

A solid blue triangle pointing to the right, positioned to the left of the main title text.

# Reverse-engineering CAN bus messages using OBD-II and correlation coefficients

Bram Blaauwendraad & Vincent Kieberl

Supervisors: Ruben Koeze & Sander Ubink, KPMG

# What is OBD-II?



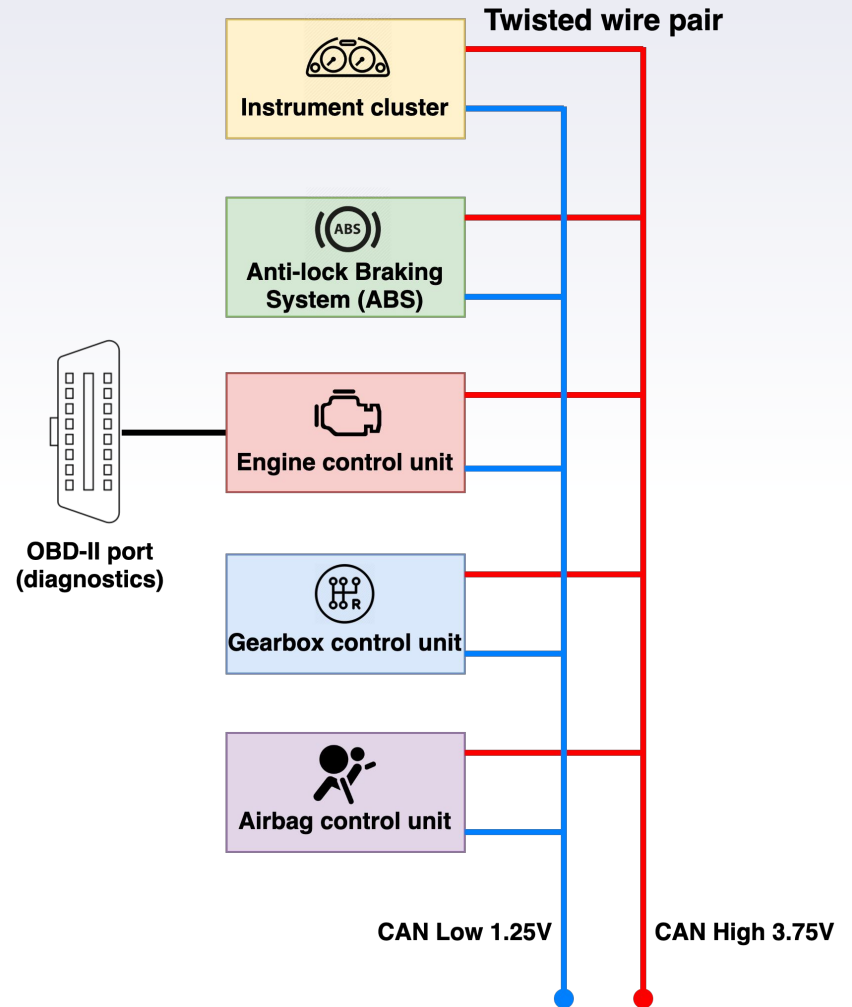
- ▶ On-Board Diagnostics II
- ▶ High level protocol that provides access to status and stored error codes of vehicle sub-systems
- ▶ Parameter identifiers (PIDs)

PID (hex)	Description
05	Engine coolant temp.
0C	Engine RPM
0D	Vehicle speed
10	Mass Air Flow rate

Source: SAE J1979 / ISO 15031-5:2015

# What is CAN?

- ▶ Controller Area Network
- ▶ Bus network: broadcast
- ▶ Saves on copper wiring costs
- ▶ CAN IDs identify message types
  - ▷ not public information
- ▶ Most CAN IDs occur regularly
- ▶ Meant for closed systems → insecure



