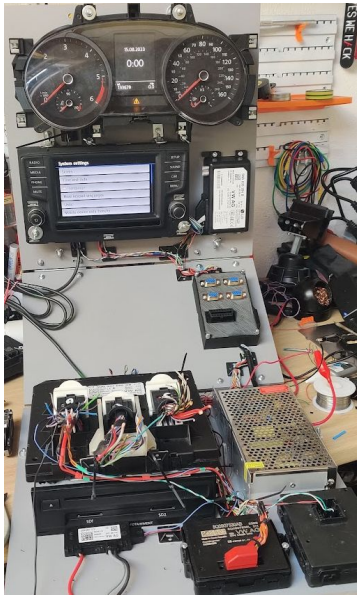
Connected Car Hacking

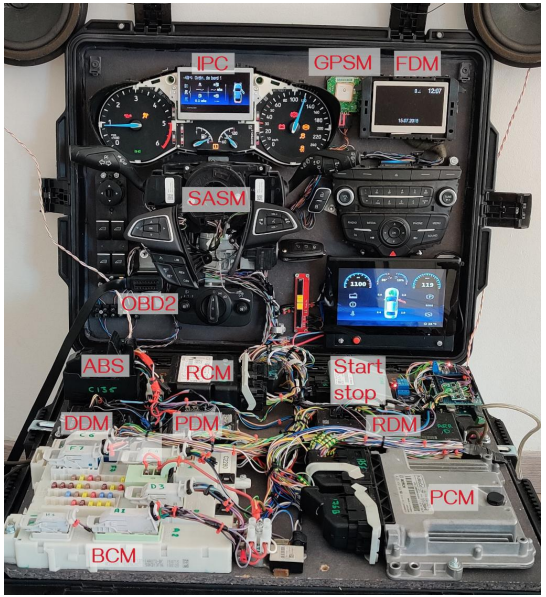
Building a test bench

Quarkslab

Usage of an automotive test-bench

- Have a functional set of **ECUs**
- No risk of **breaking your car**
- Control of the various **on-board network/bus** and **I/Os**





Which car to work on ?

- If you have choice, select a model **you could have access to**
- Beware of **too old** (missing useful functionalities/protocols) or **too recent** model (ECU price, communication encryption, ...)
- Choosing a **common model** allows to easily get access to spare ECUs

Illustration: [link]

Which ECU to focus on

- **Instrument Cluster**: provide a lot of outputs to understand what's happening on the bench, good starting point
- **Infotainment:** a lot of connectivity (Wi-Fi, Bluetooth, sometime 3GPP), verbose logs
- **Telematic:** external connectivity using 3GPP modem, most critical attack surface, but may require infotainment unit to have active traffic to backend
- **Body Control Module**: core of the car, handle most of the time the wake-up frames, locking systems...
- **Gateway**: critical security component, could also handle wake-up frames
- **Other ECUs**: depending on your needs and topics of interest (RF/RFID with RKE/PKE, Airbag crash data logger, V2G...)

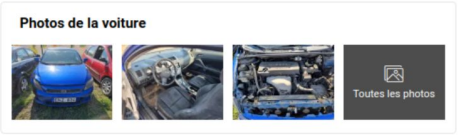
Where to find target ECUs

- **Scrapyard**
  + Could check/test on-site if the ECU is working
  + Guarantee to have access to the connectors
  - Could be difficult to find a specific model
  - Dismantling the car by yourself requires adequate tools and some time

- **Internet**
  + Large range of ECUs available
  - Sometime you may receive a broken ECU…
  - Connectors are often missing

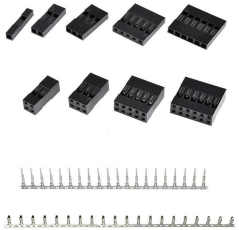
We do recommend finding **website that list other part for the same vehicle**, to avoid any enlistment issues

Moreover, spending a few more bucks on an ECU with its connector will save you time



Recommended bench tools

- **Lab voltage regulator**: you may need to power over 12V some ECU (approx. 14V when a vehicle is running), w/ peak current of 6/8A for some units
- **Wago connectors**: easily interconnect bus, power lines...
- **Dupont headers**: useful when you don't have the ECU's connector
- **120 ohms resistors**: complete missing termination on CAN bus



