APPSTACLE

(ITEA 3 – 15017)

open standard APplication Platform
for carS and TrAnsportation vehiCLEs

Deliverable: D 4.1

Concept of Vehicle to Cloud to Ecosystem to Cloud to Vehicle HMI Use Case

Work Package: 4

Experiments and Demonstrations

Task: 4.1, 4.2

Description of Demonstrator Scenarios and Connected Car Functionalities, Demonstrator Concepts Development

Document Type:	Deliverable	**Classification:**	Internal
Document Version:	final	**Contract Start Date:**	01.01.2017
Document Preparation Date:	31.03.2018	**Duration:**	31.12.2019

ITEA 3
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

History

Rev.	Content	Resp. Partner	Date
0.1	initial document structure	Robert Hoettger	05.10.2017
0.2	adapted document structure	Markus Schmidt	15.01.2018
0.3	added IDE content	Robert Hoettger	25.01.2018
0.4	added input from several partners	Markus Schmidt	29.03.2018
0.5	added Summary, Introduction, and Use Case Chapter	Markus Schmidt	17.04.2018
1.0	rework of HMI Demonstrator chapter	Markus Schmidt	11.05.2018

Contents

List of Figures

List of Tables

Summary

This deliverable is the first deliverable of the APPSTACLE Work Package 4 "Experiments and Demonstrations". It contains the results of two tasks: "Description of Demonstrator Scenarios and Connected Car Functionalities" (T4.1) as well as "Demonstrator Concepts Development" (T4.2). Logically, this document is split into three parts. One chapter describes additional use cases which are covered by the demonstrator scenarios, the following chapters describe the main demonstrator scenarios envisioned by the APPSTACLE partners, and the appendix contains the description of common software tooling and common used hardware.

Chapter 2 describes uses cases additional to those documented in Deliverables D1.1 [2] and D3.1 [3]. These uses cases are especially adapted to demonstrators scenarios developed in this document.

Chapter 3 to Chapter 5 of this deliverable describe different demonstrator scenarios, especially,

- Chapter 3: HMI demonstrator covering several uses cases,
- Chapter 4: Vehicle demonstrator with main uses cases of driver authentification and driver profiling,
- Chapter 5: Security demonstrators.

The appendix shows several topics common to more than one demonstrator, these are

- Appendix A: Software tooling - description of the development in Eclipse Che,
- Appendix B: Hardware - description of the APPSTACLE project board (Gateway),
- Appendix C: Infrastructure - description of the 5G test network.

1. Introduction

The purpose of Work Package 4 "Experiments and Demonstrations" within the APPSTACLE project is to experiment with, evaluate and demonstrate the results of the Work packages 1 to 3 in a proof-of-concept approach. Therefore, this work package is cross-sectional in relation to these other work packages. Figure 1.1 illustrates this overarching relationship of Work Package 4 with the other work packages in APPSTACLE.

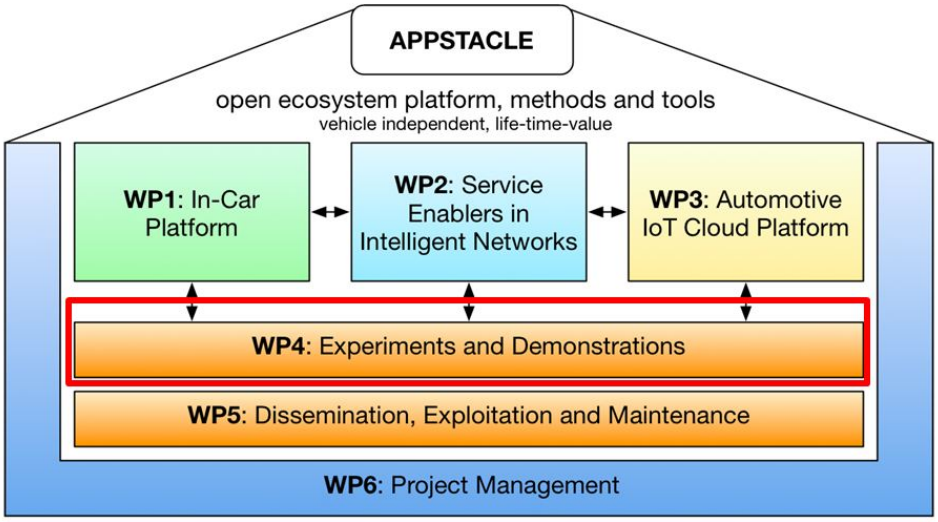

Figure 1.1.: APPSTACLE Work Package 4 and its relation to other work packages.

This document describes the scenarios and technical systems that will constitute the demonstrators for the APPSTACLE project. As such it also documents the work that has been done in Task 4.1 (Description of Demonstrator Scenarios and Connected Car Functionalities) and parts of Task 4.2 (Demonstrator Concepts Development) of the APPSTACLE project.

Due to its cross sectional nature, all demonstrator concepts presented in this deliverable (Chapter 3 to Chapter 5.1) have a strong relationship to other deliverables, in particular to

- D1.1 'Specification of In-car Software Architecture for Car2X' [2]
- D2.1 'SotA Research with regard to Car2X Communication, Cloud and Network Middleware and corresponding Security Concepts' [1]
- D3.1 'Specification of Data Management, Cloud Platform Architecture and Features of the Automotive IoT Cloud Platform'. [3]

The user scenarios and use cases to be implemented in the demonstrators are taken from these deliverables. Some of these user scenarios and use cases are adapted to the demonstrators

presented in this deliverable and some specific scenarios are added (see Chapter 2). In the future implementation of the demonstrators, subsequent results of the work packages will then be applied.

This twofold view on the results of the work packages, use cases vs. development platform (APPSTACLE, Eclipse Kuksa), should be considered for the future development and documentation of the demonstrators, as indicated in Figure 1.2.

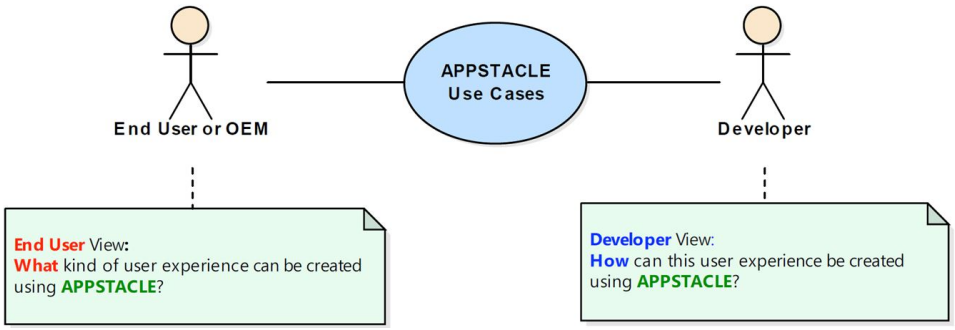

Figure 1.2.: Different views on APPSTACLE Demonstrator Use Cases.

The demonstrators should not only show what can be developed with the APPSTACLE platform, but also how the results of the APPSTACLE project can be used for this development.

Another challenge of Work Package 4 is the development of common components for the demonstrators, be they software development environments, hardware, or network infrastructure. Some of these shared capabilities and resources are presented in the appendix to this document.

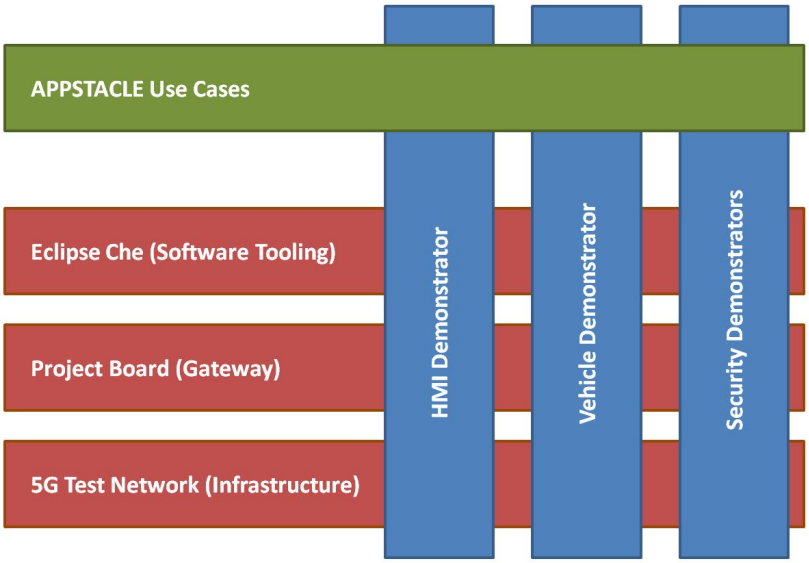

Figure 1.3.: Cross sectional structure of deliverable D4.1.

These connections and interdependencies of the topics presented in this document are illustrated in Figure 1.3. This image therefore also shows the structure of this deliverable.

2. Use Cases

To enable the Work Package 4 demonstrators to explore the possibilities of the APPSTACLE platform, typical use cases for these demonstrators are needed. In Deliverables D1.1 [2] and D3.1 [3] (the same) 22 user stories were described and presented. These might be broadly categorized as shown in Figure 2.1. They are mainly concerned with the actual end user experience or with

Figure 2.1.: Kind of use cases defined in deliverables D1.1 [2] and D3.1 [3].

the business opportunities and service offerings of car manufacturers or third parties.

In Deliverable D2.1 [1] the Use Cases 'Platooning', 'Over-The-Air-Updates', and 'Emergency Warnings' were presented. The focus of these use cases was more on deriving requirements for the performance, quality, and security of communication and connection.

Additional use cases for the demonstrators, which will be described below, mostly extend those 22 use cases and user stories presented in D1.1 [2] or D3.1 [3] Accordingly, the numbering of D3.1 [3] will be continued for the new uses cases, starting with User Story 23.

2.1. User Story 23: Climate control data collector

This scenario describes the process of collecting data concerning car climate control. The driver's behavior concerning the climate control settings, internal ECU data, and additional measurement data from the environment are sent to the cloud, which might be provided by the car manufacturer. This data can then be analyzed and used for finding defects or improvement of future control software.

Development Phase: The application developer would implement the software to connect to the car manufacturer's cloud. This would include the development of the in-vehicle side

ECUs and the gateway which have the capability to communicate to a cloud. Furthermore the algorithm for receiving and analyzing the data would be developed for the cloud. Thus a client-server relationship should be established.