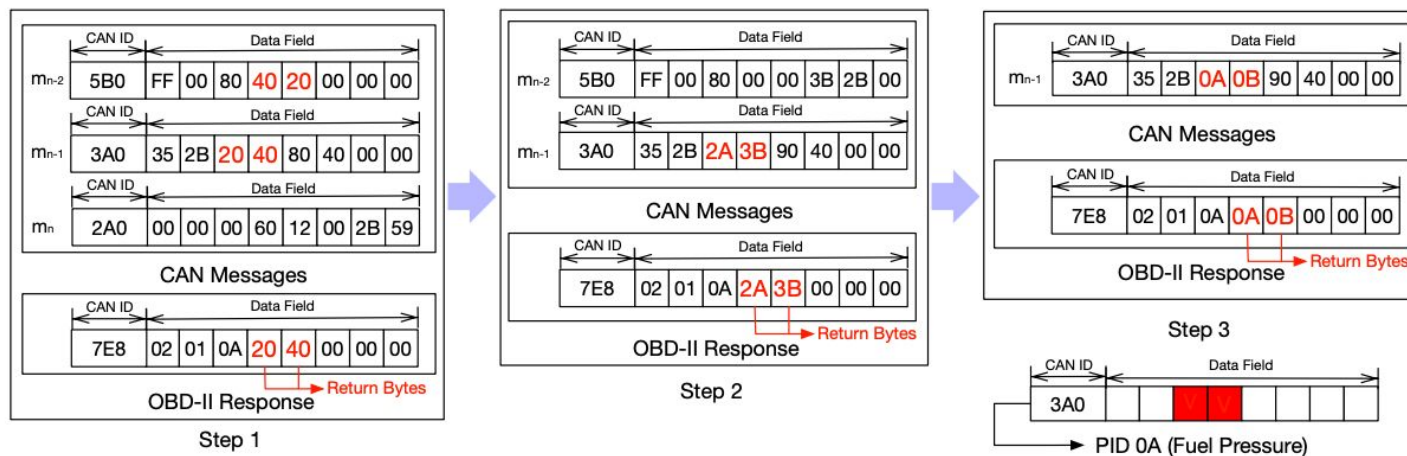
# Automotive IDSs

- ▶ Increasing amount of electronics in vehicles
- ▶ IDSs currently use features from traffic metadata<sup>1</sup>
- ▶ Content-based features may improve accuracy
  - ▷ Data plausibility checks

<sup>1</sup>Bresch, M. and Salman, N. *Design and implementation of an Intrusion Detection System (IDS) for in-vehicle networks*. Gothenburg: University of Gothenburg, 2017.

# Prior work: Kang et al.


- ▶ Automated reverse-engineering of CAN frames using OBD-II
- ▶ Matching OBD-II value to CAN data
- ▶ Process of elimination



# ▶ Prior work: Kang et al. (2)

- ▶ Only search for one-on-one matching value
- ▶ Initial experiments show that in Audi A4 B7, translation is used
- ▶ Approach Kang et al. does not work for translated values

id            data  
0x288        4A B8 18 00 00 53 69 00

  
0xB8 = 184

$184 * 0.75 - 48 = 90$  deg Celsius  
= engine coolant temperature

288 8 10ms Motor 2

---

```
byte0 xx=83,f2,29,43  
byte1 52-a5 Kühlmitteltemperatur (0,75x Wert-48°, 0xff ist Fehler) 8 Mal -> 0x420  
byte2 00 00->01->03->02->00 Bremspedal (=10 Leerlauf Bit4=ACC ?), Bremslichtschalter  
-byte3 00 =0  
-byte4 00 =0  
byte5 (5b 58 56) 00,86,87,88,89  
byte6 00 4a val (tps/maf/et al.)  
-byte7 00 =0
```

