TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Trevor Kallaste 192509IASM

VEHICLE ENGINE CONTROL AND DIAGNOSTICS SOFTWARE

Master's thesis

Supervisor: Vladimir Viies
Ph. D

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Trevor Kallaste 192509IASM

SÕIDUKITE MOOTORITE KONTROLLI JA DIAGNOSTIKA TARKVARA

Magistritöö

Juhendaja: Vladimir Viies
Dotsent

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis and this thesis has not been presented for examination or submitted for defense anywhere else. All used materials, references to the literature and work of others have been cited.

Author: Trevor Kallaste

10.05.2021

TALLINN UNIVERSITY OF TECHNOLOGY

MASTER`S THESIS TASK SPECIFICATION

Date: 20.11.2020

Student name: Trevor Kallaste
Student code: 192509IASM

Topic: Vehicle engine control and diagnostics software
Sõidukite mootorite kontrolli ja diagnostika tarkvara

Topic background: Bachelor's thesis "futureworks"

Supervisor: Vladimir Viies
Co-supervisor:
Advisor: Ivar Kallaste USParts Pärnu 55 35 666

Additional specifications: Software using OBD2 to access vehicle ECU during diagnostics

Issues to be resolved according to the following learning outcomes:

A) System aspects: DB connection with diagnostics tool
B) Software aspects: Diagnostics software using OBD KW 1281 protocol
C) Hardware aspects: Microcontroller

Exceptional conditions: Working prototype with manual

Student's signature:
(Digitally signed)

Abstract

The objective of given thesis is to study different control system interfaces, their differences and also give an accurate overview about car diagnostics, specifically on car diagnostics. Prime focus was on developing a diagnostics system in Java for older Volkswagen Group vehicles.

Main problem that occurred was the lack of proper documentation on diagnostics protocol used for those vehicles and the deviations from the developed standard.

The work has produced a working system which consists of a program to connect to the cars ECU and a server configuration that allows the application to communicate with the server and display information to the user on any device. The cars that are supported are VAG vehicles which were manufactured from the mid 1990-s to early 2000-s.

The thesis is in English and contains 74 pages of text, 3 chapters and 34 figures.

Annotatsioon

Töö eesmärk oli uurida juhtsüsteemide liideseid, anda ülevaade diagnostikast, täpsemalt autode diagnostikast. Peamine fookus oli diagnostika süsteemi loomisel, mis koosneb Java programmist Windowsi operatsioonisüsteemile ja Node-Red serverist. Kaasaegsed autod sisaldavad tohutul hulgal elektroonikasüsteeme. Kõigis neis süsteemides võib esineda vigu, mida tuleb parandada, et säilitada sujuv sõidukogemus. Võimalus juhti probleemist teavitada võib vähendada võimalust, et viga muutub aja jooksul kriitilisemaks. Lisaks on hea teada, millises alamsüsteemis viga on leitud, ja kontrollida ka teisi süsteeme, et veenduda, et need on veavabad.

On olemas palju erinevaid diagnostikavahendeid, mille kontrollimiseks on palju erinevaid rakendusi. Need rakendused võivad olla arvutitele või mobiiltelefonidele. Lisaks on palju skaneerimisvahendeid, mis ei vaja teabe kuvamiseks lisaseadet. Kõik need süsteemid võivad olla kas lihtsad, mis on tehtud tavakasutajatele, et lihtsalt kontrollida diagnostikaandmeid vigade leidmiseks, või need võivad olla professionaalsed tööriistad, mida on vaja põhjalikuks diagnostikaks. Siiski on üsna võimatu kasutada mobiiltelefoni, et ühendada vanemate autode diagnostikasüsteemiga, sest need nõuavad väga spetsiifilist riistvara, mille kasutamiseks on vaja väga spetsiifilist tarkvara ja need on kõik Windowsi jaoks. Lisaks sellele, et enamik neist süsteemidest on väga kallid, on nende funktsioonid suunatud uuematele autodele ja ei pruugi vanade diagnostikaprotokollidega toime tulla.

Töö jooksul on loodud süsteem, mis pakub funktsionaalsust, et ühendada Volkswagen Grupi autodega, mis on toodetud 1990-ndate lõpust kuni 2000-ndate alguseni. Eesmärgiks oli luua kasutajaliides, mida on tuttav kasutada neile, kes juba kasutavad mõnda autode diagnostikasüsteemi.

Peamine probleem tekkis diagnostika protokollist arusaamisel, sest dokumetatsioon on puudulik.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 74 leheküljel, 3 peatükki ning 34 joonist.

List of abbreviations and terms

ECU	Electronic Control Unit, embedded system used in cars to control electrical systems. Can only be used for Engine Control Unit, which is an ECU that control’s engine to ensure optimal performance.
BCM	Body Control Module, an ECU that control comfortability functions like electronic windows, air conditioning and also central locking.
Volkswagen	German car manufacturer.
Bosch	Electronics company, many ECUs are manufactured by Bosch.
ABS	Anti-Block System, helps to keep wheels from locking during heavy braking.
Immobilizer	Security system in most cars that can detect if the vehicle was started without the correct key.
TCM	Transmission Control Module, ECU which controls automatic transmission gearbox.
PCM	Powertrain Control Module, ECU which is found in some cars, has the functions of both Engine Control Moule and TCM.
Cruise Control	System found in many cars that lets the driver set a desired speed and the vehicle maintains it without the need from the user to use throttle pedal.
SCU	Speed Control Unit, ECU which controls the Cruise Control system.
DCU	Door Control Unit, ECU which is found is some cars, used to control all the electronics inside doors.
CAN	Controller Area Network bus, a microcontroller bus used in vehicles.
Mercedes-Benz	A German car manufacturer.
VAG-COM	A diagnostics system used for Volkswagen Group cars.
OBD	On-board Diagnostics, an electronics self-diagnostics system found in vehicles.
OBD-2	Standard version of OBD, used widely since the 1990-s

RPM	Revolutions per minute, standard of measurement for internal combustion engine rotation speed.
CPU	Central Processing Unit, the core of any computer, executes instructions of given programs.
GM	General Motors, an American car manufacturer.
Raspberry Pi	A series of single -board computers designed to teach computer science and create automatic systems.
RAM	Random Access Memory, computer memory that is used to store working data a code.
Debian	An open-source Linux distribution.
Pi OS	Debian create to use on Raspberry Pi devices.
Wi-Fi	A wireless network protocol used for local area networks and Internet access.
Bluetooth	Short-range wireless technology to exchange data between two devices over short distances.
Node-Red	Programming tool for connecting multiple devices using flows.
IoT	Internet of Things, system of connected computing devices that transfer data over network without the need of human interaction,
JavaScript	High-end programming language. Used mostly for web programming.
MQTT	Message Queuing Telemetry Transport, lightweight publish-subscribe network protocol to transport messages between devices.
Arduino	series of single-board microcontrollers to build digital devices.
ESP32	Series of affordable, low power microcontrollers with integrated Wi- Fi and Bluetooth capabilities.
DPF	Diesel Particulate Filter, device which removes harmful particles and soot from the exhaust gas of a diesel engine.
Datsun	A Japanese car manufacturer.
EGR	Exhaust gas recirculation, system to reduce exhausts of diesel and petrol engines

LED	Light-emitting diode, diode which turns electricity to light
M1	Type of vehicle with at least 4 wheels, constructed for passenger transport, with no more than 8 passenger seats.
USB	Standard for connecting devices to computer.
Torque	An Android app for car diagnostics.
Ross-Tech	Company which develops VAG-COM software and diagnostics cables.
VCDS	Developed by Ross-Tech, it is a diagnostics system for VAG vehicles. Also knows as VAG-Com.
RS-232	Standard used for Serial Port connections.
PC	Personal Computer, usually with Windows operating system.
ISO 9141-2	Base protocol for early diagnostics, describes the requirements for setting up connection between ECU and scan tools.
Windows	Operating system for personal computers.
ASCII	American Standard Code for Information Interchange, it is a character encoding standard used for electronic communication. Each character is represented by a number from 0 to 127.
group reading	In diagnostics, group reading is a process of requesting data from ECU and receiving it for up to 4 sensors, depending on the chosen group.
complement	In diagnostics for KW-1281 protocol, complement is used to check that transferred data reached its destination without errors. All bytes are sent back to the sender by subtracting its value from 0xFF. This result is called the **complement.**
intake manifold	A part of the engine that supplies the fuel/air mix to the engine.
VAG-KKL cable	Type of diagnostics cable, used before CAN bus era. Can be used for most OBD-2 functions in VAG vehicles.
Java	A class-based, object- oriented programming language.
DTC	Diagnostics Trouble Code, these are the error codes stored in the ECU.
ALLDATA	A database for cars equipment, ALLDATA provides vehicle manufacturers’ diagnostics, repair and spare parts information.

Audi	A German car manufacturer, part of Volkswagen Group.
Seat	A Spanish car manufacturer, part of Volkswagen Group.

Table of Contents

List of Figures

Introduction

Modern cars have huge amounts of electronic components. Some help optimize the engines work and reduce pollution and fuel consumption. Some others are many kinds of driver assist systems, like cruise control. And finally, some are entertainment systems like radio. All of these systems can experience faults, which have to be fixed to sustain smooth driving experience. The ability to notify the driver of the problem can reduce the chance of the error becoming more critical overtime. Additionally, it is good to know that in which subsystem the error is found in and also to check other systems to make sure they are error free. This can make the driver more relaxed while driving.

There are many different diagnostics tools, which have many different apps to control them. These apps can be for PCs or mobile phones. Additionally, there are many scanning tools, that do not need extra device to display information. All of these systems can be either easy, made for regular users to just check the diagnostics data for errors, or they can be professional tools which are needed for comprehensive diagnostics. However, a possibility to use mobile to connect to the diagnostics system of older cars is pretty much impossible, as they require very specific hardware, which requires very specific software to use and they are all for Windows. And not only are most of these systems very expensive, their functionalities are aimed towards newer cars and can struggle with older diagnostics protocols.

The aim of the thesis is to create a system that only needs affordable hardware and cross-platform software to have access to all diagnostics functionalities of Volkswagen Group cars that are manufactured from mid-1990-s to early 2000-s. The goal is to have user interface that is familiar to use for those already using some diagnostics system.

To achieve these goals, a diagnostics protocol from that era is researched.

The theory part is somewhat based on the authors experience in this field and additionally, various Internet sources are used that focus on these subjects. Some of the work is taken from the authors Bachelor thesis, including the base program for the developed system.

The work is divided into 3 chapters. The first one is about control system interfaces, specifically the ones used on the final system. The second chapter gives an overview of diagnostics and is focused on car diagnostics. The third chapter documents the created diagnostics system.

1 Control system interfaces

A control system consists of multiple subsystems with the main function to create the connection between the main device and the user. The control system interfaces are input devices or interfaces, like touchscreen, keyboard, mouse, joystick or various buttons and levers. Most of them process data both to the user and from the user. Most of them usually represent data visually, but some can also make sounds.

1.1 Engine Control Unit

All modern cars have hundreds of electronic components that are needed to control the vehicle’s driving. In addition, there are also many electronics needed to make the ride comfortable, for example: music system, air conditioning, electronic windows.

The electronics that are used to control the driving have one central control unit, called Engine Control Unit or ECU for short. The one controlling the rest of the systems is Body Control Module (BCM). Every Control Unit is essentially an Electronic Control Unit which are also called ECUs and this can cause some misunderstanding. In pre-2000-year vehicles, there was only one ECU in most cars, but with more modern cars there became more and more ECUs and the Engine Control Unit is usually referred as main ECU or engine ECU. [1]

While BCM is only found in vehicles made in this century and end of last century, ECU dates back to the 1970-s.

1.1.1 ECU

Figure 1. Volkswagen ECU by Bosch [2]

ECU (figure 1) is a computer that is partly pre-programmed by the manufacturer and also to some degree programmable. Usually to program the ECU, you need a device that is accessible to the manufacturer only.

The first need for ECUs came with requirements to make cars more economic and less polluting. They took control of how fuel was pumped into the engine to reduce waste.

Over time the ECUs also took control of diagnostics. The early versions could only show lights on your dashboard (the most common is still added to modern cars – it is the check engine light). [3]

These days as cars are a lot more sophisticated, the ECUs control almost every part of the engine's work and also the brakes, quality of life improvements like cruise control and their diagnostics capabilities are much better. Now they can show you human readable text of errors and telemetry on many different devices.

In more expensive vehicles, with a lot of comfortability functions like active suspension for a smooth ride or automatic transmission, one ECU is not enough and so the manufacturers have started adding multiple ECUs with different functions. There is a separate module for automatic transmissions. Then for some cars there is a separate module for each door to handle the electronic windows and mirror adjustments. All cars are required to have a module which handles the ABS system and in case of modern cars, this module is also responsible for calculating and displaying the vehicle's speed. [4]

1.1.2 ECU functions

As described earlier – the ECU is used as a central unit to control the vehicles electronics. They do so in the same way every other controller – by reading input, doing the necessary calculations and sending data to the outputs. The inputs are different sensors on your vehicle – for example if the speed-o-meter is showing wrong speed, it could be one of the ABS sensors giving false data. Every wheel's speed is sent to the ECU and in a straight line they are ought to be the same, if there is a discrepancy detected, the user is notified by the ECU. [5]

1.1.3 BCM

In addition to electronics controlling our vehicle's driving abilities, the rest of the important aspects of the car like comfortability and security – are also controlled by electronics. These components, which are not linked directly to the driving are controlled by the Body Control Module (BCM). [6]

In older cars, the functions like lights, windshield wipers, seat adjustments were usually directly connected to the battery and controlled by the driver – using a switch or a button to give power to them. And talking about security – most cars had the simple door unlocking and engine ignition by turning the key. For those who were interested, the ability to add a remote to central locking was also available. This system was relatively easy and the only part controlled by a computer was the immobilizer (if the received signal did not match the code stored in the ECU, then the engine would cut off).

These days however, all those mentioned functions and many new ones are controlled by the BCM. For cars that have onboard computer for checking of for example - fuel consumption or tire pressure, this info is provided by the BCM. [6]

1.1.4 Additional ECUs

Modern cars can have up to 80 different ECUs. To keep things simpler, in the context of modern cars, the Engine Control Unit, is called the “main ECU” ja Electronic Control Units are collectively referred as ECUs. [7]

As more and more electronics were added to cars, one ECU became too weak to handle it all. To create a system that is easier to handle, cars started to have more different ECUs. The main ECU was left with diagnostics and overall control as the main computer, but other ECUs were made to have control over very specific systems and report data back to main ECU.

Already in the 1990-s, when cars were fitted with ABS (anti-block system, designed to help with braking), the ABS had its own ECU and today the whole braking system is controlled by Brake Control Module, which also takes over the functions of handbrake (used to be a manual lever, but on modern cars it is just a button in your dashboard or in case of automatic vehicles, controlled automatically).

