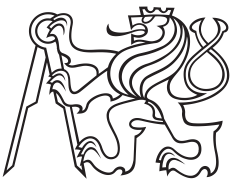
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Nástroj pro vizualizaci anomálií v komunikaci na automobilové sběrnici CAN

Vojtěch Drbohlav
Otevřená informatika

Květen 2014
Vedoucí práce: Ing. Michal Sojka, PhD.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Vojtěch Drbohlav**

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: **Nástroj pro vizualizaci anomálií v komunikaci na automobilové sběrnici CAN**

Pokyny pro vypracování:

1. Seznamte se se sběrnicí CAN a její podporou v OS Linux.
2. Vyviňte grafickou aplikaci pro OS Linux, která bude vizualizovat některé parametry provozu na sběrnici jako např. perioda, jitter a entropie jednotlivých zpráv.
3. Aplikace bude umožňovat analýzu jak živého provozu, tak dat ze záznamu. Aplikace bude navržena s ohledem na ovládání dotykem. Podrobná specifikace bude poskytnuta vedoucím práce.
4. Vše důkladně otestujte a zdokumentujte.

Seznam odborné literatury:

[1] J. Blanchette a M. Summerfield, C++ GUI programming with Qt 4. Upper Saddle River, NJ: Prentice Hall in association with Trolltech Press, 2008.

Vedoucí: Ing. Michal Sojka, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 28. 2. 2014

Poděkování / Prohlášení

Chtěl bych poděkovat vedoucímu práce za jeho odborné vedení, kvalifikované rady a pečlivé testování nástroje, který jsem v rámci této práce vytvářel. A také za cenné znalosti, které mi předával.

Dále bych chtěl poděkovat své rodině za podporu během celého studia.

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

V Praze dne 23. 5. 2014

Abstrakt / Abstract

Cílem mé bakalářské práce bylo vyvinout nástroj, který technikům firmy Volkswagen umožní zkoumat anomálie v komunikaci na automobilové sběrnici CAN. Výsledky tohoto výzkumu lze použít k vytvoření systému pro detekci útoků na sběrnici a tím zvýšit zabezpečení automobilu.

Byl vytvořen grafický nástroj, který umožňuje vizuální inspekci parametrů komunikace na sběrnici jako jsou perioda, entropie a jejich odchylky. Pracuje pod operačním systémem Linux a je naprogramován s využitím knihovny Qt. Uživatelské rozhraní je navrženo s ohledem na ovládání na dotykové obrazovce.

Nástroj byl otestován na log souboru z firmy Volkswagen, na log souboru z robota na katedře řídící techniky i v živém provozu na automobilu Škoda Octavia v laboratoři na ČVUT FEL na Karlově náměstí.

Klíčová slova: analýza komunikace, vizualizace, zabezpečení, intrusion detection system, komunikace v automobilech, CAN, Linux, C++, Qt, QML, dotykové ovládání

The goal of my bachelor thesis was to develop a tool, that would help technicians from the Volkswagen company to research anomalies in CAN bus commucation in cars. The results of this research can be used to create system to detect attacks on CAN bus and increase communication security in cars.

Created graphical tool helps to visually inspect parameters of communication on CAN bus, for example period, entropy and their deviations. This tool works on the Linux operating system and it has been developed using Qt library. Its user interface was designed for using on touchscreens.

It was tested on log files provided by the Volkswagen company, on log files captured from the robot from the department of control engineering and online in the Škoda Octavia in the lab on CTU FEL on Karlovo náměstí.

Keywords: communication analysis, visualisation, security, intrusion detection system, communication in cars, CAN, Linux, C++, Qt, QML, touch control

Title translation: A tool for visualization of anomalies in CAN bus communication in cars

Obsah

Obrázky

Kapitola 1
Úvod

V automobilech je v současné době využíváno mnoho mikrokontrolérů a komponent, které spolu musí komunikovat a předávat si potřebná data a jejich počet neustále roste. Ke komunikaci těchto zařízení se využívají sběrnice vyvinuté pro tyto účely.

Jsou na ně kladeny vysoké požadavky na spolehlivost komunikace, protože jakákoli porucha v komunikaci by mohla ohrozit bezpečnost provozu automobilu.

Mezi základní požadavky patří například doručení zprávy v daném čase, potvrzení doručení zprávy a také odolnost proti možnému rušení. Těmto požadavkům nevyhovují rozšířené protokoly pro síťovou komunikaci jako jsou například Ethernet nebo TCP/IP, proto se v automobilové komunikaci nepoužívají.

