ELISA
Enabling Linux in Safety Applications

WORKSHOP

Example System within ELISA as Cross Community Effort

with AGL, Eclipse S-Core, KernelCI, Xen, Zephyr, and more

Philipp Ahmann (ETAS), Yuichi Kusakabe (Honda Motors)

May 7-9, 2025

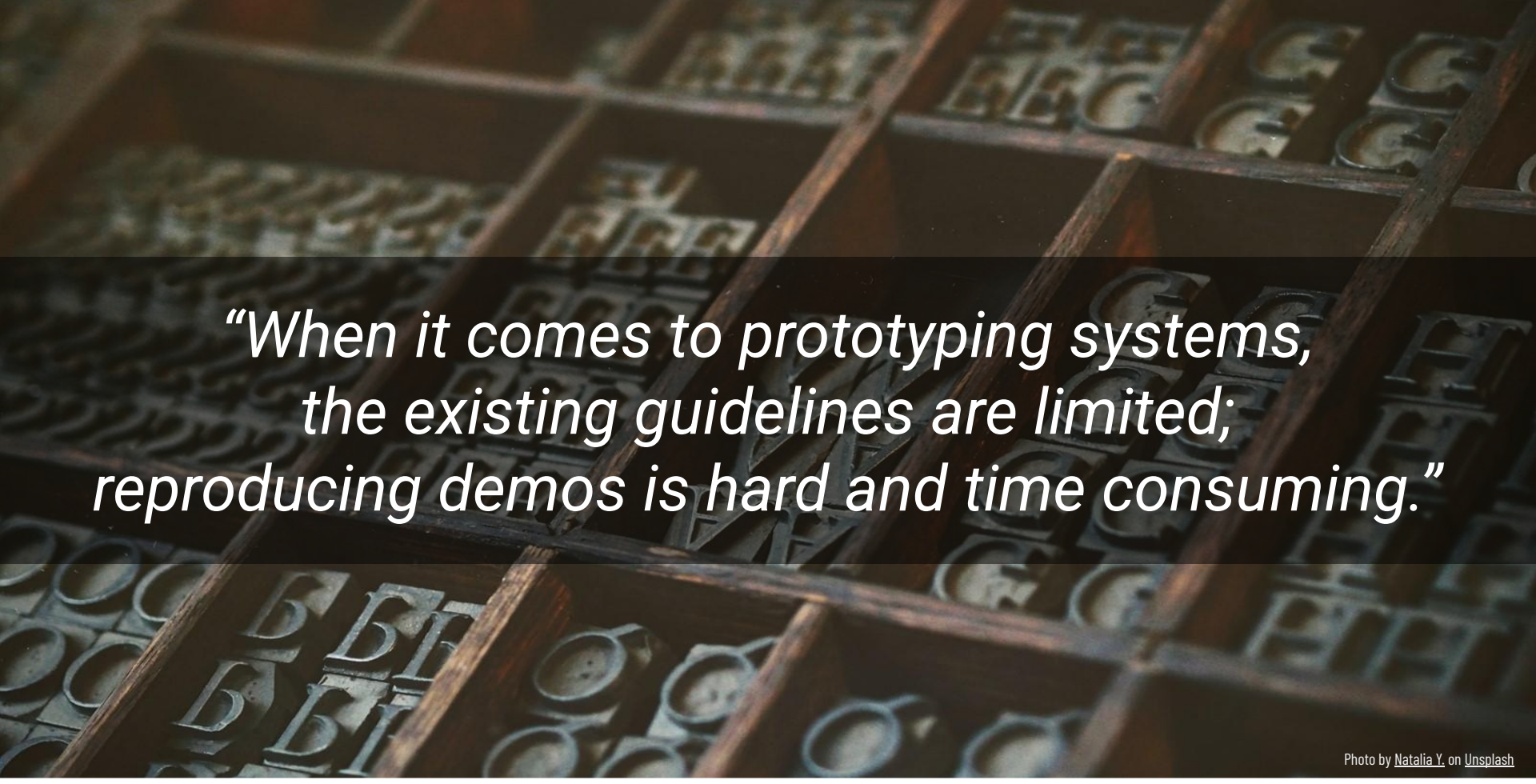


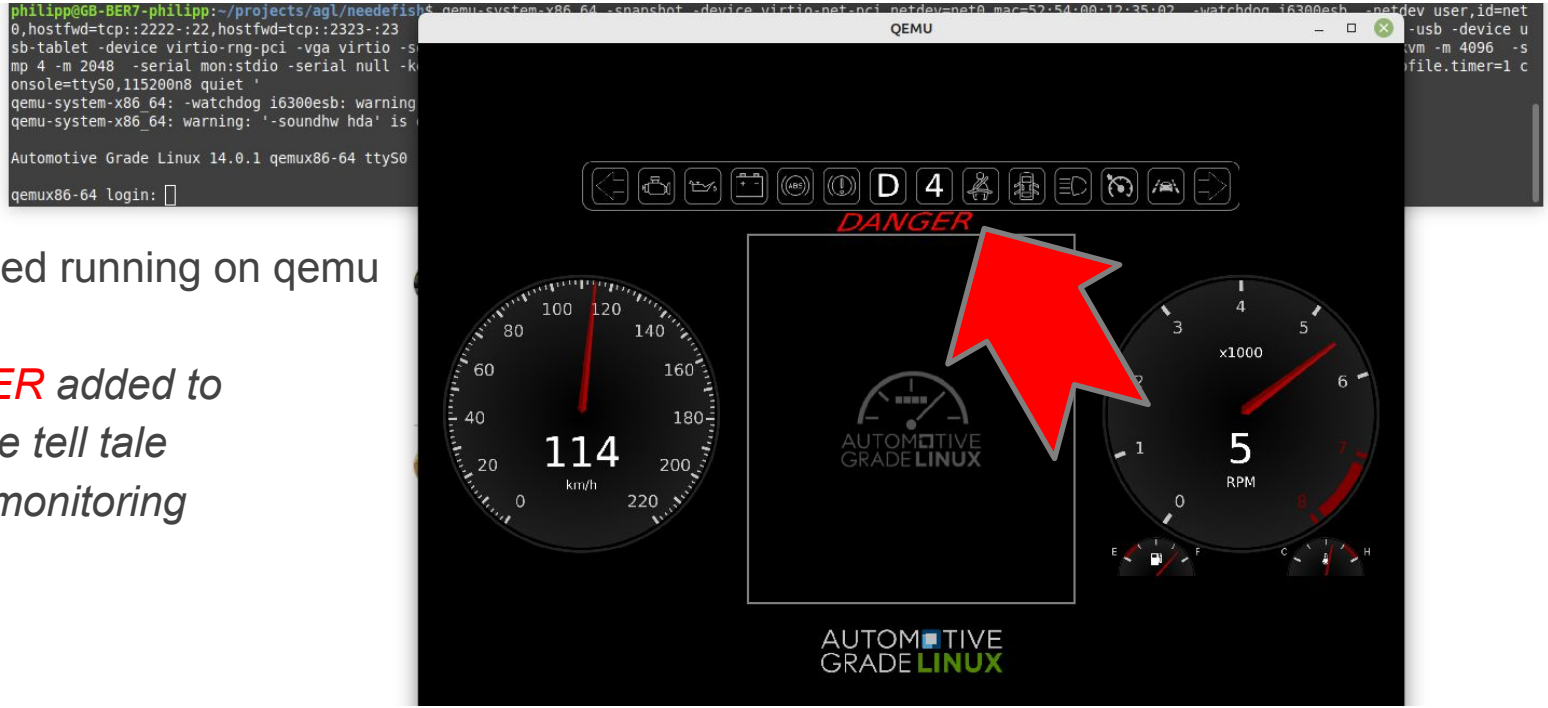
Photo by Natalia Y. on Unsplash

meta-elisa from Automotive WG

ELISA
Enabling Linux in Safety Applications

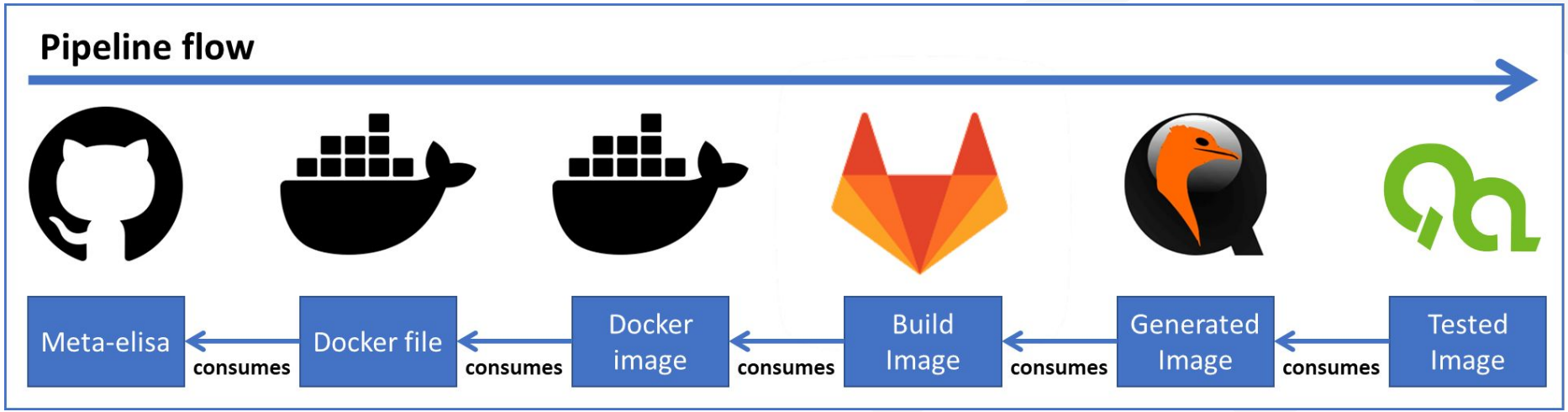
WORKSHOP

AGL Instrument Cluster Enhancements



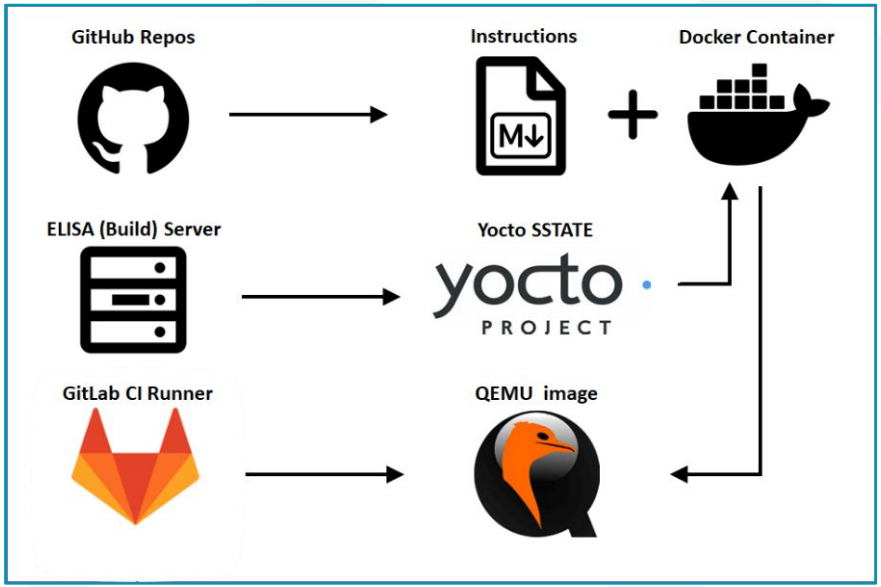
- QT based running on qemu