Additionally, there are ECUs for transmission (called Transmission Control Module or TCM) – fitted to cars that have automatic gearbox. This module controls all the gear changes in fully automatic gearboxes and in case of semi-automatic transmission it controls the clutch. The data for when to switch gears comes from sensors in gearbox, engine and also from the main ECU. Sometimes Engine Control Unit and TCM are combined into one single ECU, called Powertrain Control Module (PCM). If the vehicle is fitted with Cruise Control, then it is usually controlled by either the main ECU (or sometimes TCM) in older cars or by its own separate ECU, called Speed Control Unit (SCU). [5]

Furthermore, there is Door Control Unit (DCU), which is a minor ECU and controls all the electronic accessories inside the car door. Some vehicles can have a DCU per each door. The functionalities provided by DCU are usually: electronic windows movements, child-lock, also mirror adjustments and folding. [8]

1.1.5 CAN

A Controller Area Network (CAN bus) is a standard designed to allow microcontrollers and devices to communicate with each other’s applications without the need for host computer. It is usually found used in vehicles. Its protocol is message-based and designed for saving copper by requiring less wires (signals from different devices are sent on the same wire). Collisions are minimized by having priority list of devices and only the device with highest priority can transmit, while others back off. Every device receives the message and will discard it, if the message was not meant for them. This makes CAN bus efficient as the top priority data gets immediate bus access. [9]

CAN bus was developed in 1983 and first production car to feature CAN-based wiring system was Mercedes-Benz W140 in 1991. [10]

CAN bus is used to connect different ECUs in vehicles. In the past, when the amount of electronics in a car was low, it was no problem to have a separate wire between every device. These days however, the amount of electronics has grown so much, that all the wiring could not possibly fit. However, with CAN, all ECUs can be connected into one system, which needs a lot less wires. To accommodate such increase of data on one wire, the speeds of transmit have also increased. In the 1990-s, signals speed between ECU and diagnostics devices were between around 9600 bit/s. These days, the baudrate can reach 5 Mbit/s [11].

CAN bus is designed to be fully centralized. To keep diagnostics accessible as they were before CAN, all ECUs will communicate with the Engine Control Module, which is the only ECU needed to connect with diagnostics device and this enables access to all vehicle systems from a single diagnostics port. [9]

CAN standard consists of multiple different types of network:

1) High speed CAN bus: This is where most of the data is sent over, as it is the fastest CAN bus. This is found in almost every vehicle these days. Car diagnostics protocol use High speed CAN as their basis. [12]

2) Low speed CAN bus: This is slower than High speed CAN and can reach up to 125 Kbit/s baudrates. It is used for fault tolerance and is used to keep up the communication even when there is a fault detected. [12]

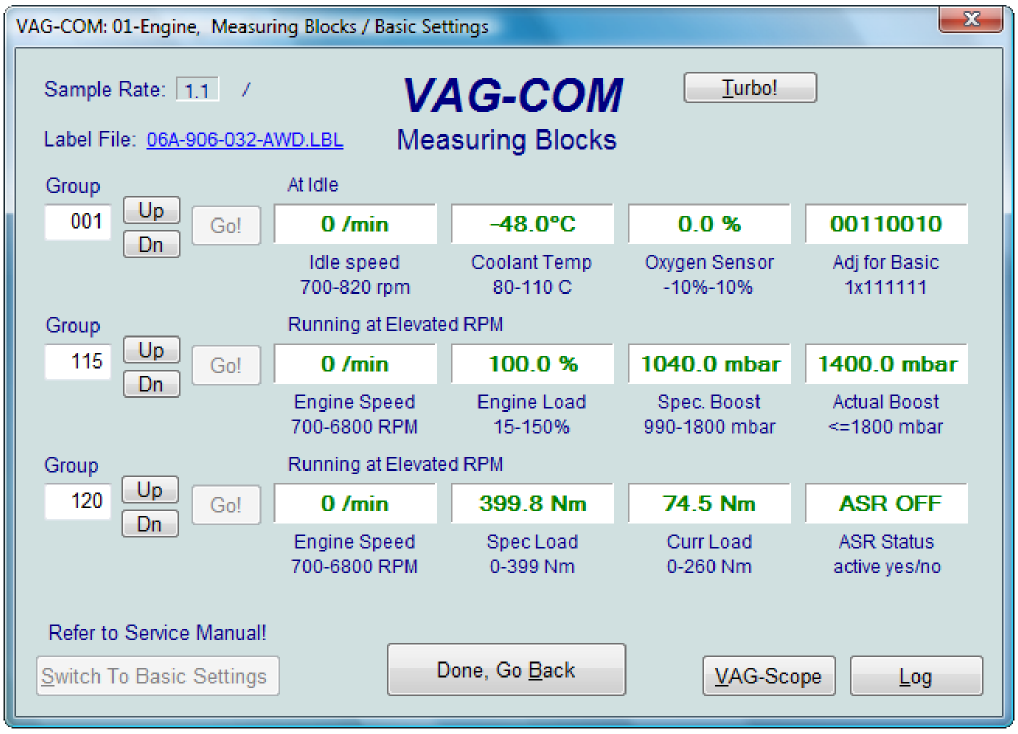
3) With modern cars, even High-speed CAN is too slow and Automotive ethernet is being rolled out fast. It supports the high bandwidth needed with Advanced Driver Assistance Systems and the quickly increasing number of cameras in all modern vehicles. Automotive ethernet while being much faster than High speed CAN, lacks some of the safety features of CAN and is therefore most likely to stay as a supplement to CAN rather than remove it in the coming years. [12]

4) With the increase of autonomous vehicles and more data being stored in cloud, CAN bus must keep getting faster and more secure to stay relevant. Faster speeds (up to 8 Mbit/s data rate) and improved security by requiring authentication is offered by CAN FD (Flexible Data Rate). This new CAN bus is already fitted to newest vehicles. [12]

1.1.6 VAG-COM

Once cars with OBD-2 became widespread in the end of the 1990-s, car mechanics and enthusiasts wanted to have access to their vehicle’s diagnostics, without the need to visit an official mechanic. However, most diagnostics systems on the market were either very expensive or had only few functionalities. The solution to this, for VAG cars is VAG-COM (figure 2).

First released in 2000, it was the first affordable diagnostics tool for VAG cars that had all the required functionalities: This tool had access to all diagnostics data and error codes. Additionally, it could delete error codes. What made VAG-COM truly great, were its functionalities that even factory tools do not have – for example, it can log data in real time and show it to the user as a graph. VAG-COM was the first diagnostics tool which had CAN support for Volkswagen/Audi cars. [13]


VAG-COM: 01-Engine, Measuring Blocks / Basic Settings

Sample Rate: 1.1 /

Label File: 06A-906-032-AWD.LBL

VAG-COM
Measuring Blocks

Group 001 — At Idle

0 /min	-48.0°C	0.0 %	00110010
Idle speed 700-820 rpm	Coolant Temp 80-110 C	Oxygen Sensor -10%-10%	Adj for Basic 1x111111

Group 115 — Running at Elevated RPM

0 /min	100.0 %	1040.0 mbar	1400.0 mbar
Engine Speed 700-6800 RPM	Engine Load 15-150%	Spec. Boost 990-1800 mbar	Actual Boost <=1800 mbar

Group 120 — Running at Elevated RPM

0 /min	399.8 Nm	74.5 Nm	ASR OFF
Engine Speed 700-6800 RPM	Spec Load 0-399 Nm	Curr Load 0-260 Nm	ASR Status active yes/no

Refer to Service Manual!

Figure 2. Reading data from different sensors using VAG-COM. It can be seen that the engine revolutions are at 0 (the car is not running). Also, an error while parsing data as coolant temperature is reported as -48 degrees. Some cars do not have all the sensors that the diagnostics program might expect them to have and can report wrong data because of it. Mechanics can compare the Spec. Boost value to the Actual Boost value and if they are too different, then it usually points to an engine problem. [14]

1.2 Raspberry Pi

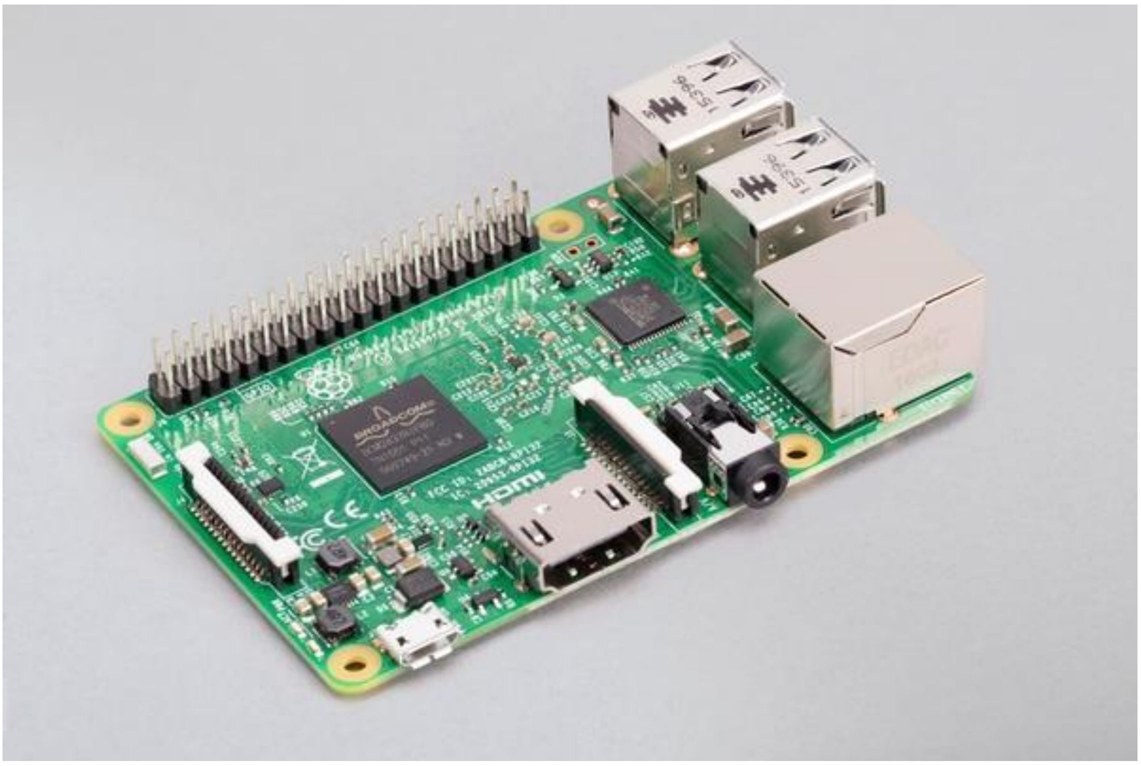


Figure 3. Raspberry Pi 3 model B [15]

Raspberry Pi (figure 3) is a series of single-board computers made for educational business. The latest model has quad-core CPU and 4GB of RAM. Their price is usually around 30-40 EUR. They are used from learning to industrial projects. In the context of control systems, Raspberry Pi devices can be used for home automation and small servers. The Raspberry Pi operates in open source ecosystem – it runs a variety of Linux distributions, mainly Debian (official version made for Raspberry Pi is called Pi OS, which is open source and uses open source software). The board itself is however not open hardware, but the schematics are released as documentation. [16]

For usage in cars, the Raspberry Pi comes with a set of input/output pins that can be programmed to control electronic components. Additionally, vehicle diagnostic systems can access the Raspberry Pi through for example, Wi-Fi or Bluetooth. This gives the user the ability to create a home server and send the vehicle telemetric data for storage and analysis.

1.2.1 Node-Red

Node-Red is a low code environment that provides flow editor to wire together different commands with the purpose of making different device talk to each other (creating an IoT network). [17]

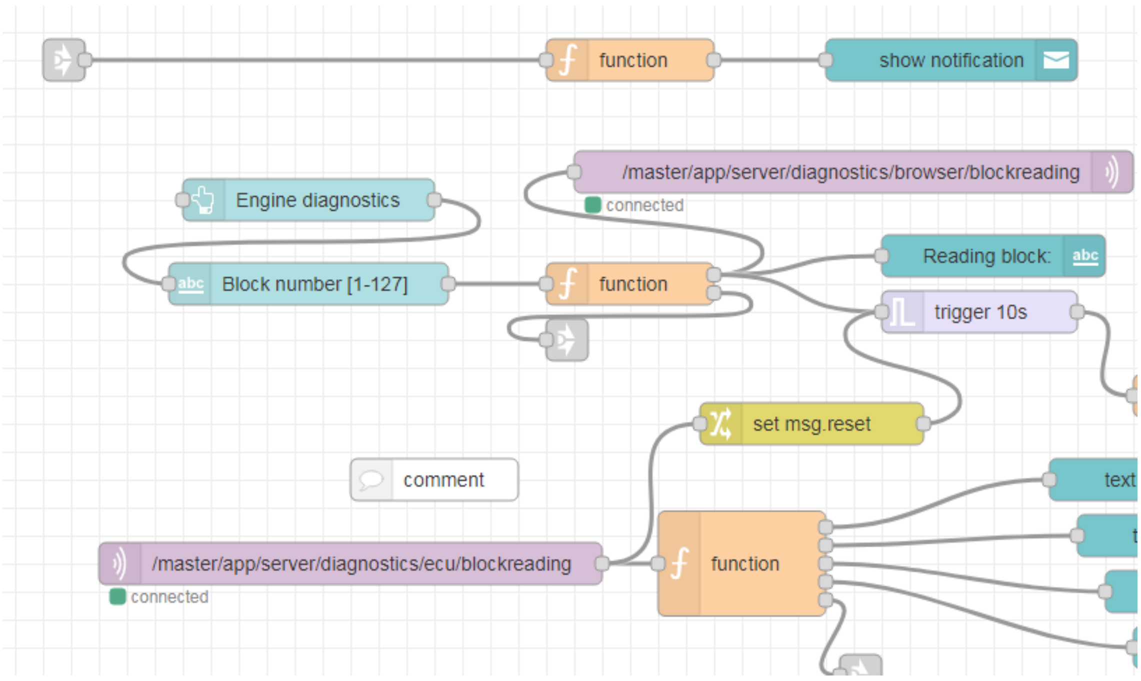


Figure 4. Node-Red Flows

Most of the work is done by dragging together different flows (figure 4) and some code can be written in JavaScript. Code is mostly needed for data parsing; all the networking is done for the user by Node-Red automatically.

Node-Red is added by default in your Pi OS and can be used to make your own IoT network quickly and easily.

For vehicle control systems, Node-Red can be used to connect to the diagnostics system to read vehicles telemetry and store data in the cloud. Additionally, you can parse diagnostics data right on your Raspberry PI and print the data in human readable way to the user.

1.2.2 MQTT

MQTT is a networking protocol which is designed for machine to machine application. The protocol works by having one device publish to certain topics and other devices that are subscribed to that topic can read the message. MQTT was designed in 1999, but has become one of the main protocols for IoT development. [18]

MQTT is built in Node-Red and is very easy to use for the main protocol to send data between Raspberry Pi server and other devices (for example car diagnostics device).

1.3 Arduino

Arduino is another open-source platform. It consists of microcontroller (figure 5) and software. The software is used to program the microcontroller. Just like the Raspberry Pi, Arduino can be used for learning and also to create different IoT systems.

Figure 5. Arduino UNO, the most widespread Arduino microcontroller [19]

1.3.1 ESP32

ESP32 is a family of microcontrollers (figure 6), which similarly to Arduinos are easy to use for different IoT systems. They can be programmed with the same software as Arduinos and the same codes can usually be used on both devices with minimal changes.

What makes ESP32 more powerful that Arduino is that the ESP32 has by default Bluetooth and Wi-Fi access, which makes it important in today’s wireless world. [20]

In vehicle diagnostics, ESP32 chip can be used to connect with the vehicle ECU through diagnostics port and send all the gathered data to the server. Only extra device needed is the connector to connect the ESP32 to the vehicle’s diagnostic port.