Setup Phase: After starting the car, the car as an entity should register itself in the cloud. After a successful registration the application gets the permission to communicate with the cloud.

Usage Phase: Every time when the driver starts the car, the connection to the cloud should automatically be established . The data collection can begin. After the driver turns off the car, the communication should be terminated.

Technical Requirements

- Runtime environment for applications on the in-vehicle platform
- APPSTACLE API for supporting different bus and network systems
- APPSTACLE platform should allow to maintain the connection permanently after it has been established

2.2. User Story 24: Climate account

The car driver can create an account to storing climate control settings and features. If the driver changes to another car, he can download all these settings via his personal climate control account.

Development Phase: The developer would implements a database and a service on the cloud. The cloud stores different accounts and the air-conditioning settings associated to the current car and driver.

Setup Phase: The driver has to create a climate control account.

Usage Phase: The driver can stores different settings using his account. When the driver changes car, he can log in into the account and download the climate control settings to get his preferred comfort.

Technical Requirements

- Runtime environment for applications on the in-vehicle platform
- APPSTACLE API for downloading and uploading data
- APPSTACLE Platform should allow for driver identification and authentification

2.3. User Story 25: Driver/Passenger Authentication

This user story describes a scenario that a passenger transformation company provides driver authorization and passenger boarding pass scanning service.

Development Phase: The app developer implements an app for a passenger transportation vehicle (most likely a bus) to provide driver authorization and passenger boarding pass scanning service during pick-up and drop-off. It sends the driver/passenger data to a third party management server on the cloud that persists driver and passenger data. The management server reads the data and provides appropriate credentials.

Setup Phase: The passenger transformation company downloads and installs the appropriate app from an app store on the passenger transportation vehicle. The app registers with the management server instance. The driver and passenger proves their credentials by reading QRCODE that is provided by the company web portal.

Usage Phase: The app displays at the HMI screen as a welcome app during the driver starts the vehicle. The driver reads the QRCODE provided by the company and authorized to drive the vehicle. If the authorization fails, the driver could not drive the vehicle. The app also scans the passenger boarding pass information by reading QRCODE provided by the company.

Technical Requirements

- Runtime environment for apps on the in-vehicle platform
- APPSTACLE API granting access the appropriate ECU for driver authorization
- Security and privacy of the data

2.4. User Story 26: Driver Profiling

Vehicle driver profiling is an increasing need for fleet management and insurance companies. Fleet management companies can reduce fuel consumption and increase vehicle engine lifetime by controlling driver's drive style. Also insurance companies can create innovative business solutions for their customers. According to these needs drivers' driving style can be evaluated by interpreting data gathered from vehicle data bus.

Development Phase: The app developer develops an in-car specific app that communicates with the cloud that in turn communicates with the third parties such as; fleet management and insurance companies.

Setup Phase: The app developer develops an in-car specific app that communicates with the cloud that in turn communicates with the third parties such as; fleet management and insurance companies.

Usage Phase: Once the app is switched-on and driver logged-in app, it gather predefined parameters of vehicle such as; gear, engine speed, speed, acceleration, steering angle change, throttle and brake pedal pressures. Then evaluates all parameters to score the driver and send that score to the cloud along with the driver and location data. The cloud app stores and visualize driver profile data and serves to the third party platform customers.

Technical Requirements

- Runtime environment for app in the in-vehicle and cloud platform
- The platform has necessary APIs, protocols to communicate with a system outside its ecosystem
- APPSTACLE API providing access to vehicle data bus, GPS and etc.

3. HMI Demonstrator

3.1. Introduction

The HMI demonstrator will be provided by Behr-Hella Thermocontrol (BHTC), an automotive Tier-1 supplier of climate control units and human-machine interface (HMI) systems. BHTC has expertise in the field of automotive electronic control units (ECUs) and in-vehicle networks. Consequently, the focus of the activities and functionalities of this demonstrator will be on the vehicle side. However, to show use cases which deal with connectivity, the communication technology and the cloud platform can not be ommited. These parts will be only be developed to that degree which is neccessary for the intended uses cases. So, the actual used communication technology in the demonstrator scenarios might not be suitable in a real car and some of the cloud platform frontend and backend software might be a mock-up.

If one considers Figure 3.1, in which the structure of the work packages is shown, mainly the results of Work Package 1, In-Car Platform, and to a lesser extent those of Work Package 3, Automotive IoT Cloud Platform, are used for this demonstrator.

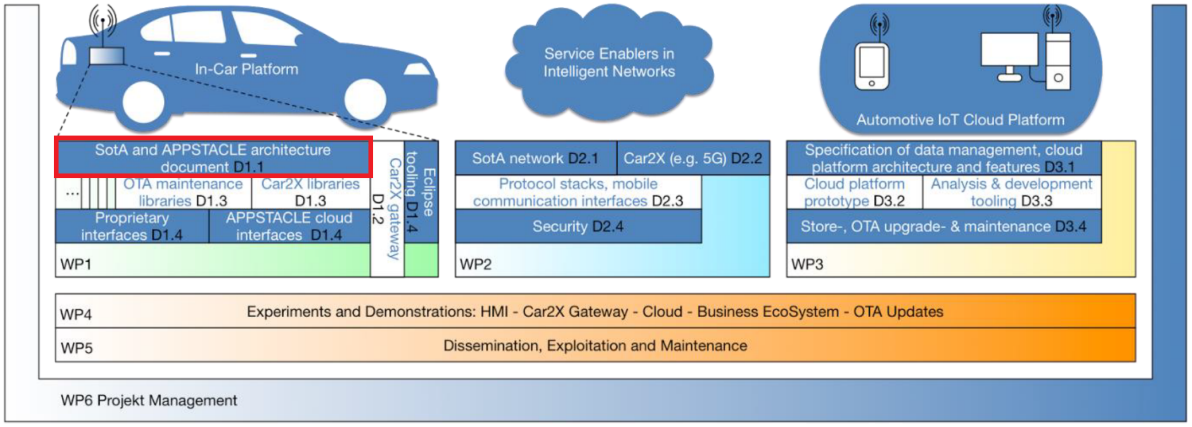

Figure 3.1.: APPSTACLE Work Packages Structure and Technical Deliverables. [2]

The results of Work Package 2 might be considered in a later stage of the demonstrator development. For a start the connectivity is taken as given and of no concern for the implementation of the uses cases.

The use cases which will be realized in this demonstrator are a selection from the uses cases presented in Deliverable D1.1 [2] (see: Figure 3.1, red box) and two additional use cases presented in this document.

The HMI demonstrator's use cases are focused on data transfer between the demonstrator and the cloud and demonstration of user experience. For ease of implementation, security issues are not adressed by the demonstrator (this is done in partner demonstrators, see Chapter 5).

3.2. Demonstrator overview

A system overview over the planned demonstrator is shown in Figure 3.2. Here the in-vehicle system is shown in the lower left part of the figure and the part realizing the services of the automotive IoT cloud platform on the upper right. Both are connectetd by a cloud symbol which indicates that the actual communication technology is abstracted away in the context of this demonstrator. In both parts, the in-vehicle system and the IoT cloud platform, the results of APPSTACLE will be utilized. Since the results of APPSTACLE will be incorporated into the Eclipse Kuksa project, every part of the demonstrator where results from APPSTACLE are used, are marked with a Kuksa symbol. These are the APPSTACLE Project board (Gateway), the APPSTACLE Community Board (HMI board), the server, and the frontend device to access the server.

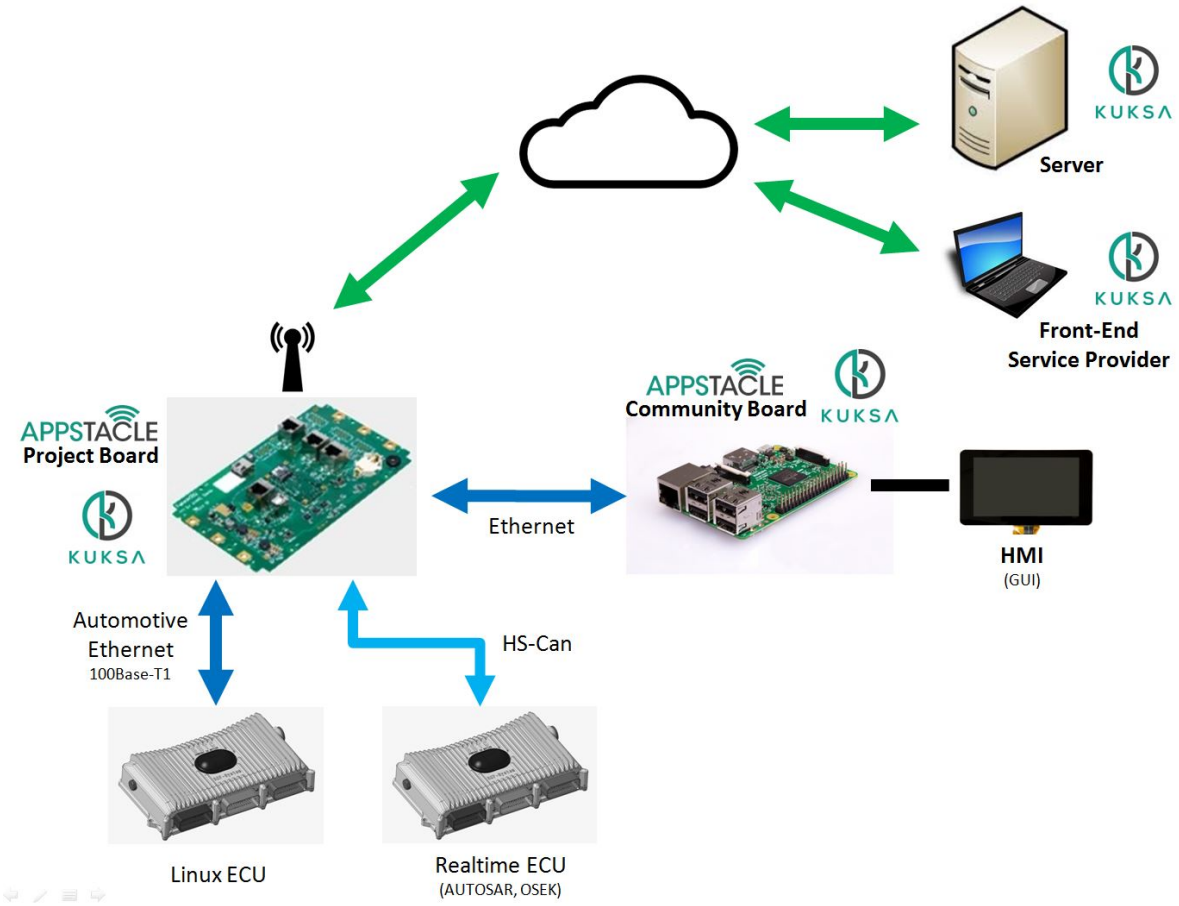

Figure 3.2.: System overview of the HMI demonstrator.

