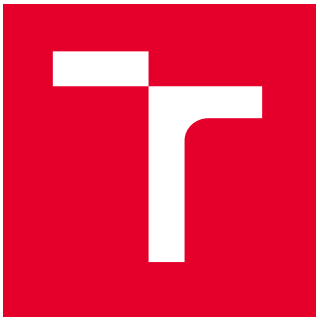

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

SECURED VEHICLE MONITORING USING OBDII AND LPWAN NETWORKS

ZABEZPEČENÝ MONITORING VOZIDEL S VYUŽITÍM OBDII PROTOKOLU A LPWAN SÍTÍ

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **Miroslav Babeľa**
AUTOR PRÁCE

SUPERVISOR — **Ing. Radek Možný**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Telecommunication and Information Systems**

Department of Telecommunications

Student: Miroslav Babeľa ***ID:*** 230530

Year of study: 3 ***Academic year:*** 2022/23

TITLE OF THESIS:

Secured vehicle monitoring using OBDII and LPWAN networks

INSTRUCTION:

In the theoretical part of bachelor's thesis, explore methods of monitoring the movement of devices via GNSS, monitoring of vehicle operating parameters (speed, fuel consumption, various temperatures, etc.), and also methods of data transmission by LPWAN (Low Power Wide Area Network).

In the practical part, design and build a device capable of monitoring given parameters and their transfer through a network, taking into account the current requirements for information security.

The result of the bachelor's thesis is a constructed hardware device capable of reading basic values from a vehicle using OBD II and showing them on an embedded display. Device also should be able to establish a communication channel to a remote server enabling the transmission of data from the device with regard to current security standards.
The output also includes the design of the system of measured values and transmitted data.

RECOMMENDED LITERATURE:

Podle pokynů vedoucího práce.

Date of project specification: 6.2.2023 ***Deadline for submission:*** 26.5.2023

Supervisor: Ing. Radek Možný

prof. Ing. Jiří Mišurec, CSc.
Chair of study program board

ABSTRACT

Cars are getting more and more complicated as both technology and government regulations evolve. This fact is of concern for individual owners as well as fleet operators as they are getting much more expensive to maintain. To reduce maintenance costs and downtime to a minimum, owners should have an easy-to-use monitoring tool that would notify them about potential problems and let them fix them before they cause more damage. This thesis aims to create a device that would be mounted inside the car and would monitor and report important engine parameters to an external server for further processing.

The theoretical part of this thesis describes current OBD II regulations and communication protocols used in the majority of vehicles currently on the road and discusses currently available low power wide networks (LPWAN) and global navigation satellite systems (GNSS). The practical part describes the design of an OBD II monitoring device with support for the most commonly used protocols in Europe, NB-IoT network, and a positioning system. The last chapter analyses the cost of a single device and its reduction for potential commercial use.

KEYWORDS

CAN BUS, GNSS, Internet of Things, K Line, KWP2000, NB-IoT, OBD, STM32, vehicle diagnostics

ABSTRAKT

Automobily sa s vývojom technológie a vládnych nariadení stávajú viac a viac komplikované. To sa však stáva problémom nie lel pre individuálnych majiteľov vozidiel ale aj pre prevádzkovateľov flotíl, kde sa im zvyšujú náklady na údržbu a prestoje. Aby majitelia vozidiel mohli znížiť svoje náklady na minimum, mali by mať nainštalované jednoduchej monitorovacie zariadenie, ktoré ich môže upozornit na potenciálne problémy a predísť tak oveľa nákladnejším opravám. Cieľom tejto práce je vytvoriť zariadenie, ktoré sa umiestni pevne vo vozidle a bude monitorovať a odosielať doležíté motorické parametre na vzdialený server pre ďalšie spracovanie.

V teoretickej časti práce sú opísané aktuálne OBD nariadenia, používané komunikačné protokoly v súčasných autách, dostupné nízkoenergetické siete pre rozsiahle územie (LPẆAN) a navigačné systémy (GNSS). Praktická časť sa zaoberá návrhom OBD II monitorovacieho zariadenia, ktoré podporuje najpoužívanejšie komunikačné protokoly v Európe, NB-IoT sieť a systém na určenie polohy. Posledná kapitola analyzuje výslednú cenu zariadenia a opisuje jej prípadnú optimalizáciu pre komerčné použitie

KĽÚČOVÉ SLOVÁ

CAN zbernica, diagnostika vozidiel, GNSS, Internet vecí, K Line, KWP2000, NB-IoT, OBD, STM32

ROZŠÍŘENÝ ABSTRAKT

Autá sa stávajú čoraz viac a viac komplikované, najmä z hľadiska obmedzovania emisií. Tento fakt spôsobuje vrásky na čele nielen individuálnym vlastníkom áut ale taktiež aj majiteľom a prevádzkovateľom väčších flotíl vozidiel, pretože sa im zvyšujú nie len náklady na údržbu ale aj zbytočné prestoje. Problému taktiež nepomáhajú aj samotný výrobcovia, ktorý v posledných rokoch začali z vozidiel odoberať rôzne dôležité budíky ako napr. teplota chladiacej kvapaliny či napätie baterky. Niektoré z nich boli nahradené kontrolkami, ktoré však často majú príliš vysokú toleranciu a rozsvietia sa až keď je neskoro a na vozidle už vznikli väčšie škody. Z týchto dôvodov vie byť prevencia veľmi zložitá nielen pre bežného používateľa ale aj mechanikov, ktorý majú na starosti údržbu flotíl.

Cieľom tejto bakalárskej práce je priniesť potenciálne riešenie tohto problému a to vo forme zariadenia, ktoré je schopné monitorovať všetky dôležité parametre motora a odosielať ich na vzdialený server pomocou nízkoenergetických sietí pre rozsiahle územie (LPWAN). Vzhľadom na to že zariadenie cieli aj na flotilových zákazníkov, malo by byť určené na pevnú montáž vo vozidle, tak aby sa zabránilo úmyselnému odpojeniu nepoctivým používateľom vozidla.

Teoretická časť obsahuje opis aktuálnych nariadení z oblasti vstavanej diagnostiky vozidla (OBD), opis najpoužívanejších komunikačných protokolov pre vnútro vozidlovú komunikáciu v európskom regióne, stručný opis dostupných polohových systémov a v neposlednej rade opis LPWAN sietí a ich protokolov. Jednotlivé témy majú svoje samostatné kapitoly, ktoré sú ďalej delené pod-témy. Prvá kapitola sa zaoberá jednotlivými časťami OBD II nariadení ako fyzické rozhranie, komunikačné protokoly, diagnostické správy a chybové kódy. Nasledujúca kapitola podrobne opisuje komunikačné protokoly používané v automobilizme, špecificky sa zameriava na protokoy ISO 9141, ISO 14230 a ISO 15765 CAN Bus.

Tretia kapitola sa venuje polohovým systémom a posledná kapitola teoretickej časti sa zaoberá samotnými LPWAN sieťami so špecifickým zameraním na Narrowband IoT (NB-IoT) siete.

Praktická časť práce bola rozdelená do dvoch nadväzujúcich častí, jedna pre semestrálnu prácu a druhú pre nadväzujúcu bakalársku prácu. Po formálnej stránke je praktická časť rozdelená na 2 časti – návrh hardwaru a návrh softwaru. V predchádzajúcej semestrálnej práci bolo zostrojené zariadenie, ktoré bolo schopné vyčítať OBD II parametre a zobraziť ich na vstavanom displeji. Táto časť práce bola taktiež prezentovaná na študentskej konferencií STUDENT EEICT 2023. Toto zariadenie zároveň poslúžilo ako základ pre odladenie programovej implementácie komunikačných protokolov ISO 9141 a ISO 14230 a implementácie samotného OBD II špecifický v móde 01 pre výčet okamžitých parametrov.

Pre implementáciu samotnej OBD II komunikácie boli preskúmané viaceré rieše-

nia, vrátane už existujúcich OBD II to UART prevodníkov. Tie boli z dôvodu zlej dostupnosti nakoniec zavrhnuté a bolo zvolené riešenie pomocou bežného mikrokontroléra. Ako základ bol zvolený mikrokontrolér STM32L431, najmä pre integrovaný CAN bus kontrolér. Ten bol doplnený o vysielače a prijímače pre jednotlivé komunikačné protokoly a v prvom zariadení aj o OLED (Organic Light-Emitting Diodes) displej na zobrazovanie nameraných hodnôt.

V nasledujúcej bakalárskej práci bol OLED displej nahradený modemom pre komunikáciu cez sieť NB-IoT – Quectel BG77. Sieť NB-IoT bola zvolená na základe jej dobrého pokrytia a dostatočnej prenosovej kapacity. Piata kapitola podrobne opisuje návrh tohto zariadenia, vrátane návrhu schém, PCB dosky aj návrhu krabičky. Modem Quectel BG77 taktiež obsahuje aj navigačný (GNSS) modul pre zisťovanie jeho polohy. Vzhľadom na finálne umiestnenie zariadenia vo vozidle, zariadenie vyžaduje na svoj optimálny chod inštaláciu aktívnej GNSS antény tak, aby mala čo najlepší výhľad na oblohu. Samotné umiestenie musí byt riešené individuálne a to z dôvodu rozličného usporiadania vnútorného priestoru vo vozidle. Rovnako musí byt pripravený aj káblový zväzok na pripojenie zariadenia k vozidlu.

Po softwarovej stránke, monitorovacie zariadenia nadväzuje na svojho predchodcu a k existujúcej implementácii OBD II pridáva nový ovládač pre modul BG77, ktorý bol inšpirovaný podobným už existujúcim ovládačom pre arduino. Šiesta kapitola podrobne opisuje fungovanie zariadenia po softwarovej stránke, vrátane flow dia gramu. Získané dáta z vozidla sú prenášané pomocou MQTT protokolu cez sieť NB-IoT na MQTT broker. Ten dáta presmeruje do NodeRED serveru, kde prebieha preklad číselných PID hodnôt na ich zrozumiteľnejšiu variantu. Takto upravené dáta sú následne uložené v InfluxDB databáze pre neskoršiu analýzu. Obe zariadenia boli intenzívne testované na viacerých vozidlách ako Renault Megane či BMW 320d E91 a sú plne funkčné.

Aktuálne najväčšia nevýhoda navrhnutého zariadenia je obmedzená podpora OBD II módov, kde zariadenie podporuje len mód 01 čo je výčet okamžitých hodnôt. Aj najpriek tomu je možné tvrdiť že zariadnie ako koncept má komerčný potenciál a môže byť v budúcnosti rozšírené o ďalšie OBD II módy, samostatnú serverovú apli káciu či predikciu potenciálnych závad. Zo stránky zadania, zariadenie spĺňa dané požiadavky a teda zadanie môže byť považované za splnené.

BABEĽA, Miroslav. *Secured vehicle monitoring using OBDII and LPWAN networks*. Brno: Brno University of Technology, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 79 p. Bachelor's Thesis. Advised by Ing. Radek Možný

Author's Declaration

Author:	Miroslav BABEĽA
Author's ID:	230530
Paper type:	Bachelor's Thesis
Academic year:	2022/23
Topic:	Secured vehicle monitoring using OBDII and LPWAN networks

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the supervisor of this thesis Ing. Radek Možný for consultations and valuable feedback on the thesis. I would also like to thank Ing. et Ing. Petr Musil for the great suggestions involving this thesis and also for lending his car for development.

Contents

Introduction **15**

1 Car diagnostics **16**
1.1 History . . . 16
1.2 OBD II/EOBD . . . 17
1.2.1 Physical interface . . . 17
1.2.2 Communication protocols . . . 17
1.2.3 Diagnostic messages and services . . . 18
1.2.4 Diagnostic trouble codes . . . 20

2 Communication protocols in cars **21**
2.1 ISO 9141 . . . 21
2.1.1 Physical layer . . . 21
2.1.2 Initialization . . . 22
2.1.3 Data messages . . . 22
2.2 ISO 14230 . . . 23
2.2.1 Physical layer . . . 24
2.2.2 Initialization . . . 24
2.2.3 Data transmission . . . 25
2.3 CAN BUS . . . 26
2.3.1 Physical layer . . . 26
2.3.2 Data transfer . . . 27
2.3.3 ISO 15765-4 . . . 29

3 Global navigation satellite systems **30**
3.1 GPS . . . 30
3.2 Galileo . . . 30
3.3 GLONASS . . . 30
3.4 BeiDou . . . 31

4 Low power wide area network **32**
4.1 Sigfox . . . 32
4.2 LoRaWAN . . . 32
4.3 LTE cat M1 . . . 32
4.4 NB-IoT . . . 33
4.4.1 Network architecture . . . 33
4.4.2 Operation modes . . . 34
4.4.3 Power saving measures . . . 34

4.5 Communication protocols used with LPWAN . . . 35
4.5.1 User Datagram Protocol . . . 36
4.5.2 Transmission Control Protocol . . . 36
4.5.3 Message Queue Telemetry Transport Protocol . . . 36
4.5.4 Constrained Application Protocol . . . 36

5 Hardware **37**
5.1 Micro controller . . . 37
5.2 CAN BUS Transceiver . . . 38
5.3 K Line Transceiver . . . 39
5.4 Power supplies . . . 40
5.5 Narrowband IoT module . . . 41
5.5.1 Main module . . . 41
5.5.2 UART level shifter . . . 42
5.5.3 Antennas . . . 42
5.5.4 SIM card . . . 43
5.5.5 USB . . . 43
5.6 PCB Design . . . 44
5.6.1 PCB . . . 45
5.6.2 Micro controller layout . . . 45
5.6.3 Power supply layout . . . 45
5.6.4 CAN and K Line transceivers layout . . . 46
5.6.5 NB-IoT module layout . . . 46
5.6.6 Peripherals & Connectors layout . . . 47
5.7 Manufacturing . . . 47
5.8 Case Design . . . 48

6 Software **50**
6.1 Main architecture . . . 50
6.2 K line and KWP2000 implementation . . . 52
6.3 CAN BUS implementation . . . 52
6.4 OBD II Parsing . . . 53
6.5 Quectel BG77 driver implementation . . . 53
6.6 Data processing . . . 54

7 Testing **55**

8 Cost analysis **58**

Conclusion **59**

Bibliography **60**

Symbols and abbreviations **66**

List of appendices **69**

A OBD II related tables **70**

A.1 List of OBD II CAN Bus identifiers 70

B Board schematic **71**

C Contents of electronic attachment **79**

List of Figures

1.1 Female OBD II connector according to SAE J1962 17
1.2 Diagnostic message format for ISO 9141-2, ISO 14230-4 and SAE J1850 19
2.1 Simplified wiring scheme for ISO9141 communication 21
2.2 Initialization sequence ISO 9141 22
2.3 Intermessage and interbyte timing ISO 9141 23
2.4 5 Baud initialization sequence ISO 14230 25
2.5 Fast initialization sequence ISO 14230 25
2.6 Message structure for ISO 14230 26
2.7 Message flow and timings ISO 14230 26
2.8 Simplified wiring diagram for ISO 11898 CAN Bus 27
2.9 CAN bus waveform 28
2.10 Structure of CAN data frame 28
2.11 Structure of CAN remote frame 29
2.12 Termination schematic for diagnostic tool 29
4.1 NB-IoT network architecture 33
4.2 NB-IoT operation modes 34
4.3 ife cycle 35
5.1 Schematic for STM32L431 38
5.2 Schematic for CAN bus transceiver 39
5.3 Schematic for K line transceiver 40
5.4 Power supply schematic 41
5.5 Voltage level translator schematic 42
5.6 NB-IoT and GNSS antennas schematic 43
5.7 SIM card slot schematic 44
5.8 USB interface schematic 44
5.9 PCB layout 47
5.10 Finished board mounted inside the case 48
5.11 Final case design 49
5.12 Finished product 49
6.1 Software flowchart 51
6.2 Data processing flow in NodeRed 54
7.1 Measurement from testdrive on 21.5.2023 – engine's RPM and vehicle's speed in time 55
7.2 Measurement from testdrive on 21.5.2023 – coolant temperature and vehicle's speed in time 56
7.3 OBD II diagnostic tool – testing in BMW E91 56
7.4 OBD II monitoring device – testing in Renault Megane 57

7.5 OBD II monitoring device – testing in Renault Megane – closeup . . . 57
B.1 Schematic – Main MCU . . . 71
B.2 Schematic – OBD II interface . . . 72
B.3 Schematic – Power supplies . . . 73
B.4 Schematic – NB-IoT module . . . 74
B.5 Schematic – Antenna interface . . . 75
B.6 Schematic – SIM card interface . . . 76
B.7 Schematic – USB interface . . . 77
B.8 Schematic – Connectors . . . 78

List of Tables

1.1 List of SAE J1979 defined diagnostic services 19
1.2 Diagnostic message format for ISO 15765-4 19
1.3 General DTC specification 20
2.1 Timing values for ISO 9141 23
2.2 Timings for ISO 14230 26
8.1 Cost summary for single unit 58
A.1 11 bit CAN identifiers defined by ISO 15765 70

Introduction

In recent years, the complexity of cars significantly increased. Newer cars are safer, have more gadgets inside them and have far better efficiency and therefore they release a lot fewer pollutants. Sometimes, these changes are pushed via government regulations. This is especially true for emission-related improvements, which are forced on the manufacturers by emission norms such as Euro 6 or Euro 7.[1]

In order to comply with the ever-increasing requirements on emission reduction, car manufacturers have to be creative and produce more and more complex cars. While these norms are beneficial to the general public, they can be problematic for individual or fleet owners, due to the added complexity. For the emissions-related systems to operate properly, the rest of the engine has to be in good condition. And as it is usual with anything mechanical and electronic, the question is not if it breaks, but when it breaks. To detect potential faults, the driver of the vehicle should keep an eye on basic engine parameters such as coolant temperature or fuel consumption. This used to be an easy task on older vehicles, but the trend today amongst manufacturers is to reduce the amount of information directly available to the driver for the sake of simplicity. In combination with certain weak points in the engine, this can lead to expensive repairs down the line if not detected early.

The main goal of this bachelor's thesis is to create a device that would be able to gather important engine parameters and send them to a remote server via Low Power Wide Area networks (LPWAN) for further processing. It also should be able to report its location to the same server via Global Navigation Satellite Systems (GNSS). The device itself should be relatively compact for it to fit inside the dashboard and it also should be designed in a way that would limit its power consumption while the car is not running

The theoretical part of this thesis describes OBD II requirements, communication protocols used in cars and the most popular LPWAN networks. It also briefly discusses the GNS systems.

The practical part consists of separate chapters for hardware, software, testing and cost analysis of the final product. The hardware chapter describes in detail the parts selection and the PCB design itself. The software chapter describes the main software architecture and focuses on its main blocks in detail. The testing and cost analysis chapters discuss the testing of the final product, discuss its cost efficiency and possible future improvements.

1 Car diagnostics

Cars are complex machines and have a lot of parameters like coolant temperature or battery voltage that need to be monitored for their trouble-free operation.

