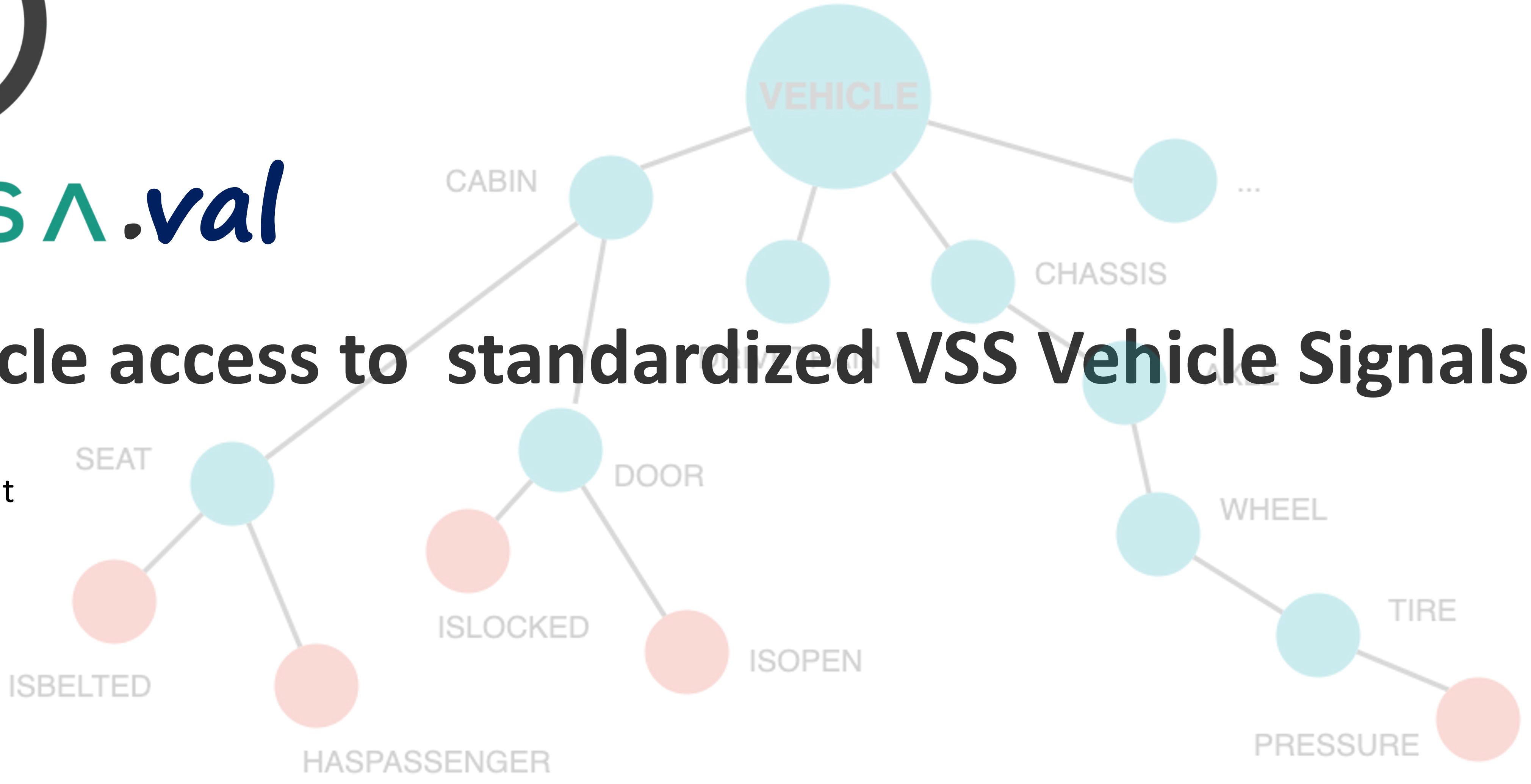
KUKSA.val

In-vehicle access to standardized VSS Vehicle Signals

FOSDEM 2023
Sebastian Schildt



KUKSA

Is an OSS software project

is part of

SDV

Eclipse Software Defined Vehicle

Is an Eclipse Working group comprised of several automotive centered OSS software projects

shares

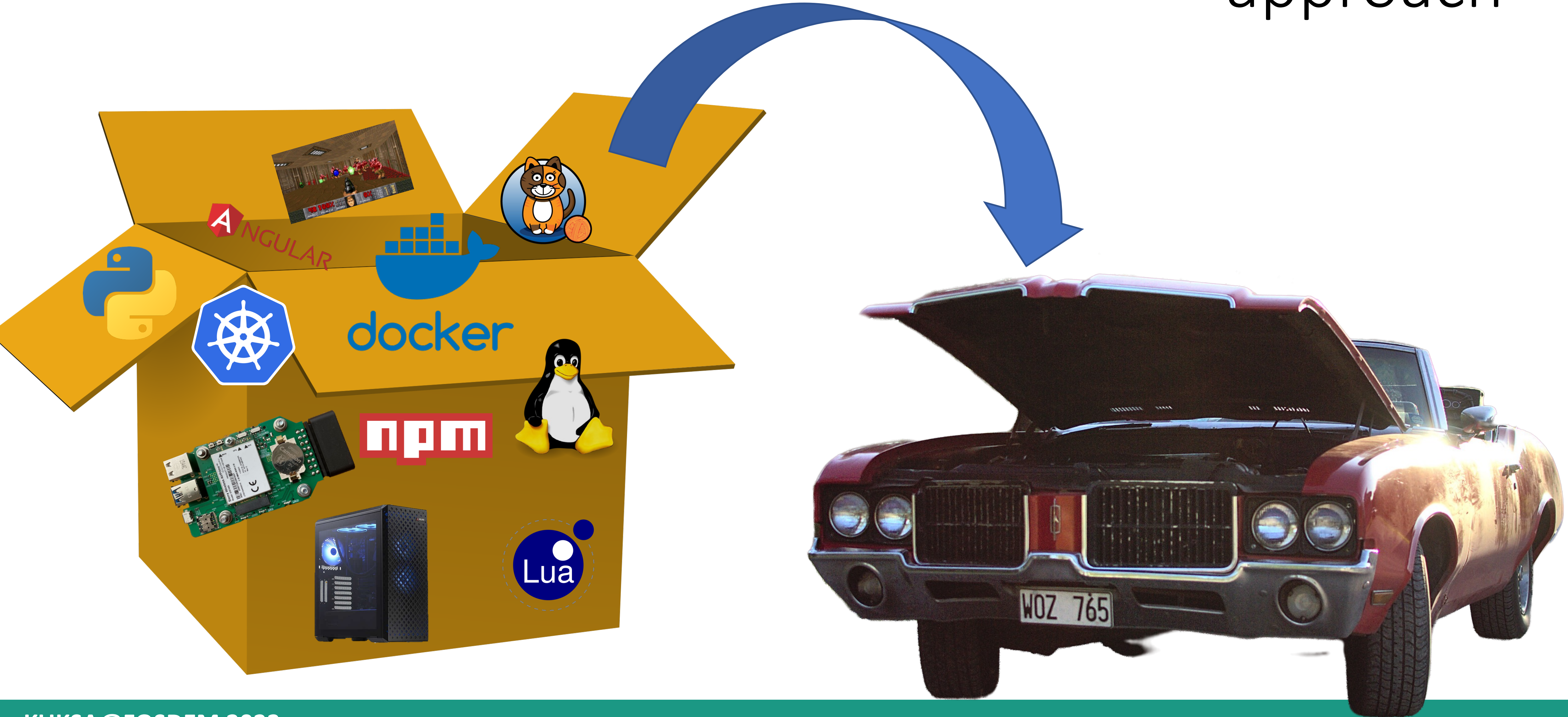
Software Defined Vehicle Mindset

Latest and greatest hype in the automotive industry

SDV promises

- Faster updates
- More “app” less “firmware”
- 10x faster development
- Reusable software
- “Cloud native” in vehicles
- Happy Users, happy devs, happy corporations, blissful happiness for all

