TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Konfigurace dataloggeru a zpracování diagnostických dat

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jiří Marek**
Vedoucí práce: Ing. Tomáš Martinec, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Datalogger configuration and diagnostic data processing

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Jiří Marek**
Supervisor: Ing. Tomáš Martinec, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří Marek**
Osobní číslo:	**M12000160**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Konfigurace dataloggeru a zpracování diagnostických dat**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s dataloggerem GL3000 od firmy G.i.N. - parametry, možnostmi konfigurace, možnostmi použití v automobilovém průmyslu.
2. Seznamte se s programovacím jazykem LTL, který se používá pro konfiguraci dataloggeru z bodu 1.
3. Vytvořte konfiguraci pro datalogger dle požadavků pro diagnostiku řídících jednotek aut.
4. Zapojte nakonfigurovaný datalogger do testovaného vozidla a proveďte požadovaná měření.
5. Vytvořte aplikaci pro zpracování a vizualizaci výstupních dat.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Kreidl Marcel, Šmíd Radislav: Technická diagnostika - senzory, metody, analýza signálu, BEN - technická literatura 2006**

[2] **Jan Moldaschl: Aplikace datalogerů v automobilech, diplomová práce, Západočeská univerzita v Plzni, 2012**

[3] **Gesellschaft für industrielle Netzwerke mbH. GL3000/GL3100/GL3200. https://gin.de. [Online] [Citace: 20. 10 2015.] http://gin.de/index.php?device=3000&lang=de.**

[4] **GL3000. [Dokument pdf] Griesheim : Gesellschaft für industrielle Netzwerke mbH, 2012.**

[5] **GL3000/GL4000 User manual. [Dokument pdf] Griesheim : Gesellschaft für industrielle Netzwerke mbH, 2011.**

[6] **LTL User Manual. [Dokument pdf] Griesheim : Gesellschaft für industrielle Netzwerke mbH, 2011.**

Vedoucí bakalářské práce: **Ing. Tomáš Martinec, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Petr Liška**
ŠKODA Auto a.s.

Datum zadání bakalářské práce: **10. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

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 tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 12.5.2016

Podpis:

Poděkování

Tímto bych rád poděkoval svému vedoucímu bakalářské práce Ing. Tomáši Martincovi, Ph.D za konzultace, pomoc a cenné rady při vytváření této práce.

Poděkování patří také kolegům z oddělení EGD, kteří mi v průběhu praktikantského pobytu ve společnosti Škoda Auto předali mnoho získaných zkušeností a cenných rad.

Abstrakt

Práce se zabývá principem testování elektrických a elektronických systémů vozidel. Na počátku se práce zaměřuje na přístroj datalogger, který je pro testování vozidel klíčový. Pro snazší pochopení problematiky je také uvedena samostatná kapitola popisující datovou sběrnici CAN, řídicí jednotku gateway a komunikaci řídicích jednotek ve vozidle. Následuje seznámení s konfigurací přístroje datalogger, specifikace zadaných požadavků pro vytvoření vlastní konfigurace a její samotná implementace. V poslední části je uveden popis, specifikace požadavků a implementace aplikace Parser. Kromě uvedeného je součástí práce i zapojení přístroje datalogger do testovaného vozidla, obecné informace o průběhu testování vozidel a použité technologie a postupy v rámci BP.

Klíčová slova

datalogger, gateway, aplikace Parser, CAN, LTL, SAX, C#

Abstract

The thesis deals with the principle of testing vehicles electrical and electronic systems. The first chapter focuses on the datalogger device, which is crucial for vehicle testing. For easier understanding of the issue, there is also a separate chapter describing the CAN data bus, a gateway control unit and communication of control units in a vehicle. The following chapters focus on familiarisation with the datalogger device configuration, specifications of the specified requirements for creating custom configuration and its actual implementation. The last section provides a description, specification of the requirements and implementation of Parser application. In addition to the above, the thesis includes a datalogger device connection to the tested vehicle, general information about the testing of vehicles and the technologies and procedures used within the thesis.

Keywords

datalogger, gateway, desktop application Parser, CAN bus, LTL, SAX, C#

Obsah

Seznam obrázků

Seznam ukázek kódu

Seznam zkratek

ASCII
American Standard Code for Information Interchange

BP
Bakalářská práce

CAN
Controller Area Network

CF
CompactFlash

ČR
Česká republika

FIFO
First In, First Out

GB
Gigabajt

G.i.N.
Gesellschaft für industrielle Netzwerke

HDD
Hard Disk Drive

IE
Internet Explorer

IDE
Integrated Development Environment

IT
Informační technologie

kB
Kilobajt

KG
Kilogram

LCD
Liquid Crystal Display

LED
Light-Emitting Diode

LIN
Local Interconnect Network

MB
Megabajt

MOST
Media Oriented System Transport

OS
Operační systém

RISC

Reduced Instruction Set Computing

ŘJ
Řídicí jednotka

SAX
Simple API for XML

SSD
Solid State Drive

ŠA
Škoda Auto

UMTS
Universal Mobile Telecommunications System

USB
Universal Serial Bus

VW
Volkswagen

W3C
World Wide Web Consortium

WFA
Windows Forms Application

WPF
Windows Presentation Foundation

XML
Extensible Markup Language

XSL
Extensible Stylesheet Language

Úvod

Uplatnění a využití IT v automobilovém průmyslu je v současné době na vzestupu. Vývoj nových produktů a technologií se zrychluje nezastavitelným tempem, vše směřuje k vyšší ohleduplnosti k životnímu prostředí, zajištění bezpečnější jízdy a poskytnutí maximálního komfortu pro řidiče a pasažéry. Při vývoji automobilů se dbá na mnoho faktorů, aby výsledný produkt splnil očekávání zákazníků. Nedílnou součástí vývoje automobilů je také jejich testování.

Téma bakalářské práce je zaměřeno na testování elektrických a elektronických systémů ve vozidle pomocí přístroje datalogger. Hlavním cílem této práce je vytvoření konfigurace pro správu dataloggeru a zpracování výstupních diagnostických dat testovaného vozidla. Práce se skládá z teoretických a praktických částí. Teoretické části slouží jako základ pro pochopení probírané problematiky. V rámci teorie je čtenář seznámen s přístrojem datalogger, popisem konfigurace přístroje a datovou sběrnicí CAN. Praktické části jsou zaměřeny na popis a implementaci vytvořené konfigurace, zapojení nakonfigurovaného dataloggeru do vozidla a vytvoření aplikace Parser.

Bakalářská práce vznikla ve spolupráci se společností Škoda Auto, kde působím jako praktikant na oddělení EGD v Technickém vývoji. Motivací pro realizaci této práce bylo získání nových znalostí nad rámec výuky a uplatnění již získaných v praxi. Další motivací bylo vytvoření užitečné aplikace, která bude využita v rámci společnosti. Při vytváření této práce jsem se seznámil s množstvím nových informací a postupů, které bych tímto rád předal i ostatním.

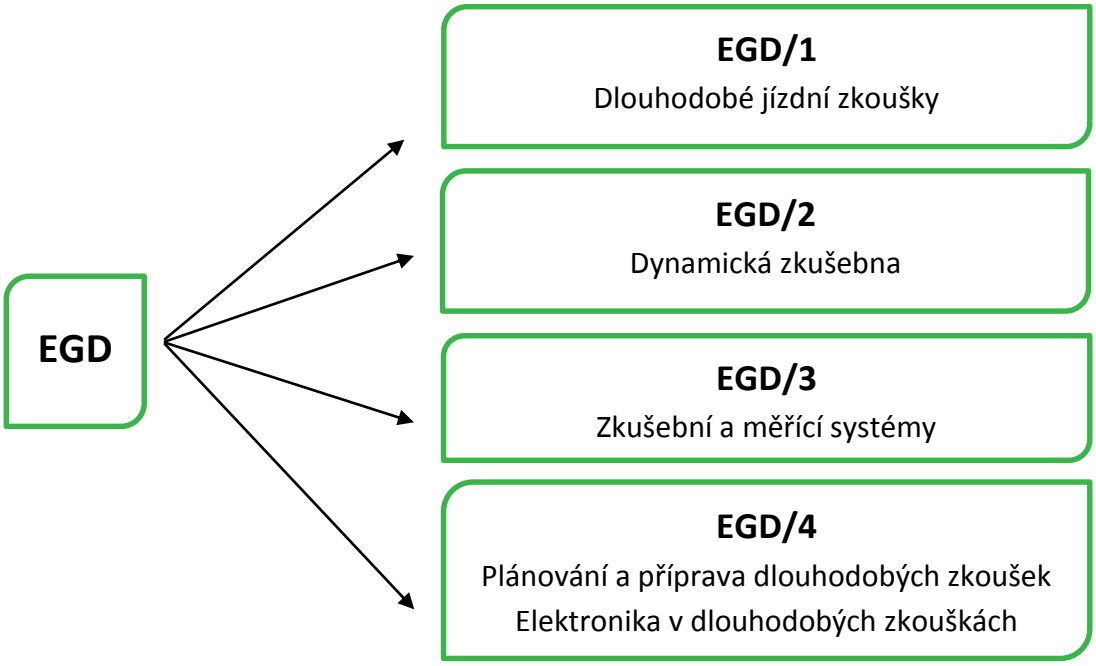
Cíle práce:

- Seznámení s dataloggerem a přístrojem GL3100 od společnosti G.i.N.
- Seznámení s programovacím jazykem LTL pro tvorbu konfigurací dataloggeru.
- Vytvoření konfigurace dle zadaných požadavků s důrazem na diagnostiku řídicích jednotek vozidla.
- Zapojení nakonfigurovaného dataloggeru do testovaného vozidla.
- Vytvoření aplikace pro zpracování výstupních diagnostických dat.

1 O Technickém vývoji a oddělení EGD

Vývoj a testování se provádí ve vývojovém centru zvaném Česana v Mladé Boleslavi. Technický vývoj se skládá z Konstrukčního centra, Technologického centra, Motorového centra a právě vznikajícího Emisního centra. Na vývoji a testování se podílí na dva tisíce specialistů, kteří spolupracují na příslušných projektech.

S vývojem nových produktů, služeb a technologií je spojeno testování kvality a funkcionality výsledného produktu. Oddělení EGD (Entwicklung Gesamtfahrzeug Dauerlauf) provádí dlouhodobé zkoušky životnosti a provozní spolehlivosti vozidel. EGD je rozděleno na čtyři pododdělení, každé z nich vykonává specifickou roli během testování vozidel.



Obrázek 1: Schéma struktury oddělení EGD

1.1 Přehled činností jednotlivých pododdělení EGD

EGD/1

- Realizace dlouhodobých jízdních zkoušek vozidel na veřejných komunikacích a zkušebních polygonech.
- Hodnocení životnosti a provozní spolehlivosti vozidel v průběhu zkoušek.

EGD/2

- Provádění pevnostních a životnostních zkoušek materiálových vzorků, komponent a celých vozů na elektrohydraulických a elektrodynamických zkušebních stavech včetně vyhodnocení dosažených výsledků.

EGD/3

- Instalace měřících snímačů a techniky, měření provozního namáhání vozidel na zkušebních tratích, vyhodnocení a analýza naměřených dat.
- Podpora a vývoj zkušebních postupů s využitím počítačových simulací.

EGD/4

- Plánování dlouhodobých jízdních zkoušek a příprava zkušebních vozidel pro jejich absolvování včetně zajištění úvodních měření a kontrol vozidla.
- Realizace dlouhodobých zkoušek elektronických systémů, sledování a zkoušení elektrokomponent v testovaných vozidlech.
- Nasazení záznamových zařízení – dataloggerů, sběr a vyhodnocení provozních a diagnostických dat.

1.2 Popis dlouhodobé jízdní zkoušky

Dlouhodobé jízdní zkoušky mají za cíl ověření životnosti jednotlivých dílů vozidla a vozidla jako celku. Dále jsou testovány elektronické systémy (např. Parkovací asistent), jejich správná činnost a stabilita v různých podmínkách. Testované vozidlo podstoupí některou z níže uvedených zkoušek na příslušné zkušební trati. Jednotlivé zkoušky jsou mimo různých zkušebních tratí odlišné v délce trvání (počet najetých km), náročnosti terénu a klimatických podmínkách.

Zkušební trati:

- Veřejné silnice
- Polygon Ehra-Lessien v Německu
- Polygony v extrémních podmínkách (teplo, zima)

Přehled některých zkoušek:

- EVP/EWP

  Kurzy EVP a EWP se provádí na polygonu Ehra-Lessien v Německu. Zkouška EVP je zaměřená na pevnost karoserie testovaného vozidla. Vozidlo je podrobeno extrémnímu zatížení karoserie po dobu 8000 km.

  V případě EWP se jedná o všestranný kurz, kdy je testované vozidlo podrobeno různým dostupným situacím, které mohou za normálního provozu nastat. Vozidlo je po celou dobu průběhu zkoušky zatížené na 80 % přípustného limitu. Standardní délka kurzu EWP je 100 000 km.

