



# Spin42

## **Reverse-engineering CAN and building ECUs using Elixir**

Thibault Poncelet

# Background

- After 7 years building an Open Banking aggregator in Elixir, We wanted to learn something new and test if Elixir could be a good fit for an automotive use case.
- In 2024, we have built an “Open Vehicle Control System”: An Elixir based devkit, allowing to upgrade any vehicle.
- “Upgrade” meaning:
  - Engine swap
  - EV retrofit
  - Infotainment system
  - Autonomous driving



# EV retrofit



2007 **VW Polo** Diesel

+



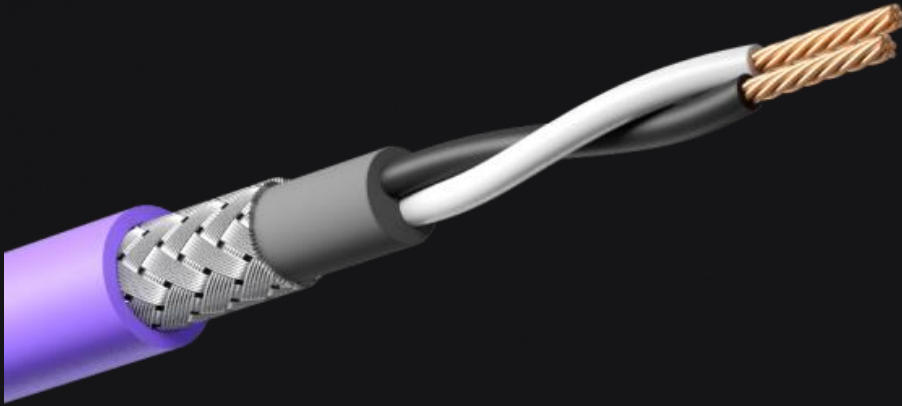
2013 **Nissan Leaf** EM57 motor

# Today's focus:

How to display the Nissan motor RPMs on the Polo's dashboard?



# CAN communication bus



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

Standard protocol in automotive, aeronautics, industry, ...

