Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1517

YellowCar: Automotive Multi-ECU Demonstrator Platform

Norbert Englisch[1], Owes Khan[1], Roland Mittag[1], Felix Hänchen[1], Ariane Heller[1] and Wolfram Hardt[1]

Abstract: Modern automotive industry develops software functionalities such as Advanced Driver Assistance Systems (ADAS) as software components, spread over several ECUs. Typical interconnection structure is a network, e.g. a CAN bus system. Up to date specifications and standards define operating system functions. During the development process, several questions such as functional test, performance evaluation, and optimization of software architecture for hardware independent implementation to enable reuse and standardization must be considered. For this reason, YellowCar has been developed as AUTSOAR based demonstration platform for evaluation of design methods and education of students in design methodology. Several applications demonstrate successfully YellowCar platform suitability.

Keywords: Automotive demonstrator platform**,** AUTOSAR, advanced driver assistance system development, evaluation of design methods, functional test

1 Introduction

YellowCar automotive Multi-ECU demonstrator platform has been developed as platform for functional tests, performance evaluation, and optimization of software architecture for hardware independent implementation. The platform is based on a miniature model of a car, well known from the market for kid´s toys. Battery based power infrastructure is provided as well as a network of actual three electronic control units (ECUs). These are connected via a CAN communication bus system. Several sensors have been added to the platform. Actors are a powertrain unit, which consists an electric motor which can move the car and a steering unit. Additionally, the platform has several lights that can be switched and a horn for acoustic signals.

All sensors have been connected to one single ECU, enabling it to act as a sensor ECU-node. Other ECUs have access to sensor data by reading the signals of sensor ECU from the communication bus. This specialized architecture enables the support of sensor data fusion. Other ECUs, connected to this communication bus implement functionalities to control the actors, e.g. MoveForward, Stop, SteerLeft, etc. By this open concept, new functionalities as well as additional ECUs can be added easily.

Considering that all sensor values and actor instructions are send via the communication bus debugging can be done easily by a bus interface like TinyCan. Thus, our demonstrator

[1] TU Chemnitz, Fakultät für Informatik, Straße der Nationen 62, 09111 Chemnitz, {enn | owes | mitro | haenf | aris | hardt}@cs.tu-chemnitz.de

doi:10.18420/in2017_151

builds a real world demonstrator for evaluation of academic approaches.

2 Requirements

For the YellowCar as an automotive Multi-ECU demonstrator platform we defined requirements similar to a modern car. Thus, YellowCar becomes a viable solution for evaluation of functionalities and design methods. The requirements cover the basic hardware and software architecture. Aspects of automotive sector`s digitalization have been considered. Every requirement will be referred by an YCR-"x", where x being the requirement number.

Hardware Architecture Requirements

The hardware requirements should support sensing of the environment as needed for ADAS-functionalities, network of several ECUs (Multi-ECU) and the ECUs themselves. These sensor related aspects can be fulfilled using **ultrasonic sensors (YCR-1)**, **camera (YCR-2)** and a **light sensor (YCR-3),** hence including a variety of sensing options. **CAN bus (YCR-4)** has been added as requirement since it is used widely in the automotive domain for ECU networks [Na00]. Such a bus system is very easy to implement and it is supported by most automotive ECUs. For the ECUs implementation we select **PowerPC (YCR-5)** as processor architecture.

All applications implemented on YellowCar automotive Multi-ECU demonstrator platform can rely on this hardware architecture features due to the specified requirements.

Software Architecture Requirements

For the software architecture we decided to choose the **AUTOSAR standard (YCR-6**) [AUT01], thereby setting it as a necessary requirement. In recent development strategies in the automotive industry as well as to latest research activities in test automation and evaluation where AUTOSAR is a widely used standard [KF09, En16]. **MISRA C (YCR-7)** [Ha04] as another requirement is met by compliance to the AUTOSAR software architecture. Well-developed tools like Matlab-Simulink [So16], dSPACE SystemDesk [DS01] can be used for implementation of applications on the YellowCar automotive Multi-ECU demonstrator platform. In this way, validation and testing of applications are easily achieved. It is also possible to test by modifying/designing architectural and behavioral aspects before implementing and realizing the actual application on YellowCar [So16]. Similar to a modern car essential applications are distributed to various ECUs of the demonstration platform.

3 Architecture

According to the specified requirements, we designed the YellowCar automotive Multi-

ECU demonstrator platform. The hardware architecture was defined and ECUs and sensors were mounted to the YellowCar chassis. The software architecture was defined also with respect to the requirements. Up to now, several applications have been implemented successfully.

3.1 Hardware Architecture

The YellowCar is controlled by three ECUs, as depicted in Fig. 1. Six ultrasonic sensors (YCR-1), a camera (YCR-2) and a light sensor (YCR-3) are connected to the I/Os of the ProcessingECU, reading the sensors data. The ProcessingECU converts all sensor data values to CAN messages and sends them to the standard CAN bus. The AssitantECU and the FeatureECU are connected to the same bus and implement the actor activation functionalities, e.g. MoveForward, Stop, SteerLeft, etc. The distance sensors, six ultrasonic sound sensors, three in the front and three in the back are activated one by one in a cyclic chain. This avoids activity of several sensors at the same time, so that they don't influence each other

For processing the hardware platform SPC560P50 from STMicroelectronics [So16] is used. It is based on a microcontroller with PowerPC-Architecture (requirement YCR-5). The electrical power supply is a 6V network separated from the power system of the powertrain. It is important to avoid interferences between both power systems.