Jedna ze sběrnic, která se pro komunikaci jednotlivých zařízení v automobilech používá, je sběrnice Control Area Network (CAN) a právě vizualizací parametrů komunikace na této sběrnici jsem se zabýval.

Vyvinutý grafický nástroj umožní zkoumat vybrané dynamické parametry komunikace na sběrnici CAN v normálním stavu a hledat anomálie při různých pokusech o ovlivnění této komunikace a při případných útocích, které se projevují změnou vizualizovaných parametrů. Výsledky této analýzy budou technici ve firmě Volkswagen využívat při zvyšování zabezpečení komunikace na sběrnici proti kybernetickým útokům. Mohou také sloužit jako základ pro návrh vzorů pro Intrusion detection system (IDS), případně při vývoji systému pro behaviorální analýzu provozu na sběrnici.

Zajímavostí je, že jedním z požadavků byl provoz nástroje na notebooku s dotykovou obrazovkou pod operačním systémem Debian GNU/Linux. Proto jsem při vývoji nástroje využíval multiplatformní knihovnu Qt a pro návrh uživatelského rozhraní jazyk QML, který tato knihovna poskytuje, a který je primárně určen pro vývoj uživatelských rozhraní pro dotykové ovládání.

Kapitola 2
Technické zázemí

2.1 Zabezpečení

Zabezpečení komunikace v automobilu proti kybernetickým útokům je velmi důležité. Elektronika v automobilu provádí rozhodování na základě dat, která si vyměňují jednotlivá zařízení prostřednictvím příslušné sběrnice. Pokud by se útočníkovi podařilo výměnu těchto dat nebo přímo tato data ovlivnit, mohlo by dojít až ke snížení bezpečnosti provozu automobilu [1–3].

K zabezpečení komunikace na sběrnici se využívají technologie, které útokům předcházejí, mezi ně patří i tzv. „Intrusion detection system (IDS)“. Dalším způsobem jak zvýšit zabezpečení je možnost detekovat již probíhající útok, který nebyl zastaven žádným firewallem nebo již zmíněným IDS. Tato technika se nazývá behaviorální analýza síťového provozu.

2.1.1 Intrusion detection system

Intrusion detection system je obranný systém, který monitoruje komunikaci na síti, případně na sběrnici a detekuje aktivitu, která by mohla znamenat, že se někdo snaží ohrozit zabezpečení.

Systém monitoruje komunikaci na síti a podle daných vzorů dokáže rozpoznat útok a případně se podle toho zařídit, například odpojit útočníka nebo zobrazit notifikaci, že se někdo pokouší ovlivnit komunikaci.

Nástroj, který jsem ve své bakalářské práci vytvořil by měl v budoucnu pomoci s návrhem takového systému, protože pomocí něj bude možné zkoumat anomálie v komunikaci při případném útoku. Díky těmto pozorováním mohou být vytvořeny vzory různých útoků, podle kterých bude moci navržený systém reagovat na hrozby [4].

2.1.2 Behaviorální analýza síťového provozu

V případech, kdy je nutné zabezpečit systém i proti útokům, u nichž neznáme předem jejich vzor chování, tak je možné využít behaviorální analýzu síťového provozu. Díky sledování zranitelností, chování zařízení na síti a detekci anomálií může tento systém podle netypického chování odhalit již probíhající útok a například upozornit administrátora sítě nebo odpojit podezřelé zařízení od sítě.

Za běžného provozu si systém vytváří model typického chování sítě a zařízení v této síti. Pokud chování nějakého zařízení neodpovídá danému modelu, tak to znamená, že zařízení je pravděpodobně napadeno nebo se někdo snaží na síť zaútočit. Tento způsob detekce napadení není tedy založen na předem známých vzorech, ale na analýze chování sítě a zařízení v síti, díky tomu dokáže odhalit i nové způsoby útoku u kterých ještě není znám daný vzor charakterizující útok.

2.2 CAN

Control Area Network (CAN) je sériová sběrnice navržená tak, aby umožňovala jednoduchou, ale přitom efektivní komunikaci prvků na síti. Při jejím návrhu byl kladen důraz hlavně na deterministický provoz na sběrnici a snadné řešení kolizí posílaných zpráv.