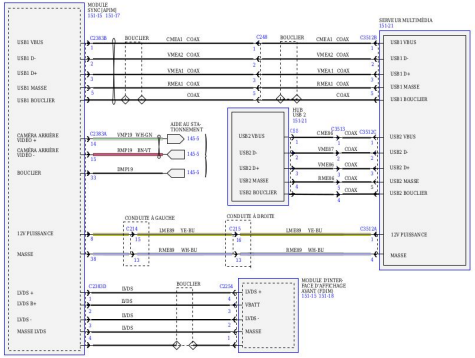
Training test-bench

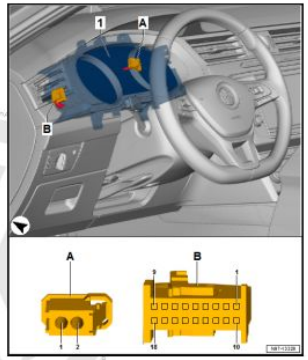
Four ECUs will be used to create a test-bench for this training:

- **Instrument Cluster (IC)**: base ECU to get a visual display of the state of the bench
- **Body Control Module (BCM) & a gateway (GW)**: sends wake-up frames, controls main input/outputs (BCM), filters and routes CAN messages between the different CAN buses (GW)
- **In-Vehicle Infotainment unit (IVI)**: multimedia unit which can be paired with smartphone for sending/receiving calls or broadcasting music
- **Telematic Control Unit (TCU)**: handles Internet connectivity and emergency calls

Getting technical information

- Look for technical documentation, to **identify ECU** (type, location), **available networks** and how the OBD-II port is wired
- Mechanic forums are good, but you'll get only a few schematics
- Every manufacturer has a website dedicated to **mechanic workshop**, but access needs subscription (could be hourly access). You will find exact **diagram**, ECU **pinout**, and sometimes you can download **firmware updates**
- Try to find CAN database (.dbc), like on comma.ai's GitHub: https://github.com/commaai/opendbc





List of manufacturers' technical data

Audi/VW	https://erwin.audi.com/ & https://erwin.vw.com/	**Mercedes**	http://www.startekinfo.com/
BMW	http://www.bmwtechinfo.com/	**Mini**	http://www.minitechinfo.com/
Citroen	http://public.servicebox.peugeot.com/pages/index.jsp	**Mitsubishi**	http://www.mitsubishitechinfo.com/
Fiat	https://www.technicalinformation.fiat.com/	**Nissan**	https://www.nissan-techinfo.com/home.aspx
Ford	https://www.etis.ford.com/	**Peugeot**	http://public.servicebox.peugeot.com/
Honda	https://techinfo.honda.com/rjanisis/logon.aspx	**Porsche**	https://techinfo2.porsche.com/
Hyundai	http://www.hyundaitechinfo.com/	**Renault**	https://newdialogys.renault.com/
Jeep	http://www.techauthority.com/	**Saab**	http://epsiportal.com/Site/SAAB
Kia	https://kiatechinfo.snapon.com/default.aspx	**Smart**	http://www.smarttekinfo.com/SmartTek/
Land Rover	http://www.landrovertechinfo.com/	**Tesla**	https://service.teslamotors.com/
Lexus	https://techinfo.lexus.com/	**Toyota**	http://techinfo.toyota.com/
Mazda	https://www.mazdaserviceinfo.com/	**Volvo**	http://www.volvotechinfo.com/

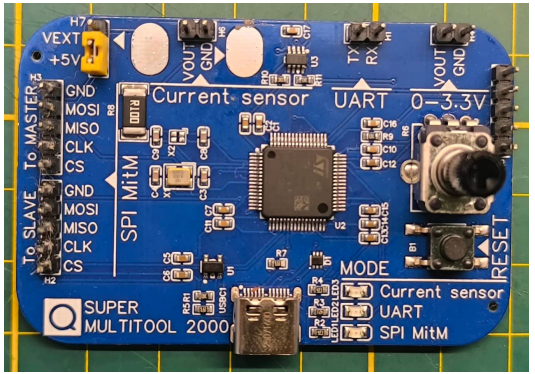
Lab 1 - Reading technical documentation

Goals

- For this lab, you'll study wiring documentation, coming from or inspired by different manufacturers technical data
- Complete challenges **Test bench - Documentation**