1.1 History

In the early days of cars, this has been accomplished by analog gauges inside of the car, usually on the dashboard. Cars mainly had gauges for speed, fuel level, battery voltage, and oil pressure. The more luxurious ones also had a rpm gauge. Everything else could have been monitored by drivers' or mechanics' instincts. This was perfectly fine, but engines from that era weren't that efficient.[2]

The invention of transistors helped to change that. Its usage in engine management allowed greater control over the engine's parameters and therefore allowed to achieve higher efficiency. The first car manufacturer to implement electronic fuel injection was Volkswagen with its Type 3 in 1968. These systems were developed further during the 70s and eventually evolved into digital engine management systems as we know them today. With the added complexity, these new engines became far more difficult to monitor and diagnose if something went wrong. This led to the creation of electronic diagnostic tools that were able to monitor more engine values and check if there are any faults with the engine.[2]

First onboard diagnostic systems used blinking LEDs or built-in displays to show error codes. Later, in the 80s, GM came up with the first onboard diagnostic system that could be connected to a computer via serial port - Assembly Line Data Link or ALDL for short. Other manufacturers soon followed their lead and came up with their diagnostic tools. [2]

To improve their vehicle inspections and combat growing pollution, the California Air Resources Board, in 1988, announced a new regulation that required new vehicles sold in California to have some sort of basic OBD capability. It is generally referred to as OBD I. The Society of Automotive Engineers (SAE) recommended a standardized connector and a set of standardized signals, but this wasn't included in the new regulation. During this era, the state of affairs was undesirable, because mechanics outside of dealer networks were forced to keep multiple sets of diagnostic tools just to be able to repair all kinds of cars. This fact combined with CARBses desire to further improve emission tests led to the creation of OBD II in 1992. In this version, recommendations from SAE were implemented and both connector and test signals became standardized. OBD II regulation was harmonized with new Federal OBD regulation from Environmental Protection Agency in 1993 and started to roll out for the model year 1994 and 1995.[3, 4]

1.2 OBD II/EOBD

OBD II is the second generation of governmental requirements concerning onboard diagnostics in vehicles. It was introduced in 1994 and it became mandatory for vehicles sold in the USA in 1996. European Union introduced similar requirements, called EOBD, in 1998 with them coming into effect in the year 2000 for petrol vehicles and 2003 for diesel ones. This requirement specifies the physical interface and its location, the car's communication protocols, specific diagnostic messages, diagnostic services, and fault codes.[4, 5]

1.2.1 Physical interface

The interface is defined by ISO 15031-3 / SAE J1962 standard. It uses 16 pin connector with a female part installed in the vehicle, which should be mounted in the driver's or passenger's side compartment and easily accessible from the driver's seat. As it can be seen in Figure 1.1, 3 pins are used for powering devices connected, and 2 pins are used for communication with the car, depending on the bus that's being utilized. The rest of the pins are left to the discretion of the manufacturer and are usually used for connecting to noncritical systems such as infotainment or climate control. [6]

1.2.2 Communication protocols

On 1st and 2nd layer of the ISO/OSI model, OBD-II supports multiple protocols such as SAE J1850, ISO 9141-2(K-Line), ISO 14230-4(KWP 2000), and ISO 15765-4 (DoCAN). The latter became mandatory in the US in 2008.[7]

SAE J1850 has two different versions which differ in the first layer of the ISO/OSI model. The first one uses variable pulse width modulation (VPW) to encode bits and it transfers them at 10.4 kbps. It is implemented by using a single wire. This

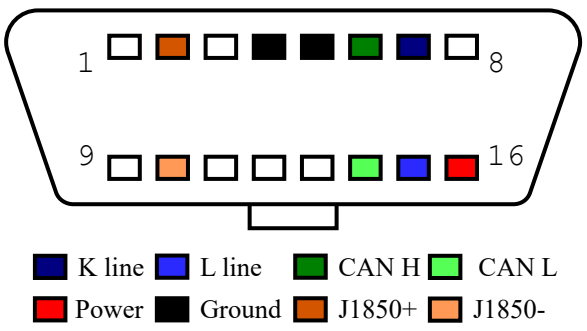

Fig. 1.1: Female OBD II connector according to SAE J1962.[8, 9]

version of SAE J1850 was mainly used by GM before 2008. The second one uses pulse width modulation(PWM) and it can transfer bits at 41.6 kbps. It uses dual wire network and it was mainly used on Ford vehicles.[10]

ISO 9141-2 or K-Line is one of the oldest protocols used by OBD-II, being that it was last updated in 1994. It uses two wires, where the first one, K Line, is bidirectional and is being used for communication between ECU and diagnostic tool, and the second one, L Line, is only unidirectional and is being used for establishing a connection with the ECU on the K-Line. This protocol is based on UART signaling and it has a data rate of 10.4 kbps. Amongst the most frequent users were mainly European and Asian manufacturers.[11]

ISO 14230-4 or Keyword Protocol 2000 covers the application and session layer of the ISO/OSI model and it shares the first and second layers with ISO 9141 protocol. It requires a special initialization sequence on L Line for the diagnostic tool to differentiate between it and the older ISO 9141 protocol. It can be also implemented on CAN Bus. Main users of KWP2000 were European and Asian car makers.[12]

ISO 15765 or DoCAN specifies onboard diagnostics over CAN Bus - ISO 11898. It uses twisted pair CAN H and CAN L to communicate with the ECU and it can transfer data at up to 1 Mbps. DoCAN became a de facto standard for all manufacturers in the 2010s.[12]

1.2.3 Diagnostic messages and services

SAE J1979 describes 10 diagnostic services, in which diagnostic tools can operate. Summary of those services can be seen in Table 1.1. These are recommended services and vehicle manufacturers are not required to implement all of them. Each service has a number of functions which either provide information to the diagnostic tool or perform certain tasks such as erasing Diagnostic Trouble Codes. Functions, that return values are reffered to as OBD II Parameter IDs or OBD II PID for short. List of these PIDs can be seen in {[14]}.[13]

SAE J1979 also defines diagnostic messages to be used in communication with vehicle's ECUs and diagnostic tools. The diagnostic message format for ISO9141-2, ISO14230-4 and SAE J1850 can be seen in Figure 1.2.[13]

The first 3 bytes, in all messages, contain header information. The first byte is different for each of those 3 protocols, in SAE J1850 and ISO 9141-2 it depends on the bit rate and the type of message. The second byte has a target address, that differs between requests and responses. In ISO 14230-4, the first byte indicates the addressing mode and the length of the data field. The second byte is the address of the receiver of the message. The third byte is the same for all of them and it has

Tab. 1.1: List of SAE J1979 defined diagnostic services.[13]

Mode [hex]	**Description**
01	Request Current Powertrain Diagnostic Data
02	Request Powertrain Freeze Frame Data
03	Request Diagnostic Trouble Codes (DTC)
04	Clear/Reset Diagnostic Information
05	Request Oxygen Sensor Monitoring Test Results
06	Request On-Board Monitoring Test Results for Specific Systems
07	Request DTCs Detected During Current or Last Driving Cycle
08	Request Control of On-Board System, Test or Component
09	Request Vehicle Information
0A	Request Permanent Diagnostic Trouble Codes

Tab. 1.2: Diagnostic message format for ISO 15765-4.[14]

Header	Data Frame							
CAN Identifier	#1	#2	#3	#4	#5	#6	#7	#8

the address of the sender of the message. The data bytes portion of a message is also the same for all of them, the first byte in it specifies diagnostic service and the rest of the bytes depends on the service selected. At the end of the message there are 2 places for non-data bytes, the first one of them called "ERR" is used for the checksum of the message. The second one, called RESP or In Frame Response, only applies to SAE J1850, where it is required to be set when using a bit rate of 41.6 kbps.[13]

On the other hand, ISO 15765-4 uses a slightly different message structure which can be seen in Table 1.2. The header only contains the CAN identifier which is used for addressing. The data portion of the frame is also slightly different, the first byte carries the length of used bytes in the data bytes. The next byte contains the diagnostic service identifier and the rest of the data portion is varied based on the specified diagnostic service.[13, 14]

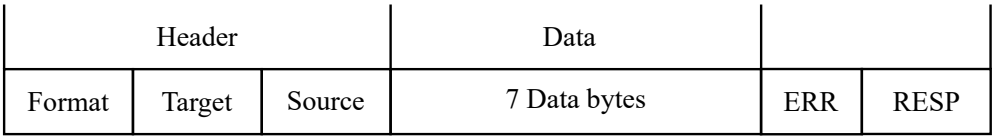

Fig. 1.2: Diagnostic message format for ISO 9141-2, ISO 14230-4 and SAE J1850.

1.2.4 Diagnostic trouble codes

Another part of OBD requirements is diagnostic trouble codes, which are triggered when a problem arises in a vehicle's ECU. These trouble codes are specified by ISO15031-6/SAE J2012 standard. These codes are generally split into 4 groups, depending on ECU which has generated them and their summary can be seen in Table 1.3. Each group is divided into 4 categories depending on the type of the fault – General open circuit, Range/Performance, Circuit Low, and Circuit High.[15]

DTCs are represented by 2 bytes in the data portion of the diagnostic message, right after the diagnostic mode byte. Their general structure can be seen in Figure 1.1. DTC begins with an alphanumeric designator, where the letter stands for the system from which the trouble code originated and the number dictates whether the code is ISO/SAE controller or is controlled by the manufacturer. This designator is followed by a 3-digit number, where the first digit is used to indicate an area of the problem. The amount of manufacturer-controlled DTC varies for each system. For example, in powertrain codes only `P1xxx` are intended for manufacturers' usage, others are solely ISO/SAE controlled or are split between the manufacturer and ISO/SAE. In chassis codes, these DTCs are split evenly between ISO/SAE and manufacturers.[15]

Tab. 1.3: General DTC specification.[15]

System	**Code**	**Hex value**	**Prefix**
Body	B0xxx – B3xxx	8xxx – Bxxx	B
Chassis	C0xxx – C3xxx	4xxx – 7xxx	C
Powertrain	P0xxx – P3xxx	0xxx – 3xxx	P
Network	U0xxx – U3xxx	Cxxx – Fxxx	U

2 Communication protocols in cars

Communication between parts of the car evolved drastically over the last 40 years. It went from using separate wires for each signal to using various buses to reduce the complexity and cost of wiring harnesses. Nowadays there are multiple different protocols used in modern cars such as ISO9141 or ISO11898.

2.1 ISO 9141

ISO 9141 or K-Line protocol is one of the oldest protocols still used in modern cars. It was created in order to unify the interchange of diagnostic information for onboard diagnostics. The first version of ISO9141 was introduced in 1989 and its later revision became part of OBD II requirements. ISO9141 is defined on the first and second layers of the ISO/OSI model.[16]

2.1.1 Physical layer

ISO9141 uses serial communication and it can operate with single-wire (K Line) or two-wire interfaces (K/L Line) connected through SAE J1962 compliant connector. K line is a bidirectional line, which is used both for initialization phase and consecutive data transfer between ECU and diagnostic tool. L line is a unidirectional line, which is only used during initialization phase in order to notify ECU of the tool's address. In all other cases, L line is idle with logic 1 state. Simplified wiring diagram can be seen in Figure 2.1. In physical layer, ISO9141 uses Non-Return-to Zero (NRZ) coding at 10.4 kbps with car's battery voltage used as logic voltage[16]

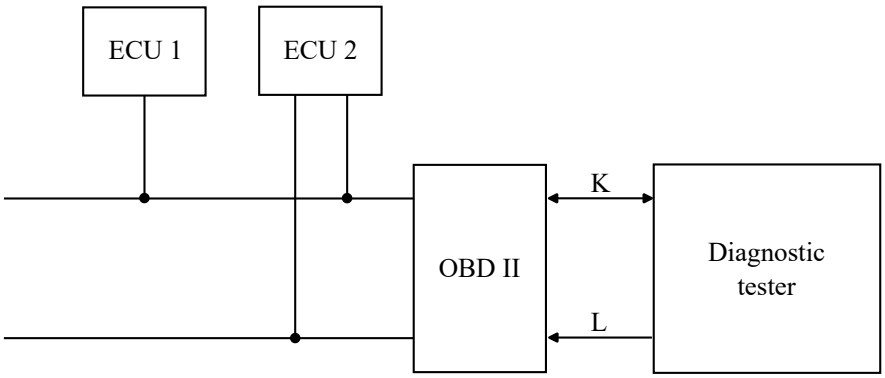

Fig. 2.1: Simplified wiring scheme for ISO9141 communication.[16]

2.1.2 Initialization

For successful communication between ECU and diagnostic tool, ISO 9141 requires a 5 bps initialization sequence. During this initialization, both the tool and ECU exchange their addresses, which are needed for subsequent communication, and agree on the bus speed. Default tester address for OBD II communication is 33_{H}.[16]

Before the initialization starts, the K-line should idle high for at least 2 ms. Initialization begins from the tester's side with the transmission of the tester address at 5 baud After at least 60 ms, but no more than 300 ms, ECU responds with a speed synchronization pattern with the value of 55_{H}. After the last bit, K-line will stay high for a time period no longer than 20 ms, in order to let the diagnostic tool reconfigure its settings.[16]

Speed synchronization is then followed by two keywords in order to inform diagnostic tool about the subsequent serial communication and hardware configuration of the diagnostic lines. Every keyword contains a start bit with logic 0, 7 bits where the LSB is sent first, odd logic 1 parity bit, and a stop bit with logic 1. Both keywords have the same value and for OBD II communication it is either 08_{H} or 94_{H}.[16]

After receiving the last keyword, the diagnostic tool will respond with the inverted value of the last keyword. ECU responds to this transmission with an inverted initialization address. This exchange is used for hand-shaking purposes. If initialization fails, the diagnostic tool has to wait for at least 300 ms before trying again. The whole initialization sequence with timings can be seen in Figure 2.2 and in Table 2.1.[16]

2.1.3 Data messages

ISO 9141 only applies to diagnostic messages defined in SAE J1979, however, other types of messages are allowed if they aren't interfering with diagnostic messages. OBD II diagnostic messages usually have a length of between 5 and 11 bytes and their

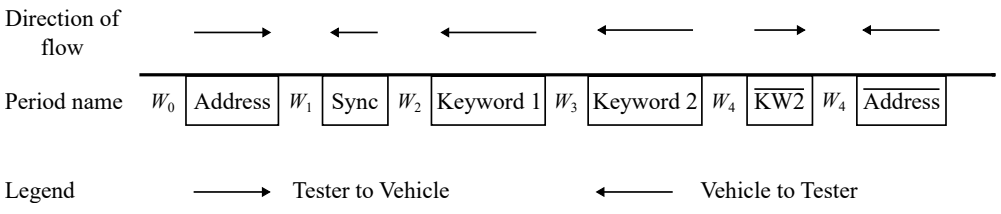

Fig. 2.2: Initialization sequence ISO 9141.[16]

Tab. 2.1: Timing values for ISO 9141.[16]

Symbol	Time [ms]		Symbol	Time [ms]	
	min	max		min	max
W_0	2	∞	P_1	0	20
W_1	60	300	P_2 (94)	0	50
W_2	5	20	P_2 (08)	25	50
W_3	0	20	P_3	55	5000
W_4	25	50	P_4	5	20

description can be seen in Section 1.2.3. Each message also contains a checksum, which is defined as the simple 8-bit sum of all the message bytes, excluding the checksum.[16]

Data transmission begins with the start bit as logic 0, followed by the data portion of the messages, where the LSB is sent first and ends with the stop bit as logic 1. Message and request timing scheme can be seen in Figure 2.3. The intermessage period depends on the type of message synchronization. If it is achieved by timing only, then the period should be at least 25 ms, and keywords 08_{H} should be used. Otherwise, the period should be 0 and the used keywords should be 94_{H}.[16]

ISO 9141 also defines a number of errors and their handling. In case of incorrect checksum detected by the vehicle, the request is ignored and ECU waits for retransmission by the tester. If the incorrect checksum is detected by the tester, it will send a new request after at least 55 ms from the last response. Incorrect message structure detected by either the vehicle or the tester is handled in the same way as the incorrect checksum.[16]

2.2 ISO 14230

ISO 14230 or Keyword Protocol 2000 can be seen as an improved version of the previously mentioned ISO 9141. It allows enhanced vehicle diagnostic outside the

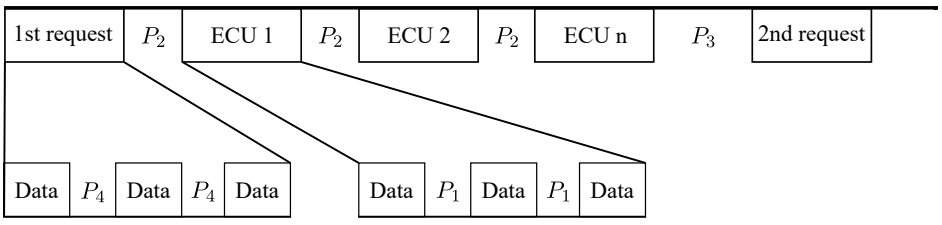

Fig. 2.3: Intermessage and interbyte timing ISO 9141.[16]

scope of OBD II and can also be used for in-vehicle networks. This standard was first published in 1998 and therefore is one of the newest standards still used in modern vehicles until recently. As opposed to ISO 9141, KWP2000 is defined on the first, seconds, and the seventh layer of the ISO/OSI model.[17, 18]

2.2.1 Physical layer

The first layer of the OSI model in KWP2000 is the same as ISO 9141. It primarily uses the K-line for both data transmission and communication initialization. Usage of the L-line is optional and it is only used for the initialization sequence. The wiring scheme can be seen in Figure 2.1. Detailed description of physical layer can be seen in Section 2.1.1.[18]

2.2.2 Initialization

In order to establish and maintain communication between the diagnostic tester and ECU, initialization of the serial bus is needed. For this task, ISO 14230 defines a special `StartCommunication` service, where the tester sends an indication, that lets ECU know what kind of initialization procedure will follow. ECU after receiving this indication, will proceed to check if a connection can be established with the selected procedure, configure its parameters if needed and send a `StartCommunication` response. If communication cannot be initialized, ECU simply won't respond and it will wait for another indication.[17]

ISO 14230 offers 3 types of initialization, 5 baud CARB initialization which is similar to ISO 9141, 5 baud or Slow initialization, and a Fast initialization at 10.4 kbps. [17]

The 5 baud initialization begins with the tester transmitting its address on both K and L lines. After receiving the last address bit, ECU will for at least 60 ms before transmitting sync pattern 55_{H} and two key bytes, which specify communication baud rate and other parameters. The tester will respond with inverted key byte 2 after waiting for at least 25 ms. To this, the ECU responds with an inverted address byte and the communication will start. The whole sequence can be seen in Figure 2.4.[17]

The Fast initialization starts with the tester transmitting a Wake-up Pattern on both K and L lines. This pattern lasts for 50 ms with 25 ms of logic 0 at the beginning after at least 300 ms of idle time on the K-line. The Wake up Pattern is immediately followed by `StartCommunication` request and ECUs response, which has information about supported communication parameters within key bytes. The whole sequence can be seen in Figure 2.5. With the Fast initialization, all ECUs have to use a baud rate of 10400 baud for both initialization and subsequent communication.[17]

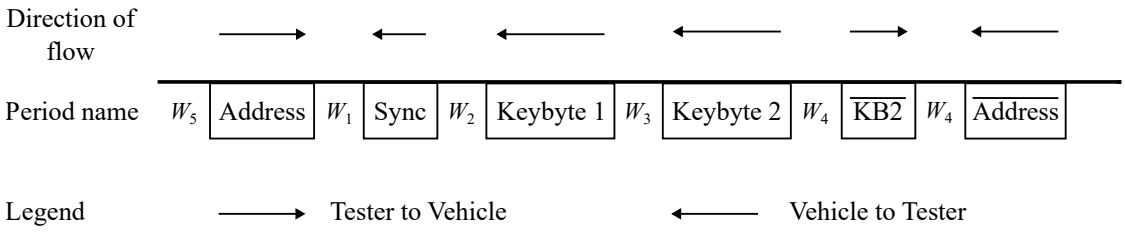

Fig. 2.4: 5 Baud initialization sequence ISO 14230.

2.2.3 Data transmission

All KWP2000 messages contain Header bytes with a maximum length of 4 bytes, data bytes with a maximum length of 255 bytes, and a checksum at the end. Message structure can be seen in Figure 2.6.[17]

The first header byte is called the format byte and it contains the addressing type and data field length for messages shorter than 64 bytes. Format byte is followed by target and source address bytes. For OBD II related communication, address 33_{H} is used for the request to the vehicle. The header ends with an optional length byte for messages longer than 64 bytes. The subsequent data portion of the message contains the Service Identification byte in the first data byte. The message ends with a checksum byte, which is defined as a simple 8-bit sum of all the bytes in the message, excluding the checksum byte. Message flow and timings can be seen in Figure 2.7 and in Table 2.2.[17]

ISO 14230 also defines a number of possible errors and their handling. In case of `StartCommunication` service error detected by the tester side, the tester will wait for at least 300 ms before transmitting a new request. If an ECU detects an error, then it will simply ignore it and wait for another one. [17, 18, 19]

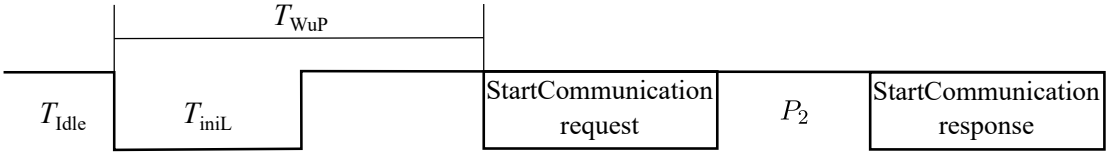

Fig. 2.5: Fast initialization sequence ISO 14230.[17]

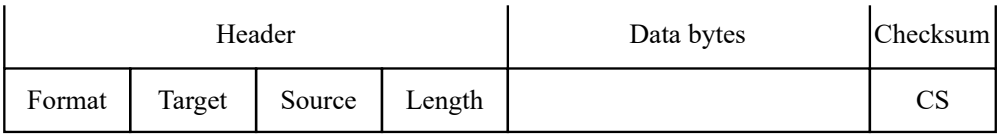

Fig. 2.6: Message structure for ISO 14230.[17]

Tab. 2.2: Timings for ISO 14230.[17]

Symbol	Time [ms]	
	min	max
P_1	0	20
P_2	25	50
P_3	55	5000
P_4	5	20

2.3 CAN BUS

CAN or Controlled area network protocol was invented by Robert Bosch GmbH. in 1986 with the intention of creating a bus, on which micro-controllers and devices can communicate with each other, without the need for a centralized computer. The latest version of the CAN protocol, CAN2.0, was published in 1991 and became standardized in 1993 as ISO 11898.

