VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

VZDÁLENÉ MONITOROVÁNÍ VYBRANÝCH SUBSYSTÉMŮ VOZIDLA

REMOTE MONITORING OF SELECTED VEHICLE SUBSYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETER DRAHOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Drahovský Peter, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové a vestavěné systémy
Název: **Vzdálené monitorování vybraných subsystémů vozidla**
Remote Monitoring of Selected Vehicle Subsystems
Kategorie: Vestavěné systémy
Zadání:

1. Prostudujte problematiku monitorovacích a řídicích systémů ve vozidlech. Zaměřte se na CAN BUS a OBD-II.
2. Seznamte se s problematikou, architekturami a prostředky pro realizaci IoT systémů.
3. Navrhněte vestavěný systém (HW i SW) a cloudovou aplikaci pro odposlouchávání a pro komunikaci na sběrnici CAN BUS ve vozidle. Umožněte lokální ukládání a akumulaci vybraných dat ve vestavěném systému, aby mohl dočasně pracovat i v offline režimu. Připojení k IoT bráně řešte vhodným bezdrátovým spojením (Bluetooth, WiFi, GSM apod.). Cloudová aplikace umožní analýzu a prezentaci jízdních dat, nastavování vestavěného systému a detekci a zpracování alarmů.
4. Navržené prostředky realizujte a ověřte funkčnost systému v reálném provozu.
5. Vyhodnoťte dosažené výsledky a diskutujte bezpečnostní aspekty tohoto projektu.

Literatura:

- Data Exchange On The CAN Bus I.
  URL: http://www.volkspage.net/technik/ssp/ssp/SSP_238.pdf
- Data transfer on CAN data bus II.
  URL: http://www.volkspage.net/technik/ssp/ssp/SSP_269_d1.pdf
- Currie, R.: Hacking the CAN Bus: Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering. 2017. URL: https://www.giac.org/paper/gcia/9927/ hacking-bus-basic-manipulation-modern-automobile-bus-reverse-engineering/133228

Při obhajobě semestrální části projektu je požadováno:

- První 3 body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Tato práce se zabývá návrhem a realizací vestavěného systému a cloudové aplikace, umožňující vzdálené monitorování subsystémů vozidla skrze sběrnici CAN. Jsou zde popsány možné způsoby získávání informací o vozidle a postup dekódování zpráv zasílaných na interní sběrnici automobilu. Pro získávání dat z dekódovaných zpráv bylo vytvořeno zařízení na bázi ESP32, které je v pravidelných intervalech zasílá skrze bezdrátové sítě do cloudové aplikace. Cloudová aplikace poskytuje webové rozhraní uzpůsobené mobilnímu telefonu, ve kterém je možné zobrazovat získaná aktuální data i jejich historický vývoj.

Abstract

This thesis describes the concept and realization of embedded system and cloud application, which is designed for remote monitoring of car subsystems over CAN bus. There are written several ways of getting information from car subsystems and process of decoding messages sent over internal car bus. Data gathering is realised by device based on ESP32, which is sending them to cloud application using wireless networks. Application UI is adapted for mobile devices and serves views containg current and historical gathered data.

Klíčová slova

CAN sběrnice, OBD, ESP32, vzdálené monitorování

Keywords

CAN bus, OBD, ESP32, remote monitoring

Citace

DRAHOVSKÝ, Peter. *Vzdálené monitorování vybraných subsystémů vozidla.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Vladimír Janoušek, Ph.D.

Vzdálené monitorování vybraných subsystémů vozidla

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením pana doc. Ing. Vladimíra Janouška Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Peter Drahovský
12. května 2021

Poděkování

Touto cestou by jsem chtěl poděkovat doc. Ing. Vladimíru Janouškovi Ph.D. za odbornou pomoc a vedení této práce.

Obsah

Kapitola 1

Úvod

1.1 Úvod

Sběrnice CAN se začala využívat v automobilech již před 20 lety a to zejména pro snížení nákladů, hmotnosti, ale i zajištění spolehlivosti komunikace. V dnešních dnech již automobily typicky využívají pro interní komunikaci několik CAN sběrnic. Informace získané odposloucháváním komunikace na těchto sběrnicích je možné využít při monitorování a ovládání některých subsystémů vozidla.

Cílem této práce je vytvořit vestavěný systém, který bude umožňovat vzdálené monitorování vozidla, vyhodnocování získaných dat a případné ovládání některých subsystémů. Prostředkem pro komunikaci s automobilem byla zvolena zmiňovaná CAN sběrnice, která bude odposlouchávaná zařízením umístěným uvnitř vozidla. Toto zařízení bude mít trvalé napájení z autobaterie, a je tedy nutné, aby bylo energeticky šetrné. V pravidelných intervalech bude komunikovat skrze vhodný bezdrátový komunikační kanál se serverem a informovat tak o aktuálním stavu vozidla. Informace získané tímto způsobem budou zpracovány a zobrazovány uživateli skrze webovou aplikaci pro mobilní telefon. Aplikace bude poskytovat jednoduché rozhraní pro zobrazení informací o automobilu a možnosti jeho ovládání.

Díky odposlouchávání interní sběrnice automobilu je možné získávat také údaje o vozidle, ke kterým uživatel běžně nemá přístup. Při zpracování by však mělo dojít k filtraci zbytečných údajů a jejich případné agregaci, aby byla práce s aplikací přehledná a přiblížila se tak k továrním řešením vzdáleného monitorování.

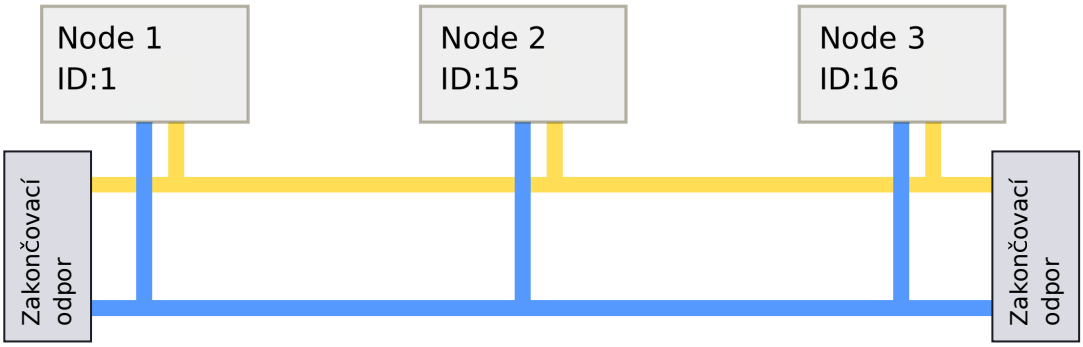
Kapitola 2

Shrnutí nastudované teorie

2.1 Protokol CAN a jeho využívání ve vozidlech VW

Controller Area Network (označován také jako CAN) je sériový komunikační protokol typu multi–master, vyvinutý společností BOSCH již v letech 1983 – 1985. V roce 1996 byla vydána specifikace `CAN2.0` [13]. Tato specifikace byla později rozšířena, zejména kvůli zvětšení maximálního možného počtu prvků na jedné sběrnici. Základní typ je označován `CAN 2.0A` a rozšířený typ `CAN2.0B`. Ten je částečně zpětně kompatibilní. V roce 1993 byl CAN definován také v mezinárodním standardu v sérii ISO 11898.

Prvky připojené ke sběrnici jsou nazývány uzly. Minimální počet uzlů na jedné sběrnici je 2, maximální závisí od verze specifikace. Teoretická maximální rychlost je 1 Mbit/s. Standardní přenosové médium je kroucená dvoulinka, maximální délka sběrnice je závislá od přenosové rychlosti (od 25 metrů pro 1 Mbit/s až po 1 Km pro 50 Kbit/s). Stavy na sběrnici jsou buď logická 1 (recesivní), nebo logická 0 (dominantní). Stav nečinnosti je reprezentován recesivní úrovní. Při vysílání log 1 musí vysílající uzel ponechat sběrnici v klidovém stavu. Této vlastnosti je využíváno také při bitové arbitráži a detekci kolizí.



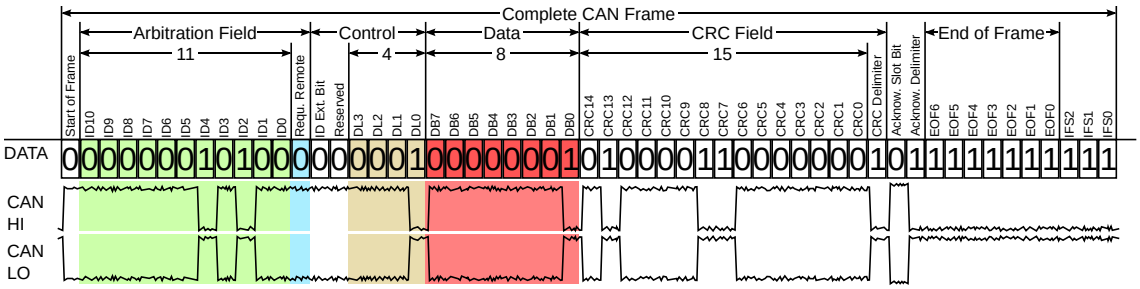
Obrázek 2.1: Ukázka propojení jednotek

Vysílající uzel musí být schopen ze sběrnice v daný moment také číst její reálný stav. Díky tomuto jsou uzly schopny detekovat kolize na sběrnici. Každý komunikující uzel na sběrnici má jedinečné ID, které je využíváno pro identifikaci, ale také při určování priority odesílání zpráv (čím nižší ID, tím vyšší priorita). Bitová arbitráž zajišťuje, že v průběhu 13 (standardní formát) nebo 33 (rozšířený formát) bitových period se určí uzel, který může vysílat na sběrnici. Ten poté nepřerušeně pokračuje v odesílání, zatímco ostatní uzly komunikaci pozastaví a čekají. Rozhodování je prováděno pomocí již zmiňovaného ID, které se

	Startovní bit	ID bitu											Zbytek rámce
		10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Uzel 15	0	0	0	0	0	0	0	0	1	1	1	1	...
Uzel 16	0	0	0	0	0	0	0	1	Zastavené vysílání				
Sběrnice	0	0	0	0	0	0	0	0	1	1	1	1	...

Tabulka 2.1: Kolize při současném vysílání uzlů 15 a 16

nachází na začátku odesílané zprávy. Uzly s nižší prioritou (vyšší ID) musí zapsat logickou jednotku dříve, než uzly s vyšší prioritou. V tomto momentě nastane na sběrnici kolize (tabulka 2.1), kterou detekuje pouze vysílající uzel s nižší prioritou (na sběrnici nevysílá log 0, ale přesto je sběrnice v takovém stavu). Ustoupí tedy a čeká na uvolnění sběrnice.



Obrázek 2.2: Datová zpráva na sběrnici CAN (převzato, upraveno)[21]

Sběrnice CAN obsahuje 4 druhy datových rámců:

Datová zpráva[21] (obrázek 2.2):

- SOF – Start Of Frame značí počátek komunikace a slouží ke synchronizaci přijímačů
- ID – Identifikátor vysílajícího uzlu. V standartu CAN2.0A má 11 bitů, v rozšířeném CAN2.0B až 29 bitů.
- RTR – Remote transmisison request, pro datové zprávy dominantní (0), pro zprávy typu žádost recesivní (1). Tím je zajištěna priorita datových zpráv případné kolizi.
- IDE – Identifier extension bit, dominantní (0) pro 11 bitový identifikátor, recesivní (1) pro 29 bitový identifikátor.
- RSV – Rezervovaný bit – nepoužívaný
- DLC – Data length code udává počet datových bytů (0-8)
- DATA – Přenášená data
- CRC – obsahuje 15 bitů dlouhý kontrolní kód pro kontrolu zprávy
- CRC – oddělení CRC a ACK, musí být recesivní (1)
- ACK – Vysílající uzel odešle recesivní bit (1) a jakýkoliv přijímající uzel potvrdí dominantním bitem (0).

- ACK – oddělení ACK od EOF, musí být recesivní (1)
- EOF – End of frame musí být recesivní (1)

Žádost o data Protokol CAN umožňuje také odeslat požadavek na získání dat. Takováto zpráva má potom stejný identifikátor jako uzel, od kterého jsou data požadována. Zpráva se od datové liší v RTR bitu a absenci dat ve zprávě. DLC takovéto zprávy určuje požadovanou, nikoliv vysílanou velikost dat.

Chybová zpráva je odvysílána každým uzlem, který detekoval chybu na sběrnici. Existují aktivní a pasivní chybové zprávy. Aktivní se skládá ze šesti dominantních bitů, pasivní potom ze šesti recesivních bitů. Následuje chybový kód (`Error flag`) skládající se ze 6 – 12 bitů, po němž následuje oddělovač (8 recesivních bitů).

Zpráva o přetížení je vysílána uzlem, který požaduje oddálení vysílání dalších zpráv.

Využívání CAN systémů u automobilů koncernu Volkswagen

CAN sběrnice se pro své vhodné vlastnosti začala využívat také v automobilovém průmyslu. Výhody použití sběrnicového systému jsou například snížení nákladů, zjednodušení kabelových rozvodů, snížení celkové hmotnosti, ale také například zjednodušení diagnostiky poruchy. V roce 1997[17] byla sběrnice CAN použita v prvním produkčním koncernovém vozidle. Jednalo se o model Passat, který používal sběrnici s rychlostí 62.5 Kbit/s, označovanou jako `convenience`. Postupně se ve vozidlech začaly využívat i další sběrnice, jejichž počet se různí mezi modely, ale i ročníky.

Touran 1T1

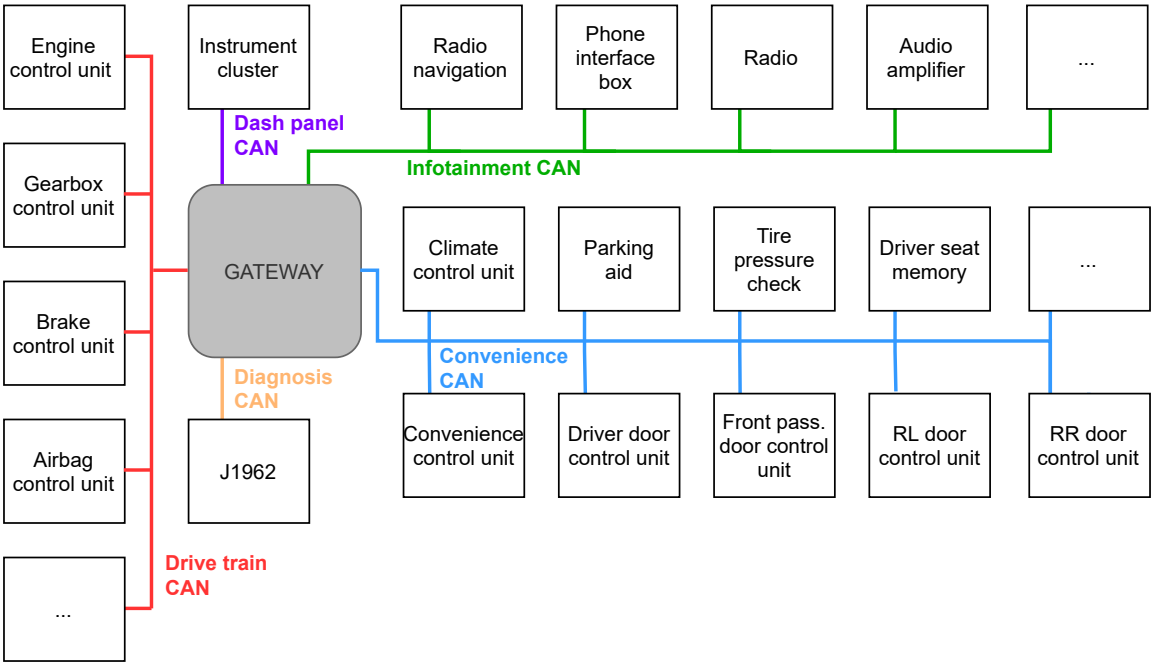
Řešení této diplomové práce je vázáno (zejména) k automobilu Touran 1T, vyrobeném v roce 2003. Princip rozdělení sběrnic je nicméně podobný i ve všech novějších vozidlech. Lišit se budou například rychlostmi sběrnic. Ve vozidle Touran se celkem nachází 5 CAN sběrnic[19].

Popis sběrnice	**Rychlost sběrnice**
Datová sběrnice CAN hnacího ústrojí	500 Kbit/s
Datová sběrnice CAN komfortního systému	100 Kbit/s
Datová sběrnice CAN informatiky	100 Kbit/s
Datová sběrnice CAN přístrojového panelu	500 Kbit/s
Datová sběrnice CAN diagnostiky	nezjištěno

Tabulka 2.2: Sběrnice CAN ve vozidle Touran 1T1

Všechny sběrnice jsou připojené k modulu zvanému brána (*v angličtině zaužívaný název gateway*)[18]. Tento modul je schopen vybrané zprávy přeposílat mezi různými sběrnicemi, co umožňuje i vzájemnou komunikaci zařízením umístěných na různých sběrnicích. Také umožňuje diagnostiku jednotek napříč sběrnicemi skrze diagnostickou sběrnici (připojená k diagnostickému portu). Propojení sběrnic je znázorněno na obrázku 2.3.

Sběrnice hnacího ústrojí zřejmě nebude z pohledu získávání relevantních dat příliš zajímavá. Nacházejí se zde např. brzdné jednotky, motorová jednotka, jednotka snímající



Obrázek 2.3: Propojení sběrnic ve vozidle Touran 1T1

natočení volantu, nebo jednotka airbagů. Monitorování a zpracování těchto dat by mělo smysl spíše pro servisní účely. Nesprávný zásah do této sběrnice by také mohl mít negativní dopad na bezpečnost provozování automobilu.

Naproti tomu na sběrnici komfortního systému se budou nacházet jednotky topení/klimatizace, dveřní jednotky, centrální zamykání, ovládání oken, ale i alarm.

Sběrnice informatiky (*v anglickém názvu Infotainment – info + entertainment*) obsahuje potom například rádio, telefonní jednotku, navigaci, nebo například informační panel před řidičem.

Datová sběrnice přístrojového panelu je jakousi agregací předešlých sběrnic. Jsou na ni přeposílány, jak stavy ze sběrnice hnacího ústrojí, tak zprávy z komfortního systému a informatiky. Data získaná napříč sběrnicemi jsou potom zobrazována na displeji před volantem (*u koncernu Volkswagen zvaný MFD plus, nebo Maxi-dot*).

Konkrétní jednotky na sběrnicích, jejich ID a odesílaná data se mi nepodařilo pro tento konkrétní model vozidla dohledat. Pro získávání dat bude tedy nutné odposlouchávat komunikaci na sběrnici a zasílaná data postupně dekódovat. Inspirací mohou být seznamy dekódovaných obsahů zpráv z jiných koncernových vozidel. Zajímavý materiálem je například odkaz na google spreadsheet `MKV GTI CAN-BUS data`[1], který se objevuje na internetu v diskusních fórech. Původní zdroj či autora nebylo možné dohledat.

Popis CAN zpráv pomocí formátu DBC

Pro své použití v real–time systémech je nutné, aby zasílané rámce obsahovaly pouze nezbytně nutná data a to pokud možno v co nejoptimálnější podobě. Datové rámce jsou tedy typicky složeny z mnoha informací a vzniká potřeba popsání jejich získání z daného rámce. Pro popis dat obsažených v rámcích je možné využít například textový formát DBC. Tento

[1] https://docs.google.com/spreadsheets/d/1eirT8LbSRl4j06BpwgsiE4PM_2BGH9UStdWLXwKvHJw/

formát umožňuje popsat jednotlivé obsahy rámců (nazývané signály) a nutné matematické úpravy pro jejich získání. S tímto formátem pracuje také mnoho softwarů pro analyzování komunikace na CAN sběrnici.

2.2 OBD

OBD, neboli On-Board Diagnostic[22] je souhrn norem primárně určených na kontrolování podsystémů vozidla příčinných na vzniku vypouštěných emisí. Postupem času se vyvinuly normy OBD I a později OBD II, které budou přiblíženy v následujících odstavcích. Pro podobnost EOBD a OBD II budou pro účely této práce po krátkém přehledu sjednoceny a nebude dbáno na jejich drobné odlišnosti.

OBD I

Norma OBD I vznikla z nařízení Kalifornského úřadu pro čistotu vzduchu a jejím účelem bylo kontrolování spalin při provozu vozidla. Pro svou jednoduchost však systém dokázal pouze kontrolovat chybovost komponentů podstatných pro vznik emisí. V případě nalezené chyby byla zaznamenána do paměti řídící jednotky a řidič byl informován skrze kontrolku MIL (Malfunction Indicator Light). Přesnější určení závady bylo možné vyvodit z frekvence blikání této kontrolky, případně skrze diagnostický port. Typ a umístnění tohoto portu nebyly jednotné pro všechna vozidla a komunikace probíhala skrze K-line.

OBD II

OBD II je novější americká norma z roku 1994, také určená k diagnostice emisních systémů osobních automobilů. V USA je jeho plnění povinné pro automobily od roku 1996. Norma je přísnější než předešlá, nařizuje sledovaní více podsystémů (účinnost katalyzátoru nebo recirkulaci spalin). V případě chyby je do paměti uložena nejen příslušná chyba, ale také stavy ostatních systémů (tzv. Freez frame). Tyto data mohou pomoci při hledání problému. Komunikace s uživatelem vozidla zůstává stále skrze kontrolku MIL (český ekvivalent je např. „*motor do dílny*“), bylo však exaktně upřesněno její chování. Kontrolka je rozsvícena před startem vozidla a při bezproblémové funkci musí po nastartování zhasnout. Pokud kontrolka zůstane svítit, případně bliká, značí to nalezenou chybu.

