VSS – A deeper Dive

Updated Q3 2024 edition

How to build things with VSS - An example

Sebastian Schildt, Erik Jägervall, Adnan Bekan
COVESA AMM, September 25th 2024

COVESA
Accelerating the future of connected vehicles

NOTE

NOTE

There are some command lines in this presentation. It seems when copying them, sometimes some whitespaces are converted not correctly so you get weird error messages like missing arguments. Somewhere ”near” this presentation you should have found an archive that contains safely copyable commands.

In the past…

… or earlier at this AMM

YOU

Where to use VSS?	Wherever you want to!
What of VSS do I need to support?	Whatever you want to!
What is VSS even?	Amazing. It is not an API, Serialization or protocol though.
Which software can I use?	Many!
I am confused	**JOIN COVESA**

This changes now

We show you an example[1,2] of how to use VSS

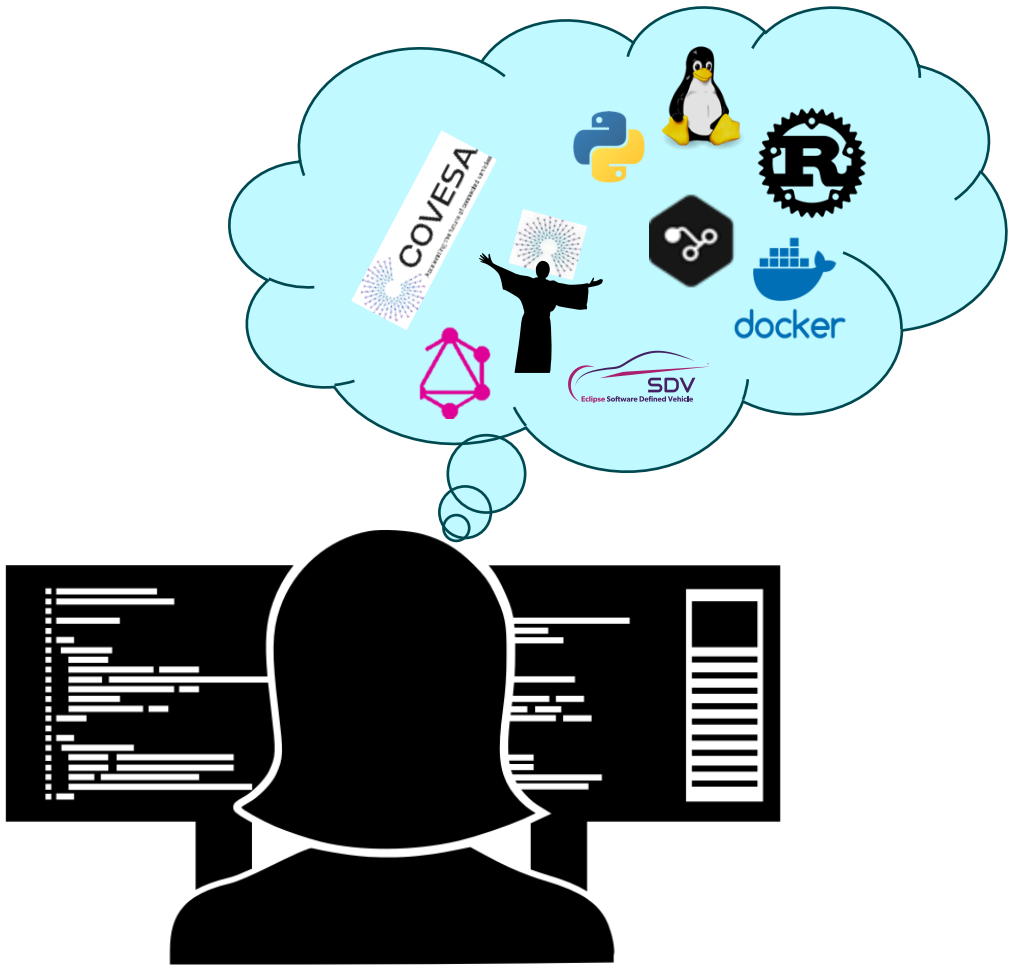
[1] We will…

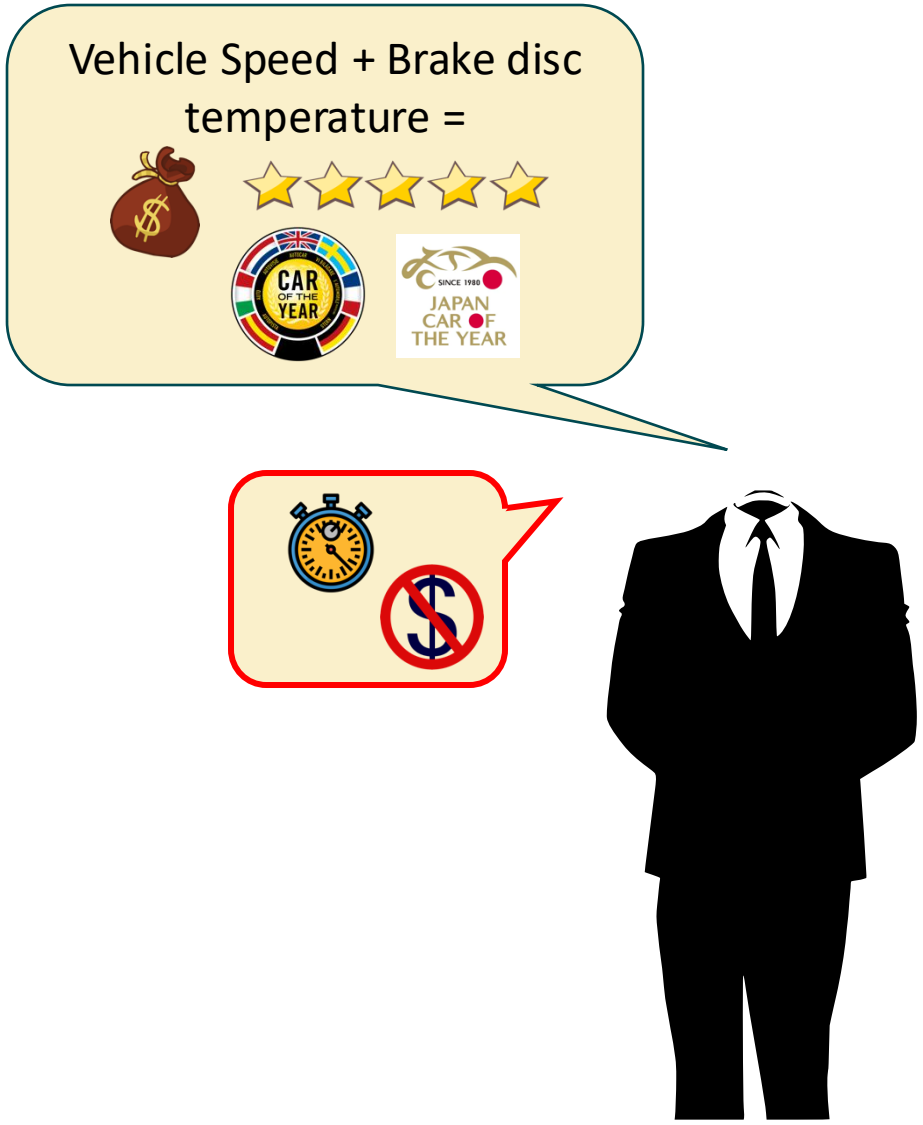
- look at and modify VSS files
- apply the VSS overlay concept
- use VSS tools
- use real CAN Data
- write real code interacting with VSS

