

# FROM A 2007 VW POLO TO AN AUTONOMOUS EV IN 1 YEAR

Thibault Poncelet

Marc Lainez

Loïc Vigneron

# L'équipe



Marc  
LINF 2007



Loïc  
SINF 2012



Thibault  
SINF 2012

2013:  Spin42

↳ 2016-2023: Ibanity (Sold to Isabel Group in 2017)

↳ 2024: Taking a sabbatical break, playing with cars...

# Why?



The transport industry is rather siloed and closed

The overall vehicle lifespan is rather limited compared to what it could be

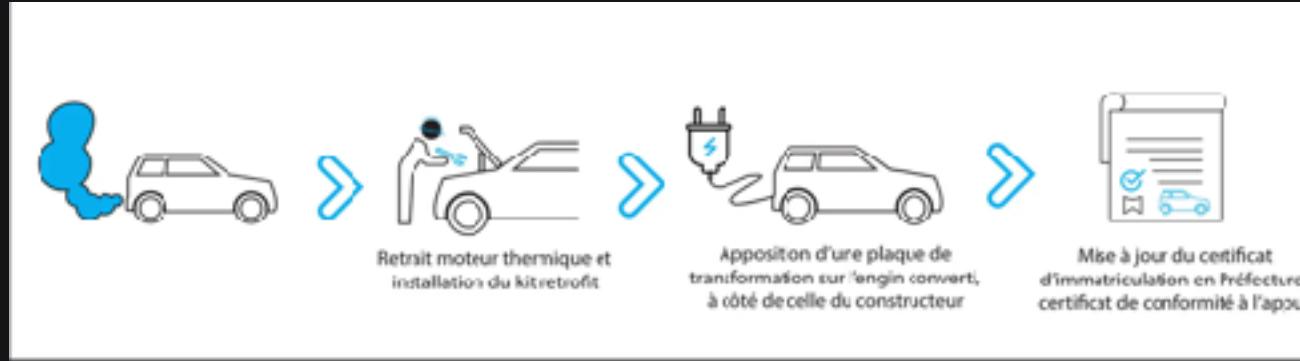
There is no “aftermarket software” for cars

Parts from different brands do not work together, too much vendor lock-in

# What is “upgrading” a vehicle?

- Bringing it on par with environmental requirements
  - “Engine swap”
  - EV Retrofit
- Adding the features we expect in today’s cars
  - Infotainment system
  - Assisted driving/autonomous driving

# EV retrofit according to Carbone4



“...this analysis confirms the relevance of EV retrofitting for all vehicle categories.”

“The economic ROI of EV retrofitting still raises questions, especially for light vehicles.”

# The “smartphones on wheels” challenges

- Until when will your manufacturer make it's software infrastructure up?
  - WM Motors bankrupt in China<sup>[1]</sup>
  - Fisker bankrupt in the USA<sup>[2]</sup>
- How much do you trust your car's software security?
  - Tesla hacked twice during the Pwn2Own conference<sup>[3]</sup>
- Can you really trust charger networks?
  - Phoenix EV contact chargers hacked at 44CON in Londres<sup>[4]</sup>

1 <https://techcrunch.com/2023/10/10/wm-motors-bankruptcy-highlights-challenges-faced-by-ev-startups-in-china/>

2 <https://www.businessinsider.com/fisker-owners-worry-about-vehicles-working-bankruptcy-2024-4>

3 <https://www.forbes.com/sites/daveywinder/2024/01/27/tesla-hacked-as-electric-cars-targeted-in-1-million-hacking-spree/>

4 <https://hackread.com/zero-day-flaws-ev-chargers-to-shutdowns-data-theft/>

What if we converted this old  
2007 Polo into an EV ?

In short...



2007 Polo Bluemotion

+



2013 Nissan Leaf EM57 motor

# 1. Getting the polo ready

# Get the diesel engine out

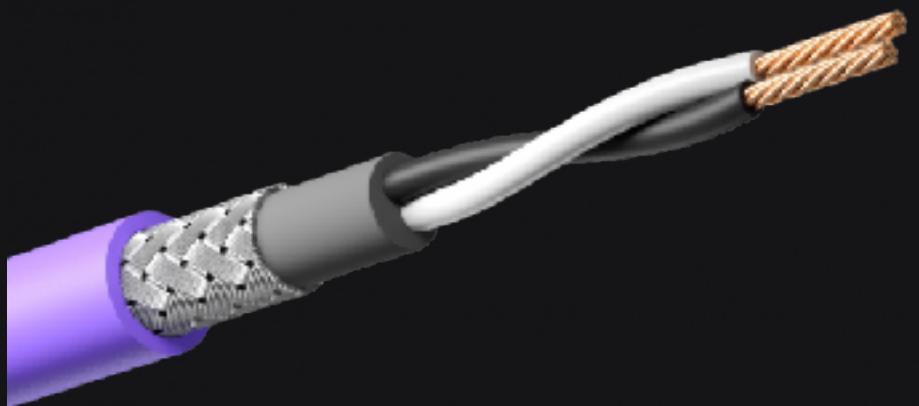


# Renovate both drivetrains



## 2. Understand the car's language

# CAN communication bus



CAN bus (Controller Area Network) is where all car components talk together

Standard protocol in automotive, aeronautics, industrial machinery, ...

