TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Datalogging hybridních vozů

Bakalářská práce

Studijní program: B6209 – Systémové inženýrství a informatika
Studijní obor: 6209R021 – Manažerská informatika

Autor práce: **Matěj Slabihoud**
Vedoucí práce: Ing. David Kubát, Ph.D., Ing.Paed.IGIP

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Akademický rok 2018 / 2019

Zadání bakalářské práce

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Matěj Slabihoud**
Osobní číslo:	E16000542
Studijní program:	B6209 Systémové inženýrství a informatika
Studijní obor:	B6209R021 – Manažerská informatika
Zadávající katedra:	katedra informatiky
Vedoucí práce:	Ing. David Kubát, Ph.D., ING.PAED.IGIP
Konzultant práce:	Ing. Vít Bečvář ŠKODA AUTO a. s.,

Název práce: **Datalogging hybridních vozů**

Zásady pro vypracování:

1. Stanovení cílů.
2. Základní popis dataloggingu a hybridních vozů.
3. Úvod do programovacího jazyka LTL.
4. Vytvoření konfigurace pro datalogging hybridních vozů.
5. Vytvoření podkladů pro externí firmu k analýze naměřených dat z hybridních vozů.
6. Popis výsledků analýzy.
7. Formulace závěrů.

Seznam odborné literatury:

EMPLEMAN, Graham. 2008. T*he Competition Car: Data Logging Manual*. Pundbury: Veloce Publishing. ISBN 9781845841621.

WATTERSON, James M. 2011. *Diagnostic Skills*. Aylesbury: ShieldCrest. ISBN 9781907692099.

CHIMATA, Raghuveer a Rajesh SINGH, et al. 2018. *Internet of Things in Automotive Industries and Road Safety: Electronic Circuits, Program Coding and Cloud Servers.* Delft: 2018

DENTON, Tom. 2016. *Electric and Hybrid vehicles*. New York: Routledge.

SIKORA, A. et al. 2014. *Communication Technologies for Vehicles.* 6th ed. New York: Springer. ISBN 9783319066431.

PROQUEST. 2018. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2018-09-30]. Dostupné z: http://knihovna.tul.cz/

Rozsah práce:	30 normostran
Forma zpracování:	tištěná / elektronická
Datum zadání práce:	31. října 2018
Datum odevzdání práce:	31. srpna 2020

L.S.

prof. Ing. Miroslav Žižka, Ph.D.
děkan Ekonomické fakulty

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

15. 4. 2019 Matěj Slabihoud

Poděkování

Chtěl bych poděkovat panu Ing. Davidu Kubátovi, Ph.D., ING.PAED.IGIP za odborné vedení práce. Dále děkuji firmě Škoda Auto a.s., která mi poskytla zařízení potřebná pro vypracování této bakalářské práce. Rovněž bych na tomto místě rád poděkoval rodičům, že mě vždy během psaní bakalářské práce podrželi a stáli po celou dobu při mně. V neposlední řadě bych rád poděkoval celému oddělení GQM-3/1 za cenné rady, které mi pomohly tuto práci zkompletovat.

Anotace

Bakalářská práce se zaměřuje na datalogging hybridních vozů pomocí elektronického zařízení, které se nazývá datalogger. V teoretické části práce jsou popsány dataloggery od společnosti G.i.N a programovací jazyk Log Task Language, který slouží k jejich konfiguraci. Poslední kapitola teoretické části se věnuje historii hybridních pohonů a jejich základnímu rozdělení. Praktická část práce se týká samotné konfigurace pro datalogging hybridních vozidel. V poslední části celé bakalářské práce je vytvořeno zadání pro externí firmu. Ta má za úkol zhotovit výstupní zprávy z jednotlivých měření získaných na základě mnou vytvořené konfigurace.

Klíčová slova

Datalogger, datalogging, hybrid, hybridní pohon, hybridní vůz, LTL, Log Task Language, G.i.N, CAN

Annotation

Title: Datalogging of Hybrid Vehicles

This bachelor thesis focuses on datalogging of hybrid vehicles with the help of a electronic device that is called datalogger. The theoretical part describes dataloggers from G.i.N company and the LTL programming language, which is used for their configuration. The last chapter of the theoretical part deals with the history of hybrid drive systems and their basic division. The practical part concerns the configuration used for datalogging of hybrid vehicles. In the last part of this bachelor thesis there is an assignment created for an external company. Their task is to do output reports from individual measurements obtained with my configuration.

Keywords

Datalogger, datalogging, hybrid, hybrid drive system, hybrid vehicle, LTL, Log Task Language, G.i.N, CAN

Obsah

SEZNAM OBRÁZKŮ

SEZNAM POUŽITÝCH ZKRATEK, ZNAČEK A SYMBOLŮ

AUX ..... Auxiliary connector
CAN ..... Controller Area Network
CCP ..... CAN Calibration Protocol
COD ..... Soubor obsahující zkompilovaný zdrojový kód
FPS ..... Frames Per Second
GPS ..... Global Positioning System
LIN ..... Local Interconnect Network
LTL ..... Log Task Language
MOST ..... Media Oriented Systems Transport
PHEV ..... Plug-in Hybrid Electric Vehicle
TTL ..... Transistor-Transistor-Logic
UART ..... Universal Asynchronous Receiver/Transmitter
UMTS ..... Universal Mobile Telecommunications Service
XCP ..... Universal Measurement and Calibration Protocol

Úvod

Mít kontrolu nad celou jízdní zkouškou a nad dynamickými vlastnostmi vozidla je v dnešní době nepostradatelnou součástí analýzy vozu před jeho uvolněním do sériové výroby. O to větší váhu má měření získané z vozidla na hybridní pohon, jelikož se do řídící jednotky motoru odesílá mnohem více CAN zpráv.

Měření je možné zaznamenat pomocí speciálního zařízení, které se nazývá datalogger. Díky tomuto zařízení tak můžeme sledovat a zaznamenávat komunikaci mezi jednotlivými jednotkami motoru. Analýza výsledného měření poté může napomoct při řešení jak chyb v komunikaci, tak při sporadických chybách. Takové chyby vyvolají takzvaný trigger, který spustí nahrávání zpráv ze všech zvolených CAN sběrnic. Trigger může být vyvolán chybovou zprávou, zaslanou přímo z řídící jednotky vozidla, stisknutím tlačítka řidičem přímo během jízdní zkoušky nebo například dosažením hraniční hodnoty sledovaného signálu.

Cílem této práce je vytipování nejvhodnějšího dataloggeru od společností G.i.N pro datalogging hybridních vozidel a seznámení se s programovacím jazykem Log Task Language, který slouží ke konfiguraci dataloggerů. Dalším cílem je přiblížení problematiky hybridních pohonů. Cílem praktické části je poté vytvoření konfigurace pro datalogger podle požadavků zadaných oddělením GQM-3/1 z firmy Škoda Auto a.s. a návrh výstupní zprávy, která ulehčí následnou analýzu vybraného měření. Tento návrh bude následně předán externí firmě ke zpracování.

1 Datalogger

Datalogger je elektronické zařízení, které slouží ke snímání signálu z různých typů zařízení pomocí senzorů. Data zaznamenaná pomocí senzorů jsou v reálném čase ukládána do paměti dataloggeru.

1.1 Oblasti použití dataloggeru

Datalogger je využíván v mnoha odvětvích, ve kterých je potřeba snímat větší množství dat, která jsou poté určena k další analýze.

Mimo automobilového průmyslu, kterému se v této bakalářské práci budeme věnovat, je datalogger hojně využíván například v meteorologických stanicích nebo při kontrole hladiny vod. Tato data jsou poté vyhodnocena a je díky nim možné predikovat různé situace, které by mohly nastat.

Se zmíněným automobilovým průmyslem souvisí také datalogging silničního provozu. Díky různým senzorům je možné dlouhodobě sledovat situaci na silnicích a ze získaných dat poté například určit časy dopravních špiček. Informováním řidičů je následně možné takovým situacím alespoň částečně předejít.

1.2 Datalogging v automobilovém průmyslu

Z počátku se využívaly pro datalogging v automobilech dataloggery, které snímaly veškeré signály z jednotlivých sběrnic nebo jiných vstupů. Omezené bylo také množství příslušenství, které bylo možné k dataloggeru připojit. Takové dataloggery byly vhodné, dokud se snímalo pouze malé množství signálů.

S postupným přibýváním jednotlivých senzorů a tím pádem i signálu se vyplatilo přejít na dataloggery postavené na vlastním, převážně Linuxovém systému. Takové dataloggery se již mohly konfigurovat a zvětšilo se i spektrum možných vstupních i výstupních zařízení, které bylo možné k dataloggeru připojit.

Díky konfiguraci dataloggeru je možné určit kdy, jak a proč se budou požadované signály snímat. Tímto způsobem se tak zamezí zahlcování paměti daty, která nejsou v danou chvíli potřebná nebo umožní snímat data pouze při určitém triggeru.

Trigger je předem definovaná změna stavu, po které začne datalogger zaznamenávat jednotlivé signály. Nejčastěji se využívá triggeru v podobě tlačítka, chybové zprávy nebo předem definované události (překročení rychlosti, kickdown, …). V případě, že nastane

některý z těchto triggerů nebo řidič automobilu v momentě sporadické události stiskne tlačítko, dojde k uložení signálů zaznamenaných za předem definovanou dobu z dočasné paměti do paměti dataloggeru.

2 Vytipování vhodného dataloggeru

V této kapitole si ukážeme jednotlivé dataloggery od firmy G.i.N, která úzce spolupracuje se Škoda Auto a.s.. V další části si poté zvolíme datalogger, který bude pro potřebná měření v rámci praktické části nejvhodnější.

2.1 Dataloggery firmy G.i.N

Firma Škoda Auto a.s., která mi poskytla možnost využívat zařízení, která jsou zapotřebí pro úspěšné vypracování této bakalářské práce, pracuje s dataloggery od firmy G.i.N. Z toho důvodu se zaměříme na dataloggery nabízené touto firmou.

2.1.1 Datalogger MultiLog

Datalogger MultiLog má vestavěný procesor s nízkou spotřebou energie a je založen na operačním systému Linux s rychlým spouštěním. Záznam signálů umožňuje ze sběrnic CAN, MOST a dvoukanálového FlexRay. Dále je možné snímat až osm kanálů LIN pomocí LINprobe.

MultiLog je volně programovatelný pomocí programu GINconf dodávaný na CD společně s dataloggerem. Pomocí konfigurace lze volně programovat pět LED indikátorů a osmiznakový alfanumerický displej.

Dále disponuje osmi digitálními vstupy i výstupy, dvěma TTL vstupy a výstupy pro připojení příslušenství, mezi které patří například tlačítko nebo přídavné LED indikátory.

Data jsou ukládána na vnitřní paměť o velikosti 100 MB a může být rozšířena pomocí Compact Flash karty o maximální velikosti 32 GB. Tento datalogger má dočasnou paměť RAM o velikosti 2 x 15 MB.

Komunikace a přenos dat je možný pomocí WLAN, USB 2.0 nebo přes CF kartu.

Díky robustní hliníkové skříni je možné datalogger využívat při teplotách v rozmezí od -40 °C do 70 °C.

2.1.2 Datalogger GL1xxx

Dataloggery první série GL1xxx disponují vlastním operačním systémem a díky programovacímu jazyku na vysoké úrovni je možné plně pracovat s triggery a systémovými objekty, jako jsou například proměnné, časovače, čítače a další. Programování je možné pomocí programu GINconf.

Signály snímá ze sběrnic CAN, LIN, analogových kanálů a také ze sériového portu RS-232. Konkrétně se jedná o dva fyzické CAN kanály, šest virtuálních CAN kanálů, dva fyzické LIN kanály a čtyři analogové kanály. Dále umožňuje snímat CCP/XCP signály.

Na přední straně jsou umístěny čtyři volně programovatelné LED diody a jedna určená pro indikaci USB připojení. Možné je také využít zabudovaného akustického generátoru pro zvukovou signalizaci.

Na rozdíl od MultiLogu nemá vnitřní paměť a záznam je ukládán na SD kartu o velikosti 2 GB, která je součástí balení. Z toho důvodu není vhodný pro dlouhodobé analýzy a může sloužit pouze pro testování vozidla na krátkých trasách.

Komunikace a přenos dat je možný pomocí vysokorychlostního USB 2.0 (480 Mbit/s).

Velkou výhodou je jeho kompaktní hliníkové provedení s odolností vůči vysokým teplotám (až do +85 °C v závislosti na použité SD kartě).

2.1.3 Datalogger GL2xxx

Další série dataloggerů od společnosti G.i.N nese označení GL2xxx. Vlastnostmi a funkčností je velmi podobný předchozím modelům ze série GL1xxx. Nicméně oproti nim snímá již signály ze čtyř fyzických a jedenácti virtuálních CAN kanálů. Dále je obohacen o AUX CAN rozhraní pro připojení příslušenství.

Na přední straně přibyla LED dioda zapnutí dataloggeru a z boku jedna další pro upozornění v případě absence nebo chyby paměťové karty.

Snímaná data mohou být ukládána jak na SD kartu, tak nově na SDHC kartu o maximální velikosti 32 GB.

Opět se jedná o menší datalogger hliníkové konstrukce. Velikostně se nemůže rovnat předchozí sérii, nicméně funkčně je na tom o dost lépe.

2.1.4 Datalogger GL3xxx/GL4xxx

S příchodem sérií GL3xxx a GL4xxx dochází k rapidnímu zvýšení výkonu a počtu kanálů, které je možné snímat. O dostatečnou výpočetní rychlost se starají dva ARM9 procesory. Stejně jako Multilog je operační systém založen na Linuxu.

Dataloggery těchto sérií disponují devíti fyzickými CAN rozhraními, sedmi virtuálními CAN kanály a dvěma LIN rozhraními, které je možné rozšířit o šestnáct LIN kanálů pomocí LINprobe. U dataloggerů GL4000/GL4200 je navíc možné snímat ze dvou FlexRay kanálů.

Díky dvěma speciálním AUX konektorům je možné připojit externí příslušenství, jako je například displej LOGview nebo 3G anténa pro přenos dat.

Dataloggery z těchto sérií (kromě GL3000) mají osmimístný alfanumerický display a je možné využít čtyř programovatelných event tlačítek. Dále pro indikaci slouží pět LED diod, které jsou také volně programovatelné.

Snímaná data jsou ukládána v případě dataloggerů GL3000/GL3100 a GL4000 na CF kartu. Co se týká verzí GL3200 a GL4200, ty už oproti tomu disponují 2.5 palcovým SATA SSD.

Pro komunikaci a přenos dat je možné využít WLAN, UMTS, USB, CF kartu nebo SSD.

Oproti GL1xxx a GL2xxx došlo k nárůstu velikosti a váhy, která je samozřejmě způsobena lepší výbavou a možnostmi dataloggerů.

2.1.5 Datalogger GL5xxx

Nejnovější série dataloggerů nese označení GL5xxx. O výpočetní rychlost se starají dva čtyřjádrové procesory Cortex-A9 Quad Core. Systém tohoto dataloggeru je opět založený na Linuxu s rychlým spouštěním. Novinkou je zde funkce dispatcher, která má za úkol monitorovat celý systém, aby nedocházelo k chybám.

Dataloggery GL5xxx disponují až dvaceti čtyřmi CAN rozhraními a jsou obohaceny o jeden AUX CAN. Dále je možné využít až šest LIN a osm UART rozhraní. Všechny verze série zvládají pracovat se dvěma kanály FlexRay a pomocí externího zařízení GLA150 lze komunikovat přes MOST150. Nově přibyla možnost připojení přes ethernet a to díky pěti ethernetovým portům pro koncovku RJ45. Samozřejmostí jsou i USB porty pro komunikaci s PC.

Pro zobrazení textu disponuje datalogger OLED displayem o velikosti 128 x 32 pixelů. Primárně slouží pro zobrazení menu, nicméně je možné celý display využít pro zobrazení vlastního textu. Pro pohyb v menu lze využít čtyř event tlačítek, která jsou plně programovatelná. Stejně jako u předchozí série se na dataloggeru nachází pět LED diod, které jsou také volně programovatelné.

Rozšířil se také počet konektorů pro připojení externího příslušenství. Dataloggery série GL5xxx mají dva AUX konektory pro připojení například externího displeje LOGview, 3G routeru nebo tlačítka. Dále disponuje AUX+, díky kterému je možné připojit zařízení GLA150 pro záznam MOST eventů.

Komunikovat s dataloggerem a přenášet nasbíraná data lze pomocí LAN/WLAN, UMTS, USB nebo přímo přes SSD.

Bohužel za nevýhodu se již může považovat jeho váha, která je cca 3500g. Nicméně jelikož došlo k rozšíření možností dataloggerů série GL5xxx, je samozřejmé, že velikost nemůže být stejná jako u předchozích modelů.

2.2 Vytipování vhodného dataloggeru

Vzhledem k možnostem, které mi Škoda Auto a.s. poskytla a samozřejmě také vzhledem k velkému množství senzorů, které současné automobily mají a tím pádem i velkému množství signálů, které se musejí zpracovávat, pro datalogging hybridních vozů využijeme jeden z nejnovějších a pro naše potřeby nejvhodnější datalogger typu G.i.N GL5350.

