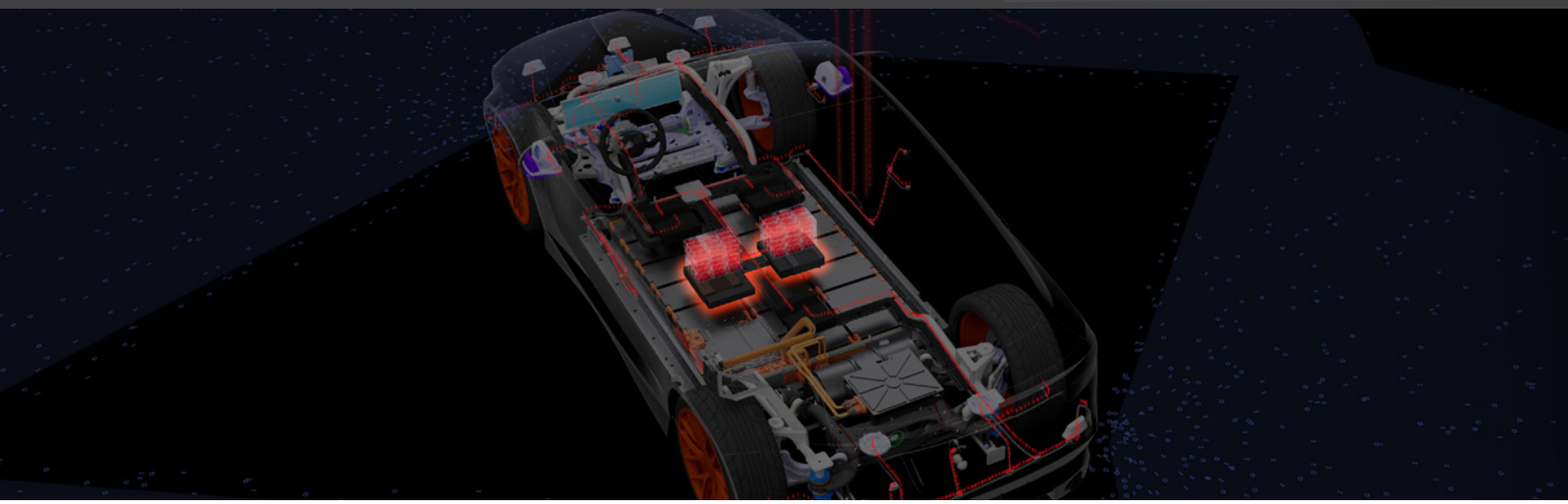


Open Server Platform Requires Shifts in Hardware and Software

The biggest strength of the software-defined vehicle is that it evolves. A vehicle leaves the factory with a certain set of capabilities, but the OEM is able to leverage usage data and expand those capabilities over the vehicle's full life cycle through over-the-air updates, enabling it to deliver better driving performance, improve the in-cabin user experience (UX), and otherwise make an aging vehicle feel new every day.

Hardware, however, does not evolve. Historically, once the vehicle left the factory, the hardware remained the same.

This dichotomy eventually presents a challenge to the software-defined vehicle. While centralizing and serverizing compute can be a cost-effective solution for sharing workloads, reallocating resources and extending its usefulness, at some point, the compute hardware will be too outdated to take on new software features that require faster processing, more memory or greater storage capacity.

OEMs can address these limitations with the right software and hardware architecture — one that not only has the flexibility to dynamically reallocate resources in real time, but also allows physical upgrades to just the specific compute components that need it the most, well into the future. Done right, the approach will also help them create independent software and hardware ecosystems while greatly extending the life of their vehicles.

