



Vehicle Abstraction in Automotive Grade Linux with Eclipse KUKSA

FOSDEM

February 3, 2024

Sven Erik Jeroschewski (SvenErik.Jeroschewski@bosch.com)

Scott Murray (scott.murray@konsulko.com)

whoami



@

ETAS



2018

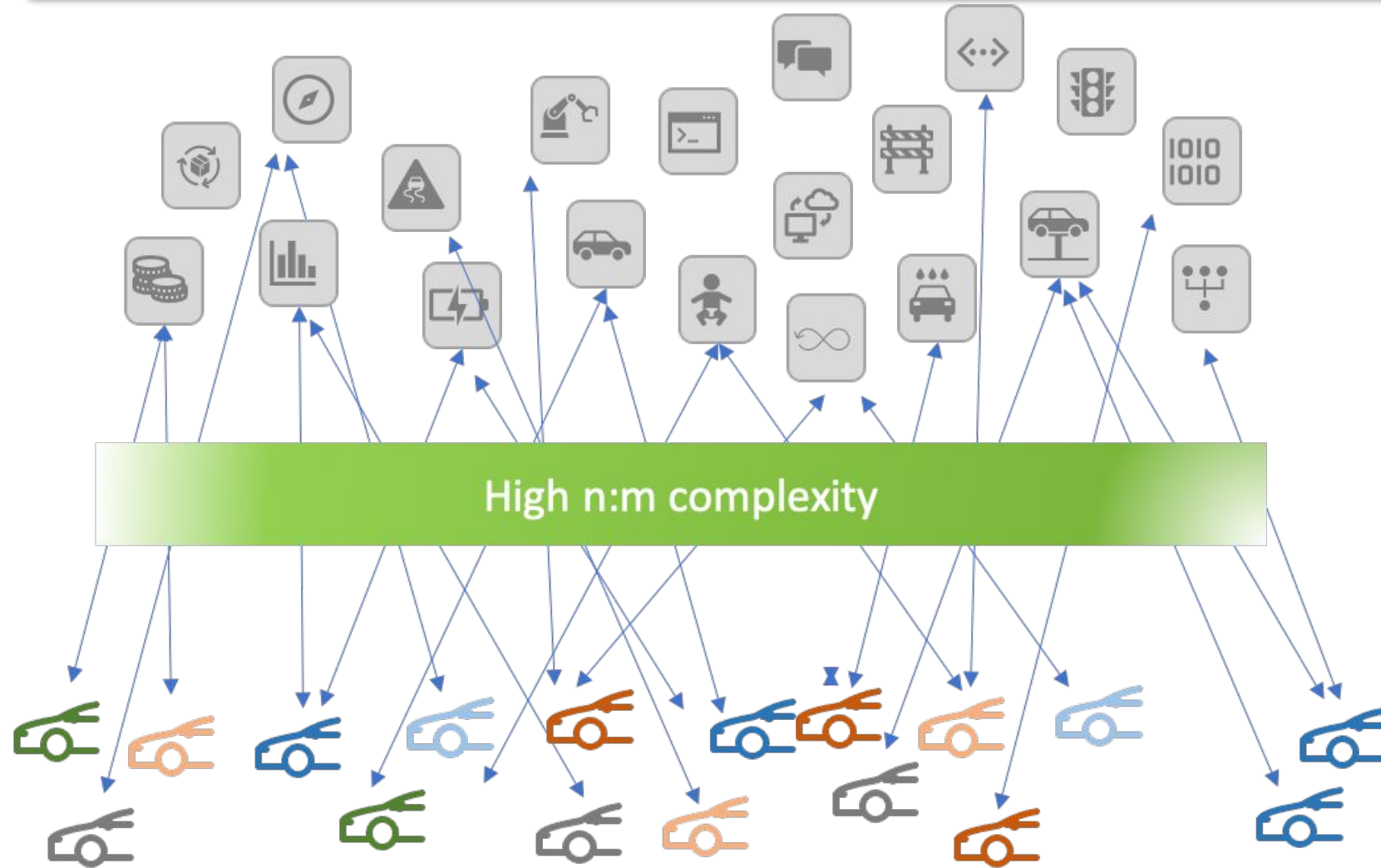


today



How to make application development for vehicles more fun and efficient?