ISO 11898 specifies 1st and 2nd layers of the ISO/OSI model for both in-car communications. Implementation of OBD communication is described in ISO 15765 which is based on ISO 11898 but also specifies 3rd layer of the ISO/OSI model.[20, 21, 22]

2.3.1 Physical layer

CAN protocol uses a two-wire differential bus for connection to every node on the network. The bus utilizes twisted pair cabling which is terminated on both ends with

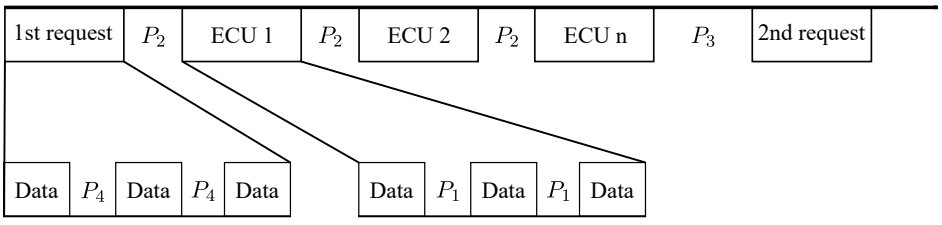

Fig. 2.7: Message flow and timings ISO 14230.

120 Ω impedance.[23] Wires in the pair are labeled CAN High and CAN Low. Each node on the network has its own CAN controller and transceiver as it is a multi-master bus – every node can initiate communication. Simplified wiring scheme can be seen in Figure 2.8.[24]

CAN messages are encoded using NRZ coding and the bus itself has two logical states: recessive and dominant. In a recessive state, all CAN nodes on the bus are turned off and the bus voltage is generated by the bus termination and internal resistance of each node's receiver. In the dominant state, at least one node sends a bit to the bus, which induces current flow through termination and creates differential voltage between the wires of the pair. CAN transceivers are usually implemented as a wired-AND gate, which represents the recessive state as logic 1 and the dominant state as logic 0. The signal waveform can be seen in Figure 2.9. CAN2.0 bus can transfer data at up to 1 Mbps. [24, 25]

2.3.2 Data transfer

In CAN networks, each node receives all of the PDUs sent by other nodes. ISO 11898 refers to these PDUs as frames and defines four different types: data frame, remote frame, error frame and overload frame.[20, 25]

Data frame

A data frame transfers data from the transmitter to all receivers on the bus. It begins with *START OF FRAME* bit field, which consists of a single dominant bit. This bit is used for the synchronization of all nodes on the bus.

Next is *ARBITRATION FIELD* which has *IDENTIFIER* and *RTR* bit. In CAN2.0 the identifier can have a length of 11 or 29 bits depending on the format of the frame – Standard or Extended. The standard format is usually used in cars and light-duty vehicles, while the extended format is usually used in heavy-duty vehicles

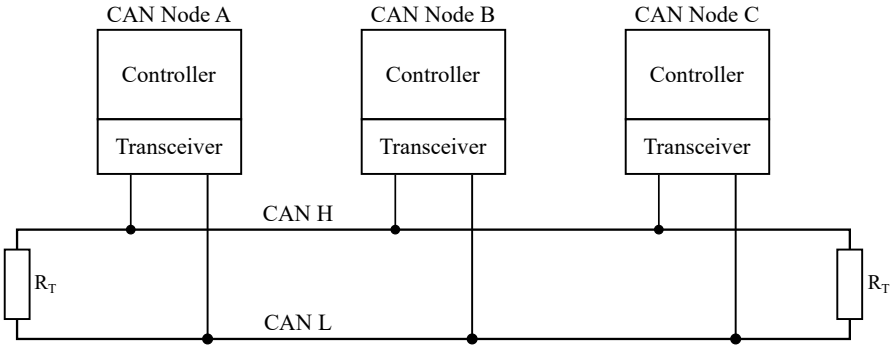

Fig. 2.8: Simplified wiring diagram for ISO 11898 CAN Bus.

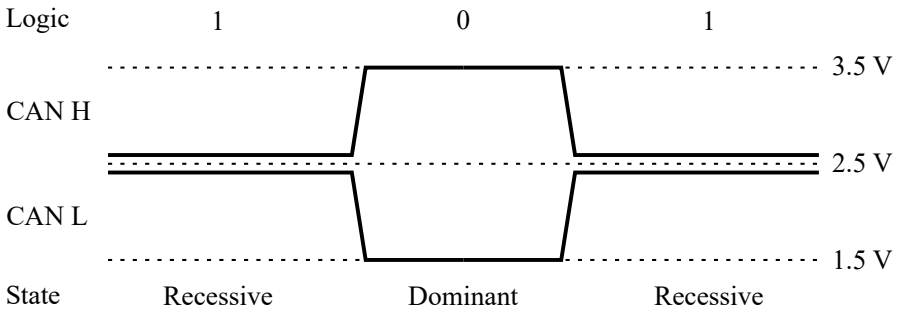

Fig. 2.9: CAN bus waveform.

SoF	Arbitration	Control	Data	CRC	ACK	EoF

Fig. 2.10: Structure of CAN data frame.[25]

like trucks and buses. *RTR* bit specifies if the frame is a data frame or remote frame and in a data frame it is always dominant.

The whole arbitration field is used for collision avoidance on the bus and prioritization of messages. The transmitter should only begin data transfer after the bus has been idle. If two frames are sent at the same time, their transmitters go into arbitration, where they compare the number of dominant bits in the arbitration field. The transmitter with the most dominant bits wins and proceeds to transmit the rest of the message. The loser discards its message and waits for another idle time on the bus. This method of arbitration means that the node with the smallest identifier has the most priority on the bus.[20]

The arbitration field is followed by *CONTROL FIELD*, which consists of four data length code bits and 2 reserved bits. In standard format one of the reserved bits is *IDE* bit, which defines the type of the format used by the frame. The extended format has this bit in the arbitration field.[25]

After the control field, follows the *DATA FIELD* which can contain up to 8 bytes of data. This field is then followed by *CRC FIELD* which has *CRC SEQUENCE* and *CRC DELIMITER*. The CRC sequence contains the CRC value for the previous fields. The CRC delimiter contains a single recessive bit in order to recognize the end of the CRC sequence.[20, 24]

The *DATA FRAME* ends with *ACK FIELD* and *END OF FRAME*. The *ACK FIELD* is used by receivers to report the successful reception of a valid message to the transmitter. The *END OF FRAME* consists of seven recessive bits and is used to indicate the end of the message. The whole data frame structure can be seen in Figure 2.10.[25]

Fig. 2.11: Structure of CAN remote frame.[25]

Remote frame

A remote frame is used to request the transmission of the data frame with the same identifier. The frame structure is similar to the data frame, with the exception of *DATA FIELD* which is not present in remote frames. The remote frame also has a recessive bit in *RTR* bit, which means that they have a lower priority than the data frames. The remote frame structure can be seen in Figure 2.11.[20]

2.3.3 ISO 15765-4

Apart from in-vehicle communication, the CAN bus can also be used for diagnostic communication between the vehicle and the diagnostic tester. Legislated OBD communication over CAN bus is described by ISO 15765.

On the physical layer, the diagnostic tool has to support only two bit rates: 250 kbps and 500 kbps. The diagnostic tool should also support both standard and extended formats of a CAN frame, which means that 4 initialization sequences should be implemented. The wiring diagram for the diagnostic tool also differs from normal CAN bus nodes. The only allowed termination on the tool's end should be an AC termination in order to reduce reflection to the bus. The usual terminating resistor between CAN H and CAN L can not be installed. A simplified wiring diagram for the tools termination can be seen in Figure 2.12.[26]

The second layer of the ISO/OSI model is the same as it is defined in ISO 11898, but ISO 15765 provides a list of CAN identifiers for use with OBD requests and responses. List of these identifiers can be seen in Table A.1.[22]

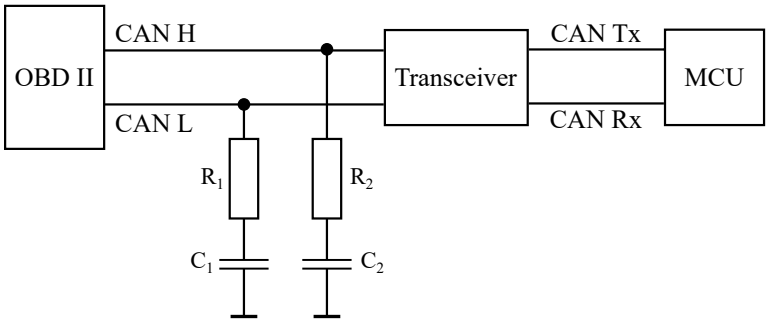

Fig. 2.12: Termination schematic for diagnostic tool.

3 Global navigation satellite systems

Ever since the introduction of global navigation satellite systems to the public, tracking the precise position of objects became a lot easier than before. Nowadays, there are multiple different GNS systems, which provide excellent coverage around the world.

3.1 GPS

GPS or Global Positioning System is the oldest and still the most utilized GNS system in the world. The initial idea of creating a three-dimensional position determination system came from the U.S. government in the 1960s. The first satellite was launched in 1978 and the system became fully functional in 1993.[27]

Currently, GPS uses 29 satellites that use two frequencies – L1 and L2.[28] The GPS is split into two services – Standard positioning service (SPS) and Precise positioning service (PPS). The L1 frequency is used in SPS, while PPS uses both L1 and L2 frequencies.[29]

3.2 Galileo

Galileo is a European GNSS, developed in order to become independent from other GNSS. It went live in 2016. Currently, Galileo uses 26 satellites with the goal of using 30 in the near future. Unlike other GNSS, Galileo is primarily a public GNSS.[30]

Galileo system provides these services [31]:

- Open Service – public GNSS
- High Accuracy Service – complementary to the Open Service
- Public Regulated Service – restricted GNSS for government authorised users only
- Search and Rescue Service – part of COSPAS-SARSAT – distress alert detection system

3.3 GLONASS

GLONASS or Global navigation satellite system, is formerly Soviet, now Russian response to GPS. The program began in the mid-1970s and the first satellite was launched in 1982. The GLONASS was proclaimed operational in 1993.[29, 30]

In function, GLONASS is similar to GPS, however, the coverage is not as consistent. It currently uses 27 satellites, which also use two frequencies – L1 and L2.

These transmit two types of signal – open for public use and obfuscated for military use.[29, 32]

3.4 BeiDou

BeiDou is a Chinese variation on satellite navigation systems. The development started in 1996 and in 2000, the first three satellites were operational. Currently, BeiDou uses 51 satellites, which provide global coverage.[30]

BeiDou provides two types of service – Radio Navigation Satellite Service and Radio Determination Satellite Service. The RNSS functions similarly to other GNSS and provides both public and military-only signals. The RDSS in combination with ground stations, is used for tracking users in China.[30, 33]

4 Low power wide area network

In recent years, there has been a big push for smart devices which are always connected to the internet. The Internet of Things extended this connectivity to various sensors and other simple devices. In order to improve range capabilities and reduce power consumption for these smart devices, new types of networks were developed. These networks are called Low power wide area networks or LPWAN. These are further divided into cellular and non-cellular-based networks.[34]

4.1 Sigfox

Sigfox is a proprietary ultra-narrowband network, which is classified as a non-cellular network. It operates at 868 MHz and 915 MHz in Europe and North America respectively.[35] The main disadvantages of this network are limited data rates of only 100 bps, and limitation to only 140 transmitted and 8 received messages per day. Sigfox network is operated by Sigfox themselves and private networks are not allowed. The maximum range for Sigfox devices is 10 km in urban setting and 40 km in rural setting.[34]

4.2 LoRaWAN

LoRaWAN is a networking protocol defined on the 2nd layer of the ISO/OSI model – the MAC layer. LoRaWAN networks are classified as non-cellular networks and usually utilize a proprietary LoRa modulation. The frequency band is shared with Sigfox – 868 MHz and 915 MHz in Europe and North America respectively. The data rates vary depending on the distance from the closest access point. The maximum data rate is 50 kbps. In contrast with Sigfox networks, LoRaWAN allows for private networks. The maximum range for LoRaWAN devices is 5 km in urban setting and 20 km in rural setting.[34, 35, 36]

4.3 LTE cat M1

LTE cat M1 is a modification to the standard LTE for IoT devices, therefore is also classified as a cellular-based network. It can be implemented on existing LTE hardware in radio towers. The theoretical maximum range is 1 km in urban and 10 km in rural setting. With the maximum data rate of 1 Mbps, LTE-M networks are suitable for sending larger amounts of data quite frequently.[34, 37]

4.4 NB-IoT

Narrowband IoT network is a variant of LTE networks, created with the goal of increasing the range and decreasing the power consumption. This network is classified as a cellular network and it is standardized by the 3GPP initiative. The maximum data rate for NB-IoT is 200 kbps with the theoretical maximum range of 1 km in urban and 10 km in rural setting.[35]

The relatively small range compared to Sigfox and LoRaWAN is compensated by a big amount of existing LTE and GSM towers, which can run NB-IoT networks on their existing hardware [35]. In the Czech Republic, the biggest NB-IoT network provider is Vodafone a.s. with stated 100 % coverage available.[38]

As the data rate for NB-IoT networks is quite small, these networks are particularly suited for devices that need to transfer only a small amount of data but frequently. With it being a cellular network, it is also suitable for communication with devices that require mobility [39].

4.4.1 Network architecture

As the NB-IoT is a variation of existing LTE networks, its networks utilize simplified EPC architecture, which sport optimizations in user and control planes. These optimizations allow the usage of both IP and message-based data transfer. The diagram for such networks can be seen in Figure 4.1. Individual NB-IoT nodes connect to base stations – eNodeB (eNB), which are connected to evolved packet core (EPC) of the NB-IoT network via `s1-lite` plane. EPC consists of several entities: the Mobility Management Entity (MME), the Service Capability Exposure Function (SCEF), the Serving Gateway (SGW), and the Packet Data Network Gateway (PGW).[40, 41]

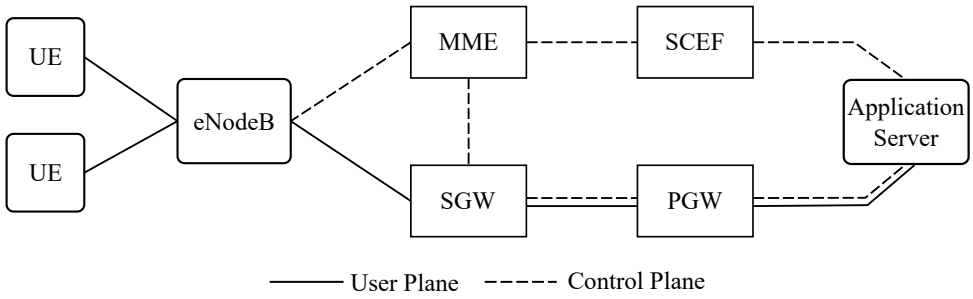

Fig. 4.1: NB-IoT network architecture.

4.4.2 Operation modes

NB-IoT uses a selection of LTE frequency bands, mainly in the lower part of the frequency spectrum. These lower frequencies are preferred as the IoT devices usually operate in difficult radio conditions, where these frequencies perform better.

The 3GPP Release 13 defines three different modes for the deployment of NB-IoT technology in the frequency spectrum, called: Standalone, Guard band, and In-band operation modes. Illustration of these modes can be seen in Figure 4.2.[40, 42]

Stand-alone

In Stand-alone operation mode, NB-IoT is deployed in between different cellular technologies such as LTE or GSM. In this case, it occupies a 200 kHz band, where 180 kHz is the actually utilized bandwidth and the remaining 20 kHz is used as guard interval – 10 kHz from both sides. [40, 42]

Guard band

Currently, the most popular operation mode for NB-IoT, because it utilizes otherwise unused resource blocks within the guard band of an LTE technology. In guard band mode, NB-IoT occupies one resource block (180 kHz) with a 10 kHz guard interval from each side. [40, 42]

In-band

In the In-band mode, NB-IoT deployment occupies one resource block within an existing LTE band. This operation mode is rarely used as it increases the NB-IoT network capacity at the expense of LTE network capacity and therefore it is costlier than previous modes. [40, 42]

4.4.3 Power saving measures

One of the main goals of NB-IoT technology is the reduction of power consumption. This is achieved with lowered transmission power from 23 dBm to 14 dBm according to a power class, as well as with the addition of two new power-saving features. The complete life cycle can be seen in Figure 4.3.[44]

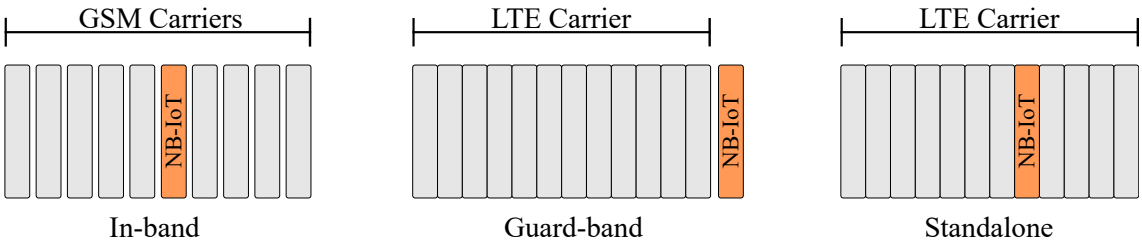

Fig. 4.2: NB-IoT operation modes.

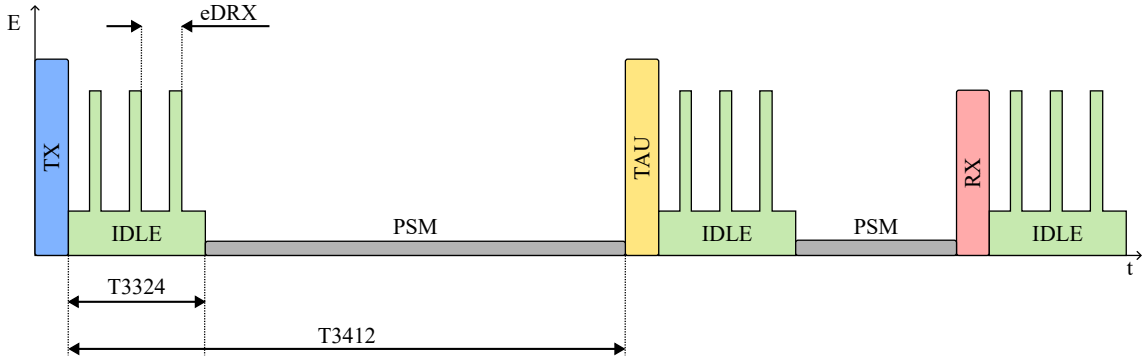

Fig. 4.3: NB-IoT life cycle.

Extended discontinuous reception

In LTE networks, the users' equipment (UE) enters an idle mode after transmission. During this mode, UE periodically checks the paging channel for potential incoming data or changes in the network. The periodicity of these checks is called the discontinuous reception (DRX) cycle and in LTE networks can last up to 2.56 s. For NB-IoT, this cycle has been extended up to 175 min and is controlled by a T3324 timer.[43, 44]

Power saving mode

If the network allows, a UE can enter a power saving mode (PSM) after the end of the idle mode. During this mode, the device remains registered to the network, but its radios are switched off and therefore it can't be accessed from the network. The duration of both idle a PSM is controlled by a T3412 timer and can last up to 413 days. After running out of the timer, UE performs either a scheduled transmission or a tracking area update (TAU). During this update, the UE informs the network about its state.[43, 44]

4.5 Communication protocols used with LPWAN

In IoT, there are numerous different communication protocols used as each IoT solution has different requirements for reliability, latency, etc.. The support for individual protocols is dependent on the module that has been chosen. A few of the most popular protocols to use in LPWAN networks are discussed in the following sections.[45, 46]

4.5.1 User Datagram Protocol

User Datagram Protocol or UDP is the simplest protocol used in IoT and on the internet in general. It is defined as a transaction-oriented protocol, which means it only sends and receives the data and does not guarantee successful delivery. The main advantage of this protocol is that it has low overhead and therefore it is fast. In IoT setting it is best suited for frequent transmission of non-critical data.[46, 47]

4.5.2 Transmission Control Protocol

Transmission Control Protocol or TCP, is one of the basic protocols used in both Internet and IoT. Apart from UDP, TCP is a connection-oriented protocol, which means that it first establishes a connection with the recipient before transmission of the data. Because of this fact, TCP is considered a reliable communication protocol. Compared to UDP, TCP has a significantly bigger overhead and is therefore suited for infrequent transmission of critical data.[46, 48]

4.5.3 Message Queue Telemetry Transport Protocol

MQTT is a protocol designed for IoT devices, which uses subscribe/publish messaging. For message transmission, it uses TCP protocol. In order to further improve the reliability of the communication, MQTT implements three levels of QoS - level 0 has the same guarantee of delivery as the underlying TCP and level 2 adds a 4-way handshake to ensure the message will be delivered exactly once. This adds flexibility to the communication because it makes prioritization of certain messages possible [49, 50]. On the other hand, MQTT adds a significant amount of overhead, especially with smaller payloads, and therefore it is not really suitable for frequent data transmission.[46]

4.5.4 Constrained Application Protocol

Constrained Application Protocol or CoAP is another communication protocol specially designed for IoT devices. It is similar to HTTP as it is based on the REST model. The server, in this case, the IoT device makes data available under the URL and the client can access them using methods like `GET, POST` or `DELETE`.[51]

For data transmission, CoAP uses UDP, which means that it has a low overhead compared to MQTT. This means very well suited for the frequent transmission of data. In order to improve reliability over standard UDP, CoAP implements two types of transmission: reliable and non-reliable. The reliable one requires `ACK` message from the recipient, otherwise, it will retransmit the message after the timeout. [46, 51]

5 Hardware

The goal of this bachelor thesis was to design and build a device, that would be capable of monitoring engine parameters via OBD, location via GNSS, and sending these pieces of information to a server via LPWAN networks in real-time. This task has been split into two segments, one for the semestral project and one for the subsequent bachelor's thesis.