[2] We will not…

- look at every tool or framework supporting VSS
- show all domains (not too much cloud in here)

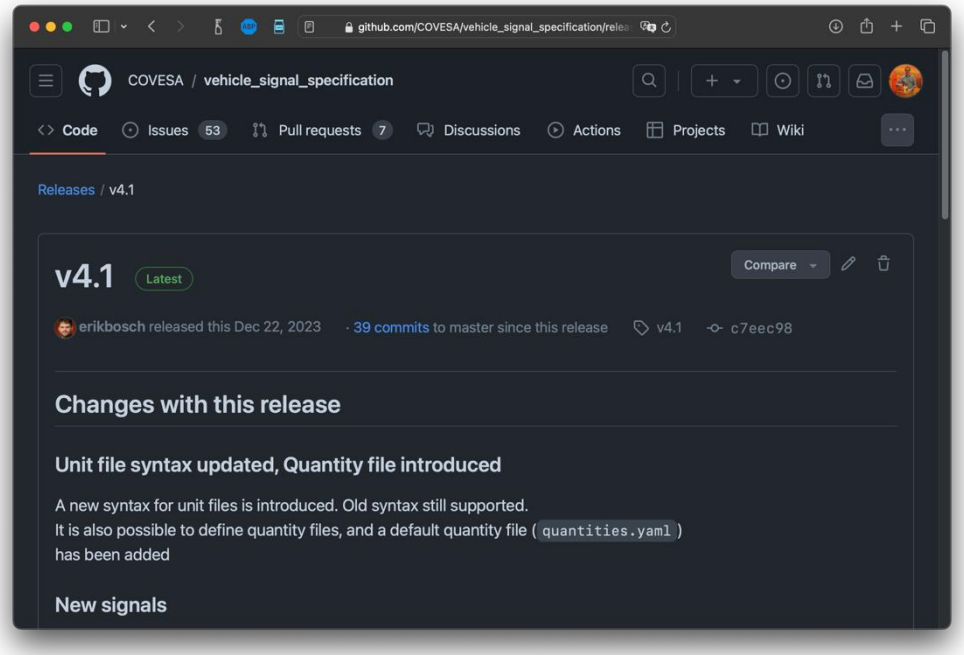
The idea

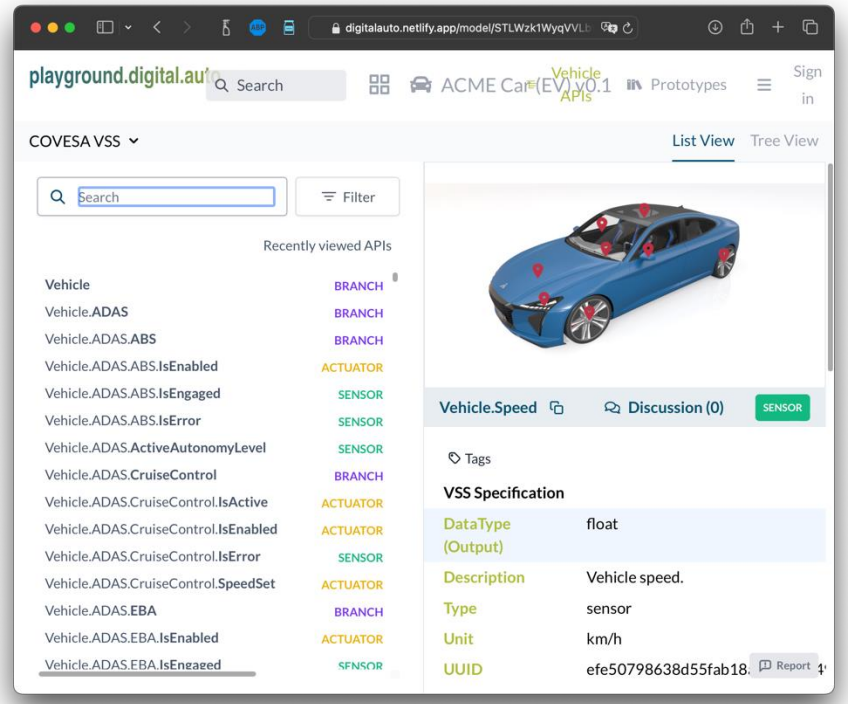




COVESA
Accelerating the future of connected vehicles

VSS – We use VSS already, are we covered?

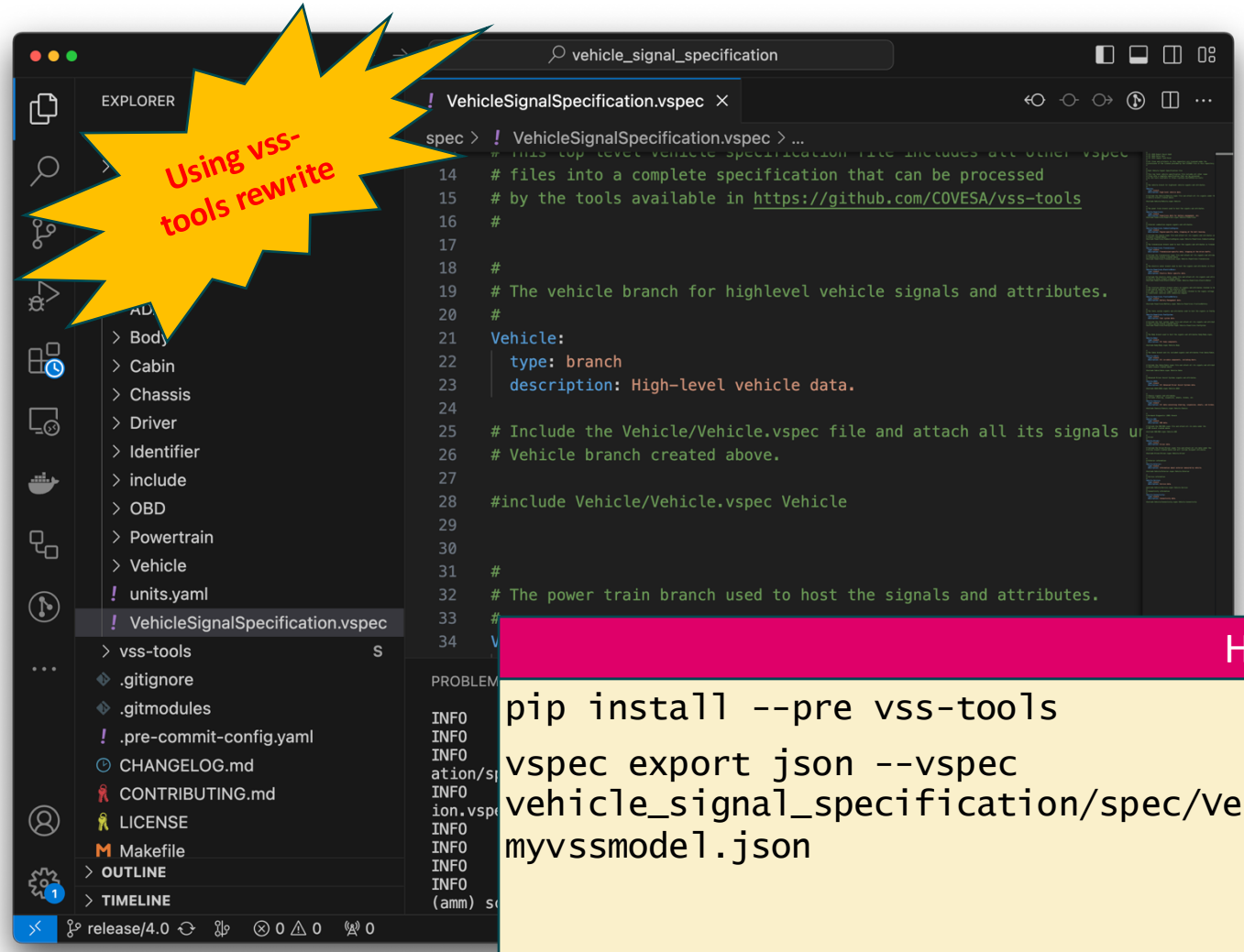





```
Vehicle.Speed
Vehicle.Chassis.Axle.Rowx.Wheel.L/R.Brake.
```
??? No temperatures