The SDV approach



Hanna Sörensson "Open Hood" https://www.flickr.com/photos/hasor/48372469486

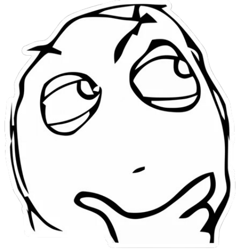
What did we achieve?

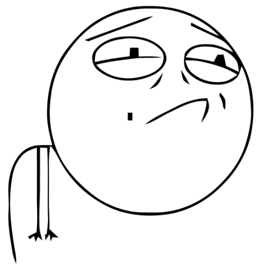
<u>The good</u>

- I can easily deploy Wordpress in my vehicle
- Probably runs Doom

<u>The bad</u>

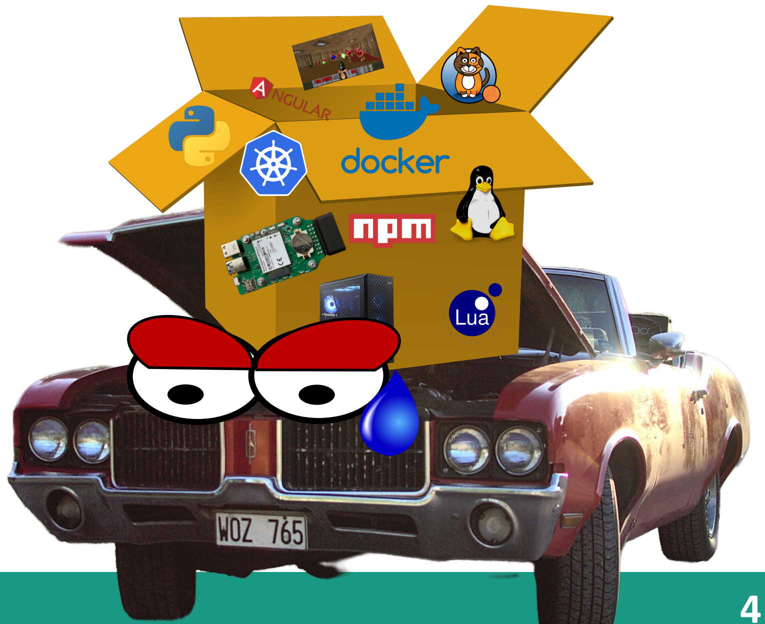
- Probably should get security and safety right





<u>The ugly</u>

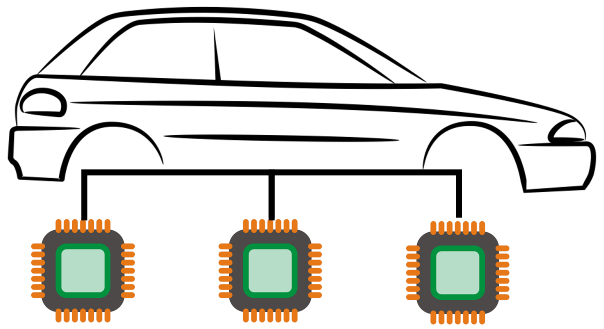
- Without any access to a vehicle‘s hardware, deploying Wordpress and running Doom is likely *all* I can do



Access to Vehicle Hardware

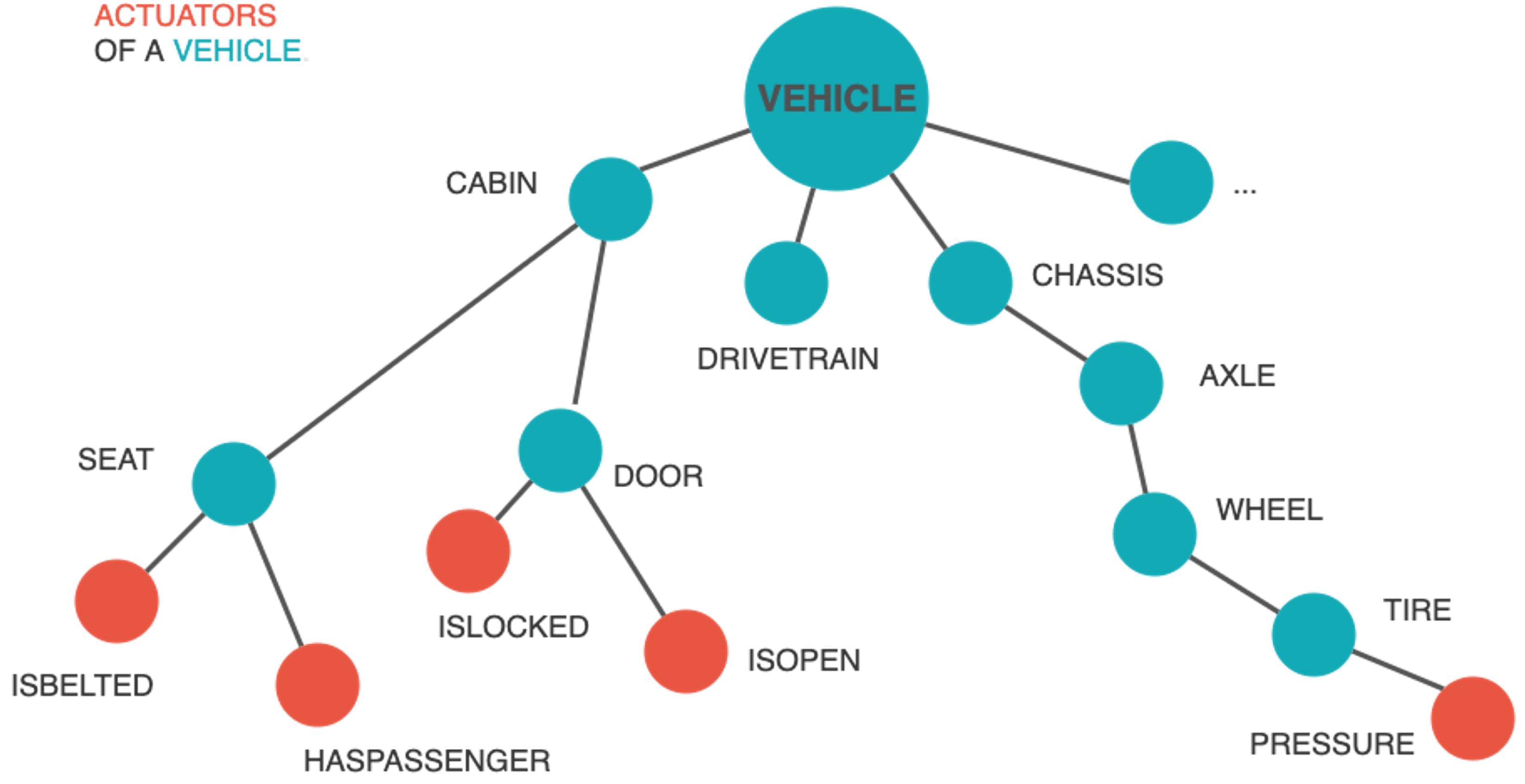
We have **sensors** (what is our current speed?) and **actuators** (e.g open the trunk!)