The first segment revolved around the creating OBD II based tool, that would be capable of reading out set OBD II parameters and showing them on an embedded display. For the OBD part, several options were researched, including direct OBD II to UART interpreters. These solutions were abandoned due to the difficult availability of said ICs and a generic microcontroller approach has been chosen.

The second segment focused on expanding the base OBD II tool with a networking capability, in order to create the monitoring device, described in the main goal of this thesis. In the networking side of this device, NB-IoT technology has been chosen for its great coverage and sufficient data capacity for the intended usage.

5.1 Micro controller

As a basis for this device, microcontroller STM32L431 from ST Microelectronics has been chosen. It is a 32-bit ARM-based MCU with 256 kB of flash memory and multiple communication interfaces such as UART, SPI, I2C, and CAN. The main reasons for choosing this particular MCU were the integrated CAN Bus controller, low costs, and good availability.

The STM32L431 is produced in multiple different packages, for this board, a LQFP64 package has been chosen in order to maintain the possibility of hand soldering of each component. The schematic concerning MCU can be seen in Figure 5.1. According to an application note from the manufacturer, each power pin (VDD) has a 100 nF decoupling capacitor placed in close proximity to the MCU itself.[52] These are accompanied by one 4.7 µF bulk capacitor. Additional filtering for analog power pin (VDDA), recommended by the manufacturer, has been omitted as the precise function of ADC is not required.

The main source for the clock signal is a 16 MHz crystal oscillator with 8 pF load capacitors. The value of these capacitors has been calculated using

$$C_1 = C_2 = 2 \cdot (C_\mathrm{L} - C_\mathrm{S}) = 2 \cdot (9 - 5) = 8\,\mathrm{pF}, \tag{5.1}$$

where C_L is load capacitance as stated by the oscillator manufacturer and C_S is stray capacitance of the tracks on the PCB.[53]

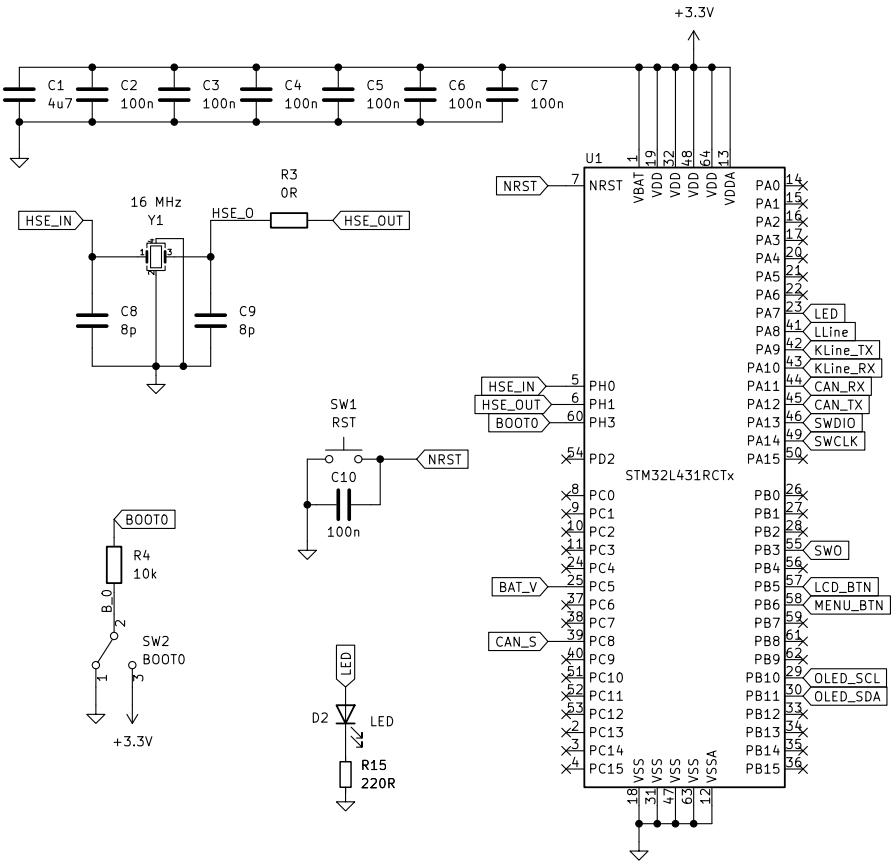

Fig. 5.1: Schematic for STM32L431.

For ease of development, both a reset button and a boot switch were added. The reset button is an SPST normally open button with 100 nF capacitor across its poles for filtering. Due to the infrequent use, the boot switch has been implemented with a pin header and a jumper to set the desired boot mode.

Programming and subsequent debugging is done via Serial Wire Debug (SWD) interface, which is modified JTAG specifically for ARM based MCUs. This interface has been implemented in full configuration with SWDIO, SWCLK, RESET and SWO pins connected to 10 pin header.

5.2 CAN BUS Transceiver

In order for the CAN controller to communicate on the CAN bus, a separate transceiver is needed for the translation of TTL logic to CAN logic. For this purpose, a CAN transceiver SN65HVD231 from Texas Instruments has been chosen. It is compatible with CAN2.0 and has a maximum bit rate of 1 Mbps. The main reason for choosing this transceiver was the ability to fully turn off the transceiver via R_s pin in order to reduce the energy consumption of the board while the vehicle

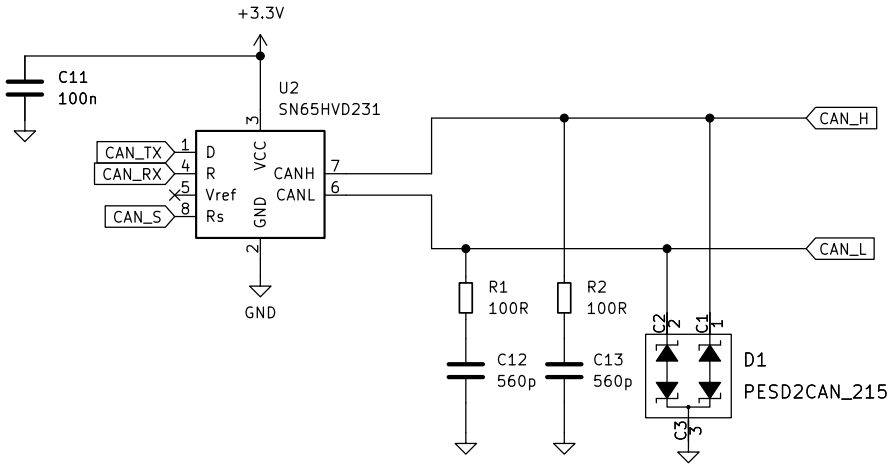

Fig. 5.2: Schematic for CAN bus transceiver.

is turned off. The schematic concerning the transceiver and CAN bus lines can be seen in Figure 5.2.

The CAN Rx and Tx pins are connected directly to the MCU, and R_s pin is also connected to MCU, where, in default, it is pulled low for High-speed mode.[54] Similar to the power pins on the MCU, the power pin on this transceiver has 100 nF decoupling capacitors placed in close proximity. Both CAN H and CAN L have 100 Ω resistor R and 560 pF capacitor for AC termination as stated in Section 2.3.3. For ESD protection, a TVS diode specifically designed for CAN has been added.

5.3 K Line Transceiver

As mentioned in Section 2.1.1, both ISO 9141 and ISO 14230 use modified UART communication. In order to connect the normal UART interface to the K-line, a K-line transceiver is needed. Standalone transceiver ICs are no longer available, the only solution is to use discrete parts to build one. The transceiver schematic is inspired by a schematic from ELM Electronics[1] and can be seen in Figure 5.3.

The transceiver used on this board consists of a voltage divider on the K-line RX pin and two NPN transistors on both K-line TX and L lines. The voltage divider is designed for reducing input voltage from nominal 12 V on which the K line runs, to 3.3 V for the MCU. The transistors serve a similar purpose in the opposite direction, they convert 3.3 V signals to 12 V logic. The 510 Ω pull-up resistors are required by ISO 9141.

[1] `https://www.elmelectronics.com/wp-content/uploads/2020/05/ELM327DSL.pdf`

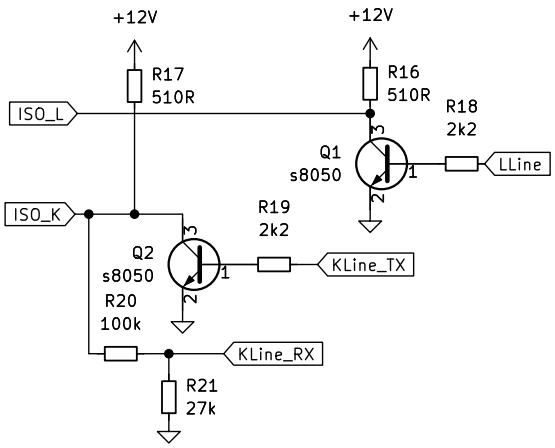

Fig. 5.3: Schematic for K line transceiver.

5.4 Power supplies

The task of powering a device with both an MCU and a separate networking module is a bit more complicated, as the networking modules usually use a lower logic voltage than the MCUs. In the case of the proposed device two different power supplies were required as the chosen MCU uses 3.3 V and the majority of the networking modules use a 1.8 V logic.

For the main 3.3V power supply, several solutions were explored with these requirements:

- Maximum input voltage of at least 15V
- Sufficient amperage
- High efficiency
- Availability

With these requirements in mind, a switching regulator was the chosen approach for the main power supply. For the task of choosing a particular regulator and designing its circuit, an online tool called *WEBENCH POWER DESIGNER*[2] from Texas Instrument, was used.

Specifically, a synchronous step-down converter circuit based on the TPS562201 regulator, was chosen. It is a switching regulator with 4.5 V to 17 V at the input and 3.3 V at 2 A output. The power supply schematic can be seen in Figure 5.4.

The input side has a 2A fuse and a bank of capacitors responsible for the input filtration and the decoupling of the main IC. The input voltage is then regulated by switching the energy on and off and filtering it on the output using an LC filter. The feedback for the main IC is provided by a high-precision voltage divider as recommended by the manufacturer.[55]

[2] `https://webench.ti.com/power-designer/`

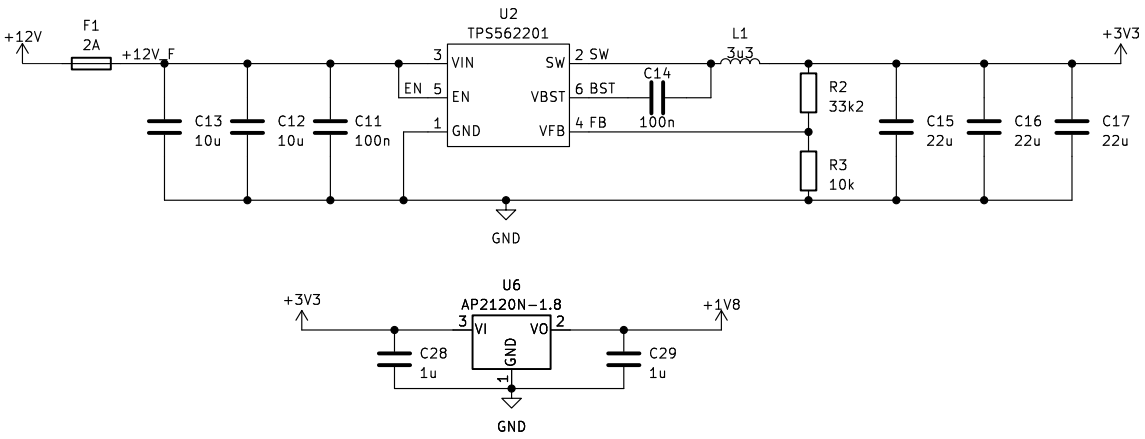

Fig. 5.4: Power supply schematic.

The second 1.8 V power supply has a lot simpler requirements, as it is only used for voltage level translation. The main requirements for this power supply were:

- Voltage regulation from 3.3 V down to 1.8 V
- Small BOM count
- Low costs

With these requirements in mind, a simple linear regulator AP2120N-1.8 from Diodes Inc. has been chosen. Because it is a linear solution, it only requires two filtering capacitors on the input and output of said IC.

5.5 Narrowband IoT module

Apart from the engine data collection aspect, the proposed device should also monitor its position and together with the rest of the data, send it to a database via NB-IoT network. For this purpose, an NB-IoT module Quectel BG77 has been chosen, mainly due to its size, availability, and integrated GNSS functionality.

5.5.1 Main module

For the basic operations, the Quectel BG77 requires only a handful of connections such as power, ground, UART, and SIM interface and radio antennas. The schematics for the majority of these connections are inspired by the BG77 Hardware Design guide and BG77 reference design.[56, 57]

This module is powered by a 3.3 V power supply with additional filter and bypass capacitors and a TVS diode for stability. The turn-on sequence is controlled by a GPIO output on the MCU with an open collector driver connected to *PWRKEY* input.

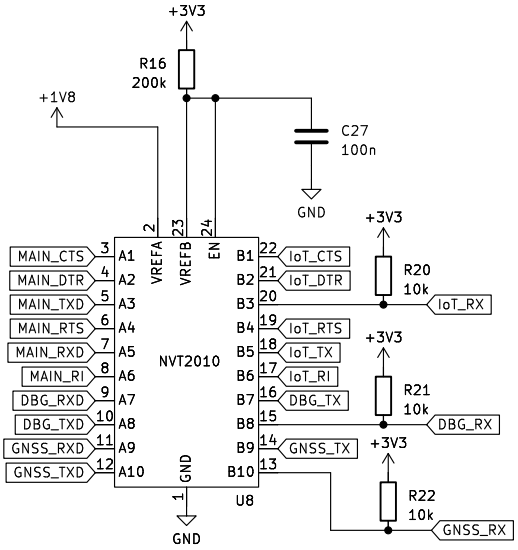

Fig. 5.5: Voltage level translator schematic.

A similar circuit is also utilized for *PON_TRIG* interface, which is used for waking up the module from PSM. This interface uses a 1.8 V logic, and therefore N-channel MOSFET has been added to the driver in order to translate the higher voltage level down. On the other hand, *AP_READY* pin, which is used for detection of the sleep status of the host MCU, uses only a simple voltage divider to step down the voltage to a 1.8 V.

5.5.2 UART level shifter

This module uses a UART interface with flow control as its main communication interface. Similarly to the previously mentioned pins, UART also uses a 1.8 V logic, while the MCU uses a 3.3 V logic. Due to this fact, the use of a bidirectional voltage translation circuit is required. For this purpose, a 10-channel voltage-level translator NVT2010 from NXP has been chosen. Its wiring diagram has been inspired by the reference design provided by the manufacturer [58] and it can be seen in Figure 5.5. This IC supports both open-drain and push-pull implementation and therefore pull-up resistors are required on the ports with higher voltage levels.

As this translator IC has 10 channels, it operates not only the main UART interface but also the GNSS UART interface for NMEA output sentences and a debug UART interface.

5.5.3 Antennas

As the Quectel BG77 supports both NB-IoT and GNSS, it sports two separate antenna interfaces with a 50 Ω impedance. Their schematic can be seen in Figure

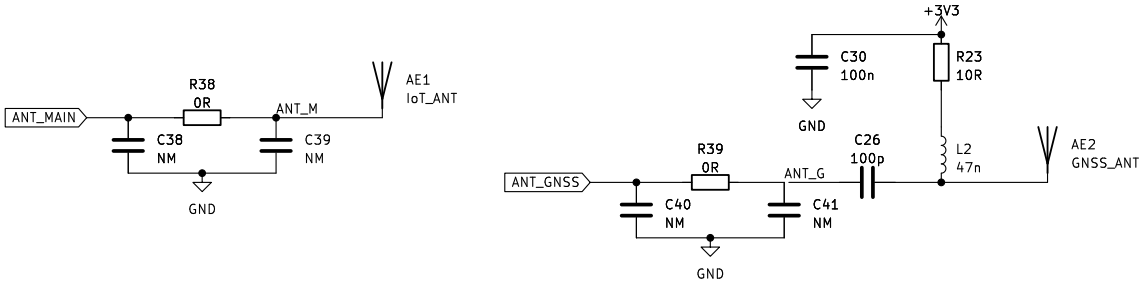

Fig. 5.6: NB-IoT and GNSS antennas schematic.

5.6. The NB-IoT has a quite high tolerance for the strength of the signal and because of this, a passive antenna has been chosen. The antenna is meant to be built-in inside the enclosure and is only connected to the main board via a u.fl connector.

The GPS on the other hand, is a lot more demanding on the signal strength and therefore requires an active antenna to improve the fix times. The chosen antenna is meant to be installed inside the car, ideally with a clear view of the sky. It uses an automotive-grade Fakra C connector to connect to the enclosure, which has a built-in adapter from Fakra C to the u.fl connector. As it is an active antenna, it requires power circuitry to supply its built-in amplifier. The schematic for this circuitry has been borrowed from the hardware design guide provided by the module's manufacturer.[56]

5.5.4 SIM card

In order to be able to register and operate on the provider's network, every UE needs a SIM card with a unique identifier.

To comply with this requirement, the proposed device uses a micro SIM card slot, mainly due to its compact size. As the SIM card slot is expected to be used by the end user, a group of TVS diodes has been added for additional ESD protection. The SIM card slot schematic can be seen in Figure 5.7.

5.5.5 USB

The Quectel BG77 uses a USB interface as its second communication interface and also as the main upgrade path for its firmware. Due to this fact and also to ease the development, a simplified USB interface has been added. Its schematic has been inspired by the reference design from the modules manufacturer and it can be seen in Figure 5.8.[57]

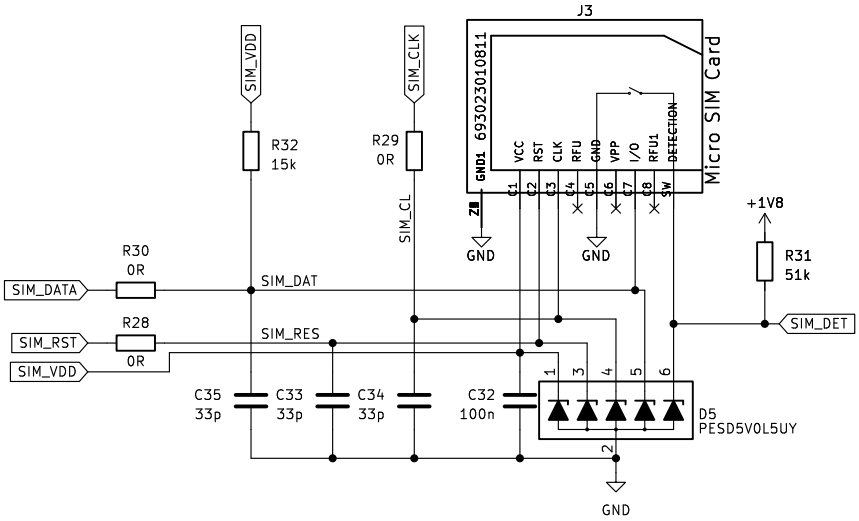

Fig. 5.7: SIM card slot schematic.

As the interface would be used only a handful of times during development, the USB interface has been implemented only with a 4x2 SMT pin header. This connector houses both the interface itself as well as jumpers for turning on the interface and booting from it.

5.6 PCB Design

Designing both the schematic and final PCB was done using KiCAD 7.0. It is an open-source software meant for electronic design automation. It consists of a Schematic capture tool, PCB layout tool, manufacturing tools for generating Gerber files and BOMs, and various calculation tools for electronic engineering.