https://github.com/COVESA/vehicle_signal_specification/releases
https://digitalauto.netlify.app

Convert VSS model



Good to know

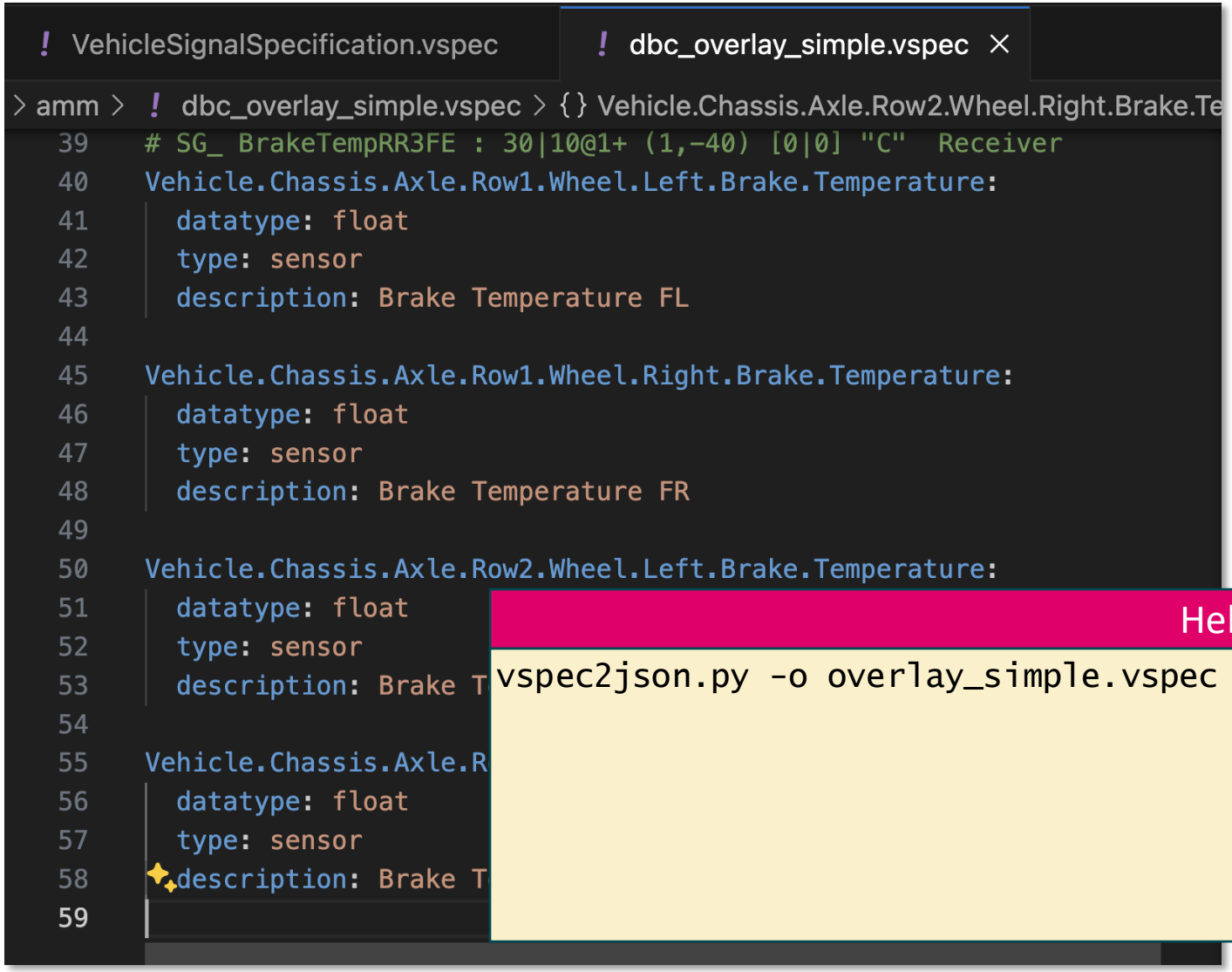
- COVESA vss-tools convert .vspec files to a variety of other formats (json, cs, ddsidl, etc.) for use in different tech stacks
- You can easily write your own converters

Helpful commands

```
pip install --pre vss-tools

vspec export json --vspec
vehicle_signal_specification/spec/VehicleSignalSpecification.vspec --output
myvssmodel.json
```


Add additional signal in overlay



Good to know

- You can *add* signals to any VSS model using an overlay
- An overlay resides in seperate files, so this a plus managing (composable) VSS models in your development workflow

Helpful commands

```
vspec2json.py -o overlay_simple.vspec ./vss_rel_4.1.yaml myextvss.json
```


Use VSS in-vehicle with KUKSA

amm — sebastian@docker: ~ — pipenv shell ▸ docker — 104×32

```
                    kuksa-client CLI
                    0.4.2

Default tokens directory: /kuksa-client/_internal/kuksa_client/kuksa_server_certificates/jwt

Connecting to VSS server at 127.0.0.1 port 55555 using KUKSA GRPC protocol.
TLS will not be used.
INFO 2024-04-05 09:33:53,241 kuksa_client.grpc No Root CA present, it will not be possible to use a secu
re connection!
INFO 2024-04-05 09:33:53,241 kuksa_client.grpc.aio Establishing insecure channel
gRPC channel connected.
Test Client> setValue Vehicle.Speed 42
OK

Test Client> getValue Vehicle.S
{
    "path": "Vehicle.Speed",
    "value": {
        "value": 42.0,
        "timestamp": "2024-04-0
    }
}

Test Client>
```

Good to know

- KUKSA is an in-vehicle server for VSS signals
- It is an Eclipse SDV Open Source project under Apache-2.0 license (commercial support available)