Although CAN is standard, the messages you transfer through it are not

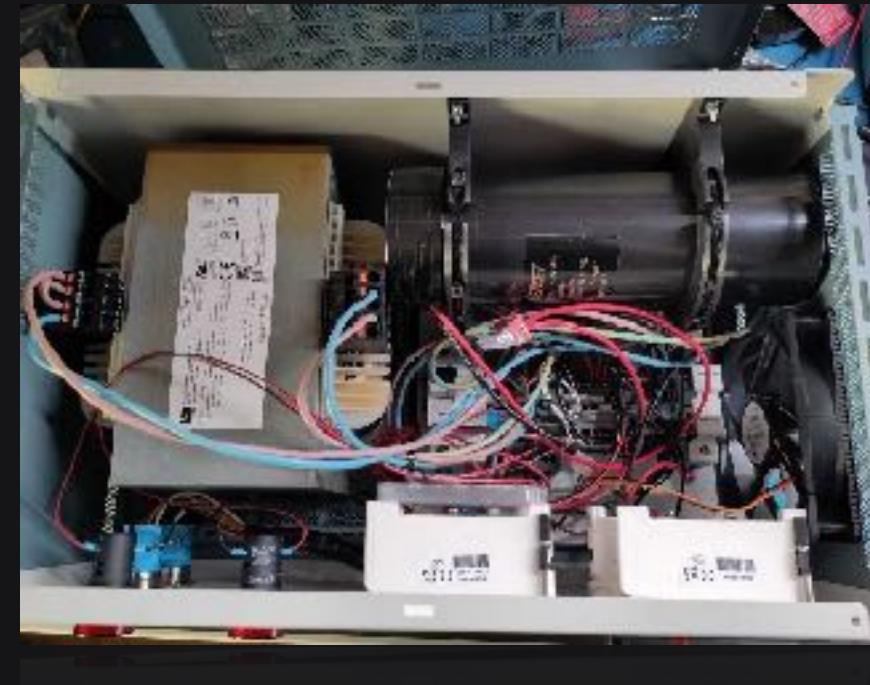
# CAN communication bus

Data exchanged on CAN is represented as a series of bytes

A “frame” with a specific ID  
is published periodically on  
the CAN

3. Get the leaf motor to spin

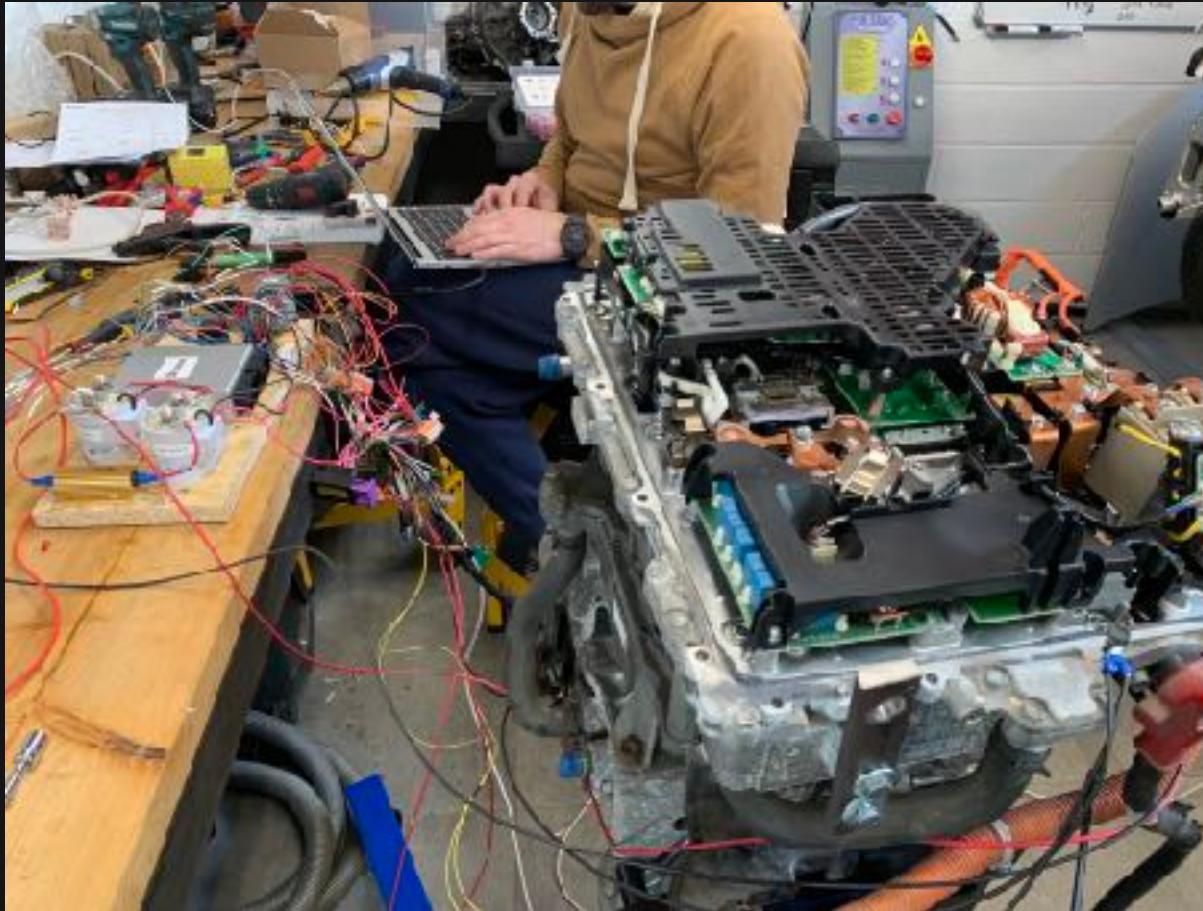
# Powering up the engine



Without batteries, we needed a way to supply 330V to the motor

We built a custom AC/DC 330V 8A power supply for our first motor tests

# Reverse engineering the Leaf motor



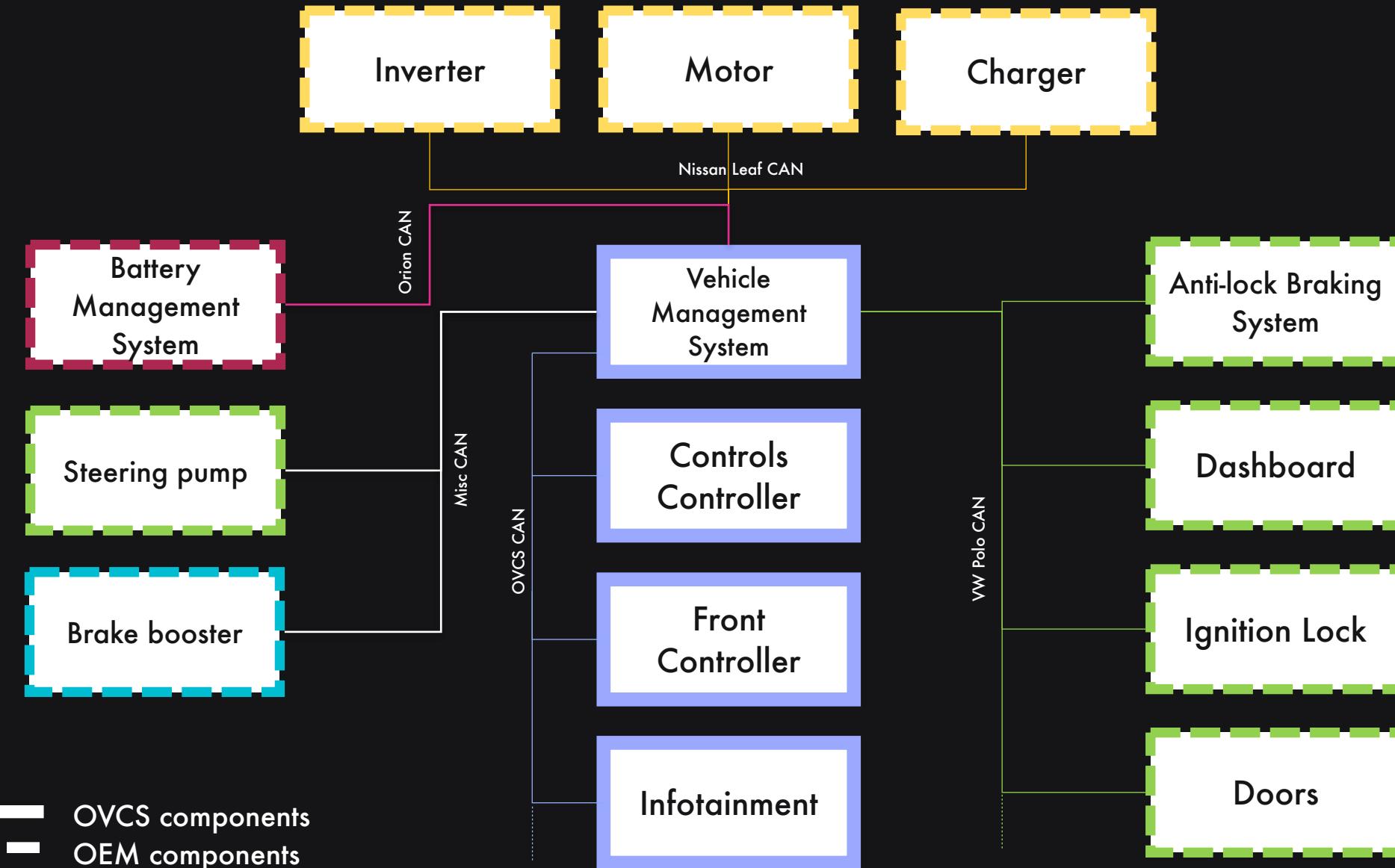
We needed to find the right CAN messages to power up the motor

We used DBC files we could find online to figure them out

# Until...



4. Use the Polo gas pedal to control the motor



# Connecting the pedal to the CAN

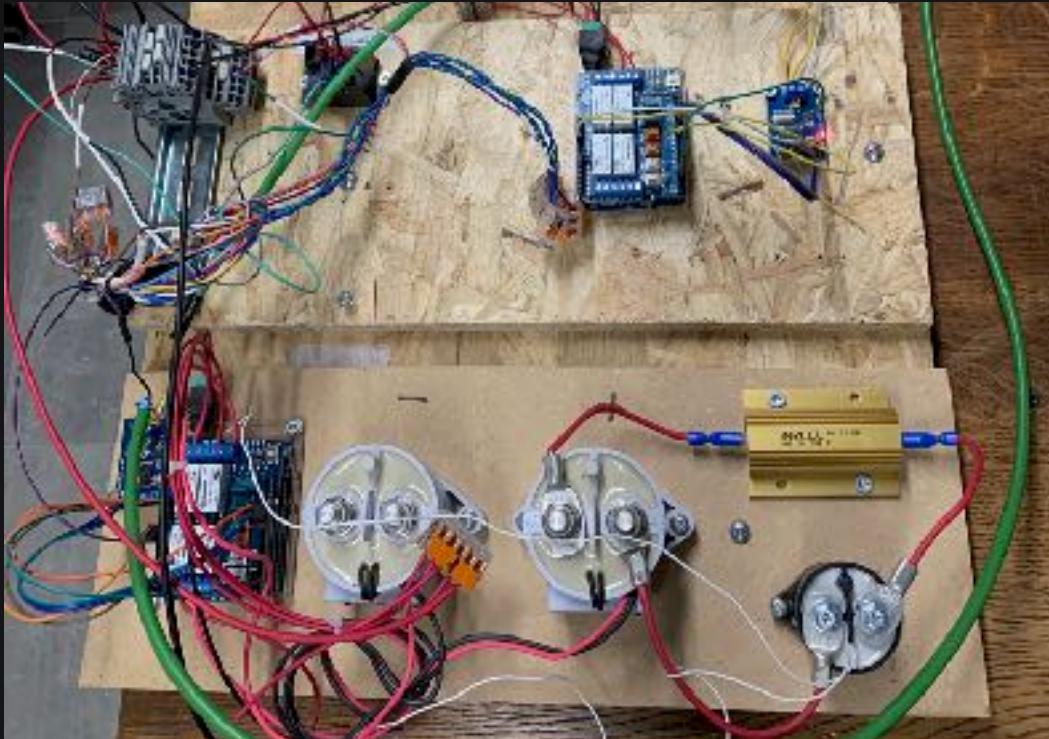
The pedal is a simple potentiometer (2 actually...)

