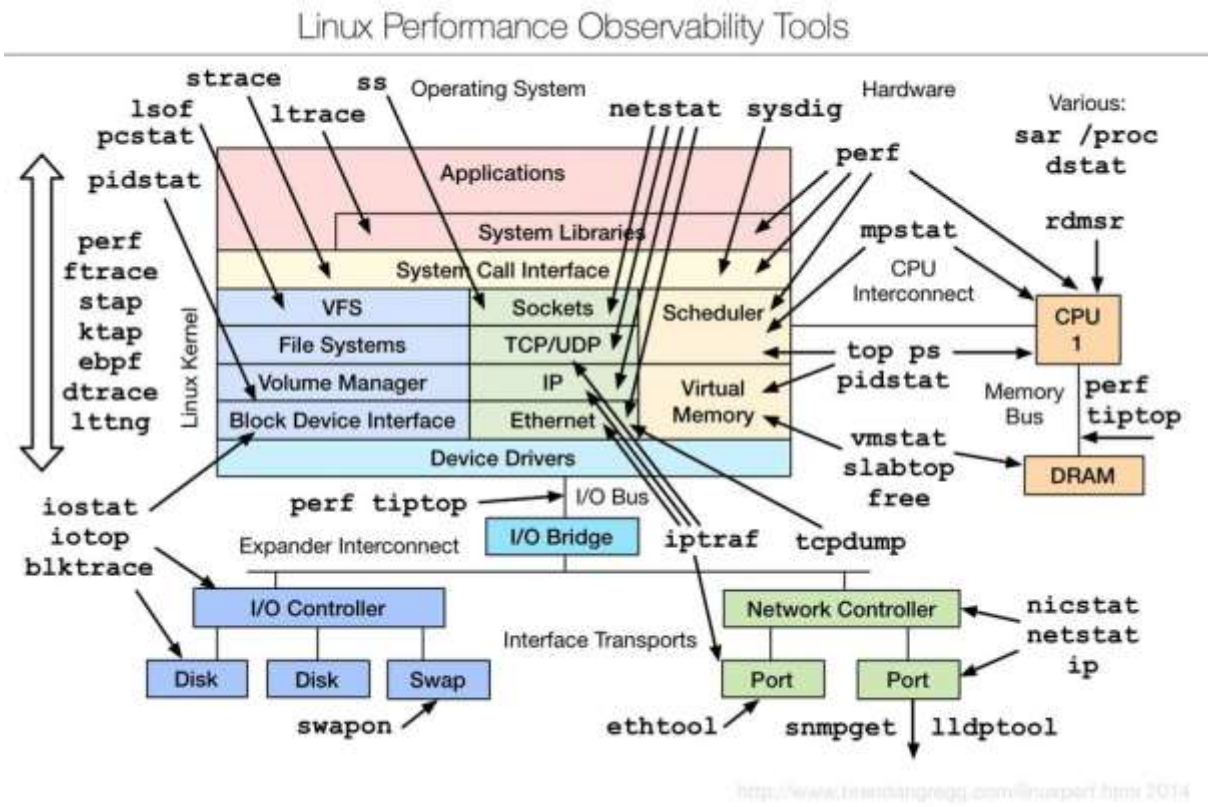

Figure 7: Linux Performance Observability Tools

With respect to the communication system, the following tools are relevant: netstat, nicstat, ip, ethtool and iptraf.

How data logging responds to requirements?

Among the requirements, the minimum compliance includes the following:

- The format of the records must be easily readable by a very wide range of tools (e.g. ASCII format, or similar).
- The format of storing must be a loss-less format. In case the data is too large, a tempting option is to use compression. The compression options should be loss-less.
- The data should be time-stamped.
- The time should be synchronized between multiple systems.

In addition, requirements of data recording exhibited in other contexts outside the context of AUTOPILOT may be relevant. Accidents of Connected Automated Vehicles are rare, and the post-crash analysis and results are rarely available to the publicly. However, the analysis of the collision of May 2016 with the Tesla car in the United States has been publicized widely [ref 2017-HWY16FH018-BMG-abstract.pdf]. The recommendations of the report that are relevant to the data recording in AUTOPILOT are the following:

- Define the data parameters needed to understand the automated vehicle control systems involved in a crash. The parameters must reflect the vehicle's control status and the frequency and duration of control actions to adequately characterize driver and vehicle performance before and during a crash.
- Define a standard format for reporting automated vehicle control systems data, and require manufacturers of vehicles equipped with automated vehicle control systems to report incidents, crashes, and vehicle miles operated with such systems enabled.

How data are delivered?

The data can be delivered in two distinct ways:

- In one way, the data is continuously uploaded from the car to the server in the fixed infrastructure. This allows keeping up with huge space demands.
- In another way, the data should be saved in a hierarchical set of memory locations in the car, including at least one element of solid state and removable media (SD card, SSD device, etc.).
- Specific devices ('black' boxes) may be used to store data, in order to be more resistant to fire, shock and other aggressions.

2.3 Pilot Site Italy

2.3.1 Short summary

The Italian IoT solution is characterized by devices (i.e. OEM in-vehicle components, inertial sensors, smartphone) that use a gateway (On Board Unit) to integrate the IoT information and to communicate to a Cloud server based on OneM2M platform.

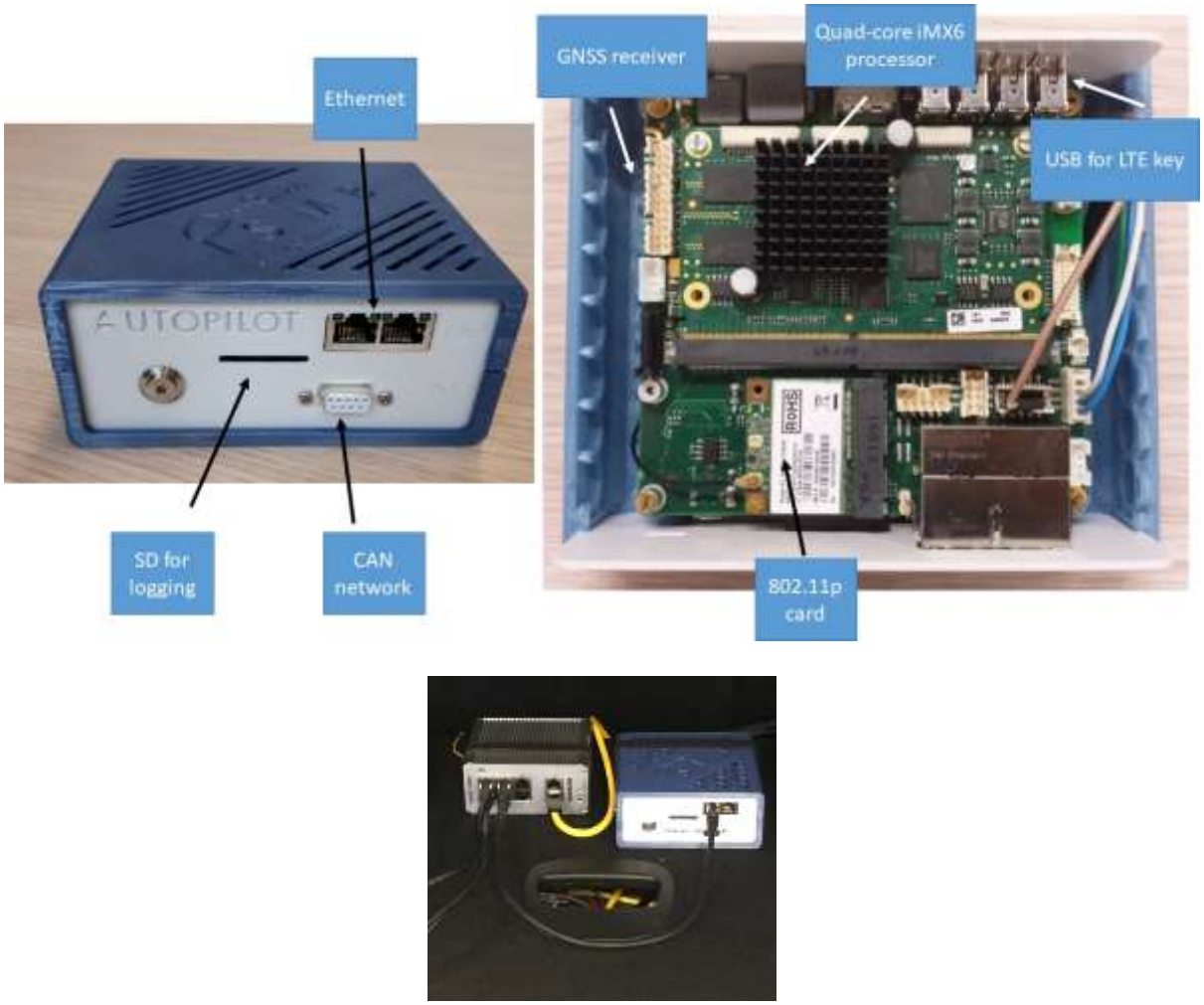

Figure 8: ISMB in-vehicle IoT platform in CRF prototype (PS Italy)

The software architecture scheme represented in Figure 5 shows the concept of IoT platform prototype in both Italian pilot site use cases:

- **Highway driving**: a cloud service merges the sensors measurements from different IoT devices as roadside sensors (e.g. pothole) or information about problems on the road (e.g. road works), in order to locate and characterize road hazards;
- **Urban driving**: focuses on the interaction with traffic lights and legacy traffic, on the robustness of the AD functions of the vehicle, safety when dealing with vulnerable road users (pedestrians and bicycle), and positioning.

In order to satisfy these use cases, the vehicle needs an on-board IoT platform to handle the various sources of data (IoT sensors like Inertial sensors, IMUs, etc.) with the various services (Local Dynamic Map-LDM, Pothole Detector, etc.).

The in-vehicle IoT platform provided by ISMB, offers communication interfaces to components that involve the following partners: TIM (provides OneM2M Cloud platform), CNIT (provides accelerometer sensor for the pothole algorithm), CRF and AVR (provide intra-vehicle sensors).

2.3.2 Software components

Figure 5 shows an overview of how the IoT platform of the Italian pilot site has to be set up in order to receive information from in-vehicle components and to connect with the cloud IoT Platform (OneM2M), also representing interfaces between the main components of this architecture.

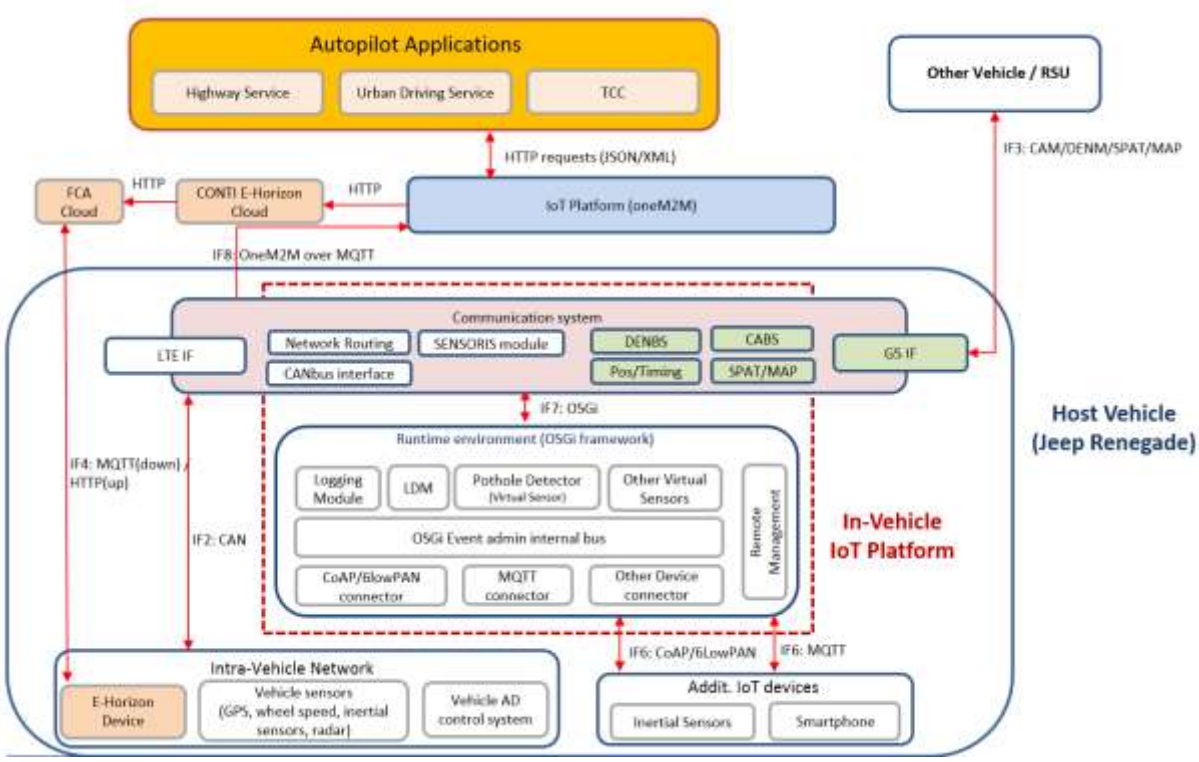

Figure 9: Vehicle scheme of software components, with In-Vehicle IoT platform

The IoT in-vehicle platform of the Italian pilot site is a modular software architecture including Application Container and Communication System, which are deployed on the On Board Unit (OBU).

The functional block Application Container is a lightweight approach to virtualization that developers can apply to rapidly develop, test, deploy, and update in-vehicle IoT platform services.

The Communication block performs the communication between devices and external components (such as OneM2M platform, RSU, other vehicle).

The “Runtime Environment” part of the OBU is composed by several software modules. As mentioned in chapter 4.2.3 of the deliverable D1.5 [1], they manage the following high-level functionalities:

- Remote Management
- Context Awareness
- Data Management
- IoT Connectors
- Data logging
    - On board logging
    - Logging towards the OneM2M platform (SENSORIS)

In the following, a brief description of each module is provided:

OSGi remote management tool: this is the software implementation of the Container Application, allows configuring the platform by adding/removing bundles, introducing the idea of remote monitoring and control of external application based on OSGi platform. Remote Services for OSGi can interact with the EventAdmin (OSGi Event Admin Internal Bus) service of the IoT in-vehicle platform. Through the Event Admin Internal Bus the connectors have the same communication interface to the bundles which they interfaced in the Application Container.

LDM: this is the database [1] where all the information about the surroundings of the ITS vehicle are saved. This data are used by the local applications to react and take decisions based on the in-vehicle sensors or other ITS station's status (such as surrounding vehicles or infrastructure).

LDM achieves integrated management of map information and vehicle information (functional requirement of Context Awareness): it contains information on real-world and conceptual objects that have an influence on the traffic flow.

CoAP/6LoWPAN connector: sensors belong to 6LoWPAN network may be connected to other IP networks through one or more edge routers (such as Ethernet, Wi-Fi or 3G/4G) that forward IP datagrams between different media.

Furthermore, CoAP is designed to use minimal resources (both on the device and on the network) and integrates with XML, JSON, or any data format chosen.

This module is used to integrate with 6LoWPAN protocol data coming from additional IoT devices (i.e. Inertial sensors), that are used by edge applications on the OBU.

MQTT connector: the MQTT connector supports both publishing (Send) and subscribing (Listen) to an MQTT broker (or server). This connector enables to integrate data to and from MQTT broker, which manages data from devices and sensors, with data from other sources accessible (connected with the functionality of IoT device adaptation and OEM communication systems).

This module is used to integrate with MQTT protocol data coming from additional IoT devices (i.e. smartphone), that are used by edge applications on the OBU.

Pothole detector: this bundle represents the implementation of the pothole detection algorithm. It is based on data fusion techniques in order to implement the concept of "virtual sensors". This module collects data from multiple sensors on the vehicle (IoT in-vehicle components or OEM in-vehicle components), processes the various data (in this case it could be established a threshold in order to be recognized or not the road holes) and sends the results of this elaboration to the cloud OneM2M platform or RSU or other vehicles (via communication system).

