Proceedings of the Fifth International Conference on Inventive Computation Technologies (ICICT-2020)
IEEE Xplore Part Number:CFP20F70-ART; ISBN:978-1-7281-4685-0

Automotive Grade Linux Software Architecture for Automotive Infotainment System

Dr. P. Sivakumar*[1], Ms. R.S. Sandhya Devi [2], Neeraja Lakshmi A[1]
Dr.B. VinothKumar[3], Dr. B. Vinod[1],
[1]Department of EEE, PSG College of Technology, Coimbatore, India,
[2]Department of EEE, Kumaraguru College of Technology, Coimbatore, India,
[3]Department of IT, PSG College of Technology, Coimbatore, India,
Email: psk.eee@psgtech.ac.in*[1] (corresponding author), sandhyadevi.rs.eee@kct.ac.in[2],
neerajakumar.20@gmail.com[1], bvk.it@psgtech.ac.in[3], bvin@mail.psgtech.ac.in[1]

Abstract—A rapid increase in the development of electric and connected vehicles has led to a complete revolution in the automotive industry; thus new solutions are needed to meet the rapid innovation in the entire industry while reducing the time to market and engineering costs. The main objective of this paper is to provide an open-source solution for Automotive In-Vehicle Infotainment to create a fast-innovating ecosystem and to reduce software time to market. Automotive Grade Linux (AGL) is a shared platform mainly used for In-Vehicle Infotainment systems (IVI), used by the automakers and suppliers to reduce fragmentation and to reuse the same code base, thus leading to rapid innovation and faster time-to-market for new products.

Keywords—In-Vehicle Infotainment, Linux Foundation, AGL

I. INTRODUCTION

In-Vehicle Infotainment (IVI) is a term referring to a device that integrates the distribution of vehicle information with passenger and driver entertainment. The IVI system comes with the Human Machine Interface (HMI) that consists of audio and video interface, touch screen, keypad, etc. to manage these services. With the aid of the HMI, it is also possible to adjust multimedia services. Additionally, the accessibility of mobile devices to the car has become a major component of the IVI network in an effort to meet the growing need for internet access and general smartphone content. The IVI network nowadays has upgraded security features designed to avoid interference of drivers [7]. When consumer electronics advances in mobile devices and integrated phones, consumers continue to demand such features to be in the car as well. Smartphones and tablets have therefore changed customer expectations for infotainment systems in the vehicle.

The main challenges faced by the automakers are to provide an automotive system that satisfies the user requirements, reducing the time to market and manufacturing costs. To overcome these challenges, the automotive industry should rely on software and hardware solutions that guarantee open standards, flexibility, interoperability, safety and security. Among the automobile systems, the In-Vehicle Infotainment plays a major role in defining customer's expectations. The software developed for modern vehicles need up to 100 million lines of code across all their various systems. While comparing this to Smartphone operating systems such as Android, it operates on 12 to15 million lines of code with more advanced features. In addition, Developing the latest in-vehicle infotainment system takes 3 to 4 years, during which time a smartphone can usually deliver 3 to 4 updates to customers. Thus a customer expects to have an IVI system which has the same features as their Smartphone. Thus automakers would like to push in-vehicle infotainment operating systems growth to about the same pace as mobile and tablet production within 12 months.

Most smartphones and tablets running on Android, iOS, or KaiOS have significant backward compatibility between versions of the same operating system. Nevertheless, each automaker uses Ubuntu, QNX, or Windows Embedded Automotive specific models. Just as manufacturers seek to standardize where they can, not only will each automaker use each operating system's modified model, but every vehicle line within each automaker also uses different custom operating systems from different suppliers. Thus a standard platform is not being followed for the development of IVI systems. This is one of the major issues faced by the automakers for implementing connected car concept.

The manufacturers needed to develop their own architecture for in-vehicle control systems without focusing on more software-centric firms that usually develop operating systems such as Apple and Google to separate themselves from the other rivals. Although this is a valid concern, drivers seldom see many car parts, just as drivers hardly communicate with many operating systems. The user interface is the key competitive differentiator as long as the inherent speed, reliability and deployment capability are

met. For the automakers, a properly operating system is a must, which is also necessary to be an open-source approach. Thus the OEMs find it beneficial to have a standard software platform to accommodate the vehicle costs during the design and manufacturing process.