- Accesible over Vehicle busses (e.g. CAN, Ethernet), originating in some embedded, often safety critical ECUs (µCs, AUTOSAR, …)
  - Challenge: Accessing those systems directly from our fancy IT stacks would be insane for safety reasons alone
- How to represent a Vehicle Speed (serialisation, identifiers, units) is not standardised. Varies from OEM to OEM, from model to model, model year, variant
  - Challenge: Semantics of Signals very much not standardised. Similar things are not represented in the same way



Challenge: No standardized signals
Solution: COVESA Vehicle Signal Specification (VSS)

TAXONOMY FOR ATTRIBUTES, SENSORS AND ACTUATORS OF A VEHICLE



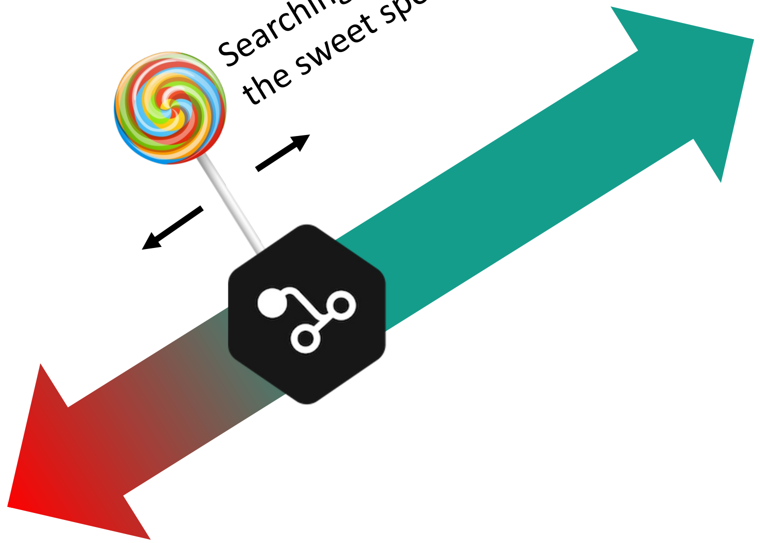
```
Vehicle.Drivetrain.Transmission.Speed
  type: sensor
  datatype: float
  unit: km/h
  description: The vehicle speed as measured by the drivetrain
```

YAML SPECIFICATION

- A simple, flexible and protocol agnostic common approach for describing vehicle data
- Extensible data model & catalog with industry supported tooling.
- Enables improved interoperability and integration, saving time and cost.

COVESA
Accelerating the future of connected vehicles

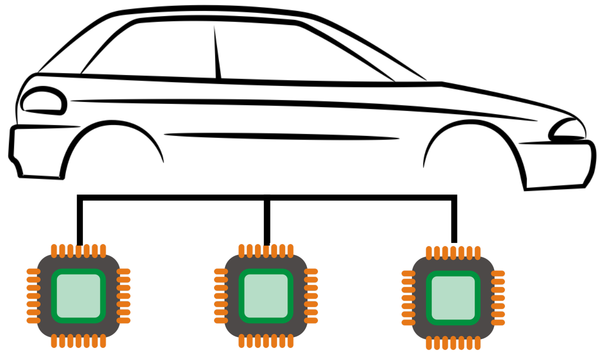
Question: Where to best leverage VSS?



<u>Backend</u>

- Aggregating data of many vehicles
- Link data to other domains

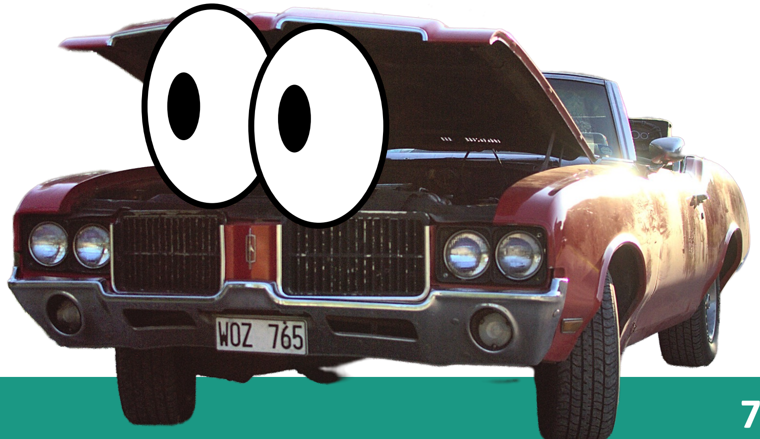
Good for VSS. Systems already in production