- SKE/SKT

  Zkoušky s označením SKE a SKT jsou prováděny na veřejných komunikacích v Mladé Boleslavi a blízkém okolí. Zkouška SKE je analogická se zkouškou EWP, pouze se provádí na veřejných komunikacích v ČR. V kurzu SKE nejsou nasazena utajená prototypová vozidla, standardní délka kurzu činí 100 000 km.

  U zkoušky SKT je kladen důraz na provoz ve městě. Zkouška napodobuje taxi provoz o délce 200 000 km.

- KL/WL

  Tyto dvě zkoušky jsou vykonávány na utajených polygonových tratích za extrémních klimatických podmínek. Kurz KL (Kaltland) představuje zkoušku v mrazivých podmínkách. Naopak WL (Warmland) představuje zkoušku při vysokých teplotách.

Testované vozidlo podstoupí jeden zkušební kurz o standardní délce nebo dva zkušební kurzy o zkrácené délce. Každá zkouška probíhá dle specifických požadavků a testovacích postupů, využívá se mnoha testovacích metodik. Vozidlo je vybaveno záznamovým zařízením (dataloggerem) a pečlivě se sleduje jeho provoz po celou dobu průběhu zkoušky. V případě neobvyklého stavu (deformace dílu, ztráta jízdních vlastností, nefunkčnost asistenčního systému…) během průběhu zkoušky se tento problém neprodleně řeší s příslušným oddělením, které se specializuje na danou problematiku. Cílem testování je nalezení, analýza a vyřešení vzniklých problémových stavů.

Průběh dlouhodobé jízdní zkoušky vozidla:

- Plánování zkoušek
- Výroba vozidla
- Příprava vozidla
- Realizace zkoušek
- Závěrečné vyhodnocení

Obrázek 2: Technický vývoj Česana

2 Datalogger

Datalogger je elektronické zařízení pro sledování a zaznamenávání dat. V testovacím vozidle je napojen na datovou sběrnici CAN, která zajišťuje vzájemnou komunikaci řídicích jednotek. Data na sběrnici CAN sleduje a umožňuje jejich uložení do své interní paměti. Následně jsou tato data exportována do výstupních souborů, jejichž obsah se zpracuje a analyzuje. Analýza výstupních dat odhalí případné chyby v řídicích jednotkách, závady na dílech nebo senzorech, a tím začíná řešení příslušné chyby či závady u testovaného vozidla. Velké uplatnění dataloggerů ve ŠA je při testování nových modelových řad nebo prototypových automobilů, využívají se i v servisní síti.

Datalogger představuje samostatné zařízení, které je propojeno s testovaným vozidlem. Při provozu vozidla sbírá data, při nečinnosti se vypne nebo zůstává v režimu spánku. Zařízení je napájeno z akumulátoru vozu, vlastní zdroj elektrické energie není součástí přístroje. S přístrojem datalogger se lze setkat i v jiných odvětvích, viz [10].

2.1 Datalogger GL3100

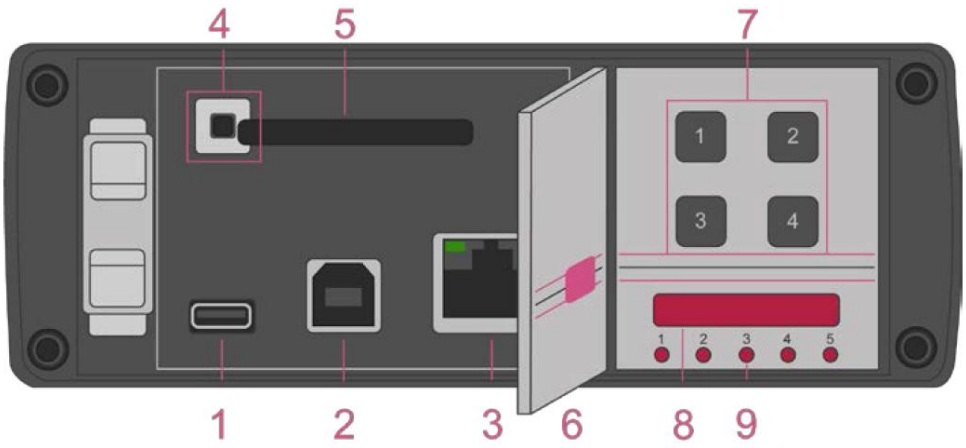
V rámci BP byl využit přístroj Datalogger GL3100 od společnosti G.i.N. Společnost G.i.N. sídlí v německém městě Griesheim a od roku 1991 se zabývá vývojem zařízení umožňujících sledování dat v rámci automobilových sběrnic. Pro více informací o společnosti viz [15]. Koncern VW, tedy i ŠA, spolupracuje se společností G.i.N., která je dodavatelem dataloggerů a příslušenství ke svým produktům.

Obrázek 3: Datalogger GL3100

Přístroj GL3100 pracuje především se sběrnicemi CAN a LIN, které se využívají ve velké míře v automobilovém průmyslu. Pomocí periferie je přístroj schopný spolupracovat i s optickou sběrnicí MOST vyvinutou speciálně pro tyto účely. Přístroj je navržen pro jednoduché rozšíření pomocí vyměnitelných modulů.

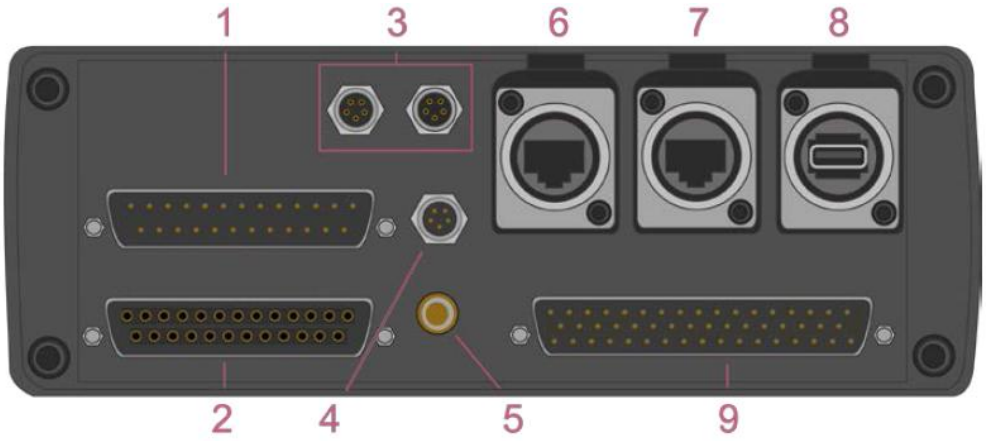
Po komunikační stránce je přístroj osazen Wi-Fi modulem pracujícím pod standardem IEEE 802.11g, a také možností přenosu dat pomocí UMTS. Konektivita s přístrojem je zajištěna mimo jiné pomocí trojice USB portů a síťových kabelů s koncovkou typu RJ-45. Další možností propojení je využití dvojice sériových portů s konektorem DB-25.

Kromě uvedených možností komunikace a konektivity je přístroj vybaven čtyřmi programovatelnými tlačítky, pěti LED diodami a malým displejem pro výpis krátkých zpráv. Vše je obsaženo v odolném kovovém provedení s rozměry 78 x 213 x 237 mm, které přístroj a jeho komponenty chrání. Dle specifikace výrobce je zařízení provozuschopné v teplotním rozmezí od −40 °C do 70 °C. Celková hmotnost přístroje bez příslušenství činí cca 2 kg. Orientační cena přístroje GL3100 s přídavnými moduly a příslušenstvím se pohybuje okolo 4 500 eur (květen 2016). Více informací lze nalézt na [17] a [18].



Obrázek 4: Detail přední strany přístroje Datalogger GL3100 (převzato z [21])

1	USB port (typ A)	6	Ochranná vrátka
2	USB port (typ B)	7	Programovatelná tlačítka
3	Ethernet (RJ-45)	8	Displej
4	Tlačítko pro uvolnění paměťové karty	9	LED diody
5	Pozice pro vložení paměťové karty		



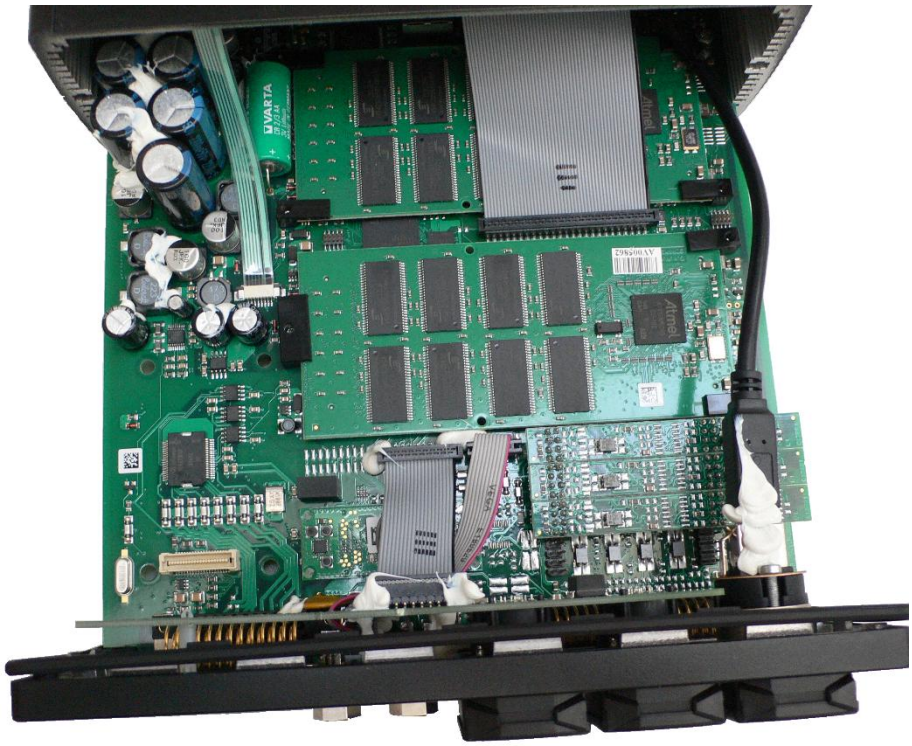
Obrázek 5: Detail zadní strany přístroje Datalogger GL3100 (převzato z [21])

1	Konektor analogových vstupů (DB-25)	6	Ethernet (RJ-45)
2	Konektor digitálních vstupů a výstupů (DB-25)	7	Ethernet (RJ-45)
3	AUX konektor	8	USB port (typ A)
4	Event konektor (unikátní konektor)	9	Hlavní konektor (DD-50)
5	Konektor pro připojení antény (pro Wi-Fi)		

Dataloggery od společnosti G.i.N. jsou programovatelná zařízení, která vykonávají příslušné operace závislé na vytvořené konfiguraci. Konfigurace se vytváří ve vývojovém prostředí, používá se unikátní programovací jazyk LTL. Tvorbě konfigurace se věnuje kapitola *Konfigurace dataloggeru*. Podrobný popis přístroje GL3100 je uveden v manuálu, viz [19].

2.2 Hardware v GL3100

Přístroj GL3100 pohání dvojice 32-bitových RISC procesorů řady ARM9 o frekvenci 240 MHz. Celková operační paměť činí 300 MB, velikost úložiště je stanovena dle vložené paměťové karty typu CF nebo připojeného USB flash disku. Pro potřeby oddělení EGD se při testování vozidel používají paměťové karty o velikosti 4 a 8 GB. Dataloggery s označením GL3200 a GL4000 mohou disponovat samostatným HDD typu SSD. Na přístroji běží speciální distribuce OS Linux s důrazem na rychlé spuštění systému. OS je uložen na paměťové kartě typu CF, nejedná se ale o stejnou kartu, na kterou se ukládají data v průběhu testování. Výrobce dodává datalogger společně s paměťovým médiem, na kterém je uložen OS. Kompletní parametry přístroje jsou uvedeny na [20] a [18].



Obrázek 6: Pohled do útrob přístroje Datalogger GL3100

Rozšiřitelnost hardwaru je zajištěna pomocí vyměnitelných modulů. Pro přístroj GL3100 existují tři možnosti rozšíření.

- Analog Board A8I – modul pro rozšíření přístroje o 8 analogových vstupů
- WiFi Board – modul pro rozšíření přístroje o WiFi připojení
- Babyboard CAN – modul pro rozšíření přístroje o kanál sběrnice CAN

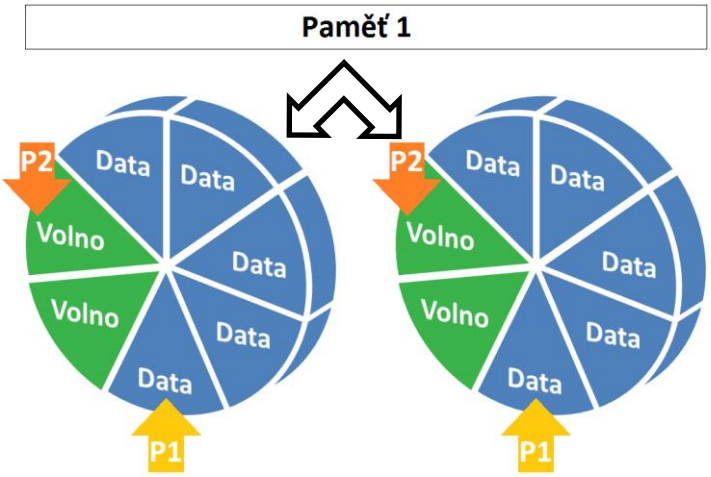
Obrázek 7: Analog Board A8I, WiFi Board a Babyboard CAN (převzato z [16])

2.3 Práce s pamětí

Datalogger GL3100 je vybaven operační pamětí o celkové velikosti 300 MB, z toho je 60 MB rezervováno pro běh OS a zbylých 240 MB slouží pro monitoring a sběr dat. Fyzicky se jedná o dvě od sebe oddělené operační paměti, které mohou pracovat

paralelně nebo nezávisle na sobě. Paměti jsou rozděleny na dva kruhové buffery o stejné kapacitě. Kapacita se definuje v rámci konfigurace, rozmezí je výrobcem stanoveno od 50 kB do 120 MB. Obě paměti musí mít vymezený příslušný prostor.