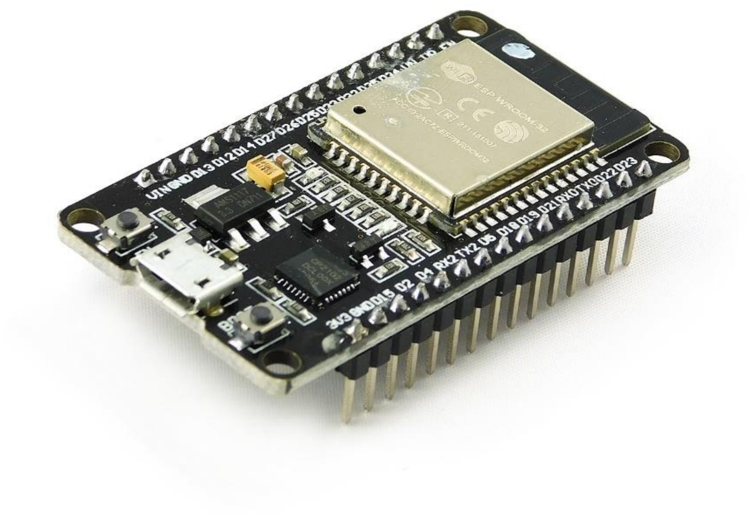


Figure 6. ESP32 microcontroller [21]

2 Diagnostics

Diagnostics is a set of functions that detect, isolate and correct malfunctions in the desired system [22]. Compiling and maintaining error logs is also part of diagnostics. In more sophisticated systems, diagnostics system is also part of carrying out the actions that are necessary to be taken on error detection.

2.1 Fault detection

Fault detection and diagnosis for is one of the important aspects in improving reliability of practical control systems [22]. Fault detection is more than just comparing values to their threshold values. It is about alerting the person responsible, about the problem and the nature of the problem. Instead of having specialists deal with the problem when it happens, technology allows to simulate every problem and create an algorithm which lets us know, what exactly is the problem and what could be causing it. [23]

Not every fault is a complete system shutdown, but can be just a reduction in performance. There are usually 2 different kinds of faults – binary ("OK" or "not OK") or a numerical value. Most fault detection systems compare this value to the stored threshold value and will give an alarm when the fault is detected. [23]

2.1.1 Alarm

The function of an alarm has always been to alert people of a fault. Usually there have been two options: either fix the fault or escape (in case of life-threatening situations). In automotive world, most faults are not life threatening, but when left unfixed, they can lead to dangerous situations (for example – car brakes failing while driving due to negligence or failing to maintain them regularly). To avoid that, all modern vehicles are filled with sensors to detect errors. However, the driver must be alerted of the problem to make sure the fault is fixed.

The alarms used in fault detection for automotive world are similar to these used elsewhere. For example, when there is a smaller fault that should be fixed soon, but is usually not critical and allows you to keep driving safely, they are displayed with yellow lights. An example for this is the diesel particle filter (DPF) light (figure 7). When the DPF gets clogged, it has an effect of performance and fuel economy. However, because it is not a critical problem, it is displayed with yellow light.

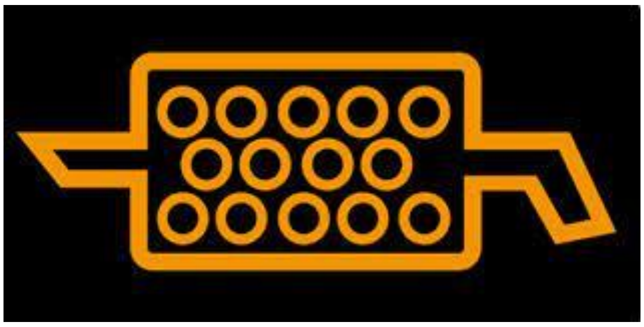


Figure 7. DPF light [24]

However, when there is a problem with brakes, the user gets a brake warning light (figure 8). Brakes are a critical system and when any fault is detected, a red light is shown to the user to fix the problem immediately. It is mostly forbidden to keep driving when a red light is shown in the car’s dashboard.

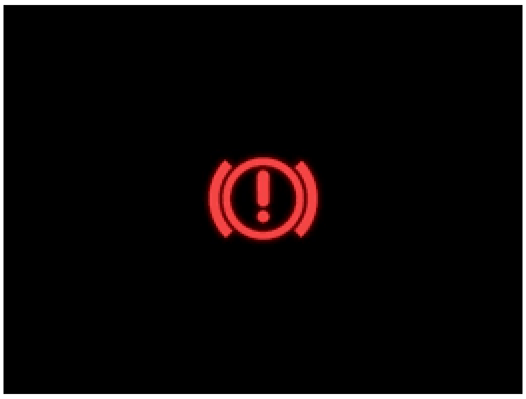


Figure 8. Brake warning light [25]

Another type of alarms know to people are sound alarms. Usually they are loud to get attention. This kind of alarms are mostly reserved for problems that require everyone around to stop whatever they were doing and deal with the fault. When there is a threat to human life, most alarms are made to produce sound, this is because a loud sound is a great way to get the human attention and also because one speaker can alert a large area and travel through walls.

In automotive world, the first alarm sounds were for alerting people that a vehicle was approaching. First there was a man walking before the vehicle with a red flag and a horn, but the horn was soon fitted to the vehicle [26]. In case of faults, alarm sounds were introduced to widespread cars around the 2000s. Earlier, it was mostly lights without sound that pointed to faults. In some modern cars, for example a short honk is played to alert the driver when there is -4 degrees air temperature outside and the road can be slippery. A regular alarm sound is played in some vehicles to alert the driver that the wiper fluid amount is under the threshold and should be filled. Both of these two alarms are usually accompanied with a light (figure 9) on the dashboard.

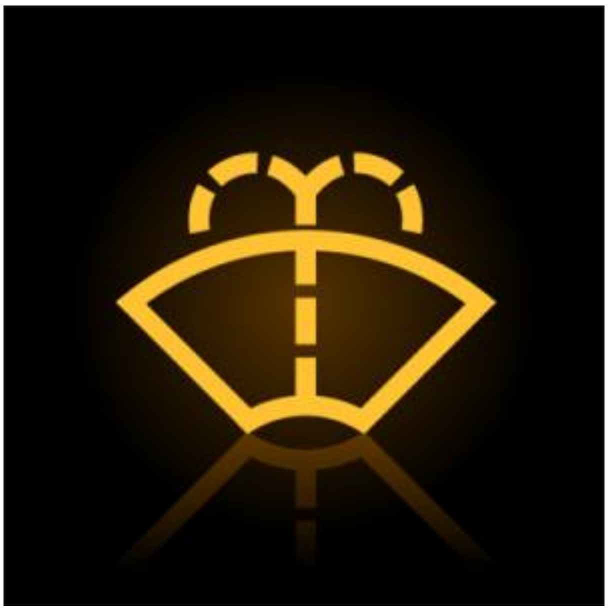


Figure 9. Low wiper fluid light. Sometimes this light is accompanied with a short alarm sound. [22]

2.1.2 Fixing and clearing errors

When a problem is detected then the next step is to eliminate it. A logical and systematic search is used to find the source of the fault. The most likely cause for a problem is found in a process of elimination – all possible causes are eliminated one by one. When all causes are eliminated, a confirmation that the system is working is needed to finish the fixing process.

2.2 Car diagnostics

All the info and figures in chapter 2.2 and its subchapters come from my Bachelor's thesis - **Android Application for Usage in Car Diagnostics**. Link: https://digikogu.taltech.ee/en/Item/3bb0538d-ba98-4b39-9cba-fda39bd6a3f4

Automotive diagnostic systems provide the car owner or mechanic with information about the different subsystems of the vehicle.

2.2.1 History

The earliest diagnostic systems were indicators and gauges on the vehicle dashboard (figure 10). Most faults had to be detected by sounds or strange smells.

Figure 10. Oil pressure gauge.

The first diagnostic system for cars was added by Volkswagen to their Type 3 in the sixties (it was the first non-experimental car to use electronically controlled fuel injection system). In addition to controlling the fuel injection, it also had the on-board diagnostics ability. Ten years later, Datsun introduced the original fuel injector systems monitoring capability.

In 1980, GM started using a small light on the dashboard (figure 11) that flashed when an engine failure was detected. The number and duration of the flashes had to be noted and then the fault they indicated had to be looked up in the manual.

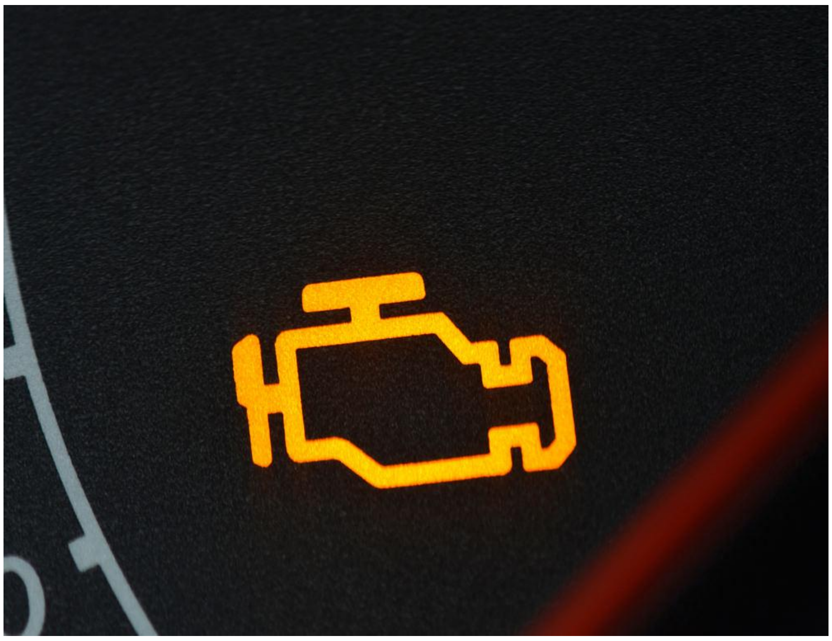


Figure 11. Check engine light lit in car dashboard

It's worth noting that up until then, each company had its own diagnostic system, there was no common standard, and so each car had to have a different adapter to communicate with the system.

The term OBD was coined in 1982 in California, when a system was created that could monitor air sensor, EGR system, car driver and fuel systems. At the time, it was useful for monitoring exhaust emissions but did not provide detailed information on other major vehicle systems. In 1991, California made it mandatory for every new car sold to have an

OBD system. In 1996, this requirement was extended to all parts of the United States. By 2001, OBD was already mandatory in Europe for petrol cars and by 2003 for diesel cars.

2.2.2 Interfaces

Over the years, different manufacturers have developed a number of different diagnostic interfaces for their vehicles, some of which had some standardization.

However, very few are in use today, and most are only in use because there are still cars on the road that use the old interfaces.

2.2.3 OBD-2

OBD-2 is an evolution of the OBD-1 protocol, which was the first standardized diagnostic interface, created in 1991 in California. The OBD-1 protocol was mostly not compatible with other car manufacturers and therefore required an expensive diagnostic system for each car manufacturer (figure 12). OBD-1 also included in-car diagnostic programs, which were switched on, for example, by holding down certain buttons on the dashboard. Different car manufacturers had different ways of displaying diagnostic data, for example Honda flashed LED lights on the dashboard in a specific pattern.

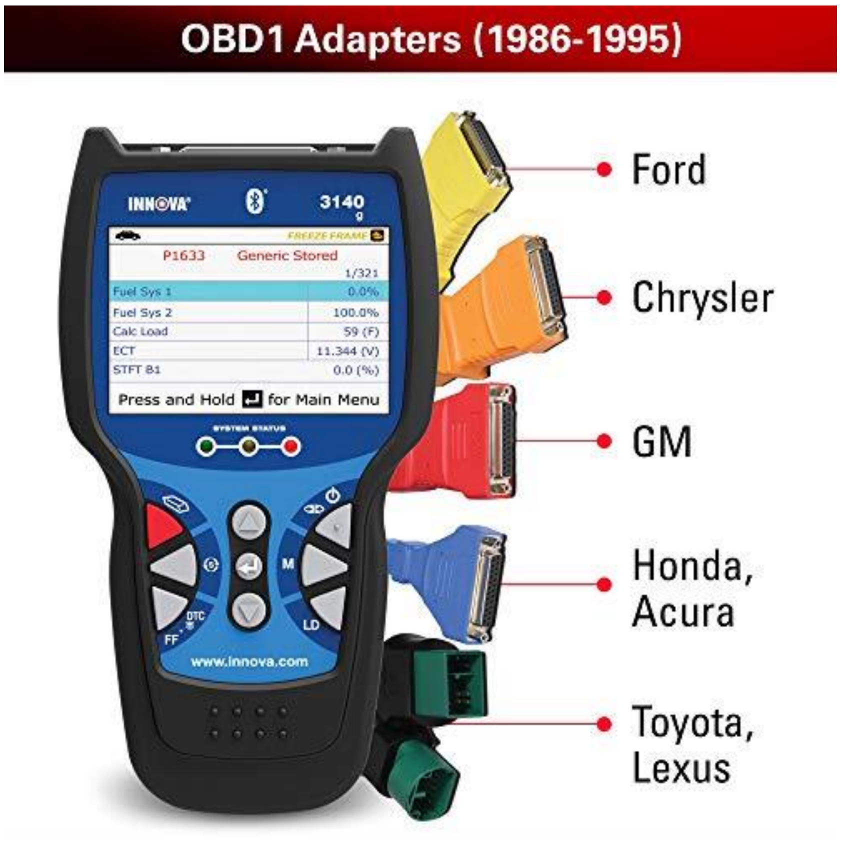


Figure 12. OBD-1 diagnostics tool which shows that different manufactures had different connection ports for diagnostics.

OBD-2 led to better standardization and more functionality. The type of connection, the protocol and the data transfer format have all been specified. In addition, there is a predefined list of parameters to monitor and how to encode the data. There is also a pin that powers the scan tool from the car battery, removing the need for a separate power source. A list of error codes is also provided as standard. Thanks to this, a single device can read the fault codes from any car equipped with OBD-2 diagnostics.

The need to simplify the increasingly complex field instrumentation was behind the creation of OBD-2. By law, only emissions data must be readable, but car manufacturers use OBD-2 to diagnose and program the entire vehicle's equipment.

The creation of OBD-2 has been of great benefit to the average user and hobby mechanics. Before, for any kind of diagnostics, you had to go to a professional who charged a high price for connecting the diagnostic equipment and identifying problems. Smaller businesses were unable to buy expensive equipment or acquire the technical skills to use it. With the advent of OBD-2, a host of affordable, user-friendly devices came on the market that almost anyone could use to diagnose problems.

Another major innovation that came with the creation of OBD-2 is the reduction of tuning options. Vehicle manufacturers are using programmable read-only memory for the driver. Before the introduction of OBD-2, they could be swapped/modified to improve engine performance (mostly at the expense of fuel consumption and not polluting the environment), but with the new standard the system is closed and swapping is, if not impossible, at least much more difficult. Tuners, on the other hand, will also benefit a little from the OBD-2 standard, as it will give them access to a large amount of vehicle data in real time, which will help in parameter selection.

There are two versions of the OBD-2 connector that are very similar, but one has a small plastic strip in the middle. The need for this arose because some vehicles use a 12V battery while others use a 24V battery. The lower voltage device must not be able to be inserted into a higher voltage vehicle. In every other way, however, the plugs are identical. Both have 16 pins (figure 13 shows the 12V version). Each manufacturer has been given a free hand to do what is done with almost any pin, the only limitations being the grounding (4th and 5th pins) and the power supply from the battery (16th pin). In addition to the predetermined limitations, there are 6 pins assigned at the start which indicate which protocol the vehicle is using (one of which is current, depending on the protocol the vehicle is using), these are pins 2, 6, 7, 10, 14 and 15. The rest is up to the manufacturer to decide.

