Linaro Automotive Strategy

by Mathieu Poirier

January 2023

Introduction

The amount of technology integrated in the latest generation of automotive products has grown to a point where it defines a significant portion of the vehicle.

A considerable amount of computational power is needed to satisfy the high number of safety critical applications, driver-assisted functionalities and immersive intelligence included in the vehicles. The automotive industry is moving from a distributed architecture where the Software Defined Vehicle (SDV) is supported by dozens of independent engine control units (ECU) to a design that converges on one or two central automotive servers. Key to this transition are the concepts of virtualization and computing domain separation. The former provides workload isolation to applications that used to run on independent ECUs while the latter ensures that real time and safety critical requirements can still be sustained.

Contents

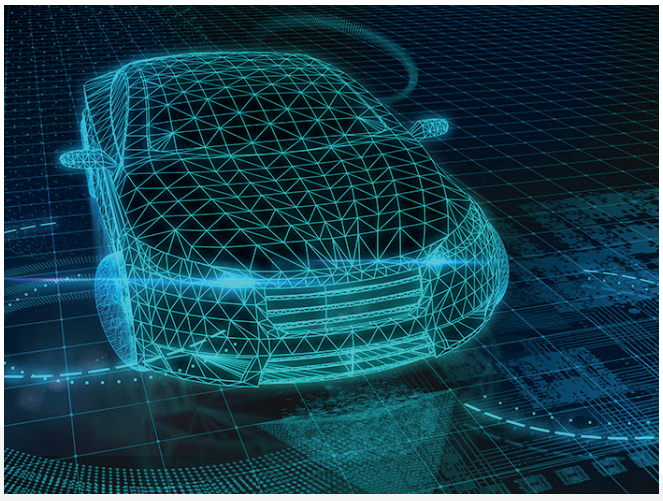
Adopting this new centralized architecture has significantly decreased the work involved in building an automobile but also increased the complexity inherent to integrating the software components that defines it. The process of configuring and integrating critical applications while meeting optimization and safety requirements is a time consuming, complicated and error prone process. The situation is exacerbated further by the proliferation of vendor specific solutions and a lack of standard interfaces allowing virtualized components to communicate.

To address these problems we are proposing to build a reference platform called the Linaro Automotive Stack. The goal of this venture is to reduce the complexity and cost associated with the Software Defined Vehicle by building an infrastructure that is open, standard, secure, hardware independent and compliant with the SOAFEE architecture specification. With the Linaro Automotive Stack, manufacturers will be able to start prototyping and building products from a well defined foundation using a hardware platform that is commonly available. The openness of the solution will provide industry convergence and overall cost reduction by making software components standard and reusable.

More specifically, the solution should be:

Secure: Every aspect of the software running in a vehicle should be attested and authenticated.

Support for mixed criticality: New computing platforms are embedding application specific cores in the same SoC, allowing for the integration of computing domains with different time sensitive and safety critical requirements while mitigating cross-domain impact.



Flexible: The concept of SDV is broad in nature and varies depending on the OEM and the underlying hardware. The software stack integrating the technology components should be flexible and not defined by the capabilities made available by the hardware.

Support real-time computing: ISO26262 defines functional safety levels that can only be achieved with the support of a real time operating system. As such a common automotive solution should accommodate real time processing in a computing domain that differs from other computing domains on the system.

Support for virtualization: This document focuses on virtualized environments. That said, the technologies detailed herein are also directly applicable to baremetal configurations.

Support for time sensitive communications: Whether taking place between the virtualized environment or the safety critical computing domains, communications with external ancillary devices are expected to impose latency requirements that need to be supported.

Architecture agnostic: User defined workload should not be constrained by the processor architecture.

Open and standard compliant: Every aspect of the solution should be open and accessible to community members.

Nota Bene

Although not related to a specific architecture, the remainder of this document uses terminology related to the Arm architecture.

System Topology

The system architecture Linaro is targeting is presented in Figure 1. It is based around the concept of computing domain separation to meet varying degrees of safety and security. As such services with ASIL-D and real time constraints will be handled by cores running a real-time operating system (RTOS) while services with less stringent requirements delegated to A-class clusters. Time sensitive computing domains will not combine hypervisor and RTOS'es to prevent scheduling decisions in the former to impact the real time characteristics of the latter.