<u>Deeply-Embedded Layer</u>

- Small µCs
- CAN/LIN
- Very proprietary
- Very limited compute resources

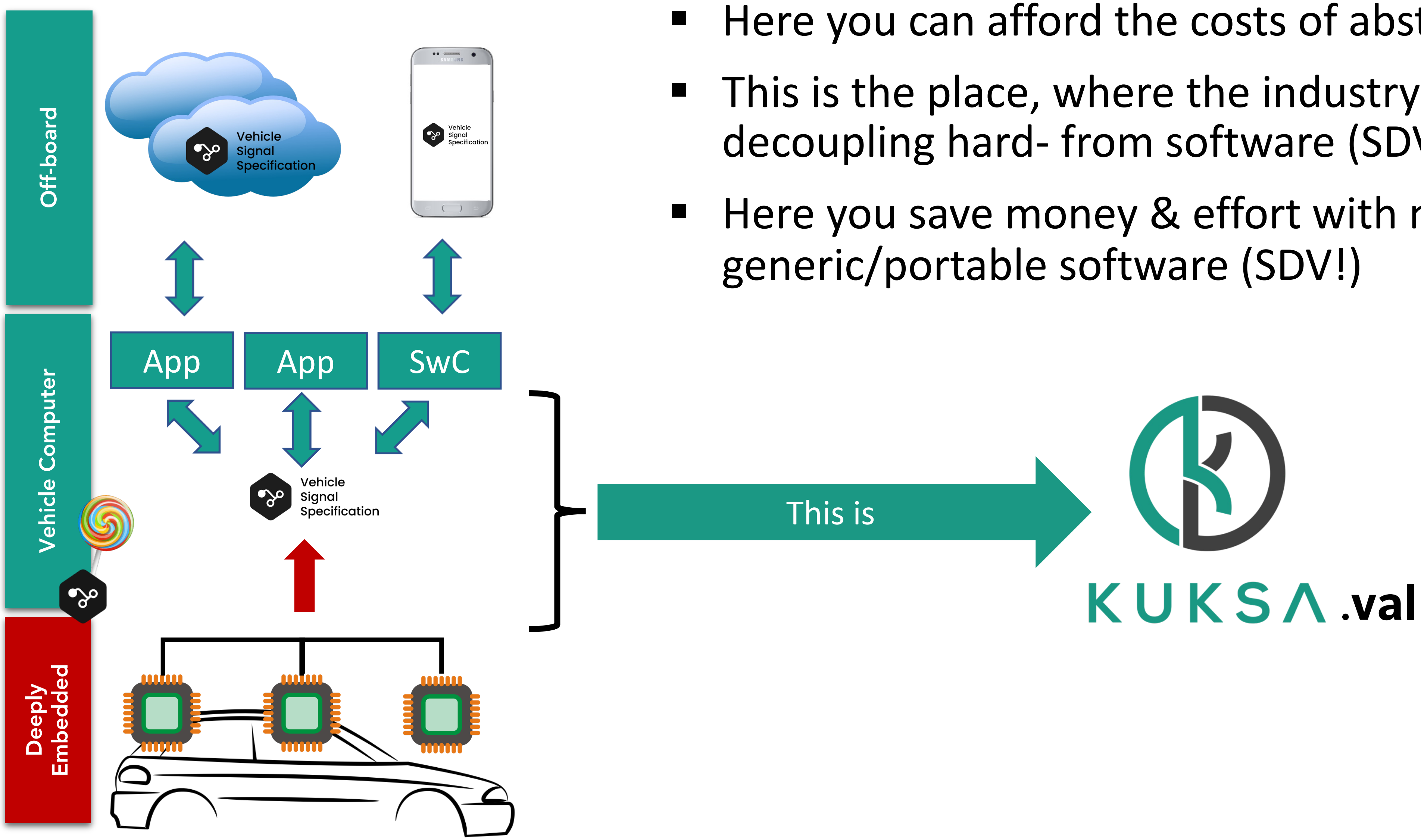
Not a happy place for VSS



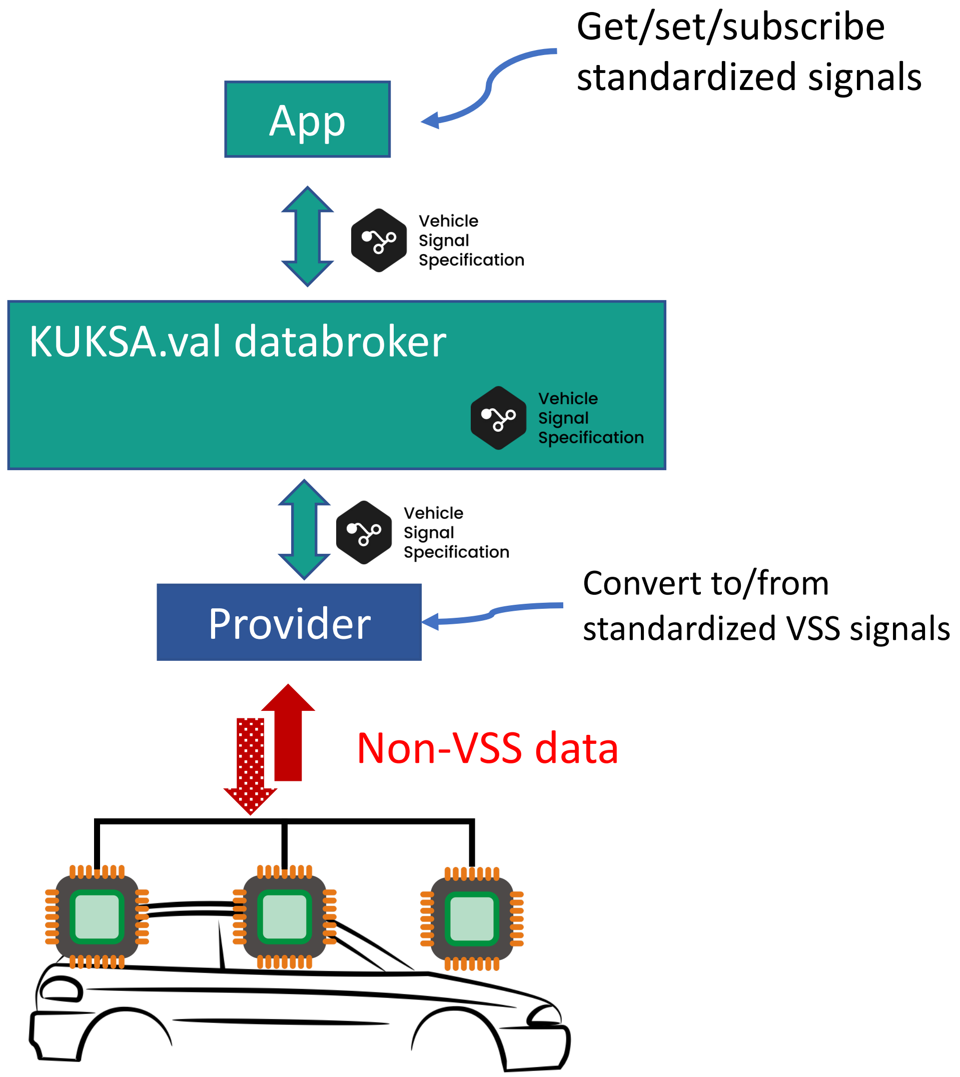
Answer: Convert in a Vehicle Computer*

- Here you can afford the costs of abstraction
- This is the place, where the industry is working on decoupling hard- from software (SDV!)
- Here you save money & effort with more generic/portable software (SDV!)

