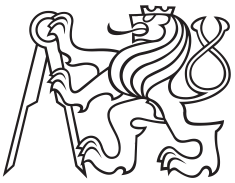
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

CANgui - Nástroj pro diagnostiku CAN sběrnice

Filip Machulda

Vedoucí: Ing. Marek Szeles
Studijní program: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Machulda** Jméno: **Filip** Osobní číslo: **483843**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

CANgui - Nástroj pro diagnostiku CAN sběrnice

Název bakalářské práce anglicky:

CANgui - Tool for CAN bus diagnostics

Pokyny pro vypracování:

- Definujte a popište CAN sběrnici/protokolu
- Proveďte sběr požadavků a případů užití pro tým eForce
- Prozkoumejte stávajících možných řešení a jejich nedostatky
- Vytvořte datový model řešení
- Navrhněte architekturu aplikace
- Vytvořte návrh uživatelského rozhraní aplikace
- Implementujte aplikaci a popište postup nasazení a použití aplikace
- Evaluujte vytvořené řešení na základě vytyčených cílů

Seznam doporučené literatury:

1. CORRIGAN, Steve. Introduction to the controller area network (CAN). Application Report SLOA101 (2002): 1-17, 2002.
2. MASSE, Mark. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly Media, Inc., 2011.
3. FETTE, Ian, a Alexey Melnikov. The websocket protocol, 2011.
4. AXELSON, Jan. Serial port complete: COM ports, USB virtual COM ports, and ports for embedded systems. Lakeview Research, 2007.
5. GARRETT, Jesse James. The elements of user experience: user-centered design for the web and beyond. Pearson Education, 2010.
6. JEŘÁBEK, Martin. Open-source a Open-hardware podpora pro CAN FD, 2019. Magisterská práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra řídicí techniky.
7. FITZ, Karel. Návrh rozšíření existujícího řadiče CAN o standard FD, 2017. Magisterská práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra mikroelektroniky.
8. DO, Duc Huy. Automatizovaný nástroj pro mapování zpráv sběrnice CAN, 2020. Bakalářská práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra počítačových systémů.
9. OLEKŠÁK, Matúš. Aplikace pro analýzu průmyslových sběrnic včetně hardwarového trigeru, 2020. Bakalářská práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra číslicového návrhu.
10. VANĚK, Jiří. CAN - USB převodník pro integraci do operačního systému GNU/Linux, 2012. Diplomová práce. ČVUT v Praze, Fakulta elektrotechnická, Katedra kybernetiky.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Marek Szeles, eForce FEE Prague Formula, FEL ČVUT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **15.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Marek Szeles podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování

Děkuji vedoucímu práce, panu inženýru Marku Szelesovi za vstřícnost a nápomoc během tvorby této práce. Dále bych rád poděkoval celému týmu eForce FEE Prague Formula za cenné zkušenosti, které jsem během působení v týmu získal a za důležitou zpětnou vazbu během tvorby této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 21. května 2021

.....

Abstrakt

CANgui je webový nástroj pro analýzu dat na Controller Area Network (CAN) sběrnici. Nástroj je primárně určen pro vývojáře elektronických řídících jednotek (Electronic Control Unit, ECU) v automobilovém průmyslu, ale lze využít v jakémkoliv poli práce s CAN sběrnicí.

Tento nástroj umožňuje uživatelům zobrazit si jednotlivé zprávy ze sběrnice (živě nebo ze záznamu), sledovat vývoj hodnot pomocí grafů a navíc si uživatelé mohou definovat i vlastní obrazovky, pro snazší monitoring dané ECU.

Pokud bude nástroj připojen ke sběrnici pomocí nástroje Ocarina IV, nástroj uživateli umožní simulovat libovolnou ECU (z aktuálně použitého modelu auta) a posílat tak zprávy i zpět na sběrnici.

Tato práce popisuje vznik nástroje CANgui, od rešerše alternativ, přes sběr požadavků, návrh a implementaci až po evaluaci vzniklého řešení.

Klíčová slova: Webová aplikace, CAN, Uživatelské rozhraní, Datová analýza

Vedoucí: Ing. Marek Szeles

Abstract

CANgui is a web tool designed for Controller Area Network (CAN) bus data analysis. The tool is primarily designed for Electronic Control Unit (ECU) developers in the automotive industry but can be used in any field of work that utilizes a CAN bus.

The tool allows the user to display individual messages from the bus (either live or from a replay file), monitor field values with charts and also allows the user to define custom windows which allow for easy monitoring of ECUs.

In case that the tool is connected to a live bus via Ocarina IV, users can simulate any ECU (from the currently used message model) and send messages back to the bus.

This thesis describes the development process of CANgui, from research of existing solutions, requirements elicitation, design, implementation and evaluation of the result.

Keywords: Web app, CAN, User interface, Data analysis

Title translation: CANgui - Tool for CAN bus diagnostics

Obsah

Obrázky

Tabulky

Úvod

Motivace

Na Fakultě elektrotechnické ČVUT od roku 2010 existuje tým eForce FEE Prague Formula, kde se desítky studentů různých oborů podílejí na vzniku unikátního projektu konstrukce elektrických závodních formulí. Každoročně tým vyrábí dva závodní monoposty (elektrický a autonomní). Oba monoposty z roku 2020 lze spolu s členy týmu vidět na obrázku 1. V rámci tohoto projektu musejí studenti čelit spoustě reálných problémů, kterým jinak čelí špičkoví inženýři v automobilovém průmyslu a vrcholovém motorsportu.

Obrázek 1: Tým eForce FEE Prague Formula [1]

Jedním z těchto problémů je i diagnostika problémů a analýza dat na datových sběrnicích. Vývojáři v automobilovém průmyslu potřebují jednoduchý způsob monitoringu elektronických řídících jednotek – Electronic Control Unit (ECU)[1], aby se ujistili, že vše funguje, jak má. Takovýto nástroj je pro vývojáře důležitý jak při vývoji, tak i při provozu, jako způsob diagnostiky problémů s danou ECU nebo sběrnicí. Toto téma se týká i týmu eForce, jelikož v jejich vozidlech jsou použity CAN sběrnice (viz kapitola 1), jako způsob přenosu informací, a to nejen pro řízení vozidla.

Pro tyto účely v dnešní době existuje řada komerčních řešení. Na základě analýzy těchto řešení (viz kapitola 3) jsme ale zjistili, že tato řešení často přehlížejí důležitost jednoduchosti ovládání a jsou omezená i v jiných aspektech. Tyto nedostatky nás motivovaly k vývoji vlastního řešení, nazvaného CANgui. Jedná se o webový nástroj, který bude využívat další nástroje a zařízení z dílny týmu eForce a zkompletuje tak balíček týkající se práce s CAN sběrnicí. Naším cílem je vytvořit robustní řešení, jež se stane součástí vývojového cyklu vývojářů ECU a datových analytiků, nejen v týmu eForce FEE Prague Formula.

Cíle

Hlavní cíle této práce jsou čtyři. Prvním cílem práce je popsat teorii CAN sběrnice, jelikož je to cílové médium pro přenos dat ve vozidlech týmu eForce. Druhý hlavní cíl analýzy má dva dílčí cíle. První dílčí cíl je sběr požadavků a případů užití. Druhým dílčím cílem je provedení analýzy existujících řešení, včetně doporučení dalšího postupu. Třetí hlavní cíl práce je popsat implementaci technického řešení zvoleného problému na základě doporučení vyplývajících z analýzy. Čtvrtý hlavní cíl je evaluace řešení, která proběhne pomocí metod kvalitativního a kvantitativního uživatelského testování.

[1] ECU jsou elektronická zařízení, obstarávající nějakou konkrétní úlohu [2][3]. V případě automobilů se může jednat například o monitorování připnutí pásů nebo ovládání stěračů.

Část I

Teorie

Kapitola 1
Controller Area Network sběrnice

Controller Area Network (CAN) sběrnice slouží jako způsob komunikace pro různá elektronická zařízení. Použití CAN sběrnice je nejvýznamnější v automobilovém průmyslu, kde se využívá již desítky let pro komunikaci mezi ECU (v kontextu CAN sběrnice je také používán termín uzel) [4]. Kromě toho je využití CAN sběrnic významné i pro lékařské vybavení nebo například v poli průmyslové automatizace [5][6].

V posledních letech se začíná prosazovat modernizovaná sběrnice Controller Area Network Flexible Data-Rate (CAN FD) [7][8]; v této práci se však zaměřujeme na sběrnici CAN 2.0, standardizovanou v normách ISO 11898-1 [9] a ISO 11898-2 [10]. Tyto normy popisují první dvě vrstvy modelu ISO/OSI, to znamená fyzickou vrstvu, jež se zaměřuje na komunikační médium, včetně způsobu kódování a synchronizaci zařízení a vrstvu datového spojení, která se mimo jiné zaměřuje na detekci chyb a serializaci dat [11][12]. Tento typ CAN sběrnice jsme zvolili, jelikož ho používá tým eForce a máme k němu dostupný CAN-USB převodník Ocarina IV (viz sekce 8.2.1). Teoreticky je možné nástroj CANgui rozšířit, aby podporoval i jiné převodníky a tím pádem podporoval i jiné druhy sběrnic, to je ale mimo rozsah této práce.

1.1 Fyzická vrstva

Definice fyzické vrstvy CAN sběrnice se v této práci zaměří pouze na klasický vysokorychlostní CAN (rychlost až 1 Mb/s) podle normy ISO 11898-2 [10].

1.1.1 Přenos informace

CAN sběrnice používá pro přenos informace kódování NRZ (z anglického Non-return-to-zero) a má dva stavy: *dominantní* (aktivní), který odpovídá hodnotě logické 0 a *recesivní* (pasivní) stav, který odpovídá hodnotě logické 1. K propojení uzlů na sběrnici se používá kroucená dvojlinka, jejíž vodiče jsou pojmenované CAN high a CAN low, též známé jako CANH a CANL, jež jsou zakončené 120 Ω rezistory na obou koncích [10].

Pro odeslání dominantního bitu se aktivně nastaví hodnota napětí na vodiči CANH na 3,5 V a CANL na hodnotu 1,5 V, čímž vzniká rozdílové napětí ($U_{roz} = U_{CANH} - U_{CANL}$) 2 V. Recesivní stav je vyznačený napětím 2,5 V na obou vodičích pomocí pull-up a pull-down rezistorů. Přijímače typicky považují rozdílové napětí do 0,5 V stále za recesivní stav. Přenos dominantního stavu vždy zvítězí nad přenosem stavu recesivního (recesivní bit se na sběrnici nepřenese) [10].

Této vlastnosti se využívá například u určení priority při odesílání zpráv na sběrnici – uzly co vysílají taktéž sledují aktuální stav sběrnice a pokud odeslaná hodnota neodpovídá aktuálnímu stavu, tak uzel přestane vysílat a pokusí se zprávu odeslat později.

1.1.2 Synchronizace

CAN sběrnice nepoužívá hodinový signál pro synchronizaci uzlů. Místo toho všechny uzly znají bitrate sběrnice a jednotlivé uzly jsou synchronizované při prvním přechodu z recesivního do dominantního stavu (první bit rámce zprávy). Uzly jsou i dále udržované v synchronizaci při následujících přechodech z recesivního do dominantního stavu [13].

Pro zajištění dostatečného počtu přechodů implementuje vrstva datového spojení takzvaný bit stuffing. Jedná se o mechanismus, který doplňuje do přenášeného rámce bity opačného stavu, pokud bylo již přeneseno 5 bitů jednoho stavu[1]. Vrstva datového spojení příjemce tyto dodatečné bity ze zprávy před dalším zpracováním odstraní [4].

1.2 Vrstva datového spojení

Definice vrstvy datového spojení CAN sběrnice se v této práci odvíjí od normy ISO 11898-1 [9].

1.2.1 Formát datových rámců

Podle novějšího standardu CAN sběrnice (CAN 2.0B) existují dva druhy datových rámců (data frame) neboli zpráv. První, takzvaný základní (base) stanovuje délku ID zprávy 11 bitů a je zpětně kompatibilní i se staršími standardy CAN sběrnice. Délka ID zprávy 11 bitů umožňuje až 2048 unikátních typů zpráv (základního druhu) na jedné sběrnici. Druhý, novější, takzvaně rozšířený (extended) druh stanovuje délku ID 29 bitů, což umožňuje použít až necelých 537 milionů různých zpráv. Rozšířené rámce jsou podporované až od standardu CAN 2.0B [4][14][15][16].

Oba druhy rámců lze vzájemně používat na stejné fyzické sběrnici, pokud jednotlivé uzly podporují standard CAN 2.0B. Podle definice lze formát datového rámce rozdělit do následujících kategorií [14]:

- **SOF (Start of Frame)**
  Značí začátek přenosu rámce
- **Arbitrážní pole**
  Určuje prioritu zprávy – ID zprávy, RTR (Remote Transmission Request) – určuje zda je rámec datový nebo takzvaně vzdálený (pro naše potřeby se budeme zabývat pouze datovými rámci)
- **Řídící pole**
  Určuje typ rámce[2], určuje délku datového pole

[1] Toto platí pouze mezi SOF a CRC oddělovačem viz sekce 1.2.1

[2] Pouze u základního rámce spadá do řídícího pole, u rozšířeného rámce je tento bit považován ještě za součást arbitrážního pole

- **Datové pole**
  Data zprávy
- **CRC Pole**
  Kontrolní součet
- **ACK Pole**
  Potvrzení o správném přijetí rámce
- **EOF (End of Frame)**
  Značí konec přenosu rámce
- **IFS (Interframe Space)**
  Místo před dalším možným rámcem (minimálně 3 recesivní bity)

Základní rámec

Formát základního rámce podle standardu CAN 2.0B lze vidět v tabulce 1.1. Oproti staršímu standardu CAN 2.0A se liší pouze v názvu bitu IDE, který se původně jmenoval r1 a byl rezervovaný pro budoucí použití. Jeho hodnota zůstala pro základní rámec stejná (dominantní – logická 0), obecně ale indikuje zda se aktuálně posílá základní nebo rozšířený rámec (recesivní – logická 1) [14][15]. Dle vlastností fyzické vrstvy CAN sběrnice tedy víme, že prioritu mají rámce základní.

Název pole	**Délka (bit)**	**Popis**
SOF	1	Naznačuje začátek přenosu – dominantní (0)
ID	11	Unikátní identifikátor zprávy
RTR	1	Pro datové rámce dominantní (0)
IDE	1	Určuje typ datového rámce – dominantní (0)
r0	1	Rezervovaný bit – dominantní (0)
DLC	4	Určuje délku datové části v byte (pouze 0 až 8)
Data	0-64	Samotná data zprávy
CRC	15 + 1	15 bitový cyklický redundantní součet (CRC-15-CAN) + 1 bit oddělovač – recesivní (1)
ACK	1 + 1	1 bit o potvrzení přijetí rámce – recesivní (1) Příjemce přepíše hodnotu na dominantní + 1 bit oddělovač – recesivní (1)
EOF	7	Naznačuje konec přenosu – 7x recesivní (1)

Tabulka 1.1: Formát a popis polí základního datového rámce

Rozšířený

Rozšířený datový rámec byl přidán ve standardu CAN 2.0B. Oproti základnímu rámci se liší pouze v délce identifikátoru zprávy, což umožňuje mít více typů zpráv na jedné sběrnici (téměř 537 milionů unikátních typů zpráv) [14][15]. Formát rozšířeného rámce lze vidět v tabulce 1.2.

Název pole	Délka (bit)	Popis
SOF	1	Naznačuje začátek přenosu – dominantní (0)
ID A	11	První část unikátního identifikátoru zprávy
SRR	1	Náhrada za RTR – recesivní (1)
IDE	1	Určuje typ datového rámce – recesivní (1)
ID B	18	Druhá část unikátního identifikátoru zprávy
RTR	1	Pro datové rámce dominantní (0)
r1, r0	2	Rezervované bity – dominantní (0)
DLC	4	Určuje délku datové části v byte (pouze 0 až 8)
Data	0-64	Samotná data zprávy
CRC	15 + 1	15 bitový cyklický redundantní součet (CRC-15-CAN) + 1 bit oddělovač – recesivní (1)
ACK	1 + 1	1 bit o potvrzení přijetí rámce – recesivní (1) Příjemce přepíše hodnotu na dominantní + 1 bit oddělovač – recesivní (1)
EOF	7	Naznačuje konec přenosu – 7x recesivní (1)

Tabulka 1.2: Formát a popis polí rozšířeného datového rámce

1.2.2 Kontrola chyb

Kontrola chyb a uzavření chyb (odpojení vadných uzlů ze sítě) jsou klíčové vlastnosti, jež dělají CAN sběrnici robustní. Na CAN sběrnici existují dvě úrovně chyb: chyby na úrovni zpráv (message level error) a chyby na úrovni bitů (bit level error) [4][17].

- Chyby na úrovni zpráv
  - **CRC error**
    Odesílatel vkládá do rámce 15 bitový redundantní součet (CRC-15-CAN). Všechny ostatní uzly si spočítají vlastní redundantní součet a porovnají ho s přijatým součtem. Pokud se hodnoty liší, uzel pošle chybový rámec a přijatý rámec zahodí.
  - **ACK error**
    Odesílatel nastaví hodnotu ACK pole na recesivní. Hodnota zůstane recesivní pouze pokud žádný jiný uzel na sběrnici rámec nepřijal. Jelikož tuto chybu dokáže detekovat vysílací uzel sám, neposílají se žádné chybové rámce.
  - **Form error**
    Tento typ chyby nastane pokud nějaký uzel detekuje dominantní hodnotu na sběrnici v případě, že má být dle definice stav sběrnice recesivní (to znamená při EOF, IFS, CRC oddělovači a nebo ACK oddělovači).

- Chyby na úrovni bitů
  - **Stuffing error**
    Uzel detekuje mezi SOF a ACK oddělovači 6 bitů stejného stavu, což je dle definice nemožné.
  - **Bit error**
    Odesílatel sleduje stav sběrnice. Pokud se stav sběrnice liší oproti poslanému bitu, odešle chybový rámec. Pro tuto chybu existují výjimky – arbitrážní pole (jiný uzel může posílat zprávu s vyšší prioritou) a ACK slot, kde naopak odesílatel očekává změnu stavu.

Část II

Analýza

Kapitola 2
Sběr požadavků a případů použití

Moderní vozidla (nejen elektrického pohonu) zpravidla obsahují elektronické řídící jednotky – ECU, které mezi sebou komunikují prostředkem sběrnice [18]. Je tedy nutné, aby vývojáři těchto jednotek měli nástroj pro odlaďování chyb, které mohou nastat při vývoji (v izolovaném prostředí), při integraci, či až v provozu (například pod zátěží). Kromě nástroje pro odlaďování chyb, je také nutné mít nástroj pro sledování stavu těchto sběrnic a všech ECU ve vozidle, jelikož nám to umožňuje provést diagnostiku problémů s vozidlem. Navíc to také umožní provádět zpětnou analýzu, což se zejména hodí pro prostředí motosportu. Tým eForce FEE Prague Formula, který se zabývá vývojem vlastních závodních formulí elektrického pohonu, používá jako způsob komunikace mezi ECU ve vozidlech CAN sběrnice. I pro tento tým je tedy kritické mít takovou sadu nástrojů.

Na základě potřeb a zkušeností členů týmu eForce byl proveden sběr požadavků a případů použití. Sběr proběhl řadou neformálních rozhovorů, primárně s vedoucím elektrické sekce a vybranými členy týmu, kteří se zabývají analýzou dat. Výsledkem těchto konverzací jsou následující požadavky na řešení. Tyto požadavky budou dále použity pro vyhodnocení existujících řešení (viz sekce 3.4). Z tohoto vyhodnocení vzejde zda tyto potřeby dokáže některé z řešení dostatečně uspokojit, či bude nezbytné vyvinout si podle těchto požadavků vlastní řešení.

2.1 Požadavky

V softwarovém inženýrství lze požadavky typicky rozdělit do dvou kategorií. První kategorií jsou takzvané funkční požadavky (functional requirements). Funkční požadavky zadávají konkrétní požadovanou funkcionalitu aplikace. Druhou kategorií jsou takzvané nefunkční požadavky (non-functional requirements), které určují vlastnosti, které by aplikace měla splňovat – typicky se týkají dostupnosti a výkonnosti aplikace.

Při sběru požadavků je taktéž důležité si jednotlivé požadavky ohodnotit dle jejich významnosti. Pro aplikaci CANgui jsme zvolili tři úrovně důležitosti:

- Kritické – Bez těchto požadavků pro nás řešení nemá hodnotu
- Důležité – Bez těchto požadavků lze řešení používat, výrazně bude ale zhoršená funkcionalita
- Užitečné – Bez těchto požadavků se obejdeme

2.1.1 Funkční požadavky

1. Zobrazit provoz sběrnice – **Kritické**
   - Log všech zpráv
   - Poslední přijatá zpráva s daným ID
2. Odesílat zprávy na sběrnici – **Kritické**
   - V jednoduché podobě (formulář podle externího modelu)
3. Namapovat obsah zpráv na externí model – **Důležité**
   - Bude použit formát CANdb JSON 2.0 (sekce 5.2.2)
4. Podporovat více sběrnic najednou – **Důležité**
5. Uložit přijaté zprávy do souboru – **Důležité**
   - ASCII Logging Files (.ASC) nebo Measurement Data Format (.MDF)
6. Přehrát uložené zprávy ze záznamu – **Důležité**
   - Podle použitého formátu na ukládání
7. Vytvořit vlastní okna – **Důležité**
   - Okna se mohou například specializovat na konkrétní ECU
   - Podpora základních komponent jako sloupcový graf pro aktuální hodnoty
   - Definice pomocí custom JSON modelu (sekce 5.3)
8. Zobrazit vývoj hodnot jednotlivých polí zpráv v čase – **Důležité**
   - Čárový graf
9. Zobrazit statistiky sběrnice a jednotek – **Užitečné**
   - Například využití sběrnice, reálná perioda zpráv, jitter zpráv
10. Sdílet data ze sběrnice online – **Užitečné**

2.1.2 Nefunkční požadavky

1. Multiplatformní – **Kritické**
   - Podpora alespoň platforem Windows a Linux
2. Jednoduchý přístup k aplikaci – **Kritické**
3. Levné licencování – **Důležité**
   - Nutné kvůli vysokému počtu vývojářů
4. Jednoduché k použití – **Důležité**
5. Použitelnost aplikace i offline – **Důležité**

2.2 Případy použití

Případy užití typicky vycházejí z předpokladů o tom, kdo bude daný nástroj používat. Kromě toho jsou případy užití úzce svázány s funkčními požadavky (nějaká konkrétní funkcionalita umožňuje daný případ užití). V našem případě se zaměřujeme na vývojáře ECU a datové analytiky v automobilovém průmyslu, proto nám ze sběru vzešly následující případy použití:

1. Monitorování živého provozu na sběrnici
2. Debugging ECU vývojářem
3. Diagnostika problémů na sběrnici
4. Vzdálený debugging vozidla
5. Zpětná analýza dat z jízdy

Kapitola 3
Stávající řešení

Na trhu již existují aplikace, které se zabývají diagnostikou CAN sběrnice. Pro porovnání jsou uvedeny tři nejrozšířenější aplikace, co se vlastností a použití týče. Jednotlivá řešení jsou porovnána na základě sběru požadavků v předchozí kapitole.

3.1 CANrunner

CANrunner je desktopová aplikace od finské společnosti Wapice Ltd [19]. Její vlastnosti splňují spoustu požadavků definované týmem eForce (viz kapitola 2), bohužel ale některé důležité body (například vizualizace dat pomocí grafických prvků) nesplňuje vůbec, nebo pouze částečně.

- Výhody
  - Analýza celé sběrnice
  - Freeware
  - Podpora pro Windows i Linux
  - Podpora více typu hardware
  - Logování zpráv
  - Zpětné přehrání ze záznamu
  - Posílání zpráv na sběrnici
  - Mapování na externí model
  - Sdílení dat po ethernetu
  - Podpora CAN FD
- Nevýhody
  - Uživatelské rozhraní není příliš intuitivní
  - V aplikaci nejsou žádné grafické prvky pro vizualizaci dat (například grafy)
  - Mapování na externí model je pro obyčejné uživatele složité
  - Sdílení dat je v základě možné pouze po lokální síti (peer-to-peer)

3.2 CANalyzer

CANalyzer je desktopová aplikace od německé společnosti Vector Informatik GmbH, která umožňuje uživateli simulovat a analyzovat jeden uzel na CAN sběrnici [20]. Tato aplikace je velice obsáhlá, co se vlastností simulace uzlu týče, tím se ale také stává její použití pro začátečníka složité.

Navíc naším cílem je možnost analýzy všech uzlů na naší CAN sběrnici s pomocí vlastních grafických obrazovek, což s CANalyzer není možné.

- Výhody
  - Pokročilá simulace uzlu pomocí vizuálního programování
  - Logování zpráv (Export i pro MATLAB)
  - Zpětné přehrání ze záznamu
  - Posílání zpráv na sběrnici
  - Mapování na externí model (proprietární formát DBC [21])
  - Podpora CAN FD
- Nevýhody
  - Uživatelské rozhraní není příliš intuitivní
  - Strmá křivka učení
  - Podpora pouze pro Windows
  - Licence vázaná na počítač
  - Drahá licence
  - Simulace pouze jednoho uzlu (ECU)
  - Vizualizace dat omezena na liniové grafy
  - Podpora pouze proprietárního hardware

3.3 CANoe

CANoe je desktopová aplikace od německé společnosti Vector Informatik GmbH [22]. Dalo by se tvrdit, že se jedná o rozsáhlejší verzi aplikace CANalyzer, zmíněnou dříve. Co se týče vlastností aplikace, tak je to pro náš případ použití asi nejvhodnější kandidát. Bohužel s sebou ale CANoe také nese nevýhody (primárně chybějící podpora pro jiné platformy než Windows a cena zřízení).

- Výhody
  - Analýza celé sběrnice
  - Možnost tvorby vlastních grafických oken
  - Pokročilá simulace uzlů pomocí vizuálního programování
  - Logování zpráv (Export i pro MATLAB)
  - Zpětné přehrání ze záznamu
  - Posílání zpráv na sběrnici
  - Mapování na externí model (proprietární formát DBC [21])
  - Podpora CAN FD
- Nevýhody
  - Uživatelské rozhraní není příliš intuitivní
  - Strmá křivka učení
  - Podpora pouze pro Windows
  - Licence vázaná na počítač
  - Drahá licence
  - Podpora pouze proprietárního hardware

3.4 Shrnutí

Pro vyhodnocení nejvhodnějšího kandidáta jsme porovnali vlastnosti všech vybraných řešení proti požadavkům a případům užití týmu eForce (viz kapitola 2). Jednotlivé vlastnosti jsou ohodnocené hodnotami z uzavřeného intervalu $[0, 1]$ (lze vyjádřit i procentuálně) podle toho, jak řešení daný požadavek či případ použití splňuje. Jednotlivým požadavkům a případům užití jsme přiřadili váhu (maximální bodové ohodnocení) podle jejich důležitosti (viz váhy kritérií v tabulce 3.1).

Kritérium	Důležitost	Váha	CANrunner	CANalyzer	CANoe	CANgui
F 1	Kritické	10	0,5	1	1	1
F 2	Kritické	10	0,5	1	1	1
F 3	Důležité	7	0,5	1	1	1
F 4	Důležité	7	1	0	1	1
F 5	Důležité	7	0,5	1	1	1
F 6	Důležité	7	1	1	1	1
F 7	Důležité	7	0	0	1	1
F 8	Důležité	7	0	1	1	1
F 9	Užitečné	5	0	1	1	1
F 10	Užitečné	5	0,5	0	0	1
Mezisoučet F			**33,5**	**53**	**67**	**72**
N 1	Kritické	10	0,7	0	0	1
N 2	Kritické	10	0,7	0,7	0,7	1
N 3	Důležité	7	1	0	0	1
N 4	Důležité	7	0,3	0,4	0,4	1
N 5	Důležité	7	1	1	1	1
Mezisoučet N			**30,1**	**16,8**	**16,8**	**41**
UC 1		5	1	1	1	1
UC 2		5	1	1	1	1
UC 3		5	1	1	1	1
UC 4		5	1	0	0	1
UC 5		5	1	1	1	1
Mezisoučet UC			**25**	**20**	**20**	**25**
Celkový součet všech kategorií			**88,6**	**89,8**	**103,8**	**138**

F = Funkční požadavek (Sekce 2.1.1)
N = Nefunkční požadavek (Sekce 2.1.2)
UC = Případ použití (Sekce 2.2)

Tabulka 3.1: Vyhodnocení řešení

Podle tabulky 3.1 vyplývá, že nejvhodnější kandidát (ze skupiny existujících řešení) pro potřeby týmu eForce by byla aplikace CANoe od společnosti Vector Informatik GmbH. Tu v současné době tým eForce využívá, bohužel má ale omezený počet licencí (které se vážou na konkrétní počítač) a vyžaduje proprietární hardware pro komunikaci s CAN sběrnicí, takže mohou danou aplikaci používat jen vybraní jedinci. Navíc aplikaci chybí podpora pro jiné platformy než Windows a křivka učení s touto aplikací je velmi strmá, což jde přímo proti zadaným požadavkům (viz sekce 2.1).

Vývoj vlastního řešení je náročnější, přidaná hodnota takového řešení ale týmu stojí za tuto časovou investici. V případě neuspokojivého řešení navíc tým může nadále používat současné řešení CANoe, takže vlastní vývoj představuje pro fungování týmu minimální celkové riziko. Z těchto důvodů jsme přišli s rozhodnutím vytvořit vlastní řešení, v podobě CANgui, které bude plně splňovat všechny zadané požadavky a umožní nám dále přidávat funkcionalitu na základě nových potřeb týmu.

Část III

Návrh

Kapitola 4
Použité technologie

Na základě sběru požadavků a případů užití (viz kapitola 2) vyplývá, že aplikace CANgui bude webovou aplikací. Aplikaci tak lze rozdělit do dvou částí (back-end a front-end), jež spolu budou komunikovat. Je tedy důležité pro tyto části zvolit vhodný balík technologií.

Volba konkrétních technologií byla založena primárně na předchozích zkušenostech a znalostech. Další kritéria (založená na nefunkčních požadavcích aplikace viz sekce 2.1), která hrála při výběru roli, byla:

- Multiplatformnost řešení
- Rychlý a přímý vývoj
- Jednoduchá udržovatelnost
- Jednoduché nasazení
- Jednoduchá distribuce aplikace

4.1 Back-end

Technologie použité pro back-end aplikace CANgui byly zvoleny tak, aby bylo možné aplikaci používat i v offline prostředí (to znamená i bez centrálního serveru, na vlastním zařízení). Pro delší část sezóny týmu eForce je tento požadavek zanedbatelný, během testování a na závodech je ale kritické mít tento nástroj dostupný i bez připojení k centrálnímu serveru. Z těchto důvodů jsme zvolili jako platformu pro vývoj CANgui Javu [23], jež je multiplatformní a nástroj lze pak distribuovat pomocí jednoho spustitelného souboru.

4.1.1 Java SE 11

Java je objektově orientovaný programovací jazyk a také platforma, původně vyvinuta společností Sun Microsystems, nyní vlastněna společností Oracle Corporation [23][24].

Jádrem platformy Java je takzvaná Java Virtual Machine (JVM), která vykonává zkompilovaný Java bytecode a je hlavním důvodem multiplatformnosti Javy [23].

Při výběru verze platformy proběhlo rozhodnutí mezi Java SE 8 a 11, jelikož to jsou takzvané Long Time Support (LTS) verze. To znamená, že mají zajištěnou podporu po dobu několika let a jsou to většinou cílové verze pro vývojáře knihoven a frameworků [25].

Java SE 11 byla vybraná, jelikož je to nejnovější LTS verze Java SE, a Java SE 8 je v dnešní době už zastaralá a obsahuje nemoderní prvky [26]. Navíc její podpora u nových knihoven nemusí být zajištěna (pro většinu případů by s tím problém být neměl, ale v tomto případě je lepší se takovým problémům přímo vyvarovat).

4.1.2 Maven

Maven je nástroj vyvinut společností Apache Software Foundation [27]. Tento nástroj je primárně používán jako nástroj pro udržování závislostí a zajištění automatického sestavovacího (build) procesu Java projektů (i když lze použít i pro ostatní platformy a nabízí mnohem více funkcí).

Srdcem Maven projektů je takzvaný Project Object Model (POM), což je XML soubor, který obsahuje veškerou konfiguraci daného projektu, včetně závislostí a pluginů pro sestavení [28].

Kromě nástroje Maven lze použít i (již zastaralý) Apache Ant nebo Gradle, jež v současné době nabírá na popularitě. Rozhodnutí pro použití Mavenu bylo pouze na základě předchozích zkušeností (zvládne vše, co tento projekt požaduje) a chybějících zkušeností s nástrojem Gradle.

4.1.3 Spring Boot

Spring Boot je komplexní framework pro platformu Java od společnosti Pivotal Software, která je nyní součástí VMware Inc [29]. Tento framework je zejména využíván na tvorbu webových služeb a serverů, lze jej ale například použít i pro vývoj distribuovaných standalone klientů [30].