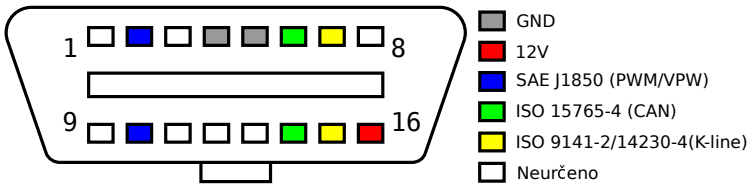
EOBD

V Evropě byla v roce 1991 vydána norma *DIN ISO 9141-2*. Obsah komunikačních protokolů, struktura a diagnostická zásuvka byla víceméně převzata z americké normy[5]. Od roku 2001 (pro benzínové motory), respektive 2003 (pro naftové motory) je tedy povinný evropský ekvivalent nazývaný EOBD.

Souhrný popis OBD II a EODB

V těchto normách je určen konektor (J1962), který musí být umístěn v dosažitelném místě z místa řidiče, maximálně však 50 cm od volantu. Na tomto 16 pinovém konektoru je dle příslušných norem vyvedených jedno z možných komunikačních rozhraní.

Norma umožňuje komunikaci skrze několik možných komunikačních protokolů. Ostatní piny, které nejsou určeny normou mohou automobilky osadit vlastním diagnostickým roz-



Obrázek 2.4: Konektor J1962 – určený standardem OBDII (převzato, upraveno)[22]

hraním. Typicky tak i udělají a vedle standardizované OBD komunikace bude na volných pinech dostupná i proprietární koncernová diagnostika.

Popis komunikačních protokolů[6][16][3]

SAE J1850 PWM Protokol využívá pro komunikaci pulzně šířkovou modulaci, používá piny 2 a 10. Hojně je využíván ve vozidlech značky Ford.

SAE J1850 VPW Podobný jako předchozí, taktéž využívá pulzně šířkovou modulaci, ale používá proměnlivou šířku impulzu. Využíván například u vozidel General Motors.

ISO 9141-2, ISO 14230-4 (KWP 2000) Využívá dvě linky K a L, na pinech 7 a 15 je nazývaný také K-line. Nevýhoda protokolu je pomalá inicializace spojení. ISO 14230-4 umožňuje rychlejší inicializaci. Využívají ho například starší vozy koncernu Volkswagen.

ISO 15765 CAN, SAE J2248 CAN Protokoly pracující na komunikační lince CAN. Vysoká rychlost přenosu, odolnost vůči rušení. Přiřazeny piny 6 a 14. Využíván je například v nových vozidlech koncernu Volkswagen.

Diagnostické režimy – v standartu OBD umožňují skrze jeden z komunikačních protokolů získávání různých údajů o vozidle. Celkem existuje 10 diagnostických režimů s adresami x01 až x0A[5]. Výrobci nejsou povinní implementovat všechny a zároveň mohou využít vyšší adresy pro implementaci vlastních módů.

Adresa	Popis
01	Měřené hodnoty
02	Freeze frame
03	Paměť trvalých závad
04	Mazání paměti závad
05	Hodnoty lambda
06	Průběžný test
07	Paměť sporadických závad
08	Testy akčních členů
09	Identifikace vozidla
0A	Paměť historie závad

Tabulka 2.3: Diagnostické režimy definované normou

- Režim 1 umožňuje sledovat naměřené hodnoty v reálném čase. Parametry vozidla, které lze sledovat jsou například rychlost vozidla, otáčky motoru, teplota kapalin, nebo složení směsi.
- Režim 2 obsahuje tzv. Freeze frames, což jsou data z ostatních monitorovaných jednotek, v momentě, kdy došlo k vzniku a uložení chybového kódu.
- Režim 3 obsahuje trvalé závady. Pokud je zde uložena závada svítí také diagnostická kontrolka.
- Režim 4 umožňuje odstranit chybové záznamy a freeze frames z paměti.
- Režim 5 obsahuje data související s lambda sondou
- Režim 6 není definován normou
- Režim 7 obsahuje chyby, které jsou natolik sporadické, že nejsou zařazeny do trvalých závad.
- Režim 8 umožňuje provádění testů akčních členů.
- Režim 9 umožňuje získání identifikace vozidla (VIN)
- Režim 10 obsahuje již smazané závady, které byly natolik vážné, že budou smazány až po několika jízdních cyklech

Rozhraní poskytuje výpis chyb z (z pohledu tvorby emisí) nejdůležitějších systémů vozidla. Diagnostika skrze OBD II může být první možností při hledání chyby na vozidle, čemu nahrává také dnešní cena diagnostik komunikujících skrze standard OBD. Dokonce je možné si takovýto základní přehled zobrazit i jako uživatel vozidla. Bohužel výsledky této diagnostiky ukazují spíše následky než příčiny problémů. Nakonec je tedy většinou nutné sáhnout po proprietární diagnostice určené pro konkrétní značku, nebo dokonce automobil.

Výsledkem diagnostiky je seznam chybových kódů (označované `DTC - Diagnostic Trouble Code`). Chybový kód se skládá z písmene a čtyř číslic. Písmeno určuje jednu z kategorií (tabulka 2.4), čísla potom konkrétní závadu v kategorii. Norma určuje běžné chybové kódy, přičemž výrobci mají možnost specifikovat také vlastní.

X	**Popis**
B	Body
C	Chassis
P	Powertrain
U	Network

Tabulka 2.4: Kategorie v chybových kódech OBD II

Získávání telemetrických dat

Tyto rozhraní jsou často využívána také k telemetriím. Využití tohoto rozhraní je dobré zejména pro svou obrovskou rozšířenost, protože jej dnes podporuje většina aut. Na druhou stranu tímto způsobem je možné typicky získat pouze data z řídících jednotek, které mají souvis s emisemi. Ovládání subsystémů vozidla je také značně omezeno.

2.3 Možnosti komunikace s vestavěným systémem

Pro zajištění komunikace s vestavěným systémem z principu přichází do úvahy pouze bezdrátová rádiová technologie. Uvedu tedy souhrn zvažovaných bezdrátových komunikačních technologií a jejich krátký popis.

Bluetooth

Bluetooth[20] pracuje na frekvenci 2,4 GHz, rychlost přenosu je dostatečná (až 255 Mbit/s). Přibližný maximální dosah je pouze pár desítek metrů a jeho použití vyžaduje druhé zařízení napojeného na počítačovou síť (tzv. bridge). Využití by tedy bylo možné například v místě pravidelného parkování auta. Spotřeba energie je přijatelná (okolo 1W)

Wifi

Wifi pracuje na frekvencích 2,4 GHz a 5 GHz. Rychlost přenosu je dostatečná. Oproti Bluetooth není typicky potřeba budovat AP, respektive lze využít běžné bezdrátové přístupové body. Spotřeba energie může být nárazově vyšší, avšak celková spotřeba energie je díky rychlé komunikaci porovnatelná jako při Bluetooth.

Běžné mobilní sítě (GSM, LTE)

Využití mobilních sítí je vhodné pro sledování vozidla za jízdy, případně při parkování mimo místa pravidelného parkování. Rychlost je v případě GSM poměrně nízká, nicméně pro malé datové přenosy je postačující. V případě LTE je pak rychlost porovnatelná s předchozími možnostmi. Mobilními sítěmi je pokryto téměř celé území Česka a Slovenska.

Díky masové rozšířenosti a nízkým přeneseným objemům je možné datové tarify získat poměrně levně. Moduly pro tyto sítě jsou také velmi dostupné, pro GSM cca 100 Kč, pro LTE potom přibližně 1500 Kč.

Narrow band LTE a LTE-M

Narrow band LTE by bylo pokrytím i spotřebou energie dostačující, problémem je požadovaná frekvence zasílání zpráv. Typicky se používá například na odpočet vodoměrů, kde jsou zasílány jednotky zpráv za měsíc. LTE-M by bylo možné použít, ale není jednoduché získat tarif pro tuto síť. U nás je používaná zejména v průmyslu, proto jsou i tarify vytvořené pouze pro firemní zákazníky (řádově stovky zařízení). Dalším problémem může být pokrytí těmito sítěmi.

LoRaWAN

Intervaly zasílání zpráv by měl být minimálně 5 minut a obsah zpráv pár desítek bytů[1]. Pro obousměrnou komunikaci s modulem ve vozidle tedy není vhodná.

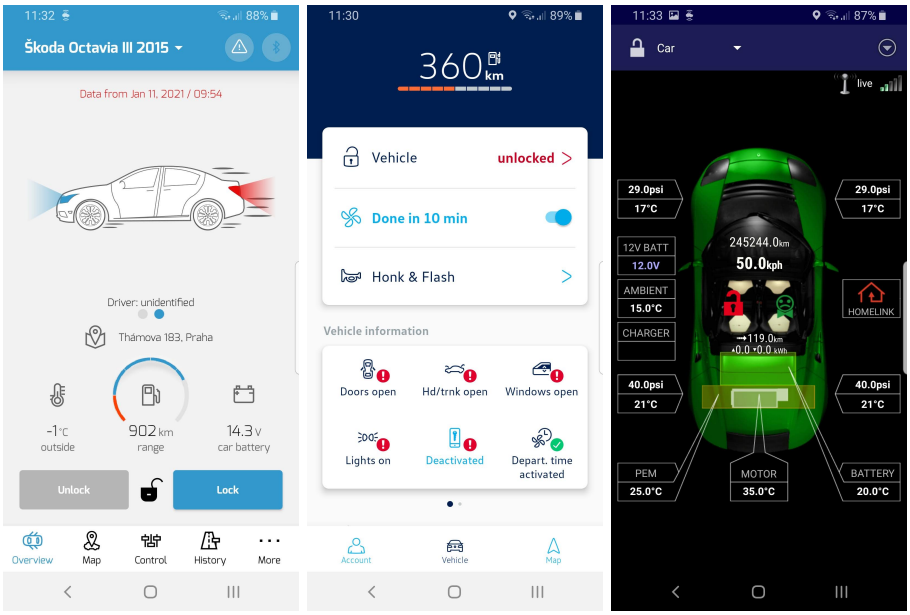
2.4 Existující řešení pro vzdálené monitorování subsystémů vozidla

Při hledání již existujících podobných řešení byly nalezeny tři přístupy k monitorování. Prví využívá diagnostický port v automobile a získává údaje dostupné skrze OBD II pro-

tokol. Tím je zaručena kompatibilita s velkým množstvím automobilů, ale jsou omezeny údaje, které je možné získat. Další možností jsou potom řešení, která jsou přímo napojena na některou ze sběrnic v automobilu. Mají mnohem větší možnosti interakce s automobilem, předně také ovládání některých jeho subsystémů. Nevýhodou je nutnost úpravy pro konkrétní automobil a komplikovanější instalace zařízení. Třetí samostatnou skupinu tvoří řešení přímo od výrobce automobilů. Ta jsou připojována dodatečně do předem připravených portů, nebo jsou zabudována do automobilů už z výroby a mají nejlepší možnost interakce.

Dojem z obslužných aplikací je velmi podobný. Základem monitorování je jedna stránka mobilní aplikace, která zobrazuje aktuální informace o vozidle, jako je například uzamknutí dveří, oken nebo mapa s polohou automobilu. V případě možností ovládání jsou zde potom umístěna také tlačítka obsluhující dané funkce. Složitější nastavení, nebo přehledy jsou typicky umístěny v dalších nabídkách, které jsou dostupné skrze postranní menu.

Následuje popis zkoumaných řešení:



Obrázek 2.5: Uživatelská rozhraní existujících aplikací [Xmarton, VW Car–Net a OVMS]

T-mobile Chytré auto

Jedná se o službu monitorování vozidla spojenou s analýzou získaných dat, asistenčními službami a distribucí internetu v rámci automobilu. Do automobilu je svépomocí instalován modul, který se zapojí do diagnostického portu automobilu. Osvojení modulu a prohlížení získaných informací probíhá skrze aplikaci v telefonu, nebo webové rozhraní.

Služba umožňuje sledovat polohu auta, číst chyby dostupné skrze OBD II a analýzu jízd s automobilem (otáčky motoru, styl jízdy, najeté kilometry). Ovládání automobilu není možné. Benefitem palubní bezdrátový AP s 5 nebo 20 GB dat. Asistenční služby nejsou předmětem porovnávání.

Měsíční poplatek za provoz je 249 Kč pro verzi s 5 GB dat, nebo 499 Kč pro verzi s 20 GB dat.

VW Car–Net / We Connect

Řešení monitorování a ovládání subsystémů vozidla od výrobce VW. U novějších vozidel je možné, že potřebný hardware je již v autě dostupný a stačí pouze zaplatit a provést aktivaci. U starších vozidel, je nutné zakoupení modulu DataPlug, který se zapojuje do diagnostického portu OBD. Služba umožňuje zobrazení polohy vozidla, prohlížení jízd a statistik, odemykání a zamykání vozidla, pomoc při hledání zaparkovaného automobilu a vzdálené vyhřátí nebo vychlazení kabiny (pokud je auto vybaveno jednotkou nezávislého topení). Výhodou je také spolupráce s palubní navigací, jenž umožňuje ukládání cílů pro navigování.

Cena za provoz je závislá od používaného balíčku a typu automobilu.

Xmarton

Tento původem český výrobce nabízí několik úrovní monitorování.[24] Z technického hlediska je nejzajímavější, nejdražší produkt `Xmarton Exclusive`. K běžným statistikám o jízdách, lokaci automobilu, nebo online přístupu, jako v předchozích možnostech přidává také větší možnosti ovládání automobilu. Skrze aplikaci je možné auto odemknout, zamknout, otevírat, nebo zavírat okna, rozsvěcovat světla případně i nastartovat. Tím je možné auto na dálku vytopit nebo vychladit. Vozidlo je také možné imobilizovat. Komunikace probíhá skrze CAN sběrnici a zařízení není zapojené do diagnostického portu vozidla. Jedná se o hotový prodávaný produkt, není tedy možné dohledat další implementační detaily.

Cena za jednotku je 12 700 Kč a měsíční poplatek 250 Kč.

Open vehicle monitoring system

OVMS je open–source [12] řešení pro monitorování, ovládání a diagnózu systémů automobilu. Po hardwarové stránce se jedná o zařízení s mikrokontrolérem ESP32, třemi CAN rozhraními a GPRS modemem kombinovaným s GPS přijímačem. Zařízení disponuje také evropskou CE a americkou FCC certifikací.

Software umožňuje sledování vlastností o vozidle získaných z CAN (případně OBDII) rozhraní, lokalizace vozidla a ovládání některých subsystémů. V základě má podporu pro 16 vozidel, přičemž všechny z nich jsou elektromobily. Podporu pro další vozidla se spalovacím motorem by bylo možné do-implementovat (jedná se o opensource).

Pro zmiňovaných 16 automobilů jsou dostupny podrobné návody pro připojení ke sběrnicím automobilu. Při ostatních vozidlech je nutné požadované údaje definovat pomocí formátu DBC (zmíněn v odstavci 2.1). Některé funkce (například vzdálené vyhřívání kabiny) potom v softwaru definovat pomocí DBC možné není.

Zařízení ve své aktuální verzi (OVMS3) stojí bezmála 3500Kč. V ceně zařízení jsou zahrnuty i poplatky za používání serveru. Datový tarif k zařízení je nutné zajistit zvlášť.

2.5 Vhodné hardwarové platformy

Při výběru byly zvažovány dva přístupy. Použití mikrokontroléru s podporou FreeRTOS, nebo jednodeskového počítače s podporou běhu OS linux. Jako zástupce prvního přístupu je uvažován mikrokontrolér ESP32, zástupce druhého potom Raspberry pi (v4).

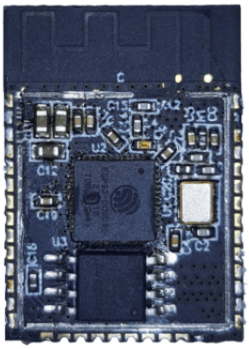
Řešení s použitím mikrokontroléru poskytuje lepší možnosti kontroly nad spotřebou energie. Je možné vypínat, nebo uspávat nepotřebné periferie, případně i celý mikrokontrolér. Nevýhodou tohoto řešení je omezení výkonu a s tím příslušná omezení ve způsobu

navrhování softwaru. Na mcu bude typicky nutné použít jazyk C / C++ a úsporně využívat dostupné prostředky (paměť a cpu).

Při použití jednodeskového počítače a běhu linuxu by bylo možné data zpracovávat i jednodušším způsobem, například v python skriptu. Výhodou by byla také možnost použití modemu pro mobilní sítě s rozhraním USB, které jsou rozšířené a cenově velmi dostupné. Díky výkonu a propustnosti zařízení by také bylo možné ze zařízení udělat palubní bezdrátový přístupový bod. Na duhou stranu prostředky pro šetření spotřebované energie jsou v tomto případě velmi omezené. Rasberry neumožňuje režim spánku, bylo by tedy nutné jej vypínat a zapínat s použitím externího obvodu. Ten by byl zřejmě obsluhován mikrokontrolérem, jednalo by se tedy o jakýsi kompromis mezi dvěma způsoby. Nevýhodou tohoto přístupu by byla větší spotřeba energie a delší reakční čas při startování systému.

ESP32

Výrobce Espressif vydal tento SoC (System on a Chip) v roce 2016 v reakci na úspěšnost jeho předchůdce ESP8266. Zatímco ESP8266 byl původně primárně určen pouze jako WIFI modul komunikující skrze AT příkazy, ESP32 byl určen od začátku pro použití jako hlavní mcu. S tím je také spojeno také vydání programovacího frameworku esp-idf spolu s mikrokontrolérem. Dnes již existuje několik revizí tohoto SoC, poslední `S2` vydaná v roce 2019. Ta má ve srovnání s první verzí několik vylepšení (novější procesor LX7, podpora flash pamětí, USB host, ...), ale chybí zde například bluetooth. Pro tento projekt je vhodnější použít původní verzi ESP32, která je stále vyráběna.



Obrázek 2.6: ESP32S bez ochranného stínění (poškozený)

Parametry a periferie: ESP32 obsahuje dvoujádrový procesor Tensilica Xtensa LX6 pracující na frekvenci až 240 Mhz a nízko odběrový ULP koprocesor, který může pracovat i při některém z režimu uspání hlavního procesoru. Dále potom 520 kB paměti RAM a 448 kB ROM. Velikosti externí flash paměti se může lišit (typicky je 4 MB), nicméně maximální podporovaná velikost je 16 MB. Následuje stručné shrnutí obsažených periferií:

I2C 2 rozhraní schopné pracovat v režimu `Master` i `Slave`.

SPI 4 rozhraní, jedno bývá využito pro již osazenou externí flash paměť. Pro uživatele jsou tedy dostupné 3 rozhraní.

UART 3 rozhraní, maximální rychlost je 5 Mb/s.

I2S 2 rozhraní, vstup i výstup.

PWM všechny výstupní piny umožňují pulzně šířkovou modulaci

Dotykový senzor 10 gpio pinů umožňující snímání změny kapacity, čehož je možné použít například při kapacitních tlačítkách

CAN umožňuje komunikaci skrze CAN rozhraní po připojení externího transcieveru splňujícího `ISO11898-1`. Implementace není kompatibilní s rozšířeným rámcem zpráv (dle `ISO11898-1 FD`). Espressif nazývá toto rozhraní TWAI (Two-Wire Automotive Interface)

A/D převodník obsahuje dva 12-bit AD registry (SAR), dohromady umožňuje až 18 kanálů měření na daných pinech. Některé kanály jsou využívány interně při bezdrátové komunikaci a není možné je volně použít

Režimy spánku: ESP32 je možné přepnout do jednoho z pěti pracovních režimů[7] :

Normal mode v tomto režimu mohou být zapnuty všechny periferie. Spotřeba energie je závislá na zapnutých periferiích, pohybuje se okolo hodnoty 200 mA. Při komunikaci skrze bluetooth a Wifi může spotřeba skokově vzrůst.

Modem sleep Při režimu uspání modemu je vypnuta periferie bluetooth a wifi, může být také snížena frekvence procesoru. Využívá se také střídavého přepínání mezi režimy modem sleep a normal mode v pravidelných intervalech, aby bylo stále možné komunikovat skrze bezdrátovou síť. Spotřeba v tomto režimu závisí na frekvenci procesoru, typicky 3 mA až 20 mA.

Light sleep podobný režim jako modem sleep. K vypínání modemu přidává použití techniky clock gating (vypínání CLK signálu). Procesor a ostatní periferie jsou tedy pozastaveny, pracuje pouze ULP koprocesor a RTC periferie. Spotřeba v tomto režimu je cca 0,8 mA.

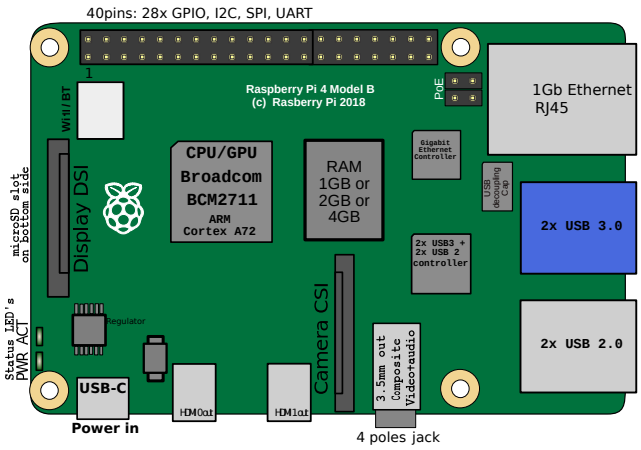
Deep sleep je nejpřísnější z režimů spánku. V tomto režimu jsou všechny komponenty odpojeny, zůstává běžet pouze RTC modul a ULP koprocesor. Spotřeba energie je cca 10 µA. Existuje zde také možnost vypnutí ULP koprocesoru, v takovém případě je spotřeba snížena na až 2,5 µA.

Probuzení ESP32 z režimu spánku je možné pomocí časovače (RTC modul), ULP koprocesoru nebo externího přerušení, které může být z dotykového senzoru, nebo změna na vstupním pinu (logická 0, logická 1 nebo změna hodnoty).