The **"Communication System"** part of the OBU, manages the following high-level capabilities:

- Send and receive of CAM and DENM packets
- Reception of SPAT/MAP messages
- Reception and decoding of CAN messages
- Managing of position and timing through the GPS hardware module
- Message routing
- Local Dynamic Map

In the following, a brief description of each module is provided:

CANBus Interface: this module reads data coming from the CAN Bus and decodes the messages previously defined in a DBC file defined by the vehicle vendor. This data are typically used to set the CAM fields that contain information about vehicle dynamics or other information (e.g. light state, etc.). The same module is used to decode important data coming from the in-vehicle sensors that are sent directly to the OneM2M platform or used by edge applications on the OBU.

Pos-Timing: this module reads the positioning data and timing information from the internal GPS receiver. This data are used to set the position on CAM and DENM messages. The timing part reads the time from the GPS and use the PPS hardware signal to align the NTP time server with a stratum 1 time source. In this way, a precise clock can be distributed to all the other software modules in the car, providing a precise synchronization.

CABS: this module takes data from the CANBus, position and time from Pos-Timing and creates a CAM message as described in the proper ETSI standard [2]. In the other hand it receives CAM messages coming from other vehicles and saves them on the LDM.

DENBS: as the previous module, this software creates and save DEN messages [3] interfacing with the proper modules to set the data and with the LDM to save the incoming messages.

SPAT/MAP: these messages are generated from a traffic light and SPAT/MAP module decodes them saving the relevant information in the LDM for further use. SPAT/MAP is just one potential technical feature of a cooperative ITS (C-ITS) roadside system. SPAT/MAP offers a potential channel for detailed information exchange between traffic systems and road users.

Network routing: this module manages the connectivity of all the in-vehicle modules that need network connectivity. Moreover, it manages the channels where CAM and DENM messages are sent. In the ISMB OBU they can be transmitted on the ETSI G5 radio channel and/or on the cellular way for debugging or other purposes.

SENSORIS module: this module takes the most important data and sends it, via cellular network, to the OneM2M platform. The data format is defined by the Autopilot Data Management Team and the information is sent using SENSORIS over MQTT.

Logging module: all the relevant data are internally logged by the OBU following the rules defined by D5.2

Continental provides E-Horizon. E-Horizon means ‘Electronic Horizon’ and relies on the fact that what a vehicle can detect is limited in range (e.g. sensors cannot detect a hazard 3km in front of itself). By crowd-sourcing data from different vehicles, a dynamic map can be enriched and uploaded to vehicles. In the result, vehicles benefit from what the others vehicles detect.

Practically, in Livorno PS, E-Horizon gathers information from OneM2M IoT platform such as road hazards detected by IoT devices, enriches its map, and notifies FCA (Fiat Chrysler Automobiles) cloud of these road hazards. FCA Cloud then notifies the vehicle that can adapt its driving speed based on this new information. In this case, E-Horizon is both the service provided by Continental Cloud and the on-board device. See below for details.

CONTI E-Horizon Cloud: collects data from TIM oneM2M cloud platform (centralized platform for IoT data), processes the data so that they can be consumed. From this data, a dynamic map update is generated and sent to FCA cloud using an HTTP request.

E-Horizon device: this module is in charge to receive the map update notification from FCA cloud through a MQTT interface, download the map update content from FCA cloud, process the map update and finally convert it to an ADASIS message to be sent over the on-board communication CAN bus, so that the information can be shared to others components (such as AD components).

2.3.3 Verification

The verification of ISMB in-vehicle IoT platform envisages the following tests:

- IT-ISMB-1: ITS G5: basic test in lab – communication between in-vehicle platform and RSU
- IT-ISMB-2: ITS G5: communication with in-vehicle CAN network
- IT-ISMB-3: ITS G5: communication between in-vehicle platform and RSUs
- IT-ISMB-4: ITS G5: communication between in-vehicle OBUs

- IT-ISMB-5: ITS G5: communication with traffic light ISMB RSU
- IT-ISMB-6: Cellular connectivity: communication with OneM2M platform
- IT-ISMB-7: 6LoWPAN: connectivity with vibration sensors
- IT-ISMB-8: Connectivity with smartphone/tablet for vibration data

Tests specifications are reported in the following tables.

Verification test ID	IT-ISMB-1 (M13)
Test Title	ITS G5: basic test in lab – communication between in-vehicle platform and RSU
Link to T2.5	Anticipates ITS-G5 type verification
High level objective of the test	Basic interoperability test between ISMB in-vehicle platform and CNIT RSU for CAM and DENM messages.
Involved software components	CAM/DENM sender/receiver, LDM.
How the test is realized	One ISMB board and one CNIT RSU are used for CAM/DENM exchange in a lab environment (not yet fully integrated in the vehicle).
Pass test criteria	Messages are correctly sent, received and decoded from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-2 (M15)
Test Title	ITS G5: communication with in-vehicle CAN network
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Communication between CRF autonomous driving platform and ISMB in-vehicle platform.
Involved software components	CANbus interface.
How the test is realized	The ISMB platform is integrated in the vehicle and exchange data with the CRF autonomous board.
Pass test criteria	Messages are correctly sent, received and decoded from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-3 (M17)
Test Title	ITS G5: communication between in-vehicle platform and RSUs
Link to T2.5	Anticipates ITS-G5 type verification
High level objective of the test	Communication test between ISMB in-vehicle platform and CNIT RSU for CAM and specific DENM messages.
Involved software components	CAM/DENM sender/receiver, LDM.
How the test is realized	One ISMB board and one CNIT RSU are used for CAM/DENM exchange in a real environment with specific DENM messages related to the use cases.
Pass test criteria	Messages are correctly sent, received and decoded from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-4 (M15)
Test Title	ITS G5: communication between in-vehicle OBUs
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Communication test between ISMB in-vehicle platforms (CAM).
Involved software components	CAM sender/receiver, LDM.
How the test is realized	One ISMB board and one CNIT RSU are used for CAM/DENM exchange in a real environment with specific DENM messages related to the use cases.
Pass test criteria	Messages are correctly sent, received and decoded from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-5 (M18)
Test Title	ITS G5: communication with traffic light ISMB RSU
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Communication test between ISMB in-vehicle platform and ISMB RSU on traffic light for SPAT/MAP and pedestrian detection.
Involved software components	SPAT/MAP DENM sender/receiver, LDM.
How the test is realized	One ISMB board and one ISMB RSU mounted on a traffic light, are used for SPAT/MAP/DENM exchange in a real environment.
Pass test criteria	Messages are correctly sent, received and decoded from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-6 (M16)
Test Title	Cellular connectivity: communication with OneM2M platform
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Communication test between ISMB in-vehicle platform and TIM OneM2M platform.
Involved software components	Network routing. Sensoris module.
How the test is realized	The in-vehicle platform sends SENSORIS messages to the OneM2M platform.
Pass test criteria	Messages are correctly sent and received by the OneM2M platform. Messages can be retrieved and correctly decoded by the OneM2M platform.
Results [yes / no]	Yes

Verification test ID	IT-ISMB-7 (M18)
Test Title	6LoWPAN: connectivity with vibration sensors
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Communication test between ISMB in-vehicle platform and the vibration sensor provided by CNIT.
Involved software components	CoAP/6LoWPAN connector.
How the test is realized	The in-vehicle platform retrieves messages from the vibration sensors via 6LoWPAN.
Pass test criteria	Messages are correctly received and decoded by the in-vehicle platform.
Results [yes / no]	Expected by middle of July 2018.

Verification test ID	IT-ISMB-8 (M18)
Test Title	Connectivity with smartphone/tablet for vibration data
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Communication test between ISMB in-vehicle platform and a smartphone/tablet on the car, to retrieve vibration data.
Involved software components	Other devices connector.
How the test is realized	The in-vehicle platform retrieves messages from a smartphone/tablet.
Pass test criteria	Messages are correctly received and decoded by the in-vehicle platform.
Results [yes / no]	Yes

2.3.4 Security Considerations

The implementation of the on board systems described in this chapter should to take into account the security requirements Described in D1.9 [3]. The current implementation of the core AD features and IoT connections, as of the current document emission date, are not yet fully compliant to the security requirements. Nonetheless care has been given to setup a development process that is taking into account the need to incorporate the requirements at a later stage. Tracking security requirements is an ongoing activity that will produce the full picture of the proposed solution compliance, together with security KPIs measurements. It must be clear that Autopilot is not developing new cybersecurity solutions, but will just instead use best practices already well established by todays secure IT, IoT, and embedded projects.

The next development stages will consider the implementation of the security requirements based on budget constraints and risk. D1.9 provides a reference risk analysis that has been specifically performed in order to prioritize security related risks. The components of the architecture described in this section of the document are an enabling factor for reducing risk. In particular we have the architectural elements to segregate communications and on board networks (CAN, WiFi, G5, etc.). One of the most crucial aspects of the current implementation is the lack of message authentication and confidentiality. For instance the current G5 implementation has ETSI security features disabled. Budget and risk will drive the selection of features to implement at a later stage.

2.3.5 Data logging and management

The Italian pilot site in-vehicle IoT platform will use the Google Protocol buffers (PROTOBUF) to perform the data logging.

Protocol buffers are a platform-and-language neutral mechanism for data serialization. They have been chosen basically for their performances figure. Compared to XML, PROTOBUF logs are 3 to 10 times smaller and 20 to 100 times faster, this is to ensure the possibility to log all data at the required frequencies (see D4.1 [4]), on a platform based on an ARM processor.

With PROTOBUF, it is possible to define a data schema (describing how data are structured) and automatically generate serializing/deserialzing code starting from it. Supported languages are Java, C++, Python, Java Lite, Ruby, JavaScript, Objective-C, and C#.

Furthermore, PROTOBUF permit to serialize data structure both in binary and JSON. In this way, it is possible to set the desired trade-off between compactness and readability of logged data depending on the situation (e.g.: logging in human-readable format like JSON during testing phase vs using a binary format in normal operations).

Finally, they permit to define a common data format between different partners only sharing the

data schema. This mitigates the problem of incompatibility between different logs files containing the same kind of information.

Considering the logging requirements, the in-vehicle platform can log all the data related to CAM and DENM messages and more in general to others V2x packets (e.g. SPAT/MAP). Moreover, all the information available on the CAN BUS and coming from the in-vehicle sensors (i.e. pothole detector) can be saved. In addition to GNSS PVT (Position, Velocity and Time), if required, the platform can log most of the receiver's raw sentences. Finally, it can log all the data transmitted and received by the cellular connection.

The logged data will be automatically sent to an FTP server (common to all actors of the Italian PS) at the end of each test session (e.g. at the end of each day of tests). Before the upload, the data will be first translated from the PROTOBUF binary format to a JSON human readable fashion.

2.4 Pilot Site The Netherlands

2.4.1 TNO prototype

2.4.1.1 Short summary

The TNO prototype consists of hardware devices (i.e., OEM in-vehicle components, sensors, and AD unit) that are integrated to a Unix machine that processes IoT and sensor data. To establish connectivity with IoT services (oneM2M platform) and with other vehicles and roadside units (ITSG5), two separate communication system units are deployed in the vehicle. Figure 10 shows the software architecture scheme defined for the following use cases:

- **Platooning**: focuses on the integration of IoT platoon service functions into the vehicle to provide platooning management and platooning formation signalling.
- **Automated Valet Parking (AVP)**: connects the vehicle to the IoT AVP service which provides parking lot availability and allocation as well as information about other road objects in the vicinity.

The in-vehicle IoT platform deployed in the TNO prototype includes components from the following partners: NXP (communication system unit) and TomTom (localization service).

2.4.1.2 Software components

Figure 10 shows the software components of the in-vehicle IoT platform of the TNO prototype vehicle and how they connect to the IoT cloud platform.

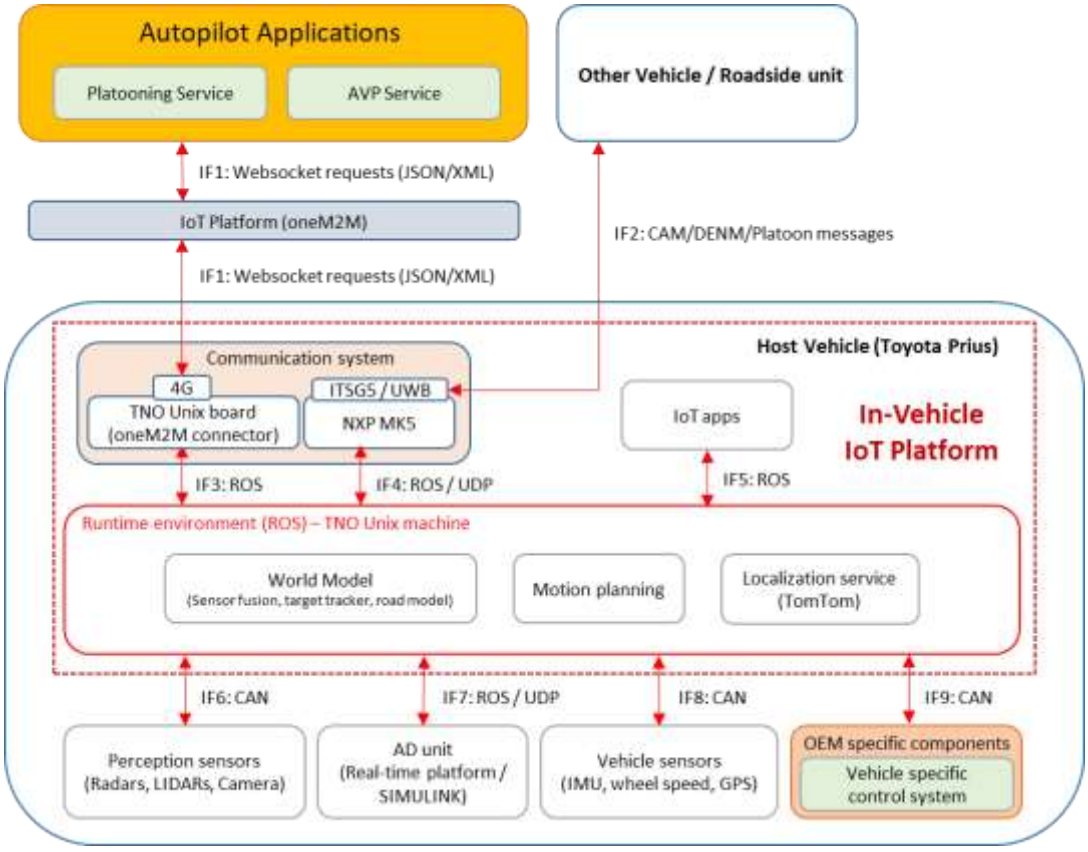

Figure 10: IoT software components architecture diagram of the TNO prototype vehicle

As shown in the previous figure, the main components of the in-vehicle IoT platform are:

- **Communication system** comprises two communication system units:
    - TNO Unix board: establishes cellular (4G LTE) connectivity to the IoT cloud via Websocket requests (JSON/XML) and interfaces with ROS components running in the TNO Unix machine.
    - NXP MK5: enables hybrid communication, i.e. combining different communication technologies (802.11 ITS-G5 and UWB), which can improve the performance and robustness of the communication. The ITSG5 radio is used to broadcast/receive both standard (CAM, DENM, SPAT, MAP) and non-standard (platooning management) messages whereas the Ultra-Wide Band (UWB) radio is used as redundant channel for non-standard messages as well as for measuring the distance to the preceding vehicle.
- **Run-time environment (ROS)** runs on the TNO Unix machine that aggregates and processes sensor and IoT data to be used by IoT apps and the AD unit:
    - World model: performs functions such as sensor fusion, target tracking and road model computation. It essentially aggregates data coming from multiple sensors, V2V communication and the IoT platform to build a road object level description of the world to the AD unit.
    - Motion planning: software component that calculates the path that the vehicle will take in the next tens of meters based on map (called occupancy map) and dynamic obstacles data. This component interfaces with other components in the run-time environment (ROS) such as the world model to gather required input such as dynamic tracked objects. The component gets the current position and the destination from the World model and returns the path that the vehicle shall follow given the environment constraints (obstacles).
    - Localization service: TomTom is a partner in the platooning use case for providing an HD map service. Using this service the most recent HD map (containing e.g. lane markings, lane center-lines, road boundaries) can be streamed to the vehicles. TNO also considers the incorporation of this HD map information into the localization algorithms of the follower vehicle in the platoon to provide and enhance lane-level localization.
- **IoT apps** generate and/or consume vehicle status and application-specific data (platooning, AVP) that are exchanged with the IoT platform (oneM2M).
- **Intra-vehicle network** comprises all other components outside the in-vehicle IoT platform:
    - Perception sensors: Radars, LIDARs and camera to be used for object and environmental perception.
    - AD unit: real-time platform running control-related algorithms.
    - Vehicle sensors: other internal sensors such as IMU, wheel speed, and GPS.
    - OEM specific components: interface with actuators in the vehicle.