You can see two signals, one is used to give the pedal position and the other one is a control value

We connected it to an arduino with a CAN module over SPI



# Deal with motor contactors



Several relays need to be activated in a specific order

Adding relays to arduino was quite straightforward

The arduinos are connected to the CAN network with CAN SPI modules

# The VMS, the car's new “brains”

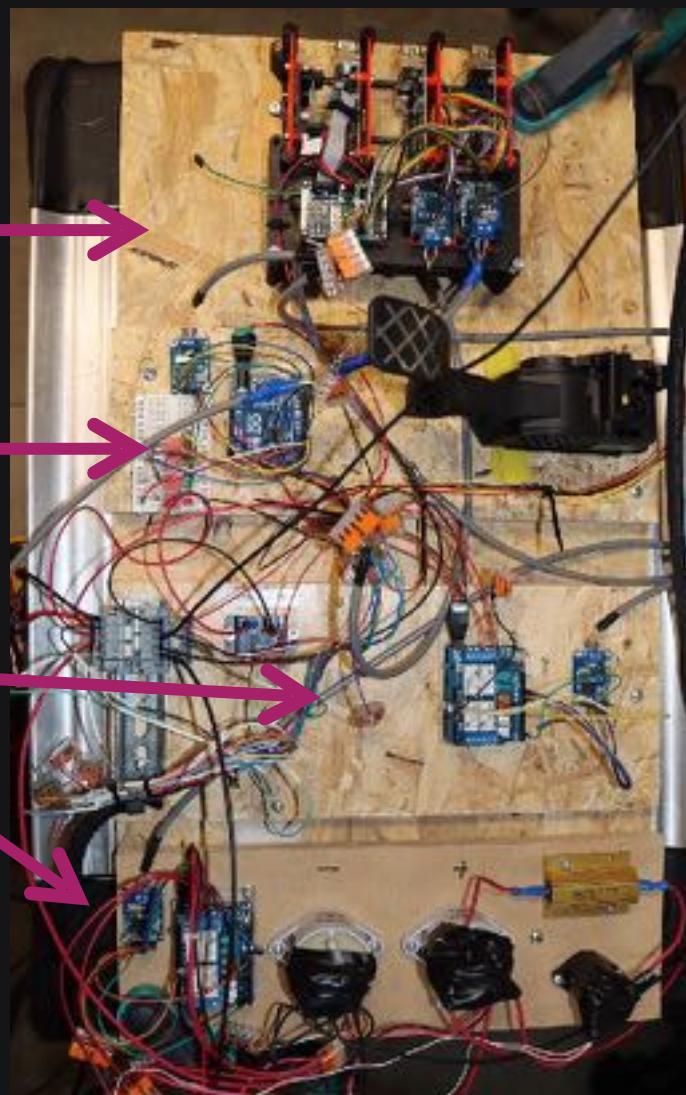


# The first “end-to-end” prototype

Vehicle Management System

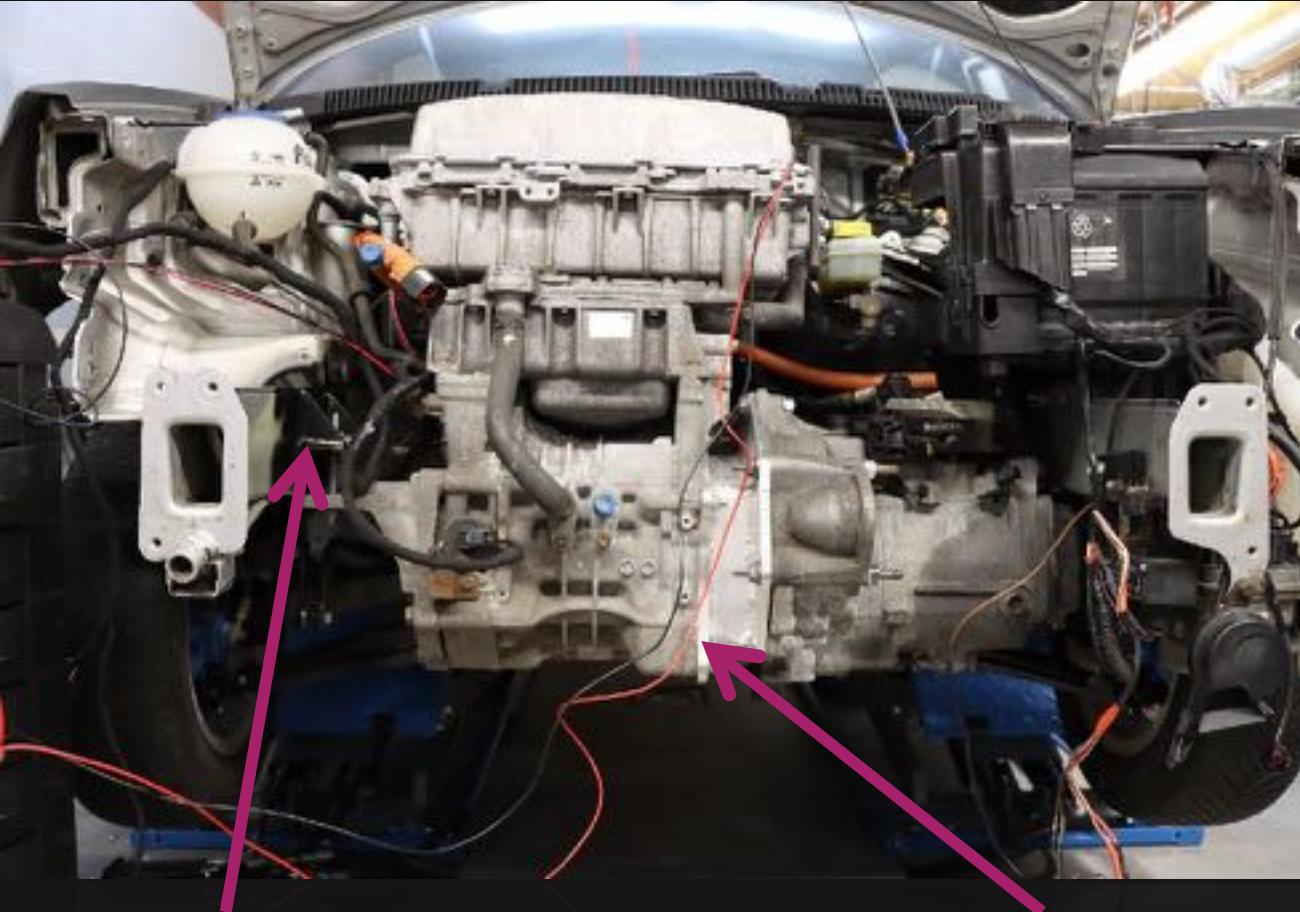
Car controls controller