- *DANGER added to illustrate tell tale safety monitoring*

meta-elisa: Pipeline Dependencies (broke last year)

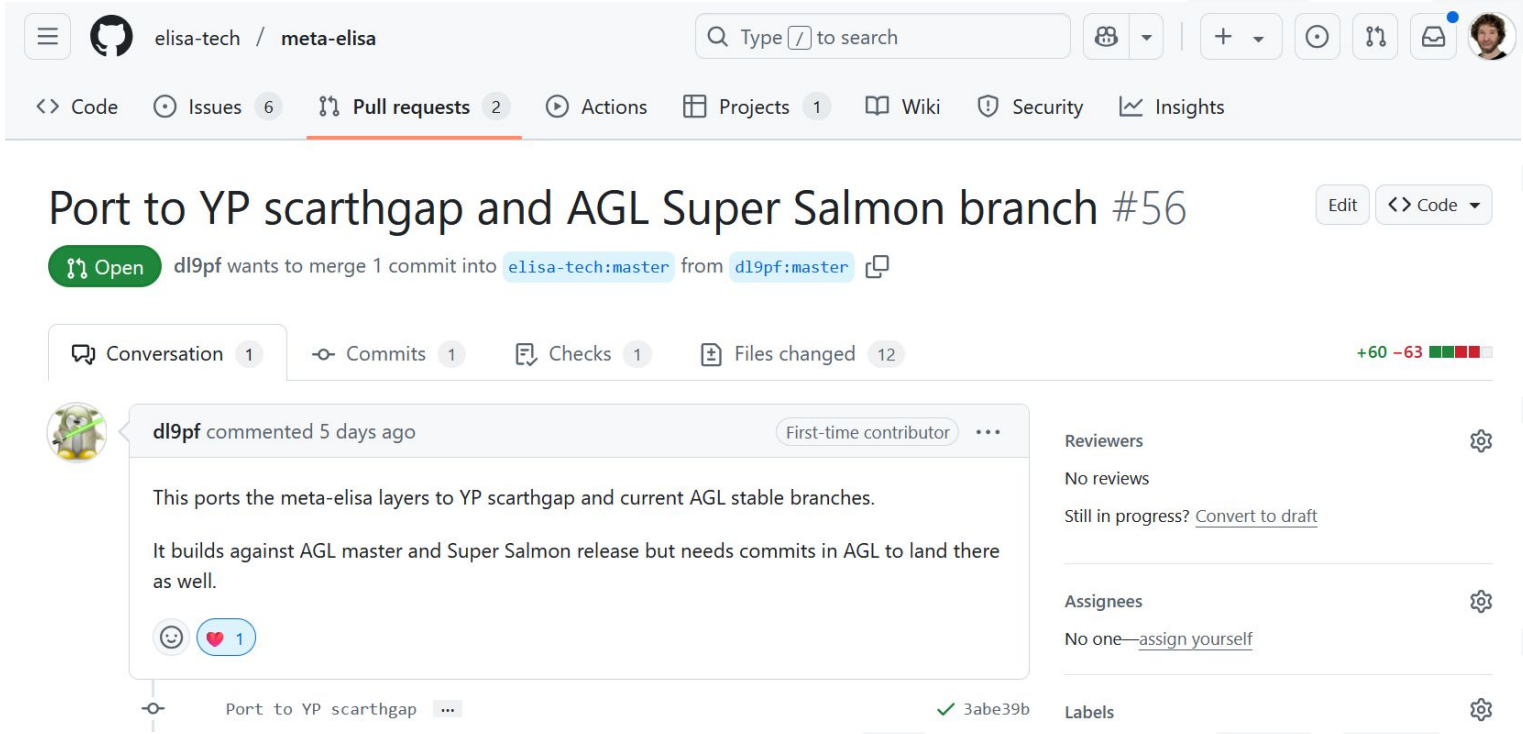


meta-elisa: Various Starting Points Provided

- Plain and native from source
  https://github.com/elisa-tech/meta-elisa
- Using docker container
  https://github.com/elisa-tech/wg-automotive/tree/master/Docker_container
- With cached build using SSTATE
  modify “`conf/local.conf`“ after the "`source`" command before the "`bitbake`” command
- Download binaries directly from build server
  https://gitlab.com/elisa-tech/meta-elisa-ci