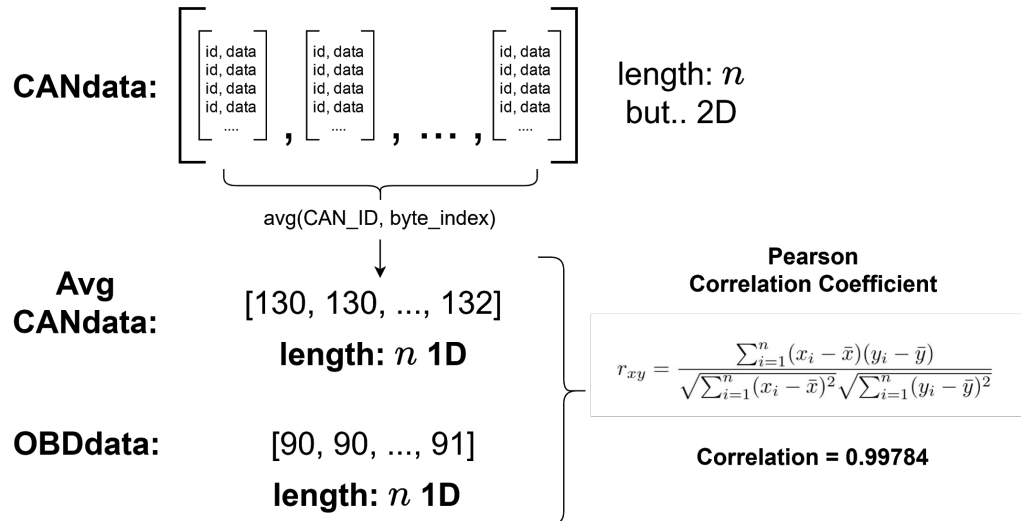
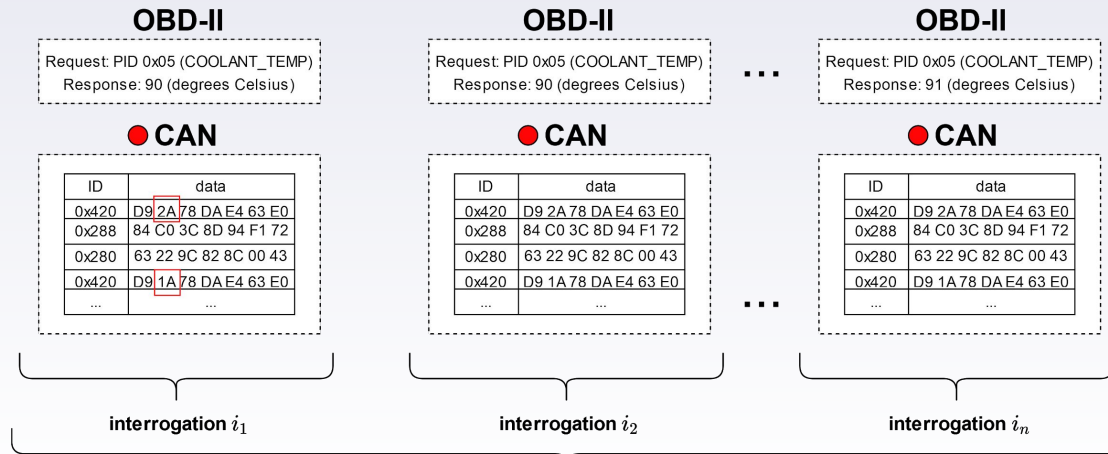
# ▶ Research question