For remote control of YellowCar, a server-client interface connecting the CAN bus with a network has been added. Therefore, a server was implemented on a netbook and connected to the CAN bus using the TinyCAN-Interface.

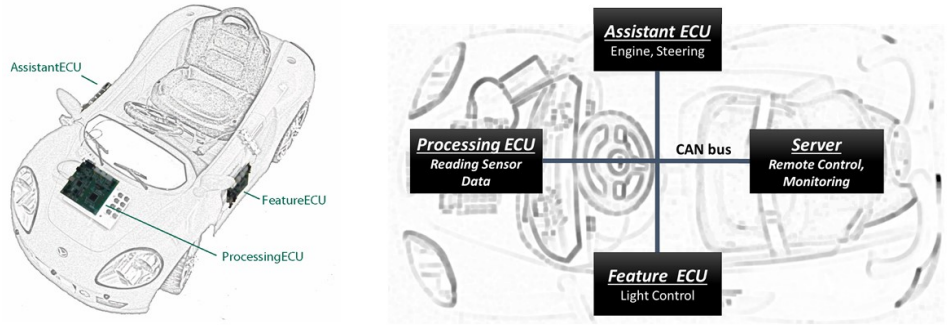

Fig. 1: Front and top view of YellowCar with ECUs and server

3.2 Software Architecture

All three integrated ECUs are realized by the system architecture AUTOSAR in version 3.1. Each ECU is separated into the horizontal layers basic software (operating system and

services), Runtime Environment (RTE) and Application Layer. The Elektrobit AutoCore and the tresos Studio are used for the configuration of the basic software and the generation of the RTE. The applications of the ECUs have been developed by the dSPACE SystemDesk and a C editor. Moreover, we have integrated Matlab/Simulink into the AUTOSAR tool chain. Main goal was to follow the recommended software structure of AUTOSAR beside the development of the applications in a reusable manner.

Especially the application of the FeatureECU contains the functionality of the light control. Our AUTOSAR compliant light control consists of more than fifteen LEDs, distributed to the car. The LEDs can be controlled by the light sensor and different action events, which are sent by the user interfaces through the server (see section 3.3). Beside the indicators and beam light, daytime light, underground, ambient interior light and puddle lights have been added. The light control was developed AUTOSAR compliant in SystemDesk (see Fig. 2) and tested in the Virtual Validation Platform (see [DHM12]) VEOS from dSPACE.

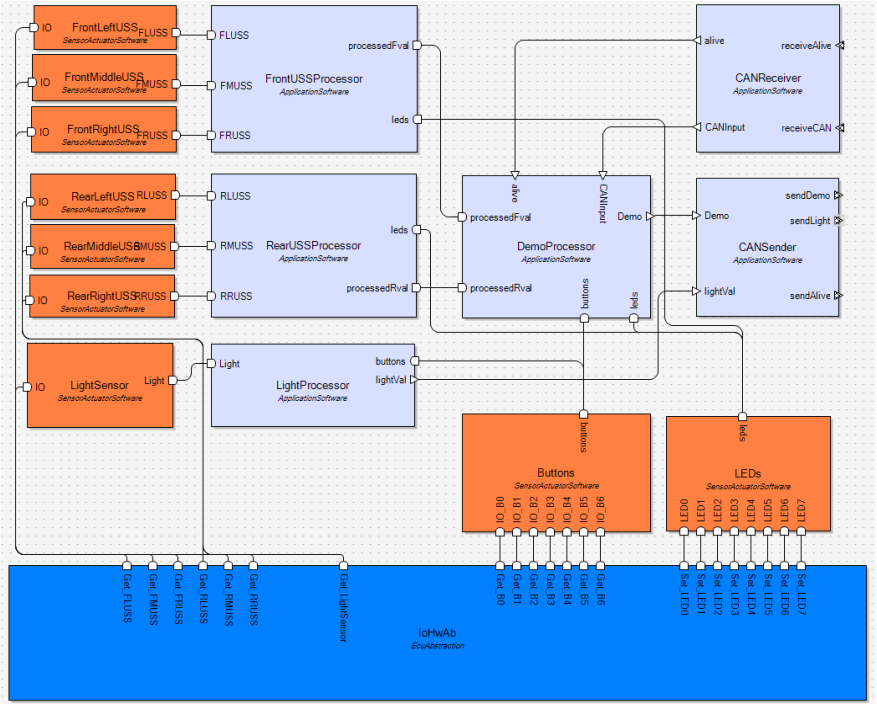

Fig. 2: Application layer diagram of light control application in AUTOSAR implemented for YellowCar

3.3 Web Interface

To enable the remote control, a server implemented in programming language C++ connects via the communication bus YellowCar with a network, like shown in Fig. 3. This server provides a webpage which represents a visualization of all sensor / control data values from the CAN bus. Moreover, by this webpage, users can remote control the YellowCar. Additionally, the same server sends data to a C# program to a configurable client, running a 3D simulation of YellowCar. This simulation represents a 3D model of the YellowCar, doing the same actions like the real car, based on the transmitted CAN messages to the server.