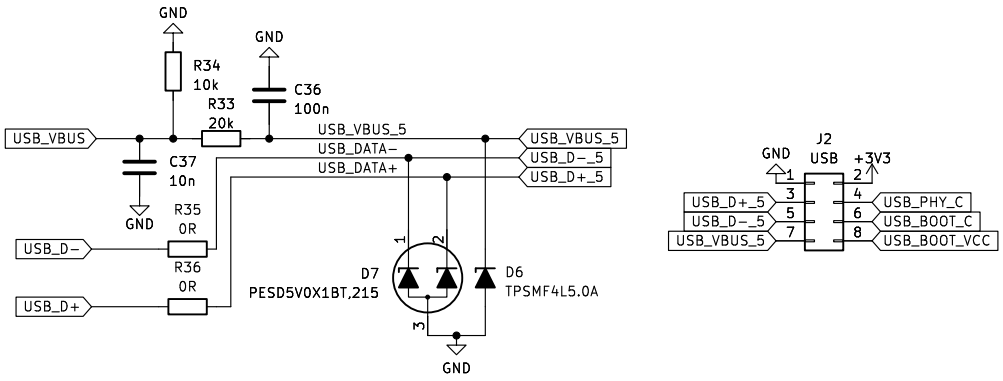

Fig. 5.8: USB interface schematic.

5.6.1 PCB

To ease the task of laying out the PCB traces, a 4-layer board has been used, with the following stack-up:

- Signal plane
- GND
- 3.3 V Power plane
- Signal plane.

The final PCB has dimensions of 50 x 50 mm and its layout can be seen in Figure 5.9. In order to fit everything, the double-sided assembly has been utilized together with the smallest components that still allow hand soldering.

5.6.2 Micro controller layout

As mentioned in Section 5.1, the main micro-controller IC uses the LQFN64 package. Each power and ground pin pair has a decoupling capacitor placed in close proximity to them. The decoupling capacitors are then connected to corresponding planes with vias, that are placed directly next to them in order to reduce ground loops. The bulk capacitor is placed as far from the power supply as possible, as recommended by the manufacturer.

The chosen crystal oscillator uses the SMD-3225 package and is also placed near the microcontroller, in order to mitigate trace capacitance. Both load capacitors are placed as close as possible to the oscillator in order to reduce ground loops and trace capacitance.

5.6.3 Power supply layout

Due to its switching nature, the main voltage regulator has been placed as far as possible from noise-sensitive components such as radio antennas. The physical layout of the regulator and its components is inspired by the reference layout from the manufacturer. As mentioned before, the voltage regulator has a pair of 0603 filtering capacitors placed on the input side together with a decoupling capacitor for the IC itself. The output side is filtered by a combination of a single inductor and a group of 0805 capacitors. The voltage feedback is set by a voltage divider from the output side.

The secondary power supply has been placed in closer proximity to the NB-IoT module itself, as it is only used for voltage-level translation. The selected voltage regulator comes in the SOT-23 package and is accompanied by a pair of 0603 capacitors on both the input and the output.

5.6.4 CAN and K Line transceivers layout

The CAN bus transceiver uses the SOIC-8 package and is located in between the microcontroller and OBD II connector. The power pin of this transceiver has a decoupling capacitor placed in close proximity, in order to reduce ground loops. The CAN TX and CAN RX are routed directly to the MCU, and the CAN H and CAN L are both routed to the OBD II connector through the TVS diode for ESD protection. The AC termination, mentioned in Section 2.3.3, is placed in close proximity to the CAN lines, after the ESD protection diode.

The K-line transceiver is composed of two S8050 transistors, which use the SOT-23 package. The routing of this transceiver is a combination of both the top and bottom layers, in order to reduce the footprint. The pull-up resistors for both K and L lines are connected to the +12 V plane on the top layer.

5.6.5 NB-IoT module layout

The selected Quectel BG77 has been placed right next to the main microcontroller, in order to minimize the length of required tracks and to make room for the antennas. This module comes in a 94-pin LGA package, which has reflected in the track width selection – only 0.1 mm in order to fit between individual pins.

Due to the size constraints, the majority of module peripherals have been located on the bottom side. These include the voltage translator, the sim card slot with TVS diodes, and the whole *PON_TRIG* circuit. The voltage translator is located directly underneath the main module and uses a TSSOP-24 package, which was selected for its hand solderability.

The exception in routing on this board is the antenna's layout, where each antenna has an impedance-matched track between the pin on the module and a u.fl connector. The GPS antenna also has a power circuit in order to support active antennas. Both of the antennas have a special plane cut out in order to ensure impedance matching and limit any interference from the rest of the board.

Another specialty in layout on this board is the USB interface, which uses a surface-mounted pin header instead of the usual USB connector. Both of the data pins form a differential pair with a 90 Ω impedance and therefore they need to be routed with a specific track width and spacing. The rest of the header is used for turning on the USB interface itself and for boot mode selection for the module. Both of these functions are done using a simple jumper. Every part of the USB interface is expected to be operated by a human and therefore each sensitive pin has a separate TVS diode for ESD protection.

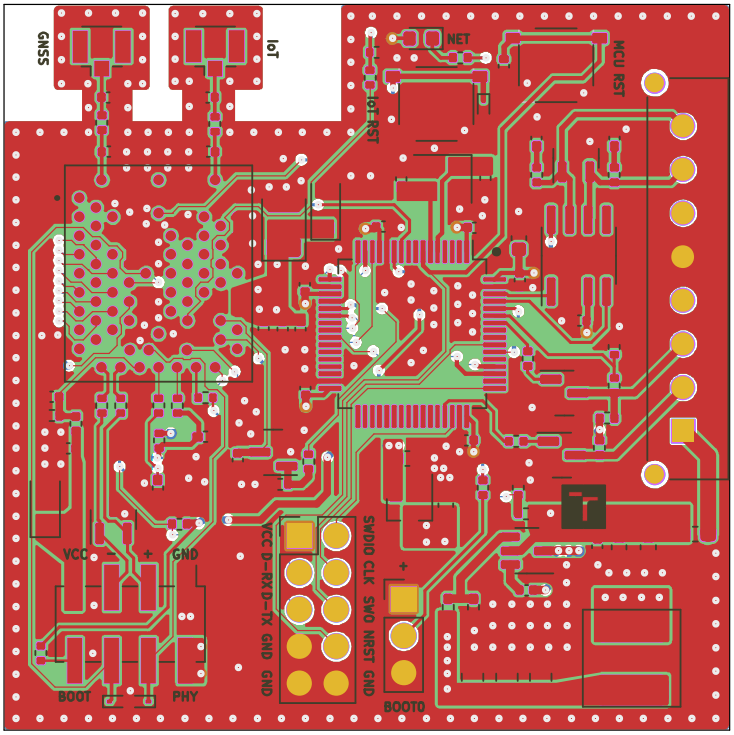

Fig. 5.9: PCB layout.

5.6.6 Peripherals & Connectors layout

The design of this board keeps the number of peripherals at a minimum necessary to achieve the desired functionality. The main peripheral is a debug connector that combines SWD and debug UART interface from the NB-IoT module. It has been placed in close proximity to both MCU and the NB-IoT module and is implemented using a 2x5 THT pin header.

The main connector provides both power and connection to the car's diagnostic interface. It has been implemented with a single-row, 8-pin connector from the Molex Micro-Fit 3.0 series. This connector has been chosen mainly for its size and current carrying capacity.

5.7 Manufacturing

The final board, which can be seen in Figure 5.10, was produced by JLCPCB, who had the best offer for 4-layer board fabrication and SMT assembly. The manufacturer offers two types of component libraries for SMT assembly – basic and extended. The basic library consists of the most used components such as passives or basic ICs. Everything else is a part of the extended library and usage of such parts comes with a 3 € fee as they have to manually load up a new reel for each one.

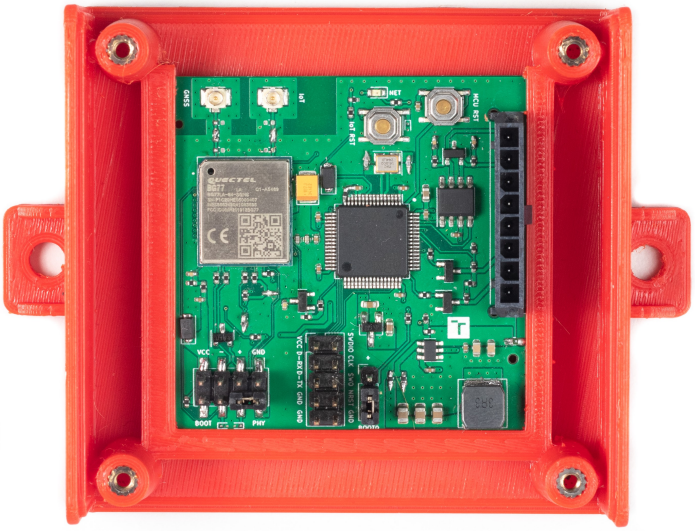

Fig. 5.10: Finished board mounted inside the case.

In order to save on manufacturing costs, a combination of automated and manual assembly has been utilized, mainly due to higher setup costs for double-sided assembly and extra fees on components from the extended library. While the extended components have an extra fee, some of them have a far better price even with the fee included compared to the retail market and therefore this combination was used to optimize the costs.

5.8 Case Design

The proposed device is meant to be installed permanently inside a vehicle, out of reach for the average person. The case design reflects this fact and it uses a simple two-part approach.

The base has a sliding feature to hold the main board in and four pegs with threaded inserts to hold the top cover on. It also sports a pair of tabs for larger screws in order to firmly secure the device inside a vehicle. These tabs support up to an M5 screw with a variety of different screw heads.

The top part has a cutout for the main connector and integrates both a place for the GPS Fakra C connector and the passive NB-IoT antenna. This part is fixed to the base using four M2.5x7 with a pan head.

The whole case has been designed using an Autodesk Inventor 2024 and it has been 3D printed in PETG plastic. The final design and a finished product can be seen in Figure 5.11 and in Figure 5.12 respectively.

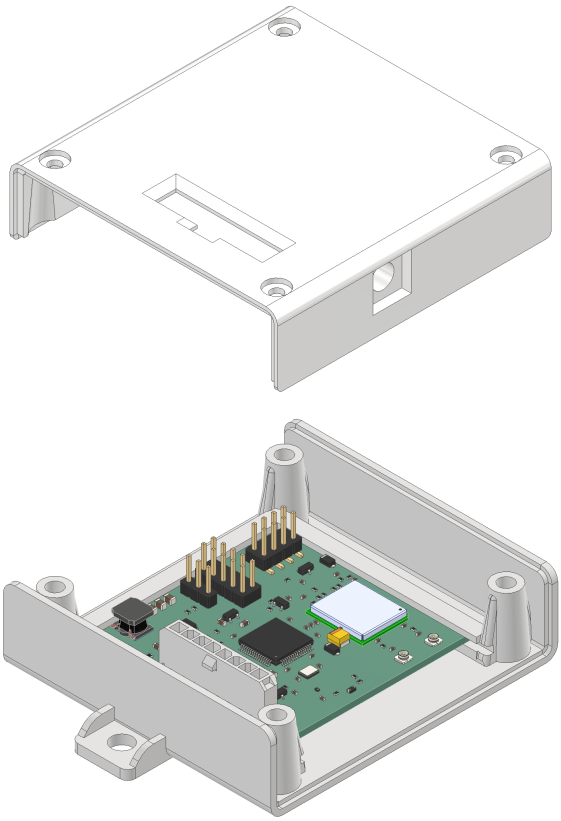

Fig. 5.11: Final case design.

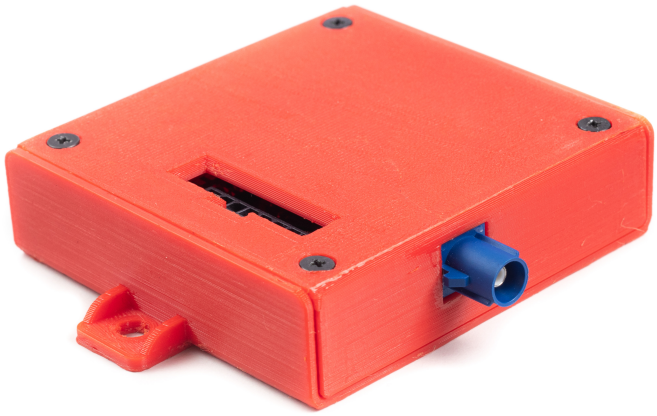

Fig. 5.12: Finished product.

6 Software

The software for this device was written in C using STM32CubeIDE 1.12 from the manufacturer of the microcontroller. This IDE provides an easy way of configuring the integrated peripherals with the included STM32CubeMX configuration tool.

6.1 Main architecture

The main architecture of the software is rather simplistic and its flowchart can be seen in Figure 6.1.

The program begins with a battery voltage check in order to preserve the vehicle's battery. After a successful voltage check, begins the NB-IoT module initialization with `module_init` function. This function turns on the module and checks if it is responding correctly on the main UART interface. The module's initialization is finished with a connection to an MQTT broker.

Successful initialization of the NB-IoT module followed by initialization of the communication with the vehicle. That is handled by a `obd2_init` function, which iterates through protocol-specific initialization functions until it receives the type of protocol used by the vehicle.

After both the module and vehicle's communication interface have been initialized, the program goes into a while cycle, where it periodically acquires the requested PIDs, forms them into a JSON format, and sends them via MQTT to a NodeRed server for further processing.

The data acquisition is handled by `acquire_vehicle_data`, which repeatedly sends an OBD II request via `obd2_request`, until a valid message is received from the vehicle's ECU.

The reception of the response to these requests is handled by the corresponding function for each of the protocols. Subsequently, the received frame is parsed by `ob2_pid_parse` function, which returns a decoded value based on the PID of the requested frame.

After decoding the value of the set PID, a PID-Value pair is saved into a 2D-array and passed through to a `create_json` function, which transforms the array's contents into a JSON style string, ready for transmission via MQTT.

The MQTT publication is handled by `mqtt_publish` function, which takes the JSON string and a topic as an argument a publishes it to the MQTT broker.

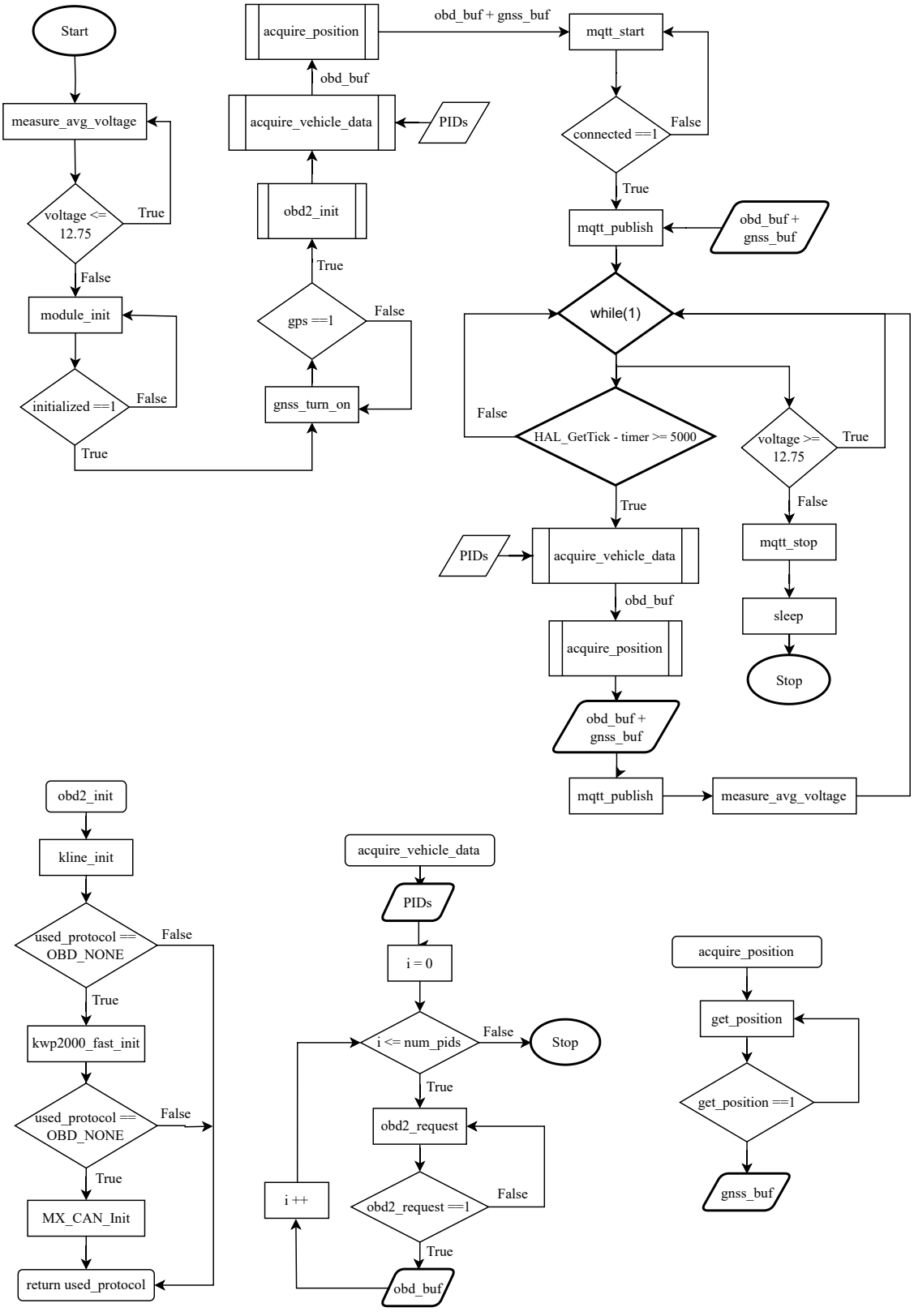

Fig. 6.1: Software flowchart.

6.2 K line and KWP2000 implementation

The OBD II initialization begins with an ISO 9141 as it is the slowest one. Its initialization procedure has been described in Section 2.1.1 and it is implemented by `kline_init` function. This function also implements a slow init procedure for ISO 14230 as they are nearly identical. The initial address transmission is done by simply toggling the GPIO output on and off and subsequent communication is done via UART in polling mode. Depending on the received bytes from the vehicle, the function returns `OBD_NONE`, `OBD_PROTO_ISO9141` or `OBD_PROTO_KWP2000_SLOW`.

If the result is `OBD_NONE`, the initialization continues with a fast init procedure for ISO 14230, which is implemented by a `kwp2000_fast_init`. This function works in a similar manner to the previously mentioned `kline_init` function, with the main difference being the data that are transmitted. The functions return depends on the received message and the function itself returns either `OBD_PROTO_KWP2000_FAST` or `OBD_NONE`.

The OBD II requests are handled by `kline_send_msg` and `kwp2000_send_msg` functions respectively. Both of these functions work in a similar manner, with the main difference being the format of the data that they transmit. The OBD II mode and PID information, provided as the input argument, is used together with a predefined frame and calculated checksum to create the final request message, which is then transmitted via UART in polling mode. Afterward, the function waits for a response message.

The reception uses UART in a DMA mode, which fires an interrupt function to let the MCU know, that a message has been received. This message is then verified and parsed into a variable using `obd2_pid_parse`. If the received message is not valid or the timer runs out, the functions return zero to let `obd2_request` know that the request was not successful.

6.3 CAN BUS implementation

If the previous initialization attempts return `OBD_NONE`, then the program resorts to a CAN bus initialization. The main initialization function `MX_CAN1_Init` is generated by the STM32CubeMX and set in a way that achieves a bus speed of 500 kbps. Subsequently, the function `can_config` is called to configure the reception filtering, in order to receive only valid OBD II defined IDs. After setting up the filter, the CAN bus controller is started and an interrupt callback for a received message is set up and activated.

Sending messages via CAN bus is handled by `can_send_msg` function, which takes a request frame as an argument. This function adds a CAN header with the

device ID, length value, and type of message information, to the request frame and transmits the whole message on the CAN bus. After a successful transmission, the function waits for a response from the vehicle.

The received messages are stored in `CAN_RX_FIFO0` register and after successful reception, an interrupt is triggered, and `HAL_CAN_RxFifo0MsgPendingCallback` function is called. This function extracts the data portion from the received frame and saves it to a data array for further processing.

If the response is received correctly, it is passed to a `obd2_pid_parse` for parsing into a variable, and the function returns one. If the reception was not successful, the function returns zero.

6.4 OBD II Parsing

The decoding of received frames is handled by `obd2_pid_parse` function, which takes the frame as an argument in the form of an array. A simple switch case is then used to determine the correct decoding equation, based on the PID value in the received frame. The decoded value is then returned in the form of a float variable, to account for all of the different types of values described in {[14, 13]}.

6.5 Quectel BG77 driver implementation

The driver for interacting with the Quectel BG77 has been written from scratch and it is inspired by an existing Arduino library for a similar Quectel BC660 made by R4sp1[1]. The module itself communicates via the UART interface with AT commands, which can be found in the manuals from the manufacturer.