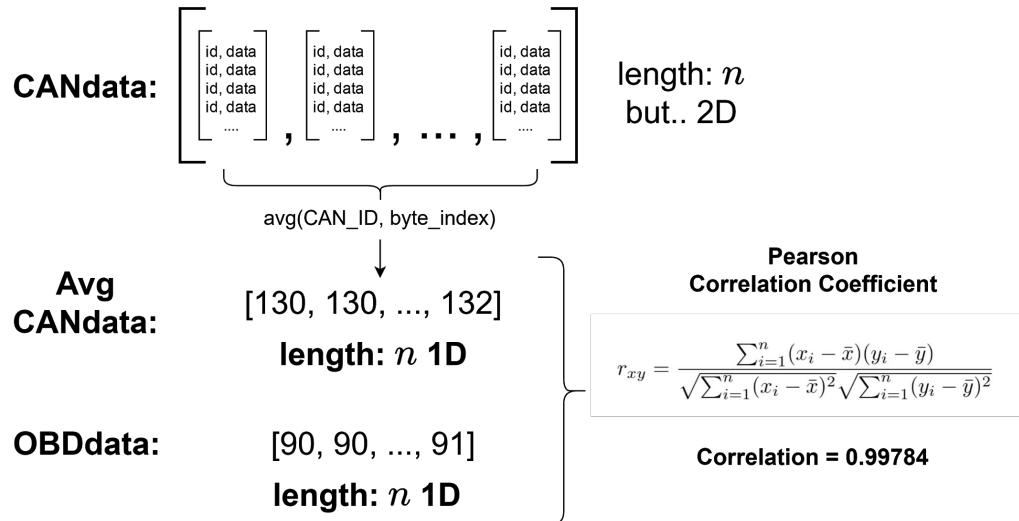
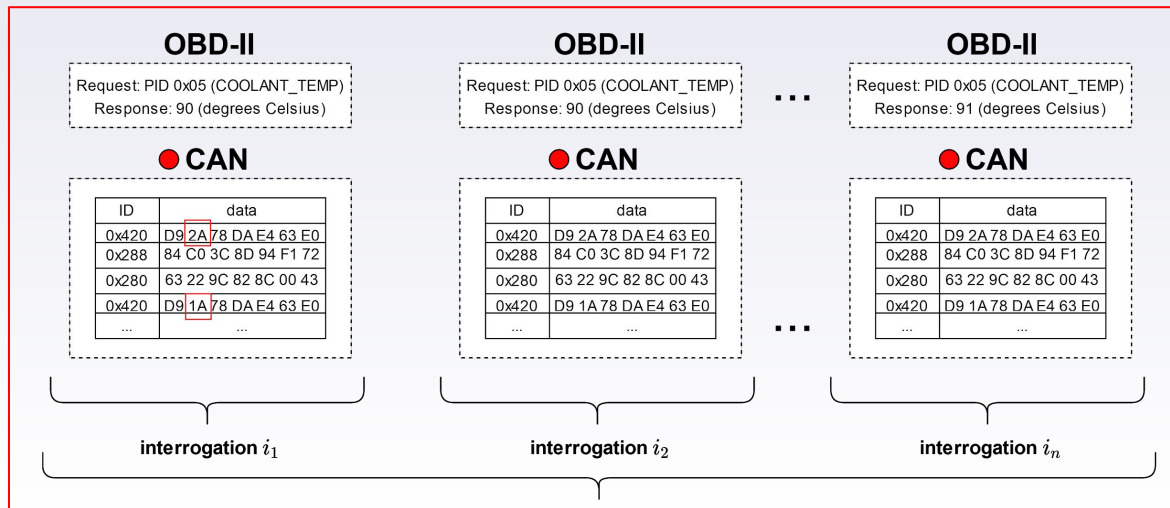
To what extent can we reverse-engineer CAN messages using OBD-II interrogations and correlation coefficients when a translation is used?