Instrument Cluster

- The **Instrument Cluster** will be the first ECU we will work with
- Having a display, it is really simple to work with and get an idea of its **state**
- We will power it using the **Super Multitool 2000** current sensor pins
- Before wiring it, perform all the required challenges to get a visual confirmation of the +5V and GND pins, to avoid any short circuit that could damage the device



Lab 2 - Identifying pin of the Instrument Cluster

Goals

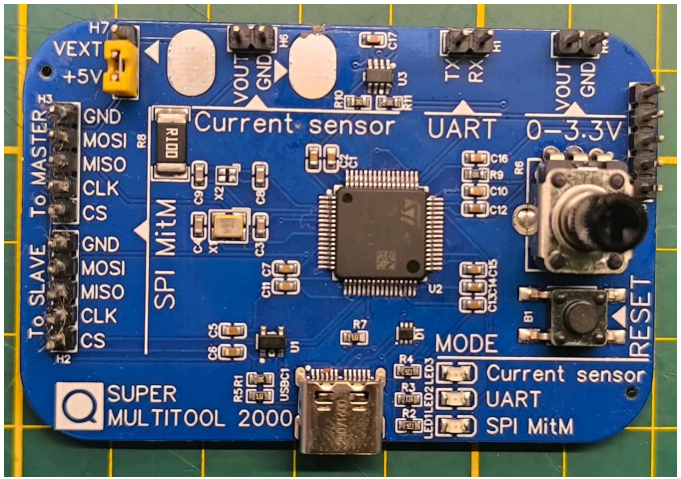
- Complete challenges **“Reference”** to **“Powering”** in the **Test bench - Instrument Cluster** category
- Pay attention on the **VOUT** and **GND** pins when connecting to the **Super Multitool 2000**

Wake-up frames

- ECUs are designed to be **power saving**, to avoid draining the car's battery
- You may need to find a **wake-up frame**
  - could be **any activity** on the CAN bus
  - or a **specific CAN frame**
- Some ECUs require a **12V input when ignition is on**, look for pin "wake signal", "ign/ignition" or "terminal 15/KL15"
- Wake-up frames are usually sent by the **Body Control Module**
- Content of the wake-up frame varies depending on the **ignition state**

Finding the wake-up frames

- **Capture** the CAN traffic of a working car and **replay it**
- Beware, sending data from **non-enrolled ECUs** may trigger some **anti-theft protection**
- Try every Arbitration ID **incrementally**, sending **8 bytes long** message starting from “00 00 00 00 00 00 00 00” to “FF FF FF FFF FF FF FF FF” and monitor the **current consumption** of your lab generator
- Having a lab generator with Ethernet/USB connectivity is a plus to automatise this search
- Otherwise, you can use a microcontroller and a current sensor to do so, like the **Super Multitool 2000**



The Super Multitool 2000

- To monitor the power consumption of an ECU, you can use the **Super Multitool 2000**
- It's based on a **STM32F4** and a current sensor **INA219**
- You can send commands to the multitool over its serial interface
- Jumper allow choosing the **power source**: **VEXT** when using a lab power generator, or **+5V** from the USB port.
- The device under test needs to be powered using **VOUT** and **GND** pins

The Super Multitool 2000 - current sensor commands

- Sending “**m**” (measure) through the UART over USB to compute the average current consumption of the device under test
- It samples 40 times the current consumption with an interval of 10ms
- Once the measurement is done, a threshold is computed
- To arm the current trigger, send “**a**” (arm). The **Super Multitool 2000** will continuously read the current on the **INA219** and send character “**!**” if the threshold is reached
- This mode can be cancelled by sending “**e**” (exit) command

The Super Multitool 2000 - current sensor commands

```
user@user: ~$ screen /dev/ttyACM0
h
Supported commands:
- [h]elp
- [e]xit current mode
==== Current sensor ====
- [m]easure
- [a]rm
[... redacted for readability …]

m
Average current: 6 mA, Max current: 6 mA, Threshold: 9 mA
a
Current detection armed, threshold: 9 mA

!Current increase detected: 10 mA (Threshold: 9 mA)
```

