VECTOR >

Charge Controller In The Zonal Architecture

Charging Communication Meets SDV

Efficient charging communication is the basis for central functions of electric vehicles. With the advent of the Software-Defined Vehicle (SDV), the question arises: How can charging communication be reliably and efficiently integrated into complex E/E architectures?

The Electric Vehicle Communication Controller (EVCC) is often the central component of a microcontroller-based onboard charger. In addition to charging communication, it also controls and monitors electrical circuits. These include the signal lines (e.g. control pilot) and the locking motor of the charging socket of the Combined Charging System (CCS). The software parts of an EVCC intended for these tasks are developed in accordance with ISO 26262 for functional safety. Cybersecurity also plays an important role in CCS-based charging systems. As charging communication is an interface to the outside world, it can be used by attackers as a potential gateway to gain access to the vehicle's internal network. As the leading standard for charging communication, ISO 15118 therefore specifies important security precautions for authenticity, integrity and confidentiality. These include the encryption of charging communication using Transport Layer Security (TLS) and the use of Public Key Infrastructure (PKI) for authentication and authorization of the vehicle and charging station. These security mechanisms require additional resources due to cryptographic operations and certificate management. The increased runtime in particular can have a negative impact on the user experience, as there can be waiting times from the start of charging communication to the actual energy transfer. However, the fast performance of cryptographic operations in software poses a major challenge for conventional microcontrollers due to the use of modern elliptic curves.

New Possibilities With the Software-Defined Vehicle

The trend towards SDV is reflected, among other things, in the structure of modern E/E architectures. These are often setup as a zonal architecture. The vehicle is divided into geographical zones. In addition to the microcontroller-based zonal and sensor/actuator ECUs, the central high-performance computers (HPC) are an integral part of the SDV. They are often executed as a system-on-chip (SoC) with powerful processors and POSIX-based operating systems. There are various approaches for implementing an EVCC in the zonal architecture. One possibility is the aforementioned onboard charger. It is integrated as a separate component in a zone of the E/E architecture (Fig. 1, 1). Another variant is the integration of the EVCC in a zonal ECU (Fig. 1, 2). A zonal ECU that is installed near the CCS charging socket is ideal for this in order to minimize the distance for Powerline Communication (PLC). The third ap-

proach is to implement the EVCC as a distributed system, whereby the individual software functions are distributed depending on the properties of existing ECUs (Fig. 1, 3).

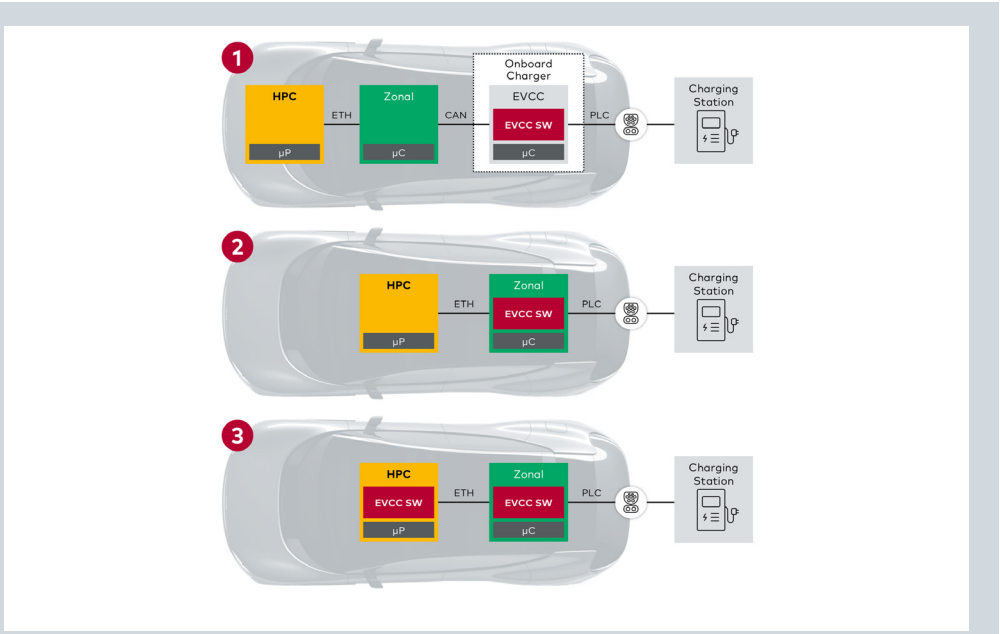

Figure 1 : Different approaches for the realization of an EVCC in the zonal architecture.