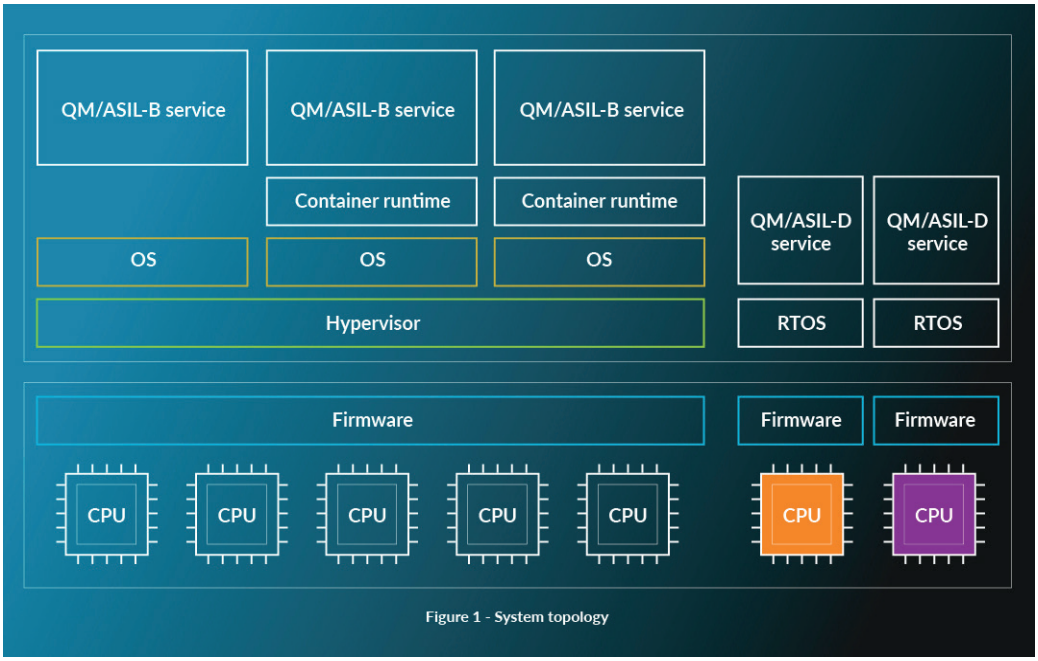


Figure 1 - System topology

Based on the hardware characteristics of the platform, the separation of the computing domains can be made at the architecture level where application specific processors, such as Cortex-M and Cortex-R, are available within a SoC. It can also happen at a logical level where a set of cores is isolated from hypervisor control at boot time.

Components and Technology

What follows is a list of components Linaro will use to build the Linaro Automotive Stack. Unless stated otherwise, we intend to integrate these technologies in our solution.

Firmware

The low level and security components of the Linaro Automotive Stack will be delivered by Linaro's Trusted Substrate firmware. Trusted Substrate is an open source project aligned with Arm's Platform Security Architecture (PSA) and System Ready. It brings standard base secure booting and over-the-air (OTA) updates along with a wealth of security features to the most demanding embedded environments such as automotive and industrial. It has a well defined roadmap that promotes the evolution of the solution in order to maintain compliance with Arm's standardisation and certification programs.

The hypervisor, operating system, container and user space programs are added as separate layers on top of the trusted substrate, making the Linaro Automotive Stack solution architecture agnostic.

Hypervisor

The Linaro Automotive Stack will use the Xen hypervisor as a default option. As a type-1 hypervisor, Xen provides a foundation to explore and experiment with, potentially highlighting shortfalls and issues to address in the early stage of development. It is a mature open source project that

includes interesting features such as static VM allocation, private guest memory space and cache coloring. Other solutions such as L4RE will be considered if there is interest from our members or if serious drawbacks with Xen come to surface.