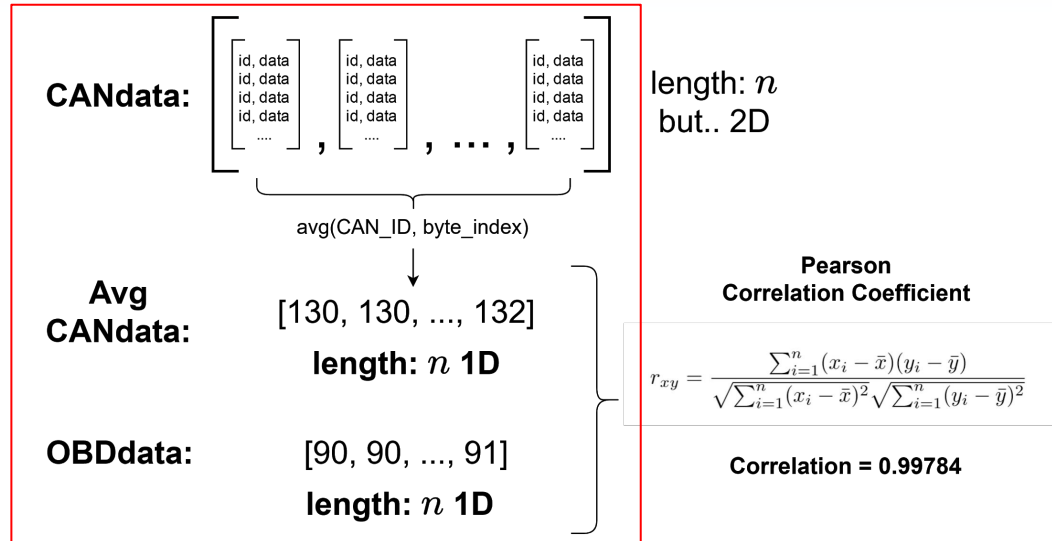
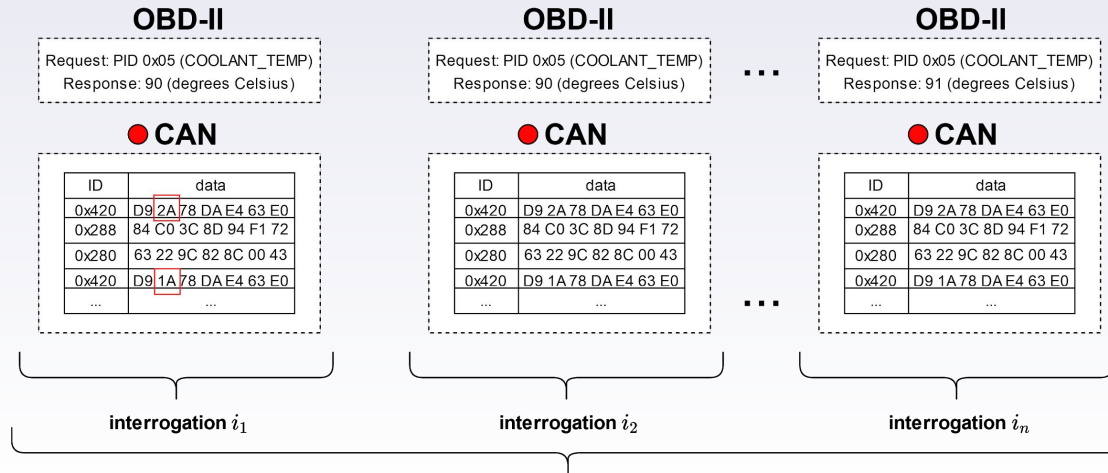