Challenge: No standardized signals



Pain Points:

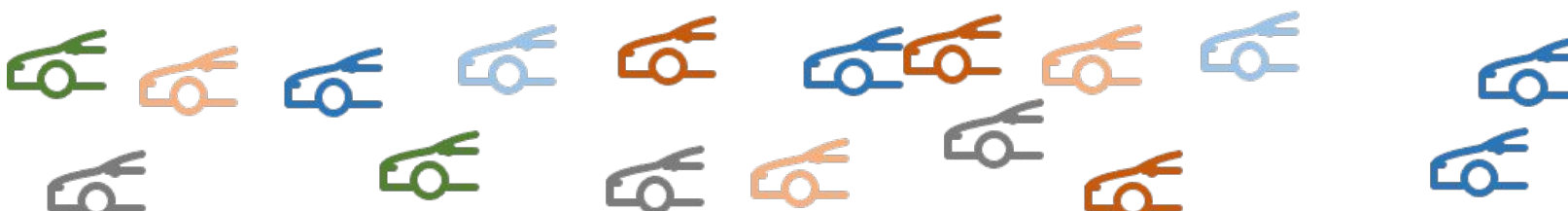
- Portability
- Scalability
- Maintenance

Leading to high complexity and data silos

Solution: Vehicle Abstraction

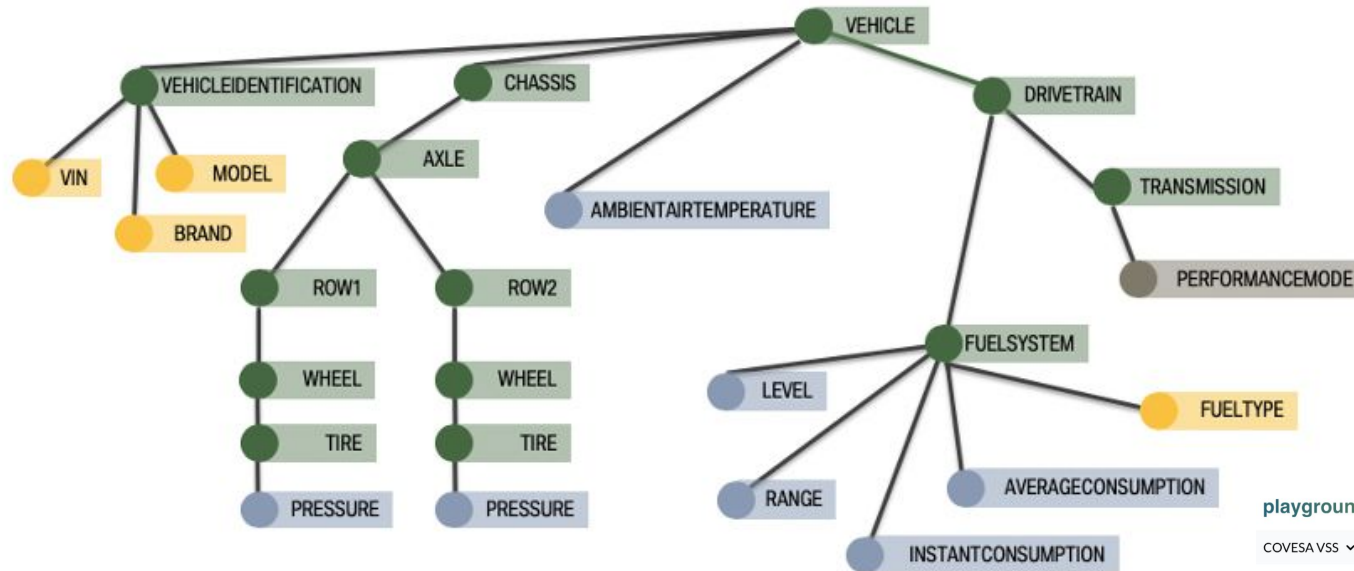


Vehicle Abstraction



- Portability:
"Write once, run everywhere"
- Scalability:
"Attract 3rd party developers"
- Maintenance:
"Realize Synergies"