VM-to-VM Communication

As stated above the concept of virtualization provides the benefit of integrating the functionality of historically disjointed processing units onto a single SoC. A side effect of this integration is the communication between processing units that would normally happen over dedicated wires now has to take place between VMs, which is orthogonal to the idea of computing domain separation brought forward by virtualization. The primary option to address this issue is Argo, a hypervisor mediated exchange allowing for communication between VMs to take place. Argo, as any other VM-to-VM communication mechanism, is tightly coupled with the hypervisor and the virtual machine manager (VMM). Its maturity, security and performance have to be assessed meticulously.

VirtIO

Linaro intends to tackle the lack of conformance and standardization in current automotive solutions by making heavy use of virtual I/O (virtIO). VirtIO is an OASIS specification describing how hardware devices are discovered, managed and accessed in virtual environments. Those hardware devices are then presented to guest operating systems as virtual devices, also known as virtio-devices. Virtio-devices are made of a frontend driver, a backend driver and a shared memory area called virtual queues (Figure 2). The nature of the information shared on the virtual queues is specific to each virtio-device type and is

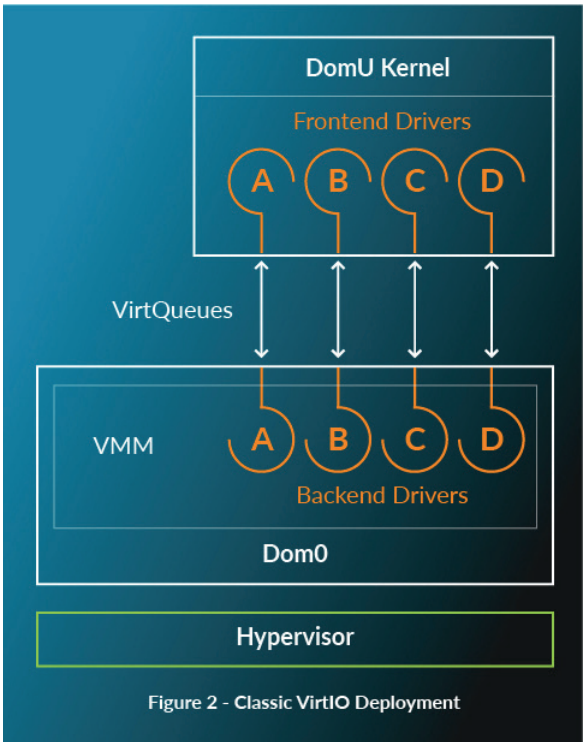


Figure 2 - Classic VirtIO Deployment

also specified on OASIS. By following the virtIO and virtio-device specifications, it is possible to write driver implementations that are independent of architecture and operating system.

The work associated with the transfer of data through virtual queues may come with a performance cost that can, for some specific use cases, be too high. Other methods such as virtual function I/O (VFIO) offer baremetal like performance but lose most of the flexibility and generic interfaces offered by virtio-devices. Unless mandated in very specific cases, Linaro does not intend to use VFIO in this design.

Vhost-User

As presented in Figure 2, backend drivers are usually implemented as part of the virtual memory manager. This situation is not optimal since the same backend drivers have to be re-implemented in different VMMs, introducing code duplication and varying degrees of supported functionality.