3.2.1. In-vehicle system structure

The core of the in-vehicle system is depicted in Figure 3.3. The different parts and their connection will be described in the following.

The APPSTACLE in-vehicle platform will be deployed onto the Project Board (Gateway)

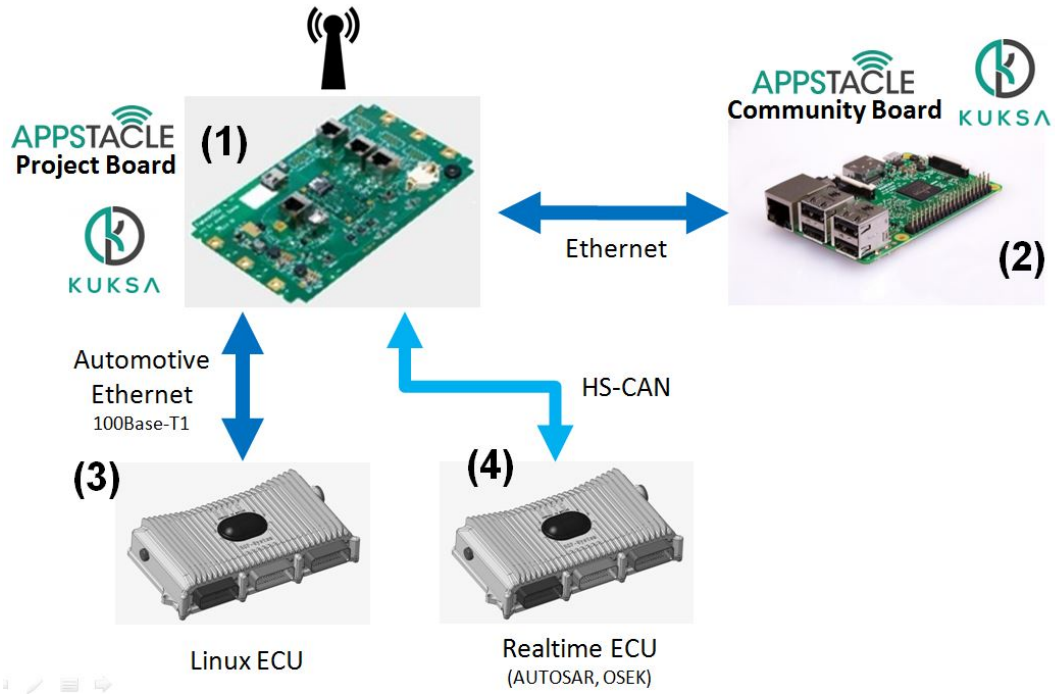

Figure 3.3.: In vehicle side of the HMI demonstrator.

and the Community Board (HMI board). The architecture of this platform was specified in Work Package 1 and is shown for reference in Figure 3.4.

The Elements of the in-vehicle system will now be described in more detail.

APPSTACLE Project Board The central element of the in-vehicle system is the APPSTACLE Project Board (1) which will be provided by taskit and is further described in Appendix B. This board is establishing the connectivity for the in vehicle part and it will be having Automotive Grade Linux (AGL) and the APPSTACLE platform (resp. Eclipse Kuksa) running on it.

The board serves as the central gateway and all other control units are directly connected to it. The gateway establishes the (wireless) connection to the cloud and controls the communication from the cloud to the ECUs and vice versa. Additionally to the gateway functionality, some functions needed for the use cases might be implemented for this board and will be running on the APPSTACLE platform

APPSTACLE Community Board The APPSTACLE Community Board (2) is a Raspberry Pi 3 with AGL and the APPSTACLE platform (resp. Eclipse Kuksa) running on it. This technological decision was taken as part of a selection process in Work Package 1. One reason for this is the general availability of the hardware and the availability of AGL for this special hardware. It is connected via Ethernet to the APPSTACLE Project Board (1). Ethernet was chosen to allow for very high data rates.

In the context of this demonstrator, this board is used to provide the car user interface and to run those applications for the APPSTACLE platform that may not run on the APPSTACLE Project Board directly due to the reduced computing power there. The provided user interface

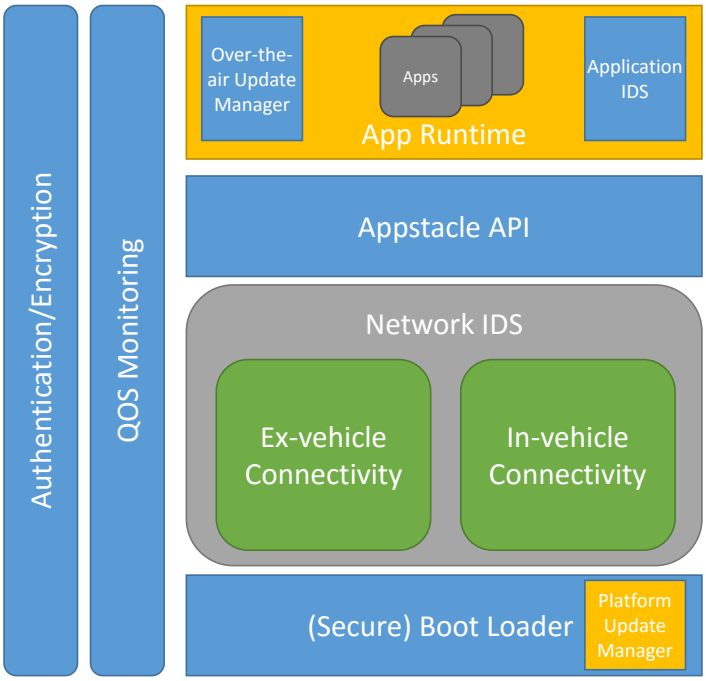

Figure 3.4.: The logical architecture of the APPSTACLE in-vehicle platform. [2]

will be a touchscreen with a graphical user interface (GUI) where the user will be able to communicate with services in the cloud or can see information provided by the cloud.

Additional control units Since some use cases require additional ECUs that are not part of the APPSTACLE connectivity domain, two additional ECUs ((3) and (4)) are connected to the gateway. One ECU (3) is connected via Automotive Ethernet to allow for use cases that need higher data rates. Consequently, the operating system of choice in this case will be some Linux system, because it will be able to deal with these data rates. For use cases which need lower datarates and realtime control capabilities, there will be a second ECU (4) which is connected via a CAN bus. The operating system on this ECU will be some automotive real time operating system like AUTOSAR OS or some other OSEK compliant operating system.

3.2.2. Cloud platform

The cloud platform has to provide several services and management capabilities. These generic capbilities are shown in Figure 3.5.

The elements of the demonstrator, where these functionalites of the (automotive) IoT cloud platform are realized, are depicted in Figure 3.6. Essentially they are a server for the backend (here most of the aforementiond functionality is located) and a possibility for accessing this server. For conceptional reasons these two elements are shown separately here, but in the actual demonstrator they might be located on the same hardware device (e.g. a laptop).

Server The server (5) is the backend of the whole demonstrator setup. Here the services and platforms for device management, data analytics, data management, telemetry, and so forth (cf.

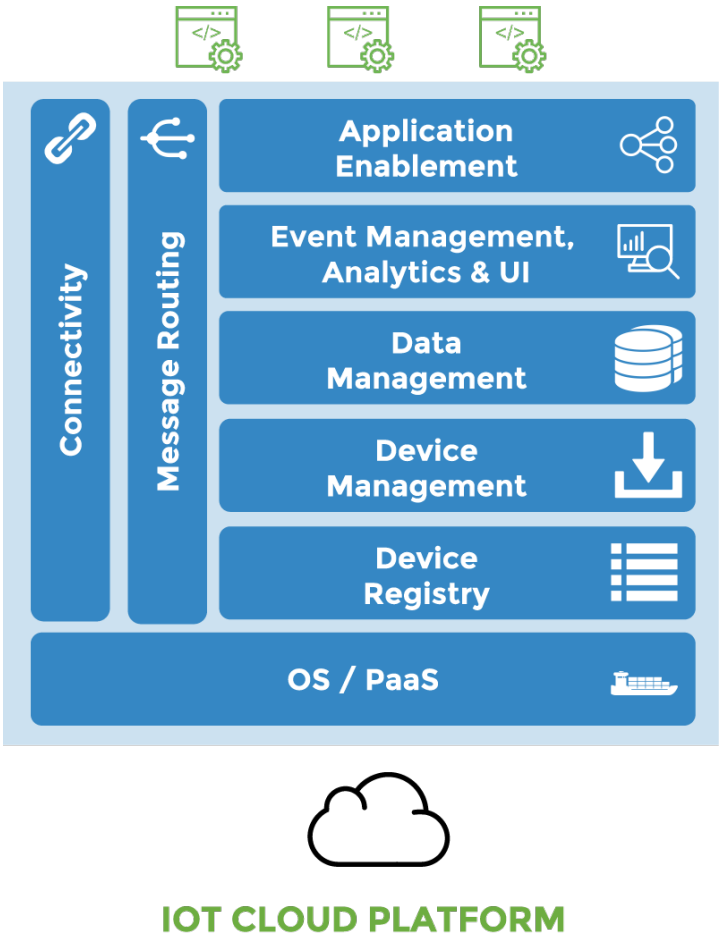

Figure 3.5.: Software stack for an IoT Cloud Platform [4].

Deliverable D3.1 [3]) are located. These might be, amongst others, Eclipse Hono for messaging or Eclipse hawkBit for device and update management [3]. Also the neccessary brokers for publisher-subscriber protocols (like MQTT [3]) are located on this server.

