

Building blocks for the Software-Defined Vehicle

B. Kreipe¹, J. Speh¹

1: Volkswagen, Germany

1. Abstract

Achieving full update capability for all vehicle functions is a key driver of the software-defined vehicle (SDV). This software-centric approach significantly impacts software allocation, function implementation, and overall electronic architecture. The most notable changes include the centralization of functions within a main control unit, a shift from domain-based to zonal architecture, and the introduction of microcontroller (MCU)-free and therefore software-free actuators.

However, transitioning to vehicle architectures without local MCUs presents several challenges. This paper explores key aspects, challenges, and solutions for the practical implementation of the SDV paradigm, with a particular focus on lighting actuators.

In classic architectures, the local MCU performs several critical tasks, including monitoring communication with other network participants, controlling driver modules, and handling diagnostics. In the SDV, these functions must be re-evaluated and reassigned.

A key consideration is the implementation of hardware abstraction. Traditionally, the local MCU serves as the bridge between the in-vehicle network, which operates on standardized protocols, and various driver devices that rely on proprietary interfaces and protocols. If this bridge is simply removed, driver-specific control must be managed by the host. On the lamp side, the execution of safety-relevant functions must be ensured in case of failure. Additionally, robust security measures are required to protect communication and prevent unauthorized hardware access. Implementing all these aspects directly within driver devices appears impractical.

A more viable approach is the combination of SDV-optimized endpoints, driver devices, and application-specific companion chips. This approach enables effective hardware abstraction, ensures multi-vendor compatibility, and supports fail-safe concepts for



microcontroller-free actuators. Furthermore, it addresses security challenges while maintaining the low-latency performance required for time-critical applications.

Keywords: software-defined vehicle, lighting control, remote control, microcontroller-free and software-free design, endpoint, smart drivers

2. Motivation

The automotive industry is currently shaped by four major electronic trends: autonomous driving [1], zonal architecture [2], 48V electrical systems [3], and the software-defined vehicle (SDV) [4]. These trends aim to address evolving demands in mobility, system efficiency, and vehicle functionality.

Ambitious goals are associated with each of these trends. Autonomous driving targets the development of new mobility paradigms, necessitating redundant perception, processing, and communication systems. Zonal architecture seeks to reduce wiring complexity and the number of electronic control units (ECUs) by enabling high levels of functional integration. The transition to 48V electrical systems promises substantial reductions in both wiring material and overall vehicle weight. Meanwhile, the SDV paradigm focuses on achieving full update capability across all vehicle functions via a software-centric approach—transforming how software is allocated, how functions are implemented, and how the electronic architecture is organised.

Among these trends, the software-defined vehicle has, in recent years, emerged as the top priority. It enables the control and continuous development of complex vehicle functions, while also supporting over-the-air updates across the fleet. These updates can be used to meet evolving regulatory requirements or to expand the functional scope of existing vehicle capabilities.

A fundamental building block in the SDV is a centralised functional intelligence that cooperates with smart modules. This allows new features to be deployed without the need to reflash subordinate ECUs.

Lighting functions in particular benefit from this architecture. Functionality can be expanded through function-on-demand services, enabling the introduction of entirely new use cases. Additionally, pixelated lighting systems can support design-on-demand concepts—allowing new visual content to be deployed dynamically. This paper examines transition strategies and key requirements for migrating from classical vehicle architectures to the software-defined vehicle.

3. Architecture transition

In classical vehicle architectures, a functional control unit—such as a lighting control module—is connected to a central host ECU via the in-vehicle network (see Fig. 1). The lighting control unit typically includes a microcontroller (MCU), which acts as a bridge, abstracting the control of downstream driver components (e.g., LED drivers) via a local bus. These drivers in turn control the actual sensors or actuators, such as LEDs. The overall lighting function is therefore distributed, with software components residing both in the central host and in the dedicated lighting control unit.

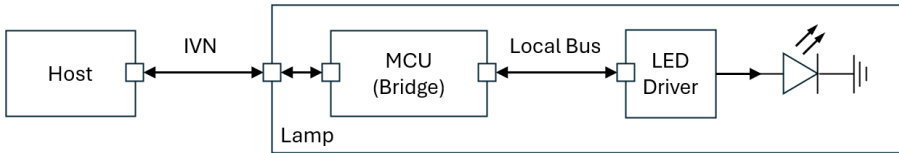


Figure 1: Classical architecture for lamp control featuring an MCU as bridge device (IVN = In-Vehicle Network, MCU = Microcontroller).