BRAIN SURGERY

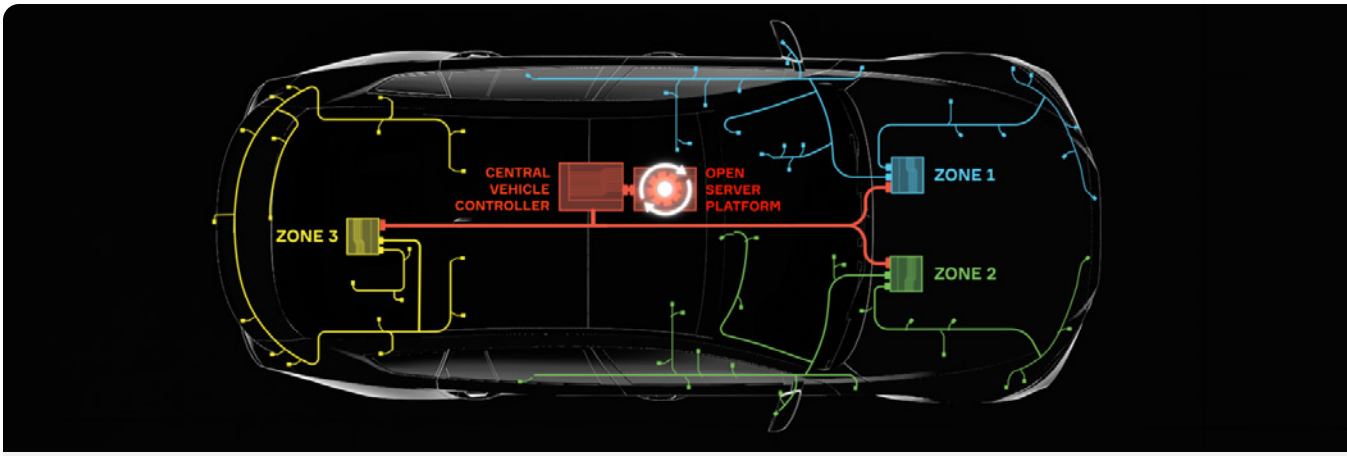
We often think of the compute in a vehicle as its "brain." After all, just as a human brain takes in information and acts based on that input, the vehicle's compute platform takes in data from sensors and other outside sources and makes decisions about what to do and what to communicate.

A human brain, however, changes and grows over time. Babies grow into children, and children grow into adults. As their brains physically change, they become capable of greater understanding and more complex reasoning.

In contrast, computers are static, with defined processing limits. As software developers create ever more sophisticated and capable applications, they push those limits until it is necessary to move to the next generation of hardware to enable the functions they create.

In the automotive world, newer hardware could also become necessary to meet evolving safety regulations or to ensure that the vehicle continues to employ the latest cybersecurity measures.

Consumers are used to this upgrade cycle in other contexts. In the mobile phone world, for example, many people upgrade to a new device every two to three years, in part to obtain a processor that can adequately support the latest apps, functions and cybersecurity protections. A key difference, of course, is that the expense of replacing an entire phone is much less than that of replacing an entire vehicle. A vehicle also is obviously far more complex, with hundreds of devices distributed throughout, which are increasingly connected to a central compute unit.



OEMs can address these limitations with the right software and hardware architecture — one that not only has the flexibility to dynamically reallocate resources in real time, but also allows physical upgrades to just the specific compute components that need it the most, well into the future. Done right, the approach will also help them create independent software and hardware ecosystems while greatly extending the life of their vehicles.

Hardware architecture

The solution in automotive is to structure the software and hardware architecture so that OEMs can upgrade just the compute needed to execute higher-level functions.

On the hardware side, OEMs are already moving to a zonal architecture, where many of the devices in the vehicle connect to their closest zone controller, which in turn aggregates data from those devices and communicates with a central vehicle controller (CVC). In addition to working with the zone controllers to handle all data communication protocols and signaling with the devices, the CVC handles body control functions, data storage, vehicle access, communication with the outside world, and potentially propulsion and chassis control.