Helpful commands

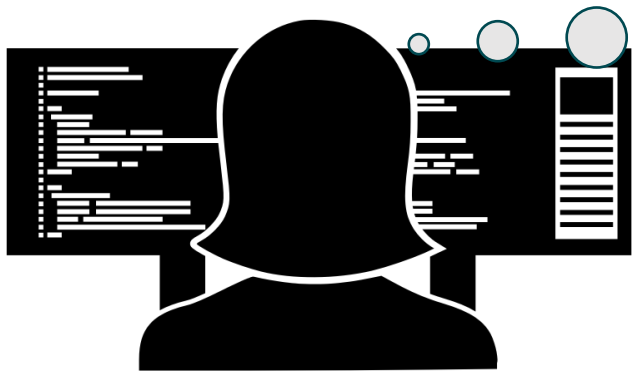
```
docker run -it --rm --net=host ghcr.io/eclipse-kuksa/kuksa-databroker:latest

docker run -it --rm --net=host ghcr.io/eclipse-kuksa/kuksa-python-sdk/kuksa-client:latest

docker run -it --rm --net=host -v $(pwd):/data ghcr.io/eclipse-kuksa/kuksa-databroker:latest --vss /data/myvssmodel.json
```

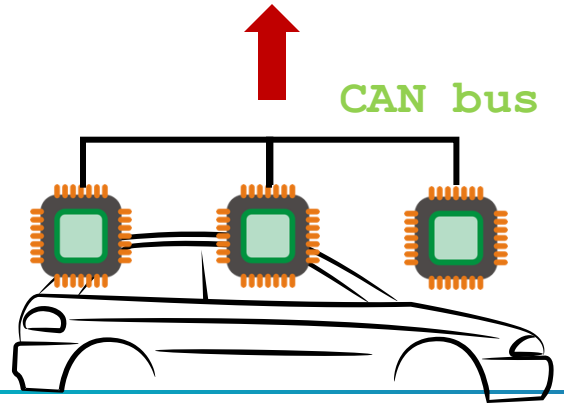

CAN – Where is my data in a vehicle?

Let’s assume raw sensor data is available on a CAN bus, but likely not in ”VSS format”
CAN signals may be described in DBC format



Vehicle.Speed ?

Vehicle.Chassis.Axle.Row1.Wheel.
Right.Brake.Temperature ?



github.com/eclipse-kuksa/kuksa-can-prov

Files | main | kuksa-can-provider / Model3CAN.dbc | Top | Edit this file | Raw

```
                                                [0|102.3] "%"  Receiver
                                                0|102.3] "%"  Receiver
                                                .0) [0|100] "%"  Receiver
                                                (1,0) [0|255] ""  Receiver
                                                4@1+ (1,0) [0|15] ""  Receiver
                                                9@1+ (1,0) [0|510] ""  Receiver
                                                DI_uiSpeedHighSpeed : 34|9@1+ (1,0) [0|510] ""  Receiver
2660      SG_ DI_uiSpeedUnits : 33|1@1+ (1,0) [0|1] ""  Receiver
2661      SG_ DI_vehicleSpeed : 12|12@1+ (0.08,-40) [-40|285] "kph"  Receiver
2662
2663    BO_ 680 ID2A8CMPD_state: 8 VehicleBus
2664      SG_ CMPD_inputHVCurrent : 32|9@1+ (0.1,0) [0|50] "A"  Receiver
2665      SG_ CMPD_inputHVPower : 21|11@1+ (10,0) [0|20000] "W"  Receiver
2666      SG_ CMPD_inputHVVoltage : 41|11@1+ (0.5,0) [0|1000] "V"  Receiver
2667      SG_ CMPD_powerLimitActive : 55|1@1+ (1,0) [0|1] ""  Receiver
2668      SG_ CMPD_powerLimitTooLowToStart : 62|1@1+ (1,0) [0|1] ""  Receiver
2669      SG_ CMPD_ready : 63|1@1+ (1,0) [0|1] ""  Receiver
2670      SG_ CMPD_speedDuty : 11|10@1+ (0.1,0) [0|100] "%"  Receiver
2671      SG_ CMPD_speedRPM : 0|11@1+ (10,0) [0|20000] "RPM"  Receiver
2672      SG_ CMPD_state : 56|4@1+ (1,0) [0|15] ""  Receiver
2673      SG_ CMPD_wasteHeatState : 60|2@1+ (1,0) [0|3] ""  Receiver
2674
2675    BO_ 1029 ID405VIN: 8 VehicleBus
2676      SG_ VINB405 m17 : 8|56@1+ (1,0) [0|7.20576E+016] ""  Receiver
2677      SG_ VINC405 m18 : 8|56@1+ (1,0) [0|7.20576E+016] ""  Receiver
2678      SG_ VINA405 m16 : 8|56@1+ (1,0) [0|7.20576E+016] ""  Receiver
```

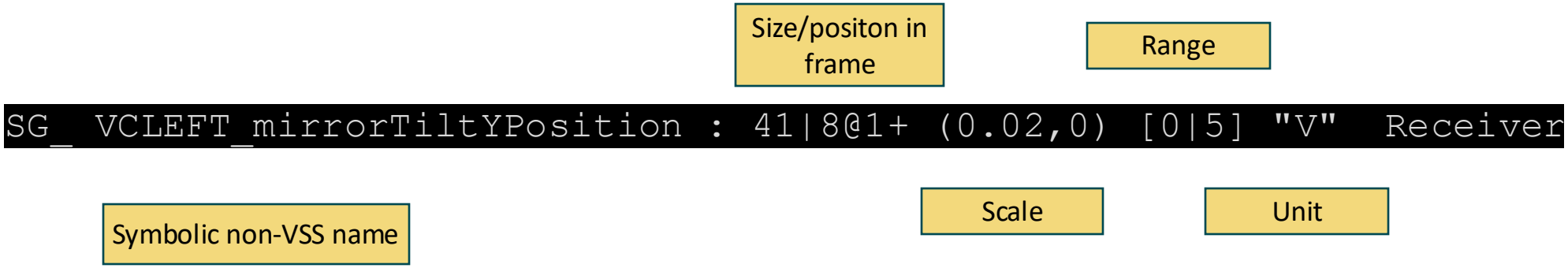
Deeply Embedded

CAN DBC

Many automotive busses and communication protocols use description of the data

.

- DBC for CAN, ARXML for various AUTOSAR communication needs, DDS-IDL for DDS systems etc.
- We focus on DBC for CAN



Can we describe the relation to VSS signals?