The SDV paradigm, which emphasises software centralisation, naturally leads to the concept of software-free electronic control units. This omits the requirement of device flashing and, if all software functionality can be removed from the ECU, even offers the possibility of an MCU-free device. In this idealised SDV scenario, the driver device is controlled directly by the central host, as shown in the frequently simplified representation in Figure 2.

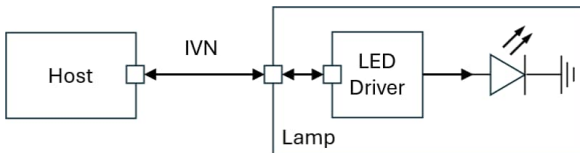


Figure 2: Commonly cited, simplified SDV architecture of a lamp without an MCU or embedded software.

However, achieving this ‘holy grail’ of an MCU- and software-free lighting actuator entails addressing a range of critical requirements. Exterior lighting systems are categorised as ASIL-B components under automotive functional safety regulations, and are subject to stringent homologation constraints. Moreover, such systems often integrate drivers from different suppliers and of varying types (e.g., LED and motor drivers), typically interfaced via proprietary on-board bus protocols. In classical architectures, much of this complexity is encapsulated in software blocks managed by the MCU.

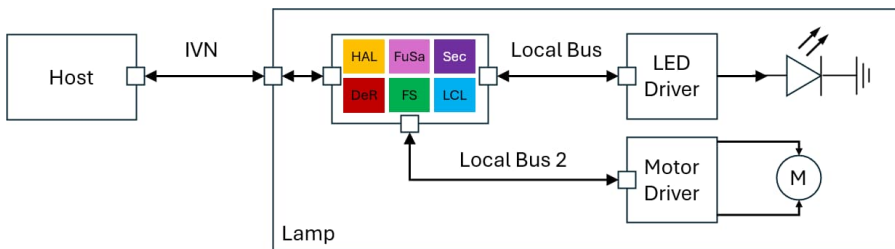


Figure 3: Detailed breakdown of lamp control elements in a classical architecture including functional blocks within the MCU.

Figure 3 depicts a detailed view of lamp control in the conventional setup. Clearly, removing the MCU in such a configuration necessitates a careful reallocation of software functions—either to the host or to the driver level—and the bridging of disparate interface protocols (Fig. 4). The resulting challenges of this architectural transition will be discussed in the following chapter.

4. Challenges

This chapter examines the functions and dependencies of the key elements involved in lamp control, with the aim of deriving requirements and design considerations for their implementation within an SDV architecture (cf. Fig. 4).

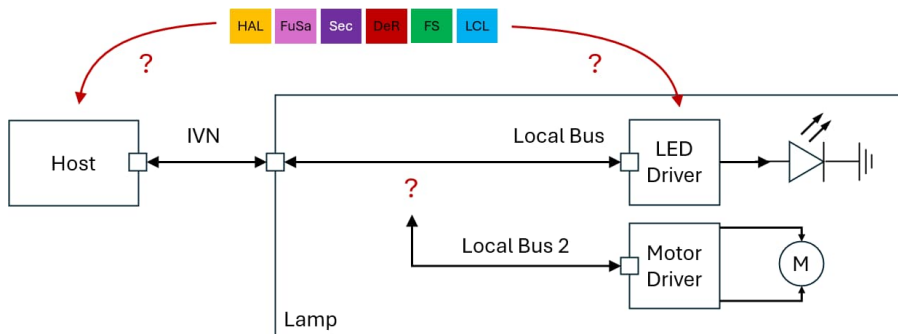


Figure 4: Architecture according to figure 3 with the MCU removed.

Bridge Functionality

Compared to internal lamp connections, in-vehicle networks (IVNs) are subject to significantly greater challenges in terms of cable length, electromagnetic interference (EMI), mechanical stress, and vulnerability. IVNs are also highly standardised, while drivers typically rely on proprietary protocols tailored to specific applications and manufacturers. A bridging component is therefore essential to translate signals and

protocols between such diverse network interfaces. The bridge also enables interface multiplication, allowing the connection of a much larger number of driver components than the IVN node count alone would support.

Hardware Abstraction

The hardware abstraction layer (HAL) maps functional software commands to the register-level control interfaces of specific drivers. This abstraction decouples functional logic from physical hardware, enhancing modularity and maintainability.

In traditional architectures, this role is typically fulfilled by Tier1 suppliers, who tailor vehicle functions to specific driver portfolios. However, shifting HAL responsibilities to the central host tightly couples abstraction to core system software. As a result, any changes require extensive integration and release testing, reducing agility.