Contactors controller



5. Put the Leaf motor into the  
Polo

# Tight fitting...



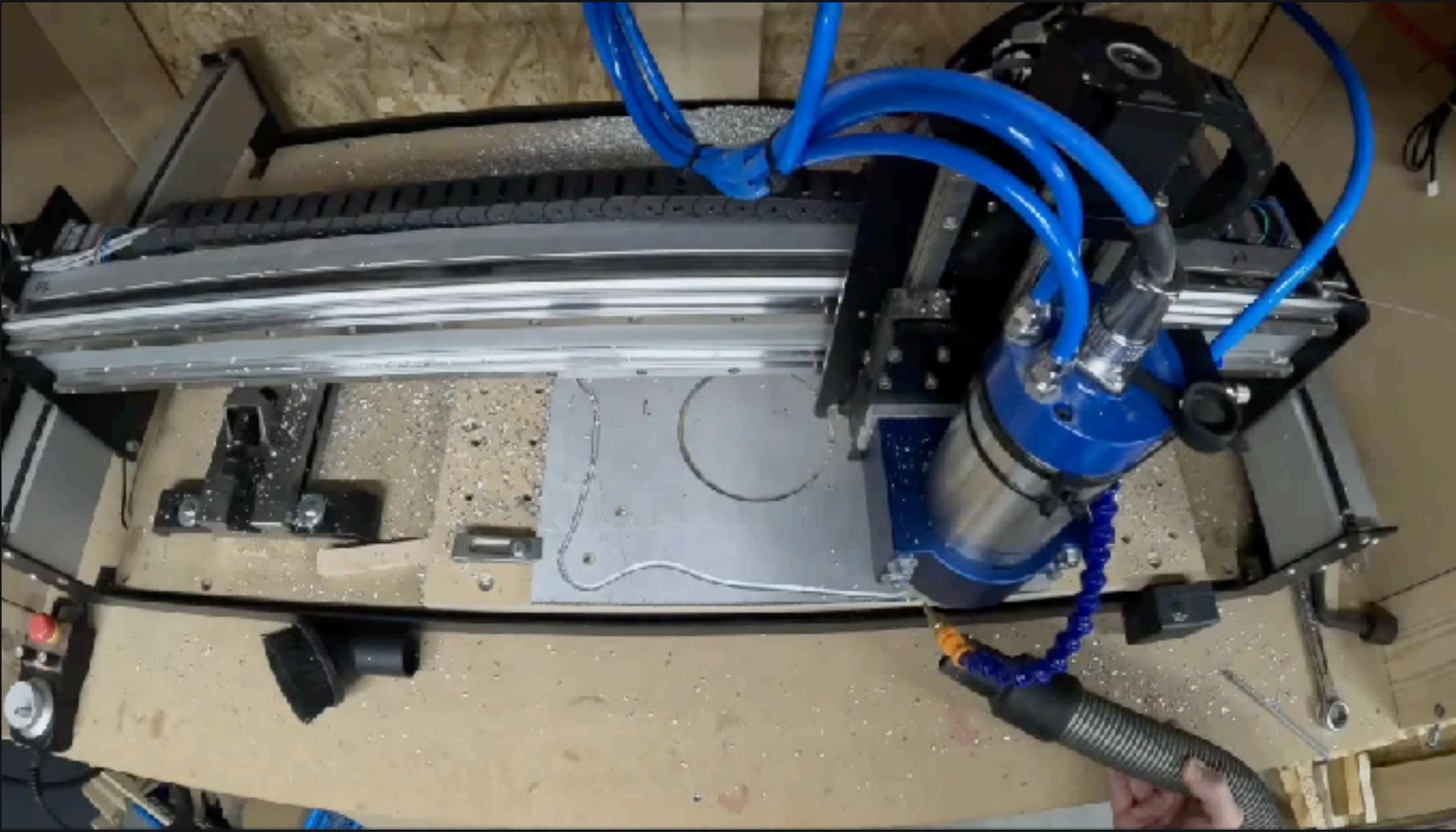
New motor support welded

Connection plates CNC'd and  
welded



Custom junction piece to connect the  
motor to the gearbox

# Parts fabrication



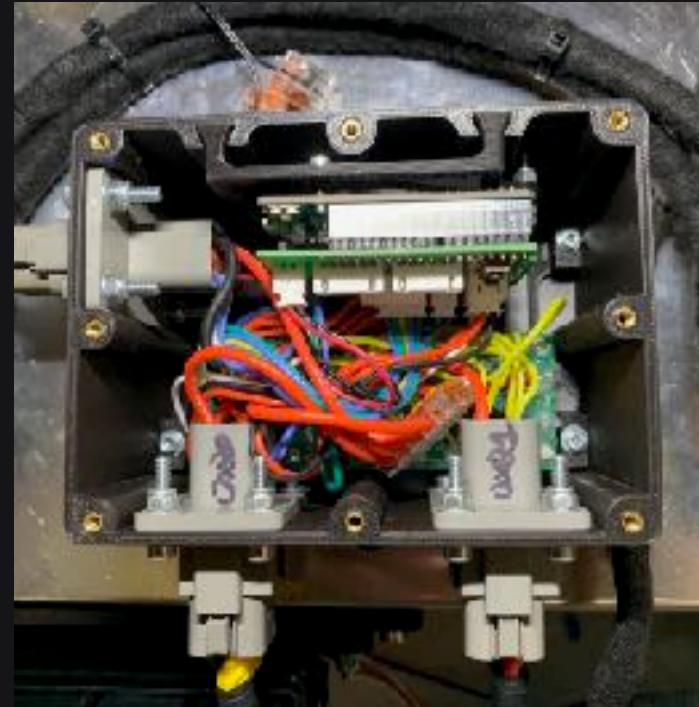
# First “in-car” test with the “plank”



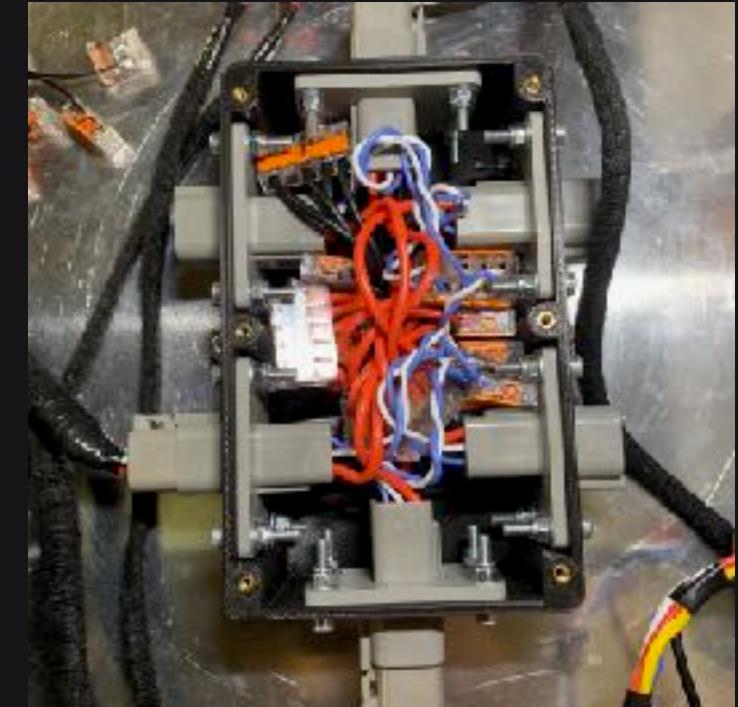
# From “plank” to safer components



VMS (x1)  
RPI4  
Custom SPI hat  
5xMCP2517FD



Generic controller (x3)  
Arduino R4 Minima  
Custom SPI hat  
MCP2517FD



OVCS Canhub (x3)  
(Just cables 😊)