Built-in support in the Linux kernel:  
`libsocketcan`

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 bus

| Sniffer |           |         |    |    |    |    |    |    |    |    |  |
|---------|-----------|---------|----|----|----|----|----|----|----|----|--|
| Delta   | Frequency | ID      | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |  |
| 0.0...  | 0 hz      | 0x00050 | 00 | 09 | 18 | 11 |    |    |    |    |  |
| 0.0...  | 463 hz    | 0x00100 | 01 | 01 | 00 |    |    |    |    |    |  |
| 0.0...  | 93 hz     | 0x00101 | 01 | 01 | 00 |    |    |    |    |    |  |
| 0.0...  | 469 hz    | 0x00110 | 01 |    |    |    |    |    |    |    |  |
| 0.0...  | 0 hz      | 0x00111 | 01 |    |    |    |    |    |    |    |  |
| 0.0...  | 245 hz    | 0x0011A | 04 | 40 | 00 | 00 | C0 | 00 | 02 | FD |  |
| 0.0...  | 0 hz      | 0x001A0 | 00 | 05 | 00 | 00 | FE | FE | 00 | 00 |  |
| 0.0...  | 239 hz    | 0x001D4 | 6E | 6E | 00 | 00 | 87 | 44 | 01 | 23 |  |
| 0.0...  | 0 hz      | 0x001DA | A5 | 32 | 18 | 00 | 00 | 01 | 02 | D5 |  |
| 0.0...  | 84 hz     | 0x00200 | FF | 3F | 93 | 0A | 66 | 05 | 03 |    |  |
| 0.0...  | 0 hz      | 0x00201 | 00 |    |    |    |    |    |    |    |  |
| 0.0...  | 0 hz      | 0x00280 | 00 | 00 | 04 | 00 | 00 | 00 | 00 | 00 |  |
| 0.0...  | 0 hz      | 0x00320 | 1A | 00 | 7F | 01 | 00 | 00 | 00 | 00 |  |
| 0.0...  | 0 hz      | 0x00388 | 0B | 02 | 09 |    |    |    |    |    |  |
| 0.0...  | 0 hz      | 0x0038A | B2 | 02 | B0 | 00 |    |    |    |    |  |
| 0.0...  | 0 hz      | 0x00390 | 24 | 00 | 30 | 02 | 00 | 8E | 21 | 33 |  |
| 0.0...  | 0 hz      | 0x00393 | 00 | 10 | 00 | 00 | 20 | 00 | 00 | 35 |  |
| 0.0...  | 0 hz      | 0x00420 | 82 | 8A | 89 | FF | 00 | 00 | 80 | 00 |  |
| 0.0...  | 0 hz      | 0x00470 | 00 | 02 | 00 | FF | 03 |    |    |    |  |
| 0.0...  | 0 hz      | 0x004A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |  |
| 0.0...  | 0 hz      | 0x0050B | 00 | 00 | 06 | C0 | 00 | 00 | 00 |    |  |
| 0.0...  | 0 hz      | 0x00520 | 40 | 42 | 01 | 9E | 2C | 5E | 4B | 03 |  |
| 0.0...  | 0 hz      | 0x0055A | 1E | 3D | 3C | 00 | 5F | 00 | 5A | E8 |  |
| 0.0...  | 0 hz      | 0x00570 | 87 | 21 | FF | 07 |    |    |    |    |  |
| 0.0...  | 0 hz      | 0x005A0 | 00 | 00 | 00 | 42 | C2 | 8E | 19 | 2B |  |
| 0.0...  | 0 hz      | 0x005D0 | 80 | 02 | 40 | A0 | 30 | 41 |    |    |  |
| 0.0...  | 0 hz      | 0x005D8 | 55 | 0C | 00 | 03 | 10 | 00 | 00 | 00 |  |
| 0.0...  | 0 hz      | 0x005E0 | 00 | FF | FF | 00 | 00 | 00 | 80 | 00 |  |
| 0.0...  | 0 hz      | 0x00600 | 03 |    |    |    |    |    |    |    |  |

# CAN bus tools

- **USB2CAN:** An USB-to-CAN device
- **SavvyCAN:** a CAN bus reverse engineering and capture GUI.
- **socketcand:** a daemon that provides access to CAN interfaces on a machine via a network interface.



# Step 1

## Reverse the motor RPM frame

- Thankfully, there is a large community of Nissan Leaf tinkerers that already reverse engineered the Leaf frames and published it on the internet.

The screenshot shows the GitHub repository page for 'leaf\_can\_bus\_messages'. The repository is public and was forked from 'baradhill/leaf\_can\_bus\_messages'. It has 1 branch (master), 0 tags, and 109 commits. A merge pull request #7 from 'majbthrd/fixspeedo' is open. The repository contains several files, including .github, AV-CAN.dbc, CAR-can\_AZE0.dbc, DatabaseEditor.PNG, EV-can\_AZE0.dbc, EV-can\_ZE0.dbc, EV-can\_ZE1.dbc, LICENSE, QC-CAN\_ALL.dbc, README.md, and canmsgs.xlsx. The README section is visible, titled 'Nissan LEAF CAN bus messages (in .DBC format!)'. It mentions that this is an evolution of a spreadsheet found at a specific Google Docs link and that proper CAN database files are also in the repository along with the original excel spreadsheet. The repository has 42 forks, 180 stars, and 39 watchers.

leaf\_can\_bus\_messages (Public)  
forked from baradhill/leaf\_can\_bus\_messages