Snadné řešení kolizí je možné díky tomu, že každá zpráva má svou prioritu. Ta se určuje podle CAN identifikátoru (CAN ID) zprávy, čím je CAN ID nižší, tím má zpráva vyšší prioritu. Pokud zařízení na sběrnici vyšlou dvě nebo více zpráv současně, tak je zachována pouze zpráva s nejvyšší prioritou, tedy nejnižším CAN ID a je doručena všem zařízením, která mohou zprávu zpracovat.

Každá zpráva obsahuje CAN ID a může obsahovat až 8 bytů dat. Jakékoliv zařízení na sběrnici může takové zprávy posílat, ale není možné, aby vysílalo více zařízení současně. Maximální přenosová rychlost sběrnice je 1 Mb/s. Pro vlastní přenos je využito Non-Return-To-Zero (NRZ) kódování.

V dnešních automobilech je velké množství senzorů a jednotek, které komunikují pomocí protokolu CAN. V autě je několik nezávislých sběrnic, které komunikují různými rychlostmi a slouží k různým účelům. Všechny sběrnice jsou propojeny dohromady pomocí CAN gateway. Jedna sběrnice může být využita pro posílání informací o provozu automobilu, jako je například rychlost. Jiná může přenášet například požadavky na stažení okénka, aktivování airbagu a podobně [5–6].

Na operačním systému Linux je podpora pro sběrnici CAN implementována v ovladačích, které jsou v jádře systému, tato implementace se nazývá SocketCAN [7]. Podobně jako existuje rozhraní *lo* pro lokální komunikaci na Ethernetu, tak lze pro CAN pomocí modulu *vcan*, vytvářet lokální virtuální CAN sběrnice na počítači vývojáře.

Pro práci s CAN sběrnicemi na Linuxu existuje balíček can-utils [8], který obsahuje různé nástroje pro práci se sběrnicí. Já jsem využíval hlavně *candump*, který slouží k zobrazení aktuálně probíhající komunikace na sběrnici a případnému uložení do souboru. Druhý nástroj který jsem používal je *cangen*, ten slouží ke generování provozu na sběrnici podle předem zadaných parametrů. Mezi další nástroje z tohoto balíčku patří například *canreplay*, který umožňuje přehrávání log souboru vygenerovaného pomocí *candump* na virtuální sběrnici.

Kapitola 3
Požadavky, návrh aplikace

Základní požadavky na grafické rozhraní aplikace a na její funkčnost jsou formulovány přímo techniky firmy Volkswagen. Podle těchto požadavků jsem navrhl jak architekturu a datové struktury aplikace, tak i její grafické uživatelské rozhraní.

3.1 Požadavky

Z firmy Volkswagen byly vzneseny tyto základní požadavky:

- Vytvořit nástroj pro vizualizaci vybraných dynamických parametrů provozu na sběrnici CAN. Mezi tyto parametry patří perioda, odchylka periody, entropie a odchylka entropie zpráv.
- Konfigurovatelný výpočet entropie - možnost zvolit jaké bity zprávy se pro výpočet používají a jaké se ignorují.
- Zobrazení zpráv s konkrétním CAN ID v čase tak, jak jsou přenášeny po sběrnici.
- Analýza online provozu po připojení počítače ke sběrnici i analýza offline dat zaznamenaných v log souboru nástrojem candump.
- Možnost posouvat se v čase při analýze online provozu i při analýze offline dat z log souboru.
- Uživatelské rozhraní navržené pro ovládání na tabletu s rozlišením 1920x1080.
- Provoz aplikace na operačním systému Debian GNU/Linux [9].
- Možnost skrýt nebo zobrazit hlavní části uživatelského rozhraní.
- Ukládat nastavení podle aktuálního načteného log souboru.
- Možnost načíst vlastní nastavení aplikace ze souboru.

Další připomínky k aplikaci vznikaly během testování a byly zapracovávány průběžně. Jednalo se hlavně o úpravy uživatelského rozhraní.

3.2 Architektura, datové struktury

Aplikaci jsem rozdělil na tři části. Na jádro aplikace, na uživatelské rozhraní a na modul, který se stará o ukládání a načítání nastavení aplikace.

Jádro aplikace získává veškeré zprávy ze sběrnice během analýzy online provozu i při načítání log souborů. Dále umožňuje spustit a pozastavit analýzu komunikace a vykonává veškeré výpočty period, entropií a jejich odchylek.

