

Paving the Way for Connected Cars with Adaptive AUTOSAR and AGL

S. Aust, *Senior Member, IEEE*

Abstract—Connected vehicles offer new business models and improved user experiences. Car manufacturers are aiming to equip their vehicles with software modules and to keep them up-to-date with new services. To ease the process of maintaining the vehicle functions and to offer basic software components over the vehicle's life-cycle the AUTOSAR (AUTomotive Open System ARchitecture) consortium started to standardize basic core functions for vehicle's Electronic Control Units (ECUs) as classic AUTOSAR real-time operating system (RTOS). However, the ever increasing user expectations require that software platforms follow recent developments in software architecture and the AUTOSAR consortium started with the design of an adaptive platform solution that is directed towards future developments. Adaptive AUTOSAR offers software components that will make it easier to deploy a standardized platform for connected cars. A missing component in adaptive AUTOSAR is the specification of a Graphical User Interface (GUI) that is required to design In-vehicle Infotainment (IVI) systems. Automotive Grade Linux (AGL) provides such appealing GUIs and is open source. In this paper, the new concept of adaptive AUTOSAR will be introduced and recent design architectures are outlined. Further, the combination of adaptive AUTOSAR and AGL are motivated to design a proof-of-concept (PoC) that is efficient to use for further performance evaluation of the software components of each software platform.

Index Terms—Connected car, classic AUTOSAR, adaptive AUTOSAR, Automotive Grade Linux, Inter-vehicle Infotainment, RESTful, over-the-air.

I. INTRODUCTION

The AUTOSAR (AUTomotive Open System ARchitecture) is a global standardization consortium that consists of car manufacturers, service providers, and software developers. The AUTOSAR consortium standardizes software platforms for automotive electronic and software systems to increase the compatibility and safety of such software components. To accommodate the need for a highly flexible and adaptive software platform the AUTOSAR consortium started with the development of so-called adaptive AUTOSAR in 2015.

Recent trends in user behavior and user expectations (UX) have indicated a strong need of new graphical user interfaces (GUIs), always connected, and up-to-date. Since the vehicle

has become more software oriented, including an Inter-Vehicle Infotainment (IVI) system, the user demand is increased and high efficient and useful content inside the

Vehicles are expected. Original Equipment Manufacturers (OEMs) and Tier1 have to follow such expectations to be successful in a competitive market and to establish the car as part of a car data value chain and monetization [1]. Automatic software uploads, security patches, and on-line monitoring are becoming a necessity for OEMs, because the user expectations are based on today's smart phone characteristics. In fact, reports have shown that millennials prefer owning a smart phone instead of owning a car which has become much more costly compared with prior decades [2]. Moreover, if new sharing services offer cost-efficient mobility whenever needed, OEMs are facing a declining market share associated with increased user expectations.

To accommodate the vehicle costs during the design and production process, it is beneficial for OEMs to standardize a common software platform. It provides the building blocks of a vehicle's software and can be modified to match different user needs and enable additional benefits [3]. For instance, the basic functions to control an Electronic Control Unit (ECU) can be standardized and used by multiple OEMs.

The AUTOSAR consortium offers such benefits for OEMs to summarize and align the software needs and to enable a common ground for discussion and technology transfer among OEMs. Instead of spending large amounts of man-hours to develop a closed-system, an open design can be more beneficial where developers talk the same language and reduces security vulnerabilities in the onset of the software development process [4], [5]. Moreover, closed systems have the disadvantage to lock-in the OEM in never-ending software maintenance procedures throughout the vehicle's entire life-cycle. Instead, the AUTOSAR real-time operating system (RTOS) is a software platform that is shared and agreed among OEMs, thus is easier to modify the software components.

However, AUTOSAR does not specify higher layer services and GUIs. In contrast, the Linux Foundation started the Automotive Grade Linux (AGL) project that aims to provide open source for automotive services and applications. In particular, the IVI system is in the focus of AGL and can be combined with other automotive platforms. In this study a combination of adaptive AUTOSAR and AGL is proposed to realize a prototype of a connected car software platform.

Stefan Aust is with NEC Communication Systems, Ltd., Kawasaki, Japan (e-mail: stefanaust@ieee.org).