3 Program GINconf

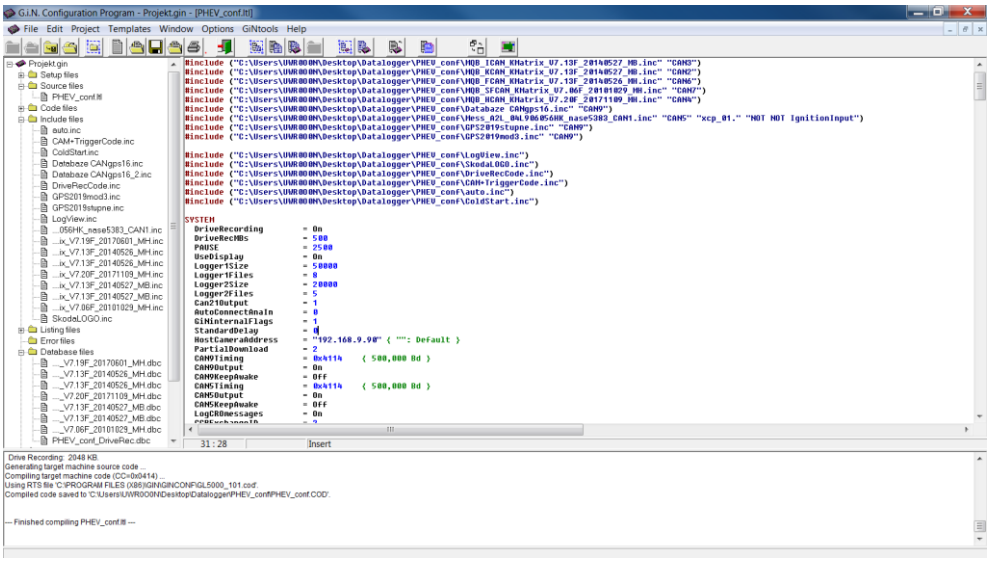
GINconf je program od společnosti G.i.N pro konfiguraci a čtení dat ze všech dataloggerů značky G.i.N. Primárně slouží ke tvorbě konfigurací a její následné kompilaci do formátu COD, který je možné pomocí konfiguračního programu nahrát přímo do dataloggeru. Reversním způsobem lze také získat použitou konfiguraci z dataloggeru zpět do konfiguračního programu. Kompilátor je dále schopný zobrazovat jednotlivé chyby a varovné hlášky, které ulehčují práci s konfiguracemi.

Kromě zdrojového kódu je možné do dataloggeru díky programu GINconf nahrát instalační soubory například s licencemi, aktualizovat firmware nebo připojit databázové soubory v podobě include souborů.

3.1 Okno programu GINconf

V této podkapitole je popsána samotná podoba aplikace. V hlavičce okna aplikace je možné vidět název samotného projektu a dynamicky je zde také zobrazována cesta k momentálně upravovanému souboru, který je součástí projektu. V hlavním menu se nachází jednotlivé záložky pro práci s aplikací. Panel nástrojů je umístěn přímo pod ním a jednotlivé nástroje jsou popsány v další části práce.

Zbytek okna je rozdělen na tři části. Na levé straně je stromové zobrazení celého projektu rozděleného do jednotlivých složek. Vpravo je editor pro psaní zdrojového kódu konfigurace a ve spodní části se zobrazuje výstup po kompilaci zdrojového kódu, kde je možné vidět celý průběh kompilace, případně chyby nebo varovné hlášky.



Obrázek 1 - Náhled okna programu GINconf

3.1.1 Hlavní menu

V hlavním menu, které je umístěno přímo mezi hlavičkou okna aplikace a panelem nástrojů, jsou tyto položky:

- File
    - V menu *File* je možné měnit nastavení projektu, paketů a souborů. Je možné zde vytvořit, otevřít, zavřít nebo uložit nový projekt, soubor nebo paket. Kromě klasického „x“ v pravém horním rohu okna aplikace, lze uzavřít okno pomocí tlačítka *Exit*, které se zde také nachází.
- Edit
    - V tomto menu se nachází funkce pro práci s editorem. Kromě klasických funkcí, kterými jsou *Cut/Copy/Paste* (Vyjmout/Kopírovat/Vložit), je tu tlačítko *Find* pro vyhledávání nebo *Replace* pro nahrazení konkrétního slova jiným slovem v celém souboru. Dále se mezi ně řadí například funkce *Undo* pro zrušení poslední provedené změny a *Select all* pro vybrání celého obsahu okna.
- Project
    - Pod záložkou *Project* se nachází menu s funkcemi pro nastavení projektu, konfigurace a pro ovládání zařízení. Nejdůležitější jsou tu funkce pro nastavení vlastností projektu, kompilátoru a nahrávání dat do dataloggeru. Je možné skrz toto menu spustit kompilaci nebo nahrát data do dataloggeru, nicméně tyto operace lze provést také přímo z panelu nástrojů.
- Templates
    - Díky tomuto menu je možné zvolit nejčastěji používané nastavení nebo si nechat vygenerovat části kódu. Jde například jednoduše definovat jednotlivé konstanty, proměnné nebo výstupy. Dále lze jedním kliknutím vygenerovat základní trigger nebo definovat jednoduchou klasifikaci.
- Window
    - Toto menu slouží pouze pro nastavení rozložení okna editoru a to kaskádově, vertikálně nebo horizontálně.
- Options
    - Pomocí tohoto menu lze nastavit všechny možnosti, jako jsou například nastavení editoru a jazyka programu. Navíc se zde nachází funkce pro manuální aktualizaci programu GINconf a je možné přes toto menu nahrát licenční programy do připojeného dataloggeru.

- GiNtools
    - Slouží pouze pro spuštění nainstalovaných podprogramů GiNconf, jako je například Visual LTL nebo MLcenter (G.i.N multi logger software).
- Help
    - Pomocí tohoto menu je možné zobrazit veškeré manuály. Nachází se zde manuály pro G.i.N konfigurační program, manuál programovacího jazyka LTL a manuály ke všem zařízením které firma G.i.N nabízí. Pro zobrazení manuálů je nutné mít nainstalovaný program pro prohlížení PDF souborů.

3.1.2 Toolbar

Toolbar neboli panel nástrojů umožňuje rychlý přístup k nejčastěji používaným funkcím.

V této části kapitoly jsou popsány jednotlivé funkce z panelu nástrojů, které jsou při programování konfigurace nejčastěji používány.

Prvních jedenáct funkcí z panelu nástrojů slouží převážně pro práci s projektem a jednotlivými soubory.

- Nový projekt
    - Tato funkce slouží pro vytvoření nového projektu. Je nutné definovat základní nastavení.
    - V prvé řadě je žádoucí nastavit cestu, kam bude projekt uložen. Primárně je cesta nastavena na složku, kde je program GINconf nainstalován, nicméně je možné jí změnit pomocí tlačítka *„Change Path“*.
    - V další části je možné nastavit typ zařízení. Na výběr je ze všech zařízení, které firma G.i.N nabízí. Lze zvolit automatickou detekci zařízení nebo vybrat konkrétní zařízení přímo ze seznamu.
    - V interface definujeme komunikační rozhraní, přes které je zařízení připojeno.
    - V neposlední řadě se musí nastavit přenosová rychlost.
    - Timeout necháváme standardní a je možné ho změnit přímo v kódu konfigurace
- Otevření projektu
    - Otevírá dialogové okno, ve kterém je možné vybrat projektový soubor, který má být otevřen. Takový soubor má příponu **.gin* a po otevření se zobrazí v okně aplikace.
- Uložit projekt

    - Uloží rozpracovaný či hotový projekt a jeho přidružené soubory.
- Zavřít projekt
    - Uzavře celý projekt. Uživatel je i přesto dotázán, zdali chce projekt uložit či nikoli.
- Možnosti projektu
    - Otevře dialogové okno s nastavením projektu. (viz Nový projekt)
- Nový soubor
    - Otevře nový prázdný soubor, který je možné při ukládání nastavit jako *source*, *setup* nebo *include* soubor.
- Přidat soubor
    - Pomocí dialogového okna, které se zobrazí při zvolení této funkce lze přidat do stávajícího projektu nový soubor.
- Uložit soubor
    - Uloží současně používaný soubor otevřený v editoru. Neuloží celý projekt.
- Odebrat soubor
    - Odebere vybraný soubor z projektu.
- Tisk souboru
    - Vytiskne vybraný soubor.
- Exit
    - Uzavře konfigurační program G.i.N..

V další části panelu nástrojů nalezneme funkce týkající se kompilace a samotného nastavení kompilátoru.

- Možnosti kompilátoru
    - Otevře dialogové okno s nastavením možností kompilátoru. Možné je nastavit například výstupní formát souboru nebo zapnout automatické vytváření databázového souboru.

- Kompilace
    - Zahájí kompilaci právě zobrazeného souboru.
- Kompilovat a nahrát
    - Zahájí kompilaci právě zobrazeného souboru a následně nahraje konfiguraci do připojeného zařízení.

- Kompilovat a vytvořit konfigurační EXE soubor
    - Zahájí kompilaci a následně vytvoří EXE soubor pro automatickou konfiguraci.
- Možnosti nahrávání konfigurace
    - Otevře dialogové okno s možnostmi nahrávání konfigurace. Jde například o nastavení změny času dataloggeru na čas počítače nebo automatické odpojení dataloggeru po ukončení nahrávání konfigurace.
- Stažení konfigurace
    - Umožňuje stažení **.COD* konfiguračního souboru a jeho následný převod do zdrojového kódu.
- Export
    - Umožňuje export souboru ve formátu **.clf*.

Poslední část Toolbaru se týká samotného vzhledu editoru.

- Přepnout editor mezi normální a maximalizovanou podobou
    - Přepne editor z normální do maximalizované podoby a naopak.
- Otevře nebo zavře eLOGview
    - Zobrazí okno s virtuálním eLOGview.

4 Úvod do LTL – Log Task Language

V této kapitole budou popsány základy jazyka LTL a ukázány jednotlivé prvky, které budou následně použity v praktické části této bakalářské práce.

4.1 Základní objekty

4.1.1 Konstanty

Konstanta přiřadí jméno pevně danému číslu. Tuto konstantu je poté možno použít i při práci s číselným datovým typem. Konstanty jsou definovány v sekci CONST v konfiguračním souboru.

Tento soubor může obsahovat neomezený počet konstant. Jednotlivé konstanty se stejným jménem mohou být v konfiguračním souboru předefinovány.

Každá konstanta má defaultní velikost o 16 bitech, nicméně může být předefinovaná pomocí hranatých závorek s požadovanou velikostí.

4.1.2 Systémové konstanty

Oproti klasickým konstantám se systémové konstanty zapisují do sekce SYSTEM a jsou předem definovány a není možné si definovat žádné vlastní. Tyto konstanty mohou nést názvy jiných objektů (klasických konstant, proměnných, atd…). Každá ze systémových konstant má předem definované defaultní hodnoty a není nutné tyto hodnoty měnit.

4.1.3 CAN zpráva

CAN zprávy jsou přenášeny přes CAN sběrnici a v LTL jsou definovány tímto způsobem:

```
CAN1 DATA 100
```

CAN1 je označení pro CAN kanál, DATA označují typ zprávy a 100 je CAN ID, který může nabývat hodnot od 0 do 2047.

4.1.4 Proměnné

Proměnné jsou jedním ze základních elementů konfigurace. Závisí na nich veškeré další části konfigurace, kterými jsou například triggery a klasifikace. Proměnné jsou definovány v sekci VAR.

Základními proměnnými jsou takzvané FREE Variables, kdy jejich velikost je určena v hranatých závorkách za danou proměnnou.

```
VAR
	Promenna = FREE [16]
```

V případě, že není velikost uvedena, je defaultně nastavena na 16 bitů.

Systémové proměnné

Kromě klasických proměnných existují i systémové proměnné, které jsou předem definované systémem. Jedná se například o proměnné pro čas.

CAN proměnné

Hodnota proměnné je odvozena z obsahu jednotlivých datových CAN zpráv. Přiřazení názvu takové zprávě a vytvoření tak proměnné, se kterou jde následně pracovat, se zapíše tímto způsobem:

```
VAR
      Název proměnné  =  CAN-Kanal  Data-typ  ID  [Data]
```

Název proměnné je volitelný, ale musí být unikátní. Každé části řídící jednotky je přiřazen jeden CAN-Kanál, ze kterého se poté sbírají jednotlivá data z různých senzorů. Na každém CAN-Kanálu se přenáší velké množství dat, a proto je nutné přesně určit s jakým typem dat chceme pracovat. V závorce se poté určuje, jaké bity se mají z vybrané části zprávy uložit do proměnné.

Určování jednotlivých CAN proměnných je velice náročné, ale v mém případě budu mít k dispozici databáze, kde jsou proměnné pro každý ze signálů definované.

Modifikátory proměnných

Stejně jako v jiných programovacích jazycích si i v tomto můžete určit, zdali bude proměnná takzvaně se znaménkem nebo bez něj. Zápis je stejný, UNSIGNED určí, že minimální možnou hodnotou bude 0. V případě 8bitové proměnné tedy bude možné definovat proměnnou v rozsahu od 0 do 255. V případě SIGNED bude možné použít i záporné hodnoty. Opět v případě 8bitové proměnné bude rozsah od -128 do -127.

Dalšími modifikátory jsou FACTOR a OFFSET.

```
VAR
      Proměnná = CAN 1 DATA 100 [0] FACTOR = 2 OFFSET = 5
```

FACTOR určuje, kolikrát se hodnota proměnné zmenší nebo zvětší. OFFSET namísto součinu hodnot funguje na bázi součtu. Říká tedy, o kolik se hodnota proměnné zvětší. Zde můžete vidět ukázku:

```
Fyzická hodnota = (Nezpracovaná hodnota * FACTOR) + OFFSET
```

Posledním modifikátorem, který využiji v praktické části je INIT, který definuje inicializační hodnotu proměnné.

```
VAR
	Proměnná = CAN 1 DATA 100 [8] INIT = 20
```

4.2 Podmínky

Podmínky se v případě jazyka LTL zapisují do závorek. Je možné určit podmínku již přímo u proměnné nebo až následně u nějakého výrazu kdekoliv ve zdrojovém kódu.

4.2.1 Jednoduché podmínky

Jednoduchými podmínkami se myslí podmínka, kde se nepracuje s logickými operátory. Porovnávají se tak pouze dvě strany, v případě že je podmínka splněna, dojde k vykonání podmíněného výrazu, pokud není splněna, výraz se ignoruje a vyhodnocuje se další řádek. V následujících příkladech jsou uvedeny některé z možných zápisů podmínky:

```
(OilPressure >= NOT CriticalPressure)   {Tlak oleje není příliš vysoký}
(Speed - SecondeTachometer > 5)         {Rozdíl rychlosti je příliš vysoký}
(DoorsOpen = 0x0F)                      {Všechny dveře jsou otevřeny}
(Speed <> SecondeTachometer)            {Odchylka mezi dvěma tachometry}
```

4.2.2 Komplexní podmínky

Komplexní podmínky se od jednoduchých liší tím, že jsou složeny z více podmínek pomocí logických operátorů. Využít můžeme dvou logických operátorů a to AND (logický součin) nebo OR (logický součet). Zde je názorný příklad:

```
(OilPressure < 20  AND  NoDoorOpen) OR (Speed >= 50)
```

Ke splnění této podmínky dojde ve dvou případech. A to v případě, že tlak oleje bude menší než 20 a zároveň nebudou otevřeny dveře, nebo že rychlost bude větší nebo rovno 50.

4.3 Složené objekty

Za složené objekty se považují například FLAGS, COUNTERS nebo TIMEOUTS. Tyto objekty budou popsány v této části práce.

4.3.1 FLAGS

Objekt definovaný v sekci FLAG slouží pro uchování hodnoty a určení změny stavu pomocí bitové proměnné. Umožňuje tak zachovat informaci o předešlém stavu proměnné a jeho změnách. A jeho zápis je takový:

```
FLAG
	VysokaRychlost		SET = (Speed >= 100)
				RESET = (Speed < 100)
```

V případě překročení rychlosti 100 Km/h, dojde k nastavení FLAG VysokaRychlost na „1“ a tuto hodnotu si bude držet, dokud se rychlost opět nesníží pod 100 Km/h.

4.3.2 Counters

Counter neboli čítač je bitový objekt, který může nabývat hodnoty 0 (status OK) při nedosažení předem definovaného limitu nebo hodnoty 1 (status Overrun) při dosažení zmíněného limitu. Zde je názorný příklad zápisu:

```
CONST
       Max = 30
       SpeedLimit = 100
COUNTER
       PrekrocenaRychlost LIMIT = Max (Speed > SpeedLimit)
```

Dokud není rychlost překročena třicetkrát je status čítače PrekrocenaRychlost OK, v případě, že se tento limit překročení rychlosti překoná, nastaví se hodnota čítače na jedna neboli na status Overrun.