Mezi nejdůležitější datové struktury jádra aplikace patří hlavně CanLog, CanInterface, CanIdGroup, CanFrame a struktury, které provádí vlastní výpočty period, entropií a jejich odchylek.

CanFrame reprezentuje jednotlivé CAN zprávy. Obsahuje informace o konkrétní zprávě jako jsou její identifikátor, CAN sběrnici na které byla zpráva přijata, její čas doručení a její obsah.

CanLog umožňuje ovládat analýzu komunikace na sběrnici, spouštět analýzu online provozu a načítat data z uživatelem specifikovaného log souboru. Podle potřeby také vytváří CanInterface pro jednotlivé CAN sběrnice, která se v komunikaci vyskytují.

CanInterface slouží k vlastnímu získávání zpráv z příslušné CAN sběrnice, případně jejich přehrávání z log souboru. Pro jednotlivé CAN ID, která se v komunikaci na dané sběrnici vyskytují, vytváří CanIdGroup.

CanIdGroup provádí vlastní výpočty period, entropií a jejich odchylek a poskytuje rozhraní pro získání těchto hodnot ostatním částem aplikace.

Druhou částí aplikace je uživatelské rozhraní. To zahrnuje definici uživatelského rozhraní popsaného jazykem QML a modely, které získávají data z jádra aplikace a připravují je pro zobrazení v uživatelském rozhraní. Tato část také obsahuje struktury, které udržují stav uživatelského rozhraní.

Poslední částí je modul, který umožňuje ukládat a načítat nastavení aplikace. Je využíván oběma již zmíněnými částmi aplikace.

Kapitola 4
Implementace

Pro vývoj aplikace jsem použil integrované vývojové prostředí (IDE) Qt Creator [10] a programoval jsem ji v programovacím jazyce C++, protože s ním mám největší zkušenosti a přišel mi pro tento účel vhodný. Aplikace je poměrně výkonově náročná, protože se často aktualizují hodnoty period a entropie a na sběrnicích se pohybuje velké množství zpráv.

Využíval jsem knihovnu Qt verze 5.2 [11–12] od firmy Digia. Knihovna je multiplatformní, takže bez problémů funguje na požadovaném operačním systému Debian GNU/Linux. Kromě toho že je knihovna multiplatformní patří mezi její výhody usnadnění práce se soubory, systém signálů a slotů pro komunikaci mezi objekty, základní podpora načítání a ukládání nastavení aplikace, jazyk QML a připravené datové struktury pro komunikaci s uživatelským rozhraním.

Jak jsem již popsal v sekci 3.2 (Architektura, datové struktury), aplikaci jsem rozdělil na tři hlavní části, na jádro aplikace, uživatelské rozhraní a modul pro načítání a ukládání nastavení.

Všechny tři části spolu komunikují pomocí systému signálů a slotů. Jedná se o systém, který umožňuje snadné a rychlé předávání zpráv mezi objekty. Objekt, který chce reagovat na vyvolání nějakého signálu připojí svůj slot k tomuto signálu a když je signál vyvolán, tak jsou automaticky zavolány všechny sloty, které jsou na něj připojené.

Pro tvorbu uživatelského rozhraní jsem použil jazyk QML, který je primárně určen k tvorbě uživatelského rozhraní pro dotyková zařízení. Dalším z důvodů pro výběr QML je to, že se v něm dá rozhraní vytvářet velmi rychle, což bylo vhodné při prvotním návrhu i při dalším zpracování připomínek od vedoucího práce a od techniků firmy Volkswagen.

