Eclipse Software Defined Vehicle

Leda Incubator

22nd September 2022
Mike Haller
ETAS GmbH
Agenda

- Eclipse Leda
- Incubator Goals
- Cloud Connector
- Vehicle Update Manager
- Self Update Agent
- OpenTelemetry Collector
- Roadmap
SDV?

- New **challenges** to be solved
- Global, collaborative ecosystem is needed
- There is **no "One size fits all"**
- Separating hardware and software lifecycle
- Automotive SW is a highly specialized domain
- A lot of new code will be written
- It will have to be maintained for a long time
- A lot of computers-on-wheels need to be administrated
Looking into the crystal ball

- Automotive SW development must become more open, more attractive, it needs to spark enthusiasm
- Methodology and technology from IoT and Cloud will find its way into the vehicles
- DevOps will become normal for millions of vehicles
- The build–deploy–monitor DevOps cycle will continuously become faster and more effective thanks to containerized applications
- Vehicle Abstraction Layers are laying the ground for innovation
- Technology needs evaluation, ideas need to be tested
- A lot of variations and alternatives will be possible
Leda?

It takes time to ...
... find the right components
... design a working stack
... build the full stack
... deploy on a device
... configure the base services
... and NOW you can start developing your app.

How about...
... download Leda
... run on qemu/docker or some affordable HW device
... and directly start deploying your Vehicle Applications?

* Shamelessly stealing the Kuksa ideas >:-D
## The diff to commercial

<table>
<thead>
<tr>
<th></th>
<th>Eclipse Leda &amp; Incubator</th>
<th>Commercial projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sources</td>
<td>Open Source only</td>
<td>OSS + proprietary</td>
</tr>
<tr>
<td>Releases</td>
<td>Multiple variations of the SDV stacks (full vs minimal vs rescue-mode)</td>
<td>Single target device only (as optimized as possible)</td>
</tr>
<tr>
<td>Hardware</td>
<td>Common, affordable, available</td>
<td>Specialized, expensive</td>
</tr>
<tr>
<td>Customization</td>
<td>Sane defaults for quickstart image</td>
<td>Full customization</td>
</tr>
<tr>
<td></td>
<td>High flexibility when building from scratch</td>
<td>No invest into &quot;quickstart&quot;</td>
</tr>
<tr>
<td>Documentation</td>
<td>Community</td>
<td>Commercial, Trainings</td>
</tr>
<tr>
<td>Validation</td>
<td>Briefly - low hanging fruits</td>
<td>Strict, Lots of regulations, certification requirements, &quot;100% Coverage&quot;</td>
</tr>
<tr>
<td></td>
<td>Focus: demo the process</td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>Community, best-effort</td>
<td>Commercial, long-term contracts</td>
</tr>
<tr>
<td>Entry barrier</td>
<td>Low, Accept contributions</td>
<td>High</td>
</tr>
</tbody>
</table>
Deliverables

**Leda**

**Build recipes** for SDV-related packages
Meta-Layer (Yocto, OpenEmbedded)

**Quickstart Images (Distro)**
Sane default configurations
Pre-Integrated Documentation, How-Tos

Runs on Raspi, QEMU, ... Docker

**Incubator**

**Source Code**
Build recipes
Documentation

No releases!
Leda Incubator Goals

A place for new SDV components for integration into the Leda quickstart images.

Includes new components, experimental, pre-mature, temporary implementations etc. to fill the current gaps.

Leda Incubator can be a home until the "right" upstream projects are identified, contributions are worthy or the component may even become a standalone project.

Low entry barrier regarding overhead (don't need project proposals, project websites, build environment etc. › Leda can be reused)
Use Cases

OTA Updates
- OS Update
- App Containers

App Monitoring
- Metrics
- Logs
- Traces

Digital Twin
- CPU, Memory, Version
- CPU, Logs
Incubator Components

- Cloud Connector
- Self Update Agent
- RAUC Update Service
- VApp
- Vehicle Update Manager
- OpenTelemetry Connector
Cloud Connector

Connect the vehicle to a cloud backend and route messages

- Connectivity (Azure IoT Hub)
- Authentication & Security
- Device Identity
- Connectivity Recovery
- Device Telemetry (D2C)
- Command & Control (C2D)
- Extensible Message Routing (Apps)

Potential future upstream project:

Status: Implemented > CQ
Vehicle Update Manager

Dispatch remote requests for different types of updates

- Routing Decision: Self Update vs Container vs other domains
- Delegate to corresponding target service
- Define message protocol and behavior (UpdateAgent API)
- Update requests follow "desired state" paradigm

Potential future upstream project:

Status: Implemented > CQ
Self Update Agent

Enable Over-the-Air (OTA) updates of the operating system and base packages

- Triggered by remote API via PubSub from the cloud backend
- Determine need for update (current vs desired)
- Dispatch to local update mechanism, such as RAUC
- Send progress information to cloud

Potential future upstream project:

Status: Implemented > CQ
OpenTelemetry Collector

Collect and publish general device telemetry and applications logs, metrics and traces

- Use OTel / OTLP implementations
- Example configurations
- Adapter for mqtt integration

Potential future upstream project:

Status: Implemented > CQ
OpenTelemetry Collector

- Minimized, custom container image
- Open OTEL/OTLP protocols
- Logs from native services (eg journald)
- Logs from containers
- Velocitas V-Apps: Logs + DAPR traces
- Host Metrics (eg CPU load, memory)
- Metrics per container
  - cpu & memory per node/pod/container

Example monitoring backends:

- Zipkin
- jaeger
- elasticsearch
- Prometheus
- Grafana
- loki
- ...
Current Work

- Support for Yocto **Kirkstone** LTS until Apr’24
- Migrate to better build and layer tool: siemens/\texttt{kas}
- Recipes for ESRLabs **Northstar**: embedded container runtime
- Build recipes for first **incubators**
- ETAS sponsoring an Eclipse Leda introduction video
- Setup build environment on Eclipse infrastructure

Roadmap

- Support for Yocto **Dunfell**
- Support for Yocto **Langdale**
- **Prepare binary OSS build** ("nightly")
- Setup Eclipse ORT license scanning
- Setup of dependency track + CVE checks
- Dockerized qemu images
Join us!

You can help us by using Leda!
We're grateful for feedback

Mailing List:
leda-dev@eclipse.org

We're looking for ...
• additional hardware to support
• ideas for use cases
• contributions (ideas, code, docs)
THANK YOU