2.4.1.3 Verification

The verification of the TNO in-vehicle IoT platform envisages the following tests:

- NL-TNO-1: ITS G5/UWB: basic test in lab
- NL-TNO-2: ITS G5/UWB: integration test in-vehicle
- NL-TNO-3: Cellular IoT connectivity - Platooning
- NL-TNO-4: V2V IoT connectivity – Platooning
- NL-TNO-5: V2I IoT connectivity to Traffic Lights (ITS-G5)
- NL-TNO-6: Data Model – Platooning
- NL-TNO-7: Motion planner integration
- NL-TNO-8: Cellular IoT connectivity – AVP
- NL-TNO-9: Data Model – AVP and Platooning

Tests specifications are reported in the following tables.

Verification test ID	NL-TNO-1 (M14)
Test Title	ITS G5/UWB: basic test in lab
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Basic test with NXP concerning their hybrid communication unit (ITSG5 + Ultra-wide band communication channels).
Involved software components	Communication system.
How the test is realized	Two NXP communication units are used for basic data exchange in a lab environment (not yet fully integrated in the vehicle).
Pass test criteria	Messages are correctly sent and received from one communication unit to another.
Results [yes / no]	Yes

Verification test ID	NL-TNO-2 (M18)
Test Title	ITS G5/UWB: integration test in vehicle
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Integration test with NXP concerning their hybrid communication unit (ITS-G5 + Ultra-wide band communication channels).
Involved software components	OEM specific components, world model, communication system.
How the test is realized	Messages are defined and generated for the platooning use case and travel from OEM components up to the communication system which broadcasts to other vehicles with the ITS-G5 channel. The ultra-wide band (UWB) channel is used for exchanging signals and providing ranging information.
Pass test criteria	Messages are correctly sent and received from one vehicle to another. Ranging measurement is correctly provided to the world model component.
Results [yes / no]	Yes

Verification test ID	NL-TNO-3 (M14)
Test Title	Cellular IoT connectivity - Platooning
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	End-to-end information exchange test between vehicle and oneM2M IoT platform for the platooning use case. Connectivity based on commercial cellular network.
Involved software components	OEM specific components, world model, IoT bridge, vehicle IoT apps, communication system, IoT platform (oneM2M)
How the test is realized	Messages are defined and generated for the platooning use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes

Verification test ID	NL-TNO-4 (M18)
Test Title	V2V IoT connectivity - Platooning
Link to T2.5	Anticipates ITS-G5 and Vehicle_safety_platooning type verification
High level objective of the test	Vehicle-to-vehicle platooning data exchange based on ITS-G5.
Involved software components	OEM specific components, world model, communication system.
How the test is realized	Messages are defined and generated for the platooning use case and travel from OEM components up to the communication system (temporary TNO unit) which broadcasts to other vehicles with the ITS-G5 channel.
Pass test criteria	Messages are correctly sent and received from one vehicle to another.
Results [yes / no]	Yes

Verification test ID	NL-TNO-5 (M18)
Test Title	V2I IoT connectivity to Traffic Lights
Link to T2.5	Anticipates ITS-G5 type verification.
High level objective of the test	Integration test with traffic light units via IoT.
Involved software components	OEM specific components, world model, communication system.
How the test is realized	Messages are generated from traffic light units and sent to IoT platform (oneM2M) which re-publishes the data to interested vehicles.
Pass test criteria	Messages are correctly sent from traffic light units and received correctly by the vehicle.
Results [yes / no]	Yes

Verification test ID	NL-TNO-6 (M14)
Test Title	Data Model - Platooning
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Verification of preliminary platooning data model (data specification of IoT messages to be exchanged).
Involved software components	IoT bridge, vehicle IoT apps, communication system, IoT platform (oneM2M).
How the test is realized	Messages are defined and generated for the platooning use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Data exchanged (messages) are correctly set as specified in the data model and verified via logging outputs. Time synchronization of the logging in different components is confirmed as required for post-analysis.
Results [yes / no]	Yes

Verification test ID	NL-TNO-7 (M18)
Test Title	Motion planner integration
Link to T2.5	Anticipates IoT_platform and Vehicle_safety_platooning type verification.
High level objective of the test	Integration test of path/motion planning algorithm.
Involved software components	World model, vehicle AD planning and control application.
How the test is realized	Vehicle path planning algorithm from receives and processes data from the world model to generate path trajectories to the vehicle control system.
Pass test criteria	Path trajectories are generated based on world model data.
Results [yes / no]	Yes

Verification test ID	NL-TNO-8 (M18)
Test Title	Cellular IoT connectivity - AVP
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	End-to-end information exchange test between vehicle and oneM2M IoT platform for the AVP use case. Connectivity based on commercial cellular network.
Involved software components	OEM specific components, world model, IoT bridge, vehicle IoT apps, communication system, IoT platform (oneM2M).
How the test is realized	Messages are defined and generated for the AVP use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes

Verification test ID	NL-TNO-9 (M18)
Test Title	Data Model – AVP and Platooning
High level objective of the test	Verification test of final data models for both platooning and AVP use cases.
Involved software components	IoT bridge, vehicle IoT apps, communication system, IoT platform (oneM2M).
How the test is realized	Messages are defined and generated for both platooning and AVP use cases and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Data exchanged (messages) are correctly set as specified in the data model.
Results [yes / no]	Yes

2.4.1.4 Data logging and management

All data is recorded with available ROS tooling for logging. This log data is stored in ROS *bag* files, which are the standard file format for logging in the ROS environment.

Every data message that goes to and from the IoT platform is extracted from the ROS bag files and based on it a new CSV file is generated to meet all data logging requirements as defined in WP4.1.

The generated CSV file is then uploaded to a FTP server directory that has been allocated for each use case in the project for evaluation.

2.4.2 NEVS prototype

2.4.2.1 Short summary

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 are AVP and Platooning. The control interfaces are accessible through a prototyping environment (i.e. dSpace MABX). This can provide access to interior sensor readings regarding vehicle dynamic states, e.g. rotational or translational acceleration, steering angle, brake pressure, wheel speed, etc.

2.4.2.2 Software components

Figure 11 shows the software components of Vehicle control system of the NEVS prototype vehicle and the integration to the In-Vehicle IOT platform and above layers.

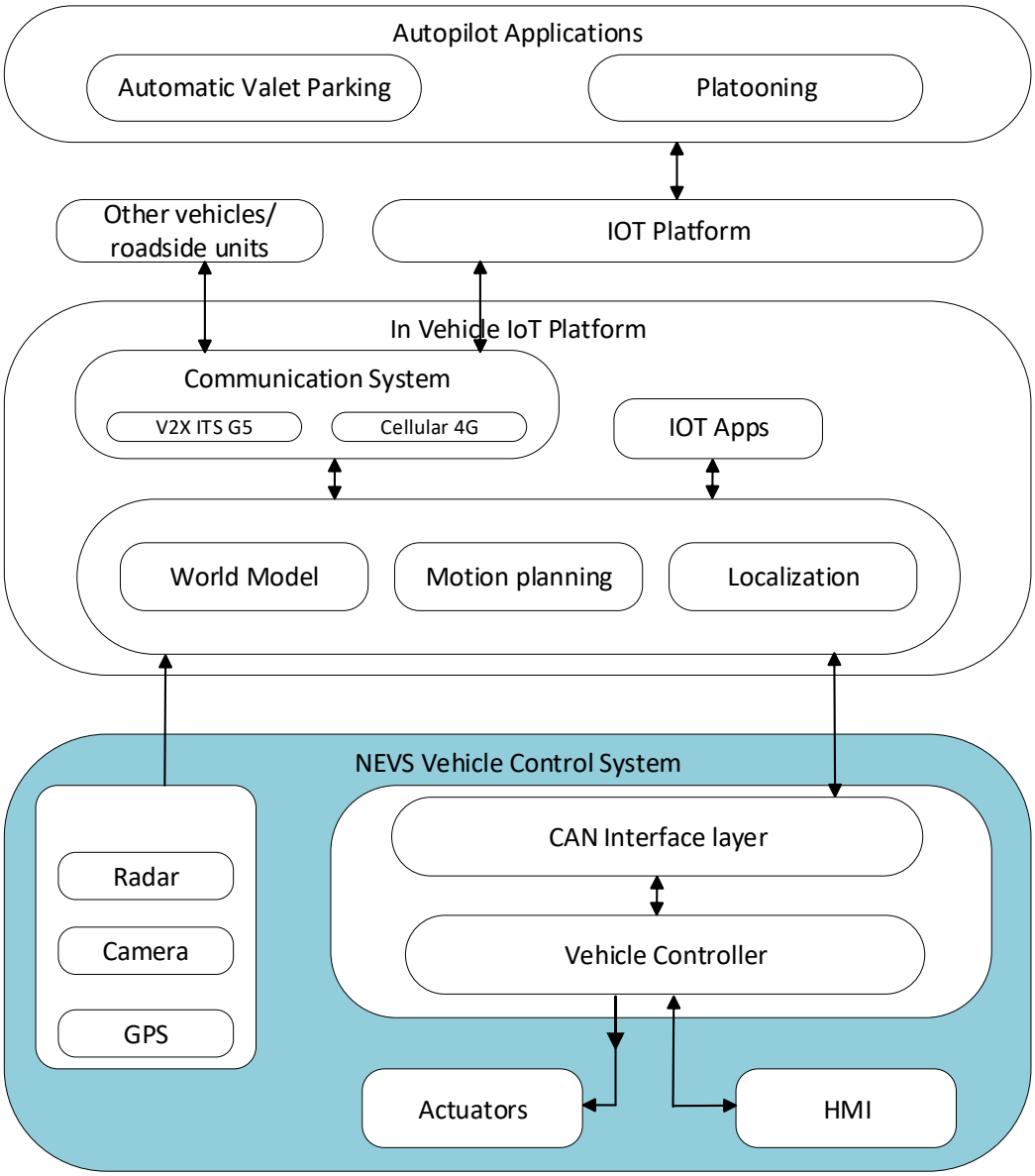

Figure 11: NEVS vehicle control system

CAN Interface layer: This layer is responsible for sending and receiving CAN messages to and from external systems. This layer interfaces with the in-vehicle IoT platform and the rest of the vehicle.

Vehicle controller: This module is responsible for executing the actuation in the vehicle and interfaces with the rest of the control units in the vehicle. This also interfaces with the HMI for communicating with the driver.

Sensors: This unit comprises a radar, camera and GPS components. This sends the object information to the in-vehicle IoT platform.

2.4.2.3 Verification

The verification of TNO in-vehicle IoT platform envisages the following tests:

- NL-NEVS-1: Integration test for CAN interfaces
- NL-NEVS-2: Integration test for ADAS unit

Tests specifications are reported in the following tables.

Verification test ID	NL-NEVS-1
Test Title	Integration test for CAN interfaces
High level objective of the test	Test of CAN communication between In-vehicle IoT and Vehicle control system.
Involved software components	CAN Interface layer.
How the test is realized	Using CAN tools.
Pass test criteria	Messages are correctly sent and received .
Results [yes / no]	Planned; basic IoT integration test is done by chapter 2.4.1.3.

Verification test ID	NL-NEVS-2
Test Title	Integration test for ADAS unit
High level objective of the test	Test of communication between In-vehicle IoT and ADAS unit.
Involved software components	ADAS unit.
How the test is realized	Using CAN tools.
Pass test criteria	The objects with relevant information and position data are received.
Results [yes / no]	Planned; basic IoT integration test is done by chapter 2.4.1.3

2.4.2.4 Data logging and management

The Vehicle controller shown in Figure 11 is connected to the vehicle network and has access to the vehicle information. The necessary vehicle information is packaged in the CAN interface layer and sent over CAN to in-vehicle IoT platform.

2.4.3 TUEIN prototype

The TUEIN prototype vehicle and in-vehicle IoT platform, whose software components are shown in Figure 16, provide the needed support for the TU/e use case **Urban driving / Rebalancing**, to pilot a **driverless car rebalancing service** on the Eindhoven University campus. The University Campus has a 2-km road network and a 30 km/h speed limit. On the campus, there are neither cross walks nor traffic lights.

The main goal of the rebalancing use case is to demonstrate a set of vehicles driving autonomously within the constraints of TU/e Campus (Urban environment) using both environmental sensor data as well as data available through IoT platform to improve the world model & Local Dynamic Maps embedded in the vehicle.

The technical setup is of the in-vehicle IoT platform is based on the connectivity requirements with the different separate IoT platform in the Urban Driving / Rebalancing use case represented in Figure 12.

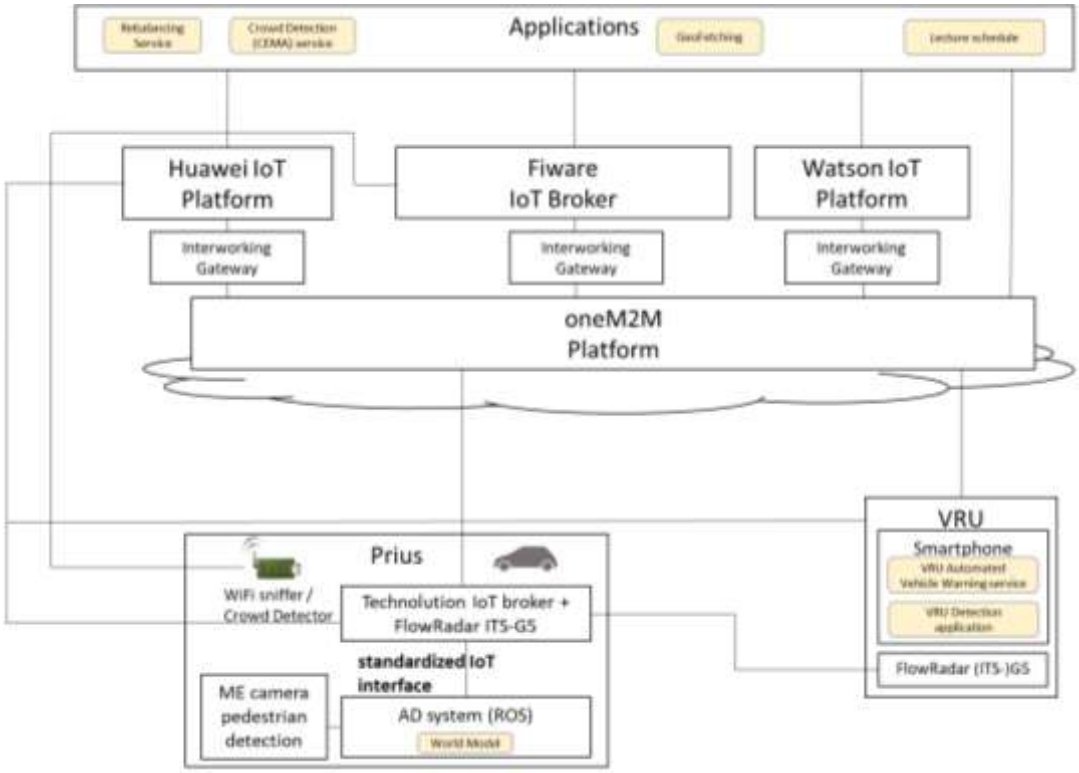

Figure 12: Overall IoT connectivity architecture to which the TUe vehicle needs to connect

2.4.3.1 Short summary

The TU/Eindhoven Campus Test Site provides 1 vehicle (Toyota Prius) for AUTOPILOT: Technolution contribute with 3 ITS-G5 devices (1 in-vehicle / 2 on people) and has integrated the ITS-G5 device into the in-vehicle IoT platform, such that using G5 Vehicles, Vulnerable Road Users and Road Side Units can contribute to the Rebalancing use case.