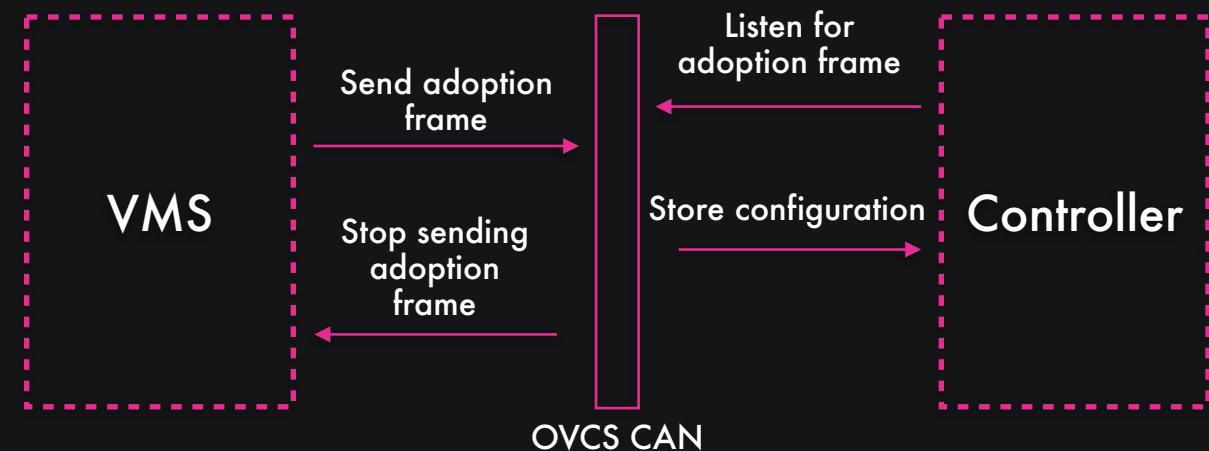
# Generic controller

OVCS Function	Physical Pin	OVCS Pin
UART Receive	D0	
UART Transmit	D1	
Adopt button	D2	
SPI CAN Int	D3	
Digital	D4	0
Software PWM	D5	0
Software PWM	D6	1
Digital	D7	1
Digital	D8	2
Software PWM	D9	2
SPI CAN CS	D10	
SPI CAN COPI	D11	
SPI CAN CIPO	D12	
SPI CAN SCK	D13	
DAC	A0	0
Analog In	A1	0
Analog In	A2	1
Analog In	A3	2
I2C SDA - MOSFET	A4	
I2C SCL - MOSFET	A5	
Digital	MOSFET0-0 -> 7	3 -> 10
Digital	MOSFET1-0 -> 7	11 -> 18
Hardware PWM	PiC32 over UART	0 -> 3

All controllers run the same code

Their function is determined by the VMS during adoption

A button on the controller makes it go into adoption mode



# The infotainment

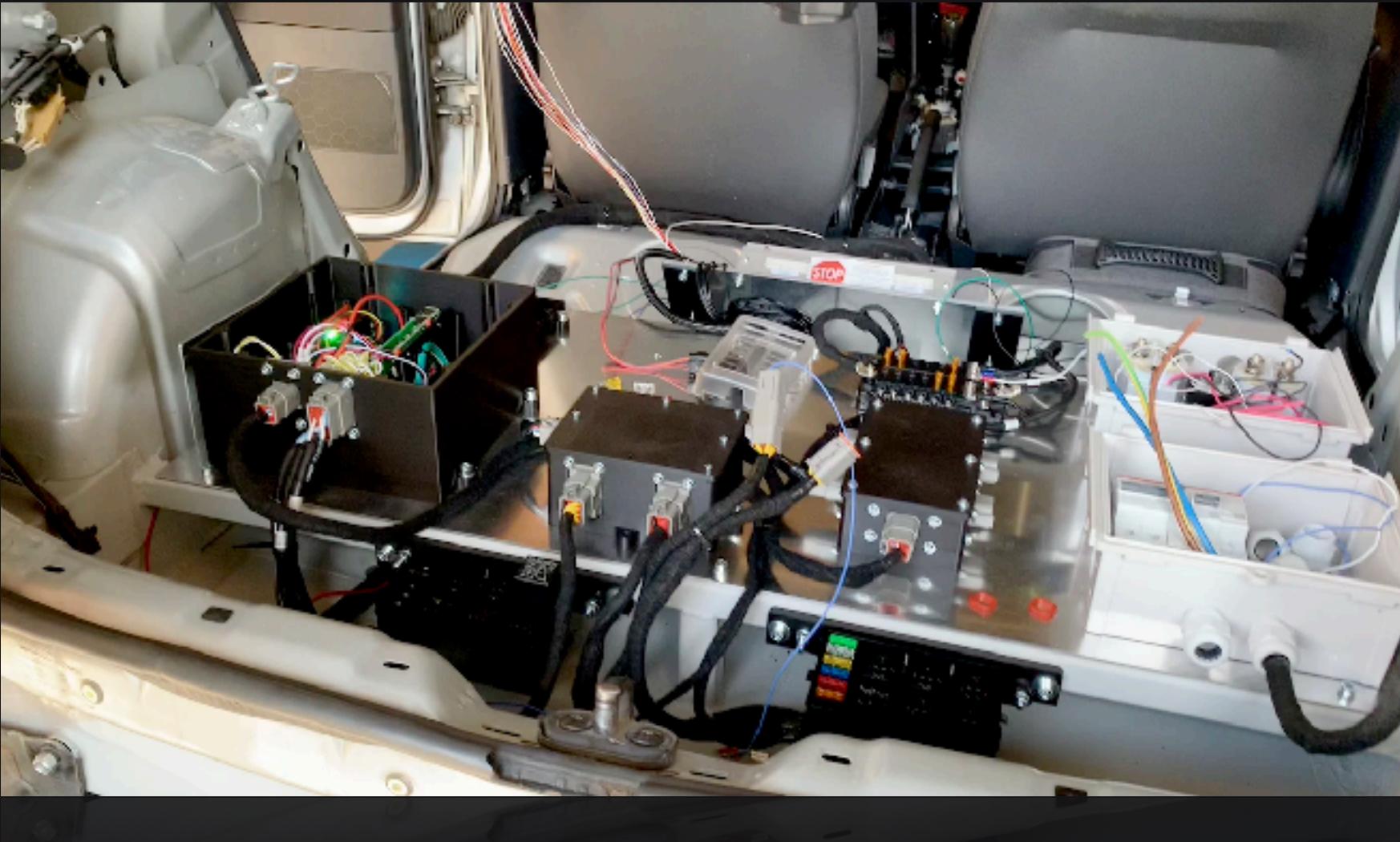
Gives information and diagnostics about car features

Replaces the gear selector (PRND)