Functional Safety

Exterior lighting systems are safety-critical and must comply with functional safety (FuSa) standards. Primary lighting functions are usually classified as ASIL B. To ensure safe data transmission, end-to-end (E2E) protection mechanisms are applied, including watchdog timers, checksums (CRC), dynamic identifiers, and message counters. IVNs are generally subject to stricter safety validation than internal lamp buses. However, current driver ICs are often not equipped to implement all required FuSa mechanisms independently.

Fail-Safe Operation

To meet safety requirements, lamps must transition to a safe state in the event of communication failure with the central controller. This necessitates both bus diagnostics and self-monitoring capabilities. Safe states may be static—hardcoded into the driver—or dynamic, dependent on vehicle status. However, dynamic fallback logic cannot typically be implemented within the driver alone, presenting a challenge in MCU-free architectures.

Security

As networking technologies evolve and communication paradigms shift from signal-based to remote-controlled devices, cybersecurity requirements are increasing. Measures such as hardware authentication and message authentication (e.g. MACsec, SecOC) are essential. However, drivers without embedded MCUs generally lack the resources to support advanced cryptographic operations or key management. Thus, a direct host-driver communication cannot fulfill security requirements and necessitates a bridge, which may connect dedicated external authentication components capable of performing these tasks in hardware.

LED Binning and Derating

MCUs typically offer numerous GPIOs and high-precision ADC interfaces used for voltage measurements tasks such as temperature-based derating and identifying LED binning resistors. If the MCU is removed, these functions must be performed by either the driver or bridge. Existing drivers usually offer only limited ADC precision due to reference voltage and resolution constraints. This requires detailed analysis and tolerance evaluation to ensure reliable measurement. In some cases, binning data could be stored in an EEPROM to minimise interface demands.

Local Control Loops and Time-Critical Applications

Certain applications require real-time, local data processing due to the latency and jitter introduced by the IVN. A classic example is anti-pinch protection in electric windows, which shall operate independently of central communication.

In lighting systems, the control of stepper motors is a similarly time-critical function. In traditional setups, these were handled by local MCUs. Within the SDV architecture, such functions must be offloaded to dedicated SoCs capable of real-time control.

5. Direct host-driver-communication

One potential approach for realising the SDV paradigm is the direct connection of smart drivers to the central host computer (cf. Fig. 2). In this scenario, local bus protocols such as UART may be transmitted over a robust physical layer (e.g. via CAN or LVDS) to the host, or alternatively, in-vehicle network (IVN) interfaces such as CAN-FD may be integrated directly into the driver ICs.

While this architecture may offer cost advantages, it cannot fulfil all the system-level requirements outlined in the previous chapter. Several key limitations emerge:

1. **Lack of Standardisation:**
This approach inherently depends on a closed, single-source ecosystem, as no standard exists for on-board driver interfaces. Interoperability between suppliers is therefore not achievable.
2. **Limited Functional Safety and Security Capabilities:**
As discussed in section 4, driver ICs generally lack the computational and hardware resources required to implement comprehensive functional safety (FuSa) and cybersecurity measures, such as end-to-end safety mechanisms or message authentication.
3. **Scalability Constraints:**
Without a port-multiplying bridge, the number of addressable driver devices is

restricted to the maximum node count supported by the bus system. Moreover, since bus bandwidth and node count are inversely related, increasing system complexity may result in performance bottlenecks. Although higher-channel-count drivers can partially mitigate this issue, they do not offer a scalable solution for complex, distributed lighting systems.

From the authors' perspective, this architectural model is therefore suitable only for isolated, low-complexity applications, rather than as a general solution for the SDV lighting domain.

6. Endpoints

An alternative solution emerges with the availability of endpoints, introduced for scalable zonal architectures [5]. These tailored bridge devices are optimised for the remote control of particular applications—such as lighting—and are capable of interfacing with a wide variety of driver ICs. Endpoints also act as port multipliers, effectively extending the connectivity of the in-vehicle network (IVN). Beside the classical IVN interfaces such as LIN or CAN, they support different Ethernet types, such as 10BASE-T1S or 100BASE-T1 for a seamless integration in Ethernet-based zonal architectures. As they require only parameterisation and contain no user-programmable software, endpoints can be classified as MCU-free. These characteristics make them highly suitable for software-defined vehicle (SDV) implementations in which sensors or actuators are remotely controlled directly from a central host computer.