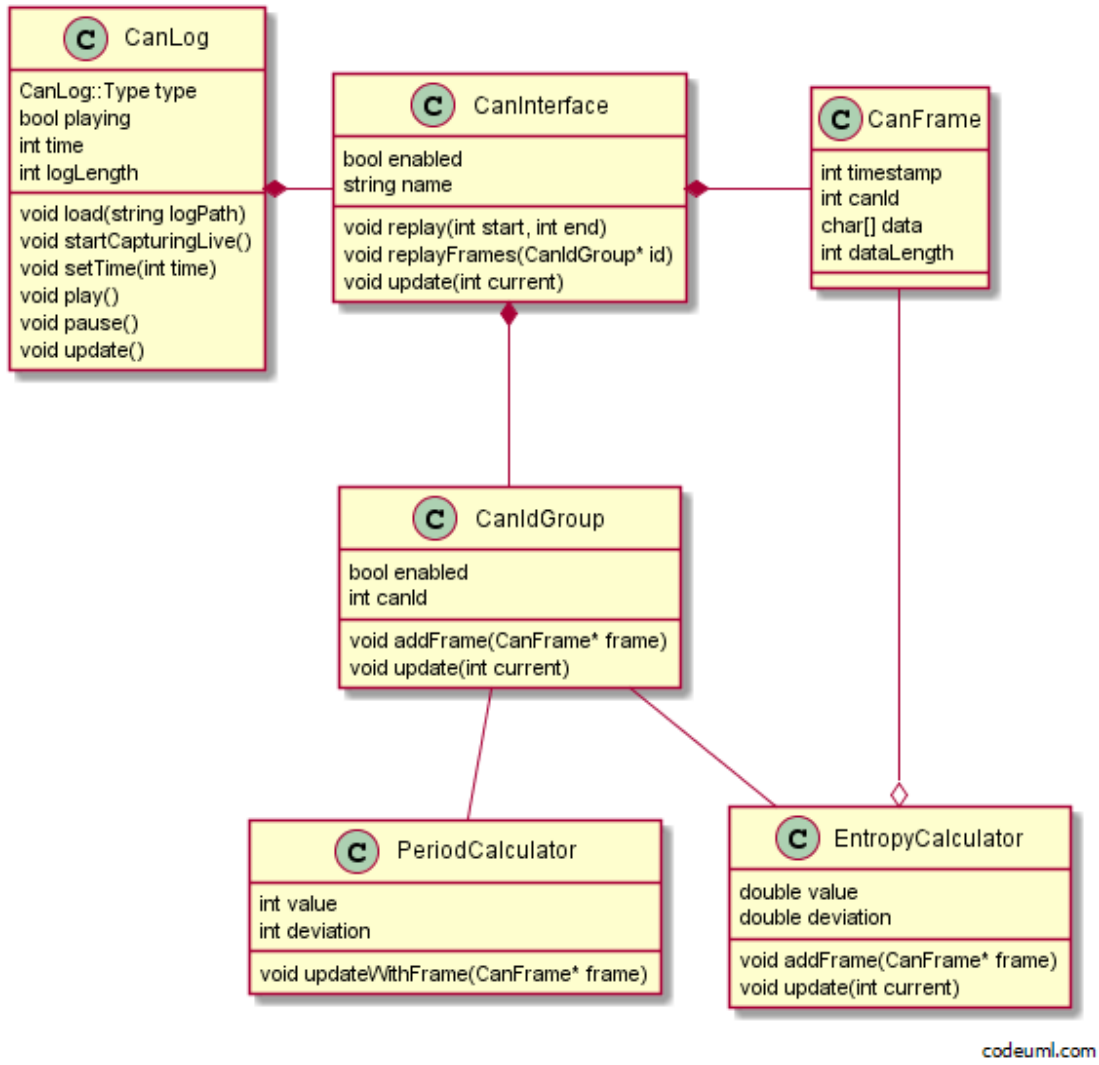
Díky tomu, že QML je vícevláknové, tak výpočty hodnot, které provádí jádro aplikace, mohou probíhat v hlavním vlákně aplikace a neblokují vykreslování uživatelského rozhraní, které je díky tomu velmi plynulé.

Během vývoje jsem komunikoval s vedoucím této práce a techniky z firmy Volkswagen pomocí systému Redmine, což je nástroj pro správu projektu a umožňuje snadnou komunikaci týmu.

4.1 Rozdělení na moduly

V této kapitole popisuji podrobněji implementaci jednotlivých částí aplikace. Všechny názvy tříd a struktur jsou uváděny tak, jak jsou pojmenovány ve zdrojovém kódu, tedy v anglickém jazyce.

4.1.1 Jádro aplikace



Obrázek 4.1. Základní struktura jádra aplikace.

Jak jsem již naznačil v sekci 3.2, jádro aplikace obsahuje několik tříd, které slouží k ovládání analýzy provozu na sběrnici a k jeho vlastní analýze. Struktura tříd jádra aplikace je znázorněna na obrázku 4.1.

Základem je třída CanLog, která umožňuje načíst data z log souboru vytvořeného nástrojem candump nebo spustit a pozastavit analýzu online provozu, případně se posouvat v čase v momentálně probíhající analýze provozu.