COVESA Vehicle Signal Specification



<https://digitalauto.netlify.app/>



ATTRIBUTE
BRANCH
SENSOR
ACTUATOR

```

- Drivetrain.Transmission.Speed:
  type: sensor
  datatype: uint16
  unit: km/h
  min: 0
  max: 300
  description: The vehicle speed, as measured by the drivetrain.
    
```

playground.digital.auto

Search ACME Car (EV) v0.1 Vehicle APIs Prototypes

COVESA VSS

Vehicle.Cabin.Seat

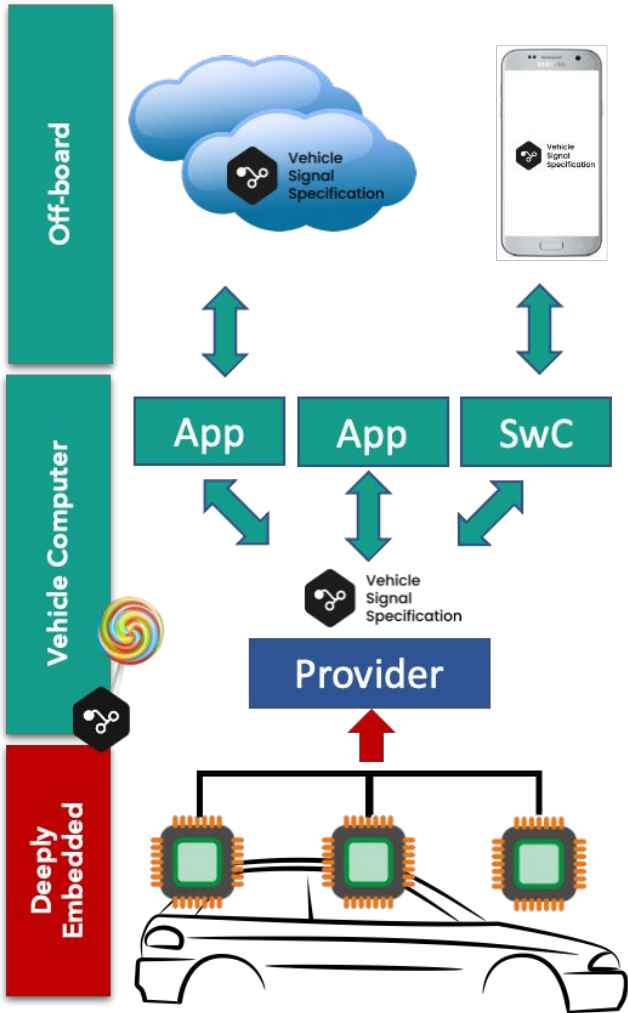
Recently viewed APIs

- Vehicle.Cabin.Seat BRANCH
- Vehicle.Cabin.Seat.Row1 BRANCH
- Vehicle.Cabin.Seat.Row1.Pos1 BRANCH
- Vehicle.Cabin.Seat.Row1.Pos1.Airbag BRANCH
- Vehicle.Cabin.Seat.Row1.Pos1.Airbag.IsDeployed SENSOR
- Vehicle.Cabin.Seat.Row1.Pos1.Backrest BRANCH
- Vehicle.Cabin.Seat.Row1.Pos1.Backrest.Lumbar BRANCH
- Vehicle.Cabin.Seat.Row1.Pos1.Backrest.Lumbar.Height ACTUATOR
- Vehicle.Cabin.Seat.Row1.Pos1.Backrest.Lumbar.Support >
- Vehicle.Cabin.Seat.Row1.Pos1.Backrest.Recline ACTUATOR