Raspberry pi

Raspberry Pi Foundation se zaobírá již od počátku vývojem a prodejem jednodeskových počítačů. První z nich byl Raspberry Pi Model B[23] vydaný v roce 2012. Počítač je pravidelně vydáván v aktualizované verzi (aktuálně verze 4). Existují modelové řady A, B, Zero a Compute modules. Modely se liší velikostmi, ale i osazenými komponentami. Rozdíl

mezi Compute modulem a ostatními verzemi je v připojitelnosti. Compute modul je primárně určen na využití do produkčních systémů a nemá tedy vyvedeny běžné konektory jako ostatní modely. Místo toho je zde pouze jeden 200-pin SO-DIMM konektor, ve kterém jsou vyvedeny všechny vstupy a výstupy. Je tedy primárně určen pro osazení do vlastního plošného spoje. Pro vývoj je možné také využít desku Comute Module IO Board.



Obrázek 2.7: Modul Raspberry pi v4 (převzato, upraveno [23])

RPi 4 Model B je aktuálně posledním vydaným počítačem z roku 2019. Obsahuje 64 bitový 4 jádrový ARM Cortex-A72 pracující na frekvenci 1.5 GHz. Modul může být osazen 1 až 8 GB operační paměti. Ze síťových rozhraní je zde ethernet (1GB/S), wifi (2.4/5 GHz) a bluetooth (5.0). Obsahuje také rozšiřitelné porty pro připojení kamery a displeje. Pro připojení dalších modulů je možné využít 40 pinový konektor s GPIO rozhraním.

Připojení periférií: GPIO konektor obsahuje 17 GPIO pinů. Ty mají hw podporu pro jedno UART rozhraní, 4 SPI a 4 I2C rozhraní. Hardwarové PWM je zde na dvou pinech, ostatní mají podporu softwarového PWM. Pro připojení periferií je možné využít i USB rozhraní (2x USB2.0 a 2x USB3.0).

Spotřeba energie: V klidovém režimu má Raspberry spotřebu cca 600 mA na 5 V. V případě vytížení může spotřeba vzrůst až na 1.25 A. Raspberry neposkytuje podporu pro režim spánku. Je zde možné vypínat nevyužívané periferie (ethernet, wifi, bluetooth,...), nicméně celý modul uspat není možné.

Souborový systém: Raspberry používá pro načtení systému paměťovou kartu. Je možné tedy osadit MicroSD dle požadované velikosti, nicméně je třeba klást nároky na spolehlivost a rychlost paměťového média.

Kapitola 3

Návrh řešení

Pro své odlišnosti byla práce rozdělena na několik částí, kterých návrhy řešení budou představeny v následujících odstavcích.

3.1 Zajištění přístupu ke sběrnici a dekódování zasílaných zpráv

Prostředkem pro komunikaci se systémy vozidla byla zvolena interní sběrnice CAN. Na rozdíl od použití OBD bude sice řešení nekompatibilní s jinými vozidly, ale bude možné získat zajímavější data o vozidle a možná také ovládat některé subsystémy.

Pro získávání dat ze sběrnice je nutné se ke sběrnici fyzicky připojit a případně ji přivést na jiné, dostupnější místo. Primárně byla pro komunikaci s vozidlem vybrána sběrnice komfortních systémů. Z pohledu získávání relevantních dat o automobilu se zdají být jednotky komunikující na této sběrnici nejzajímavější. Na tuto sběrnici je možné se napojit prakticky kdekoliv, protože propojuje jednotky napříč celým automobilem. Jako nejrozumnější místo k připojení bylo zvoleno místo pod volantem, kde se díky umístění modulu brány nacházejí všechny sběrnice. V případě zájmu tak bude možné čerpání dat rozšířit i o další sběrnice vozidla. Spolu se zmiňovanou sběrnicí bude vhodné vyvést také trvalé napájení z baterie automobilu, které bude využíváno při pozdějším osazení jednotky a trvalém provozování zařízení.

Pro proces dekódování zpráv je zapotřebí vybavení počítače CAN rozhraním. Jedna z možností je využití USB -> CAN adaptéru. Tyto adaptéry jsou většinou dodávány i se softwarem umožňujícím monitorování sběrnice. Jejich ceny se odvíjí od rychlosti, počtu CAN rozhraní a kvality softwaru. Pohybují se v řádech jednotek až desítek tisícikorun. Tyto adaptéry mají nesporně (vyjma své ceny) spousty výhod a jsou většinou dodávané také s kvalitní obslužnou aplikací. Cenově dostupnějším řešením by mohla být výroba vlastního USB -> CAN adaptéru.

V počítači bude po připojení možné v obslužném softwaru monitorovat dění na sběrnici. Pro monitorování je možné využít například soubor linuxových `can-utils`[3]. Ty umožňují zobrazování všech přijatých zpráv(`candump`), zasílání (`cansend`) nebo generování náhodné komunikace (`cangen`). Dalším je například `cansniffer`, který ulehčuje hledání změn ve zprávách jejich agregováním. Systematickým pozorováním zpráv na sběrnici a porovnáváním se stavem vozidla potom bude možné postupné zjištění obsahu zasílaných zpráv. Pro porovnávání plánuji použít také údaje získané skrze diagnostické rozhraní vozidla diagnostikou VCDS. Tímto způsobem je možné se dostat i k údajům, které jsou pro běžného uživatele

vozidla nedostupné (například teploty v interiéru vozidla). Popis obsahu zpráv bude použit v dalších částech návrhu.

3.2 Návrh zařízení umístěného uvnitř vozidla

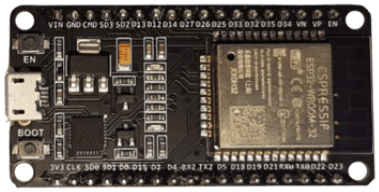
Součástí řešení je návrh a výroba zařízení zprostředkovávajícího komunikaci se subsystémy vozidla. Zařízení bude napojeno na sběrnici automobilu a bude mít k dispozici trvalé napájení z baterie vozidla, aby mu bylo umožněno komunikovat i při jeho odstavení. Pro zjednodušení návrhu bude prototyp spojen z několika modulů. Všechny součásti je možné koupit i samostatně, po ověření funkčnosti tedy bude tedy možné při finálním produktu osadit komponenty na jeden tištěný spoj a zmenšit tak celkové rozměry zařízení.

Výběr mikrokontroléru

Pro řešení jsem se rozhodl použít mikrokontrolér ESP32 od výrobce Espressif[4]. Poskytuje dostatečný výkon (dvou-jádrový procesor Xtensa LX6 o frekvenci až 240 MHz) a má přijatelnou spotřebu (včetně podpory několika režimů spánku). Pro komunikaci se systémy automobilu je možné využít integrovaný CAN kontrolér, případně připojit jiný CAN transciever například skrze SPI rozhraní. Mikrokontrolér má také tři UART rozhraní, které bude možno využít pro komunikaci s GSM modemem, nebo případným GPS přijímačem. Pro komunikaci skrze Wifi bude použito rozhraní, které je jeho součástí.

Pro správnou funkci je třeba zajistit stabilní napájení v rozmezí 2,3 V až 3,6 V a je třeba počítat s nárazovým odběrem až 500 mA. Pro šetření energie během parkování vozidla budou využity sleep režimy, pomocí kterých je možné odběr snížit až na několik málo µA.

Na prototyp zařízení je možné využít vývojové desky osazené tímto procesorem, které mají vyvedenou většinu používaných vstupně–výstupních pinů a dodržují doporučené zapojení modulu výrobcem. Jedním z nich je například ESP32 Devkit V1 (obrázek 3.1), označovaný také jako `Nodemcu`. Nevýhodou použití tohoto modulu může být nevyvedení konektoru pro připojení externí antény. Pokud se při testování ukáže tato absence jako stěžejní, bude zvážena možnost přidání antény, případně bude modul vyměněný za jiný.



Obrázek 3.1: ESP32 Devkit V1

Komunikace se zařízením

Z dostupných možných komunikačních protokolů bylo zvoleno GSM a Wifi. GSM je ve střední Evropě i přes trend vypínání v jiných částech světa stále hojně využívané a cenově velmi dostupné. Pokud by v budoucnu došlo k vypnutí GSM sítě, je možné modem nahradit jiným, podporujícím například LTE. Změny v komunikaci s modulem jsou minimální, neboť komunikace probíhá skrze standardizované AT příkazy. Technologie Wifi byla zvolena jako doplněk ke GSM. Při parkování vozidla na místě pokrytém Wifi sítí bude možné při jejím

využití značně urychlit a zefektivnit přenos dat. Tím bude možné dosáhnout nižší celkovou spotřebu energie a omezit vybíjení baterie automobilu.

Pro komunikaci skrze Wifi bude využíváno rozhraní v ESP32, skrze mobilní síť potom GSM modulu SIM800L (obrázek 3.2) připojeného skrze UART rozhraní. Modul ESP32 obsahuje také Bluetooth, který by bylo možné použít pro přímou komunikaci mezi mobilním telefonem a automobilem (například v místech bez pokrytí běžnými bezdrátovými sítěmi).

Komunikace se zařízením bude probíhat skrze HTTP requesty, jejichž obsahem budou zprávy zapsané ve formátu JSON. Celá komunikace musí být šifrovaná skrze https.

Obrázek 3.2: GSM modul SIM800L

Získávání polohy automobilu

V kontextu monitorování dat získaných z automobilu se nabízí také monitorování jeho polohy. Pokud by vozidlo bylo vybaveno originální navigací, bylo by pravděpodobně možné získávat údaje o poloze skrze CAN. V tomto případě byla tovární navigace nahrazena domontovaným autorádiem, které informace o poloze skrze sběrnici CAN neposílá. Nabízí se tak možnost přidání modulu pro určování polohy(GPS), případně lokalizace skrze dostupné sítě GSM a Wifi. Přesnost určování polohy bez GPS modulu je závislá na množství nalezených okolních vysílačů, ale pohybuje se v řádech stovek metrů.

V případě přidání GPS modulu je možné uvažovat také nad spojením s výměnou modulu pro mobilní sítě za takový, který také podporuje lokalizaci skrze GPS. Výrobce `SIMCOM` poskytuje několik takovýchto kombinovaných modulů [15]. Liší se zejména podporou novějších mobilních sítí a cenou.

Napájení zařízení

Komponenty používané v zařízení mají různé požadavky na napájení, není proto možné je napájet jedním společným napájecím napětím. Rozhodl jsem se tedy pro dvě napájecí větve, jedna s napětím 3,3 V pro ESP32 a druhá 4,0 V pro GSM modem. Větev s napětím 3,3 V bude trvale napájená LDO regulátorem a větev 4,0 V bude napájená spínaným `stepdown konvertorem`, který je možné vypnout pro šetření energie.

Software

Pro ESP32 byl vydán Espressif IoT Development Framework (zkráceně `esp-idf`). Ten nabízí nejlepší možnost obsluhy všech dostupných periferií a modulů ESP32. Pro masovou rozšířenost platformy Arduino je možné ESP32 programovat také v Arduino frameworku. Ten je ale proti `esp-idf` značně omezen a interně i tak používá framework od Espressifu. Jeho výhoda je dostupnost mnoha hotových knihoven na práci s periferiemi. Další možností je programování skrze Micropython. Ten, jak už název napovídá, je z velké míry kompatibilní implementací Python 3. ESP32 je Micropythonem podporováno a to včetně mnoha periferií.

Pro lepší využití omezených zdrojů na mikrokontroléru jsem se rozhodl pro implementaci použít jazyk C / C++ a framework esp-idf.

U zařízení umístěného mimo dosahu počítače je vhodné umožnit vzdálenou aktualizaci softwaru. ESP32 má podporu pro OTA, nicméně k jejímu využívání je potřeba mít dostatečně velkou flash paměť s dostatkem volného místa. Důležitý je také spolehlivý a rychlý komunikační kanál pro stažení aktualizace. Bude dobré, pokud by se podařilo OTA aktualizace umožnit. Pokud by se ukázalo že přenos firmwaru skrze GSM (repsektive GPRS) je nespolehlivý a pomalý, bylo by možné update firmwaru provádět pouze přes Wifi.

Lokální ukládání a akumulace dat

Požadavek na lokální ukládání a akumulaci vybraných dat (v zadání) byl uveden z důvodu možnosti použití bezdrátových komunikačních technologií krátkého dosahu (Bluetooth, Lora, ...) a z toho plynoucí potřeby ukládání a dávkového zasílání naměřených dat. Od tohoto řešení bylo po přezkoumání možností připojení odstoupeno a bylo zvoleno použití (téměř) vždy dostupného připojení (kombinace WiFi + GSM). Z tohoto důvodu byla nutnost lokálního ukládání zúžena na ukládání událostí o překročení nastavených prahových hodnot.

U každé monitorované hodnoty senzoru je možné definovat prahovou hodnotu (treshold) a porovnávací funkci, kterou bude zařízení kontrolovat. V případě jejího překročení bude vytvořena událost (event), kterého zaslání na server bude zaručeno. V případě nedostupnosti internetového připojení bude tento event uložen do trvalé paměti zařízení a bude odeslán při nejbližším úspěšném připojení.

Komunikační protokol

Komunikace mezi cloudovým serverem a zařízením v automobilu budou ve formátu JSON. Zpráva bude rozdělena do třech částí (uk. 3.1): `carState`, `deviceState` a `cmdAnswer`.

```
{
   "carState":
   {
      "doorContacts": [0,0,0,1,0,0],
      "VIN": "VWZG123456",
      "intTemp": 12,
      "rpm": 1230,
   },
"deviceState":
   {
      "runtime": 120,
      "spiffsError": false
   },
"cmdAnswer":
   {
      "lock": true
   }
}
```

Ukážka 3.1: Návrh formátu zpráv zasílaných ze zařízení

carState bude obsahovat informace zjištěné ze sběrnice vozidla. Obsahy těchto zpráv budou moci být přímo ukládány do databáze.

deviceState bude obsahovat informace o monitorovacím zařízení. Tyto informace nebudou určeny k ukládání do databáze. Jedná se o zprávy sloužící k zajištění chodu a debugování komunikace.

cmdAnswer je volitelná část zprávy a bude obsahovat stav provedení zaslaných příkazů. (`true` nebo `false`).

```
{
   "commands":
   [
      {
         "cmd": "setWakeupFreq",
         "val": 600
      }
   ]
}
```

Ukážka 3.2: Návrh formátu zpráv zasílaných do zařízení

Komunikace opačným směrem (do zařízení) bude probíhat prostřednictvím seznamu příkazů (`commands`), které budou zasílány do zařízení (uk. 3.2).

3.3 Zpracování a ukládání dat do databáze

Data získaná ze senzorů automobilu budou odesílána do cloudové aplikace, kde budou zpracována a ukládána do databáze. Pro implementaci jsem se rozhodl použít prostředí NodeJS. Implementace serverové části aplikace bude tak sdílet implementační jazyk s webovým rozhraním (popsaný v 3.4), takže bude možné případné znovupoužití částí kódu.

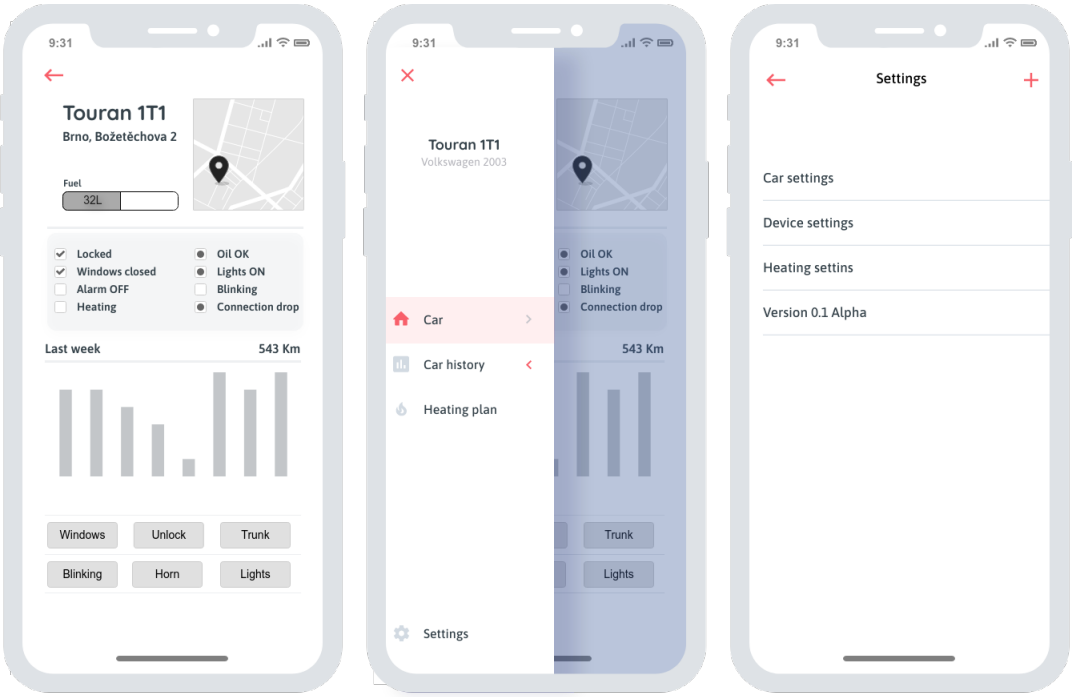
Při odhlédnutí od komplikovanějšího způsobu získávání dat se jedná o běžnou webovou aplikaci. Ukládaná data budou časové řady, nabízí se tedy použití databáze uzpůsobené na ukládání takovýchto dat. Mezi zvažovanými databázemi byly InfluxDB, MongoDB a TimescaleDB. Všechny tři nabízí podporu ukládání časových řad. InfluxDB a TimescaleDB mají lepší kompresi a nižší nároky na hardware. Na druhou stranu MongoDB je rozšířenější, lepší podporu knihoven a umožní také jednoduché ukládání i ostatních dat, nejen časových řad. Při této velikosti projektu (ukládání dat z jednoho automobilu) je databáze MongoDB dostačující. Při sběru dat z mnoha automobilů by bylo zřejmě lepší použití například InfluxDB pro časové řady v kombinaci s jnou databází pro ostatní data.

Serverovou část implementace je možné rozdělit na dvě části:

- Komunikace se zařízením
- Poskytování dat webové aplikaci

3.4 Monitorování a přístup k zaznamenaným datům

Přistupování k uloženým datům a ovládání subsystémů bude probíhat skrze webovou singlepage aplikaci. Ta bude se serverem komunikovat skrze definované aplikační rozhraní. Primárně by měla být aplikace dostupná skrze mobilní telefon, čemu bude také přizpůsobeno její rozhraní.



Obrázek 3.3: Mockup aplikace

Webová aplikace (PWA) bude koncipovaná podobně jako klasické aplikace na androidu (obrázek 3.3). Bude se skládat z několika pohledů, mezi kterými bude možné přepínat pomocí postranní vytahovací nabídky. Na první stránce budou dostupné často používané a základní informace o vozidle, to je například jeho poloha, statistiky z posledních jízd nebo uzamčení vozidla. Další stránky aplikace budou zobrazovat historii vozidla (poslední jízdy a parkování), stav zařízení umístěného ve vozidle a upravování jeho parametrů (např. různé frekvence zasílání získaných informací).

Kapitola 4

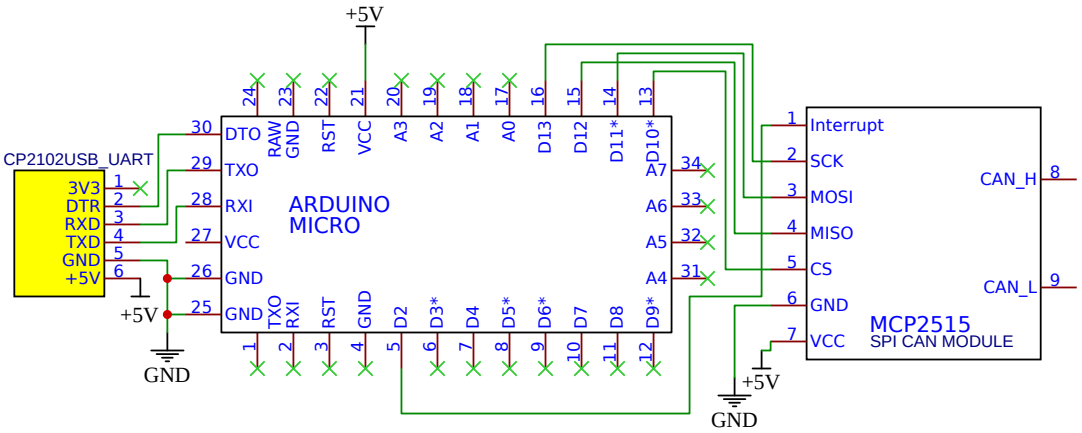
Dekódování zpráv a implementace

4.1 Dekódování zpráv

V následujících odstavcích je popsán postup dekódování zpráv zasílaných na interních sběrnicích automobilu.

Zajištění připojení ke sběrnici

Pro připojení počítače ke sběrnici jsem se rozhodl použít `USB -> CAN` převodník na bázi Arduino Micro. Ten se skládá z `USB -> UART` převodníku, Arduino Micro (`ATmega32U4`) a `MCP2515` CAN kontroléru s SPI rozhraním. Jejich zapojení je znázorněno na obrázku 4.1. Cena komponent tohoto vyrobeného převodníku je asi 200 Kč. Do mikrokontroléru byl nahrán opensource firmware `arduino-canbus-monitor`[1]. Ten umožňuje komunikaci s CAN rozhraním skrze `UART` protokolem nazývaným `SLCAN`, nebo také `Serial CAN`. Maximální podporovaná rychlost sběrnice, se kterou tento převodník zvládne komunikovat, je 500 Kbit/s. Ve zkoumaném automobilu se nenachází sběrnice s větší rychlostí.