4.3.3 Timeouts

Časové limity se používají pro monitorování časových vztahů mezi CAN zprávami. Stejně jako čítač je TimeOut bitový objekt a nabývá hodnoty 0 (Status OK) v případě, že doba mezi poslední přijatou zprávou a tou předchozí je v předem nastaveném limitu. Pokud dojde k překročení tohoto limitu, je hodnota objektu nastavena na 1 (Status Overrun). Zde je opět názorný příklad:

```
TIMEOUT ChybaTachometru    TIME = 100 (CAN1 DATA 200h)
```

V případě, že jsou po sobě jdoucí zprávy v časovém limitu maximálně 100 ms, je status objektu ChybaTachometru udržován na OK. V případě, že je mezi zprávami tento limit překročen, dojde k nastavení statusu Overrun.

4.4 Základní funkce

V této části kapitoly si uvedeme nejpoužívanější funkce jazyka LTL. Uvedeme si funkci pro výpočty a funkci pro nahrávání CAN zpráv z dataloggeru a její modifikace. A na závěr kapitoly se podíváme na funkci EVENT a popíšeme si jednotlivé typy této funkce, které budou následně použity v praktické části.

4.4.1 Funkce CALC

V sekci CALC je možné provádět veškeré výpočty a aritmetické operace s proměnnými typu FREE. V potaz se musí brát bitová velikost těchto proměnných, aby byl použit správný zápis.

V případě proměnných o velikosti jednoho bitu, který nabývá hodnot 0 a 1, musí být při přiřazení uvedena hodnota v závorkách, aby kompilátor rozpoznal, že se jedná o bitovou hodnotu.

```
VAR
	BitovaHodnota = FREE [1]
CALC
	BitovaHodnota = (0) {nebo}
	BitovaHodnota = (1)
```

Pro práci s proměnnými o velikosti 8 bitů nebo 16 bitů využíváme aritmetické výrazy.

```
CONST
	B = 10
	C = 15
VAR
	A = FREE[8]
CALC
	A = (B+B)*C
```

V sekci CALC je možné také využít volitelného výrazu WHEN, díky kterému docílíme toho, že výsledek bude záviset na zvolené podmínce.

```
CALC
	X = A WHEN (A>B)
	X = B WHEN (A<B)
```

4.4.2 Funkce TRANSMIT

Díky funkci TRANSMIT může být požadováno odesílání dat periodicky i během nahrávání v jakékoliv části kódu. Pro použití této funkce musí být aktivovaný příslušný CAN výstup. Aktivaci provedeme pomocí systémové konstanty CanOutput nebo Can*Output, kde * nahradíme požadovaným číslem CAN sběrnice, který má být snímán.

```
SYSTEM
	Can1Output = On
EVENT
	ON CYCLE (10) BEGIN
		TRANSMIT CAN1 DATA 100 [Speed] WHEN (podmínka)
	END
```

Po aktivaci CAN výstupu a v případě, že je definována bitová adresa pro proměnnou Speed, jsou odesílány informace o rychlosti každých 10 ms, ale to pouze pokud je splněna námi zvolená podmínka.

Funkce TRANSMIT může být dále modifikována pomocí klíčových slov, kterými jsou například LOG a LOG ONLY.

```
EVENT
	ON CYCLE (100) BEGIN
		TRANSMIT CAN1 DATA 100 [1]
		TRANSMIT CAN1 DATA 200 [2] LOG
		TRANSMIT CAN1 DATA 300 [3] LOG ONLY
```

V případě tohoto příkladu, je zpráva 100 odesílána přes CAN1 každých 10 ms (bez nahrání do paměti dataloggeru). Zpráva 200 je oproti zprávě 100 navíc i nahrána do paměti dataloggeru. Na závěr zpráva 300 je pouze nahrána do paměti dataloggeru, ale není odeslána.

Poslední využívaná modifikace funkce TRANSMIT je označena klíčovým slovem ONCHANGE.

```
EVENT
	ON CYCLE (1000) BEGIN
		TRANSMIT CAN1 DATA 100 [Speed Kilometer] LOG ONLY ONCHANGE
	END
```

V tomto případě se zpráva nahraje do paměti dataloggeru, ale pouze pokud během posledního požadavku došlo ke změně požadovaných hodnot proměnných.

4.5 Události

V této kapitole budou popsány jednotlivé typy událostí neboli eventů, s kterými je možné pracovat, a mohou být zpracovány.

Struktura bloku kódu událostí byla naznačena v předchozí kapitole. Po klíčovém slovu EVENT se definuje typ události a v závorkách se uvádí požadované parametry, které jsou pro každý typ jiné. Blok kódu, kde jsou následně využívány funkce jazyka LTL, je uzavřen mezi klíčovými slovy BEGIN a END. V jedné sekci po klíčovém slovu EVENT je možné definovat více událostí o různých typech.

```
EVENT
	Typ události 1 (parametr) BEGIN
		Blok kódu
	END
	Typ události 2 (parametr) BEGIN
		Blok kódu
	END
```

4.5.1 ON CYCLE (...)

Tento typ události pracuje s interními hodinami zařízení. Parametrem je čas uvedený v milisekundách v rozsahu od 1 do 60000 ms. Zvolený parametr určuje, jak často se bude daná událost periodicky opakovat.

```
VAR n = FREE [16]
EVENT
	ON CYCLE (1000) BEGIN
		CALC n = n+1
	END
```

Tento příklad zapříčiní, že se proměnná n zvětší o 1 každých 1000 milisekund.

4.5.2 ON RECIEV (...)

Událost tohoto typu je spuštěna, pouze pokud je obdržena námi zvolená CAN zpráva, která se uvádí jako parametr události.

```
VAR n = FREE[16]
EVENT
	ON RECIEVE (CAN1 DATA 100) BEGIN
		CALC n = n+1
	END
```

Stejně jako v příkladu u předchozí události je zvyšována hodnota proměnné n o 1. Nicméně momentálně dojde k provedení výpočtu pouze při obdržení zprávy 100.

4.5.3 ON SET (...), ON RESET (...)

Tyto dva typy událostí jsou spuštěny při nastavení a resetování námi definovaného objektu typu FLAG.

```
VAR n = FREE[16]
FLAG StartMotoru  SET = (Otacky > 800) RESET = (Otacky = 0)
EVENT
	ON SET (StartMotoru) BEGIN
		CALC n = n+1
	END
```

Opět zvyšujeme hodnotu proměnné n o 1, ale tentokrát počítáme, kolikrát dojde k nastartování vozidla. Při nastartování vozidla dojde ke splnění podmínky události a hodnota proměnné n se zvýší o 1. K dalšímu zvýšení opět dojde až po vypnutí motoru a jeho opětovném nastartování.

4.5.4 ON CALC (...)

Tuto událost je možné použít například jako switch nebo se hojně používá k tvoření sekvenčních funkcí.

```
VAR
	a = FREE[16]
	b = FREE[16]
	c = FREE[16]
EVENT
	ON CYCLE (100) BEGIN
		CALC	a = a+1
			a = 0 WHEN (a = 2)
	END
	ON CALC (a = 1) BEGIN
		CALC	b = b + 1
	END
	ON CALC (a = 2) BEGIN
		CALC	b = b + 10
	END
```

V tomto příkladu jsou sekvenčně přičítány hodnota 1 a 10 k proměnné „b“ v závislosti na hodnotě proměnné „a“, která se mění každých 100 ms.

Tento zápis jde zapsat také tímto způsobem a výsledek je zcela stejný.

```
EVENT
	ON CYCLE (100) BEGIN
		CALC	a = a + 1
			a = 0		WHEN (a = 3)
			b = b + 1	WHEN (a = 1)
			b = b + 10	WHEN (a=2)
		END
```

4.5.5 ON SYSTEM (...)

U tohoto typu události můžeme použít mnoho různých parametrů, kterými jsou systémové proměnné.

- ON SYSTEM (Startup)
  - V případě použití tohoto parametru se při startu dataloggeru provede nejdříve vše, co je uvedeno v bloku této události.
- ON SYSTEM (PauseOver)
  - Při použití této události dojde k provedení operací v přilehlém bloku opět pouze při startu po uplynutí doby, na kterou je předem nastavená systémová proměnná PAUSE.
- ON SYSTEM (Logger1/2_Trigger)
  - Blok kódu, který náleží této události, se provede pouze tehdy, když dojde ve vyrovnávací paměti (Ring Buffer) ke změně z *„Trigger Lead“* na *„Trigger delay“,* což znamená, že podmínka STOP zapříčinila nahrávání.

- ON SYSTEM (Logger1/2_Stop)
    - Pokud dojde ve vyrovnávací paměti ke změně *„Trigger delay“* na stav *„zastaveno“*, dojde k provedení této události. V případě, že je *„Trigger delay“* nastaven na 0, vykoná se ihned po ON SYSTEM (Logger1/2_Trigger).
- ON SYSTEM (Logger1/2_Start)
    - V tomto případě je událost spuštěna v opačném případě než u *„Logger1/2_Stop“*. Událost se vykoná v případě, že dojde ke změně u podmínky START ze stavu *„zastaveno“* na *„nahrávání“* (Trigger Lead). Pokud je podmínka START konstantně ve stavu TRUE, událost bude provedena ihned po ON SYSTEM (Logger1/2_Stop).
- ON SYSTEM (Shutdown)
    - Příkazy uvedené v bloku této události budou provedeny těsně před vypnutím dataloggeru nebo jeho přechodu do režimu spánku. Všechny LOG ONLY zprávy nebo jiné záznamové výstupy budou dále generovány a tím pádem nedojde ke ztrátě dat.

4.5.6 Lokální objekty v bloku kódu událostí

Na závěr této kapitoly je možné zmínit, že veškerá deklarace vně bloku kódu, který je součástí události, není závislá na zbytku konfigurace. Tudíž objekty jako jsou například konstanty nebo proměnné, které jsou vytvořeny v bloku kódu, existují pouze v něm a nikde jinde se s nimi nedá pracovat. Za zmínění také stojí, že unikátnost názvů například proměnných se musí dodržovat pouze vně kódu, nicméně nemusí být unikátní pro celou konfiguraci.

```
VAR
	A = FREE[1]
EVENT
	ON CYCLE (100) BEGIN
		VAR A = FREE[8]
	END
	ON RECEIVE (CAN1 DATA 100) BEGIN
		VAR A = FREE[16]
	END
```

Taková část konfigurace by byla v pořádku zkompilována a nehlásila by žádnou chybu.

4.6 INCLUDE soubory

V konfiguraci pro jakýkoliv typ dataloggeru od společnosti G.i.N lze využít vkládání externích include souborů, které se připojují pomocí klíčového slova #include.

```
SYSTEM
	#include "include.txt"
CONST
	#include ("include.txt")
```

Takto vložené soubory jsou čteny zcela stejným způsobem, jako kdyby byl obsah těchto souborů vložen přímo ve zdrojovém kódu konfigurace.

4.6.1 #include s parametry

Vložené soubory je možné specifikovat pomocí určitých parametrů.

```
#include ("databaze.inc" "CAN1")
```

Tato přiložená databáze proměnných obsahuje definice jednotlivých signálů a vedený parametr určuje s jakou CAN sběrnicí má uvedená databáze komunikovat.

```
VAR
	%1%_Speed = %1% DATA 100 [0]
	%1%_Temp = %1% DATA 100 [1]
```

Taková deklarace proměnných je identická deklaraci, kterou je možné využít přímo v hlavním kódu konfigurace.

```
VAR
	CAN1_Speed = CAN1 DATA 100 [0]
	CAN1_Temp = CAN1 DATA 100 [1]
```

4.6.2 Cesta souboru

Cestu k souboru je možné definovat více způsoby.

Pokud je include soubor vložen přímo ve složce projektu konfiguračního souboru, není potřebné definovat celou cestu.

```
#include ("soubor.inc")
```

Pokud include soubor není umístěn ve složce projektu, je potřebné definovat celou cestu k *.*inc* souboru.

```
#include ("C:\Uzivatel\ … \soubor.inc")
```

5 Datalogging pomocí dataloggeru GL5xxx

5.1 Uspořádání paměti

V případě série dataloggerů GL5xxx je paměť určená pro ukládání nasbíraných dat rozdělena na dvě nezávislé části. S každou částí je možné nakládat paralelně a nejsou na sobě nijak závislé.

Každá paměť je rozdělena do dvou částí a velikost těchto pamětí (bloků ring bufferu) může být definována samostatně pomocí systémových konstant Logger1Size a Logger2Size.

```
SYSTEM
      Logger1Size = 5000 {velikost první paměti 5 MB}
      Logger2Size = 2000 {velikost druhé paměti 2 MB}
```

Velikost ring bufferu je možné definovat mezi 50 KB a 400 000 KB.

Dalšími systémovými konstantami, které je možné deklarovat jsou Logger1Files a Logger2Files. Definovaná velikost říká, kolik triggrovaných nahrávání může být maximálně uloženo do dočasné paměti. V případě, že dojde k překročení nastaveného limitu, jsou nejstarší záznamy vymazány pokaždé, když dojde k dalšímu triggeru.

```
SYSTEM
      Logger1Files = 100
      Logger2Files = 1000
```

5.2 STOP a START kritéria

Obě paměti 1 a 2 je možné kontrolovat pomocí START/STOP. Z důvodu, že jsou obě paměti nezávislé, může být nahrávání první paměti zastaveno nebo spuštěno bez toho, aby byla ovlivněna druhá paměť.

Princip fungování START/STOP je takový, že pokud zařízení nahrává, kontroluje se neustále kritérium pro STOP nahrávání a v případě, že dojde k zastavení nahrávání, začne se kontrolovat kritérium pro START nahrávání.

```
STOP 1/2 (podmínka) nebo (podmínka)
START 1/2 (podmínka) nebo (podmínka)
```

V případě, že není pro paměť definováno START kritérium, záznam nebude znovu spuštěn. Zaznamenána bude pouze událost, která spustila trigger jako první, nicméně další záznam nebude proveden do vymazání paměti.

Pokud je START kritérium definováno a je splněno, dojde k opětovnému záznamu. Tento záznam bude uložen do dalšího bloku.

V případě, že není definováno ani jedno kritérium, daná paměť nemůže být nikdy spuštěna.

5.3 Víceúrovňové Trigger podmínky

Ke kontrole nahrávání se kromě START/STOP může použít funkce počínající klíčovým slovem Trigger_Configuration.

```
TRIGGER_CONFIGURATION
      Trigger_Nazev ("Jmeno_Triggeru", 1, 100, On, 50) BEGIN
            Level1 = (podmínky)
            Level2 SET = (podmínky) RESET = (podmínky)
            ...
            Level32 = (podmínky)
END
```

Nejprve je nutné definovat název objektu Trigger_Configuraton.

Jmeno_Triggeru je volitelné a bude se zobrazovat ve správci souborů. Pokud definováno není, bude zobrazen zvolený název objektu.

V další části se definují čtyři parametry:

- První parametr je pro určení, s jakou pamětí budeme pracovat.
- Jak často bude trigger znovu vypočítáván, je určen druhým parametrem v milisekundách.
- On/Off pouze říká, zdali je daný trigger aktivní. V případě stavu Off není třeba trigger vymazat nebo zakomentovat.
- Poslední parametr určuje minimální prodlevu mezi triggery.

Level1/…/32 jsou pevně dané názvy pro individuální trigger podmínky. Tyto podmínky jsou definovány stejně jako objekty typu FLAG a je s nimi stejným způsobem pracováno. V případě, že dojde k resetu již nastaveného stupně během průchodu, všechny ostatní vyšší stupně jsou resetovány. V případě definování pouze několika stupňů triggeru, například pouze podmínek pro Level 1 a Level 10, chybějící stupně jsou automaticky přeskočeny.

5.4 Funkce Drive Recorder

Tato funkce umožňuje zaznamenávat data v čase až do úplného zaplnění paměti. Proto lze funkci využít pro dlouhodobé nahrávání.

Nahrávat pomocí funkce Drive Recorder je možné až po nastavení systémové konstanty DriveRecording na hodnotu On (zapnuto).

Díky systémové proměnné DriveRecSwitch lze vypínat a zapínat nahrávání pomocí změny její bitové hodnoty.

```
CALC DriveRecSwitch = (On)
```

V případě, že chceme nahrávat data, je nutné si alokovat velikost operační paměti. Pro tento účel slouží další systémové konstanta DriveRecMBs. Rozsah je omezen pouze velikostí operační paměti (v našem případě 64 MB).

6 Hybridní vozidla

Otázka alternativních pohonů se řeší již mnoho let, nicméně v poslední době jde o velmi diskutované téma. Hybridní vozidla jsou takové dopravní prostředky, které kombinují více zdrojů energie. Vedle klasického spalovacího motoru je tu navíc například setrvačník, který má za úkol snížení emisí při rozjezdu vozidla. Další možností je rozšíření spalovacího motoru o elektromotor a akumulátor. Realizace této možnosti patří v současné době k těm více oblíbeným.

I přesto, že účinnost elektromotorů je větší, než při použití klasického spalovacího motoru, spalovací motor v kombinaci s elektromotorem je stále lepší variantou. Je to dáno faktem, že infrastruktura zatím není plně připravena na čistě elektrické pohony. Nicméně tento problém se velmi intenzivně řeší a za posledních pár let došlo k velkému pokroku, na kterém se podílelo mnoho institucí i automobilek, včetně Škoda Auto a.s..