Contents of CAM messages from the vehicle are provided through OneM2M via MQTT using cellular 3/4G.

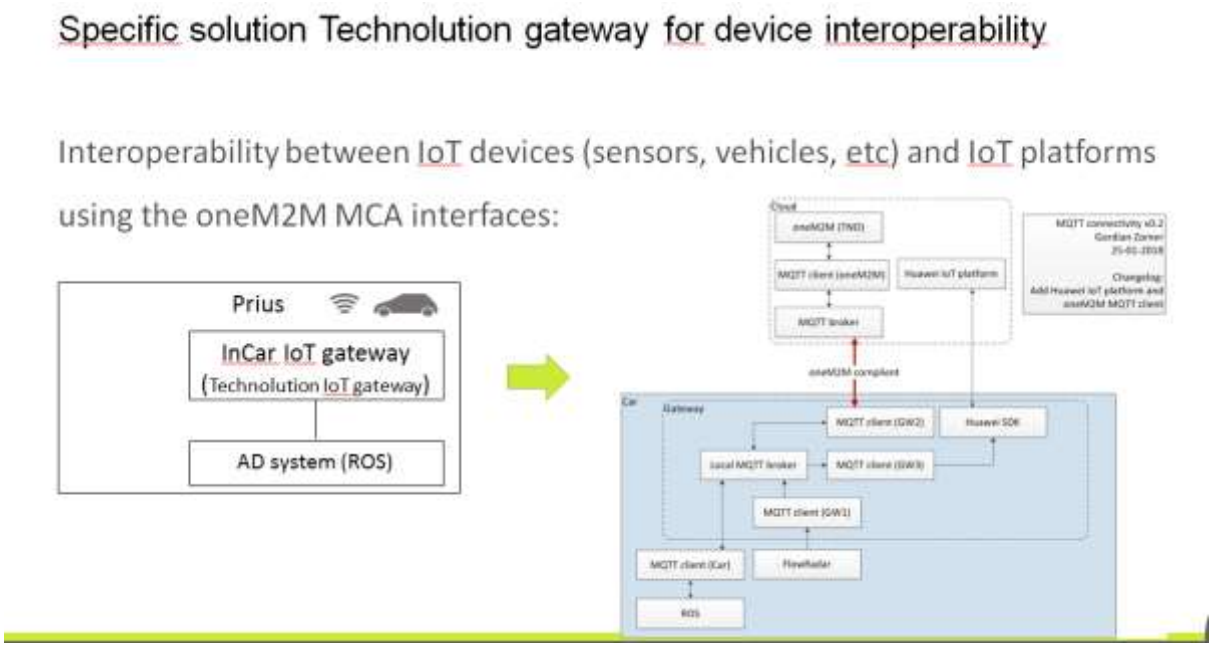

Figure 13: Solution IoT gateway with In-Vehicle IoT platform

Figure 14: Technolution IoT gateway (4G) with Technolution FlowRadar (ITS-G5) (both labelled with AUTOPILOT)

Concerning the VRU IoT solution, the G5 broadcast of CAM messages (specifically identified as VRU) is used to localize those VRU's using position, speed and heading. Every vehicle on-board IoT gateway (Host vehicle communication system) will receive these VRU G5 CAM messages and broadcasts those VRU G5 CAM messages, using cellular communication (3/4G), towards the Brainport OneM2M IoT platform.

Concerning the in-vehicle IoT Platform, the prototype uses an on-board unit to implement the in-vehicle IoT Platform, which will communicate with the different sensors (IoT devices) that reside inside the vehicle and with the external IoT platforms, including the Brainport pilot site OneM2M platform, FIWARE, HUAWEI OceanConnect & IBM Watson, as depicted in Figure 12.

Detailed description of Technolution IoT gateway:

- **Communication interfaces**: These interfaces are provided by the Technolution OBU (on-board unit) where the in-vehicle IoT platform is implemented.

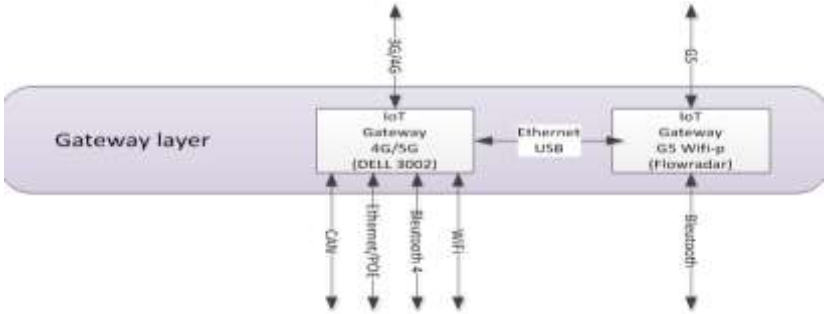

Figure 15: Communication interfaces of Gateway layer

- **In car internal connectivity of the gateway layer**
  - UDP to connect to the TU/e in-vehicle Runtime environment
- **In car external connectivity of the gateway layer**
  - 3G/4G LTE to connect to the central IoT platform for receiving smartphone data
- **Gateway protocols**
  - G5 gateway protocols: sub-set of CAM messages according to the ETSI standards

    - 4G/5G gateway protocols: OneM2M standards + connectivity with HUAWEI Ocean Connect IoT platform

- **IoT Module**:
  - **Gateway additional functions**
    - GPS (gps time used for time synchronization of G5 messages)
    - Log files/local storage: for evaluation proposed
    - Security: non-functional requirement
  - **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 messages, adapted to VRU identification in this use case.
  - **Connection with OneM2M MQTT**
    - **MQTT broker:** the gateway will also be positioned as a MQTT broker for the vehicle information. The vehicle than has its own information broker on board to share information between devices and applications with publish-subscribe mechanism.
    - **MQTT connector**: the MQTT connector supports both publishing (Send) and subscribing (Listen) to an MQTT broker (or server). This connector enables to integrate data to and from MQTT broker, which manages data from devices and sensors, with data from other sources accessible (connected with the functionality of IoT device adaptation and OEM communication systems).

2.4.3.2 Software components

The figure below shows an overview of the software components of the in-vehicle IoT platform of the TU/e Brainport pilot site and how these are connected, as well as how they connect with the IoT cloud platform.

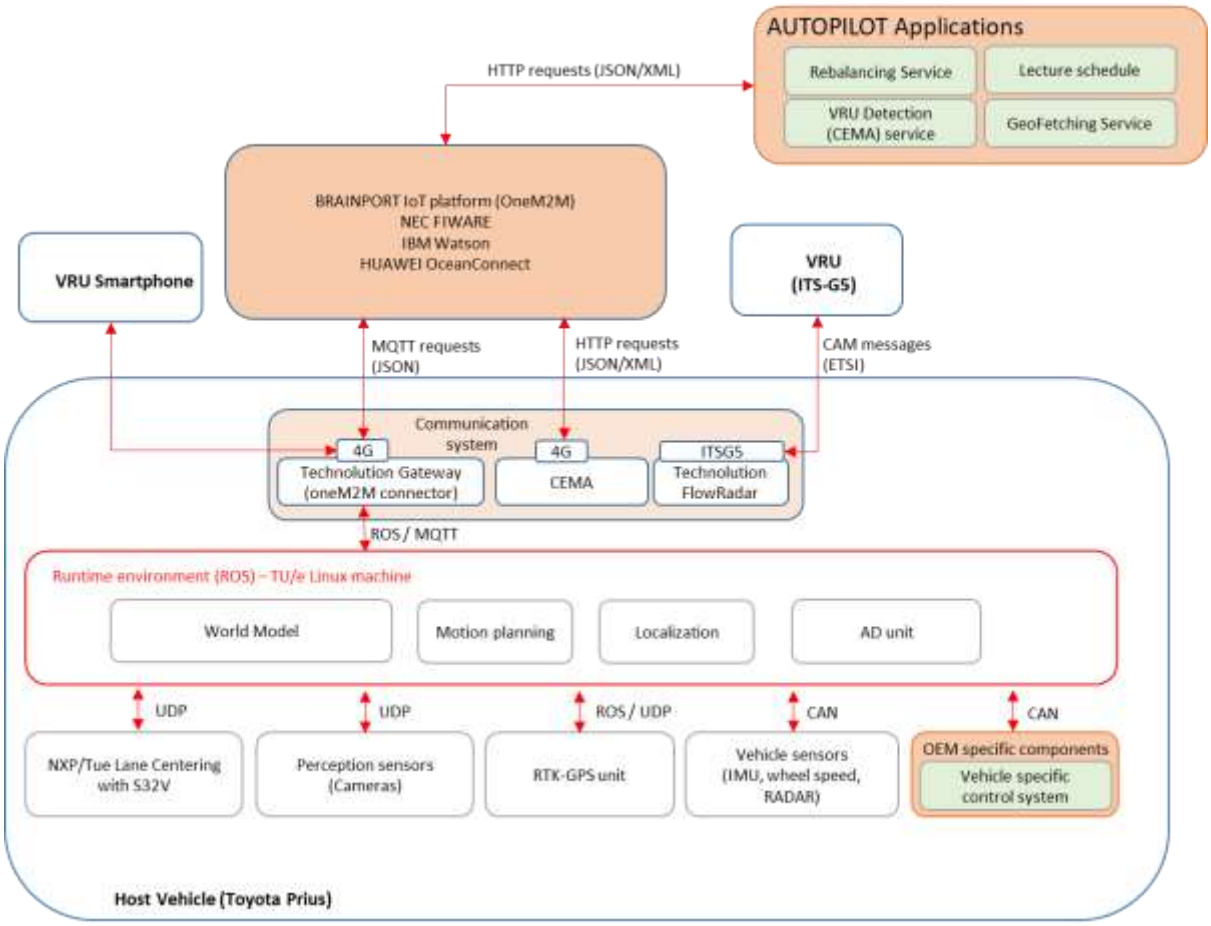

Figure 16: Vehicle scheme TUEIN prototype of software components, with In-Vehicle IoT platform

As shown in the previous figure, the main modules that compose the in-vehicle IoT platform are:

- **Communication system** comprises two communication system units:
  - The communication system provided by Technolution combines both ITS-G5 as well as 4G communication to the OneM2M platform and HUAWEI OceanConnect platform.
    - Technolution Gateway: establishes cellular (4G LTE) connectivity directly to the OneM2M & HUAWEI OceanConnect platforms and interfaces with ROS components running in the TU/e Unix machine.
    - Technolution FlowRadar: establishes ETSI ITS G5 to connect the vehicle to VRU (only 2 VRU will be equipped with ITS-G5 devices for this)
  - Separately, there is a direct connection from the NEC CEMA crowd estimation device in the vehicle towards the NEC FIWARE platform.
- **NEC Crowd Detector** is a separate hardware device that is in the vehicle and directly sends the number of detected devices within range of the device to the CEMA service from NEC. Data from the CEMA service is then received back into another vehicle using the schematic above.
- **Run-time environment (ROS)** runs on the TU/e Unix machine that aggregates and processes sensor and IoT data to be used by IoT apps and the AD unit:
    - World model: performs functions such as sensor fusion. It essentially aggregates data coming from multiple sensors, V2V communication and the IoT platform to give an overview of the obstacles around the vehicle to the AD unit.

- Motion planning (provided by NEC): software component that calculates the path that the vehicle will take in the next tens of meters based on obstacle data.
- Localisation: using RTK-GPS combined with vision based localization algorithms to localize the ego vehicle.
- NXP / TU/e Lane centering module: a low latency automated lane-centering system is developed and deployed on basis of open platform technology. This is implemented on a separate hardware platform. A fully programmable open vision pipeline allows OEMs to customize and optimize the vision processing. To that end a low power, small form-factor, automotive vision processor that embodies open standards (Open CL and Open CV) with a completely programmable vision pipeline is incorporated with various dedicated cores to achieve high performance, low power, and standards adherence. It also supports a high bandwidth MIPI (CSI-2) interface; a widely used camera interface in the IoT (mobile, wearables) industry. The sensor also considerably reduces bandwidth requirements, thus enabling real-time analytics in the IOT world. The algorithm employs principles of IoT based deep-learning in a more traditional model-based algorithm. The algorithm exploits the concept of hierarchical classification from deep learning. However, unlike deep learning, classification at each hierarchical level is engineered instead of being trained through images. This makes it more predictable as well as verifiable. The algorithm runs significantly faster than current lane-recognition systems, allowing significantly smoother automated driving. In addition, the performance of inference of neural networks on the vision processor is evaluated. Although in general it may be thought that large scale floating point GPU's are needed for inference, (as appears from the large deployment of NVIDA PX2 in automotive), results show that excellent results can be obtained on the relatively resource constraint IoT vision processor by deploying fixed point-neural networks. To that end, a newly developed lateral control algorithm to perform automated lateral control is deployed

- **Intra-vehicle network** comprises all other components outside the in-vehicle IoT platform:
  - Perception sensors: Radars, LIDAR and camera to be used for object and environmental perception
  - AD unit: real-time platform running control-related algorithms
  - Vehicle sensors: other internal sensors such as IMU, wheel speed, and GPS
  - OEM specific components: interface with actuators in the vehicle

2.4.3.3 Data logging and management

TU/e uses the same data management approach as TNO (see 2.4.1.4).

Next to this, also data from and to the OneM2M platform is recorded in the Technolution IoT gateway described above.

2.4.3.4 Verification

The verification of TUEIN in-vehicle IoT platform comprises the following tests:

- NL-TUE-1: ITS G5/IoT integration: basic test in lab
- NL-TUE-2: Cellular IoT connectivity OneM2M – Urban Driving / Rebalancing
- NL-TUE-3: Cellular IoT connectivity HUAWEI OceanConnect – Urban Driving / Rebalancing
- NL-TUE-4: Cellular IoT connectivity NEC FIWARE – Urban Driving / Rebalancing
- NL-TUE-5: Data Model – Urban Driving / Rebalancing

Verification test ID	NL-TUE-1 (M14)
Test Title	ITS G5/IoT integration: basic test in lab
High level objective of the test	Basic test with Technolution concerning their in-vehicle IoT platform with ITS-G5 functionality.
Involved software components	Communication system, Technolution FlowRadar (ITS-G5 unit).
How the test is realized	Two ITS G5 FlowRadar communication units are used for basic data bridging to the 3G/4G network in a lab environment (not yet fully integrated in the vehicle).
Pass test criteria	The ETSI CAM Messages are correctly bridged from the G5 network to the IOT 3G/4G network and sent and received from one 3G/4G communication unit to another 3G/4G communication unit.
Results [yes / no]	Yes (only first set of positioning messages)

Verification test ID	NL-TUE-2 (M18)
Test Title	Cellular IoT connectivity OneM2M – Urban Driving / Rebalancing
Link to T2.5	Anticipates verification of 2-way communication (over 4G connection) to OneM2M.
High level objective of the test	End-to-end information exchange test between vehicle and oneM2M IoT platform for the Urban Driving use case. Connectivity based on commercial cellular network.
Involved software components	OEM specific components, world model, Technolution IoT Gateway, vehicle IoT apps, communication system, IoT platform (oneM2M).
How the test is realized	Messages are defined and generated for the Urban Driving use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes (tested in Plugfest #2, May 2018)

Verification test ID	NL-TUE-3 (M18)
Test Title	Cellular IoT connectivity HUAWEI OceanConnect – Urban Driving / Rebalancing.
Link to T2.5	Anticipates verification of 2-way communication (over 4G connection) to HUAWEI OceanConnect.
High level objective of the test	End-to-end information exchange test between vehicle communication system (Technolution IoT gateway) and proprietary IoT platforms from HUAWEI. Connectivity based on commercial cellular network.
Involved software components	OEM specific components, Technolution IoT Gateway, vehicle IoT apps, communication system, IoT platform (HUAWEI OceanConnect).
How the test is realized	Messages are defined and generated for the Urban Driving use case and travel from OEM components up to the IoT platform (OceanConnect) in the cloud.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes

Verification test ID	NL-TUE-4 (M18)
Test Title	Cellular IoT connectivity NEC FIWARE – Urban Driving / Rebalancing
Link to T2.5	Anticipates verification of 1-way communication (over 4G connection) from NEC CEMA device to NEC FIWARE and 1-way communication towards vehicle platform of CEMA data.
High level objective of the test	End-to-end information exchange test between NEC CEMA crowd estimation device in the vehicle and proprietary NEC FIWARE IoT platform. Connectivity based on commercial cellular network.
Involved software components	NEC CEMA device in-vehicle, NEC FIWARE IoT platform, Technolution IoT Gateway, communication system.
How the test is realized	Messages are defined and generated for the Urban Driving use case and travel from NEC CEMA device up to the IoT platform NEC FIWARE platform.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Yes

Verification test ID	NL-TUE-5 (M18)
Test Title	Data Model – Urban Driving / Rebalancing
Link to T2.5	Anticipates verification of Urban Driving / VRU detection applications.
High level objective of the test	Verification of preliminary VRU detection data model (data specification of IoT messages to be exchanged).
Involved software components	Technolution IoT Gateway, vehicle IoT apps, communication system, IoT platform (oneM2M).
How the test is realized	Messages are defined and generated for the use case and travel from OEM components up to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Data exchanged (messages) are correctly set as specified in the data model and verified via logging outputs. Time synchronization of the logging in different components is confirmed as required for post-analysis.
Results [yes / no]	Yes

2.4.4 VALEO prototype

2.4.4.1 Short summary

The Valeo prototype consists of hardware devices (i.e., OEM in-vehicle components, Valeo sensors, and AD unit) that are integrated to a Windows machine that processes IoT and sensor data. To establish connectivity with IoT services (oneM2M platform) and with other vehicles and roadside units (ITS-G5), we use our own Valeo shark antenna. The following Figure 10 shows the software architecture scheme defined for the following use case:

- **Highway Pilot (HP)**: focuses on the integration of IoT in a vehicle and with road cameras to detect road defects and set AD instructions for the following vehicles.