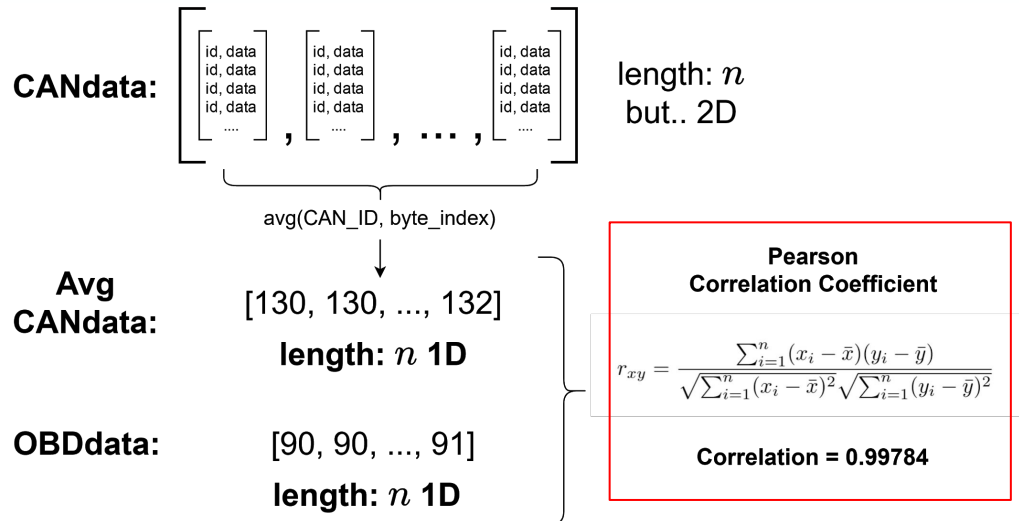
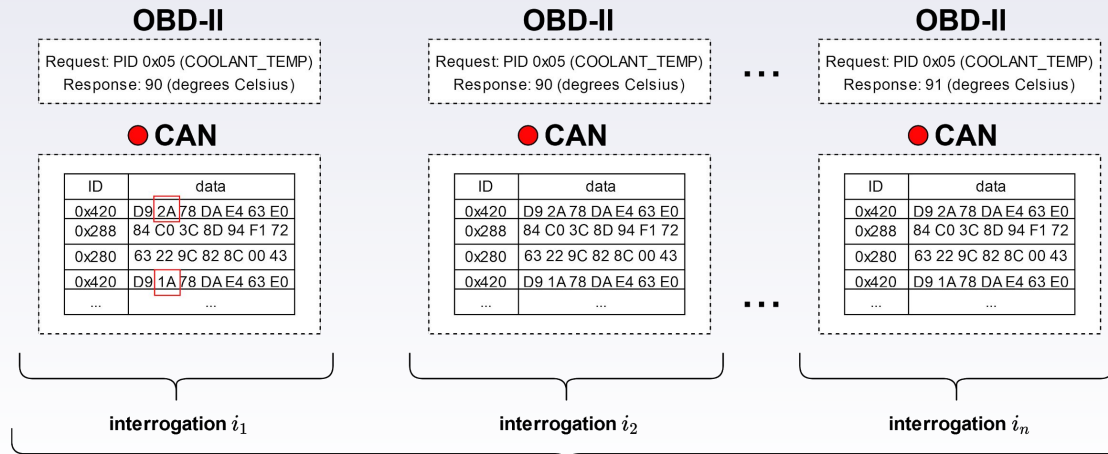
# Methodology: theory

1. Start listening on CAN bus
2. Do OBD-II request for supported PID
3. Stop listening on CAN bus
4. **Compute averages** for every unique CAN ID + byte index pair
5. Calculate Pearson Correlation Coefficient [OBDdata][CANdata]



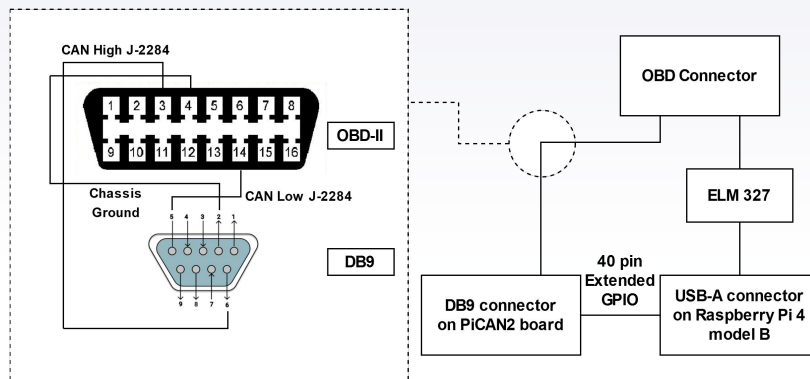






# Methodology: practical

- ▶ Audi and Hyundai
- ▶ 100 / 200 interrogations
- ▶ Testing procedure



# Methodology: proof-of-concept

- ▶ Python 3
- ▶ Multithreading
  - ▷ Get CAN data asynchronously

```
# Get OBD response
OBD_response = OBDCONN.query(obd.commands[PID_name])
if OBD_response.is_null():
    print("Error: no OBD response received!")
    sys.exit(-1)
OBDdata = OBD_response.value.magnitude
```