* Something with a processor and a full blown (POSIX) OS

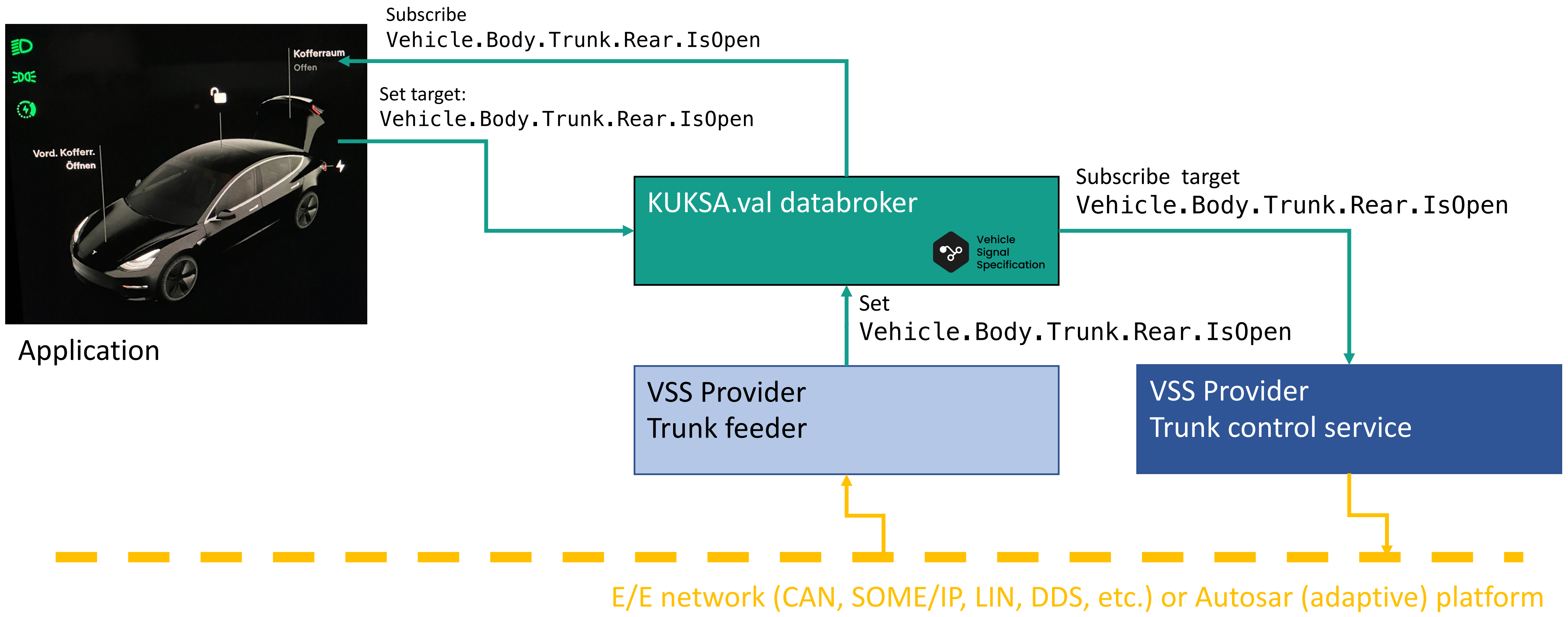

Off-board
Vehicle Computer
Deeply Embedded
Vehicle Signal Specification
Vehicle Signal Specification
App
App
SwC
Vehicle Signal Specification
This is
KUKSA .val

KUKSA.val Scope and Design Choices


Get/set/subscribe standardized signals
App
Vehicle Signal Specification
KUKSA.val databroker
Vehicle Signal Specification
Vehicle Signal Specification
Provider
Convert to/from standardized VSS signals
Non-VSS data

- 100% Open Source Eclipse Project (Apache 2.0 license)
- ”In-vehicle digital twin” based on VSS
- Lightweight (core written in RUST)
- Only providing “current” view (no historic data)
- Easy to use language-agnostic interface (GRPC)
- VSS Providers/Feeders to transform data to VSS

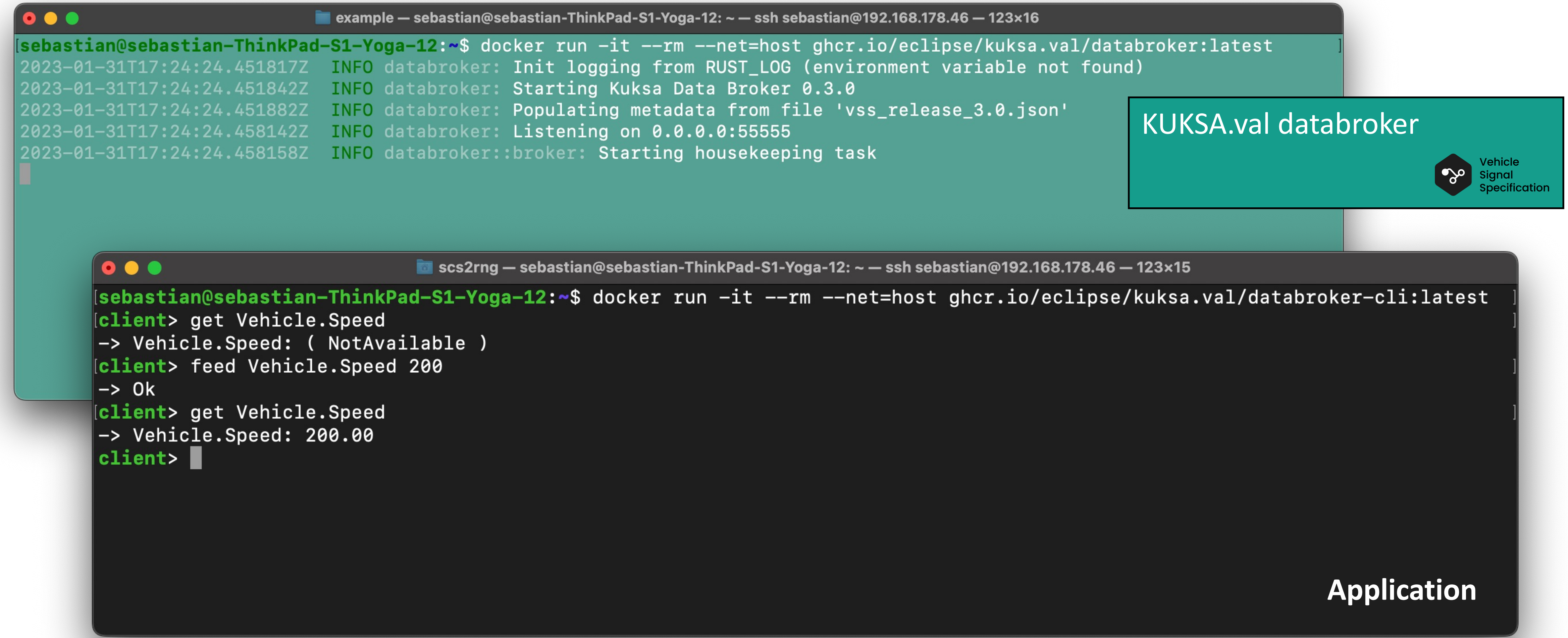
Sensors & Actuators in KUKSA.val



Subscribe
Vehicle.Body.Trunk.Rear.IsOpen
Set target:
Vehicle.Body.Trunk.Rear.IsOpen
Kofferraum
Offen
Vord. Kofferr.
Öffnen
Application
KUKSA.val databroker
Vehicle Signal Specification
Subscribe target
Vehicle.Body.Trunk.Rear.IsOpen
Set
Vehicle.Body.Trunk.Rear.IsOpen
VSS Provider
Trunk feeder
VSS Provider
Trunk control service
E/E network (CAN, SOME/IP, LIN, DDS, etc.) or Autosar (adaptive) platform

(How)does this work?