The in-vehicle IoT platform deployed in the Valeo prototype includes components from the following partners: Vicomtech (Road anomaly detection) and TomTom (Live maps and AD instructions).

2.4.4.2 Software components

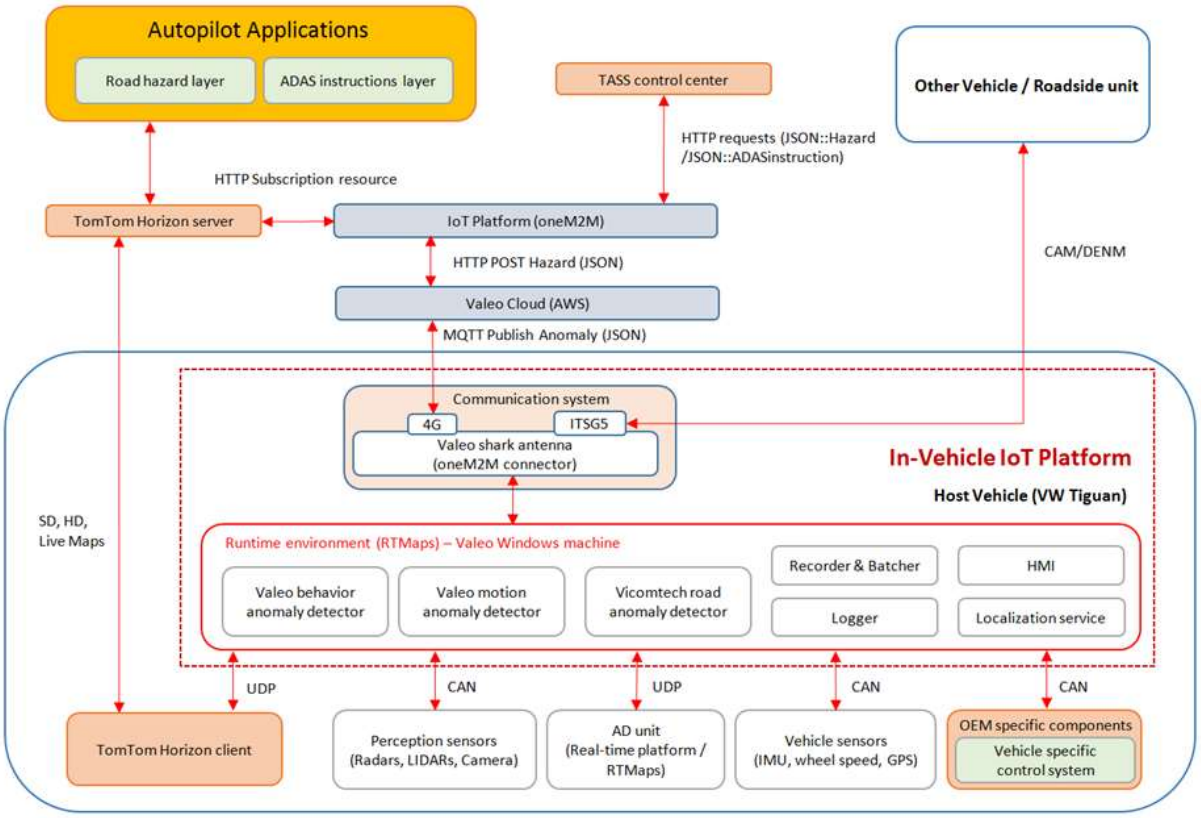

Figure 17: Software components of the in-vehicle IoT platform of the Valeo prototype vehicle

This figure shows the software components of the in-vehicle IoT platform of the Valeo prototype vehicle and how they connect to the IoT cloud platform.

- **Communication system** comprises two type of communication thanks to:
  - VALEO shark antenna: establishes cellular (4G LTE) connectivity to the Valeo cloud via MQTT Publish Anomaly (JSON) and interfaces with RTMaps components running in the Valeo Windows machine. Combining different communication technologies (802.11 ITS-G5), it can improve the performance and robustness of the communication. The ITSG5 radio is used to broadcast/receive both standards (CAM, DENM).
- **Run-time environment (RTMaps)** runs on the Valeo Windows machine that aggregates and processes sensor and IoT data to be used by IoT apps and the AD unit:
  - Valeo behavior anomaly detector: This is an algorithm based on the vehicle behaviour to detect abnormal driving.
  - Valeo motion anomaly detector: This is an algorithm based on the vehicle 3D motions to detect abnormal road.
  - VicomTech anomaly detector: This is an algorithm based on the front camera and the lidar to detect road defects.
  - Recorder & Batcher: This is a raw recorder of all data to replay the set of data and to do machine learning as post-processing in the Cloud.
  - Logger: This block is the watcher of the IoT platform which will notify every single event in the in-vehicle IoT platform.
  - HMI: This HMI will let testers understand the known and incoming road defects and associated AD instructions.
  - Localization service: GPS signals and HD map based localization service output might

be fused to achieve an enhanced lane-level vehicle positioning.

- **Intra-vehicle network** comprises all other components outside the in-vehicle IoT platform:
  - Perception sensors: Radars, LIDARs and camera to be used for object and environmental perception.
  - AD unit: real-time platform running control-related algorithms.
  - Vehicle sensors: other internal sensors such as IMU, wheel speed, and GPS.
  - OEM specific components: interface with actuators in the vehicle.
- **TomTom Horizon server/client** is an available solution provided by TomTom to securely implement AD instruction associated to road hazards met. The server contains a specific layer of road hazards and another for AD instructions.
- **TASS Control Center** is the only human in the loop. Its task is associated AD instructions to each road hazard.

2.4.4.3 Verification

The verification of VALEO in-vehicle IoT platform envisages the following tests:

- NL-VCDA-1: Detection of road anomaly
- NL-VCDA-2: Data logging
- NL-VCDA-3: Recording data
- NL-VCDA-4: Cloud connectivity
- NL-VCDA-5: ADASIN reception
- NL-VCDA-6: Road hazard representation (HMI)
- NL-VCDA-7: ADASIN application
- NL-VCDA-8: Data model

Tests specifications are reported in the following tables.

Verification test ID	NL-VCDA-1 (M14)
Test title	Detection of road anomaly
High level objective of the test	Through car sensors, these algorithms: driving behaviour, motion anomaly and road viewing anomaly have to generate a JSON::Anomaly.
Involved software components	Algorithms of detection and sensors.
How the test is realized	Go through a road hazard several times and each algorithm has to generate a JSON::Anomaly.
Pass test criteria	At least, a JSON::Anomaly is sent to the cloud every time.
Results [yes / no]	

Verification test ID	NL-VCDA-2 (M14)
Test title	Data logging
High level objective of the test	Each event in, through and out the vehicle has to be archived as described in D4.1.
Involved software components	Logger and communication system.
How the test is realized	Generate each event and confirm that the logger react to the change.
Pass test criteria	For each event in the vehicle, a logging has been done.
Results [yes / no]	

Verification test ID	NL-VCDA-3 (M14)
Test title	Recording data
High level objective of the test	Keep a trace of what has been done with the vehicle.
Involved software components	Recorder, sensors and logger.
How the test is realized	Before launching a record, verify that every sensor is sending data, every algorithm is ready to detect anomalies and the communication system is operational.
Pass test criteria	Be able to replay a record as we are doing it.
Results [yes / no]	

Verification test ID	NL-VCDA-4 (M14)
Test title	Cloud connectivity
High level objective of the test	Keep the Cloud connectivity on and received the ADASIN at the right time.
Involved software components	Communication system, AD unit and TomTom Horizon client.
How the test is realized	Send regularly messages from the vehicle to the Cloud and vice-versa for a long period (more than 20 minutes).
Pass test criteria	90% of received messages from both sides.
Results [yes / no]	

Verification test ID	NL-VCDA-5 (M14)
Test title	ADASIN reception
High level objective of the test	Define the relevance and the potential application of the ADAS instruction in a giving driving situation.
Involved software components	AD unit, communication system, HMI and sensors.
How the test is realized	Apply suitable ADASIN and unsuitable ADASIN and display the status on the HMI.
Pass test criteria	The AD unit is able to apply on its own the ADASIN.
Results [yes / no]	

Verification test ID	NL-VCDA-6 (M14)
Test title	Road hazard representation (HMI)
High level objective of the test	Show the enhancement of the IoT through the display of incoming road hazards and incoming ADASIN.
Involved software components	HMI, AD unit and communication system.
How the test is realized	Go next to an annotated road hazard and ADASIN and display them on the HMI of the vehicle.
Pass test criteria	Display them correctly and have enough time to read the information.
Results [yes / no]	

Verification test ID	NL-VCDA-7 (M14)
Test title	ADASIN application
High level objective of the test	When the ADASIN is suitable (see NL-VCDA-5), the vehicle respects the ADASIN until this end.
Involved software components	AD unit, communication system, HMI and sensors.
How the test is realized	Apply and display the time/distance from when it will be apply until when (countdown).
Pass test criteria	Apply the ADASIN before the road hazard and remove the ADASIN after the road hazard.
Results [yes / no]	

Verification test ID	NL-VCDA-8 (M14)
Test title	Data Models
High level objective of the test	Verification of all data model (Anomaly, Hazard and ADASIN).
Involved software components	Algorithms of detection, sensors, logger and communication.
How the test is realized	Messages are defined and generated for from the vehicle to the IoT platform (oneM2M) in the cloud and back to the vehicle.
Pass test criteria	Data exchanged are correctly set as specified in the data model and verified via logging outputs. Time synchronization of the logging in different components is confirmed as required for post-analysis.
Results [yes / no]	

2.4.4.4 Data logging and management

Each functional block from Figure 17 has his own data logging. The transfer of these logs from a component to other respects the connections on this diagram.

Each log file is in “csv” file type. They shall contain the header of each column. Unix epoch timestamps are long value with the number of milliseconds since 1-1-1970 UTC (Ex :1521453406869). The log_data field must be bracketed in quotes: " ".

The log data fields and their occurrences are listed the following tables:

Responsible	Vicomtech Valeo

CSV Log File Name	`log.tiguan_vicomtech_component.csv`
	`log.tiguan_data_manager.csv`
	`log.tiguan_imu_component.csv`
	`log.tiguan_secure_agent.csv`
	`log.tiguan_ad_command.csv`

Log Data Field	Occurrences
Log_timestamp	<unix epoch in ms>
log_stationid	TIGUAN
log_applicationid	VICOMTECHCOMPONENT DATAMANAGER IMUCOMPONENT
log_action	SEND NOTIFICATION ALIVE RECEIVE AUTHENTIFICATION CONNECTION PROCESSING
log_medium	INTERNAL 4G
log_type	DATA ERROR INFO NONE
log_data	JSON::Anomaly JSON::Hazard JSON::ADASIN Map::Hazard Map::ADASIN <error message> <message> NONE

All these logs are locally stored in the test vehicle. They have to be sent manually (TBC) to the TNO logging platform after each performed test.

2.4.5 IBM Ireland prototype

2.4.5.1 Short summary

IBM IE is leading the car-sharing/ ridesharing use case in Brainport. This use case will provide a cloud-based application to book and dispatch vehicles to accommodate ride demands. In addition, the service will be coupled with other use cases in the Brainport area, i.e., platooning, car rebalancing, AVP.

IBM IE does not have access to vehicles per se, but it will make use of partners' vehicles to integrate its solution. In particular, the vehicles will be equipped with GPS/radio receivers, so to communicate with the ridesharing application on the cloud.

We are developing a Watson IoT API call, which vehicles will be able to use to publish/subscribe to events and receive updates. For example, vehicles will be able to post if a road is blocked (jammed), as well as other communication protocols to communicate directly with the ridesharing/carsharing service.

The In-Vehicle IoT platform will be also coupled with the ones of other use cases, so to acquire events that may be useful for the IBM IE service.

2.4.5.2 Software components

A general picture is shown in Figure 18. One can see how the vehicles will be equipped with an on board unit that can send/receive information to the ridesharing application directly, and that can make use of a Watson IoT API call to publish and subscribe to events.

The on board unit consists of:

(i) A GPS receiver with navigation capabilities; this will serve to guide the vehicles to pick-up and delivery locations;

(ii) A Data log/ process component; this will serve to log the available data and process the received/ to be sent information. For example, the log will store the customers that the vehicle has serviced, and the vehicle status. If the fuel is running low, the data processing will inform the ridesharing/car-sharing application;

(iii) A radio communication module, so to receive and send data (either via the API directly, possibly via OneM2M, or via MQTT directly to the application).

As for Hardware, the on board unit will be a suitably modified tablet, or smart phone, capable of displaying routing information and directions to the user.

As for Software, the on board unit will have a navigation software for the GPS in (i), some simple data logging and processing for (ii), and simple APIs/ OneM2M connectors for (iii).

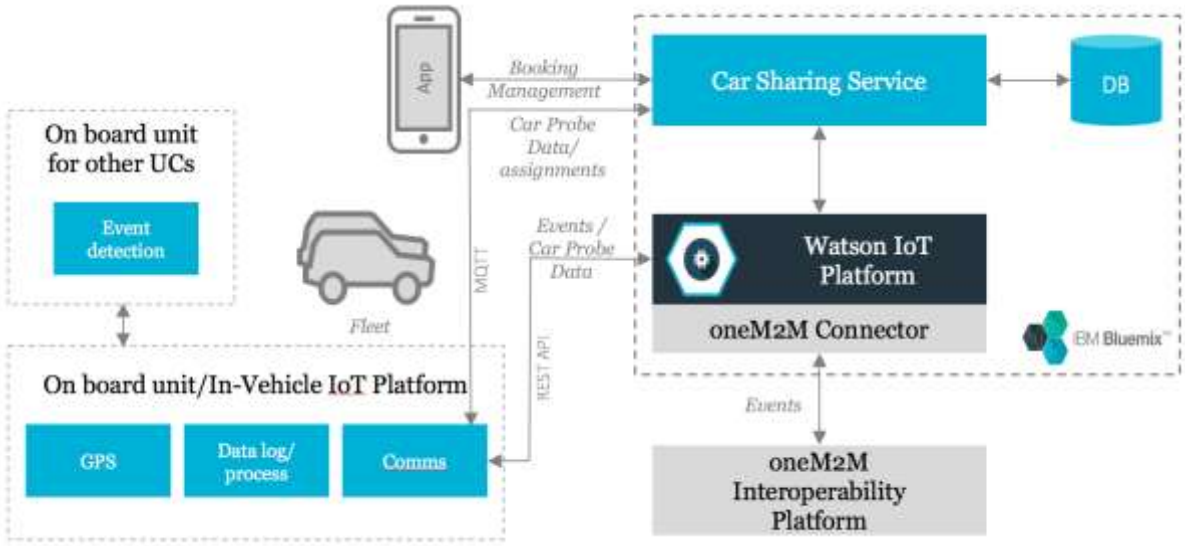

Figure 18: Vehicle scheme of software components, with In-Vehicle IoT platform

2.4.5.3 Verification

The verification of IBM IE in-vehicle IoT platform encompasses these tests, which are specified in the tables that follow.