DBC overlay: Describe VSS-DBC relation in VSS

```
Vehicle.Body.Mirrors.DriverSide.Tilt:
  datatype: int8
  type: actuator
  dbc2vss:
    signal: VCLEFT_mirrorTiltYPosition
    transform:
      math: "floor((x*40)-100)"
```

Good to know

- VSS allows you to define custom metadata that will be processed by VSS-tools

Standard VSS definition

Reference DBC signal name

Apply transform
(example converts 0..5V into -100/+100 range demanded by VSS)

Helpful commands

```
vspec2json.py -o overlay_simple.vspec -o ./dbc_overlay.vspec -e dbc2vss --json-
./vss_rel_4.1.yaml myextvsswithdbc.json
```

Variants?

```
Vehicle.Body.Mirrors.DriverSide.Tilt:
  datatype: int8
  type: actuator
  dbc2vss:
    signal: VCLEFT_mirrorTiltYPosition
    transform:
      math: "floor((x*40)-100)"
```

Good to know

- VSS allows you to define custom metadata that will be processed by VSS-tools

Standard VSS definition

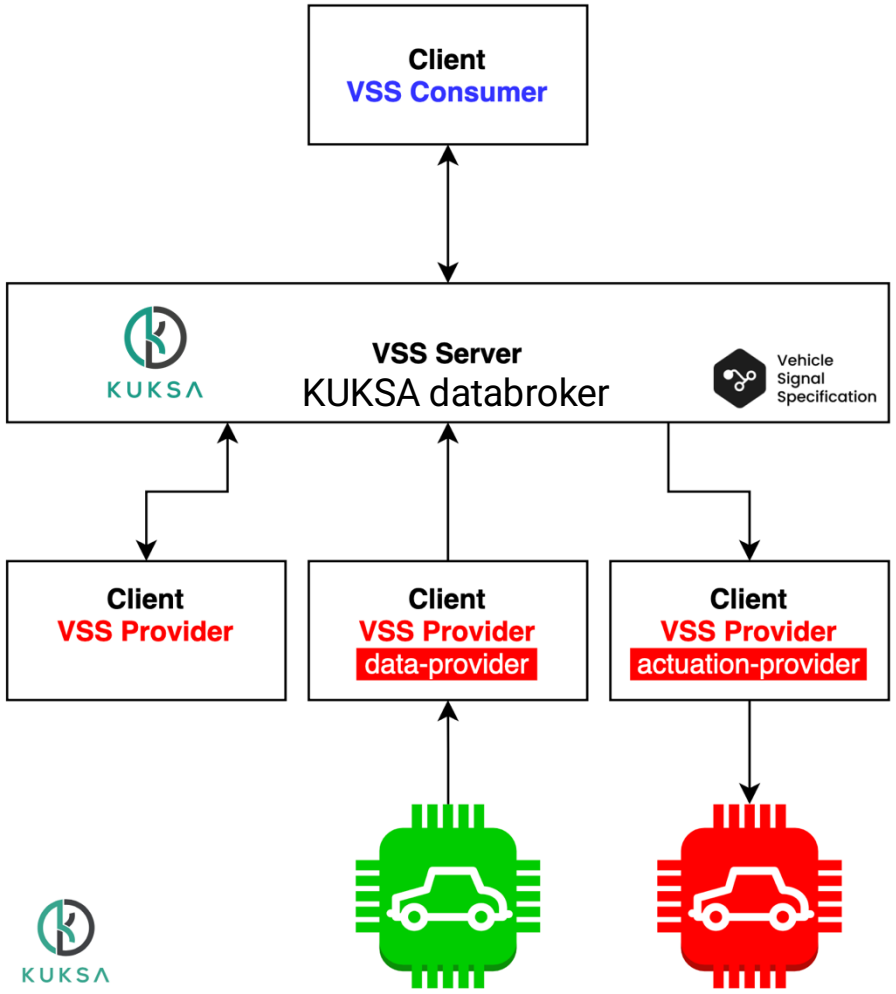
Reference DBC signal name

Apply transform
(example converts 0..5V into -100/+100 range demanded by VSS)

"tilt sensor A" "tilt sensor B" "without tilt"

