ElixirConf EU

Retrofitting a car and running it with Elixir

Thibault Poncelet

Marc Lainez

Loïc Vigneron

ElixirConf EU

Who we are

Marc

Loïc

Thibault

2013: Founded Spin42

- 2017: First Elixir project by Loïc (Elicopter Elixirconf EU 2017)
  - 2016-2023: Founded and sold Ibanity (thank you Elixir...)
    - 2024: Sabbatical break, playing around with cars...

The problem

Automotive world is closed

Car's average lifespan is 8-12 years although a lot is still usable/salvageable

Aftermarket software support and public doc inexistent (future cybersecurity risk?)

Parts from different brands don't work together (to low compatibility between brands)

OVCS

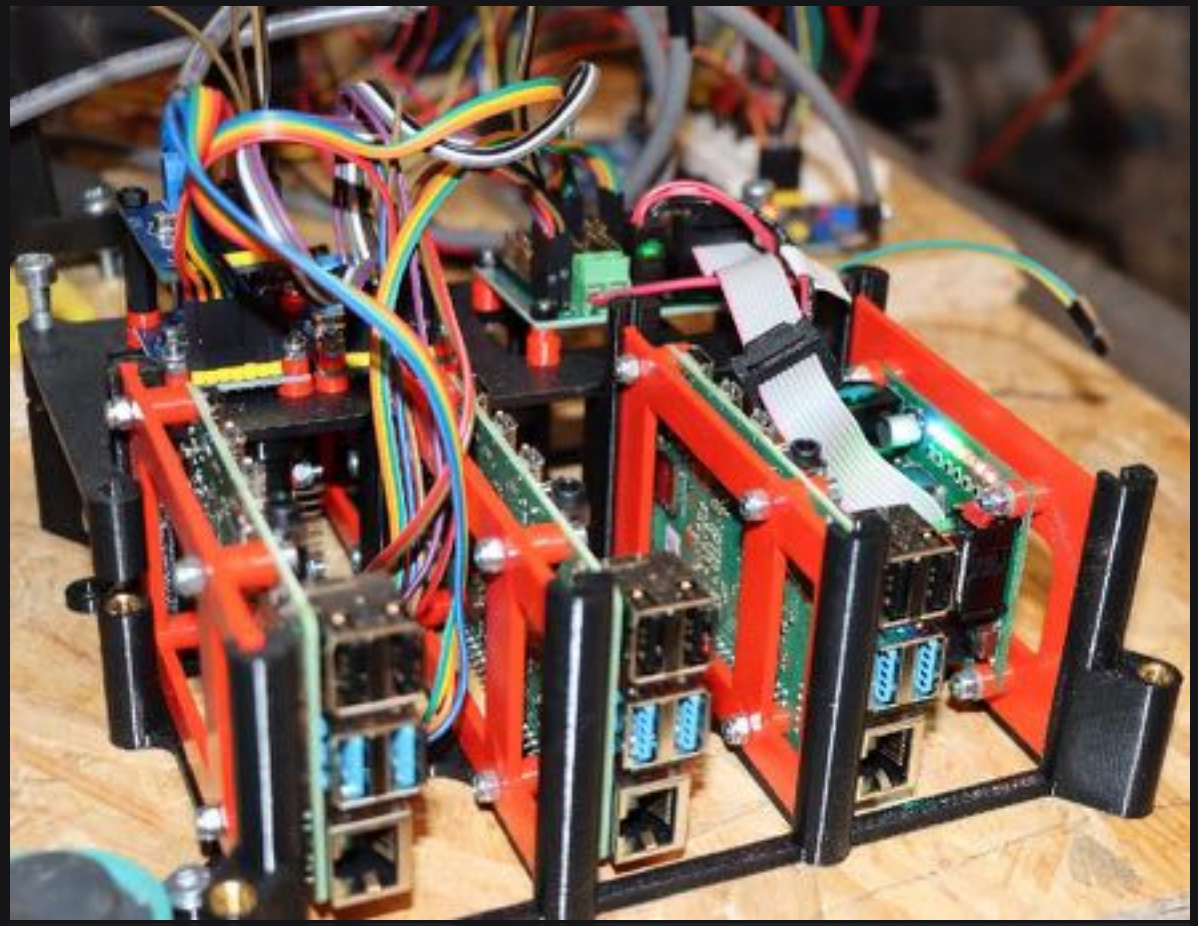
Open vehicle control system

Open software and open hardware components

Abstraction layer for vehicles hardware and software

More developer friendly than traditional automotive technologies

Allows prototyping and monitoring for vehicle projects



The first step

2007 Polo Bluemotion

+

EM57 2013 Nissan Leaf Motor

ElixirConf EU

Disclaimer

NOT ROAD SAFE YET

Use at your own risk, we are not responsible…

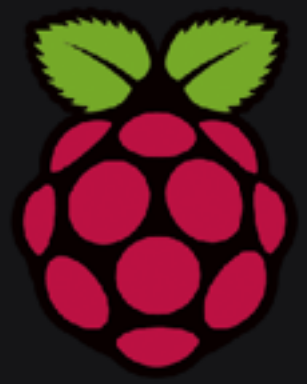
ElixirConf EU

DESIGN CHOICES

BUILDING BLOCKS

EVERYTHING TOGETHER

FURTHER WORK



RPI 4 Nerves R4 Minima

Nothing automotive grade yet...

```
const post_calibration_enabled = (calibrationEnabled) => {
    return axios.post(import.meta.env.VITE_BASE_URL + "/api/calibration", {
      calibrationModeEnabled: calibrationEnabled,
    });
  }
```

```
defmodule VmsApiWeb.Api.CalibrationController do
 ...
  def create(conn, %{"calibrationModeEnabled" => true} = _params) do
    :ok = ControlsController.enable_calibration_mode()
    ....
  end
```

```
defmodule VmsCore.Controllers.ControlsController do
  def enable_calibration_mode() do
    GenServer.call(__MODULE__, :enable_calibration)
  end

 @impl true
  def handle_call(:enable_calibration, _from, state) do
    state = put_in(state, [:car_controls, :calibration_status], "started")
    {:reply, :ok, state}
  end
  ...
```