II. AUTOSAR

The AUTOSAR consortium standardizes an automotive open system architecture software planform for car electronic and software systems. It aims to reduce the ever increasing complexity in vehicle software development by standardizing a common software platform. AUTOSAR partners can utilize the platform, modify it, and to design the software based on their needs. The re-use of standardized software components and the integration of third party solutions based on the AUTOSAR standard, make the design process highly cost-efficient. AUTOSAR is a partnership program, consisting of core partner, premium partner, development partner, and associate partner. Core Partners are responsible for defining the main activities of the project. A list of current Core Partners is given [6]. Premium partners are allowed to participate actively in the standardization process, including access to all necessary documentation. Development partners are provided with limited access to development documents, but allowed to participate. Associate partners are allowed to license the automotive platform license-free. There are two main directions in AUTOSAR, *classic AUTOSAR* and *adaptive AUTOSAR* as show as follows:

A. Classic AUTOSAR

Classic AUTOSAR was the first realization of a standardized vehicle platform. The main target was the software system for ECUs. The so-called Basic Software (BSW) is the middleware between the ECU functionalities and higher application layer. Although basic functions to connect the car to an outside communication infrastructure were introduced, the functional concept does not allow a highly flexible software platform. AUTOSAR standardized three software layers, namely the *application layer*, *runtime environment* (RTE), and *basic software* (BSW) [3]. In classic AUTOSAR the application layer handles the communication from application software components to the ECU through the RTE. The RTE acts as an abstraction layer among ECUs and establishes inter-ECU communication via Controller Access Network (CAN), FlexRay, and Media Oriented Systems Transport (MOST).

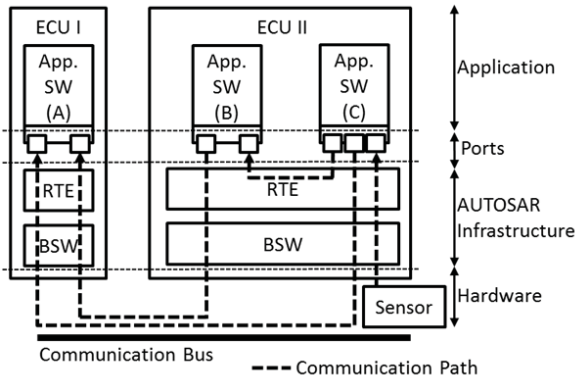


Fig. 1: Schematics on how to connect ECUs with classic AUTOSAR.

To further interact with the ECU the BSW provides the necessary functions to run the ECUs software components and acts as a standardized software layer. In Fig. 1 the classic AUTOSAR system is depicted including multiple ECUs connected via the runtime environment.

B. Adaptive AUTOSAR

The development of adaptive AUTOSAR has become the main driving force in the AUTOSAR consortium in recent years. Various OEMs are joining the standardization process of adaptive AUTOSAR. The consortium is multi-national and includes partners from the U.S., Europe, and Asia. Meetings are held on a regular basis. Additionally, specific *all member meetings* are conducted once per year to report about project results. Further, an AUTOSAR Open Conference (AOC) is held annually in different locations, introducing showcases and proof-of-concept (PoC) systems to the public [7].

In Fig. 2 the adaptive AUTOSAR software architecture is depicted. It shows the functional clusters and its application programming interfaces (APIs). Main clusters include execution management (EMO), persistency (PER), identity and access management (IAM), time sensitive networks (TSN), platform health management (PHM), crypto, update and configuration management (UCM), network management (NM), diagnostics (diag), and signaling (signal). The ara:com interface contains the communication APIs.

AUTOSAR has a detailed specification release plan for adaptive AUTOSAR. The latest release date was scheduled for the end of March 2018, named as *R18-03*. The next release is planned for October 2018, named as *R18-10*. Each release aims to implement new functions. For R18-10 the main focus is to implement the standard work related to automotive software updates. AUTOSAR standardizes the vehicle's software components, but not the software itself. The software implementation is done by the OEMs. PoCs are being developed by AUTOSAR to show the usefulness of the software standard. This is to further increase the acceptance rate of the standardization among OEMs as suggested in [3].