Spring Boot má v základu výchozí konfiguraci všech prvků a tím pádem umožňuje rychlý vývoj. V případě, že je nutné zajistit jiné, nevýchozí chování, lze konfiguraci přepsat. Spring Boot je navíc velice modulární. Podle konkrétních potřeb lze do projektu přidávat další knihovny (například pro zabezpečení aplikace nebo komunikaci s databází), které jsou přímo kompatibilní s tímto frameworkem. Dále stojí za zmínku, že Spring Boot do výsledného JAR souboru automaticky vkládá webový server (v základě webový server Tomcat) a při spuštění aplikace tento server i spustí, což nesmírně usnadňuje nasazení a i distribuci aplikace [30].

Volba Spring Boot pro CANgui byla kromě výše zmíněných výhod také založena na dobrých zkušenostech s frameworkem v minulosti.

4.1.4 jSerialComm

jSerialComm je Javová knihovna, vyvinuta společností Fazecast, Inc. Jejím účelem je poskytnout univerzální rozhraní pro komunikaci po sériovém portu. Jelikož se jedná o nízkoúrovňový způsob komunikace, tak jSerialComm pracuje pouze s daty na úrovni jednotlivých bytů [31].

Na trhu existuje mnoho dalších knihoven, které se zabývají stejným problémem. Ze zkušeností je ale jSerialComm nejlepší řešení, jelikož kromě toho, že umožňuje komunikaci po několika portech současně, navíc i umožňuje prohledat všechny sériové porty podle jmenných deskriptorů. Tuto funkcionalitu jsme u žádné další knihovny nenašli a jelikož předem nedokážeme určit, do kterého portu bude zařízení připojeno, tak je tato funkcionalita kritická pro hladký uživatelský zážitek.

CANgui používá tuto knihovnu jako způsob komunikace mezi Ocarina Klientem (viz sekce 8.2.3) a Ocarinou IV (viz sekce 8.2.1), pomocí Ocarina Protokolu (viz sekce 8.2.2).

4.2 Front-end

Jelikož je CANgui interaktivní webová aplikace, tak většina použitých technologií pro front-end je v dnešní době považovaná za standard. Výběr rozšiřujících knihoven byl zpravidla založený na předchozích zkušenostech.

4.2.1 HTML 5

Hypertext Markup Language (HTML) je značkovací jazyk určený pro tvorbu webových stránek [32]. Verze 5 je nejnovější verzí HTML a oproti starším verzím disponuje mimo jiné například novými značkami (které jsou primárně určeny pro lepší sémantiku stránky) a také podporou offline aplikací [33].

4.2.2 CSS

Cascading Style Sheets (CSS) je jazyk, který definuje způsob, jakým se mají prvky značkovacího jazyka (primárně HTML) zobrazovat. Dosahuje tak pomocí takzvaných selektorů, kterými definuje které prvky budou ovlivněny a vlastností, které daným prvkům definují konkrétní pravidla, podle kterých se prvky zobrazí. Tyto vlastnosti sahají od změny barev až po umístění na stránce [34].

4.2.3 Sass

Syntactically Awesome Style Sheets (Sass) je rozšíření jazyka CSS, které obsahuje spoustu nových vlastností a funkcí, které umožňují jednodušší způsob stylování. Tyto vlastnosti zahrnují mimo jiné například možnost použití proměnných, vnořování (nesting) pravidel nebo také dědičnost selektorů. Před použitím na stránce se Sass soubory kompilují do standardních CSS souborů [35].

4.2.4 JavaScript

JavaScript je další klíčovou technologií pro tvorbu moderních webových aplikací. Jedná se o programovací jazyk, který se spouští na stroji klienta. Mimo jiné nám JavaScript umožňuje vytvářet dynamické webové stránky, jelikož máme přístup přímo k Domain Object Model (DOM) [36] a můžeme ho také upravovat. Dokážeme tak za běhu stránky měnit její obsah – například můžeme ze serveru načíst nová data pomocí technologie AJAX a vložit je do stránky [37][38].

4.2.5 React

React je JavaScriptová knihovna od společnosti Facebook, jež se zaměřuje na vývoj interaktivního uživatelského rozhraní pro webové aplikace [39]. React, stejně jako Spring Boot, umožňuje vývojáři začít programovat samotnou aplikaci během pár minut s pomocí takzvaného Create React App (CRA). CRA za vás založí projekt, nastaví základní závislosti a také provede základní konfiguraci [40].

4.2.6 Material-UI

Material-UI je knihovna pro usnadnění tvorby uživatelského rozhraní v Reactových aplikacích [41]. Tato knihovna poskytuje komponenty, které odpovídají komponentám specifikovaných ve směrnících Material Design od společnosti Google [42].

4.2.7 µPlot

µPlot je JavaScriptová knihovna pro tvorbu grafů. Tato knihovna poskytuje relativně obsáhlé API pro nastavení jednotlivých grafů, jež lze dále konfigurovat pomocí vlastních callbacků a pluginů. Kromě renderování samotných dat také vykresluje a obstarává osy a popisky grafů [43]. Tyto vlastnosti ve spojení s výkonem samotné knihovny (rychlé renderování a malé vytížení paměti oproti ostatním knihovnám) je důvod pro zvolení této knihovny.

Mezi zvažovanými možnostmi byla i například populární knihovna Recharts [44], jež je přímo implementovaná pro React, bohužel ale výkonnostně absolutně nestačila na renderování v reálném čase. V ohledu výkonnosti existují řešení, která jsou o něco lepší oproti knihovně µPlot, jelikož využívají hardwarovou akcelerace pomocí WebGL API [45][46], samotná integrace těchto knihoven je ale mnohem náročnější, jelikož zpravidla umí pouze vykreslit čárové grafy na základě vložených dat. To znamená, že osy, ovládání přiblížení, popisky a podobné vlastnosti bychom museli implementovat sami. S ohledem na složitost tohoto úkolu, v porovnání s použitím knihovny µPlot, jež všechny tyto vlastnosti podporuje automaticky, nás přivedlo k rozhodnutí použít zmíněnou knihovnu µPlot.

4.3 Komunikace mezi front-endem a back-endem

Jelikož front-end a back-end aplikace jsou rozděleny, je nutné zařídit způsob přenosu dat do front-endu pro prezentaci dat.

4.3.1 HTTP

Hypertext Transfer Protocol (HTTP) je bezestavový protokol aplikační vrstvy síťového modelu TCP/IP určený primárně ke komunikaci s webovými servery [47]. Pro lepší sémantiku dotazů HTTP definuje takzvané dotazovací metody, které specifikují jakou akci chceme s daným zdrojem provést. Nejpoužívanější metody jsou: `GET`, `POST`, `PUT`, `DELETE` [48].

Součástí HTTP odpovědi jsou takzvané stavové kódy, které indikují úspěšnost daného dotazu a případně prozrazují kde nastala chyba [49].

Naše aplikace bude kromě přenosu obsahu webových stránek k uživatelům, používat HTTP pro komunikaci s REST API (viz sekce 6.5) webového serveru.

4.3.2 WebSocket

WebSocket je komunikační protokol umožňující obousměrnou komunikaci po TCP spojení [50]. Tento protokol je zejména používán, pokud si přejeme dostávat od serveru data (bez explicitního vytvoření požadavku) a také na server data plánujeme posílat [51]. Významné příklady použití WebSocket protokolu jsou chatovací aplikace nebo online editory.

4.3.3 STOMP over WebSocket

Simple (or Streaming) Text Oriented Message Protocol (STOMP) je textově orientovaný komunikační protokol, který slouží pro asynchronní komunikaci mezi různými klienty. Server slouží pouze jako prostředník této komunikace. STOMP definuje základní příkazy (a odpovědi), orientované kolem komunikace mezi těmito klienty. Hlavními příkazy jsou *SUBSCRIBE*, který uživatele přihlásí k odběru zpráv pro specifikovanou *adresu* a *SEND*, který zprávy na *adresu* posílá. Specifikovaná adresa pro tyto příkazy je dle definice STOMP libovolný řetězec znaků, sémantiku této adresy určuje až konkrétní implementace STOMP serveru [52].

Implementace STOMP serveru ve Spring Boot definuje dva různé komunikační modely. První jsou takzvané fronty (queue), které jsou určené pro komunikaci mezi dvěma klienty (point-to-point). Druhý typ (který používá CANgui) jsou témata (topic). Témata jsou určena pro komunikaci mezi několika klienty zároveň, podle návrhového vzoru publish-subscribe [53][54].

STOMP není limitovaný pouze na protokol WebSocket, jeho použití po WebSocketu ale pro náš případ užití dává největší smysl. V naší aplikaci budeme za témata považovat Ocarina relace (viz sekce 6.4), kde obsahem zpráv budou přijatá data na CAN sběrnici nebo požadavky na simulaci provozu na sběrnici (odchozí zprávy, odeslané z uživatelského rozhraní).

Kapitola 5
Datový model

Aplikace CANgui pracuje na všech vrstvách aplikace s daty z CAN sběrnice. Je tedy nutné správně definovat jakým způsobem budou tato data ze sběrnice interpretována v aplikaci. K vyobrazení podčástí datového modelu aplikace jsou použity diagramy tříd UML.

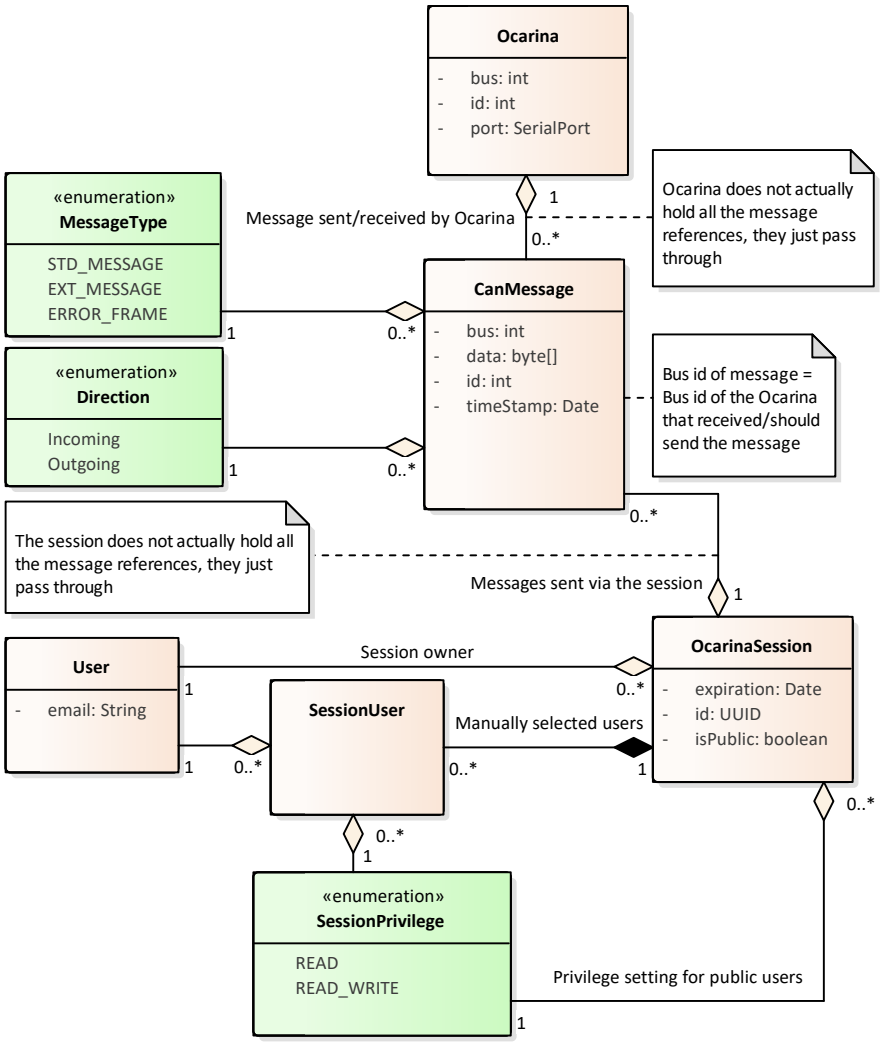
5.1 Datový model komunikace v CANgui

Datový model komunikace (viz obrázek 5.1) primárně popisuje formát zpráv, které jsou základním stavebním kamenem aplikace. Zprávy prochází všemi vrstvami aplikace – od CAN sběrnice/Ocarina Klientu, až po uživatelské rozhraní CANgui, kde jsou zprávy dočasně ukládané a pak dále interpretované a zobrazené v grafických prvcích aplikace. Zprávy prochází komunikačním kanálem takzvaných Ocarina relací (viz sekce 6.4), do kterých se připojují uživatelé z uživatelského rozhraní CANgui.

Dále model popisuje i autorizační nastavení Ocarina relací. Uživatel, co vytváří relaci přes Ocarina Klient, si může zvolit kdo bude mít oprávnění se k relaci připojit (číst data). Kromě toho může ještě určit, zda může na sběrnici i tento uživatel zprávy posílat. Je možné buď explicitně vyjmenovat jednotlivé uživatele, kterým bude povolení přiděleno, nebo lze použít takzvaných veřejných nastavení, která udělí povolení (číst či i zapisovat) všem uživatelům.

5.2 Datový model CAN sběrnice

Jedním z hlavních požadavků (funkční požadavek č. 3) na aplikaci, je možnost mapování zpráv na externí datový model. To nám v první řadě umožní interpretovat přijatá data do formy čitelné pro člověka. Dále nám to také umožní vytvořit příjemné uživatelské rozhraní, pro zadávání jednotlivých polí zpráv v reálných jednotkách, což zlehčí uživatelům práci při simulování ECU (viz funkční požadavek č. 2).



Obrázek 5.1: Datový model komunikace

5.2.1 CANdb

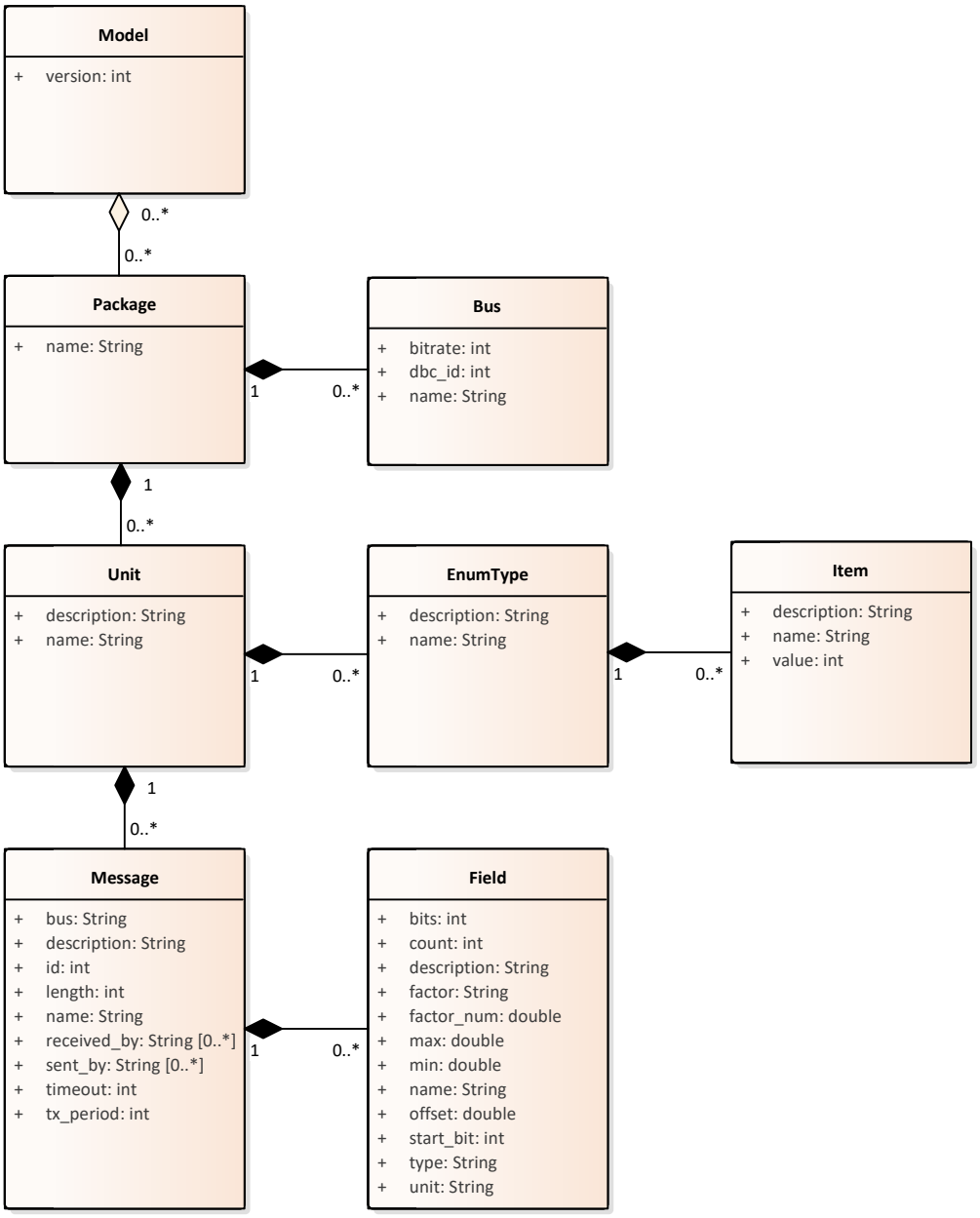
CANdb je webová aplikace vyvinuta v předchozích letech Ing. Martinem Cejpem, pro účely týmu eForce FEE Prague Formula. Jejím hlavním účelem je centrálně udržovat datový model CAN sběrnic jednotlivých generací formulí týmu [55].

CANdb nabízí kromě základních CRUD operací také spousty užitečných funkcí pro vývojáře ECU v týmu eForce. CANdb například umožňuje generování kódu pro odesílání a přijímání všech relevantních zpráv dané ECU. Pro CANgui je ale nejdůležitější CANdb API, které mimo jiné nabízí takzvaný CANdb JSON 2.0, jež obsahuje datový model CAN sběrnic žádané generace formule v JSON formátu [55].

5.2.2 CANdb JSON 2.0

CANdb JSON 2.0 je datový formát, který poskytuje detailní informace o datovém modelu CAN sběrnic (formát není omezený na počet sběrnic). Tento formát je srovnatelný například s proprietárním formátem DBC [21] od společnosti Vector Informatik GmbH, který je mimo jiné využívaný v jejich aplikaci CANoe (viz sekce 3.3).

Data v tomto formátu obsahují informace o datovém modelu celého vozidla. Jak lze vidět na obrázku 5.2, tak tento model je rozdělený na takzvané balíčky (package), které pomáhají rozdělit datový model daného vozidla do logických celků.



Obrázek 5.2: CANdb JSON 2.0

Jednotlivé balíčky mohou například obsahovat pouze ECU specifické pro dané vozidlo, nebo naopak soubor ECU/nástrojů, které dané vozidlo využívá, ale není specifické pouze pro něj (v případě elektromobilu to může být například nabíječka). Tyto balíčky kromě informací o ECU také zahrnují informace o CAN sběrnicích, které jsou pro daný balíček relevantní.

Tento model je kritický pro naší aplikace, jelikož právě díky němu jsme schopni interpretovat přijaté zprávy z CAN sběrnice do formy čitelné pro člověka. Navíc, tento formát není omezený pouze na formule týmu eForce FEE Prague Formula, ale umožňuje definovat si jakýkoliv model zpráv, co chodí po CAN sběrnici.

CANgui si vybraný model ukládá perzistentně v paměti prohlížeče pomocí localStorage API. LocalStorage je API webových prohlížečů, které umožňuje aplikaci číst a zapisovat data do paměti prohlížeče v textové formě. Tato data jsou dostupná pouze pro stránky na stejné doméně [56].

5.3 Custom window model

Jedním z požadavků (funkční požadavek č. 7), jež dělá CANgui všestranným nástrojem, je umožnit uživatelům vytvořit si vlastní obrazovky. Hlavním záměrem této funkcionality je zrychlit analýzu dat a usnadnit tak uživatelům diagnostiku problémů. Tyto obrazovky mohou například shrnovat kritické informace daného vozidla nebo jen vybrané ECU.

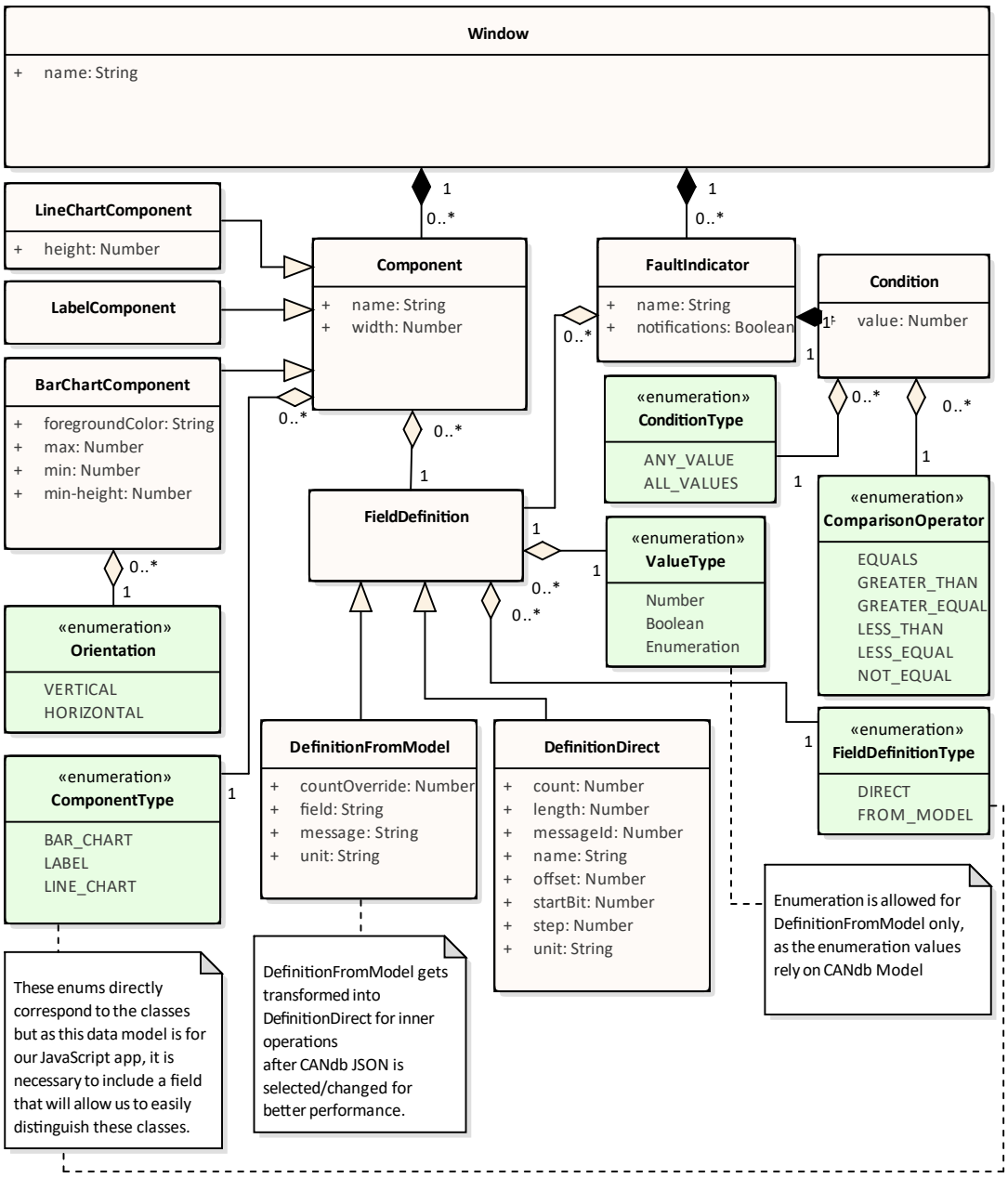
Pro tyto účely jsme vytvořili vlastní datový model (viz obrázek 5.3), který popisuje jednotlivé komponenty obrazovek a i jakým způsobem jsou definovaná data, která jsou v těchto komponentách zobrazená.

Tento datový model v současné podobě umožňuje vkládat do obrazovky tři typy komponent:

- Sloupcový graf
  - Pro sledování aktuální hodnoty vůči očekávané maximální/minimální hodnotě
- Čárový graf
  - Pro sledování vývoje hodnot
- Label
  - Pro zobrazení aktuální hodnoty jako textovou hodnotu

V budoucnu je možné model rozšířit o další druhy komponent, prozatím by ale tyto komponenty měly stačit pro většinu uživatelů. Kromě těchto komponent každá obrazovka umožňuje přidat indikátory chyb, jež se zobrazí v případě, že nastane nějaký chybový stav.

Stejně jako datový model CAN sběrnice si CANgui definice obrazovek udržuje v paměti prohlížeče pomocí localStorage API.



Obrázek 5.3: Custom window model

Kapitola 6
Architektura aplikace

V této kapitole jsou popsány hlavní části aplikace a jak jsou v kontextu celé aplikace provázány. Kromě toho jsou zde popsány i základní architektonické styly, jež CANgui používá.

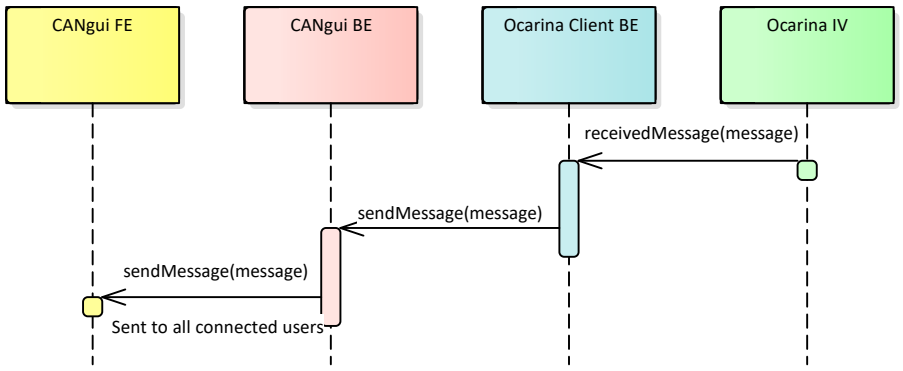
6.1 Klient-server

Komunikační model klient-server je založen na principu, že klient (zpravidla aplikace u koncového uživatele, která není z pohledu internetu veřejně přístupná) vytváří a posílá požadavky na server (typicky tato komunikace probíhá po internetu, server ale klidně může být na stejném zařízení jako klient – jako v případě Ocarina Klientu). Server tyto požadavky zpracuje a vrátí klientovi odpověď. Jedná se tedy o oboustranný způsob komunikace, komunikaci ale pokaždé musí zahájit klient svým požadavkem [57][58]. Za speciální případ tohoto komunikačního modelu lze považovat komunikační protokol WebSocket, jež umožňuje serveru odesílat data klientovi libovolně, samotné otevření komunikačního kanálu ale musí zahájit klient.

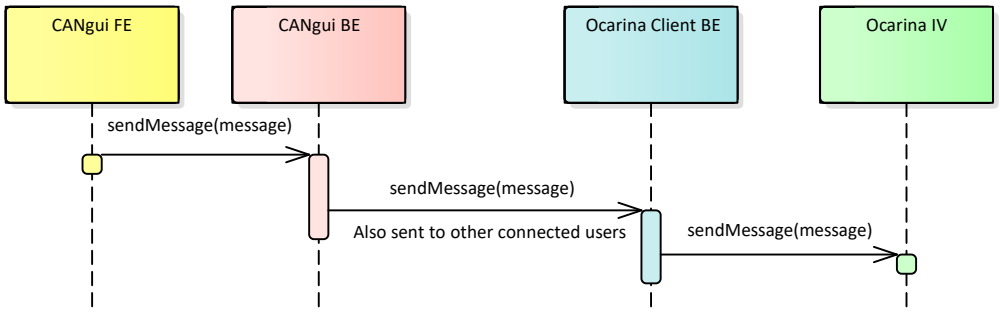
Typicky je tento komunikační model u webových aplikací použit pro komunikaci mezi back-endem a front-endem aplikace, v našem případě je ale také použit pro komunikaci mezi Ocarina Klientem a CANgui Web serverem.

6.2 CANgui Web server

CANgui Web server je nejdůležitější prvek celé aplikace. Kromě toho, že poskytuje přístup k uživatelskému rozhraní aplikace CANgui, je také zodpovědný za zprostředkování Ocarina relací, přes které probíhá oboustranná komunikace mezi uživatelským rozhraním CANgui a CAN sběrnicí (viz obrázky 6.1 a 6.2).



Obrázek 6.1: Příjem zprávy z CAN sběrnice



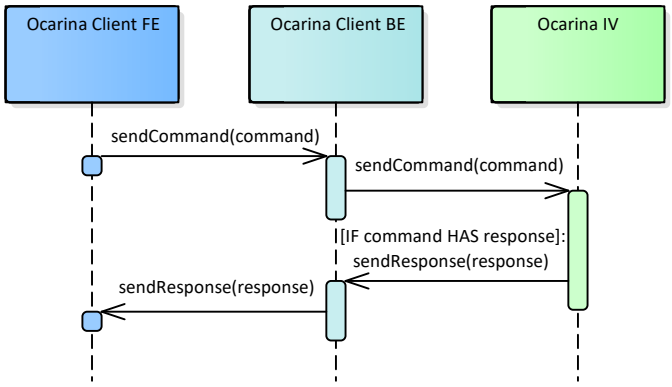
Obrázek 6.2: Poslání zprávy na CAN sběrnici

6.3 Ocarina Klient

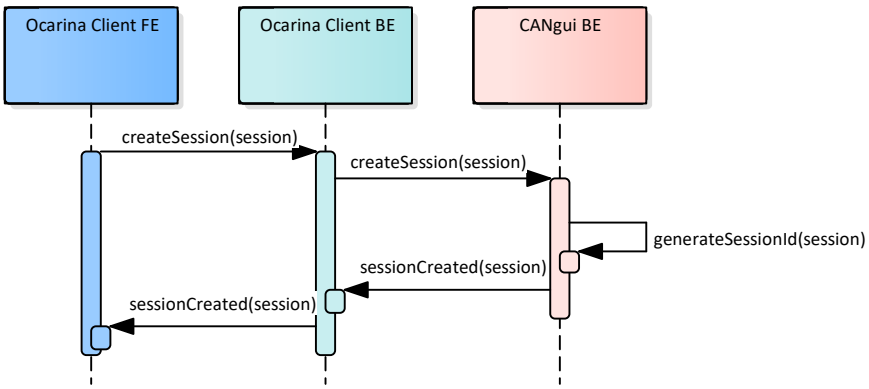
Ocarina Klient je druhou komponentou celku CANgui. Ocarina Klient je interně také webová aplikace. Webový server (a uživatelské rozhraní) slouží pouze ale jako rozhraní ke konfiguraci připojených zařízení Ocarina IV a je spouštěn přímo na zařízení koncového uživatele, jako desktopová aplikace. V kontextu komunikačního modelu klient-server lze tedy považovat Ocarina Klient za klientskou aplikaci. V kontextu celé aplikace se ale jedná o poskytovatele služeb – ve smyslu poskytování dat pro uživatelské rozhraní CANgui.

Hlavní zodpovědností této komponenty je přímá komunikace se zařízením Ocarina IV (viz sekce 8.2.1) po sériovém rozhraní (viz obrázek 6.3) pomocí Ocarina Protokolu (viz sekce 8.2.2), jež Ocarina Klient plně implementuje.

Další zodpovědností Ocarina Klientu je přenos informací z CAN sběrnice (z Ocariny) na CANgui Web server a zpět. Proto musí v první řadě Ocarina Klient poslat požadavek na založení relace (viz obrázek 6.4), po které bude komunikace mezi CANgui a Ocarina Klientem probíhat.



Obrázek 6.3: Komunikace se zařízením Ocarina IV



Obrázek 6.4: Vytvoření Ocarina relace

6.4 Ocarina relace

Ocarina relace (session) je základní komunikační prvek aplikace CANgui. Probíhá po nich komunikace mezi CAN sběrnicí a uživatelským rozhraním CANgui (konkrétně jsou data posílána z Ocarina Klientu na CANgui Web server, jež je přeposílá všem připojeným uživatelům).

Relace jsou založené na základě požadavků Ocarina Klientů a musí být udržované naživu daným Ocarina Klientem. Uživatelé, kteří zakládají danou relaci mají možnost konfigurace autorizačních nastavení pro tuto relaci. To znamená, že můžou podle emailových adres ručně přidat uživatele, se kterými si přejí tuto relaci sdílet. Dále mohou nastavit, zda daným uživatelům chtějí povolit posílání zpráv. Kromě vyjmenování konkrétních uživatelů mohou povolit přístup všem uživatelům (kteří znají ID dané relace) a vybrat, zda jim také povolí posílat zprávy, či nikoliv.

Implementačně to jsou takzvané STOMP témata (topic) (viz sekce 4.3.3) s přidanou autorizační logikou, takže veškerý přenos zpráv v těchto relacích probíhá po protokolu STOMP (over WebSocket).

6.5 Representational state transfer API

Representational state transfer (REST) je architektonický styl pro tvorbu aplikačních rozhraní (Application Programming Interface – API), zpravidla určených pro přístup a interakci s daty, pomocí základních HTTP požadavků [59][60].