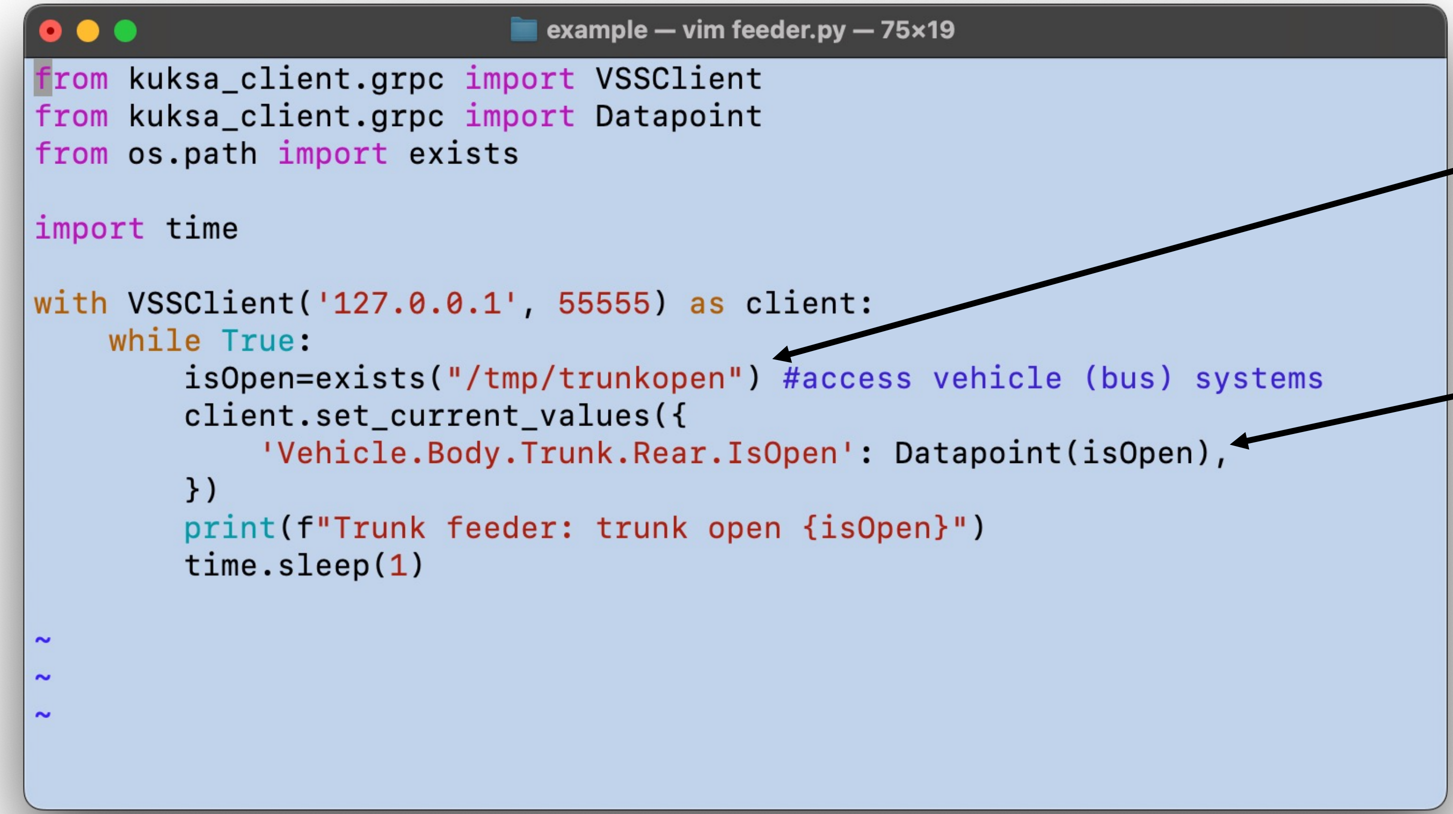
Is this written in Powerpoint, or what?


example — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~ — ssh sebastian@192.168.178.46 — 123×16
KUKSA.val databroker
Vehicle Signal Specification
scs2rng — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~ — ssh sebastian@192.168.178.46 — 123×15
Application

```
sebastian@sebastian-ThinkPad-S1-Yoga-12:~$ docker run -it --rm --net=host ghcr.io/eclipse/kuksa.val/databroker:latest
2023-01-31T17:24:24.451817Z  INFO databroker: Init logging from RUST_LOG (environment variable not found)
2023-01-31T17:24:24.451842Z  INFO databroker: Starting Kuksa Data Broker 0.3.0
2023-01-31T17:24:24.451882Z  INFO databroker: Populating metadata from file 'vss_release_3.0.json'
2023-01-31T17:24:24.458142Z  INFO databroker: Listening on 0.0.0.0:55555
2023-01-31T17:24:24.458158Z  INFO databroker::broker: Starting housekeeping task
```

```
sebastian@sebastian-ThinkPad-S1-Yoga-12:~$ docker run -it --rm --net=host ghcr.io/eclipse/kuksa.val/databroker-cli:latest
client> get Vehicle.Speed
-> Vehicle.Speed: ( NotAvailable )
client> feed Vehicle.Speed 200
-> Ok
client> get Vehicle.Speed
-> Vehicle.Speed: 200.00
client> 
```