Parsování dat log souboru je implementováno funkcí sscanf ze standardní knihovny jazyka C, tato funkce načítá data z textového řetězce podle zadaného formátu. Zkoušel jsem i parsování regulárními výrazy, které jsou implementované v knihovně Qt, ale to bylo až desekrát pomalější, proto jsem zvolil parsování pomocí sscanf. Při parsování

jsou načteny všechny CAN sběrnice, které se v logu vyskytují a pro každou je vytvořena instance třídy CanInterface. Během načítání také probíhá parsování CAN zpráv a vytváření instancí třídy CanFrame pro každou zprávu, které jsou předávány příslušným instancím třídy CanInterface.

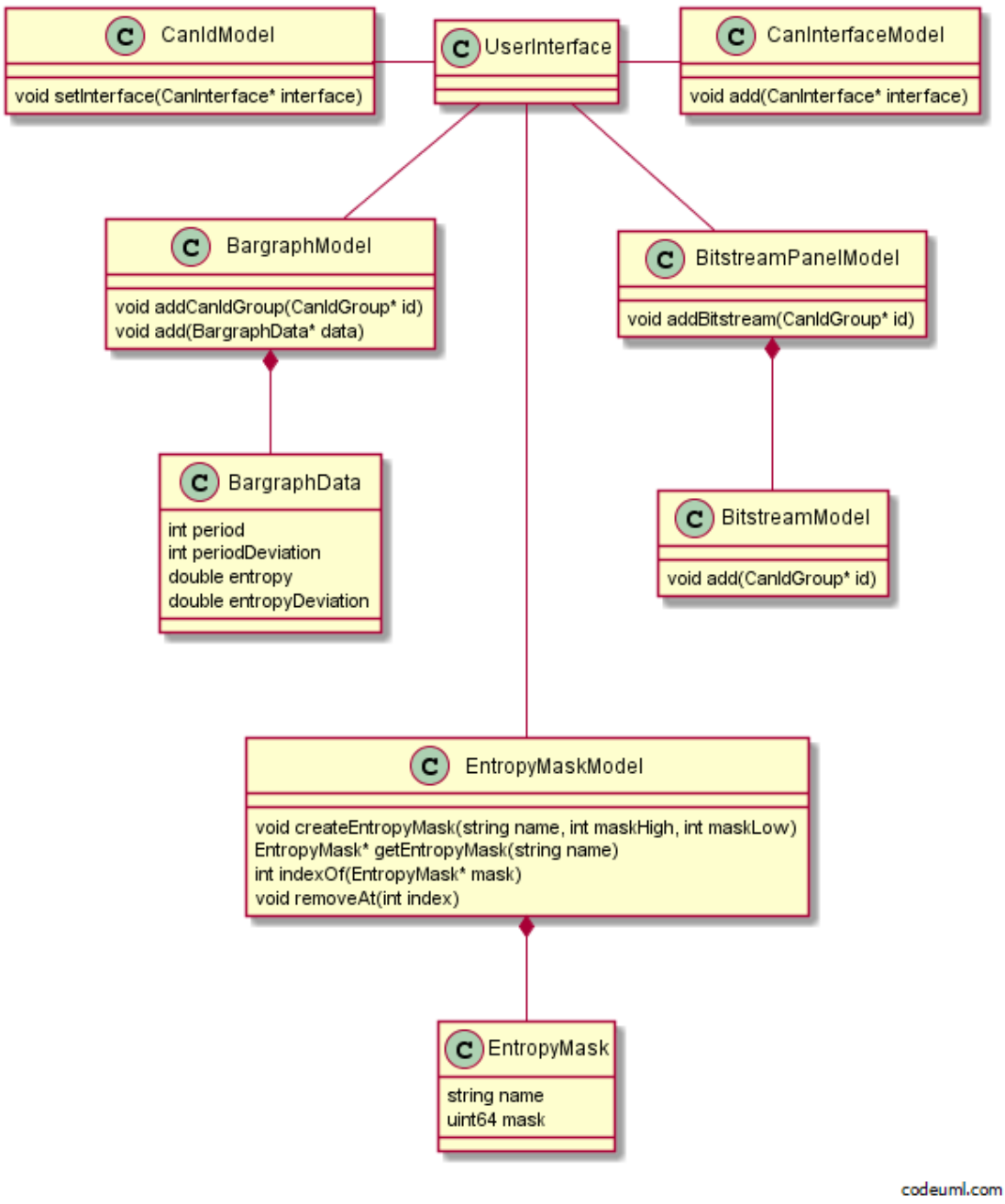
Při spuštění analýzy online provozu se využívá knihovna libudev, která slouží k získávání informací o zařízeních v počítači s operačním systémem Linux. Pomocí knihovny se najdou všechny CAN sběrnice v systému a pro ty jsou následně vytvořeny instance třídy CanInterface. V tomto případě CanLog nenačítá žádné CAN zprávy, ale nastaví CanInterface do módu, kdy je otevřen socket na příslušnou CAN sběrnici a třída CanInterface se poté sama stará o příjem zpráv z konkrétního sběrnice a vytváření instancí třídy CanFrame.