6.1 Historie hybridních vozidel

Historie hybridních vozidel sahá až na samý počátek automobilismu. I přes fakt, že v dnešní době je využití hybridních pohonů stále značně pozadu oproti klasickým spalovacím motorům, první hybridní automobily se objevily již s příchodem automobilů se spalovacím pohonem.

6.1.1 Raný automobilismus

První vůz na hybridní pohon byl vůz Mixte od společnosti Lohner-Porsche a byl představen již roku 1898. Jednalo se o vůz postavený na sériovém uspořádání, měl náhon přední nápravy a kola poháněl elektrokolomotor. Navzdory tomu, že se jednalo o první hybridní automobil, účinnost přenosu výkonu byla až 83%.

Dalším hybridním vozem, který přišel s inovativním řešením, byl druhý model vozu Mixte a byl představen roku 1903. Jako první vůz měl náhon na všechna čtyři kola a na svou dobu dosahoval vysoké rychlosti a to až 110km/h.

Významným konstruktérem, kterého je potřeba zmínit, byl Henri Pieper. Jako první přišel s hybridním pohonem s paralelním uspořádáním a to roku 1905. Vůz byl vybaven silným spalovacím motorem, elektromotorem a mechanickou pákou, kterou bylo možné přepínat jednotlivé jízdní režimy. Při nízkých rychlostech byl využíván pouze elektromotor a k přepnutí na spalovací motor mohlo dojít pouze manuálně. Za zmínění jistě stojí i to, že vůz si během jízdy rekuperoval energii.

Jedním z nejprodávanějších hybridních vozidel své doby, kterého se vyrobilo přes 600 kusů, byl od společnosti Woods Motor a jeho modelové jméno bylo „Dual Power“, což bylo vzhledem k pohonu velmi výstižné.

6.1.2 Moderní automobilismus

Po dlouhé pauze, trvající od 30. do 60. let 20. století, se objevily zmínky o testovacím automobilu od společnosti General Motors s označením GM 512H. Tohoto vozidla byl vyroben pouze jeden kus a z důvodu nesplnění očekávání byl jeho další vývoj pozastaven.

Další pouze testovací vozidlo bylo roku 1973 od společnosti VW a neslo název Taxi. Tento hybridní automobil byl testován na území USA a až na pár exemplářů, mezi kterými byly i čistě elektrické vozy, se také v dalším vývoji nepokračovalo.

V roce 1989 byl představen vůz od společnosti Audi, opět s výstižným názvem DUO. Opět se jednalo pouze o testovací vůz, nicméně roku 1997 se dostala do výroby jeho třetí verze (DUO III). Bylo vyrobeno pouze 60 kusů a účinnost motoru nedosahovala žádných závratných hodnot.

Velký automobilový boom nastal roku 1997, kdy Japonská společnost Toyota představila první generaci modelu Prius, který je známý především proto, že se jedná o vozidlo s hybridním pohonem. Tento automobil vyvolal značný zájem o hybridní automobily.

Od této doby se začaly objevovat další hybridní vozy napříč celým světem automobilismu. Zásadně se do historie hybridních vozů zapsalo USA v roce 1999 a to vozem Honda Insign. V roce 2000 se objevila další verze Toyoty Prius, o dva roky později se do rodiny hybridních vozů přidala také společnost Honda s vozem Civic Hybrid, která byla ihned následována roku 2004 druhou generací Toyoty Prius II, která dosahovala vysokých prodejů a velmi zvýšila povědomí o hybridních vozidlech.

6.2 Dělení hybridních pohonů podle uspořádání

6.2.1 Sériové uspořádání

Méně používaným druhem uspořádání je sériové. Vozidlo je v tomto případě poháněno především elektromotorem. Elektrický proud potřebný pro pohon trakčních motorů nebo dobíjení baterie vozidla vzniká přes generátor, který je poháněn spalovacím motorem. Jak napovídá název tohoto uspořádání, jednotlivé části hybridního pohonu v sériovém uspořádání jsou uspořádány za sebou. Nevýhodou tohoto uspořádání je samozřejmě mnohonásobná přeměna energie. Z toho důvodu dosahuje mechanická účinnost maximálně 55 %.

Vzhledem k tomu, že využití spalovacího motoru není v mnoha situacích vhodné, je možné zvolit optimální rozsah otáček, při kterých bude spalovací motor využíván k pohonu vozidla s maximální účinností. Nemusíme se tak bát zvýšených emisí při rozjezdu vozidla nebo při volnoběhu.

V případě, že dojde k poklesu energie v baterii a tím pádem nebude schopná plně napájet celé vozidlo, dojde k automatickému nastartování spalovacího motoru.

6.2.2 Paralelní uspořádání

Kromě spalovacího motoru a elektromotoru je v případě paralelního uspořádání celý systém obohacen o mechanickou převodovku, která propojuje kola i oba motory. Jednou z možností je také umístění elektromotoru/generátoru mezi převodovku a spalovací motor. V tomto případě pak elektromotor plní funkci alternátoru a startéru.

V případě, že je v provozu pouze jeden motor, druhý motor stále běží ve volnoběhu a nedodává žádný výkon. Součtem obou okamžitých momentů motoru spojených hřídelí dostaneme výsledný moment motoru.

Jak už bylo řečeno v úvodu kapitoly, spalovací motor v obvyklém režimu dodává většinu výkonu. Elektrický motor slouží v tomto případě převážně k akceleraci.

V případě tohoto typu uspořádání jsou dále například různé výhřevy, klimatizace, posilovač řízení a další elektrické systémy poháněny energií získanou z elektromotoru. Tím pádem dojde ke zvýšení účinnosti spalovacího motoru.

Baterie hybridního vozidla má větší napětí než baterie v klasických automobilech a slouží k akumulaci elektrické energie vzniklé během jízdy vozidla. Výhodou je tedy možnost rekuperace.

6.2.3 Kombinované uspořádání

Kombinované uspořádání je kombinací paralelního a sériového typu uspořádání. Toto uspořádání bylo aplikováno z důvodu částečné neefektivity jednotlivých, již zmíněných, uspořádání.

Využitím kombinovaného uspořádání dochází k větší účinnosti celé sestavy motoru. Díky tomuto uspořádání dokáže automobil využívat při akceleraci, nízkých rychlostech nebo při střídavém zatížení sériové uspořádání a tím pádem využít vysokou účinnost spalovacího

motoru. Naopak v případě konstantní jízdy a vysokých rychlostech dokáže využít paralelního uspořádání a využít tak mechanického přenosu výkonu, který mu toto uspořádání nabízí.

6.3 Dělení hybridních pohonů podle stupně hybridizace

6.3.1 Micro hybrid

V případě prvního stupně hybridizace, který nese název „Micro hybrid“ se jedná v jednoduchosti pouze o klasický Start/Stop systém. V momentě úplného zastavení vozidla dojde k vypnutí klasického spalovacího motoru a při puštění brzdového pedálu opět k jeho opětovnému nastartování. Díky tomu se šetří jak spotřeba paliva, tak dochází i k redukci emisí, které vozidlo vytváří. Největší efekt nastává při využití vozidla, vybaveného tímto systémem, například ve městě, kde dochází k častému zastavení automobilu. V současné době je systémem „Start/Stop“ vybaveno velké množství vozidel a všechny proto můžeme označit jako vozidla s prvním stupněm hybridizace.

6.3.2 Mild hybrid

Vozidla označovaná jako Mild hybrid jsou poháněna konvenčním spalovacím motorem s elektromotorem s výkonem kolem 15 kW. V případě varianty mild hybrid je zachován spalovací motor a je odebrán startér motoru a alternátor, z tohoto důvodu se jedná pro automobilové výrobce o vcelku nenákladnou verzi hybridního pohonu.

Tento stupeň hybridizace napomáhá vozidlu především při startu a rozjezdu vozidla. V případě zastavení vozidla na krátkou chvíli umožní vypnutí motoru a obstarává následně start (Start/Stop systém) a napomáhá malým výkonem při rozjezdu.

Oproti vozidlu s klasickým spalovacím motorem dochází až k 15% úspoře paliva. Příkladem může být model Silverado od společnosti General Motors, kde docházelo až k 12% úspoře paliva.

6.3.3 Full hybrid

Typ hybridizace nazývaný jako Full hybrid označuje vozidla na hybridní pohon, která jsou schopna jet na spalovací pohon, elektrický pohon nebo na kombinaci těchto dvou pohonů.

To, na jaký pohon vozidlo pojede, si volí sám řidič nebo může nechat volbu typu pohonu na řídící jednotce motoru. Ta sama vyhodnotí, jaký pohon bude v danou chvíli nejvhodnější.

Při jízdě na čistě elektrický pohon jsou nulové emise, z toho důvodu je vhodné například ve městech zvolit elektrický pohon (smogová omezení). Oproti tomu například na dálnici

nedosahuje elektrický pohon větší účinnosti, než klasický spalovací, z toho důvodu je vhodnější zvolit jízdu na spalovací motor.

6.3.4 Plug-in hybrid

Vozidla na takzvaný Plug-in hybrid neboli PHEV, jsou označovány jako mezistupeň mezi vozidly na čistě elektrický pohon a klasickými hybridními automobily.

Vozidla s tímto typem hybridizace je možné nabíjet z jakéhokoliv elektrického zdroje, který je schopný baterii dobít. Jelikož vysokonapěťové baterie nemají stále tak velkou kapacitu jako baterie elektromobilů. Dobíjení tak trvá mezi 2-3 hodinami, v závislosti na velikosti baterie.

7 Praktická část

V této kapitole se budu věnovat praktické části bakalářské práce. Mým primárním úkolem je naprogramovat konfiguraci pro logování dat z řídící jednotky hybridního vozu. Konfigurace bude pro datalogger GL5350 od společnosti G.i.N.

Dalším úkolem je vytvořit požadavky pro externí firmu, která na základě měření, získaného díky mnou vytvořené konfiguraci, vytvoří vyhodnocení, které představím v závěru praktické části.

7.1 Konfigurace pro hybridní vozidla

Konfigurace pro logování dat z řídící jednotky vozidla s hybridním pohonem se nijak zvlášť neliší od konfigurace pro logování dat z vozidla na klasický spalovací pohon. Nicméně v konfiguraci přibudou další prvky zaměřené na signály z hybridní CAN sběrnice.

Konfigurace by měla splňovat požadavky, které budou popsány v další části této kapitoly.

7.1.1 Požadované vlastnosti konfigurace

Hlavním požadavkem je, aby byly snímány veškeré CAN sběrnice, po kterých jednotlivé části vozidla komunikují s řídící jednotkou hybridního vozidla. Z toho důvodu musí být implementovány databáze, kde jsou popsány informace o jednotlivých zprávách, ve kterých se signály nacházejí.

Nicméně ne každý signál je pro budoucí analýzu vždy důležitý a snímání všech CAN sběrnic po celou dobu by mělo za následek zbytečné zaplnění paměti dataloggeru. Z toho důvodu bylo potřeba si vytipovat jednotlivé signály, které pro nás mají nejvyšší prioritu. Tyto signály je poté potřeba logovat stále. Všechny ostatní signály musí být logovány pouze v případě, že nastane nějaká chyba nebo pokud dojde k manuálnímu vyvolání triggeru (zmáčknutí tlačítka).

Dalším požadavkem je zobrazování různých informací přímo na speciálním displeji LOGview. Na tomto displeji by se měly ukazovat především informace o nefiltrované rychlosti vozidla, různé teploty, které mohou ovlivňovat chod motoru, sklon vozovky, zrychlení vozidla a také například informace o posledním stisku tlačítka. V neposlední řadě je zapotřebí zobrazit informace z různých přídavných zařízení, jako jsou například data z GPS antény nebo z kamery.

Dále by měla být možnost zaznamenávat obraz pomocí kamery a dále také zvukový záznam přes speciální tlačítko „VoCAN Button“, které je obohacené o mikrofon. Všechna tato zařízení by v závěru měly napomoct při analýze chyb způsobených během jízdní zkoušky.

Požadavkem jsou také klasifikace, které budou sloužit pro další analýzu. Musí být vytvořena například klasifikace pro kontrolu otevírání dveří v průběhu celého měření.

7.1.2 Popis prvků konfigurace

V této podkapitole budou popsány jednotlivé části konfigurace, která by měla splňovat předem určené požadavky.

Include soubory

Databáze

Před tím, než je možné s jednotlivými signály pracovat, je potřeba dataloggeru sdělit, v jaké zprávě se nachází jednotlivé signály, jakou mají bitovou adresu a především po které CAN sběrnici se přenáší. K tomu je potřeba využít databázi, kde jsou všechny tyto informace zapsány.

```
#include ("C:\Users\UWR0O0N\Desktop\Datalogger\All-in_conf\MQB_ACAN.inc" "CAN5")
#include ("C:\Users\UWR0O0N\Desktop\Datalogger\All-in_conf\MQB_ICAN.inc" "CAN3")
```

Zde je ukázka INCLUDE příkazu, kterým lze přiřadit databázi k projektu konfigurace. V první části se určí celá cesta k databázi a druhá část definuje, ke které CAN sběrnici databáze náleží. Je nutné podotknout, že každý výrobce dataloggerů má jinak zapojený svazek pro připojení k řídící jednotce motoru, z toho důvodu je nutné si dávat pozor na správné přiřazení CAN sběrnice.

```
ANT_ESP_21_BZ                  = %1% DATA ANT_ESP_21 [X(X:X)]
ANT_BR_Eingriffsmoment         = %1% DATA ANT_ESP_21 [X(X:0) 1(X:X)]
                                 FACTOR = 1  OFFSET = -XXX
                                 VALUETABLE = [ XXXX = "Init" XXXX = "Fehler" ]
```

Přiložený soubor obsahuje databázi všech signálů, které je možné sledovat na určité CAN sběrnici. Každá CAN proměnná je definována názvem zprávy a bitovou adresou. V případě zvolení špatných hodnot může dojít k získávání nesmyslných informací nebo nebudou logována žádná data.

Další include soubory

Dále je možnost si vytvořit neomezené množství include souborů, které mohou obsahovat část konfigurace.

```
#include ("C:\Users\UWR0O0N\Desktop\Datalogger\All-in_conf\LogView.inc")
#include ("C:\Users\UWR0O0N\Desktop\Datalogger\All-in_conf\SkodaLOGO.inc")
#include ("C:\Users\UWR0O0N\Desktop\Datalogger\All-in_conf\DriveRecCode.inc")
```

Tato možnost velmi ulehčí orientaci v kódu konfigurace a je poté přehlednější a snadněji se provádí jeho změny.

Systémové proměnné

V další části konfigurace jsou definovány systémové proměnné. Systémové proměnné jsou uvedeny pod klíčovým slovem SYSTEM a bez jejich správného definování by nebylo možné správně využít všech požadovaných vlastností dataloggeru a jeho příslušenství.

V první řadě je potřeba otevřít komunikaci s požadovanými CAN sběrnicemi. Toho docílíme systémovou proměnnou CANxOutput, kterou nastavíme na hodnotu „On“. Dále je nutné nastavit přenosovou rychlost pomocí proměnné CANxTiming. Rychlost CAN sběrnice se liší podle typu vozidla. Poslední proměnnou, která se týká CAN sběrnice je CANxKeepAwake, která určuje, zdali má být datalogger stále aktivní i po vypnutí KL15 (druhá pozice klíčku v zapalování). V případě, že je nastavena na hodnotu „On“, datalogger bude měřit, dokud budou z příslušné CAN sběrnice chodit zprávy. Uvedené „x“ musí být nahrazeno příslušným číslem CAN sběrnice.

Dále se v této části nachází proměnné, které určují maximální počet packetů dat a jejich velikost. Jak bylo zmíněno v teoretické části, datalogger má dvě paměti, se kterými může pracovat. V našem případě si deklarujeme Logger1/2Size na velikost 100 MB a Logger1/2Size na 100 packetů.

Vzhledem k tomu, že podmínkou je logování dat pomocí funkce DriveRec, musíme si definovat i systémové proměnné, které se týkají této problematiky. Pro možnost logování tímto způsobem si tedy nastavíme DriveRecording na hodnotu „On“ a musíme také zvolit maximální velikost nahrávaných dát. Pro to nám slouží proměnná DriveRecMBs, kterou nastavíme na 5000. Tato velikost bude dostačující vzhledem k počtu signálů, které hodláme snímat.

Proměnné v tomto odstavci se týkají především příslušenství, které bude k dataloggeru během měření připojené. Vzhledem k faktu, že chceme pořizovat video záznam, je nutné nastavit IP

adresu externí kamery. IP adresu nastavíme pomocí proměnné HostCameraAddress. Dále budeme využívat i externí display LOGview. Pro jeho používání musíme nastavit proměnnou UseDisplay na hodnotu „On“ a v případě potřeby je možné nastavit i zobrazování jednotlivých stránek pomocí DisplayAutoPages. Práce s touto proměnnou je jednoduchá, pouze si pomocí nul a jedniček navolíme stránky, které se mají zobrazit nebo skrýt a poté bitovou hodnotu převedeme do hexadecimálního čísla. Výsledkem je poté například hodnota 0x000B, která říká, že první dvě stránky budou viditelné, třetí bude skrytá a čtvrtá poté opět viditelná.