Porting meta-elisa to latest AGL release



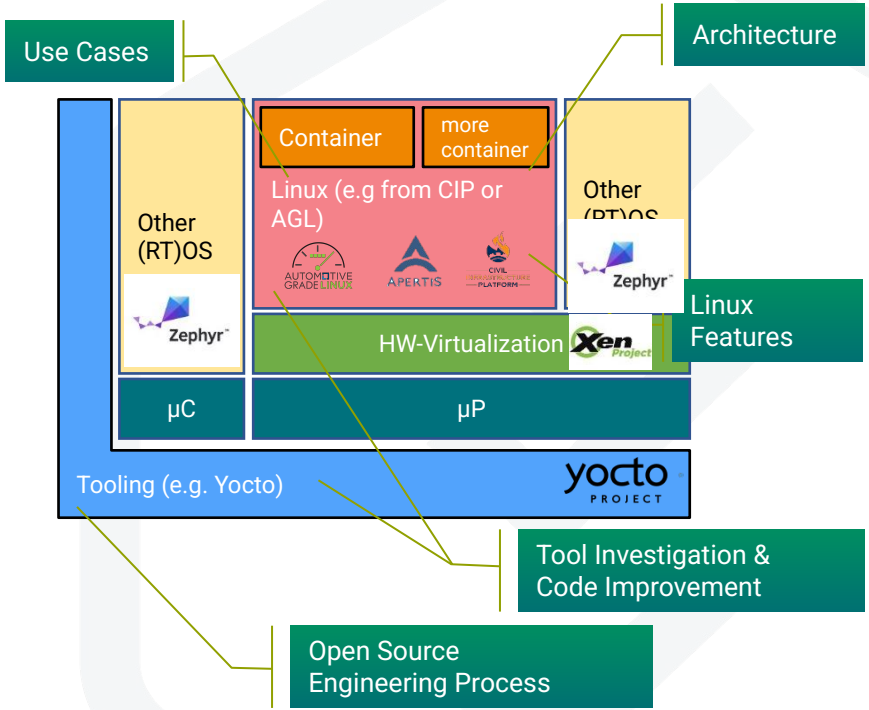
ELISA
Enabling Linux in Safety Applications

WORKSHOP

Existing Example Systems WG

ELISA Working Groups - Fit in an Exemplary System

- **Linux Features**, **Architecture** and Code Improvements should be integrated into the reference system directly.
- **Tools** and **Engineering process** should serve the reproducible product creation.
- **Medical, Automotive, Aerospace** and future WG use cases should be able to strip down the reference system to their use case demands.



Interactions Between the Communities

- Open source projects focusing on safety-critical analysis

- Open source projects with safety-critical relevance and comparable system architecture considerations

- Further community interactions

"If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple.
But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas

— George Bernard Shaw

COLLABORA

Example System - At Embedded World 2024

- Xilinx ZCU102 running Xen, Zepyhr, Linux
- Qemu version also exists

- Software built in ELISA CI
- Focus on reproducibility
- Examples provided as base for extension

- Detailed documentation available

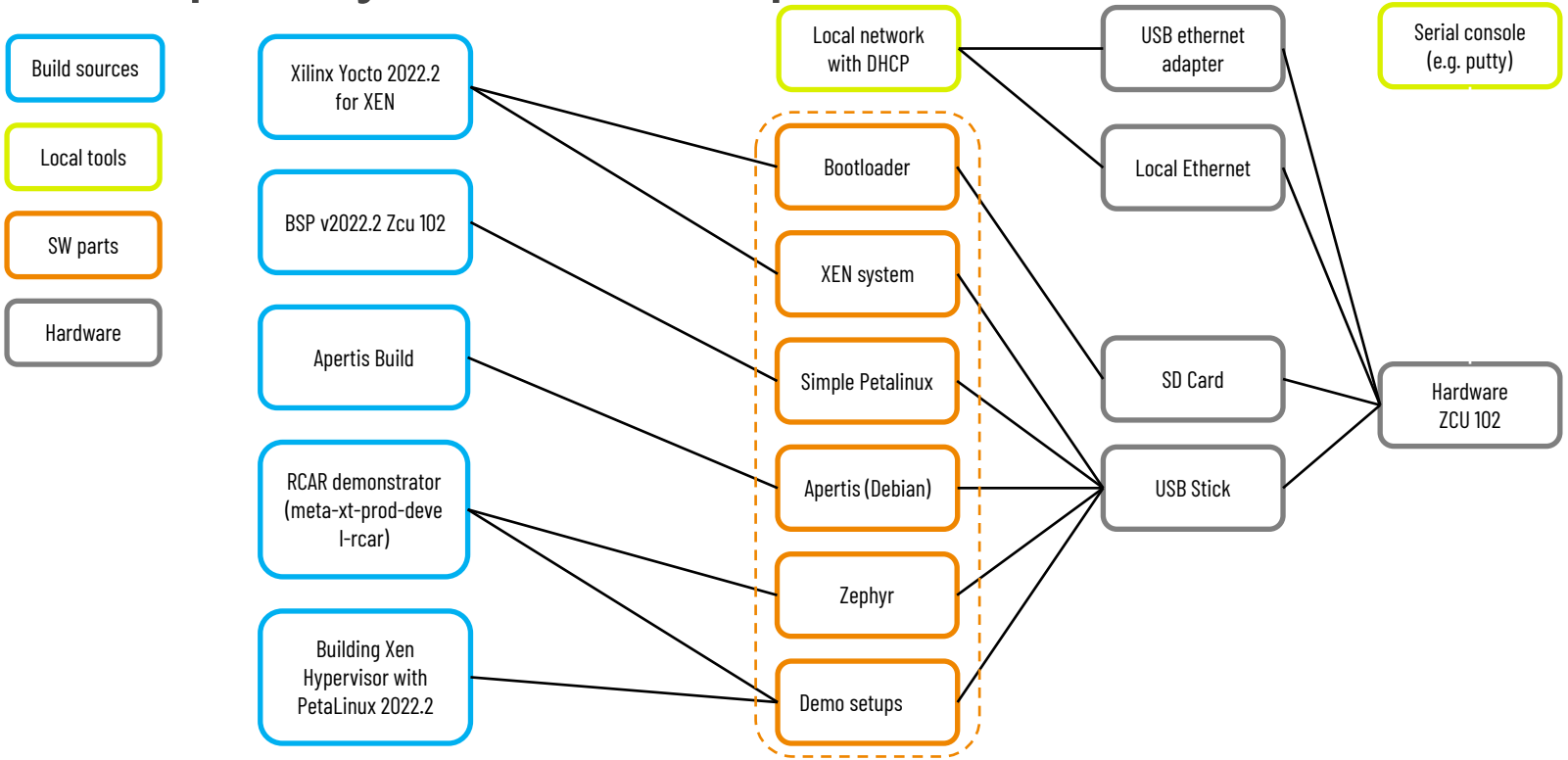
Looking for new hardware in 2025
Candidates: ARM-qemu, Xilinx Kria, Pi5, …
(open for your suggestions and contribution)

https://elisa.tech/blog/2024/04/09/elisa-project-at-embedded-world/

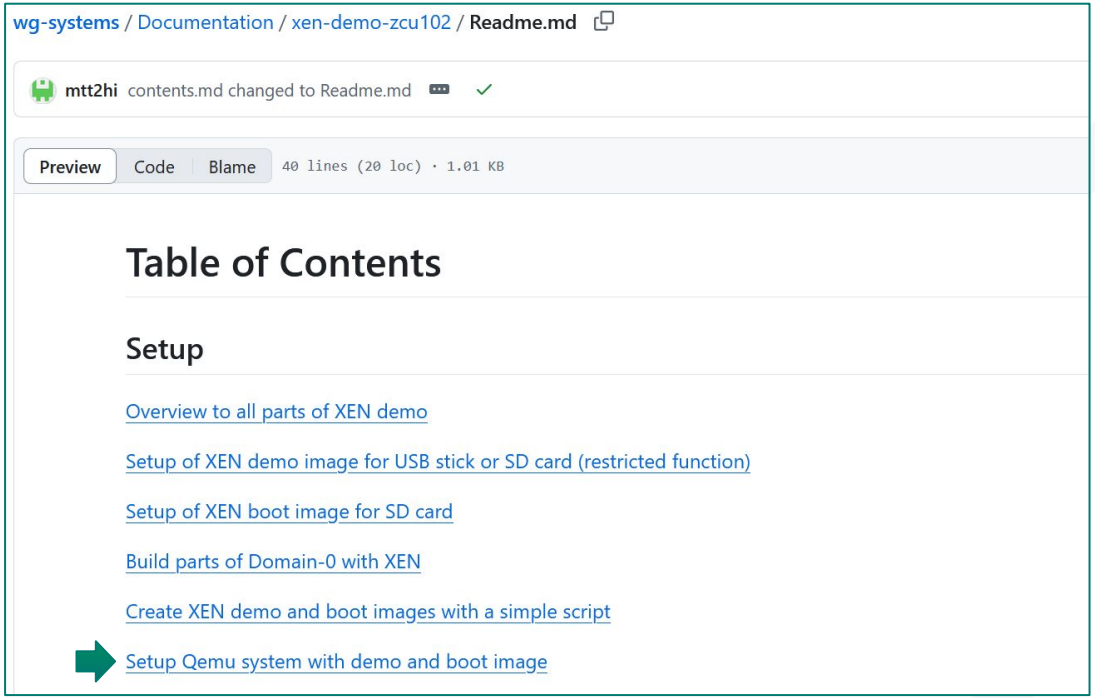
ELISA
Enabling Linux in Safety Applications