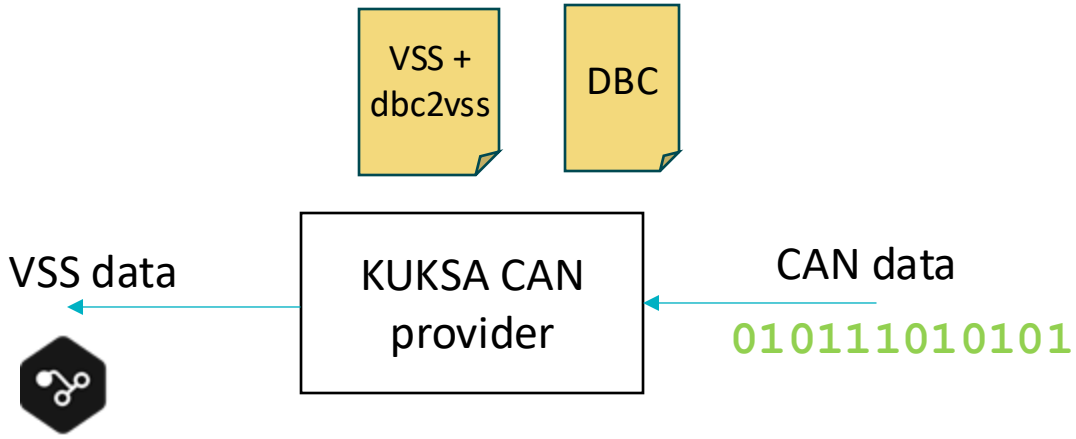
VEHICLE SPECIFIC MAPPINGS

KUKSA CAN provider

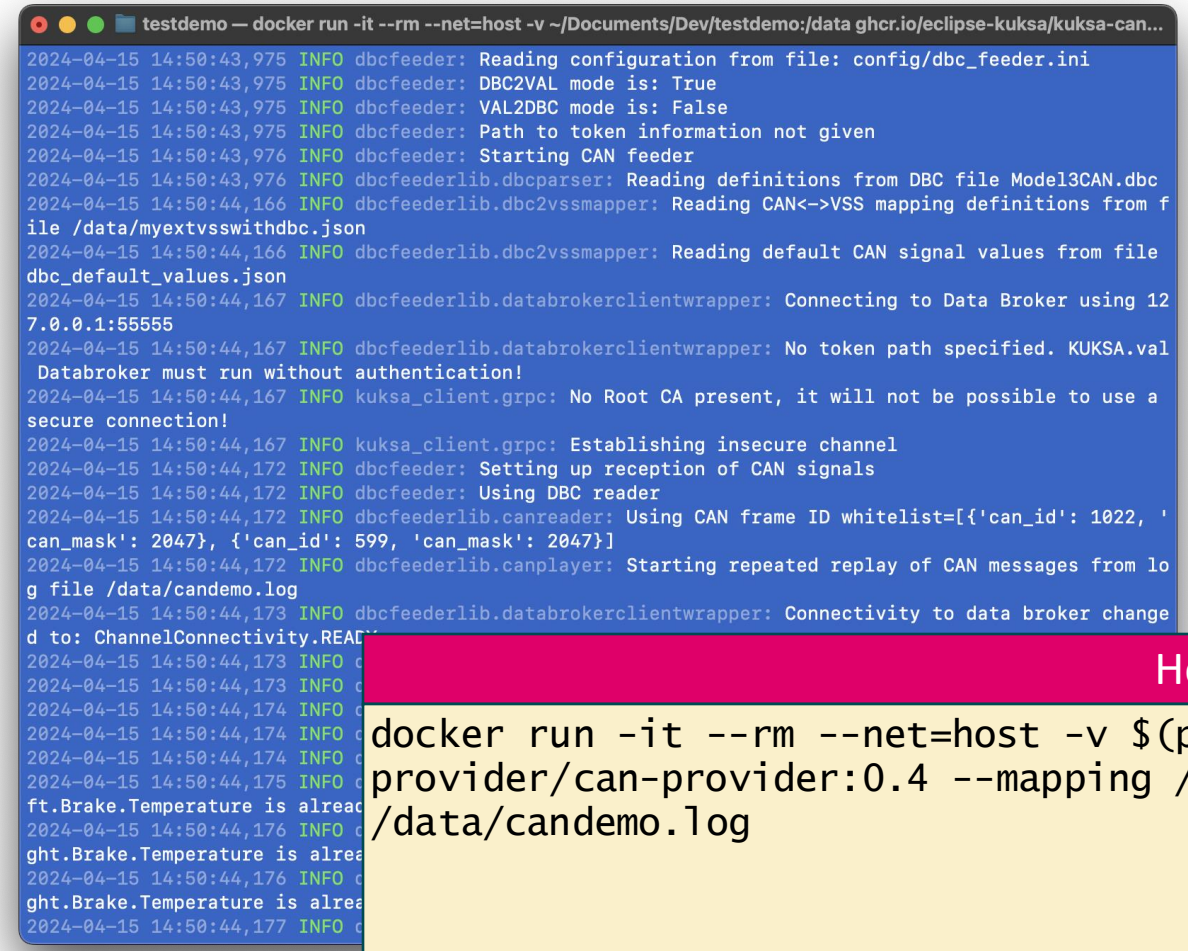


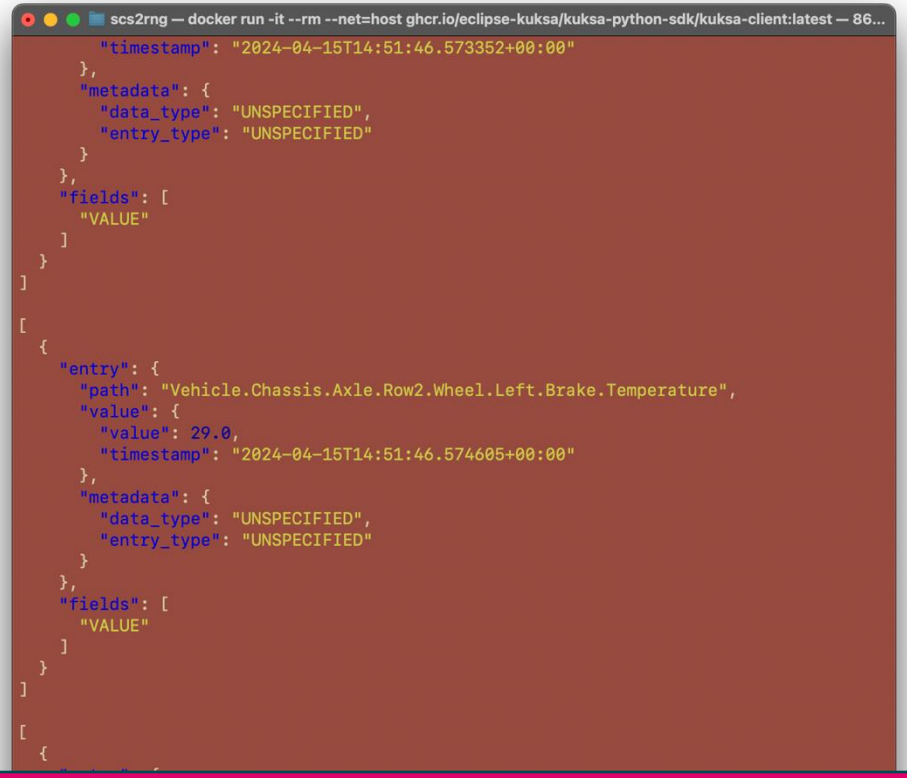
VSS provider syncs of the vehicle with VSS model of the server

- data-provider makes sure that the actual state of a vehicle is represented in VSS (historically known as “feeder”)
- actuation-provider ensure that the target value of a VSS actuator is reflected by the actual state of a vehicle



Running CAN provider





Helpful commands

```
docker run -it --rm --net=host -v $(pwd):/data ghcr.io/eclipse-kuksa/kuksa-can-provider/can-provider:0.4 --mapping /data/myextvsswithdbc.json --dumpfile /data/candemo.log
```

Making an App

```python
async def present():
    while True:
        print("⚙ Processed: " + str(nbr_vss_signals) + " VSS signals!")
        print("🩸 FL brake: " + str(vss_values['Vehicle.Chassis.Axle.Row1.Wheel.Left.Brake.Tempe
        print("🩸 FR brake: " + str(vss_values['Vehicle.Chassis.Axle.Row1.Wheel.Right.Brake.Temp
        print("🩸 RL brake: " + str(vss_values['Vehicle.Chassis.Axle.Row2.Wheel.Left.Brake.Tempe
        print("🩸 RR brake: " + str(vss_values['Vehicle.Chassis.Axle.Row2.Wheel.Right.Brake.Temp
        print("🚀 Max speed since last time: " + str(vss_values['Vehicle.Speed']))
        reset_vss_values()
        await asyncio.sleep(5)
```

Good to know

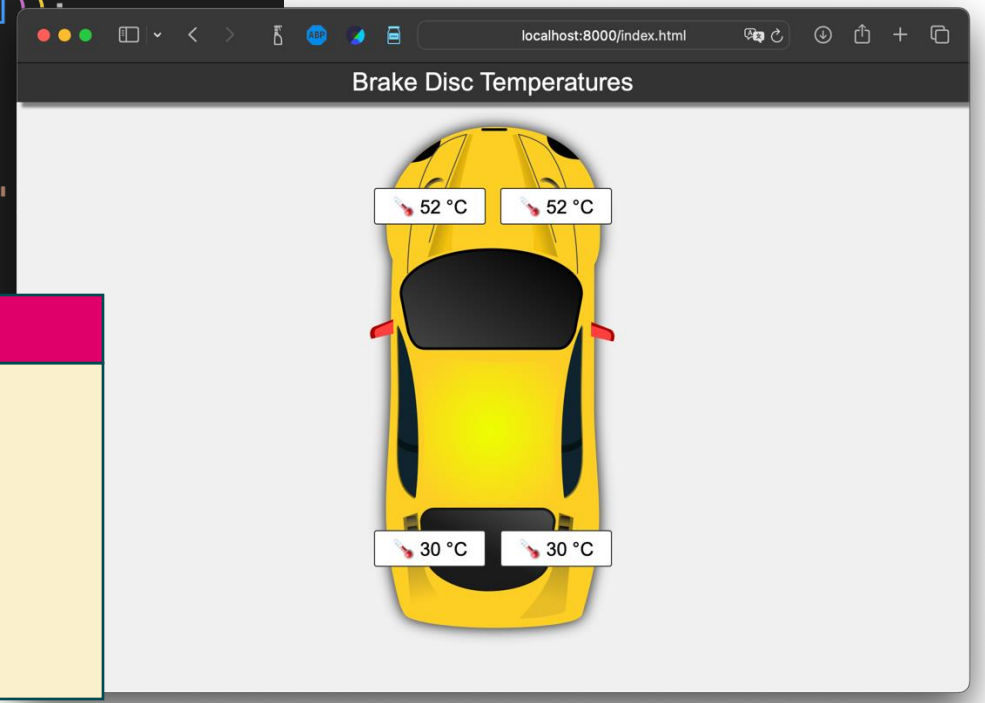
- KUKSA Python SDK is an easy way to use the KUKSA GRPC API
- You can use GRPC directly in any supported language
- You can use the KUKSA Android SDK for Smartphones, Tablets and Android Automotive