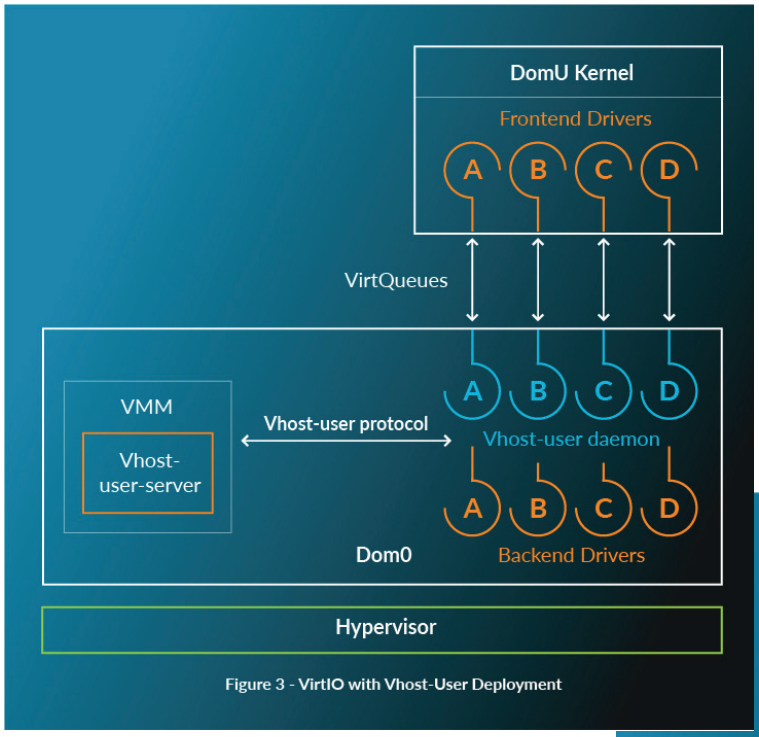


Figure 3 - VirtIO with Vhost-User Deployment

Linaro intends to address this problem by replacing the backend drivers in the VMMs with a vhost-user-server. The backend drivers will then be converted to stand alone user space services called vhost-user daemons, using the vhost-user protocol to share information about virtual queues between them (Figure 3). That way, backend drivers become separate entities that can be implemented and maintained independently from the virtual machine managers, focusing development and bug fixes in a central location. It also allows to change the VMM as required without needing to implement new backend drivers.

Linaro has done a lot of work with vhost-user compatible backend drivers. Interoperability has been demonstrated with QEMU, Xen and the Gunya hypervisors. We also intend to supplement missing vhost-user daemons with work found in the QEMU project where command line options have been added to instantiate vhost-user daemons rather than virtual machines, as presented in Figure 4.

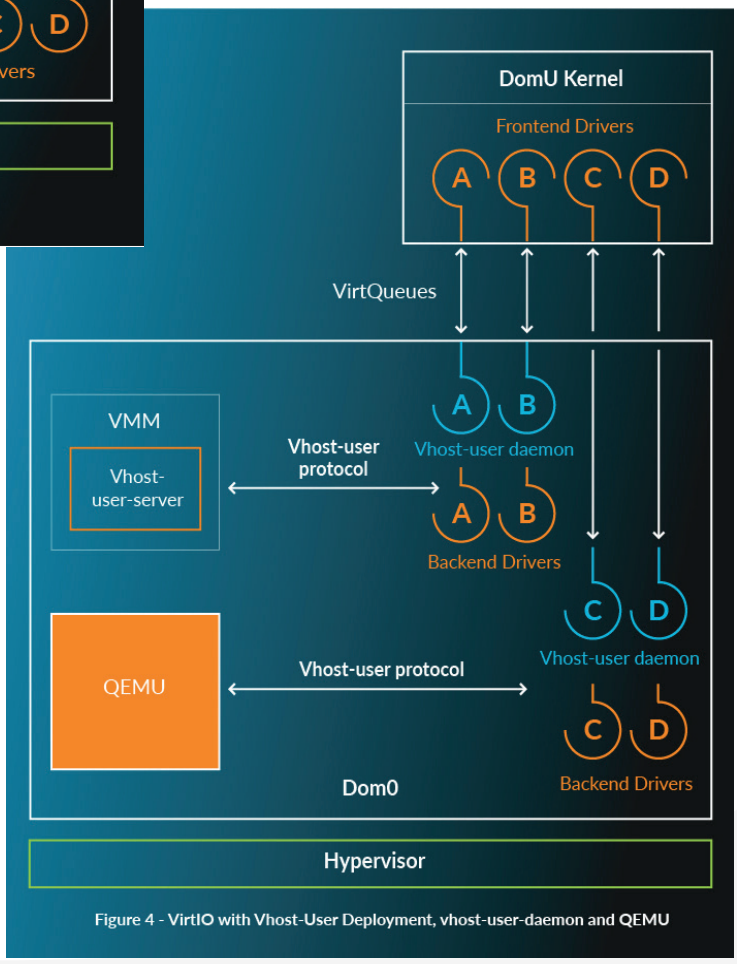


Figure 4 - VirtIO with Vhost-User Deployment, vhost-user-daemon and QEMU

VirtIO and Safety Critical Computing Domains

Safety critical workloads will be executed on CPUs that are completely disjointed from the main cluster units in order to meet latency and certification requirements. In such an architecture, hardware devices connected to the safety critical domains such as a camera and a speed sensor are not expected to be accessible by CPUs in the virtualized environment to avoid resources contention (Figure 5).