Jako poslední je třeba určit, po jaké době od spuštění dataloggeru se mají začít logovat data a jak dlouho má být datalogger aktivní po vypnutí KL15. Tuto problematiku nám vyřeší systémová proměnná PAUSE, kterou nastavíme na hodnotu 0, pro okamžité snímání signálů. Dále definujeme SleepSeconds, kterou musíme nastavit na nejméně 30 sekund, aby datalogger nalogoval vypnutí všech systémů.

Proměnné, konstanty a značky

Pod klíčovými slovy VAR, CONST a FLAG se nacházejí tentokrát již námi zvolené proměnné, konstanty a značky. Z důvodu, že budou značně využívány v dalších částech této praktické práce, zmíním je jmenovitě až v průběhu této práce.

Trigger

První a pravděpodobně nejdůležitější funkcí je trigger, který v první řadě po stisku tlačítka nebo v případě chyby spustí logování všech CAN sběrnic.

```
FLAG
      TriggerFlag                SET    = (Key1)
                                 RESET = (NOT Key1) SOUND (HI)
```

Tato část funkce vytvoří značku na základě stisku tlačítka řidičem během jízdy. Po stisku tlačítka se změní hodnota FLAG „TriggerFlag“ na 1 a po uvolnění zpět na 0. Díky tomu se spustí EVENT „ON SET“ a pomocí funkce TRANSMIT nahraje značky do paměti DriveRec. V měření je poté jasně vidět kde v jaký moment bylo tlačítko zmáčknuto.

Obdobným způsobem se udělá i značka triggeru na základě chyby, kterou chceme sledovat.

Aby byly sledovány všechny otevřené CAN sběrnice po stisku tlačítka, je třeba definovat funkci „TRIGGER_CONFIGURATION“, kterou si pojmenujeme jako „Trigger“, budeme ji logovat na první paměť, opakovaně je možné ho provést po 30 sekundách a bude se logovat zpětně 30 sekund záznamu.

```
TRIGGER_CONFIGURATION
      TriggerConfig_1_02 ("Trigger" 1, 1, On, 30000) BEGIN
      Level1 = (TriggerFlag)
END
```

Logování vybraných signálů pomocí funkce DriveRec

Pro větší přehlednost zdrojového kódu konfigurace je vše, co se týká logování signálů pomocí funkce DriveRec, v samostatném INCLUDE souboru.

I přestože by bylo ideální, kdyby byla možnost snímat všechny signály po celou dobu měření, z důvodu velkého objemu dat to není reálné. Proto v našem případě využijeme funkci DriveRec, která nám umožňuje logovat pouze vybrané signály a to po celou dobu měření. Zápis této funkce je velmi jednoduchý, stačí pouze určit, že hodláme odesílat nějaká data pomocí TRANSMIT a za klíčovým slovem DriveRec poté určíme signály, které se mají logovat.

Bylo nutné si tedy nejdřív vytipovat jednotlivé signály, které by nás při následné analýze mohly zajímat. Jedná se například o signály obsahující informaci o rychlosti vozidla, otáčkách motoru, napětí baterie a různé teploty vody, oleje atp. Dále jsou potřebné také signály týkající se elektromotoru. Po celou dobu měření je vhodné snímat například spotřebu energie elektromotoru, rekuperaci, procentuální nabití baterie nebo opět teplotu, ale v tomto případě elektromotoru.

Aby bylo snímání signálů aktivní, je třeba nejprve sdělit dataloggeru, že snímání pomocí funkce DriveRec požadujeme. Toho docílíme pomocí přepnutí systémové proměnné DriveRecSwitch na hodnotu „On“. To provedeme ihned při zapnutí dataloggeru díky systémové proměnné Startup.

```
EVENT
      ON SYSTEM (Startup) BEGIN
            CALC DriveRecSwitch = (On)
      END
```

Dále bylo nutné určit, jak často se mají signály snímat a zdali se mají logovat stále nebo pouze pokud dojde ke změně hodnoty signálu. Rychlost, otáčky motoru nebo například různé teploty je potřeba snímat co nejčastěji, proto je zvolený cyklus nastaven na 100 ms a stálé logování do paměti dataloggeru.

```
EVENT
       ON CYCLE (100) BEGIN {100 ms}

{-----------------------------------ANTRIEB---------------------------------------}
              TRANSMIT  DriveRec  18  [ANT_MO_Drehzahl_01  ANT_MO_Fahrpedalrohwert_01
              ANT_ZAS_Kl_15      ANT_BCM1_Aussen_Temp_ungef      ANT_ESP_Laengsbeschl
              ANT_ESP_Querbeschleunigung] LOG ONLY

              TRANSMIT     DriveRec     2     [ANT_MO_Ladedruck     ANT_ESP_Bremsdruck
              ANT_MO_Kuppl_schalter ANT_MO_Kuehlmittel_Temp] LOG ONLY
```

Nicméně například souřadnice určující GPS polohu vozidla stačí sledovat každou 1s a díky modifikaci ONCHANGE je možné signál uložit do paměti dataloggeru pouze v případě, že došlo ke změně hodnoty.

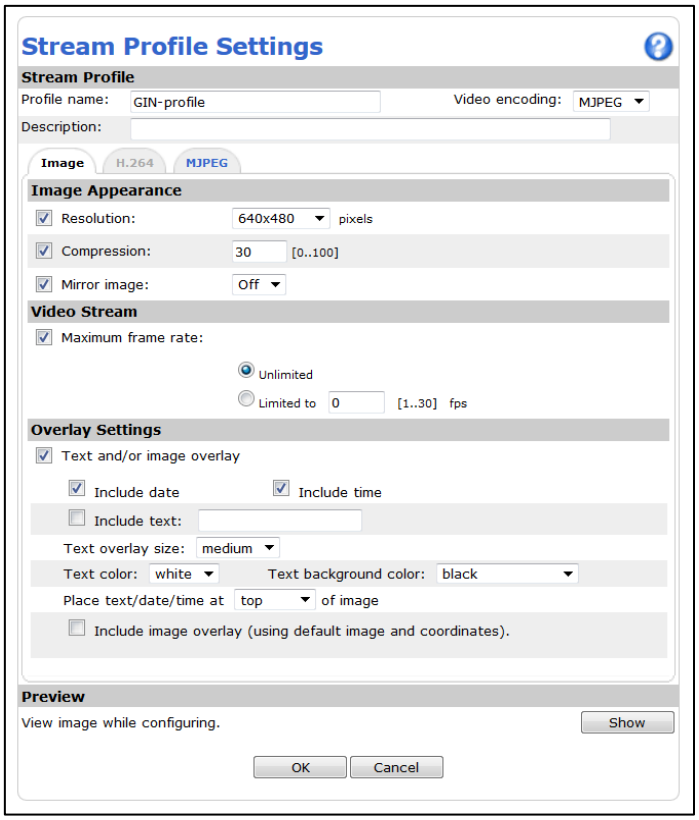
```
EVENT
       ON CYCLE (1000) BEGIN {1000 ms}
{-----------------------------------KOMFORT----------------------------------------}
              TRANSMIT DriveRec 8 MOTOROLA
              [KOM_NP_LatDegree.HIGH KOM_NP_LatDegree.LOW] LOG ONLY ONCHANGE
              TRANSMIT DriveRec 9 MOTOROLA
              [KOM_NP_LongDegree.HIGH KOM_NP_LongDegree.LOW] LOG ONLY ONCHANGE
```

Nahrávání video záznamu

Pro možnost zaznamenat videozáznam během jízdy se využívá speciální síťová kamera AXIS P1214-E s rozlišením 720p.

Nastavení kamery

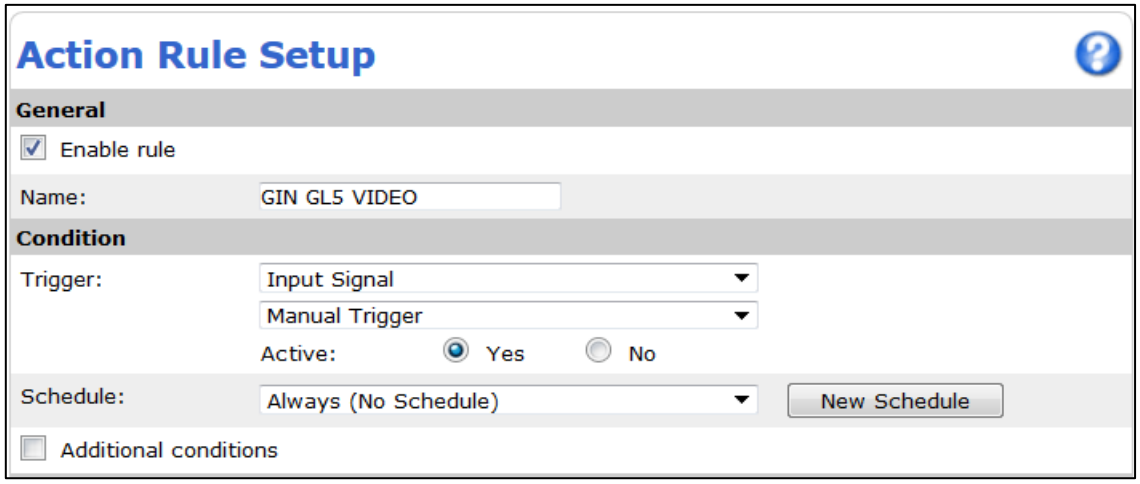
Před tím než budeme moci nastavit podmínky záznamu, je nutné vytvořit streamovací profil, kde nastavíme rozlišení a kódování video záznamu. Toto nastavení provedeme ve webové aplikaci IP kamery, ke které je možné se připojit pomocí prohlížeče po zadání IP adresy zařízení.



Obrázek 2 - Nastavení kamery - Streamovací profil

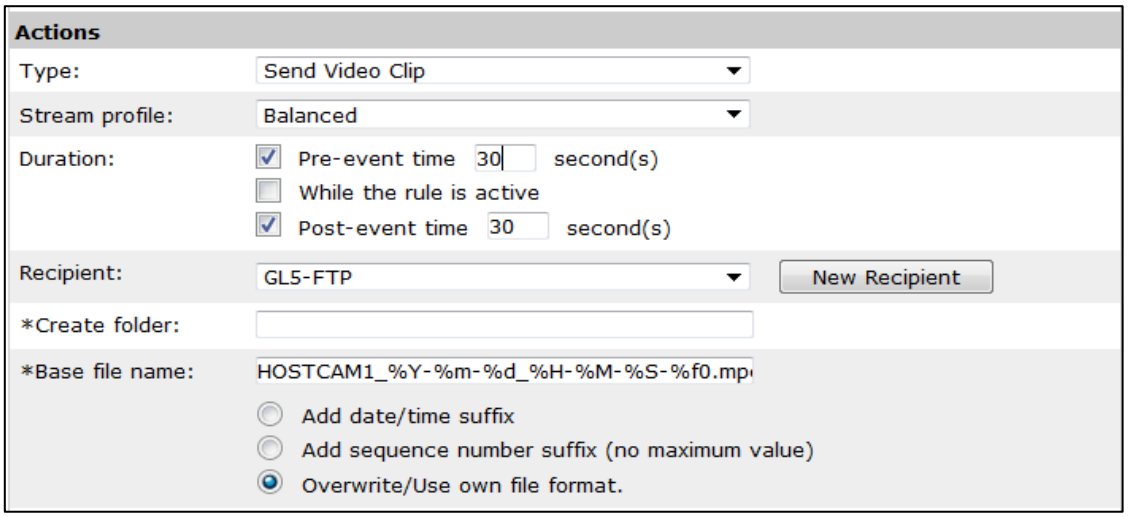
Optimálním rozlišením je 640x480 a kódování MPEG, při použití většího rozlišení je objem dat příliš vysoký. Pro představu, jedna minuta záznamu při rozlišení 480p zabírá 15MB, kdežto při použití rozlišení 720p dojde téměř k trojnásobnému nárůstu na 40MB.

Dále je zapotřebí nastavit pravidla nahrávání video záznamu. V prvé řadě je potřeba si zvolit název pravidla, pomocí kterého budeme moci jednotlivá nastavení rozlišovat. V další části nastavení se volí typ triggeru, jinak řečeno nastavení toho, v jakém případě má začít kamera nahrávat. V našem případě volíme jako trigger manuálně vyvolaný vstupní signál, neboli stisknutí tlačítka dataloggeru.



Obrázek 3- Nastavení kamery - Trigger

V poslední části už pouze nastavíme typ akce „Send Video“, zvolíme námi vytvořený streamovací profil a nastavíme frekvenci snímků na 30FPS což je v našem případě dostačující. V poslední části zvolíme, kolik vteřin záznamu požadujeme při obdržení triggeru. Pro další analýzu je vhodné, aby byl uložen záznam dlouhý minimálně jednu minutu. Z toho důvodu nastavíme „Post-trigger time“ i „Pre-tigger time“ na 30 sekund. Také je možné nastavit název videa.



Obrázek 4 - Nastavení kamery - Typ akce, pre/post trigger, název souboru

Konfigurace kamery

Pro správnou komunikaci dataloggeru s kamerou se musí nastavit IP adresa kamery, jak bylo uvedeno dříve.

Pro uložení video záznamu využijeme opět FLAG značku „TriggerFlag“. V případě, že dojde ke zmáčknutí tlačítka, provede se EVENT ON SET. V ten moment se nastaví systémová proměnná HostCameraTrigger na hodnotu „On“ a tímto způsobem dojde k uložení posledních 30s záznamu a příštích 30s bude záznam pokračovat, tak jak bylo nastaveno v kameře.

```
EVENT
       ON SET (TriggerFlag) BEGIN
       CALC
              HostCameraTrigger = (On)
END
```

Následně po uvolnění tlačítka se díky EVENT ON RESET se systémová proměnná nastaví na „Off“, aby nedocházelo k opakovanému záznamu videa.

```
EVENT
       ON RESET (TriggerFlag) BEGIN
       CALC
              HostCameraTrigger = (Off)
END
```

Zobrazení stavu kamery na displej LOGview

Opět si vypíšeme informace o stavu kamery na display LOGview. Díky systémové proměnné HostCameraStatus si můžeme zobrazit jednotlivé stavy.

Informace o kameře si vypíšeme na šestou stránku displeje na první řádek. HostCameraStatus.0 nám říká, že není kamera připojena a nemůžeme ji používat. Dále tu je HostCameraStatus.1 a pokud je splněn tento stav, znamená to, že kamera je připojena a je připravena nahrávat. A pokud dojde k předchozímu stavu a zároveň platí, že proměnná LEDonTriggerMemory1 > 0, nahrávání video záznamu je aktivní.

```
DISPLAY
       printp (6, 4, 4, "Camera:   Error ") WHEN (HostCameraStatus.0)
       printp (6, 4, 4, "Camera:   Online") WHEN (HostCameraStatus.1)
       printp (6, 4, 4, "Camera:   No Lic") WHEN (HostCameraStatus = 0xF1)
       printp  (6,  4,  4,  "Camera:      REC      ")  WHEN  (HostCameraStatus.1  AND
       LEDonTriggerMemory1 > 0)
```

Nahrávání zvukového záznamu

Stejně jako video záznam, je možné nahrávat i zvuky přímo po čas jízdy při stlačení samostatného tlačítka. Nicméně je opět potřeba speciální zařízení, které se nazývá „VoCAN Button“.

Samotné tlačítko se připojuje do AUX konektoru a z toho důvodu je potřebné nastavit systémovou proměnnou AUX_SwitchBox na „On“, aby byl záznam zvuku možný.

Dále si vytvoříme FLAG značku VoCAN, které přiřadíme tlačítko. Je nutné dát si pozor na proměnnou tlačítka, která je v tomto případě Key8. Stisknutí tlačítka doplníme zvukovým signálem, aby byl řidič uvědomen, že tlačítko opravdu zmáčkl.

```
FLAG VoCAN = (Key8) SOUND (HI 50)
```

Aby nebyl zvukový záznam moc dlouhý, ale přesto měl řidič dostatek času na sdělení problému, nastaví se po EVENT ON SET systémová proměnná VoiceRecording na hodnotu 10000 což odpovídá 10s.

```
EVENT
	ON SET (VoCAN) BEGIN
		CALC VoiceRecording = 10000
	END
```

Pro vizuální signalizaci navíc využijeme diod, které jsou na tlačítku. V případě, že není zaznamenáván zvukový záznam, svítí pouze červená dioda. Pokud řidič stiskne tlačítko, dojde k rozblikání zelené diody. Dioda oranžové barvy svítí, pokud je spuštěn pouze trigger dat bez záznamu zvuku.

Přenos dat přes 3G

Nastavení routeru

Pro odeslání dat na server externí firmy je potřebné připojit 3G router od firmy Sierra Wireless s označením AirLink LS300.

V první řadě je nutné do routeru vložit datovou SIM kartu. Poté co je SIM karta vložena, musí být router nakonfigurován. Toho docílíme připojením routeru k počítači pomocí LAN kabelu a zadáním síťové adresy do okna prohlížeče se dostaneme do webové aplikace.