Sponsor Watch 39 Fork 42 Star 180

master 1 Branch 0 Tags  
Go to file Add file Code

This branch is 105 commits ahead of, 2 commits behind baradhill/leaf\_can\_bus\_messages:master

dalathegreat Merge pull request #7 from majbthrd/fixspeedo 5031032 · 3 months ago 109 Commits

|                    |                                                |              |
|--------------------|------------------------------------------------|--------------|
| .github            | Create FUNDING.yml                             | 5 years ago  |
| AV-CAN.dbc         | Initial version of AV-CAN                      | 5 years ago  |
| CAR-can_AZE0.dbc   | Merge pull request #7 from majbthrd/fixspeedo  | 3 months ago |
| DatabaseEditor.PNG | Add screenshot of database editor              | 6 years ago  |
| EV-can_AZE0.dbc    | Update EV-can_AZE0.dbc                         | 4 months ago |
| EV-can_ZE0.dbc     | Add cluster ECO findings                       | 2 years ago  |
| EV-can_ZE1.dbc     | Update EV-can_ZE1.dbc                          | 3 months ago |
| LICENSE            | Initial commit                                 | 6 years ago  |
| QC-CAN_ALL.dbc     | Update descriptions for V2X                    | 2 years ago  |
| README.md          | Response IDs added                             | 3 years ago  |
| canmsgs.xlsx       | Add some detail for GOM / battery bars (0x5a9) | 4 years ago  |

README GPL-3.0 license

### Nissan LEAF CAN bus messages (in .DBC format!)

This is an evolution of the spreadsheet found at <https://docs.google.com/spreadsheets/d/1EHa4R85BttuY4jZ-EnssH4YZddpsDVu6rUFm0P7ouwg/edit#gid=1>

Proper CAN database files are also found in this repository along with the original excel spreadsheet. The .dbc files are easier to use when working with CAN messages. Please note that there are major differences

Releases: No releases published

Sponsor this project: [patreon.com/dala](https://patreon.com/dala)

Packages: No packages published

[https://github.com/dalathegreat/leaf\\_can\\_bus\\_messages](https://github.com/dalathegreat/leaf_can_bus_messages)

# 0x1DA

## Inverter Status Frame

Signal Parameters

Name: MG\_OutputRevolution

Start Bit:

BITS ->

7

6

5

4

3

2

1

0

0

1

2

3

4

5

6

7

|                           |    |    |    |    |                          |                     |    |
|---------------------------|----|----|----|----|--------------------------|---------------------|----|
| 7                         | 6  | 5  | 4  | 3  | 2                        | 1                   | 0  |
| 15                        | 14 | 13 | 12 | 11 | 10                       | 9                   | 8  |
| 23<br>MG_ErrorCodes       | 22 | 21 | 20 | 19 | 18<br>MG_EffectiveTorque | 17                  | 16 |
| 31                        | 30 | 29 | 28 | 27 | 26                       | 25<br>MG_ErrorCodes | 24 |
| 39<br>MG_OutputRevolution | 38 | 37 | 36 | 35 | 34                       | 33                  | 32 |
| 47                        | 46 | 45 | 44 | 43 | 42                       | 41                  | 40 |
| 55                        | 54 | 53 | 52 | 51 | 50                       | 49<br>MG_Clock      | 48 |
| 63<br>CRC_1DA             | 62 | 61 | 60 | 59 | 58                       | 57                  | 56 |

Bit Length: 15

Byte Order: ☐ LSB First (Little Endian)

Type: UNSIGNED INTEGER

Scale: 1

Bias: 0

Min Value: -16382

Max Value: 16382

Units Name: rpm

Receiving Node: Vector\_XXX

Multiplexing: ☒ None ☐ Multiplexed ☐ Multiplexor ☐ Extended

Multiplex Low Value: 0

Multiplex High Value: 0

Multiplex Parent:

Comment:

Value Table:

# Step 2

## Reversing the Polo RPM Frame

- **Pro tip:** Do not remove the original engine before recording a complete CAN trace
- We could not find a DBC File for the 2007 Polo online
- VW is using similar frames on multiple cars of the same generation.

```
parse_can_logs / VW CAN IDs Summary.md
Preview Code Blame 111 lines (103 loc) · 3.81 KB Code 55% faster with GitHub Copilot
• 0x1AC
  ◦ (?) b1 - saw on 0/128 levels, ACC braking
  ◦ (?) b4 - noisy, correlates d(speed)
  ◦ (?) b5 - braking with engine, only on cruise ( bit7 )?
• 0x280
  ◦ b0 - clutch ( bit3 ) and acceleration ( bit0 ) pedals
  ◦ b1 = b4 = b7 - torque or engine load
  ◦ b3<<8+b2 - RPM
  ◦ b5 - acceleration pedal or cruise
  ◦ (?) b6 - some smooth graph
```

[https://github.com/v-ivanyshyn/parse\\_can\\_logs](https://github.com/v-ivanyshyn/parse_can_logs)

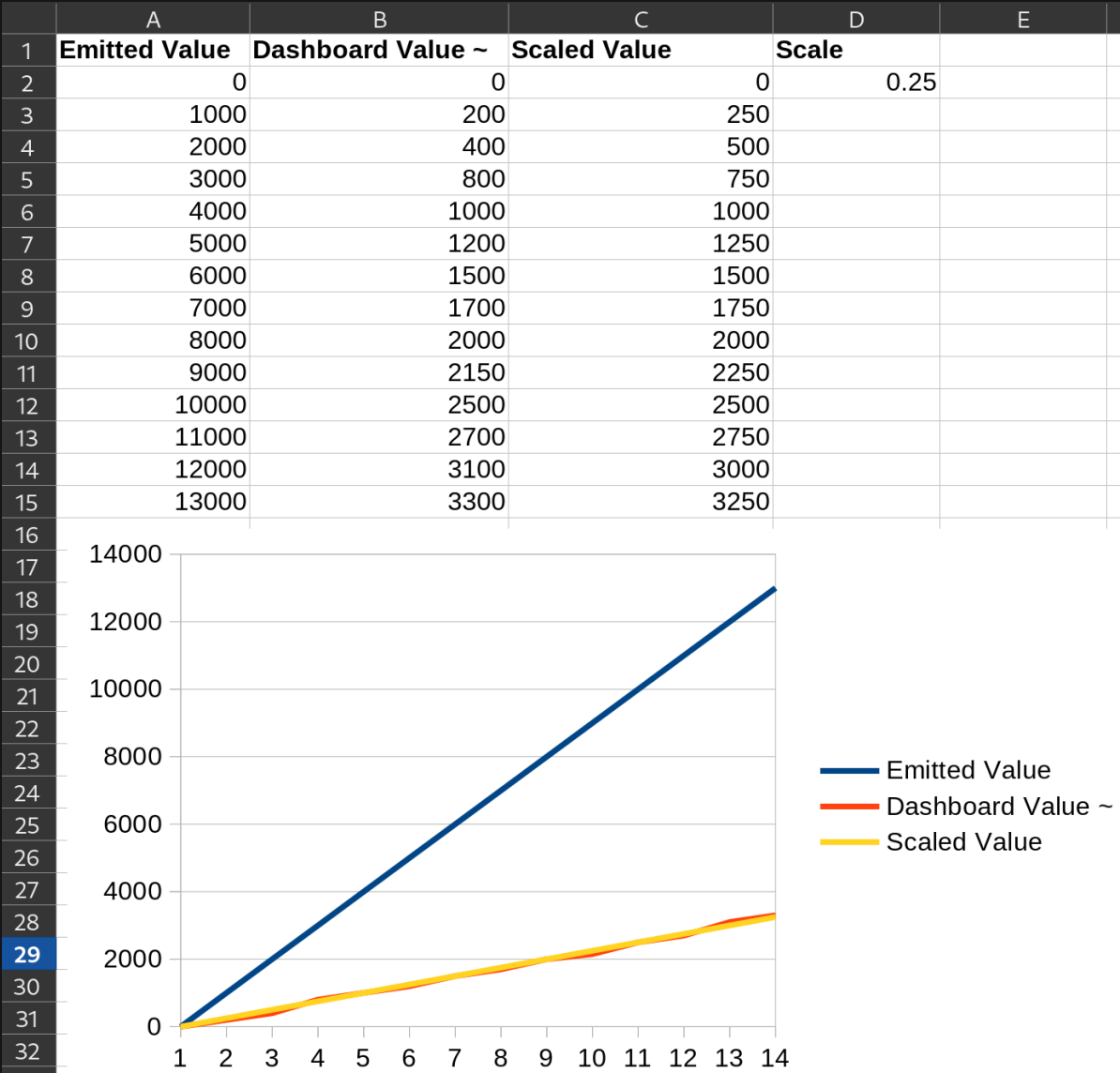
# Emitting a custom frame with SavvyCAN

Frame Sender

|   | En                                  | Bus | ID    | Len | Ext                      | Rem                      | Data                                    | Trigger | Modifications | Count |