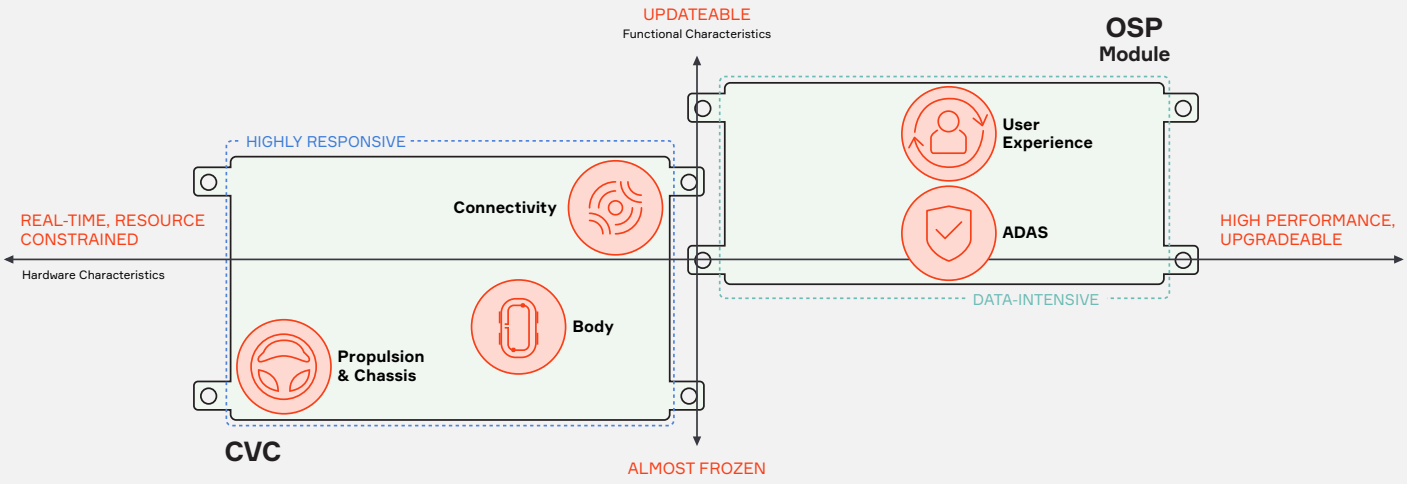
In other words, the CVC and zone controllers manage all of the lower-level functions necessary to run a vehicle. This architecture allows higher-level functions to be supported by a separate compute platform: the open server platform, or OSP.

The OSP is the brain of the vehicle, performing functions that require complex and data-intensive compute capabilities, usually related to advanced driver-assistance systems (ADAS) or in-cabin UX. It might actually be more precise to say that the OSP is the cerebrum while the CVC is the cerebellum, or "small brain," because the CVC translates directives coming from the OSP into actions carried out by the vehicle.

Separating the two architecturally makes a lot of sense. The lower-level functions and communication with vehicle input/output do not change much over the life of a vehicle. But higher-level functions are continuously evolving as developers solve for more use cases or create more innovative UX features. Higher-level functions also require more powerful processing, more memory and a graphics processing unit.

Separate and Unequal

Different vehicle software domains have different needs, so the hardware architecture must be designed to best accommodate those differences. That means running functions either on the central vehicle controller (CVC) or the open server platform (OSP), as appropriate.



Software architecture

The software architecture must also be structured appropriately to support the separation. The key factors are abstraction and interface standardization — that is, presenting a standard, consistent interface to higher levels of software so that those higher levels do not have to be concerned with the details of how the lower levels work.

At the lowest level is the device abstraction layer, or DAL. Devices include everything from sensors (such as radars, cameras and thermal sensors) to actuators (such as seat controls, door locks and window lifters). The zone controllers handle all direct communication with devices, and the DAL presents a standard set of application programming interfaces (APIs) through microservices to higher levels of software for control and diagnostics.

For example, the higher-level software could request through a DAL API that a window be lowered, without having to know how to talk to the window lifter motor; instead, the zone controller would format the appropriate signal through a Local Interconnect Network bus and send that to the window motor. Even sensor data streaming would be standardized.