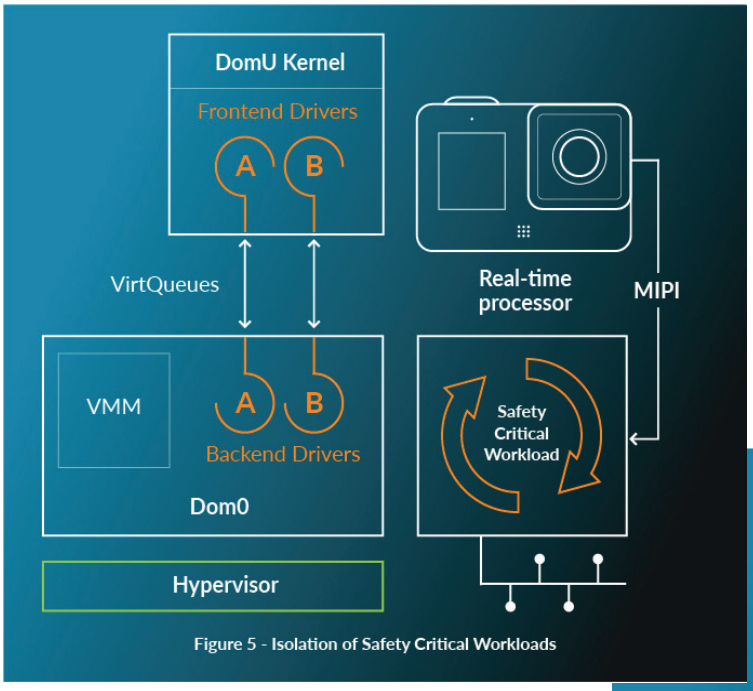


Figure 5 - Isolation of Safety Critical Workloads

Although desirable from an isolation point of view, several use cases are calling for the consumption of data collected from these devices by workloads in the virtualized domain. As such, it is easy to envision a scenario where input from a camera and speed from a sensor are rendered on a cockpit application.

In order to standardize input generated by devices in isolated domains, Linaro intends to work with its members to bring the virtIO protocol used by the Linux remoteproc subsystem up to the same standards as the virtIO protocol used in virtualized environments. From there, accessing devices connected to an isolated processor or a host system will be done using the same frontend drivers and virtIO protocol, as presented in Figure 6.

Using virtio-devices to format input from isolated devices is also key when considering cloud native development, as presented later in this document.