CANgui Web server i Ocarina Klient implementují REST API. CANgui Web server poskytuje API pro interakci s Ocarina relacemi (zakládání, upravování, obnovení). Ocarina Klient kromě API pro interakci s Ocarina relacemi také poskytuje API pro konfiguraci připojených Ocarina IV zařízení.

6.6 Single-page application

Single-page application (SPA) je způsob tvorby webových stránek, kde při navigaci po stránce dochází k dynamickému generování obsahu, místo načítání celého HTML obsahu znovu [61]. To způsobuje větší pocit plynulosti a responzivity stránky pro uživatele [62][63]. Tím, že není stránka neustále aktualizovaná, nám mimo jiné umožňuje držet si data pro konkrétní relaci pouze lokálně. To je pro náš případ použití kritické, jelikož nám to umožňuje si držet přijaté zprávy přímo u uživatele a při navigaci po stránce tato data neztratíme.

Tohoto výsledku se dosahuje použitím JavaScriptu. Zpravidla se používá nějaká knihovna, která generování a správu obsahu obstarává za vás (v našem případě React), stejného výsledku lze ale dosáhnout i bez použití jakýchkoliv knihoven [61][62][63].

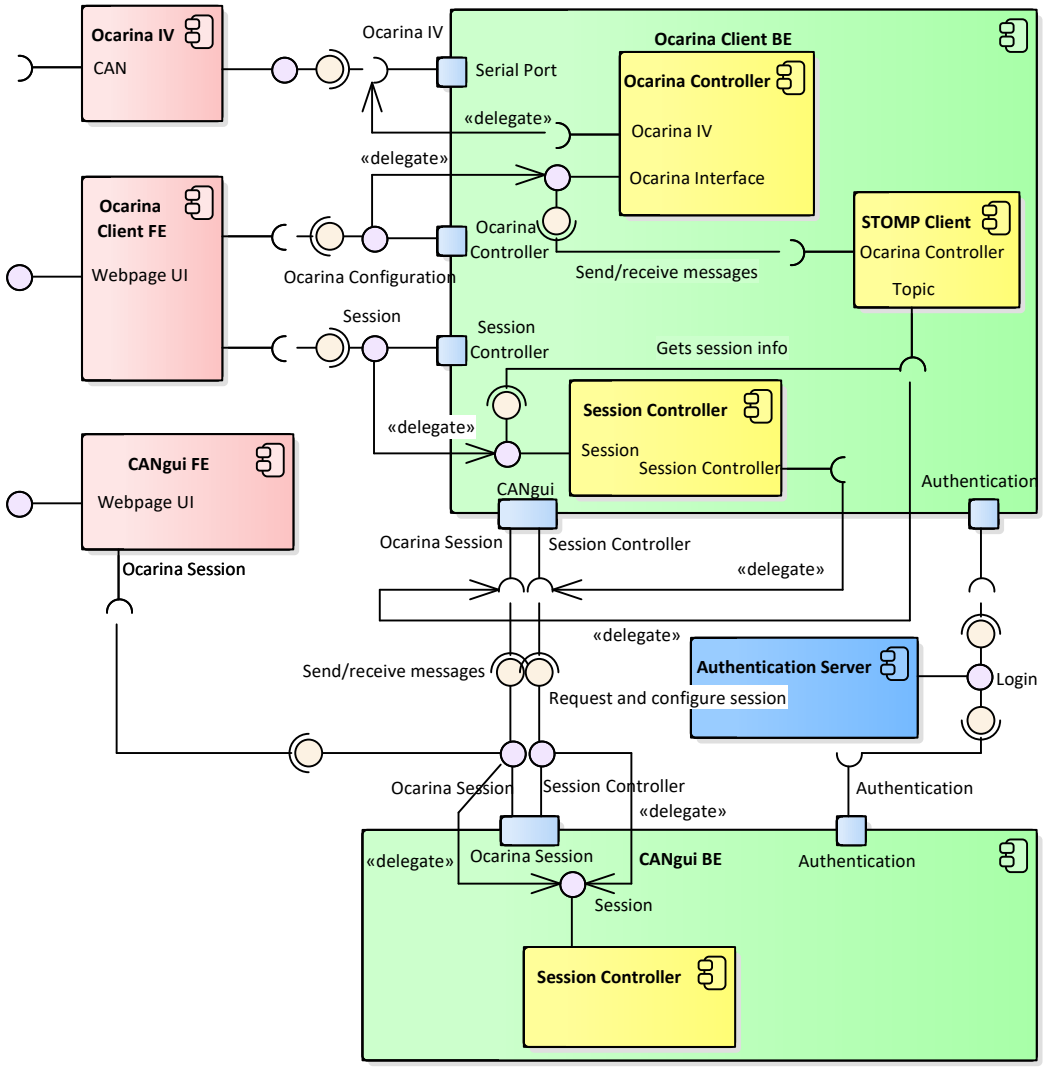
Pokud daná aplikace slouží pro prezentaci dat, která musí být načtena ze serveru, používá se k načtení těchto dat technologie Asynchronous JavaScript and XML (AJAX) [61][62][63]. Z názvu se zdá, že lze přenášet pouze data ve formátu XML, tomu ale není pravda – lze přenášet libovolný datový formát [38].

Pokud se jedná o aplikaci, která chce data přijímat i bez explicitního požadavku (například chatovací aplikace), lze k tomu použít například technologie WebSocket. WebSocket rozhodně není jediné řešení, pomocí kterého dosáhneme tohoto výsledku, v dnešní době se ale považuje za nejlepší řešení pro tuto problematiku.

Oba front-endy (CANgui i Ocarina Klientu) jsou implementovány pomocí Reactu a jsou tedy Single-page application (SPA).

6.7 Diagram komponent

Diagram komponent (viz obrázek 6.5) popisuje rozložení systému do funkčních celků a jejich rozhraní, pomocí kterých bude komunikace mezi jednotlivými komponentami probíhat.

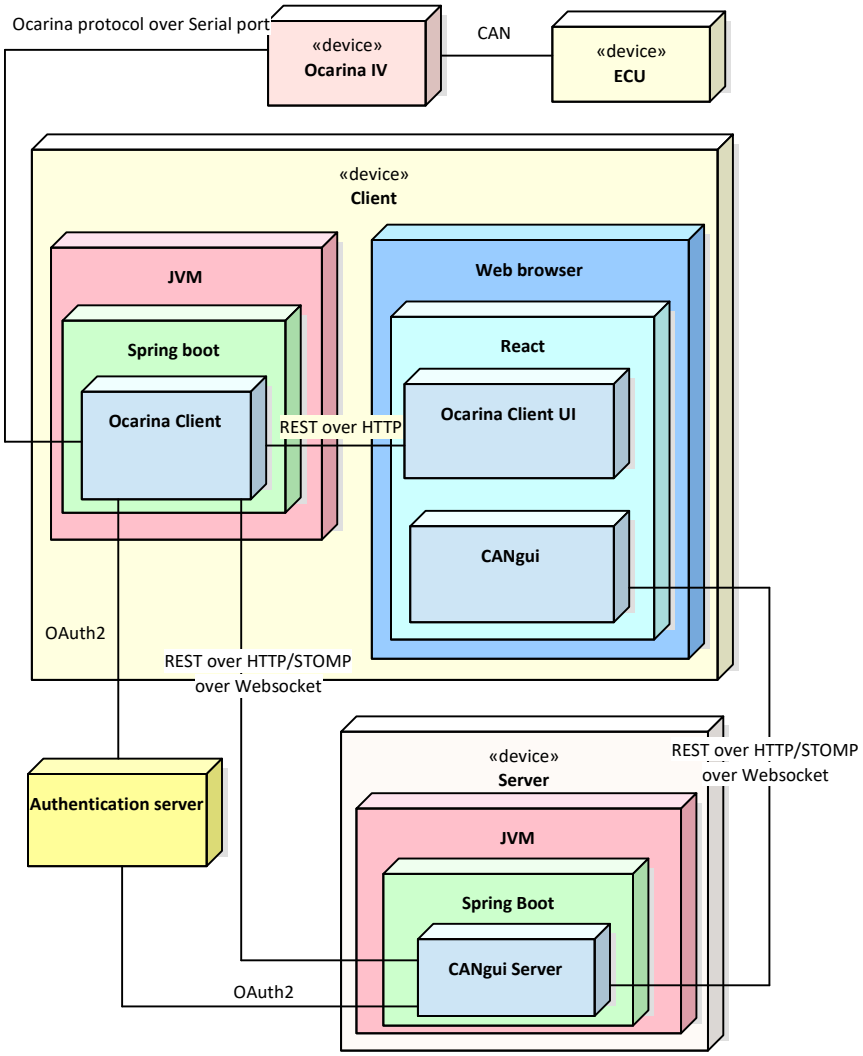


Obrázek 6.5: Diagram komponent

Aplikace je konkrétně rozdělena na 6 komponent, jež lze dále rozdělit na dílčí komponenty. Hlavními komponentami jsou v tomto ohledu back-end části aplikací CANgui a Ocarina Klient. Tyto komponenty spolu komunikují po dvou rozhraních, jejichž poskytovatel je back-end CANgui. Prvním rozhraním je ovládací rozhraní Session Controller, jež umožňuje Ocarina Klientu zakládat nové relace a udržovat je v běhu. Druhé rozhraní, úzce spojené, je rozhraní přes které samotné relace probíhají. Přes toto rozhraní prochází oboustranná komunikace mezi front-endem CANgui a back-endem Ocarina Klientu. Prostředníkem druhého rozhraní je back-end CANgui, jež zajišťuje rozposlání zpráv mezi všechny připojené účastníky.

6.8 Diagram nasazení

Diagram nasazení (viz obrázek 6.6) popisuje jednotlivá zařízení a protokoly, pomocí kterých budou mezi sebou zařízení v CANgui komunikovat.



Obrázek 6.6: Diagram nasazení

Tento diagram konkrétně popisuje scénář, kdy uživatel používá zařízení Ocarina IV jako zdroj dat pro aplikaci CANgui a je tedy připojen ke CAN sběrnici. Ocarina Klient je spouštěn na zařízení uživatele, jelikož je nutné zařídit přístup k sériovému rozhraní počítače pro komunikaci se zařízením Ocarina. Kromě Ocarina Klientu je na zařízení uživatele také spuštěn webový prohlížeč, přes který uživatel zařízení Ocarina může konfigurovat a dále založit Ocarina relaci, ke které se později připojí on nebo i další uživatelé. Zařízení na kterém je spuštěn CANgui Server je v principu libovolný počítač, ke kterému se dokáže Ocarina Klient připojit. Pro případy offline použití aplikace lze tedy spustit CANgui server i na zařízení uživatele. V tomto případě nepoužívá ani jedna komponenta autorizační server.

Část IV

Vývoj

Kapitola 7
Uživatelské rozhraní

Jak bylo již zmíněno v sekci o použitých technologiích, uživatelské rozhraní CANgui je implementováno pomocí knihovny React od společnosti Facebook [39].

Hlavním cílem při návrhu bylo poskytnout uživatelům uživatelské rozhraní, jež je přímočaré a jednoduché k použití i bez předchozích zkušeností nebo nutnosti proškolení uživatelů. Návrh se drží doporučení Material Design pro web od společnosti Google [42].

7.1 Prototypování

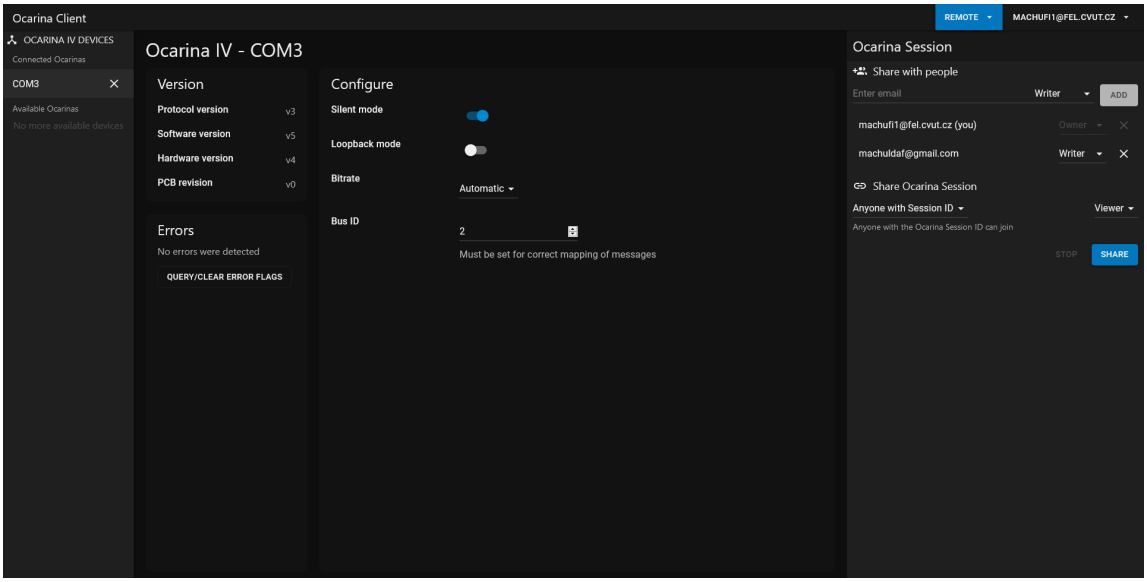
Před samotnou implementací uživatelského rozhraní byl vytvořen prototyp, jenž popisuje základní rozložení aplikace a ovládacích prvků. Prototyp byl vytvořen v aplikaci Adobe XD [64]. Tento prototyp lze vidět v příloze A.

Tvorba prototypu probíhala iterativně. V první řadě bylo navržené rozložení uživatelského rozhraní pomocí wireframů. Postupně se přidávala další míra detailu a až nakonec se přidaly barvy. Prototyp je vytvořen pouze v tmavém režimu, samotné uživatelské rozhraní nabízí ale i režim světlý.

Celý proces tvorby prototypu uživatelského rozhraní byl konzultován s vybranými členy týmu eForce. Na základě poznatků členů byl prototyp v průběhu tvorby upravován, aby lépe plnil potřeby týmu. Tvorba prototypu před samotnou implementací nám umožnila pracovat kreativně a vytvářet rychlé změny v návrhu rozložení stránky. Dále nám tvorba prototypu pomohla vytvořit vizuální identitu aplikace, které jsme se při vývoji mohli držet.

7.2 Ocarina Klient

Uživatelské rozhraní Ocarina Klientu slouží v první řadě k připojení a synchronizaci zařízení Ocarina IV. Pro synchronizované zařízení umožňuje upravovat jejich konfiguraci (viz obrázek 7.1), dle možností zařízení viz příloha C.



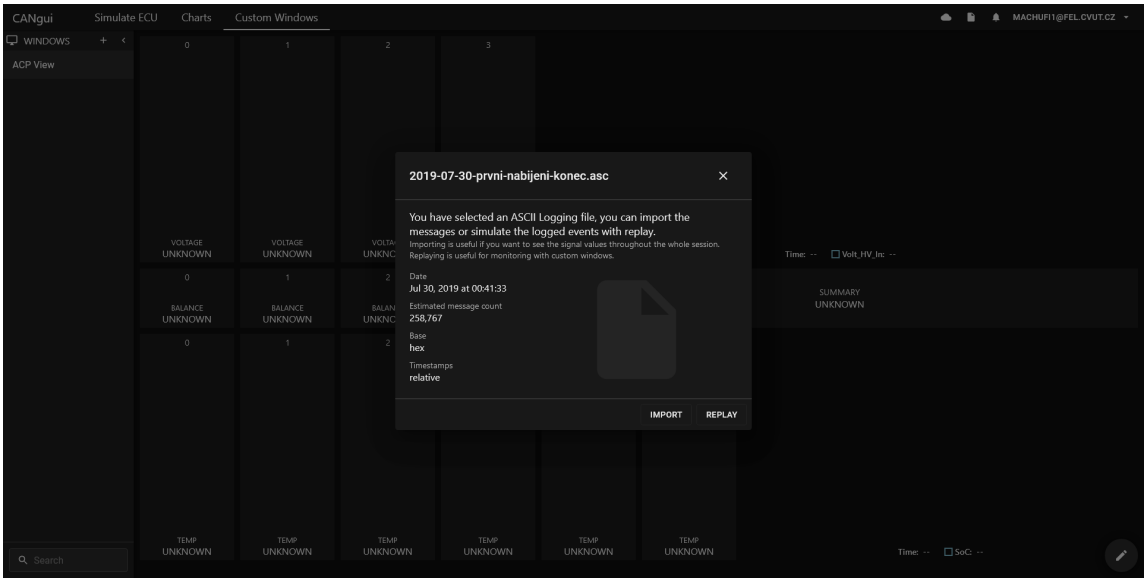
Obrázek 7.1: Ocarina Klient – Dashboard

Kromě operací se zařízeními Ocarina, Ocarina Klient umožňuje konfiguraci Ocarina relací. Uživatel může vytvořit novou relaci s připojenými zařízeními Ocarina a upravovat její autorizační nastavení (viz sekce 6.4).

7.3 CANgui

Uživatelské rozhraní CANgui (většina obrazovek lze vidět v příloze B) je o něco rozsáhlejší a umožňuje tak uživateli o poznání více možností. V první řadě se uživatel musí přihlásit do aplikace.

CANgui využívá autentizaci a autorizaci na bázi protokolu OAuth 2, který umožňuje, aby autorizační server (*poskytovatel identity*) a samotná aplikace byly vzájemně nezávislé. Z praktických důvodů je však v první verzi jediným podporovaným autorizačním serverem interní *eForce SSO*[1].

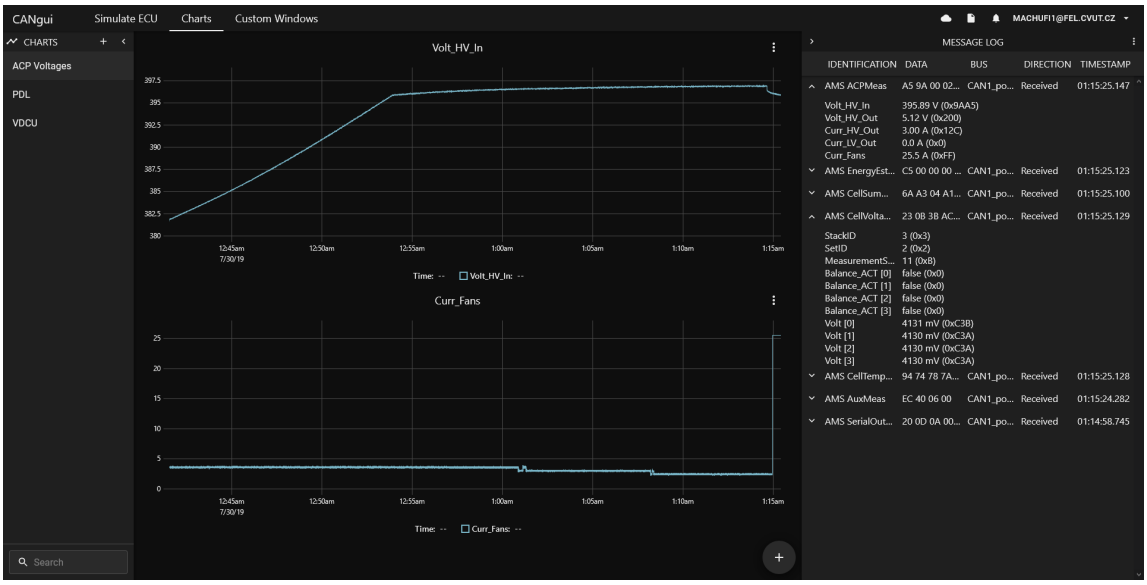


Obrázek 7.2: CANgui – Import dialog

Po přihlášení je důležité zvolit zdroj dat. Uživatelé mají dvě možnosti: připojit se k Ocarina relaci pomocí ID, nebo nahrát do aplikace soubor se záznamem (viz obrázek 7.2). Po zvolení zdroje dat se uživateli zobrazí seznam přijatých zpráv. Tento seznam má dva módy: poslední přijatá zpráva (zobrazí od každé zprávy pouze poslední přijatou zprávu – podle ID), nebo všechny zprávy (seřazených podle času přijetí). Pokud má přijatá zpráva definici ve zvoleném datovém modelu CAN sběrnice (viz sekce 5.2), tak lze zprávu rozkliknout a podívat se na interpretované hodnoty jednotlivých polí zprávy.

Je samozřejmé, že seznam přijatých zpráv pro kvalitní analýzu dat nestačí. Proto CANgui umožňuje uživateli sledovat hodnoty jednotlivých polí pomocí čárových grafů (viz obrázek 7.3). Toto řešení je vhodné pokud uživatel nepotřebuje komplexní obrazovku s různými typy grafických prvků, ale stačí mu pouze sledovat vývoj hodnot vybraných polí. Pokud si uživatel přeje vyšší úroveň přizpůsobení, má možnost si definovat vlastní obrazovky (viz obrázek 7.4).

[1] Single sign-on (SSO) umožňuje uživateli přihlásit se do vícero (zpravidla propojených) systémů se stejnými přihlašovacími údaji [65].



Obrázek 7.3: CANgui – Obrazovka s grafy

Obrázek 7.4: CANgui – Vlastní okno při replayi

Kapitola 8
Implementační řešení

V této kapitole je popsán proces praktické implementace aplikace a hlavní výzvy, které byly v rámci implementace řešeny. Jmenovitě se jedná o zabezpečení STOMP témat, komunikace s CAN sběrnicí a práci s logovacími soubory.

8.1 Zabezpečení

Aplikace CANgui podporuje dva módy použití. V prvním (online) módu se uživatel připojuje k veřejně přístupnému serveru CANgui, jež je zabezpečený pomocí OAuth2. V druhém (offline) módu aplikace spouští uživatel CANgui server na svém zařízení. Jelikož se předpokládá s případem, že uživatel nebude mít internetové připojení, tak v tomto módu aplikace zabezpečená není.

Ocarina Klient při spuštění autentizaci uživatele nepožaduje, jelikož se předpokládá s případem, že se bude uživatel chtít připojit k lokálnímu CANgui serveru. Autentizace je tedy vyžádaná pouze v případě, že se snaží připojit k veřejnému CANgui serveru.

8.1.1 Zabezpečení Ocarina relací

Kromě základního zabezpečení aplikace, pomocí kterého omezujeme přístup do aplikace pouze na uživatele registrované v dané organizaci, je také důležité spravovat autorizaci uživatelů pro konkrétní Ocarina Relace (viz sekce 6.4).

Uživatel jež relaci zakládá mimo jiné určuje autorizační nastavení. Má možnost určit konkrétně kterým uživatelům povolí se k této relaci připojit a zda jim dovolí do relace zprávy posílat. Kromě konkrétního určení uživatelů je možné udělat relaci takzvaně veřejnou, což znamená, že se k relaci může připojit kdokoliv, kdo zná její ID.

Tyto relace na pozadí fungují jako STOMP témata (viz sekce 4.3.3). Pro náš případ užití je kritické omezit, jaký uživatel se může k relaci může připojit (má autorizační právo pro čtení) a pak dále omezit který z již připojených uživatelů může zprávy do relace posílat (autorizační právo pro zápis). Ani jedna z těchto vlastností v základu pro STOMP neexistuje.

Bylo tedy nutné si přizpůsobit chování pro náš konkrétní případ užití. Naštěstí Spring Boot umožňuje do řetězce, jež zpracovává přijaté zprávy v komunikačním kanále STOMP serveru, vložit interceptor, který nám umožní reagovat na konkrétní STOMP příkazy podle naší autorizační logiky.

Pro příkaz *SUBSCRIBE* (který přihlásí uživatele k přijímání zpráv pro dané téma) jsme tedy rozšířili logiku o dvě ověření. V první řadě se ujistíme, že se uživatel snaží připojit k existující relaci. Dále pokud relace existuje, tak si ověříme, že pro danou relaci má uživatel autorizační práva pro čtení. Pokud obě dvě ověření příkaz splňuje, tak necháme Spring Boot v řetězci zpracování pokračovat dál. Pokud ale některé z těchto ověření nesplní, tak řetězec přerušíme výjimkou s příslušnou chybovou hláškou. STOMP server si v tomto případě neuloží uživatele jako odběratele a odešle uživateli rámec *ERROR*, na jež přijetí musí klient dle STOMP definice ukončit připojení.

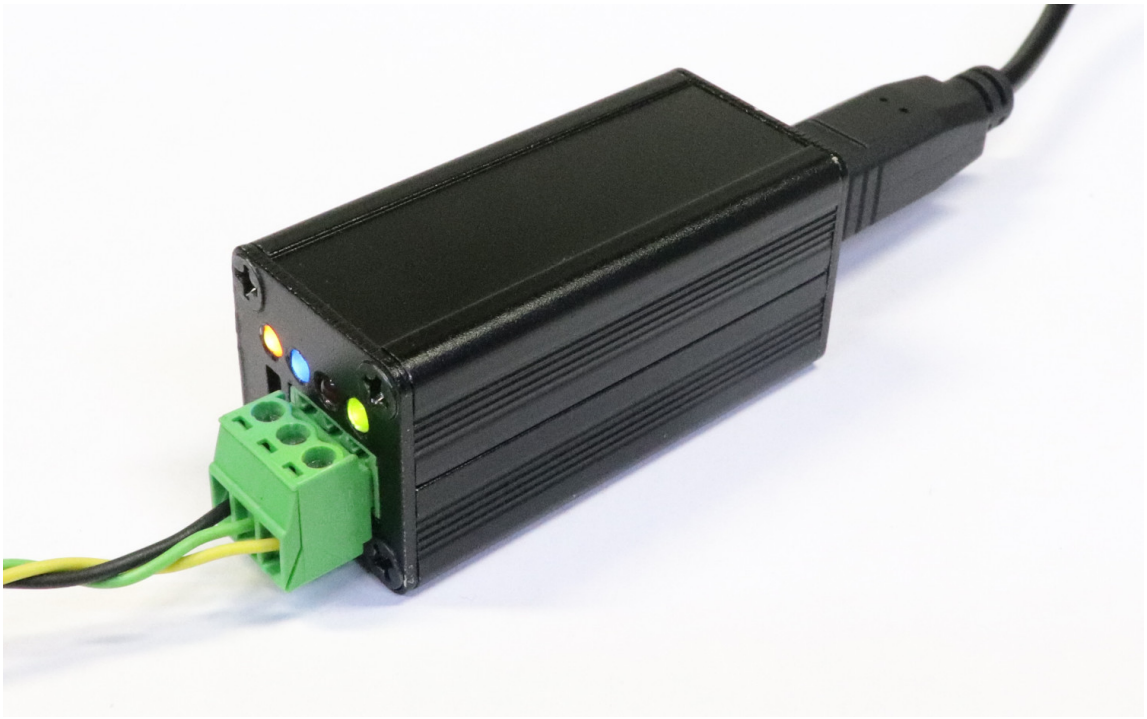
Podobně je vyřešená i autorizační logika pro příkaz *SEND*. Nejdříve ověříme, že se uživatel snaží poslat zprávu do existující relace (může například nastat případ, kdy se relace ukončí, ale klient o tom neví) a poté ověříme, že má uživatel příslušná autorizační práva pro odeslání této zprávy.

8.2 Komunikace s CAN sběrnicí

Jedním z hlavních prvků aplikace CANgui je schopnost čtení a posílání zpráv na CAN sběrnici. Pro tuto funkcionalitu tedy musíme definovat způsob, jak bude komunikace mezi webovou aplikací a CAN sběrnicí probíhat.

8.2.1 Ocarina IV

Ocarina IV (lze vidět na obrázku 8.1) je zařízení, které poskytuje rozhraní pro komunikaci s CAN sběrnicí po sériovém portu počítače [66]. Jejím aktuálním vývojářem je Ing. Patrik Bachan z týmu eForce FEE Prague Formula.



Obrázek 8.1: Ocarina IV připojená ke CAN sběrnici a PC [68]

Jak název napovídá, jedná se již o čtvrtou generaci tohoto zařízení, kompletně vyvinutém v prostředí týmu eForce. Mezi nejdůležitější vlastnosti toho zařízení patří přijímání, posílání zpráv a také detekce chyb (viz sekce 1.2.2) na CAN sběrnici.

Jednou z hlavních motivací pro vývoj tohoto zařízení byla možnost konkurence proprietárním, komerčním řešením, jako například zařízení VN7610 [69] od společnosti Vector Informatik GmbH, jelikož tým eForce má omezený počet zařízení a licencí k jejich software. Vlastní řešení kromě přidané úrovně volnosti totiž umožňuje rozšíření tohoto zařízení mezi vlastní vývojáře, což jim nadále pomůže při vývoji svých ECU.

Spolu s Ocarinou, Ing. Patrik Bachan také vyvinul referenční API v Pythonu, které plně implementuje protokol Ocariny. Toto API lze mimo jiné ovládat z příkazové řádky [66].

8.2.2 Ocarina Protokol v3

Jelikož je Ocarina IV komplexní zařízení, které poskytuje spousty funkcionality, bylo nutné vyvinout vlastní protokol[1] pro komunikaci s Ocarinou.

Datovou jednotkou komunikace jsou rámce, které přenáší veškeré zprávy mezi Ocarinou a PC. Zprávou může být myšleno náš příkaz, odpověď (na některé příkazy Ocarina asynchronně posílá odpověď) nebo vnitřně vyvolaná událost. Veškeré zprávy protokolu, včetně jejich formátu a funkcionality lze vidět v příloze C.

Formát jednotlivých rámců (viz tabulka 8.1) podléhá kódovacímu schématu známému jako Type-length-value (TLV). To znamená, že je zpráva rozdělená na 3 části.

1. Typ zprávy
   - Hodnota pole odpovídá ASCII hodnotě daného znaku
2. Délka datové části
   - Hodnota odpovídá počtu byte datové části
3. Datová část
   - Proměnlivá délka
   - Pořadí byte odpovídá endianitě little-endian

T	L	V0	V1	...	Vn
znak	n	data(LSB)	data	...	data(MSB)

Každé pole tabulky odpovídá velikosti právě 1 byte.

Tabulka 8.1: Ocarina IV Protokol – Formát rámců

[1] Veškeré informace týkající se tohoto protokolu jsou odvozené z oficiální dokumentace projektu [67].

8.2.3 Ocarina Klient

Ocarina Klient v kontextu komunikace s CAN sběrnicí slouží jako aplikační rozhraní pro jednoduchou komunikaci se zařízením Ocarina IV. Komunikace se zařízením Ocarina IV probíhá po sériovém rozhraní a Ocarina Klient pro tuto komunikaci používá knihovnu jSerialComm (viz sekce 4.1.4). Ocarina Klient plně implementuje celý Ocarina Protokol (viz sekce 8.2.2) a poskytuje uživatelské rozhraní (viz sekce 7.2), pomocí kterého mohou uživatelé jednoduše Ocarinu konfigurovat.

Ocarina Klient podporuje komunikaci s (teoreticky) neomezeným počtem Ocarin najednou, pro případy, že vozidlo na kterém daný uživatel pracuje má více CAN sběrnic.

8.3 Přehrávání a ukládání záznamů

Další důležitou součástí aplikace je možnost ukládat a zpětně přehrávat záznamy zpráv z CAN sběrnic(e). I přes to, že se nabízí možnost nahrát a zpracovat tato data na serveru, tak kvůli potenciální velikosti přenášených dat jsme se rozhodli veškerou práci se soubory provádět přímo v prohlížeči uživatele.

8.3.1 Datový formát

Datových formátů pro ukládání provozu CAN sběrnic existuje v současné době relativně dost [70][71]. Jedná se o jednoduché formáty (například CSV) až po velice sofistikované a komplexní formáty. Pro účely aplikace CANgui jsme zvažovali následující možnosti:

- ASAM MDF
  - Standard od roku 2019
  - Data v binární podobě
- BLF
  - Proprietární formát od společnosti Vector GmbH
  - Data v binární podobě
- ASC
  - Proprietární formát od společnosti Vector GmbH
  - Data v textové podobě
- Vlastní formát

Velice lákavou možností bylo pokusit se implementovat formát ASAM MDF [72], jež je v současnosti považovaný za průmyslový standard. Bohužel po hlubší studii jsme ale zjistili, že tento formát je díky své obsáhlé funkcionalitě ale velice komplexní a bohužel pro účely této práce je mimo naše možnosti.

Další zvažovanou možností byl formát BLF. Tento formát členové týmu eForce používají v současnosti nejčastěji, při práci s nástrojem CANoe od společnosti Vector GmbH (viz 3.3). Nejlákavější vlastnost tohoto formátu je velikost samotných souborů. Jelikož se jedná

o binární data, tak jsou soubory malé a jednoduše přenosné. Bohužel ale tím, že se jedná o proprietární binární formát, tak by pro nás bylo prakticky nemožné algoritmus pro kódování a dekódování reverzním inženýrstvím zreplikovat.

Jako další varianta je další proprietární formát od společnosti Vector GmbH, takzvaný ASCII Logging Files (.ASC) (ukázka formátu lze vidět ve výpisu 8.1). Oproti formátu BLF se ale jedná o textový formát, takže i přes to, že je samotný formát proprietární, tak lze celkem jednoduše zjistit, které části výstupu co znamenají.