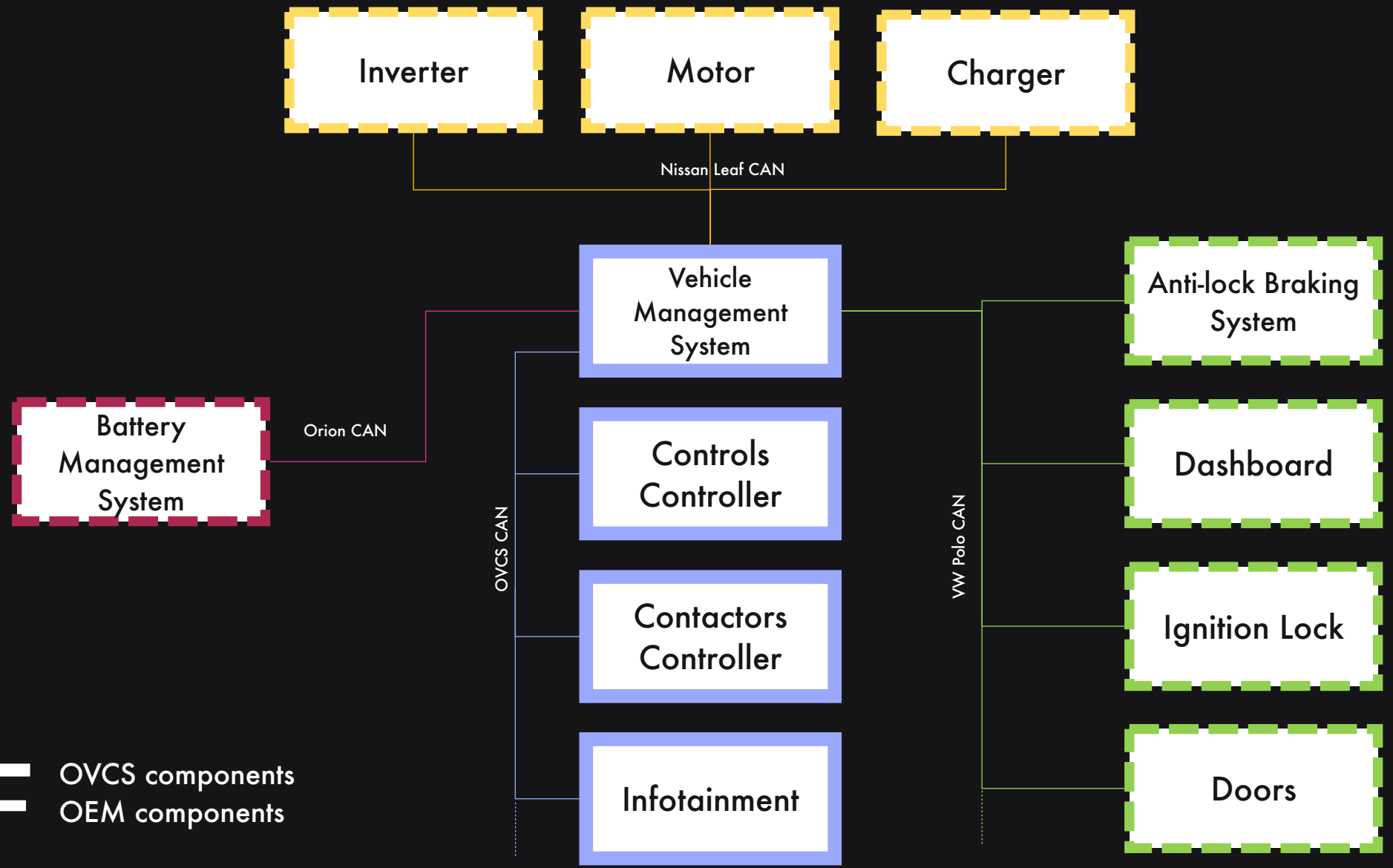
ElixirConf EU

DESIGN CHOICES

BUILDING BLOCKS

EVERYTHING TOGETHER

FURTHER WORK


Inverter
Motor
Charger
Nissan Leaf CAN
Vehicle Management System
Battery Management System
Orion CAN