Serves as a basis for a future multimedia system and control interface



# Placing the components in the car



# Used technologies



Vue.js  
Frontend web  
VMS



Nerves  
Firmware builder based on buildroot  
All non-arduino components



Phoenix  
Backend API  
VMS + Infotainment



Flutter  
Frontend embedded  
Infotainment

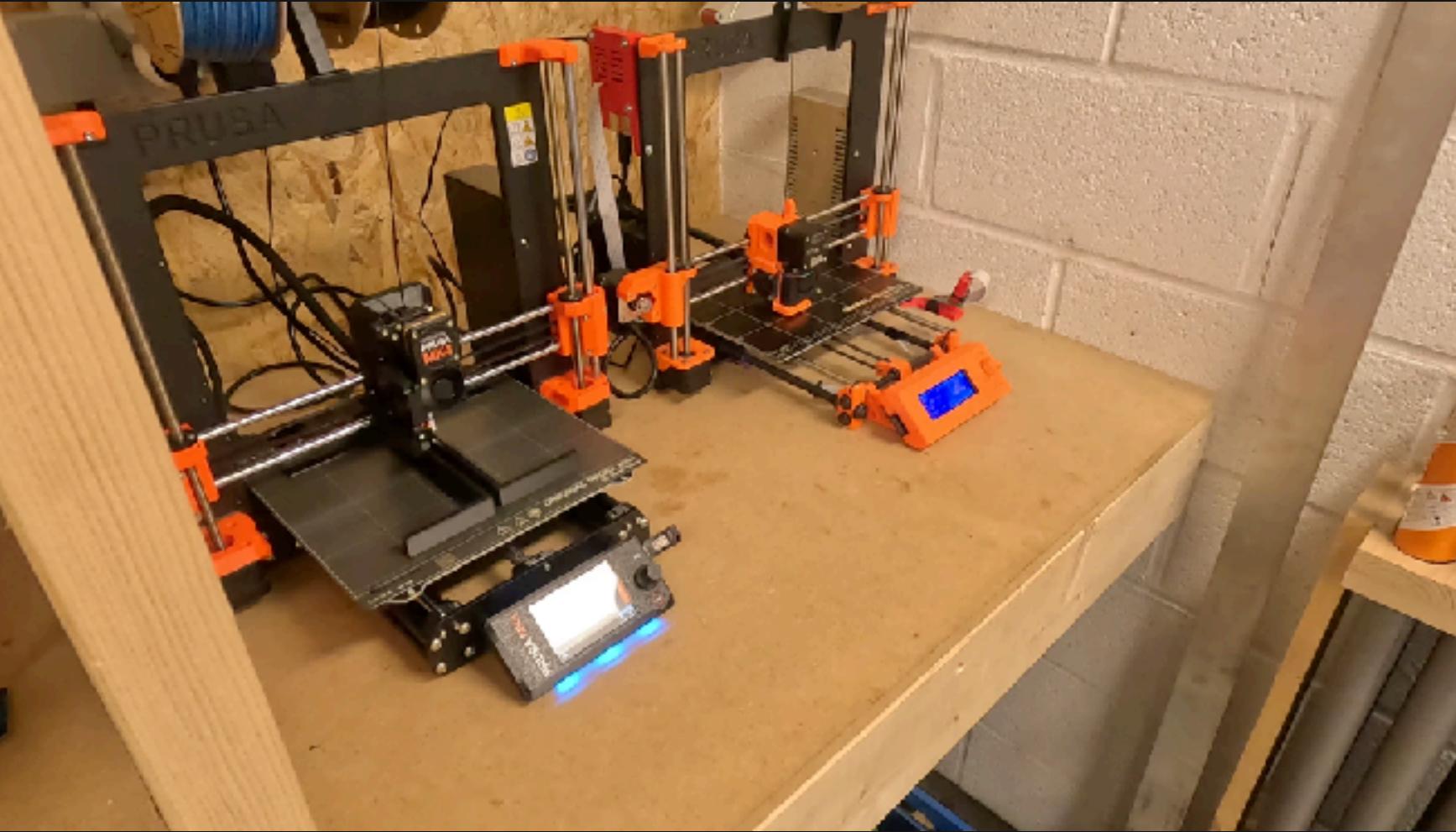


Elixir  
All non-arduino  
components



C++  
Microcontrollers  
All arduinos

# 3D printing



# 6. Adapting the car

# Changing the servo-brakes

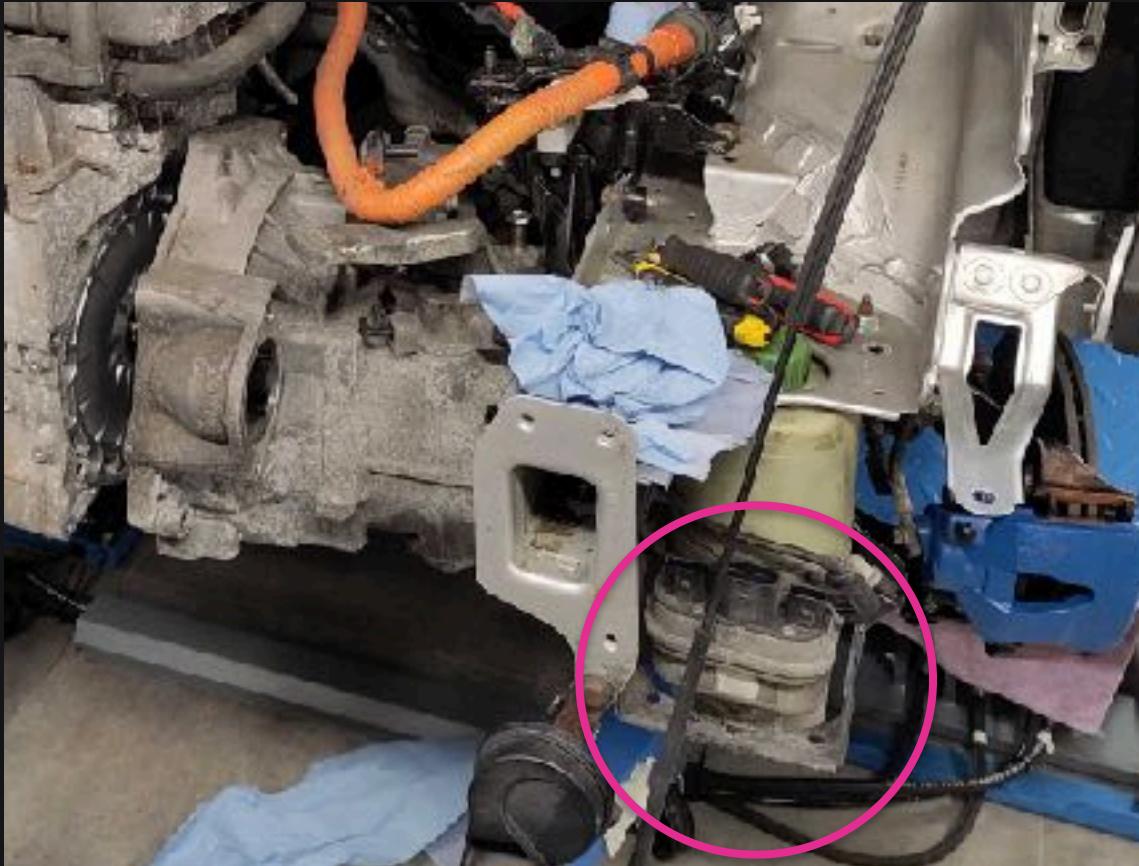
The polo had a servo-brake that used the depression from the thermal engine to provide brake assistance

Tesla's "brake boosters" are popular in old car renovations

We simply installed a gen2 Tesla iBooster to solve this issue



# Controlling the steering hydraulic pump



The pump starts when the thermal engine is started

It knows it's started when the RPMs on the CAN are the "idle RPM" of the thermal engine...

We are controlling it separately through the VMS by faking the engine presence and RPM

# 7. Testing the car

# First test drive



8. What about the battery?

# The battery containers



Three custom made aluminum containers were fabricated

We put two under the trunk and one where the fuel tank used to be

We are using an Orion BMS

We have several nv200 modules for a total of 80kwh capacity

# Wiring the battery



What ifs we transformed our Polo  
EV into an autonomous vehicle?

# Required components

Plan trajectory

Perceive environment and localize

Control the vehicle

ODD (Operational Design Domain)

Decision algorithms, route  
planners, ...

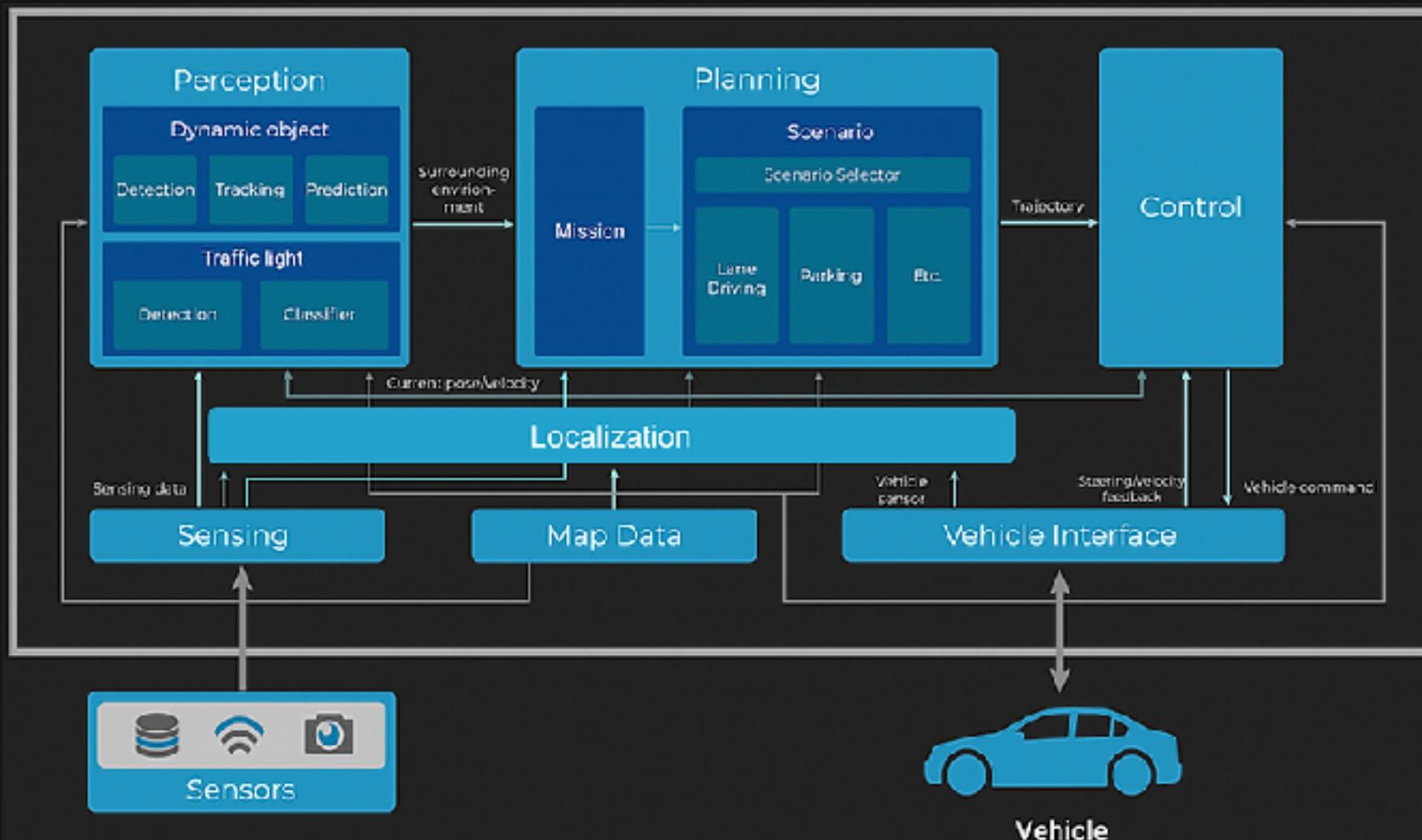
Sensors, GNSS, cameras,  
telemetry...