WORKSHOP

Example System - Composition



Example System - Reproducibility & Documentation



https://github.com/elisa-tech/wg-systems/blob/main/Documentation/xen-demo-zcu102/Readme.md

AGL SDV EG System

About myself(Yuichi Kusakabe)

HONDA
The Power of Dreams

- 2005/4 Automotive Tier1 in Japan
  - Renesas H8S AVN power control(Non-OS)
  - Panasonic UniPhier USB-Audio(Fastest in the industry with HDD transfer function)
- 2009/12 Working together with in Japan OEM
  - Engaged in the R&D of SoC for next-generation IVI and multimedia OS
- 2011/1 Paradigm shift to use OSS
  - Renesas RMA1 Linux-PF Project Leader, AGL Board Member
  - Renesas R-Car M2/H3/M3 Fast Boot, Java, HTML5 browser
- 2020/6 Join Honda Motor in IVI software development team
  - Qualcomm SA8155P AAOS based IVI Lead Architect
- 2024/4 Launched Honda OSPO(Open Source Program Office)
  - SDV(Software Defined Vehicle) SW Chief Architect and OSPO Tech Lead

SPDX LITE

OPENCHAIN 2.1 ISO/IEC 5230:2020 Open Source License Compliance Program

OPENCHAIN 1.1 ISO/IEC DIS 18974 Open Source Security Assurance Program

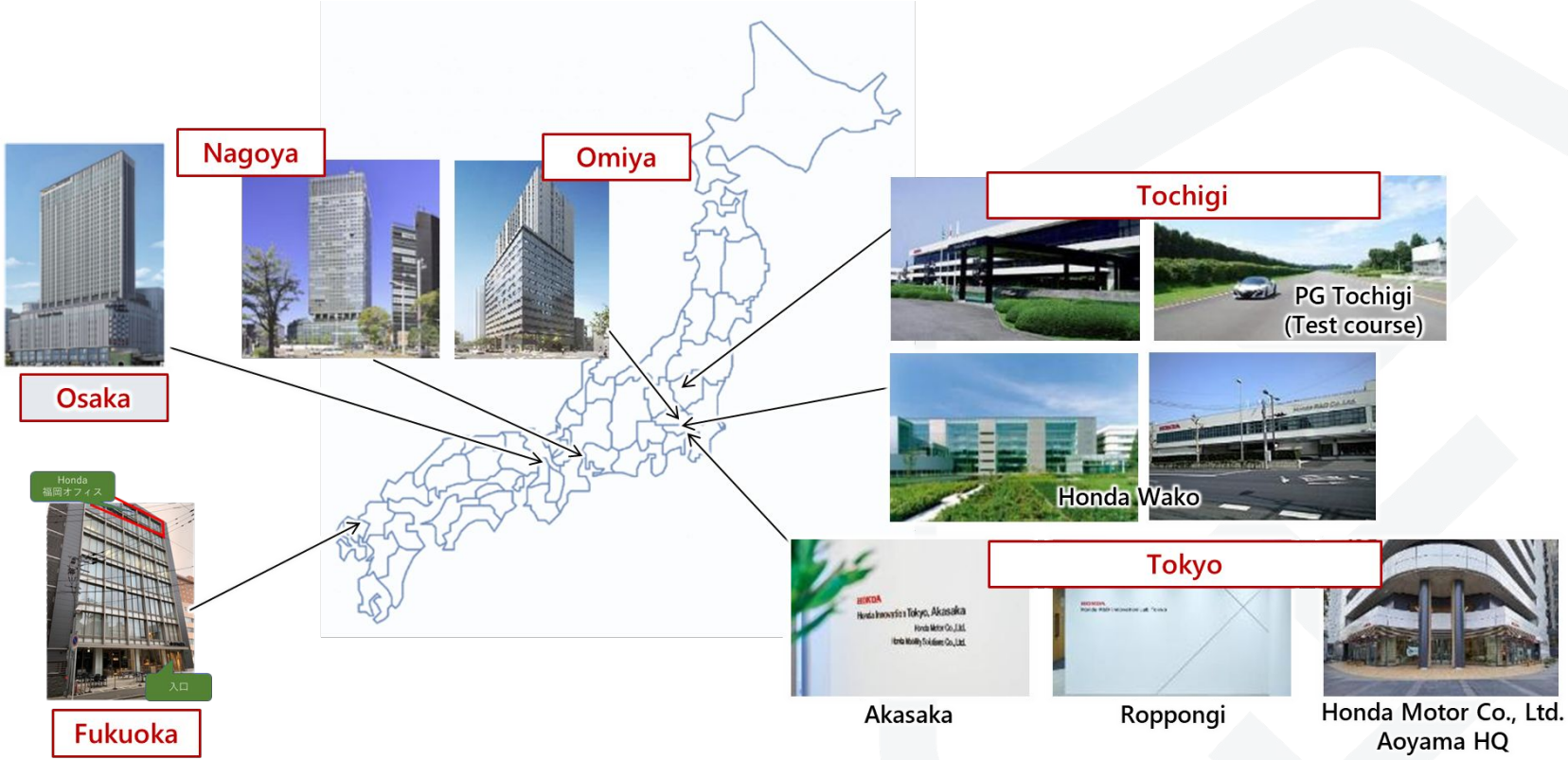
AUTOMOTIVE GRADE LINUX

ELISA
Enabling Linux in Safety Applications

ELISA
Enabling Linux in Safety Applications

WORKSHOP

Honda Software Center in Japan

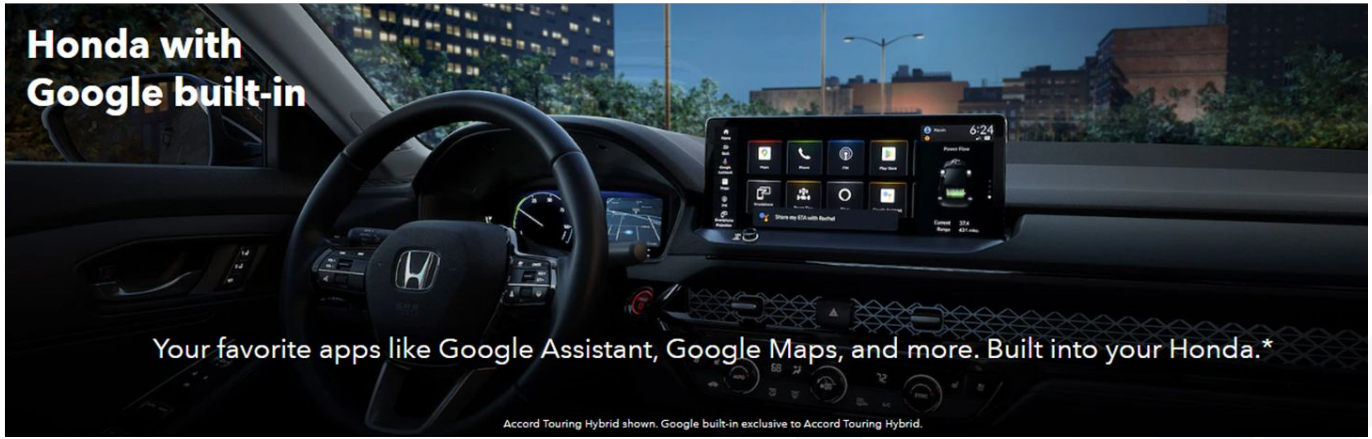


23M/Y US ACCORD IVI system

■Achievement

The First-Ever Honda with Google built-in. The 2023 Accord Touring Hybrid with Google built-in also offers an electrifying hybrid powertrain, 12-speaker Bose premium sound system, Head-Up Display, and more.