Lab 3 - Finding the wake-up frame

Goals

- Using the **Super Multitool 2000**, you'll use the current sensor to find the wake-up frame
- Complete challenges **"Wake-up frames"** in **Test bench - Instrument Cluster**
- When the wake-up frame is received, the current consumption increase occurs within a few milliseconds, it is recommended to try 3 to 4 times each Arbitration ID
- If you are not comfortable to use serial interface with Python, a base script is accessible through the hints

Message database & UDS scanning

Once the pin discovery is performed and the ECU is working, it is a good practice to:

- **Identify Arbitration ID sent**: to establish a CAN database
- **Find out UDS identifiers resistance**: use main UDS Services to find TX/RX arbitration ID
- **List supported UDS Services**: could be helpful to monitor ECU state, find forgotten developer options or help during reverse engineering phases

Illustration: [link]

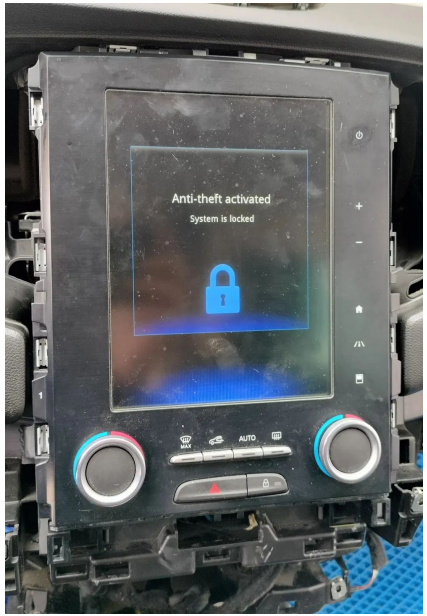
Lab 4 - Finding CAN and UDS dataset

Goals

- Now the **Instrument Cluster** is started, identify the Arbitration ID used for the CAN messages sent by the IC
- Build a scanner to find the **Arbitration ID** used for **UDS requests**
- Complete the challenge **UDS TX/RX ID** from **Test bench - Instrument Cluster**

Anti-theft protection

- Some ECUs implements an **anti-theft** detection
- If triggered, the ECU could be **disabled** or **work partially**
- It could be triggered by an **invalid message** or a **missing CAN message**
- It is based on a shared “**secret**” by other ECU being part of the enrollment
- On older **IVI**, it uses a **PIN** system



Lab 5 - Bypassing the anti-theft protection

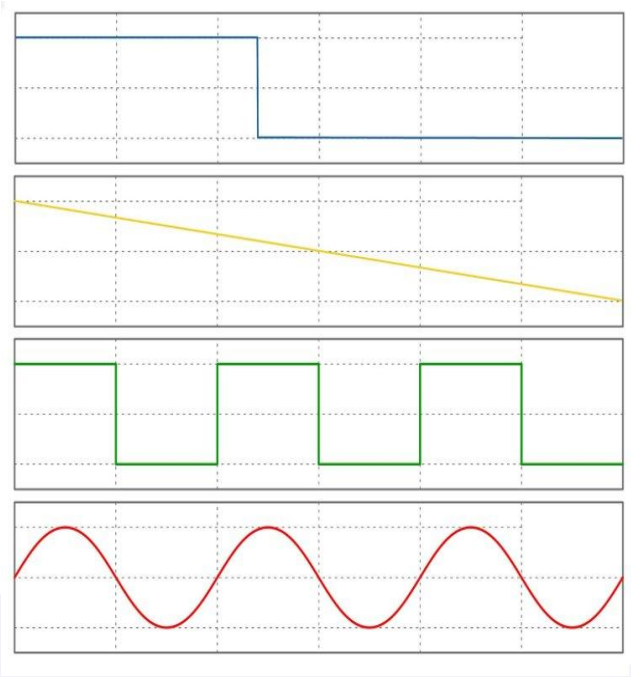
Goals

- Try to bypass the anti-theft protection. To do so, you'll first read the **VIN** of the vehicle, using **UDS** request **ReadDataByIdentifier**, then analyse a capture from a working vehicle
- Complete the challenges **VIN** and **Bypassing anti-theft protection** from **Test bench - Instrument Cluster**