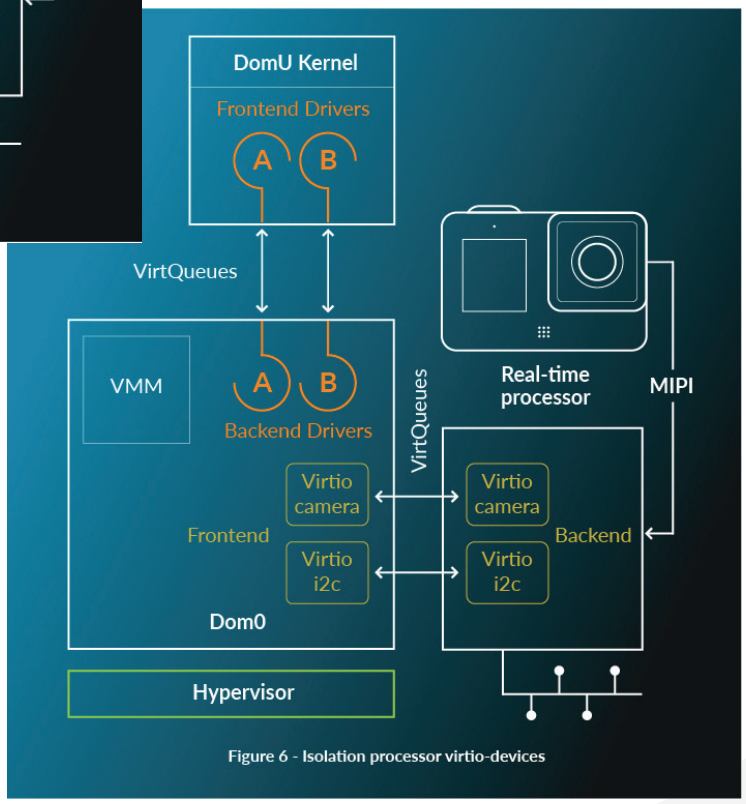


Figure 6 - Isolation processor virtio-devices

Bridging the Gap with Virtio-bridge

The data generated by devices confined to the safety critical domain is consumed by user applications running in virtual machines, and not at the host level as presented in Figure 6. To address this situation Linaro has been experimenting with the concept of "virtio-bridge" where front end drivers in a virtual machine are accessing backend drivers in the safety critical domain using virtual queues and a virtio-bridge to relay interrupts, as shown in Figure 7. Details on that concept are beyond the scope of this document but can be expanded further upon request.

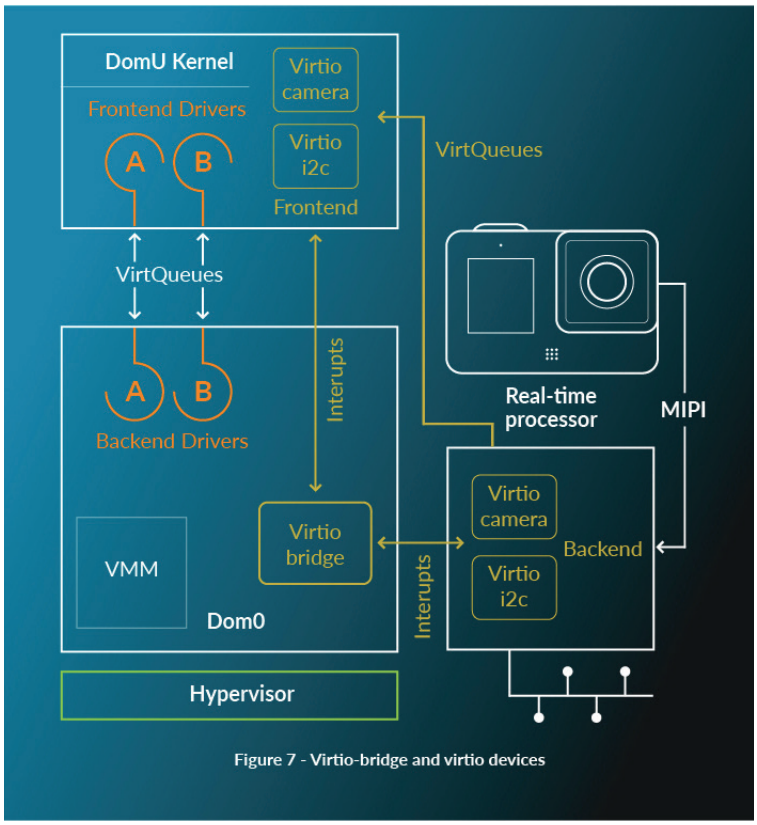


Figure 7 - Virtio-bridge and virtio devices

Operating Systems

Real-Time

Safety critical and time sensitive requirements in automotive systems demand a high level of certifications and almost invariably mandate using an operating system that can satisfy hard real time constraints. Linaro will partner with its members to provide a sample real time solution or use the Zephyr operating system. Depending on latency requirements, it may also be possible to use the Linux kernel with the PREMPT_RT patchset applied. Although not certified as hard real time, this option provides enough real time capability to make a solid proof of concept. As stated earlier, time sensitive processing should be done on dedicated CPUs rather than mixed with a virtualized environment. The latter introduces scheduling delays and interrupt delivery latencies that are incompatible with safety critical requirements.