Poté již jen stačí nastavit PIN SIM karty a restartovat router.

Konfigurace dataloggeru

Pro odeslání dat přes 3G síť pomocí dataloggeru slouží systémové proměnné TransferRequest + ConnectionRequest3G. Aby bylo možné systémové proměnné použít, je nutné přiřadit další systémové proměnné SystemRequest výše zmíněné proměnné. Poté je odesílání zahájeno.

```
SystemRequest = TransferRequest + ConnectionRequest3G
```

Jelikož je mnohdy odesílání delší, není možné odesílat data ihned po jízdě, aby nedošlo ke zbytečnému vybíjení baterie automobilu. Z toho důvodu je odesílání provedeno až při začátku dalšího měření.

Tím nastává další problém, spuštění routeru trvá zhruba dvě a půl minuty. Vzhledem k tomu je posunut začátek odesílání dat o tři minuty.

```
EVENT
	ON CYCLE (18000) BEGIN
		CALC cnt = cnt +1
	End
	ON SET (cnt=10) BEGIN
		CALC SystemRequest = TransferRequest + ConnectionRequest3G
	END
```

GPS pozice vozidla

Konfigurace dataloggeru

Pro logování pozice vozidla je nezbytné připojit k dataloggeru speciální GPS anténu CANgps a ke konfiguraci přiřadit správnou databázi.

Následně stačí opět pomocí funkce DriveRec uložit souřadnice a informace z GPS antény. Nicméně jediným problémem je fakt, že signál pro zeměpisnou délku a šířku je rozdělen, z důvodu, že se jedná o 16bitové signály. Pro jejich spojení je nutné využít takzvaný formát MOTOROLA.

```
TRANSMIT DRIVEREC 19 MOTOROLA
[GPS_Longitude.HIGH GPS_Longitude.LOW] LOG ONLY
TRANSMIT DRIVEREC 20 MOTOROLA
[GPS_Latitude.HIGH GPS_Latitude.LOW] LOG ONLY
TRANSMIT DRIVEREC 21
[GPS_Year GPS_Month GPS_Day GPS_Hours GPS_Minutes GPS_Seconds] LOG ONLY
TRANSMIT DRIVEREC 22
[GPS2_LongitudeM GPS2_LongitudeS GPS2_LatitudeM GPS2_LatitudeS] LOG ONLY
```

Zobrazení GPS souřadnic na displeji

Opět je možné si zobrazit GPS informace a souřadnice v reálném čase na displeji LOGview.

GPS je aktivní a má dostatečnou přesnost po tom, co je připojena alespoň k pěti satelitům. Do té doby je zobrazena na displeji jako neaktivní.

```
DISPLAY
printp (6, 4, 12, "GPS:")
	printp (6, 4, 20, "Neni aktivni    ") when (GPS2_NoActSat < 5)
	printp (6, 4, 20, "Je aktivi       ") when (GPS2_NoActSat >= 5)
END
```

Po tom, co je připojeno dostatečné množství satelitů, zobrazí se jako aktivní. Na displeji poté můžeme sledovat informace o tom, kolik satelitů je připojeno, jaká je přibližná odchylka a také se zobrazí celé souřadnice, na kterých se vozidlo zrovna nachází.

```
printp (6, 4, 28, "Satelity: %2f   ", GPS2_NoActSat) when (GPS2_NoActSat >= 5)
printp (6, 4, 36, "Odchylka: %3fm  ", GPS2_PDOP) when (GPS2_NoActSat >= 5)
printp (6, 4, 44, "LONG: %2f°%2f'%2f''  ", GPS2_LongitudeD, GPS2_LongitudeM,
GPS2_LongitudeS) when (GPS2_NoActSat >= 5)
printp (6, 4, 52, "LAT:  %2f°%2f'%2f''      ", GPS2_LatitudeD, GPS2_LatitudeM,
GPS2_LatitudeS) when (GPS2_NoActSat >= 5)
```

Klasifikace

Pomocí dataloggeru je možné tvořit klasifikace, které slouží k základnímu vyhodnocení dat z jízdy. Je možné tak nahradit z části vyhodnocení, které pro naše oddělení, kterého jsem součástí, vytváří externí firma.

Otevírání dveří

Z jiného oddělení byl vznesen požadavek na kontrolu otevírání dveří během jízdy. Na vyhodnocení musí být vidět, kolikrát a v jakou dobu byly otevřeny jednotlivé dveře vzhledem k venkovní teplotě.

Vzhledem k požadavkům, použijeme typ klasifikace COUNT, který nám jednotlivá otevření dveří bude počítat po dobu celého měření. Otevření dveří bude započítáváno podle venkovní teploty a podle času během kterých k otevření dveří došlo. Vzhledem k tomu, že se tato klasifikace skládá z více částí, je vhodné si tuto skupinu pojmenovat, aby bylo vyhodnocení přehlednější.

```
CLASSIFY Dvere
       TIME
       DverePL COUNT (KOM_ZV_FT_offen)
       OVER ANT_BCM1_Aussen_Temp_ungef (20 CLASSES OF 2 BASE -20)
              OVER Hour (24 CLASSES OF 1 BASE 0)
       DverePP COUNT (KOM_ZV_BT_offen)
              OVER ANT_BCM1_Aussen_Temp_ungef (20 CLASSES OF 2 BASE -20)
              OVER Hour (24 CLASSES OF 1 BASE 0)
       DvereZL COUNT (KOM_ZV_HFS_offen)
              OVER ANT_BCM1_Aussen_Temp_ungef (20 CLASSES OF 2 BASE -20)
              OVER Hour (24 CLASSES OF 1 BASE 0)
       DvereZP COUNT (KOM_ZV_HBFS_offen)
              OVER ANT_BCM1_Aussen_Temp_ungef (20 CLASSES OF 2 BASE -20)
              OVER Hour (24 CLASSES OF 1 BASE 0)
```

Jak je vidět ve zdrojovém kódu konfigurace a také ve výsledném vyhodnocení, teplota je rozdělena po 2 stupních od hodnoty -20 do teploty +20.

Další klasifikace

Dále například sledujeme počet takzvaných „KickDown“. Jde o moment, kdy je pedál plynu sešlápnut na 100 %. Jedná se o velmi jednoduchou klasifikaci a kromě pojmenování není potřeba ji zařadit do zvláštní samostatné skupiny klasifikací.

```
KickDown COUNT (FAH_MO_Kickdown)
```

Další klasifikaci, která je zařazena do stejné skupiny jako ta předchozí, slouží ke sledování otáček motoru. Jedná se o klasifikaci, díky které můžeme vidět, při jakých otáčkách motoru řidič nejčastěji jel.

```
TIME OVER ANT_MO_Drehzahl_01 (|12| CLASSES OF 500 BASE 0)
```

Vzhledem k tomu, že se jedná o klasifikaci typu TIME, počítá se tedy jednotlivý čas v sekundách strávený v rozmezí, které si nastavím. Otáčky motoru jsou rozděleny od 0 ot./s do 6000 ot./s. Výsledkem je tabulka, kde je možné vidět, kolik sekund řidič jel při jednotlivých otáčkách motoru.

Displej LOGview

Včetně informací, které jsem uvedl v předchozích kapitolách, jsou na displeji LOGview zobrazeny grafické prvky a další informace, které usnadňují sledování dat přímo během jízdní zkoušky.

Informace zobrazované na displeji

Na displeji jsou během jízdy zobrazovány v reálném čase informace o nefiltrovaných teplotách motoru, venkovní teploty nebo například chladicí kapaliny. Dále je možné sledovat aktuální napětí baterie.

```
DISPLAY
      printp (3, 4, 13 "RTs:      %5f °C", ANT_MO_Kuehlmittel_Temp)
      printp (3, 4, 21 "VTs       %5f °C", ANT_KBI_Aussen_Temp_gef)
      printp (3, 4, 29 "RT:       %5f °C", RozdilTeplot)
      printp (3, 4, 37 "VT:       %3f °C", Vzduch)
      printp (3, 4, 45 "Voda:     %3f °C", Voda)
      printp (3, 4, 53 "Baterie: %3f  V", ANT_BEM_UBDM)
```

Na další stránce displeje je vidět maximální dosažená rychlost a také maximální zrychlení vozidla, které bylo během jízdy dosaženo.

```
EVENT
      ON CYCLE (100) BEGIN
            CALC    maxRychlost    =    ANT_ESP_v_Signal    WHEN    (maxRychlost    <
            ANT_ESP_v_Signal)
            CALC   maxZrychleni   =   ANT_ESP_Laengsbeschl   WHEN   (maxZrychleni   <
            ANT_ESP_Laengsbeschl)
      END

ON CYCLE (100) BEGIN
      DISPLAY
            printp (4, 4, 46 "MaxR:       %3f km/h", maxRychlost)
            printp (4, 4, 54 "MaxZ:       %3f m/s^2", maxZrychleni)
      END
```

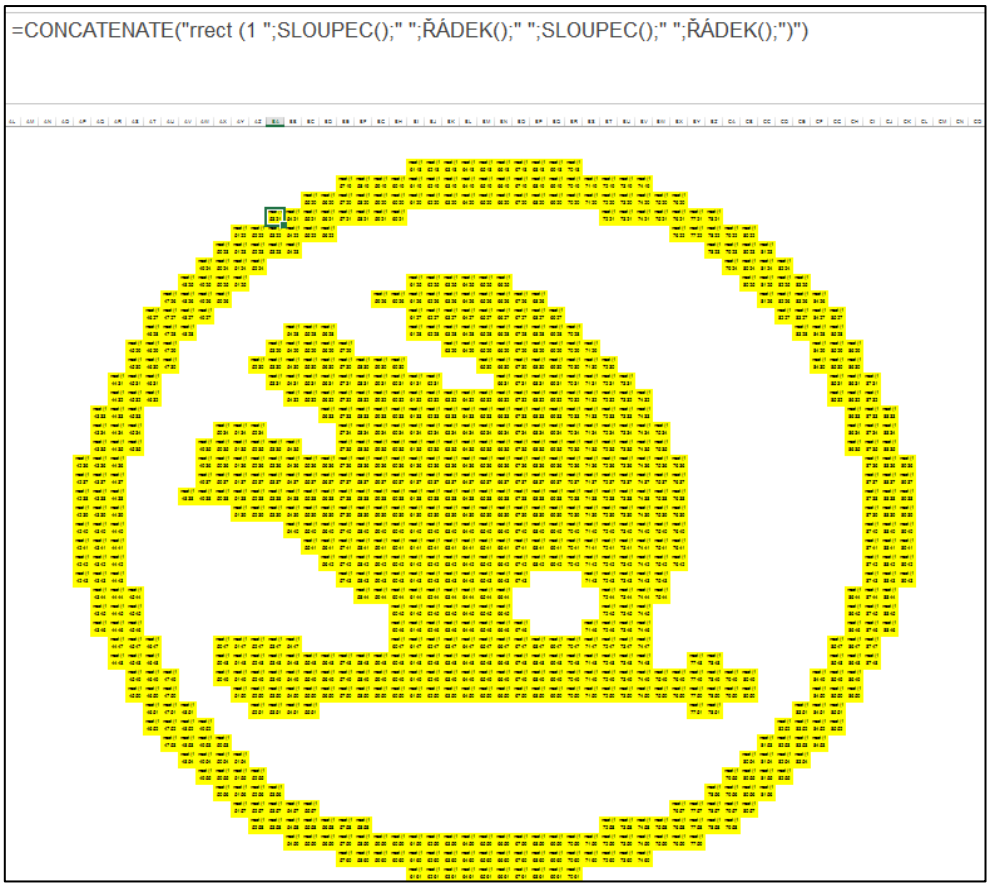
Logo Škoda Auto a.s.

Naše oddělení vzneslo požadavek na firmu G.i.N, zdali by byli schopni vytvořit na úvodní straně displeje logo Škoda Auto a.s.. Bylo nám řečeno, že to není možné. Proto jsem byl požádán, abych se o to pokusil.

Vzhledem k tomu, že je možné ohraničit jednotlivé informace pomocí rrect(strana x1 y1 x2 y2 inv), docílil jsem vykreslení bodu na displeji a to tak, že jsem tento objekt umístil pouze na jeden pixel displeje.

```
rrect (1, 30, 5, 30, 5)
```

Poté co jsem zjistil, že je to možné, stačilo si pomocí Microsoft Excel v buňkách nechat vygenerovat kód všech pixelů displeje a následně vytvořit logo Škoda Auto a.s..



Obrázek 5 - Logo Škoda Auto a.s. - Microsoft Excel

Výsledkem bylo, že po nahrání konfigurace do dataloggeru se na první obrazovce, vždy při spuštění dataloggeru, zobrazí logo Škoda Auto a.s.

Obrázek 6 - LOGview - Logo Škoda Auto a.s.

7.2 Výstupní zpráva

Závěrem této práce je výstupní zpráva, kde budou uvedeny jednotlivé informace o jízdě, které budou popsány v další části této zprávy. Tuto zprávu měla vytvořit externí firma, ale vzhledem k tomu, že na našem oddělení byla dána priorita na jiný typ vozu, bude výsledná zpráva znázorněna mnou.

7.2.1 Požadavky na výstupní zprávu

V této podkapitole budou jednotlivě popsány veškeré požadavky na výstupní zprávy, které byly mnou a mými kolegy definovány. Ke každé straně s požadavky bude přidán náhled z mnou navržených vzorových výstupních zpráv vytvořených v programu Microsoft PowerPoint.

Každá strana bude mít v horní části uvedený název zprávy a vedle toho posledních osm znaků VIN vozidla.

Podoba stránky bude podle předpisů Škoda Auto a.s.. V patičce každé stránky bude číslo stránky, název oddělení a datum.

Postupem času jsme došli k závěru, že bude vhodné vytvořit dvě výstupní zprávy. První se bude týkat především jízdních dat a bude se nazývat „FAHRPROFIL PHEV“. Ve druhé zprávě budou pouze informace týkající se jednotlivých dobíjení vozidla a bude se nazývat „AUFLADEN PHEV“.

7.2.2 Výstupní zpráva - FAHRPROFIL PHEV

Strana 1

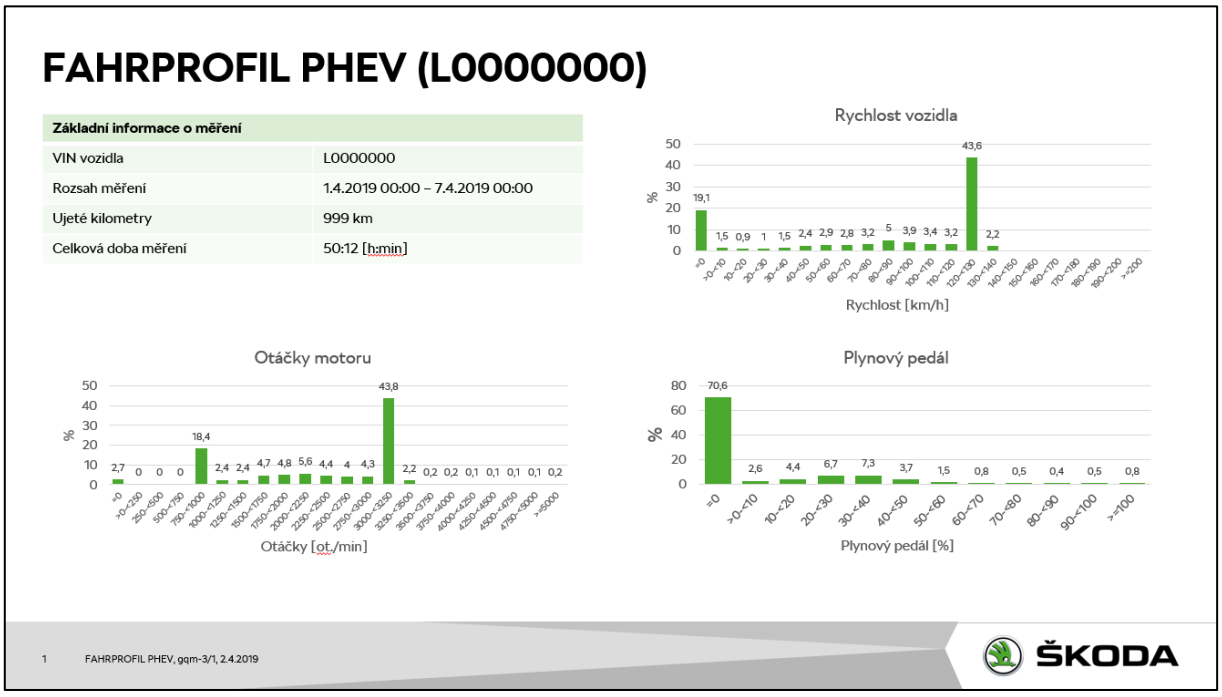
Na první straně se budou nacházet základní informace získané během jízdy.

Tato strana bude rozdělena do čtyř částí. Vlevo nahoře bude tabulka s údaji o délce měření, které analyzujeme. Dále zde bude celkový nájezd kilometrů po dobu vybraného měření a celková doba měření.