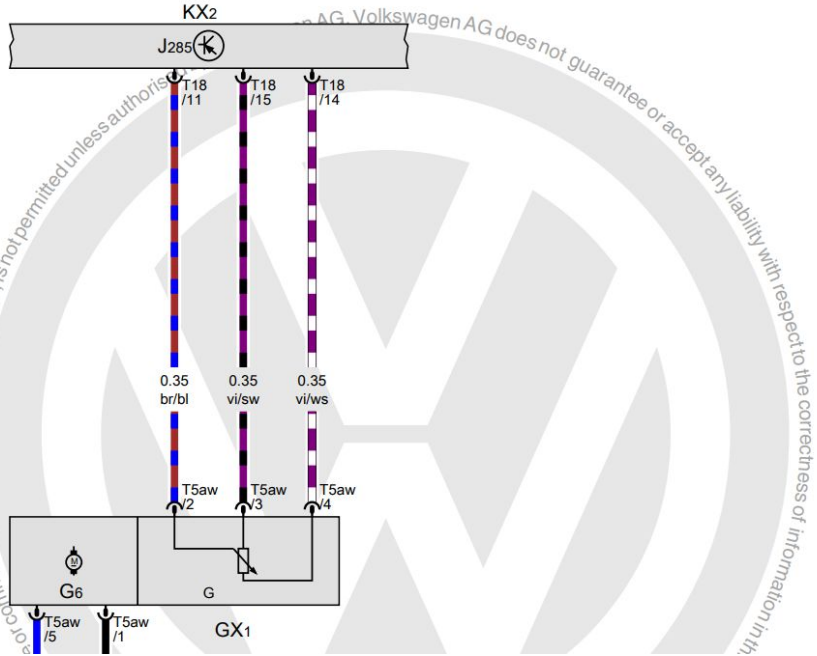
Emulating sensors

Sensors have 4 types of output:

- **Relay to ground (or +12V)**
  Input read 1 if connected to GND (or +12V)
- **Variable resistance**
  Mostly 5V signal going through 1-5K potentiometer
- **Square signal**
  Also 5V signal
- **Wave signal**
  More complex to emulate, need additional hardware



Instrument Cluster sensor diagram

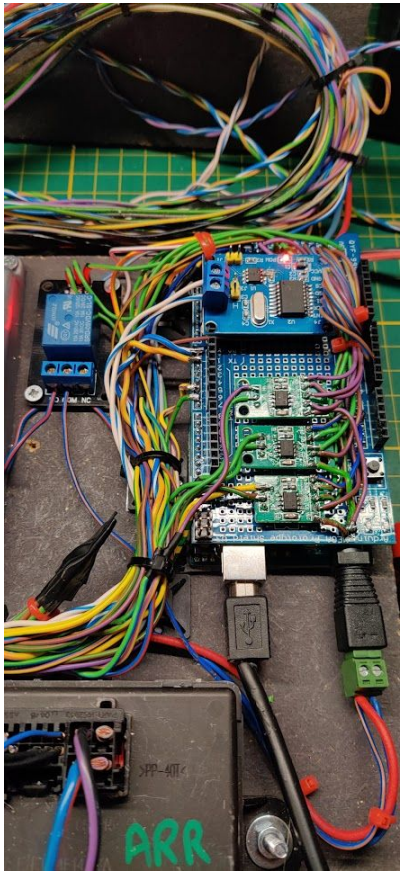


Fuel delivery unit, Fuel pump control unit, Dash panel insert

G	Fuel gauge sender
GX1	Fuel delivery unit
G6	Fuel system pressurisation pump
J285	Control unit in dash panel insert
J538	Fuel pump control unit
KX2	Dash panel insert
T5aw	5-pin connector, black
T5ax	5-pin connector, black
T18	18-pin connector, black
(480)	Earth connection, in fuel tank wiring harness
(720)	Earth point on right B-pillar