Which platforms and technologies will ultimately be integrated into Eclipse Kuksa and thus become relevant for this demonstrator has not yet been fully determined. WP3 is working on that.

Frontend for Service Providers The frontend (6) provides the access to the backend on the service provider side. In an automotive setting this might be an employee of a car manufacturer who wants to access telemetry data from several cars. For example, Grafana might then be used for data visulisation. To demonstrate the value of APPSTACLE, e.g., a webpage could be built (at least a simple cloud program), which stores features (e.g. climate features). Those can be downloaded to the ECUs via the APPSTACLE, e.g., to enhance a better climate comfort.

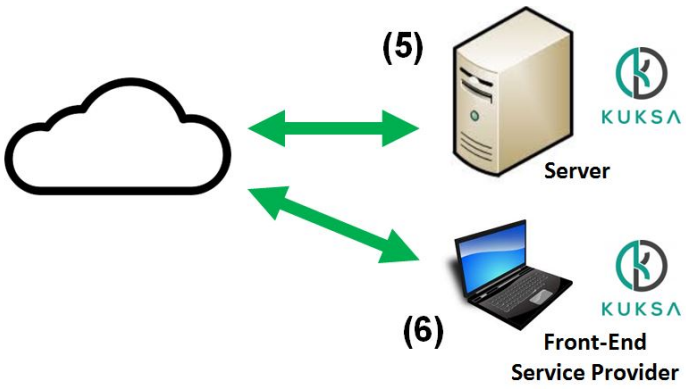

Figure 3.6.: Cloud platform of the HMI demonstrator.

3.3. Considered Use Cases

The aim of the HMI demonstrator is to implement and showcase selected APPSTACLE use cases. Some use cases and scenarios are taken from Deliverable D1.1 [2] or Deliverable D3.1 [3] (where they are also listed). For the HMI demonstrator these are in particular (numbering according to D3.1 [3]):

- User Story 02: Vehicle Tracking,
- User Story 03: Wrong Way Driver Warning,
- User Story 04: Augment vehicle functionality,
- User Story 05: Data Collection Fleet Learning,
- User Story 07: Driver Seat Conguration,
- User Story 18: System Surveillance and Maintenance,
- User Story 20: In-vehicle behavior learning.

Two more use cases for this demonstrator have been presented in Chapter 2:

- User Story 23: Climate control data collector,
- User Story 24: Climate account.

One might argue that User Story 23 and User Story 05 and also Story 24 and User Story 07 are essentially the same. However, since User Story 23 and User Story 24 will be the ones actually implemented, they are added to the list. Both use cases are explained in more detail below.

3.3.1. Climate control data collector

In this use case data concerning the car's climate control system will be collected and sent to a server where this data can be analyzed and evaluated. There are essentially three kinds of data, which would be collected: the first type is measurement data from the car's environment (e.g., the interior temperature), the second type is internal data from the ECU, and the third

type is user input data (change of settings) which comes from the user interface. There are two main ideas behind this data collection. The first idea is to monitor the operation of the climate control functionality in the car. In case of deviation from the desired behavior, parameter or software updates might be initiated or the driver might be advised to go to a service station. The advantage compared to the current standard in-car diagnostic features is that the evaluation of data on the back-end server side might learn and be improved over time. The second idea is to take the data from many cars and learn how to improve the development of future climate control algorithms

Figure 3.7 illustrates the communication between the different hardware for this data collecting use case (in a simplified manner).

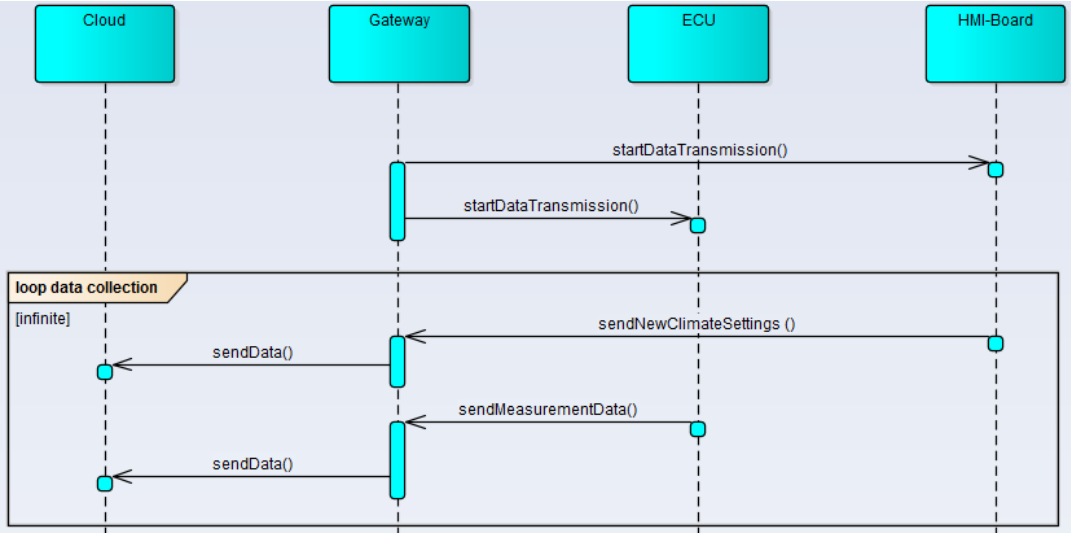

Figure 3.7.: Simplified behavior for the data collecting use case.

The gateway has to establish the connection to the cloud. After the connection is established to the cloud, an application (which runs on the APPSTACLE platform on the gateway) sends commands to the climate control ECU and the HMI-Board for starting the data collection and data transfer to the cloud (Figure 3.7). After the connection is established, the data is sent continuously. The HMI-Board has to send its data only at start and then on change of the climate settings. The data from the ECU (measurement data and internal control data) should be sent continuously.

3.3.2. Climate Account

The climate account use case is about the storage of prefered climate control settings in the cloud and synchronizing with those stored settings stored later on. This means that the comfort settings, which a user has set in a car, can be stored online. Furthermore, this settings can be downloaded to another car (same make and model, or settings are generalized over different models) to get the prefered air-conditioning. Since these settings will depend highly on the weather situation, several settings for typical weather conditions might be saved. For this intend the user must have some kind of an account on the back-end side, but for this demonstrator the account itself and its management (e.g. authentification) will not be implemented in detail. The focus will be on the connectivity issues like storing the profile and downloading it.

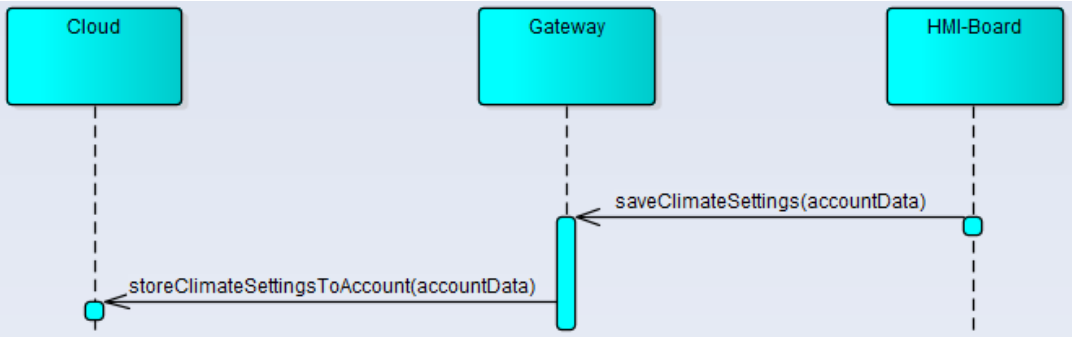

Figure 3.8.: Simplified behavior for the saving climate settings.

First of all the user has to select on the HMI-Board the appropriate GUI menu to save his climate settings. After that the HMI-Board then sends this settings to the gateway. The gateway itself should send this settings, which are associated to the users account, to the cloud (see Figure 3.8).

The other way around, the user has the possibility to download his climate settings to the current used car. To show this use case, the user has to select the GUI menu for getting the climate settings on the HMI-Board. After that, the APPSTACLE application should communicate with the gateway. This service has to forward the request to the cloud and receive the settings. Notably is the point, that the gateway must send the climate settings to the HMI-Board as well as to the ECU. This is very critical, because of the synchronisation between what the ECU get and what the GUI application shows as the current settings. This behavior can be seen in figure 3.9.

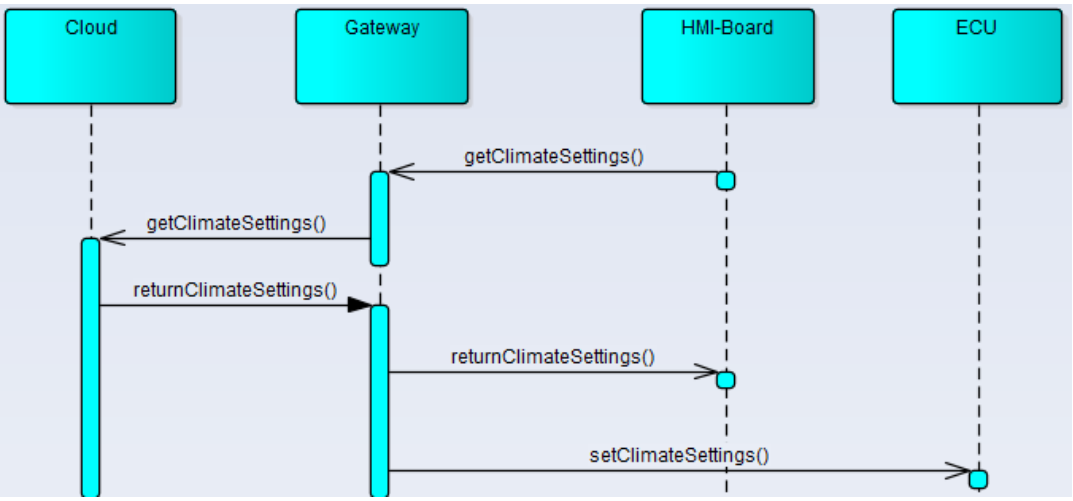

Figure 3.9.: Simplified behavior for the receiving climate settings.

4. Vehicle Demonstrator

4.1. Driver Authentication Demonstrator

This is one of the demonstrator scenarios focused by Turkish consortium. The idea in this scenario is to offer a driver authentication service for public transformation vehicles. The demonstrator shall be in the production facilities of the automotive manufacturing company Otokar settled in Sakarya, Turkey.