Solution of a Distributed EVCC

This third approach was investigated as part of a pre-development project at Vector. The distribution of the software is primarily carried out in accordance with the requirements of functional safety and cybersecurity. The EVCC software is split between two ECUs. The safety-related applications for controlling electrical circuits are executed in a microcontroller-based gateway with Vector's MICROSAR Classic basic software in order to benefit from the real-time capability. The gateway can be implemented within a zonal ECU (Figure 2) or as a stand-alone sensor/actuator ECU. In contrast, the load communication with the resource-intensive software components for cybersecurity is largely executed on an HPC. With the help of powerful processors, shorter runtimes can be achieved for cryptographic operations. The HPC is operated with Android Automotive OS (AAOS). Figure 2 shows the distribution of the various software functions within the system.

The gateway handles communication with the charging station via Powerline Communication and with the battery management system (BMS) via the internal vehicle network. It integrates the PLC chip and controls the hardware I/Os of the charging socket. The first steps of the charging sequence, such as Signal-Level Attenuation Characterization (SLAC) and SECC Discovery Protocol (SDP), also take place there. The gateway also forwards messages from the charging station to the Android system in the HPC. This is done via a NAT-like (Network Address Translation) router within the gateway, which only allows messages from the charging protocol to pass through to the vehicle network. Unwanted or unknown messages are filtered out. The NAT-like router contactor protects the internal network and the external PLC interface. It supports the requirements of ISO 21434. The gateway and the HPC are connected via Ethernet over two channels. One is used for the loading protocol, the other for synchronizing the states between the Android system and the gateway. As the gateway starts faster, individual charging steps can be initiated before the HPC is available. SLAC and SDP start without any input from the Android system, which shortens the time until charging actually begins. A large part of the charging communication is processed in the Android Automotive OS of the HPC. In addition, cryptographic operations are executed and the required certificates and keys are managed, which is essential for Plug & Charge, among other things. Charging communication is implemented in AAOS as a charging service. It is based on the well-known charging communication software MICROSAR Charge from Vector and contains the protocol-specific implementation including the state machines, the EXI (Efficient XML Interchange) encoder/decoder and XML Security for the application messages. The architecture has a modular setup. External components such as the Crypto library or certificate and key management can be integrated interchangeably. The charge service is integrated into the Android framework as a native service. This ensures a quick start and monitoring by the service manager. The latter starts the service, monitors it and restarts it if necessary. The charge service can also be developed and maintained with the help of the Vendor Native Development Kit (VNDK). The allocation of the charge service within AAOS is important, as Android systems have several partitions with different purposes and authorizations (Figure 3). The system partition with the components of the Android core system has the highest privileges and unrestricted API access. However, it is not intended for user-specific extensions. The charge service is kept in the vendor partition, as this has access to all necessary system APIs and offers pro-

tection mechanisms against unauthorized changes to the data.

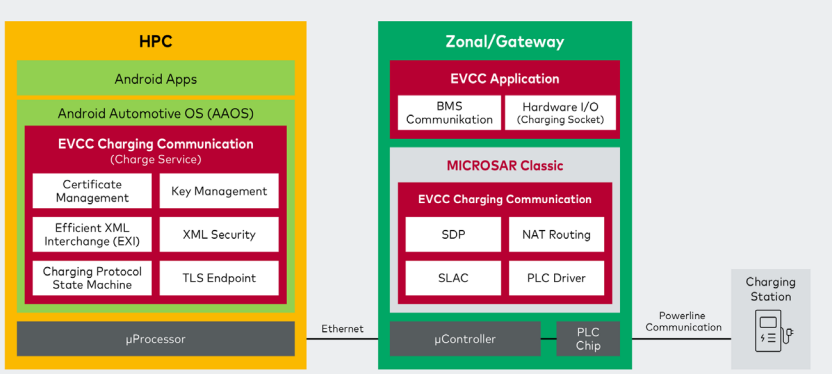

Figure 2: Distribution of various software functions within a system.