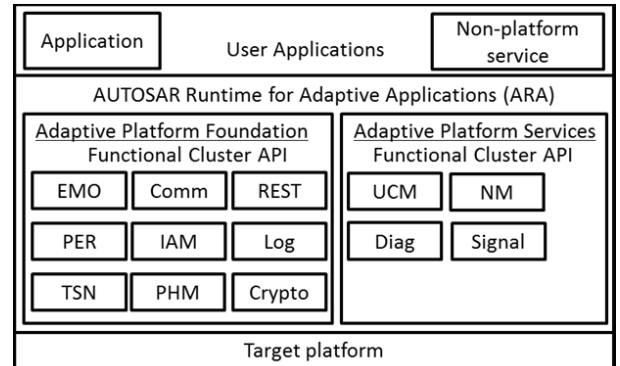


Fig. 2: Adaptive AUTOSAR platform R18-03 with AUTOSAR Runtime for Adaptive Applications (ARA), functional clusters and APIs.

C. Working groups in AUTOSAR

AUTOSAR is comprised of several working groups that are targeted to a specific automotive software system component. Working groups include a *Lead Work Package* (Lead WP), a *Work Package* (WP), and *Feature Teams* (FTs). The Lead WP builds the backbone of the entire architecture group and is supervising the development process. The WP consists of *software architecture, methodology and templates*, and *application interface*. Of interest are the so-called FTs, in which recent developments are taking place, including communication management (FT-CM), execution management (FT-EMO), diagnostics (FT-DIAG), methodology & manifest (FT-MN), persistency (FT-PER), and update & configuration management (FT-UCM). In particular, the FT-UCM has gained some attention recently, because the team members are discussing software update functions and over-the-air (OTA) strategies for the vehicle-wide software update process of multiple ECUs.

Further, feature teams exist for safety (FT-SAF), security (FT-SEC), demonstrator integration (FT-DI), system tests (FT-ST), and automated driving interfaces (FT-ADI). FT-ADI was founded in 2017 to address the needs for a standardized interface for vehicle sensors. A complete list of working groups is given in [8].

The philosophy behind the FT's is mainly to support the so-called V-model of a verified software implementation process. The V-model is widely accepted in the automotive industry to define the software process for efficient product development. The model comprises the project definition and defines the project test and integration procedure. Verification and validation is part of a feedback process to improve the project definition when needed (executed by the FTs).

III. AUTOMOTIVE GRADE LINUX

AGL is a *code-first* software project within the Linux Foundation. Several AGL members were engaged in the GENIVI automotive software project [9] and experienced a significant implementation delay based on postponed specifications. Thus, the AGL community was motivated to ship their software platform at a higher speed and utilized Tizen OS in the beginning [10]. AGL specifies several software modules and provides a basic software kernel based on Linux OS. Software modules include communication, web services, graphics, navigation, and automotive services. The latest AGL Unified Code Base (UCB) release 5.0, named as *Electric Eel*, was in January 2018 [11], and is based on Yocto 2.3 (Pyro branch) which is widely used for embedded Linux systems [12]. The release includes a Software Development Kit (SDK) and addresses all vehicle's software including, infotainment, instrument cluster, telematics, heads-up display (HUD), advanced driver assistant systems (ADAS), functional safety, and autonomous driving. In Fig. 3 the software components of AGL are depicted.

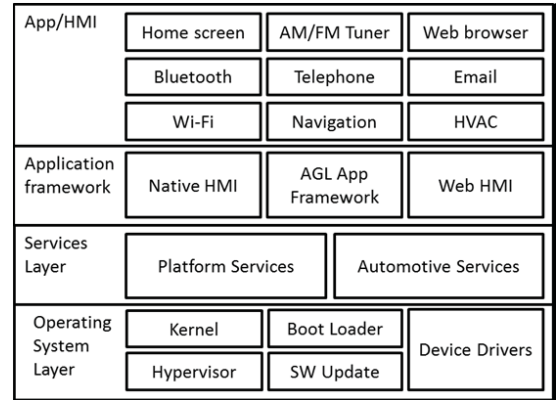


Fig. 3: The AGL software platform, consisting of Application Layer and Human Machine Interface (HMI), Application Framework, Services Layer, and Operating System Layer, containing the required device drivers [13].

It shows the Application Layer that contains the software modules to interact with the vehicle's driver, including web browser, Wi-Fi, Bluetooth, etc. Device drivers and kernel are provided in the Operating System Layer. AGL supports QT to display dashboard information with reduced delay [13].