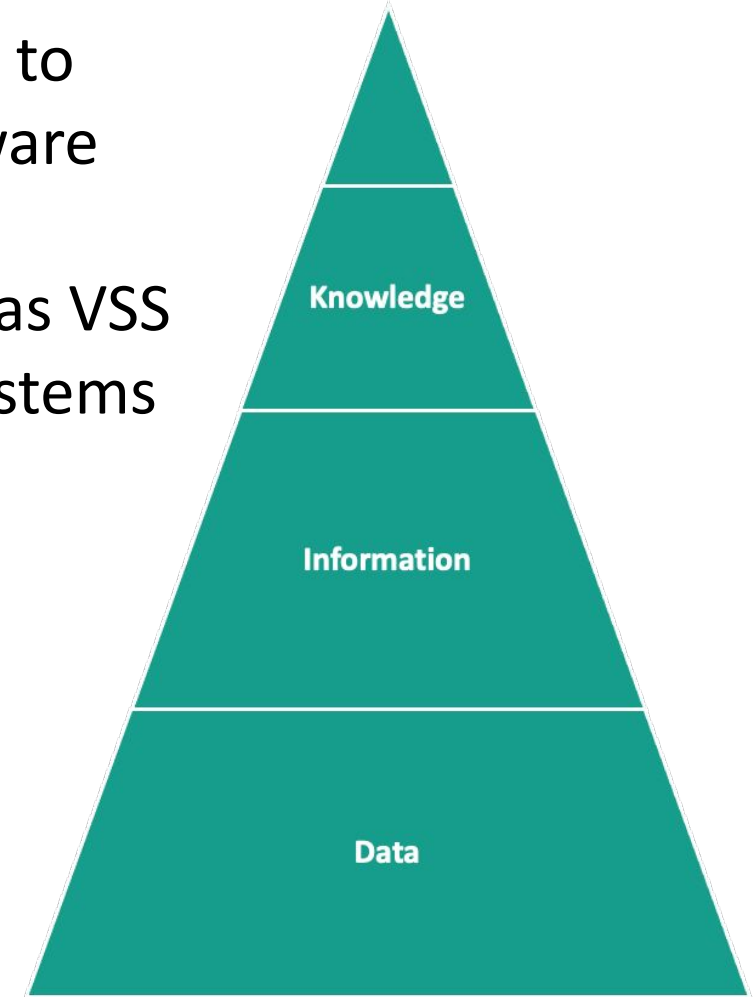
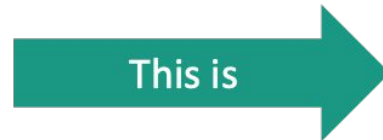


Source: https://covesa.github.io/vehicle_signal_specification/introduction/overview/