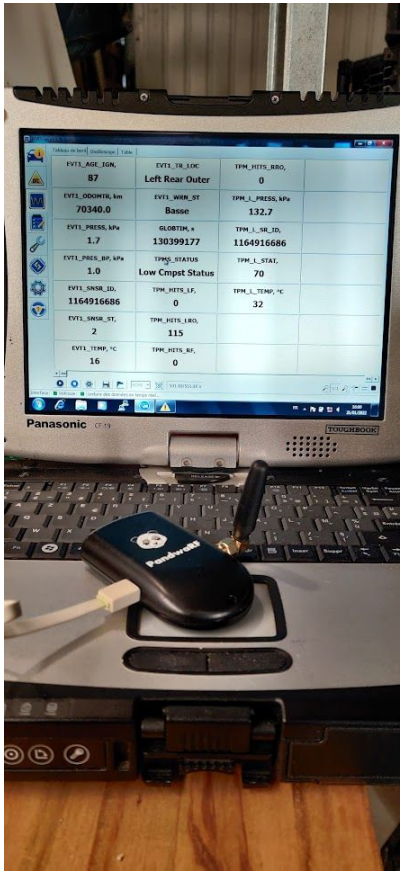
Example of sensors emulation

- Switching pins mode (input/output) is used to have **signal to ground** control
- **12V output** is handled by a relay
- **Digital potentiometer** emulates temperature and fuel level sensors
- An **Arduino** board adapts values through its **UART** interface



Not all sensors are wired

- Direct **Tire Pressure Monitoring System** (**TPMS**) use radio-frequency
- Each sensor **sends periodically** data to the BCM
- Signal are crafted with the Rfcat library and supported devices based on a **CC1101 transceiver**



Diagnostic tools

- Having access to official or 3rd party **diagnostic tools** is a time saver for this tasks
- It allows to **requests read value** from the ECU and evaluate **valid inputs**
- Some tools allow to modify **ECU configuration**, using under the hood UDS Service Write Data By Identifier, which can be useful to revert crash mode for example

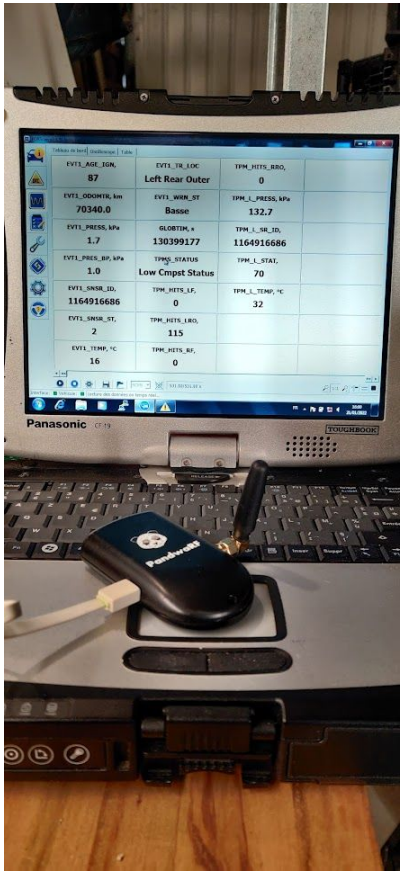


Illustration: [link]

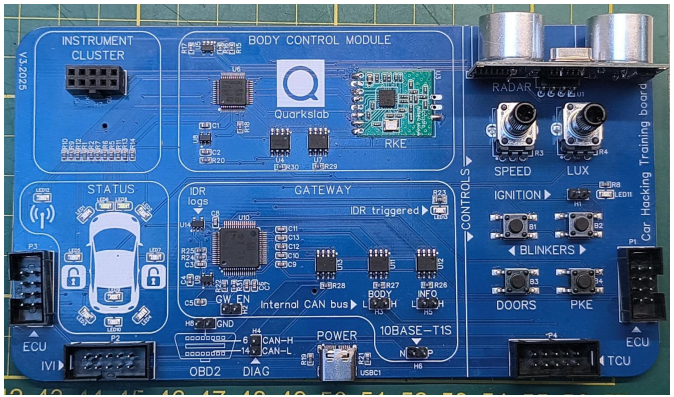
Lab 6 - Sensor emulation

Goals

- Using the **Super Multitool 2000**, you will emulate a basic sensor on the pin 6, tied to the **Lux reader**
- Complete the remaining **Test bench - Instrument Cluster** challenges