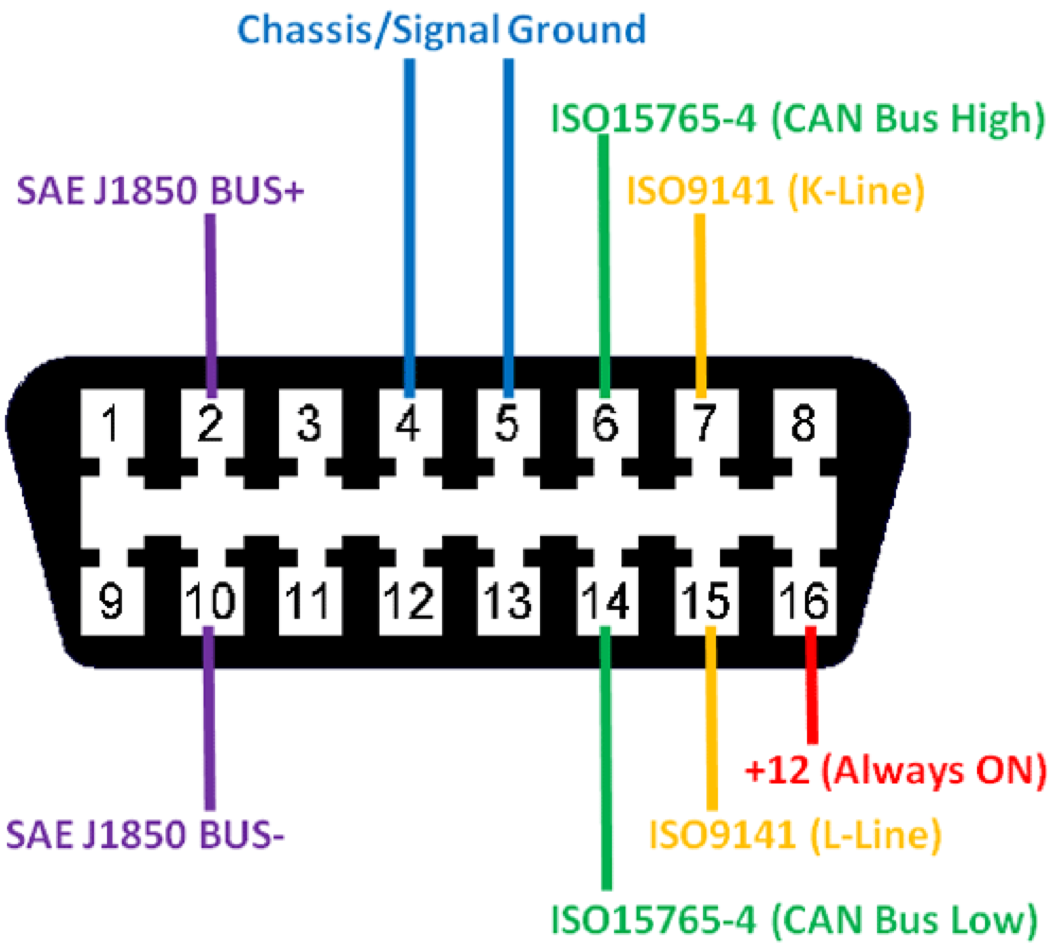


Figure 13. OBD-2 pinout with all default pins marked, the KW-1281 uses K-Line and also sometimes L-line. Also, every diagnostics device must always be connected to +12V and ground.

2.2.4 EOBD

The EOBD regulations are a European variant of OBD-2 and apply to all M1 category vehicles with up to 8 seats and a gross vehicle mass of less than 2 500 kg, first registered in a Member State of the European Union after the beginning of 2001 for petrol-engine vehicles and after the beginning of 2004 for diesel-engine vehicles. The law was adopted in 1998, and the EOBD regulation applies to all models created after the adoption of the Directive, one year earlier, i.e. at the beginning of 2000 for petrol cars and at the beginning of 2003 for diesel cars.

For vehicles with more than 8 seats or a gross vehicle weight exceeding 2 500kg, the law will apply to petrol cars in 2002 and to diesel cars in 2007. In 2017, all the other protocols were repealed, as 25 of them had been created in 38 years.

Japan, like the European Union, has introduced its own variant of OBD-2, called JOBD.

2.3 Scanning tools

For diagnostics, a large number of different devices have been made that are compatible with the OBD socket to access diagnostic functions. There are simple and inexpensive devices for the general user, high-end professional tools for dealerships provided by vehicle manufacturers, and devices for vehicle telematics (the science where the system takes information and does something useful with it).

2.3.1 Hand-held scanning devices

There are a number of different types of small hand-held scanning tools, from simple fault code readers (figure 14) and erasers for general users to professional tools, which in turn can be divided into several types. Most of these provide real-time access to vehicle diagnostic data, while the better ones can even check electrical connections for problems with the vehicle's electronics. The more expensive ones also include devices that offer suggestions on how to fix the faults found.

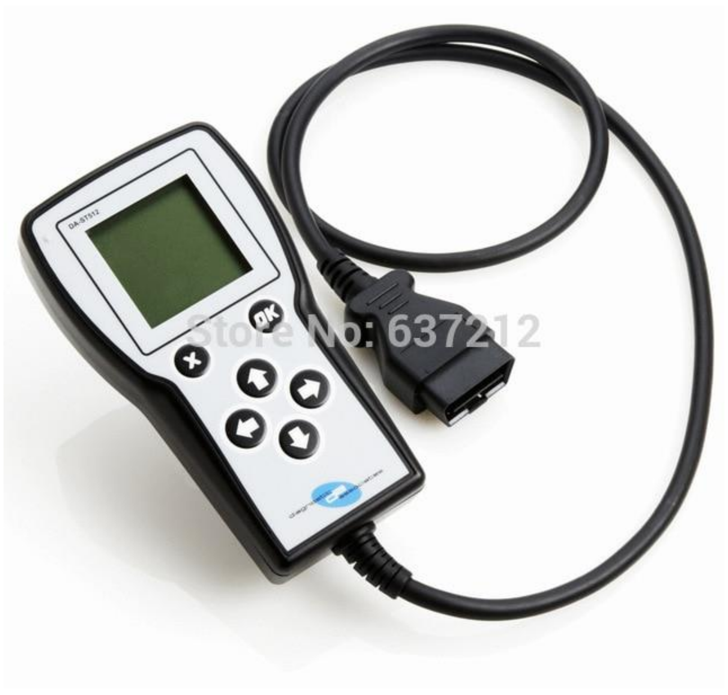


Figure 14. An example of a handheld scanning device. This one is cheap and can only show data / error codes. Other functionalities require a more expensive device.

2.3.2 Mobile apps

Mobile diagnostic apps use an OBD adapter connected via Bluetooth, Wi-Fi or USB cable to read (and, in better apps, manipulate) data from the vehicle's ECU. Mobile apps are convenient because they can be used on your mobile phone, which most people have these days. A very large number of mobile applications (figure 15 shows a widespread diagnostics app called Torque) have been developed with varying levels of functionality and cost, but most of them only work on newer cars due to the complexity of implementing the protocols used on older cars (lack of documentation).

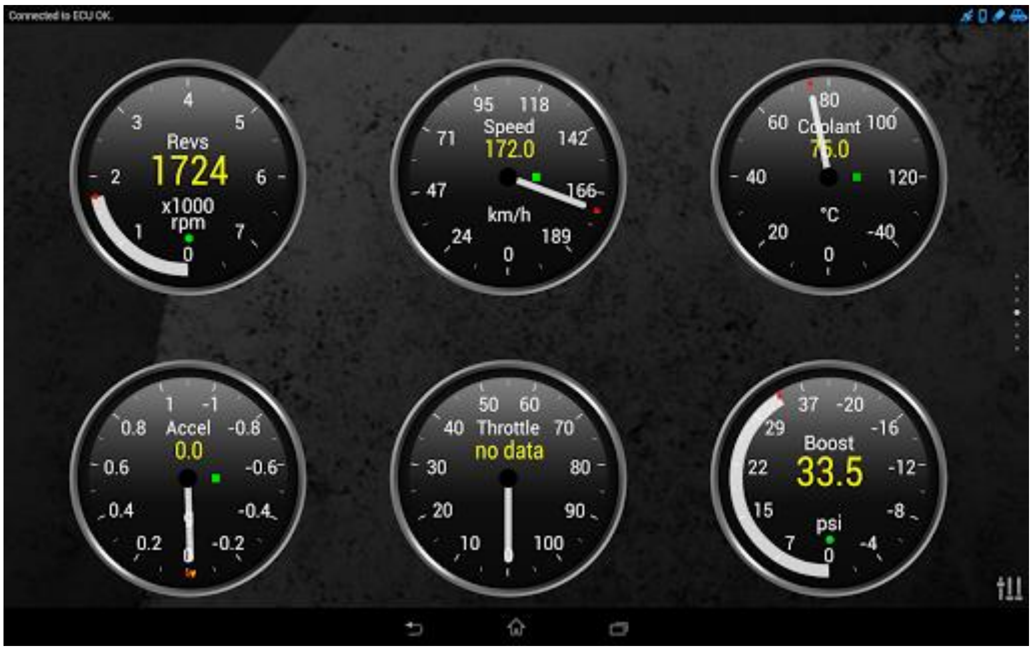


Figure 15. Screenshot of a mobile app called Torque. This app allows you to connect to your car with your phone using a Bluetooth OBD-2 diagnostics device. It sends data to your phone and the app displays the car's engine revolutions, car speed and coolant temperature. It can also be seen that the app failed to receive the throttle position. This could be because the diagnostics protocol the vehicle uses in not fully supported by the diagnostics device or the app itself.

2.3.3 Applications for personal computers

Desktop diagnostic applications convert OBD-2 signals into standardized USB or serial port data. Compared to hand-held scanning tools, desktop diagnostic applications have more features, such as: the ability to store large amounts of data, the possibility to use higher resolution screens (able to display much more detailed information) and the ability to run multiple applications at the same time, adding flexibility. Well-known desktop diagnostic applications are for example: Ross-Tech VAG-COM VCDS (works only on Volkswagen Group cars) which was written about in the first chapter.

2.3.4 ELM interface

The ELM is a microcontroller programmed by ELM Electronics which interprets most of the OBD-2 interfaces used in cars. Several versions of ELM have been developed over the years, the most well-known and functional of which is the ELM327 (figure 16).

The ELM327 offers a simple interface that can be connected via USB, RS-232, Bluetooth or Wi-Fi. Today there are many programs that connect to ELM, most of them are made for mobile devices.

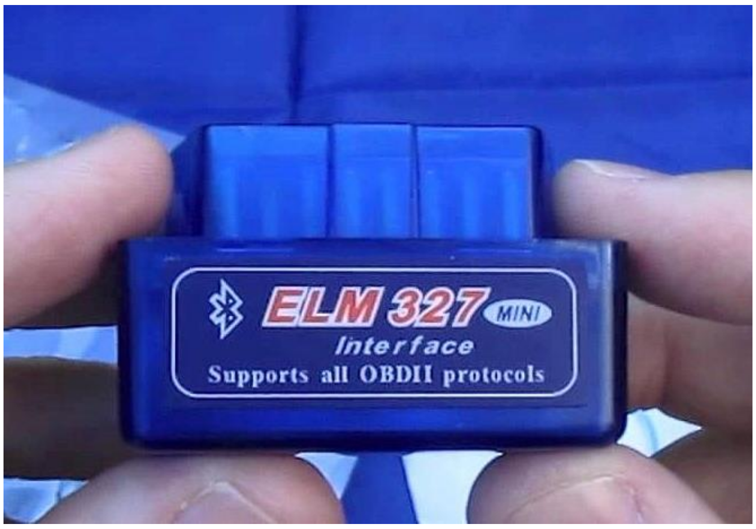


Figure 16. A very compact ELM device that is plugged into the vehicles OBD-2 socket and then connected to the phone with Bluetooth. This device can read data from most OBD-2 vehicles, but is missing the support for older vehicles.

3 T-Diagnostics

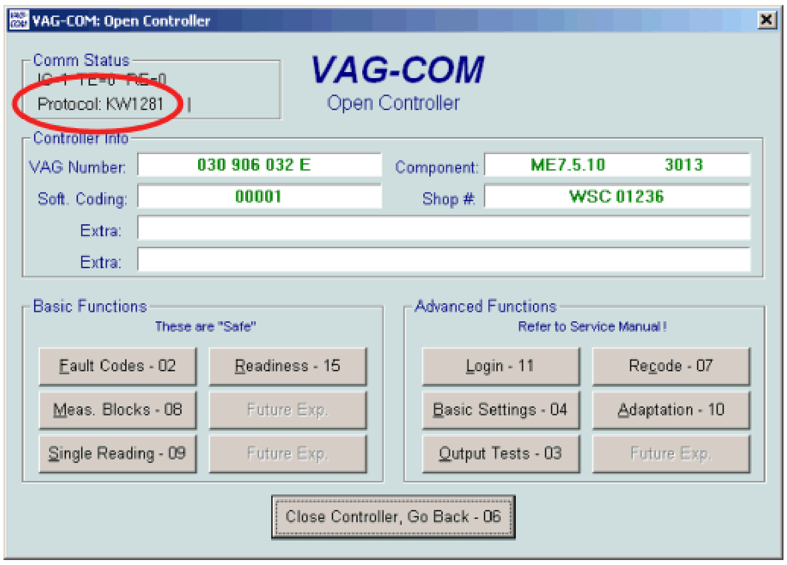
The developed program (named T-Diagnostics) is made for older cars that only have one ECU. Therefor the Engine Control Unit is from here on called ECU.

3.1 Diagnostics protocol description

Some older VAG cars (Golf 3 for example) use KW-1281 protocol when communicating with ECU via OBD2 port. This was created before CAN was used in cars and is usually ignored by most diagnostics' software (the exceptions are VAG-COM and some high-end professional tools, like Delphi). All of them have their bad sides (for example, VAG-COM only works on Windows computers and you must have all the right drivers, with right cables and all the professional tools are very expensive). For the regular user, there is not much to choose from. The ELM interface is supporting every other OBD-2 protocol, except for KW-1281. The reason for this is the missing documentation for KW-1281.

Of course, there are some programs made by people that can be found online, but most of them require more tools, in addition to your VAG OBD2 cable. If, however you are used to VAG-COM and are ready to use a few microcontrollers (in my example, an Arduino and Raspberry PI) you can get the same information and more, while having the possibility to access it all from your phone instead of PC.

Not much information exists about KW-1281 protocol. The manufacturer has only published general information and most chip tuners who know the information are holding it in secret as their trade secrets. The best source for information is VAG-COM, but you will need some way to read the signals on hardware level. It is also worth mentioning that KW-1281 has a similar protocol called ISO 9141-2 and they can be confusing. It can usually be checked with VAG-COM to see which protocol your vehicle uses (figure 17).