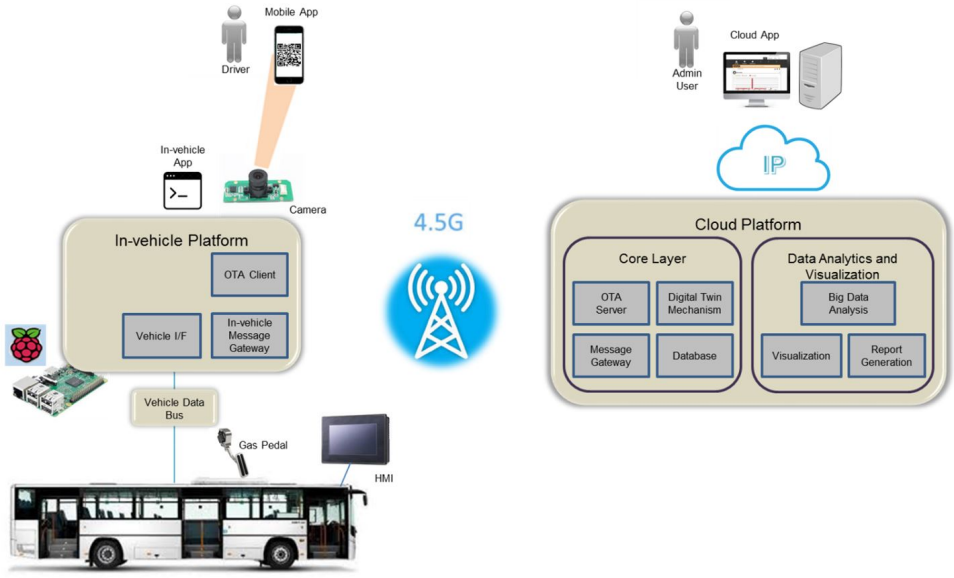

Figure 4.1.: Driver Authentication Demonstrator

As illustrated in Figure 4.1, Driver Authentication Demonstrator, the scenario demonstrates the following APPSTACLE components:

In-vehicle Platform: It is a platform that includes OTA client, in-vehicle message gateway and vehicle I/F software built on RPi3 board. OTA client performs software installation and update of in-vehicle applications according to the command messages coming from OTA server located at core layer of cloud platform. In-vehicle message gateway provides communication between in-vehicle platform and cloud platform. In-vehicle platform receives the vehicle data from vehicle data bus and send the commands to there through vehicle I/F that is a bridge between in-vehicle platform and vehicle. In the scenario, the status of vehicle gas pedal shall be the vehicle data and managed by cloud platform services. An in-vehicle application software shall handle this. Driver shall authenticate the system by a QRCODE that is generated by a mobile application.

Cloud Platform: It is a platform to support automotive IoT cloud services. The core layer of the platform consists of message gateway, OTA server, digital twin mechanism and database

services. The message gateway allows to connect vehicles through various protocols to business applications in the IoT manner. OTA server is a backend solution to install and update in-vehicle application software running on in-vehicle platform. Digital twin mechanism enables IoT applications to manage digital twins of in-vehicle platform entities. Database persists vehicle data in the cloud infrastructure. The data analytics and visualization layer consists of big data analysis, visualization and report generation components. Big data analysis processes the data stored in database for further analysis. Visualization allows visualizing data within flexible dashboards. Report generation generates business reports. A cloud application provides frontend and backend services to demonstrate the scenario.

In the scenario, Admin user and Driver are the main actors and they actuate the APPSTACLE components as described in the following steps:

- Cloud registration operation of in-vehicle platform
  - When In-vehicle platform is booted successfully, a registration message (HTTP, MQTT etc.) shall be sent to the Message gateway to register In-vehicle platform as a device in the cloud platform.
  - Message gateway in the cloud platform shall get the device registration message and send it to OTA server and Digital twin mechanism for further processing and In-vehicle platform shall be registered to Message gateway, OTA server and Digital twin mechanism as a device.
  - OTA server shall create In-vehicle platform as a target device into OTA server database.
  - Digital twin mechanism shall create a digital twin including metadata (chassis number, manufacturer, firmware version etc.) and feature data (location information and status of In-vehicle platform – by default it is online - and vehicle gas pedal - by default it is disabled) and store it in its database as a thing.
  - When In-vehicle platform powers down, it shall be disabled in the cloud components. At this stage, if In-vehicle platform sends a deregistration message, it shall be deregistered.
  - In-vehicle platform shall send heartbeat event messages to the cloud platform periodically. Cloud platform shall scan the registered and online objects periodically as well. If any heartbeat message is not received from an online object in a specified time, its connection status shall be set to offline.
- In-vehicle application software installation/update
  - Admin user shall be able to perform a command to install or update In-vehicle application software from management web page of OTA server. For this purpose, OTA server shall send a message to the OTA client running on In-vehicle platform. This message shall include the name of last version of the software and binary file repository link of the software.
  - OTA client shall handle the message coming from OTA server and install or update In-vehicle application software from the binary file repository link. OTA client shall reply a message to the OTA server about the status of installation (succeeded or failed).

  - OTA client shall display a status message on HMI after software installation process is completed.
  - OTA server shall keep and maintain the data as a time series data in OTA server database.

- Cloud application web portal admin operations
  - Admin user shall register Driver to the cloud platform with the data such as name, surname and driver's licence id.

- Driver mobile application software installation and QRCODE generation operations
  - Mobile application shall be installed from Android or Apple app store and Driver shall login with the user id registered by Admin user to the cloud platform.
  - Driver shall press a button to generate a dynamic QRCODE. A request message shall be sent to the cloud platform and Cloud application backend service shall generate a dynamic QRCODE with a specified duration. Cloud platform shall cache the QRCODE for this duration if the generation is succeeded.
  - Cloud platform shall reply a message to Mobile application about the status of QRCODE caching (succeeded or failed).
  - Mobile application shall display the QRCODE code generated if the cloud platform caching operation succeeds.

- In-vehicle and cloud platform messaging operations
  - Driver shall scan the QRCODE displayed on Mobile application from the camera plugged on In-vehicle platform. In-vehicle application shall receive it be and informed about the identity of Driver (name, surname and driver's licence id) and company name.
  - In-vehicle application shall pass the information provided by the dynamic QRCODE to the In-vehicle message gateway.
  - In-vehicle message gateway shall pass the information provided by In-vehicle application as well as chassis number and location information to cloud platform.
  - Message gateway at the cloud platform shall receive the message from In-vehicle message gateway and persist the data into Database. It shall also inform backend service of Cloud application for data processing.
  - Cloud application backend service shall compare the QRCODEs coming from In-vehicle platform and cached in the cloud platform. According to the comparison, a message shall be sent to Digital twin mechanism related to the proper state change in Vehicle gas pedal located in In-vehicle platform digital twin.
  - Vehicle I/F shall get the information related to digital twin from the cloud and send a message to Vehicle data bus to command Vehicle gas pedal.
  - The state of Vehicle data bus shall be changed according to the information coming from Vehicle I/F.

- Data analytics and visualization operations

- Big data analytics engine shall fetch the batch data from database including time series information such as Vehicle gas pedal state, Driver id, Driver plate and location and perform analysis.
- Visualization component shall display the stream and batch data fetch from database.
- Reports shall be generated according to data fetched from database.

Technical contributions: Turkish consortium will build together the scenario according to the APPSTACLE architectural components illustrated in Figure 1: Driver Authentication Demonstrator and each partner in the consortium contributes them according to their expertise and business interests: Kocsistem: In-vehicle platform Netas: Cloud platform Otokar: Vehicle data bus and HMI

4.2. Driver Profiling Demonstrator

This is one of the two demonstrator scenarios focused by Turkish consortium. The idea in this scenario is to figure out driver classification and driving style recognition using inertial sensors and environmental conditions. Driver behavior affects traffic safety, fuel/energy consumption and gas emissions. The purpose of driver behavior profiling is to understand and have a positive influence on driver behavior. The demonstrator shall be in the production facilities of the automotive manufacturing company Otokar settled in Sakarya, Turkey.

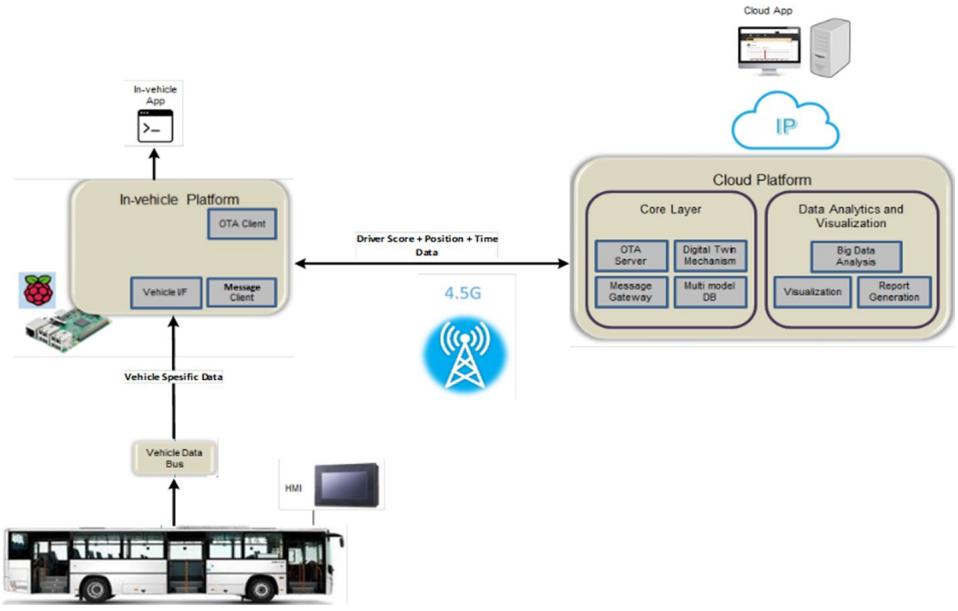

Figure 4.2.: Driver Profiling Demonstrator

Figure 2: Driver Profiling Demonstrator As illustrated in Figure 4.2, Driver Profiling Demonstrator, the scenario demonstrates the following APPSTACLE components: In-vehicle Platform: It is a platform that includes OTA client, in-vehicle message gateway and vehicle I/F software built on RPi3 board. OTA client performs software installation and update of in-vehicle applications according to the command messages coming from OTA server located at

core layer of cloud platform. In-vehicle message gateway provides communication between in-vehicle platform and cloud platform. In-vehicle platform receives the vehicle data from vehicle data bus and send the commands to there through vehicle I/F that is a bridge between in-vehicle platform and vehicle. A dedicated in-vehicle application gets vehicle data and process it to make inferences about drivers' behavior. Then this information is sent to cloud platform. Cloud Platform: It is a platform to support automotive IoT cloud services. The core layer of the platform consists of message gateway, OTA server and database services. The message gateway allows to connect vehicles through various protocols to business applications in the IoT manner. OTA server is a backend solution to install and update in-vehicle application software running on in-vehicle platform. Database persists vehicle data in the cloud infrastructure. Visualization allows visualizing data within flexible dashboards. Report generation generates business reports. A cloud application provides frontend and backend services to demonstrate the scenario.