Body Control Module

- Next ECU will be the **Body Control Module**, which will handle the wake-up frames and provide some inputs/outputs
- It will be powered through its **USB-C plug**
- On the **Body Control Module** PCB, there is also a **Gateway** and an **OBD-II** plug for diagnosis purposes (not used in this training)
- The PCB provides pins to connect to the two **CAN buses** (**BODY** and **INFO**) and the **10BASE-T1S** network



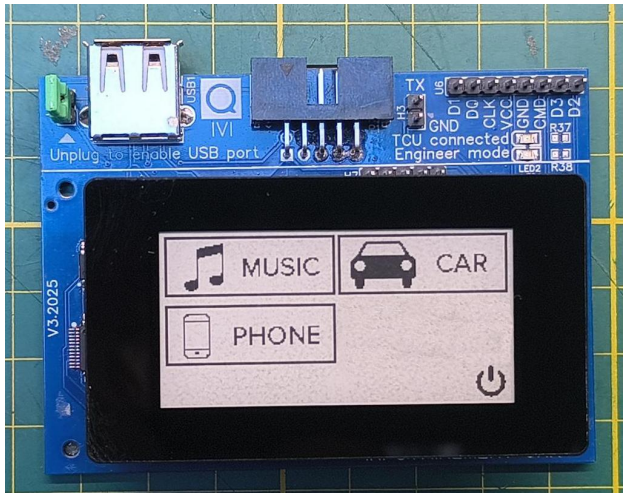
Lab 6 - Body Control Module

Goals

- For the **Body Control Module (BCM)** we will focus on its **CAN messages** and also find the Arbitration ID to send **UDS** requests
- Connect the **Instrument Cluster** to the **Body Control Module** and power it using a USB-C cable
- Complete challenges **Test bench - Body Control Module**

In-Vehicle Infotainment

- Using a 10-pin ribbon cable, connect the **In-Vehicle Infotainment (IVI)**
- Its **USB plug** can be used when **removing the jumper**, which disables its 10BASE-T1S connectivity
- The **ePaper screen** has a **touch capability**, to open various menus



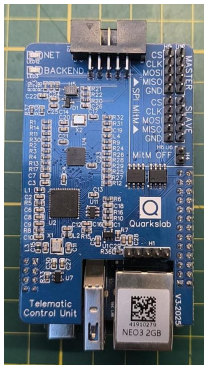
Lab 7 - In-Vehicle Infotainement unit

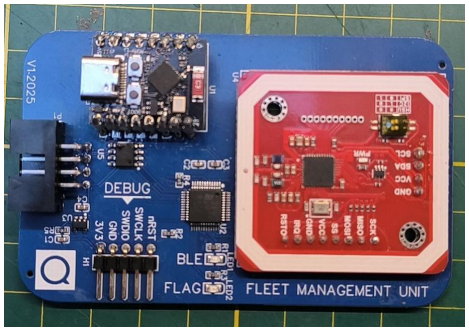
Goals

- We will focus on the Automotive Ethernet connectivity of the **IVI**
- Connect the **IVI** to one of the two **10-pins IDC connector**
- Complete challenges **Test bench - Infotainment**

Telematic Control Module & Fleet Management Unit

- The last two ECUs to add to our bench are the **Telematic Control Unit** and the **Fleet Management Unit**
- The **Telematic Control Unit** will handles connectivity to an emulated **LTE network** using ZMQ protocol
- The **Fleet Management Unit** has **RFID** and **Bluetooth Low Energy** capabilities to allow car sharing





Lab 8 - Telematic Control Unit diagnostic access

Goals

- Assemble all the parts of your test bench
- Wire the **IVI** and its **display**
- Connect the **USB port** and check if it works by inserting a **USB drive**
- Complete the **Test bench - Telematic** challenges

Thank you

Contact information:

Email: contact@quarkslab.com

Phone: +33 1 58 30 81 51

Website: www.quarkslab.com

@quarkslab

Quarkslab