VAG-COM: Open Controller

Comm Status
IC=1 TE=0 RE=0
Protocol: KW1281

VAG-COM
Open Controller

Controller Info
VAG Number: 030 906 032 E
Component: ME7.5.10 3013
Soft. Coding: 00001
Shop #: WSC 01236
Extra:
Extra:

Figure 17. Checking what OBD2 protocol you ECU is using with VAG-COM. [28]

3.1.1 Data received

The ECU will send data from sensors in 2 integers. It is up to the diagnostics program to calculate the correct result. This information is also scarce and the best source for the author was to check the open source codes for similar programs. However, as it turns out, every car can have small differences. If the data received is incorrect then a program called ALLDATA can be used to find you which sensor your car uses. Each sensor sends their data by a change in resistor. Knowing that we can assume what data is sent to the ECU by the sensor and therefor figure out the data send by the ECU to the diagnostics device. A good example for this is the throttle pedal position. There is a rheostat attached to the pedal and pushing it changes the voltage that is received by the ECU. The ECU then sends the voltage converted to an integer (b) and also the maximum (a) to the diagnostics device. Then the device calculates the position (100% * (b/a)). If the pedal is in the middle, then b value is around 66/67 and if it is completely pushed down then b is equal to 133. If the pedal is in its normal state then the b is equal to 0. Absolutely every sensor behaves the same, except that sometimes instead of maximum value there can be either minimum value or sometimes only 1 value is used for calculations and the other one is not used (but the protocol dictates that 2 values must always be sent). An example when only one value is necessary is a valve attached to the air filter that is fully opened or fully closed, depending if the incoming air is “COLD” or “WARM”. For that only 1 or 0 is required and no other calculation is done.

3.2 Communication

The communication between diagnostics device and ECU is established by following KW-1281 protocol. The communication takes place in synchronous manner and there is always one device that sends and another device that reads. A small fault detection system is built into the protocol, but as the communication speed is rather slow and amount of data sent is small, this system is rarely needed, but helps to keep the data integrity in check.

3.2.1 Initialization

There are 2 phases for communication: wake up and actual communication.

One reason why most devices ignore this protocol is it’s wake up stage. While most of the communication will take part in widespread baudrates (4800, 9600, 10400 etc.) the wakeup stage requires every ECU using this protocol to use a baudrate of 5 bits/s to start the communication and then switch to faster baudrate. This caused many problems as most devices do not support such low baudrates. In addition, there is also a difference in parity check and data size in package (parity check disappears and data size increases to 8 from 7). With Windows, users can find themselves with the problem that switching the baudrate takes too long and some data is lost, this makes the ECU to end the communication. While the wakeup data may not be important, KW-1281 protocol requires that all data sent, must be sent back to the sender, before new data is sent (this is used synchronize and check the data). If the computer lags behind while changing baudrate, then ECU will not send any new data and timeout occurs. [28]

To get this to work with your PC, instead of changing the baudrate, each bit is sent with a 200ms delay (when using 9600 baudrate) to simulate slower baudrate. Another weird part of this protocol is the fact that the data in the first package is sent reversed. So, if you want to send the byte 1 (000 0001) then instead you must send (100 0000). [28]

3.2.2 Actual communication

After you have woken up one of the control units (they will be described later, each one is belonging to a group of 4 sensors and sends their own data), the actual communication can start.

No matter which control unit is chosen, the communication always follows the same pattern. A few notes to remember first:

All data is in hexadecimal notation, during communication and also when described here for consistence sake. When either side sends a package, then a calculation is done (0xFF – data) and the result is sent back. This is used to check that all the data that is sent is correct and also to synchronize the communication (there is no additional parity check). The result is also called **complement**.

In addition, whenever PC sends any data, then it is immediately echoed back by the chip in OBD cable. They should be ignored as they are not important, unless for debugging purposes.

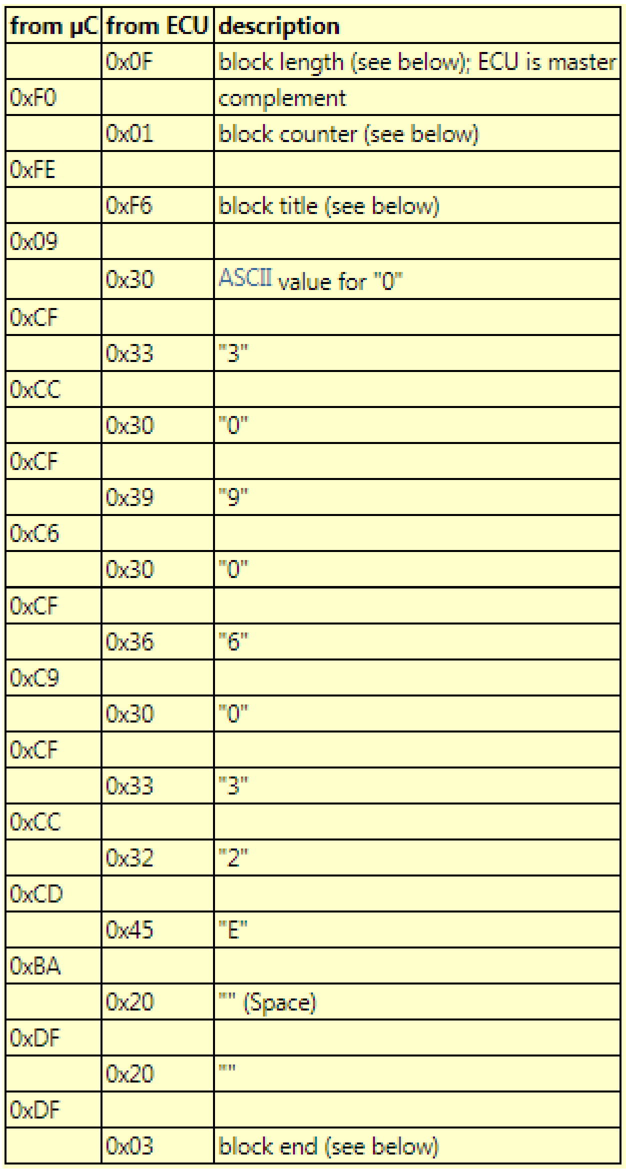
It cannot be stressed enough how important it is to determine the correct baudrate after wake-up part. While 9600 is most widespread, some ECUs can use other speeds, for example 10400. When using wrong speed, you will only get scrap data. For a device that can be used on multiple cars, the program must take this into account and try different speeds.

When using a microcontroller there is another problem that can occur (will not happen with Windows devices which have much slower I/O). When your device receives data and immediately sends back the complement data, the ECU will not have enough time to react and communication will fail. To fix this, before sending the complement, there needs to be a slight pause (5ms will do). It is important that the pause is not too long. ECU has about <500ms timeout period. If the timeout occurs then everything has to start from the beginning. [28]

3.2.3 Diagnostics data

After a control unit has woken up, it sends introduction data to the diagnostics device. First, it sends some sync data which is used to make sure that the communication is established correctly, if ECU response cannot be parsed, the connection did not establish correctly. When the ECU receives the complement, it sends the first block of data. For example, let’s use control unit 1 (also known as engine in VAG-COM): [28]

First, ECU sends the device data. It consists of 4 blocks, each block consists of block length, block counter, block title, a string of multiple bytes long (depending on the block length sent by the ECU) and a block end byte. One full block is shown in figure 18. [28]



from µC	from ECU	description
	0x0F	block length (see below); ECU is master
0xF0		complement
	0x01	block counter (see below)
0xFE		
	0xF6	block title (see below)
0x09		
	0x30	ASCII value for "0"
0xCF		
	0x33	"3"
0xCC		
	0x30	"0"
0xCF		
	0x39	"9"
0xC6		
	0x30	"0"
0xCF		
	0x36	"6"
0xC9		
	0x30	"0"
0xCF		
	0x33	"3"
0xCC		
	0x32	"2"
0xCD		
	0x45	"E"
0xBA		
	0x20	"" (Space)
0xDF		
	0x20	""
0xDF		
	0x03	block end (see below)

Figure 18. ECU sends controller ID given to the ECU by the manufacturer [28]

Some notes: The block length is actually the block length without block end byte.

Block counter is the same for ECU and the diagnostics device, both have to increment it, other ways it means that there has been communication error and ECU stops the communication.

Block title shows what type of block is sent, useful for the algorithm to understand what kind of data to expect. Some titles are reserved to only be sent to the ECU and are never sent by the ECU (for example 0x05 – clear all ECU errors).

No complement is sent after block end.

After one side receives a data block, it always has to send an acknowledge block to continue the communication (if neither side has nothing to send, then they keep sending acknowledge blocks to each other).

This block is similar to the data block seen previously, but is much shorter, only 1 byte is sent as data (0x09 – ACK).

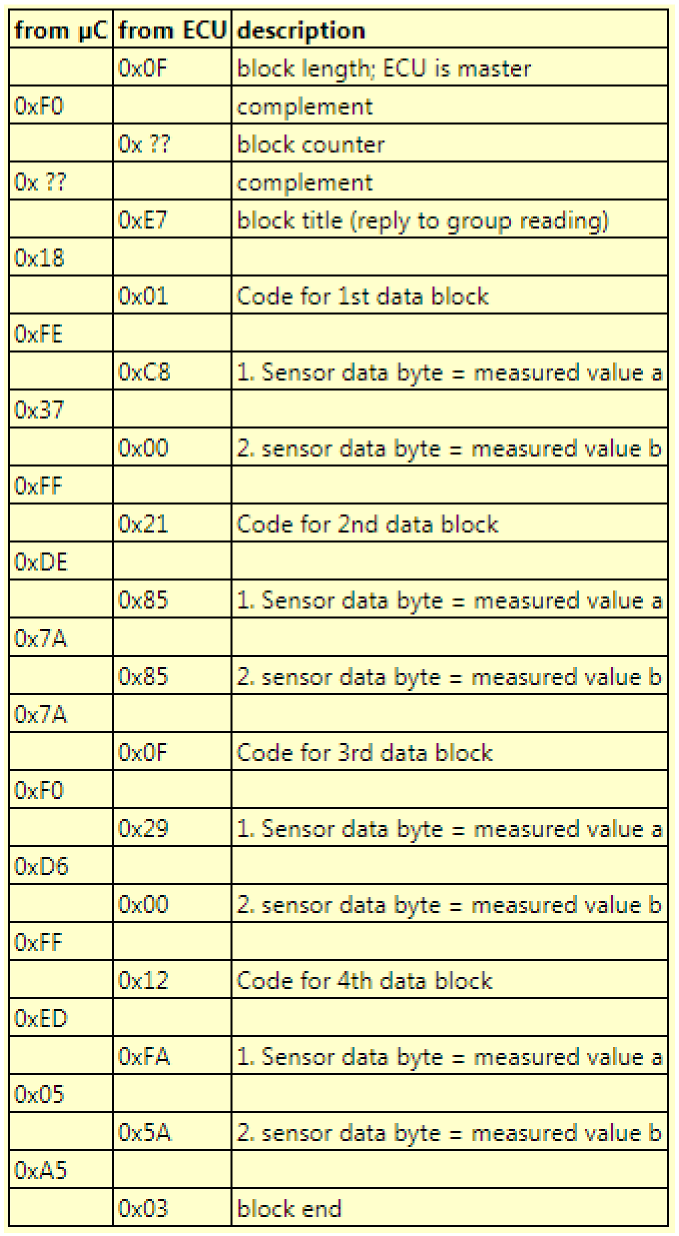
After the ECU has sent 4 data blocks and received the acknowledge for each one, then both devices will stay in acknowledge loop.

After the ECU has send an acknowledge block and the user has picked a new action. For this the device has to send a block called **group reading**. This is the main diagnostics action and it requests data from up to 4 sensors, grouped to 255 blocks. Group reading is only available for engine unit. As this car does not have so many sensors, many groups share the same sensors and some might send no data at all.

Group reading is sent the same way as the acknowledge block, only changing the title and data bytes (title is now 0x29 and data is a byte from 0 to 255, depending on the block you wish to read).

The ECU responds to this by sending a “reply to group reading” block (identified as 0xE7). Its data fields consist of 3 bytes per each 4 sensors (sensor number, first int and second int). The sensor number shows which of the 255 possible sensors is read and the 2 integers sent are ‘a’ and ‘b’ which the diagnostics device has to put into an equation which then provides the calculated value.

An example is on figure 19:

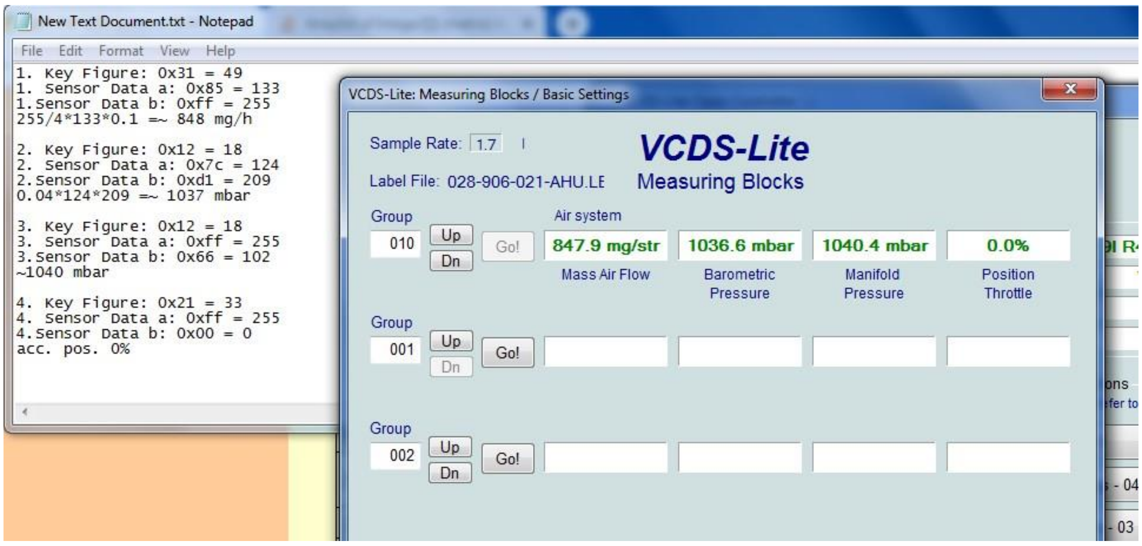


from µC	from ECU	description
	0x0F	block length; ECU is master
0xF0		complement
	0x ??	block counter
0x ??		complement
	0xE7	block title (reply to group reading)
0x18		
	0x01	Code for 1st data block
0xFE		
	0xC8	1. Sensor data byte = measured value a
0x37		
	0x00	2. sensor data byte = measured value b
0xFF		
	0x21	Code for 2nd data block
0xDE		
	0x85	1. Sensor data byte = measured value a
0x7A		
	0x85	2. sensor data byte = measured value b
0x7A		
	0x0F	Code for 3rd data block
0xF0		
	0x29	1. Sensor data byte = measured value a
0xD6		
	0x00	2. sensor data byte = measured value b
0xFF		
	0x12	Code for 4th data block
0xED		
	0xFA	1. Sensor data byte = measured value a
0x05		
	0x5A	2. sensor data byte = measured value b
0xA5		
	0x03	block end

Figure 19. Reply from ECU [28]

3.3 Parsing data

When creating the diagnostics tool, the author must find somehow to test that the equations used to calculate the data sent by the vehicle are correct. There is no official documentation about it and only trustworthy source is VAG-COM.



```
1. Key Figure: 0x31 = 49
1. Sensor Data a: 0x85 = 133
1.Sensor Data b: 0xff = 255
255/4*133*0.1 =~ 848 mg/h

2. Key Figure: 0x12 = 18
2. Sensor Data a: 0x7c = 124
2.Sensor Data b: 0xd1 = 209
0.04*124*209 =~ 1037 mbar

3. Key Figure: 0x12 = 18
3. Sensor Data a: 0xff = 255
3.Sensor Data b: 0x66 = 102
~1040 mbar

4. Key Figure: 0x21 = 33
4. Sensor Data a: 0xff = 255
4.Sensor Data b: 0x00 = 0
acc. pos. 0%
```