Výpis 8.1: Formát .ASC souboru

```
date Wed May 05 12:37:48 PM 2021
base hex timestamps absolute
Begin Triggerblock Fri Aug 02 12:45:32 PM 2019

  0.000000 1  390        Rx D 8  E9  93  81  94  E6  FE  00  23
  0.010000 1  390        Rx D 8  E9  93  81  94  26  FF  00  23
  0.019000 1  390        Rx D 8  E9  93  81  94  D5  FE  00  24
  0.025000 1  394        Rx D 8  9D  00  00  00  00  00  00  01
  0.028000 1  497        Rx D 7  82  80  80  80  7A  7C  78
  0.030000 1  390        Rx D 8  E9  93  81  94  26  FF  00  24
  0.032000 1  496        Rx D 8  35  03  87  3B  B8  83  EB  B7
End Triggerblock
```

Největší nevýhodou tohoto formátu je velikost výstupních souborů. Jelikož se jedná o textový formát, tak při vysokém počtu zpráv je velikost výstupu mnohokrát větší oproti binárnímu protějšku.

Poslední zvažovanou variantou bylo si vyvinout vlastní formát, přímo pro účely týmu eForce. Tato možnost byla ale skoro okamžitě zamítnuta, kvůli znemožnění integrace s ostatními nástroji mimo ekosystém týmu eForce. Nástroj CANgui sice uživatelům poskytuje spousty funkcionality, není ale všemocný a proto je nutné zachovat možnost práce i s ostatními nástroji – konkrétně se jedná o CANoe a MATLAB.

Na základě těchto poznatků jsme udělali rozhodnutí, že bude v současné době nástroj CANgui podporovat formát .ASC. V budoucnu je ale určitě možné rozšířit, jaké formáty bude nástroj CANgui podporovat.

8.3.2 Přehrávání záznamu

Nástroj CANgui nabízí uživateli dva způsoby vložení dat ze souboru. První způsob je *import*. To znamená, že se do aplikace vloží všechna data najednou. Druhý způsob je *replay*, což znamená, že se zprávy do aplikace budou přidávat postupně, jako by byl uživatel živě připojený přímo ke sběrnici. Obě varianty ale fundamentálně fungují hodně podobně.

Jelikož při čtení dat ze souboru nechceme zastavit ovládání uživatelského rozhraní a základní chování JavaScriptu v prohlížeči je takové, že se veškeré skripty spouští v jednom vlákně, musíme využít takzvaných WebWorkerů. Pro jednoduchost to je způsob jak spustit skripty, ve vlastním vlákně. Jediný způsob komunikace s těmito Workery je pomocí událostí. Tyto události mohou obsahovat data, tato data jsou ale při přenosu kopírována (výjimkou jsou speciální typovaná pole, ty my ale nevyužíváme) [73][74].

Samotné dekódování ze souboru je rozdělené na dvě části. V první části čteme hlavičku souboru, ze které se dozvíme více o samotném formátu druhé části. Hlavní tři části, které nás z hlavičky zajímají jsou: reálný čas první zprávy, formát časových razítek (timestamp) zpráv a číselná soustava použítá pro kódování identifikátoru a datové části zprávy.

Druhá, datová část je rozdělena na jednotlivé události podle řádků. Je tedy pro nás jednoduché soubor rozdělit na řádky a jednotlivé řádky dekódovat zvlášť. CANgui podporuje dva typy událostí: zprávy a chybové rámce. Ostatní události CANgui ignoruje.

Každá individuální zpráva (nebo chybový rámec) je poslán do hlavního vlákna individuálně. Důvod pro toto rozhodnutí bylo takové, že kvůli tomu, že se všechna přenášená data (mezi vlákny) kopírují, tak přenos velkého počtu zpráv zastavoval hlavní vlákno aplikace a aplikace v ten moment nereagovala na uživatelovu interakci. Ke stejnému závěru došla i společnost Red Hat s aplikací Log Reaper, kdy navíc při přenosu vysokého počtu zpráv nastával problém s překročením limitu paměti zařízení [75]. Jedná se tedy o kompromis mezi výkonností a stabilitou algoritmu, spolu se zvýšenou responzivitou uživatelského rozhraní.

Pokud uživatel vybere, že chce zprávy ze souboru importovat, hlavní vlákno aplikace si přijaté zprávy z importovacího vlákna zprávy dočasně ukládá. Po tom, co hlavní vlákno dostane signál, že už jsme celý soubor zpracovali, vloží všechny zprávy do logu aplikace.

Replay funguje na podobné bázi. Hlavní vlákno si ukládá přijaté zprávy z vedlejšího vlákna, tentokrát ale tyto zprávy postupně odebírá a přidává je do logu průběžně. To uživateli dává dojem, jako by byl připojený přímo ke sběrnici. Zprávy do logu přidáváme v závislosti na zdrojových datech (časový rozdíl mezi jednotlivými zprávami) a na době renderování předchozího snímku. Pro každý snímek do logu přidáme pouze ty zprávy, které by reálně byly za ten časový okamžik přijaté. Pro uživatele je tedy nerozpoznatelné, zda zprávy přidáváme do logu po jedné (jako bychom je přijali) nebo ve skupinách.

8.3.3 Ukládání záznamu

Ukládání zpráv do souboru je o něco jednodušší, základní princip je ale podobný. Vytvoříme vedlejší vlákno pomocí WebWorker API [74], kterému předáme zprávy z logu. Jelikož serializace objektů se zdá být rychlejší jak deserializace, tak nám i při vysokém počtu zpráv (přes dva miliony) přišlo snesitelné předat všechny zprávy vedlejšímu vláknu najednou.

V první řadě je do souboru vložená statická hlavička, jež definuje jakým způsobem funguje naše implementace kódovacího algoritmu. Potom jsou postupně jednotlivé zprávy kódovány a vkládané do výsledného souboru, který následně uživateli nabídneme k uložení na disk.

Kapitola 9

Nasazení a použití aplikace

Naším cílem bylo udělat přenos a nasazení aplikace co nejjednodušší pro koncového uživatele, jelikož Ocarina Klient spouští uživatel na svém zařízení. Dále se počítá s případem, že uživatel bude potřebovat spustit také CANgui server i lokálně, například při testování nebo na závodech.

9.1 Nasazení

Aplikace používá pro sestavení (build) nástroj Maven (viz sekce 4.1.2). Spring Boot (viz sekce 4.1.3) automaticky při sestavení přibalí do Java Archive (JAR) i webový server Tomcat. Kromě webového serveru jsou Mavenem přibalené i ostatní závislosti projektu. Jelikož chceme distribuci aplikace udělat co nejsnazší, tak je do výsledného JAR souboru přibalená i sestavená verze uživatelského rozhraní. Tohoto dosahujeme pomocí dvou pluginů, jež se spouští při sestavení aplikace. První plugin se stará o samotné sestavení uživatelského rozhraní a druhý o zkopírování vygenerovaných souborů do výsledného JAR souboru.

Jelikož počítáme s případem, že uživatel bude chtít CANgui server spustit i lokálně, tak CANgui nabízí pro sestavení také Maven profil *local*. Pokud se aplikace sestaví s tímto profilem, aplikace nepoužívá autorizační server týmu eForce a je tedy použitelná i bez internetového připojení.

Sestavený JAR poté můžeme standardně spustit jako jakoukoliv jinou spustitelnou Java aplikaci (například z příkazové řádky jako `java -jar $NAZEV_APLIKACE.jar`). Při zapnutí aplikace spustí Spring Boot webový server na předem specifikovaném portu. V našem případě se jedná o port 8059 pro Ocarina Klient a port 10301 pro CANgui server (CANgui server navíc používá prefix */cangui*).

Tento způsob nasazení je i v současné době použit na veřejném serveru týmu eForce, odkud je aplikace CANgui dostupná.

9.2 Použití aplikace

Použití CANgui se dá rozdělit do několika skupin, dle požadavků daných uživatelů. Tyto požadavky se přímo odráží v případech užití aplikace (viz sekce 2.2) a nefunkčních požadavcích na aplikaci (viz sekce 2.1.2).

1. Pro základní použití aplikace (lze použít přehrání záznamu nebo připojit se k existující relaci) nemusí uživatel nic stahovat ani instalovat. Stačí se pouze přihlásit do aplikace skrz webový prohlížeč.
2. Pokud uživatel plánuje přímo komunikovat s CAN sběrnicí pomocí zařízení Ocarina IV, bude nutné aby si nejprve stáhl Ocarina Klient. Ocarina Klient uživateli umožní vytvořit novou relaci a poté se k ní pomocí webového prohlížeče standardně připojit.
3. Pokud uživatel potřebuje, aby mohl aplikaci používat i bez internetového připojení (v případě eForce například při testování nebo závodech), bude si uživatel muset stáhnout i webový server pro CANgui a v Ocarina Klientu zvolit, že chce komunikovat s lokálním webovým serverem, místo výchozího veřejného serveru.

Pro spuštění Ocarina Klientu nebo webového serveru CANgui, bude nutné, aby měl uživatel na svém systému nainstalovaný JRE 11. Uživatelské rozhraní počítá s použitím nejnovější verze prohlížeče Google Chrome nebo Mozilla Firefox. Kompatibilita s ostatními prohlížeči není zajištěna.

Kapitola 10
Evaluace řešení

Pro evaluaci použitelnosti nástroje CANgui jsme zvolili metodu uživatelského testování. Testování mělo kvalitativní a kvantitativní část, a probíhalo ve dnech 11.–12. 5. 2021. Testování bylo navrženo se dvěma hlavními cíli. Prvním cílem bylo zjistit, zda aplikace v současné době vyhovuje požadavkům týmu eForce a zda by pro ně aplikace byla přínosná. Druhým cílem bylo vypozorovat v čem lze aplikaci dále zlepšovat a získat od účastníků konkrétní náměty na další funkcionalitu.

Pro testování byli vybrání členové elektrické skupiny týmu eForce, jelikož se jedná o naší cílovou skupinu. Pro testování bylo zvoleno 6 účastníků s rozdílnými zkušenostmi s prací s CAN sběrnicí. V našem případě tyto zkušenosti i přímo odpovídají s dobou působení v týmu. Účastníky jsme tedy rozdělili do následujících kategorií:

- Nováčci – Nemají prakticky žádné zkušenosti (1 účastník)
- Seniorní členové – Mají základní až pokročilé zkušenosti (3 účastníci)
- Bývalí členové – Mají pokročilé zkušenosti i z prostředí mimo eForce (2 účastníci)

K tomuto rozložení účastníků jsme se rozhodli, jelikož očekáváme, že s aplikací budou převážně interagovat již zaběhlí členové týmu. Bývalí členové byli přizváni kvůli jejich cenným zkušenostem s touto problematikou z týmového i komerčního prostředí.

10.1 Průběh testování

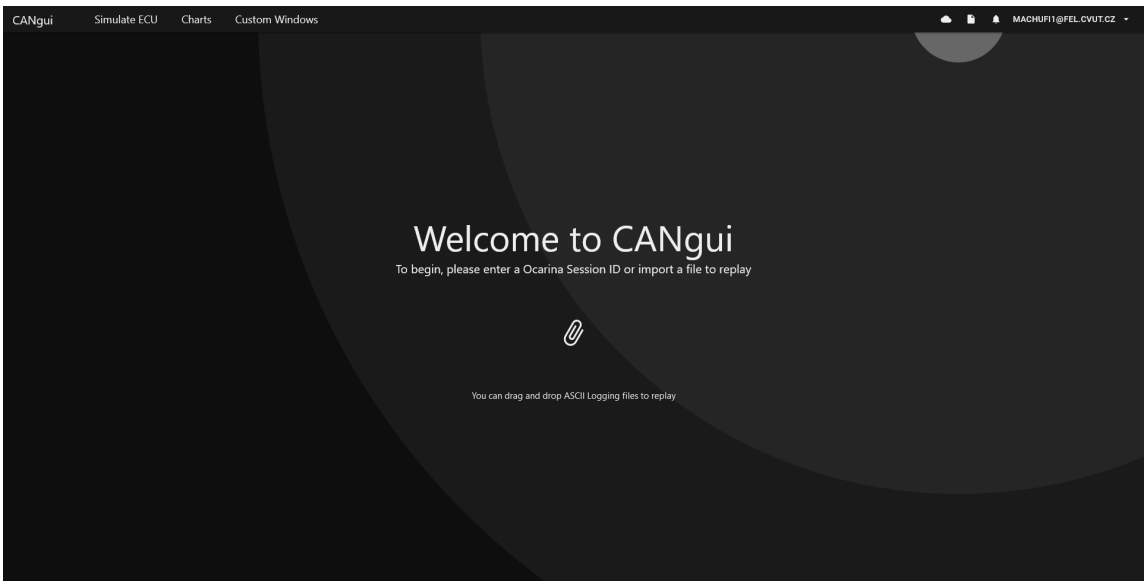
První část testování byla strukturovaná do série úkolů, které měli účastníci splnit. Úkoly byly záměrně strukturované více obecně, abychom zjistili, jak budou s aplikací interagovat uživatelé, kteří aplikací používají poprvé. Účastníci byli při plnění úkolů pod dohledem. Úkoly byly zvoleny tak, aby uživatel postupně prošel funkcionalitou celé aplikace.

1. Připojte se k zařízení Ocarina IV a nakonfigurujte jej pro loopback.
2. Vytvořte veřejnou Ocarina relaci.
3. Připojte se k vytvořené relaci.

4. Pošlete přes tuto relaci zprávu.
    - Tento úkol navíc testuje zda uživatel pochopí, že zatím nemá vybraný datový model CANdb a musí si ho nastavit
5. Vytvořte si okno s grafy a přidejte do něj alespoň 2 různé grafy.
6. Importujte do aplikace zprávy z .ASC souboru.
7. Podívejte se na signály ze souboru ve vámi přidaných grafech, v časovém rozmezí pěti minut
    - Konkrétní čas byl určen v závislosti na konkrétním souboru
    - Tento úkol navíc testoval, zda je intuitivní přibližování grafů µPlot
8. Vytvořte si vlastní okno a přidejte do něj alespoň 3 různé komponenty.
9. Přehrajte zprávy z .ASC souboru a vyzkoušejte si monitorování signálů pomocí vámi přidaných komponent.
10. Vyexportujte data z logu do .ASC souboru.

Z pozorování účastníků jsme zjistili, že většina úkolů pro ně byla relativně bezproblémová a přímočará, vyskytly se ale také části, ve kterých jsme vypozorovali opakované problémy.

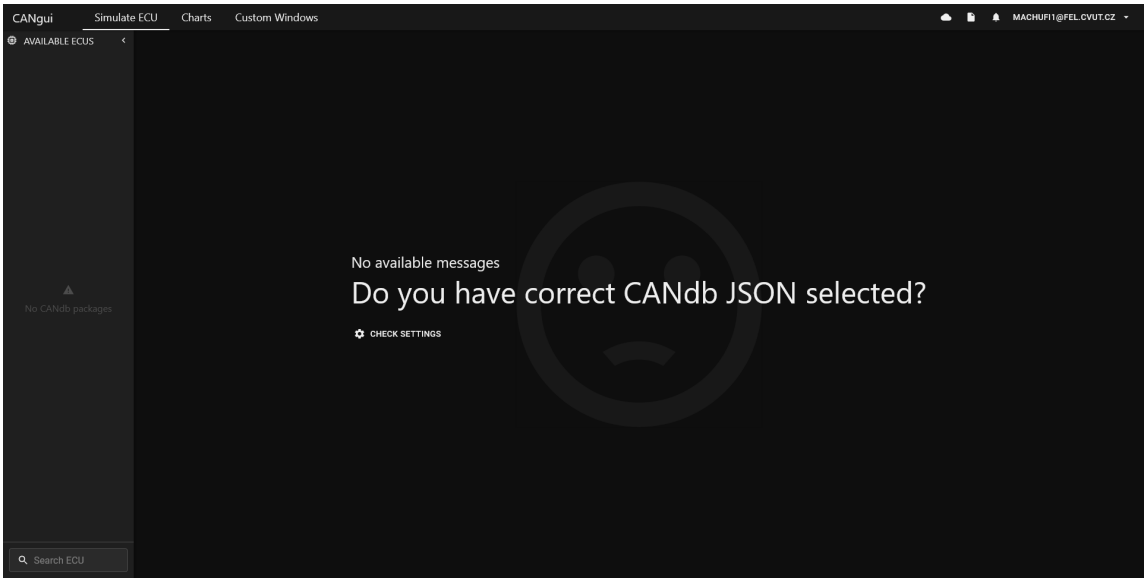
První problém, se kterým se až na jednoho účastníka potýkali všichni, bylo najít ikonu pro připojení se k vytvořené relaci. Každý účastník tuto ikonu eventuálně našel, bylo ale poznat, že to bylo pro účastníky složité a ne příliš intuitivní. Záměrně je tato ikona vyznačena na úvodní stránce pulzujícími kruhy (viz obrázek 10.1). Tyto kruhy některé účastníky správně nasměrovaly, bohužel samotná ikona pro ně ale nesignalizovala dostatečně, že je to funkcionalita, kterou hledají.



Obrázek 10.1: Úvodní obrazovka aplikace

Další úkol, který byl pro část účastníků problémový, bylo poslání zprávy na sběrnici. U tohoto úkolu jsme identifikovali dva problémy. Někteří účastníci nebyli vůbec schopni identifikovat stránku, která by tuto funkcionalitu měla umožňovat – zmátl je název. Druhý problém byl spojený s tím, že účastníci neměli nastavený datový model CANdb a nedošlo jim, že si ho musí v první řadě nastavit. Jeden z účastníků nevěděl, že takový model existuje, takže nerozuměl co po něm aplikace požaduje. To bylo zapříčiněno tím, že je v týmu nováček a zatím se s tímto modelem nesetkal při vývoji ECU.

Aplikace se snaží uživatele navézt k tomu, aby si zkontroloval nastavení, zda má nastavený správný datový model (viz obrázek 10.2). Účastníci, kteří měli s tímto úkolem problém si tohoto textu ale vůbec nevšimnuli. Překvapivě si ale všimnuli chybové hlášky v menu, které na ně oproti domovské stránce vyskočilo.



Obrázek 10.2: Problémová obrazovka pro odeslání zprávy

Z pozorování se žádné další problémové části už nevyskytnuly. Jednalo se spíše o individuální potíže, které účastník během pár vteřin vyřešil.

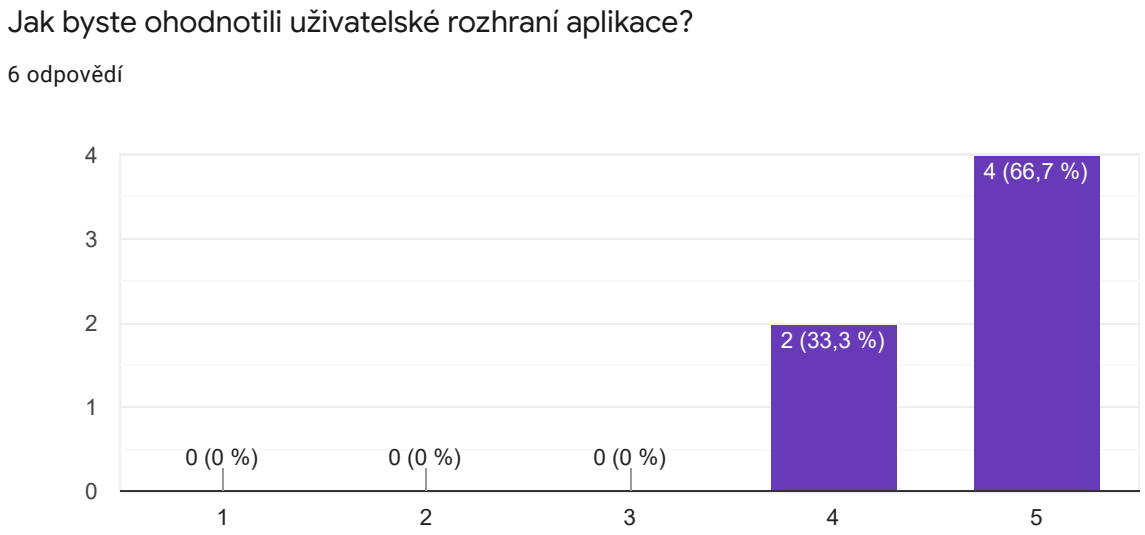
Bezprostředně po průchodu aplikací dostali účastníci dotazník[1] (viz příloha D). Tento dotazník lze rozdělit do dvou části. První část je zaměřena na ohodnocení aktuálního stavu aplikace pomocí kvantitativních otázek. Druhá část je zaměřená spíše kvalitativně, jelikož skrz odpovědi na tyto otázky se snažíme získat námět na další vývoj.

[1]Konkrétní výstupy z tohoto dotazníku lze vidět v příloze E.

10.2 Shrnutí

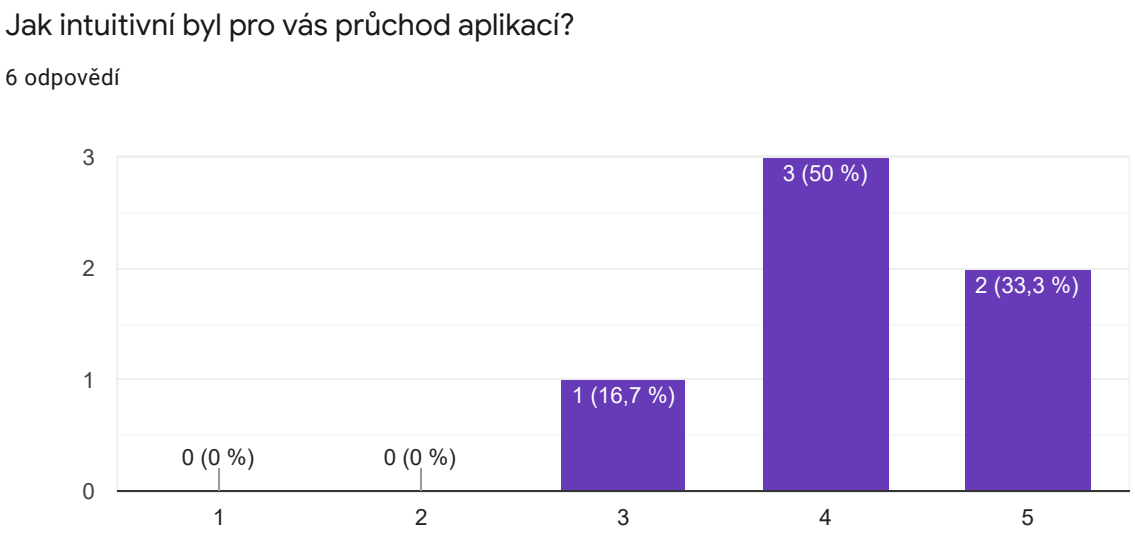
Z kvalitativní části testování jsme identifikovali, že vzniklé problémy většinou způsobovala nečekaná formulace popisků. Bylo ale zajímavé sledovat, jak rozdílný může být způsob, jak uživatel řeší situaci, kdy se v aplikaci ztratí. Někteří uživatelé začali po aplikaci zběsile klikat, takže si ani nečetli, kam se je aplikace snaží nasměrovat. Druhá skupina byla naopak velice konzervativní – ve smyslu, že se báli na cokoliv kliknout, ve strachu že aplikaci rozbijí. Na základě těchto pozorování budeme aplikaci dále upravovat a testovat, abychom podobné stavy mitigovali.

Výsledky kvantitativní části jsou v souladu s vypozorovaným chováním při plnění úkolů. Pro evaluaci tedy budeme používat hodnocení účastníků v následujících třech kategoriích. První kategorií je hodnocení uživatelského rozhraní aplikace. Hodnocení bylo v rozmezí 1 (velmi negativní) až 5 (velmi pozitivní). Z odpovědí (viz obrázek 10.3) vyplývá, že účastníci hodnotí uživatelské rozhraní pozitivně.



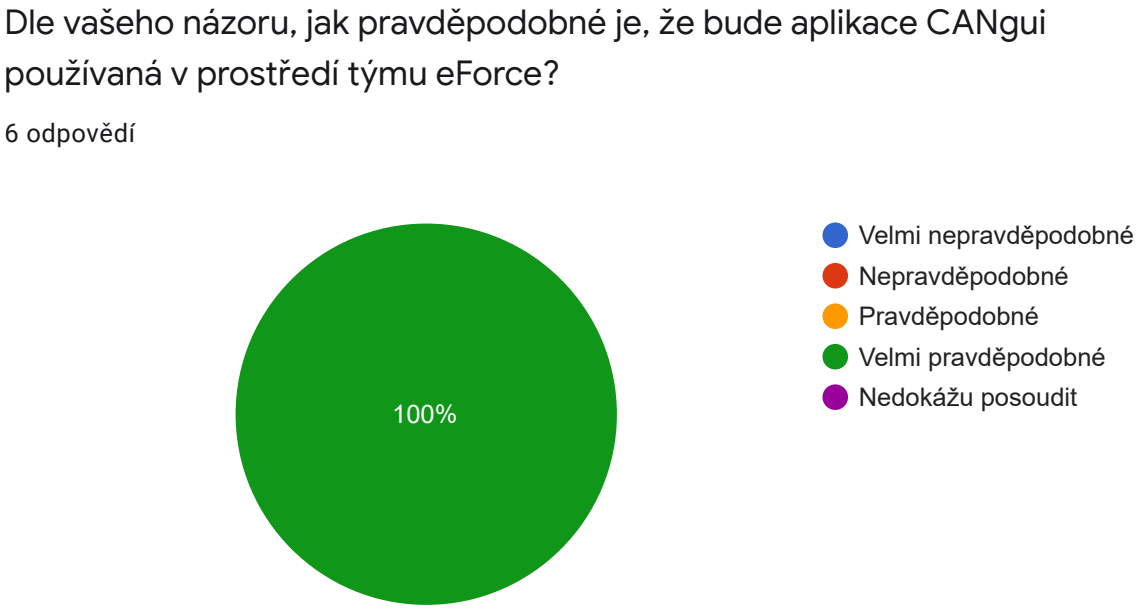
Obrázek 10.3: Hodnocení uživatelského rozhraní

Další kategorií bylo hodnocení intuitivnosti průchodu aplikací. Toto hodnocení bylo také v rozmezí 1 (průchod nebyl intuitivní) až 5 (průchod byl intuitivní). Výsledek hodnocení (viz obrázek 10.4) je stále pozitivní, oproti předchozí kategorii, ale převyšují hodnocení 'spíše pozitivní'. Toto hodnocení odpovídá naším očekáváním, jelikož se dost velká část účastníků setkala s úkolem, kde se museli na chvíli zastavit.



Obrázek 10.4: Hodnocení intuitivnosti

Poslední kategorií bylo hodnocení použitelnosti aplikace v prostředí týmu eForce. Výsledek tohoto hodnocení (viz obrázek 10.5) nás mile překvapil, jelikož všichni účastníci vnímají nástroj CANgui jako přínosný a myslí si, že bude velmi pravděpodobně používaný v týmovém prostředí.



Obrázek 10.5: Hodnocení použitelnosti řešení

Z výsledků hodnocení vyplývá, že je aplikace hodnocená velice pozitivně a bude používaná v týmu eForce. I přes to ale v některých částech lze uživatelský zážitek zlepšit, primárně lepší formulací popisků. Některé připomínky účastníků testování jsou již zapracované, ty složitější budeme řešit formou dalšího rozvoje.

Závěr

Hlavním cílem této práce bylo vytvořit robustní webovou aplikaci pro diagnostiku dat na CAN sběrnici. Tento cíl se nám rozhodně podařilo splnit v plném rozsahu všech dílčích cílů.

První dílčí cíl je zaměřený na teorii CAN sběrnice, jako cílového média pro přenos dat ve vozidlech týmu eForce FEE Prague Formula. Tato část textu definuje základní vlastnosti této sběrnice, podle norem ISO 11898-1 a ISO 11898-2, které odpovídají prvním dvěma vrstvám ISO/OSI modelu.

Druhý, analytický dílčí cíl se dělí na dvě části. V první části jsme provedli sběr požadavků a případů užití pro tým eForce. V druhé části jsme udělali analýzu existujících řešení, které jsme vyhodnotili vůči definovaným požadavkům. Z výsledku této analýzy jsme došli k závěru, že pro potřeby týmu eForce bude nejlepší vyvinout vlastní řešení, oproti pořízení licencí pro některé z existujících komerčních řešení.

Třetí dílčí cíl se zabývá návrhem a vývojem tohoto řešení. To znamená, že jsme v první řadě určili technologie, které bude CANgui používat. Dále jsme popsali datový model celé aplikace a v neposlední řadě jsme vytvořili návrh architektury aplikace.

V implementační části jsme se zaměřili na popis uživatelského rozhraní aplikace a implementační detaily specifické pro vývoj CANgui. Kromě toho jsme popsali i způsob nasazení a použití aplikace.

Poslední, čtvrtý cíl se týká evaluace řešení. Pro evaluaci jsme použili metody uživatelského testování. Z této evaluace plyne, že je CANgui pro tým eForce velký přínos a že bude nástroj používaný v průběhu celé sezóny (to znamená od vývoje ECU, až po samotné závody po celé Evropě).

Nástroj CANgui bude nadále vyvíjen v prostředí týmu eForce. Mezi příležitosti k dalšímu rozvoji patří zejména nové požadavky, jež vznikly z uživatelského testování. Jmenovitě se jedná o přeskakování při přehrávání ze souboru nebo vkládání více signálů do jednoho grafu. Mezitím budeme zjišťovat jak dále zlepšovat uživatelský zážitek a jak udělat aplikaci ještě přístupnější pro nové členy a uživatele.

Literatura

[1] KRUŽLIAK, Andrej a Miroslav MARTÍNEK. *Tým eForce FEE Prague Formula* [fotografie]. Praha: Strahov, 2020. Formát: 5016 x 3344

[2] RANGAM, Kartik. What Is An ECU? Electronic Control Unit (ECU) Explained. In: *The GoMechanic Blog* [online]. GoMechanic, October 4, 2020. [cit. 2021-05-21]. Dostupné z: https://gomechanic.in/blog/ecu-electronic-control-unit-explained/

[3] KINGSTON, Lewis. What is an Electronic Control Unit? PH Explains. In: *PistonHeads UK | Cars for Sale | Car News | Motoring Forum* [online]. London, United Kingdom: Pistonheads Holdco Limited, Tuesday, March 27, 2018. [cit. 2021-05-21]. Dostupné z: https://www.pistonheads.com/news/ph-features/what-is-an-electronic-control-unit-ph-explains/37771

[4] PARIKH, Bijal. CAN Protocol: Understanding the Controller Area Network Protocol. In: *Engineers Garage* [online]. Cleveland, Ohio, United States: WTWH Media, 2021. [cit. 2021-01-08]. Dostupné z: https://www.engineersgarage.com/article_page/can-protocol-understanding-the-controller-area-network-protocol/

[5] CANopen Explained - A Simple Intro (2020). In: *CAN bus data loggers - Simple. Pro. Interoperable.* [online]. Aabyhoej, Denmark: CSS Electronics, 2020. [cit. 2021-02-18]. Dostupné z: https://www.csselectronics.com/screen/page/canopen-tutorial-simple-intro

[6] Medical. In: *Kvaser | Advanced CAN Solutions* [online]. Mission Viejo, California, United States: Kvaser Inc. [cit. 2021-02-18]. Dostupné z: https://www.kvaser.com/about-us/applications/medical/

[7] CAN FD Explained - A Simple Intro (2021). In: *CAN bus data loggers - Simple. Pro. Interoperable.* [online]. Aabyhoej, Denmark: CSS Electronics, 2021. [cit. 2021-05-21]. Dostupné z: https://www.csselectronics.com/screen/page/can-fd-flexible-data-rate-intro/language/en

[8] CAN FD - The basic idea. In: *CAN in Automation (CiA): Controller Area Network (CAN)* [online]. Nuremberg, Germany: CAN in Automation (CiA) [cit. 2021-05-21]. Dostupné z: https://www.can-cia.org/can-knowledge/can/can-fd/

[9] ISO 11898-1:2015. *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling.* ICS 43.040.15. 2nd ed. December 2015. International Organization for Standardization.

[10] ISO 11898-2:2016. *Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit.* ICS 43.040.15. 2nd ed. December 2016. International Organization for Standardization.