Helpful commands

```
pip install kuksa-client
python app.py
```


Make a Web App/PWA


```
function parseData(js) {
    if ( js.hasOwnProperty("subscriptionId") ) {
        if (js['subscriptionId'] == SUB_ID_SPEED) {
            setSpeed(js['data']['dp']['value']);
            return;
        }
        else if (js['subscriptionId'] == SUB_ID_FL) {
            $('.front-left').html(tempTostr(js['data']['dp']['value']));
            return;
        }
        else if (js['subscriptionId'] == SUB_ID_FR) {
            $('.front-right').html(tempTostr(js['data']['dp']['value'
            return;
        }
```

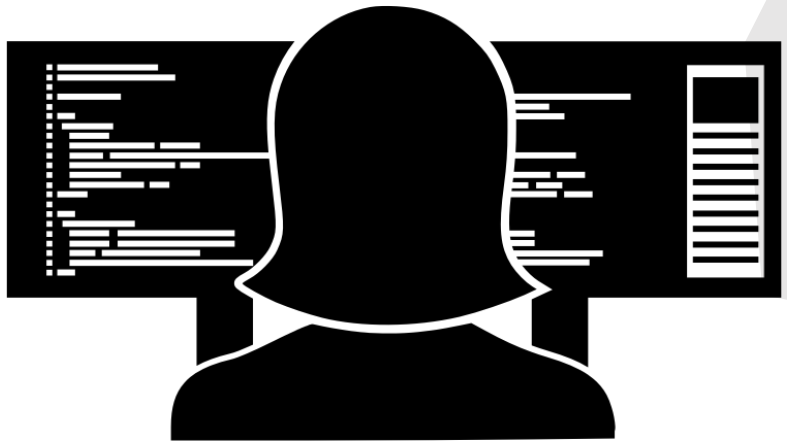


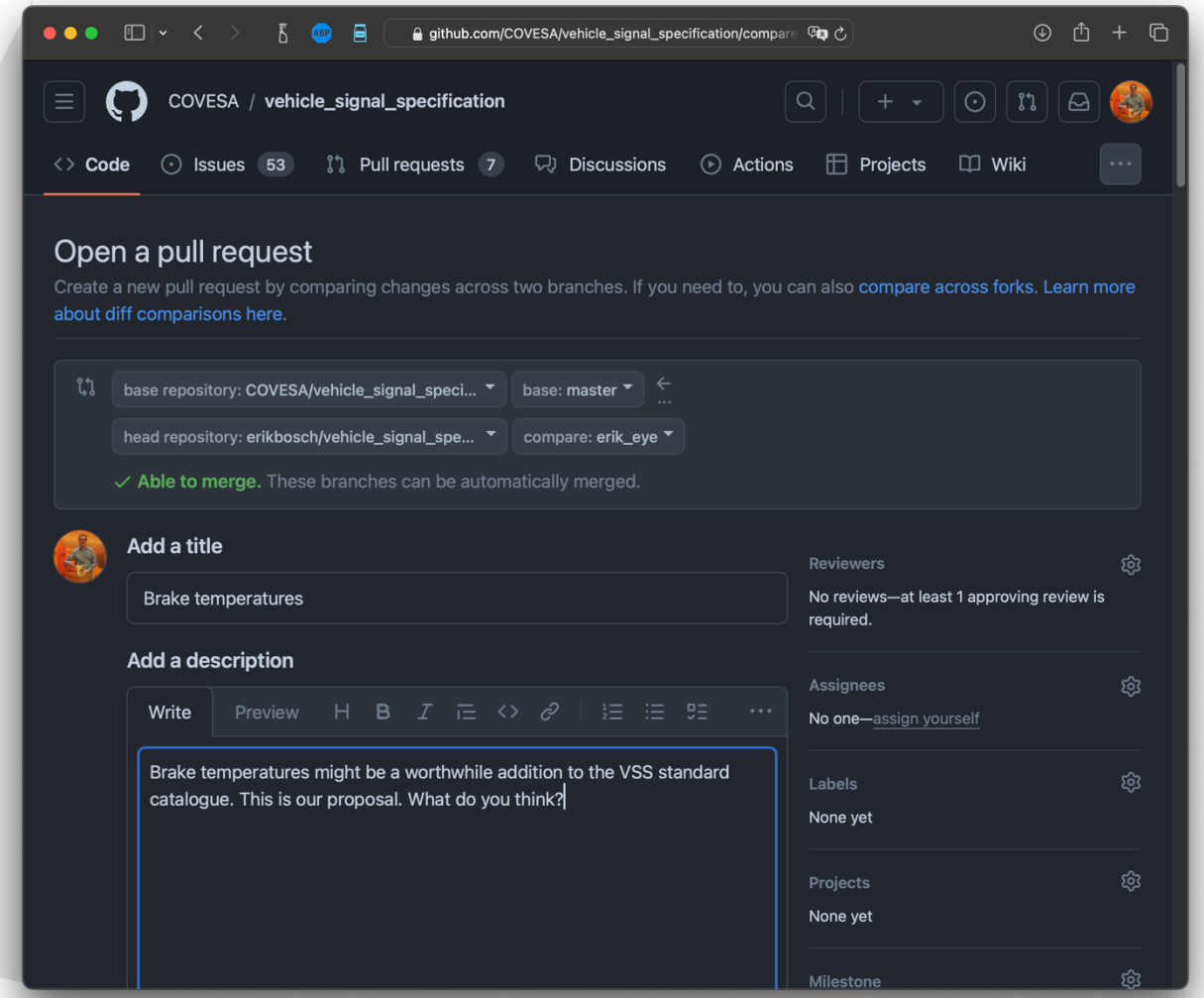
Helpful commands

```
python -m http.server
```