Vpravo nahoře bude graf s procentuálním znázorněním toho, jaké rychlosti bylo během měření dosahováno. Sledované rychlosti budou v rozsahu 0 až 200 km/h. Díky tomuto grafu lze ihned rozpoznat, zdali šlo o jízdu po dálnici, v obci nebo mimo ní.

Na grafu vlevo dole bude umístěn graf opět s procentuálním znázorněním otáček motoru. Tentokrát rozsah sledovaných hodnot bude od 0 do 5000 ot./min. Bude tak možné posoudit, zdali styl jízdy řidiče neměl vliv na případnou závadu vozidla.

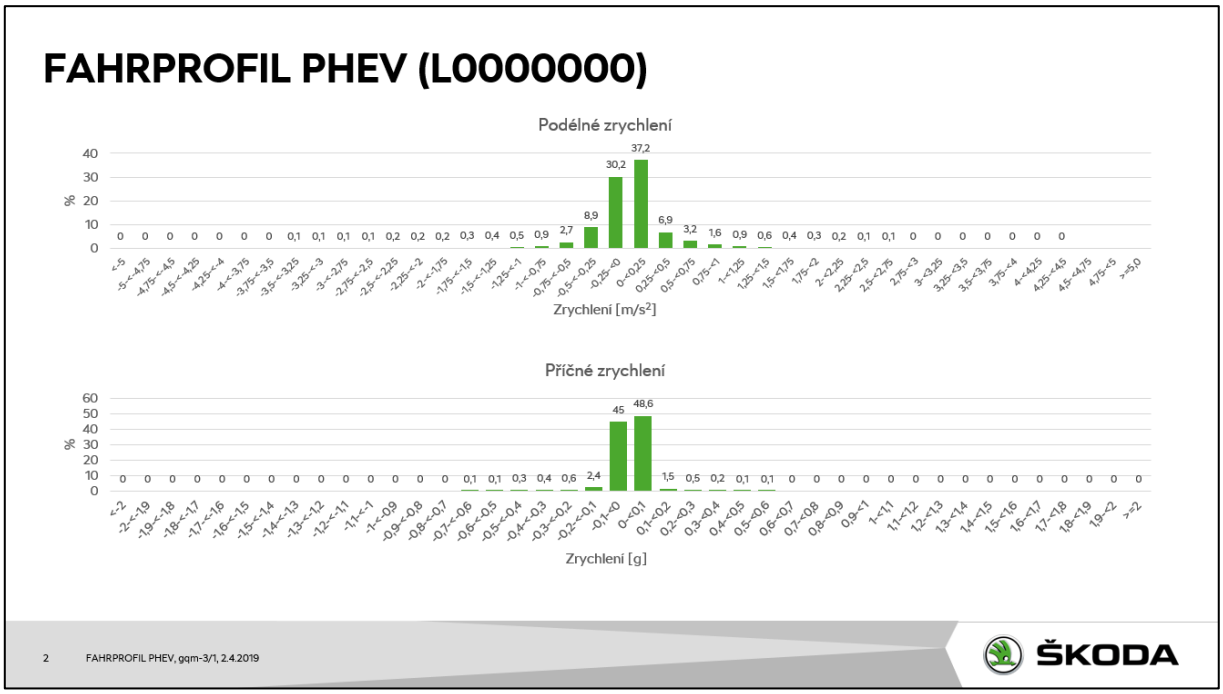
Na posledním grafu, který bude orientován vpravo dole, bude procentuální znázornění sešlápnutí plynového pedálu. Signál sešlápnutí plynového pedálu je v procentech a bude sledován celý rozsah.



Obrázek 7 - FAHRPROFIL PHEV strana 1

Strana 2

Na druhé stránce budou dva grafy s procentuálním znázorněním podélného a příčného zrychlením vozidla. Jelikož tyto signály dosahují kladných i záporných hodnot, bude rozsah grafu s podélným zrychlením -5 až 5 m/s^2 a u grafu s příčným zrychlením bude rozsah -2G až 2G. Tyto rozsahy by měly být více než dostačující.

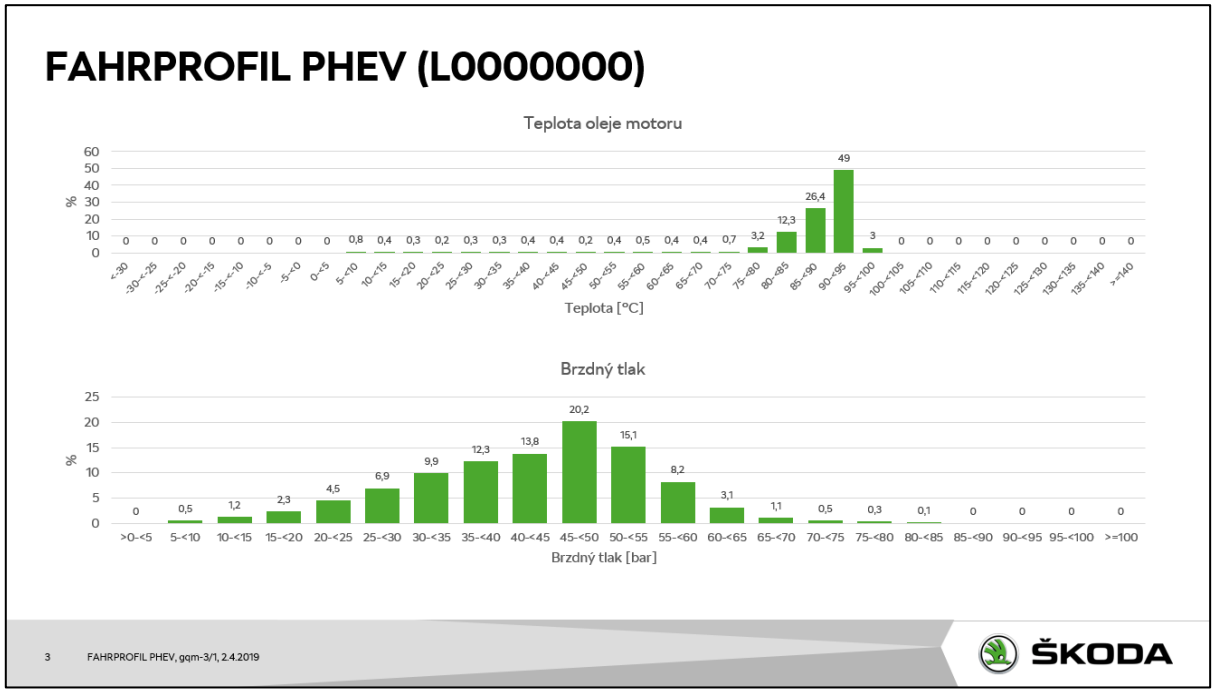


Obrázek 8 - FAHRPROFIL PHEV strana 2

Strana 3

Další dva grafy se budou nacházet na třetí straně a půjde opět o procentuální znázornění. Prvním grafem bude teplota oleje motoru, kde bude rozsah od -30 °C do 140 °C. Díky tomuto grafu bude možné snadněji odhalit závady.

Druhý graf se bude týkat brzdového tlaku. Rozsah je 0 až 100bar.

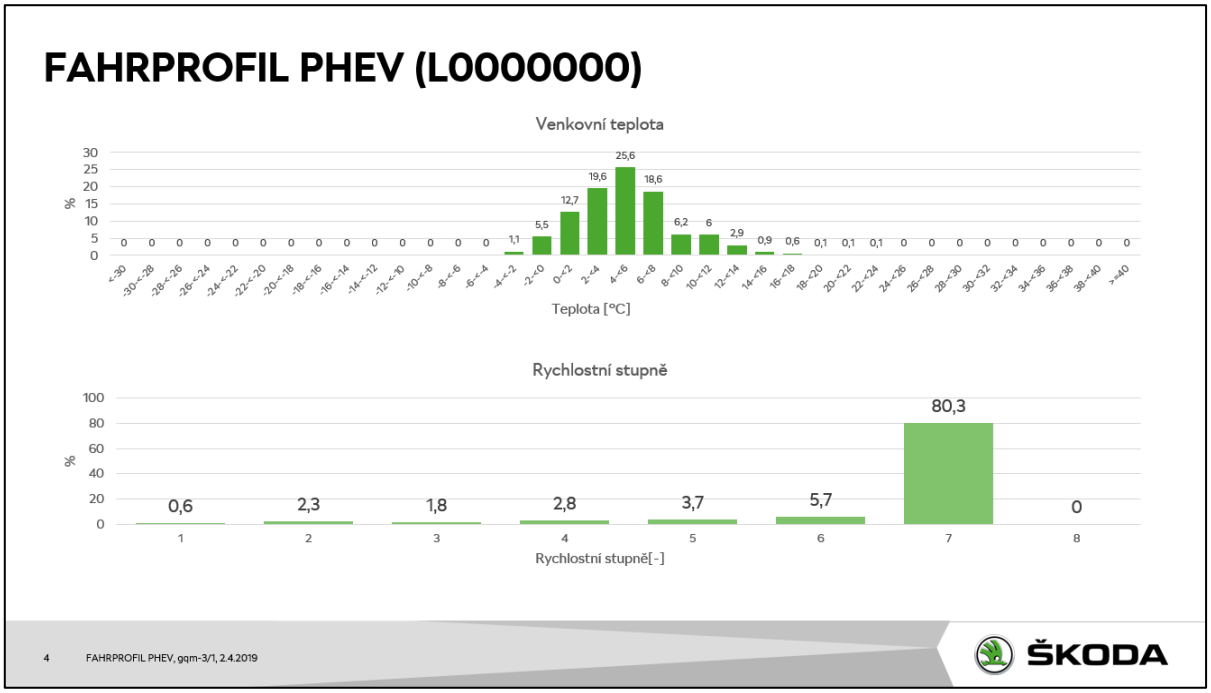


Obrázek 9 - FAHRPROFIL PHEV strana 3

Strana 4

Opět dva grafy se budou nacházet na čtvrté straně výsledné zprávy. Půjde o sloupcový graf ukazující procentuální znázornění venkovní teploty. Rozsah grafu pro venkovní teplotu bude od -30 °C do 40 °C.

Druhý graf bude opět v procentech ukazovat, jaký byl zařazen rychlostní stupeň. Jelikož jde o vozy s automatickou převodovkou, můžou se objevit až 8 stupňové převodovky.



Obrázek 10 - FAHRPROFIL PHEV strana 4

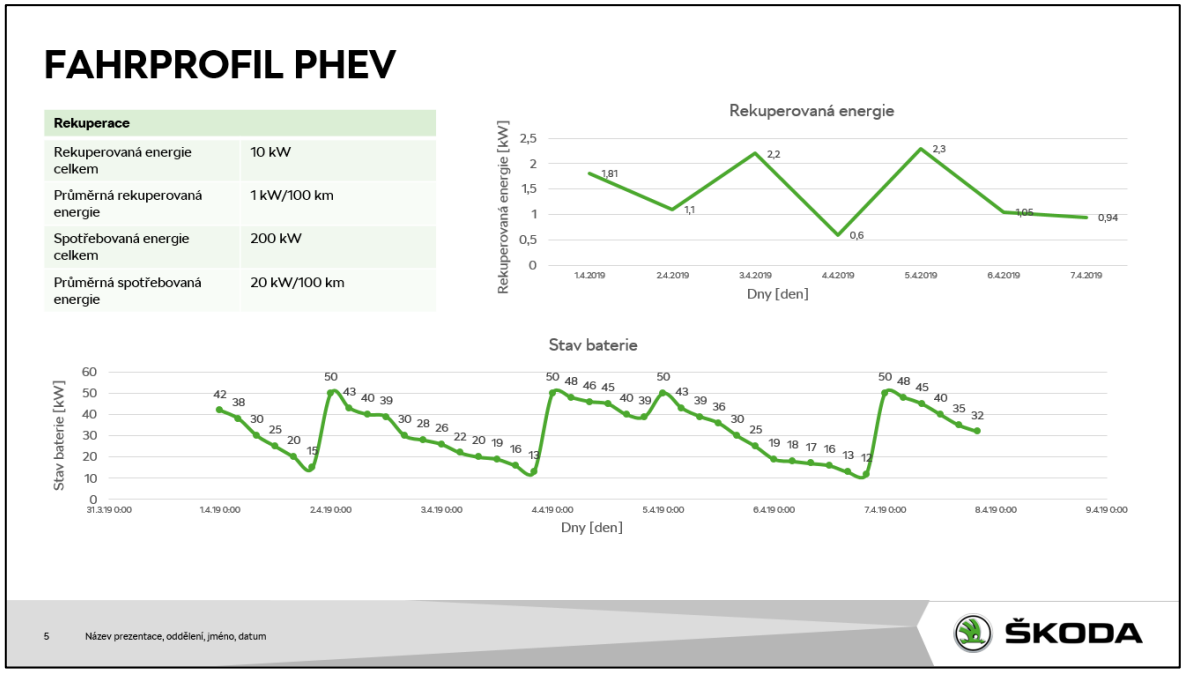
Strana 5

Jelikož jde o hybridní vůz, který disponuje elektromotorem, na této straně se budou nacházet informace týkající se rekuperace energie a spotřeby energie elektromotoru.

Vlevo nahoře bude tabulka s informací o celkové rekuperaci energie po čas vybraného měření. Na dalším řádku bude průměrná rekuperace energie na 100km. Dále zde bude informace o celkové spotřebě elektromotoru a nakonec průměrná spotřeba energie na 100km.

Dále se bude na této stránce nacházet spojnicový graf znázorňující průběh vybíjení vysokonapěťové baterie za každý den z vybraného měření. Na tomto grafu bude zobrazován průběh bez dobíjení baterie. To znamená, že nárůst energie v případě připojení konektoru nabíječky bude z tohoto grafu vyřazen.

Na dalším sloupcovém grafu bude znázorněno množství rekuperované energie za každý den.



Obrázek 11 - FAHRPROFIL PHEV strana 5

Strana 6

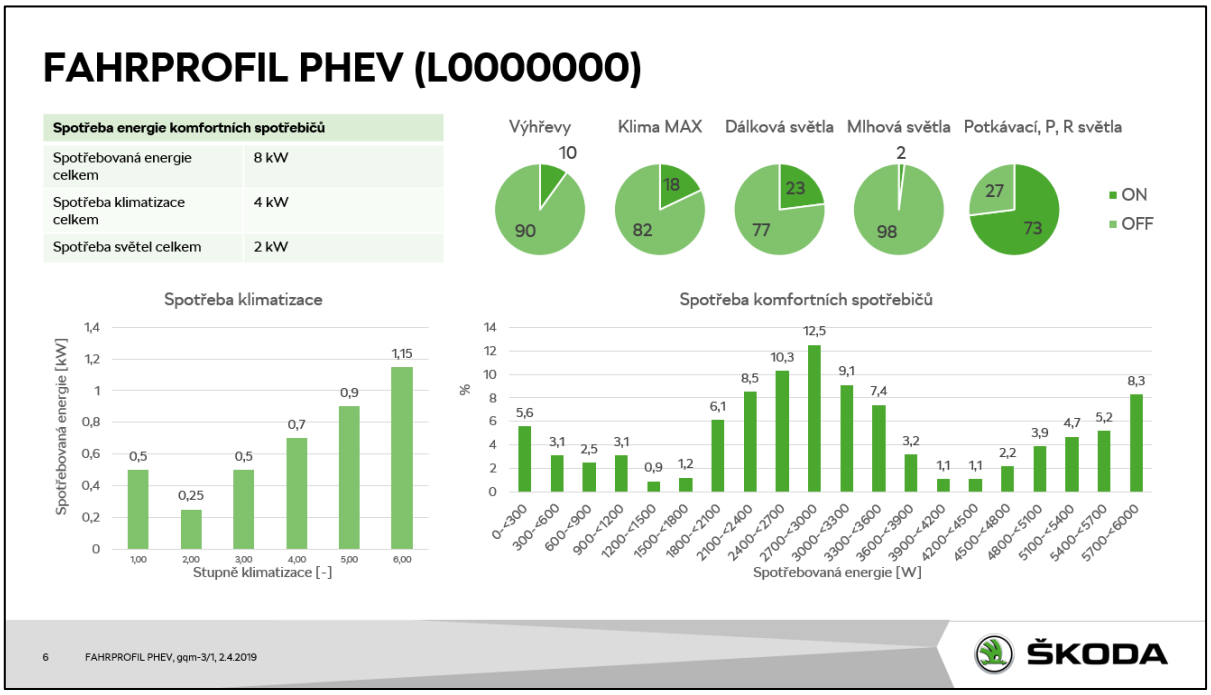
Tato stránka se bude týkat spotřeby energie komfortních spotřebičů, které mohou mít vliv na vybíjení vysokonapěťové baterie.

Vlevo nahoře bude opět tabulka. V této tabulce bude uvedena celková spotřeba energie komfortních spotřebičů. Dále zde bude celková spotřeba energie klimatizační jednotky a celková spotřeba energie všech světel.

Pod touto tabulkou se bude nacházet sloupcový graf s procentuálním používáním jednotlivých stupňů klimatizace.

Další sloupcový graf bude ukazovat celkovou spotřebu energie rozdělenou procentuálně po 300mW až po zjištěné maximum, které by nemělo překročit 6000mW. Tím bude docíleno grafického znázornění toho, v jakých mezních hodnotách se nejčastěji spotřeba energie komfortních spotřebičů pohybovala během zvoleného měření.