OVCS CAN
Controls Controller
Contactors Controller
Infotainment
VW Polo CAN
Anti-lock Braking System
Dashboard
Ignition Lock
Doors
OVCS components
OEM components

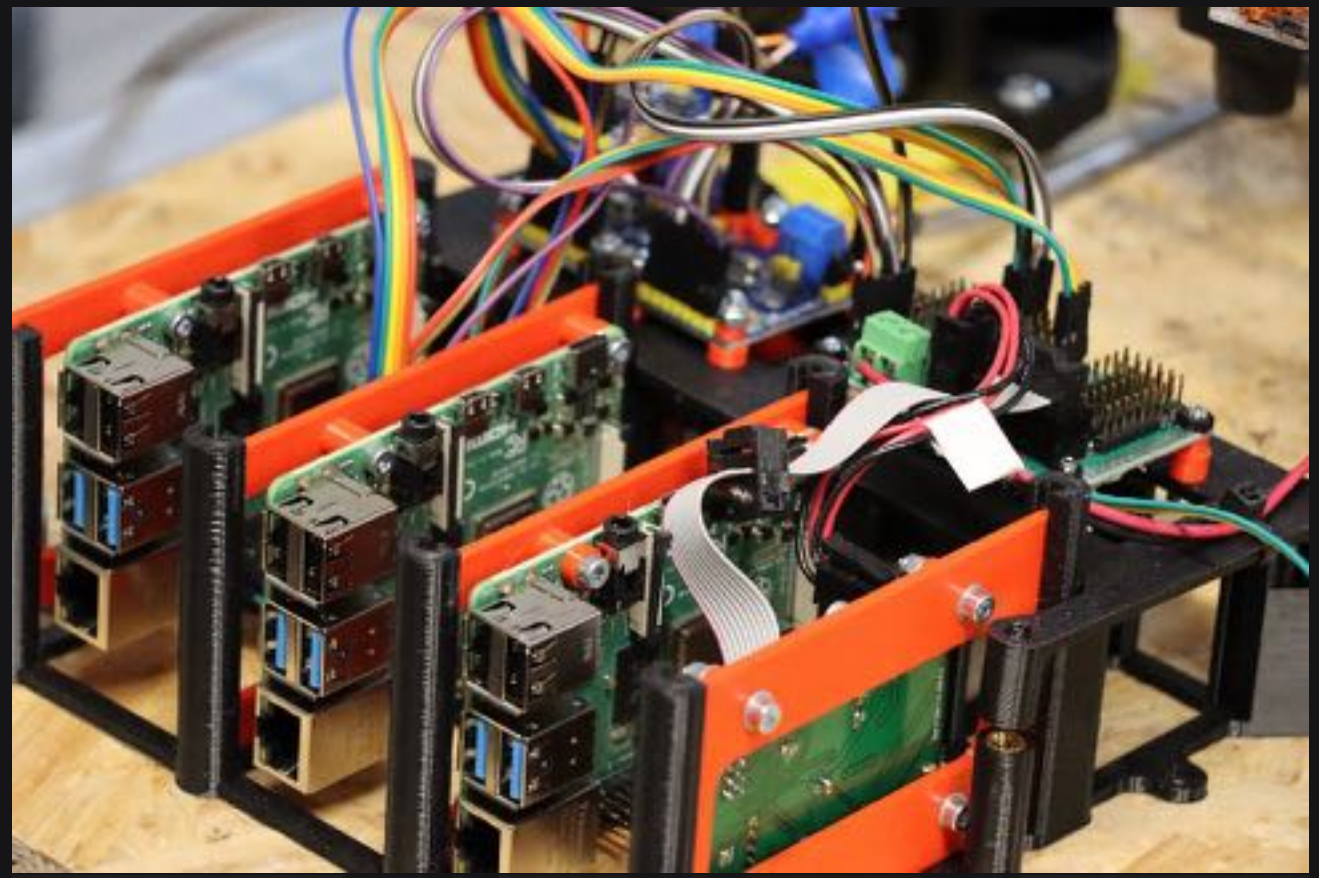
VMS

The car's "brains"