Kruhový buffer představuje homogenní datovou strukturu, která se skládá z pole o fixní délce a dvou ukazatelích. První ukazatel (P1) ukazuje na začátek kruhového bufferu, tedy na místo v poli, kde se nachází první obsazený prvek. Tento prvek obsahuje data, která byla zapsána jako první, a jsou tedy nejstarší. Druhý ukazatel (P2) ukazuje na první volný prvek, tedy na místo v poli, kam je možné data zapsat.



Obrázek 8: Kruhový buffer

Datalogger využívá princip kruhového bufferu pro uchování části komunikace o příslušné délce, která je definovaná kapacitou bufferu. Cíleně dochází k opakovanému přepisování dříve uložených dat – dříve uložená data jsou nahrazena novými daty (FIFO). Tento způsob zajišťuje aktuálnost zaznamenané komunikace. Celý proces se cyklicky opakuje, dokud se neprovede trigger (viz podkapitola *Pojem Trigger a Trace*).

Pomocí triggeru dojde k uložení obsahu kruhového bufferu na paměťové médium. Přesun a odstranění dat vyžaduje určitou režii, zároveň je ale nutné zajistit uložení příchozích dat do paměti. Vzniklou situaci řeší dvojice kruhových bufferů.

Příchozí data se ukládají pouze do jednoho bufferu, druhý buffer se nevyužívá, ale je připraven k zápisu. Po splnění podmínky triggeru se obsah prvního bufferu přesouvá a maže, příchozí data jsou přesměrována na druhý buffer, a ten je zaznamenává. Při dalším splnění podmínky triggeru se obsah druhého bufferu přesouvá a maže, příchozí data putují zpět do prvního bufferu a takto se celý postup opakuje. Výše popsané je možné díky shodným velikostem obou bufferů. Pro podrobnější popis viz [25].

2.4 Příslušenství

Většina výrobců nabízí ke svým produktům originální příslušenství, které umožňuje rozšířit funkce přístroje a zároveň zajišťuje kompatibilitu. Ani společnost G.i.N. není výjimkou. Kompletní paleta příslušenství je rozsáhlá (viz [16]), podrobněji představen bude LogView a CANgps.

2.4.1 LogView

LogView představuje externí zobrazovací zařízení. Jedná se o monochromatický LCD displej s rozlišením 128 x 64 pixelů a LED podsvícením. Displej dokáže zobrazit sledovaná data v textové podobě s možností využití některých grafických obrazců, záleží na naprogramování. LogView nabízí až 16 programovatelných stránek, přepínání mezi nimi je zajištěno pomocí tří postranních tlačítek. Tlačítka jsou programovatelná, lze je nastavit například i pro vykonání triggeru. Zařízení je propojeno s dataloggerem pomocí konektoru AUX, komunikace je zajištěna pomocí sběrnice CAN. Spuštění a vypnutí přístroje je závislé na stavu dataloggeru.

Obrázek 9: Log View

LogView je vhodný pro sledování dat při samotném testování. Na rozdíl od miniaturního displeje dataloggeru GL3100 jsou data přehledně vypsaná a ihned k dispozici. Rozměry přístroje jsou 66 x 89 x 28 mm.

2.4.2 CANgps

CANgps je GPS modul pro sledování polohy testovaného vozidla při zkoušce. Během jízdy jsou zaznamenávány např. údaje o aktuálním čase, poloze, směru jízdy, rychlosti a nadmořské výšce. CANgps komunikuje s dataloggerem přes sběrnici CAN, a tím je zajištěno provázání dat – data testovaného vozidla jsou propojena s daty z GPS modulu. To přináší cenné

Obrázek 10: CANgps

informace navíc při následné analýze výstupních dat. Modul obsahuje i standardní sériový port (RS-232, DE-9).

Analýza výstupních dat se provádí v aplikaci LogGraph, vizualizaci jízdních tras je možné provést v aplikaci Google Earth. V rámci BP se analyzují diagnostická data, tomuto tématu se věnuje kapitola *Aplikace Parser*.

2.5 Zapojení dataloggeru do testovaného vozidla

Při zapojení přístroje datalogger do vozidla se využívá kabelového svazku. Svazek zajišťuje propojení dataloggeru se sběrnicemi CAN, po kterých vede veškerá komunikace (viz *Datová sběrnice CAN*). Kabelový svazek je buď integrovaný v hlavním svazku, nebo je nutná dodatečná montáž. Z hlediska připojení je jednodušší první z uvedených možností.

2.5.1 Integrovaný svazek

Testované vozidlo je často vybaveno integrovaným svazkem již z výroby. To platí tehdy, pokud se jedná o prototypové vozidlo. Integrovaný svazek představuje kabel se dvěma svorkovnicemi zapojený do hlavního svazku. Hlavní svazek je složen z velkého množství kabelů a konektorů, propojuje a napájí řídicí jednotky a jednotlivé senzory ve vozidle. Jeden z konektorů hlavního svazku vede do řídicí jednotky gateway.

Integrovaný svazek je vyveden pod palubní deskou na straně spolujezdce (u A sloupku). Propojení svazku s dataloggerem zajišťuje kabel s dvojicí svorkovnic na jedné straně a hlavním konektorem (DD-50) na straně druhé. Svorkovnice kabelu a integrovaného svazku se spojí podle své barvy a hlavní konektor se připojí k dataloggeru. Pro připojení GPS modulu je kabel opatřen sériovým portem (konektor DE-9).

Obrázek 11: Kabel propojující svazek a datalogger, vl. svorkovnice, vp. hlavní a GPS konektor

U integrovaného svazku je napájení dataloggeru realizováno přímo z akumulátoru testovaného vozidla přes samostatnou pojistku.

2.5.2 Dodatečná montáž

V případě, že testované vozidlo není vybaveno integrovaným svazkem, je nutné provést dodatečnou montáž modulového svazku. Modulový svazek obsahuje dvojici konektorů pro připojení k řídicí jednotce gateway na jednom konci a dvojici svorkovnic na druhém. Svazek se nazývá modulový, protože vyvedené svorkovnice umožňují připojení různých kabelů nebo zařízení.

Modulový svazek funguje na principu přesměrování dat pomocí klasické Y redukce. Při dodatečné montáži se z řídicí jednotky gateway odpojí konektor, který vede z hlavního svazku. Odpojený konektor se připojí k černému konektoru modulového svazku. Červený konektor modulového svazku je zapojen do konektoru řídicí jednotky gateway. Tímto jednoduchým způsobem se zajistí průchod signálů po modulovém svazku. Data jsou z modulového svazku vedena do gatewaye (červený konektor), a také do dataloggeru (dvojice svorkovnic). Propojení modulového svazku s dataloggerem je vyřešeno stejným způsobem jako u integrovaného svazku.

Obrázek 12: Modulový svazek, vlevo dvojice konektorů, vpravo dvojice svorkovnic

Modulový svazek vede z řídicí jednotky gateway nejen data, ale také poskytuje napájení. Připojený datalogger je tedy napájen z řídicí jednotky gateway.

2.5.3 Konstrukce a umístění dataloggeru

Před zapojením přístroje do testovaného vozidla je přístroj upevněn na speciální plechový nástavec, aby se zamezilo jeho pohybu. Konstrukce nástavce je navržena pro

upevnění k figuríně. Během testování je vozidlo podrobeno určité zátěži, toho se docílí pomocí figurín naplněných vodou. Figurína je posazena na místo spolujezdce a na zadní sedačky. V případě potřeby se zatíží i nákladový prostor vozidla.

Obrázek 13: Konstrukce pro umístění přístroje GL3100 do testovaného vozidla

Nástavec s dataloggerem je umístěn na figuríně sedící jako spolujezdec. Celá konstrukce je i s figurínou pevně přivázána k sedačce spolujezdce a připoutána bezpečnostním pásem. Umístění dataloggeru není náhodné, na vrcholu konstrukce se nachází ovladač (E2T2L), pomocí kterého je možné provést trigger. Zkušební řidič v případě potřeby stiskne tlačítko na ovladači a tím zatriggruje. Pojmu „trigger“ se podrobněji věnuje kapitola *Konfigurace dataloggeru*.

Obrázek 14: Datalogger v provozu, umístěný na sedadle spolujezdce

3 Datová sběrnice CAN

CAN (Controller Area Network) je datová sběrnice využívaná pro vzájemnou komunikaci řídicích jednotek v automobilu. Pomocí sběrnice CAN je možné přepravovat velké množství informací mezi mnoha řídicími jednotkami, rychlost přenosu dat může dosáhnout až 1 Mb/s. Komunikace probíhá pomocí datových protokolů. Více informací nabízí [3] a [4].

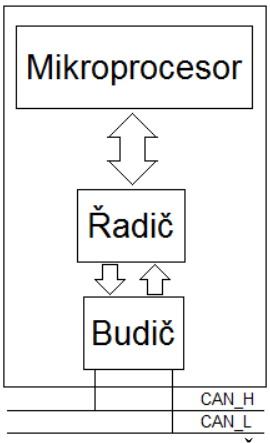
Složení sběrnice CAN:

- Řadič

  Řadič je obsažen v řídicí jednotce (**1**, **2**) a propojuje mikroprocesor a vysílač. Hlavní funkcí je příprava a předávání dat mezi těmito články. Mikroprocesor zpracovává data přijatá ze sběrnice a rozhoduje, která data mají být poslána. Řadič obdrží data, ta připraví a předá na patřičný článek.

- Budič (přijímač a vysílač)

  Budič je napojen přímo na sběrnici CAN a slouží pro transformaci dat. Přijímá elektrické signály ze sběrnice, které mění na data a následně je předá řadiči. Opačnou cestou dochází k přijetí dat od řadiče, jejich přeměně na elektrické signály a odeslání na sběrnici. Budič je součástí řídicí jednotky (**1**, **2**).



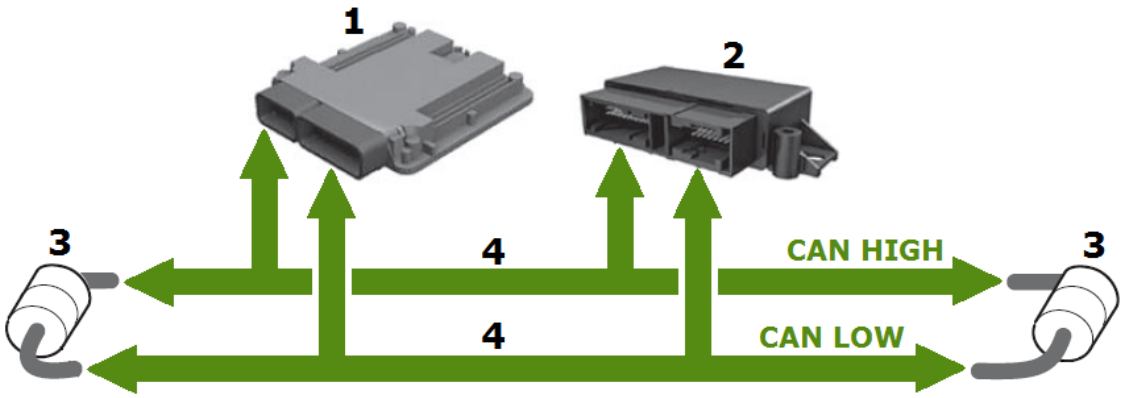
Obrázek 15: Struktura ŘJ

- Vedení datové sběrnice (**4**)

  Jedná se o obousměrné vedení a slouží k přenosu dat. Je provedeno kroucenou dvoulinkou pro zamezení rušení a případné porušení poslaných dat.

- Ukončení datové sběrnice (**3**)

  Ukončení datové sběrnice je zajištěno pomocí dvojice odporů, které zabraňují odrazům elektrických signálů.



Obrázek 16: Datová sběrnice CAN

1	Řídicí jednotka motoru	3	Ukončení datové sběrnice
2	Řídicí jednotka airbagu	4	Vedení datové sběrnice

Výhody datové sběrnice CAN:

- Snížení množství potřebných kabelů k propojení

  Veškerá data se přenáší po dvou vedeních – CAN_H (high) a CAN_L (low), což přináší menší množství potřebných kabelů pro vzájemné propojení řídicích jednotek ve vozidle. S menším množstvím kabelů se docílí úspory místa, snížení hmotnosti a nákladů.