In the scenario, end user and driver are the main actors and they actuate the APPSTACLE components as described in the following steps:

- In-vehicle application software installation/update
  - Admin user shall be able to perform a command to install or update In-vehicle application software from management web page of OTA server. For this purpose, OTA server shall send a message to the OTA client running on In-vehicle platform. This message shall include the name of last version of the software and binary file repository link of the software.
  - OTA client shall handle the message coming from OTA server and install or update In-vehicle application software from the binary file repository link. OTA client shall reply a message to the OTA server about the status of installation (succeeded or failed).
  - OTA client shall display a status message on HMI after software installation process is completed.
  - OTA server shall keep and maintain the data as a time series data in OTA server database.
- In-vehicle application driver profiling operations
  - Driver Authentication Scenario completed successfully.
  - Dedicated driver profiling application collects related vehicle information from vehicle I/F and feeds supervised artificial neural network to make inferences about driver behavior.
  - Driving style information output is sent to cloud platform for evaluating info in the light of environmental parameters such as; traffic density, daylight and etc.
- Cloud driving behavior operations
  - The output which is sent by in-vehicle platform driver profiling application is also evaluated with environment information as a second phase.
  - Second phase evaluation output is stored with related driver, date and time data in the database.
  - Visualization component shall display the data fetch from database.

Technical contributions: Turkish consortium will build together the scenario according to the APPSTACLE architectural components illustrated in Figure 2: Driver Profiling Demonstrator and each partner in the consortium contributes them according to their expertise and business interests: Kocsistem: In-vehicle platform Netas: Cloud platform Otokar: Vehicle data bus and HMI

5. Security Demonstrators

5.1. Demonstrator - SecurityMatters

5.1.1. Scope

This demonstrator is used as a showcase on the security aspects of the APPSTACLE platform. It specifically focuses on real-time monitoring as well as threat detection by identifying anomalies in the network. This process is based on the analysis of network packets that are exchanged within the APPSTACLE platform by an Intrusion Detection System (IDS), that is a network monitoring hardware device along with the software to support the logic for detecting potential attacks. The presence of the IDS in the APPSTACLE platform is transparent, since it is a fully passive solution that has a minimal impact in the platform's performance. Furthermore, the IDS uses existing state-of-the-art as well as novel algorithms that are developed in WP1 (in-vehicle IDS) and WP2 (V2X IDS) to recognize existing or novel attacks by looking at deviations from the normal system's behavior. Such algorithms allow the early-stage detection and analysis of malicious activities in the APPSTACLE platform.

5.1.2. Objective

Through the use of the NIDS the user of the APPSTACLE platform can get real-time notification for security threats, operational hazards and diagnostics that will allow him/her to schedule security or maintenance updates.

5.1.3. Overview

An overview of the final demonstrator that is currently being developed is presented in Figure 5.1.

This Figure presents the three main communication layers that are usually encountered in in-vehicle environments and are considered as a part of the APPSTACLE project. In particular, the in-vehicle layer includes all the subnetworks that are required for the correct functionality of the vehicle and have a direct interface with the vehicle electronics, such as the body-control, chassis, powertrain and infotainment subnetwork. The NIDS receives network traffic from the APPSTACLE in-vehicle HW (i.e. gateway) that is provided by taskit (Section 5.2). Accordingly, the platform layer features the APPSTACLE in-vehicle platform as well as its connections to the NIDS through different interfaces, such as the connection between the APPSTACLE in-vehicle HW and the monitoring sensor of the NIDS. This connection is handled through the CAN-Bus DSUB-9 interface that is shown in the in Figure.

This Figure illustrates the connections that are used to mirror all the network traffic that is taking place inside and outside of the vehicle. Moreover, these connections allow the NIDS to operate as a passive solution that is able to detect potential security threats or misconfigurations. In the same time it is capable of minimizing overall impact in terms of performance on the overall system. The NIDS is also connected to a Web server or an application running on

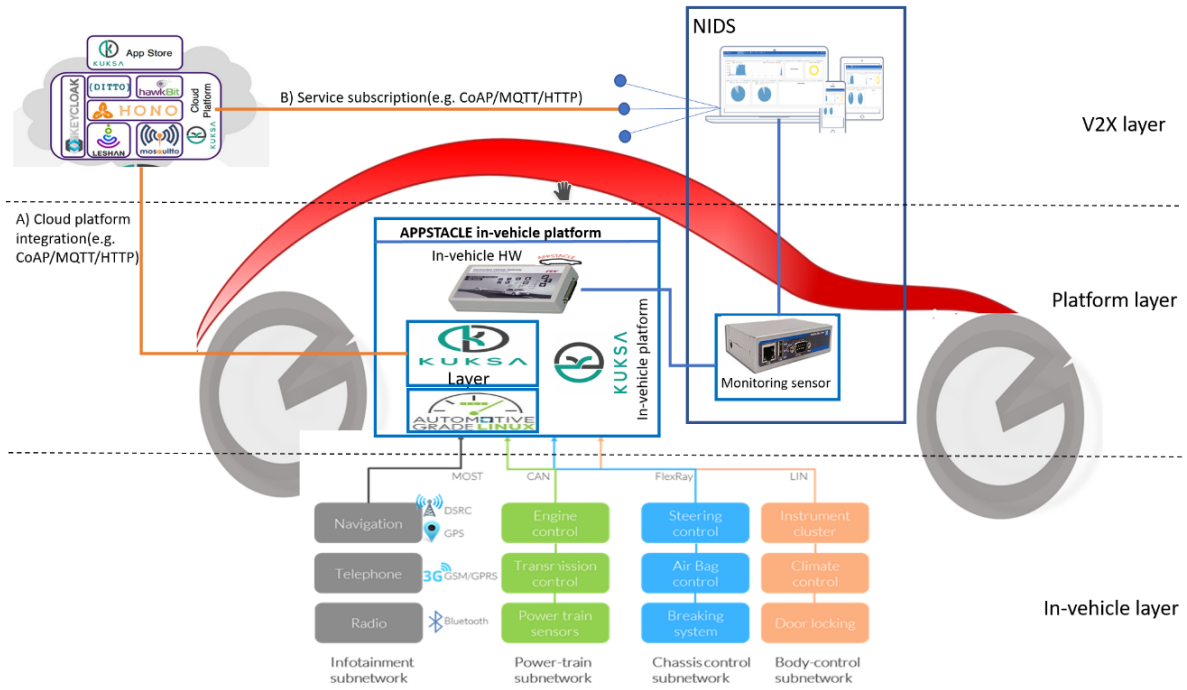

Figure 5.1.: Interactions of NIDS with the APPSTACLE in-vehicle platform

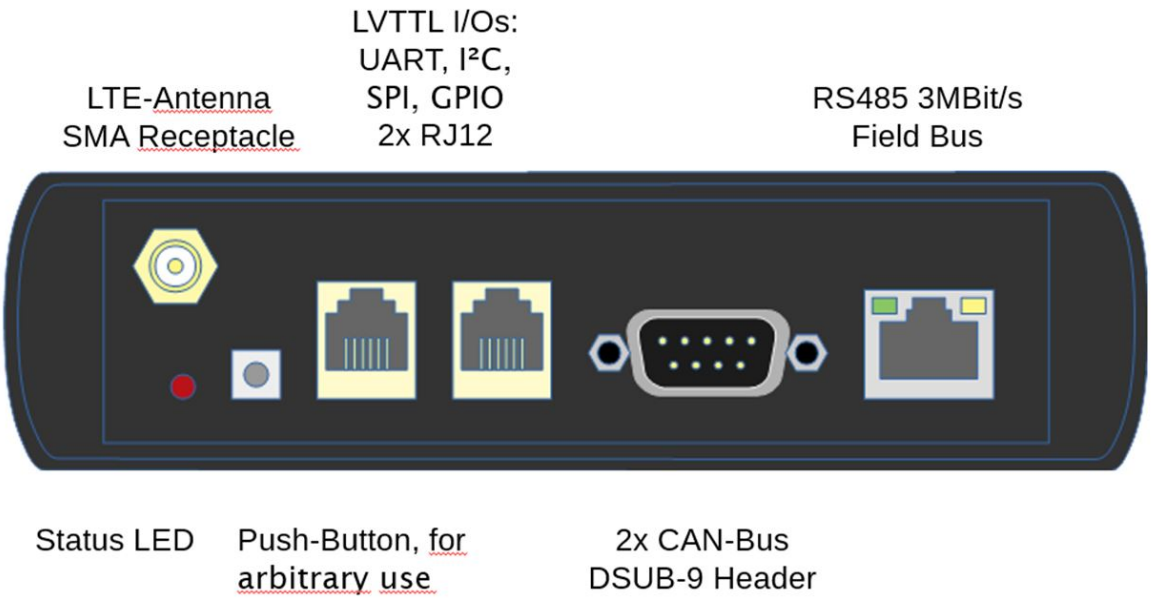

Figure 5.2.: APPSTACLE gateway interfaces

an embedded device (e.g. smartphone/tablet) that provides a user-friendly environment for the representation of the monitored environment and the detected threats or misconfigurations on it. This environment is provided as a service to the APPSTACLE Cloud platform that resides

on the V2X layer, namely the layer that represents all the ex-vehicle interactions. The APPSTACLE Cloud platform can also receive monitored in-vehicle communication events directly from the APPSTACLE in-vehicle platform through the Eclipse Kuksa layer [1].