[11] The OSI Model - Features, Principles and Layers. In: *Studytonight - Best place to Learn Coding Online* [online]. Studytonight Technologies Pvt. Ltd., 2021. [cit. 2021-05-20]. Dostupné z: https://www.studytonight.com/computer-networks/complete-osi-model

[12] Layers of OSI Model. In: *GeeksforGeeks | A computer science portal for geeks* [online]. Noida, India: GeeksforGeeks, 04 Feb, 2020. [cit. 2021-05-20]. Dostupné z: https://www.geeksforgeeks.org/layers-of-osi-model/

[13] RICHARDS, Pat. *Understanding Microchip's CAN Module Bit Timing* [online]. Chandler, Arizona, United States: Microchip Technology Inc, 2001. [cit. 2021-02-19]. DS00754A. Dostupné z: http://ww1.microchip.com/downloads/en/AppNotes/00754.pdf

[14] VOSS, Wilfried. CAN Bus Guide – Message Frame Format. In: *Copperhill Technologies - SAE J1939 & CAN Bus Prototyping* [online]. Greenfield, Massachusetts, United States: Copperhill Technologies, January 27, 2017. [cit. 2021-03-10]. Dostupné z: http://www.copperhilltechnologies.com/can-bus-guide-message-frame-format/

[15] VOSS, Wilfried. CAN Bus Guide – Extended CAN Protocol. In: *Copperhill Technologies - SAE J1939 & CAN Bus Prototyping* [online]. Greenfield, Massachusetts, United States: Copperhill Technologies, January 30, 2017. [cit. 2021-03-10]. Dostupné z: http://www.copperhilltechnologies.com/can-bus-guide-extended-can-protocol/

[16] CAN data link layers in some detail. In: *CAN in Automation (CiA): Controller Area Network (CAN)* [online]. Nuremberg, Germany: CAN in Automation (CiA) [cit. 2021-03-10]. Dostupné z: https://www.can-cia.org/can-knowledge/can/can-data-link-layers/

[17] WEBERMANN, Hauke a Andreas BLOCK. *CAN Error Injection, a Simple but Versatile Approach* [online]. Hanover, Germany: esd electronics gmbh, 2012. [cit. 2021-03-25]. Dostupné z: https://esd.eu/sites/default/files/CAN_Error_Injection.pdf

[18] ALI, Adam. What is Vehicle CAN bus and why do you need to care. In: *Digital Transformation and Cyber Resilience | Earth2 Digital* [online]. Rhodes, NSW, Australia: Earth2 Software Pty Ltd, March 02, 2019. [cit. 2021-05-20]. Dostupné z: https://www.earth2.digital/blog/what-is-vehicle-can-bus-ecu-evoque-adam-ali.html

[19] CANrunner. In: *Software. Electronis. Inovation. | Wapice Ltd* [online]. Vaasa, Finland: Wapice Ltd, 2021. [cit. 2021-05-20]. Dostupné z: https://www.wapice.com/products/canrunner

[20] CANalyzer. In: *Vector* [online]. Regensburg, Germany: Vector Informatik GmbH, 2021. [cit. 2021-05-20]. Dostupné z: https://www.vector.com/cz/en/products/products-a-z/software/canalyzer/

[21] HENNESSY, Bryan. An Introduction to J1939 and DBC files. In: *Kvaser | Advanced CAN Solutions* [online]. Mission Viejo, California, United States: Kvaser Inc, September 13, 2019. [cit. 2021-05-20]. Dostupné z: https://www.kvaser.com/developer-blog/an-introduction-j1939-and-dbc-files/

[22] CANoe. In: *Vector* [online]. Regensburg, Germany: Vector Informatik GmbH, 2021. [cit. 2021-05-20]. Dostupné z: https://www.vector.com/cz/en/products/products-a-z/software/canoe/

[23] What is Java?: A Beginner's Guide to Java and Its Evolution. In: *Edureka* [online]. Bengaluru, Karnataka, India: Brain4ce Education Solutions Pvt., 2021. [cit. 2021-01-08]. Dostupné z: https://www.edureka.co/blog/what-is-java/

[24] Oracle Buys Sun. In: *Oracle: Integrated Cloud Applications and Platform Services* [online]. Redwood Shores, California, United States: Oracle, Apr 20, 2009. [cit. 2021-01-08]. Dostupné z: https://www.oracle.com/corporate/pressrelease/oracle-buys-sun-042009.html

[25] Java is Still Free 2.0.3. In: *Medium: Java Champions* [online]. San Francisco, California, United States: A Medium Corporation, Mar 3, 2019. [cit. 2021-01-08]. Dostupné z: https://medium.com/@javachampions/java-is-still-free-2-0-0-6b9aa8d6d244

[26] From Java 8 to Java 11. In: *Codete: Software Development Company* [online]. Kraków, Poland: Codete, July 3, 2018. [cit. 2021-01-08]. Dostupné z: https://codete.com/blog/java-8-java-11-quick-guide/

[27] Maven: Introduction. In: *Apache Maven Project* [online]. Forest Hill, Maryland, United States: The Apache Software Foundation, 2020. [cit. 2021-01-08]. Dostupné z: https://maven.apache.org/what-is-maven.html

[28] Maven: Introduction to the POM. In: *Apache Maven Project* [online]. Forest Hill, Maryland, United States: The Apache Software Foundation, 2020. [cit. 2021-01-08]. Dostupné z: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

[29] VMware Completes Acquisition of Pivotal. In: *VMware: Delivering a Digital Foundation For Businesses* [online]. Palo Alto, California, United States: VMware, 2019. [cit. 2021-01-08]. Dostupné z: https://www.vmware.com/company/news/releases/vmw-newsfeed.VMware-Completes-Acquisition-of-Pivotal.3b73174e-4485-4ff9-8c5d-56c54de6db86.html

[30] *Spring Boot* [online]. Palo Alto, California, United States: VMware, 2021. [cit. 2021-01-08]. Dostupné z: https://spring.io/projects/spring-boot

[31] *jSerialComm* [online]. Nashville, Tennessee, United States: Fazecast, Inc., 2021 [cit. 2021-01-08]. Dostupné z: https://fazecast.github.io/jSerialComm/

[32] What is HTML. In: *W3Schools Online Web Tutorials* [online]. Sandnes, Norway: Refsnes Data, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.com/whatis/whatis_html.asp

[33] HTML5. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla, 2021. [cit. 2021-01-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[34] What is CSS. In: *W3Schools Online Web Tutorials* [online]. Sandnes, Norway: Refsnes Data, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.com/whatis/whatis_css.asp

[35] Sass Basics: Sass. In: *Sass: Syntactically Awesome Style Sheets* [online]. Sass. [cit. 2021-01-08]. Dostupné z: https://sass-lang.com/guide

[36] Introduction to the DOM. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Apr 9, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[37] What is JavaScript? In: *W3Schools Online Web Tutorials* [online]. Sandnes, Norway: Refsnes Data, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.com/whatis/whatis_js.asp

[38] What is AJAX? In: *W3Schools Online Web Tutorials* [online]. Sandnes, Norway: Refsnes Data, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.com/whatis/whatis_ajax.asp

[39] *React: A JavaScript library for building user interfaces* [online]. Menlo Park, California, United States: Facebook, 2021. [cit. 2021-01-08]. Dostupné z: https://reactjs.org/

[40] Create a New React App. In: *React: A JavaScript library for building user interfaces* [online]. Menlo Park, California, United States: Facebook, 2021. [cit. 2021-01-08]. Dostupné z: https://reactjs.org/docs/create-a-new-react-app.html

[41] *Material-UI: A popular React UI framework* [online]. Paris, France: Material-UI SAS, 2021. [cit. 2021-05-17]. Dostupné z: http://material-ui.com/

[42] *Material Design* [online]. Mountain View, California, United States: Google, 2021. [cit. 2021-05-17]. Dostupné z: https://material.io/

[43] SOROKIN, Leon. µPlot. In: *GitHub* [online]. San Francisco, California, United States: GitHub, Inc., 2021. [cit. 2021-05-17]. Dostupné z: https://github.com/leeoniya/uPlot

[44] *Recharts* [online]. Recharts Group, 2020. [cit. 2021-01-08]. Dostupné z: http://recharts.org/en-US

[45] WebGL: 2D and 3D graphics for the web. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Mar 3, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

[46] WebGL Overview. In: *The Khronos Group Inc* [online]. Beaverton, Oregon, United States: The Khronos® Group Inc., 2021 [cit. 2021-05-20]. Dostupné z: https://www.khronos.org/webgl/

[47] HTTP. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Mar 16, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP

[48] HTTP request methods. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Dec 4, 2020. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[49] HTTP response status codes. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, May 19, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

[50] *websocket.org* [online]. Tenefit Corporation. [cit. 2021-05-20]. Dostupné z: https://www.websocket.org/

[51] Window.localStorage. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Apr 24, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[52] STOMP Protocol Specification, Version 1.2. In: *STOMP* [online]. [cit. 2021-05-17]. Stomp Spec Group, 10/22/2012. Dostupné z: https://stomp.github.io/stomp-specification-1.2.html

[53] STOMP. In: *Web on Servlet Stack* [online]. Palo Alto, California, United States: VMware, 2021-05-12. [cit. 2021-05-20]. Dostupné z: https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#websocket-stomp

[54] What is Pub/Sub Messaging? In: *Amazon Web Services (AWS) - Cloud Computing Services* [online]. Seattle, Washington, United States: Amazon Web Services 2021. [cit. 2021-05-17]. Dostupné z: https://aws.amazon.com/pub-sub-messaging/

[55] CEJP, Martin. *Communication protocol database (CANdb)*. Design report. Praha: eForce FEE Prague Formula, ČVUT v Praze, 2019.

[56] Window.localStorage. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Apr 24, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

[57] Client Server Architecture. In: *CIO Wiki* [online]. CIO Wiki, 6 February 2021 [cit. 2021-05-20]. Dostupné z: https://cio-wiki.org/wiki/Client_Server_Architecture

[58] BANGER, Er R S. What is Client Server Architecture: Diagram, Types, Examples, Components. In: *Digital Thinker Help* [online]. DigitalThinkerHelp, September 9, 2020. [cit. 2021-05-20]. Dostupné z: https://digitalthinkerhelp.com/what-is-client-server-architecture-diagram-types-examples-components/

[59] RESTful Web Services. In: *W3schools Online Programming Tutorials* [online]. Indore, India: W3schools, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.in/restful-web-services/intro/

[60] REST Methods. In: *W3schools Online Programming Tutorials* [online]. Indore, India: W3schools, 2021. [cit. 2021-01-08]. Dostupné z: https://www.w3schools.in/restful-web-services/rest-methods/

[61] SPA (Single-page application). In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, Apr 24, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/SPA

[62] Single-page application vs. multiple-page application. In: *Neoteric – Medium* [online]. San Francisco, California, United States: A Medium Corporation, Sep 16, 2020. [cit. 2021-05-20]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[63] GIMON, Zee. What is a Single Page Application? In: *HUSPI* [online]. Rzeszów, Poland: Huspi, 2019. [cit. 2021-05-20]. Dostupné z: https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications

[64] Adobe Inc. *Adobe XD* [software]. Duben 2021. [přístup 2021-05-20]. Dostupné z: https://www.adobe.com/products/xd.html [Požadavky na systém: Operační systém macOS X v10.14 nebo novější, Windows 10 (64-bit) – Version 1809 (build 10.0.17763) nebo novější, displej 13 palců nebo větší při rozlišení alespoň 1280x800 pro platformu Windows a 1400x900 pro platformu macOS, operační paměť 4 GB, grafická karta s podporou alespoň Direct 3D DDI Feature Set: 10, grafická karta Intel s ovladačem z roku 2014 nebo novější]

[65] How Does Single Sign-On (SSO) Work? In: *OneLogin* [online]. San Francisco, California, United States: OneLogin, Inc., 2021. [cit. 2021-05-20]. Dostupné z: https://www.onelogin.com/learn/how-single-sign-on-works

[66] BACHAN, Patrik. *Ocarina IV*. Doprovodná dokumentace. Praha: eForce FEE Prague Formula, ČVUT v Praze, 2019.

[67] BACHAN, Patrik. *Ocarina Protocol v3*. Praha: eForce FEE Prague Formula, ČVUT v Praze, 2019.

[68] BACHAN, Patrik. *Ocarina IV* [fotografie]. Praha: ČVUT v Praze, Fakulta elektrotechnická, 2019. 1500 x 937

[69] VN7610 - FlexRay/CAN FD Interface for USB. In: *Vector* [online]. Regensburg, Germany: Vector Informatik GmbH, 2021. [cit. 2021-05-20]. Dostupné z: https://www.vector.com/cz/en/products/products-a-z/hardware/network-interfaces/vn7610/

[70] Logging Formats. In: *Vector KnowledgeBase* [online]. Regensburg, Germany: Vector Informatik GmbH, 2021. [cit. 2021-05-17]. Dostupné z: https://support.vector.com/kb?id=kb_article_view&sysparm_article=KB0011536

[71] THORNE, Brian. python-can. In: *GitHub* [online]. San Francisco, California, United States: GitHub, Inc., 2020. [cit. 2021-05-20]. Dostupné z: https://github.com/hardbyte/python-can/tree/master

[72] ASAM MDF. In: *Association for Standardization of Automation and Measuring Systems* [online]. Höhenkirchen, Germany: ASAM e. V., 2021. [cit. 2021-05-20]. Dostupné z: https://www.asam.net/standards/detail/mdf/wiki/

[73] Using Web Workers. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, May 4, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers

[74] Web Workers API. In: *MDN Web Docs* [online]. Mountain View, California, United States: Mozilla Corporation, May 7, 2021. [cit. 2021-05-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

[75] Communicating Large Objects with Web Workers in javascript. In: *Red Hat Developer* [online]. Raleigh, North Carolina, United States: Red Hat Software, 2014. [cit. 2021-05-17]. Dostupné z: https://developers.redhat.com/blog/2014/05/20/communicating-large-objects-with-web-workers-in-javascript

Rejstřík zkratek

.ASC ASCII Logging Files. 14, 53, 58

.MDF Measurement Data Format. 14

AJAX Asynchronous JavaScript and XML. 28, 40

API Application Programming Interface. vii, 28, 29, 32, 34, 40, 51, 54

CAN Controller Area Network. vi, vii, ix, xi, 2, 5–8, 13, 17, 18, 21, 29, 31–34, 37–39, 42, 47, 49–52, 56, 57, 63, 105, 107, 110, 112, 113

CAN FD Controller Area Network Flexible Data-Rate. 5, 17–19

CRA Create React App. 28

CSS Cascading Style Sheets. vii, 27

DFU Device Firmware Upgrade. viii, xi, 109

DOM Domain Object Model. 28

ECU Electronic Control Unit. vi, ix, 2, 5, 13–15, 18, 31, 32, 34, 51, 59, 63, 88, 89

HTML Hypertext Markup Language. vii, 27, 40

HTTP Hypertext Transfer Protocol. vii, 29

HW Hardware. 104

IN Input. 110

JAR Java Archive. 26, 55

Java SE Java Platform, Standard Edition. vii, 25, 26

JRE Java Runtime Environment. 56

JSON JavaScript Object Notation. vii, ix, 14, 32, 33

JVM Java Virtual Machine. 25

LTS Long Time Support. 25, 26

OUT Output. 110

POM Project Object Model. 26

REST Representational state transfer. vii, 29, 40

RX Receive. 108–110

Sass Syntactically Awesome Style Sheets. vii, 27

SPA Single-page application. vii, 40

SSO Single sign-on. 47

STOMP Simple (or Streaming) Text Oriented Message Protocol. vii, 29, 39, 49, 50

SW Software. 104

TCP Transmission Control Protocol. 29

TCP/IP Internet protocol suite. 29

TLV Type-length-value. 51

TX Transmit. 108–110

USB Universal Serial Bus. 5, 110

XML Extensible Markup Language. 26, 40

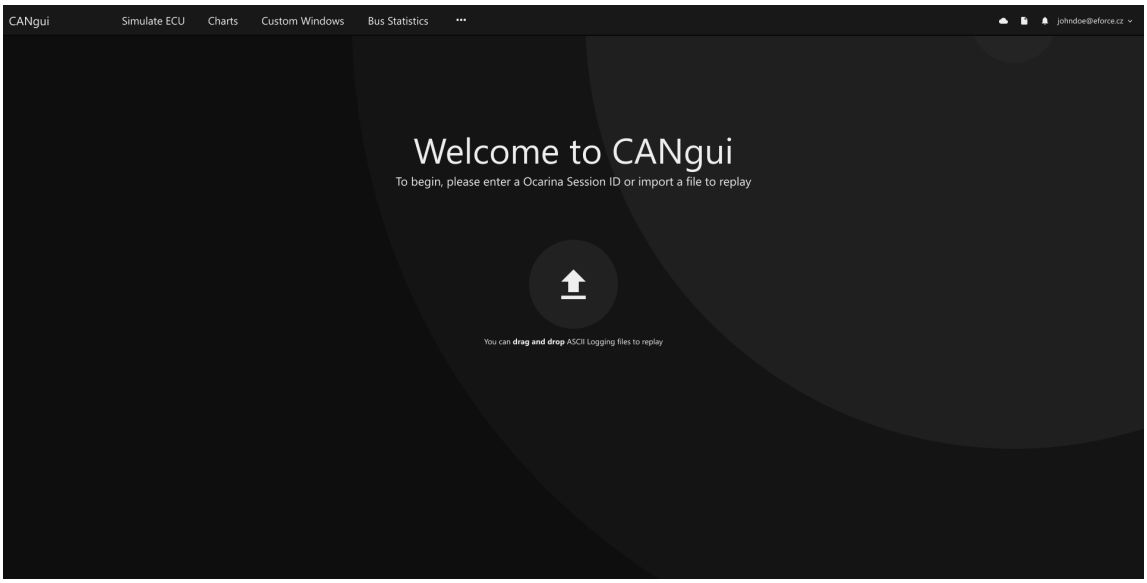
Přílohy

Příloha A

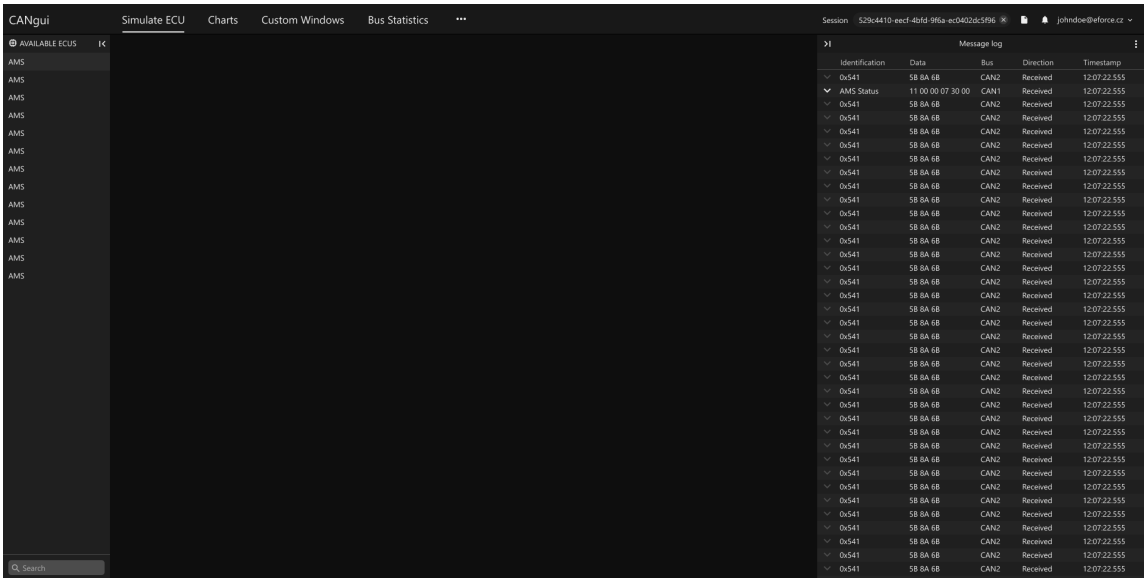
Prototyp CANgui

Prototyp nástroje CANgui byl vytvořen v aplikaci Adobe XD [64]. Tento prototyp byl před implementací iterován, jak je popsáno v kapitole 7.

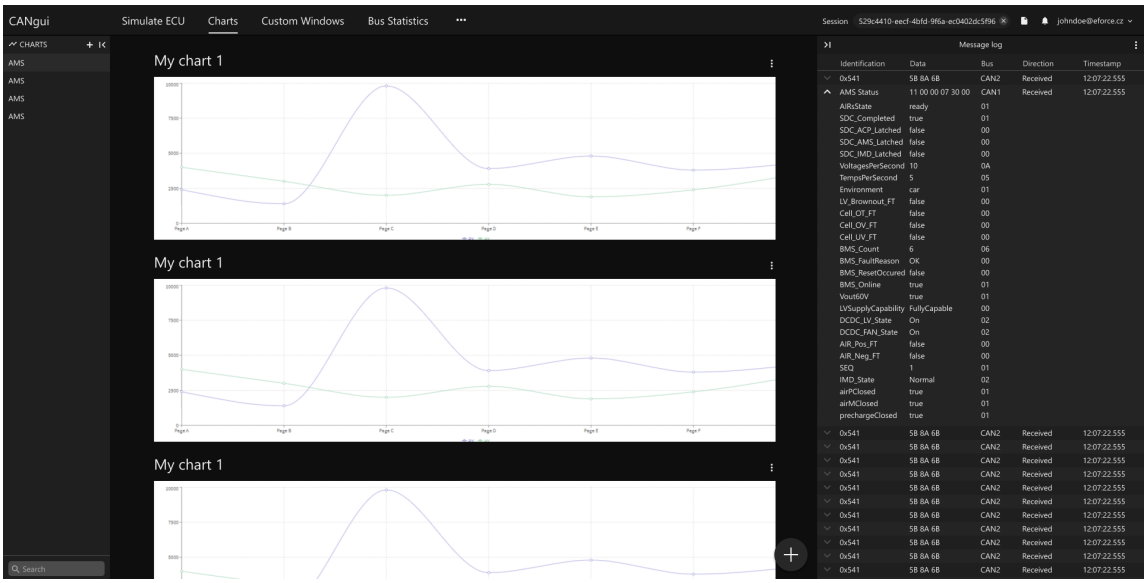
Obrázek A.1: Prototyp CANgui – Přihlašovací obrazovka



Obrázek A.2: Prototyp CANgui – Úvodní obrazovka



Obrázek A.3: Prototyp CANgui – Rozložení stránek aplikace



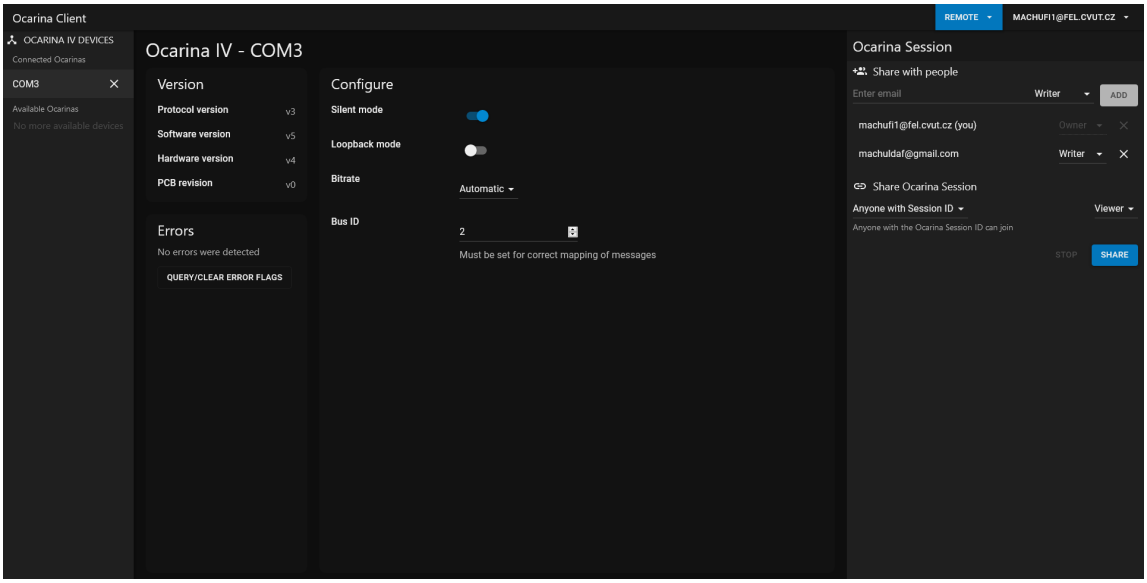
Obrázek A.4: Prototyp CANgui – Rozložení obrazovky s grafy, včetně ovládacích prvků

Příloha B

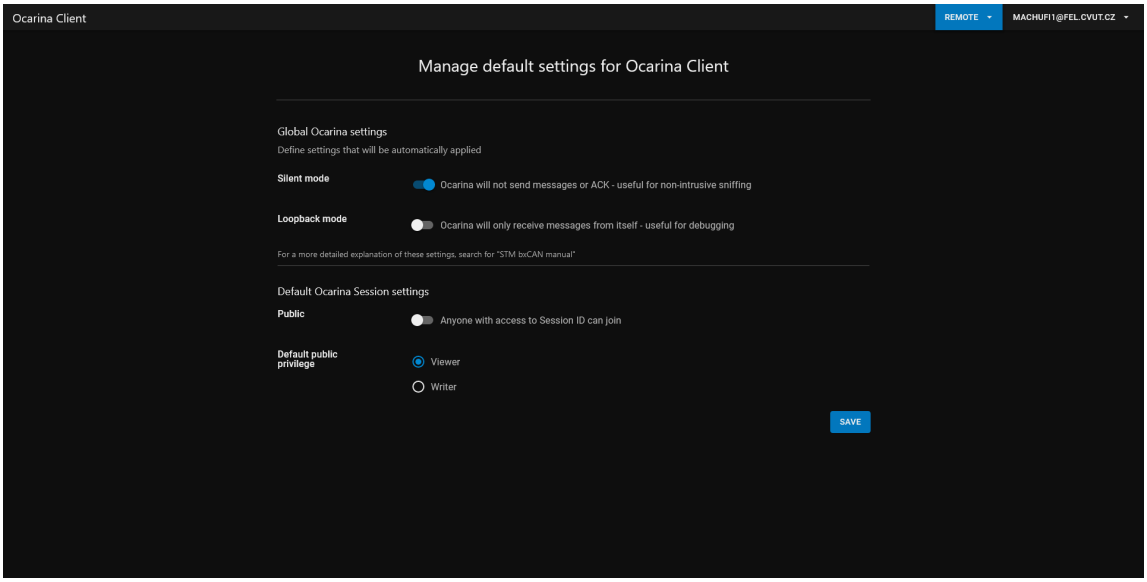
Snímky z obrazovek

Následuje znázornění snímků obrazovek, obou implementovaných částí aplikace.

B.1 Ocarina Klient



Obrázek B.1: Ocarina Klient – Dashboard

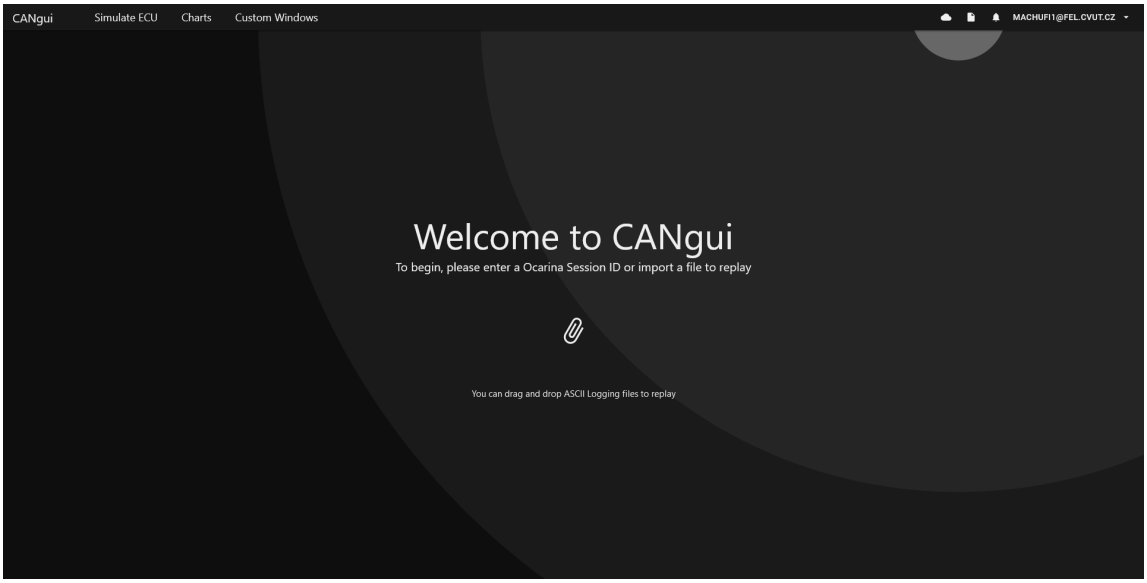


Obrázek B.2: Ocarina Klient – Nastavení výchozích hodnot

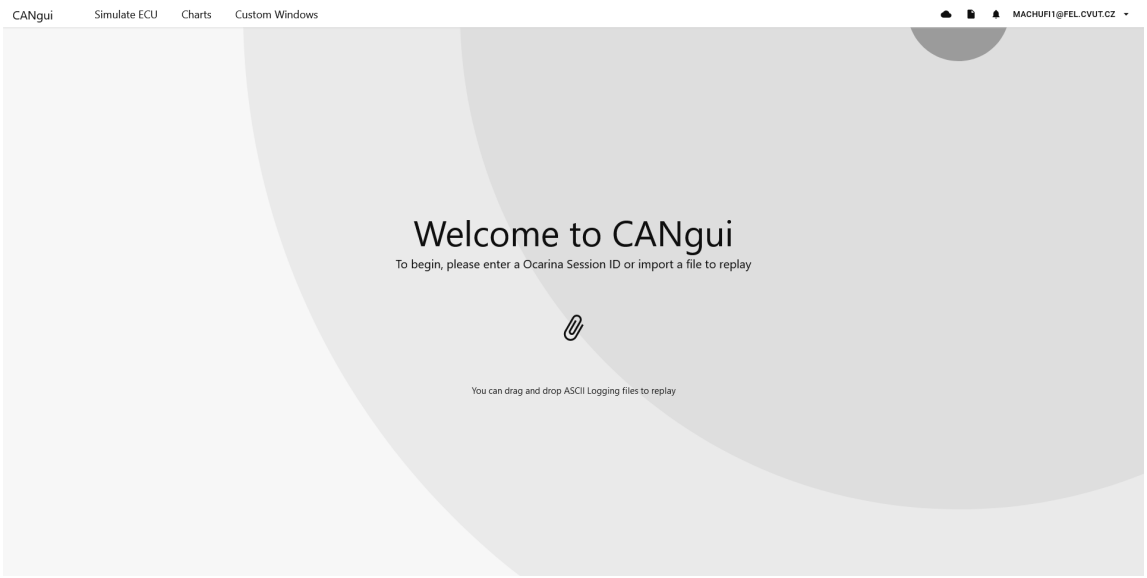
B.2 CANgui

Obrázek B.3: CANgui – Přihlašovací obrazovka (Tmavý režim)

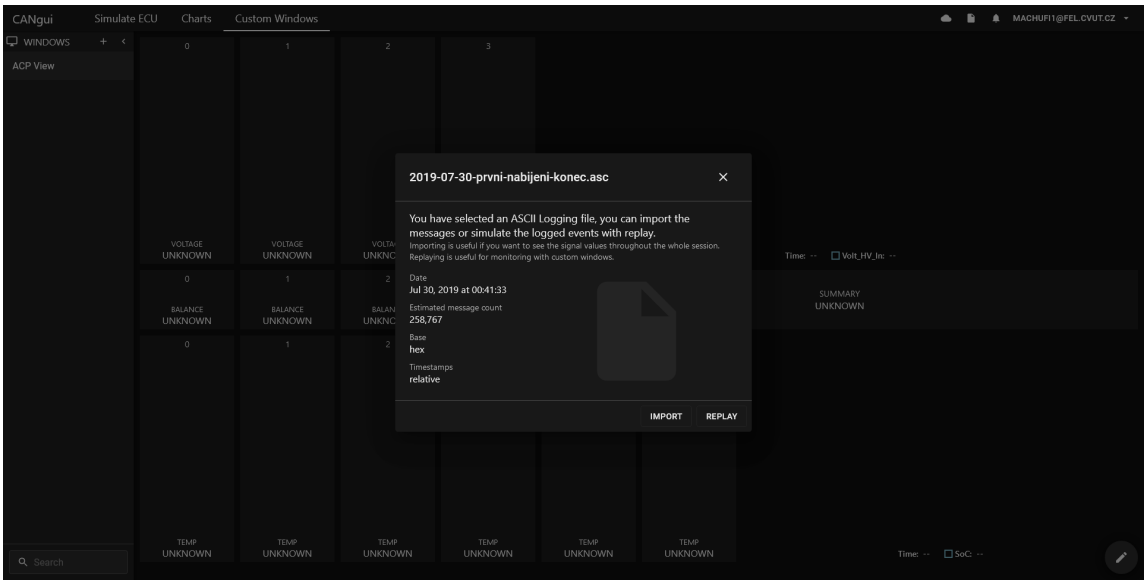
Obrázek B.4: CANgui – Přihlašovací obrazovka (Světlý režim)



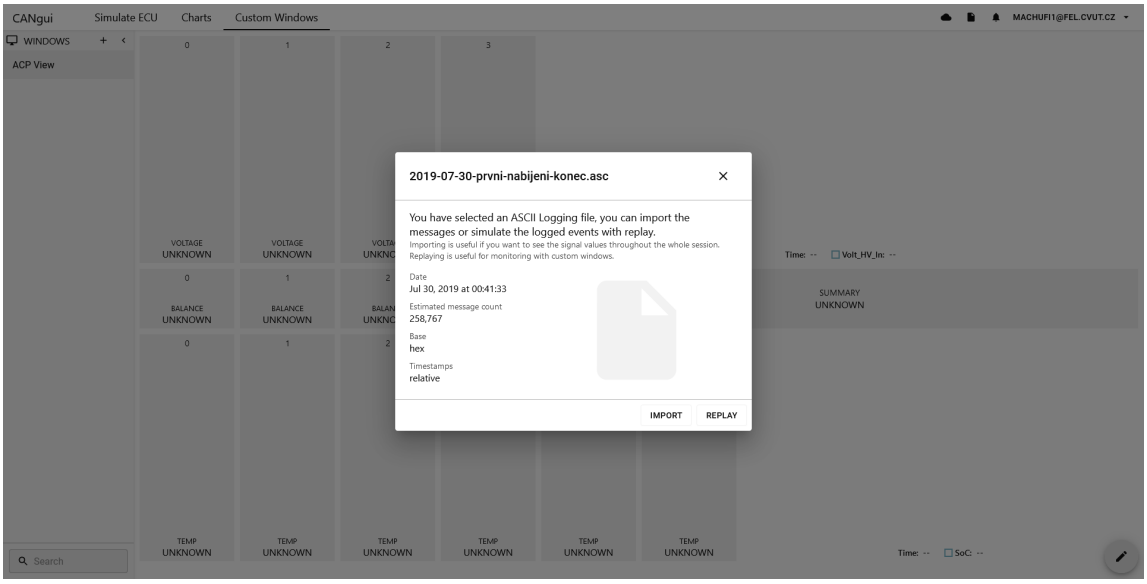
Obrázek B.5: CANgui – Úvodní obrazovka po přihlášení (Tmavý režim)