- Rychlý přenos dat mezi řídicími jednotkami vozidla

- Rozšiřitelnost

  V případě přidání další řídicí jednotky do datové sítě postačí jen její připojení na příslušnou sběrnici CAN. Implementace probíhá pouze na softwarové úrovni.

- Normalizace

  CAN sběrnice je normalizovaná po celém světě, a tím je možné vybavit vozidlo řídicími jednotkami různých výrobců – kompatibilita (výměna dat) je zajištěna.

V současnosti roste počet elektronických systémů ve vyráběných vozidlech, a s tím i počet řídicích jednotek. Datová síť dnešních automobilů je rozdělena na několik samostatných CAN sběrnic. Samostatné CAN sběrnice mají jednoznačné označení a zajišťují propojení příslušných řídicích jednotek. Všechny datové sběrnice vozidla jsou propojeny pomocí řídicí jednotky gateway.

3.1 Datový protokol

Pomocí datového protokolu dochází ke komunikaci mezi řídicími jednotkami vozidla. Komunikace probíhá ve velmi krátkých časových intervalech ve formě datových protokolů (zpráv). Protokol se skládá z mnoha po sobě jdoucích bitů, jejichž počet je závislý na velikosti datového pole (DATA). Datový protokol je vytvořen podle datového rámce, struktura rámce je znázorněna na obrázku níže a detailně popsána na [8].

Obrázek 17: Struktura datového rámce (Data Frame)

- Počáteční pole (SOF)

  SOF (Start of Frame) značí začátek datového protokolu, velikost pole je 1 bit.

- Stavové pole (AF)

  AF (Arbitration Field) obsahuje prioritu datového protokolu a je zde uveden název zprávy (např. poloha plynového pedálu). Vyšší priorita značí přednost při odeslání datového protokolu. Velikost pole je 11 bitů.

  Součástí stavového pole je často označován i bit RTR (Remote Transmission Request). Tento bit určuje, zda příslušný datový protokol obsahuje data (Data Frame) nebo se jedná o žádost o data (Remote Frame). Hlavní rozdíl mezi oběma datovými rámci je, že žádost neobsahuje datové pole (DATA).

- Řídicí pole (CF)

  CF (Control Field) obsahuje počet informací, které se nachází v datovém poli. Pomocí toho lze zkontrolovat úplnost zaslané zprávy. Velikost pole je 6 bitů.

- Datové pole (DATA)

  DATA (Data Field) přenáší informace (zprávy) pro ostatní řídicí jednotky. Jedná se o pole s největším množstvím informací, maximální velikost je 64 bitů.

- Kontrolní pole (CRC)

  CRC (Cyclical Redundancy Check) slouží pro zjištění chyb během přenosu. Využívá se metoda založená na cyklickém výpočtu kontrolního kódu dat před a po přenosu. Velikost pole je 16 bitů.

- Potvrzovací pole (ACK)

  ACK (Acknowledgement Field) slouží pro potvrzení přijetí zaslané zprávy. Příjemce sdělí odesílateli, že datový protokol byl korektně přijat. Byla-li zjištěna chyba, je odesílatel s touto skutečností seznámen a dochází k opětovnému zaslání zprávy. Velikost pole jsou 2 bity.

- Ukončovací pole (EOF)

  EOF (End of Frame) představuje konec datového protokolu. Dochází zde k poslední kontrole úspěšnosti přijetí a celistvosti zaslané zprávy. Při chybě se přenos přeruší a následuje opakované zahájení přenosu. Velikost pole je 7 bitů.

V rámci sběrnice CAN existují tři hlavní datové rámce, pomocí kterých se vytváří datový protokol. Nejběžnějším z nich je datový rámec (Data Frame) obsahující data. Druhý rámec představuje žádost o data (Remote Frame) a využívá se, pokud některá z řídicích jednotek žádá o příslušná data. Třetí rámec se označuje jako chybový (Error Frame) a vzniká při detekci chyby v některé ze zaslaných zpráv. Více informací na [7].

Za zmínku stojí i pojem Extended Frame, neboli rozšířený datový rámec. Vše výše uvedené se týká základního datového rámce (Base Frame). Rozdíl mezi nimi je ve velikosti stavového pole (11 bitů u základního, 29 bitů u rozšířeného), jinak jsou stejné. Struktura rozšířeného datového rámce je k dispozici na [6].

3.2 Přenos dat

Přenos dat po sběrnici CAN probíhá pomocí datových protokolů (zpráv). Při přenosu datového protokolu není určen jednoznačný příjemce, zaslanou zprávu obdrží všechny řídicí jednotky v rámci příslušné sběrnice. Každá zpráva obsahuje unikátní identifikátor, který určí příslušnost zaslané zprávy. Po přijetí a kontrole zaslané zprávy se řídicí jednotka rozhodne, zda jsou data potřebná a budou zpracována nebo je lze ignorovat.

Průběh datového přenosu lze rozdělit na následující části:

- Příprava dat

  O přípravu dat se stará řadič řídicí jednotky. Data pro odeslání obdrží od mikroprocesoru, který zpracovává přijatá data a vyhodnocuje aktuální situaci.

- Odeslání dat

  Odeslání dat zajišťuje budič, který přebírá připravená data od řadiče. Dojde k převodu dat na elektrické signály a následnému odeslání.

- Přijetí dat

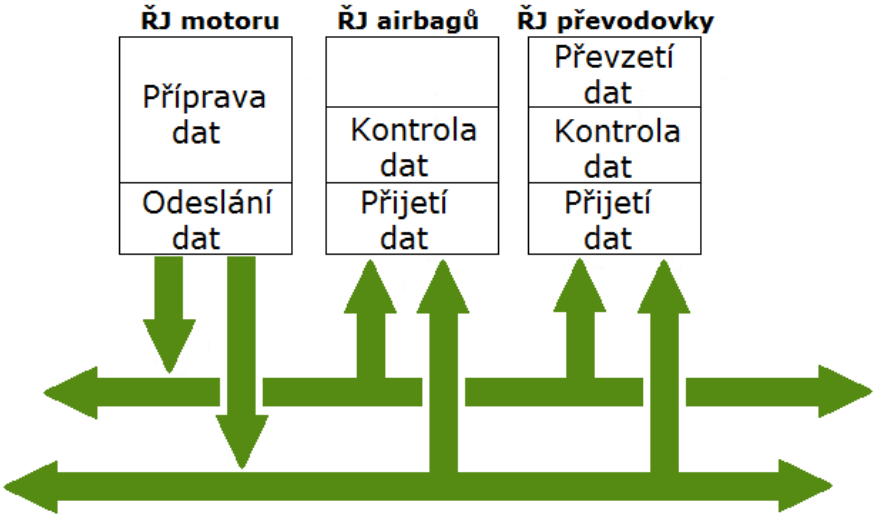
  Pokud některá z řídicích jednotek odešle data, ostatní řídicí jednotky tato data automaticky přijmou. To platí, pokud se uvedené řídicí jednotky nachází v rámci jedné datové sběrnice CAN.

- Kontrola dat

  Kontrolu provádí řadič řídicí jednotky, který přijatá data filtruje. Na základě unikátního identifikátoru zaslané zprávy se rozhodne, zda data budou převzata a zpracována. Nejsou-li přijatá data zapotřebí, řídicí jednotka je ignoruje.

- Převzetí dat

  Pokud jsou zaslaná data vyhodnocena jako potřebná, dojde k jejich převzetí. Následuje předání těchto dat z řadiče na mikroprocesor, kde jsou zpracována.

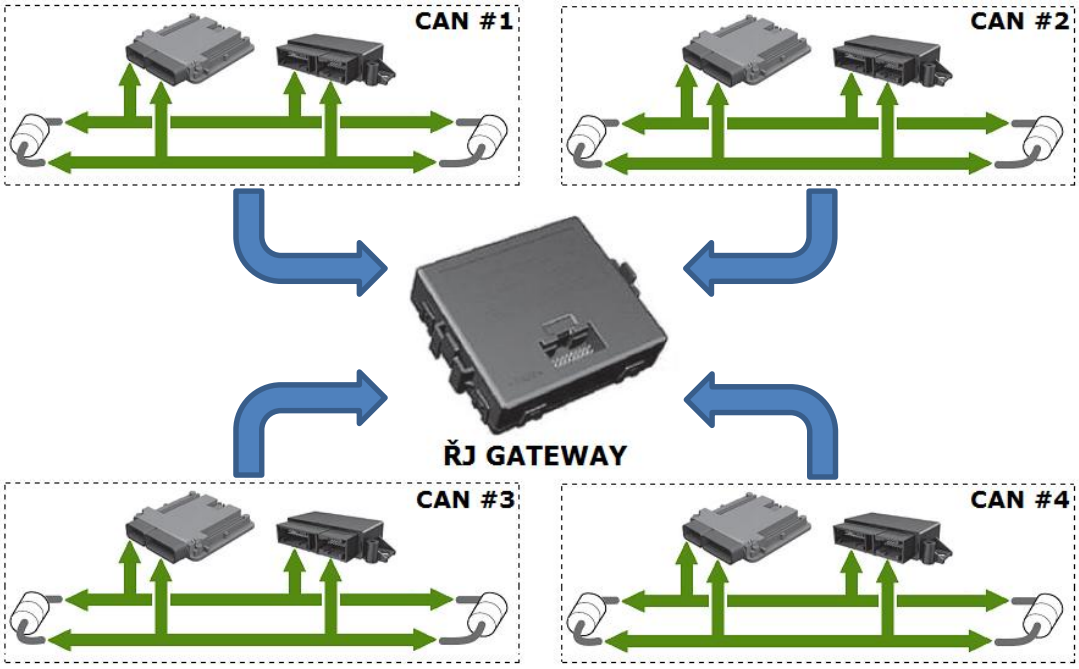


Obrázek 18: Přenos dat mezi řídicími jednotkami po sběrnici CAN

V případě, kdy chce více řídicích jednotek odeslat svůj datový protokol současně, dochází k porovnání priorit. Každý datový protokol je opatřen podle své důležitosti jedenáctibitovým kódem uloženým ve stavovém poli (AF). Priorita datového protokolu se stanoví postupným vyhodnocením tohoto kódu. Více informací k dispozici na [5].

3.3 Gateway

Komunikace mezi více datovými sběrnicemi probíhá pomocí řídicí jednotky gateway. Gateway tvoří komunikační uzel mezi jednotlivými datovými sběrnicemi a zajišťuje přeposílání zpráv (komunikace) mezi nimi. Jednotlivé CAN sběrnice mají za úkol propojit příslušné řídicí jednotky a jsou vedeny do řídicí jednotky gateway. Gateway představuje centrální bod, do kterého vede veškerá komunikace. Následně dochází k přeposílání zpráv mezi jednotlivými sběrnicemi podle stanovené specifikace.



Obrázek 19: Propojení jednotlivých CAN sběrnic s řídicí jednotkou gateway

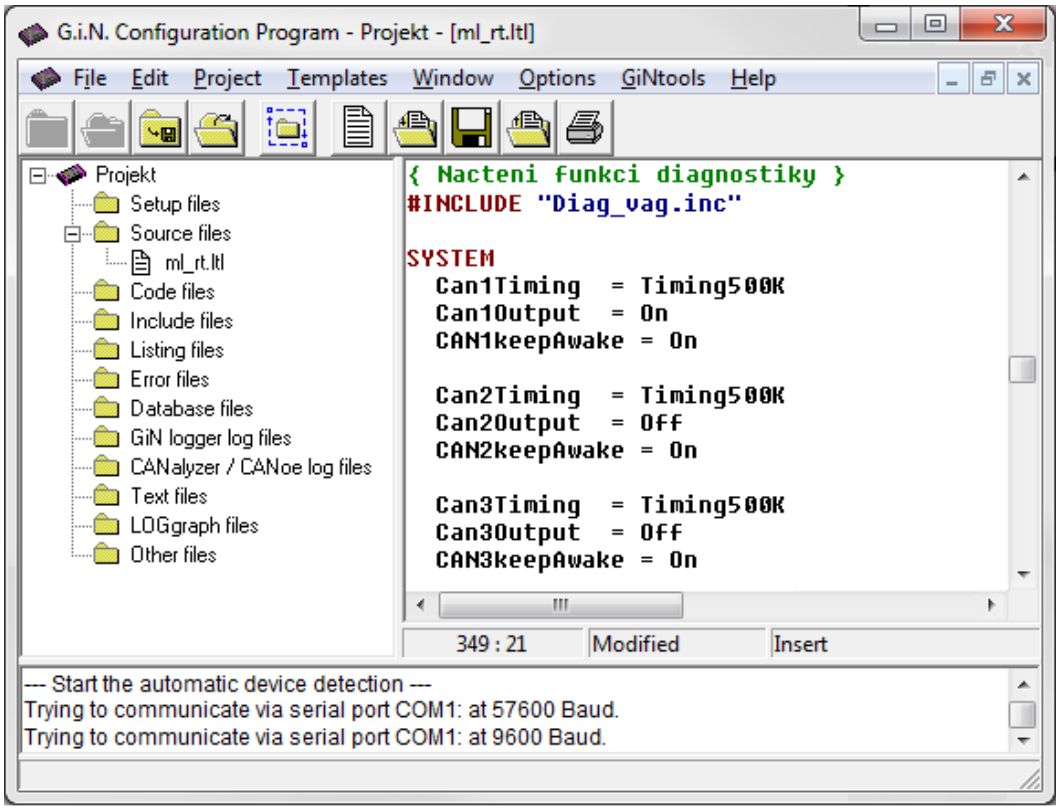
Kromě datových sběrnic je k řídicí jednotce gateway připojen i datalogger. Pomocí tohoto propojení dokáže datalogger naslouchat probíhající komunikaci ve vozidle. V závislosti na dané konfiguraci se sledují a zaznamenávají části komunikace. Tématu konfigurace dataloggeru se věnuje následující kapitola, tématu propojení dataloggeru s vozidlem se věnuje podkapitola *Zapojení dataloggeru do testovaného vozidla*. Více informací o řídicí jednotce gateway a její funkci lze nalézt na [27] a [28].