5.1.4. Ongoing work

Existing state-of-the-art algorithms for the NIDS have been demonstrated in the deliverable D1.1: Specification of In-car Software Architecture for Car2X Applications as well as in the deliverable D2.1: SotA Research with regard to Car2X Communication, Cloud and Network Middleware and corresponding Security Concepts. The current work focuses in the integration of the developed algorithms in the Eclipse Kuksa layer, as well as in the development of novel intrusion detection algorithms based on deep packet inspection inside the vehicle as well as in its connections with other entities in the V2X environment (i.e. other vehicles, Road Side Units, pedestrians or even the interfaces to the Cloud).

5.2. Demonstrator - Technolution

Within the context of the APPSTACLE project, Technolution recognizes that the key concern of an OEM is vehicle security. Today, these core interests are already under threat, as third parties and end users are becoming aware of the lack of security in traditional vehicle buses.

This threat will only increase in the connected world as supported by the APPSTACLE platform. There are, however, legitimate reasons for access to on-vehicle systems which significantly enhance the utility and capability of the vehicle. The requirement is therefore to separate the legitimate actor from the malevolent one for read (and optionally write) access to the on-vehicle busses.

5.2.1. Demonstrator scenario

Some critical functions of a vehicle are owned by the OEM and are not accessible by APPSTACLE because these are safety affecting functions. Typically these might be brakes, motor management etc. Today ‘safety affecting’ and ‘comfort’ functions are often partitioned onto separate networks with narrow gateway functions between them (e.g. accessing engine performance information via the instrument module) but in a future architecture it shall be necessary for more nuanced access between these networks; A typical example might be the requirement for key-off running which is required by emergency service users to provide power for critical equipment when a vehicle is at the site of an incident.

Conversely, functionalities that have not historically been considered safety-affecting (e.g. route planning and comfort functions) can indeed have an impact on the safe transit of the vehicle when mis-configured; The classic example of this is routing an over height vehicle to collide with a constrained height bridge.

Overall, it is obvious that it is essential that access to any on-vehicle system must be viewed as a safety affecting functionality.

Within the constraints of APPSTACLE access to the OEM owned critical functions will not be implemented, but access to ‘comfort’ functions and APPSTACLE specific must still be controlled.

[1] https://projects.eclipse.org/proposals/eclipse-kuksa

In general therefore it can be seen the credential based fine grained access control is needed and a strong separation of APPSTACLE owned and OEM owned domains also needs to be established. This separation will also offer an additional barrier in case the security features of APPSTACLE are breached and an attacker managed to reach the safety critical functions of a vehicle.

5.2.2. Scenario description

A hacker has the intention to disable braking systems of vehicles. A possible attack vector could be the (APPSTACLE) wireless communication interfaces. Once the attacker managed to breach the APPSTACLE security barriers and managed to reach the inside of the Gateway or ECU, it should not be possible to influence the vehicle safety by reaching the safety domain or by disable/change safety related functions.

5.2.3. Demonstrator scope

The demonstrator focusses on software separation on single processor units which are often used in in vehicle systems. Techniques will be based on Microkernel systems specifically intended for safety/security applications.

A. Description of APPSTACLE's Software Development Environment

Initial efforts were spent to form a platform accessing both, the in-vehicle software and the cloud software. Therefore, a flexible and yet easy to use environment had to be identified that can easily switch between SDKs, libraries, development views, tools, or configurations. Since APPSTACLE addresses IoT technologies and state-of-the-art architectures, whilst C- and Cpp development workspaces mostly require immense configuration efforts, Eclipse Che has been chosen to alleviate the mentioned challenges. Being Cloud-based, Eclipse Che can be setup individually and shared among developers with all its configurations, SDKs, libraries and so on. Just the url-link must be entered into an arbitrary browser and no further setup is required to start either in-vehicle or spring-boot (cloud) application development.

Because APPSTACLE makes use of an immense amount of technologies, sophisticated concepts and architectural decisions had to be carefully elaborated and evaluated. In addition, those investigations should be compliant to an easy to use as well as established platform with, in the best case, a community, in order to build up upon and integrate the various ideas.

The APPSTACLE consortium finally came up with the architectural principle shon in Figure A.1.

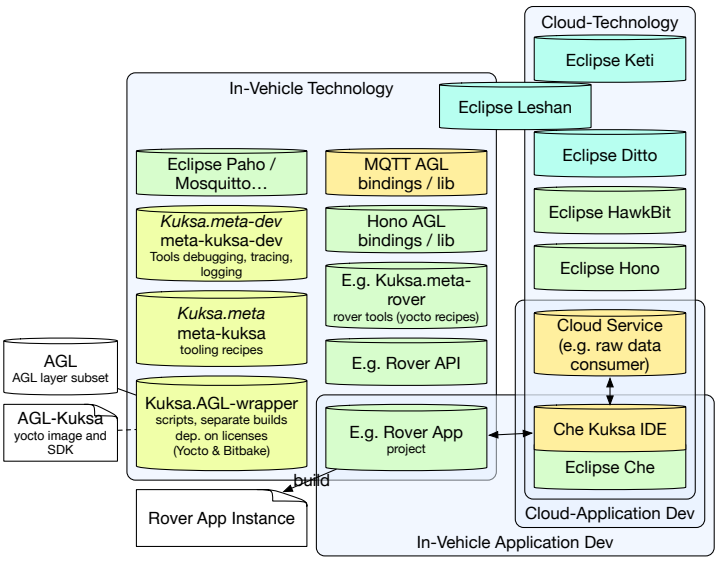

Figure A.1.: APPSTACLE Dev Architecture

Since in-vehicle development activities will be based on Automotive Grade Linux (AGL), as specified in work package 1, the AGL SDK will be required to be included to Eclipse Che as well as patched towards the needs of APPSTALCE such that bindings for the cloud (Hono), protocols (MQTT / LWM2M, REST), gateway APIs, security concepts, and more are included.

B. Description of Gateway

B.1. Overview

From the perspective of hardware development, it is therefore primarily the development of a Car2X Gateway. For this, the gateway uses developed communication protocols, interfaces and libraries. WP 4 covers the ECU and Car2X Gateway developments including maintenance functions. This task addresses the connection of the HMI to the ecosystem in order to ensure the interoperability between the technical solutions of the work packages 1-3 in the form of a continuous data exchange over the network as well as next to others integrate maintenance functionalities.

B.2. APPSTACLE Project Board

The main task of the distinct APPSTACLE project board consists in carrying out the connection between the central CPU and the Cloud via LTE and, later on, 5G. This involves all necessary software protocols and security checks. Apart from that, the Gateway serves as the central router for various in-vehicle fieldbus interfaces, which may include Automotive Ethernet, CAN, CAN-FD, LIN, (MOST, Flexray). Furthermore, ex-vehicle interfaces, e.g. ITS-G5, may be connected via USB or Ethernet.

The APPSTACLE Project board is built up with a baseboard with all connectors onboard. On that baseboard taskits DropA5D2 CoM runs with an ARM-Cortex-A5-processor. It provides one the newest microprocessors of Atmel, the SAMA5D22. This chip has the right price-to-performance ratio with the world's lowest power consumption for all MPUs in its class. Its ARM Cortex-A5 core includes NEON (floating point and parallelizing unit), as well as cryptographic functions like AES, SHA and a true random number generator (TRNG).

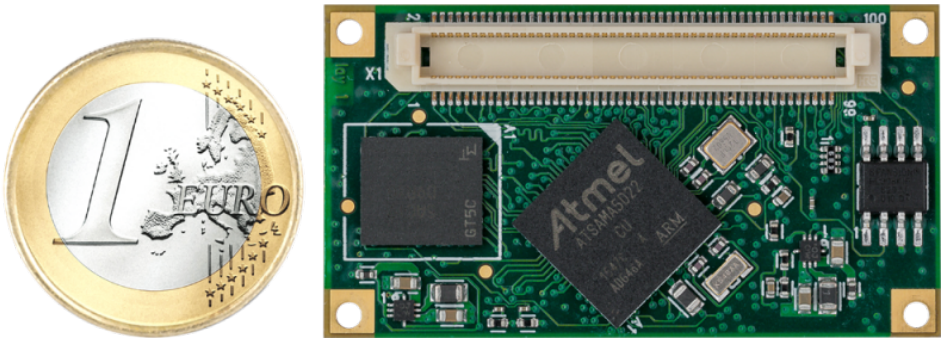

Figure B.1.: Processor module.

Most conspicuous feature of the DropA5D22 is the very small size: taskit shrinked the board to only 46 mm x 25 mm (1.81 in x 0.98 in) and further decreased power consumption.

Supply current in standby mode is less than 1 mA while keeping the entire RAM contents. The DropA5D22 wakes up from standby in well below 0.1 milliseconds. The Drop boots from its on-board serial flash and/or microSD card, or optionally from external flash memory. DropA5D22 is shipped with a full featured Linux 4.7 operating system. The comprehensive multiplexing of the 72 PIO-pins provides for various interface configurations.

If it is needed taskit delivers the APPSTACLE project board with a Human-Machine-Interface (HMI). This may consist of a TFT (e.g. 7") and a capacitive touchscreen. It could be used as a debug-screen or to show up traffic, events and action while demonstrating the use cases. The APPSTACLE project board should provide three Automotive Ethernet (e.g. IEEE 802.3 100Base-T1) and some other interfaces:

- GPS-Antenna SMA Receptacle
- LTE-Antenna SMA Receptacle
- Vin = 8..28V
- Terminal Block 3.5mm
- Ethernet 100Base-T
- USB 2.0 480MBit/s Host and Device
- LVTTL I/Os: UART, I^2C, SPI, GPIO
- 2x RJ12 RS485 3MBit/s Field Bus
- Status LED
- Push-Button, for arbitrary use
- 2x CAN-Bus in one DSUB-9 Header

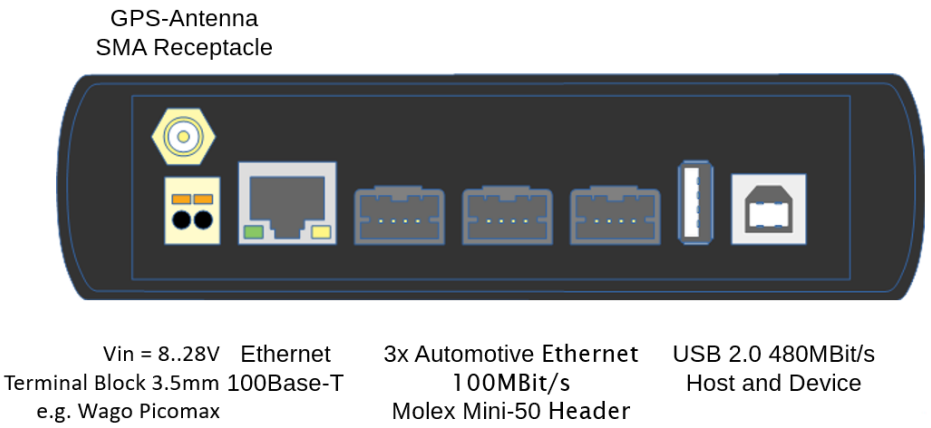

Figure B.2.: Gateway Interfaces 1.