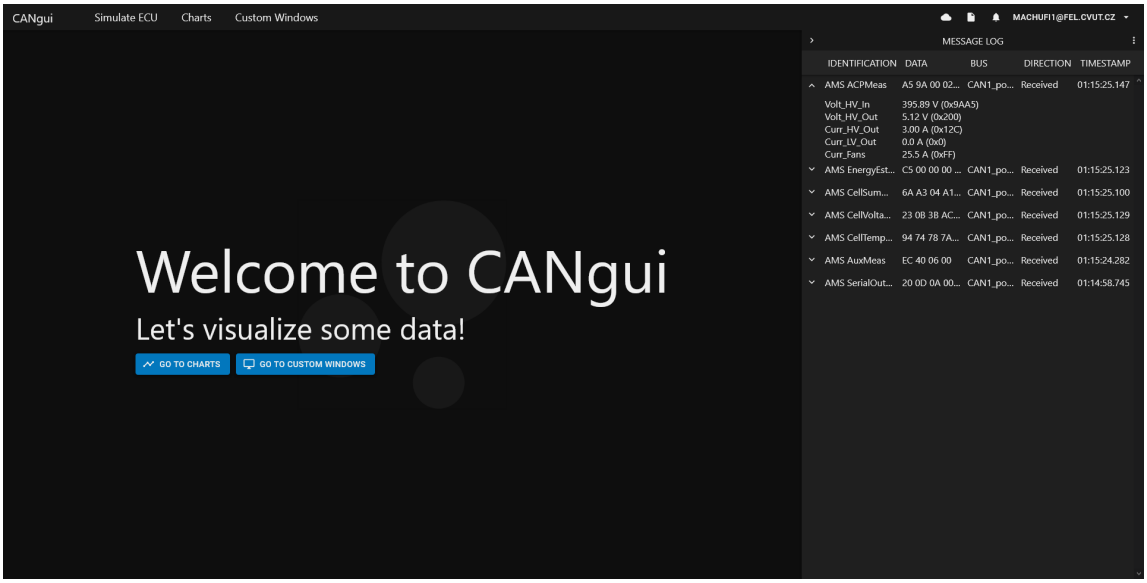
Obrázek B.6: CANgui – Úvodní obrazovka po přihlášení (Světlý režim)



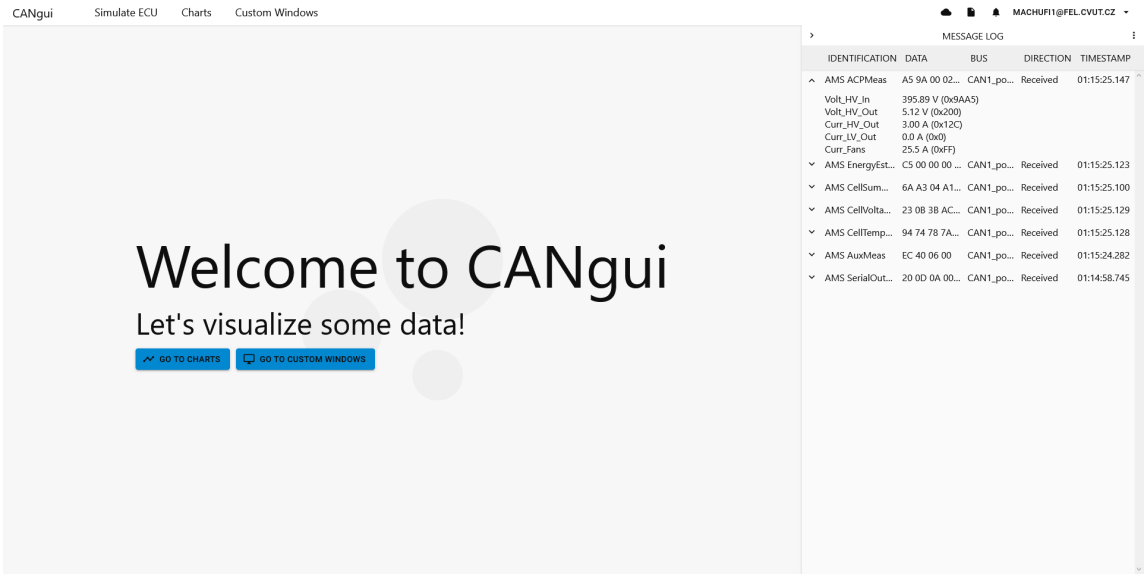
Obrázek B.7: CANgui – Import dialog (Tmavý režim)



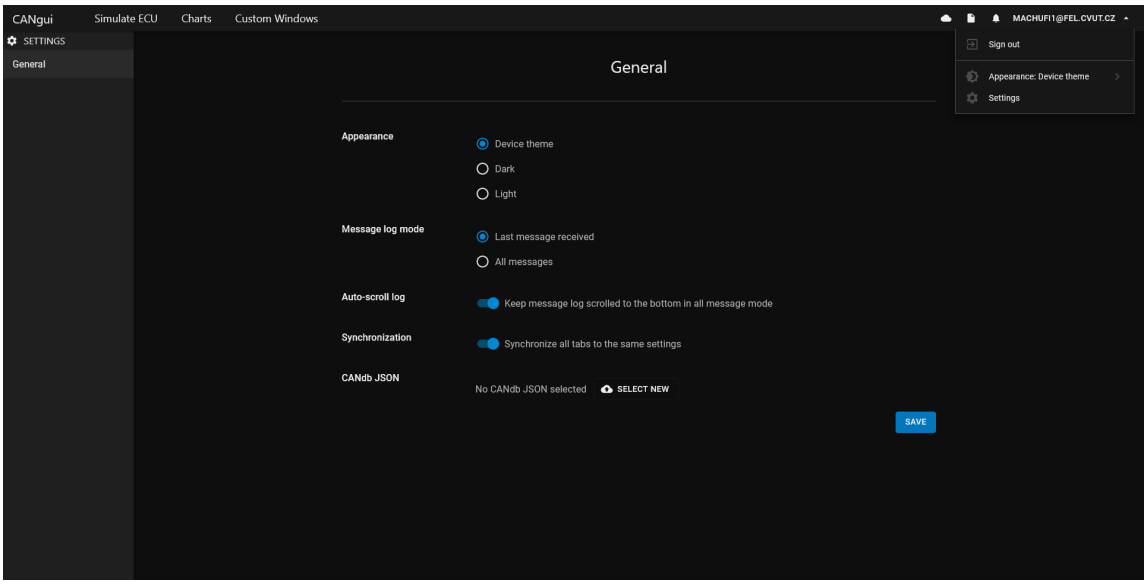
Obrázek B.8: CANgui – Import dialog (Světlý režim)



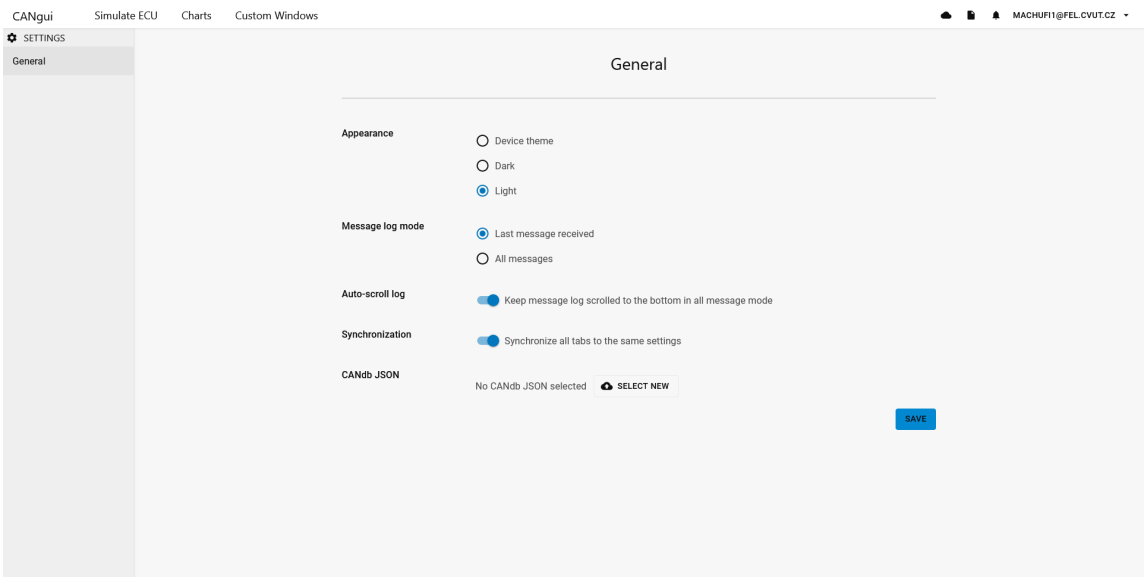
Obrázek B.9: CANgui – Úvodní obrazovka po vybrání zdroje dat (Tmavý režim)



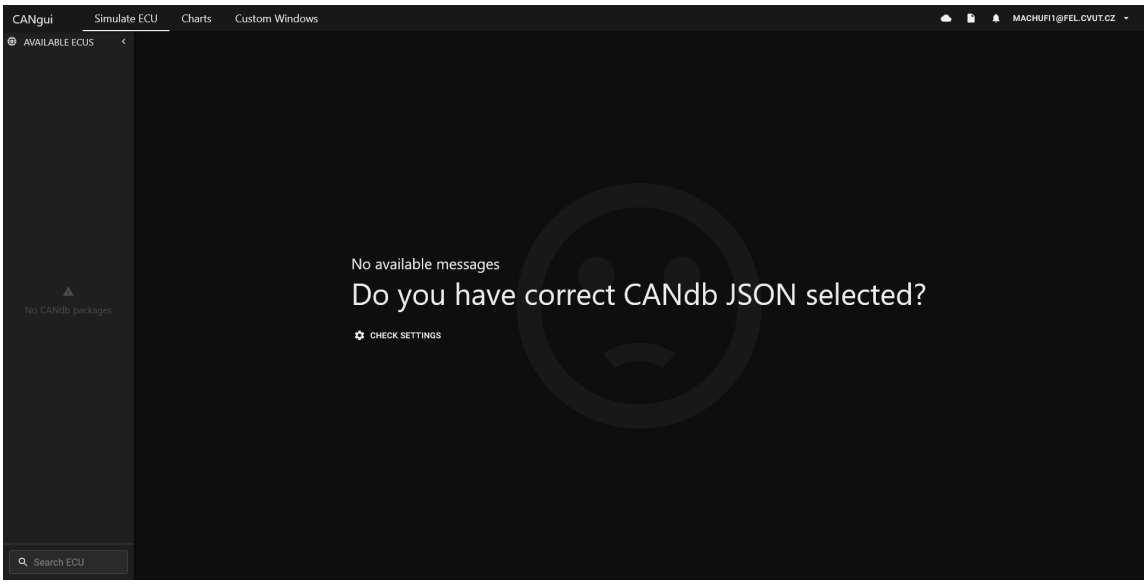
Obrázek B.10: CANgui – Úvodní obrazovka po vybrání zdroje dat (Světlý režim)



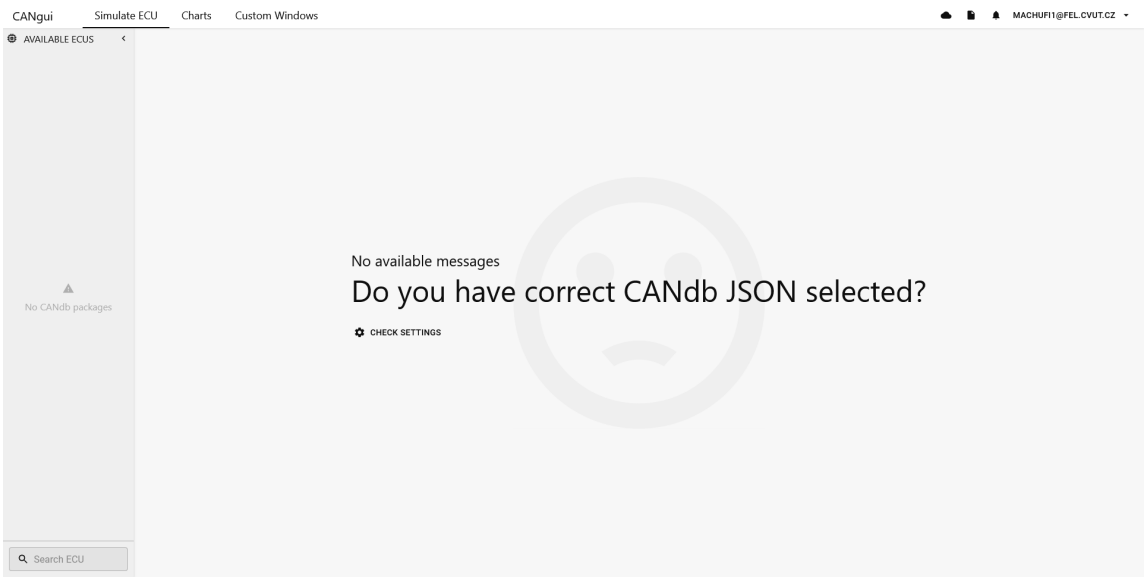
Obrázek B.11: CANgui – Obrazovka s nastavením aplikace (Tmavý režim)



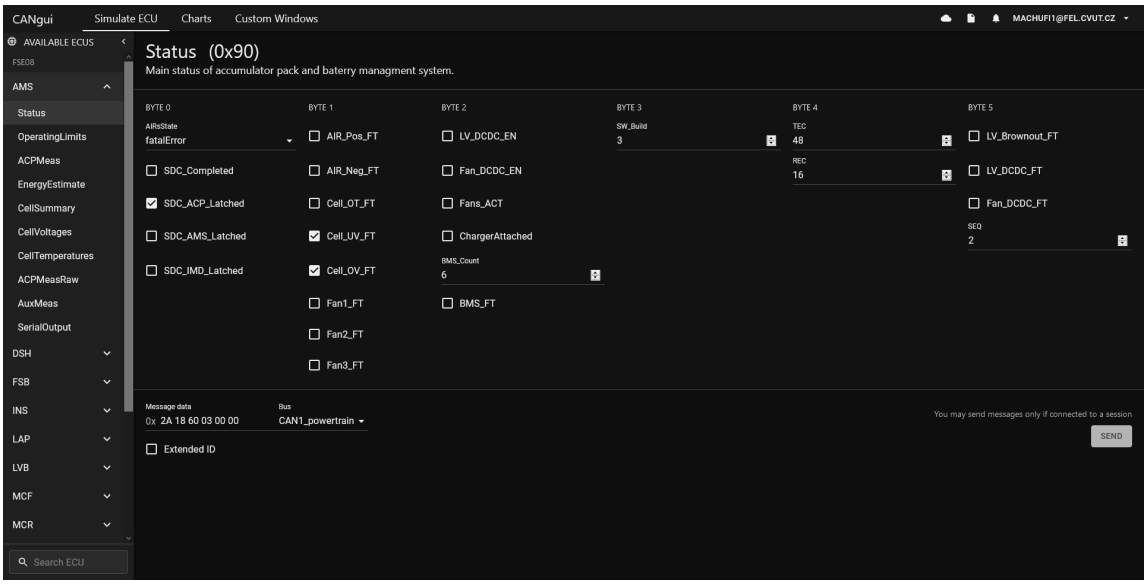
Obrázek B.12: CANgui – Obrazovka s nastavením aplikace (Světlý režim)



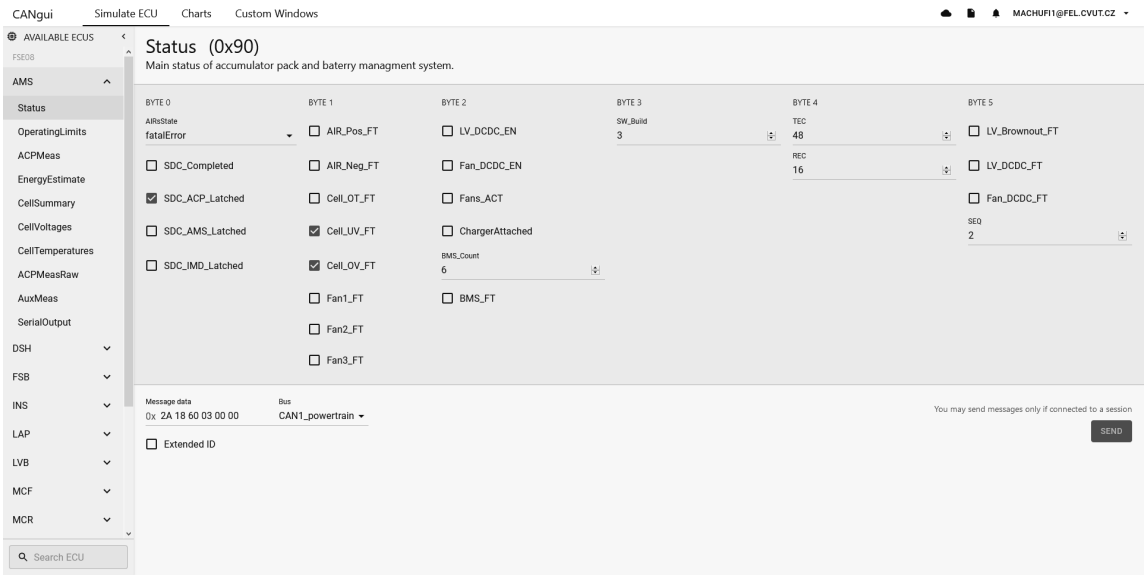
Obrázek B.13: CANgui – Obrazovka pro simulaci ECU bez validního CANdb modelu (Tmavý režim)



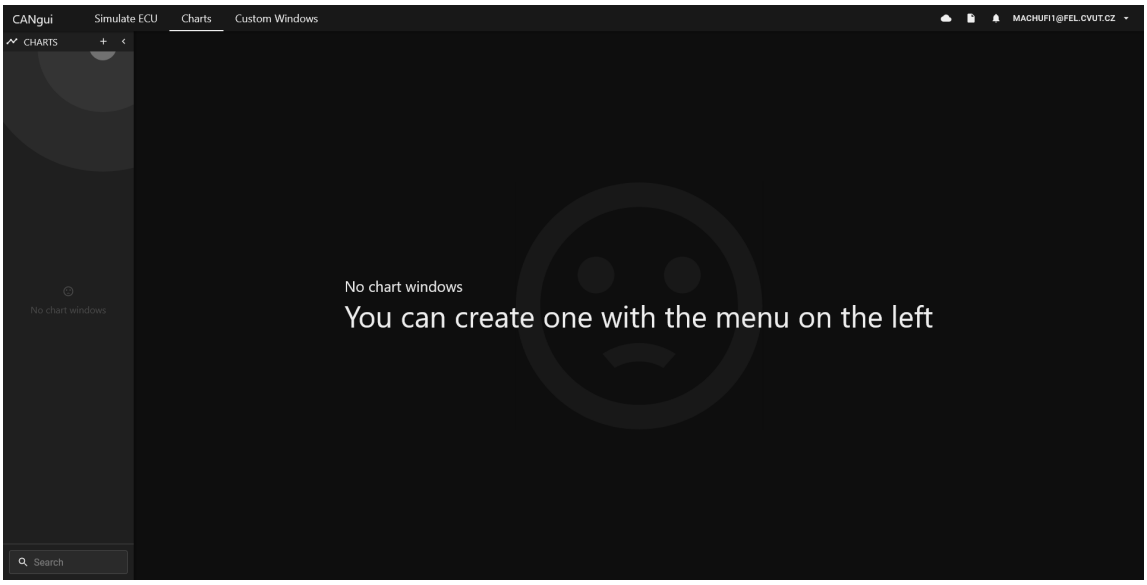
Obrázek B.14: CANgui – Obrazovka pro simulaci ECU bez validního CANdb modelu (Světlý režim)



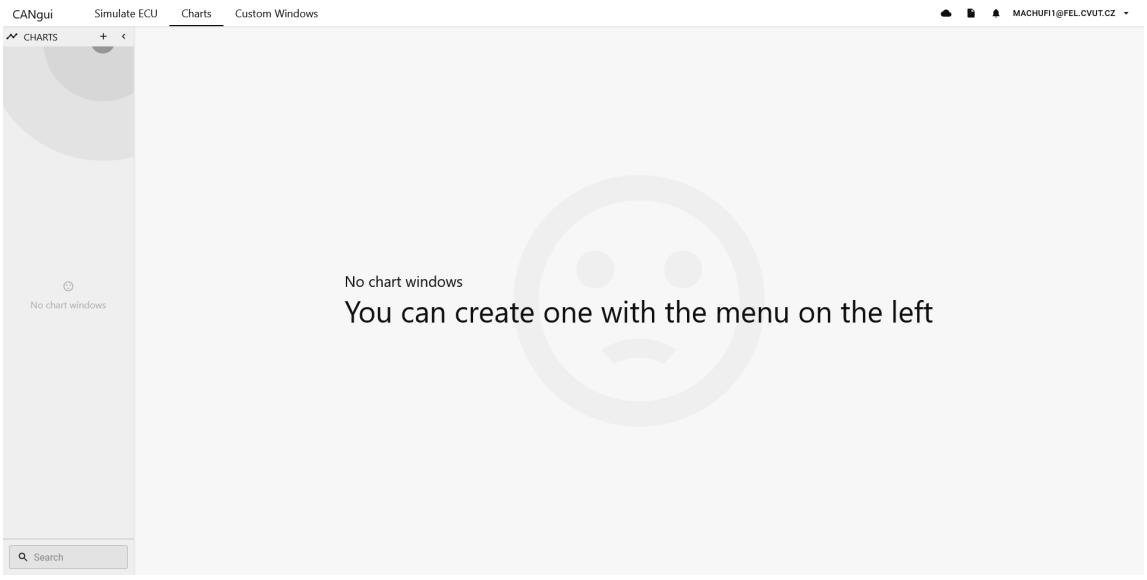
Obrázek B.15: CANgui – Obrazovka pro simulaci ECU (Tmavý režim)



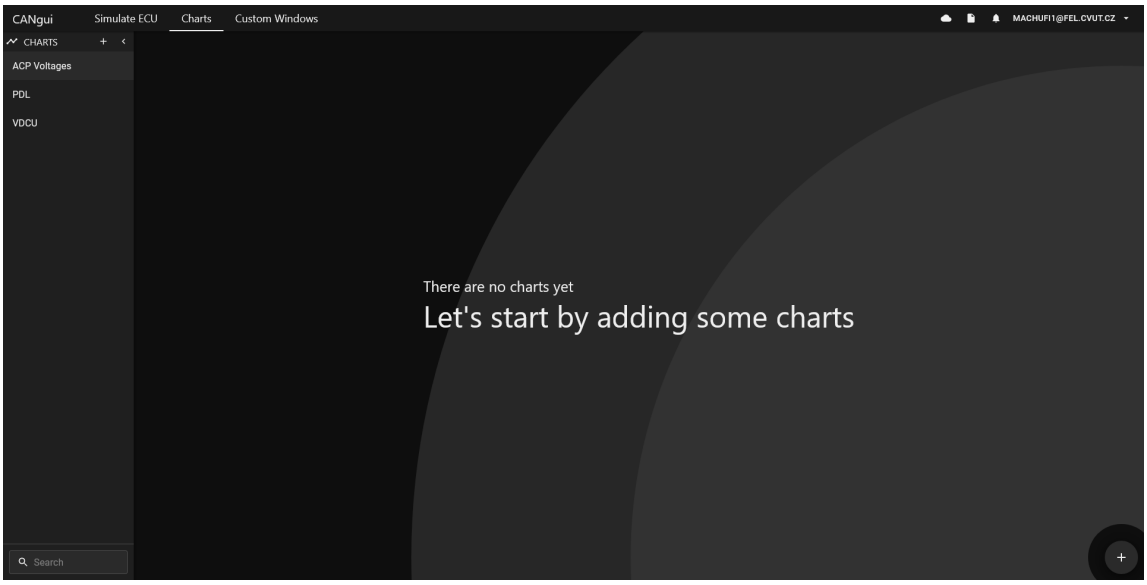
Obrázek B.16: CANgui – Obrazovka pro simulaci ECU (Světlý režim)



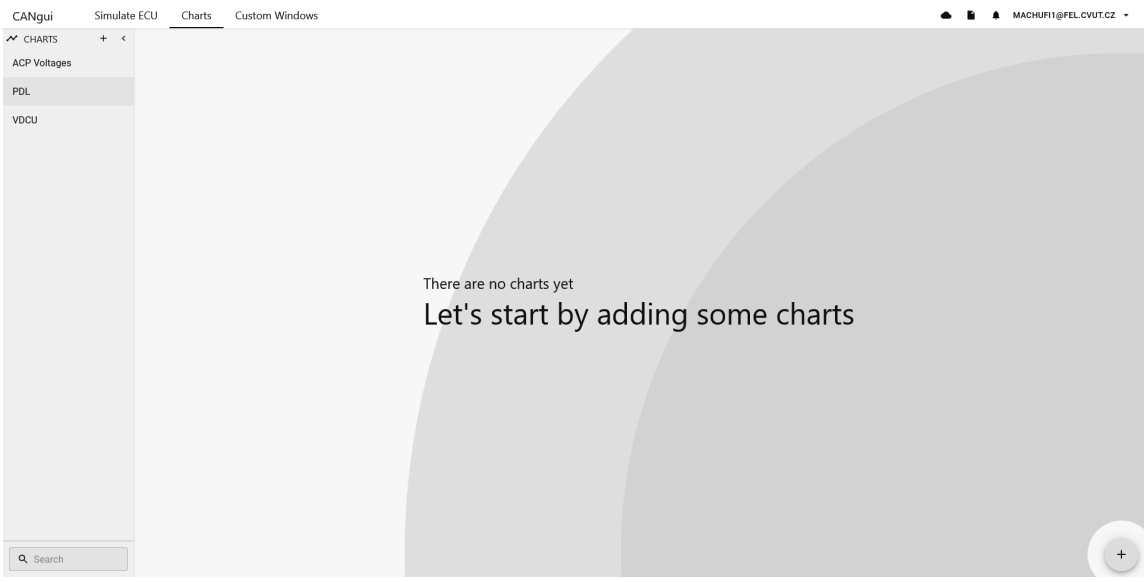
Obrázek B.17: CANgui – Obrazovka s grafy bez záložek (Tmavý režim)



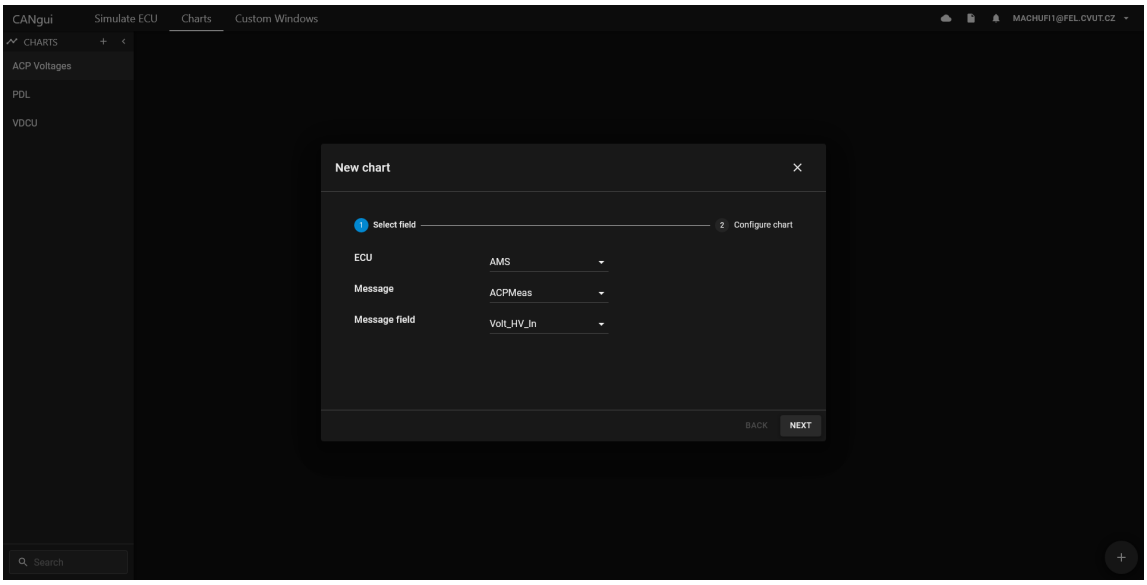
Obrázek B.18: CANgui – Obrazovka s grafy bez záložek (Světlý režim)



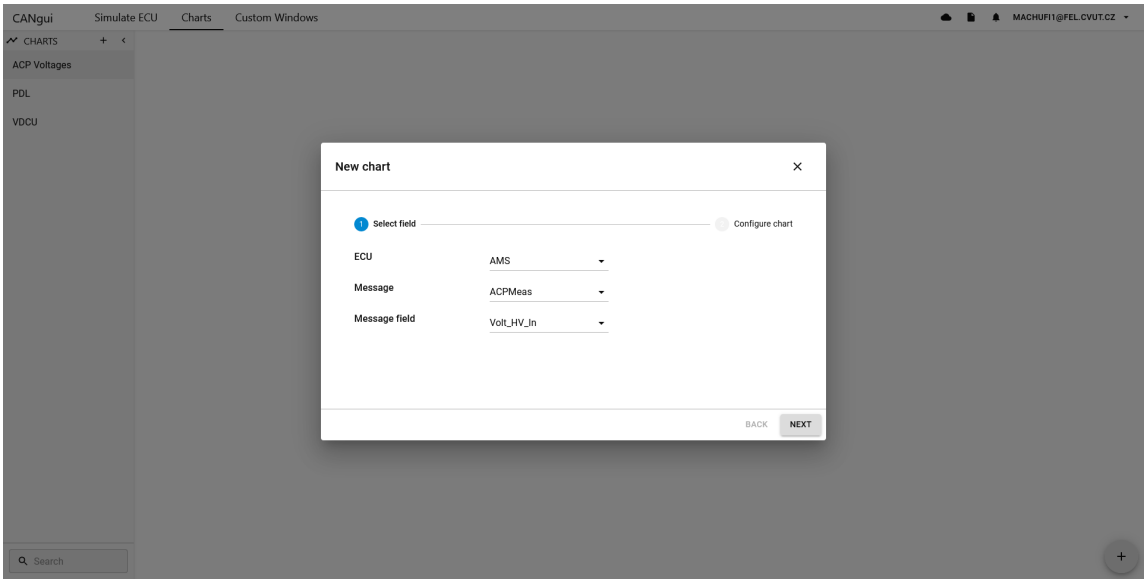
Obrázek B.19: CANgui – Obrazovka s grafy po vytvoření záložky (Tmavý režim)



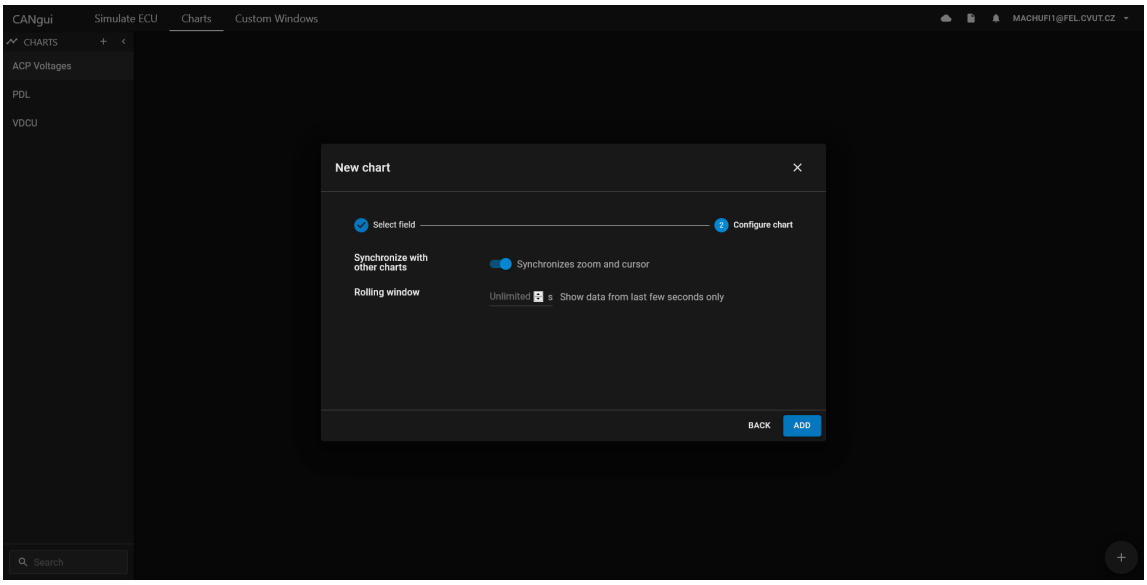
Obrázek B.20: CANgui – Obrazovka s grafy po vytvoření záložky (Světlý režim)