4 Konfigurace dataloggeru

V průběhu testování se nabízí mnoho možností, jak a co přesně sledovat. Každý testovaný vůz je jiný, ať už modelovým označením, typem motoru a převodovky nebo výbavovým stupněm. Pro jednoznačné určení parametrů, sledovaných při testování, slouží konfigurace. Konfigurace je tvořena konfiguračním souborem, který jasně definuje, co vše bude při testování sledováno a zaznamenáváno. Konfigurační soubor se poté nahraje do paměti dataloggeru. Datalogger podle kódu konfiguračního souboru vykonává zadané příkazy a jejich výsledky zaznamenává do výstupních souborů.

4.1 Programovací jazyk LTL

Konfigurační soubory se programují v jazyce LTL. Log Task Language (LTL) je unikátní programovací jazyk pro tvorbu konfiguračních souborů kompatibilních s dataloggery od společnosti G.i.N. Zdrojový kód se vytváří v programu G.i.N. Configuration Program, který představuje jednoduché vývojové prostředí (IDE). Výsledný kód je zkompilován a připraven k použití, hlavní konfigurační soubor má příponu ltl.



Obrázek 20: Vývojové prostředí G.i.N. Configuration Program

Kód konfigurace je složen z jednotlivých částí, každá část je uvozena klíčovým slovem (např. *CONST, EVENT, SYSTEM*…). Obsah jednotlivých částí se liší, je možné vnořit i více částí do sebe. Pro lepší přehlednost bývá dobrým zvykem provést odsazení obsahu od levého okraje. Příklady syntaxe programovacího jazyka LTL jsou uvedeny v následujících podkapitolách. Podrobnější popis syntaxe lze nalézt na [23] a [24].

Vytvořený program se provádí opakovaně po celou dobu zapnutí přístroje. Datalogger vykonává kód konfigurace chronologicky. Konfigurace se vytváří na míru, podle specifických požadavků. Rozdělení konfigurace do více souborů a jejich následné propojení v hlavním souboru vede k vyšší přehlednosti.

4.2 Pojem Trigger a Trace

Pojmy Trigger a Trace spolu souvisí a pro porozumění následujících částí práce je vhodné se s nimi seznámit.

Trigger

Trigger je událost, při které dojde k uložení dat z kruhového bufferu na paměťové úložiště. Provedení triggeru závisí na nadefinovaných podmínkách uvedených v konfiguračním souboru. Při splnění některé z uvedených podmínek se událost provede. Triggery lze rozdělit do následujících dvou skupin.

- HW trigger

  Datalogger je opatřen dálkovým ovladačem se dvěma tlačítky. Pokud řidič během testování zaregistruje neočekávaný stav nebo nestandardní chování, vyvolá ručně trigger pomocí stisku tlačítka (neboli zatriggruje). O provedení události rozhoduje testovací řidič.

Obrázek 21: Ovladač E2T2L

- SW trigger

  Softwarový trigger představuje automatické vyvolání události (zatriggrování) v případě splnění některé z uvedených podmínek. Definice podmínek je záležitostí programátora, který je dle potřeby uvede v konfiguraci. O provedení události nerozhoduje testovací řidič. Pro více informací o triggeru viz [25].

Trace

Trace představuje záznam dat o příslušné časové délce. Většinou se jedná o kratší úsek komunikace mezi CAN sběrnicemi testovaného vozidla. Záznam dat (trace) je vytvořen po provedení triggeru, uložená data jsou označována jako trace.

Existuje také pojem Full-trace, kdy se již nejedná o kratší úsek komunikace, ale o kompletní komunikaci mezi CAN sběrnicemi po celou dobu testování. Zjednodušeně lze říci, že dochází k zaznamenávání veškeré komunikace po celou dobu průběhu testování. Kvůli enormnímu množství výstupních dat a s tím i spjaté datové náročnosti se full-trace standardně nevyužívá (v rámci oddělení EGD).

4.3 Specifikace požadavků

Požadovaná konfigurace je zaměřena na diagnostiku řídících jednotek testovaného vozidla. Důraz je kladen na identifikaci a výpis chybové paměti příslušné řídící jednotky. Konfigurace také pracuje s triggery a příslušenstvím CANgps pro určení polohy testovaného vozidla.

Specifikace požadavků konfigurace:

- Provedení diagnostiky všech řídicích jednotek v rámci jednoho měření

  Uložení (vyčtení) diagnostických dat ze všech řídicích jednotek testovaného vozidla ve stanovenou dobu. Data obsahují identifikační údaje a výpis chybové paměti příslušné řídicí jednotky. Uložení dat na paměťové médium proběhne každé měření po splnění následujících podmínek:

  - Před začátkem ranní směny
  - Zapnuté zapalování a nulová rychlost vozidla po dobu deseti vteřin

  Pojem „jedno měření“ představuje 24 hodin (3 směny).

- Definice seznamu řídicích jednotek v konkrétním vozidle

  Vytvoření kompletního seznamu řídicích jednotek s možností inicializace vzhledem k testovanému vozidlu. Seznam zajistí možnost znovupoužití konfigurace i pro jiná testovaná vozidla.

- Diagnostika vybraných řídicích jednotek při prvním zápisu do chybové paměti

  Pokud se při testování objeví chyba v některé z řídicích jednotek, ihned se provede diagnostika (identifikace a výpis chybové paměti) a uložení diagnostických a snímaných dat na paměťové médium (SW trigger).

- Diagnostika řídicí jednotky motoru při každém zápisu chyby do paměti

  Sledování chybové paměti řídicí jednotky motoru. Identický postup jako v předchozím bodě, ovšem při zápisu každé chyby.

- Hardware trigger

  Ruční trigger (HW trigger) pomocí tlačítka na dálkovém ovladači. Testovací řidič provede uložení všech snímaných dat po stisknutí tlačítka.

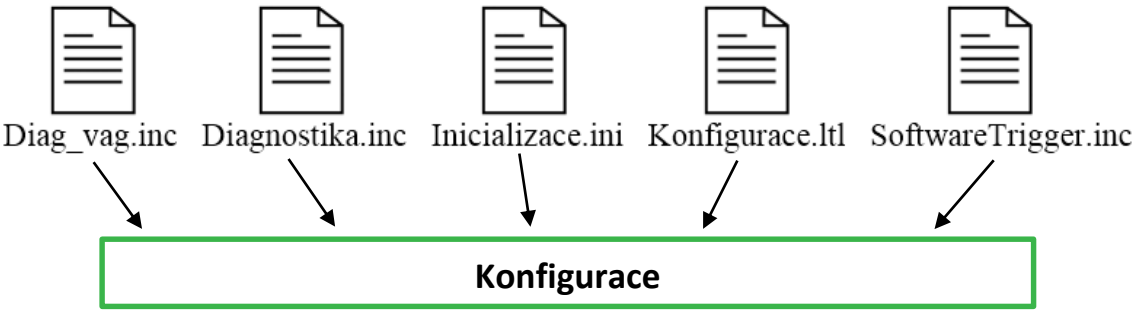
- Záznam GPS souřadnic při triggeru

  Propojení GPS modulu CANgps s dataloggerem GL3100 (fyzicky i softwarově). Pokud při testování proběhne trigger, uloží se data o aktuální poloze testovaného vozidla.

Výstupem popsané konfigurace jsou soubory, které se následně zpracovávají a analyzují. Mezi nimi je i soubor Data1DIAG, který je zpracován pomocí aplikace Parser. Tomuto tématu se podrobně věnuje kapitola *Aplikace Parser*.

4.4 Implementace konfigurace

Vytvořená konfigurace byla naprogramována dle zadaných specifikací. Skládá se z pěti souborů, které jsou propojené v hlavním programu. Podrobná grafická struktura vytvořené konfigurace se nachází v příloze, viz str. 58.



Obrázek 22: Struktura vytvořené konfigurace

4.4.1 Soubor Diag_vag.inc

Podpůrný soubor pro provedení diagnostiky testovaného vozidla, dodaný výrobcem dataloggeru. Tento soubor je nutné využít v případě, že datalogger pracuje v režimu diagnostiky. Obsahem souboru jsou konstanty pro různé módy diagnostiky, adresy řídicích jednotek a kódy pro provedení základních příkazů diagnostiky.

```
CONST
  A_Motorelektronik      = $0101 [16]    {ŘJ motoru}
  A_Getriebeelektronik   = $0202 [16]    {ŘJ převodovky}
  A_Bremselektronik      = $0303 [16]    {ŘJ brzd}
  A_Airbag               = $0515 [16]    {ŘJ airbagu}
  A_Lenkhilfe            = $0944 [16]    {ŘJ servořízení}
```

Ukázka kódu 1: Definice konstant pro provedení diagnostiky

Uvedený kód obsahuje definici konstant a přiřazení adresy příslušné řídicí jednotky. Velikost jednotlivých konstant může být odlišná, výchozí hodnotou je nejmenší možná velikost pro uložení příslušné hodnoty (v případě hodnoty 101h to je 9 bitů). Hranaté závorky slouží pro určení explicitní velikosti konstanty.

4.4.2 Soubor Diagnostika.inc

Tento soubor zajišťuje provedení diagnostiky všech řídicích jednotek testovaného vozidla. Diagnostika se provede po splnění uvedených podmínek a to pouze jednou v rámci příslušného měření. Soubor obsahuje kompletní seznam aktuálně používaných řídicích jednotek během testování vozidel ve společnosti ŠA. Také je zde uveden seznam příslušných požadavků s modifikátory, pomocí kterých se diagnostika provádí.

Seznam požadavků a jejich modifikátory:

- F187 (Číslo dílu)
- F191 (Číslo dílu HW)
- F197 (Označení systému)
- F1A3 (Verze HW)
- F189 (Verze SW)
- F19E (Kód souboru)
- F1A2 (Verze souboru)
- F1A0 (Číslo dílu sady parametrů)
- F1A1 (Verze sady parametrů)
- 0600 (Kódování)

V rámci diagnostiky dojde k oslovení uvedených řídicích jednotek pomocí dataloggeru. Datalogger postupně osloví každou řídicí jednotku množinou uvedených

požadavků (requestů). Odpovědi na tyto požadavky pak tvoří identifikační část. Kromě požadavků zaměřených na identifikaci příslušné řídicí jednotky je také vyslán požadavek pro oslovení chybové paměti. Odpověď na tento požadavek pak tvoří výpis chybové paměti.

```
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0x87(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0x91(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0x97(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0xA3(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0x89(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0x9E(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0xA2(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0xA0(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0xF1(7:0) 0xA1(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[0x22(7:0) 0x06(7:0) 0x00(7:0)] LOG ONLY NOCLOSE WHEN(_01_Motor)
TRANSMIT DIAG CAN1 A_MOTORELEKTRONIK
[UDS_FSPL_FF UDS_FSPL_only_VAS] LOG ONLY NOCLOSE WHEN(_01_Motor)
```

Ukázka kódu 2: Oslovení řídicí jednotky motoru v rámci diagnostiky

Uvedený kód slouží pro oslovení řídicí jednotky motoru. Nejprve se provede identifikace, tu tvoří prvních deset požadavků, a poté se osloví chybová paměť. Podmínka *WHEN* na konci každého požadavku slouží pro ověření, zda se diagnostika příslušné řídicí jednotky provede. Provedení je závislé na hodnotě uvedené konstanty, pokud konstanta nabývá hodnoty *TRUE*, dojde k provedení diagnostiky. Hodnoty těchto konstant se nastavují v hlavním programu, konkrétně v seznamu řídicích jednotek testovaného vozidla. Stejným způsobem se provede diagnostika všech řídicích jednotek testovaného vozidla.

4.4.3 Soubor Inicializace.ini

Jedná se o soubor s výchozím nastavením parametrů pro datalogger GL3100. Parametry nastavení se v rámci tohoto souboru nemění, jejich editace je možná v hlavním programu konfigurace. Jedná se tedy o šablonu s výchozím nastavením, kterou je možné použít v libovolném přístroji GL3100.