Evaluation

A distributed EVCC solution has a particularly positive effect on runtime behavior. In real test scenarios, the runtime of a TLS 1.3 handshake with client and server authentication in accordance with ISO 15118-20 was improved by a factor of 8-10 compared to a microcontroller-based system. In addition, all time limits required by the standard were met during the entire loading sequence.

The memory requirements for the certificate chains are less significant, as SoC platforms have significantly larger memory resources than conventional microcontrollers. In addition, the microcontroller for controlling the charging socket can be designed much smaller, as the charging communication is shifted to the HPC. This leads to significant cost savings. Furthermore, the update capability and start-up time are important properties of an EVCC. The update strategy is often manufacturer-specific and depends on the technical properties of an ECU. The software distribution listed above offers the possibility of updating the charging protocol centrally in the HPC, which is particularly advantageous when using multiple charging sockets. An example therefor are electrically powered trucks with multiple charging sockets for the CCS and MCS (Megawatt Charging System). In addition, variant management can be simplified by dynamically configuring functions in the charging service.

A fast start-up time leads to a positive user experience thanks to a short waiting time during the charging process. Microcontroller-based systems enable startup times in the millisecond range, whereas HPCs often require several seconds. In a distributed EVCC solution, the start-up behavior must be designed in such a way that charging communication can begin as soon as the charging cable is plugged in. Therefore, the microcontroller executes the first steps of the charging sequence. In addition to the positive aspects, however, the distributed EVCC solution leads to increased complexity. For example, the number of communication paths within the vehicle increases. Different hardware platforms and software ecosystems are also involved in the charging process. This increases the effort required to test the overall system, as the Test Setup becomes more complex due to the involvement of several ECUs and the number of test cases increases.

Conclusion

Vector's pre-development project illustrates the advantages and disadvantages of different integration approaches of an EVCC. The approach selected should be based on the available hardware and software distribution within the zonal architecture. In addition, future hardware platforms with more powerful processors and hardware accelerators may result in new integration scenarios. However, suitable charging communication software must be flexible and portable. In the SDV, economic and project-specific factors of ECU development must be taken into account more

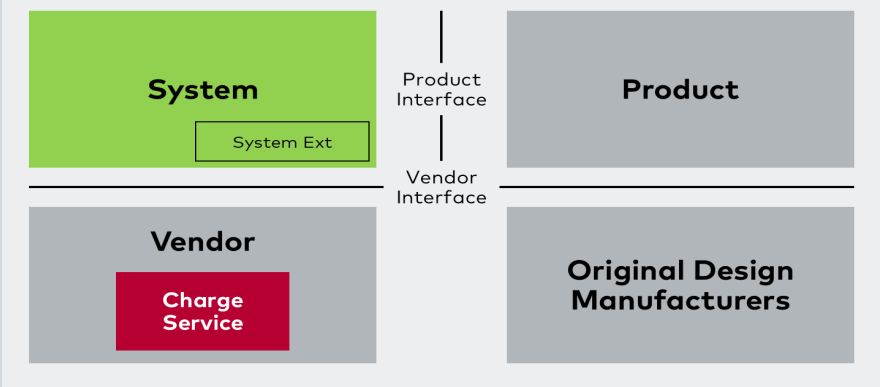

Figure 3: Overview of partitions in Android Automotive OS

than ever. From a technical point of view, nothing stands in the way of implementing efficient charging communication in the SDV.

Tobias Schneider
worked in various departments at Vector after studying electrical engineering. Since 2016, he has been working as a project manager in pre-development, where he is responsible for various topics in the field of embedded software.

Michael Vick
has been working on the topic of charging communication in various roles at Vector since 2018. He is currently Solution Manager for the charging communication software MICROSAR Charge.

Translation of a German publication in ELEKTRONIKPRAXIS 1/2026
Image rights: Vector Informatik GmbH