VCDS-Lite: Measuring Blocks / Basic Settings

Sample Rate: 1.7

Label File: 028-906-021-AHU.LE

VCDS-Lite
Measuring Blocks

Group 010 — Air system

Mass Air Flow	Barometric Pressure	Manifold Pressure	Position Throttle
847.9 mg/str	1036.6 mbar	1040.4 mbar	0.0%

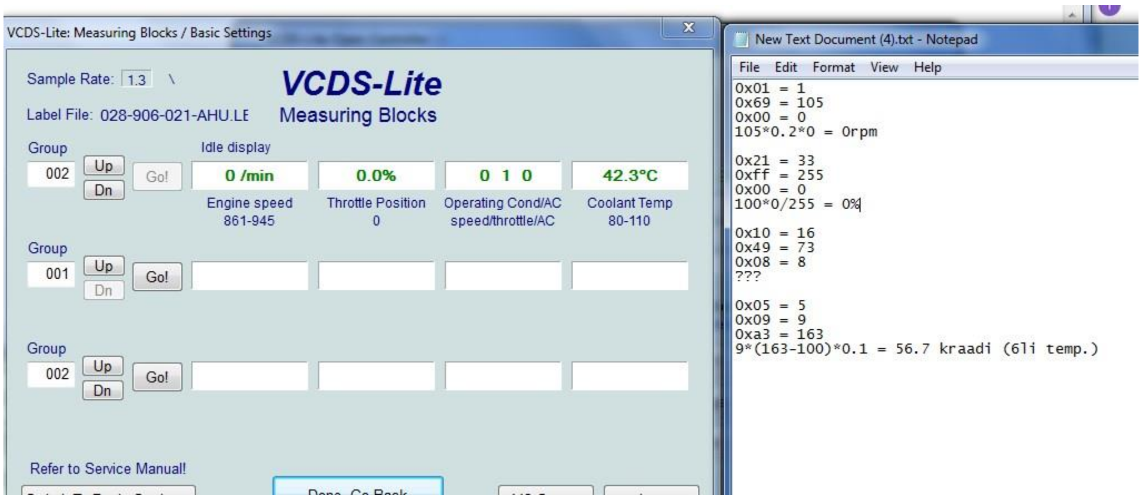
Group 001

Group 002

Figure 20. First test – engine running, read sensors in group 10

On the left in figure 20 there are the values sent by the authors cars when connected to the developed program. Along the data there are also the equations that are published online. Using them, 4 values are calculated – intake air flow, outside air pressure, manifold pressure (air pressure inside the intake manifold) and throttle position. On the right, there is VAG-COM, connected to the same car, at the same time and reading the same data. It can be seen that the data sent to the developed program and the found equations give us the same results as they do with VAG-COM (the small differences come from the fact that VAG-COM refuses connection on the same COM port as other devices so there is a little timeframe between 2 programs connecting to the vehicle). To be sure about the correctness of data, more tests have to be run, but it can be seen that the testing has been successful.

3.4 Further testing


VCDS-Lite: Measuring Blocks / Basic Settings

Sample Rate: 1.3

Label File: 028-906-021-AHU.LE

VCDS-Lite
Measuring Blocks

Group	Idle display			
002	0 /min	0.0%	0 1 0	42.3°C
	Engine speed 861-945	Throttle Position 0	Operating Cond/AC speed/throttle/AC	Coolant Temp 80-110
001				
002				

Refer to Service Manual!

New Text Document (4).txt - Notepad

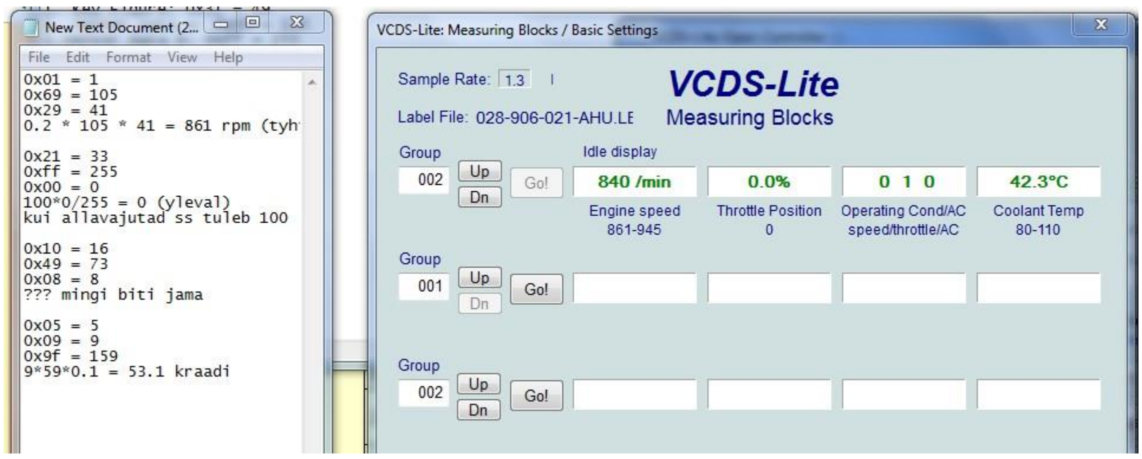
```
0x01 = 1
0x69 = 105
0x00 = 0
105*0.2*0 = 0rpm

0x21 = 33
0xff = 255
0x00 = 0
100*0/255 = 0%

0x10 = 16
0x49 = 73
0x08 = 8
???

0x05 = 5
0x09 = 9
0xa3 = 163
9*(163-100)*0.1 = 56.7 kraadi (6li temp.)
```

Figure 21. Next test – engine stopped


New Text Document (2...

```
0x01 = 1
0x69 = 105
0x29 = 41
0.2 * 105 * 41 = 861 rpm (tyh

0x21 = 33
0xff = 255
0x00 = 0
100*0/255 = 0 (yleval)
kui allavajutad ss tuleb 100

0x10 = 16
0x49 = 73
0x08 = 8
??? mingi biti jama

0x05 = 5
0x09 = 9
0x9f = 159
9*59*0.1 = 53.1 kraadi
```

VCDS-Lite: Measuring Blocks / Basic Settings

Sample Rate: 1.3

Label File: 028-906-021-AHU.LE

VCDS-Lite
Measuring Blocks

Group	Idle display			
002	840 /min	0.0%	0 1 0	42.3°C
	Engine speed 861-945	Throttle Position 0	Operating Cond/AC speed/throttle/AC	Coolant Temp 80-110
001				
002				

Figure 22. Next test – engine running

The previous two pictures (figures 21 and 22) have the same query, but on the first one the engine did was stopped, on the second one it is running. The first value read was the number of revolutions of the engine, if the engine is not running then the number of revolutions is 0. However, before sending the other query, the car was started and a value of 861 or 840 was obtained (there were a few minutes between the queries to record the data and transfer the control of OBD device to another application, during which time there was a slight difference in the RPM, which is normal for this vehicle as the RPM is higher immediately after start-up and then slowly falls down). These figures were compared with the readings on the dashboard tachometer. The second value obtained from the query was again the position of the accelerator pedal. At the time of each query, the throttle pedal was in up position. The third value is not stated in the documentation but VAG-COM links it to the climate control, which is not present in this car and therefore shows default values. The fourth value is the coolant temperature. It should be noted that this measurement was taken in the garage in winter, after a short drive which warmed up the coolant a little and then left it to cool down. It should also be noted that in one of the pictures it is referred to as the oil temperature, as it was written in the documentation, but now it is known that there is an error (small things like this make it difficult to create a working application, as each thing has to be checked several times on several cars to be sure of the correctness of the data). This car does not have a sensor to check oil temperature and even VAG-COM does not understand that.

3.5 Used hardware

Current system consists of multiple devices that are all connected to each other. Starting from the car, the first device, plugged into the OBD-2 port is a VAG-KKL cable (figure 23). It is a standard cable for diagnostics with older VAG cars. They can connect to VAG-COM, but usually no other program can detect it as diagnostics tool. They use virtual COM port to communicate with the chip inside them that changes 5V serial communication to 12V. Also reading and writing to OBD-2 take place on the same pin, instead of 2 different pins like in serial or USB cables.

Figure 23. VAG-KKL cable. [29]

Connected to the VAG-KKL cable is the diagnostics program. It is a Java program which takes care of the communication of KW-1281 protocol which is described at the beginning of this chapter. The program code is in Appendix 2.

The program relays all the data to the server, which is running on Raspberry Pi using Node-Red architecture.

3.5.1 Server

After the diagnostic tools accepts data from ECU, it is first checked for errors, using KW-1281 standard. It states that after a byte is received, a calculation is done on it and then sent back. The calculation is quite simple, the received byte is subtracted from 0xFF. When sending a byte, the device must wait for the connected device to send back the calculated result and it is then checked that if the data is correct (the sender checks that 0xFF minus the sent byte is equal to the received byte) and sends another byte. If it is wrong then usually the connection has to be restarted (the ECU will always disconnect in case of wrong answer sent back).

The reason for that kind of fault detection is mainly that the OBD-2 tool always echoes the received byte back to the sender, which is usually ignored. However, if the fault detection would send exactly the same byte, then it would be easy to mix them up.

If the diagnostics tool receives correct data, then it is relayed to the server. In this project, a Raspberry Pi is used as the server. Communication between the program and server is done by MQTT protocol. The diagnostics tool will publish the received data to a topic which corresponds to the current activity. The program is also subscribed to a topic where it will get instructions from the user.

Server is built with Node-Red where flows will begin if a command is sent to a topic. Parsing of diagnostics data is also done by the server and then printed to the user in human readable form. All commands to the diagnostics tool are given by the user by sending request via browser and the server will relay them to the program by publishing the command and it’s parameters (like which sensors to read) to a specific topic, which corresponds to one of the main diagnostics activities (they are – reading diagnostics data, reading error codes or deleting error codes).

3.5.2 Flows

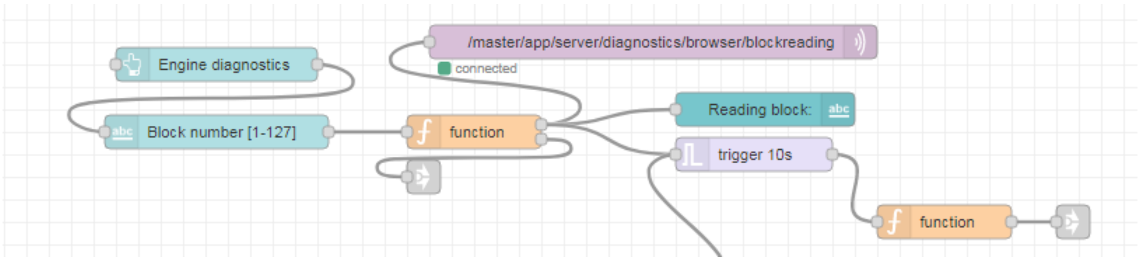
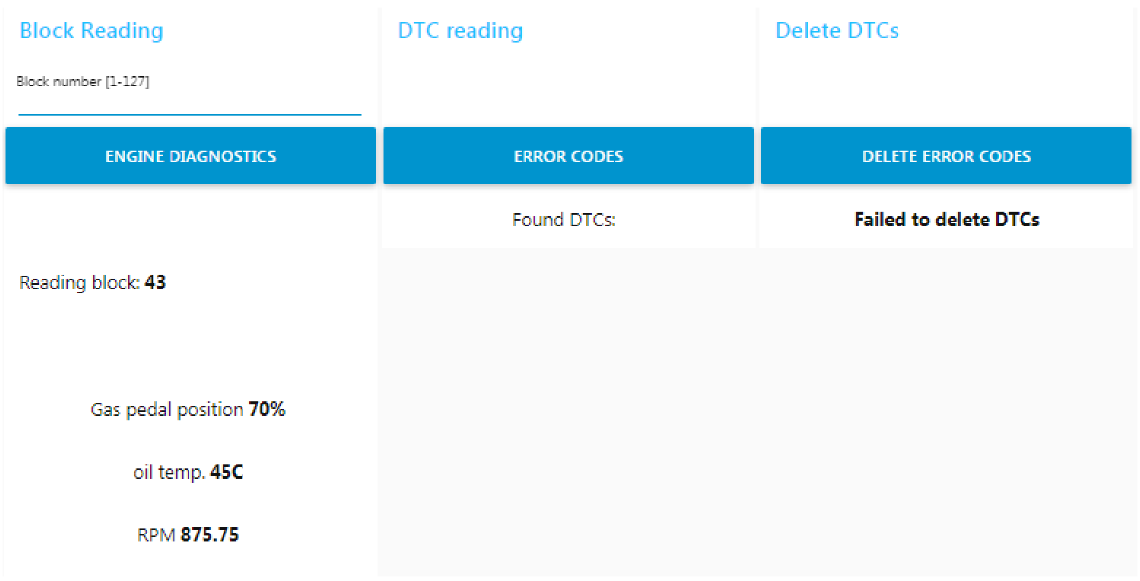


Figure 24. Send request flow

The first flow is to send requests to the diagnostics tool. The user inputs their request (figure 25) by writing the number of a block they want to read (every block corresponds to 4 sensors and the user must either know them or just try till they find what they need. The blocks can be checked in VAG-COM, but no official documentation exists.). Then they can press the send button and their command is sent to the diagnostics tool. If the number inputted is wrong, then they will see an error message (all error messages are linked by the grey boxes on the flows and can be seen on figure 26). If the server fails to respond before timeout, an error is shown.



Found DTCs:

Failed to delete DTCs

Reading block: **43**

Gas pedal position **70%**

oil temp. **45C**

RPM **875.75**

Figure 25. User interface

In figure 24, it is shown that the user wanted to read sensors in block 43. The vehicle responded with 3 values for this block. These values are for gas pedal position, oil temperature and engine revolutions. The data is parsed with the help of VAG-COM and also has the same error – this vehicle is actually lacking oil temperature sensor and is instead providing coolant temperature, while other vehicles are providing actually oil temperature for block 43. However, the data is correct and the mechanic will know that it is coolant temperature.

On the right side of figure 25, it is shown that the user has tried to delete the error codes (DTCs), but for some reason has failed. This happened because no error codes where provided by the vehicle (the user must always read the error codes first as a safety measure).

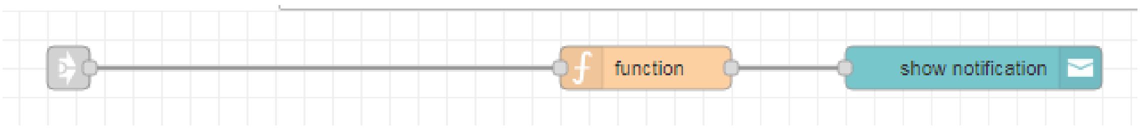


Figure 26. Error messages flow

When an error message is requested by the server, it is compiled into a human readable form to print out the error code and a message to the screen.