Šablona obsahuje konstanty a systémové proměnné s inicializací výchozích hodnot. Konstanty slouží pro slovní pojmenování příkazů tvořených hodnotou. Hodnota pro vykonání příslušného příkazu je přiřazena do konstanty a ta se pak používá v rámci konfigurace (viz níže). Systémové proměnné uchovávají výchozí hodnoty určené od výrobce.

```
CONST
  Timing500K = 0x4114
  Timing100K = 0x4914
  ConnectionRequest      = 0x0001
  ShutdownRequest        = 0x0002
  ConnectionRequestWLAN  = 0x0080
```

Ukázka kódu 3: Definice konstant v inicializačním souboru

Konstanty přináší vyšší čitelnost a přehlednost kódu konfigurace. Místo hodnoty příkazu se použije název konstanty, který tuto hodnotu uchovává.

```
SYSTEM
  Logger1Size   = 5000
  Logger2Size   = 5000
  Logger1Files  = 100
  Logger2Files  = 2
  Can1Timing    = Timing500K
  Can1Output    = Off
  CAN1keepAwake = On
  CAN1Wakeup    = On
```

Ukázka kódu 4: Definice systémových proměnných v inicializačním souboru

Proměnné *Logger1Size* a *Logger2Size* slouží pro nastavení velikosti kruhového bufferu obou pamětí. Velikost kruhového bufferu se udává v kB a určuje délku zaznamenané části komunikace (trace). V proměnných *Logger1Files* a *Logger2Files* je definován maximální počet vygenerovaných souborů při triggeru.

Následující čtyři proměnné slouží pro nastavení CAN kanálu s příslušným číselným označením. Proměnná *Can1Timing* slouží pro nastavení rychlosti, na které připojený CAN kanál běží. *Can1Output* určuje mód provozu dataloggeru – *On* pro diagnostiku, *Off* pro naslouchání komunikace a sběr dat. Zbylé dvě proměnné popisují závislost mezi příchozími zprávami a režimem spánku dataloggeru.

Při tvorbě konfigurace ve vývojovém prostředí G.i.N. Configuration Program se inicializační soubor *Inicializace.ini* automaticky vygeneruje a připojí ke konfiguraci během kompilace.

4.4.4 Soubor Konfigurace.ltl

Hlavní částí celé konfigurace je soubor *Konfigurace.ltl*, který propojuje všechny ostatní soubory, z nichž se konfigurace skládá. Právě zde se nachází hlavní program, podle kterého datalogger pracuje. Konfigurační soubor obsahuje definici projektu, editaci výchozího nastavení dataloggeru, seznam řídicích jednotek, ruční trigger, nastavení pro bezdrátový přenos dat a GPS modul, základní ovládání výstupních zařízení dataloggeru (displej a LED diody) a další parametry pro správnou funkcionalitu konfigurace.

Na začátku konfiguračního souboru je uvedena definice projektu. Definice projektu jednoznačně určí platformu testovaného vozidla a jeho identifikaci. Následuje seznam řídicích jednotek obsažených v testovaném vozidle a nastavení možností diagnostiky.

```
VAR
  PQ26    = FREE [1]        {Fabia, Rapid}
  MQB_A1  = FREE [1]        {Octavia}
CALC
  MQB_A1         = (On)     {Zvolen projekt s označením MQB_A1}
  {Seznam ŘJ ve vozidle}
  _01_Motor  = On    _02_Getriebe    = On    _03_ABS_ESC    = On
  _08_Klima  = On    _13_ACC_Radar   = Off   _44_Lenkhilfe  = Off
  {Nastavení možností diagnostiky ŘJ}
  Diagnostika      = On
  SW_Trigg_Motor   = On          SW_Trigg_Getriebe    = On
  SW_Trigg_ABS     = On          SW_Trigg_Lenkhilfe   = Off
```

Ukázka kódu 5: Definice projektu testovaného vozidla

Soubor dále obsahuje část s editací výchozího nastavení dataloggeru, definovaného v inicializačním souboru *Inicializace.ini*. Za touto částí následuje deklarace vlastních a CAN proměnných, nastavení časovačů a dalších prostředků, s kterými se v rámci konfigurace pracuje.

```
#INCLUDE "Diag_vag.inc"          {Načtení funkcí diagnostiky}
SYSTEM                           {Editace výchozího nastavení}
  Can1Timing       = Timing500K
  Can1Output       = On
  Logger1Size      = 8000
  Logger1Files     = 150
VAR
  Nadm_vyska       = CAN9 DATA 050h [7 6] {CAN proměnné pro GPS}
  Zem_sirka        = CAN9 DATA 051h [3 2]
  Zem_delka        = CAN9 DATA 051h [7 6]
TIMER
  T_Trigger_Display     TIME = 5000    (Trigger_Display)
```

Ukázka kódu 6: Editace nastavení, deklarace CAN proměnných a nastavení časovače

Po deklaraci a nastavení všech potřebných parametrů se v programu vyskytují části zvané *EVENT*, představující událost. U každé události se určí, kdy se má provést (např. cyklicky, při změně hodnoty…) a blok operací, které se provedou. Všechny události jsou opatřeny klíčovými výrazy *BEGIN* a *END*.

```
EVENT                        {Automatické spuštění WLANu}
  ON SET (Hour = 5 AND Minute = 30) BEGIN
  CALC
  SysReq = ConReqWLAN WHEN (ESP_Signal = 0 AND NOT Kl_15)
  END
```

Ukázka kódu 7: Událost v konfiguraci

Uvedená událost slouží pro automatické spuštění bezdrátové sítě a následný přenos nasbíraných dat. Událost se provede při změně stavu proměnných *Hour* a *Minute* na uvedené hodnoty. Dalším požadavkem pro úspěšné provedení události je splnění zadaných podmínek v části *WHEN*. Pokud je vše splněno, dojde k bezdrátovému přenosu nasbíraných dat.

Za seznamem událostí je uvedena část kódu pro ovládání výstupních prvků dataloggeru. Ve vytvořené konfiguraci se jedná o jednoduché ovládání displeje a diod.

```
DISPLAY
  PRINT (0, "TRIGGER")    WHEN (Trigger_Display AND StorDev = 1)
  PRINT (0, "RECORD")     WHEN (NOT DiagActive AND Odpocet > 60)
  PRINT (0, "DIAGNOSE")   WHEN (DiagActive)
OUTPUT
  LED1  = (On)
  LED5  = (Odpocet = 122)
```

Ukázka kódu 8: Manipulace s displejem a diodami dataloggeru

Část *DISPLAY* slouží pro ovládání displeje dataloggeru. Na displeji se zobrazí některý z uvedených textů, pokud jsou splněny zadané podmínky. Manipulace s LED diodami je obsažena v části *OUTPUT*. Dioda s označením *LED1* je aktivní po celou dobu zapnutí přístroje. Dioda s označením *LED5* svítí v případě, že na připojených CAN kanálech probíhá komunikace mezi řídicími jednotkami vozidla.

4.4.5 Soubor SoftwareTrigger.inc

Poslední soubor, tvořící konfiguraci, je *SoftwareTrigger.inc*. Slouží pro provedení diagnostiky uvedených řídicích jednotek. Diagnostika příslušné řídicí jednotky se provede, pokud se při testování objeví záznam v její chybové paměti. Sledování záznamů chybových pamětí zajišťují proměnné, provedení diagnostiky řeší události. Každá z událostí

má stanovenou podmínku, při jejímž splnění dojde k provedení diagnostiky příslušné řídicí jednotky.

```
VAR
  Getriebe = CAN5 XDATA 17F00077h [7(7)] PERSISTENT
  WHEN (SW_Trigg_Getriebe)
  Diagnostika_Getriebe_provedena = FREE [1] PERSISTENT
CALC
  Stop1Trigger.6 = (Getriebe)
{Vyčtení chybové paměti ŘJ převodovky po SW triggeru}
EVENT
  ON SET (Getriebe AND NOT Diagnostika_Getriebe_provedena) BEGIN
   CALC
    Diagnostika_Getriebe_provedena    = (On)
    TRANSMIT DIAG CAN1 A_Getriebeelektronik [0x22(7:0) 0xF1(7:0)
    0x87(7:0)] LOG ONLY NOCLOSE WHEN (_02_Getriebe)
                                ...
    TRANSMIT   DIAG   CAN1   A_Getriebeelektronik   [UDS_FSPL_FF
    UDS_FSPL_only_VAS] LOG ONLY WHEN (_02_Getriebe)
  END
```

Ukázka kódu 9: SW trigger a následné provedení diagnostiky řídicí jednotky převodovky

V kódu se nejprve definuje CAN proměnná *Getriebe*, která slouží pro sledování chybové paměti řídicí jednotky převodovky. Uvedená podmínka ověřuje nastavení možností diagnostiky v hlavním programu, na tom závisí případné provedení události. Dále je uvedena deklarace vlastní proměnné. Část *CALC* naplní šestý bit proměnné *Stop1Trigger*, tato proměnná se v hlavním programu vyhodnotí a způsobí vyvolání triggeru (SW trigger).

Při splnění podmínky události se provede diagnostika řídicí jednotky převodovky. Příkaz *TRANSMIT* odešle na připojený CAN kanál (zde CAN1) zprávu s požadavkem o data (request). Zpráva s požadavkem je adresována příslušné řídicí jednotce. Zpětnou vazbou na zaslaný požadavek je odpověď. Tímto způsobem dojde v rámci diagnostiky k oslovení řídicích jednotek pomocí dataloggeru.

4.4.6 Výstupní data

Vytvořená konfigurace generuje následující výstupní soubory:

- Soubor Data1DIAG s diagnostickými daty
- Soubory obsahující části komunikace (trace)

Kód konfigurace byl vytvořen podle specifikace programovacího jazyka LTL, viz [22].

5 Aplikace Parser

5.1 Popis a funkce aplikace

Aplikace Parser slouží pro zpracování výstupních diagnostických souborů. Tyto soubory vytváří datalogger a obsahují diagnostická data sledovaných řídicích jednotek testovaného vozidla. Vstupem aplikace je právě tento diagnostický soubor, který aplikace zpracuje. Výstupem je dvojice souborů, tvořící přehledný výpis požadovaných dat. Kompletní požadavky pro vytvoření aplikace lze nalézt v podkapitole *Specifikace požadavků*.

Vyhledávání informací v diagnostických souborech je poměrně obtížné a vyžaduje přinejmenším znalost struktury těchto souborů (podrobněji uvedeno dále). Soubory bývají rozsáhlé (cca tisíc řádků a více) a všechna data jsou zde uložena v hexadecimální soustavě. Vyhledání příslušné informace může být časově velice náročné, pokud se připočítá i potřebný čas pro přeložení informace do znakové sady ASCII. A právě zde přináší aplikace podstatné vylepšení. Všechna data jsou v rámci zpracování vstupního souboru rozdělena podle označení řídicí jednotky a přeložena do znakové sady ASCII. Uživateli se zobrazí vygenerovaný výpis dat se všemi potřebnými informacemi.

Součástí výpisu dat je výpis chybové paměti každé řídicí jednotky testovaného vozidla. Jedná se o tabulku, která obsahuje případné chyby v řídicí jednotce. Pokud některá z tabulek obsahuje chybu, dochází k počátku řešení vzniklého problému u testovaného vozidla. Hlavní funkcí aplikace Parser je tedy analýza diagnostického souboru.

5.2 Struktura diagnostického souboru

Diagnostická data jsou uložena ve formě textového souboru s názvem Data1DIAG. Data1DIAG představuje diagnostický soubor a obsahuje kompletní sledované údaje všech řídicích jednotek příslušného testovaného vozidla. Velikost souboru je proměnlivá a závisí na počtu řídicích jednotek vozidla. Struktura souboru je rozdělena na požadavek (request) a odpověď (response).

```
Diagnosedaten:
==============

25.02. 08:43:49.04  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $22  Data: $F1 $9E
25.02. 08:43:49.30  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $62  Data: $F1 $9E $5A $35 $35 $31
25.02. 08:43:49.30  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $22  Data: $F1 $A2
25.02. 08:43:49.33  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $62  Data: $F1 $A2 $5A $35 $35 $31
```

Ukázka kódu 10: Obsah souboru Data1DIAG

5.2.1 Požadavek

Pomocí požadavku dochází k oslovení řídicí jednotky. Požadavek je identifikován kódem *$22* nebo *$19* a žádá příslušnou řídicí jednotku o informace. Součástí každého požadavku je modifikátor, který přesně definuje, o kterou informaci má požadavek zájem. Kromě kódu a modifikátoru je v požadavku obsaženo id řídicí jednotky a datum a čas, kdy byl požadavek zaslán.

```
25.02. 08:43:49.04  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $22  Data: $F1 $9E
------------------------------------------------------------
01                  ID oslovené řídicí jednotky
22                  Označení (kód) požadavku
F1 9E               Modifikátor požadavku
25.02. 08:43:49.04  Datum a čas oslovení ŘJ
```

Ukázka kódu 11: Popis požadavku

Požadovaná množina požadavků a jejich modifikátory:

- F187 (Číslo dílu)
- F191 (Číslo dílu HW)
- F197 (Označení systému)
- F1A3 (Verze HW)
- F189 (Verze SW)
- F19E (Kód souboru)
- F1A2 (Verze souboru)
- F1A0 (Číslo dílu sady parametrů)
- F1A1 (Verze sady parametrů)
- 0600 (Kódování)

Sledované údaje se dělí na identifikační a chybovou část. V identifikační části jsou obsaženy informace o řídicí jednotce (např. číslo dílu, verze hardware, verze software,

kódování…). Chybová část neboli oslovení chybové paměti slouží pro ověření stavu řídicí jednotky. Požadavek s kódem *$22* je určen pro identifikaci, požadavek s kódem *$19* oslovuje chybovou paměť.