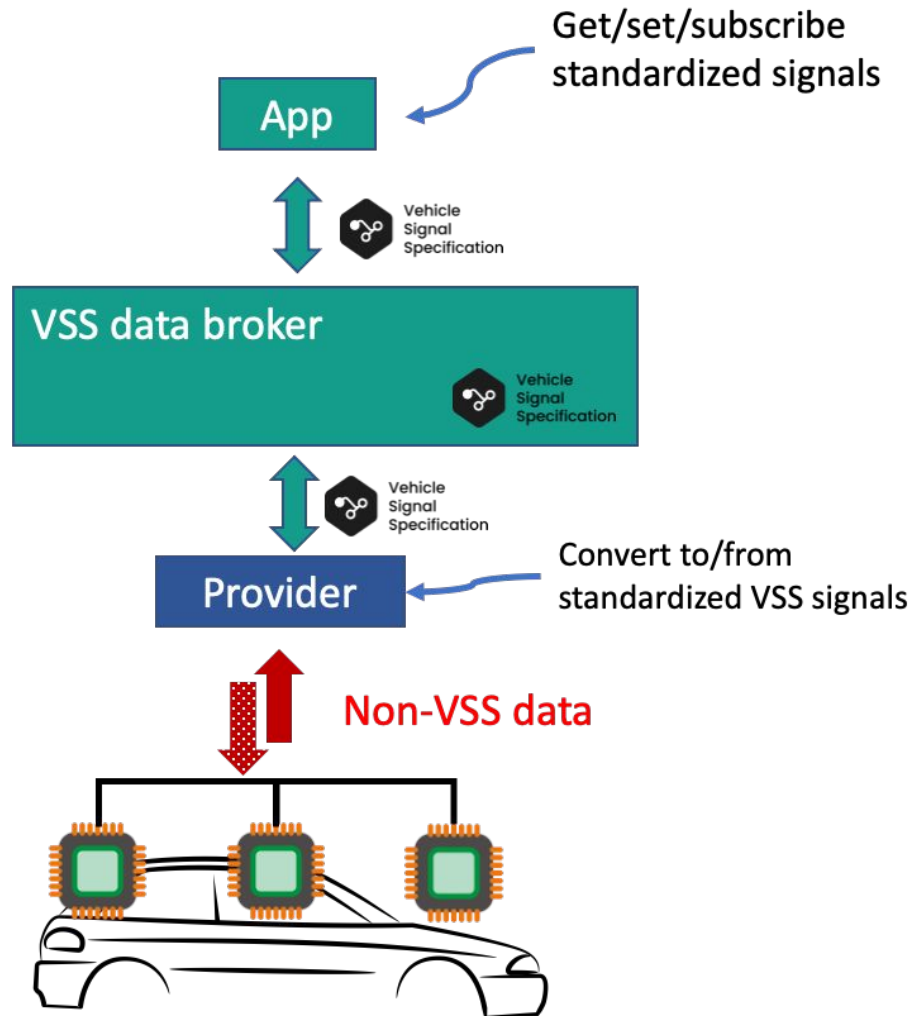
KUKSA.val



- Vehicle Computer as place to decouple hard- from software (SDV)
- Add an API to access data as VSS from deeply embedded systems



KUKSA.val Scope



- Open Source Project at Eclipse Foundation under Apache 2.0
- “In-Vehicle digital twin” based on VSS
- Only provide current and target view (no history data)
- VSS providers to transform data to VSS

Data Broker:

- Written in Rust
- < 4MB statically compiled
- language agnostic interface (gRPC):
Get, Set, Subscribe

Let's get to the news

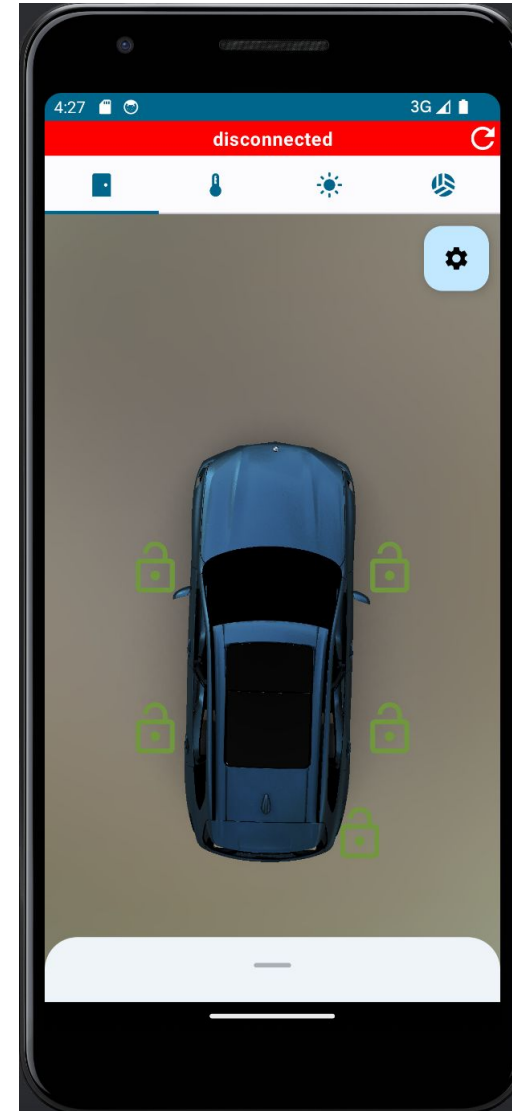
- KUKSA Android SDK released
- Vehicle Mock Service Available
- Leda on AGL