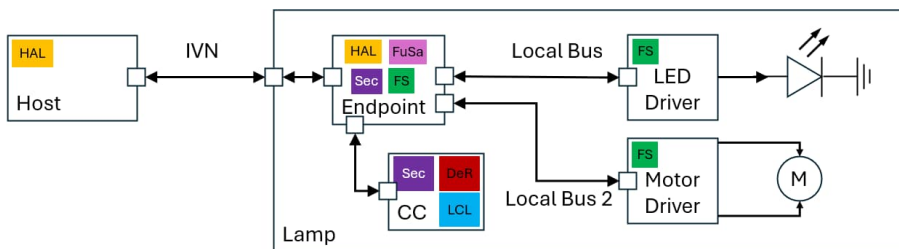


Figure 5: SDV architecture based on endpoints and companion chips (CC).

For cost-sensitive designs, it is neither necessary nor desirable to implement all required functions within the endpoint itself. To retain system scalability and flexibility, endpoints may be complemented by application-specific companion chips, such as ASICs or SoCs. This allows for the following functional repartitioning:

- Essential tasks, such as functional safety mechanisms and basic security (e.g. watchdogs, CRC checks), are implemented within the endpoint.

- Advanced features, such as complex cryptographic operations or application-specific control loops, are offloaded to the companion chip.
- Some functions, including hardware abstraction or fail-safe state logic, may be distributed across the host, endpoint, or driver, depending on the overall architecture and the functional capabilities of each component.

Figure 5 illustrates this concept, showing how endpoints and companion chips (CCs) can be used to implement an SDV-compatible lighting control system that supports flexible and scalable function allocation.

Although the endpoint-based approach introduces additional components compared to direct host–driver communication (cf. Section 5), it offers considerable potential for cost reduction and printed circuit board (PCB) space savings relative to traditional microcontroller-based solutions. Figure 6 presents a side-by-side comparison between a legacy light control unit using a general-purpose MCU and an SDV-optimised design incorporating endpoints for the remote control of all headlamp functions.



Figure 6: Comparison of a conventional light control unit with MCU (100-pin, left) and an SDV-optimised control unit with endpoint (32-pin, right).

7. Summary

Among all in-vehicle domains, exterior lighting exhibits one of the strongest demands for remote control. Only a highly flexible control architecture—capable of handling ever-increasing resolution—can fully unlock the potential of function-on-demand and design-on-demand content. As such, lighting represents an ideal application area for software-defined vehicle (SDV) architectures.

Achieving this vision requires the development of software- and microcontroller-free (MCU-free) lighting control units. However, the removal of the MCU from legacy control units represents the most complex aspect of this transition. Endpoints, particularly when combined with application-specific companion chips, offer a practical and scalable migration path. This architectural approach supports modularity, functional safety, and system scalability, without sacrificing cost efficiency.

In particular, the integration of endpoints with standardised in-vehicle Ethernet interfaces provides a robust foundation for future-proof SDV lighting systems. Nevertheless, to realise the full benefits of this paradigm, coordinated development efforts are required—especially in the co-design of smart drivers—and significant progress must be made towards the standardisation of driver interfaces, diagnostics, and register sets.

8. References

- [1] L. Chen et al., “*End-to-end Autonomous Driving: Challenges and Frontiers*”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, doi:10.1109/TPAMI.2024.3435937
- [2] J. Maier et al. “*Design of Zonal E/E Architectures in Vehicles Using a Coupled Approach of k-Means Clustering and Dijkstra’s Algorithm*”, Energies, 2023, doi:10.3390/en16196884
- [3] A. K. Kumawat et al., “*A Comprehensive Study of Automotive 48-Volt Technology*”, International Journal of Mechanical Engineering, 2017, doi: 10.14445/23488360/IJME-V4I5P103
- [4] S. Jiang et al., “*Vehicle E/E Architecture and Key Technologies Enabling Software-Defined Vehicle*”, SAE WCX World Congress Experience, 2024, doi:10.4271/2024-01-2035.
- [5] Kreipe et al., “*Ethernet-based zonal lighting-architecture and applications for automotive exterior lighting*”, AEC 2023, WEKA Fachmedien GmbH, 2023.

9. Abbreviations

ADC	Analogue-to-Digital Converter
ASIL	Automotive Safety Integrity Level
CAN	Controller Area Network
CC	Companion Chip
CRC	Cyclic Redundancy Check
DeR	Derating

E2E	End-to-End
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMV	Electromagnetic Compatibility
FuSa	Functional Safety
FS	Fail Safe
GPIO	General-Purpose Input/Output
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
IVN	In-Vehicle Network
LCL	Local control loop
LED	Light Emitting Diode
LIN	Local Interconnect Network
LVDS	Low-Voltage Differential Signalling
MACsec	Media Access Control Security
MCU	Microcontroller Unit
SDV	Software-Defined Vehicle
SecOC	Secure Onboard Communication
SoC	System on Chip
UART	Universal Asynchronous Receiver-Transmitter