The main aim of this paper is to address the development time, fragmentation, and lack of code reuse and other factors faced by the Automakers. Automotive Grade Linux (AGL) from The Linux Foundation hopes to serve as the de facto industry standard in-vehicle infotainment operating system, before addressing all the software needs of vehicles, including advanced driver assistance systems, autonomous driving, telematics, and other in-vehicle displays, such as instrument clusters and heads up displays.

II. EXISTING SYSTEMS

The benefits and drawbacks of various networks, and identifies all the attainable techniques to boost the Quality of service (QoS). Additionally, two classifications of automotive gateways are given together with a short discussion regarding constructing a comprehensive in-vehicle communication system with completely different networks and automotive gateways [1].

The proof-of-concept design, whose main contribution is an automotive-oriented extension of Google Android that has options for combining extendibility and safety necessities, It includes Google Android as a third party application extension. This paper provides the result that, unworthy applications cannot access vehicle functionalities to operate a vehicle in a secure manner [2].

Certain Open technologies, like HTML5 and JavaScript have been discussed in paper [3] for IVI application development. It utilizes WebGL for advanced graphical effects. A Java-based framework was created to permit movability of cluster and IVI applications to the planned application environment.

The integration of applications with an in-vehicle infotainment system using different consumer electronic devices. These applications are integrated into the user interface while minimizing the development cost. The control elements correlate in Volkswagen and Audi systems with the control buttons in the lower part of the display or, respectively, the control buttons on the edges [4].

The AUTOSAR (AUTomotive Open System ARchitecture) consortium [5] [10], which provides a standard core functions for vehicle electronic control units (ECUs) [15][19]. It also discusses Adaptive AUTOSAR that provides software components to make the deployment of a standardized platform easier for connected cars [18] [16].

The identification of drivers with the help of the finger print identification model is discussed in this paper [17]. This is mainly used in connected cars for enhancing driver profiling and car safety. The model is based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (Long Short-Term Memory) RNN / LSTM which includes data collected from smartphone sensors [6].

III. PROPOSED SYSTEM

A. AUTOMOTIVE GRADE LINUX (AGL)

The Automotive Grade Linux (AGL) was launched in 2012 by the Linux Foundation. It was developed as an open-source project to help the automakers rely on Linux based software platform. It was originally designed for IVI systems, later it emerged as a platform to support instrument clusters, telematics, HMI, etc.

The main concept is that the AGL software will act as a reference framework that can be supported and used by OEMs and suppliers to manufacture their own commercial product by integrating their own innovations. This leads to the development of a personalized and custom user interface on top of it [9]. AGL's important goal is to develop and implement a fully open source IVI vehicle software platform at a rapid pace. To do this, AGL aims to bring the automakers and automotive suppliers in a single thought to develop and rely on a common platform that provides full user interface access to OEMs [9]. Similar to the approach of GENIVI, AGL forms a base platform for OEM, while developing some innovations on top of it, rather than spending human efforts and resources on developing separate, individual solutions [11].

B. TECHNOLOGY

The most important concern in developing a new product is the reuse of existing work and economizing capital. The AGL provides a Unified Code Base (UCB) that includes an operating system, middleware components and application framework. This provides 70-80% of the starting point for the production of the IVI system. Thus automakers and suppliers can concentrate on customizing remaining 20-30% of the system based on customer requirements. AGL is designed to work with different platforms for applications like telematics, HMI, instrument clusters, etc. or it may start the development from scratch [13]. The AGL platform's basic building blocks are quite similar to other embedded Linux architectures, as the same kernel and many of the same middleware and open source modules can be reused.

The AGL platform is compliant with the GENIVI compliance specification to some extent as both were developed by the Linux Foundation [14]. Since AGL IVI platform is based on Tizen IVI, an operating system profile originally created for mobile and embedded devices, adding additional middleware components and user interface to it, builds on an extensively tested and functional OS architecture. In addition, most of AGL's inventions are returned upstream to Tizen [12].