Happy End

Would have been even easier if the signal had already been there

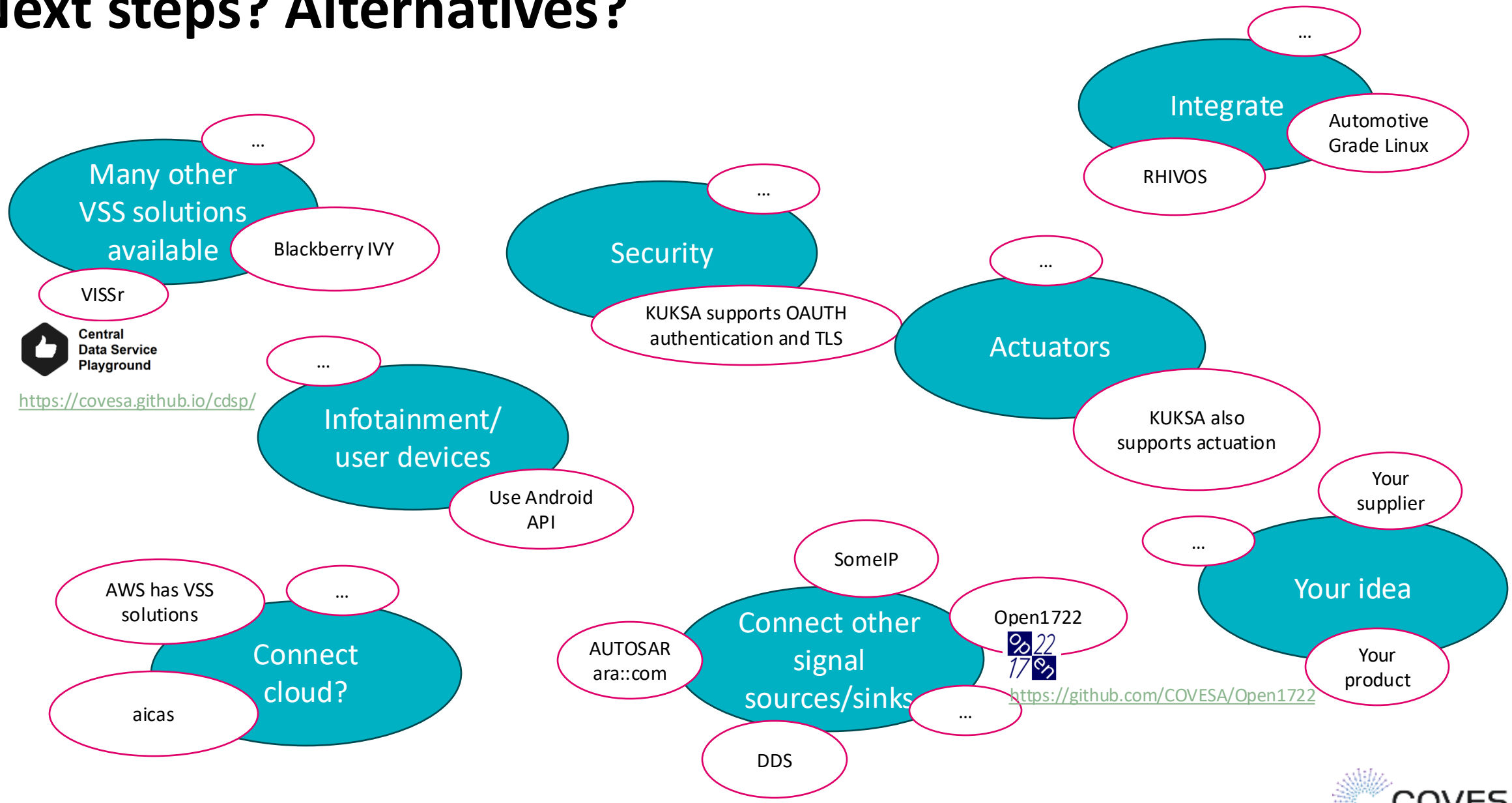




Summary

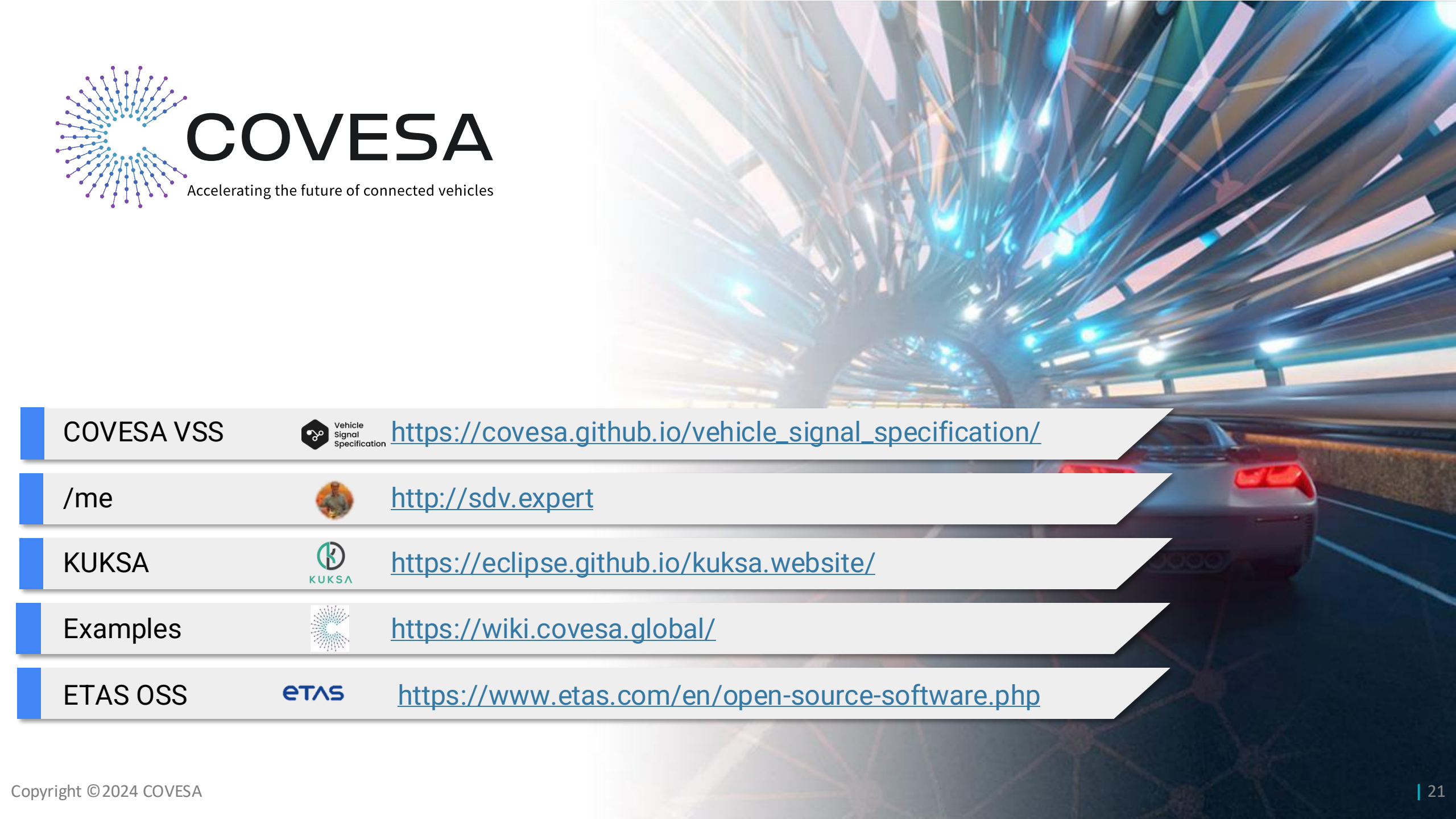
We have seen
▪ How to use the vss-tools to convert VSS into other useful formats ▪ How to add custom VSS signals and converting them using the tools ▪ How to model and add custom VSS metadata VSS signals and converting them using the tools ▪ How to run the Eclipse KUKSA software to work with VSS signals in-vehicle ▪ How to use a CAN provider automatically converting raw CAN signals into valid VSS signals using a configuration ▪ How to write an application using VSS signals ▪ How to create a webpage visualizing VSS data

Next steps? Alternatives?



COVESA

Accelerating the future of connected vehicles



COVESA VSS	Vehicle Signal Specification	https://covesa.github.io/vehicle_signal_specification/
/me		http://sdv.expert
KUKSA	KUKSA	https://eclipse.github.io/kuksa.website/
Examples		https://wiki.covesa.global/
ETAS OSS	ETAS	https://www.etas.com/en/open-source-software.php