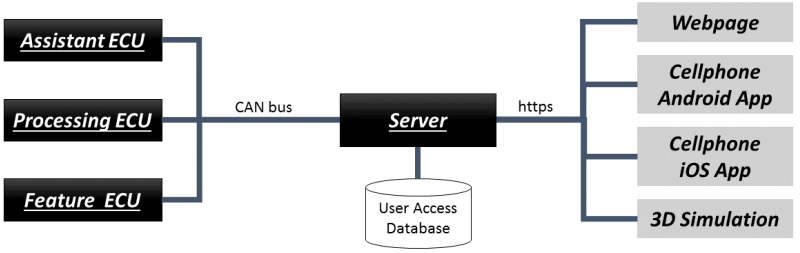

Fig. 3: Web based control, Apps and 3D Simulation of YellowCar

4 Results

As followed, we name three application examples that have been implemented on our YellowCar automotive Multi-ECU demonstrator platform. The implementations meet various pre-defined requirements. **Light Control (LC-App):** Based on light sensor data and manual inputs through web interface/switches, the LC-App controls the lights of the YellowCar. LC-App meets the requirements YCR-3, YCR-4, YCR-5, YCR-6 and YCR-7. The Feature-ECU implements this application. The Processing-ECU, to which all the sensors are connected, informs the Feature-ECU via CAN bus. **Traffic Sign Detection (TSD-App):** The YellowCar, while driving, automatically detects various road signs. The camera sensor provides raw data, preprocessing implements traffic sign detection and traffic sign recognition. CAN messages give this information to the Assistant-ECU. E.g. when a stop sign is detected and recognized, the engine-stop-message is sent. The requirements YCR-2, YCR-4 and YCR-7 have been met here. **Obstacle detection (OTD-App):** The Processing ECU detects any obstacle on the driving way of YellowCar. The ultrasonic sensors detect obstacles. OTP-App meets requirements YCR-1, YCR-4, YCR-5, YCR-6 and YCR-7. With respect to the detection, Assistant-ECU implements typical parking-assistance behavior.

Based on the presented application from the automotive domain, research approaches have been evaluated and improved. For example, the semi-automated and application specific test case generation and test execution for AUTOSAR basic software and RTE [En15] and

an approach for static analysis [SL10, En16] of AUTOSAR configurations and source code are successfully evaluated.

5 Conclusion

The automotive demonstrator YellowCar is a research platform with similar behavioral and architectural properties as compared to a modern real world car in E/E context at the Technische Universität Chemnitz. The platform supports easy extension of integration and implementation of new upcoming applications realized on automotive standards like MISRA C and AUTOSAR. Moreover, the YellowCar builds a perfect base for checking automotive academic approaches for real world applications, especially in the area of automated test case generation, for the demonstration of which several applications have been implemented.

References

[AUT01] AUTOSAR, Layered Software Architecture; Date: 26.06.2017: http://www.autosar.org/fileadmin/files/standards/classic/4-2/software-architecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf.

[SL10] A. Spillner, T. Linz and H. Schaefer: Software Testing Foundations: A Study Guide for the Certified Tester Exam, Rocky Nook, 2014, ISBN 978-1937538422.

[KF09] O. Kindel und M. Friedrich: Softwareentwicklung mit AUTOSAR: Grundlagen, Engineering, Management in der Praxis, dpunkt Verlag, 2009, ISBN 978-3898645638.

[En15] N. Englisch et al.: Application-Driven Evaluation of AUTOSAR Basic Software on Modern ECUs; Proceedings of the 13th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2015. ISBN: 978-1-4673-8299-1.

[En16] N. Englisch et al.: Efficiently Testing AUTOSAR Software Based on an Automatically Generated Knowledge Base; 7th Conference on Simulation and Testing for Vehicle Technology; Springer International Publishing, 2016; ISBN: 978-3-319-32344-2.

[Na00] M. Di Natale: Scheduling the CAN bus with earliest deadline techniques, 21st IEEE Proceedings of Real-Time Systems Symposium, 2000.

[So16] Soltani, Saeed: Dynamic Architectural Simulation Model of YellowCar in MATLAB/Simulink Using AUTOSAR System, 2016.

[DS01] dSPACE SystemDesk Product page; Date: 26.06.2017: https://www.dspace.com/de/gmb/home/products/sw/system_architecture_software/systemdesk.cfm.

[Ha04] L. Hatton: Safer language subsets: an overview and a case history, MISRA C, Information and Software Technology (2004).

[DHM12] M. Deicke, W. Hardt and M. Martinus: Virtual Validation of ECU Software with Hardware Dependent Components Using an Abstraction Layer; Simulation und Test für die Automobilelektronik, 2012. ISBN: 978-3-8169-3121-8.