VSS Provider: Trunk feeder


example — vim feeder.py — 75×19

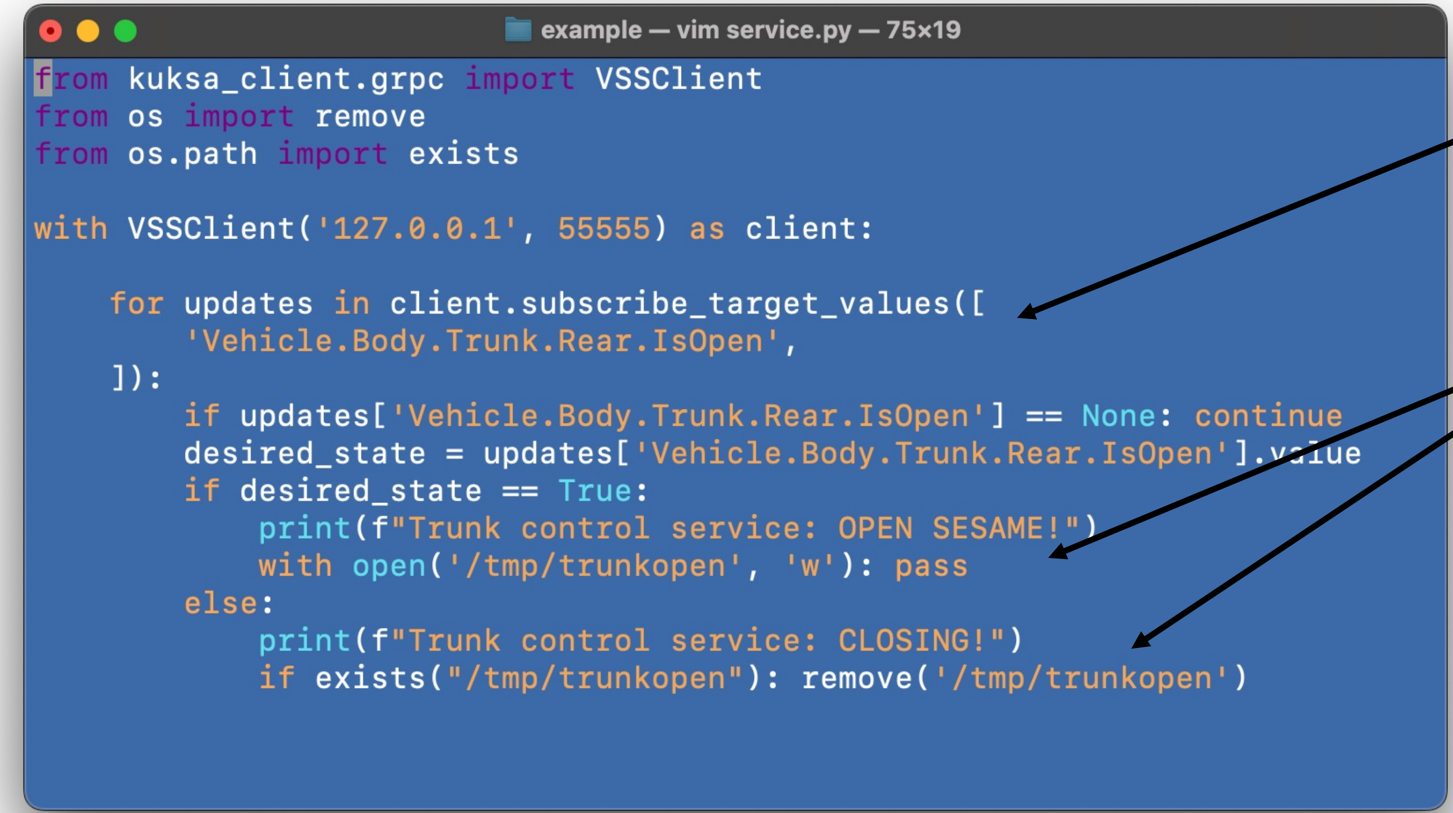
```
from kuksa_client.grpc import VSSClient
from kuksa_client.grpc import Datapoint
from os.path import exists

import time

with VSSClient('127.0.0.1', 55555) as client:
    while True:
        isOpen=exists("/tmp/trunkopen") #access vehicle (bus) systems
        client.set_current_values({
            'Vehicle.Body.Trunk.Rear.IsOpen': Datapoint(isOpen),
        })
        print(f"Trunk feeder: trunk open {isOpen}")
        time.sleep(1)

~
~
~
```

If we detect the trunk is currently open,

Update the current value in KUKSA.val

VSS Provider:
Trunk feeder

VSS Provider: Trunk control service



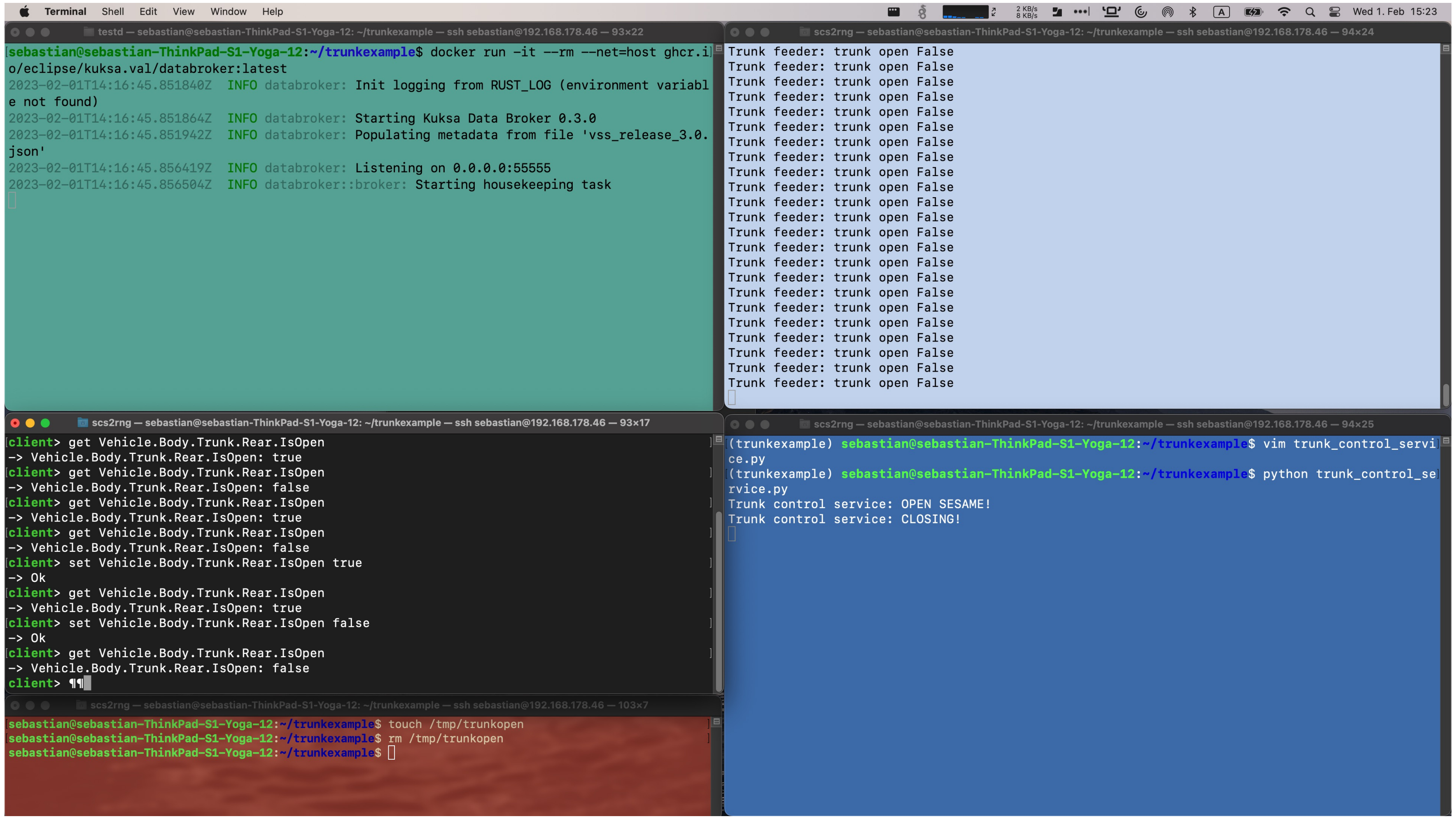
```python
from kuksa_client.grpc import VSSClient
from os import remove
from os.path import exists

with VSSClient('127.0.0.1', 55555) as client:

    for updates in client.subscribe_target_values([
        'Vehicle.Body.Trunk.Rear.IsOpen',
    ]):
        if updates['Vehicle.Body.Trunk.Rear.IsOpen'] == None: continue
        desired_state = updates['Vehicle.Body.Trunk.Rear.IsOpen'].value
        if desired_state == True:
            print(f"Trunk control service: OPEN SESAME!")
            with open('/tmp/trunkopen', 'w'): pass
        else:
            print(f"Trunk control service: CLOSING!")
            if exists("/tmp/trunkopen"): remove('/tmp/trunkopen')
```