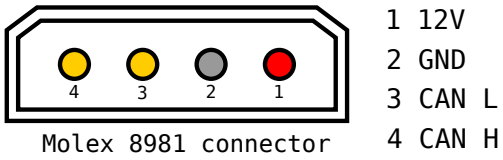


Obrázek 4.1: Schéma `USB -> CAN` převodníku

Fyzické připojení na sběrnici komfortních systémů bylo provedeno v místě připojení komfortní jednotky (v blízkosti centrální gateway), skrze již existující připojení domontovávaného alarmu. Dále je zde připojeno napájení 12 V a 0 V. Všechny 4 vodiče jsou

[1] https://github.com/latonita/arduino-canbus-monitor

vyvedeny do konektoru (obrázek 4.2) umístěného v blízkosti pojistkové skříně automobilu, kde je také dostatek místa na finální umístění jednotky. K propojení byl použit konektor Molex 8981, používaný například pro napájení počítačových komponent. Při výběru bylo dbáno zejména na zabránění nechtěnému propojení a tím způsobenému zkratu při odpojení konektoru a při jeho připojování. Přívod 12V byl osazen 5A pojistkou.



Obrázek 4.2: Popis konektoru určeného pro připojení k automobilu

Proces dekódování zpráv

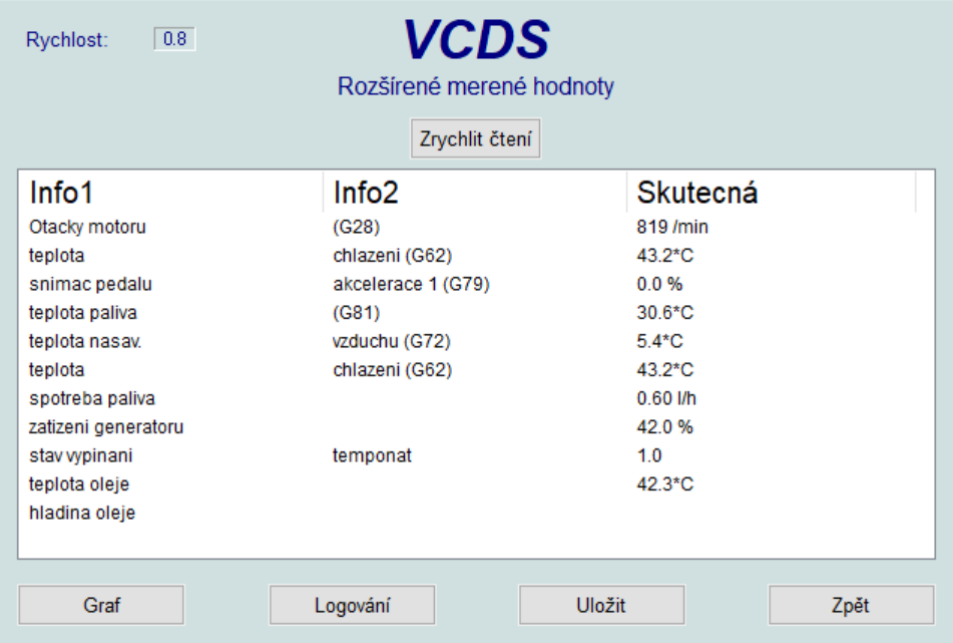
Po zajištění připojení počítače ke sběrnici automobilu skrze převodník bylo možné zahájit postupné dekódování zpráv zasílaných na sběrnici. Program `cansniffer` z linuxového balíčku `can-utils` umožňuje filtrování a agregování zpráv dle identifikátoru odesílatele a zobrazování zpráv v binárním formátu. To se spolu s možností zvýraznění změněných bitů ukázalo, jako nejlepší cesta k dekódování zpráv. Postup dekódování konkrétního parametru je možné ukázat například na stavu otevření dveří.

Příklad – stav otevření dveří Hledání parametru ve zprávách je lepší provádět se zapnutým zapalováním. Při vypnutém zapalováni dochází k postupnému uspávání jednotek a při jejich hromadném probuzení se hledaná zpráva v komunikaci typicky lehce přehlédne. Program cansniffer se po chvíli ustálí a zobrazí přehled zpráv. Ty obsahují ID, frekvenci zasílání a obsahy zpráv se zvýrazněnými měnícími se částmi. Ze zobrazovaného seznamu je ještě možné smazat již známé zprávy z jiných řídících jednotek pomocí filtru. Dále po otevření dveří bude vidět změna ve zprávách, které sledovaný údaj obsahují, nebo na něj reagují. Takovýchto zpráv může být i více a proto je většinou nutné proces několikrát opakovat. Například u zkoumaných dveří řidiče je možné změnu zjistit u dvou zpráv. První je z jednotky ve dveřích řidiče, protože obsahuje také stavy jiných ovládacích prvků osazených ve dveřích (ovládání oken, náklonu zrcátka, atd.). Pro další zpracování byla v tomto případě použita zprávu z druhé jednotky, protože obsahuje souhrnné data o všech dveřních kontaktech v automobilu. (Tabulka 4.1).

ID	**Délka [bytů]**	**Zpráva [bin]**	**Reprezentace**
470	5	0000 0001 0000 0000 0000	LP Dveře otevřené
470	5	0000 0010 0000 0000 0000	PP Dveře otevřené
470	5	0000 0100 0000 0000 0000	LZ Dveře otevřené
470	5	0000 1000 0000 0000 0000	PZ Dveře otevřené
470	5	0001 0000 0000 0000 0000	Kufr otevřen
470	5	0010 0000 0000 0000 0000	Přední kapota otevřena

Tabulka 4.1: Dekódovaná část obsahu zpráv

Údaje které nejsou ve vozidle přímo zjistitelné byly porovnávané s údaji získanými skrze diagnostiku VCDS (obrázek 4.3). Zjištění přesné číselné hodnoty usnadní hledání, které už je samo osobě ztíženo matematickými úpravami při zakódovávání zpráv.



Obrázek 4.3: Ukázka hodnot zjištěných skrze diagnostiku VCDS

Příklad – čas zobrazený na palubním počítači je zasílaný spolu s dalšími údaji v rámci zpráv s dentifikátorem 0x65D. Pro zjištěni zápisu byly pozorovány krátké změny (v rámci minut a sekund) a poté simulovány delší časové skoky (manuální změnou času na palubním počítači). Celý postup dekódování aktuálního času je možné vidět na obrázku 4.4.

```
65D#3599460300B0DE06 ──────────→ 11:55:17

65D#xxxxxxxxxxAxBCDE ──────────→ HH:MM:SS

HH = A + ((D & 0x1)*16)
MM = (BC & 0x7D) >> 1
SS = DE *2 + (BC & 0x40)
```

Obrázek 4.4: Výpočet času posílaného jednotkou palubního počítače

Údaje dekódované ze zpráv

Ze sběrnice se mi prozatím podařilo dekódovat následující údaje o vozidle:

- Poloha klíčku v zapalování

- Stav otevření a zamčení dveří
- Stav otevření oken
- Otáčky motoru
- Aktuální rychlost vozidla
- Zařazený převodový stupeň
- Stav stěračů
- Stav světel a varovných světel
- Venkovní teplota
- Vnitřní teplota
- Teplota chladící kapaliny
- Teplota oleje
- VIN vozidla
- Napětí baterie
- Aktuální čas
- Stav najezděných kilometrů
- Stav varovných indikátorů (nízká hladina oleje, prázdná palivová nádrž, ...)

Možnosti ovládání subsystémů automobilu

Při monitorování je možné, aby byl CAN modul přepnut do `listen_only` režimu a nijak tedy neovlivňoval dění na sběrnici. Ovládání jakýchkoliv částí systému vyžaduje zasílání zpráv na sběrnici, je tedy nutné modul přepnout do režimu umožnující zasílání. Tím vzniká větší riziko způsobení chyby na sběrnici, nebo na některé z připojených jednotek.

Problémem je také, že některé (z pohledu automobilu) kritické zprávy jsou ošetřeny kontrolními součty, nebo je v nich obsaženo počítadlo, které se při každé zprávě inkrementuje. Tohoto počítadla se paradoxně využívá i při snaze o odklonění původní jednotky a jejím kompletním nahrazením[2]. Takovéto „odstavení“ systémů ale není předmětem této práce.

Při správném reprodukování nahraných zpráv by mělo být možné některé subsystémy vozidla také ovládat. Systematickým pozorováním dění na sběrnici je možné nalézt části zpráv, zodpovědné za hledané změny v systému. Na příkladě z jednotky dveří řidiče (tabulka 4.2) je možné vidět dva nalezené bity, kterých změnou je možné docílit otevření zavazadlového prostoru, nebo víka nádrže. Zajímavostí je, že dveřní jednotka je pravděpodobně univerzální a obsahuje tedy i možnost otevření zámku zavazadlového prostoru, i když dveře takovýmto spínačem osazeny nejsou. Systémy pracující na vybrané sběrnici nejsou kritické z pohledu bezpečnosti a jízdních vlastností automobilu, nicméně sběrnice jsou mezi sebou propojené a chyba by mohla být zavedena i do jiných kritických systémů. Z tohoto důvodu budou všechny experimenty prováděny pouze u stojícího automobilu.

ID	DATA [bin]						Akce
381	1000000	00000110	000000010	10001100	00000000	00000000	x
381	1000000	00000110	000000010	10001100	**1**0000000	00000000	víko nádrže
381	1000000	00000110	000000010	10001100	0**1**000000	00000000	zav. prostor

Tabulka 4.2: Hledání možností ovládání skrze jednotku dveří

Při zpětném zaslání zaznamenané zprávy se opravdu vykoná požadovaná akce (v tomto případě otevření víka nádrže). Tento naivní přístup má však významný problém, kterým je možný konflikt na sběrnici. Sběrnice CANBUS používá arbitr spoléhající se na splnění požadavku na unikátnost identifikátoru každé vysílané jednotky (popsáno v 2.1). Většina jednotek zasílá svůj stav několikrát za vteřinu a tento předpoklad může být při takovémto injektování porušen. Při specifickém zaslání může nastat na sběrnici konflikt, který arbitr nebude schopen vyřešit. I při správném načasování může nastat zmatení jednotky, která může očekávat specifické konsekvence k dané změně bitu, které takovýmto umělým sepnutím nenastanou. Problémem je také periodické zasílání dvou různých zpráv (z pravé a ze smyšlené jednotky), které mohou (a typicky také jsou) protichůdné. Chování takovéto jednotky je potom nepředvídatelné[10].

Pro zaslání příkazu k zamknutí dveří je zřejmě tento způsob dostatečný a může být použit, nicméně jsou části systému, které takto jednoduše není možné ovládat. Příkladem může být nastavení cílové teploty interiéru. Tato hodnota se v komunikaci skutečně objevuje a je zasílána jednotkou ovládání klimatizace. Zmiňovaná jednotka ale bohužel hodnotu odesílá a není možné jí tímto zjištěným kanálem systému vnutit. (podobný problém je popisován v *Hacking the CAN Bus: Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering*[3]) Tento problém se ukazuje u většiny ovládaných periférií. Zaznamenané zprávy jsou buď k ovládání vhodné, ale jednotky je periodicky zasílají a je tak jejich podvrhnutí komplikované, nebo je sledovaný parametr pouze zasílán, ale není možné jej nastavovat (nastavování probíhá pouze uvnitř jednotky a ven je zasílán pouze nastavený stav). V tabulce 4.3 je možné vidět proces zasílání navzájem odporujících si zpráv na sběrnici. Okno (cílený akutátor) se v tomto případě přerušovaně otevírá a zastavuje.

Čas [ms]	Odesílatel	ID	Zasílaná data [hex]	Vliv na okno
0	ŘJ dveře řidiče	0x381	82 0C 00 8C 00 00	okno se nehýbe
50	fiktivní zařízení	0x381	82 0C 00 8D 00 00	otevírání okna
100	ŘJ dveře řidiče	0x381	82 0C 00 8C 00 00	okno se nehýbe
150	fiktivní zařízení	0x381	82 0C 00 8D 00 00	otevírání okna
200	ŘJ dveře řidiče	0x381	82 0C 00 8C 00 00	okno se nehýbe

Tabulka 4.3: Konflikt zasílaných zpráv

Další možností by mohlo být ovládání s využitím identifikátoru, který není periodicky zasílán. Jednotky mohou reagovat i na jiné zprávy, například při domontování dodatečné výbavy. Problémem je objevení takovýchto identifikátorů a jejich obsahů. Mohly by být nalezeny odposlechnutím takovéto výbavy (problematické), nebo například zasíláním náhodných zpráv s náhodnými identifikátory a sledováním změny stavu. U tohoto řešení je však riziko zaslání zprávy, jejíchž následky by mohly být nevratné (například vystřelení airbagů). Tento problém by se dal vyřešit například izolováním části sběrnice na které je prováděno hledání. V tomto případě by bylo možné zasílat náhodně generované zprávy a

pozorovat reakci na sledované části systému. Takovýto proces zjišťování může být velmi zdlouhavý a proveditelnost izolování některých částí je také diskutabilní. Sledovaná jednotka může ke své aktivaci například vyžadovat přepnutí zapalování do zapnuté polohy, což by bylo třeba správně simulovat například zasíláním nahrané komunikace.

Jinou možností je dekódování a zasílání diagnostických zpráv, které typicky obsahují také testovací sekvence ovládající různé aktuátory automobilu. Při pokusech byla odposlouchávána komunikace na pozorované sběrnici, přičemž byl spuštěn skrze diagnostiku `VCDS` proces *Test akčních členů*. Na sledované sběrnici se začaly objevovat zprávy s novým, dosud nezaznamenaným identifikátorem. Diagnostika se připojuje k automobilu skrze diagnostický port a komunikace s jednotkami probíhá skrze gateway. Bohužel prosté zaznamenání a zreprodukování zaznamenané komunikace v tomto případě nefungovalo. Pro ovládání jednotek tímto způsobem by bylo zapotřebí zjistit obsah zpráv zasílaných z diagnostiky skrze gateway.

Všeobecně jde snaha o ovládání součástí automobilu proti snaze výrobce, který se snaží sběrnici a její komunikaci co nejvíce zabezpečit a takovémuto konání zamezit. Mimo kontrolní součty (CRC), sekvenční číslování kontrolních částí zpráv může mít také ovlivňovaná jednotka nastavené podmínky, při kterých umožní požadovanou akci provést. U ovládání zámku dveří pravděpodobně takovéto chování nebude implementováno, nicméně například u natáčení nápravy (používané při samočinném parkování) může být[10].

Pro následující akce byly odhaleny příkazy, které je vyvolávají a je možné je v projektu použít:

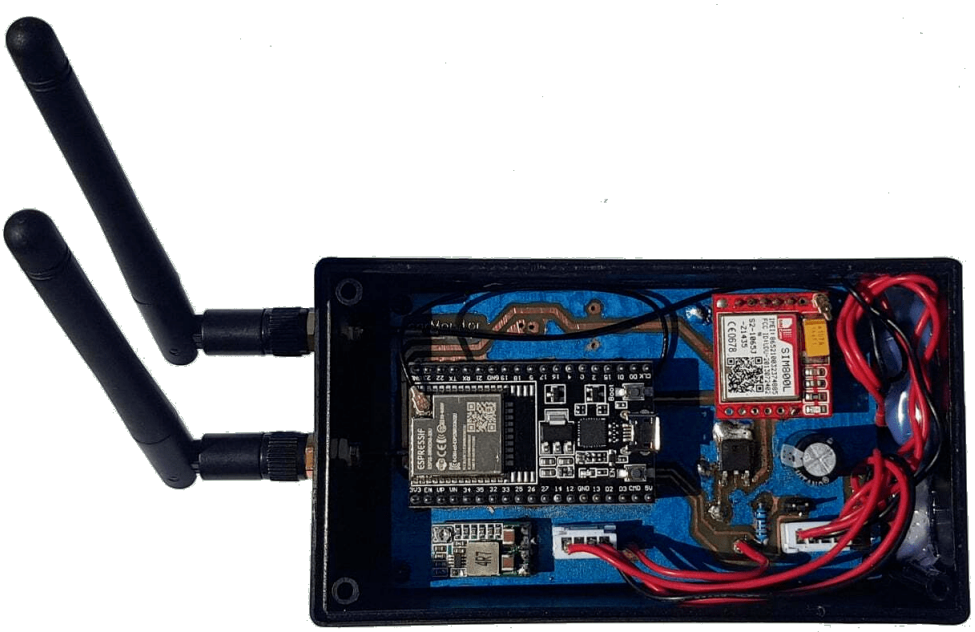
- Zamknutí automobilu (se spuštěním alarmu)
- Zamknutí automobilu (bez spuštění alarmu)
- Odemknutí automobilu (alarm zůstává spuštěn, není možné jej vypnout)
- Otevření a zavření oken (částečné i úplné)
- Otevření zavazadlového prostoru
- Otevření víka nádrže

Všechny akce lze vyvolat i pokud jsou jednotky automobilu v režimu spánku (při příjmu zprávy se probudí) a využívají pro odeslání identifikátor komfortní jednotky, která většinu času aktivně nevysílá.

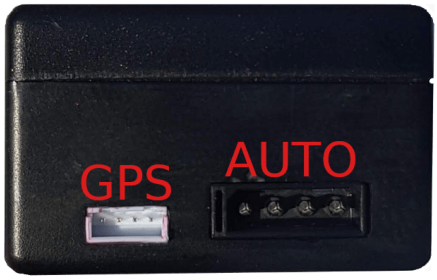
4.2 Hardware zařízení v automobilu

Tato sekce popisuje proces vývoje hardwaru zařízení určeného pro monitorování subsystémů automobilu a zasílání získaných informací na server pro zpracování.

Dle návrhu byl vytvořen prototyp zařízení pro monitorování a komunikaci s vozidlem. Mezi použité komponenty patří `NodeMCU ESP32S`, CAN transceiver `MCP2551`, GSM modul `SIM800L` a step-down konvertor. Komponenty jsou osazeny na univerzálním plošném spoji a jejich zapojení je možné vidět na obrázku v příloze 4.7. Navíc byl oproti návrhu modul osazen také externím UART GPS modulem Ublox–6M, pro rozšíření monitorování také o polohu automobilu.



Obrázek 4.5: Hotové zařízení bez vrchního krytu



Obrázek 4.6: Zařízení – pohled na stranu s konektory

Napájení zařízení

Celé zařízení musí být napájeno z baterie automobilu, přičemž klidový odběr celého systému automobilu v uspaném stavu je dle měření 0.4 A. Spotřeba implementovaného zařízení by tedy měla být při uspání co nejnižší, aby nedocházelo při delší odstávce automobilu k vybití baterie. Pro snížení spotřeby energie a dodržení předepsaných napětí všech modulů byly vytvořeny dvě napájecí větve:

3,3 V je napájecí napětí první větve. Napájí ESP a CAN transciever. Oba tyto moduly potřebují stálé napájení, aby bylo zařízení schopné probudit se z režimu spánku.

Modul MCP2515 je při ukončení komunikace na sběrnici přepnut do režimu spánku. V tomto režimu je jeho spotřeba energie snížena[9] z 5 mA na 1 µA. Nevýhoda tohoto režimu je snížená propustnost, nicméně ihned po startu ESP je modul opět přepnut do normálního módu.

ESP vyžaduje pro provoz také 3,3 V a jeho spotřeba v aktivním režimu může být až 0,3A. Při režimu spánku je snížena až na 10 µA. Z tohoto režimu je přepnuto zpět do normálního režimu v případě obnovení komunikace na sběrnici, nebo v předem nastaveném časovém intervalu.

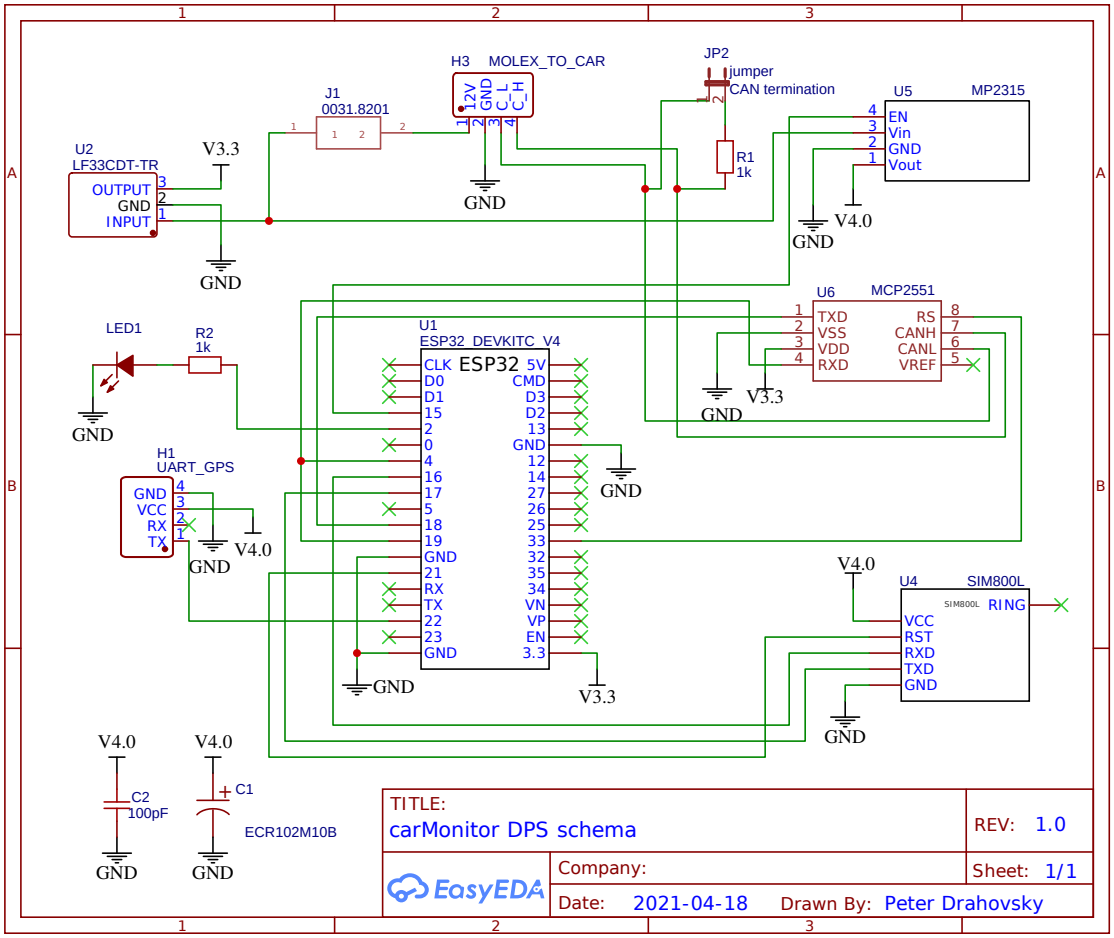
4,1 V je napájecí napětí druhé větve. Touto větví je napájen GSM modul a GPS přijímač. Napájecí větev je generována step-down konvertorem a napájecí napětí je zvoleno průnikem doporučených napětím obou modulů. Špičkový odběr GSM modulu při připojování k síti může být i 2A, proto jsou v blízkosti modulu osazeny kondenzátory. Použitý konvertor má povolovací vstup a je jej možné při režimu spánku vypnout. GPS ani GSM modul není potřebný při spánkovém režimu.