RPI4 + Nerves

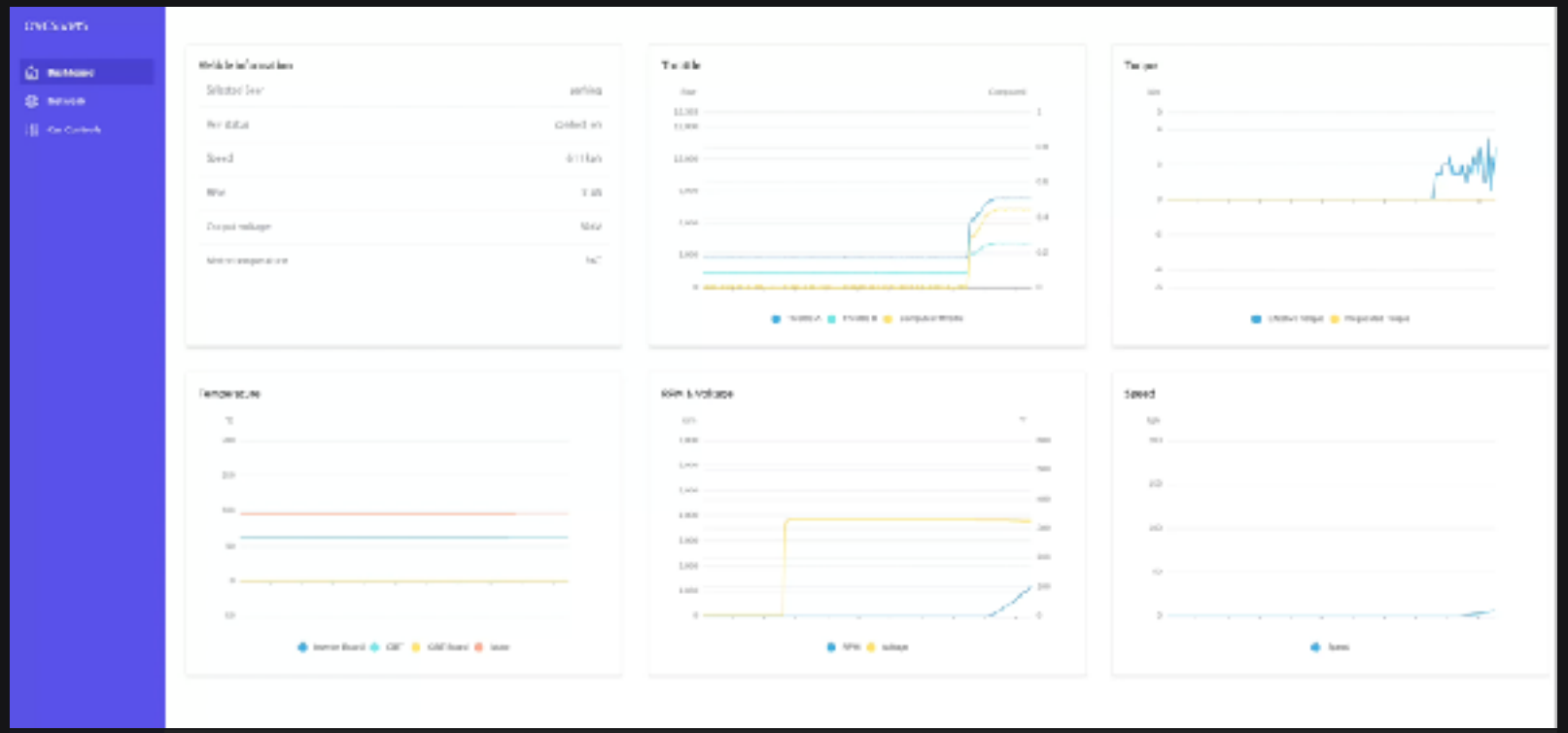
Connected to all networks

Translates and routes messages



https://github.com/open-vehicle-control-system/ovcs/tree/main/vms

VMS - monitoring



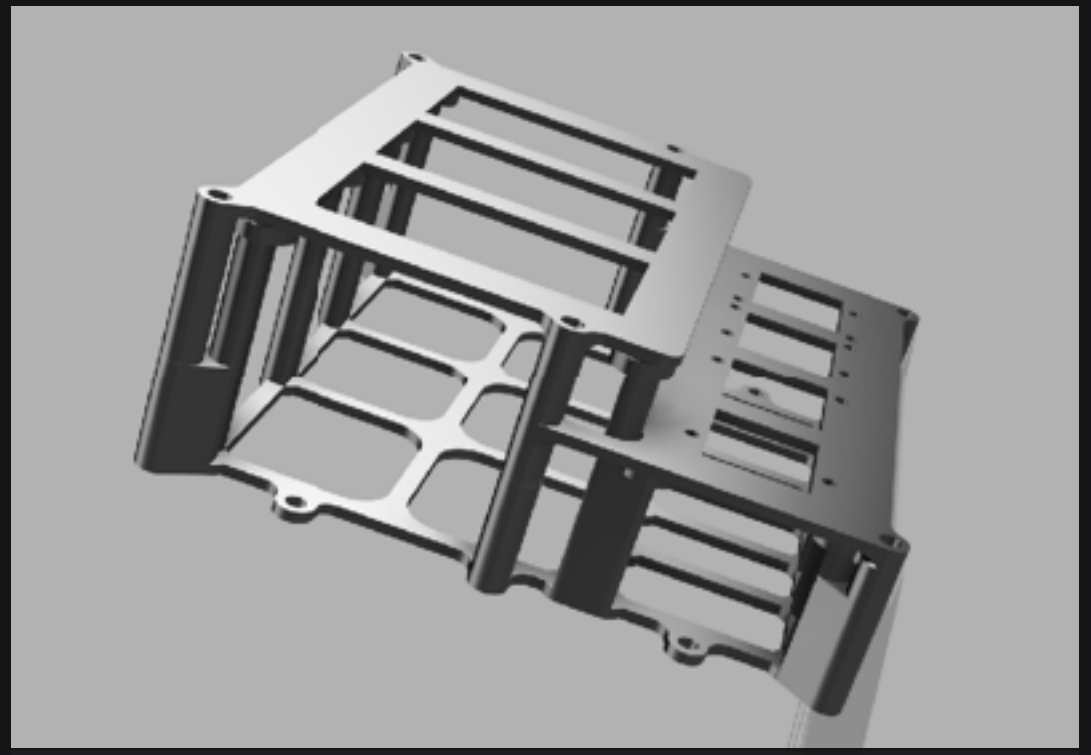