If KUKSA.val pushes a new target state,

Interact with vehicle systems to let it happen

VSS Provider:
Trunk control service

Demo



testd — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~/trunkexample — ssh sebastian@192.168.178.46 — 93×22

```
[sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$ docker run -it --rm --net=host ghcr.i
o/eclipse/kuksa.val/databroker:latest
2023-02-01T14:16:45.851840Z  INFO databroker: Init logging from RUST_LOG (environment variabl
e not found)
2023-02-01T14:16:45.851864Z  INFO databroker: Starting Kuksa Data Broker 0.3.0
2023-02-01T14:16:45.851942Z  INFO databroker: Populating metadata from file 'vss_release_3.0.
json'
2023-02-01T14:16:45.856419Z  INFO databroker: Listening on 0.0.0.0:55555
2023-02-01T14:16:45.856504Z  INFO databroker::broker: Starting housekeeping task
```

scs2rng — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~/trunkexample — ssh sebastian@192.168.178.46 — 94×24

```
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
Trunk feeder: trunk open False
```

scs2rng — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~/trunkexample — ssh sebastian@192.168.178.46 — 93×17

```
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: true
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: false
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: true
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: false
client> set Vehicle.Body.Trunk.Rear.IsOpen true
-> Ok
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: true
client> set Vehicle.Body.Trunk.Rear.IsOpen false
-> Ok
client> get Vehicle.Body.Trunk.Rear.IsOpen
-> Vehicle.Body.Trunk.Rear.IsOpen: false
client> ¶¶
```

scs2rng — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~/trunkexample — ssh sebastian@192.168.178.46 — 94×25

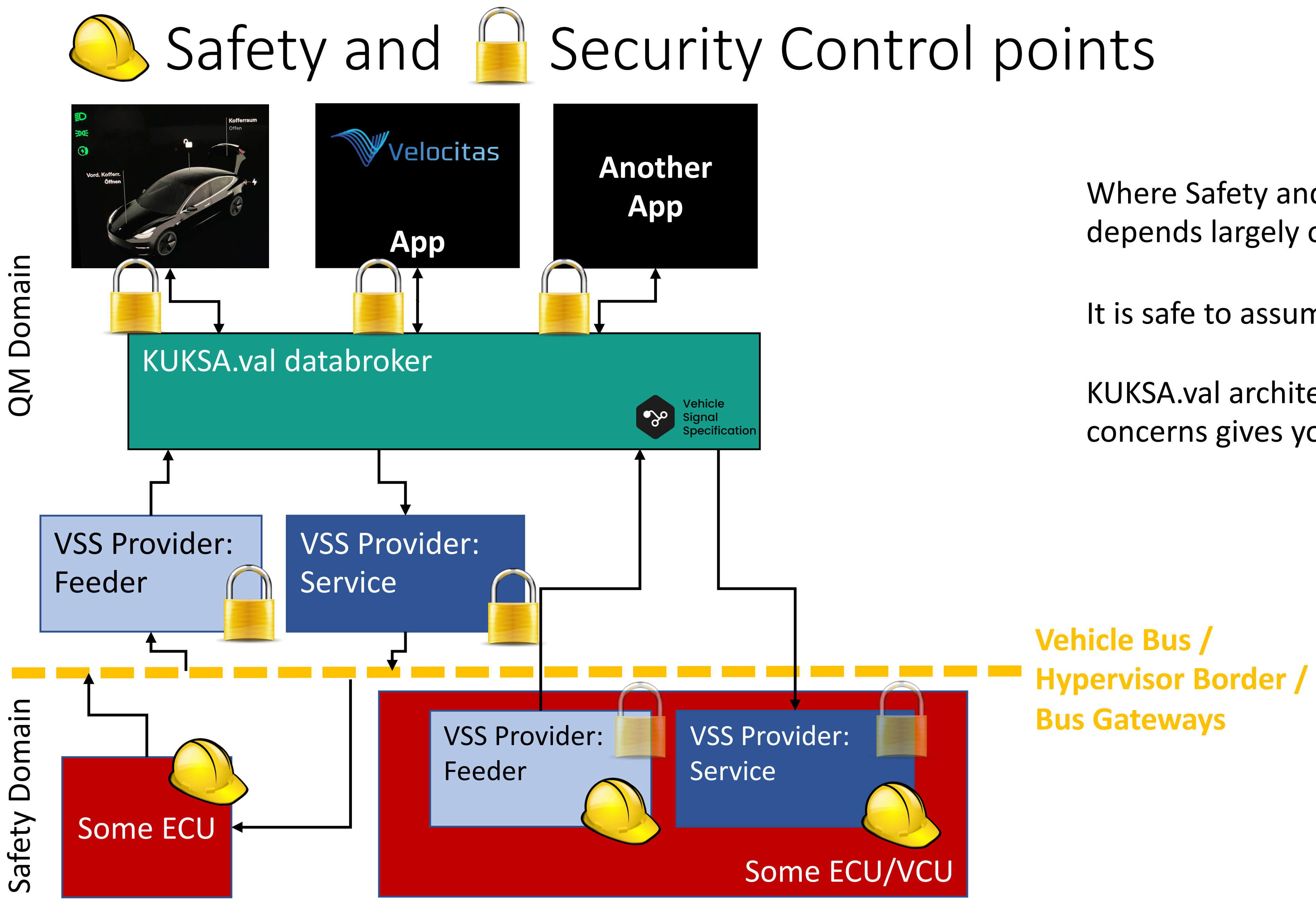
```
(trunkexample) sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$ vim trunk_control_servi
ce.py
(trunkexample) sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$ python trunk_control_se
rvice.py
Trunk control service: OPEN SESAME!
Trunk control service: CLOSING!
```

scs2rng — sebastian@sebastian-ThinkPad-S1-Yoga-12: ~/trunkexample — ssh sebastian@192.168.178.46 — 103×7

```
sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$ touch /tmp/trunkopen
sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$ rm /tmp/trunkopen
sebastian@sebastian-ThinkPad-S1-Yoga-12:~/trunkexample$
```

https://youtu.be/fD6My8za4jY

Safety and Security Control points



Where Safety and Security come into play depends largely on on your application.

It is safe to assume, they *will* play a role.

KUKSA.val architecture and seperation of concerns gives you several control points.

Enabling SDV

Challenge: Letting any application access lower level vehicle systems is insane.

- ✓ KUKSA.val gives you a control point
- ✓ Architecture allows integation of safety controls on different levels depending on your requirements

Challenge: Semantics of Signals very much not standardised. Representation of similar signals in different vehicles are not the same .

- ✓ KUKSA.val leverages standard COVESA VSS signals enabling portable applications

KUKSA

Sa(m|n)e interfaces, faster development.

Stay in contact

Github		https://github.com/eclipse/kuksa.val
Me		http://sdv.expert
Eclipse SDV		https://sdv.eclipse.org
COVESA VSS		https://covesa.github.io/vehicle_signal_specification/
Eclipse Velocitas		https://websites.eclipseprojects.io/velocitas/

Thank you

KUKSA

Sa(m|n)e interfaces, faster development.