|---|-------------------------------------|-----|-------|-----|--------------------------|--------------------------|-----------------------------------------|---------|---------------|-------|
| 1 | <input checked="" type="checkbox"/> | 0   | 0x280 | 8   | <input type="checkbox"/> | <input type="checkbox"/> | 0x00 0x00 0x08 0x18 0x00 0x00 0x00 0x00 | 100ms   |               | 220   |
| 2 | <input type="checkbox"/>            |     |       |     | <input type="checkbox"/> | <input type="checkbox"/> |                                         |         |               |       |

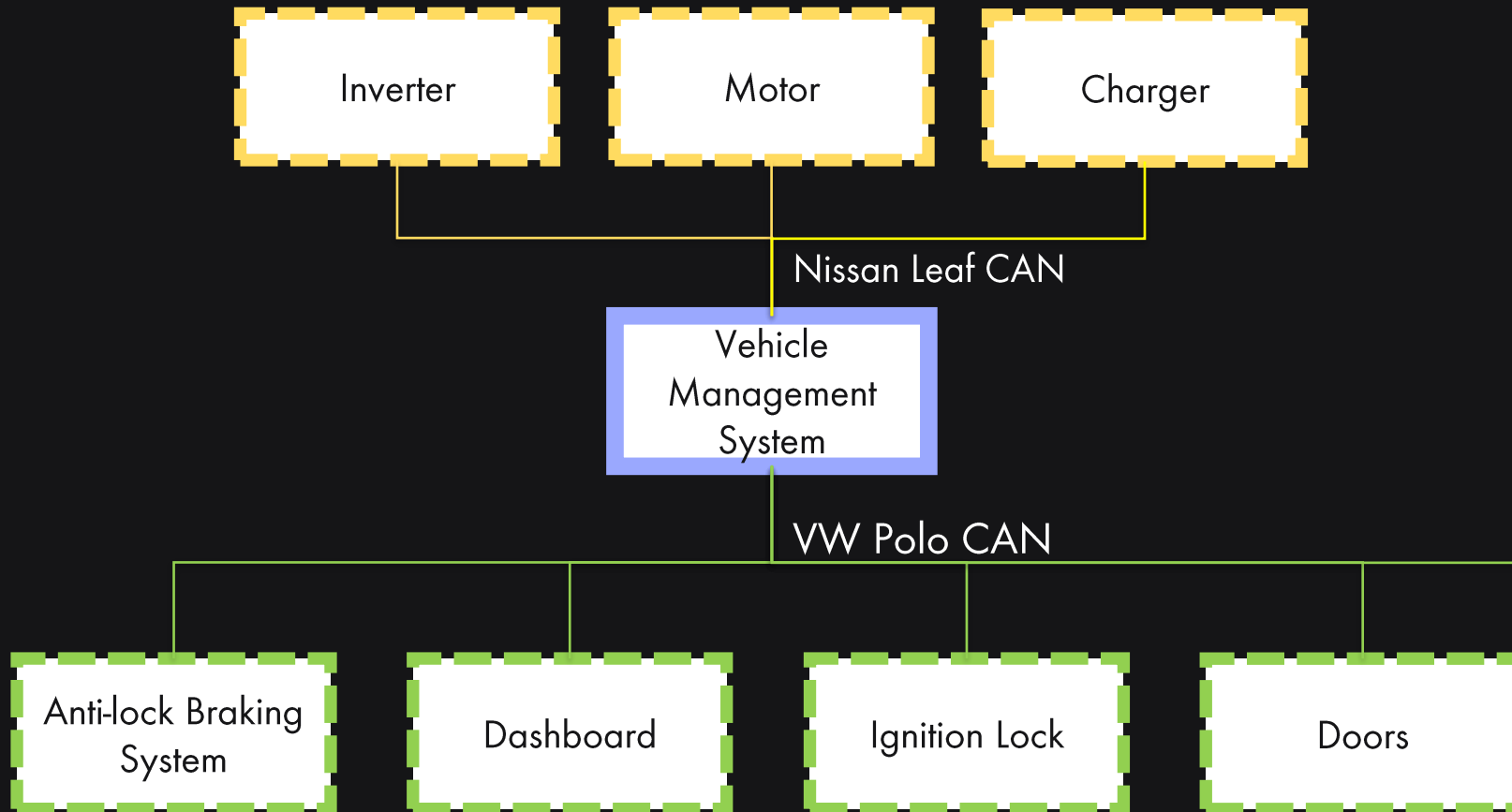
- Based on these findings, we tried to emit the 0x280 frame on the VW CAN bus at various frequencies.
- Some values of bytes 2 and 3 were actuating the RPM needle at 10Hz!
- Still need to translate real RPMs into the VW format
- $\text{physical value} = \text{offset} + \text{scale} * \text{raw\_decimal\_value}$