2023 Honda Accord Google Services - YouTube



AGL SDV Reference PF

What is Automotive Grade Linux?

Automotive Grade Linux is a collaborative, open source project that brings together automakers, suppliers, and technology companies for the purpose of building Linux-based, open source software platforms for automotive applications that can serve as de facto industry standards.

AGL address all software in the vehicle: infotainment, instrument cluster, heads-up-display (HUD), telematics, connected car, advanced driver assistance systems (ADAS), functional safety, and autonomous driving.

Adopting a shared platform across the industry reduces fragmentation and allows automakers and suppliers to reuse the same code base, which leads to rapid innovation and faster time-to-market for new products.

AGL is a Linux Foundation project and its goals are as follows:

- Build a single platform for the entire industry
- Develop 70 to 80% of the starting point for a production project
- Reduce fragmentation by combining the best of open source
- Develop an ecosystem of developers, suppliers, and expertise that all use a single platform

You can find additional overview information on the "About Automotive Grade Linux" page. You can find information on the AGL Unified Code Base on the "Unified Code Base" page.

Example of SDV(Software Defined Vehicle) system

Virtual, digital twin development

Out-Car

Virtual development environment

Data utilization base

Data driven development

Utilize standard architecture/ technology/tool

Hardware-less development/ verification environment

End To End environment including other ECUs

OTA Manager

Cloud Connector

Safety App

In-Car Core ECU

API

Vehicle Data Lake

Service

Container

Container

CP

CP

OS

Adaptive-Core

Classic-Core

UX app, high-speed evolution

Real time, safety

Soft area: provision area of most of the services

Hard area: with knowledge and actual results

Enhance development efficiency by automation

Data driven development/ analysis environment

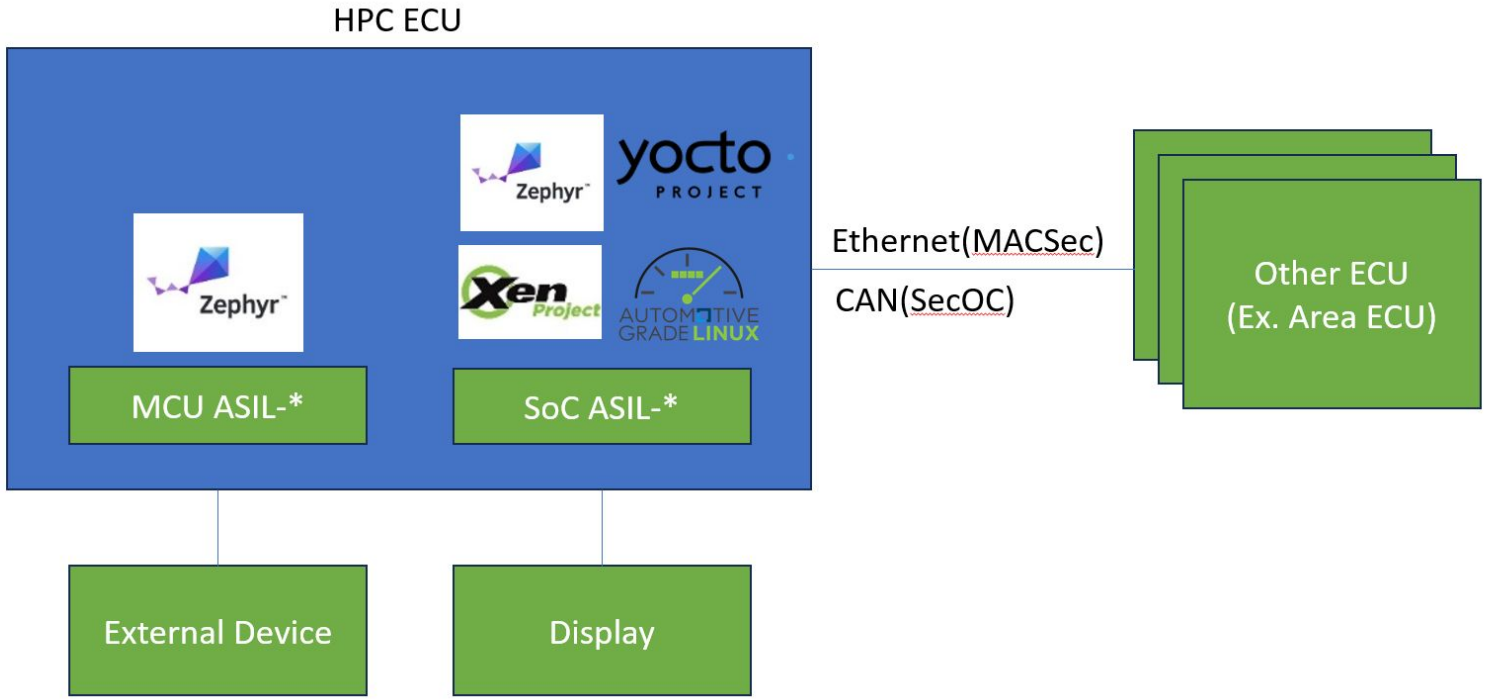
Cross-generation SW development (shift to one-branch)

ELISA Enabling Linux in Safety Applications

WORKSHOP

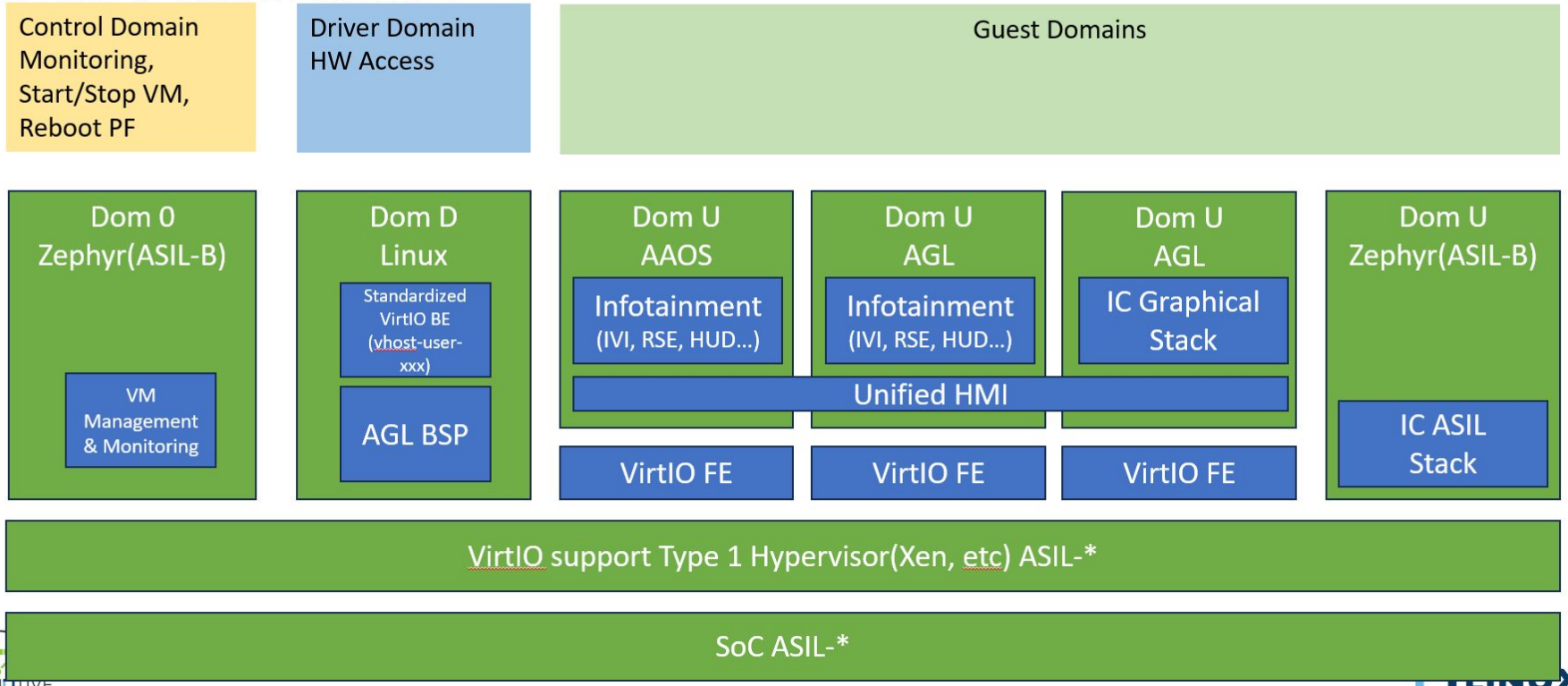
https://sched.co/1eKvl

Example SDV System Block Overview



SDV Architecture Overview(SoC ASIL-A/B)