- NL-IBMIE-1: On board unit: basic tests in the lab
- NL-IBMIE-2: On board unit: basic tests in the vehicle
- NL-IBMIE-3: Connectors: basic tests in the lab
- NL-IBMIE-4: End-to-end connectivity and data transmission
- NL-IBMIE-5: In-car data managing: data models, logging, displaying

Verification test ID	NL-IBMIE-1 (M17)
High level objective of the test	On board unit: basic tests in the lab
Link to T2.5	Anticipates IoT_platform type verification.
Involved software components	On board unit software.
How the test is realized	Messages are defined and sent to the on board unit that has to display the information and send back messages.
Pass test criteria	Messages are correctly sent and received from one end to another. Navigation is correctly displayed.
Results [yes / no]	Planned

Verification test ID	NL-IBMIE-2 (M17)
High level objective of the test	On board unit: basic tests in the vehicle
Link to T2.5	Anticipates IoT_platform type verification.
Involved software components	On board unit software.
How the test is realized	Messages are defined and sent to the on board unit that has to display the information and send back messages. The vehicle purposely does not follow the GPS directions: GPS navigator changes the directions in real-time.
Pass test criteria	Messages are correctly sent and received from one end to another. Navigation is correctly displayed.
Results [yes / no]	Planned

Verification test ID	NL-IBMIE-3 (M17)
High level objective of the test	Connectors: basic tests in the lab
Link to T2.5	Anticipates IoT_platform type verification.
Involved software components	IBM Watson API, IBM adapter, and MQTT.
How the test is realized	Messages are defined and sent back and forth from the on board unit to the cloud via IBM Watson API, IBM adapter, and MQTT.
Pass test criteria	Messages are correctly sent and received from one end to another.
Results [yes / no]	Planned

Verification test ID	NL-IBMIE-4 (M18)
High level objective of the test	End-to-end connectivity and data transmission
Link to T2.5	Anticipates IoT_platform type verification.
Involved software components	IBM Watson API, IBM adapter, and MQTT, on board software
How the test is realized	Messages are defined and generated for the vehicles and travel to Watson IoT and back. The vehicle is purposely driving not following the GPS directions: data logs are sent to the cloud and new directions are issued.
Pass test criteria	Messages are correctly sent and received from one end to another (first and last components).
Results [yes / no]	Planned

Verification test ID	NL-IBMIE-5 (M18)
Test Title	In-car data managing: data models, logging, displaying
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Verification of the in-car data management functionalities.
Involved software components	IBM Watson API, IBM adapter, and MQTT, on board software.
How the test is realized	Messages are defined and generated for the car-sharing use case and travel from on board unit up to the IoT platform in the cloud and back to the vehicle.
Pass test criteria	Data exchanged (messages) are correctly set as specified in the data model and verified via logging outputs. Time synchronization of the logging in different components is confirmed as required for post-analysis. Navigation is correctly displayed in the vehicle and changes are reported in real-time, if the vehicle does not follow directions.
Results [yes / no]	Planned

2.4.5.4 Data logging and management

The data that are needed for the car sharing service are sent to the service and are available for evaluation. Data will be logged at the service side and stored in different formats depending on the nature of the data (typically routes, ride-assignments, GPS positions, etc.). The collected data will be available for evaluation for work package 4.

2.4.6 DLR prototype

2.4.6.1 Short summary

The DLR prototype FASCarE is a fully electric Volkswagen e-Golf. The vehicle was modified to allow access to the OEM systems, particularly OEM sensors and actuators for longitudinal and lateral control. Furthermore, it was equipped with additional hardware. This includes radar and laser sensors, a differential GPS system, several industrial PCs for running the AD functionality, a 4G communication unit and a custom dashboard display. Figure 19 shows the software scheme for the following use case:

- **Automated Valet Parking (AVP):** connects the vehicle to the IoT AVP service which provides parking lot availability and allocation, routing as well as information about other road objects / obstacles in the vicinity.

2.4.6.2 Software components

The main components of the vehicle software architecture are:

- **Communication system**
  - Mobile communication unit (4G): provides Internet connectivity to the IoT platform
  - IoT Gateway: communicates with the IoT platform over MQTT
  - Logging: logs IoT communication
- **Run-time environment ROS** runs a Ubuntu Linux machine and is responsible for processing and fusing sensor and IoT data to be used by the AD functions:
  - World model: performs functions such as sensor fusion, target tracking and road model computation. It essentially aggregates data coming from multiple vehicle sensors and the IoT platform to build an environment model of the world.
  - Data Logging: logs data from sensors
- **Run-time environment Dominion** runs on a Ubuntu Linux machine and responsible for
  - AVP Function Management: controls and monitors the general flow of the AVP use case, e.g. dropoff – park – pickup and is responsible for activating the relevant function modules based on the data received through the IoT gateway
  - Tactical Planner: plans the vehicle's behaviour on a tactical level with a time horizon of several seconds and parametrizes trajectory planning accordingly, based on e.g. recommended speed and static obstacles
  - Trajectory planning: software component that computes a path for the vehicle with certain constraints such as obstacle avoidance
  - High Level Control: is responsible for keeping the vehicle on the path by sending actuator setpoints to the vehicle systems
  - Logging: logs data specific to functionality running on Dominion RTE
- **ROS-Dominion-Bridge**

- Is responsible for relaying data between both run time environments

- **Intra-vehicle network** comprises all other components outside the in-vehicle IoT platform:
  - Perception sensors: Radars, LIDARs and camera to be used for object and environmental perception
  - RTK GPS: GPS localization system with real-time kinematic positioning (RTK)
  - Vehicle specific components: real-time platform for low level actuator control and vehicle OEM sensors (odometry, RADAR, camera)
  - In-vehicle HMI: Custom dashboard display

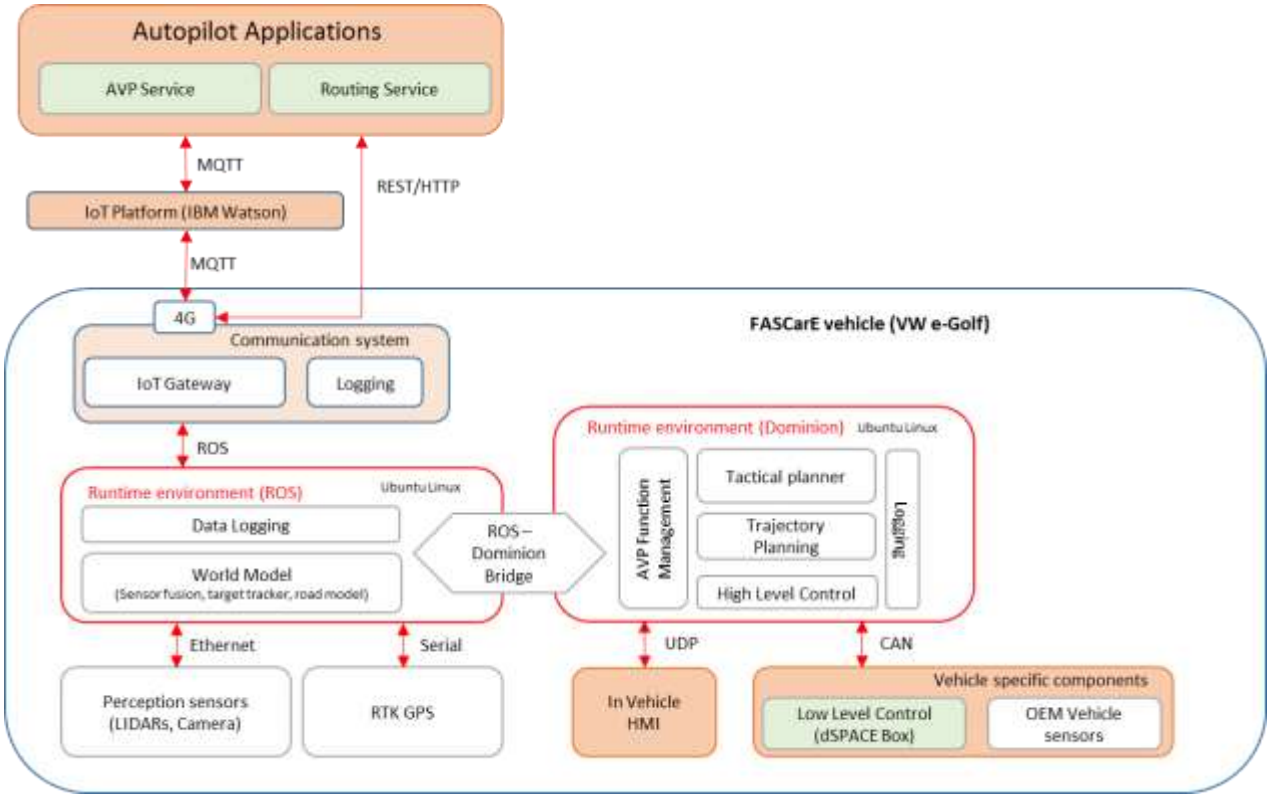

Figure 19: DLR vehicle scheme for software components

2.4.6.3 Verification

Verification of the vehicle platform is done using the following tests:

Verification test ID	NL-DLR-1 (M18)
Test Title	Integration ROS – Dominion Bridge
High level objective of the test	Test data transfer between ROS and Dominion RTEs.
Link to T2.5	None
Involved software components	ROS and Dominion RTE.
How the test is realized	ROS and Dominion applications are started as well as the ROS Dominion Bridge. The bridge is configured to relay specific data from ROS to Dominion and vice versa.
Pass test criteria	The test is successfully if data available in Dominion can be accessed from a ROS application and data published in ROS can be read from a Dominion application.
Results [yes / no]	Yes

Verification test ID	NL-DLR-2 (M18)
Test Title	IoT connectivity
High level objective of the test	Test data transfer between vehicle platform and IoT platform.
Link to T2.5	Being tested as part of T2.5.
Involved software components	Communication system, IoT platform, AVP IoT application.
How the test is realized	See test cases IoT_platform_1-7 from T2.5.
Pass test criteria	IoT data can be sent / received and parsed correctly.
Results [yes / no]	Yes

Verification test ID	NL-DLR-3 (M18)
Test Title	LIDAR ROS Integration
High level objective of the test	Receive LIDAR data over ROS.
Link to T2.5	None
Involved software components	LIDAR, ROS
How the test is realized	Enable LIDAR, read and process LIDAR data in a ROS application.
Pass test criteria	LIDAR data arrives reliably and processing output appears valid.
Results [yes / no]	Yes

Verification test ID	NL-DLR-4 (M18)
Test Title	GPS ROS Integration
High level objective of the test	Receive GPS data over ROS.
Link to T2.5	None
Involved software components	GPS RTK, ROS
How the test is realized	Enable GPS, read and process GPS data in a ROS application.
Pass test criteria	GPS data arrives reliably and appears valid.
Results [yes / no]	Yes

2.4.6.4 Data logging and management

There are three logging components involved

- **Logging in ROS**
  - Data exchanged in ROS is logged in ROS bag files, the standard format for logging in ROS
- **Logging in Dominion**
  - Use Case / evaluation relevant data exchanged in Dominion is logged in csv files
- **Communication Logging**
  - Every message sent / received from the IoT platform is logged in csv files according to WP4.1 requirements

2.5 Pilot Site Spain

2.5.1 Short summary

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

Concerning the IoT solution for the vehicles, all the prototypes use an on-board unit to implement the in-vehicle IoT platform, which will communicate with the different sensors that reside inside the vehicle and with the external IoT platform, including the pilot site IoT platform or the Central IoT Platform, being both oneM2M platforms.

This in-vehicle IoT solution, which software components are shown in Figure 2, provides the needed support for both Spanish use cases:

- **Automated Valet Parking**: AD functions in a parking environment where the user can interact with the Valet Parking functionalities of the vehicle and the vehicle can communicate with the parking control center through IoT.
- **Urban driving**: AD functions in urban environment while the IoT in-vehicle platform focuses on the interaction with traffic lights, vulnerable road users/obstacles and different hazards and events, such as traffic jams, roadworks or accidents.

2.5.2 Software components

In the figure below is shown an overview of the software components of the in-vehicle IoT platform of the Spanish pilot site and how these are connected, as well as how they connect with the IoT cloud platform.

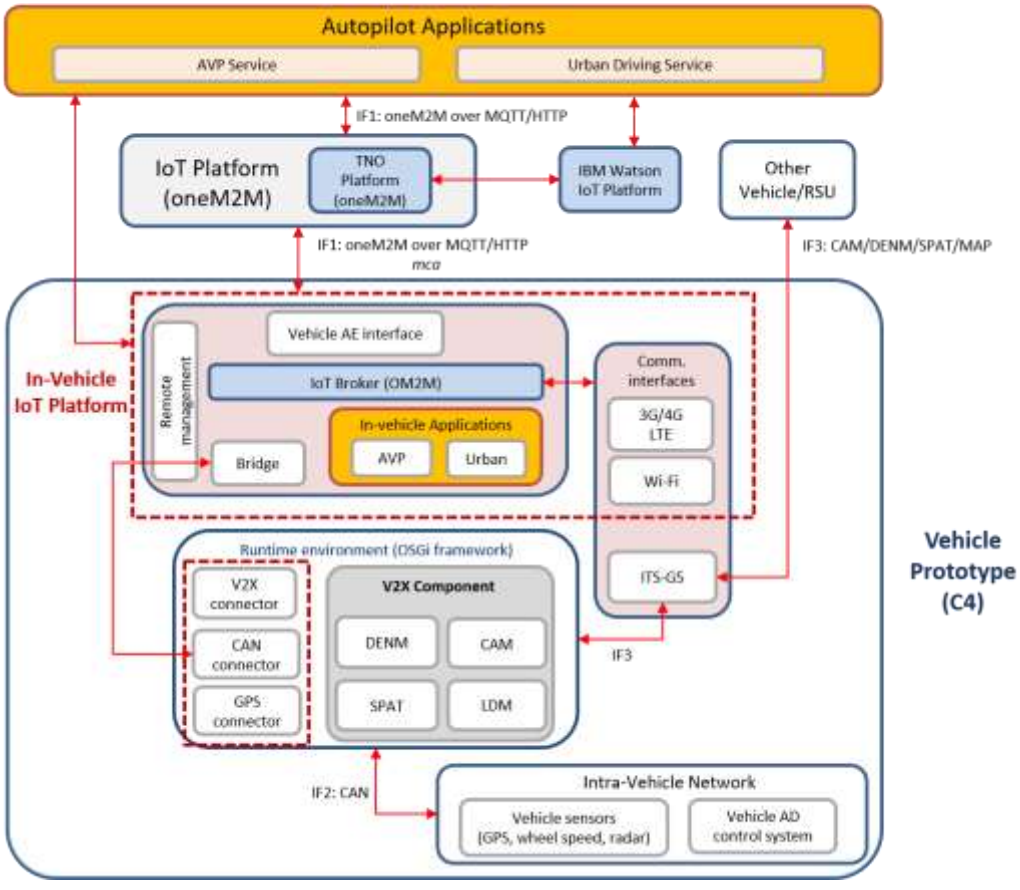

Figure 20: IoT software components architecture diagram of the Spanish pilot site

As shown in the previous figure, the main modules that compose the in-vehicle IoT platform are:

- **Communication interfaces**: These interfaces are provided by the OBU (on-board unit) where the in-vehicle IoT platform is implemented.
  - Cellular (3G/4G LTE): Cellular interface to connect to the cloud
  - Wi-Fi: Wireless interface to connect to the cloud
  - ITS-G5: Wireless interface used by the V2X component to connect with other vehicles or infrastructure
- **IoT Module**: OM2M implementation of the oneM2M standard, using an OSGi framework with its broker and the different applications implemented. This module is the one that translates the information that comes from the vehicle into oneM2M messages and translates any oneM2M message into understandable information for the vehicle.
  - IoT Broker: OM2M based ASN-CSE (oneM2M) that acts as an IoT Gateway. It provides the HTTP and MQTT connectivity to the IoT in-vehicle platform.
  - Bridge: Application that translates all the information from the vehicle into oneM2M and is responsible to publish it and provide any needed methods to obtain this data.
  - In-vehicle Applications: The needed applications that will carry the use cases mentioned before; AVP and Urban driving, inside the vehicle platform. These applications will interact with their respective cloud versions in order to provide the full functionality expected in the use cases.
  - Vehicle AE Interface: Application that forwards the in-vehicle IoT information to the TNO/Central IoT platform. This module allows an *mca* connection between the IoT platforms, avoiding the *mcc* connector not yet available between the different oneM2M IoT platforms.
  - Remote management: this software component allows the remote monitoring and control of different IoT devices/applications.
- **Runtime environment**: OSGi framework that contains the stack that enables the V2X communication.
  - V2X Component: Contains several modules that are able to process data coming from V2X communication through ITS-G5. Provides the encoding/decoding for the SPAT/MAP, CAM and DENM messages.
  - Connectors: Provides the in-vehicle IoT platform with various connectors to transfer information from different sources. The connectors give the IoT module access to the CAN bus, GPS information and other V2X data that can be received via ITS-G5.
- **Intra-vehicle network**: Internal modules of the car (no-IoT).
  - Vehicle sensors: The different sensors or sources of useful data for the vehicle.
  - AD control system: Module responsible for the AD functions.