# Reversing the RPM *scale* value



# Step 3

Concrete implementation using Elixir



# CAN on Elixir

- Elixir is a dynamic, functional language running on the BEAM (aka the Erlang VM).
- The BEAM has been designed to build massively scalable **soft real-time** systems.
- Since Erlang was originally designed by Ericsson to build telephony switches, all the binary primitives are available out-of-the-box.
- Pattern matching is also helping a lot in parsing CAN frames.

# Receiving and parsing one frame

# Introducing **cantastic**

- An Elixir library doing all the heavy-lifting for sending and receiving CAN frames.
- Clean YAML DSL to describe your entire CAN Network
- Automatically spawns:
  - 1 receiver process (GenServer) per CAN network
  - 1 emitter process (GenServer) per emitted frame

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

# Receiving the Leaf RPM value

```
---
can_networks:
  leaf_drive:
    bitrate: 500000
    received_frames:
      name: inverter_status
      id: 0x1DA
      frequency: 10
      signals:
        - name: rotations_per_minute
          kind: integer
          endianness: big
          sign: signed
          value_start: 32
          value_length: 16
          unit: rpm
> polo_drive: ...
```

```
defmodule Leaf.Inverter do
  use GenServer
  ...
  def init(_) do
    Cantastic.Receiver.subscribe(self(), :leaf_drive, "inverter_status")
    {:ok, %{}}
  end

  def handle_info({:handle_frame,
    %Cantastic.Frame{name: "inverter_status", signals: signals}},
    state)
  do
    %{
      "rotations_per_minute" => %Cantastic.Signal{value: rotation_per_minute}
    } = signals

    Polo.Dashboard.set_rotation_per_minute(rotation_per_minute)
    {:noreply, state}
  end
  ...
end
```