KUKSA Android SDK

- SDK for Android to communicate with KUKSA.val Data Broker
 - Available in Maven Central as `org.eclipse.kuksa.kuksa-sdk`
- Released Example Companion Application based on SDK to F-Droid Store



Source: <https://f-droid.org/>



Vehicle Mock Service

- Mock vehicle behavior:
 - Example: move current value for 10 seconds to target after it has been set
- Python script accepting behavior definition in the form of specific `mock.py`
- https://github.com/eclipse/kuksa.val.services/tree/main/mock_service

mock.py - Example

```
mock_datapoint(  
    path="Vehicle.Cabin.Seat.Row1.DriverSide.Position",  
    initial_value=0,  
    behaviors=[  
        create_behavior(  
            trigger=create_event_trigger(EventType.ACTUATOR_TARGET),  
            action=create_animation_action(  
                duration=10.0,  
                values=["$self", "$event.value"],  
            ),  
        )  
    ],  
)
```

Sneak Preview: Eclipse Leda on AGL

```
Automotive Grade Linux 16.0.2 \n \l (GNU/Linux 5.15.124-yocto-standard x86_64)

Hostname.....: qemux86-64
Uptime.....: 0 days, 0:6 hours
System load...: 0.12 (1min) | 0.10 (5min) | 0.05 (15min)
Disk Space....: Root: 9.1G | Used: 2.9G | Free: 5.8G
                Data: | Used: | Free:
RAM .....: Total: 1.9G | Used: 662.5M | Free: 98.4M
Network.....: Interface: | IPv4 Address:
                eth0 | 10.0.2.15
Containers....: 6 running | 1 stopped | 0 exited
To check SDV services health, run $ sdv-health
Use $ sdv-device-info to display device configuration
Use $ sdv-provision to generate self-signed device certificates

root@qemux86-64:~# sdv-health
[SDV Info]
* OS Release:      : Automotive Grade Linux 16.0.2 (pike)
* Image Version:   : Automotive Grade Linux 16.0.2 \n \l
* Build Time:      : 20231103082640

-----
[CAN Status]
* can0             : OK

-----
[SDV Ports]
* OpenSSH          : OK      { :::22 }
* Kanto CM         : OK      { /run/container-management/container-mana
* Mosquitto Server : OK      { 127.0.0.1:1883 :::1883 }

-----
[SDV Services]
* containerd       : OK
* rauc              : FAILED! (Unit rauc.service could not be found.)
* container-management : OK
* kanto-update-manager : OK

-----
[SDV Optional Services]
* sshd.socket      : OK
* systemd-networkd : OK
* systemd-timesyncd : N/A (Unit systemd-timesyncd.service could not be found.)

-----
[Kanto CM Containers]
* cloudconnector   : WARNING (NOT FOUND)
* databroker       : OK
* sua              : WARNING (Stopped)

-----
[Kanto CM Containers (OPTIONAL)]
* seat-service-example : OK
* hvac-service-example : OK
* node-red-example     : OK
* feeder-can         : OK
* feeder-gps          : OK
* otelcol-sdv-exporter : WARNING (NOT FOUND)
* otelcol-sdv-agent    : WARNING (NOT FOUND)

-----
[SDV Connectivity]
* Ping [Internet]   : OK (ping 1.1.1.1)
* DNS Lookup [Internet] : OK (142.250.185.238)
* Cloud Connector   : FAILED! ({"connected":false,"timestamp":1706635304,"cause":"PROVISIONING_MISSING"})
* Device ID         : FAILED! (Connector did not respond)
```

```
Automotive Grade Linux 16.0.2 \n \l (GNU/Linux 5.15.124-yocto-standard x86_64)

Hostname.....: qemux86-64
Uptime.....: 0 days, 0:12 hours
System load...: 0.00 (1min) | 0.00 (5min) | 0.00 (15min)
Disk Space....: Root: 2.0G | Used: 1.2G | Free: 659M
                Data: | Used: | Free:
RAM .....: Total: 1.9G | Used: 398.2M | Free: 1.1G
Network.....: Interface: | IPv4 Address:
                eth0 | 10.0.2.15
Containers....: 0 running | 0 stopped | 0 exited
```

whoami #2

- Linux user/developer since 1994
- Embedded Linux developer since 2000
- Principal Software Engineer at Konsulko Group since 2014
- Working on AGL on contract since 2016
 - Yocto Project maintenance
 - Demo development, integration, and maintenance