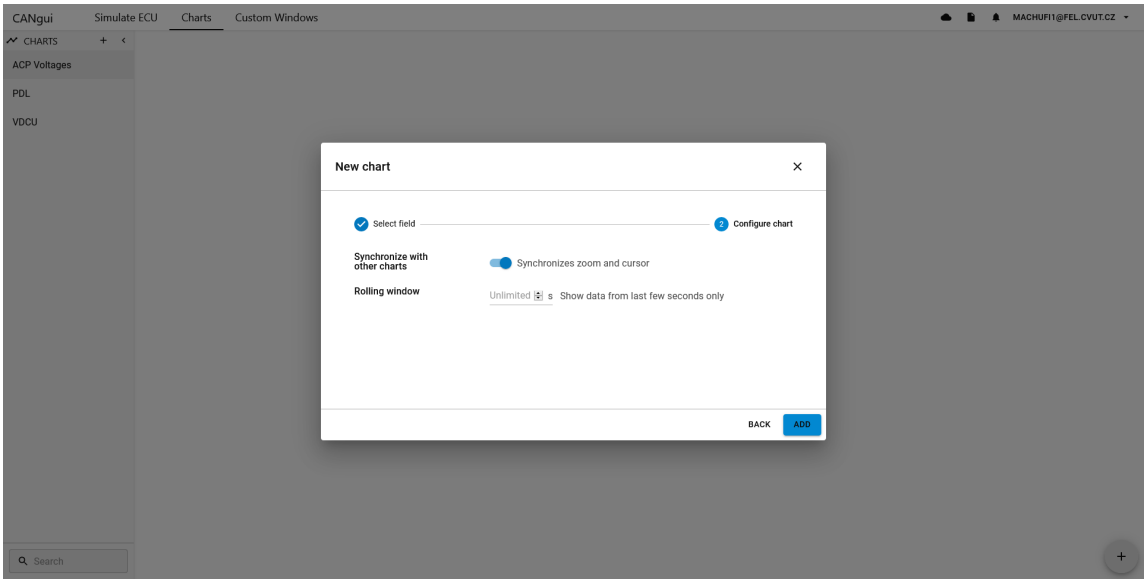
Obrázek B.21: CANgui – Dialog pro definici grafu – definice pole (Tmavý režim)



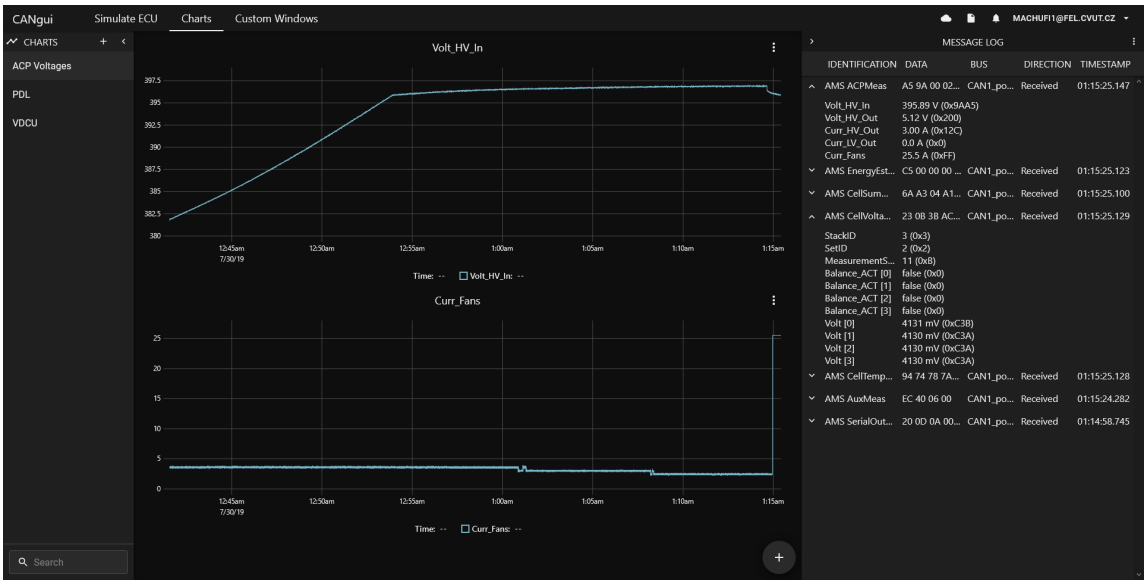
Obrázek B.22: CANgui – Dialog pro definici grafu – definice pole (Světlý režim)



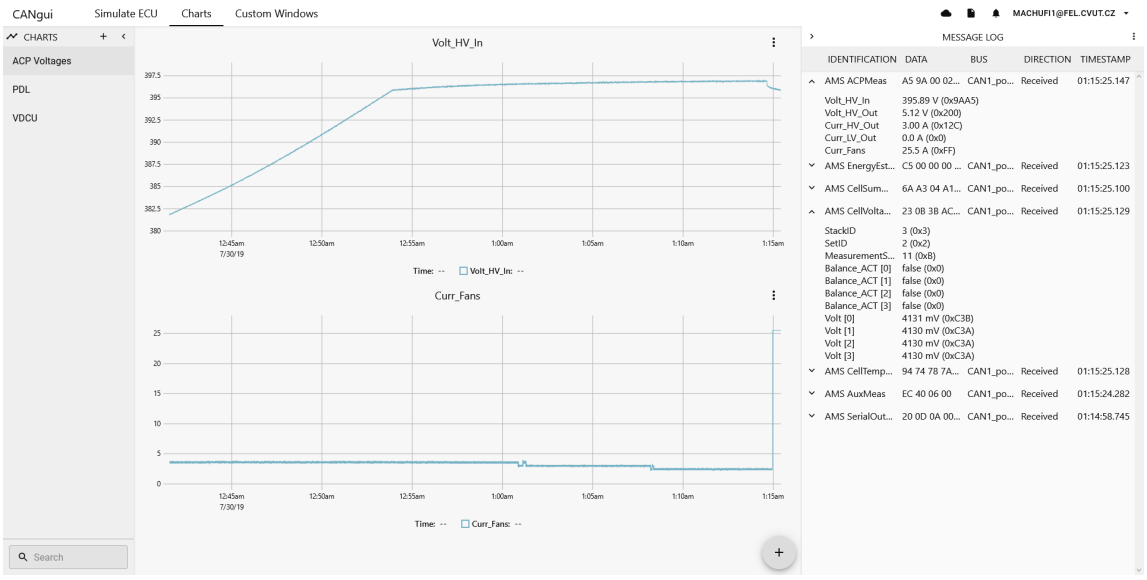
Obrázek B.23: CANgui – Dialog pro definici grafu – nastavení grafu (Tmavý režim)



Obrázek B.24: CANgui – Dialog pro definici grafu – nastavení grafu (Světlý režim)



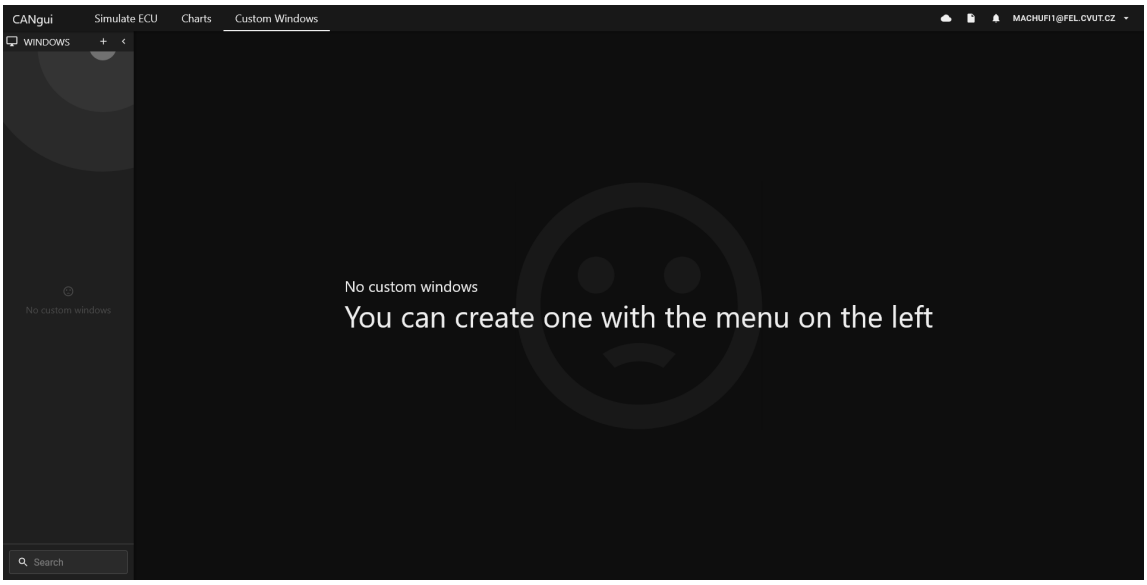
Obrázek B.25: CANgui – Obrazovka s grafy (Tmavý režim)



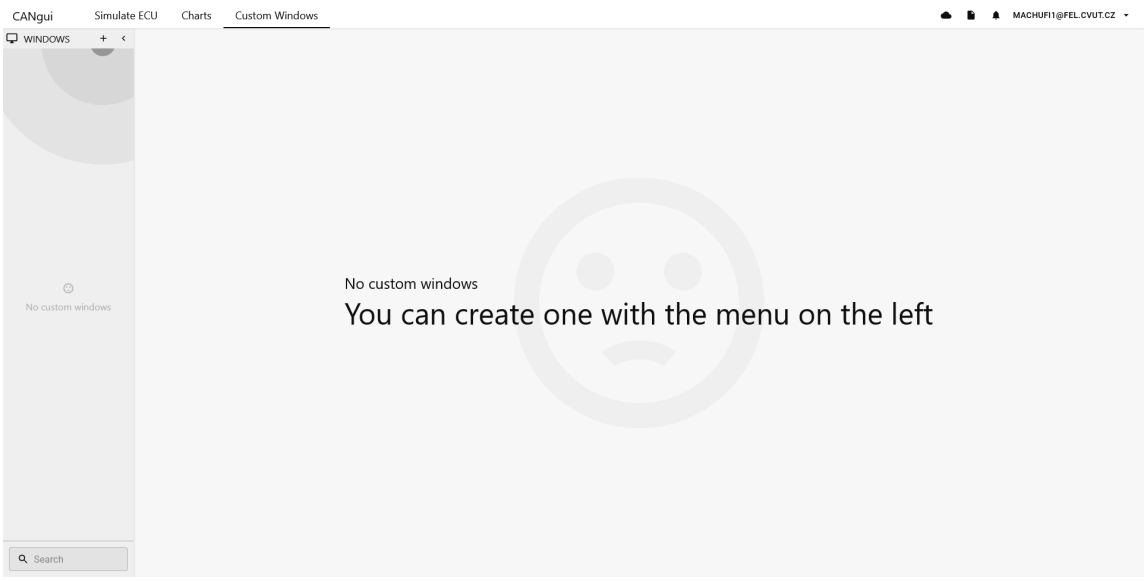
Obrázek B.26: CANgui – Obrazovka s grafy (Světlý režim)

Obrázek B.27: CANgui – Obrazovka s grafy – pole hodnot po přiblížení (Tmavý režim)

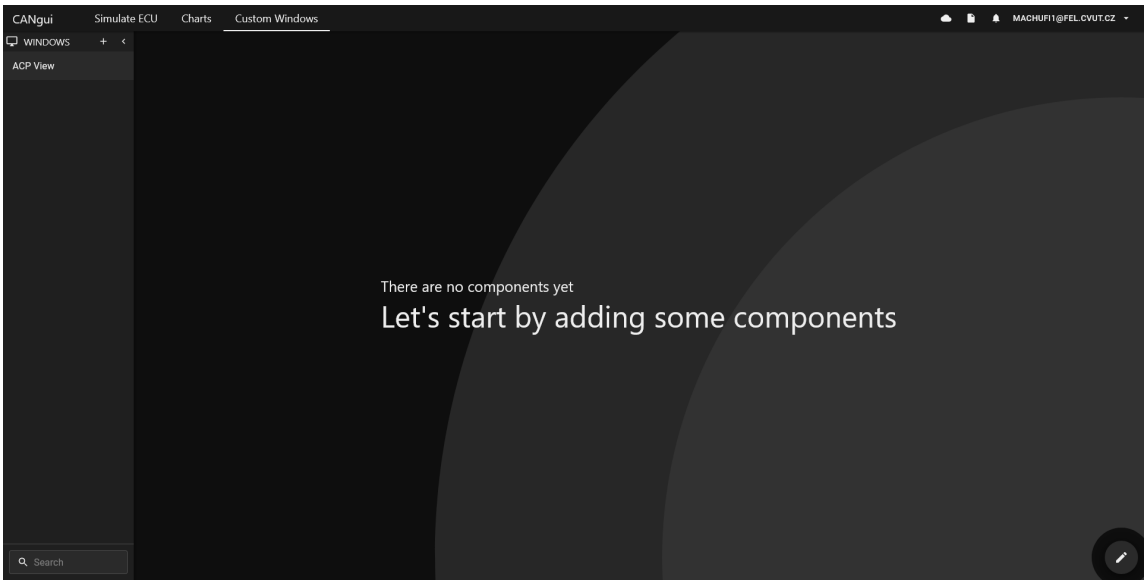
Obrázek B.28: CANgui – Obrazovka s grafy – pole hodnot po přiblížení (Světlý režim)



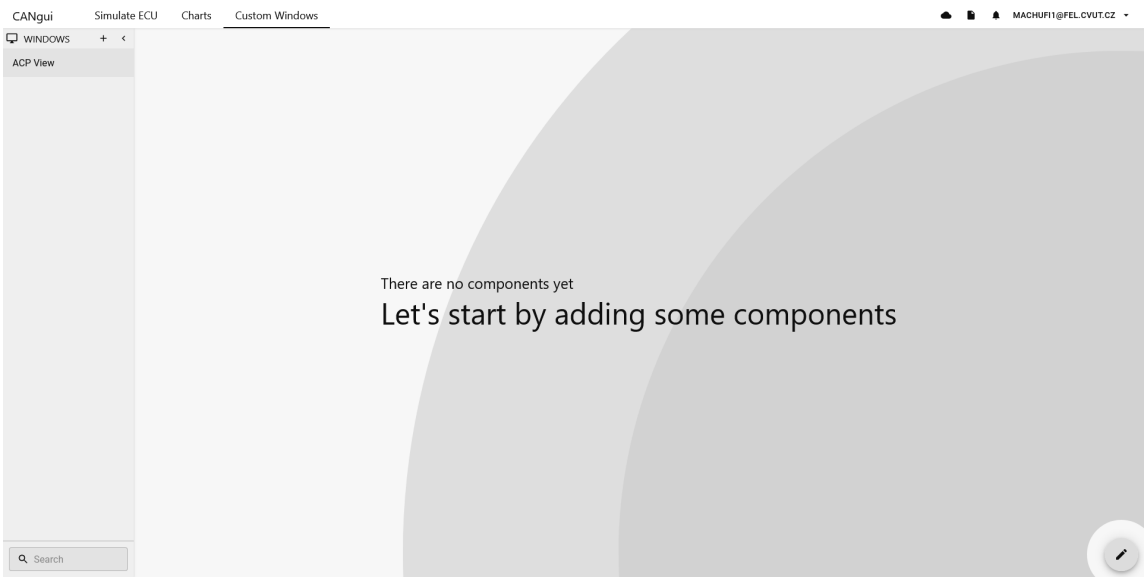
Obrázek B.29: CANgui – Obrazovka s vlastními obrazovkami bez záložek (Tmavý režim)



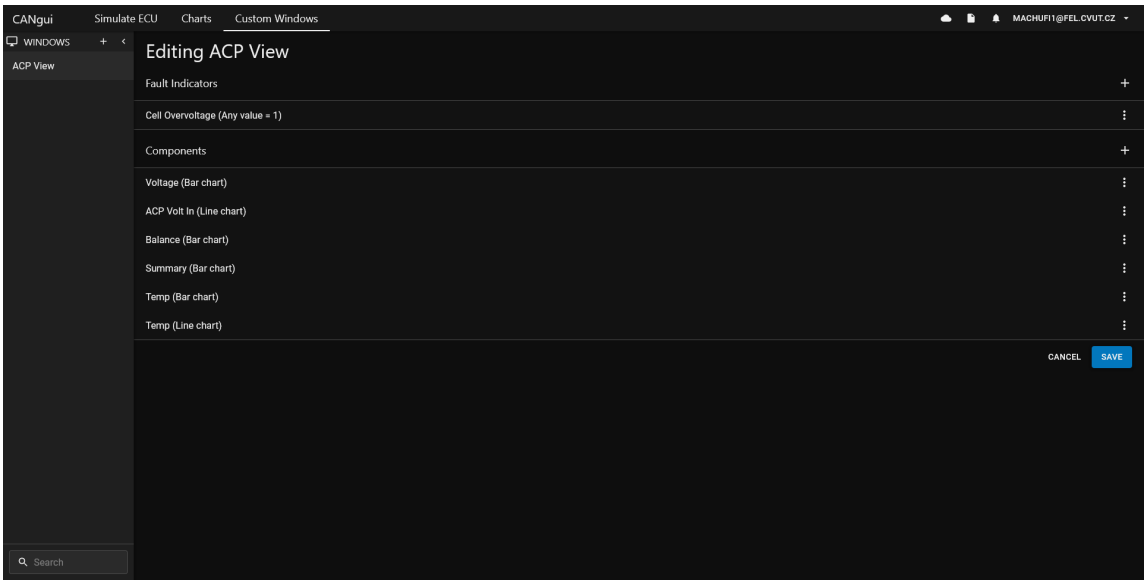
Obrázek B.30: CANgui – Obrazovka s vlastními obrazovkami bez záložek (Světlý režim)



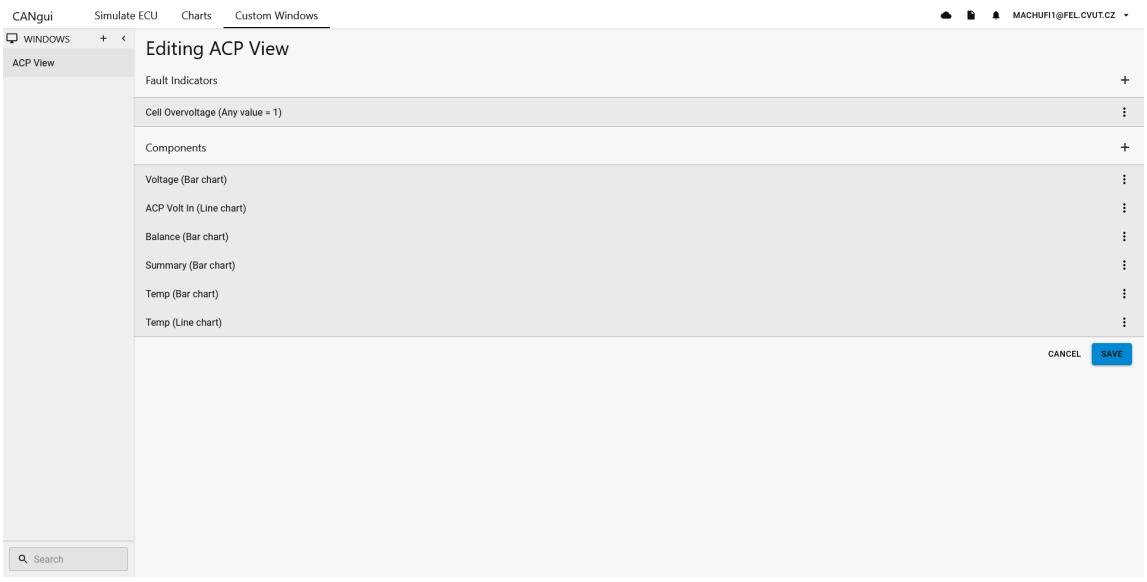
Obrázek B.31: CANgui – Vlastní obrazovka po vytvoření (Tmavý režim)



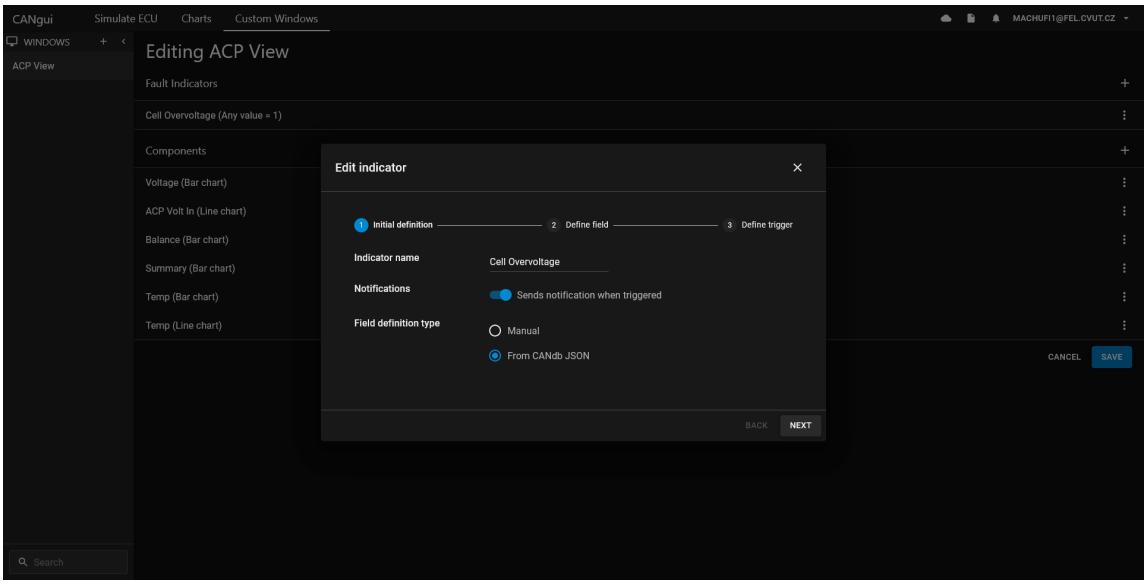
Obrázek B.32: CANgui – Vlastní obrazovka po vytvoření (Světlý režim)



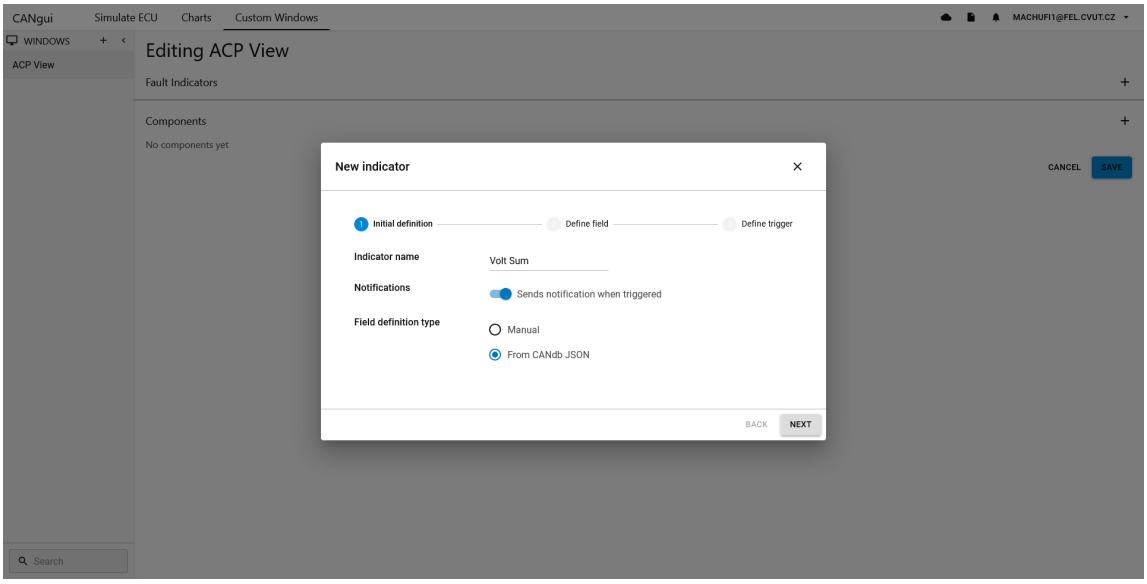
Obrázek B.33: CANgui – Editační obrazovka vlastní obrazovky (Tmavý režim)



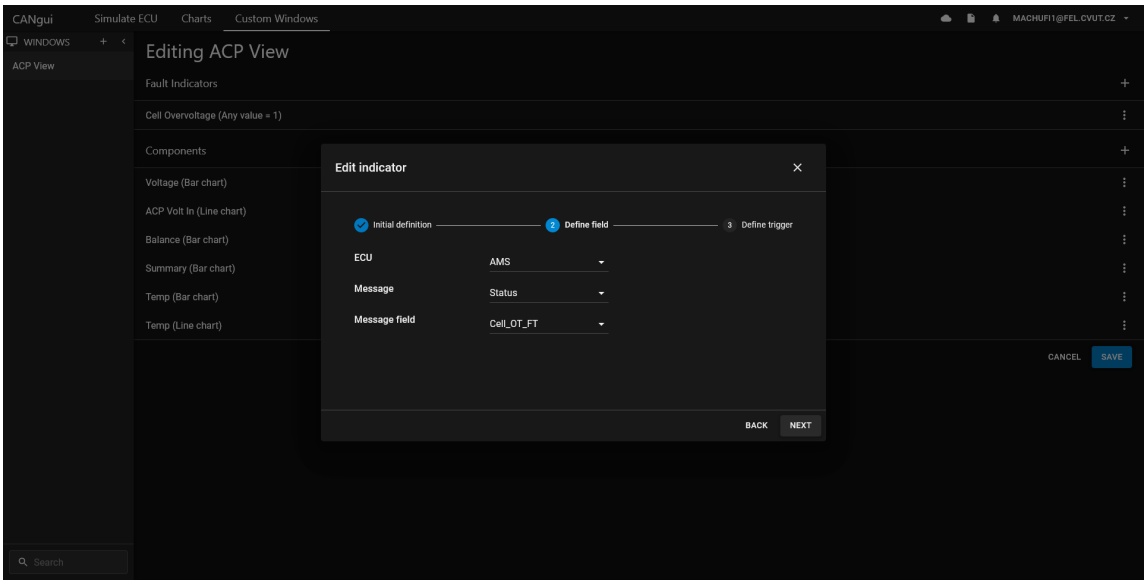
Obrázek B.34: CANgui – Editační obrazovka vlastní obrazovky (Světlý režim)



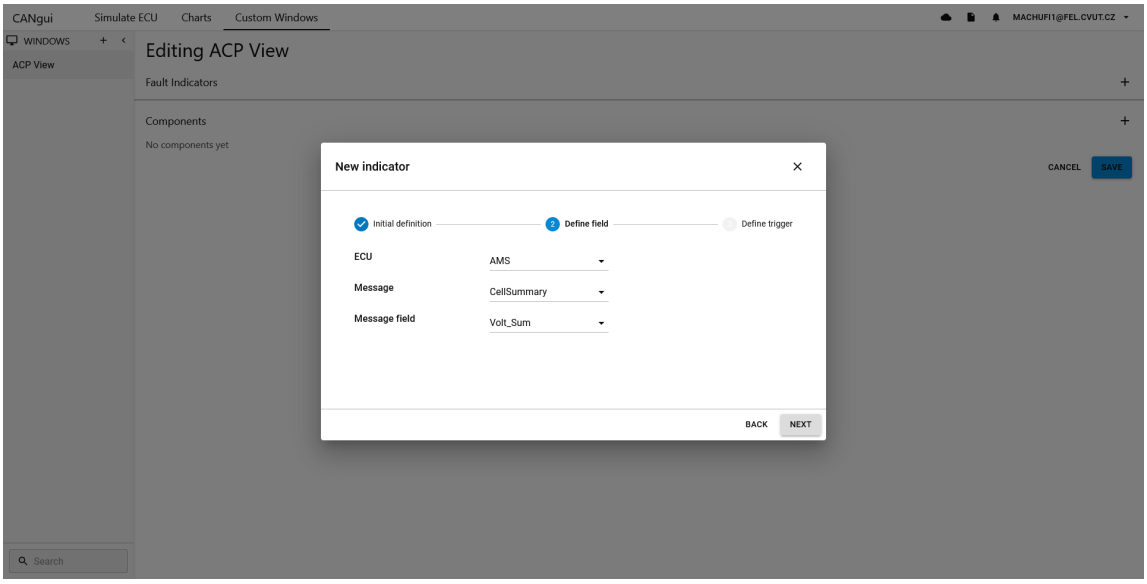
Obrázek B.35: CANgui – Dialog pro definici indikátoru – základní nastavení (Tmavý režim)



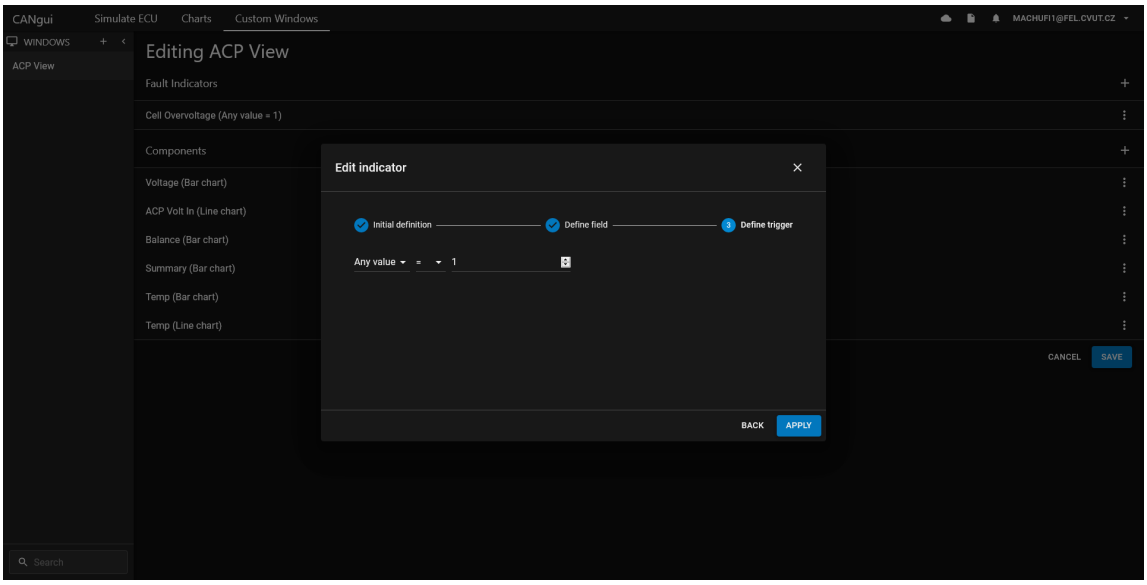
Obrázek B.36: CANgui – Dialog pro definici indikátoru – základní nastavení (Světlý režim)



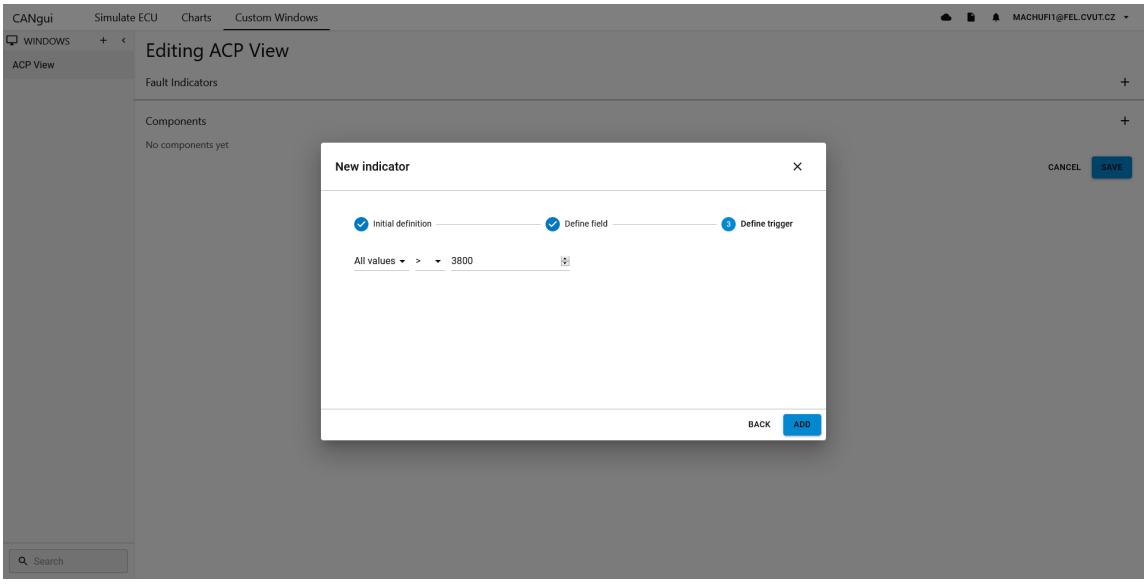
Obrázek B.37: CANgui – Dialog pro definici indikátoru – definice pole (Tmavý režim)