## Steps

1. Get supported PIDs
2. Get CAN and OBD data for each PID
3. Compute averages
4. Compute correlation and save to CSV

	A	B	C	D	E	F	G
1	CAN ID (Dec)	CAN ID (Hex)	Index	PID	Correlation	OBD Data	CAN Data
2	640 280		3	RPM	0.997369715	[1149.0, 1333.0, 1272.0,	[17.0, 20.0, 19.0, 19.0, 19.0, 20.0, 19.0,
3	640 280		5	RPM	0.76934128	[1149.0, 1333.0, 1272.0,	[1.0, 30.0, 33.0, 33.0, 34.0, 34.0, 33.0, 20.0, 2
4	896 380		2	RPM	0.76102077	[1149.0, 1333.0, 1272.0,	[2.0, 32.0, 33.0, 33.0, 34.0, 34.0, 33.0, 18.0, 2
5	1416 588		4	RPM	0.733531356	[1149.0, 1333.0, 1272.0,	[105.0, 105.0, 106.0, 106.0, 107.0, 107.0, 107.
6	640 280		4	RPM	0.698583493	[1149.0, 1333.0, 1272.0,	[6.0, 43.0, 51.0, 51.0, 52.0, 52.0, 51.0, 33.0, 2
7	640 280		1	RPM	0.696430727	[1149.0, 1333.0, 1272.0,	[6.0, 43.0, 51.0, 51.0, 52.0, 52.0, 51.0, 33.0, 2
8	1160 488		2	RPM	0.694699297	[1149.0, 1333.0, 1272.0,	[6.0, 43.0, 50.0, 51.0, 52.0, 52.0, 51.0, 32.0, 2
9	640 280		7	RPM	0.635676569	[1149.0, 1333.0, 1272.0,	[1.0, 43.0, 50.0, 51.0, 52.0, 52.0, 51.0, 28.0, 2
10	1100 44C		2	RPM	0.632512892	[1149.0, 1333.0, 1272.0,	[197.0, 196.0, 196.0, 196.0, 196.0, 196.0, 196.
11	1160 488		1	RPM	0.630863151	[1149.0, 1333.0, 1272.0,	[1.0, 44.0, 50.0, 51.0, 52.0, 52.0, 51.0, 27.0, 2
12	896 380		3	RPM	0.609736989	[1149.0, 1333.0, 1272.0,	[35.0, 13.0, 20.0, 21.0, 22.0, 22.0, 21.0, 5.0, 5
13	1344 540		1	RPM	0.540933141	[1149.0, 1333.0, 1272.0,	[116.0, 106.0, 106.0, 107.0, 107.0, 108.0, 108.
14	1504 5E0		2	RPM	0.528736185	[1149.0, 1333.0, 1272.0,	[55.0, 56.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0,
15	1088 440		4	RPM	0.489807006	[1149.0, 1333.0, 1272.0,	[131.0, 131.0, 131.0, 131.0, 131.0, 131.0, 131.0, 131.
16	648 288		5	RPM	0.433755331	[1149.0, 1333.0, 1272.0,	[76.0, 76.0, 76.0, 76.0, 76.0, 76.0, 76.0, 76.0,

# Methodology: fitting

- ▶ Reverse-engineer formula used on CAN data
- ▶ When correlation  $> 0.9$
- ▶ We assume  $y = ax + b$ 
  - ▷ Fit value in 8-bit integer and allow negative values
  - ▷ E.g. COOLANT\_TEMP =  $0.75x - 48$

```
byte1 52-a5 Kühlmitteltemperatur (0,75x Wert-48°, 0xff ist Fehler) 8 Mal -> 0x420
```