The main communication is handled by a `send_command` function, which takes both the command and expected reply as an argument. The transmission is handled by UART in polling mode and the reception uses an interrupt mode, which is set to receive only one byte. Together with the transmission of the command, a timer is started to limit the wait time for a response from the module.

After the reception of one byte, the interrupt function `nb_rx_callback` saves the byte into an array and restarts both the UART reception and the timer. When the timer eventually runs out, an interrupt is triggered to let the MCU know that either a message has been received or the connection timed out. Subsequently, the `send_command` function checks the received reply and if the expected reply matches with the received one, the function returns 1, otherwise, it returns 0.

[1] `https://github.com/R4sp1/Quectel-BC660`

The rest of the functions in this driver use the same principle: send the AT command via `send_command` and if the response is valid, check the response for additional data concerning the command used. The exceptions to this are the `power_on` and `power_off` functions, which toggle the GPIO on and off on the *PWRKEY* pin.

6.6 Data processing

The received data from both the car and GNSS module are processed in a Node-Red server and fed into InfluxDB, which is used for both storage and visualization of the measured data. The final flow in Figure 6.2.

The data is published via MQTT protocol separately using two topics. The OBD data is received as JSON string, afterwards, PIDs are translated to human-readable form, parsed into JSON, and pushed towards InfluxDB. On the other hand, the location data does not have a need for further processing and therefore it is received and immediately parsed into JSON and pushed towards the InfluxDB node with different measurement setting.

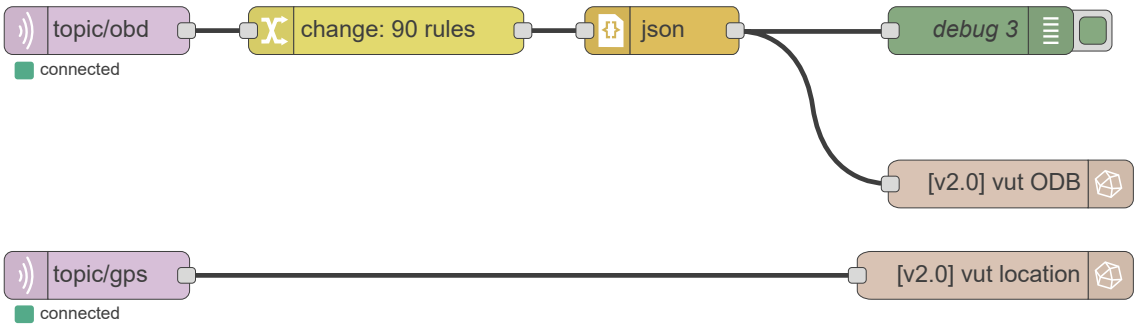

Fig. 6.2: Data processing flow in NodeRed

7 Testing

As mentioned before, during this thesis, two different devices were produced and both of these have been thoroughly tested on various vehicles, namely 2011 Renault Megane and 2005 BMW 320d E91, which were mainly used as a test mule for the development of both devices. Most of the test drives were carried out around the Brno area and some of the results from these can be seen in the following figures.

Figures 7.1 and 7.2 show an example of collected information from a vehicle's ECU during a test drive. Figure 7.1 shows a correlation between the engine's RPM and the vehicle's speed in time, where the sudden changes in RPMs correspond to vehicle acceleration and deceleration.

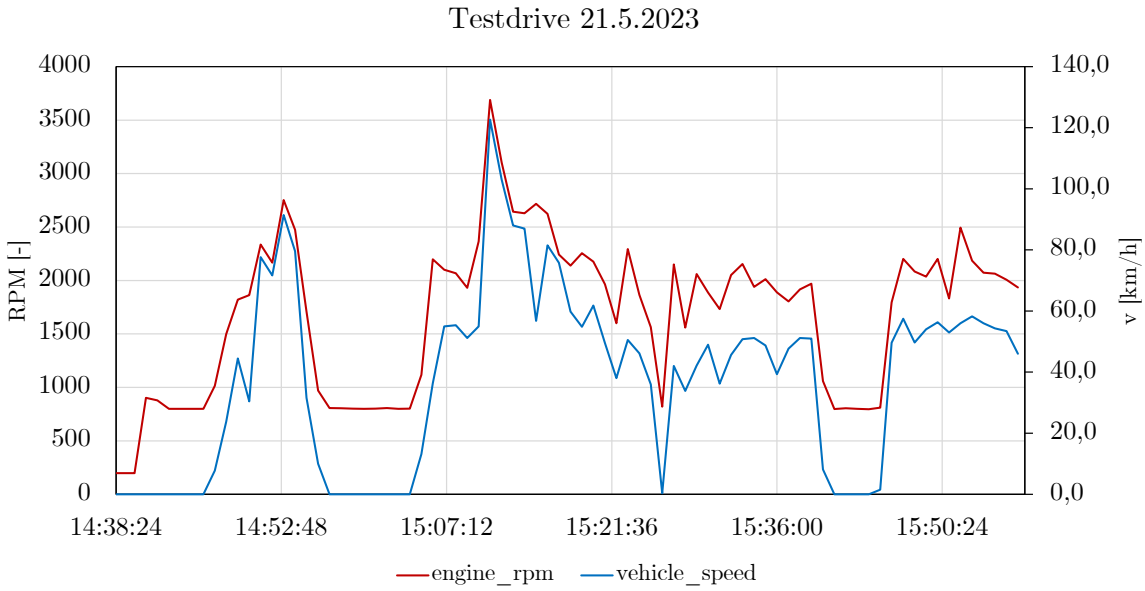

Fig. 7.1: Measurement from testdrive on 21.5.2023 – engine's RPM and vehicle's speed in time.

A similar correlation can be seen in Figure 7.2, where it is a correlation between the vehicle's speed and coolant temperature. Small temperature increases can be seen when the vehicle is stationary as there is a lack of airflow through the radiator. The same effect can be observed when the vehicle starts accelerating to a certain speed, where the increased load produces additional heat in the cooling system. When the vehicle starts to decelerate, the temperature slightly drops as the load decreases but the airflow is still sufficient.

Both of these figures show that the device works reliably as the measured data meets the expectations. Figures 7.3, 7.4 and 7.5 show working devices during the development stage.

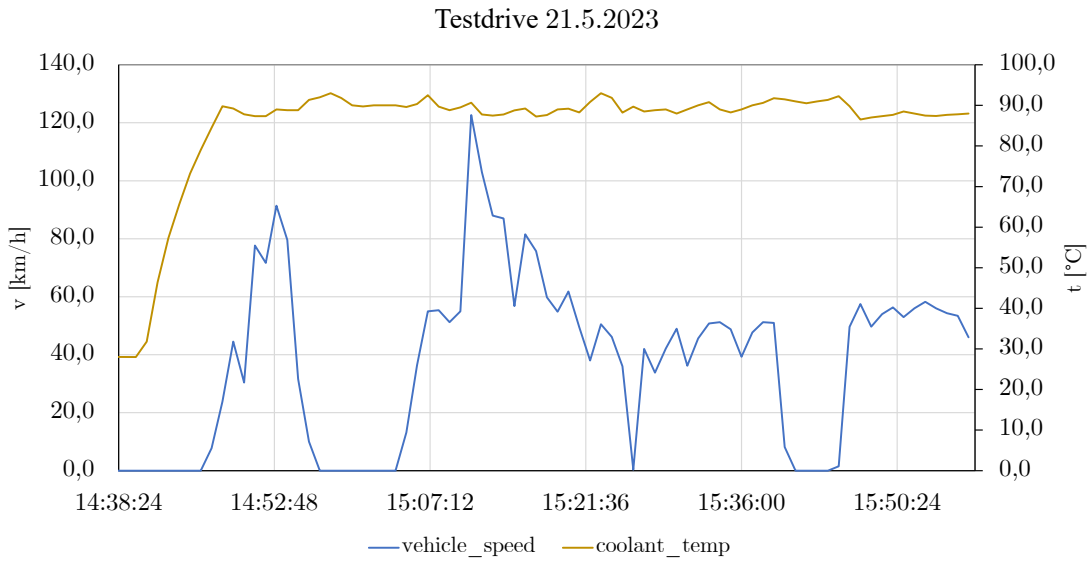

Fig. 7.2: Measurement from testdrive on 21.5.2023 – coolant temperature and vehicle's speed in time.

Fig. 7.3: OBD II diagnostic tool – testing in BMW E91.

Fig. 7.4: OBD II monitoring device – testing in Renault Megane.

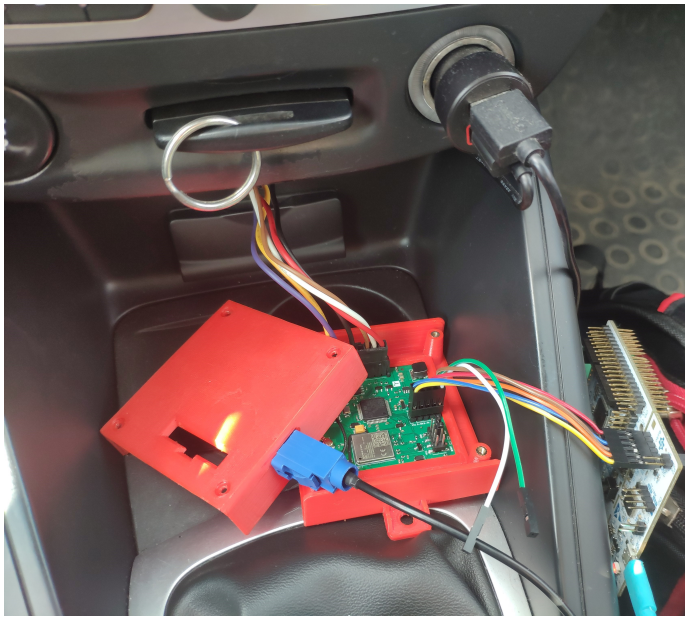

Fig. 7.5: OBD II monitoring device – testing in Renault Megane – closeup.

8 Cost analysis

The main advantage of the proposed device is the usage of modern LPWAN networking instead of old and soon-to-be deprecated GSM/GPRS networking. While the device itself is mainly focused on a fleet customer rather than a sole end user, it still needs to be cost-effective in order to be competitive with other solutions on the market. The cost summary for a single device can be seen in Table 8.1. The proposed device comes at 50 € for a single unit.

There are a number of departments, where the costs could be optimized further, such as the board assembly. For this thesis, only two devices were produced and therefore fixed manufacturing costs are quite high. But with a higher production volume, the economies of scale come to play and the single unit cost starts to come down. Another option to optimize would be the components around the NB-IoT module, mainly the voltage translator which could be replaced with a smaller one, as both the debug and GNSS UARTs are not used.

Tab. 8.1: Cost summary for single unit.

Item	**Supplier**	**Price [€]**
PCB	JLCPCB	0.476
SMT Assembly		0.35
Stencil		0.282
Components		4.825
Extended parts fee		2.948
Setup fee		1.508
Components for Hand Soldering	Farnell	0.69
	Mouser	11,25
NB-IoT Antenna	Aliexpress	2,65
GPS Antenna		5,70
3D printed case		2
Shipping	Fedex	1,932
Taxes		3,8
Summary		**38,41**

Conclusion

The topic of easy diagnosis for cars became a lot more prominent in recent years, as cars became a lot more complex. This fact is of concern not only for individual owners but also fleet managers as unexpected service items can increase the downtime of a fleet.

The main goal of this bachelor's thesis was to design and build a device, that would be capable of monitoring set engine parameters via OBD II and transmitting them to a remote server via Low Power Wide Area networks (LPWAN). This goal was realized by designing a custom board built around the STM32L431 microcontroller. To make this device suitable for a wide range of cars, the proposed device supports several different communication protocols used in cars sold in Europe. This OBD II part has been developed and tested on a previously designed OBD II diagnostic tool, which also has been presented at a student conference STUDENT EEICT 2023.

A modern Narrowband IoT network has been selected as the ideal solution for this proposed device. The network support has been implemented using a Quectel BG77 NB-IoT module combined with a passive antenna for NB-IoT and an active antenna for Global Navigation Satellite Systems (GNSS).

At the time of writing this conclusion, the board was capable of measuring a whole set of OBD II PIDs via the most used protocols and was also able to transmit them to a remote server for further processing. It is also capable of reporting its location to the same remote server, but the server itself currently cannot show the location directly on the map.

The main disadvantage of the proposed device is the currently limited support for OBD II modes as it only supports mode 01. Despite this, it could be argued that the proposed device as a concept has commercial potential and it could be further expanded with support for multiple OBD II modes, a dedicated server application, and possibly a fault prediction system.

With that being said, according to the thesis assignment, the results of this bachelor's thesis can be considered successful as the goals given in the assignment have been accomplished.

Bibliography

[1] Commission proposes new Euro 7 standards to reduce pollutant emissions from vehicles and improve air quality, 10 November 2022. European Commission [online]. [Accessed 6 December 2022]. Available from: https://ec.europa.eu/commission/presscorner/detail/en/ip_22_6495

[2] NEDELCHEV, Alek, 2022. AUTOMOTIVE DIAGNOSTICS — A BRIEF HISTORY IN HOWEVER MANY CHAPTERS. *Abrites.com* [online]. 2022. [Accessed 30 October 2022]. Available from: https://abrites.com/blog/automotive-diagnostics-a-brief-history-in-however-many-chapters

[3] DENTON, Tom, 2006. *Advanced automotive fault diagnosis*. 2nd ed. Burlington: Elsevier Butterworth-Heinemann. ISBN 978-075-0669-917.

[4] *On-Board Diagnostic (OBD) Regulations and Requirements: Questions and Answers*, 2003. [online]. 2003. U.S. Environmental Protection Agency. Available from: https://nepis.epa.gov/Exe/ZyPDF.cgi/P100LW9G.PDF?Dockey=P100LW9G.PDF

[5] *DIRECTIVE 98/69/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*, 1998. [online]. 1998. Available from: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1998L0069:19981228:EN:PDF

[6] *Diagnostic Connector Equivalent to ISO/DIS 15031*, 2002. . USA: Society of Automotive Engineers. SAE J1962

[7] *On-board diagnostics.*, 2010. [online]. 2010. USA: United States Environmental Protection Agency. Available from: https://www.govinfo.gov/content/pkg/CFR-2010-title40-vol18/pdf/CFR-2010-title40-vol18-sec86-005-17.pdf

[8] ZANDIN, Johan. OBD-II female connector pinout.svg. *Wikipedia: the free encyclopedia* [online]. [Accessed 11 December 2022]. Available from: https://commons.wikimedia.org/wiki/File:OBD-II_female_connector_pinout.svg

[9] BABELA, Miroslav, MUSIL, Petr and , Radek Možný, 2023. OBD Diagnostic tool. In: *Proceedings I of the 29th Student EEICT 2023*. 2023. ISBN 978-80-214-6153-6.

[10] *Class B Data Communications Network Interface*, 2001. . USA: Society of Automotive Engineers. SAE J1850

[11] MAHAJAN, Gauri, PARCHANDEKAR, S.K. and TAHIR, Mohammad, 2017. Implementation and Validation of K Line (ISO 9141) Protocol for Diagnostic Application. In: *International Research Journal of Engineering and Technology* [online]. Maharashtra, India. 2017. p. 708-713. [Accessed 26 November 2022]. Available from: `https://www.irjet.net/archives/V4/i7/IRJET-V4I7181.pdf`

[12] FALCH, Martin, 2022. OBD2 Explained: A Simple Intro [2022]. *CSS Electronics* [online]. 2022. [Accessed 30 October 2022]. Available from: `https://www.csselectronics.com/pages/obd2-explained-simple-intro`

[13] *E/E Diagnostic Test Modes*, 2002. . USA: Society of Automotive Engineers. SAE J1979

[14] *Road vehicles - Communication between vehicle and external equipment for emissions-related diagnostics - Part 5: Emissions-related diagnostic services*, 2006. . Switzerland: International Organization for Standardization. ISO 15031-5

[15] *Diagnostic Trouble Code Definitions*, 2002. . USA: Society of Automotive Engineers. SAE J2012

[16] *Road vehicles - Diagnostic systems: Part 2: CARB requirements for interchange of digital information*, 1994. . Switzerland: International Organization for Standardization. ISO 9141-2

[17] *Road vehicles - Diagnostic communication over K-Line (DoK-Line) - Part 2: Data link layer*, 1999. . Switzerland: International Organization for Standardization. ISO 14230-2

[18] *Road vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 4: Requirements for emission-related systems*, 2000. . Switzerland: International Organization for Standardization. ISO 14230-4

[19] *Road vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 3: Application layer*, 1999. . Switzerland: International Organization for Standardization. ISO 14230-3

[20] *Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling*, 2003. . Switzerland: International Organization for Standardization. ISO 11898-1

[21] *Road vehicles - Controller area network (CAN) - Part 2: High-speed medium access unit*, 2003. . Switzerland: International Organization for Standardization. ISO 11898-2

[22] *Road vehicles - Diagnostic communication over Controller Area Network (DoCAN) - Part 2: Transport protocol and network layer services*, 2016. . Third edition. Switzerland: International Organization for Standardization. ISO 15765-2

[23] *Self-study programme 269: Data transfer on CAN data bus II*, 2003. . Wolfsburg. Volkswagen AG

[24] *Self-study programme 238: Data Exchange On The CAN Bus I*, 2001. . Wolfsburg. Volkswagen AG

[25] *CAN Specification: Version 2.0*, 1991. . Stuttgart. Robert Bosch GmbH

[26] *Road vehicles - Diagnostics on Controller Area Networks (CAN) - Part 4: Requirements for emissions-related systems*, 2005. . Switzerland: International Organization for Standardization. ISO 15765-4

[27] Global Positioning System History, 2017. *National Aeronautics and Space Administration* [online]. 2017. [Accessed 5 December 2022]. Available from: `https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html`

[28] GLOBAL POSITIONING SYSTEM (GPS) OPERATIONAL SATELLITES. *N2YO* [online]. Available at: `https://www.n2yo.com/satellites/?c=20`

[29] KAPLAN, Elliott D. and HEGARTY, Christopher J., 2006. *Understanding GPS: Principles and Applications*. 2nd ed. Boston: Artech House. ISBN 15-805-3894-0.

[30] MILLNER, David H., MAKSIM, Stephen and HUHMANN, Marissa. BeiDou: China-s GPS Challenger Takes Its Place on the World Stage. *National Defense University Press* [online]. [Accessed 5 December 2022]. Available from: `https://ndupress.ndu.edu/Media/News/News-Article-View/Article/2999161/beidou-chinas-gps-challenger-takes-its-place-on-the-world-stage/`

[31] Galileo Services, 2022. *EUSPA* [online]. 2022. [Accessed 5 December 2022]. Available from: `https://www.euspa.europa.eu/galileo/services`

[32] GLONASS OPERATIONAL SATELLITES, *N2YO* [online]. [Accessed 5 December 2022]. Available from: `https://www.n2yo.com/satellites/?c=21`

[33] BEIDOU NAVIGATION SYSTEM SATELLITES, *N2YO* [online]. [Accessed 5 December 2022]. Available from: https://www.n2yo.com/satellites/?c=35

[34] HOSSAIN, Mohammad Istiak and MARKENDAHL, Jan I., 2021. Comparison of LPWAN Technologies: Cost Structure and Scalability. In: *Wireless Personal Communications* [online]. Springer Science+Business Media. 2021. p. 887-903. Available from: https://link.springer.com/10.1007/s11277-021-08664-0

[35] MEKKI, Kais, BAJIC, Eddy, CHAXEL, Frederic and MEYER, Fernand, 2019. A comparative study of LPWAN technologies for large-scale IoT deployment. In: *ICT Express* [online]. The Korean Institute of Communications Information Sciences. 2019. p. 1-7. Issue 1, Volume 5. Available from: https://linkinghub.elsevier.com/retrieve/pii/S2405959517302953

[36] What is LoRaWAN® Specification, *LoRa Alliance* [online]. [Accessed 12 December 2022]. Available from: https://lora-alliance.org/about-lorawan/

[37] Implementing Low-Power Wide-Area Network (LPWAN) Solutions with AWS IoT: LTE - M, 2022. *Amazon* [online]. 2022. [Accessed 5 December 2022]. Available from: https://docs.aws.amazon.com/whitepapers/latest/implementing-lpwan-solutions-with-aws/lte-m.html

[38] Co je NB — IoT, 2022. *Vodafone.cz* [online]. 2022. [Accessed 10 December 2022]. Available from: https://www.vodafone.cz/podnikatele/internet-veci/nb-iot1/

[39] WANG, Shie-Yuan, CHANG, Jui-En, FAN, Hsin and SUN, Yi-Hsiu, 2020. Performance Comparisons of NB-IoT, LTE Cat-M1, Sigfox, and LoRa Moving at High Speeds in the Air. In: *2020 IEEE Symposium on Computers and Communications (ISCC)* [online]. IEEE. 2020. p. 1-6. ISBN 978-1-7281-8086-1. Available from: https://ieeexplore.ieee.org/document/9219557/

[40] SCHLIENZ, J and RADDINO, D. Narrowband Internet of Things: Whitepaper. [online]. [Accessed 20 May 2023]. Retrieved from: https://www.rohde-schwarz.com/cz/applications/narrowband-internet-of-things-white-paper_230854-314242.html

[41] ZAYAS, Almudena Diaz and MERINO, Pedro, 2017. The 3GPP NB-IoT system architecture for the Internet of Things. *2017 IEEE International Conference on Communications Workshops (ICC Workshops)* [online]. 2017. P. 277-282.