The next level is the vehicle abstraction layer, or VAL, which abstracts the more complex body control functions managed by the CVC. High-level software would interact with VAL APIs to get data or take actions.

To continue the window-lifter example, abstraction at the VAL level could include a service that manages all aspects of window operation, including user controls, the window lifter motor and so on. Say the in-cabin UX function received a request from a user via the infotainment system to lower all the windows by 50 percent. The UX software would send a command to the window service with those instructions, which would then send individual commands to the DAL APIs for the window lifters on each of the four windows for the duration necessary to lower the windows by 50 percent.

Software containers package together all of the files and libraries with the application that needs them, making the code relatively independent from its host environment.

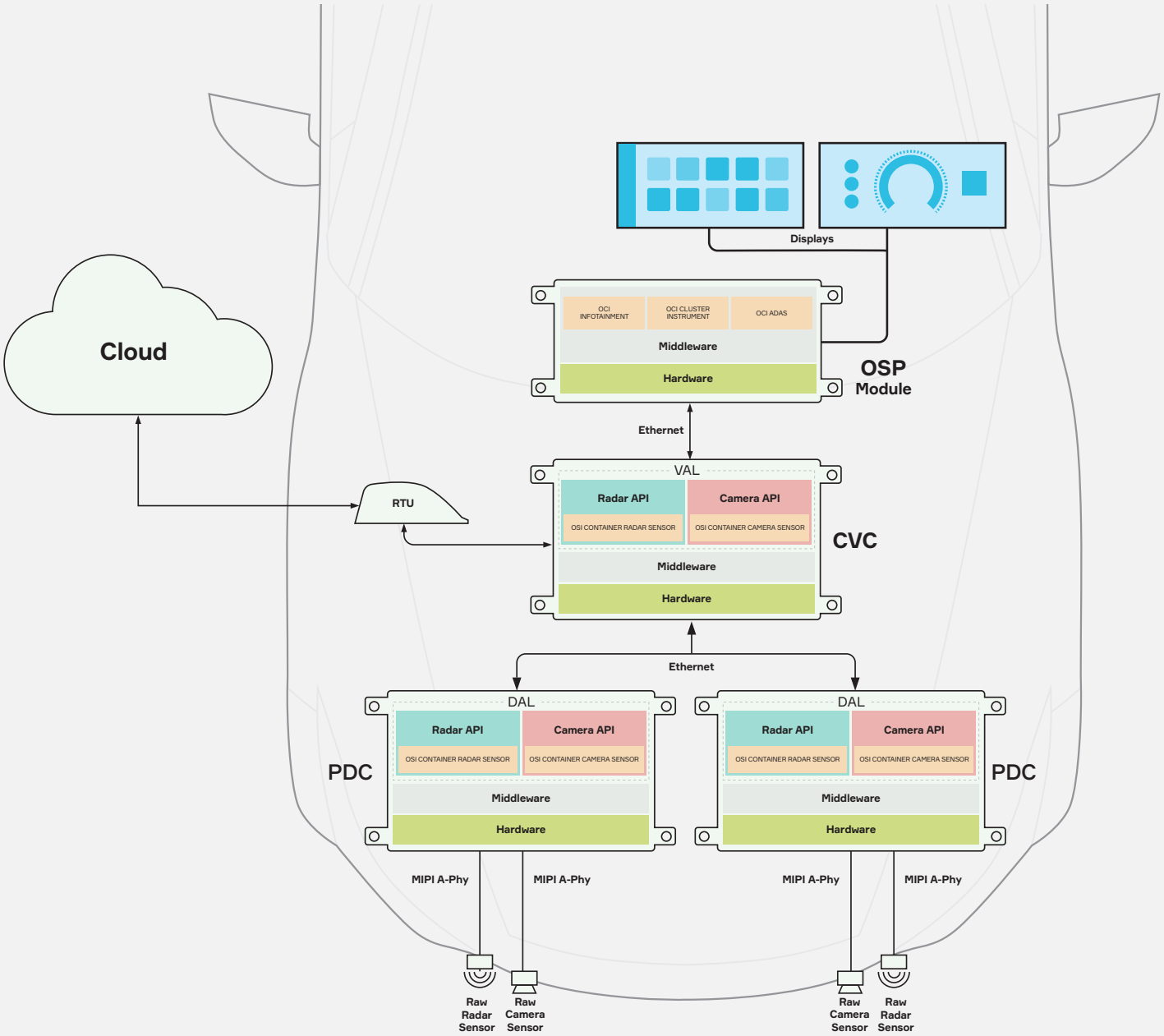
Containerization

Another important piece of the software architecture is containerization. Software containers package together all of the files and libraries with the application that needs them, making the code relatively independent from its host environment. This approach helps to create a service-oriented architecture where individual services, each tied to its own container, can be updated individually without affecting other software.

Containerization also helps disconnect software development from the hardware it runs on. That means that containerized services could potentially be moved from one compute platform to another. It also means that containerized applications are not tied to the hardware they run on, so the hardware that comprises an OSP could be changed — from one system-on-a-chip to another, for example.

Layers of Abstraction

Each device in the architecture uses software abstraction to enable higher levels of functionality. Power Data Center (PDC) zone controllers handle communications with devices and present a Device Abstraction Layer (DAL) to the central vehicle controller (CVC), which in turn presents more complex functions to applications running on the open server platform (OSP).



A FLIPPED SCRIPT

With those major architectural elements in place, the OSP becomes swappable. The hardware is structured so that the higher-level functions are isolated on the OSP. The software is structured so that any applications on the OSP communicate to the vehicle through standard interfaces. And software containerization abstracts the applications' access to compute on the OSP. As the OSP ages and becomes unable to run the latest applications or receive the latest operating system updates, the owner can take it to a dealership to exchange it for a newer and more capable OSP — without disturbing the rest of the vehicle architecture.