2.5.3 Verification

The verification tests performed by the Spanish pilot site are the following:

- SP-CTAG-1: IoT in-vehicle platform – REST Urban
- SP-CTAG-2: IoT in-vehicle platform – AVP
- SP-CTAG-3: In-vehicle platform availability – basic connectivity
- SP-CTAG-4: IoT in-vehicle platform to TNO platform
- SP-CTAG-5: Cellular connectivity
- SP-CTAG-6: Wi-Fi connectivity
- SP-CTAG-7: Urban data models
- SP-CTAG-8: AVP data models
- SP-CTAG-9: AVP events

In the following tables are the specifications of the tests:

Verification test ID	SP-CTAG-1 (M18)
Test Title	IoT in-vehicle platform – REST Urban
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Basic test of connectivity between a REST Service and the in-vehicle platform.
Involved software components	Communication interfaces, In-vehicle IoT Broker, In-vehicle Urban application, (Urban Service).
How the test is realized	With a mock urban service running, communicate the in-vehicle platform with it and send/receive messages.
Pass test criteria	In-vehicle IoT platform is able to request and receive information from the Urban Service with a REST API.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-2 (M18)
Test Title	IoT in-vehicle platform – AVP
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Basic test of connectivity between an IoT Service and the in-vehicle platform.
Involved software components	Communication interfaces, In-vehicle IoT Broker, In-vehicle AVP application, Vehicle AE Interface, (AVP Service).
How the test is realized	Perform some communication between the AVP Service and the in-vehicle AVP app through the IoT infrastructure.
Pass test criteria	In-vehicle IoT platform is able to request and receive information from the AVP Service with IoT messaging.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-3 (M18)
Test Title	In-vehicle platform availability – basic connectivity
Link to T2.5	Anticipates IoT_platform type verification.
High level objective of the test	Basic test of connectivity directly to the in-vehicle IoT platform.
Involved software components	Communication interfaces, In-vehicle IoT Broker.
How the test is realized	IoT in-vehicle platform is able to receive and send messages.
Pass test criteria	In-vehicle IoT platform is available and can receive and process IoT messages.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-4 (M18)
Test Title	IoT in-vehicle platform to TNO platform
Link to T2.5	Anticipates IoT_platform type verification
High level objective of the test	Connectivity between the in-vehicle IoT Platform and the TNO IoT Platform through an AE.
Involved software components	Communication interfaces, In-vehicle IoT Broker, Vehicle AE Interface, TNO IoT Platform.
How the test is realized	Registration of the Vehicle AE into the TNO Platform.
Pass test criteria	Successful registration and communication.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-5 (M18)
Test Title	Cellular connectivity
Link to T2.5	Anticipates IoT_platform type verification
High level objective of the test	Basic test of connectivity only with cellular networking.
Involved software components	Communication interfaces (Cellular), In-vehicle IoT Broker.
How the test is realized	Disable other communication interfaces and check the connectivity with the cellular module only.
Pass test criteria	Messages are correctly sent and received.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-6 (M18)
Test Title	Wi-Fi connectivity
Link to T2.5	Anticipates Vehicle_safety_urban_drivingtype verification.
High level objective of the test	Basic test of connectivity only with 802.11 Wi-Fi networking.
Involved software components	Communication interfaces (Wi-Fi), In-vehicle IoT Broker.
How the test is realized	Disable other communication interfaces and check the connectivity with the Wi-Fi module only.
Pass test criteria	Messages are correctly sent and received.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-7 (M18)
Test Title	Urban data models
Link to T2.5	Anticipates Vehicle_safety_urban_driving type verification.
High level objective of the test	Verification of preliminary urban data models (traffic lights, VRUs, hazards, etc.)
Involved software components	Communication interfaces, in-vehicle IoT broker, in-vehicle Urban application, (Urban Service).
How the test is realized	Messages relative to the Urban use case (traffic light, VRUs, hazards) are sent and received and can be understood by the in-vehicle IoT platform.
Pass test criteria	Messages are correctly processed as traffic lights, VRUs or hazards by the in-vehicle IoT platform and verified via logging outputs.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-8 (M18)
Test Title	AVP data models
Link to T2.5	Anticipates Vehicle_safety_valet_parking type verification.
High level objective of the test	Verification of preliminary AVP data models (VRUs).
Involved software components	Communication interfaces, in-vehicle IoT broker, in-vehicle AVP application, (AVP Service).
How the test is realized	Messages relative to the Urban use case (VRUs) are sent and received and can be understood by the in-vehicle IoT platform.
Pass test criteria	Messages are correctly processed as VRUs by the in-vehicle IoT platform and verified via logging outputs.
Results [yes / no]	Yes

Verification test ID	SP-CTAG-9 (M18)
Test Title	AVP events
Link to T2.5	Anticipates Vehicle_safety_valet_parking type verification
High level objective of the test	Verification of IoT events received by in-vehicle IoT platform.
Involved software components	Communication interfaces, in-vehicle IoT broker, in-vehicle AVP application, (AVP Service).
How the test is realized	Event messages (drop-off, pickup) are sent and received by the in-vehicle IoT platform, and processed correctly by the in-vehicle AVP application.
Pass test criteria	AVP in-vehicle application correctly processes the pickup and drop-off events.
Results [yes / no]	Yes

2.5.4 Data logging and management

The Spanish pilot site in-vehicle IoT platform will be logged according the Autopilot requirements. The data will be stored following the format defined by the Autopilot consortium in each of the communication points as shown in the figure below.

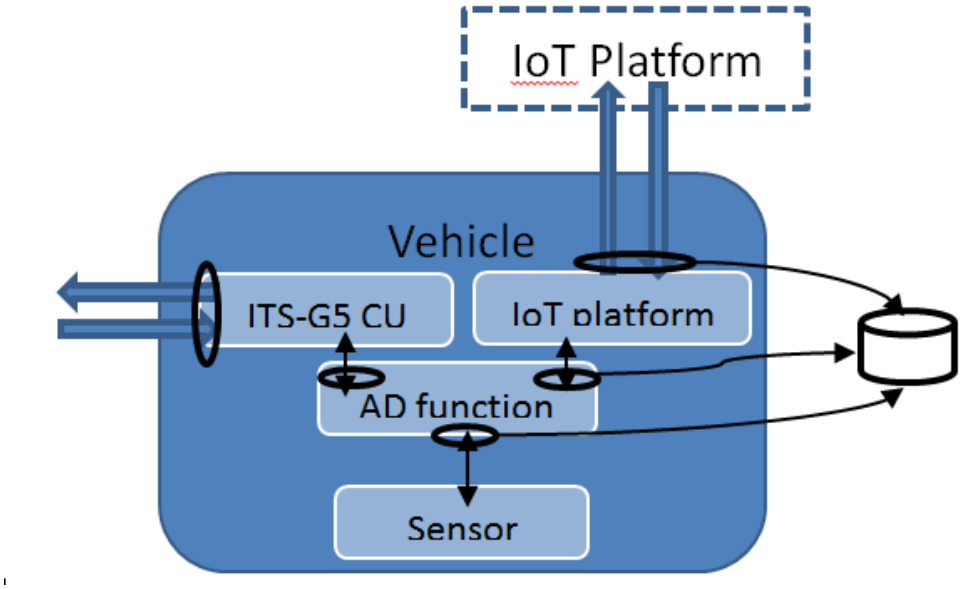

Figure 21: IoT software components architecture diagram of the CTAG prototype vehicle

The in-vehicle IoT platform will use the Google Protocol buffers (PROTOBUF) for some of the in-vehicle data. Furthermore, InterCor's format will also be used for logging ITS-G5 data, and this format will be extended to allow the logging of the IoT messages.

3 Data recording and management in the in-vehicle IoT platform

3.1 Overview

In addition to its IoT capabilities, the in-vehicle IoT platform of T2.1 should handle logging and accurate transfer of data emanating from the vehicle, as it is the gateway between the vehicle and the IoT world.

WP4 sets requirements of the data that has to be collected, within D4.1 [4] and in a related spreadsheet [5], and gets feedback from WP3 on their availability [6]. WP4 also proposes a format in which data should be available at the Central Test Server, in order to be correctly interpreted by analysts. The approach has been to start from Intercor [7] and extend it for AUTOPILOT usage [8].

T3.4 defines a methodology for data collection, including data from vehicles and their sensors [9]. I.e., T3.4 should ensure that data get converted in the proper format before being available to analysts at the CTS. Of course, the earlier this is done (e.g. in the design of an IoT device/vehicle) the better.

Therefore, T2.1 verifies that on-board unit do not only have IoT capabilities, but can also log in a proper format, and specifically, as treated in this chapter, T2.1 defined the vehicle data log format for AUTOPILOT.

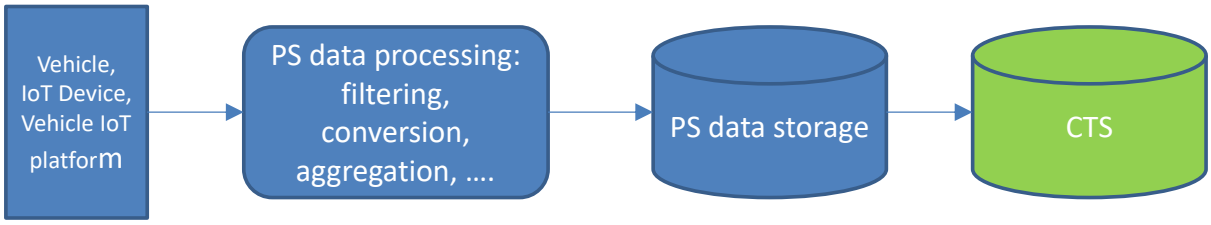

Figure 22: Data management chain. WP4 sets requirements for the CTS. Data conversion may happen in any step before.

AUTOPILOT needs to agree on these common aspects regarding the data uploaded at the Central Test Server (CTS):

1. **Logical organisation (grouping) of data into data sets/files**
   e.g. split Vehicle Data in GPS data, target detection, sensor types
   e.g. split communication into files of same message types
2. **Definition of parameters in data set (name, type, value range, frequency, …)**
   e.g. define speed in m/s (see AUTOPILOT_WP4_DataReqs_0.5.xlsx)
3. **Definition of file encoding and format**
   e.g. define encoding and file format (PROTOBUF, json, xml, UPER, csv, sql)

Data types that should be recorded in the on-board system include:

- Vehicle data
- V2X On-board communication logging
- IoT On-board communication logging
- Event data
- Situational data

T2.1 has focused on Vehicle data, as WP4 requested a specific contribution to T2.1, in the data format definition. This is the topic of the next sections. Concerning the rest of the data, the on-board platforms are expected to follow the guidelines given by AUTOPILOT. Hereafter, a brief overview is given.

For V2X On-board communication, data about the transmitted and received CAM and DENM should

be logged, as needed by the technical evaluation. The format at the end of the chain should follow the Intercor Common Communication [10] and its extensions currently being defined within AUTOPILOT.

For IoT data, the reference is T2.3 IoT data model and the IoT Data Model Task Force. Specifically, the messages sent from vehicles to the IoT platform e.g. for car sharing, AVP, and platooning use cases should be compliant to the "vehicle package" based on SENSORIS (reference is a dedicated GitLab repository at https://gitlab.com/autopilot/iot-data-model and wiki at https://gitlab.com/autopilot/iot-data-model/wikis/home). Also other use cases, e.g. those included in the Italian pilot site, follow a similar model. For evaluation purposes, it is expected that the in-vehicle IoT platform should at least record: sent message "uuid" (Universal Unique Message Identifier), sent message contents and sending timestamp, and received message "uuid" and reception time.

Event data types refer to the logged Autonomous Driving (AD) function, and to the underlying service logics. This is the main data type to use for extracting the IoT added value, as it includes the actions generating IoT data aimed at the in-vehicle system, the triggers and control actions taking place in the vehicle system upon reception of IoT data, and the effects of IoT data on (changes in) AD behaviour. Generalizing, events include also Human Machine Interface (warnings), e.g. in case of partially or non-automated driving, or also in case an HMI simulates a manoeuver which cannot be done due to safety reasons. Each of these cases is considered as an event model. The reference is AUTOPILOT_CommonApplicationLogFormat_extension_v0.7.7.xlsx, where an example was provided by TNO on Platooning. Every partner (i.e. every pilot site) has to define an EVENT MODEL (ref. Intercor §5.2) to clarify the sequence of event related to the specific Use Case. This action is left at general IoT level, but it impacts on the different on board platform prototypes, which need to include given event logging corresponding to given pilot site/use case.

Situational data are external situations which may affect AD functions, and possibly cannot be detected by the in-vehicle sensors (or they are detected from a parallel source). These include traffic congestions, traffic management decisions, incident validation, traffic light control data, weather conditions etc. Situational data are generally recorded at Test Site level (e.g. weather). Some vehicle-related situational data are expected to be received through V2X/IoT (e.g. the Signal Phase and Time – SPAT received on board) and, from the logging point of view, are treated as V2X and IoT messages respectively.