- Phoenix + Vue.js app for real time monitoring
- Enable throttle pedal calibration
- Alerts when controllers are not present or defective

VMS - challenges

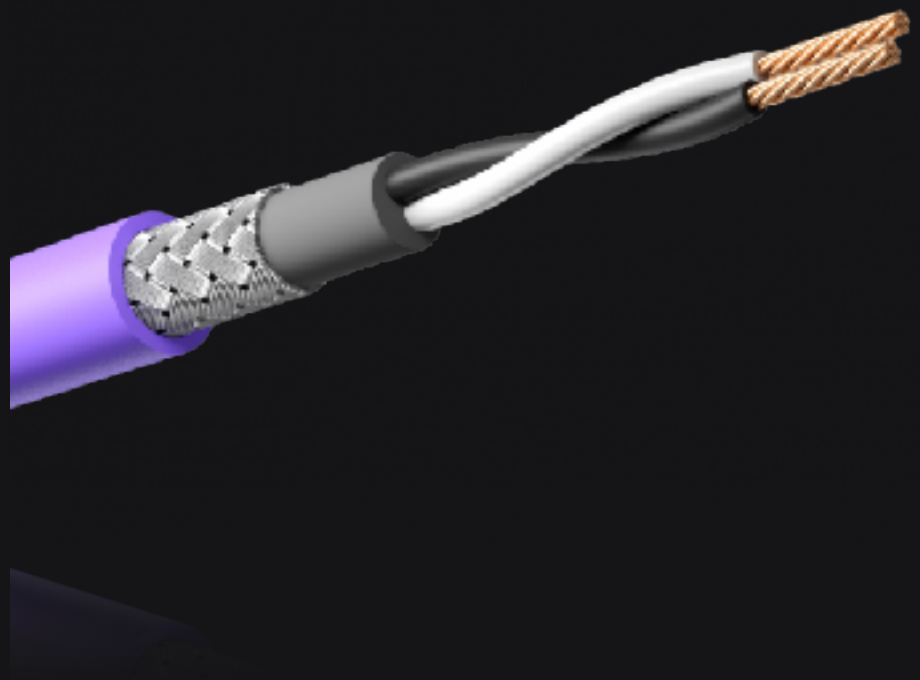
Testing on hardware is hard...

3D printing can fail...