Automotive Grade Linux

- A collaborative open source project that is bringing together automakers, suppliers, and technology companies to build a Linux-based, open software platform for automotive applications
- Founded in 2014
- Currently over 150 members
 - 10 major OEMs and many Tier 1 and Tier 2 suppliers
- Code first model (as opposed to specification driven)
- Used in production vehicles from Toyota and Subaru
- <https://www.automotivelinux.org/>

AGL Provides...

- A base automotive oriented Linux distribution built with Yocto Project (<https://www.yoctoproject.org/>)
- Goal of providing 70-80% of the base platform for production
- Focus was initially on in vehicle infotainment (IVI) targets
- Expansion into instrument cluster (IC) and telematics based on member interest
- Expert groups for various areas of interest, with open biweekly meetings
- Biannual releases (nominally February & August)

KUKSA.val in AGL?

- Up until 2020, a lot of AGL development went into a demonstration application framework
 - Included CAN and higher level signal abstraction APIs
- Members indicated they were not interested in further effort going into the application framework
 - OEMs already have frameworks in hand
- AGL shifted towards a bit more of a FOSS technology demonstrator model for its integration demos
- The timing aligned well with respect to the releases of VSS and KUKSA.val

KUKSA.val in AGL? (cont)

- KUKSA.val server was initially added in the Marlin (13.0) release in March 2022
 - Replacement for previous agl-service-can-low-level and agl-service-signal-composer
- A BitBake recipe to build the server (and now databroker) is carried in the meta-agl-demo layer
 - Custom AGL VSS generated by applying overlay vspec file on top of base VSS
- The CAN feeder is also built and packaged with a BitBake recipe in meta-agl-demo
 - Uses CAN database (DBC) file with minimal "agl-vcar" CAN signal definitions

KUKSA.val Integration in AGL

- Magic Marlin (13.0) - Spring 2022
 - KUKSA.val 0.2.1 integrated
 - VSS 2.2
 - Using C++ server with VIS WebSocket API
 - kuksa-dbc-feeder CAN feeder for demos
- Nifty Needlefish (14.0) - Summer 2022
 - Upgraded to KUKSA.val 0.2.5 and VSS 3.0
- Optimistic Octopus (15.0) - Spring 2023
 - Upgraded to KUKSA.val 0.3.1 and VSS 3.1.1
 - Switch to using vspec overlay with vss-tools

KUKSA.val Integration in AGL (cont)

- Prickly Pike (16.0) - Summer 2023
 - Still using KUKSA.val 0.3.1
 - Databroker included in images for evaluation and testing
 - Using Rust 1.68 mixin layer for Yocto kirkstone
- Quirky Quillback (17.0) - Spring 2024
 - KUKSA.val 0.4.2 and VSS 4.0
 - 0.4.2 released by community to get a working RISC-V build of the databroker
 - Fully switched over to the databroker
 - Rust 1.70 mixin layer for Yocto kirkstone recently published

VSS Applications in AGL

- Pure VSS signal observers
 - e.g. IC dashboard applications
 - Read "sensors" in VSS terminology
- VSS signal actors
 - e.g. services like agl-service-hvac
 - Implement "actuators" in VSS terminology
- Some applications also set actuators
 - HVAC, audio controls, navigation, etc.