3.2 Vehicle data recording and management

3.2.1 Basic assumptions

Similarly to the other data types, the basic assumptions for vehicle data are taken from the Intercor Project, which states [10]:

- *Every vehicle, platform and device provides its own logging, and manages the integrity of its logging with unique identifiers and time stamping*
- *Log data is provided per experiment, test run or test session. Data loggers should manage the size, test sessions and chronological order of log data. When logging is provided in separate files, the filenames should make should make this explicit by including the log_stationid, log_applicationid and a starting timestamp in the log file names (…)*
- *All stations and applications that generate logging are time synchronized. Time synchronization issues cannot be fixed afterwards*
- *All timestamps are logged in a common time format, time zone and time unit: Coordinated Universal Time (UTC) in milliseconds since Unix epoch (number of milliseconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds (in ISO 8601: 1970-01-01T00:00:00Z).*

- *Timestamps in other time formats are converted in the logging to avoid a posteriori conversion and interpretation issues in other software tools.*
- *If timestamps in the original message are essential for identification of the message, referencing or analyses, then these should obviously be logged and appended, together with the converted value in UTC.*

- *Locations or positions are defined in WGS84 coordinates: latitude, longitude, bearing/heading. Latitude and longitude should be in degrees with 10^-7 precision. Locations may be supplemented with roadid, direction, lane id, etc. for reference.*
- *Data element names should be unique. When data element names are reused within a log station, log application or message type, they are assumed to have the same semantics and units. To avoid issues in conversion between tools, it is recommended to use only lower case characters, digits and underscores ("_"). No spaces in the names. Date element names with capital letters should also be unique when all letters are converted to lower case letters.*

3.2.2 Logical Organization in files/datasets

File naming is an AUTOPILOT modification of the from Intercor definition [10] and uses

<messagetype>_<log_stationid>_<utc_time_iso8601>[_<formattype>].<filetype>

In particular, since almost all the IoT on-board units, i.e. the in-vehicle IoT platforms, are expected to integrate ITS G5 to send Cooperative Awareness Message (ETSI ITS G5 CAM message); and since prototypes are generally associated with pilot sites and the related data management, the following coding has been decided:

<log_stationid> = Country Code (2 digits) + ITS G5 station id (2 digits)

For instance, Pilot Site Italy OBU numbering can be: 3901, 3902, etc.; and Pilot Site the Netherlands 3101, 3102, etc.

Conventionally, the OBU maintains this coding also when going to another pilot site than the one originally associated. For instance, a French vehicles going to another pilot site outside France, will maintain their numbers (3301, 3302, etc.).

All components, levels, loggers, sensors, communication units, etc. in a single vehicle should have the same stationid = log_stationid. The principle is that everything that travels along the same trajectory and with the same speed is part of the same physical station.

To distinguish the logged parameters within a *stationid*, every component, level, logger, sensor, communication unit, IoT platform, can have a log-applicationid that is unique within the same stationed, similarly to Intercor (the reference is Intercor Common Data Logging [7] , and specifically sections 2.2 and 3.1-3.3).

3.2.3 Definition of parameters in data set

AUTOPILOT identifies Data types, which are similar to the InterCor Layers. The classification comes from AUTOPILOT WP4 data requirements [ref. AUTOPILOT_WP4_DataReqs_0.5.xlsx]. From a practical point of view, T2.1 working group made reference to the pilot site feedback spreadsheet [AUTOPILOT_DataRequirements_PSFeedBack.xlsx].

Data types include

- Vehicle data (topic of the present task)
  - Vehicle Sources
  - Vehicle Data
  - Derived Data
  - Positioning

- V2X Messages
- IoT Messages
- Events
- Situational Data

Concerning Vehicle Data and Derived Data, the general rule to be followed is that WP4 does not need raw sensors data, but rather the information of detections by in-vehicle sensors.

Given the differences in data availability from pilot sites' on board units, another guideline is to leave out completely the columns of datasets that are systematically unavailable, while leaving empty fields when data are occasionally unavailable.

3.2.4 In-vehicle data log format

Using the general approach and guidelines described in 3.1 and coming mainly from Intercor, T2.1 issued a spreadsheet which is a comprehensive definition of the schema for vehicle data logging to a central repository for evaluation. This spreadsheet is organized in Excel panels (or tables), and includes

- two introductory tables with explanation and versioning, respectively
- one table (named "each table") with meta data for the data that need to be logged with every message in a file or row(s) in a table
- the following table corresponding to the (log) message contents, namely:
    - vehicle
    - positioning_system
    - vehicle_dynamics
    - driver_vehicle_interaction
    - environment_sensors

The format tables are reported in Table 1.

Table 1 – table "each table" (meta-data)

Name	Type	Range	Unit	Description	ADA
rowid	serial	0..	[N/A]	-- sequence of row numbers to uniquely identify a log line by <log_stationid, log_timestamp, rowid>, only necessary when a subtable is logged	C
log_timestamp	long	from 0 to 4398046511103 (= 2^{42}-1)	msec	-- timestamp at which the log_stationid logs (writes) the data row. elapsed time since midnight January 1st 1970 UTC	M
log_stationid	long	from 0 to 4294967295 (= 2^{32}-1)	[N/A]	-- unique id of the host (e.g. stationid, server id, IoT platform or device id, cloud service id, ...) that logs this log data row. Log_stationid can be another host than the source generating the data to be logged	M
log_applicationid	long	from 0 to 4294967295 (= 2^{32}-1)	[N/A]	-- unique id of the application, instance or thread, on the log_stationid host that logs this log data row. Applicationid is at least unique within the log_station. ApplicationId is mandatory if multiple components on a host log to the same table or if the application logging into a table is not trivial (e.g. it is trivial that a CAM Basic Service is the only application logging CAM messages in the cam	O

Table 2 – table "vehicle"

Name	Type	Range	Unit	Description
speed	double	from 0 to 163.82	[m/s]	Speed over ground, meters per second.
outsidetemperature	double	from -60 to 67	[°C]	Vehicle outside temperature during trip.
insidetemperature	double	from -60 to 67	[°C]	Vehicle inside temperature during trip.
batterysoc	double	from 0 to 100	[%]	Percentage of the battery of the vehicle.
rangeestimated	double	from 0 to 1000	[km]	Range estimated with the actual percentage of the battery and/or available fuel.
fuelconsumption	double	from 0 to 1	[L/km]	Average fuel consumption during a route or trip.
enginespeed	int	from 0 to 10000	[1/min]	Engine speed calculated in terms of revolutions per minute.
owndistance	double	from 0 to 5000	[km]	Total kilometrage per day or trip or road type etc.

Table 3 – table "positioning_system"

Name	Type	Range	Unit	Description
speed	double	from 0 to 163.82	[m/s]	Speed over ground, meters per second. Measured by GNSS receiver.
longitude	double	from -90 to 90	[degree]	Longitude
latitude	double	from -180 to 180	[degree]	Latitude
heading	double	from 0 to 360	[degree]	Heading
ggasentence	string		[NMEA format]	GGA - Fix information.
gsasentence	string		[NMEA format]	GSA - Overall Satellite data.
rmcsentence	string		[NMEA format]	RMC - recommended minimum data for gps.
vtgsentence	string		[NMEA format]	VTG - Vector track an Speed over the Ground.
zdasentence	string		[NMEA format]	ZDA - Date and Time.

Table 4 – table "vehicle_dynamics"

Name	Type	Range	Unit	Description
yawrate	double	from -327.66 to 327.66	[°/s]	Vehicle rotation around the centre of mass of the empty vehicle. The leading sign denotes the direction of rotation. The value is negative if the motion is clockwise when viewing from the top.
acclateral	double	from -16 to 16	[m/s2]	Lateral acceleration of the vehicle.
acclongitudinal	double	from -16 to 16	[m/s2]	Longitudinal acceleration of the vehicle.
accvertical	double	from -16 to 16	[m/s2]	Vertical acceleration of the vehicle.
speedwheelunitdistance	double	from 0 to 163.82	[m/s]	Sensor on free running wheel for increased accuracy. Speed measured from wheels (???).

Table 5 – table "driver_vehicle_interaction"

Name	Type	Range	Unit	Description
throttlestatus	int	from 0 to 100	[%]	Position of the throttle pedal (% pushed). Modify to boolean (i.e., 0->NOT PUSHED, 1-> PUSHED) if % is not available on the car.
clutchstatus	int	from 0 to 100	[%]	Position of the clutch pedal (% pushed). Modify to boolean (i.e., 0->NOT PUSHED, 1-> PUSHED) if % is not available on the car.
brakestatus	int	from 0 to 100	[%]	Position of the brake pedal (% pushed). Modify to boolean (i.e., 0->NOT PUSHED, 1-> PUSHED) if % is not available on the car.
brakeforce	double	from 0 to 300	[bar]	Measure of master cylinder pressure.
wipersstatus	enum	['OFF' 'ON']	[N/A]	Position of the windscreen wipers (boolean). Extend the enumeration if more details are available (e.g., ['OFF', 'SLOW', 'FAST'], ['OFF', 'SLOW1', 'SLOW2', 'FAST1', 'FAST2']).
steeringwheel	double	from -720 to 720	[°]	Position of the steering wheel.

Table 6 – table "environment_sensors"

Name	Type	Range	Unit	Description
longitude	double	from -90 to 90	[degree]	Main object transformed to geolocalized coordinates longitudinal (log_applicationid identifies the sensor providing this measurement (e.g., camera, LIDAR, radar...)).
latitude	double	from -180 to 180	[degree]	Main object transformed to geolocalized coordinates lateral position (log_applicationid identifies the sensor providing this measurement (e.g., camera, LIDAR, radar...)).
obstacle_ID	int	from 0 to 1000	[-]	ID of the obstacle detected by environmental sensors.
x	double	from 0 to 500	[m]	Main object relative distance longitudinal / x-direction (log_applicationid identifies the sensor providing this measurement (e.g., camera, LIDAR, radar...)).
y	double	from -50 to 50	[m]	Main object relative distance lateral / y-direction (log_applicationid identifies the sensor providing this measurement (e.g., camera, LIDAR, radar...)).
obstacle_covariance	float64			Covariance matrix of positions of longitude, latitude, altitude of RADAR detected objects.
ObjectClass	int	from 0 to 65	[-]	65 classes from Mapillary dataset[1]
lanewidthsensorbased	double	from 0 to 10	[m]	Lane width measured by on-board sensor(s).
lanewidthmapbased	double	from 0 to 10	[m]	Lane width from map information.
trafficsigndescription	string		[N/A]	signrecognition[2]
speedlimit_sign	double	from 0 to 250	[km/h]	signrecognition [3]
servicecategory	enum	['dangerWarning', 'regulatory', 'informative', 'publicFacilities', 'ambientCondition', 'roadCondition']	[N/A]	signrecognition [4]
servicecategorycode	int	[11, 12, 13, 21, 31, 32]	[N/A]	signrecognition[5]
countrycode	string		[N/A]	signrecognition [6]
pictogramcategorycode	int	from 0 to 999	[N/A]	signrecognition [7]
VRU_pedestrian_class	int	from 0 - 3	1 = children, 2 = adults, 3 = elderly	Sub classes of pedestrians.
VRU_cyclist_class	int	from 0 - 3	1 = children, 2 = adults, 3 = elderly	Sub classes of cyclists/riders.
confidence_levels	double	from 0 - 100	[%]	Indication for false positive detections (minimum default level).
Environ_info	int	from 1 - 6	[-]	1=sunny/day, 2=raining/day, 3=snow/day, 4=night/dry, 5=raining/night, 6=snow/night
Road_hazard	int	from 0 to 42	[N/A]	No standardized dataset available --> current proposal: pothole detection, slippery road, black ice etc.
sensor_position	int	from 0 to 1000	[mm]	Position of sensor on vehicle wrt. CoG. required for correlating to environmental detection with IoT detections.
process_delay	int	from 0 to 1000	[ms]	Is processing delay known or unknown?

3.2.5 In-vehicle data log encoding

For the data encoding, InterCor proposes: SQL, CSV, XML (§4.2) [add ref.], but any human readable format is acceptable. For instance, ITALY proposes PROTOBUF for all logs from IoT platform.

4 Conclusions

Within T2.1, the integration of IoT in the vehicle has been performed. Within the vehicle an on board component, called “on-board IoT platform”, enables the connectivity of AUTOPILOT vehicles, the exchange of vehicle data with the IoT, thus making the vehicle an “IoT device”, and includes processing functionalities such as to contribute to IoT applications and enable to the autonomous driving use cases of Automated Valet Parking, Urban Driving, Highway Driving, Highway Pilot and Car rebalancing for Shared Vehicles. Being Table 7 summarizes the on board IoT integration in the different prototypes. It highlights the main interfaces, in line with the general concept scheme outlined in the introduction (Figure 1), namely:

- connectivity of the vehicle with neighbour entities, for vehicular networking
- main connection by the in-vehicle IoT platform to the general IoT platform
- other IoT connections, e.g. on-board services with their own cloud, synchronised with the IoT platform independently from the in-vehicle-IoT platform.
- interface to the intra-vehicle network (OEM-vehicle network)
- interface to local IoT devices and applications (additional IoT network in the vehicle)

Table 7 – Summary of IoT integration in AUTOPILOT vehicles: main interfaces

Pilot Site/Prototype	**Neighbour entities connection**	**Main IoT connection**	**Other IoT connections**	**Interface to intra-vehicle network**	**Interface to IoT devices and applications in the vehicle**
PS Finland	ETSI ITS G5	MQTT (LTE/4G)	-	CAN	DDS
PS Italy	ETSI ITS G5	ETSI OneM2M over MQTT (LTE/4G)	Conti connected eHorizon (IoT connection in the cloud)	CAN	To Inertial sensors: 6LoWPAN (CNIT vibration sensor), CAN (CRF IMU) and MQTT over WiFi (smartphone)
PS France	ETSI ITS G5	ETSI OneM2M HTTP requests		CAN	ROS
PS NL/ TNO & NEVS	ETSI ITS G5 /UWB	ETSI OneM2M via Websocket requests (LTE/4G)	-	CAN	ROS/UDP
PS NL/TUE	ETSI ITS G5	OneM2M over MQTT requests HUAWEI OceanConnect	NEC motion planning NEC Crowd Detector via	CAN	ROS

Pilot Site/Prototype	Neighbour entities connection	Main IoT connection	Other IoT connections	Interface to intra-vehicle network	Interface to IoT devices and applications in the vehicle
		over HTTP (LTE/4G)	NGSI (FIWARE)		
PS NL/VALEO	ETSI ITS G5	MQTT publish anomaly to VALEO cloud	TomTom Horizon server/client	CAN	UDP
PS NL/IBM	-	OneM2M/MQTT (LTE/4G)	-	-	
PS NL/DLR	-	MQTT (LTE/4G)	-	CAN	ROS
PS Spain	ETSI ITS G5	ETSI OneM2M over HTTP/MQTT		CAN	OM2M

As a whole IoT integration choices have been rather heterogeneous in AUTOPILOT, especially for locally connected IoT devices and applications. However, in order to fulfil the use cases, basic principles have been followed, namely to ensure the interaction with the pilot site IoT platform, the usage of ITS G5 for vehicular networking and the access to the intra-vehicle network when needed, both to control the vehicle and to obtain more precise data on vehicle dynamics.

The developed systems have been checked by prototype leaders in order to get ready for system verification in T2.5 and then adapted and extensively tested within WP3. The few remaining checks will be handled within the piloting activity.

Another important topic for the sake of the pilots is the data recording within the vehicle. This aspect is key to evaluate the autonomous driving behaviour, provided by T2.2. Beyond addressing the requirements from other AUTOPILOT tasks, this task T2.1 has contributed to the definition of in-vehicle data format.

References

[1] AUTOPILOT deliverable D1.5 “Initial Open IoT Vehicle Platform Specification”, version 2.1, 2 October 2018
[2] AUTOPILOT deliverable D2.5 “Readiness verification approach, version 2.1, 29 September 2018
[3] AUTOPILOT deliverable D1.9 “Initial Specification of Security and Privacy for IoT-enhanced AD”, 27 September 2018
[4] AUTOPILOT deliverable D4.1 “Methodology for evaluation”, version 1.0, 31 January 2018
[5] AUTOPILOT WP4 Data requirement spreadsheet, 19 March 2018. Explained in D4.1
[6] File “AUTOPILOT_DataRequirements_PSFeedBack.xlsx” spreadsheet , 25 May 2018
[7] Netten, B., “Intercor Common Data Logging”, version 0.7.7, 10 April 2018
[8] File “AUTOPILOT_CommonApplicationLogFormat_extension_v0.7.7.xlsx” spreadsheet, 3 May 2018
[9] AUTOPILOT deliverable D3.6 “Data collection and Integration methodology”, 23 April 2018
[10] File “InterCor_CommonCommunicationLogFormat_v0.7.7.xlsx” spreadsheet, 10 April 2018