Výroba plošného spoje

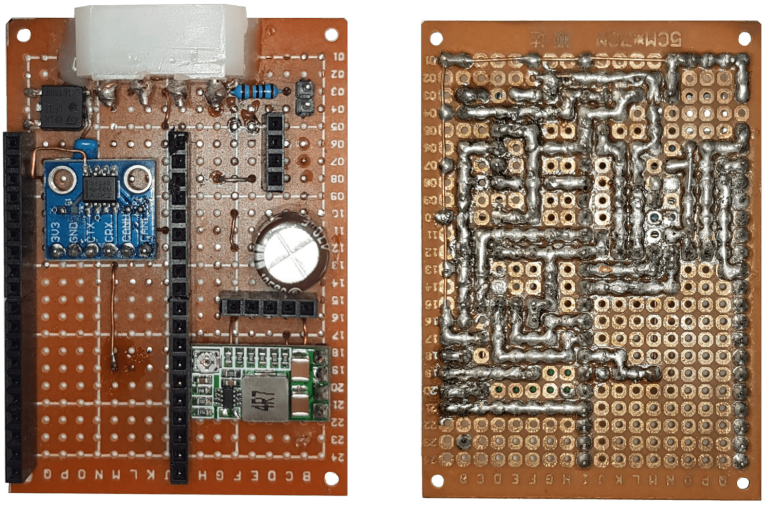
Pro účely vývoje byl vytvořen prototyp na univerzálním plošném spoji (obr. 4.8). Výhoda takovéhoto řešení je možnost relativně jednoduché změny v zapojení, nebo přidání nových komponentů. Oproti nepájivému poli je potom prototyp mnohem robustnější a odolnější proti nahodilému odpojení, nebo narušení zapojení při přenášení (což je při testování v prostředí automobilu zásadní). Zapojení bylo tedy postupně odladěno na tomto prototypu a zaneseno do schématu (obrázek 4.7).

Prototyp na univerzálním plošném spoji je plně funkční, nicméně vzhledově ani trvalostí není ideální. Rozměrově a rozložením by také bylo komplikované vytvoření vhodného obalu. Použitý modul s mcu. ESP32 neobsahuje konektor na připojení externí antény a není tedy možné použít větší anténu, která by pomohla se zvýšením síly signálu WiFi. Po zvážení zmíněných problémů, bylo rozhodnuto o vytvoření nového DPS, který bude splňovat následující požadavky:

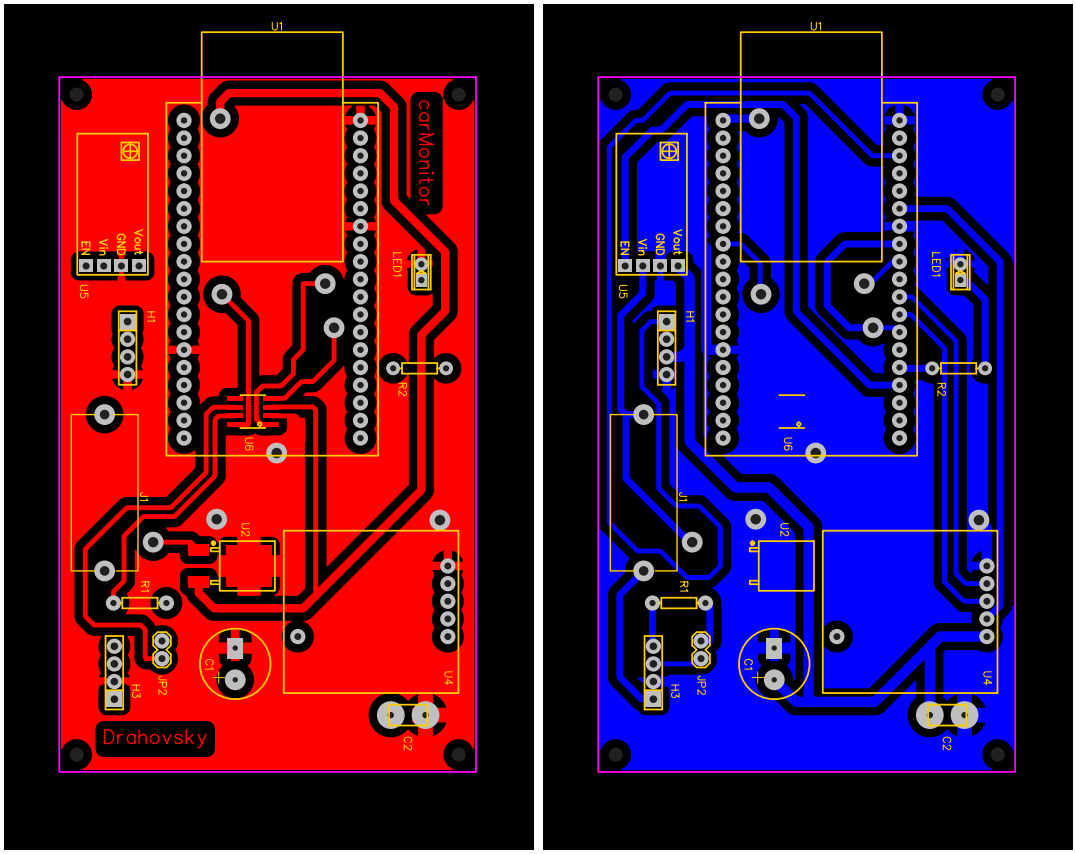
- rozměry desky vhodné pro umístění do krabice
- použití modulu ESP32 s konektorem pro připojení externí antény (jiné rozložení vstopně-výstupních pinů)
- klíčové komponenty, které by mohly být v budoucnu vyměněny (ESP32 a GSM modul) nechť jsou osazeny s pomocí rozebíratelného spoje



Obrázek 4.7: Schéma zapojení vytvořeného plošného spoje



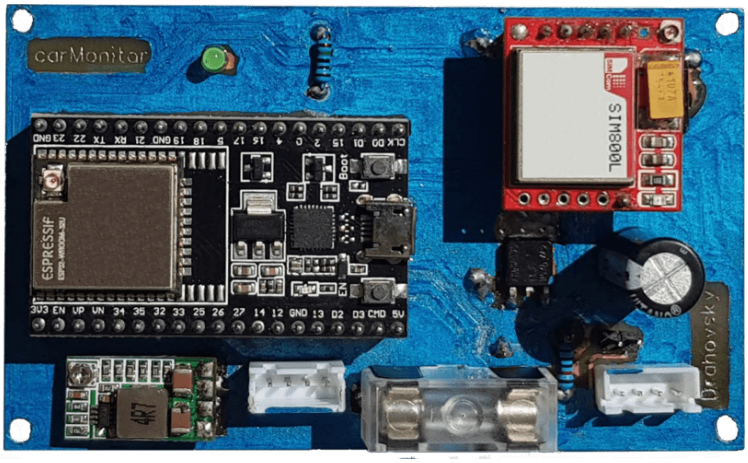
Obrázek 4.8: Prototyp zařízení vytvořený na univerzálním plošném spoji (bez zapojeného modulu ESP32 a SIM800L)



Obrázek 4.9: Návrh plošného spoje

Při návrhu desky byl kladen důraz také na zvýšení tolerance k možným chybám při výrobě desky. Toho bylo dosaženo zvětšením průměru provrtávaných děr, zvětšením pájecích ploch, snížením množství propojů mezi stranami desky a zvětšením mezer mezi jednotlivými cestami na DPS (*anglicky clearence*). Po těchto úpravách bylo možné desku vyrobit i v domácích podmínkách (návrh je možné vidět na obrázku 4.9).

Hotový návrh byl vytisknut na laserové tiskárně na nesavý papír, odkud byl zahřátím žehličkou přenesen na cuprexit [8]. Po postupném odmáčení papíru zůstal na desce pouze zapečený inkoust (maska označující místa, kde měď zůstává). Poté byl ponořen do směsi určené na leptání DPS (chlorid železitý). Po vyleptání, opláchnutí a provrtání děr byly na plošný spoj postupně osazeny všechny komponenty. Průchodky propojující vodivé dráhy na protilehlých stranách desky byly vytvořeny připájením drátové propojky. Poté byl plošný spoj nalakován pro zakrytí nechráněné mědi a zvýšení ochrany vodivých spojů (modrá barva). Zhotovený spoj osazený všemi součástkami je možné vidět na obrázku 4.10.



Obrázek 4.10: Osazený plošný spoj – povrch je ošetřen modrým lakem

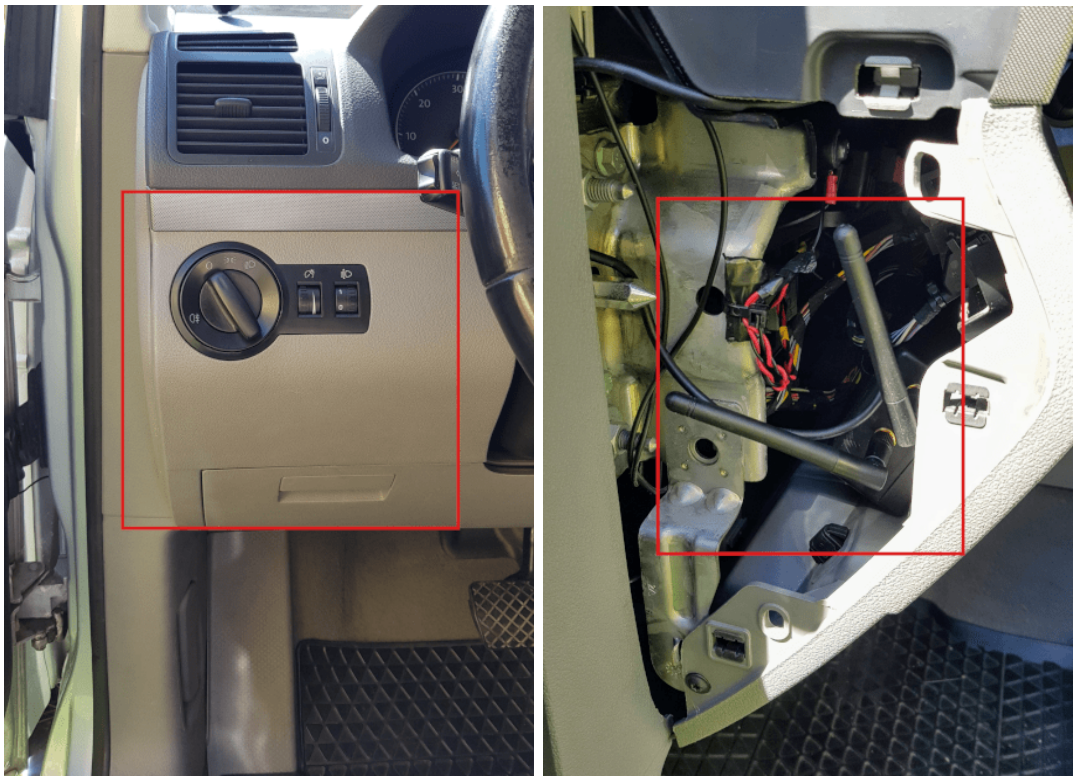
Zajímavé části implementace

Kvalita signálu bezdrátových sítí – funkčnost obou použitých komunikačních kanálů (GSM i Wifi) je závislá od kvality signálu. Při testech s prototypem na univerzálním plošném spoji (který pro komunikaci využíval pouze malé integrované antény) se ukázala potřeba přidání konektorů na osazení externích antén. Zařízení mělo většinu času signál dostatečný, nicméně při jízdách skrze místa s horším pokrytím signálem GSM spojení vypadávalo. Při parkování byla kvalita spojení WiFi velmi ovlivněna umístěním prototypu v rámci vozidla. Při přílišném zastínění obklopenými plechy karoserie a nesprávném natočení vůči používanému přístupovému bodu byl signál nestabilní. Po osazení finálního zařízení externími anténami se tyto problémy velmi zredukovaly. Pokud by se v budoucnu opět ukázalo, že je signál nedostatečný, je možné vyměnit krátké tyčové antény za antény s delším přívodním kabelem, které bude poté možné namontovat na vhodnější místo, ve větší vzdálenosti od zařízení (například pod čalounění sloupku u předních dveří).

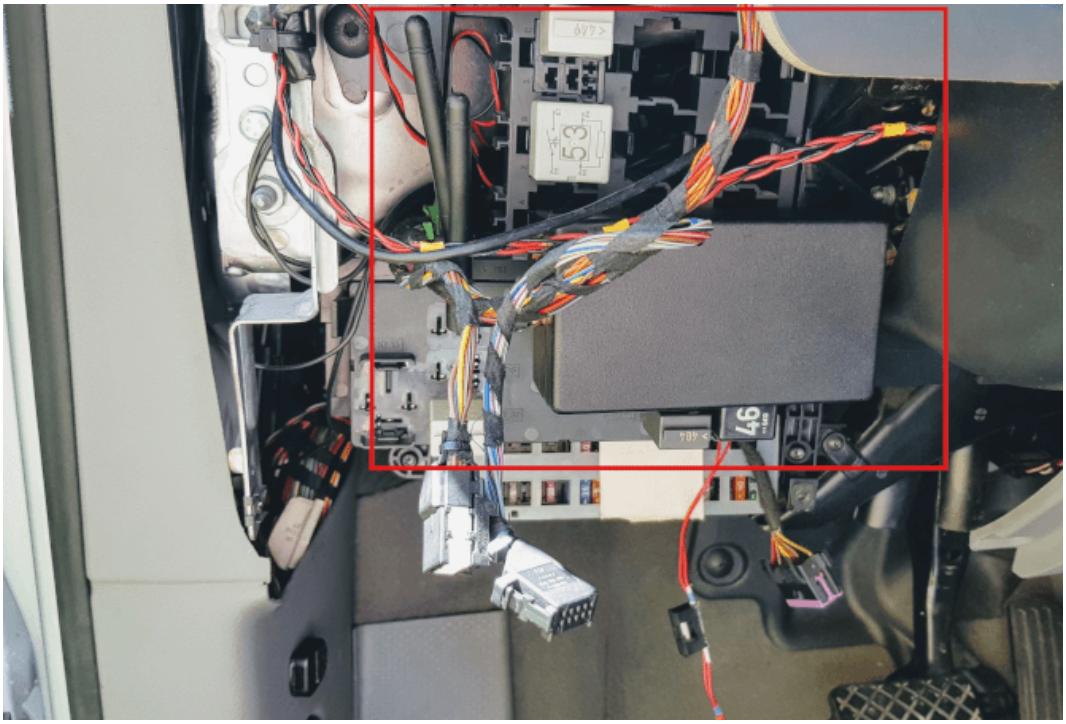
Použití propojovacích konektorů – stěžejní části zařízení (GSM modem, ESP32 a GPS modul) byly připojeny k plošnému spoji skrze konektory. Výhodou tohoto řešeni je jejich snadná výměna v případě poškození, nebo ukončení podpory používané technologie (v případě GSM modulu). Pokud by tyto části nebyly osazovány tímto způsobem, mohlo by být přistoupeno k vytvoření jednoho plošného spoje, který by v sobě zahrnoval i zmiňované moduly. Obtížnost návrhu je v takovémto případě vyšší, protože je třeba důkladněji dbát například na správné odstínění, nebo délky vodivých drah. Takovýto plošný spoj by také zřejmě nebylo možné osadit v domácích podmínkách, jelikož by obsahoval spoustu SMD komponent (také komponenty, které mají pájecí plošky ze spodní strany, které vyžadují pro své osazení speciální vybavení).

Optimalizace napájení – průměrná spotřeba zařízení se při běhu mírně liší dle typu připojení – 0.9 W při komunikaci skrze WiFi a 1.1 W při komunikaci skrze GSM. Při běhu motoru automobilu a dobíjení baterie, je tato spotřeba zanedbatelná, avšak při parkování by překonala samotnou klidovou spotřebu automobilu (0.3 W). Po uspání zařízení klesá spotřeba na méně než 0.1 W, což by nemělo mít na vybíjení autobaterie zásadní vliv.

Umístění ve vozidle – zařízení bylo do vozidla umístěno (dle návrhu) v místě před řidičem, pod krycím plastem v blízkosti pojistkové skříně (obr. 4.11 a 4.12). V tomto místě je jednoduchá dostupnost k napájení, ke všem sběrnicím a je zde dostatek místa. GPS anténa byla nalepena pod přístrojovou desku, která je plastová a nebrání tak prostupu signálu. Pro uchycení krabičky zařízení byly zvoleny stahovací pásky.



Obrázek 4.11: Uložení zařízení v automobile (zařízení vyznačeno červeným ohraničením)



Obrázek 4.12: Montáž zařízení do automobilu – pohled bez krycího plastu (zařízení vyznačeno červeným ohraničením)

4.3 Software zařízení v automobilu

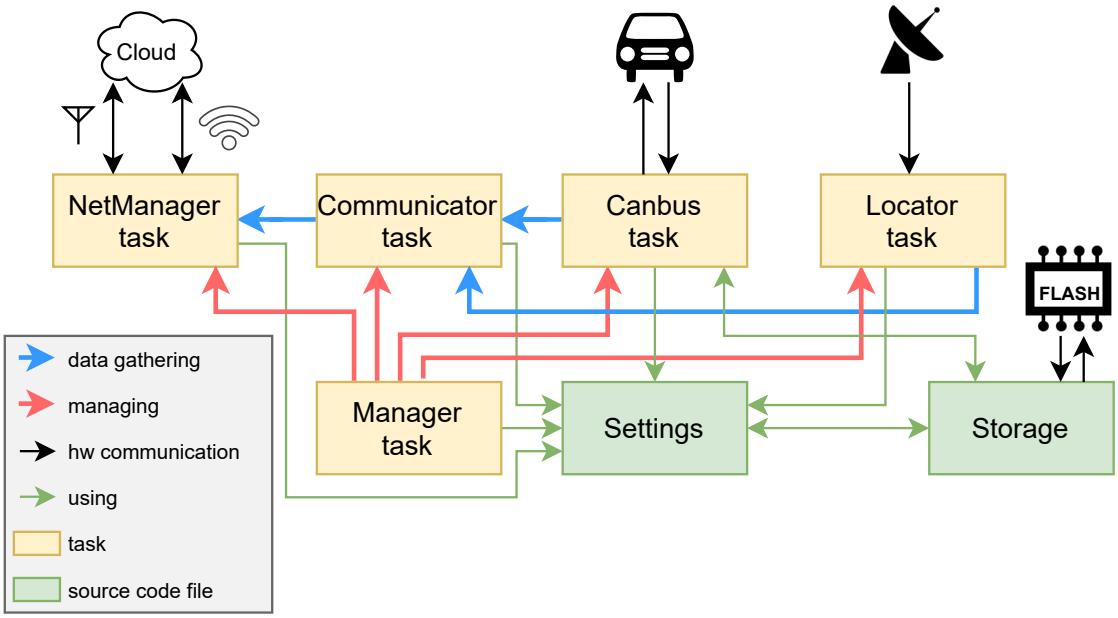
`ESP-IDF` (vývojový framework pro moduly ESP od výrobce Espressif) interně využívá FreeRTOS, který mimo jiného umožňuje využití preemptivního multitaskingu. Xtensa lx6 (mikroprocesor obsažený v ESP32) obsahuje dvě výpočetní jádra, což umožňuje opravdu souběžný běh dvou podprogramů. Softwarová implementace byla rozdělena do pěti částí dle jejich funkce. Každé části je přidělena výpočetní úloha (*anglicky task*) s příslušnou prioritou a o přidělení procesoru se již stará FreeRTOS.

Popis funkčních částí

Krátký přehled funkčních částí běžících souběžně, spuštěných po startu systému:

NetManager obsluhuje připojení k internetu. Při pokynu k připojení vyhledá bezdrátové sítě v dosahu a připojí se skrze GSM či WiFi. Rozhodnutí závisí na dostupnosti, síle a chybovosti dostupných Wifi sítí (v případě vysoké chybovosti je připojování skrze Wifi pozastaveno a je přepnuto výhradně na GSM). Po úspěšném připojení modul netManager zajistí synchronizaci času (skrze SNTP) a nastaví stav sítě na připojený. Poté vyčkává na pokyn odpojení, znovupřipojení, nebo na případný výpadek připojení. Samotný modul pouze provádí zadané akce (připojení a odpojení) a zprostředkovává stavy připojení sítě.

Canbus zajišťuje komunikaci s vozidlem skrze sběrnici CAN. Při detekování příchozích zpráv jsou porovnány identifikátory zpráv se seznamem monitorovaných parametrů.



Obrázek 4.13: Znázornění propojení funkčních částí zdrojového kódu

V případě shody je ze zprávy vypočtena hodnota sledovaného senzoru a je uložena do příslušného objektu v paměti reprezentujícího daný senzor. Při nastavování hodnoty je také kontrolováno překročení nastavené prahové hodnoty a případně je vytvořena nová událost (dle návrhu 3.2) Takováto událost je v paměti uložena do doby potvrzení odeslání na server. V případě neúspěšnosti odeslání je před uspáním (nebo restartováním) uložena do paměti, odkud je při příštím startu načtena a odeslána. Pro každý senzor je možné definovat jednu prahovou hodnotu a porovnávací operátor (větší, menší, rovno, nerovno).