[Accessed 20 May 2023]. DOI: 10.1109/ICCW.2017.7962670. Retrieved from: http://ieeexplore.ieee.org/document/7962670/

[42] MANGALVEDHE, Nitin, RATASUK, Rapeepat and GHOSH, Amitava, 2016. NB-IoT deployment study for low power wide area cellular IoT. *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* [online]. 2016. P. 1-6. [Accessed 22 May 2023]. DOI: 10.1109/PIMRC.2016.7794567. Retrieved from: http://ieeexplore.ieee.org/document/7794567/

[43] POPLI, Sakshi, JHA, Rakesh Kumar and JAIN, Sanjeev, 2019. A Survey on Energy Efficient Narrowband Internet of Things (NBIoT): Architecture, Application and Challenges. *IEEE Access* [online]. 2019. Vol. 7, p. 16739-16776. [Accessed 20 May 2023]. DOI: 10.1109/ACCESS.2018.2881533. Retrieved from: https://ieeexplore.ieee.org/document/8536384/

[44] FELTRIN, Luca, TSOUKANERI, Galini, CONDOLUCI, Massimo, BURATTI, Chiara, MAHMOODI, Toktam, DOHLER, Mischa and VERDONE, Roberto, 2019. Narrowband IoT: A Survey on Downlink and Uplink Perspectives. *IEEE Wireless Communications* [online]. 2019. Vol. 26, no. 1p. 78-86. [Accessed 20 May 2023]. DOI: 10.1109/MWC.2019.1800020. Retrieved from: https://ieeexplore.ieee.org/document/8641430/

[45] *IoT technologies and protocols*, 2022. Azure [online]. 2022. [Accessed 12 December 2022]. Available from: https://azure.microsoft.com/en-us/solutions/iot/iot-technology-protocols/

[46] STUSEK, Martin, ZEMAN, Krystof, MASEK, Pavel, SEDOVA, Jindriska and HOSEK, Jiri, 2019. IoT Protocols for Low-power Massive IoT: A Communication Perspective. In: *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* [online]. IEEE. 2019. p. 1-7. ISBN 978-1-7281-5764-1. Available from: https://ieeexplore.ieee.org/document/8970868/

[47] *User Datagram Protocol*, 1980. [online]. ISI. Available from: https://datatracker.ietf.org/doc/html/rfc768 RFC 768

[48] *TRANSMISSION CONTROL PROTOCOL*, 1980. [online]. Marina del Rey: Information Sciences Institute. Available from: https://datatracker.ietf.org/doc/rfc761/ RFC 761

[49] MQTT: The Standard for IoT Messaging, 2022. [online]. 2022. [Accessed 10 December 2022]. Available from: https://mqtt.org/

[50] Quality of Service (QoS) 0,1, & 2 MQTT Essentials: Part 6, 2022. *HiveMQ* [online]. 2022. [Accessed 10 December 2022]. Available from: https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[51] *The Constrained Application Protocol (CoAP)*, 2014. [online]. Bremen: Internet Engineering Task Force. Available from: https://datatracker.ietf.org/doc/rfc7252/ RFC 7252

[52] *AN4555 Application note: Getting started with STM32L4 Series and STM32L4+ Series hardware development*, November 2022. [online]. Rev 9. [Accessed 28 November 2022]. Available from: https://www.st.com/resource/en/application_note/an4555-getting-started-with-stm32l4-series-and-stm32l4-series-hardware-development-stmicroelectronics.pdf STMicroelectronics NV

[53] *AN2867 Application note: Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs*, August 2022. [online]. Rev 16. [Accessed 28 November 2022]. Available from: https://www.st.com/resource/en/application_note/cd00221665-oscillator-design-guide-for-stm8afals-stm32-mcus-and-mpus-stmicroelectronics.pdf STMicroelectronics NV

[54] *SN65HVD23x 3.3-V CAN Bus Transceivers*, 2018. [online]. Rev O. [Accessed 28 November 2022]. Available from: https://www.ti.com/lit/ds/symlink/sn65hvd232.pdf Texas Instruments Incorporated

[55] *TPS56220x 4.5-V to 17-V Input, 2-A Synchronous Step-Down Voltage Regulator in 6-Pin SOT-23*, 2020. [online]. Revision B. [Accessed 28 November 2022]. Available from: https://www.ti.com/lit/ds/symlink/tps562201.pdf Texas Instruments Incorporated

[56] *BG77 Hardware Design*, 2021. . Shanghai, China: Quectel Wireless Solutions Co.

[57] *BG77 Reference Design*, 2019. . V1.0. Shanghai, China: Quectel Wireless Solutions Co.

[58] *NVT2008; NVT2010: Bidirectional voltage-level translator for open-drain and push-pull applications*, 2014. [online]. Rev. 3. Netherlands: NXP B.V. [Accessed 23 May 2023]. Retrieved from: https://www.nxp.com/docs/en/data-sheet/NVT2008_NVT2010.pdf

Symbols and abbreviations

ACK	Acknowledge
ALDL	Assembly Line Data Link
BOM	Bill of materials
CAN	Controller Area Network
CARB	California Air Resources Board
CoAP	Constrained Application Protocol
DMA	Direct Memory Access
DRX	Discontinuous reception
DTC	Diagnostic trouble codes
ECU	Electronic control unit
eDRX	Extended discontinuous reception
eNB	eNodeB
EOBD	European On-board Diagnostics
EPC	Evolved Packet Core
ESD	Electrostatic discharge
GLONASS	Global navigation satellite system
GNSS	Global navigation satellite system
GPIO	General-purpose input/output
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IC	Integrated circuit
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things

JTAG Joint Test Action Group interface

JSON JavaScript Object Notation

KWP2000 Keyword Protocol 2000

LED Light-emitting diode

LoRa Long Range

LPWAN Low power wide area network

LSB Least significant bit

LTE Long-term Evolution

MCU Microcontroller unit

MME Mobility Management Entity

MOSFET Metal–oxide–semiconductor field-effect transistor

MQTT Message Queue Telemetry Transport

NB-IoT Narrowband Internet of Things network

NRZ Non-return-to Zero coding

OBD On-board diagnostics

OLED Organic light-emitting diode

PCB Printed Circuit Board

PDU Protocol data unit

PGW Packet Data Network Gateway

PID Parameter ID

PSM Power saving mode

PWM Pulse width modulation

RDSS Radio Determination Satellite Service

RNSS Radio Navigation Satellite Service

SAE Society of Automotive Engineers

SCEF	Service Capability Exposure Function
SGW	Serving Gateway
SIM	Subscriber identity module
SMT	Surface mount technology
SPDT	Singe Pole Dual Throw switch
SPI	Serial Peripheral Interface
SPST	Singe Pole Single Throw switch
SWD	Serial Wire Debug
TAU	Tracking area update
TCP	Transmission Control Protocol
THT	Through-hole technology
TTL	Transistor-Transistor Logic
TVS	Transient-voltage-suppression diode
UART	Universal asynchronous receiver-transmitter
UDP	User Datagram Protocol
UE	User equipment
USB	Universal serial bus
VPW	Variable pulse width modulation
3GPP	The 3rd Generation Partnership Project

List of appendices

A OBD II related tables **70**
A.1 List of OBD II CAN Bus identifiers 70

B Board schematic **71**

C Contents of electronic attachment **79**

A OBD II related tables

A.1 List of OBD II CAN Bus identifiers

Tab. A.1: 11 bit CAN identifiers defined by ISO 15765

Identifier	Description
7DF	Functionally addressed request from diagnostic tool
7E0	Physical request from diagnostic tool to ECU #1
7E8	Physical response from ECU #1 to diagnostic tool
7E1	Physical request from diagnostic tool to ECU #2
7E9	Physical response from ECU #2 to diagnostic tool
7E2	Physical request from diagnostic tool to ECU #3
7EA	Physical response from ECU #3 to diagnostic tool
7E3	Physical request from diagnostic tool to ECU #4
7EB	Physical response from ECU #4 to diagnostic tool
7E4	Physical request from diagnostic tool to ECU #5
7EC	Physical response from ECU #5 to diagnostic tool
7E5	Physical request from diagnostic tool to ECU #6
7ED	Physical response from ECU #6 to diagnostic tool
7E6	Physical request from diagnostic tool to ECU #7
7EE	Physical response from ECU #7 to diagnostic tool
7E7	Physical request from diagnostic tool to ECU #8
7EF	Physical response from ECU #8 to diagnostic tool