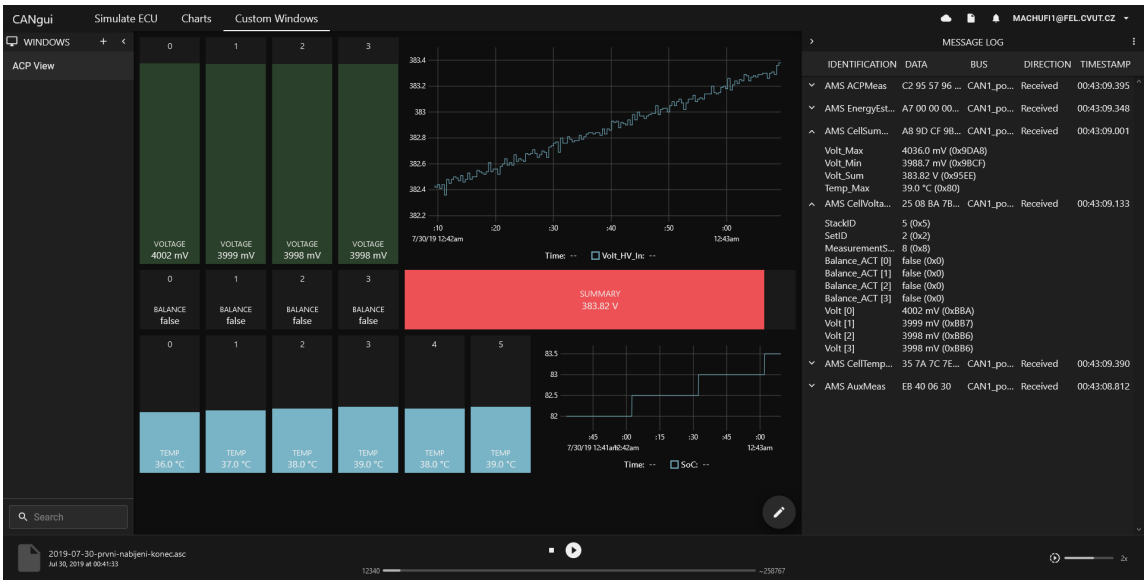
Obrázek B.38: CANgui – Dialog pro definici indikátoru – definice pole (Světlý režim)



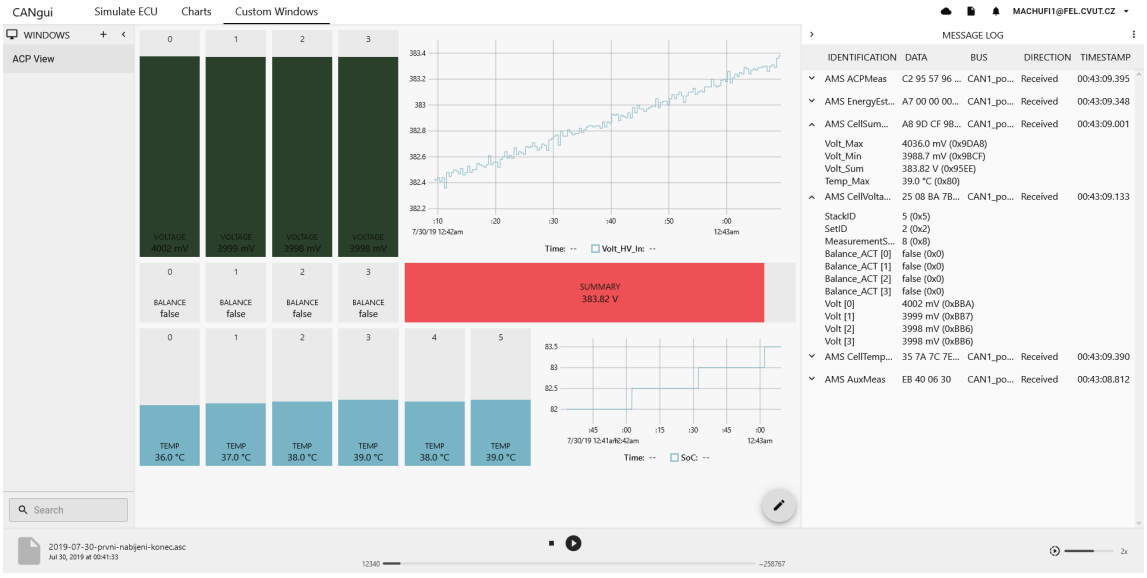
Obrázek B.39: CANgui – Dialog pro definici indikátoru – definice podmínky (Tmavý režim)



Obrázek B.40: CANgui – Dialog pro definici indikátoru – definice podmínky (Světlý režim)



Obrázek B.41: CANgui – Vlastní obrazovka při přehrávání ze souboru (Tmavý režim)



Obrázek B.42: CANgui – Vlastní obrazovka při přehrávání ze souboru (Světlý režim)

Příloha C

Zprávy Ocarina Protokolu v3

Následuje specifikace zpráv Ocarina Protokolu v3 [67], používaného ke komunikaci mezi zařízením Ocarina IV [66] a osobním počítačem.

C.1 Connect

- Příkaz
  - Slouží k synchronizaci se zařízením
  - Musí být poslán jako první příkaz po připojení k zařízení
  - Způsobí reset CAN rozhraní (no fwd, silent, auto bitrate) a vyprázdnění vyrovnávací paměti (FIFO)
  - Nevyčistí chybové příznaky
  - Formát viz tabulka C.1

T	**L**
'r'	0

Tabulka C.1: Ocarina Protokol v3 – Příkaz **connect**

- Odpověď
  - Speciální formát (není ve formátu TLV!)
  - Ocarina pošle sekvenci 24x 0xAA

C.2 Query Version

- Příkaz
  - Dotaz pro zjištění verze zařízení
  - Slouží k detekci kompatibility
  - Formát viz tabulka C.2

T	L
'v'	0

Tabulka C.2: Ocarina Protokol v3 – Příkaz **query version**

- Odpověď
  - Formát viz tabulka C.3
    - **PROT:** verze komunikačního protokolu (aktuálně verze 3)
      - Důležitá položka, která by se měla zkontrolovat ihned po spojení se zařízením
    - **SW:** verze Software zařízení
    - **HW:** verze Hardware zařízení (4 pro Ocarina IV)
    - **HW revision:** revize desky zařízení

T	L	V0	V1	V2	V3
'v'	4	PROT	SW	HW	HW revision

Tabulka C.3: Ocarina Protokol v3 – Odpověď na příkaz **query version**

C.3 Auto bitrate

- Příkaz
  - Zapíná takzvaný auto bitrate mód
  - Ocarina postupně zkusí všechny podporované hodnoty bitrate (viz tabulka C.6), než se jí povede přečíst zprávu ze sběrnice
  - Dokud Ocarina nenajde validní bitrate sběrnice, ignoruje ostatní nastavení periferie (silent, loopback a data forwarding)
  - Formát viz tabulka C.4

T	L
'a'	0

Tabulka C.4: Ocarina Protokol v3 – Příkaz **auto bitrate**

C.4 Manual bitrate

- Příkaz
  - Nastaví bitrate CAN periferie manuálně
  - Formát viz tabulka C.5
    - **BR:** hodnota bitrate – diskrétní hodnota mapovaná podle výčtu podporovaných bitrate (viz tabulka C.6)
      - Hodnota UNKNOWN není pro příkaz **manual bitrate** validní a způsobí chybu

T	L	V0
'b'	1	BR[3:0]

Tabulka C.5: Ocarina Protokol v3 – Příkaz **manual bitrate**

Bitrate (kbit/s)	BR[3:0]
UNKNOWN	0
10	1
20	2
50	3
100	4
125	5
250	6
500	7
800	8
1000	9

Tabulka C.6: Ocarina Protokol v3 – Výčet podporovaných hodnot bitrate

C.5 Query config

- Příkaz
  - Dotaz ke zjištění konfigurace CAN periferie
  - Formát viz tabulka C.7

T	L
'g'	0

Tabulka C.7: Ocarina Protokol v3 – Příkaz **query config**

- Odpověď
  - Formát viz tabulka C.8
    - **FWD:** zda je zapnuto přeposílání zpráv ze sběrnice (viz sekce C.10 a C.11)
    - **LOOP:** zda je zapnutý loopback mód (viz sekce C.8 a C.9)
    - **SILENT:** zda je zapnutý silent mód (viz sekce C.6 a C.7)
    - **BR:** aktuálně nastavený bitrate periferie dle výčtu podporovaných bitrate (viz tabulka C.6)
      - Hodnota UNKNOWN v tomto případě znamená, že je zařízení v auto bitrate módu (viz sekce C.3)

T	L	V0
'g'	1	FWD[6] LOOP[5] SILENT[4] BR[3:0]

Tabulka C.8: Ocarina Protokol v3 – Odpověď na příkaz **query config**

C.6 Enable silent mode

- Příkaz
  - Zapne takzvaný silent mód
    - CAN periferie blokuje veškerý přenos (včetně posílání zpráv, potvrzení (ACK) a chybových rámců)
    - Vhodné pro sledování provozu, bez narušení chodu sběrnice
  - Formát viz tabulka C.9

T	L
'S'	0

Tabulka C.9: Ocarina Protokol v3 – Příkaz **enable silent mode**

C.7 Disable silent mode

- Příkaz
  - Vypne takzvaný silent mód
  - Opak příkazu viz sekce C.6
  - Formát viz tabulka C.10

T	L
's'	0

Tabulka C.10: Ocarina Protokol v3 – Příkaz **disable silent mode**

C.8 Enable loopback mode

- Příkaz
  - Zapne takzvaný loopback mód
    - Všechny zprávy co zařízení odesílá, také přijímá
    - V loopback módu zařízení nepřijímá ostatní zprávy ze sběrnice
    - Užitečné pro debugging komunikace se zařízením a stress testing (především v kombinaci se silent módem)
  - Formát viz tabulka C.11

T	L
'L'	0

Tabulka C.11: Ocarina Protokol v3 – Příkaz **enable loopback mode**

C.9 Disable loopback mode

- Příkaz
  - Vypne takzvaný. loopback mód
  - Opak příkazu viz sekce C.8
  - Formát viz tabulka C.12

T	L
'l'	0

Tabulka C.12: Ocarina Protokol v3 – Příkaz **disable loopback mode**

C.10 Enable received data forwarding

- Příkaz
  - Zapne takzvaný received data forwarding
    - Všechny přijaté zprávy z CAN sběrnice jsou přeposlány hostovi po seriovém portu
    - Nutné pro příjem CAN zpráv
  - Formát viz tabulka C.13

T	L
'F'	0

Tabulka C.13: Ocarina Protokol v3 – Příkaz **enable received data forwarding**

C.11 Disable received data forwarding

- Příkaz
  - Vypne takzvaný received data forwarding
  - Pokud je vypnutý silent mód, tak Ocarina pořád potvrzuje přichozí zprávy (ACK)
  - Opak příkazu viz sekce C.10
  - Formát viz tabulka C.14

T	L
'f'	0

Tabulka C.14: Ocarina Protokol v3 – Příkaz **disable received data forwarding**

C.12 Query counters

- Příkaz
  - Dotaz na počet přijatých a odeslaných zpráv
  - Formát viz tabulka C.15

T	L
'c'	0

Tabulka C.15: Ocarina Protokol v3 – Příkaz **query counters**

- Odpověď
  - Vrátí počet přijatých a odeslaných zpráv
  - Formát viz tabulka C.16
    - **RX:** počet přijatých zpráv (ne nutně přeposlaných viz sekce C.10)
    - **TX:** počet odeslaných zpráv

T	L	V0	V1	V2	V3	V4	V5	V6	V7
'c'	8	RX[7:0]	RX[15:8]	RX[23:16]	RX[31:24]	TX[7:0]	TX[15:8]	TX[23:16]	TX[31:24]

Tabulka C.16: Ocarina Protokol v3 – Odpověď na příkaz **query counters**

C.13 Reset counters

- Příkaz
  - Příkaz vyresetuje čítače RX a TX (viz sekce C.12)
  - Formát viz tabulka C.17

T	L
'C'	0

Tabulka C.17: Ocarina Protokol v3 – Příkaz **reset counters**

C.14 Reboot to DFU

- Příkaz
  - Restartuje zařízení do Device Firmware Upgrade (DFU) módu pro flashování
  - Formát viz tabulka C.18

T	L
'd'	0

Tabulka C.18: Ocarina Protokol v3 – Příkaz **reboot to DFU**

C.15 Query error flags

- Příkaz
  - Dotaz k získání chybových příznaků zařízení
  - Dotaz také chybové příznaky **resetuje**
  - Formát viz tabulka C.19

T	L
'e'	0

Tabulka C.19: Ocarina Protokol v3 – Příkaz **query error flags**

- Odpověď/Událost
  - Kromě odpovědi na příkaz, Ocarina pošle chybovou událost (tuto zprávu) v případě, že nastane jakákoliv chyba
  - Formát viz tabulka C.20
    - **ERR:** chybové příznaky zařízení (hodnoty viz tabulka C.21)

T	L	V0	V1	V2	V3
'e'	4	ERR[7:0]	ERR[15:8]	ERR[23:16]	ERR[31:24]

Tabulka C.20: Ocarina Protokol v3 – Odpověď na příkaz **query error flags**

ERR bit	Popis
0	Přetečení zásobníku USB IN zařízení
1	Přetečení zásobníku USB OUT zařízení
2	Přetečení zásobníku CAN RX zařízení
3	Přetečení zásobníku CAN TX zařízení
4	Neplatný příkaz pro odeslání zprávy
5	Příkaz není rozpoznán
6	Pokus o odeslání zprávy před inicializací CAN periferie (bitrate)
7	Příkaz očekával jinou délku datové části
8	Délka datové části byla příliš dlouhá

Tabulka C.21: Ocarina Protokol v3 – Chybové příznaky

C.16 Query device ID

- Příkaz
  - Dotaz k získání ID zařízení
  - Formát viz tabulka C.22

T	L
'i'	0

Tabulka C.22: Ocarina Protokol v3 – Příkaz **query device ID**

- Odpověď
  - Vrací String s ID zařízení
  - Formát viz tabulka C.23

T	L	V0	V0	V0	...	Vn
'i'	n	'O'	'c'	'a'	...	...

Tabulka C.23: Ocarina Protokol v3 – Odpověď na příkaz **query device ID**

C.17 Send STD ID message

- Příkaz
  - Požadavek k odeslání zprávy se standardním ID na sběrnici
  - Formát viz tabulka C.24

T	L	V0	V1	V2	V3	...	Vn
'm'	n	ID[7:0]	ID[10:8]	DATA0	DATA1	...	DATA(n-2)

Maximální délka datové části je 8 byte → n je v rozmezí od 2 do 10

Tabulka C.24: Ocarina Protokol v3 – Příkaz **send STD ID message**

C.18 Send EXT ID message

- Příkaz
  - Požadavek k odeslání zprávy se rozšířeným ID na sběrnici
  - Formát viz tabulka C.25

T	L	V0	V1	V2	V3	V4	...	Vn
'M'	n	ID[7:0]	ID[15:8]	ID[23:16]	ID[28:24]	DATA0	...	DATA(n-4)

Maximální délka datové části je 8 byte → n je v rozmezí od 4 do 12

Tabulka C.25: Ocarina Protokol v3 – Příkaz **send EXT ID message**

C.19 STD ID message received

- Událost
  - Vnitřně vyvolaná událost Ocariny, pokud CAN periferie přijme zprávu se standardním ID
  - Formát viz tabulka C.26
    - **TS:** Časové razítko o velikosti 5 byte, s rozlišením 1 μs

T	**L**	**V0**	**V1**	**V2**	**V3**	**V4**
'm'	n	TS[7:0]	TS[15:8]	TS[23:16]	TS[31:24]	TS[39:32]

V5	**V6**	**V7**	...	**Vn**
ID[7:0]	ID[10:8]	DATA0	...	DATA(n-2)

Maximální délka datové části je 8 byte → n je v rozmezí od 7 do 15

Tabulka C.26: Ocarina Protokol v3 – Událost **STD ID message received**

C.20 EXT ID message received

- Událost
  - Vnitřně vyvolaná událost Ocariny, pokud CAN periferie přijme zprávu se rozšířeným ID
  - Formát viz tabulka C.27
    - **TS:** Časové razítko o velikosti 5 byte, s rozlišením 1 μs

T	**L**	**V0**	**V1**	**V2**	**V3**	**V4**
'M'	n	TS[7:0]	TS[15:8]	TS[23:16]	TS[31:24]	TS[39:32]

V5	**V6**	**V7**	**V8**	**V9**	...	**Vn**
ID[7:0]	ID[15:8]	ID[23:16]	ID[28:24]	DATA0	...	DATA(n-4)

Maximální délka datové části je 8 byte → n je v rozmezí od 9 do 17

Tabulka C.27: Ocarina Protokol v3 – Událost **EXT ID message received**

C.21 Error on CAN occured

- Událost
  - Vnitřně vyvolaná událost Ocariny, pokud detekuje na CAN sběrnici chybu
  - Formát viz tabulka C.28
    - **TS:** Časové razítko o velikosti 5 byte, s rozlišením 1 μs
    - **TEC:** Počet chyb při přenosu
    - **REC:** Počet chyb při příjmu
    - **ERR:** Hodnota odpovídá výčtu chybových hodnot (viz tabulka C.29)
    - **BUS:** Hodnota odpovídá výčtu stavů sběrnice (viz tabulka C.30)

T	L	V0	V1	V2	V3	V4	V5	V6	V7
'E'	8	TS[7:0]	TS[15:8]	TS[23:16]	TS[31:24]	TS[39:32]	TEC	REC	ERR[7:4]BUS[1:0]

Tabulka C.28: Ocarina Protokol v3 – Událost **error on CAN occured**

ERR	Popis
0	Žádný chyba nebyla detekován
1	Stuffing chyba
2	Chyba ve formátování
3	Zpráva nepotvrzena (ACK)
4	Sběrnice zaseknutá v recesivním stavu
5	Sběrnice zaseknutá v dominantním stavu
6	CRC mismatch
7	Zaviněno softwarem
15	Neznámá chyba

Tabulka C.29: Ocarina Protokol v3 – Chybové kódy CAN sběrnice

BUS	Popis
0	CAN sběrnice provozní
1	Pouze pasivní interakce (kvůli chybám)
2	Odpojeno ze sběrnice (kvůli chybám)

Tabulka C.30: Ocarina Protokol v3 – Stavy CAN sběrnice

C.22 Heartbeat

- Událost
  - Indikuje provozní Ocarinu
  - Perioda zprávy se pohybuje kolem 1 s
  - Formát viz tabulka C.31

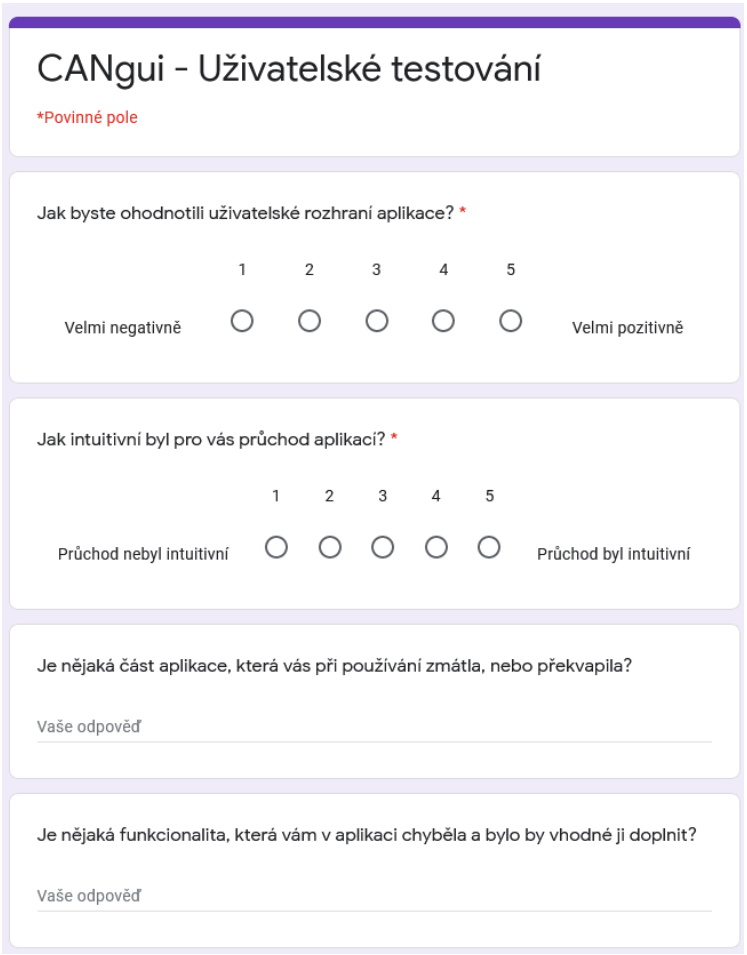
T	L
'h'	0

Tabulka C.31: Ocarina Protokol v3 – Událost **heartbeat**

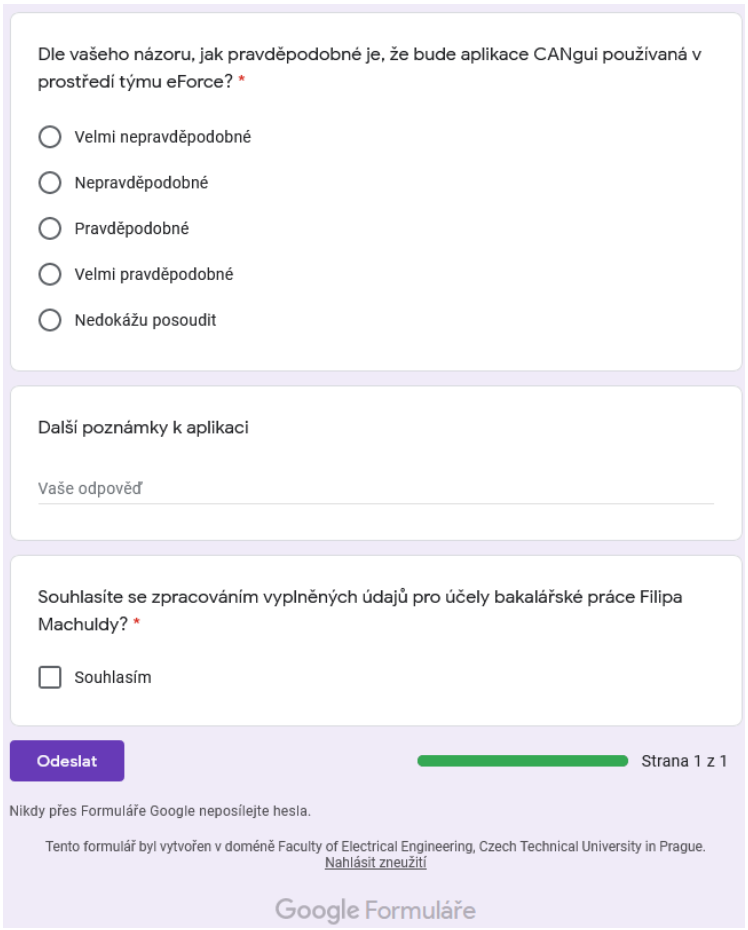
Příloha D

Dotazník pro uživatelské testování

Dotazník byl navržen se dvěma cíli - zhodnotit momentální stav aplikace a získat náměty pro její další rozvoj. První cíl pokrývají kvantitativní otázky zaměřené na spokojenost uživatelů. Druhý cíl naopak pokrývají kvalitativně zaměřené otázky, kde bylo možné sesbírat konkrétní náměty skrze otevřené odpovědi. Dotazník byl navržen tak, aby ho bylo možné vyplnit do pěti minut.



Obrázek D.1: Dotazník pro uživatelské testování (1. část)



Obrázek D.2: Dotazník pro uživatelské testování (2. část)

Příloha E

Odpovědi dotazníku uživatelského testování

V této příloze jsou uvedeny odpovědi z dotazníku v surové formě. Šetření bylo provedeno elektronickou formou bezprostředně po testování. Účastníci jsou rozděleni do tří skupin, dle jejich délky působení v týmu: N (Nováček), S (Seniorní člen), A (Alumni – Bývalý člen)

E.1 Jak byste ohodnotili uživatelské rozhraní aplikace?

Účastník	Hodnocení
1 (S)	5/5
2 (S)	5/5
3 (N)	5/5
4 (S)	5/5
5 (A)	4/5
6 (A)	4/5

1/5 = Velmi negativně

5/5 = Velmi pozitivně

Tabulka E.1: Odpovědi na otázku č.1

E.2 Jak intuitivní byl pro vás průchod aplikací?

Účastník	Hodnocení
1 (S)	4/5
2 (S)	5/5
3 (N)	5/5
4 (S)	4/5
5 (A)	3/5
6 (A)	4/5

1/5 = Průchod nebyl intuitivní

5/5 = Průchod byl intuitivní

Tabulka E.2: Odpovědi na otázku č.2

E.3 Je nějaká část aplikace, která vás při používání zmátla, nebo překvapila?

Účastník	Odpověď
1 (S)	Chýbal Json, Automatický upload jasnu by bol dobrý
2 (S)	Jen pozitivně překvapen. Aplikace umí o mnoho víc věcí, než jsem čekal a nad to vypadá velmi moderně.
3 (N)	Trošku jsem byl zmatený z toho, že musím nahrát data do aplikace, abych si mohl dát zobrazit nějaký graf. Ale toto bylo zejména způsobeno tím, že jsem nepracoval s předchozí verzí, která tuto funkcionalitu představila.
4 (S)	Změna nastavení – musím udělat dva kliky abych se do něj dostal a nevadilo by mi mít kolečko nastavení na horní liště, piktogramy k tlačítku ocarina session (připomíná uložení/stažení na cloud)
5 (A)	Hned na začátku jsem nevěděl, že se aplikace skládá ze 2 částí (ocarina klient a GUI). Vizuálně jsem měl problémy s nízkými kontrasty mezi prvky, které spolu sousedili, ale jinak byly nezávislé. Když jsem chtěl poslat CAN zprávu, nenapadlo mě hledat pod "Simulate ECU", nechtěl jsem simulovat chování nějaké ECU. Nejčastěji používané akce by měly mít ovládací prvky více na očích. Vyhledávání v levém stromě by mohlo hledat v celém stromu a ne jen top level. Tmavé motivy mám rád, ale podle mě tu chybí nějaký "akcent", barva, která by právě zvýraznila akční prvky atp. Přidávání grafu, kde člověk na jedné obrazovce vybere jednotku, potvrdí, vybere zprávu potvrdí,... mi připadalo strašně zdlouhavé. Strašně moc by se mi líbila funkcionalita, kterou má napčíklad Visual Studio Code – každná funkce se dá aktivovat z popup "příkazového řádku". Pokud zůstaneme pri zemi, tak minimálně při přidávání grafu by mi mnohem víc sedělo fulltextové vyhledávání v názvech jednotek, zprávách a fieldech najednou. Např klíčové slovo cell, už by mi odfiltrovalo 95% fiieldů a z těch pár bych si už vybral. Potvrzování voleb kávesou enter mi hodně chybělo. Zobrazení zpráv v logu mělo na mě zbytečně velké řádkování.
6 (A)	Po stránce architektury je aplikace rozdělena na webovou službu a program běžící u uživatele, který obstarává připojení k fyzické sběrnici. Toto rozdělení je reflektováno i v uživatelském rozhraní, což však z uživatelského pohledu nepovažuji za příliš intuitivní. Tento problém se však týká především počáteční konfigurace; uživatelské rozhraní při záznamu a zpracování dat je již celistvé.

Tabulka E.3: Odpovědi na otázku č.3

E.4 Je nějaká funkcionalita, která vám v aplikaci chyběla a bylo by vhodné ji doplnit?

Účastník	Odpověď
1 (S)	Automatický load Jason suboru by bol dobrý, Viac linek do jedného grafu, Defaultne custom windows, Preskakovaní v logu, Faster replay speed, Periodické posielanie správ, Vyhledávaní na základe parametrú nejaké zprávy, Export/import nastavení custom windows
2 (S)	Exportování grafů a vlastních oken v nějakém snadno editovatelném formátu (ideálně JSON). Možnost přeskakovat na libovolné časy v režimu Replay Rozšíření možností simulace ECU – například periodické odesílání dvou či více zpráv s předprogramovanými daty. Možné využití by bylo v okamžiku, kdy jedna zpráva obsahuje data z měření, ale vedle ní je druhá zpráva obsahující pravdivostní hodnotu indikující validitu měřených dat. Pro simulaci by se hodilo mít možnost nastavit periodické odesílání obou, ve Status zprávě navtrdo nastavit validitu na “true”, následně nastavovat hodnotu měření a sledovat vliv na testované elektronické systémy.
3 (N)	*Bez odpovědi*
4 (S)	Ukládání/stažení na/z cloud, export nastavení, zvukové efekty :D, export grafu do souboru a možná, měření času tzn. mít dva kurzory a někde v rohu číselně čas, y-kurzor, změna času z relativního na absolutní (mít možnost změny).
5 (A)	Součástí předchozí odpovědi.
6 (A)	Uvítal bych možnost zobrazení více signálů ve společném grafu (např. teploty měřené v různých bodech)

Tabulka E.4: Odpovědi na otázku č.4

E.5 Dle vašeho názoru, jak pravděpodobné je, že bude aplikace CANgui používaná v prostředí týmu eForce?

Účastník	Odpověď
1 (S)	Velmi pravděpodobné
2 (S)	Velmi pravděpodobné
3 (N)	Velmi pravděpodobné
4 (S)	Velmi pravděpodobné
5 (A)	Velmi pravděpodobné
6 (A)	Velmi pravděpodobné

Tabulka E.5: Odpovědi na otázku č.5

E.6 Další poznámky k aplikaci

Účastník	Odpověď
1 (S)	Super práce
2 (S)	I s velkým množstvím zpracovávaných dat funguje aplikace hladce a responzivně. Výborné jsou notifikace v případě detekce problémového stavu pomocí skupiny Fault Indicators. Ve srovnání s uživatelským rozhraním terminálu se jedná po grafické stránce o obrovský skok kupředu pro moui každodenní práci. Nastavování vlastních oken i grafů je velice přímočaré a po dvou třech zopakováních se člověk cítí jako doma. Zobrazování syrových hexadecimálních dat vedle dekodovaných je pro programátora velmi příjemné. Mechanismus importu a exportu logů by snad již nemohl být jednodušší s použitím drag&drop a klávesové zkratky ctrl+S. Fantastická je možnost sdílení jedné Ocarina session mezi mnoha uživateli, díky které bude výrazně usnadněno vzdálené ladění či získávání pomoci starších členů. Možnost připojit k formuli jeden počítač pro interakci s Ocarinou a všechnu analýzu dat přesunout do vedlejší místnosti bude nedocenitelná v okamžiku, kdy by na formuli potřebovali naráz pracovat elektrikáři i mechanici a navzájem by si překáželi. Přihlašování na SSO je velmi jednoduché a skvěle zapadá k ostatním týmovým nástrojům. V neposlední řadě si cením přehledné a hlavně transparentní konfigurace, například výběr JSON definující formát zpráv.
3 (N)	Velmi se mi líbila možnost nastavit dark theme. Velmi uživatelsky přívětivá aplikace.
4 (S)	Aplikace je velmi rychlá na to že běží v prohlížeči, grafy vypadají velmi hezky, aplikace je velmi použitelná ale nejvíce zajímavé je vzdálené připojení.
5 (A)	Pokud bude aplikace nadále rozvíjena, nemá konkurenci.
6 (A)	Celý koncept aplikace je nepochybně významným krokem vpřed v rámci týmu eForce, nicméně ani mezi komerční "konkurencí" nevím o nástroji, který by takto kombinoval moderní rozhraní s pokročilými funkcemi. Velkým přínosem je také hluboká integrace s týmovým ekosystémem, která by při použití cizího řešení byla velmi náročná vzhledem k absenci otevřených standardů v oblasti diagnostiky sběrnice CAN. Nabízí se také široké možnosti dlouhodobého rozvoje (např. trvalý záznam dat do databáze)

Tabulka E.6: Odpovědi na otázku č.6

E.7 Souhlas se zpracováním vyplněných údajů

Účastník	Souhlas
1 (S)	Ano
2 (S)	Ano
3 (N)	Ano
4 (S)	Ano
5 (A)	Ano
6 (A)	Ano

Tabulka E.7: Souhlas účastníků se zpracováním odpovědí

Příloha F

Zdrojové soubory

Zdrojové soubory pro obě části aplikace jsou v digitální formě samostatně odevzdané s prací v rámci souboru **CANgui-zdrojove-soubory.zip**

Příloha G

Spustitelný Ocarina Klient

Spustitelná část aplikace *Ocarina Klient* je v digitální formě samostatně odevzdaná s prací v rámci souboru **ocarinaclient-1.0.0-RELEASE.jar**

Příloha H

Spustitelný CANgui Server

Spustitelná část aplikace **CANgui Server** (lokální verze) je v digitální formě samostatně odevzdaná s prací v rámci souboru **cangui-1.0.0-RELEASE.jar**.