5.2.2 Odpověď

Odpověď poskytuje informace, které byly v rámci požadavku dotazovány. Obsah odpovědi je v hexadecimální soustavě, za každým bajtem odpovědi následuje mezera. Ve většině případů se využívá překladu hexadecimální hodnoty na ASCII – jeden bajt je roven písmenu nebo číslici. V některých případech se obsah odpovědi nepřekládá a ponechá se v šestnáctkové podobě. Ne vždy je ale typ odpovědi stejný, celkem může dojít ke třem typům odpovědí:

- Kladná odpověď (kód *$62* nebo *$59*)
- Záporná odpověď (kód *$7F*)
- Žádná odpověď

Ideálním stavem je kladná odpověď – nejprve se zopakuje modifikátor, který byl použit v požadavku, a následně poskytne požadované informace. Záporná odpověď je v souboru uvedena, ale požadované informace neposkytuje. Pokud jsou ve vstupním souboru dva požadavky bezprostředně za sebou, znamená to, že nedošlo k žádné odpovědi.

```
25.02. 08:43:49.30  TP-Adresse = $00  5-Bd-Adresse = $01
  Code = $62  Data: $F1 $9E $5A $35 $35 $31
-------------------------------------------------------
01                  ID oslovené řídicí jednotky
62                  Označení (kód) požadavku
F1 9E               Modifikátor požadavku
5A 35 35 31         Obsah odpovědi
25.02. 08:43:49.30  Datum a čas oslovení ŘJ
```

Ukázka kódu 12: Popis odpovědi (kladná odpověď)

5.3 Specifikace požadavků

Aplikace Parser byla vytvořena na základě zadaných požadavků. Jak již bylo uvedeno, ne všechny informace obsažené ve výstupním souboru jsou požadovány, některé chybí

nebo nejsou uvedeny vůbec. Aplikace musí být připravena vyřešit všechny tyto stavy. Mimo jiné jsou na aplikaci kladeny i další požadavky, jejich přehled je uveden níže.

Specifikace požadavků aplikace:

- Konzolová aplikace bez uživatelského rozhraní (GUI)

  Není požadováno GUI, pouze spustitelný soubor, který vykoná svoji činnost a ukončí se. Po spuštění aplikace je kladen důraz na minimální interakci s uživatelem.

- Možnost spuštění aplikace s parametrem

  Aplikace by měla být implementována také pro spuštění s parametrem. Parametr zadaný v příkazovém řádku představuje cestu k příslušnému vstupnímu souboru. Při spuštění aplikace pomocí spustitelného souboru se možnost zadat parametr neuvažuje. Tento požadavek není možné řešit pomocí nabídky (menu), ve které si uživatel vybere volbu, a to z důvodu minimální interakce aplikace s uživatelem.

- Přenositelnost spustitelného souboru

  Spustitelný soubor bude možné zkopírovat do libovolného adresáře, kde se nachází vstupní soubor Data1DIAG a ten bude zpracován. Pokud se v příslušném adresáři požadovaný soubor nenachází, je tato skutečnost sdělena uživateli.

- Korektní zpracování souboru s diagnostickými daty

  Aplikace musí umět zpracovat požadovanou množinu požadavků a odpovědí, které se v souboru nachází.

- Přehledný výpis dat

  Po zpracování vstupního souboru je požadován přehledný výpis dat. Zobrazená data jsou rozdělena podle názvů řídicích jednotek do přehledných bloků. Doporučeným formátem pro výpis dat je XML.

5.4 Struktura aplikace

Aplikace Parser byla vytvořena v programovacím jazyce C#. Aplikace využívá objektového návrhu (OOP) a lze rozdělit do tří hlavních částí (modulů), které navzájem spolupracují. Při zdárném průběhu je výsledkem přehledný výpis požadovaných dat. Podrobná grafická struktura aplikace se nachází v příloze, viz str. 59.

Hlavní části aplikace:

- LoadFile
- Processing
- Generate

5.4.1 Modul LoadFile

Modul LoadFile je zaměřen na vyhledání příslušného vstupního souboru a nahrání jeho obsahu do paměti. Cestu k souboru je možné určit dvěma způsoby. První a jednodušší způsob je spuštění aplikace ve stejném adresáři, kde se nachází požadovaný soubor. Druhým způsobem je spuštění aplikace se zadaným parametrem pomocí příkazového řádku. V parametru je nutné uvést cestu ke konkrétnímu adresáři, ve kterém se nachází požadovaný soubor Data1DIAG.

Po spuštění aplikace prvním či druhým způsobem se nejprve kontroluje existence souboru v konkrétním adresáři. V případě jeho nenalezení je vykonání programu ukončeno a uživateli se zobrazí chyba, která o této skutečnosti informuje. Pokud soubor existuje, dojde ke kontrole jeho obsahu, a následnému nahrání do paměti. Obsah souboru se vloží do dynamické datové struktury seznam (list).

Struktura zpracovávaného souboru Data1DIAG je rozdělena na požadavek a odpověď. Znění požadavku i odpovědi zabere vždy dva řádky textu. Aplikace tyto dva řádky spojí a uloží celé znění požadavku i odpovědi do seznamu. Následně jsou uložená data zpracována modulem Processing.

5.4.2 Modul Processing

Modul Processing zajišťuje zpracování nahraných dat. Data se nejprve rozdělí podle označení řídicí jednotky, a následně je vytvořen nový objekt pro každou řídicí jednotku obsaženou v seznamu. Poté se nastaví hodnoty jednotlivých atributů. Všechny řídicí jednotky mají stejnou strukturu.

Atributy řídicí jednotky:

- ID
- Název
- Číslo dílu
- Číslo HW
- Označení systému
- Verze HW
- Verze SW
- Kód souboru
- Verze souboru
- Číslo dílu sady parametrů
- Verze sady parametrů
- Kódování
- Chyby

Před nastavením hodnot jednotlivých atributů je zapotřebí příslušné hodnoty z dat získat. Hodnoty atributů jsou uloženy v rámci odpovědí v hexadecimální podobě. Všechny uvedené atributy, kromě atributu *Kódování*, uchovávají ASCII hodnotu a před jejich nastavením proběhne převod dat z šestnáctkové soustavy na ASCII hodnotu. Atribut *Kódování* zůstává v hexadecimální podobě.

```
public long HexToDec(string hex)
{
      long dec = 0;
      if (hex != "")
          dec = Convert.ToInt64(hex, 16);
      return dec;
}
```

Ukázka kódu 13: Funkce pro převod dat do znakové sady ASCII

Výše uvedené představuje identifikační část. Ve výpisu dat je zahrnuta i chybová část, kde jsou uvedeny případné chyby řídicích jednotek. Pokud je obsažena chyba, uloží se její označení, typ a datum, kdy k chybě došlo, do seznamu chyb.

Problematika zpracování dat není složitá, v praxi však dochází k různým situacím, které je třeba řešit.

- Délka zpracovávaného souboru

  Vstupní soubor Data1DIAG nemá pevně stanovenou délku a může obsahovat libovolné množství dat. Obvykle však velikost souboru nepřesáhne 1 MB.

- Neznámý počet řídicích jednotek

  Nikdy není předem známo, kolik řídicích jednotek se ve vstupním souboru nachází. Počet řídicích jednotek je proměnlivý podle modelu či výbavového stupně testovaného vozidla.

- Vícenásobné oslovení příslušné řídicí jednotky

  Ve vstupním souboru může být obsaženo několikanásobné oslovení příslušné řídicí jednotky.

- Různé typy odpovědí

  Není zajištěno, že je odpověď vždy kladná. Při detekci záporné nebo žádné odpovědi je příslušný atribut nastaven na výchozí hodnotu.

- Neznámý typ požadavku nebo odpovědi

  Vstupní soubor může obsahovat neznámý typ odpovědi nebo požadavku. Další možností je neznámý modifikátor, který není zahrnutý v rámci množiny požadavků.

- Kompletní obsah chybové paměti

  Pokud se stejná chyba objeví v rámci příslušné řídicí jednotky opakovaně, je i opakovaně uvedena ve výpisu.

Po zpracování dat následuje jejich přehledný výpis, to zajišťuje modul Generate.

5.4.3 Modul Generate

Poslední část aplikace obstarává výpis požadovaných dat. Data jsou vložena do formátu XML, který je vygenerován a následně spuštěn ve webovém prohlížeči. Spuštění vygenerovaného souboru proběhne automaticky, zajistí jej sama aplikace. Výchozím prohlížečem pro zobrazení vygenerovaného souboru je IE.

```
using (XmlWriter xw = XmlWriter.Create(umisteni, settings))
{
      xw.WriteStartDocument();
      xw.WriteProcessingInstruction("xml-stylesheet", link);
      xw.WriteStartElement("jednotky");
      foreach (RidiciJednotka rj in units)
      {
           xw.WriteStartElement("jednotka");
           if (rj.isEmpty(rj.Nazev) == true)
            xw.WriteElementString("nazev","---neuvedeno---");
           else
            xw.WriteElementString("nazev",rj.Nazev);
                                ...
      xw.WriteEndElement();
      }
```

Ukázka kódu 14: Část funkce pro vygenerování XML souboru

Společně s vygenerovaným souborem typu XML se vygeneruje i soubor typu XSL. Tento druhý soubor slouží jako šablona pro vložení a zobrazení dat. Soubor XML odkazuje na soubor XSL a pro korektní výpis dat je nutné, aby oba soubory byly obsaženy ve stejném adresáři. Obrázek níže znázorňuje rozdíl mezi zobrazením výpisu.

```
<?xml version="1.0" encoding="utf-8" ?>
<jednotky>
  <jednotka>
    <nazev>Řídicí jednotka motoru</nazev>
    <cislo_dilu>3G0906259B</cislo_dilu>
    <cislo_hw>06K907425B</cislo_hw>
    <os>2.0l R4 TFSI</os>
    <verze_hw>H13</verze_hw>
    <verze_sw>0002</verze_sw>
    <ks>EV_ECM20TFS0203G0906259B</ks>
    <verze_souboru>001003</verze_souboru>
    <cdsp>---neuvedeno---</cdsp>
    <vsp>---neuvedeno---</vsp>
    <kod>0C 1D 00 32 24 26 05 0B 30 04</kod>
    <chybovka>
      <id_chyby_hex>003B6B</id_chyby_hex>
      <id_chyby_dec>15211</id_chyby_dec>
      <typ_chyby>28</typ_chyby>
      <datum>03.02. 08:45:45</datum>
    </chybovka>
  </jednotka>
```

Řídicí jednotka motoru

Číslo dílu	3G0906259B
Číslo HW	06K907425B
Označení systému	2.0l R4 TFSI
Verze HW	H13
Verze SW	0002
Kód souboru	EV_ECM20TFS0203G0906259B
Verze souboru	001003
Číslo dílu sady parametrů	---neuvedeno---
Verze sady parametrů	---neuvedeno---
Kódování	0C 1D 00 32 24 26 05 0B 30 04

ID chyby(hex)	ID chyby(dec)	Typ chyby	Datum
003B6B	15211	28	03.02. 08:45:45

Obrázek 23: Výpis dat pomocí XML (vlevo) a pomocí XML společně s XSL (vpravo)

6 Použité technologie a postupy

6.1 C# a .NET Framework

C# je objektově orientovaný programovací jazyk vyvíjený společností Microsoft. Jazyk spadá do skupiny jazyků s virtuálním strojem (virtual machine) a je postaven na základech C++ a Javy. C# využívá .NET Framework, jedná se o prostředí pro běh aplikací. Poskytuje také mnoho knihoven pro vývoj a známé vývojové prostředí (IDE) Visual Studio. C# představuje moderní programovací jazyk s pokročilými možnostmi vývoje softwaru. Jazyk nabízí různé možnosti vývoje software, mezi nejznámější patří konzolové aplikace a okenní aplikace (WFA, WPF).

V rámci této bakalářské práce byla v jazyce C# vytvořena aplikace Parser. Aplikace byla naprogramována ve vývojovém prostředí Microsoft Visual Studio Enterprise 2015 společně s .NET Framework verze 4.6. Specifikace jazyka je volně ke stažení na [11], více informací o jazyce C# a .NET frameworku lze nalézt na [9] a [1].

6.2 XML a XSL

Extensible Markup Language (zkráceně XML) je značkovací jazyk používaný pro publikaci dokumentů a výměnu dat mezi aplikacemi. Zabývá se strukturou obsahu příslušného dokumentu, nezabývá se jeho vzhledem. Dokument ve formátu XML je rozdělen do jednotlivých částí pomocí nadefinovaných elementů, elementy obsahují značky (tagy). O definici vzhledu XML dokumentu a jeho zobrazení se stará jazyk XSL.

Extensible Stylesheet Language (zkráceně XSL) představuje skupinu jazyků pro popis formátování a transformace XML souborů. Pojem XSL zahrnuje následující jazyky.