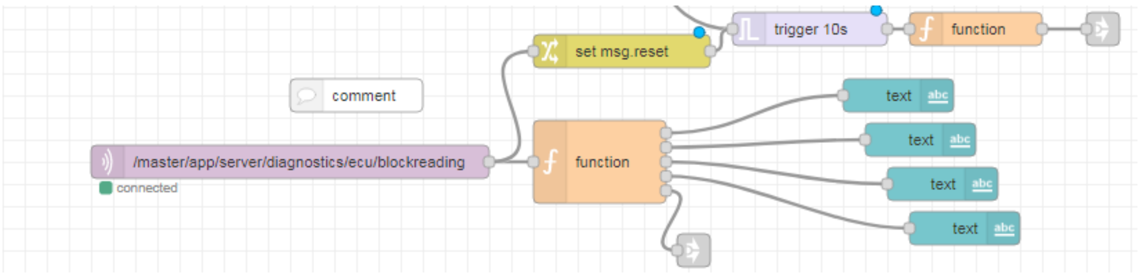
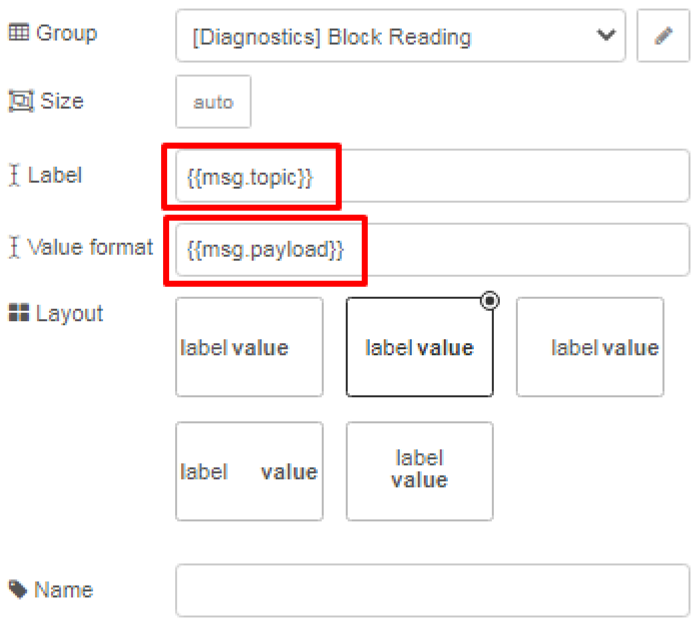


Figure 27. Response to a group reading flow

When the vehicle responds to a group reading request (figure 27), the data is split into 4 different text boxes (1 for each sensor). The data is parsed by assigning the sensor name to the number (provided by the vehicle) and the received data in calculated to print a value which is understandable to humans (temperature is given in degrees, engine revolutions in RPMs). The text is divided so that sensor name is in topic and data in message payload, this makes it possible to print easily (figure 28). It can be seen that the label field is topic and value field is message payload.



Group: [Diagnostics] Block Reading

Size: auto

Label: {{msg.topic}}

Value format: {{msg.payload}}

Layout

Name

Figure 28. Text box description

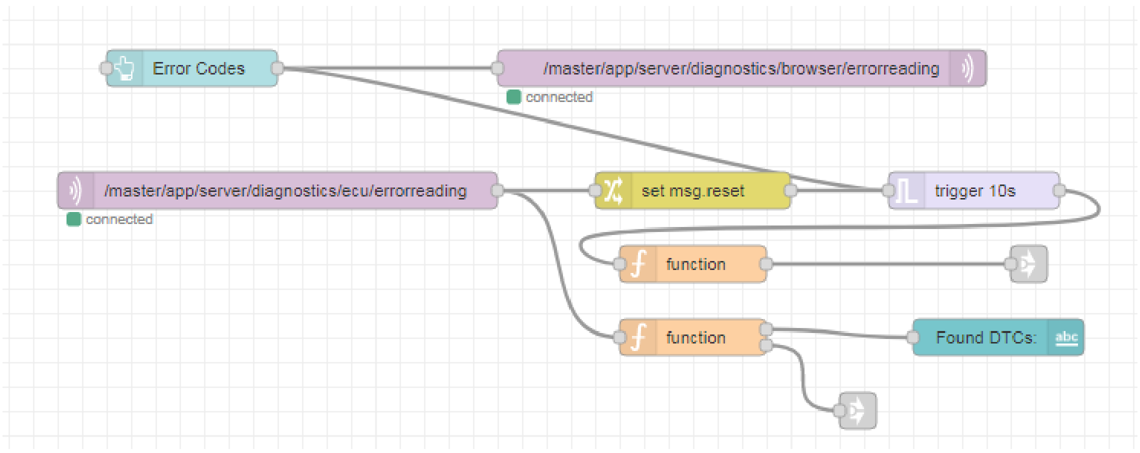

Error Codes
/master/app/server/diagnostics/browser/errorreading
connected
/master/app/server/diagnostics/ecu/errorreading
connected
set msg.reset
trigger 10s
function
function
Found DTCs:
abc

Figure 29. Send command to read error codes flow

When the user wants to read error codes from the car’s ECU (figure 29), then all they need is a click of a button (figure 25, DTC reading). The server publishes a command to *errorreading* topic that the diagnostics tool is subscribed to. If the error codes are received before the timeout, then they are printed under found DTCs field. If an error occurs, then an error message is displayed to the user.

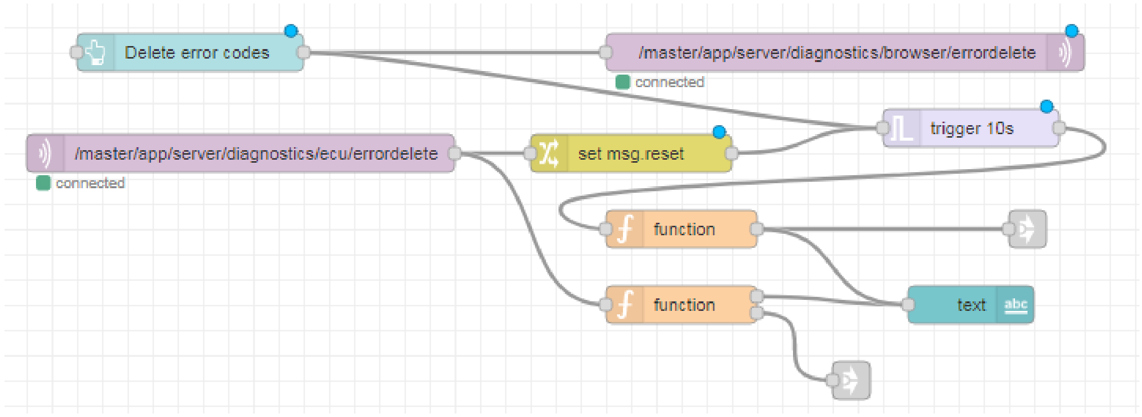

Delete error codes
/master/app/server/diagnostics/browser/errordelete
connected
trigger 10s
/master/app/server/diagnostics/ecu/errordelete
connected
set msg.reset
function
function
text

Figure 30. Delete DTCs flow

Deleting error codes (figure 30) is similar to reading them. Again, the user only needs to press a button, which makes the server publish a message in *errordelete* topic. When no problems arise, the user is told that all errors are deleted (after this it is recommended to read errors again to make sure they do not reappear. If they do then the problems are usually not fixed). In case of problems, the user is told that the program failed to delete error codes and a notification is shown which can give some hint about the problem.

3.5.3 Additional features

The error codes are stored as a 5-number digit in ECU's memory. To know what error the code represents, one must consult the Internet, as there is no official database for OBD-2 error codes. Each manufacturer has their own codes and formats, the 5-digit version is used by VAG.

There is however a solution, provided by Ross-Tech, who also are the developers of VAG-COM. Their webpage provides explanation to each error codes used by VAG.

To help the user, diagnostics tools can search the error codes in Ross-Tech forums and instead of displaying the error code to the user, the explanation is provided instead.

3.6 Testing the system

The development started in autumn, when it was already quite cold outside. Because there is little documentation about the subject, a lot of hours were first spent sitting in the car, and it can get quite cold. To fix this, the system must be useable without the vehicle.

Solution – make the car ECU work inside your home, using 220V instead of the 12V used in cars.

Because VAG cars are known for their reliability, they quite widespread even today, more than 20 years after they stopped manufacturing engines with KW-1281 protocol diagnostics. This makes it easy to find spare parts for them, including ECUs. One of them has been acquired by the author, which is quite similar to their own car (difference is that the ECU is for Golf 4, not 3, but they share the same engine and therefore the same diagnostics protocol).

First thing is to make sure which pins are required to power the ECU, send OBD-2 data and also to power the ignition wire, which is required for diagnostics. To find them, ALLDATA (a database used by mechanics for all spare parts information, wiring graphs and so much more for each car model) was used. It turns out that for that engine the ALLDATA has wrong/missing information and it took a few hours of testing to actually power the ECU. For power reasons, a 12V adapter is connected to all required wires. A spare OBD port was soldered to diagnostics wires to enable a diagnostics cable connection. This was before CAN became widespread and would not be possible with more modern cars.

Figure 31 shows the built device. On the left there is an OBD-2 cable together with OBD-2 port, which is connected to the Golf 4 ECU in the middle. On the right is the 12V adapter.

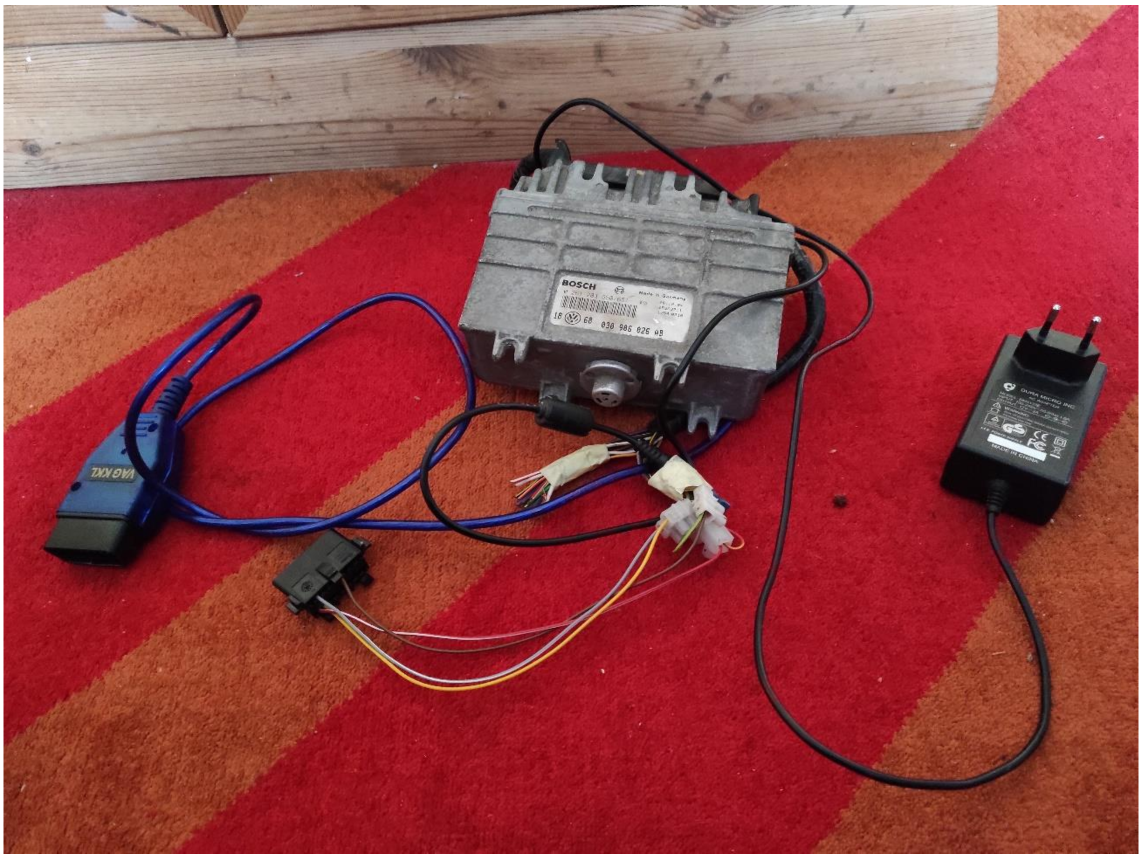


Figure 31. ECU with 12V adapter and OBD-2 cable

Before connecting the adapter to the ECU, make sure that you test its voltage first. The ECU can handle up to 18V, but the diagnostics cables are fragile and can be seriously damaged if not handled correctly.

3.7 Use cases for the diagnostics tool

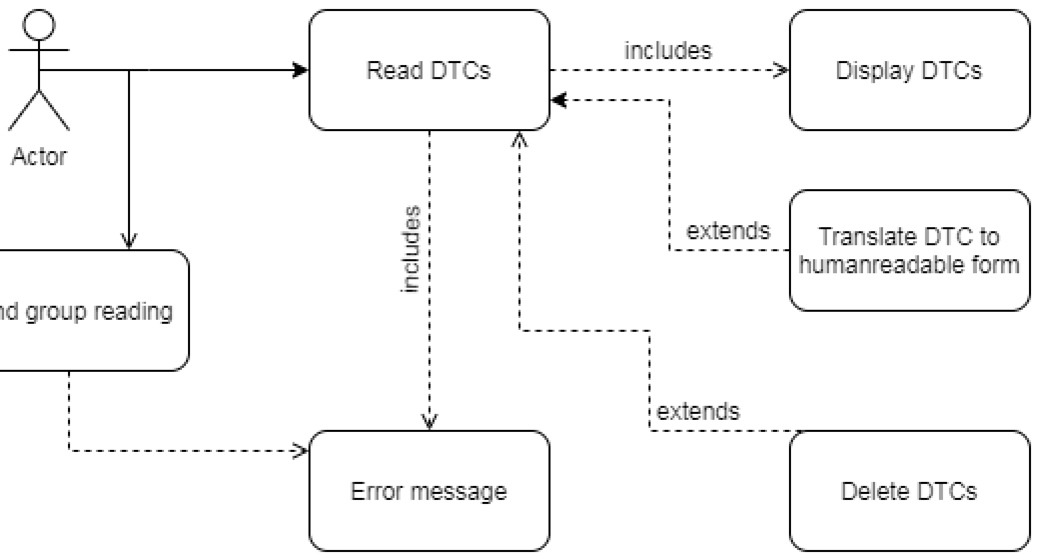


Figure 32. Use cases for the diagnostics tool

The figure 32 represents the most typical use cases for this system. The ECU allows three known operations: read diagnostics data, read DTCs (the error messages) and delete error messages.

3.7.1 Use case – reading data from sensors

Prerequisite: the user has connected the diagnostics device to the car.

Purpose: To read the desired data through the selected control module and display it on the device display.

Result: The user can see the status of the selected subsystem of the vehicle on the display of his device.

Scenario:

1. The user connects the device to the vehicle and opens the application.

2. The user inputs the desired group's number which sensors he desires to access.

3. The device sends a message to the vehicle to initiate the connection. The vehicle selects which module the device wants to connect to.

4. The module transmits its data and waits for further instructions.

5. The user reads the required data from the screen and prompts for new data until all the required data has been received.

6. The user closes the application and the device sends the vehicle the end of connection message. The vehicle will stop diagnostics program.

Extensions:

3.a The device fails to connect to the vehicle, displays an error message to the user, offering solutions to correct the problem.

3.7.2 Use case – access DTCs.

Prerequisite: A hazard light is illuminated on the dashboard of the vehicle and the mechanic has connected the device to the vehicle.

Purpose: Read from the car's ECU what the problem is that has caused the hazard light to be displayed.

Result: fault code(s) are displayed on the screen and, if necessary, the mechanic can immediately obtain the fault code explanation from the Internet.

Scenario:

1. The user connects the device to the vehicle.

2. The user requests DTCs to be read from ECU.

3. The device sends a message to the vehicle initiating the connection, and that it requires all DTCs to be read.

4. The module transmits all fault codes (if any) and waits for further instructions.

5. The user reads the error codes from the screen. The device is disconnected from the vehicle.

Extensions:

3.a The device fails to connect to the vehicle, displays an error message to the user, offering to provide solutions to fix the problem

5.a The device shall access the Ross-Tech forum and copy the DTC explanation to display to the user.