Non Real-Time

For non real-time workloads, Linaro intends to use the Linux kernel and a generic root file system. Guest VMs will also be oriented toward Linux but that choice is only for demonstration purposes - different operating systems can be selected based on the application workloads being executed.

Containers

Container Runtime

Regarding container runtime, Linaro has decided to favour podman. Unlike Docker, which is a monolithic application covering all aspects of the containerization process, podman is a set of command line tools targeting specific functions. Tools are conformant with the Open Container Initiative (OCI) standard, making them compatible with Docker and podmand containers alike. It offers a wealth of interesting features but most importantly and contrary to Docker, it endorses a daemon-less approach. Daemons have elevated privileges and are a primary target for attackers. Other container technologies such as Kata and RunX will be considered should podman not perform as expected.

Container Attestation

Linaro is presently assessing various strategies for container attestation as part of the Trusted Substrate program. This topic is especially tedious to address since no standard method has emerged, with OEM and solution vendors coming up with their own proposal. We intend to work with our members to find an open solution that can be integrated to the Trusted Substrate program and subsequently inherited by the Linaro Automotive Stack.

Application Services

In partnership with our members, Linaro's automotive blueprint program has been integrating The Autoware Foundation's open autonomous driving project (OpenADK) with the AVA platform. OpenADK is an open source undertaking that provides a software stack for self-driving vehicles. It includes modules for localization, object detection, route planning and mechanical controls (Figure 8).

Integration of OpenADK in the Linaro Automotive Stack is provided as a starting point and offers a good benchmark for what an automotive stack should provide to the workloads they host. Other projects such as Android Automotive will also be integrated at the application service level.