The IVI system is the central point of information of a connected car. The vehicle's driver utilizes the IVI to control services and functions of the car. Further, the IVI provides services to support the route planning and to inform of road-side services. The Automotive Grade Linux (AGL) foundation is focusing on standardizing such open source IVI. Further, APIs to the car's software are considered and web services are an integral part of AGL. AGL can be setup to display car data and road-side information and map services.

Note that both projects, AUTOSAR and AGL, are utilizing similar on-line software development tools for Continuous Integration (CI) and Automatic Testing (AT). Whereas in AUTOSAR assigned FTs exist to verify specification content and prototype implementations, software developers in AGL check their own implementations (future AGL releases may include cross-check validations).

IV. THE CONNECTED CAR

The realization of a connected car demands a high degree of digitalization thus, requires more software components compared to a standard vehicle. Multiple software modules are involved in the driving process, including steering, vehicle network, switches, cameras, IVI, drive recording, ADAS, and autonomous driving functions. The increased use of software inside a car adds up to 100 million lines of codes (LoC) compared to a plane with only 6 million LoCs. Further, the monetary costs of a today's vehicle are mostly dominated by its software costs which already accounts for 40% [14]. Active road safety applications, including intersection collision warning, overtaking vehicle warning, and pre-crash sensing, require such high level of software integration.

A. In-car networks & systems

The in-car networks can be summarized as standard and legacy systems, including CAN, FlexRay, and MOST. Recently there has been some effort to standardize vehicle networks for higher bandwidth and low-latency demands, e.g., to support ADAS and other vehicle's life-critical communication. Such new networks include automotive Ethernet, audio-video bridging (AVB) and time-sensitive networks (TSN) [15]. Adaptive AUTOSAR has started some activities to integrate TSN functions to the software platform.

Next, in-car services can be realized with different software platforms to match different user expectations. For instance, the IVI can be integrated as Automotive Grade Linux (AGL) solution, or as Tizen OS. It can also be realized as a complete solution using Linux-GENIVI. IVI systems can support wireless communication via Wi-Fi or Bluetooth to connect the user's mobile devices, such as smart phones, to the vehicle via smart link connections.

B. Out-car networks

The connected car is attached to cloud services via wireless carrier systems, including LTE, 5G, IEEE 802.11 (Wi-Fi), and IEEE 1609.x (WAVE) protocols [4]. Adaptive AUTOSAR and AGL offer basic software modules to setup communication links via RESTful API to realize a connected car. A schema of a connected car is shown in Fig. 4. Because classic AUTOSAR components have been widely installed, a vehicle may be comprised of both, classic and adaptive AUTOSAR software modules.

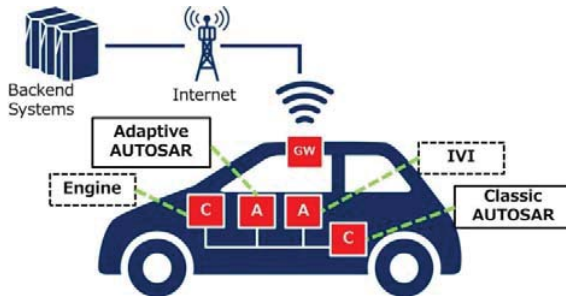


Fig. 4: The connected car utilizing classic AUTOSAR and adaptive AUTOSAR software components.

For instance, the engine control can be managed by a classic AUTOSAR ECU whereas the IVI is realized as an adaptive AUTOSAR ECU. An ECU for accessing the back-end cloud services via mobile carrier systems can be realized as a central gateway (GW) located inside the car. Vehicle's data and sensing information can be sent to services in the cloud. Additionally, control functions or maintenance can be started from the cloud and can be forwarded to the vehicle.

C. Over-the-air software updates

The connected car requires a frequent updates of the installed software modules to minimize the risk of cyber-attacks [16], [17]. Both, AUTOSAR and AGL, have active

development teams working on OTA software solutions to keep the vehicle's software up-to-date. OSTree has been proposed in AGL for software updates [18]. OSTree enables version updates of Linux-based systems and performs atomic software upgrades of entire filesystems and kernel updates. Note that the *UPTANE framework* considers several helpful vehicle service requirements and provides security insights issued by the National Highway Traffic Safety Administration (NHTSA) [19].