5.b After reading the more detailed explanation, the mechanic closes the application.

3.7.3 Use Case – delete all DTCs from ECU memory

Prerequisite: The mechanic has completed fixing the problem, but hazard light is still illuminated on the dashboard of the vehicle and the mechanic has connected the device to the vehicle.

Purpose: Read from the car's ECU the DTCs and attempt to delete them.

Result: fault code(s) are deleted from the ECU and if fix was successful, they do not show up if DTCs are read from the ECU again.

Scenario:

1. The user connects the device to the vehicle.

2. The user requests DTCs to be read from ECU, deleting is not possible without requesting the DTCs first.

3. The device sends a message to the vehicle initiating the connection, and that it requires all DTCs to be read.

4. The module transmits all fault codes (if any) and waits for further instructions.

5. The user requests the DTCs to be deleted.

6. The device sends a message to the car’s ECU to delete all fault codes.

7. User gets a message that all DTCs are deleted.

8. To make sure deleting was successful, user requests all DTCs to be read again.

9. Diagnostics tool requests the ECU to send all DTCs.

10. The user reads the error codes from the screen. The device is disconnected from the vehicle.

Extensions:

3.a The device fails to connect to the vehicle, displays an error message to the user, offering to provide solutions to fix the problem.

4.a If the device failed to start reading DTCs, an error is displayed to the user.

6.a The device failed to initiate deleting procedure and displays an error to the user.

10.a If the device fails to read DTCs, then an error is displayed to the user that there was an error reading fault codes (not to be confused with the case when there are no error codes stored in ECU)

10.b If no fault codes are received, user is notified that the ECU has no error codes stored, indicating that the deleting was successful and the error is fixed.

3.8 Functional Requirements

System can connect to the ECUs of VAG vehicles with engine that uses KW-1281 OBD protocol.

All the sensors in connected vehicle are available for diagnostics.

The system can read and delete DTCs.

The system can translate DTCs into human readable form.

3.9 Non-Functional Requirements

The device connected to the ECU is running Windows OS.

The server is built with Node-Red.

The diagnostics protocol is predetermined by the manufacturer.

The diagnostics cable must use virtual serial port for connection with the system.

The system UI must be comfortable to use for those already familiar with VAG-COM.

3.10 Conclusion

In conclusion, a system is created that has access to all diagnostics processes in Volkswagen Group cars from around 1994 to cheaper cars manufactured in 2004. The system is tested on 1996 Golf Mk3 with 1.9 liters turbodiesel engine. The same engine was manufactured for a long time and used on many different cars for both Volkswagen and other VAG cars like Audis or Seats and will most likely work with them too.

Summary

The objective of this thesis is to give an overview of control system interfaces that people can see in their everyday life, with focus on passenger cars. All of the mentioned systems are used to create a diagnostics system.

The system created, called T-Diagnostics is a Java program created for Windows environment. It can access a Node-Red server, running on Raspberry Pi, via MQTT. This system is designed to connect with VAG cars manufactured from mid-1990-s to early 2000-s. Diagnostics functions which are required are the ability to access various sensors data and read/delete error codes from ECU memory.

Conclusions from the thesis: To make the best system, it is necessary to consider how the user will be most comfortable with using the service. This can be determined by observing which programs the users have used and try a similar approach. Try to see what the users are most annoyed by and fix those problems.

References

[1] Denso, "Denso," Denso, 24 March 2017. [Online]. Available: https://www.denso-am.eu/media/corporate-news/2017/march-2017-newsletter-a-history-of-engine-management-systems-according-to-denso/. [Accessed 23 December 2020].

[2] Bosch, "Bosch," Bosch, 2021. [Online]. Available: https://www.bosch-mobility-solutions.com/en/solutions/control-units/eengine-control-unit/. [Accessed 26 December 2020].

[3] blaineautocare, "Blaine Auto Care," 29 September 2020. [Online]. Available: https://www.blaineautocare.com/2020/09/29/a-complete-history-of-the-check-engine-or-service-engine-soon-light/. [Accessed 26 December 2020].

[4] K. Rangam, "Go Mechanic," 4 October 2020. [Online]. Available: https://gomechanic.in/blog/ecu-electronic-control-unit-explained/. [Accessed 6 Januray 2021].

[5] Layton, "Layton Remaps," Layton, 1 September 2018. [Online]. Available: https://www.laytonecuremaps.co.uk/financial-help/. [Accessed 4 Janruay 2021].

[6] Continental, "Continental," Continental, [Online]. Available: https://www.continental-automotive.com/en-gl/Passenger-Cars/Vehicle-Networking/Control-Units/Body-Control-Modules. [Accessed 6 January 2021].

[7] C. Ebert and J. Capers , "Embedded Software: Facts, Figures, and Future," *Computer,* vol. 42, pp. 42-52, 2009.

[8] Continental, "Continental," Continental, [Online]. Available: https://www.continental-automotive.com/en-gl/Passenger-Cars/Vehicle-Networking/Control-Units/Door-Control-Units. [Accessed 8 January 2021].

[9] "CAN in Automation (CiA)," CAN in Automation (CiA), [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/. [Accessed 16 January 2021].

[10] "CAN Newsletter," [Online]. Available: https://can-newsletter.org/engineering/applications/160322_25th-anniversary-mercedes-w140-first-car-with-can/. [Accessed 16 Januray 2021].

[11] "CAN Newsletter," [Online]. Available: https://can-newsletter.org/engineering/engineering-miscellaneous/181220_at-the-turn-of-the-year-classical-can-can-fd-and-can-xl. [Accessed 17 January 2021].

[12] "csselectronics," CSS Electronics, 2021. [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en. [Accessed 17 January 2021].

[13] "Ross-Tech," Rocc-Tech, [Online]. Available: https://www.ross-tech.com/vag-com/. [Accessed 4 February 2021].

[14] "Ross-Tech," Ross-Tech, [Online]. Available: http://www.ross-tech.com/vag-com/tour/704-screens/meas_blocks.png. [Accessed 9 Feburary 2021].

[15] "Raspberry Pi," The Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed 12 February 2021].

[16] "Raspberry Pi," The Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Accessed 12 February 2021].

[17] C. Mobberley, "Learn Adafruit," AdaFruit, 25 January 2014. [Online]. Available: https://learn.adafruit.com/raspberry-pi-hosting-node-red. [Accessed 13 February 2021].

[18] S. Cope, "Steve's internet Guide," [Online]. Available: http://www.steves-internet-guide.com/mqtt/. [Accessed 12 February 2021].

[19] "Arduino," Arduino, [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardEthernet/. [Accessed 13 February 2021].

[20] "ESP32," Espressif Systems, [Online]. Available: http://esp32.net/. [Accessed 13 February 2021].

[21] "Nettigo," Nettigo, [Online]. Available: https://nettigo.eu/products/esp-32-development-board-with-wifi-bluetooth-30pin. [Accessed 13 February 2021].

[22] Y. Dai and J. Zhao, "Advanced Methods of Risk Assessment and Management," *Methods in Chemical Process Safety*, 2020.

[23] "enertiv," 16 January 2019. [Online]. Available: https://www.enertiv.com/resources/faq/what-is-fault-detection-diagnostics. [Accessed 7 March 2021].

[24] "holtsauto," Holt Lloyd International Limited, [Online]. Available: https://www.holtsauto.com/redex/support/cars-dpf-warning-light/. [Accessed 9 March 2021].

[25] "Volkswagen," Volkswagen, [Online]. Available: https://www.volkswagen.co.uk/owners-and-drivers/my-car/warning-light/brake-system. [Accessed 10 March 2021].

[26] "COGApa," COGApa, [Online]. Available: http://www.cogapa.com/history.htm. [Accessed 16 March 2021].

[27] T. A. M. C. S. Team, "Casestudies Atlantic Motorcar," Atlantic Motorcar Center, 14 April 2015. [Online]. Available: http://casestudies.atlanticmotorcar.com/audi-

q7-washer-solvent-warning-light-sensors-and-rain-x-deicer/. [Accessed 4 March 2021].

[28] F. Schäffer, "blafusel," 15 July 2009. [Online]. Available: https://www.blafusel.de/obd/obd2_kw1281.html. [Accessed 11 December 2020].

[29] "okidoki," okidoki, 20 April 2021. [Online]. Available: https://www.okidoki.ee/item/diagnostika-vag-com-kkl-4091-audi-skoda-seat-vw/10440247/. [Accessed 29 April 2021].

[30] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Appendix 1 – Non-exclusive license for reproduction and publication of a graduation thesis

I Trevor Kallaste

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis “Vehicle engine control and diagnostics software”, supervised by Vladimir Viies.

    1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

    1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.

3. I confirm that granting the non-exclusive license does not infringe other persons’ intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

10.05.2020

Appendix 2 – algorithm ja program code.

The code for java program that connects to the ECU is uploaded to: https://gitlab.pld.ttu.ee/trkall/T-Diagnostics

Additionally, the repository also includes the Node-Red server config file to set up the server and also MQTT Arduino project code to set up connection between diagnostics tool and server (if necessary). The Arduino code is built for ESP-32 system and might not work on all Arduino based systems.

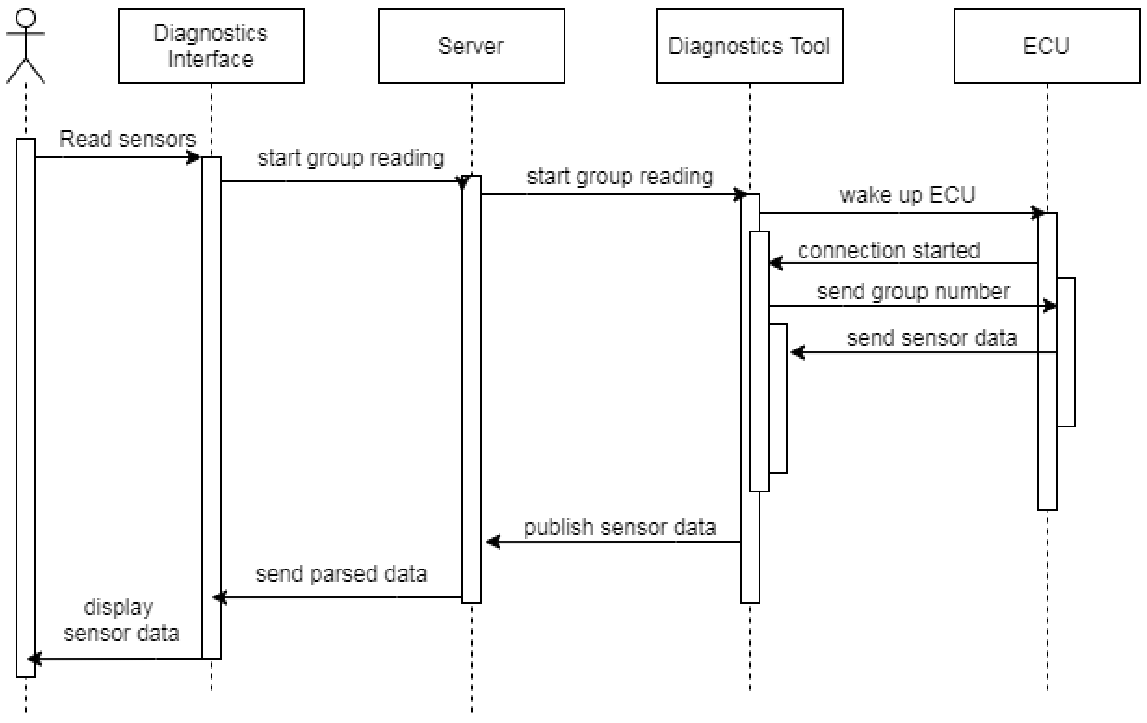


Figure 33. Send group reading algorithm

Figure 33 shows the algorithm of sending a group reading to access the sensor data in ECU. If the user wants to read the measurement data from one or more of sensors, a group reading is initiated, where a group number which consists the desired sensors is sent to the server via the diagnostics interface. The package to the server is sent with MQTT, publishing the group number to the group reading topic. This is sent forward to the diagnostics tool by the server using MQTT again. The diagnostics tool first initiate connection with ECU by sending the wake-up package. If the Ecu was woken up successfully, the group reading package is sent. Then ECU responds with the sensors data, which the tool forwards to the server. In the server, the data is calculated to show the data in human readable form, which is displayed on the interface.

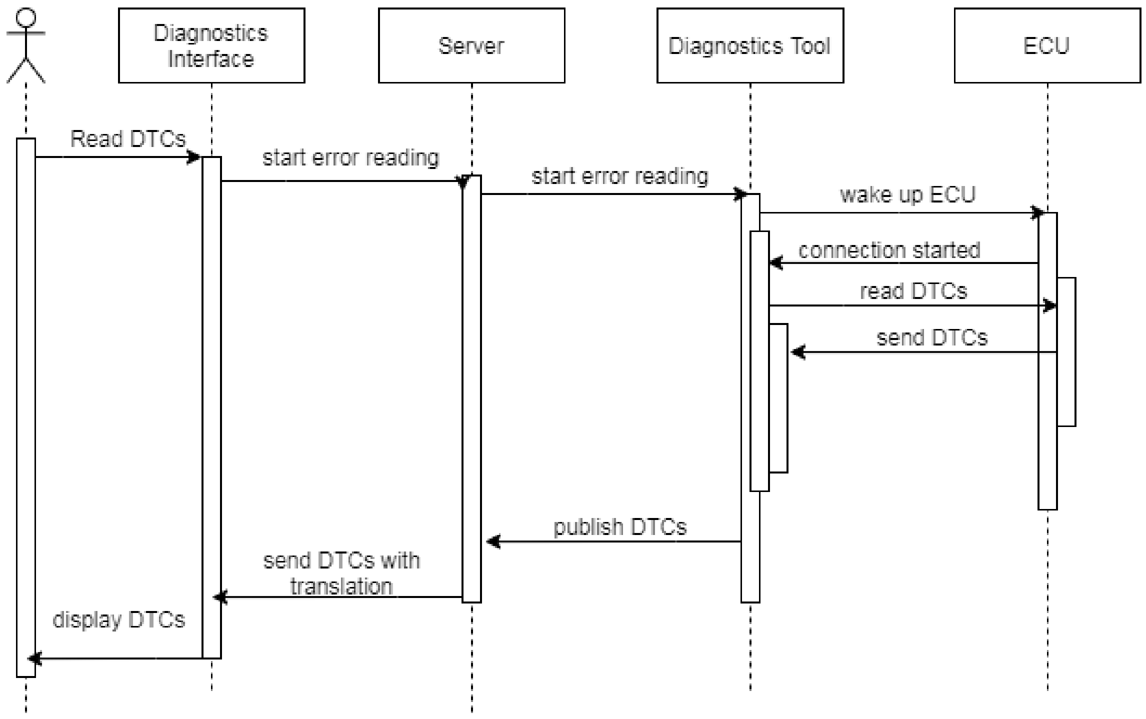


Figure 34. read DTCs algorithm

Reading error codes from the ECU is similar to reading sensor data. The request is sent by the user using the diagnostics interface to the server using MQTT, and is forwarded to the diagnostics tool. Again, the connection is started with the wake-up package and followed by “read DTCs” package. Unlike group reading, all DTCs are access together, there are no groups. The accessed DTCs are then sent to the server and displayed to the user. To make it easier to understand the errors, the codes are parsed by using Ross-tech forum.

Deleting the error codes is similar to reading them, only difference is the message which is sent to the user on successful or unsuccessful deletion.