AGL SDV Reference PF



Some OSS key components of SDV

VSOMEIP

vSomeIP

Overview: The vSomeIP stack implements the http://some-ip.com/ (Scalable Service-Oriented Middleware over IP (SOME/IP)) protocol.
The stack consists of:

- a shared library for SOME/IP (libvsomeip3.so)
- a shared library for SOME/IP's configuration module (libvsomeip3-cfg.so)
- a shared library for SOME/IP's service discovery (libvsomeip3-sd.so)
- a shared library for SOME/IP's E2E protection module (libvsomeip3-e2e.so)

Optional:

- a shared library for compatibility with vsomeip v2 (libvsomeip.so)

Lead: Gonçalo Almeida (Critical Techworks), Diogo Pedrozza (Critical Techworks)

Resources:

- Project page

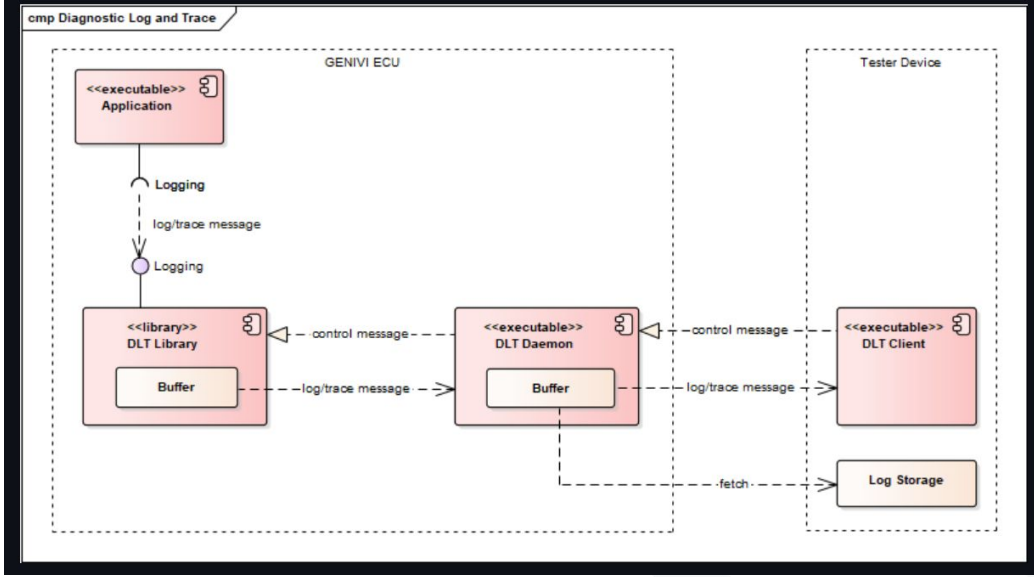
https://covesa.global/project/vsomeip/

DLT(Diagnostic Log and Trace)

Overview

COVESA DLT provides a log and trace interface, based on the standardised protocol specified in the AUTOSAR Classic Platform R19-11 DLT. It is used by other COVESA components but can serve as logging framework for other applications without relation to COVESA.

The most important terms and parts are depicted in the following figure. Please refer to Glossary for a full overview over DLT-specific terms.



https://github.com/COVESA/dlt-daemon

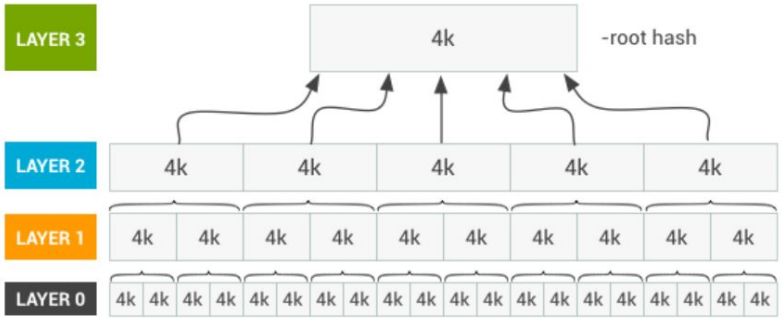
Verified Boot

Android 4.4 and higher supports Verified Boot through the optional device-mapper-verity (dm-verity) kernel feature, which provides transparent integrity checking of block devices. dm-verity helps prevent persistent rootkits that can hold onto root privileges and compromise devices. This feature helps Android users be sure when booting a device it is in the same state as when it was last used.

Potentially Harmful Applications (PHAs) with root privileges can hide from detection programs and otherwise mask themselves. The rooting software can do this because it is often more privileged than the detectors, enabling the software to "lie" to the detection programs.

The dm-verity feature lets you look at a block device, the underlying storage layer of the file system, and determine if it matches its expected configuration. It does this using a cryptographic hash tree. For every block (typically 4k), there is a SHA256 hash.

Because the hash values are stored in a tree of pages, only the top-level "root" hash must be trusted to verify the rest of the tree. The ability to modify any of the blocks would be equivalent to breaking the cryptographic hash. See the following diagram for a depiction of this structure.



https://source.android.com/docs/security/features/verifiedboot/dm-verity

MACSec

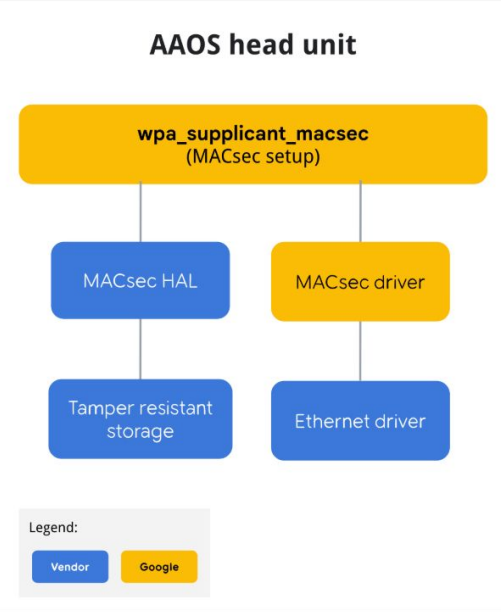
Enable MACsec for Ethernet features

This page explains how to enable MACsec for Ethernet features.

Use MACsec to authenticate and encrypt the Ethernet communication used by in-vehicle infotainment (IVI) for different ECU units, protecting the data from tampering, replay, or information disclosure. Do this buy enabling MACsec IEEE 802.11AE for the Ethernet network.