Building an Elixir CAN library from scratch



Communication bus



CAN bus (Controller Area Network) allows to communicate efficiently amongst all vehicle components

Standard in automotive, ships, planes, machinery and more

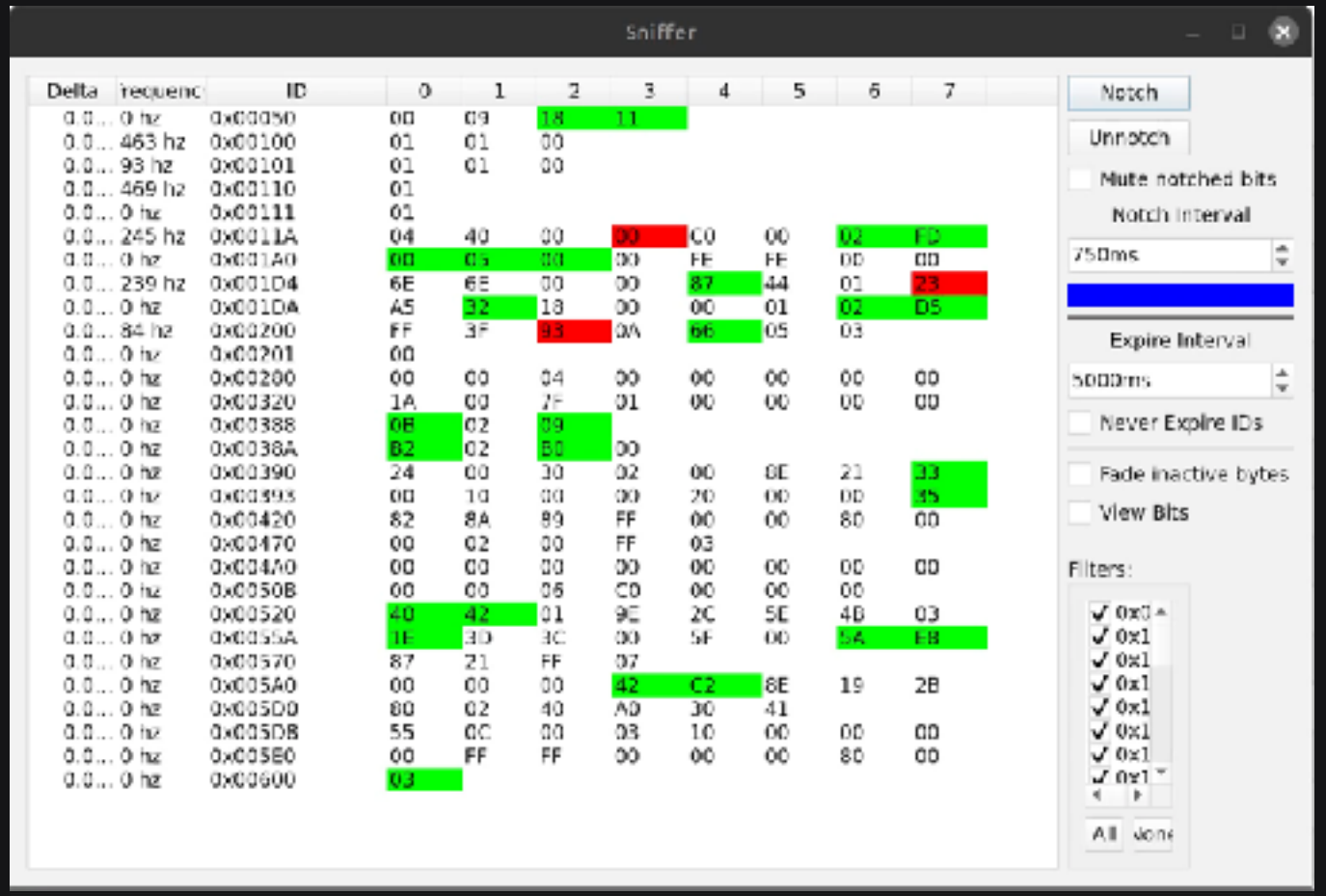
While CAN is standard, the message payload are not, and almost never documented by manufacturers

Communication bus

Sending and receiving data over CAN is not the same as using a REST API

Data is just a set of bits

Frames are broadcasted to all module and continuously



Cantastic

Based on a declarative YAML file, Cantastic allows you to emit and receive frames in a convenient way.

```yaml
- name: abs_status
  id: 0x5A0
  frequency: 20
  signals:
    - name: speed
      unit: km/h
      value_start: 8
      value_length: 16
      scale: "0.0075"
- name: gear_status
  id: 0x600
  frequency: 20
  signals:
    - name: selected_gear
      kind: enum
      value_start: 0
      value_length: 8
      mapping:
        0x00: drive
        0x01: neutral
        0x02: reverse
        0x03: parking
```

```elixir
@impl true
def handle_info({:handle_frame,  %Frame{signals: signals}}, state) do
    %{"speed" => %Signal{value: speed}} = signals
    {:noreply, %{state | speed: speed}}
end
```

```elixir
def select_gear(gear) do
  :ok = Cantastic.Emitter.update(:ovcs, "gear_status", fn (data) ->
    %{data | "gear" => gear}
  end)
end
```