In contrast, adaptive AUTOSAR specifies a novel automotive software update technology named as *Update and Configuration Manager* (UCM) which is specified within the FT-UCM group. UCM supports a connection to a back-end server (out-car communication). UCM consists of a basic Linux package manager, data encoder, and software versioning function. A candidate for a software update backend server is *hawkBit* (TM) and can be integrated into Eclipse [20]. Future UCM version will include a vehicle-wide software update process, named as *Vehicle Update Manager* (VUM) and advanced software rollback solutions.

V. EMBODIMENT OF ADAPTIVE AUTOSAR AND AGL

Adaptive AUTOSAR provides the software platform to connect the vehicle's ECUs to outside road services. However, to show the significance of adaptive AUTOSAR a GUI is needed, which is not provided by adaptive AUTOSAR. In contrast, AGL provides a useful GUI implementation which is easy to implement. Thus a combination of adaptive AUTOSAR and AGL is beneficial. Moreover, it provides the platform to further investigate the functionality of the standardized functional clusters in adaptive AUTOSAR. For instance, it is of high importance to attach a wide range of sensors, including cameras, radar, lidar, pressure, temperature, speed, acceleration, etc., which are used by different car services.

The setup of adaptive AUTOSAR and AGL is proposed in this study to create a PoC that enables adaptive AUTOSAR prototype implementation using the hardware target platform. The target platform is based on two different processor architectures, namely the open source hardware *Minnowboard* [21] and the specialized automotive embedded reference board *Renesas R-Car H3* [22]. Adaptive AUTOSAR enables both platforms independently to proof their prototype implementation for different automotive single on chip (SoC) solutions and architectures, e.g., Intel, ARM 32, and ARM 64. Support for QEMU to run the same platform within a Linux virtual box is also supported.

Next, AGL can run on the same target platforms and system kernels can be compiled for different architectures. In Fig. 5 the schematics of the proposed prototype are displayed. A touch-pad is used to utilize the vehicle driver's touch sensing information. The touch-pad is connected via HDMI with the target board. Further, touch sensing information is sent from the touch-pad via USB to the AGL platform.

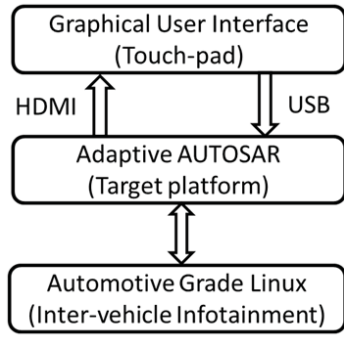


Fig. 5: Schematics of the proposed prototype running adaptive AUTOSAR and AGL on a target platform.

Next, the target platform runs adaptive AUTOSAR to provide the necessary communication protocols, which are used to connect the vehicle to out-car services via Wi-Fi (IEEE 802.11n). Information is transferred among the ECUs using adaptive AUTOSAR. Data is displayed on the touch-pad and the driver can interact with the system by using it.

In Fig. 6 the Intel Minnowboard is shown which is used for the PoC. It runs AGL 5.0 and connects to a touch-pad for user interaction. The platform is connected via USB to the touch-pad.

The sequence chart of the PoC is presented in Fig. 7. The sequence chart indicates the communication starting from the cloud indicating a new software version which is available for download. The back-end server (hawkBit) utilizes the REST protocol (HTTP) for communication with the ECU1. ECU1 runs adaptive AUTOSAR and informs the user via the connected touch-pad that is supported by ECU2 running AGL 5.0. ECU1 and ECU2 are connected via Ethernet as in-car network configuration. After acknowledging, the download process starts and is acknowledged by the user when finished.

In Fig. 8 the PoC is displayed running adaptive AUTOSAR and AGL. Further, it shows the used touch-pad for user interaction. The adaptive AUTOSAR software and AGL run on two ECUs.