# Results

- ▶ PIDs with High correlation on all tests
- ▶ Examples:

<b>PID</b>	<b>CID + Byte Index (int 100 &amp; 200)</b>	<b>Correlation</b>
Audi RPM	0x280 - 3	~0.997
Audi INTAKE_PRESSURE	0x588 - 4	~0.999
Audi MAF	0x288 - 6	~0.962
Hyundai COOLANT_TEMP	0x329 - 1	~0.992
Hyundai THROTTLE_POS	0x329 - 5	~0.972

# Results (2)

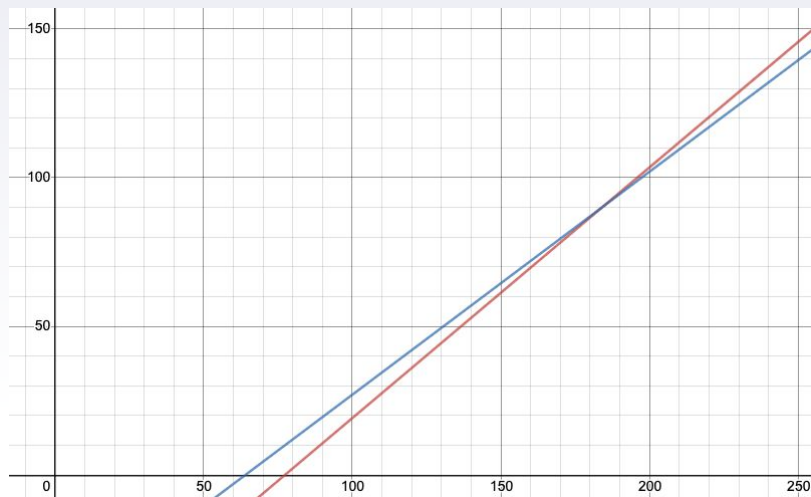
- ▶ PIDs with no matches (correlation < 0.9)
- ▶ Examples:
  - ▷ Audi - ENGINE\_LOAD
  - ▷ Audi - INTAKE\_TEMP
  - ▷ Hyundai - AMBIENT\_AIR\_TEMP
  - ▷ Hyundai - EVAPORATIVE\_PURGE
- ▶ Potentially a combination of CAN values (e.g. 0x280-2 + 0x360-4)

# Results (3)

- ▶ PIDs with ambiguous result
- ▶ Example:
  - ▷ COOLANT\_TEMP on Audi matches on both coolant temperature and oil temperature
  - ▷ In certain driving conditions, these behave almost identically
  - ▷ Different testing procedure solved this problem

# Result (4)

- ▶ Exact formula not found, however:
  - ▷ Close approximation when range is known
  - ▷ Lower resolution through averages
  - ▷ Still useful for IDS



**Suspected Formula**

$$y = 0.75x - 48.0$$

**Found Formula**

$$y = 0.84x - 65.2$$

# Discussion

It works, however...

Practical considerations:

- ▶ What parameter are you looking for
- ▶ Fluctuations in environment variables
- ▶ Amount of CAN messages in test vehicle

Thus: not *one* optimal setup

# Conclusion

- ▶ Correlation can be used to map CAN ID and byte indices to OBD values and formulas can be approximated with some limitations.
- ▶ Limitations
  - ▷ No correlation (possibly a formula)
  - ▷ Testing Procedure matters
  - ▷ 1 on 1 match will not be found (correlation = n/a)
  - ▷ Only works on one byte values (max. 255)

# Future work

- ▶ Bigger sample size
  - ▷ Limited through OBD port, (security) gateways
- ▶ Conclusive proof
  - ▷ Reverse-testing
- ▶ Extensive testing procedure
- ▶ Performance