The ramifications go well beyond the OSP, however. Taken together, these factors enable independent software and hardware ecosystems.

- **Machine ecosystems:** The hardware can be developed and certified independently of any application software. Performance can be characterized by standard benchmarks such as processing speed and memory capacity. Development cycles can be 18 months or less.
- **Device ecosystems:** All devices in the vehicle will combine proprietary hardware and software but will be aligned to a standard API. They will connect via standard buses. Their development will not be connected to any particular machine or application.
- **Application ecosystems:** Containerized applications can move from one hardware platform to another without a software change. Standard UX and ADAS stacks with well-defined APIs can help scale. And all applications can be developed in a cloud environment.

Because these ecosystems are independent, vendor lock-in becomes obsolete. Devices, applications and machines can all be replaced without disturbing the other ecosystems.

Having swappable OSPs also allows OEMs to easily differentiate among models or trim levels. Two models with the same devices and CVC could have different ADAS or UX capabilities, just by connecting different OSPs with hardware that correlates with their software capabilities.

When a vehicle is built with these hardware and software foundational elements, the way is clear for realizing the true potential of software-defined vehicles, ensuring that they have the most up-to-date capabilities for years to come.

THE VIEW FROM SVA™

Aptiv's Smart Vehicle Architecture™ technologies take all of these factors into account as they lay the groundwork for future enhancements.

From a hardware perspective, our OSP modules can be designed to either stand alone or install directly atop a CVC, with a liquid cooling plate between them to keep temperatures down. The configuration further optimizes hardware cost and makes the OSPs easy to swap out when an upgrade is needed.

Automotive Ethernet and PCI Express connect the OSPs to the CVC. Current versions of ADAS-focused compute solutions connect directly to some sensors, but future versions will route all of those sensor connections through the zone controllers and CVC. However, some UX-focused OSP implementations may continue to connect directly to high-definition displays to support their high bandwidth requirements within the cost constraints of the architecture.