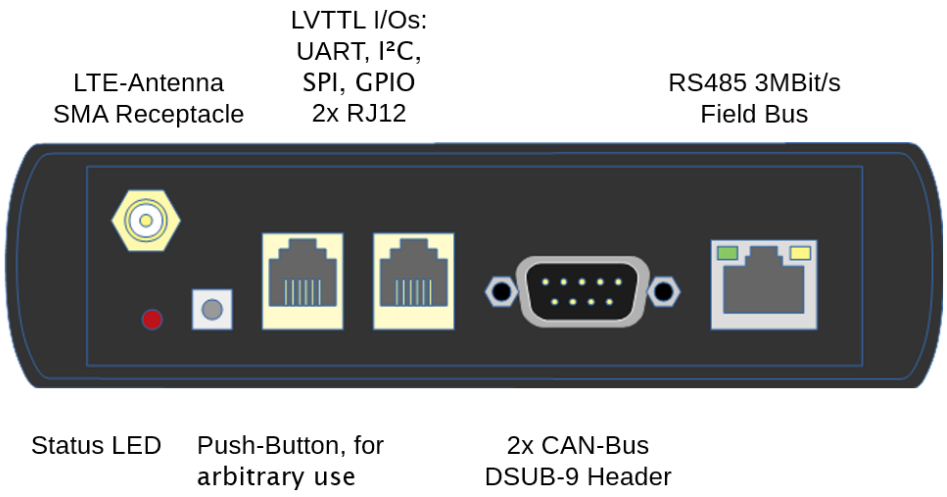

Figure B.3.: Gateway Interfaces 2.

B.3. Details of Block Diagram

The central unit is taskits APPSTACLE project board for the demonstrator. In the following block diagram, it is colored in cyan. The connection to the cloud is done via a Gemalto LTE module. With the separate co-processor Cortex M4 it provides RS 485/RS422 fieldbus with 3 Mbit/s through RJ45. Further more it gives 8 x GPIO and 2 CAN-FD-Transceiver to the users. The optional developer-HMI-touchscreen will be capacitive. Finally one standard Ethernet and three Automotive Ethernet connections offer most variety connect ability to every project partner.

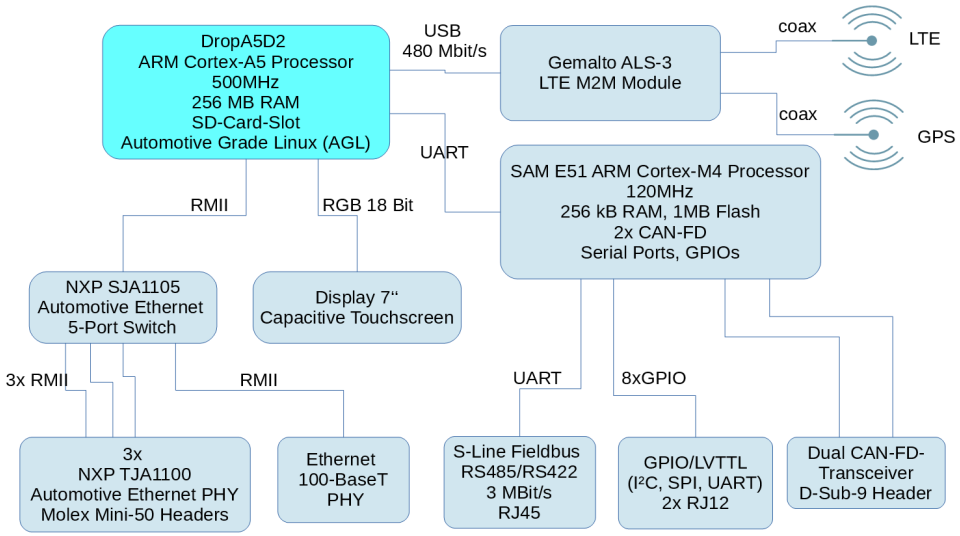

Figure B.4.: Block diagram of the APPSTACLE project board.

B.4. Software and Housing

As the best possible base for APPSTACLE AGL (Automotive Grade Linux) was selected, because a variety of basic functionality is already covered and the community as well as dissemination within the automotive environment is quite large.

taskit has already implemented AGL on the community board (Rasperry Pi) and after initial analysis also assumes that it will be able to run on the CoM of the gateway, since in case of doubt it would also be possible to use light versions of AGL, that run on an ECU. In the end, AGL will run on the APPSTACLE project board.

To make it more handle an aluminium case was chosen to be stable and to offer a safe home to the electronic parts and the screen.

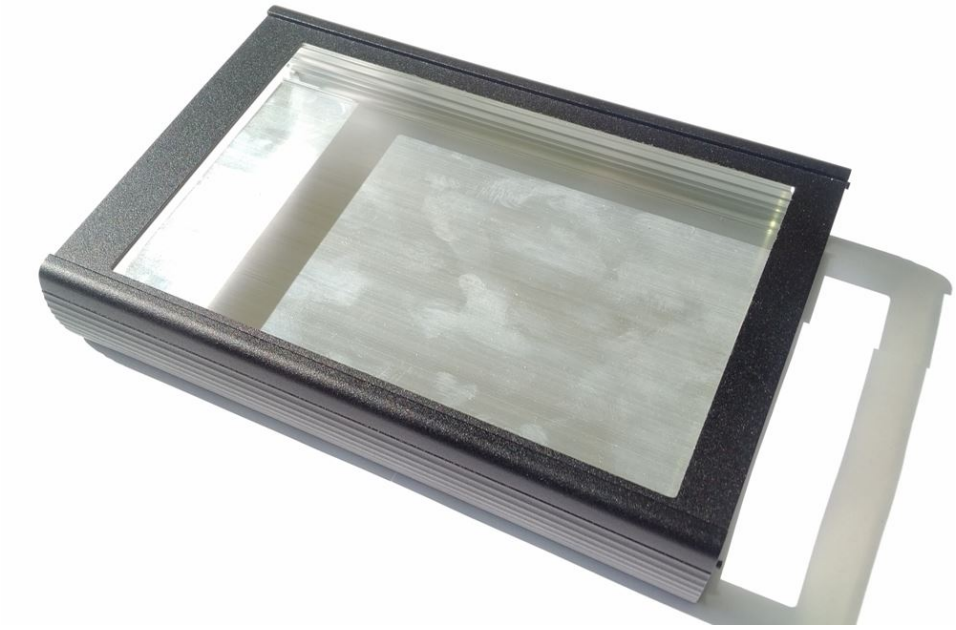

Figure B.5.: Optional Housing for the APPSTACLE project board.

C. Demonstrator for 5G Connectivity

The available or soon available components of the 5G Test Network have been described in APPSTACLE deliverable D2.1 Section 3.3. The components most likely to be used in APPSTACLE Demonstrations are the private 4G Macrocell supporting NB-IoT devices, the 5G proof-of-concept mmWave radio, the Cloud core and mobile edge computing, depending on the demonstrated use case. Figure C.1 represents a high-level picture of the 5G Test Network at University of Oulu.

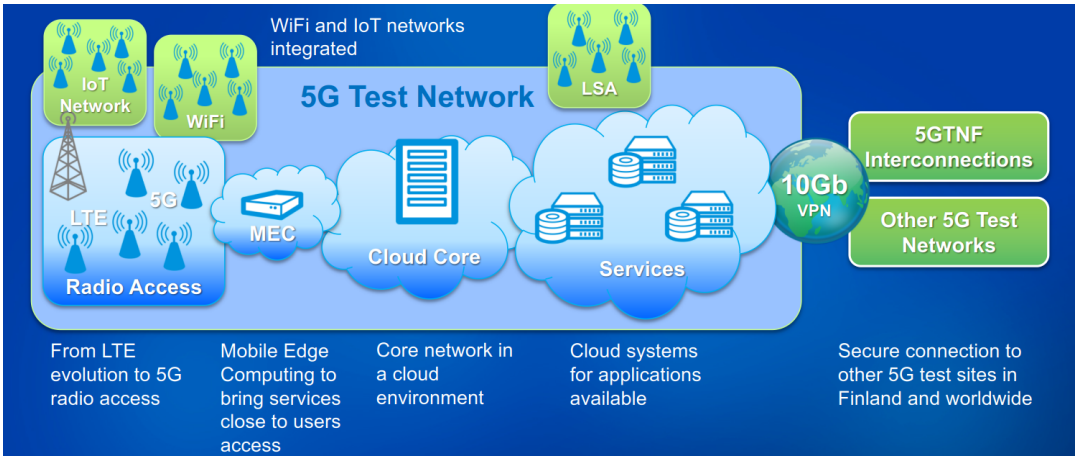

Figure C.1.: 5G Test Network at University of Oulu

Bibliography

[1] APPSTACLE: Deliverable D2.1 - SotA Research with regard to Car2X Communication, Cloud and Network Middleware and corresponding Security Concepts. August 2017. – Research report. Deliverable Report

[2] APPSTACLE: Deliverable D1.1 - Specification of In-car Software Architecture for Car2X. January 2018. – Research report. Deliverable Report

[3] APPSTACLE: Deliverable D3.1 - Specification of Data Management, Cloud Platform Architecture and Features of the Automotive IoT Cloud Platform. January 2018. – Research report. Deliverable Report

[4] Eclipse: *Open source stack for IoT Cloud Platforms*. `https://iot.eclipse.org/cloud/`. 2017. – Accessed: 2017-07-18