C. AGL SOFTWARE ARCHITECTURE

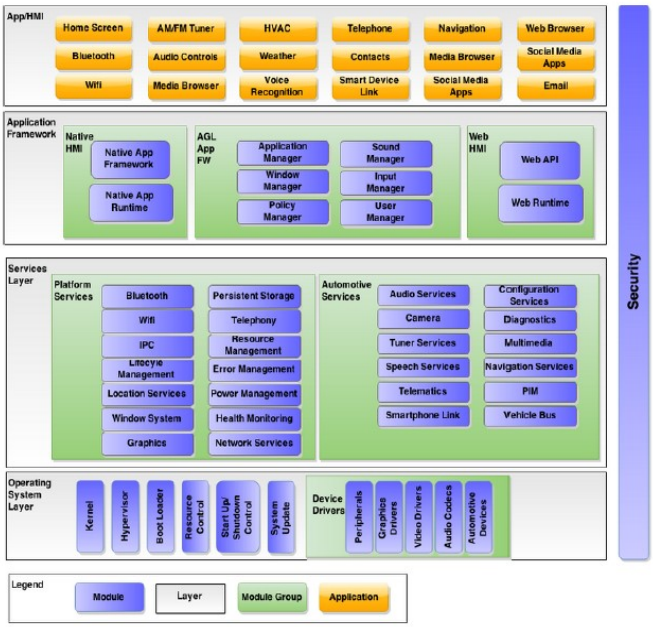

Fig. 1 AGL software architecture

The software architecture of AGL can be represented with four layers, starting with an application and an HMI layer at the top of the stack, followed by a system layer that provides APIs for application development and communication. Below is a service layer that contains all software available user-space resources. At the bottom of the stack, is an operating system layer that includes the kernel, different device drivers, and common operating system utilities [15]. The user interface and functionality of the AGL framework are made entirely in JavaScript and HTML5, and the platform communicates with the car via an Automotive Message Broker (AMB) using Tizen IVI web runtime to allow applications to transmit data to and from the vehicle. Some of the features currently included in AGL's user experience are Home Screen, Dashboard, Heating Ventilation and Air Conditioning (HVAC) system, Google Maps, Internet & News Service, etc.

IV. SYSTEM IMPLEMENTATION

A. SYSTEM BLOCK DIAGRAM

Figure 2 shows the overall system implementation as a Block Diagram. The development of an Infotainment system using AGL involves building an AGL image on the target platform. The image can be built using a suitable Software Development Kit (SDK). The target platform must be configured to boot the image. After the image is booted on the target, the applications can be created and deployed on the hardware. With the help of a touch screen or monitor connected to the target board, the AGL image can be viewed and used In-vehicle infotainment system.

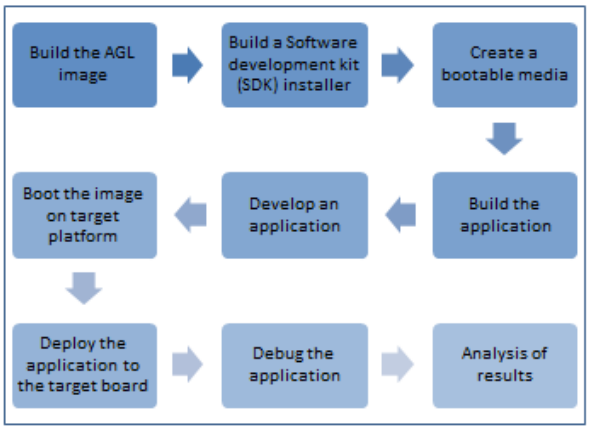

Fig. 2 System block diagram

A. SOFTWARE REQUIREMENTS

It is recommended that the build host is a native Linux machine running a distribution sponsored by Yocto Project to use the AGL program. Essentially, AGL image is supported by a recent version of Ubuntu, Fedora, openSUSE, CentOS, or Debian. The AGL image development requires at least 50GB of memory to be available in the build host. Thus a native Linux platform is most suited to download the source files and build the image. If the build host is not a native Linux computer, a Docker can be used to construct a container that allows us to function as if we're using a host-based on Linux. The container has the same development environment as a properly prepared build host running a licensed Linux distribution (i.e. distros, packages, and so on). Thus a Docker image can be used to eliminate the host dependency issues.