Demo services

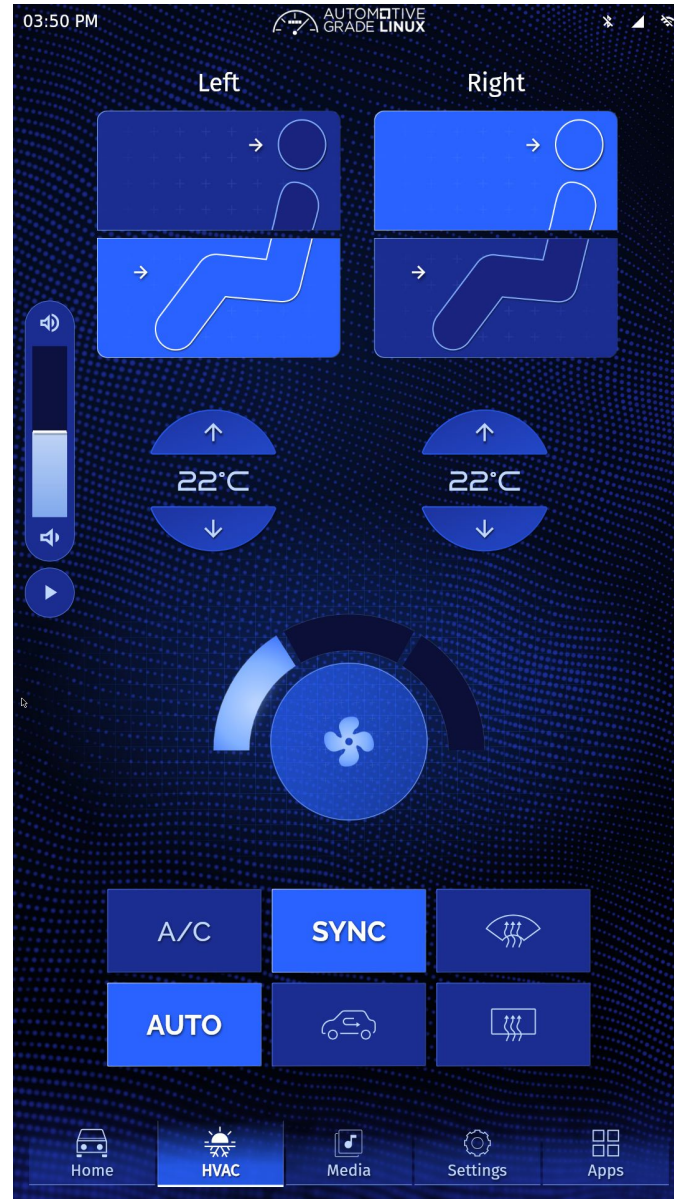
- agl-service-hvac
 - Listens to fan speed and temperature actuator signals
 - Pushes fan speed updates to HVAC controller via CAN
 - Pushes temperature updates to LEDs in demo unit via GPIO
- agl-service-audiomixer
 - Listens to VSS volume and some AGL custom audio control actuator signals
 - Pushes changes to WirePlumber

Qt Demo Applications

- VSS signal using applications:
 - Homescreen
 - Dashboard
 - IC dashboard
 - HVAC
 - Navigation
- Client code is abstracted in Qt library (libqtappfw-vehicle-signals) to reduce code duplication
 - Originally VIS WebSocket based, now gRPC with databroker

Flutter Demo Applications

- VSS signal using applications:
 - Homescreen
 - Combines dashboard, HVAC, media, etc. into a unified application a bit more realistic with respect to current OEM designs
 - UI designed for AGL by ICS for CES 2024
 - IC dashboard
- Client code is currently duplicated in each application, but is not large
- Some potential for switching to using a wrapped non-Dart gRPC implementation, e.g. Rust's tonic
 - Toyota has indicated they do something along these lines



More information

- Vehicle Abstraction with Eclipse Kuksa and Eclipse Velocitas - Sven Erik Jeroschewski, Bosch Digital

https://static.sched.com/hosted_files/aglammspring2023/5c/VehicleAbstractionwithEclipseKuksaandEclipseVelocitas.pdf

<https://www.youtube.com/watch?v=LHJnBKb1Ta8>

- Vehicle Signaling Specification and KUKSA.val in AGL

https://static.sched.com/hosted_files/aglammspring2023/8f/VSS%20and%20KUKSA.val%20in%20AGL.pdf

https://www.youtube.com/watch?v=RhSocQDu_DY

- AGL table in AW building on Sunday!

Want to start hacking? Join us @



BCM24

**Bosch
Connected
Experience**

February 26–28
in Berlin

Questions?