Overview

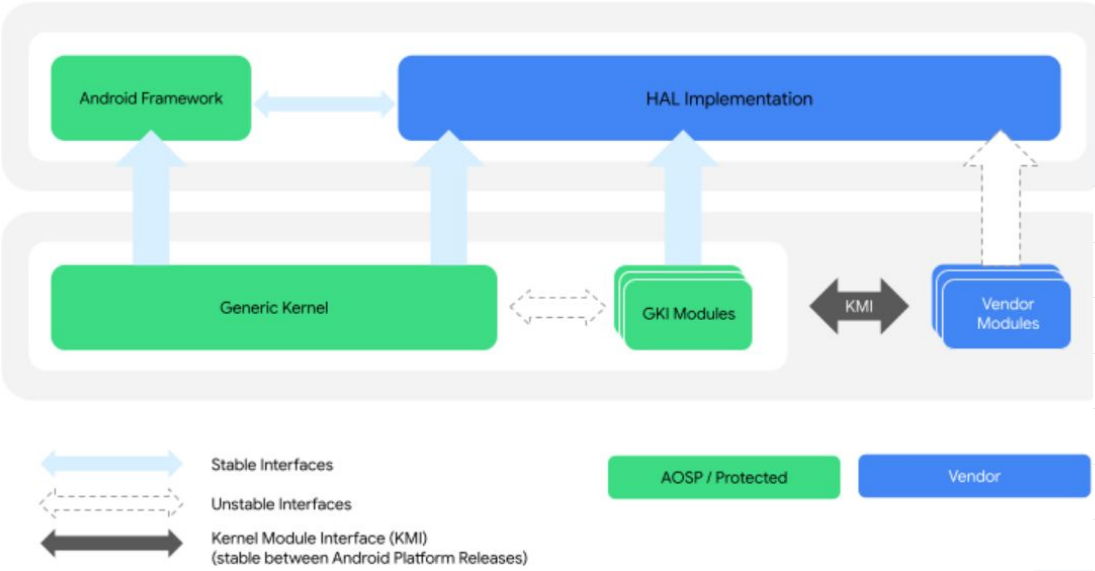
To enable MACsec, `wpa_supplicant` is used as the daemon for handling the MACsec Key Agreement (MKA) handshake. A MACsec HAL is defined for storing the MACsec pre-shared-key, called the connectivity association key (CAK) securely. The MACsec HAL only supports CAK. This vendor-specific MACsec HAL stores the CAK securely in a tamper resistant storage. Provisioning the key depends on the vendor implementation.



https://source.android.com/docs/security/features/verifiedboot/dm-verity

GKI(Generic Kernel Image)

The GKI kernel interacts with hardware-specific *vendor modules* containing system on a chip (SoC) and board-specific code. The interaction between the GKI kernel and vendor modules is enabled by the *Kernel Module Interface (KMI)* consisting of symbol lists identifying the functions and global data required by vendor modules. Figure 1 shows the GKI kernel and vendor module architecture:



ACK branch	Launch date	Support lifetime (years)	EOL
android11-5.4	2019-11-24	6	2026-01-01
android12-5.4	2019-11-24	6	2026-01-01
android12-5.10	2020-12-13	6	2027-07-01
android13-5.10	2020-12-13	6	2027-07-01
android13-5.15	2021-10-31	6	2028-07-01
android14-5.15	2021-10-31	6	2028-07-01
android14-6.1	2022-12-11	6	2029-07-01
android15-6.6	2023-10-29	4	2028-07-01
android16-6.12	2024-11-17	4	2029-07-01

https://source.android.com/docs/core/architecture/kernel

A/B Software update

Android OTA updates occasionally contain changed files that don't correspond to code changes. They're actually build system artifacts. This can occur when the same code, built at different times, from different directories, or on different machines produces a large number of changed files. Such excess files increase the size of an OTA patch, and make it difficult to determine which code changed.

To make the contents of an OTA more transparent, AOSP includes build system changes designed to reduce the size of OTA patches. Unnecessary file changes between builds have been eliminated, and only patch-related files are contained in OTA updates. AOSP also includes a build diff tool, which filters out common build-related file changes to provide a cleaner build file diff, and a block mapping tool, which helps you keep block allocation consistent.

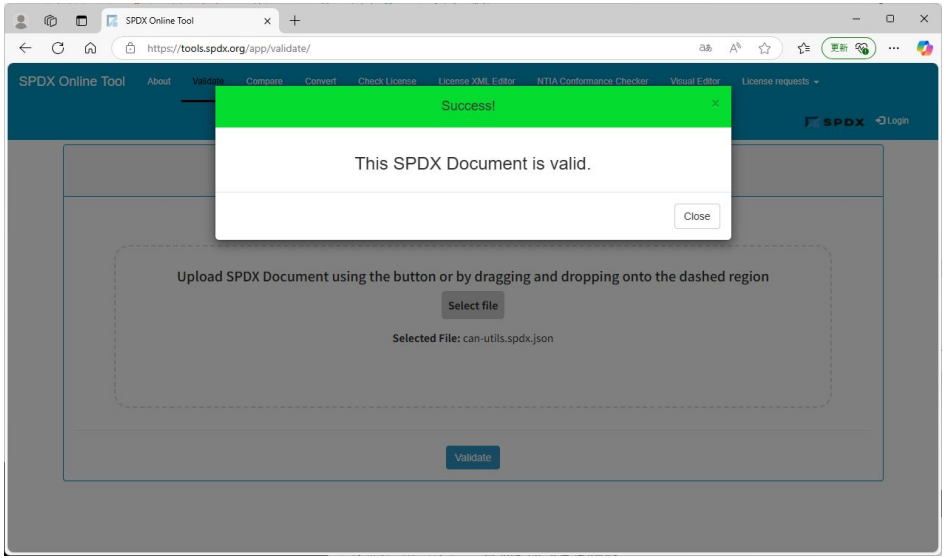
A build system can create unnecessarily large patches in several ways. To mitigate this, in Android 8.0 and higher, new features were implemented to reduce the patch size for each file diff. Improvements that reduced OTA-update package sizes include the following:

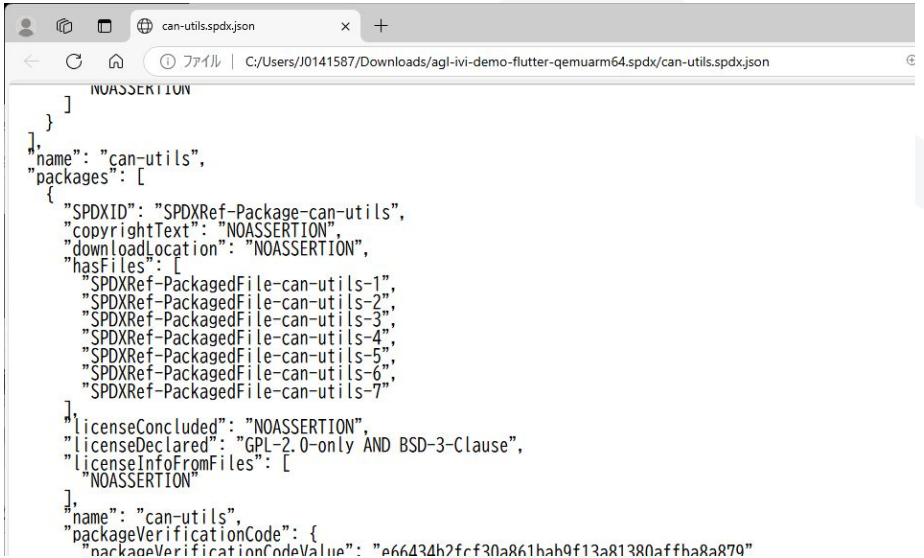
- Usage of **ZSTD**, a generic-purpose, lossless-compression algorithm for full images on non-A/B device updates. **ZSTD** can be customized for higher compression ratios by increasing compression level. Compression level is set during OTA generation time and can be set by passing the flag `--vabc_compression_param=zstd,$COMPRESSION_LEVEL`
- Increasing the compression window size used during OTA. The maximum compression window size can be set by customizing the build parameter in a device's `.mk` file. This variable is set as `PRODUCT_VIRTUAL_AB_COMPRESSION_FACTOR := 262144`

https://source.android.com/docs/core/ota/reduce_size

SBOM and ASIL topics

SBOM(SPDX and SPDX Tools)





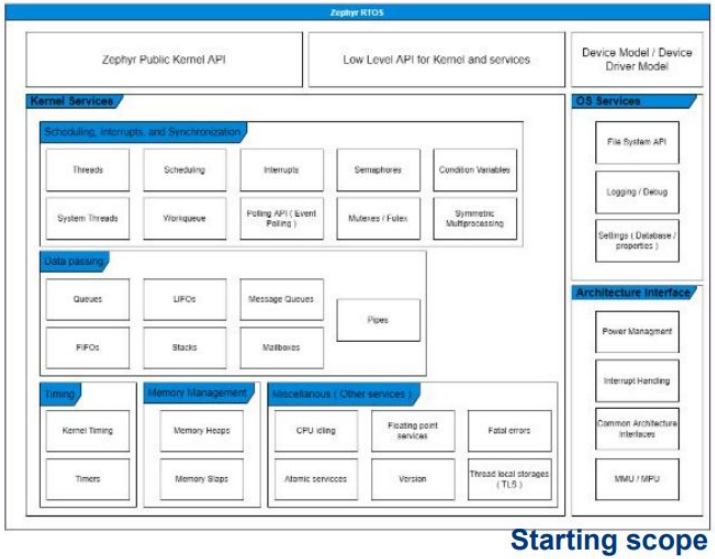
NOASSERTION: Package Download Location

Package Name	SPDX Identifier	Package Version	Package FileName	Package Supplier	Package Originator	Home Page	Package Download Location	Packag
can-utils	SPDXRef-Package-can-utils	2023.03		Organization: OpenEmbedded ()			NOASSERTION	