https://github.com/open-vehicle-control-system/cantastic

Controllers

"Dumb" logic, rely on VMS decisions for safety

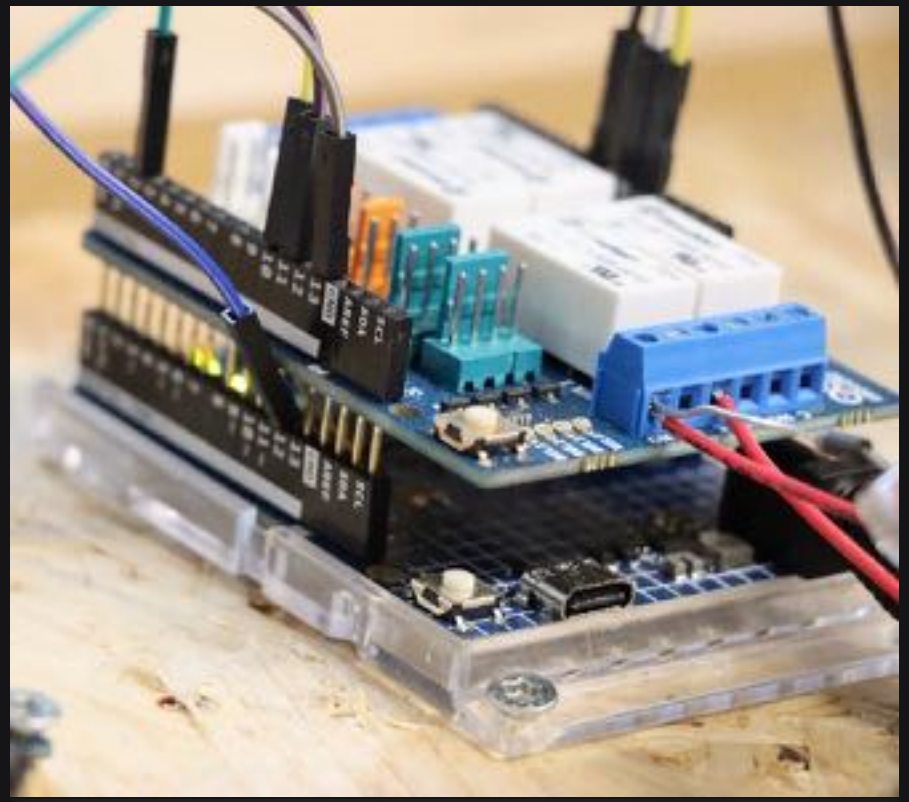
Connects to CAN with SPI module (mcp2515)

Used to enhance existing capabilities with CAN (analog pedal, fuel gauge, ...)

Gear selector

https://github.com/open-vehicle-control-system/ovcs/tree/main/controllers

Controllers - challenges



- Large relays interfered with Arduino and crashed them
- Using ethernet cables instead of real CAN cables was unreliable
- Bad USB cables (not enough power when using mcp2515)
- Relay hat conflicting with SPI pin
- Next: look for automotive grade alternative