Během přehrávání třída CanLog periodicky aktualizuje jednotlivé instance třídy CanInterface, které v případě analýzy log souboru a analýzy historie online provozu na sběrnici vytvářejí instance třídy CanIdGroup, pokud ještě neexistují, podle toho, jaké CAN ID se v tu chvíli v komunikaci vyskytovaly. Při analýze online provozu jsou instance CanIdGroup vytvářeny ihned, kdy je zpráva s daným CAN ID přijata a CanIdGroup ještě neexistuje. CanInterface při aktualizaci nebo přijetí nové zprávy předá zprávu příslušné instanci CanIdGroup podle identifikátoru přijaté zprávy.

CanIdGroup při přijetí nové zprávy aktualizuje pomocí tříd PeriodCalculator a EntropyCalculator statistiky a vyvolá příslušné signály při změně hodnot.

PeriodCalculator a EntropyCalculator počítají periodu, entropii a odchylku těchto hodnot. K výpočtu využívám vzorec pro výpočet klouzavého průměru s vhodnou konstantou zapomínání. Výpočet odchylky probíhá tak, že se počítají dva průměry s různou konstantou zapomínání a odchylka se rovná rozdílu těchto dvou průměrů.

4.1.2 Uživatelské rozhraní



Obrázek 4.2. Základní struktura C++ části uživatelského rozhraní.

Jak jsem již uvedl vlastní uživatelské rozhraní je napsané v jazyce QML, který je poskytován knihovnou Qt. Uživatelské rozhraní obsahuje definice všech dialogů které se v aplikaci vyskytují, hlavní okno aplikace, dialogy pro nastavení jednotlivých CAN sběrnic a CAN ID, atd. Také obsahuje komponenty, které určují, jak se budou zobrazovat data poskytovaná jednotlivými modely.

Mimo vlastní uživatelské rozhraní jsou součástí tohoto modulu podpůrné třídy napsané v jazyce C++, jejich struktura je znázorněna na obrázku 4.2. Jedná se převážně o modely, které získávají data z jádra aplikace a formátují je a poskytují vlastnímu uži-

vatelskému rozhraní k zobrazení. Modely jsou s jádrem aplikace propojeny systémem signálů a slotů, který umožňuje snadnou a rychlou komunikaci mezi objekty.

Modely, které uživatelské rozhraní využívá jsou BargraphModel, BitstreamModel, BitstreamPanelModel, CanIdModel, a EntropyMaskModel.

BargraphModel vytváří instance třídy BargraphData ke každé instanci třídy CanIdGroup. Třída BargraphData je napojena na signály třídy CanIdGroup a ukládá a formátuje hodnoty, které poskytuje třída CanIdGroup. Při změně hodnot třída BargraphData vyšle signál o změně, ten se dostane do modelu a poté se z modelu dostane k vlastnímu uživatelskému rozhraní, které aktualizuje hodnoty v sloupcových grafech.