B. HARDWARE REQUIREMENTS

On various hardware platforms including Intel 64-bit hardware boards, Renesas R3 development kit, Minnow board, Raspberry pi 2/3, etc., AGL image can be supported. In this paper, Raspberry Pi3 is used to construct the AGL image as a target platform.

C. DEVELOPMENT OF AGL IMAGE

The AGL image creation process consists of setting up the system (i.e. the construct host) that creates the image and finishes by using the Yocto Project to create a specific hardware-oriented image. Once the build host has been determined, by downloading the AGL source files, it can generate an AGL image. The new stable release version of AGL or the "cutting-edge" (i.e. "master" branch) files can be used depending on the development goals.

Building the AGL image

The part of the AGL software downloaded is a setup script to initialize the built environment. The script accepts several options that allow us to define parameters

like target hardware (i.e. the machine), create a directory, etc. An AGL function is a configuration that takes into accounts the different settings and dependencies necessary for a specific project. Specifying the "agl-demo" function, for example, ensures that the aglsetup.sh script generates configuration files that the AGL demo needs to build the image. Creating the AGL image involves running a defined target for BitBake. In general, there are many different ways to create and construct an application. The Standard Software Development Kit (SDK) is an important key to designing an application that is suitable for target hardware.

V. HARDWARE SETUP FOR AGL

A. BUILDING THE AGL IMAGE FOR RASPBERRY PI 3

The Raspberry Pi is a small computer suitable for computer language learning. Some options must be specified when running the script files to build the AGL prototype image for a Raspberry Pi board. The AGL image can be built using the following command

$ bitbake agl-demo-platform

It must be deployed to the target board after the image has been created (i.e. Raspberry pi 3). Deploying the prototype image of the AGL consists of copying the image to a MicroSD file, plugging the card into the board of the Raspberry Pi, and booting it. The entire hardware setup is shown in Figure 3.

Fig. 3 Hardware setup for AGL image development

B. OUTPUT

The home screen will be shown on the Pi-configured monitor or touchscreen after booting the AGL image to Raspberry pi 3. The AGL Home screen is intended for use with touch presses; the HMI guide provides access to all pre-installed AGL test applications as well as access to user-installed applications in the future. The sample feature list includes some vehicle apps such as HVAC-control, Navigation or Dashboard, as well as some infotainment applications such as television, digital, etc. Figure 4 and Figure 5 shows the home screen menus and dashboard obtained from the AGL image.

Fig. 4 Homscreen options in AGL

Fig. 5 Dashboard options in AGL

C. SIMULATION RESULTS

The different releases of AGL Unified Code Base over the years have integrated some added features for the infotainment system. Some of them include ready-to-produce audio solutions, Vehicle to cloud infrastructure, Voice recognition services, virtualization, etc. The AGL framework tends to provide fast and secure boot, software update, and Diagnostic log and trace facilities, etc. The AGL virtualization approach helps to provide better performance and power consumption. The study shows that the virtualized ECUs have a faster boot execution time, for example, it takes less than 5 seconds for instrument cluster applications and 10 seconds for IVI systems. Thus it helps to manage the execution priorities to ensure that the system is always available for urgent needs.

VI. CONCLUSION

Automotive Grade Linux is still under production, achieving its first commercial launch in the 2018 Toyota Camry system of a major original vehicle manufacturer of equipment. No other automobile operating systems have the supporting development targets to function as a de facto industry standard for in-vehicle infotainment marketing. By reducing product complexity and reusing proprietary elements of the code base of the operating system, the manufacturers easily incorporate AGL and all other technologies in their infotainment. AGL platform also improves the price and performance value by eliminating the license cost for the automakers. Thus the overall lifecycle costs of the infotainment system in a vehicle are considerably reduced. In future, AGL software is expected to be developed for other functions like telematics, instrument cluster, etc. in an automobile, apart from the IVI systems. This helps in providing a completely open standard solution for connected vehicles. With Linux as a base for AGL platform, a high level of security is guaranteed for the IVI system, thus minimizing the vulnerabilities and attacks during a software upgrade. Thus AGL helps in building a promising technology for automobiles and connected vehicles.