Acceleration, braking,  
steering

# ODD examples



# Autoware



# Autonomous driving levels

Level 0: no autonomy

Level 1: basic autonomy (adaptive cruise control, lane assist, ...)

Level 2: a human needs to be ready to take over at all times (Tesla autopilot)

Level 3: a human needs to take control in extreme situations (mercedes drive pilot)

Level 4: no human intervention necessary in certain areas (geofencing, robot taxis)

Level 5: no human intervention needed, in any conditions and places

# Controlling the Polo

Acceleration



Braking



Steering



# Braking



Tesla's brake boosters can be controlled via CAN

The CAN messages allow to control the rate of fluid going through the booster

From gen1 DBC files and some CAN traces we found, we were able to reverse the right CAN messages

Braking ✓



# Steering

The original steering column is not motorised

We tried reversing a 2019 Polo steering column (2Q1909144) with no success

We stripped the 2Q1 of it's ECU and motor and simply connected another servo and angle sensor



# Steering ✓

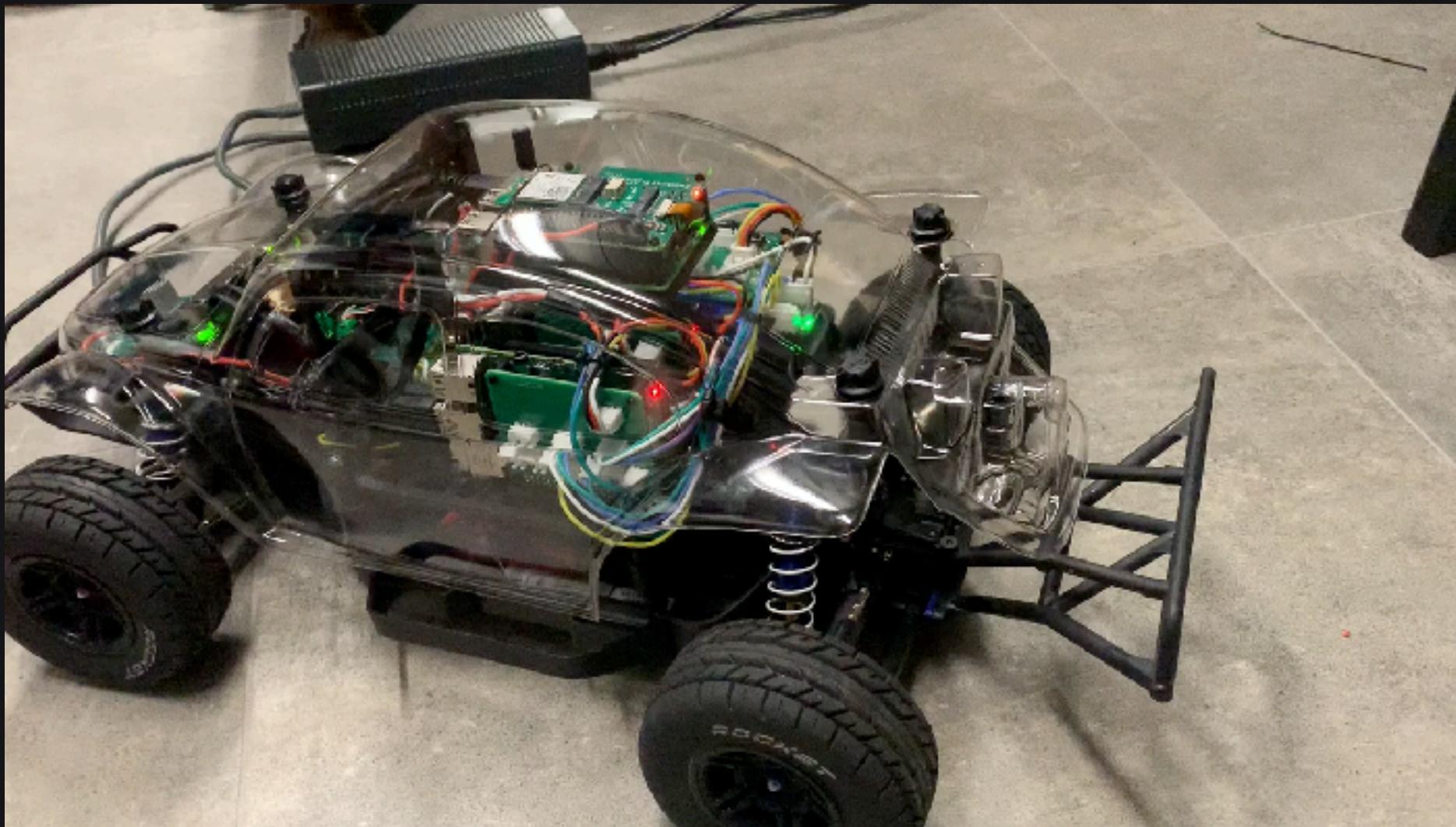


# All together...



How are we going to **test** our  
**autonomous driving** components?