Infotainment

Connected to OVCS and Car drive CAN

Running WPEWebkit in Kiosk mode

3D printed enclosure that fits in the car's radio bay

https://github.com/open-vehicle-control-system/ovcs/tree/main/infotainment

Infotainment

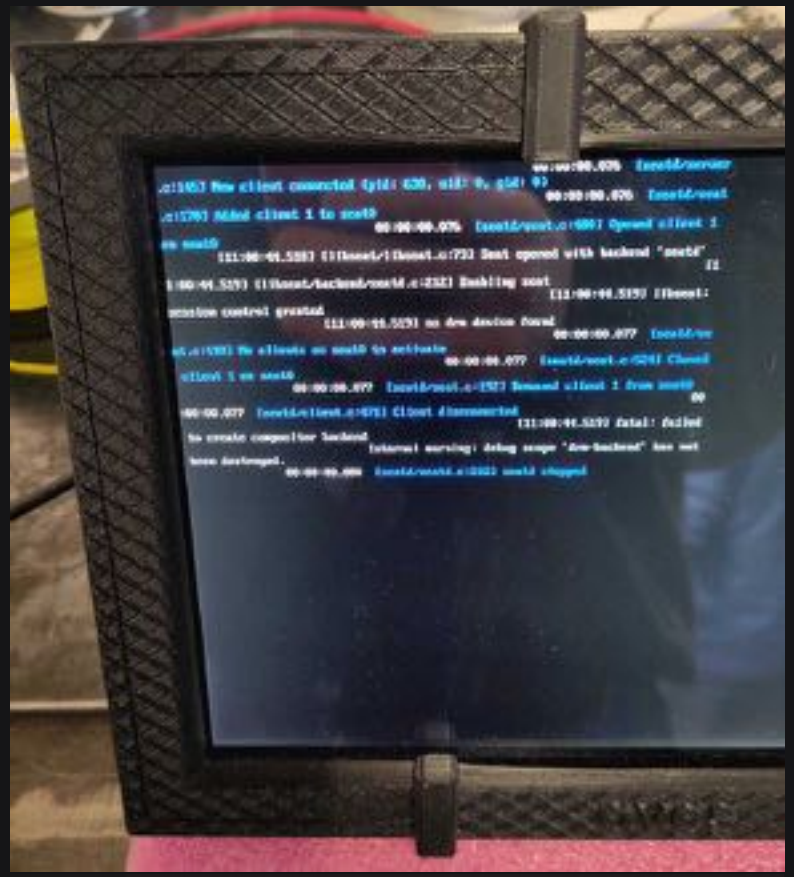
Infotainment - challenges

Making it work in kiosk mode

- Lots of buildroot tinkering
- Slow WPEWepkit compile time

Still some intermittent issues with the Direct Rendering Manager and Wayland on RPI4 at boot time

Next: try standard minimal Debian with live-build



ElixirConf EU

DESIGN CHOICES

BUILDING BLOCKS

EVERYTHING TOGETHER

FURTHER WORK

OVCS1 Frankencar

2007 Polo Bluemotion

+

EM57 2013 Nissan Leaf Motor

Mechanical work

Welded new motor support

CNC cut and welded aluminum connection plates

Fabricated custom junction between motor and gearbox

330V DC power supply

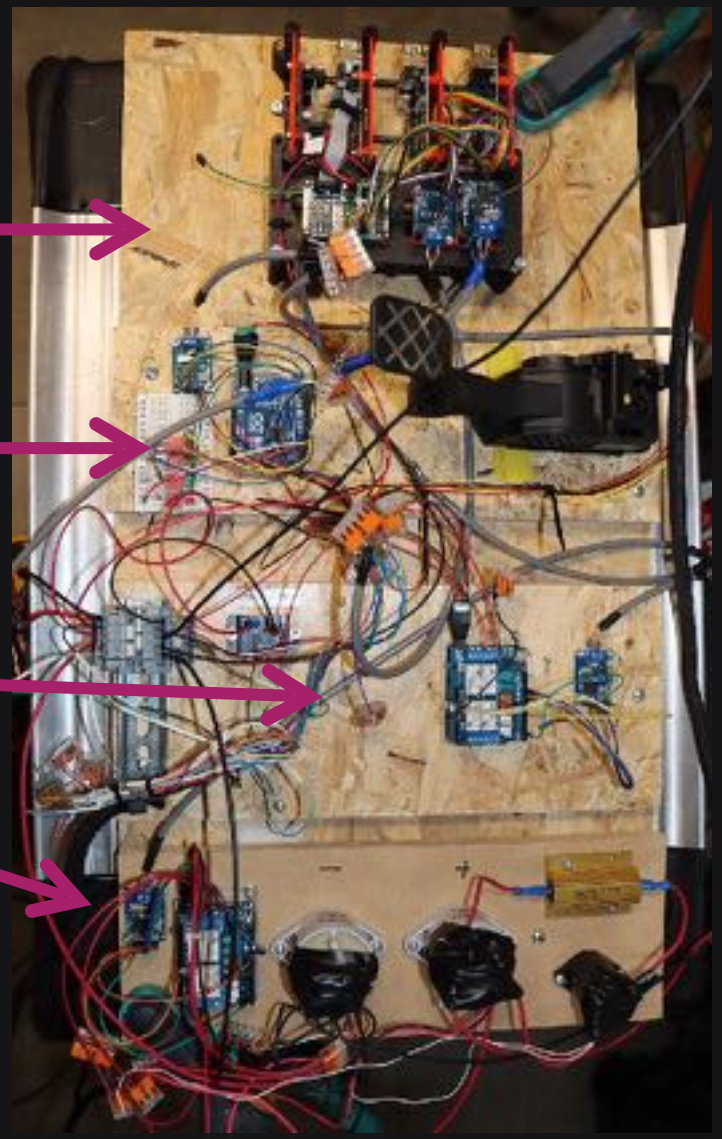
Prototype board

Vehicle Management System

Car controls controller

Contactors controller

And lots of wires...



The wheels on the...

DESIGN CHOICES

BUILDING BLOCKS

EVERYTHING TOGETHER

FURTHER WORK

Next steps

Connect batteries and Battery Management System

Test the car in a controlled environment

Add redundancy to VMS

Use automotive grade components?

Self-driving in Elixir?

ElixirConf EU

Any questions?

Special thanks to people who helped us:

Joël Vigneron

Jean-Luc Delvaux