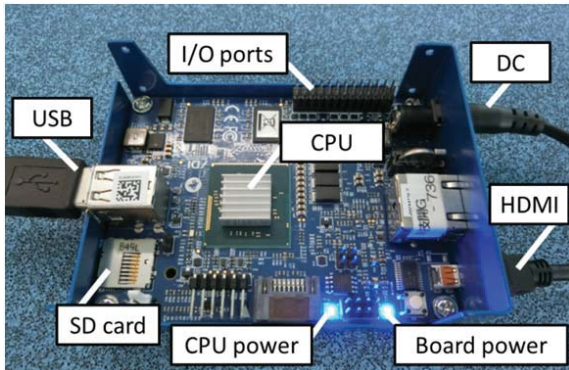


Fig. 6: Illustration of the implemented prototype as part of a PoC realization.

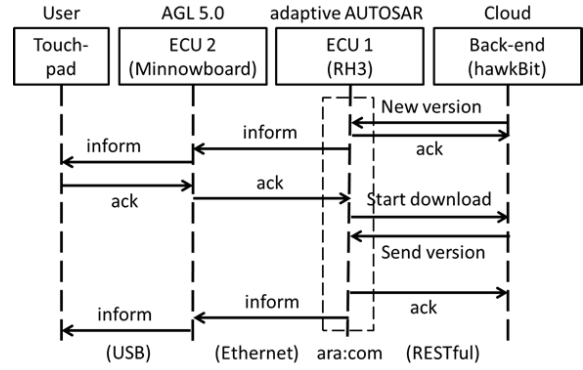


Fig. 7: Illustration of the sequence chart of the proposed PoC.

AGL runs on the Minnowboard, whereas adaptive AUTOSAR runs on the RH3 which provides more active threads.

The PoC is allowing investigating the software functional clusters, such as communication, logging, and persistency. Car data can be displayed and analyzed for delay requirements. For instance, ADAS require ultra-low latency transmission. Such delay measurements can be conducted with the help of this POC. Additionally, software updates can be initiated to verify software versions and the upload process. The POC has proven to be efficient to satisfy such evaluation tasks of adaptive AUTOSAR and AGL.



Fig. 8: Illustration of the implemented prototype as part of a PoC realization.

Finally, findings on the inter-ECU communication are presented. The communication performance of the two ECUs was evaluated. Of interest is the transmission delay when both, RH3 and Minnowboard are used as in-car network configuration, because these boards include two relevant CPU architectures used by OEMs. This is of importance, because the implemented software should proof its effectiveness. The intra-delay was measured with the *Linux Trace Toolkit: next generation* (LTTng) software toolkit to measure various network applications [23]. The results of the inter ECU communication latency measurement are shown in Table 1.

Latency	RH3-Minnow (R17-10)	Minnow-Minnow (R17-10)
Average [μ s]	4455.37	978.51
Standard Deviation [μ s]	1720.06	259.13

Table 1. Measured inter ECU latency (RH3 and Minnowboard)..

The results in Table 1 indicate a significant delay when RH3 is applied to the network. The Minnowboard showed a better communication performance compared to RH3. Since the CPU load of both RH3 and Minnowboard was not fully applied, it is suggested that the difference in the Ethernet IC contributes to the significant delay. Additionally, difference in the kernel configuration were identified which affects the network performance. Furthermore, differences in the adaptive AUTOSAR software releases (R17-03 vs R17-10) were found to have a significant influence of the network performance, indicating an improvement of 59% for RH3-Minnowboard latency performance when using the latest AUTOSAR version. Future work will include further measurements at a wider network scale with up to 10 ECUs.

VI. CONCLUSIONS

To support a common software platform that provides standardized interfaces the AUTOSAR consortium started the development of adaptive AUTOSAR. It standardizes functional clusters, including persistency, security, update configuration, logging, and communication. The cluster architecture is beneficial because they are based on common technologies and compatibility. If car manufacturers have to provide different functionalities, modifications can be applied. It reduces the challenges faced with closed software systems which are difficult to maintain over a vehicle's entire life-cycle. The adaptive AUTOSAR real-time operating system (RTOS) does not provide a Graphical User Interface (GUI). In contrast, Automotive Grade Linux (AGL) offers an open source in-vehicle infotainment (IVI) implementation. Thus, a combination of Automotive Grade Linux (AGL) with adaptive AUTOSAR was proposed and implemented as a proof-of-concept (PoC). The PoC verified that a combination of adaptive AUTOSAR and AGL can be realized. The AUTOSAR consortium may consider the advanced AGL IVI

features to provide a GUI. The benefit is that AUTOSAR does not need to create a new GUI. Instead, the AUTOSAR software specification activities can focus on new networking trends, including time sensitive networks and secure software update over-the-air.