Dále se zde bude nacházet pět koláčových grafů, kde bude znázorněno, z kolika procent z jízdy, se používaly námi zvolené komfortní spotřebiče. První graf se bude týkat výhřevů, další maximálního výkonu klimatizace, třetí zapnutí dálkových světel, čtvrtý mlhových světel a pátý potkávacích, parkovacích a couvacích světel.



Obrázek 12 - FAHRPROFIL PHEV strana 6

7.2.3 Výstupní zpráva - AUFLADEN PHEV

Strana týkající se jednotlivých dobíjení

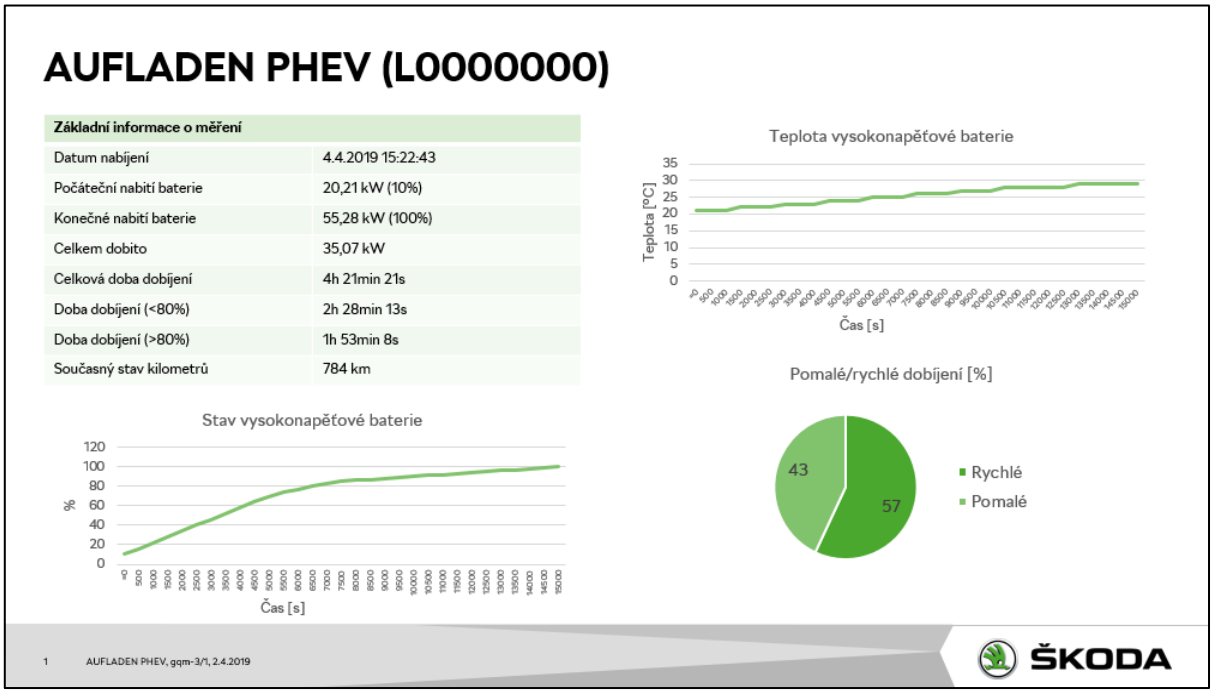
V tomto případě nelze specifikovat, na jaké stránce se bude nabíjení nacházet, jelikož jich může být větší množství. To znamená, že pokud dojde k více dobíjení vozidla během vybraného měření, každá strana bude obsahovat pouze jedno dobíjení.

Vlevo nahoře se bude nacházet tabulka s vybranými informacemi o nabíjení. Na první řádku bude datum a čas začátku dobíjení a hned pod tímto údajem počáteční a konečná úroveň dobití baterie v procentech i kilowatech, celkový čas dobíjení a také celkově dobitá energie opět v kilowatech. Jako další zde musí být čas dobíjení do 80 % kapacity baterie a poté také čas dobíjení od 80 % kapacity baterie a to z důvodu, že po dobití 80 % kapacity baterie je dobíjení vysokonapěťové baterie výrazně zpomaleno. Na posledním řádku bude aktuální počet najetých kilometrů

Pod zmíněnou tabulkou bude spojnicový graf s celým průběhem dobíjení vysokonapěťové baterie v procentech.

Vpravo nahoře se bude nacházet další spojnicový graf s informací o změně teploty vysokonapěťové baterie v průběhu celého dobíjení.

Pod tímto grafem vpravo dole se bude nacházet koláčový graf s procentuálním znázorněním poměru mezi pomalým a rychlým dobíjením vysokonapěťové baterie.

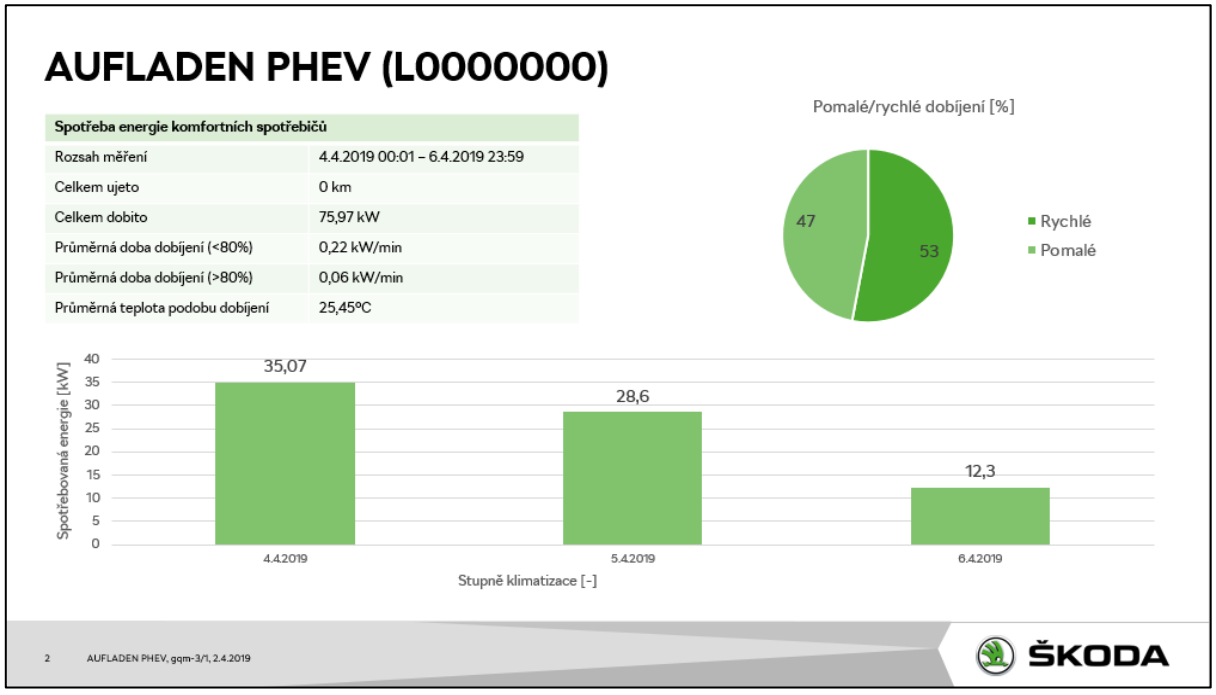


Obrázek 13 - AUFLADEN PHEV strana 1

Závěrečná strana výstupní zprávy „AUFLADEN PHEV"

Na poslední straně této výstupní zprávy bude vyhodnocení ze všech sledovaných dobíjení vysokonapěťové baterie.

Vlevo nahoře se bude opět nacházet tabulka s informací o celkovém nájezdu kilometrů a množstvím dobité energie v kilowatech během sledovaného měření. Hned pod touto informací bude, kolik kW se v průměru dobilo za jednu minutu při pomalém i rychlém dobíjení baterie. Na posledním řádku bude průměrná teplota vysokonapěťové baterie během všech nabíjení.



Obrázek 14 - AUFLADEN PHEV - Poslední strana

8 Výstupní zpráva vypracovaná externí firmou

V zadání bylo uvedeno, že podle mnou zadaných požadavků bude vypracována výstupní zpráva externí firmou, která dlouhodobě spolupracuje s naším oddělením GQM-3/1.

Nicméně výstupní zpráva nebyla zpracována z důvodu, že se v průběhu psaní této bakalářské práce změnily priority našeho oddělení. Bohužel musela být upřednostněna výstupní zpráva pro jiný typ vozu a z toho důvodu externí firma neměla kapacitu na vytvoření mnou požadované výstupní zprávy včas.

Pro představu, jak bude výsledná výstupní zpráva vypadat, jsem vytvořil dokument v Microsoft PowerPoint. Náhledy jednotlivých stránek jsou umístěny v kapitole s požadavky na výstupní zprávu.

Závěr

V teoretické části bakalářské práce jsem představil krátký úvod do dataloggingu jako takového. Dále jsem provedl vytipování nejvhodnějšího typu dataloggeru od společnosti G.i.N. Jako nejvhodnější se ukázal datalogger s označením GL5350, který patří mezi ty nejnovější, má možnost snímat velké množství CAN sběrnic a také je uzpůsoben pro připojení mnoha externích zařízení. V předposlední kapitole teoretické části jsem představil programovací jazyk Log Task Language pro tvorbu konfigurací. Byly zde popsány jednotlivé prvky konfigurace, které jsem následně použil v praktické části této bakalářské práce. V poslední kapitole teoretické části jsem nastínil historii hybridních pohonů a jejích základní rozdělení.

Praktická část této bakalářské práce je primárně zaměřena na konfiguraci vytipovaného dataloggeru. Tato konfigurace měla být uzpůsobena pro datalogging hybridních vozidel a musela splňovat všechny požadavky zadané oddělením GQM-3/1, které spadá pod firmu Škoda Auto a.s.. Veškeré požadavky jsem splnil a konfiguraci jsem navíc obohatil o různé detaily, jako je například logo Škoda Auto a.s. zobrazené na externím displeji dataloggeru. Dále jsem vytvořil požadavky na zpracování výstupní zprávy na základě naměřených dat mnou vytvořenou konfigurací. Toto zadání je určeno pro externí firmu, která pro naše oddělení měření zpracovává a stará se o tvorbu automaticky generovaných výstupních zpráv. Zadání bylo doplněno o grafický návrh v Microsoft PowerPoint.

V průběhu psaní této bakalářské práce došlo ke změně priorit našeho oddělení a bohužel byla upřednostněna analýza dat z elektrických vozidel, a proto musela být upřednostněna tvorba výstupní zprávy pro elektrický vůz. Z toho důvodu externí firma nestihla vytvořit požadovanou výstupní zprávu a nemohla být v praktické části představena.

Během psaní této bakalářské práce jsem si prohloubil znalosti v oblasti dataloggingu a seznámil jsem se s problematikou hybridních pohonů, který se stává čím dál víc oblíbený v oblasti automobilismu. Navíc jsem se naučil plně programovat v jazyce LTL. Znalost tohoto jazyka je nepostradatelná v případě sbírání dat pomocí dataloggerů od společnosti G.i.N.

Seznam použité literatury

G.I.N. GMBH. *LTL (Log Task Language): Programming Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4039/ltl_manual_e.pdf

G.I.N. GMBH. *LOGview: User Manual*. Germany, 2003. Dostupné také z: http://www.gin.de/files/4037/logview_manual_e.pdf

G.I.N. GMBH. *GL1000 / GL1010 Series: User Manual*. Germany, 2018. Dostupné také z: www.gin.de/files/4014/gl1000_manual_e.pdf

G.I.N. GMBH. *GL2000 series: User Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4016/gl2000_manual_e.pdf

G.I.N. GMBH. *GL3000 / GL4000 Series: User Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4018/gl3000_gl4000_manual_e.pdf

G.I.N. GMBH. *GL5000 Series: User Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4475/gl5000_manual_e.pdf

G.I.N. GMBH. *CANgps: User Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4004/cangps_manual_e.pdf

G.I.N. GMBH. *Network Camera (HostCAM): User Manual*. Germany, 2018. Dostupné také z: http://www.gin.de/files/4222/hostcam_manual_e.pdf

G.I.N. GMBH. *GiNconf: User Manual*. Germany, 2017. Dostupné také z: http://www.gin.de/files/4012/ginconf_manual_e.pdf

MOLDASCHL, Jan. *Aplikace datalogerů v automobilech*. Plzeň, 2012. Diplomová práce. ZÁPADOČESKÁ UNIVERZITA V PLZNI. Vedoucí práce Ing. Michal Kubík.

MAREŠ, Milan. *Měření zrychlení vozidla*. Brno, 2012. Diplomová práce. Mendelova univerzita v Brně. Vedoucí práce Ing. Jiří Čupera, Ph.D.

HORČÍK, Jan. Historie hybridních aut: 1. díl. In: *Hybrid* [online]. Hybrid, 2006, 30.9.2009 [cit. 2019-04-15]. Dostupné z: http://www.hybrid.cz/clanky/historie-hybridnich-aut-1-dil

HORČÍK, Jan. Historie hybridních aut: 2. díl. In: *Hybrid* [online]. Hybrid, 2006, 30.9.2009 [cit. 2019-04-15]. Dostupné z: http://www.hybrid.cz/clanky/historie-hybridnich-aut-2-dil

HORČÍK, Jan. Historie hybridních aut: 3. díl. In: *Hybrid* [online]. Hybrid, 2006, 30.9.2009 [cit. 2019-04-15]. Dostupné z: http://www.hybrid.cz/clanky/historie-hybridnich-aut-3-dil

ONORI, Simona, Lorenzo SERRAO a Giorgio RIZZONI. *Hybrid electric vehicles: Energy Management Strategies*. Ilustrované vydání. New York, NY: Springer Berlin Heidelberg, 2015. ISBN 978-1-4471-6779-2.

MILLER, John M. *Propulsion systems for hybrid vehicles*. Ilustrované vydání, dotisk. Stevenage, UK: Institution of Electrical Engineers, c2004. ISBN 08-634-1336-6.

Hybrid and Electric Vehicles: The Electric Drive Plugs In [online]. 2011, **June 2011** [cit. 2019-04-12]. Dostupné z: http:/www.ieahev.org/assets/1/7/IA-HEV_2010_annual_report_6MB.pdf

ANDERSON, Curtis D. a Judy ANDERSON. *Electric and hybrid cars: a history*. 2nd ed. Jefferson, N.C.: McFarland, c2010. ISBN 978-0-7864-3301-8.

MI, Chris a Abul MASRUR. *Hybrid electric vehicles: principles and applications with practical perspectives principles and applications with practical perspectives*. Second edition. Hoboken, NJ, USA: Wiley, 1918. ISBN 978-111-8970-560.

HROMÁDKO, Jan. *Speciální spalovací motory a alternativní pohony: komplexní přehled problematiky pro všechny typy technických automobilních škol*. Praha: Grada, 2012. ISBN 978-802-4744-551.

TEMPLEMAN, Graham. *The Competition Car Data Logging Manual*. England: Veloce Publishing, 2008. ISBN 978-1-84584-162-1.

MCBEATH, Simon. *Competition car data logging*. 2nd ed. Newbury Park, Calif.: Haynes North America, 2008. ISBN 978-184-4255-658.

SMITH, Craig. *The car hacker's handbook: a guide for the penetration tester*. San Francisco: No Starch Press, [2016]. ISBN 978-1-59327-703-1.

EHSANI, Mehrdad, Yimin GAO, Stefano LONGO a Kambiz EBRAHIMI. *Modern electric, hybrid electric, and fuel cell vehicles*. Third edition. Boca Raton, [2018]. ISBN 978-1-4987-6177-2.

DRAGE, Chris. *Primary ICT Handbook: Science* [online]. United Kingdom: Nelson Thornes, 2001 [cit. 2019-04-14]. ISBN 9780748737277. Dostupné z: https://books.google.cz/books?id=0fHv6xrg_r0C&printsec=frontcover&hl=cs#v=onepage&q&f=false

DOYLE, Stephen. *Information System for you* [online]. Third Edition. United Kingdom: Nelson Thornes, 2001 [cit. 2019-04-14]. ISBN 0748763678. Dostupné z: https://books.google.cz/books?id=c8OWnIG-A8EC&pg=PA169#v=onepage&q&f=false

WATTERSON, J.M. *Diagnostic Skills* [online]. United Kingdom: ShieldCrest, 2011 [cit. 2019-04-14]. ISBN 9781907692099. Dostupné z: https://books.google.cz/books?id=iOj97CQdiXEC&printsec=frontcover&hl=cs#v=onepage&q&f=false

PISTOIA, Gianfranco, ed. Electric and hybrid vehicles: power sources, models, sustainability, infrastructure and the market. Boston: Elsevier, [2010]. ISBN 978-0-444-53565-8.

PFM STAFF. Silverado Hybrid…Emergency Power. In: *Hendon: Media group* [online]. Illinois: Hendon Media Group, 2006, 2006 [cit. 2019-04-15]. Dostupné z: http://www.hendonpub.com/resources/article_archive/results/details?id=3782