B Board schematic

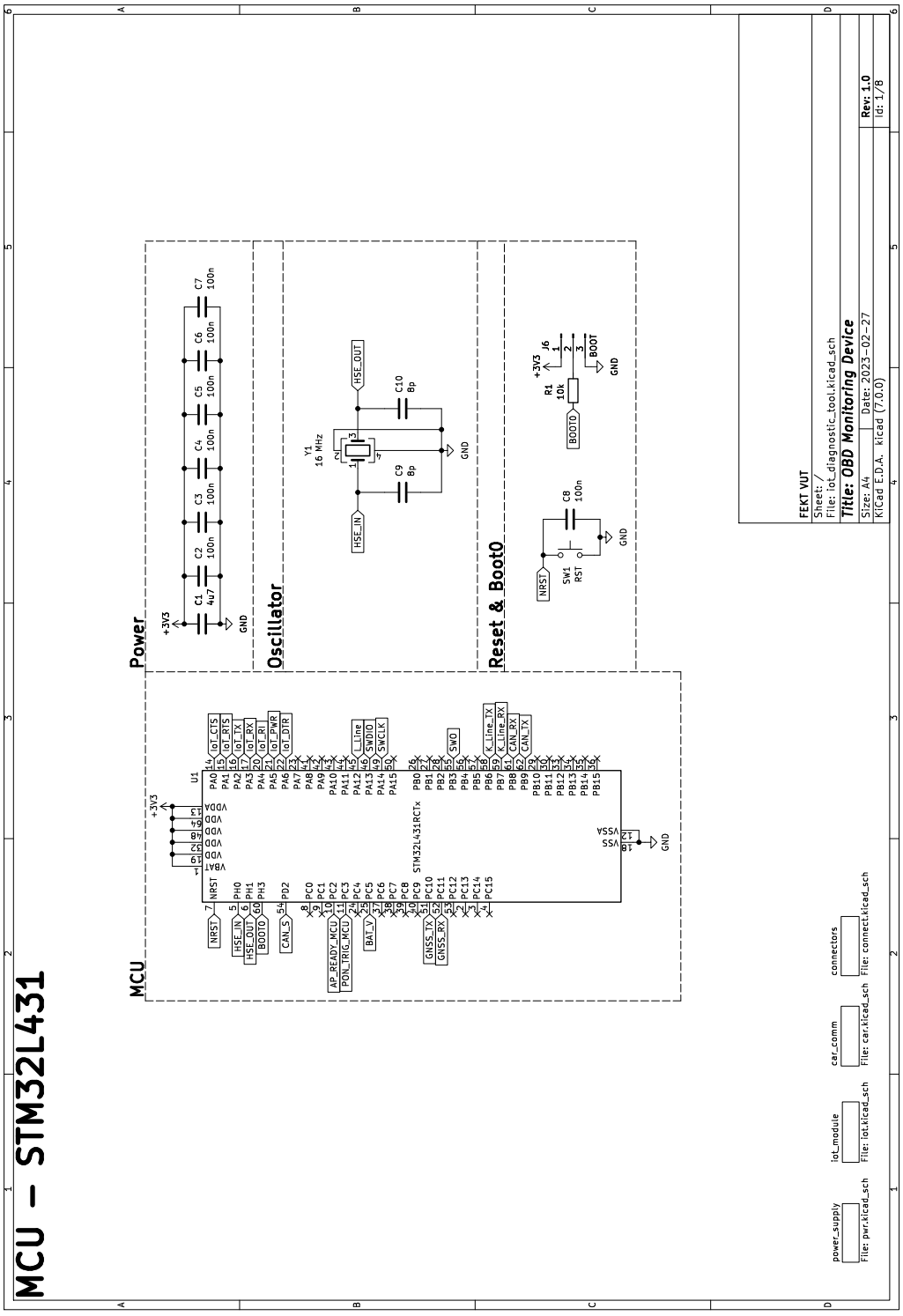

Fig. B.1: Schematic – Main MCU

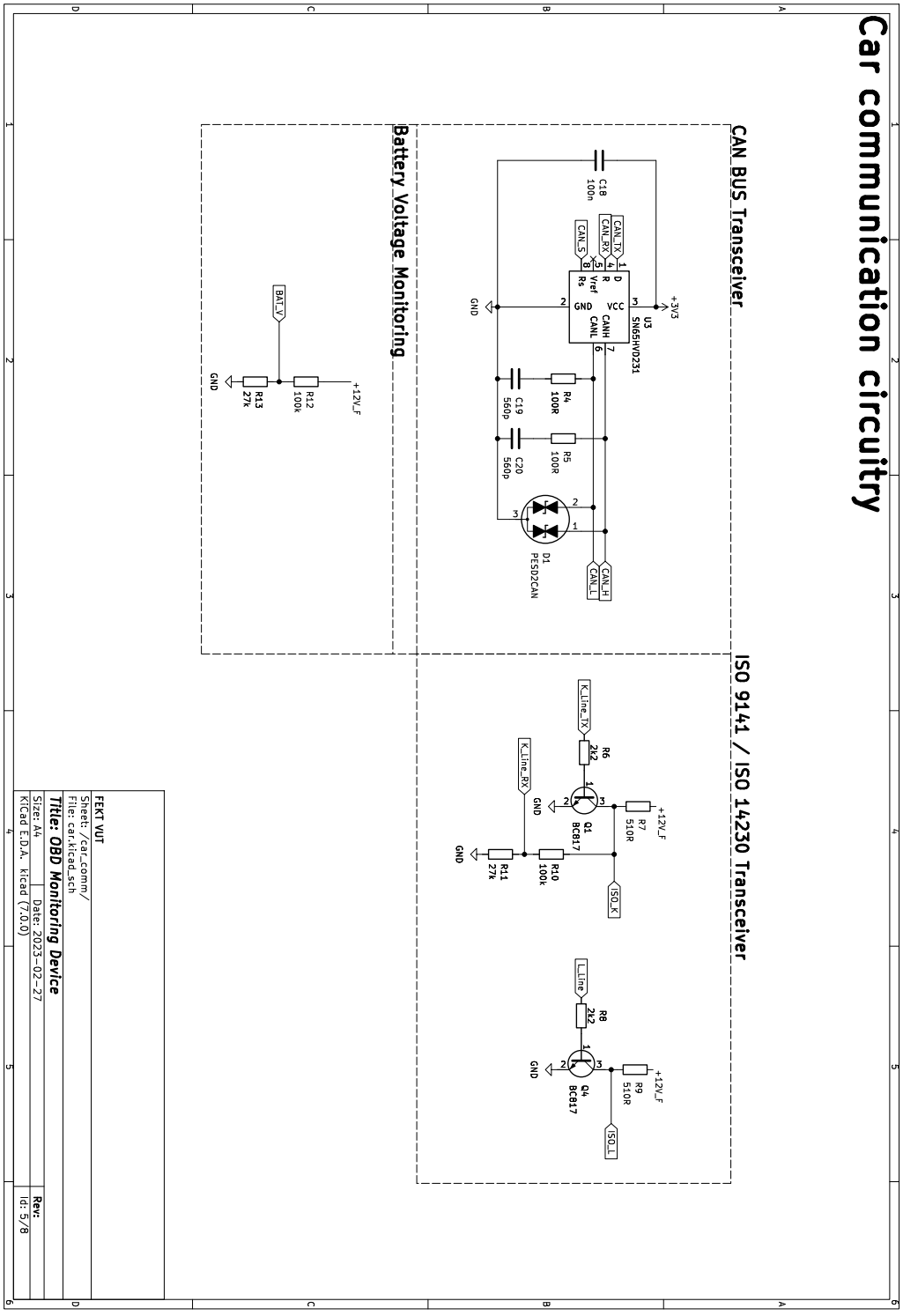

Fig. B.2: Schematic – OBD II interface

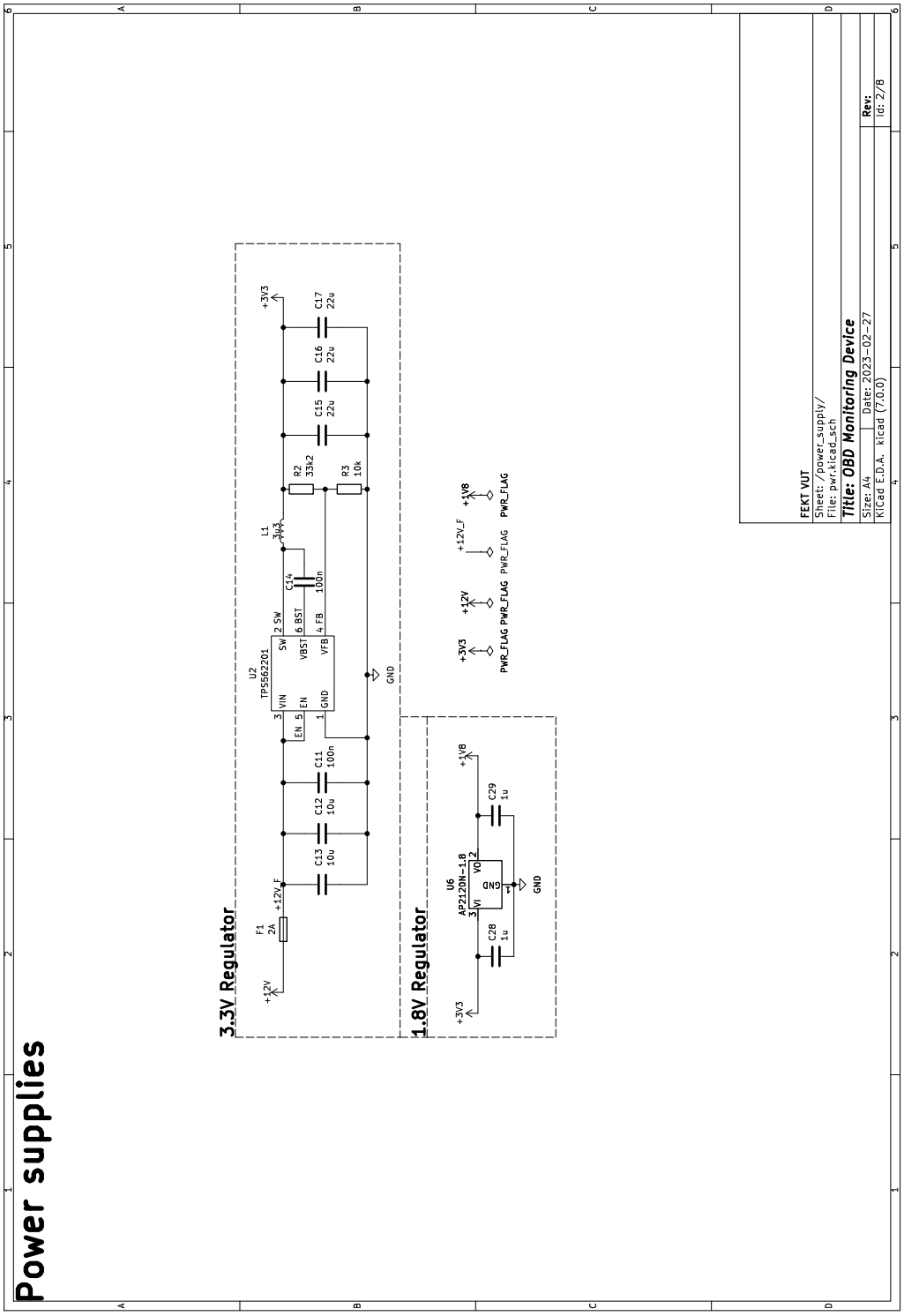

Fig. B.3: Schematic – Power supplies

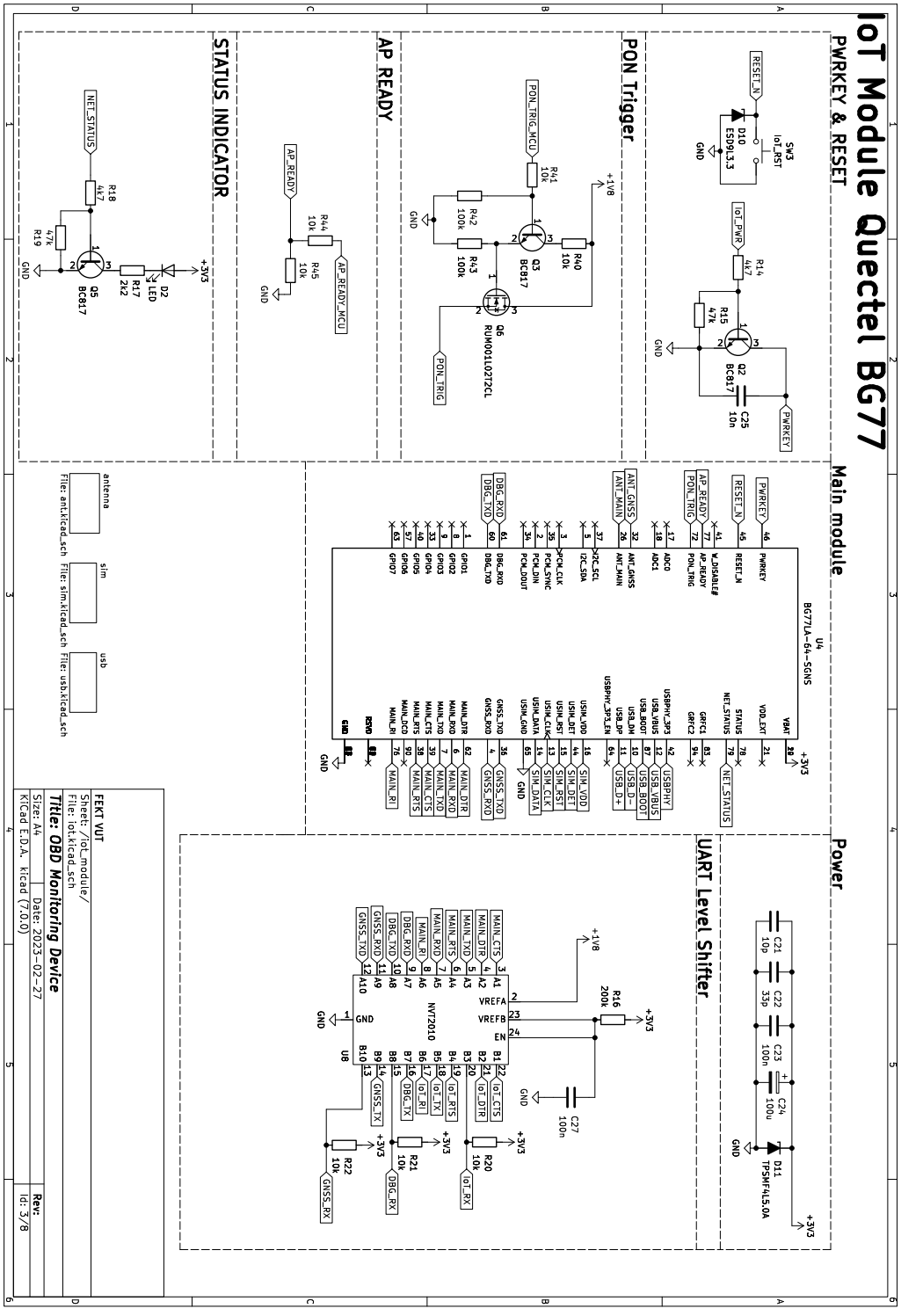

Fig. B.4: Schematic – NB-IoT module

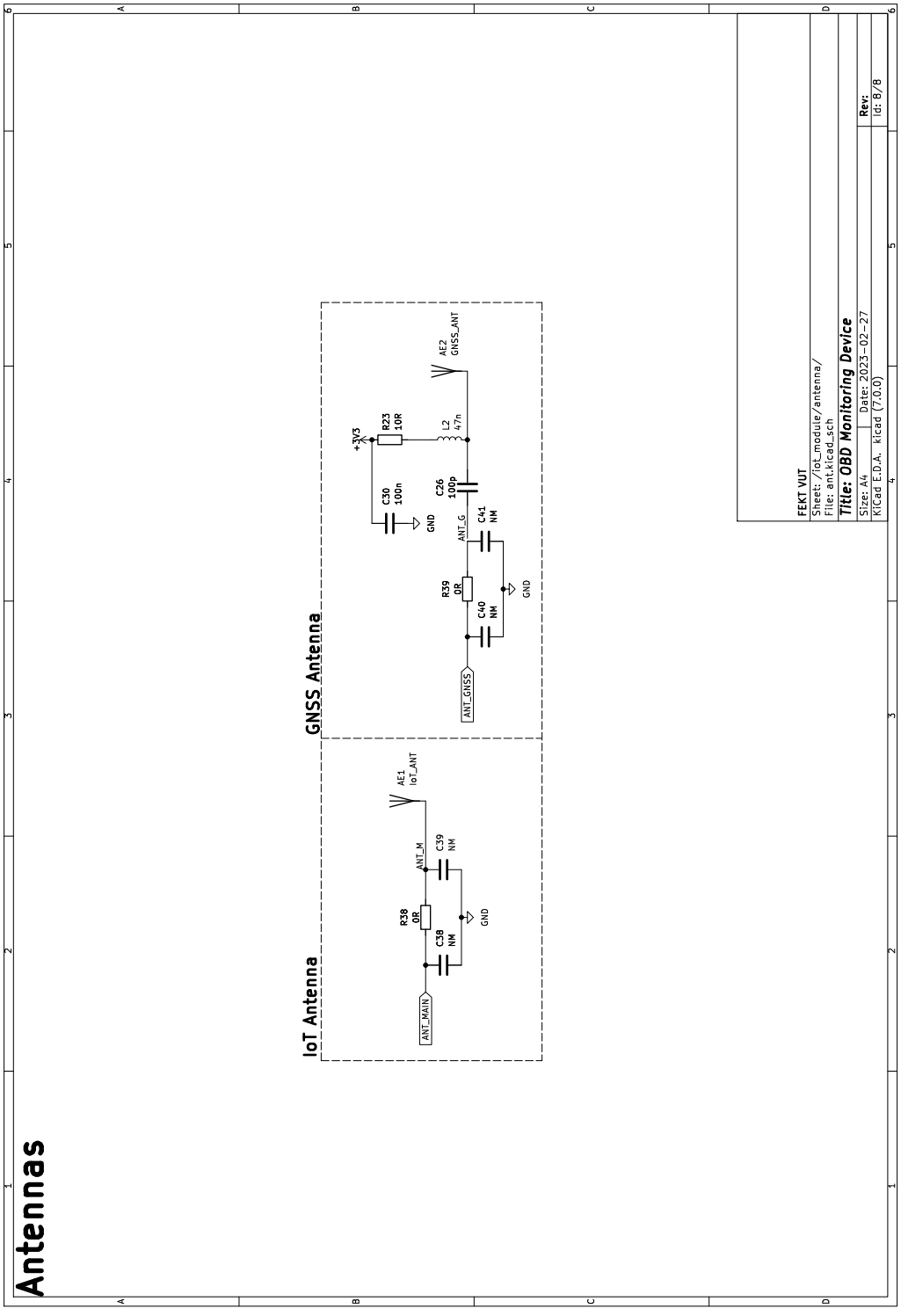

Fig. B.5: Schematic – Antenna interface

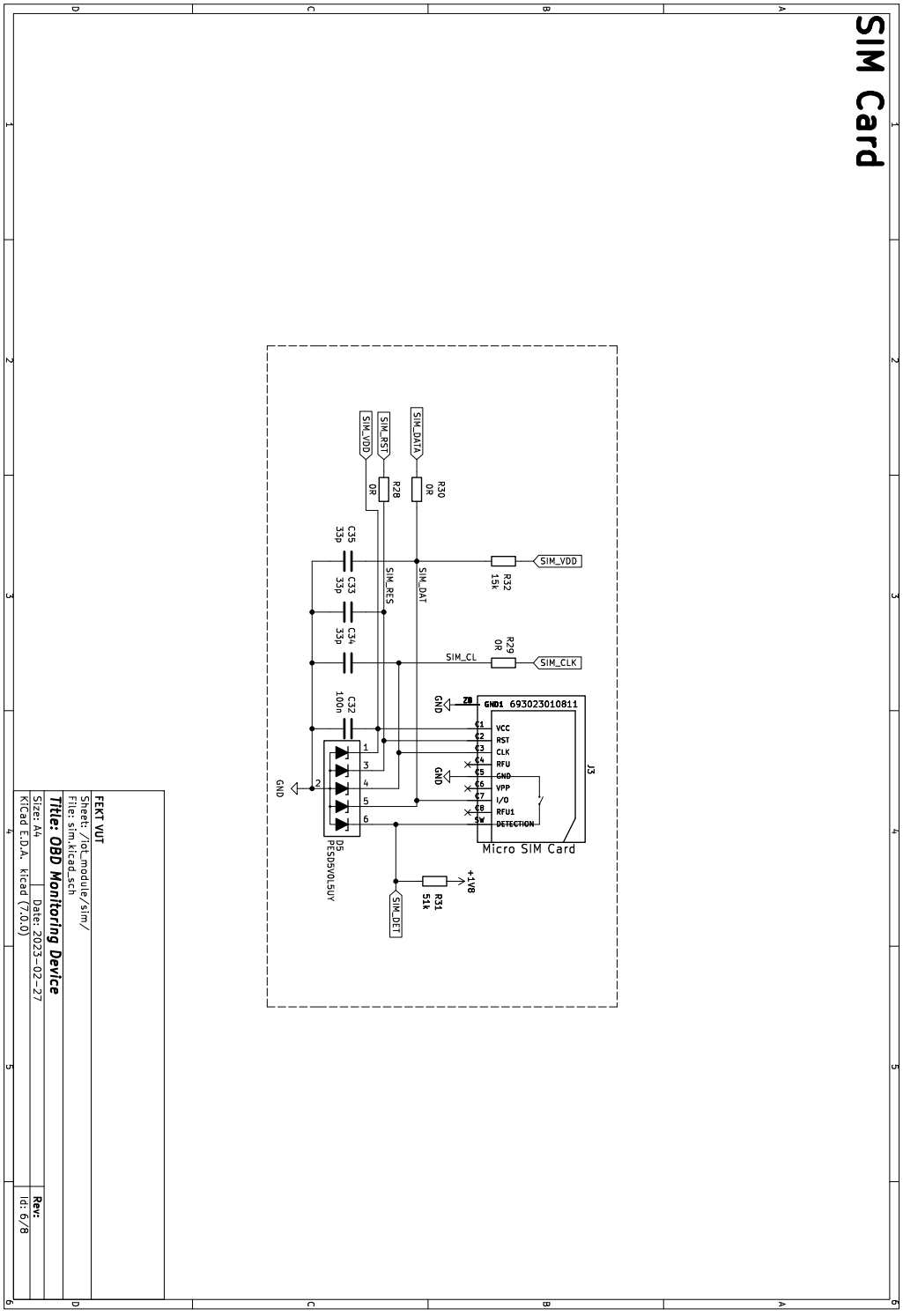

Fig. B.6: Schematic – SIM card interface

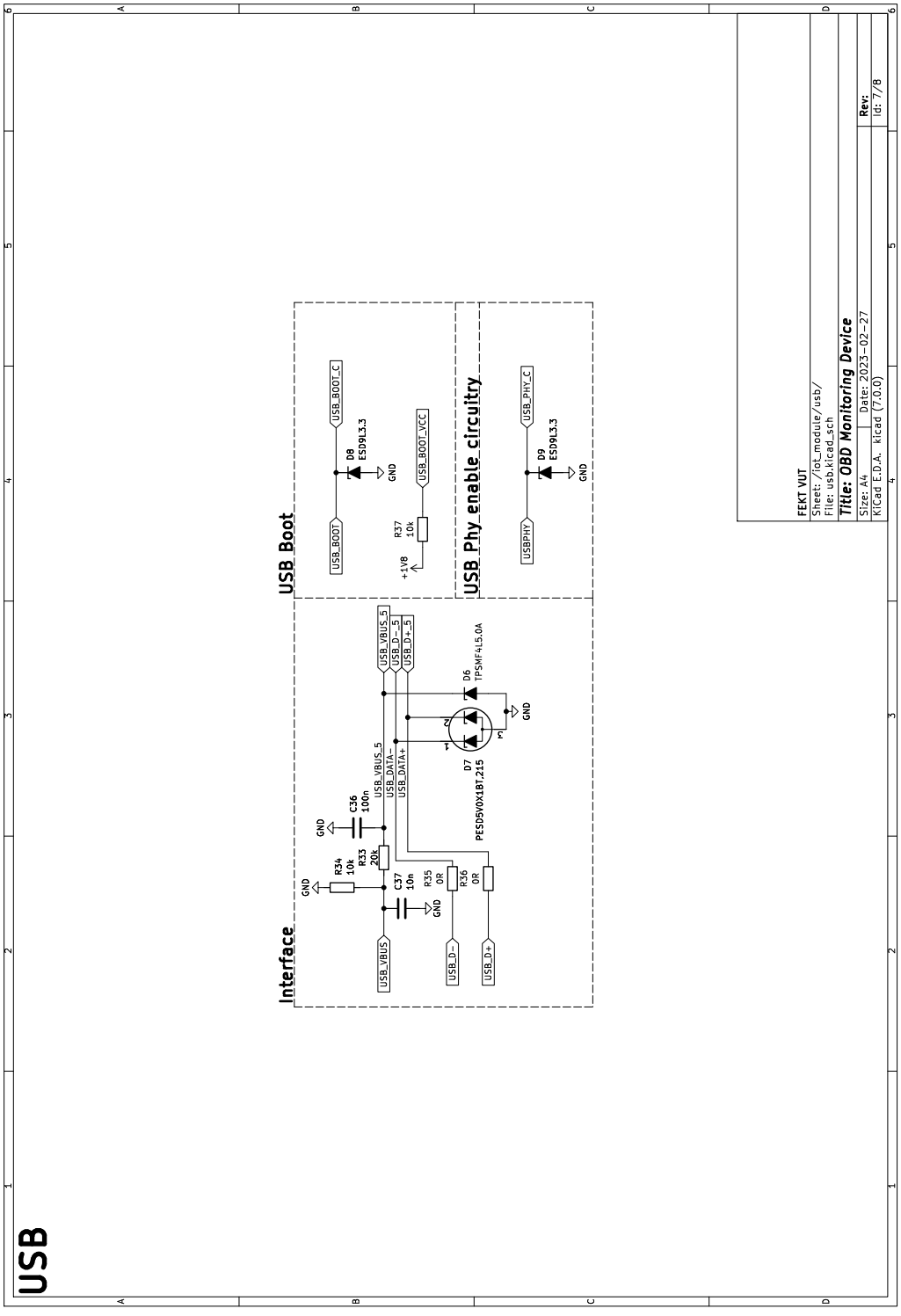

Fig. B.7: Schematic – USB interface

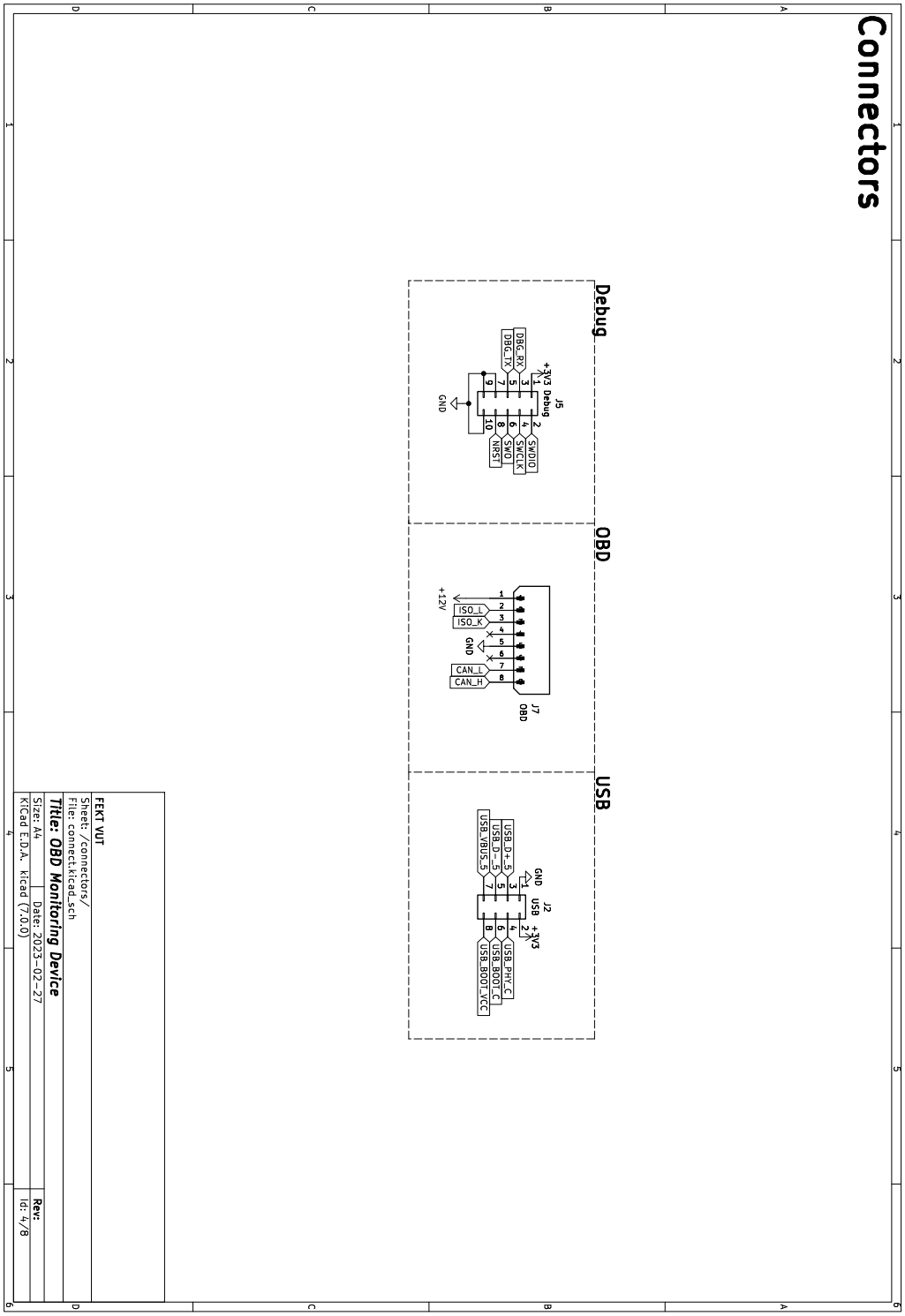

Fig. B.8: Schematic – Connectors

C Contents of electronic attachment

The electronic attachement contains KiCAD project for the PCB, gerber files for PCB manufacturing and STM32 project with the whole program for the MCU. The KiCAD project has been made using KiCAD 7.0.0. The case has been designed using Autodesk Inventor 2024.

The software portion of the attachment has been made using STM32CubeIDE 1.12.1. The latest version can be found on github[1].

- `/` — Root directory of the attached archive
  - `Case` — Design files for the 3D printed case
    - `base.ipt` — Base part – Inventor part
    - `base.stl` — Base part – STL part ready for 3D printing
    - `case.iam` — Sase assembly – Inventor Assembly
    - `iot_diagnostic_tool.iam` — Board assembly – Inventor Assembly
    - `top.ipt` — Top part – Inventor part
    - `top.stl` — Top part – STL part ready for 3D printing
  - `iot_obd_monitoring_device` — Software project
    - `Core` — Main source code
    - `Drivers` — Drivers and libraries
    - `.cproject`
    - `.mxproject`
    - `.project`
    - `iot_obd_monitoring_device.ioc`
    - `STM32L431RCTX_FLASH.ld`
  - `iot_diagnostic_tool` — KiCAD project directory
    - `jlcpcb` — Manufacturing files
      - `assembly` — BOM and POS files for assembly
      - `gerber` — Gerber files for board manufacturing
    - `ant.kicad_sch` — KiCAD schematic for antennas
    - `car.kicad_sch` — KiCAD schematic for car communication
    - `connect.kicad_sch` — KiCAD schematic for connectors
    - `iot.kicad_sch` — KiCAD schematic for IoT module
    - `iot_diagnostic_tool.kicad_pcb` — KiCAD main PCB
    - `iot_diagnostic_tool.kicad_prl`
    - `iot_diagnostic_tool.kicad_pro` — Main KiCAD project
    - `iot_diagnostic_tool.kicad_sch` — Main KiCAD schematic
    - `pwr.kicad_sch` — KiCAD schematic for power supplies
    - `sim.kicad_sch` — KiCAD schematic for SIM card slot
    - `usb.kicad_sch` — KiCAD schematic for USB
    - `vut_logo.kicad_mod` — VUT logo for silkscreen
  - `Schematic` — PDF version of board schematic
    - `iot_diagnostic_tool.pdf`

[1] `https://github.com/slimakk/secured-vehicle-monitoring`