From a software perspective, we are moving toward signal-to-service abstraction, and our acquisition of Wind River gives us access to its VxWorks® real-time operating system and support for Open Container Initiative-compliant containers in safety-critical environments.

As SVA™ technologies evolve, our approach will provide for the fail-operational redundancy needed for Level 3 and above automated driving. With the flexibility that comes from containerization, applications can move from one OSP to a second OSP for redundancy, which the OSP modules installed atop the CVC are ideally suited to support.

When a vehicle is built with these hardware and software foundational elements, the way is clear for realizing the true potential of software-defined vehicles by ensuring that they have the most up-to-date capabilities for years to come.

Built for Exchangeability

Ideally, open server platform (OSP) modules should be designed so that a technician can easily change them out when an upgrade becomes necessary.

MULTI-FUNCTIONAL

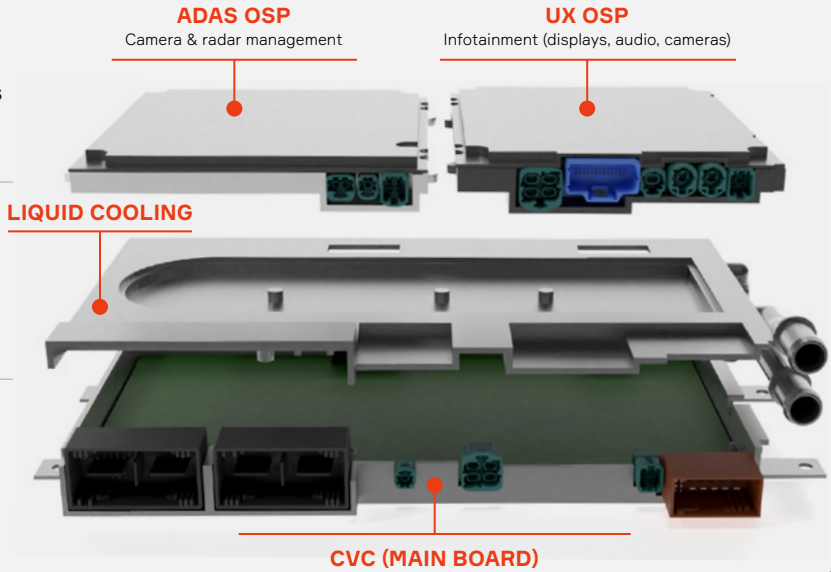
Handles ADAS & UX or provides additional data-acceleration features such as AI/ML, gateway routing, etc.

MODULAR

Exchangeable OSP modules cooled by the CVC unit

EFFICIENT THERMAL MANAGEMENT

Liquid cooling designed to last vehicle lifetime



ABOUT THE AUTHORS

Martin Bornemann
Vice President, Advanced Technology & Architecture,
Chief Technology Office

Martin Bornemann is responsible for the advanced technology and architecture development in Aptiv's CTO office. He has been with Aptiv for more than 20 years, holding positions in innovation management, project management and hardware development. Before joining Aptiv, he designed telecom equipment for Ericsson and conducted wireless LAN research for Bosch.

Benjamin Gould
Product Management Director, Compute
Global Product Organization

Benjamin Gould is responsible for defining Aptiv's next generation of compute, the brain of the software-defined vehicle. Benjamin joined Aptiv in 2021 as chief customer engineer, and prior to that he was senior project manager at Mobileye. Benjamin started his career at Intel, where he spent 20 years in industrial engineering, application engineering, technical marketing and product marketing management. Benjamin has a bachelor's degree in electrical engineering and applied physics from Case Western University.

Cezary Klimasz
Technical Program Manager, Compute
Global Product Organization

Cezary Klimasz leads the design and development of Aptiv's central vehicle controllers and open server platforms. His team is working on next-generation computing platforms leveraging high-performance hardware with state-of-the-art software solutions. Cezary began his career with Aptiv in 2012 as a hardware engineer. Since then he has held various roles in product and technology management.

LEARN MORE AT APTIV.COM/ADAS →