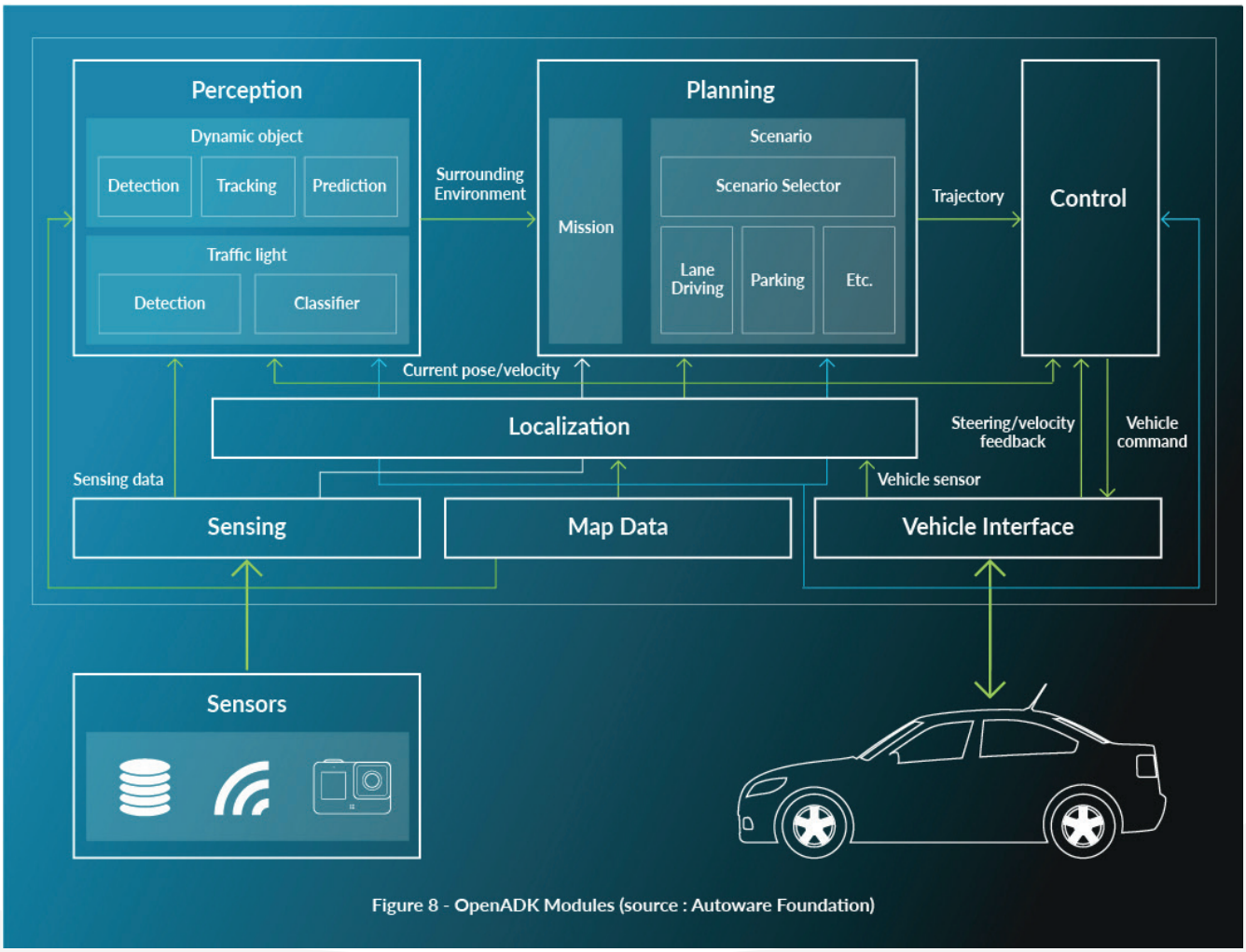


Figure 8 - OpenADK Modules (source : Autoware Foundation)

Time Sensitive Networking

With the convergence of functionality toward central processing servers, the connections between critical sensors and ECUs is being replaced by Time Sensitive Networks (TSN) capable of service guarantees and deterministic latency. As such the challenge is not to receive the information but to make sure it is processed in a timely manner.

TSN endpoints can be managed at the hypervisor level or by CPUs running real-time operating systems, depending on the hardware configuration. In the former case, controlling the latency introduced by the delivery of interrupts from the hypervisor to the virtual machines can be complex and tightly coupled to a specific implementation, as described in this Linaro blog. Another solution is to route data packets to a real-time core and use virtio-devices to convey information to the virtualized domain, following the strategy introduced in the above section on VirtIO and Safety Critical Domains.

Regardless of the strategy, properly integrating TSN in automotive products is highly dependent on the use cases to cover and the capabilities of the hardware platform.

Cloud Native Development

The Linaro Automotive Stack is entirely based on interactions with virtio-devices, making it easy to port the solution to a cloud native environment. In such a scenario a machine specification script that is identical to the target hardware platform is instantiated by a VMM running in the cloud environment. Once booted, this virtual platform can support user space elements in a way that is identical to the target hardware.

The bulk of the work resides in building a virtual machine configuration that closely replicates inputs that normally come from hardware devices, something that can be difficult when it comes to secure devices and real time processing. Other types of actuators like a camera feed or a pre-processed driving scenario can be fetched from files and transmitted via virtio-backend drivers the same way they originate from actual hardware devices. Although interesting on many fronts, cloud native development is a long term goal and not part of the immediate strategy.

SOAFEE Compliance and CI/CD

The SOAFEE gitlab repository includes a test suite to determine if a software stack is compliant with the specification. That test suite has been integrated in Linaro's Trusted Reference Stack (TRS) quality assurance roster and is being run as part of the continuous integration (CI/CD) process engineered by Linaro. The continuous integration infrastructure is built on top of TuxBuild, Linaro's massively parallel compilation engine and LAVA, Linaro Automation and Validation Architecture.

The immediate availability of this test architecture makes it easy to concentrate on new test suites specifically tailored for the Linaro Automotive Stack and quickly isolate problems as they arise during the development process.

Taking the Next Step

Building on prior experience from the mobile industry, Linaro strongly believes in adopting a well defined software platform that is standards compliant, secure and open source as key to managing the cost and complexity the automotive industry is transforming to. As a result, Linaro has aligned itself with the SOAFEE SIG to steer the industry into a sustainable software ecosystem built on open collaboration solving key engineering challenges.

Linaro has been working on the technologies mentioned in this document for several years. We are taking the next step by integrating these components in the Linaro Automotive Stack. We invite you to join the conversation and collaborate on this project by contacting Linaro at contact@linaro.org.