REFERENCES

[1] 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.

[2] G. Macario, M. Torchiano and M. Violante, "An in-vehicle infotainment software architecture based on google android," *2009 IEEE International Symposium on Industrial Embedded Systems*, Lausanne, 2009, pp. 257-260.

[3] R. Ostojic, J. Pesic, M. Z. Bjelica and G. Stupar, "Java-based graphical user interface framework for In-Vehicle Infotainment units with WebGL support," *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Berlin, 2016, pp. 184-186.

[4] F. Hueger, "Platform independent applications for in-vehicle infotainment systems via integration of CE devices," *2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Berlin, 2012, pp. 221-222.

[5] S. Aust, "Paving the Way for Connected Cars with Adaptive AUTOSAR and AGL," *2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops)*, Chicago, IL, USA, 2018, pp. 53-58.

[6] E. Mekki, A. Bouhoute and I. Berrada, "Improving Driver Identification for the Next-Generation of In-Vehicle Software Systems," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7406-7415, Aug. 2019.

[7] In-Vehicle Infotainment (IVI). Techopedia. Accessed:2020-01-20. [Online]. Available: http://www.techopedia.com/de_nition/27778/in-vehicle-infotainment-ivi

[8] R. N. Charette. (2009). This Car Runs on Code. IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code/.

[9] Melissa Logan. JVC Kenwood, Linaro, and Opensynergy Join Automotive Grade Linux. Automotive Grade Linux (AGL). Accessed: 2020-01-20. [Online]. Available: https://www.automotivelinux.org/news/announcement/2014/11/jvc-kenwood-linaro-and-opensynergy-join-automotive-grade-linux

[10] P.Sivakumar, B.Vinod, R.S.Sandhya Devi, S.Poorani Nithilavallee(2015). Model Based Design Approach In Automotive Software and Systems. *International Journal of Applied Engineering Research, Volume 10, Number 11 (2015) pp. 29857-29865*

[11] The Linux Foundation explains the benefits of open-source collaboration with Automotive Grade Linux. Telematics Wire. Accessed: 2020-01-20. [Online]. Available: http://telematicswire.net/the-linux-foundation-explains-the-bene_tsof-open-source-collaboration-with-automotive-grade-linux/

[12] AGL. Automotive Grade Linux (AGL). Accessed: 2020-01-20. [Online]. Available: https://www.automotivelinux.org/

[13] Automotive Grade Linux Requirements Definition. Automotive Grade Linux (AGL). Accessed: 2015-03-21. [Online]. Available: https://download.automotivelinux.org/POC/PoC Spec/MASTER COPY AGL Spec v0.82.pdf

[14] Srihari, M. M., and P. Sivakumar. "Implementation of Multi-Function Printer for Professional Institutions." In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 1-3. IEEE, 2018.

[15] Sandhya Devi R.S., Sivakumar P., Balaji R. (2019) "AUTOSAR Architecture Based Kernel Development for Automotive Application"*International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018. ICICI 2018*. Lecture Notes on Data Engineering and Communications Technologies, vol 26. Springer, Cham

[16] Sivakumar, P., Devi, M. R. S., & Kumar, B. V. "Analysis of Software Reusablity Concepts Used In Automotive Software Development Using Model Based Design and Testing Tools" *In Alliance International Conference on Artificial Intelligence and Machine Learning (AICAAM), 2019,* pp.78-89.

[17] Ashwin, S., Loganathan, S., Kumar, S. S., & Sivakumar, P. "Prototype of a fingerprint based licensing system for driving", *In 2013 International Conference on Information Communication and Embedded Systems, 2013,*pp. 974-987.

[18] Kumar, P. S., Vinod, B., & Devi, R. S. (2016). Model Based Design approach for Component Design Implementation in Automotive System Development. *Asian Journal of Research in Social Sciences and Humanities*, *6*(12), 927-943.

[19] Devi, R. S., Sivakumar, P., & Sukanya, M. Offline Analysis of Sensor Can Protocol Logs Without Can/Vector Tool Usage.*International Journal of Innovative Technology and Exploring Engineering, Volume-8 Issue-2S2 December, 2018.*