V případě neaktivity sběrnice po nastavenou dobu modul nastavuje příslušnou událost (konkrétně `CANBUS_NO_MESSAGE`), na kterou mohou reagovat ostatní moduly (typicky to znamená postupné ukončení komunikace a přechod do režimu spánku).

Locator zpracovává data zasílané GPS modulem. Ty jsou ve formátu `NMEA 0183` (National Marine Electronics Association) a pro zpracování je použita knihovna `nmea_parser`, poskytovaná jako součást frameworku esp–idf. GPS modul posílá mnoho informací, pro sledování automobilu jsou relevantní hlavně zeměpisné souřadnice a nadmořská výška. Ostatní odesílané informace byly tedy zredukovány na minimum a frekvence zasílání zpráv byla nastavena na 1 zprávu za sekundu. (skrze aplikaci ublox center). Při zafixování polohy jsou přijaté údaje uloženy a připraveny k zaslání na server (při nejbližší odchozí komunikaci).

Communicator zabezpečuje pravidelné odesílání zpráv na server a případné zpracování příchozích příkazů. Při startu vyčkává na pokyn (event bit `CONNECTED`). Po jeho přijetí začne odesílat zprávy.

Tělo každé odesílané zprávy (popsaná v sekci 3.2) obsahuje všechny dostupné informace v čase jejího odesílání. Formát pro komunikaci je json a pro serializaci a deserializaci je využívána knihovna `ArduinoJson`[2]. V případě dostupnosti internetového připojení jsou zprávy sestaveny a odesílány s nastavenou frekvencí. Následně jsou zpracovány příkazy

[2]https://arduinojson.org/v6/doc/

zaslané serverem v odpovědi na zprávu. V případě zpracovávání příkazu je ignorována nastavená zasílaná frekvence a zpráva obsahující odpověď na provedený příkaz je odeslána ihned.

Modul manager zpracovává události zasílané ostatními moduly, které potom určují celkový stav systému. Modul cyklicky kontroluje stav připojení k internetu, úspěšnost odesílání zpráv i aktivitu CAN sběrnice. Při zjištění nedostupnosti internetového připojení dává pokyny k znovupřipojení. Pokud počet neúspěšných pokusů přesáhne nastavený práh, restartuje nebo uspává zařízení (v případě neaktivity automobilu zařízení přechází do režimu spánku pro snížení odběru energie). Stejně tak je spuštěn proces uspání při detekování neaktivity CAN sběrnice. Proces spočívá ve vyvolání události (`EVENT_TO_SLEEP`), po které mají všechny ostatní moduly určen čas na případné korektní ukončení (a případné uložení informací do trvalé paměti). Poté jsou nastaveny události umožňující probuzení zařízení (timer a externí přerušení ze sběrnice CAN), je vypnuto napájení periférií (GSM a GPS) a CAN kontrolér je přepnut do standby režimu. Nakonec je přepnut do režimu spánku také samotný mikrokontrolér.

Další části implementace jsou v souborech `Settings` a `Storage`. Modul storage obsahuje metody umožňující načítání a ukládání souborů do trvalé paměti. `Settings` obsahuje logiku serializace a deserializace parametrů používaných v celém systému. V případě nekorektního načtení parametrů uložených v paměti jsou použity předvolená nastavení (například při vadných sektorech paměti, nebo při prvním spuštění). V ostatních případech jsou použity načtená nastavení. Nastavování těchto parametrů je uvedeno v sekci 4.3

Zajímavé části implementace

Synchronizace mezi úlohami – probíhá z využitím api `xEventGroup`. Jedná se o soubor funkcí umožňující synchronizaci a komunikaci prostřednictvím událostí a je součástí esp-idf (také FreeRTOS). Pro synchronizaci jsou vytvořeny dvě skupiny událostí – `managerEvents` a `networkEvents`. V první jsou spravovány události týkající se celého systému, v druhé potom události týkající se pouze připojení k internetu. Každá skupina má definovaných několik bitů, které reprezentují definované události. Výhoda takovéto synchronizace je v možnosti uspání úlohy (odevzdání procesoru) a pasivnímu čekání do nejbližší změny sledovaného bitu.

Šifrování komunikace – jedním z požadavků při návrhu bylo také šifrování komunikace (srze https). Tento požadavek se podařilo splnit. Zařízení komunikuje se serverem výhradně skrze https a k ověřování používá kopii veřejného klíče uloženého v paměti. Samotné šifrování potom používá self_singed certifikáty. Využití stejného certifikátu jako při webovém přístupu k aplikaci by bylo možné, ale zkomplikovalo by řešení nutností pravidelného (a bezpečného) stahování nového veřejného klíče. Certifikát má nastavenou relativně dlouhou dobu platnosti (oproti certifikátům LetsEncrypt použitých v ostatní komunikaci) a jeho aktualizace je možná aktualizací firmwaru.

Aktualizace firmwaru – zařízení také umožňuje vzdálené aktualizování firmwaru (OTA – over the air). Jeho distribuce probíha na vyžádání (příkazem `updateFirmware`) a je funkční skrze Wifi i GSM připojení. OTA aktualizace ve frameworku esp–idf také podporují návrat k poslední správné verzi firmwaru při nahrání chybné aktualizace. Zařízení má dvě oblasti paměti vyhrazené pro uložení firmwaru které cyklicky využívá. V případe

chybné aktualizace a následnému pádu celého systému je automaticky přepnuto zpět na předchozí verzi firmwaru.

Výběr verze frameworku esp-idf – při implementaci byla několikrát změněna verze. Některé využívané části obsahovaly chyby a pomohl až update na vyšší verzi. Příkladem může být PPPoS (Point to Point Protocol over Serial) používaný při komunikaci s modemem. V první testované verzi (4.0.x) bylo spojení nestabilní. Aktualizací na verzi 4.1.x se problém vyřešil.

Použité knihovny – při implementaci byla využita knihovna ArduinoJson[3] a knihovny frameworku esp–idf. ArduinoJson umožňuje práci s dokumenty ve formátu `JSON`. Podporuje dynamické i statické alokování paměti, serializaci, deserializaci i vyhledávání klíčů. V softwaru je využívána při příchozích a odchozích zprávách a při ukládání nastavení a souborů do trvalé paměti.

Synchronizace systémového času – základem systematického sbírání dat je přesné zaznamenávání času získání naměřených hodnot. Zařízení nedisponuje RTC (real time clock) modulem, je proto nutné získávat z připojených zařízení. Software získává čas při připojení na internet skrze SNTP, nebo z GPS modulu při dostupnosti dostatečného množství družic. Zařízení je schopno také nastavovat čas získaný s CAN sběrnice automobilu, ale od tohoto způsobu získávání času bylo upuštěno. Automobil neeviduje datum, pouze čas a zajištění času předešlými dvěma způsoby se ukázalo být dostatečné.

Všechny zaznamenávané časy jsou zapisovány ve formátu `unix timestamp` a mají časové pásmo UTC.

Zpracování a vytváření událostí o překročení prahové hodnoty – pomocí příkazů `addTreshold` a `delTreshold` (popsány v 4.3) je možné přidávat a odebírat sledování překročení nastaveného prahu. Při zjištění nové hodnoty senzoru je porovnána s prahovou hodnotou a v případě jejího překročení je vytvořena nová událost. Ta se skládá z názvu senzoru, který hodnotu překročil, hodnoty, jakou jej překročil a čas kdy tato událost nastala. Záznam je uložen do paměti a při nejbližším odesílání je odeslán. Pokud odesílání selže (nebo ani nenastane z důvodu nepřipojení sítě), záznam je uložen do trvalé paměti, odkud je při příštím spuštění načten a korektně odeslán (pokud ne proces se opět opakuje).

Trvalá paměť – pro ukládání nastavení a neodeslaných událostí je použita část SPI flash paměť osazená na modulu ESP32. Její velikost se liší napříč moduly, u použitého modulu je 4096 Kb. Tato paměť je rozdělena na několik particií (tabulka 4.4) a jsou do ní uloženy také samotné firmwary zařízení (`ota_0` a `ota_1`), konfigurace integrovaného Wifi modulu (`phy_init` a `nvs`) a informace o zapsaných verzích fimwaru (`otadata`). Pro zmíněné ukládání dat se používá část `storage` o velikosti 512 Kb. Návrh ukládání do trvalé paměti byl vypracován tak, aby byla paměť co nejméně degradována častými zápisy. Použitá paměť by měla být schopna 100 000 přepisovacích cyklů.

Nemožnost iniciovat komunikaci ze strany serveru – z důvodu šetření energie jsou bezdrátové komunikační moduly většinu času uspány (GSM i WiFi), což znemožňuje možnost oboustranného iniciování komunikace. Z pohledu monitorování je dostatečné iniciování

[3] https://arduinojson.org/

Jméno	Typ	Podtyp	Ofset	Velikost	Flag
nvs	data	nvs	0x9000	0x4000	x
otadata	data	ota	0xD000	0x2000	x
phy_init	data	phy	0xF000	0x1000	x
ota_0	app	ota_0	0x10000	1280K	x
ota_1	app	ota_1	x	1280K	x
storage	data	spiffs	x	512K	x

Tabulka 4.4: Tabulka určující rozdělení paměti – partitions.csv

komunikace pouze ze strany zařízení v automobilu, avšak při snaze automobil ovládat vzniká možný problém, kdy uživatel chce vykonat příkaz a zařízení je v režimu spánku. Tyto situace by bylo možné částečně eliminovat systematickými změnami ve frekvenci probouzení zařízení. Například by zařízení mohlo přejít do režimu spánku až jednu hodinu po zaparkování automobilu. Dále by se probouzelo relativně často (například každých pět minut) a s delším odstupem času od poslední aktivity automobilu by byly prodlužovány intervaly ne-aktivity.

Další možností je záměrné zvýšení aktivního času (frekvencí probouzení nebo prodloužením času vyčkávaného před přepnutím do režimu spánku) v časových intervalech, které jsou typické pro ovládání automobilu. Například pokud by zřízení podporovalo vzdálený výhřev kabiny automobilu a řidič jej typicky spouštěl vždy v 7:30 ráno, zařízení by v tento čas bylo probuzeno a čekalo na příkaz.

Obě tyto možnosti jsou proveditelné a záleží od serverové implementace, protože zařízení umožňuje krátkodobé i trvalé změny v nastavení.

Nastavitelné parametry zařízení

Chování zařízení je možné upravit změnou následujících parametrů:

messageFreq – nastavuje časový údaj (v sekundách), který bude systém čekat před odesláním zprávy. Čekání začíná posláním předešlé zprávy, je tedy možné, že při pomalém připojení budou zprávy chodit s většími časovými rozestupy.

canbusTimeout – určuje po kolika sekundách neaktivity bude sběrnice považována za neaktivní. Po této době modul `Canbus` nastavuje event `CAN_NO_MESSAGE` a postupně začíná uspávání zařízení.

wakeupFreq – určuje nastavení časovače k probouzení z režimu spánku

errorWakeupFreq – určuje nastavení časovače k probuzení z režimu spánku a je použita při opakovaných problémech s připojením k internetu. Používá se aby se předešlo k zbytečnému vybíjení baterie při parkování v oblastech bez pokrytí GSM sítěmi.

authToken – token používaný při veškeré komunikaci se serverem

webApi – určuje adresu serveru, která je používaná při komunikaci

stayAlive – pomocný parametr umožňující vynucení neuspávání systému po danou dobu.

sendOnlyChanges – experimentální parametr upravující selekci odesílaných zpráv tak, aby byly posílané pouze parametry, které byly od posledního odeslání změněny. Při současné implementaci se nevyužívá, nicméně mohl by být použit pro zmenšení přenášeného objemu dat.

Popis příkazů určených na ovládání zařízení

Příkazy jsou zasílány jako odpověď na zprávu se senzorovými daty a umožňují měnit nastavení zařízení, nebo ovlivňovat připojený automobil. Formát zprávy je opět `JSON` obsahující pole `commands`(dle návrhu 3.2). Hodnoty ovlivňované příkazy obsahující hodnotu `{pernament: true}` jsou natrvalo uložené do paměti, ostatní mají efekt pouze do nejbližšího restartu systému. V systému jsou definované následující příkazy:

setMessageFreq: upravuje hodnotu `messageFreq` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val` a jednu volitelnou `pernament`. Hodnota `val` musí být nezáporná celočíselná konstanta.

setCanbusTimeout: upravuje hodnotu `canbusTimeout` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val` a jednu volitelnou `pernament`. Hodnota `val` musí být nezáporná celočíselná konstanta.

setWakeupFreq: upravuje hodnotu `wakeupFreq` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val` a jednu volitelnou `pernament`. Hodnota `val` musí být nezáporná celočíselná konstanta.

setErrorWakeupFreq: upravuje hodnotu `errorWakeupFreq` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val` a jednu volitelnou `pernament`. Hodnota `val` musí být nezáporná celočíselná konstanta.

stayAlive: nastavuje hodnotu `stayAlive` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val`. Hodnota `val` musí být nezáporná celočíselná konstanta. Tento příkaz neumožňuje uložení typu `pernament` pro zamezení chyb vedoucích k vybití baterie.

updateFirmware: spouští OTA aktualizaci firmwaru. Obsahuje jednu povinnou hodnotu `val` typu string, s URL adresou k binárnímu souboru obsahující nový firmware. URL adresy jsou omezeny pouze na ty, které budou distribuovány skrze HTTPS zašifrované uznávaným certifikátem.

setApiUrl: nastavuje parametr `apiUrl` používaný pro autorizaci u odcházejících zpráv. Obsahuje jednu povinnou hodnotu `val` typu string, určující nový token. Druhý volitelný parametr `pernament` určuje zdali bude token uložen do trvalé paměti.

getWifiAps: zašle jména uložených bezdrátových sítí typu WiFi, které zařízení používá. Odpověď je zaslána obratem v v sekci `cmdAnswer` pod hodnotou wifiAps. Parametr `pernament` by v tomto případě nedával smysl.

setWifiAp: přidá zaslanou kombinaci názvu a přístupového hesla k bezdrátové sít do seznamu naučených sítí. Obsahuje povinnou hodnotu `ssid` a `pass`. Obě hodnouty jsou typu string. Parametr pernament je v tomto případě zbytečný, změny jsou vždy uloženy do trvalé paměti.

delWifiAp: odebere uvedený název WiFi sítě z uloženého seznamu. Obsahuje jednu povinnou hodnotu `val` typu string, definující název sítě. Parametr pernament je implicitně nastaven .

restart: provede korektní restartování zařízení (s uložením a postupným vypnutím všech částí). Neobsahuje žádné volitelné ani povinné hodnoty.

hardRestart: provede „tvrdý" restart zařízení (bez uložení a vypínání částí). Neobsahuje žádné volitelné ani povinné hodnoty.

resetToDefaults: nastaví všechny nastavitelné parametry na přednastavené hodnoty a smaže uložená nastavení z trvalé paměti. Neobsahuje žádné volitelné ani povinné parametry.

saveSettings: uloží aktuálně používané parametry (i ty původně nezaslané s hodnotou `pernament`) do trvalé paměti zařízení. Nemá žádné parametry.

sendOnlyChanges: upravuje hodnotu `sendOnlyChanges` (popsáno v 4.3). Příkaz obsahuje jednu povinnou hodnotu `val` a jednu volitelnou `pernament`. Hodnota `val` je typu boolean.

addTreshold: umožňuje přidání prahové hodnoty a porovnávací funkce k některému z monitorovaných senzorů. Obsahuje povinné parametry `tType` (hodnoty e, n, l, g), `sensor` (hodnotou je název senzoru, kterému je nastavován treshold) a `val`, což je samotná prahová hodnota, proti které bude vykonáváno porovnání (popsáno vs 4.3). Příkaz může obsahovat volitelný parametr `pernament`.

delTreshold: slouží k odebrání nastaveného sledovaného překročení práhu. Obsahuje povinný parametr `val` určující jméno senzoru, kterému bude sledování překročení prahu odebráno (popsáno vs 4.3). Parametr `pernamet` je v tomto případě implicitní.

nop: odešle skrze CAN sběrnici zprávu, která nemá vliv na žádnou jednotku na sběrnici. Po odeslání jsou řídící jednotky připojené na sběrnici na krátkou dobu probuzeny a je možné zaznamenat jejich stavy. Nemá žádné parametry

lock: určeno k odeslání sekvence zpráv, která má za následek uzamknutí dveří automobilu. Nemá žádné parametry.

lock: určeno k odeslání sekvence zpráv, která má za následek uzamknutí dveří automobilu a spuštění alarmu. Nemá žádné parametry.

unlock: určeno k odeslání sekvence zpráv, která má za následek odemknutí dveří automobilu. Nemá žádné parametry.

unlock: určeno k odeslání sekvence zpráv, která má za následek odemknutí dveří automobilu. Nemá žádné parametry.

openWindows: určeno k odeslání sekvence zpráv, která má za následek otevření oken. Nemá žádné parametry.

windowsDown: určeno k odeslání sekvence zpráv, která má za následek otevření oken. Nemá žádné parametry.

windowsUp určeno k odeslání sekvence zpráv, která má za následek zavření oken. Nemá žádné parametry.

4.4 Zpracování dat a jejich distribuce

Data získaná prostřednictvím monitorovacího zařízení je potřeba mírně zpracovat a systematicky ukládat do databáze, odkud mohou poté být distribuovány klientovi do mobilní aplikace. Tyto úlohy řeší serverová aplikace, která je popsána v této sekci.

Pro její implementaci byl dle návrhu zvoleno prostředí NodeJS. Pro usnadnění práce se zpracováním HTTP požadavků byl použit framework `express`. Javascript-ový správce balíčků `npm` poskytuje veliké množství hotových modulů a knihoven, které je možné použít. V této serverové implementaci byly použity následující balíčky:

- `mongoose`– knihovna pro práci s databází MongoDB
- `moment` – knihovna pro práci s časovými údaji
- `espress` – framework
- `express-session`, `passport` a `connect-mongo` – knihovny pro zajištění autentizace uživatele a pro správu a ukládání cookies

Implementace byla rozdělena na dvě části, první zpracovává příchozí a odchozí zprávy ze zařízení v automobilu, druhá potom zprostředkovává předzpracování a distribuci dat pro webového klienta.

Komunikace se zařízením

Část aplikačního rozhraní (API), kterou úlohou je komunikace se zařízením má pouze jeden endpoint (`/api/carlogger`). Sem jsou zasílána zaznamenaná data ze senzorů (popsána v sekci 3.2) skrze protokol HTTP metodou `POST`. Data jsou v případě absence povinných částí doplněna o předvolené hodnoty. Zasílané údaje v části `carState` jsou potom rovnou uložena do databáze. Zbylé části `deviceState` a `cmdAnswer` jsou uloženy pro přeposlání klientovi. V případě absence zasílaných příkazů je zařízení odpověděno pouze stavovým kódem 200 (OK). Pokud jsou nachystané nějaké příkazy k odeslání, jsou přiloženy k zasílané odpovědi (dle návrhu, uk. 3.2).

Autorizace je zajištěna pomocí tokenu `x-access-token` v hlavičce zpráv, které přikládá zařízení na veškeré odchozí zprávy. V případě jeho absence nebo uvedení neplatného tokenu odpovídá server stavovým kódem `401`.

Tato část rozhraní je dostupná skrze HTTPS komunikaci, která používá jiný pár klíčů než zbytek sytému(v tomto případě self signed certifikát). Je to z důvodu zjednodušení

validace certifikátu na straně zařízení a zároveň možnosti použití běžného a uznávaného certifikátu při ostatních webových službách (například od cert. autority LetsEncrypt).

Komunikace s klientem

Druhá část implementace serveru se zabývá předzpracováním a distribucí dat pro webovou aplikaci. Data jsou k dispozici pomocí cca 15 definovaných zdrojů (API endpointů). Zasílaná data jsou ve formě `JSON` a jsou získávaná z databáze pomocí `agregation pipeline` (MongoDB obdoba SQL dotazů). Některé komplikovanější endpointy jsou získávány pomocí několika dotazů do databáze, které jsou poté následně zpracovány v prostředí Javascriptu.

Autentizace probíhá skrze plugin `passport`, který podporuje ukládání uživatelských přístupů v databázi MongoDB. Při autentizaci je ve webovém klientovi uloženo HTTP cookie, které zajišťuje budoucí autentizaci.

Zajímavé části implementace

Získání dat pro zobrazení MapView: Pro zobrazení denního přehledu jízd a parkování je potřeba přepracovat a roztřídit data z databáze, které jsou ve formě záznamů agregovaných dle času (24 záznamů do dne). Postupně jsou vyfiltrovány záznamy ve kterých nebyla uložena poloha automobilu a záznamy které jsou nadbytečné, protože rozlišení detekce bylo nastaveno na jednu minutu. Následně jsou rozděleny na jízdy (parametr `engineOn` má hodnotu `true`) a na parkování (`engineOn` = `false`). U parkování je předpokládáno, že auto nebylo v pohybu, stačí tedy jeden záznam o poloze. U jízd je počet zobrazovaných poloh snížen na jeden údaj o poloze za minutu. Pro potřeby zobrazování je to poměrně dostatečná přesnost a tato úprava redukuje problematický velký počet záznamů při dlouhých jízdách.