# OVCS Mini



# A Mavlink bridge for OVCS

“Micro Air Vehicle Link”, mostly used  
in aerial drones

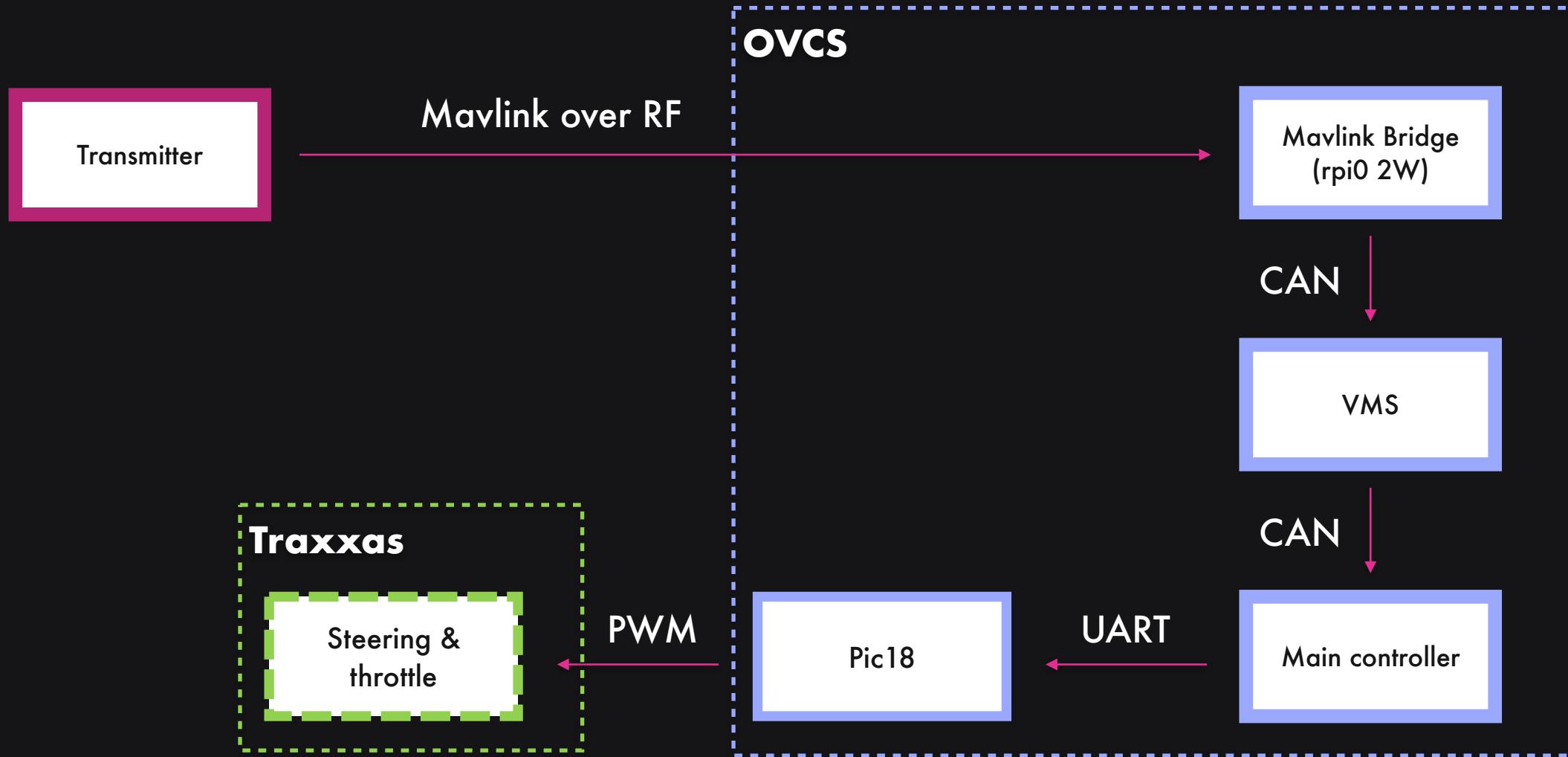
Also supports “rover” types of drones

Open protocol which can be  
extended with our own messages

Supported by several controllers,  
libraries and tools



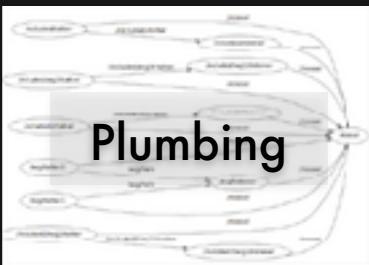
# Controlling OVCS mini through a Mavlink transmitter



# A ROS2 for OVCS



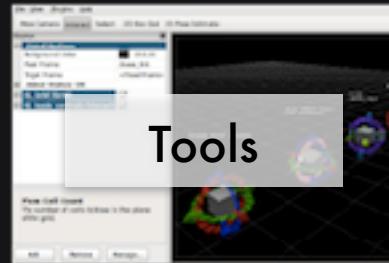
=



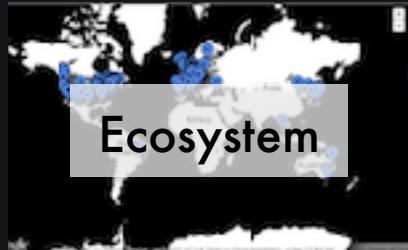
+



+



+



Robot Operating System

Standard in the robotic's world

Rclex, an Elixir client for Nerves

No need to run Ubuntu 🎉

# The OVCS remote (wip)

Multi protocol remote  
(Mavlink, ROS, ?)

Allows us to test new features  
that are not supported by off-  
the-shelf transmitters

And... it's just cool to build  
one 😎



# A homemade stereo camera (wip)



Using 2 cheap USB cameras

Leverages on Rclx to send raw and compressed images in ROS2 topics

TODO:

- Add the depth map
- Perform a proper camera calibration
- Add an IMU

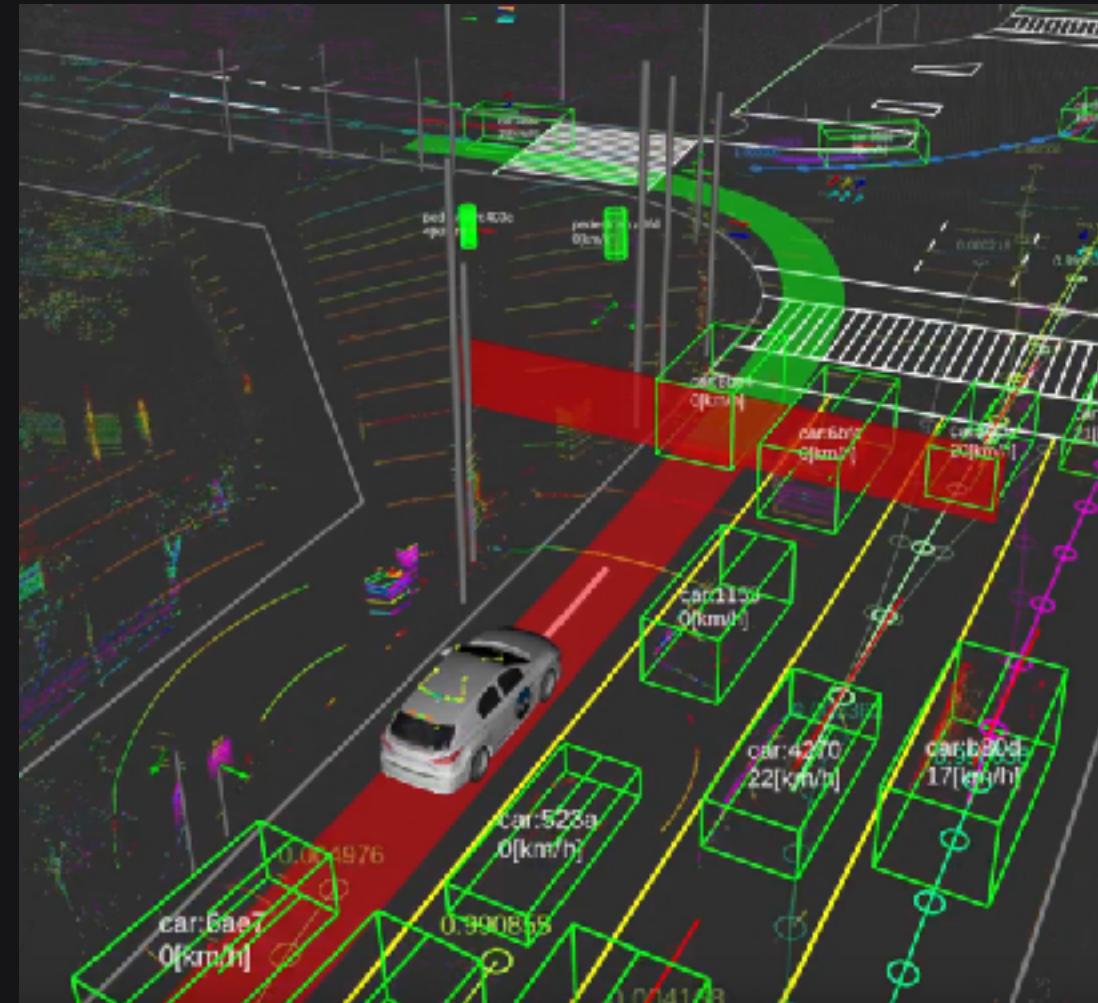
# Next steps to reach autonomous driving

Finish up our ROS2 bridge

Start tinkering with Autocar

Perform the first “manual” and  
“remote driving” tests in a  
controlled space with the Polo

Install the appropriate sensors  
and devices on the Polo’s roof



What will all this be used for?

# Some (maybe) future products & services

## Products

Remote control modules for existing vehicles 

Autonomous driving for existing vehicles 

"Aftermarket" OS/CAN gateway for end-of-life vehicles 

Renting the Polo for research purposes 

## Services

Vehicle EV/autonomous retrofit 

CAN/ISOBUS/... reverse engineering as a service 

Vehicle parts interoperability 

Industrial machinery upcycling/retrofitting 



That's it! 😊

Any ideas, suggestions, questions?

<https://github.com/open-vehicle-control-system>