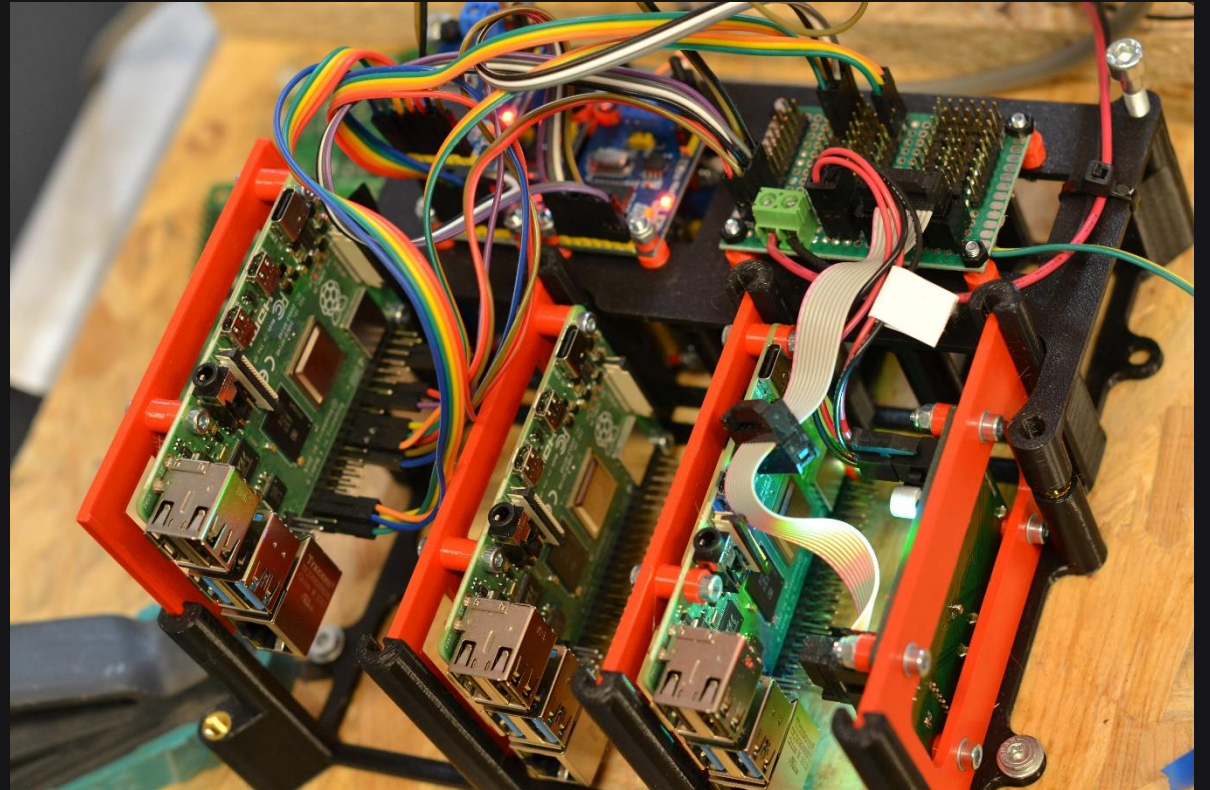
# Emitting the **Polo** RPM value

```
---
can_networks:
> leaf_drive: ...
  polo_drive:
    bitrate: 500000
    emitted_frames:
      name: engine_status
      id: 0x280
      frequency: 100
      signals:
        - name: rotations_per_minute
          kind: integer
          unit: rpm
          value_start: 16
          value_length: 16
          scale: "0.25"
```

```
defmodule Polo.Dashboard do
  use GenServer
  ...
  def init(_) do
    :ok = Cantastic.Emitter.configure(:polo_drive, "engine_status", %{
      parameters_builder_function: :default,
      enable: true,
      initial_data: %{
        "rotations_per_minute" => 0
      }
    })
    {:ok, %{}}
  end

  def set_rotation_per_minute(rotation_per_minute) do
    Cantastic.Emitter.update(:polo_drive, "engine_status", fn (data) ->
      %{data | "rotations_per_minute" => abs(rotation_per_minute)}
    end)
  end
  ...
end
```

# Let's push this on a Raspberry PI



# And... **It works!**





## Want some **more?**

- Loïc will present more details about the robotic part tomorrow at **13:35** in **UB2.147**.
- Marc will present the OVCS project tomorrow at **16:00** in **K.1.105**.
- [ovcs.be](http://ovcs.be)