4.5 Ukládání dat

Zvolené databázové řešení je NoSQL databáze MongoDB, která pracuje s dokumenty (pojmenování jednoho záznamu v prostředí MongoDB) ve formátu JSON Pro zjednodušení validace, vyplnění předvolených hodnot a pro všeobecné zjednodušení práce s databází byla použita zmiňovaná knihovna `mongoose`. Ta umožňuje definování schém a modelů, vůči kterým jsou poté vkládané záznamy validovány. Součástí schém je také nastavení indexů, které zrychlují vyhledávání v záznamech. Návrh schémy používanou pro ukládání dat do databáze je možné vidět v uk. 4.1.

```
{
   "hourTimestamp": {
      "type": "Date",
      "index": "true",
      "unique": "true",
      "default": "Date.now"
   },
   "odometerMax": "Number",
   "odometerMin": "Number",
   "numberOfReadings": "Number",
   "carWasRunning": {
      "type": "Boolean",
      "default": false
   },
   "alert": {
      "type": "Boolean",
      "default": "false"
   },
   "sensorReadings": ["SensorReadingdata"]
}
```

Ukážka 4.1: Schéma dat ukládaných do databáze

Při naivním ukládání dat do databáze (1 soubor dat ze senzorů = jeden dokument) by se velmi rychle navyšoval celkový počet dokumentů a s tím přibližovali spojené problémy (např. velikosti vyhledávacích indexů). Pro ukládání dat ze senzorů byla tedy zvolena technika (angl.nazývaná „bucketing"[14]), která snižuje celkový počet dokumentů ukládáním časových záznamů do souborů definovaných jednotným časovým intervalem (v tomto případě jedna hodina). Do jednoho dokumentu jsou tedy uloženy všechny záznamy ze senzorů získané během jedné hodiny (uk. 4.1). U dat zaznamenávaných v této databáze většinou nebude důvod prohlížení jediného záznamu, proto agregace po hodinách také ulehčí práci při jejich získávání. Důležité data ze senzorů, dle kterých by mohlo být vyhledáváno, jsou také agregovány (příslušnou funkcí) do samotného dokumentu reprezentujícího danou hodinu. Je tedy možné filtrovat například pouze hodiny, ve kterých bylo auto v pohybu.

Pro popis dat získávaných z databáze je v případě MongoDB využíváno `MongoDB Query Language`[11] (`MQL`). Pro vytvoření komplexních dotazů je možné využít `agregation pipeline`. V té je možné postupně zřetězit několik běžných dotazů a docílit tak získání žádaných dat.

Databáze MongoDB byla zvolena také pro možnost využití cloudového řešení Atlas[4]. Ten obsahuje grafické rozhraní umožňující procházení uložených záznamů, pomocníka pro vytváření `agregation pipeline`, zobrazování statistik či vytváření grafů. V bezplatné verzi umožňuje uložení databáze o velikosti max 500 MB, což na období ladění a vývoje plně postačující. Při placené verzi umožňuje také využití distribuovaných databázových clusterů umožňujících lepší rozložení zátěže a možnosti škálování. Pro účely této práce však byla databáze vyexportována a přesunuta na stejný server, kde běží i ostatní serverové aplikace. Vytížení databáze pro účely monitorování jednoho automobilu nebude nijak zásadní a není tedy nutné dlouhodobě využívat takovéto řešení.

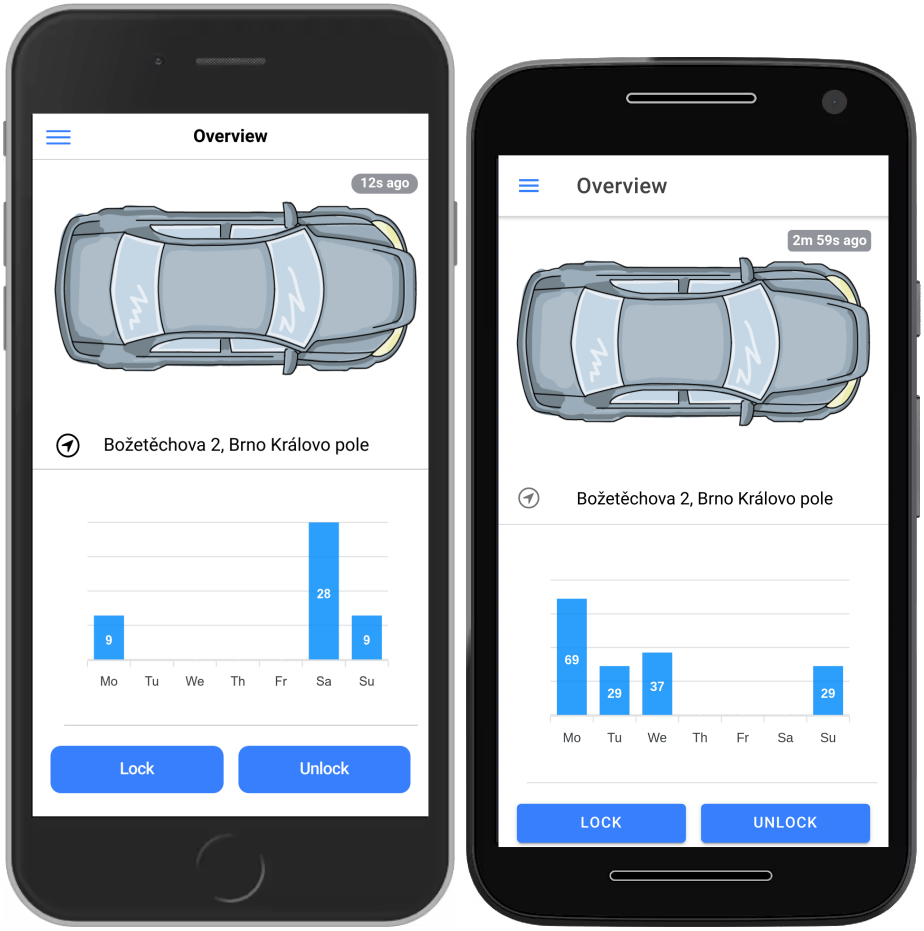
[4] https://mongodb.com/cloud/atlas

Velikost databáze

Velikost uložených dat závisí od frekvence používání auta, protože při neaktivitě (parkování automobilu) se vytváří pouze pár záznamů do hodiny. Při běžném provozu automobilu se velikost databáze zvětšila za 10 dní asi o 4,5 MB, z čehož dopočítaná roční velikost databáze je asi 1650 MB. Pro snížení velikosti databáze by bylo možné docílit například průměrováním ukládaných hodnot, redukcí frekvence ukládání dat, nebo jejich selektivním mazáním s odstupem času. Při výběru vhodného řešení je nutné se zamyslet, jak konkrétní a jak stará data jsou potřeba ukládat, k čemuž je nutno znát účel jejich ukládání. V tomto případě (a stejně tak i v případě továrního řešení `VW-Carnet`) je smysl sbírání dat víceméně krátkodobého charakteru. Horizontem určujícím nutnost ukládání veškerých detailních průběhů jízd byl určen jeden rok stáří záznamu. Do této doby bude tedy možné sledovat konkrétní naměřené hodnoty tak, jak byly naměřeny a po této době budou data zredukována.

4.6 Přístup a ovládání skrze webového klienta

Pro přístup k zaznamenaným údajům a ovládání byla vytvořena webová aplikace. Aplikace je primárně přizpůsobena rozhraní mobilního telefonu, nicméně je možné ji používat i v běžném desktopovém prostředí pomocí internetového prohlížeče.



Obrázek 4.14: Ukážka přizpůsobení vzhledu na systémech iOS a android

Implementace

Při implementaci byl použit framework `Ionic`, určený pro tvorbu mobilních webových aplikací v kombinaci s frameworkem `vue` – `ionic-vue`[5].

Výhodou použití webové stránky je jednodušší vývoj a udržování aplikace napříč různými ekosystémy. Pokud webová stránka plní požadavky PWA ([6]), je možné ji nainstalovat do podporujících systémů, kde se potom tváří jako běžná aplikace. Tento přístup není nejoptimálnější a není možné při něm dosáhnout stejného komfortu jako při nativních aplikacích, avšak pro jednoduché aplikace je dostačující. Framework Ionic podporuje odlišení uživatelského prostředí dle používaného systému, čímž je dosaženo přizpůsobení vzhledu prvků aplikace běžným aplikacím daného systému (obrázek 4.14).

Pro získávání dat a komunikaci s automobilem je na pozadí využíváno aplikační rozhraní popsané v sekci 4.4. U většiny stránek (u kterých to dává smysl) probíhá periodická aktualizace dat na pozadí. Manuálně lze data aktualizovat gestem „potáhnutím z hora“.

V aplikaci je využíváno mapového aplikačního rozhraní `Google geolocation API`. Používá se pro zobrazování údajů v mapě a pro převod souřadnic na textový popis adresy (*anlg. reverse geolocation*). Tyto služby jsou zpoplatněné, nicméně Google poskytuje měsíčně kredit 200 €, který je možné využít k jejich zaplacení. Aplikace se (i v režimu vývoje) vyšplhala na sumu cirka 10€ za měsíc, na což by měl stačit zmiňovaný kredit i s dostatečnou rezervou. Alternativně by mohly být využito opensource řešení OpenStreetMap, které také poskytuje podobné aplikační rozhraní bezplatně. Výhodou použití Google řešení je lepší podpora na mobilních zařízeních a možnost propojení s mapovou aplikací Google Maps.

Při vývoji bylo využito pomocného rozhraní pro vývoj ve frameworku Ionic, které vytvoří lokální webový server a zabezpečuje propagaci změn ve zdrojovém kódu klientovi. Toto řešení značně urychlí vývoj, protože umožňuje prakticky okamžité sledování následků provedených změn.

Popis uživatelského rozhraní

Samotná aplikace byla rozvržena do čtyrech stránek, mezi kterými je možné přepínat pomocí postranní nabídky (obr. 4.15).

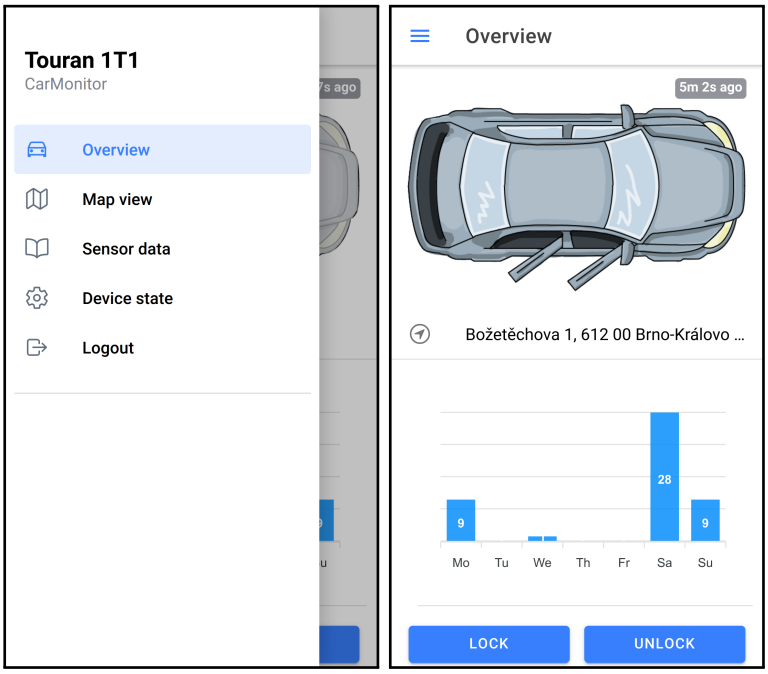
Overview – toto je první stránka, kterou uživatel uvidí po přihlášení do aplikace, čemuž byly také uzpůsobeny zobrazované údaje (náhled v obr. 4.15). Nacházejí se zde informace o automobilu, které typicky uživatel bude často zobrazovat. V její vrchní části se nachází obrázek interpretující stav otevření dveří, kufru a kapoty automobilu. Vpravo nad obrázkem je stáří zobrazovaných údajů určené časem uplynulým od poslední komunikace se zařízením. Pokud by zařízení nezaslalo zprávu po dobu delší než je obvyklé, ukazatel času změní barvu, aby indikoval abnormální stav.

Pod miniaturou je adresní popis polohy automobilu získaný ze souřadnic pomocí `Google reverse geolocation API` (popsáno v sekci 4.6). Adresný popis je zároveň odkazem na přesnou polohu do aplikace Google maps.

Ve spodní části stránky se nachází graf zobrazující kilometry najezděné za poslední týden, tlačítka vyvolávající akce v automobilu a informace z vybraných senzorů.

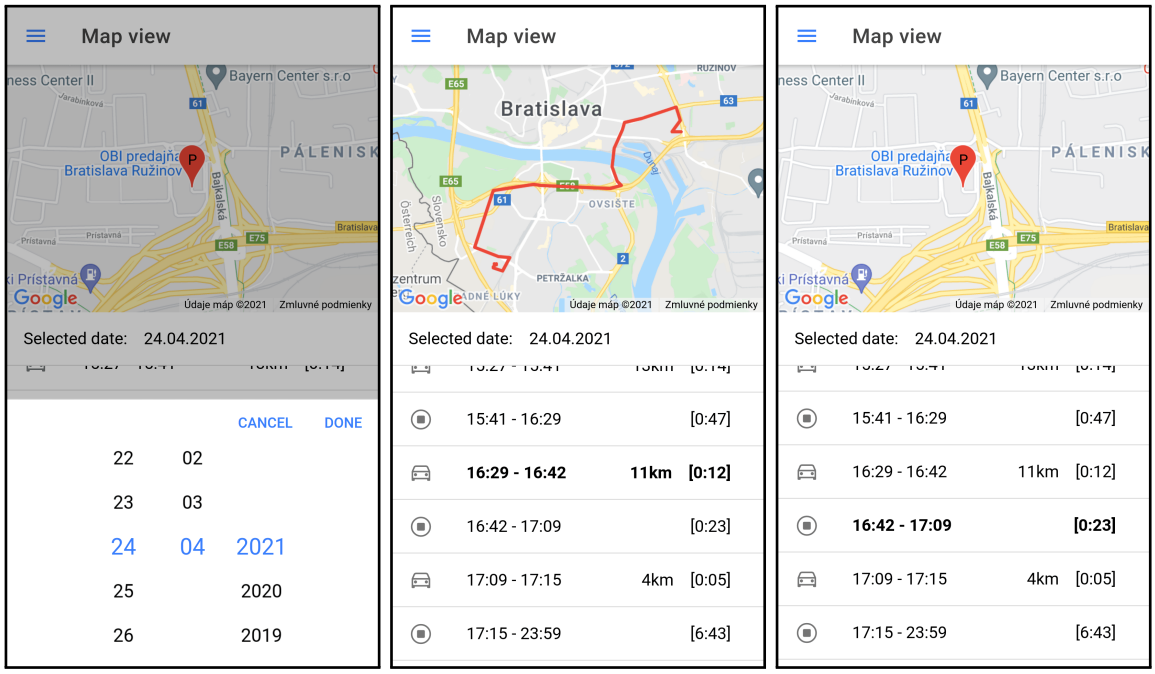
Zobrazovaná data jsou automaticky aktualizována.

[5] https://ionicframework.com/vue
[6] https://web.dev/progressive-web-apps/



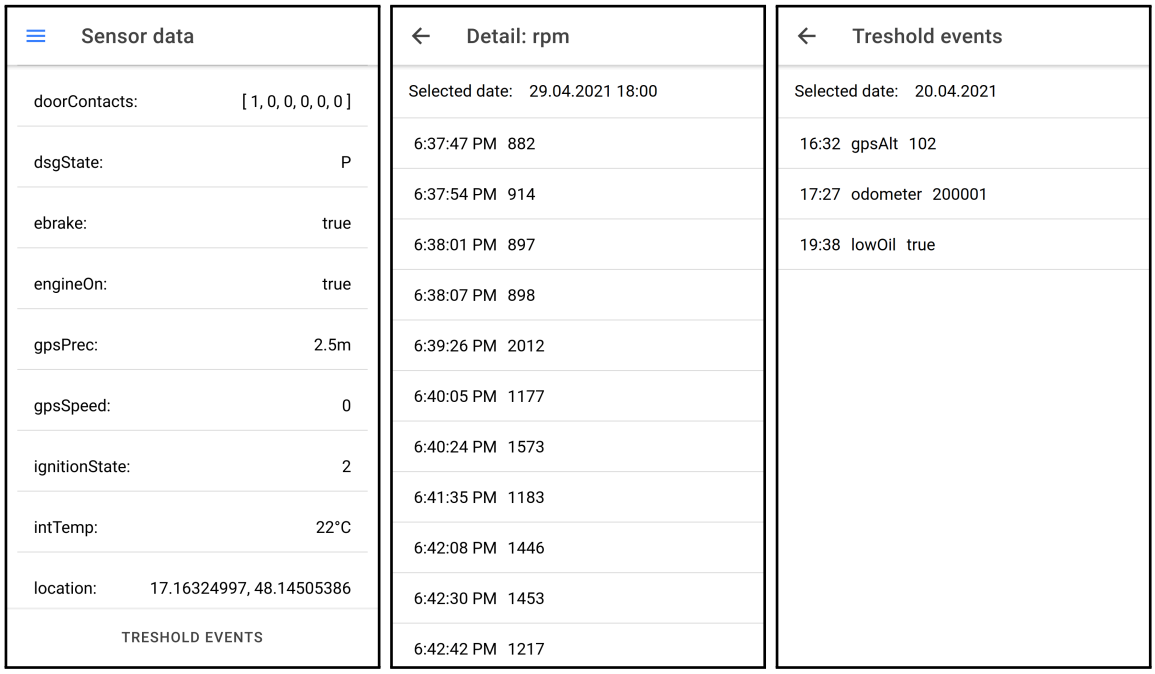
Obrázek 4.15: Postranní nabídka aplikace a stránka `Overview`

Map view – umožňuje prohlížení historie polohy automobilu na mapovém podkladu. Po výběru data je zobrazen chronologicky uspořádaný seznam událostí z daného dne. Dostupné typy událostí jsou parkování a jízda. U každé z nich je možné vidět začátek a konec události, délku trvání a případně u jízdy také její délku v kilometrech. Po zvolení konkrétní události je její průběh vyobrazen na mapě ve vrchní polovině obrazovky. V případě parkování automobilu je poloha vyznačena značkou, v případě jízdy potom čárou vykreslující její průběh. Ukážku je možné vidět na obr. 4.16.



Obrázek 4.16: Stránka `Map view` zobrazující výběr zobrazovaného dne, zaznamenanou jízdu a parkování

Sensor data – obsahuje aktuálně získaná data, prohlížení jejich historie a zobrazování událostí o překročení nastavených prahových hodnot (možno vidět na obr. 4.17). Aktuální naměřené hodnoty obsahují data, která byla získána při poslední komunikaci s automobilem. Pokud ve zprávě z automobilu data o zobrazovaném senzoru chyběla, hodnota není zobrazena. Kliknutím je možné vybrat senzor u kterého bude otevřeno zobrazení jeho zaznamenaná historie. Zobrazení historických dat je možné uzpůsobit zvolením data a hodiny, ve vrchní části obrazovky. Podobně funguje také výpis překročených nastavených prahových hodnot, který je možné otevřít stisknutím tlačítka „Treshold events" ve spodní části obrazovky.



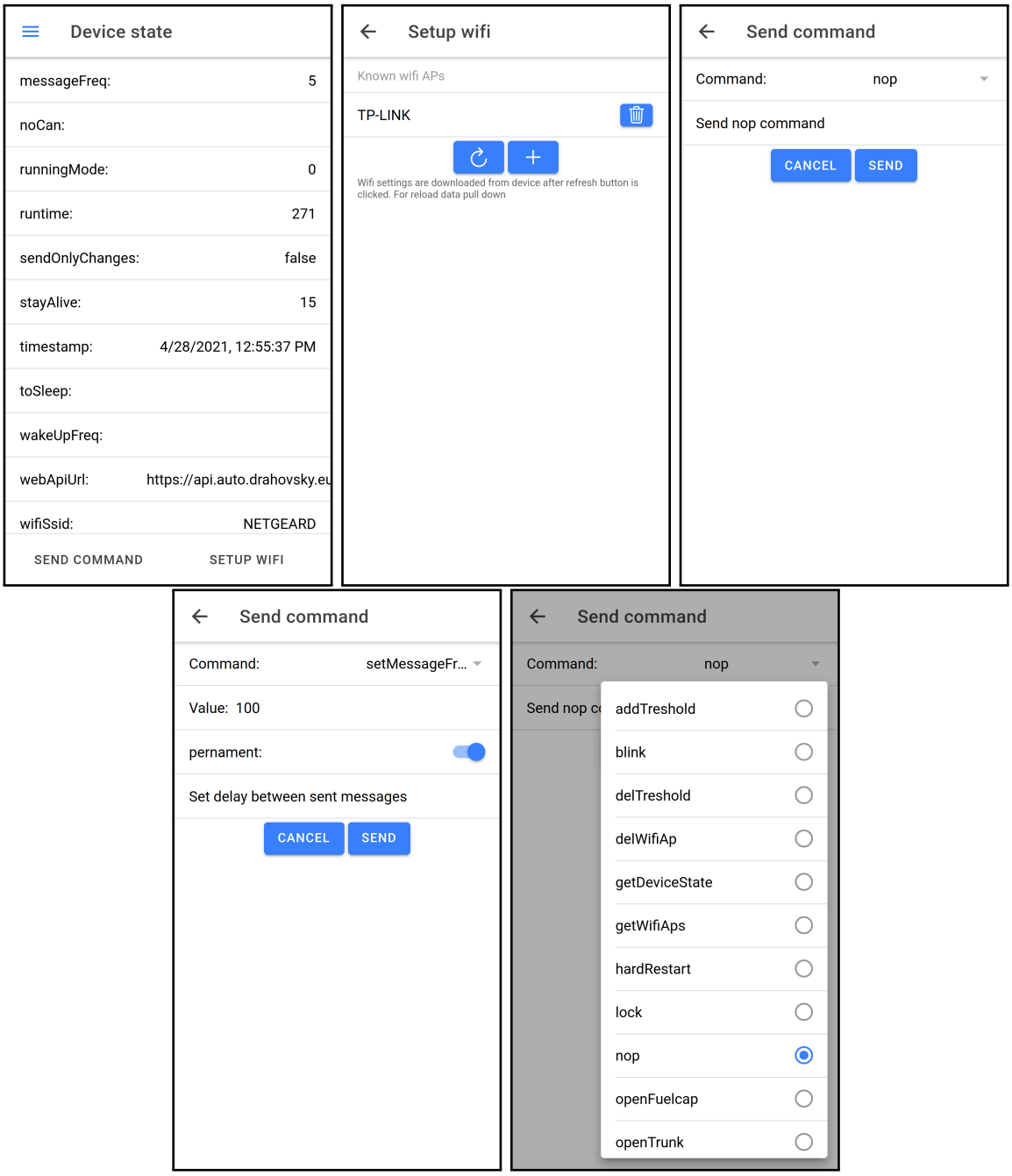
Obrázek 4.17: Nabídky stránky `Sensor data` – aktuální zobrazení, zaznamenaná historie senzoru `rpm`a uložené překročení prahových hodnot

Device state – slouží k zobrazování a nastavování parametrů zařízení, zasílání příkazů a správě uložených přístupových údajů k bezdrátovým přístupovým bodům. Všechny informace v tomto souhrnu podstránek mají význam spíše při nastavování a ladění zařízení než při běžném používání aplikace. Náhled popisovaných stránek je možné vidět na obr. 4.18

První zobrazený přehled ukazuje aktuální nastavení zaslané zařízením. To jej zasílá po obdržení příkazu `getDeviceState`, nebo při každém restartu a změně parametrů. Ve spodní části se nachází nabídky `Send command` a `Setup Wifi`.