- XSLT (Extensible Stylesheet Language Transformations)

  Zajišťuje převod dat uložených v XML do jiného formátu, nejčastěji do HTML.

- XSL-FO (Extensible Stylesheet Language – Formatting Objects)

  Značkovací jazyk na bázi XML určený pro formátování XML dokumentů.

- XPath (XML Path Language)

  Dotazovací jazyk pro adresování částí XML dokumentu. Popisuje, která část dokumentu bude zpracována. XPath bývá implementován jako součást XSLT.

Specifikace uvedených jazyků jsou volně a bezplatně k dispozici na stránkách konsorcia W3C. Více informací o specifikacích lze nalézt na [13] a [29]. Obecný popis jazyka XML a skupiny jazyků XSL je uveden na [12] a [14].

6.3 SAX

Simple API for XML (zkráceně SAX) představuje rozhraní pro práci s XML dokumenty. Hlavním cílem je zajištění přístupu k XML souborům a jejich zpracování. Rozhraní umožňuje generování a čtení dokumentů. Čtení i zápis souborů probíhá sekvenčně, což přispívá k menší paměťové náročnosti a v některých případech i k vyšší rychlosti zpracování. Více informací o rozhraní SAX a jeho použití je k dispozici na [2] a [26].

7 Závěr

Během zpracování bakalářské práce jsem se seznámil s reálným využitím IT v automobilovém průmyslu. Na počátku jsem o principu testování elektrických a elektronických systému vozidel věděl pouze nejzákladnější fakta, s postupem času se ale získané znalosti rozšiřovaly.

Seznámení s dataloggerem GL3100, jeho parametry a možnostmi využití v praxi proběhlo úspěšně. V rámci vytvoření vlastní konfigurace dataloggeru jsem se také seznámil se správou přístroje a jeho vnitřním nastavením. Vlastní konfigurace je naprogramována podle zadaných požadavků, provádí diagnostiku řídicích jednotek testovaného vozidla a zajišťuje sledování a zaznamenávání požadovaných dat.

Vytvořená aplikace Parser přináší rychlé a přehledné zpracování výstupních diagnostických souborů. Oproti složitému a zdlouhavému vyhledávání informací v uvedeném souboru je aplikace elegantním řešením. Podstatnou výhodou je také překlad obsahu souboru z hexadecimální soustavy do znakové sady ASCII. Všechna zpracovaná data jsou zobrazena v přehledném výpisu. Možnost vylepšení či rozšíření aplikace shledávám v pokročilejším grafickém výstupu.

Zvolené téma je poměrně složité a rozsáhlé. Téměř každou z hlavních částí této práce by bylo možné rozšířit na samostatnou publikaci – to ale není zamýšleným cílem. Zamýšleným cílem je předání získaných informací a poznatků zájemci o toto téma.

Splnění cílů

- Seznámil jsem se s dataloggerem a přístrojem GL3100 od společnosti G.i.N. ✓
- Seznámil jsem se s programovacím jazykem LTL ✓
- Vytvořil jsem konfiguraci dle zadaných požadavků ✓
- Provedl jsem zapojení nakonfigurovaného dataloggeru do testovaného vozidla ✓
- Vytvořil jsem aplikaci pro zpracování výstupních diagnostických dat ✓

Použitá literatura

[1] .NET Framework – Wikipedia, the free encyclopedia *Wikipedia, the free encyclopedia* [online]. [cit. 2016-03-30]. Dostupné z: https://en.wikipedia.org/wiki/.NET_Framework

[2] About SAX. *SAX* [online]. [cit. 2016-03-30]. Dostupné z: http://www.saxproject.org/

[3] CAN bus – Wikipedia, the free encyclopedia *Wikipedia, the free encyclopedia* [online]. [cit. 2016-03-18]. Dostupné z: https://en.wikipedia.org/wiki/CAN_bus

[4] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN)* [online]. Revised 07/2008. Dallas: Texas Instruments, 2008. [cit. 2016-03-18]. Dostupné z: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[5] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN): Arbitration* [online]. Revised 07/2008. Dallas: Texas Instruments, 2008, s. 4-5. [cit. 2016-03-18]. Dostupné z: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[6] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN): Extended CAN* [online]. Revised 07/2008. Dallas: Texas Instruments, 2008, s. 4. [cit. 2016-03-18]. Dostupné z: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[7] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN): Message Types* [online]. Revised 07/2008. Dallas: Texas Instruments, 2008, s. 5-6. [cit. 2016-03-18]. Dostupné z: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[8] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN): Standard CAN* [online]. Revised 07/2008. Dallas: Texas Instruments, 2008, s. 3. [cit. 2016-03-18]. Dostupné z: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[9] C Sharp – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2016-03-30]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

[10] Data logger – Wikipedia, the free encyclopedia *Wikipedia, the free encyclopedia* [online]. [cit. 2016-02-15]. Dostupné z: https://en.wikipedia.org/wiki/Data_logger

[11] Download C# Language Specification 5.0 from Official Microsoft Download Center. *Microsoft – Official Home Page* [online]. [cit. 2016-03-30]. Dostupné z: https://www.microsoft.com/en-us/download/details.aspx?id=7029

[12] Extensible Markup Language – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2016-03-30]. Dostupné z: https://cs.wikipedia.org/wiki/Extensible_Markup_Language

[13] Extensible Markup Language (XML) 1.0 (Fifth Edition). *World Wide Web Consortium (W3C)* [online]. [cit. 2016-03-30]. Dostupné z: https://www.w3.org/TR/2008/REC-xml-20081126/

[14] Extensible Stylesheet Language – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. [cit. 2016-03-30]. Dostupné z: https://cs.wikipedia.org/wiki/Extensible_Stylesheet_Language

[15] G.i.N. mbH – Intelligente Datenlogger | Company. *G.i.N. mbH – Intelligente Datenlogger | Home* [online]. [cit. 2016-03-18]. Dostupné z: http://gin.de/index.php?id=3013&lang=en

[16] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *G.i.N. Catalog: Accessories* [online]. Griesheim: G.i.N., 2015, s. 31-57. [cit. 2016-04-10]. Dostupné z: http://gin.de/files/4123/gin-katalog2015_en.pdf

[17] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *G.i.N. Catalog: GL3000 series* [online]. Griesheim: G.i.N., 2015, s. 19-21. [cit. 2016-04-10]. Dostupné z: http://gin.de/files/4123/gin-katalog2015_en.pdf

[18] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *Flyer GL3000* [online]. Griesheim: G.i.N., 2015. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/3954/flyergl3000en.pdf

[19] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *GL3000/GL4000 User Manual* [online]. Version 12-2015. Griesheim: G.i.N., 2015. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4018/gl3000_gl4000_manual_e.pdf

[20] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *GL3000/GL4000 User Manual: Features* [online]. Version 12-2015. Griesheim: G.i.N., 2015, s. 7-9. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4018/gl3000_gl4000_manual_e.pdf

[21] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *GL3000/GL4000 User Manual: Ports, Operating and Display Elements* [online]. Version 12-2015. Griesheim: G.i.N., 2015, s. 11. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4018/gl3000_gl4000_manual_e.pdf

[22] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *LTL User Manual* [online]. Version 10.2015. Griesheim: G.i.N., 2015. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4039/ltl_manual_e.pdf

[23] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *LTL User Manual: Basic elements of the Language* [online]. Version 10.2015. Griesheim: G.i.N., 2015, s. 15-21. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4039/ltl_manual_e.pdf

[24] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *LTL User Manual: Basic Objects* [online]. Version 10.2015. Griesheim: G.i.N., 2015, s. 22-45. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4039/ltl_manual_e.pdf

[25] G.i.N. mbH [Gesellschaft für industrielle Netzwerke mbH]. *LTL User Manual: ML and GL3/4 Data Logging* [online]. Version 10.2015. Griesheim: G.i.N., 2015, s. 110-118. [cit. 2016-02-15]. Dostupné z: http://gin.de/files/4039/ltl_manual_e.pdf

[26] SAX Quickstart. *SAX* [online]. [cit. 2016-03-30]. Dostupné z: http://www.saxproject.org/quickstart.html

[27] Sumitomo Electric Industries, Ltd. | Newsletter "SEI WORLD" Central Gateway. *Sumitomo Electric Industries, Ltd*. [online]. [cit. 2016-03-18]. Dostupné z: http://global-sei.com/sn/2015/448/3a.html

[28] Škoda Auto a.s. *Dílenská učební pomůcka č. 54: Gateway*. Mladá Boleslav: Škoda Auto, 2004, s. 15-18.

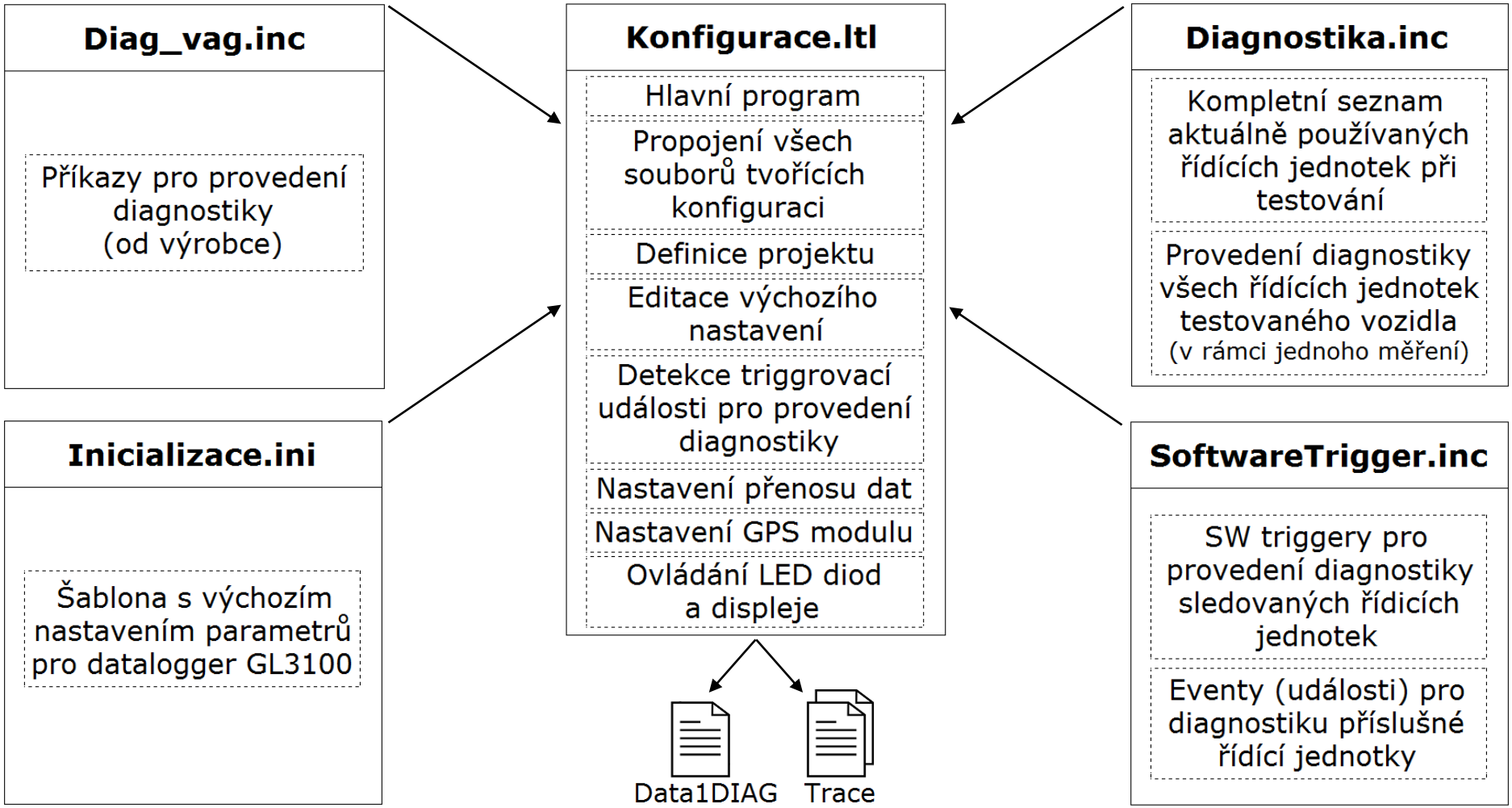
[29] The Extensible Stylesheet Language Family (XSL). *World Wide Web Consortium (W3C)* [online]. [cit. 2016-03-30]. Dostupné z: https://www.w3.org/Style/XSL/

Přílohy

A Obsah CD

- text bakalářské práce v elektronické podobě
    - bp_2016_Jiri_Marek.docx
    - bp_2016_Jiri_Marek.pdf
- adresář Aplikace Parser
    - spustitelný soubor Parser.exe
    - soubor s diagnostickými daty Data1DIAG.txt
- adresář Konfigurace – kompletní vytvořená konfigurace
    - soubor Diag_vag.inc
    - soubor Diagnostika.inc
    - soubor Inicializace.ini
    - soubor Konfigurace.ltl
    - soubor SoftwareTrigger.inc

B Grafická struktura vytvořené konfigurace



C Grafická struktura aplikace Parser