VII. REFERENCES

- [1] McKinsey & Company, "Car data: paving the way to value-creating mobility – Perspectives on a new automotive business model," in *Advanced Industries*, 2016.
- [2] Millennials Would Rather Ditch Car Than Smartphone or Computer: Zipcar Survey. (2018, May). [Online]. Available: https://www.huffingtonpost.com/2013/03/01/millennials-car-ownership_n_2789454.html
- [3] S. Martinez-Fernandez, C. P. Ayala, and X. Franch, "A Survey on the Benefits and Drawbacks of AUTOSAR," *IEEE WASA '15*, pp. 19–26, May 2015.
- [4] Experimental Security Assessment of BMW Cars: A Summary Report (2018, May). Homepage. [Online]. Available: https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf.
- [5] M. Harris. (2015, Sept.). Researcher Hacks Self-driving Car Sensors—\$60 Lidar spoofing device generates fake cars, pedestrians and walls. *IEEE Spectrum*, [Online]. Available: <https://spectrum.ieee.org/cars-that-think/transportation/self-driving/researcher-hacks-selfdriving-car-sensors>.
- [6] AUTOSAR (2018, May). Core Partners. [Online]. Available: <https://www.autosar.org/about/current-partners/core-partners/>
- [7] AUTOSAR (2018, May). Homepage. [Online]. Available: <https://www.autosar.org/news-events/details/call-for-papers-11th-autosar-open-conference-2018-apr-18/>
- [8] AUTOSAR (2018, May). Working Groups. [Online]. Available: <https://www.autosar.org/working-groups/>
- [9] M. Aichouch, J.-C. Prevotet, F. Nouvel, "Evaluation of the overheads and latencies of a virtualized RTOS", in *Int. IEEE Symposium on Industrial Embedded Systems (SIES)*, 2013, pp.1–5.
- [10] Tizen In-Vehicle Infotainment (2018, May). Homepage. [Online]. Available: <https://www.tizen.org/about/devices/vehicle-infotainment>.
- [11] Automotive Grade Linux (2018, May). Software. [Online]. Available: <https://www.automotivelinux.org/software>.
- [12] Yocto Project (2018, May). Main Page. [Online]. Available: https://wiki.yoctoproject.org/wiki/Main_Page.
- [13] Automotive Grade Linux, "Automotive Grade Linux Requirements Specification," Version 1.0, May 28, 2015.
- [14] R. N. Charette. (2009, Feb.). This Car Runs on Code. *IEEE Spectrum*. [Online]. Available: <https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code/>.
- [15] Time Sensitive Networking Task Group. (2018, May). Homepage. [Online]. Available: <http://www.ieee802.org/1/pages/tsn.html>
- [16] W. Zeng, A. S. Khalid, and S. Chowdhury, "In-Vehicle Networks Outlook: Achievements and Challenges," in *Int. IEEE Communications Surveys & Tutorials*, vol. 18, No. 3, third quarter, 2016, pp.1552–1571.
- [17] M. Berret, F. Mogge, M. Bodewig, E. Fellhauer, C. Sondermann, and M. Schmidt, "Global Automotive Supplier Study 2018—Transformation in light of automotive disruption," Roland Berger and Lazard Automotive teams, Dec. 2017.
- [18] OSTree (2018, May). Introduction. [Online]. Available: <https://ostree.readthedocs.io/en/latest/manual/introduction/>
- [19] T. K. Kuppusamy, L. A. DeLong, and J. Capps, "UPTANE—Security and Customizability of Software Updates for Vehicles," *IEEE Vehicular Technology Magazine*, pp. 66–73, March 2018.
- [20] Eclipse hawkBit (TM) (2018, May). IoT. Update. Device. [Online]. Available: <https://www.eclipse.org/hawkbit/>
- [21] Minnowboard. Open source hardware for small spaces. (2018, May). Homepage. [Online]. Available: <https://minnowboard.org/>
- [22] Renesas R-Car H3 (2018, May). Homepage. [Online]. Available: <https://en.wikichip.org/wiki/renesas/r-car/h3>
- [23] LTTng (2018, May). The LTTng Documentation. [Online]. Available: <https://lttng.org/docs/v2.10/>