`Send command` umožňuje zaslání předdefinovaného příkazů do zařízení. Každý příkaz obsahuje krátký popis shrnující jeho vliv na systém a u některých je možné také definovat přizpůsobující parametry.

Na podstránce `Setup Wifi` jsou zobrazeny názvy uložených bezdrátových přístupových bodů a je zde možnost mazat uložené záznamy a přidávat nové. Tlačítkem aktualizace se odešle příkaz do zařízení, které poté zašle seznam uložených sítí.



Obrázek 4.18: Nabídka `Device state`

Kapitola 5

Zhodnocení dosaženého řešení

Interní CAN sběrnice automobilu se ukázala být dobrým zdrojem dat pro systematický sběr informací o vozidle. Vybraná sběrnice obsahuje zajímavé údaje, kterých analýzou lze získat podobný přehled o aktivitách automobilu, jako tovární řešení vzdáleného monitorování.

Používaný způsob zjišťování obsahů zpráv zasílaných na monitorované sběrnici byl poměrně přímočarý a vedl k získání zajímavých dat. Z počátku bylo zdlouhavé odfiltrování zpráv, kterých obsahy se neustále měnily (většinou kontrolní počítadla) a znemožňovaly tak nalezení právě hledaného parametru. Vybraný prostředek pro monitorování bych druhý krát nezvolil, protože jeho výroba a odladění zabrala neúměrně mnoho času v poměru k ušetřeným nákladům. Raději než znovu vytvářet USB -> CAN převodník bych zvolil hotový produkt. Jako vhodný se zdá být open-source řešení CANable[1], který se dá pořídit za cenu okolo 1500 Kč.

Zvolená komunikační technologie se zdá být vhodná. Přidání WiFi ke GSM komunikaci při dostupnosti přístupového bodu výrazně urychlilo zasílání zpráv. Při zasílání skrze GSM odesílání zprávy trvá i několik desítek sekund, kdežto skrze WiFi je zpráva typicky odeslána do jedné sekundy. Tato optimalizace umožnila zkrácení celkového aktivního času zařízení, protože automobil je většinu času zaparkován v dosahu některé z definovaných WiFi sítí. Při přenášení dat skrze GSM jsou data účtovaná dle přeneseného objemu. V testovacím období zařízení nepřeneslo více než 15MB dat za měsíc. Reálná spotřeba dat se ukáže až s běžným používáním automobilu (mimo pandemická omezení, které v době vývoje platí), protože spotřeba dat roste se zvyšováním četnosti provozu automobilu.

Vytvořené hardwarové řešení by bylo možné navrhnout a uložit do menší krabičky (optimalizovat by se dala velikost plošného spoje, stejně jako zbytečné volné místo v krabičce), ale při přihlédnutí na množství místa v automobilu to nebyla priorita. Prioritou bylo kompletní vytvoření a osazení v domácích podmínkách, čehož se nakonec podařilo dosáhnout.

Výběr soc (System On Chip) ESP32 je z pohledu výkonu a spotřeby energie pro tento projekt velmi vhodný. Použitím například Raspberry by se zařízení dostalo sice implementačně do úplné jiné kategorie a bylo by možné jej rozvíjet jinými směry (například analýza dat přímo v automobilu), ale pro účely monitorování je současný výkon dostatečný. Oproti osmibitovým mikro–kontrolérům (MCU) (například `ATmega 328P` používaný ve vývojových deskách Arduino Uno) je práce s 32 bitovým dvou-jádrovým kontrolérem mnohem pohodlnější a výsledný kód tak může být přehlednější. I z těchto důvodů bylo při implementaci zvoleno použití frameworku esp–idf místo frameworku Arduino, který je přeci jen koncepčně vhodný spíše pro parametrově slabší MCU. Na druhou stranu esp–idf budí dojem

[1] https://canable.io/

neodladěného a nedokončeného nástroje, který musel být při vývoji několikrát aktualizován (změněná v API nebyla zpětně kompatibilní). Některé používané knihovny jsou stále ve vývojovém procesu a jsou vnich tedy poměrně zásadní změny (například podpora protokolu Point-to-Point Protocol over Serial).

Zvolená serverová implementace v NodeJS se ukázala být pro zpracování dat, jejich ukládání do databáze a distribuce klientovi dostatečná. Při dramatickém zvýšení počtu klientů a monitorovaných vozidel by zřejmě řešení vyžadovalo optimalizaci, nicméně při monitorování jednoho automobilu je možné server provozovat i na relativně slabém hardwaru.

Vytvořená webová aplikace umožňuje sledování živých a zaznamenaných dat v mobilním telefonu i v desktopovém prohlížeči. Díky využití konceptu PWA je možné aplikaci do zařízení nainstalovat, což umožnilo upuštění od nutnosti otevírání prohlížeče pro spuštění aplikace. Použití webových aplikací není nejoptimálnější a nativní aplikace pro všechny používané systémy by zřejmě umožnila plynulejší běh, nicméně vytvořená aplikace není nikterak náročná (nejedná se například o hru, kde by zajištění maximálního výkonu bylo stěžejní) a pro její běh je takovýto přístup dostatečný.

5.1 Možné směry vylepšení

Při vytváření řešení bylo nalezeno několik oblastí ve kterých by mohlo být vytvořené řešení vylepšeno, nebo rozšířeno.

Změna bezdrátové technologie mobilních sítí – použitý modul `SIM00L` používá pro komunikaci technologii GSM, která je sice pro zasílání takto malých zpráv dostačující, ale v budoucnu bude pravděpodobně vypnuta. Vypínání GSM v jiných krajinách již začalo a postupně se zřejmě dostane i k nám, nicméně dá se očekávat, že vypnutí tohoto u nás nejstaršího používaného typu sítě nebude tak rychlé, jako v ostatních zemích. Zejména je to způsobeno nedostatečným pokrytí sítěmi typu 3G, což by způsobilo nedostupnost signálu pro velikou část území. V případě dlouhé životnosti vytvořeného řešení bude nutné vyměnit osazený bezdrátový modul. V současné době existují moduly podporující 4G sítě, které také používají UART rozhraní (stejně jako používaný `SIM800L`). Jedná se například o modul `SIM7600A`, kterého cena se pohybuje okolo 1500 Kč. V obchodech je k dostání také v podobné formě jako používaný GSM modul, po hardwarové stránce by se tedy mohlo jednat opravdu pouze o odpojení starého modulu a zapojení modulu nového. Po softwarové stránce budou nutné mírné úpravy v části kódu obsluhující připojení k internetu (modul `NetManager`) a to zřejmě pouze v inicializační sekvenci. Komunikace probíhá stejně, skrze `AT commands`.

Rozšíření o záložní baterii – pokud by mělo zařízení fungovat podobně jako alarm a umožnit i monitorování vozidla po jeho odcizení, nabízí se možnost přidání přídavné baterie, která by zařízení udržela v provozu i po výpadku autobaterie (při jejím záměrném odpojení). Pro pár hodin běhu zařízení by měla stačit i relativně malá baterie (např. 18650). Otázkou by byla životnost takovéto baterie, protože by byla většinu času udržována plně nabitá (v zásadě je to podobný problém jako u baterií do UPS, kde se používají speciální – na to uzpůsobené akumulátory).

Rozšíření lokálního ukládání – v současném řešení je pouze minimální část dat ukládaná lokálně. Jedná se pouze o záznamy o překročení nastavených prahových hodnot

(`treshold events`), které jsou ukládány do flash paměti v modulu ESP. Nabízí se možnost rozšíření zařízení o čtečku paměťových karet, na které by bylo možné ukládat podstatně větší množství dat, která by mohla být na server zasílána dávkově. K takovémuto řešení by mohlo být přistoupeno například při vypustění komunikace skrze mobilní sítě a spoléhání se čistě na připojení skrze WiFi.

Ovldádání skrze Bluetooth LE – zajímavým rozšířením by mohlo být také použití integrovaného Bluetooth modulu v ESP32. Bluetooth by se dalo použít pro detekování přítomnosti a plné probuzení zařízení, případně by bylo možné implementovat skrze něj také ovládání. Výhodou Bluetooth je nižší spotřeba energie a také případná dostupnost i v oblastech bez pokrytí jinými bezdrátovými sítěmi. Otázkou je zdali by takovéto řešení stálo za do-implementování a problémy s tím spojené. Záleží, jak často se auto v takovémto místě bez pokrytí běžně používanými sítěmi nachází.

Rozšíření počtu monitorovaných sběrnic – v tomto automobilu se mimo monitorované sběrnice komfortních systémů nacházejí také další CAN sběrnice. Monitorování „motorové" sběrnice by mohlo přinést zajímavé data z pohledu ekonomiky stylu jízdy, servisních úkonů, nebo vlivu kvality paliva na provoz automobilu. K řešení těchto námětů by však bylo zapotřebí dostudování mnoha teorie z jiných oborů a smysl takovéhoto řešení by byl pouze u rozšíření počtu podporovaných a monitorovaných automobilů.

Dalším zajímavým předmětem zkoumání by mohla být sběrnice „infotainment", která by dle dosavadního zkoumání mohla umožnit projekci vlastních informací na zobrazovacích zařízeních automobilu. Nabídky zobrazovaných funkcí na displeji přístrojové desky před řidičem by mělo být možné definovat právě skrze sběrnici „infotainment". Při dekódování obsahů zasílaných zpráv a jejich správném napodobení by tak mělo být možné zobrazovat na integrovaném displeji vlastní, jinak nedostupné, informace o automobilu, nebo klidně jiná zajímavá data získaná z internetu (například počasí).

Po hardwarové stránce by pro rozšíření počtu monitorovaných sběrnic bylo nutné přidat další CAN kontroléry. CAN kontrolér v ESP32 podporuje pouze připojení jednoho externího CAN transceiver-u, bylo by tedy nutné použít například kontrolér MCP2515, který umožňuje obsluhu dalšího CAN transciever-u a je jej možné připojit skrze rozhraní SPI.

5.2 Bezpečnostní aspekty dosaženého řešení

V otázce bezpečnosti systému a provozu automobilu se nabízí několik možných oblastí, které budou projednávány v následujících odstavcích.

Nebezpečí nechtěného ovlivnění systémů automobilu

Při připojování neschválených zařízení na interní sběrnice automobilu, které k tomuto účelu nejsou určeny, vzniká možnost ovlivnění procesů, které jsou skrze tuto síť iniciovány. Při omezení připojovaných zařízení pouze na čtení, by neměla být sběrnice nijak ovlivňována. Nabízí se pouze možnost chyby hardwaru vedoucí například k propojení vodičů sběrnice mezi sebou, zemí nebo 12 V. Sběrnice musí být v tomto směru dobře chráněná, protože takovéto chyby se mohou nastat i bez nutného zásahu, například degradací izolace vodičů věkem, teplotním namáháním, nebo chybou v některé z jednotek vozidla. Dokazují to i chybové výpisy z diagnostiky, např. `Chyba sběrnice - zkrat na kostru`.

Při povolení zápisu na sběrnici je další možností vzniku chyby zaslání zprávy, která by měla za následek nepříznivé ovlivnění zbytku systému. Nastat by tato událost mohla například z následujících důvodů:

- Nepřiměřeně časté zasílání zpráv s vysokou prioritou (nízkým ID), kdy by sběrnicový arbitr začal nestíhat vybírat zprávy zasílané uzly s nižší prioritou, které by poté nebyly doručovány. Nastalo by tak úplné zamezení odesílání zpráv pro některé uzly.
- Zasíláním jedné, nebo několika zpráv, které v běžném provozu automobilu nedávají smysl a dovedou některou z jednotek do nedefinovaného stavu.

Celkově je zasílání zpráv na sběrnici nebezpečnější, protože ať už v případě chyby v programovém kódu, nebo jiném selhání existuje možnost ovlivnění systému. Toto ovlivnění by mohlo mít kritické následky například při nechtěném blokování zpráv připravujících automobil na havárii (přitahování pásů, vystřelení airbagů, atd.). Na zkoumané sběrnici se nenachází tyto kritické části systému, nicméně opatrnost je na místě.

Z těchto důvodů bylo při řešení přistoupeno k blokování zápisu na sběrnici při detekování zapnutého motoru. U zaparkovaného vozidla není tolik kritických rutin, kterých nechtěné narušení by mohlo mít fatální následky. Zároveň při řešení nebyly implementovány úkony, kterých používání by dávalo smysl u jedoucího vozidla.

Riziko odposlechu komunikace

Monitorované senzory poskytují poměrně citlivé údaje (např. souřadnice o poloze) a mohly by teoreticky být motivací pro odposlouchávání komunikace. Pokud připustíme možnost odposlechu zasílaných dat (například v místě přenosu skrze bezdrátový komunikační kanálu), nechtěnému monitorování by měla zabránit zvolená šifrovaná komunikace skrze protokol HTTPS. Při komunikaci je ověřován certifikát serveru s pomocí veřejného klíče uloženého v paměti zařízení. Odchozí komunikace je poté šifrována tímto klíčem (asymetrická kryptografie) a privátní klíč, kterým je zprávu možno dešifrovat je v serverové části řešení. Odolnost samotného zařízení vůči napadení (například. odposlech komunikace s modemem skrze UART), nebyla v tomto případě řešena, protože při fyzické dostupnosti zařízení je pro případného útočníka jednodušší připojení vlastního zařízení.

Riziko převzetí kontroly nad zařízením

Komunikace je zabezpečena šifrováním, stejně jako vzdálená aktualizace firmwaru. Možnou cestou by byla změna firmwaru v zařízení, která by ale vyžadovala fyzickou přítomnost. U zařízení uloženého v útrobách automobilu je nepravděpodobné, aby se k němu útočník dostal. V případě produkčního řešení je možné využít šifrování flash paměti, které by zamezilo extrahování a i případnému nahrání jiného firmwaru.

Kapitola 6

Závěr

Cílem této práce bylo zajistit přístup k informacím o řídících systémech vozidla a navrhnout vestavěný systém, který bude spolu s cloudovou aplikací umožňovat odposlouchávání a monitorování komunikace mezi řídícími jednotkami na sběrnici CAN BUS.

Nejvhodnějším zdrojem informací se ukázala být CAN sběrnice komfortních systémů. Při experimentech a zkoumání dění na sběrnici se mi podařilo odhalit více než 20 monitorovatelných stavů, které jsou zasílány řídícími jednotkami na této sběrnici. Podařilo se také zprávy zasílat a zasláním různých sekvencí zpráv docílit například uzamčení, nebo odemčení vozidla.

Pro odposlouchávání a zprostředkovávání informací cloudové aplikaci bylo navrženo a vytvořeno zařízení na bázi ESP32, které je umístěné ve vozidle a v pravidelných intervalech zasílá data ze senzorů skrze bezdrátové sítě (GSM a WiFi). V práci bylo také důkladně řešeno napájení tohoto zařízení, které je zajištěno z autobaterie a je tedy nutné dbát na zamezení nadměrnému vybíjení (zejména u zaparkovaného automobilu). Zařízení bylo doplněno také o GPS modul pro rozšíření monitorování o geografickou polohu a s tím spojené možnosti při analýze.

Data zajištěná a zaslaná zařízením jsou ukládána do databáze a je možné je zobrazit pomocí webové aplikace. Ta je uzpůsobena zejména pro rozhraní mobilního telefonu a je v ní možné sledovat jak aktuální, tak historii zaznamenaných dat.

Vývoj zařízení (zejména hardwaru) byl značně ovlivněn pandemickými opatřeními probíhajícími v době vývoje, jež omezila možnosti nákupu komponentů a přístup nástrojům při výrobě.

Vzniklé řešení je možné dále rozvíjet. Bez úpravy hardwaru je možné (vzdálenou aktualizací softwaru) monitorování rozšířit o nově nalezené senzory nebo přidat novou sekvenci příkazů k ovládání subsystémů automobilu. Zajímavým směrem by mohlo být také rozšíření o připojení k další sběrnici vozidla, což by však vyžadovalo přidání dalšího CAN kontroléru.

Literatura

[1] *Limitations of LoRaWAN* [online https://www.thethingsnetwork.org/docs/lorawan/limitations.html]. 2019 [cit. 2020-12-20].

[2] Chris Valasek, C. M. *CAN Message Injection - OG Dynamite Edition* [online]. 2017 [cit. 2020-04-18]. Dostupné z: https://packetstormsecurity.com/files/142306/CAN-Message-Injection-OG-Dynamite-Edition.html.

[3] Currie, R. *Hacking the CAN Bus: Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering* [https://www.giac.org/paper/gcia/9927/hacking-bus-basic-manipulation-modern-automobile-bus-reverse-engineering/133228]. 2017 [cit. 2021-04-20].

[4] Espressif Systems. *ESP32 Series Datasheet – Version 3.4* [online]. 2020 [cit. 2021-01-06]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[5] Horpatzká, M. *Systémy sériové diagnostiky motorových vozidel*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav automobilního a dopravního inženýrství. Dostupné z: http://hdl.handle.net/11012/60135.

[6] Jantolák. [https://www.autodiagnostika.jantolak.sk/?%E8l%E1nok-5-%FAvod-do-diagnostiky-obd-obd2-eobd-zariadenia,259]. 2012 [cit. 2020-12-20].

[7] LastMinuteEngineers.com. *Insight Into ESP32 Sleep Modes and Their Power Consumption* [online]. 2021 [cit. 2021-01-11]. Dostupné z: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/.

[8] Michal, OK1WMR. *Výroba plošných spojů nažehlovací metodou* [online]. 2020 [cit. 2020-04-26]. Dostupné z: https://ok1kvk.cz/clanek/2008/vyroba-plosnych-spoju/.

[9] Microchip. *MCP2515 Stand-Alone CAN Controller with SPI Interface Data Sheet* [online]. 2019 [cit. 2021-01-06]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf.

[10] Miller, C. a Valasek, C. *Adventures in Automotive Networks and Control Units* [http://www.carmelowalsh.com/wp-content/uploads/2014/05/Car_Hacking_Hacktivity_2013_whitepaper.pdf]. 2014 [cit. 2020-12-20].

[11] MongoDB. *MongoDB vs MySQL Differences*. 2021 [cit. 2020-04-27]. Dostupné z: https://www.mongodb.com/compare/mongodb-mysql.

[12] Open vehicle monitoring system. *OVMS Documentation* [online]. 2020 [cit. 2021-01-08]. Dostupné z: https://github.com/openvehicles/Open-Vehicle-Monitoring-System-3.

[13] Robert BOSCH GmbH, D.-. S. . *CAN sSpecification 2.0* [online]. 1991 [cit. 2020-12-18]. Dostupné z: https://www.kvaser.com/software/7330130980914/V1/can2spec.pdf.

[14] Robert Walters. *Time Series Data and MongoDB* [https://www.mongodb.com/blog/post/time-series-data-and-mongodb-part-2-schema-design-best-practices]. 2019 [cit. 2020-04-27].

[15] SIMCom Wireless Solutions Co., L. *SIMCom Product Catalogues 2019* [online]. 2019 [cit. 2021-01-06]. Dostupné z: https://simcom.ee/documents/SIMCom%20Product%20Catalogues_2019.pdf.

[16] Sparkfun: TONI_K. [https://learn.sparkfun.com/tutorials/getting-started-with-obd-ii/obd-ii-protocols]. [cit. 2020-15-04].

[17] VOLKSWAGEN AG, W. *SSP238 - Data Exchange On The CAN Bus I* [online]. [cit. 2020-12-19]. Dostupné z: http://www.volkspage.net/technik/ssp/ssp/SSP_238.pdf.

[18] VOLKSWAGEN AG, W. *SSP307 - The Touran Electrical system (Design and function)* [online]. [cit. 2020-12-30]. Dostupné z: http://www.volkspage.net/technik/ssp/ssp/SSP_238.pdf.

[19] VOLKSWAGEN AG, W. *SSP306 – Touran* [online]. 2003 [cit. 2020-12-19]. Dostupné z: https://www.volkswagenclub.net/manual_download.php?id=155.

[20] Wikipedia contributors. *Bluetooth — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Bluetooth&oldid=996313492]. 2020 [cit. 2020-12-20].

[21] Wikipedia contributors. *CAN bus — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=CAN_bus&oldid=994118908]. 2020 [cit. 2020-12-28].

[22] Wikipedia contributors. *On-board diagnostics — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&oldid=993657894]. 2020 [cit. 2020-12-19].

[23] Wikipedia contributors. *Raspberry Pi — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=999550702]. 2021 [cit. 2021-01-11].

[24] Xmarton. [https://www.xmarton.com/funkce/]. 2020 [cit. 2020-12-30].

Příloha A

Obsah přiloženého paměťového média

```
root
├── doc
├── device_firmware
├── backend
├── frontend
├── plosny_spoj
├── xdraho11-DIP.pdf
└── README.md
```