Zephyr ASIL(ISO26262)

Zephyr Initial Certification Focus

- Start with a limited scope of kernel and interfaces
- Initial target is IEC 61508 SIL 3 / SC 3 (IEC 61508-3, 7.4.2.12, Route 3s)
- Option for 26262 certification has been included in contract with certification authority should there be sufficient member interest



Starting scope

Scope can be **extended** to include **additional components** with associated **requirements** and **traceability** as determined by the safety committee

ELISA Enabling Linux in Safety Applications

WORKSHOP

https://sched.co/1urXZ

Xen ASIL(ISO26262)

Safety Certification Challenges

This page captures the efforts to certify Xen on ARM for ISO 26262 ASIL B. Discussions happen during the regular Xen on ARM Community Calls organized on xen-devel by Stefano Stabellini and Julien Grall. See [1] for Minutes.

We have identified the following requirements, all of them need an owner:

- **Code style requirements, a subset of MISRA**
  - Next step (Lars): find public documents that describe the code style requirements and publish them to xen-devel.
- **Create a subset of functions that need to go through certification**
  - Next step (Stefano): create a small Kconfig for Xen as a reference, using Renesas Rcar as starting point.
  - Start a discussion on which features we need to have. For instance real time schedulers might be required in some configurations but not all.
- **Understand how to address dom0:**
  - We need a plan for a non-Linux dom0.

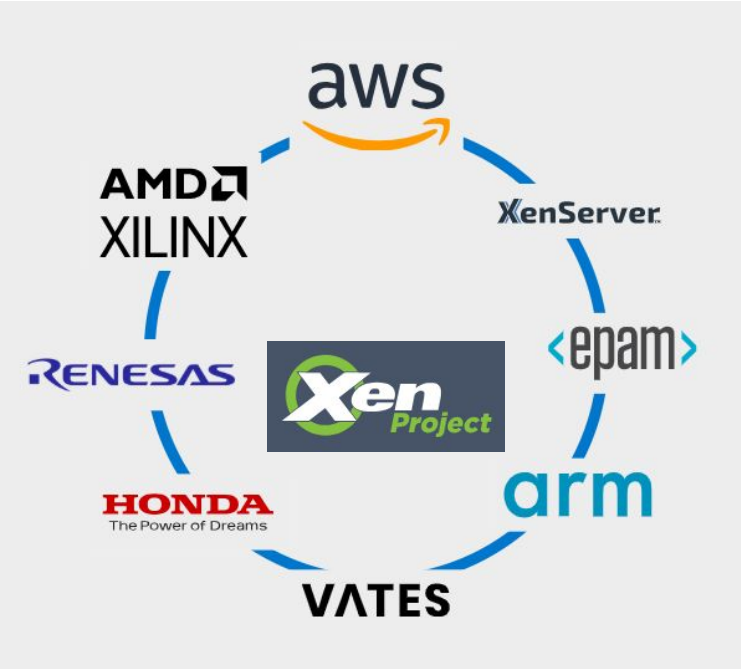
    It looks like **FreeRTOS** dom0 could be a good option.
    - Next step (Artem): Find out more information about FreeRTOS on Xen. Reach out to people that worked on it (Dornerworks? Galois?).

    An alternative may be a **dom0-less** option

    Until now, we discussed this topic under the name of "create multiple guests from device tree". There are no patches (as far as I know), but it was submitted as the Xen on ARM project for Outreachy this year.

    There are patches for a different project to setup shared memory regions from the xl config file (no need for grant table or xenbus support).
    - Next step (Praveen Kumar): volunteered to investigate
- **Create artifacts, such as docs, fault analysis, prove fault tolerance, safety management docs, document development processes.**
  - Next step: find a company or a certification body that would guide us through the process.

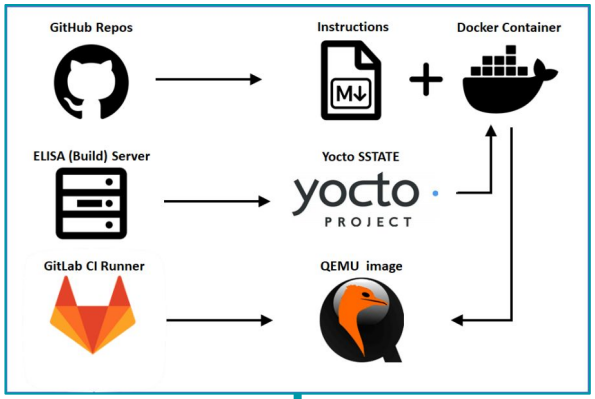
Renesas strives to realize OSS-based Safety solutions

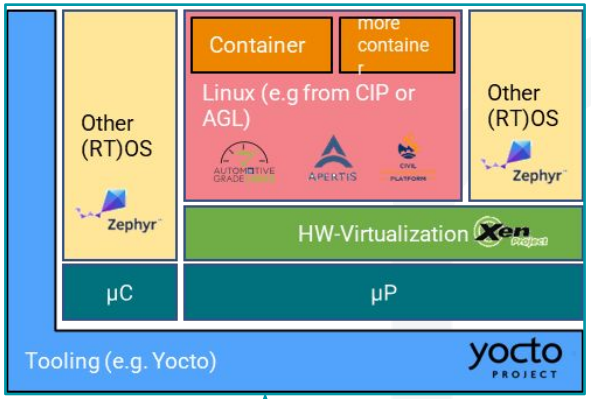


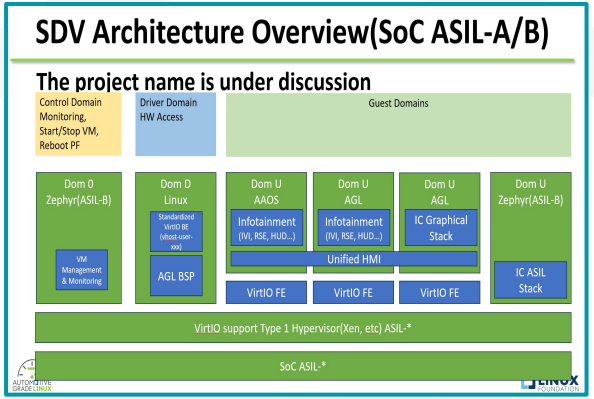
- Linux Foundation
  - Sliver
  - Serving the board director
- AGL
  - Gold, Advisory board and other roles
  - Reference platform provider
- Xen
  - Long-term contributor (via EPAM)
  - Newly joined as Premier to boost FUSA
- Zephyr
  - Upgraded to Platinum, to boost FUSA

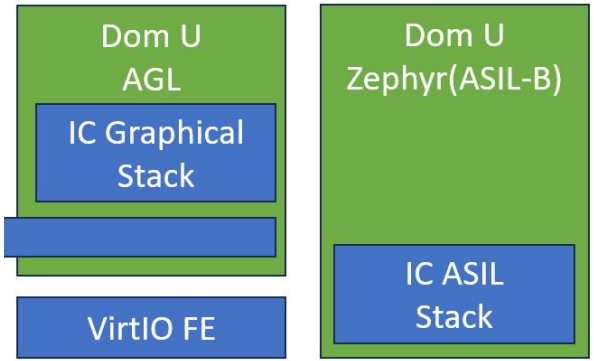
Let us start the discussion!

How to move forward? What - Who - Where - …?









ELISA Enabling Linux in Safety Applications

WORKSHOP

Additional projects

- S-Core → Put in DomU AGL
- KernelCI
  → Testing Linux Kernel and Link to Requirements
  → Continuous Compliance in Systems WG
- SPDX → Complete System BOM in AGL?
- More?