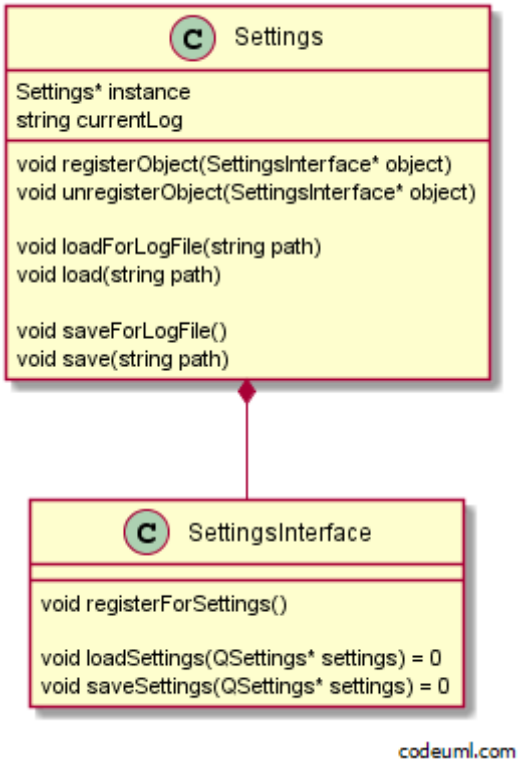
BitstreamModel je model, který je napojen přímo na instance třídy CanIdGroup a slouží k zobrazování zpráv v čase tak, jak se zprávy na sběrnici objevují. Může být napojen na několik instancí třídy CanIdGroup zároveň a tedy zobrazovat i více CAN ID současně.

BitstreamPanelModel slouží ke správě instancí třídy BitstreamModel, kterých se v aplikaci může vyskytovat libovolné množství. Umožňuje jejich přidávání i odebírání.

CanIdModel slouží k zobrazení seznamu CAN ID, které se vyskytují na konkrétní sběrnici, v dialogu pro nastavení CAN sběrnice. Tento model umožňuje zobrazení, nebo skrytí jednotlivých CAN ID z bargraph panelu.

EntropyMaskModel slouží k zobrazení a úpravě masek, které se využívají při výpočtu entropie. Tyto masky vyjadřují, jaké bity zpráv se mají při výpočtu entropie ignorovat.

4.1.3 Modul pro načítání a ukládání nastavení aplikace



Obrázek 4.3. Základní struktura modulu pro ukládání a načítání konfigurace.

Tento modul obsahuje pouze abstraktní třídu SettingsInterface a třídu Settings, vizte obrázek 4.3. Třída Settings je singleton, který je využíván napříč celou aplikací. Umožňuje explicitně načítat a ukládat nastavení aplikace. Také se stará o automatické načí-

tání a ukládání nastavení podle toho, jaká aktuálně probíhá analýza. Rozlišuje konfigurace podle jmen a velikostí log souborů, které uživatel analyzuje. Je tedy možné mít pro každý log soubor jiné nastavení aplikace, jako jsou například masky pro výpočet entropie nebo zobrazené či skryté CAN sběrnice.

Pokud chce mít nějaká třída možnost načítat nebo ukládat nastavení, tak musí dědit abstraktní třídu SettingsInterface. Ta poskytuje čistě virtuální metody loadSettings a saveSettings, které volá třída Settings při načítání, případně ukládání nastavení. Také poskytuje metodu registerForSettings, kterou by měla třída, která chce používat nastavení, volat po své inicializaci. Po zavolání této metody se už nastavení bude automaticky načítat i ukládat, o vše se stará třída Settings.

4.2 Kompilace

V této části popíši postup kompilace a podmínky, které jen nutné splnit aby bylo možné aplikaci zkompilovat. Při uvádění názvů balíčků budu uvádět názvy pro operační systém Debian GNU/Linux, protože je to systém, na kterém měla aplikace primárně fungovat a na kterém probíhalo její testování.

4.2.1 Závislosti

Pro kompilaci aplikace je nutné mít nainstalovaný C++ kompilátor, například GCC. Dále je nutné mít nainstalované knihovny libudev, Qt alespoň verze 5.2, pluginy pro Qt Declarative a hlavičkové soubory knihoven a pluginů. Seznam balíčků, které je nutné nainstalovat:

- build-essential – obsahuje mimo jiné i GCC kompilátor a hlavičkové soubory standardní knihovny jazyka C a C++
- libudev-dev
- qt5-default ($\geq$ 5.2)
- qtdeclarative5-dev ($\geq$ 5.2)
- qtdeclarative5-qtquick2-plugin ($\geq$ 5.2)
- qtdeclarative5-quicklayouts-plugin ($\geq$ 5.2)
- qtdeclarative5-window-plugin ($\geq$ 5.2)
- qtdeclarative5-controls-plugin ($\geq$ 5.2)
- qtdeclarative5-models-plugin ($\geq$ 5.2)

Tyto balíčky je možné v operačním systému Debian Linux nainstalovat například příkazem `apt-get install`.

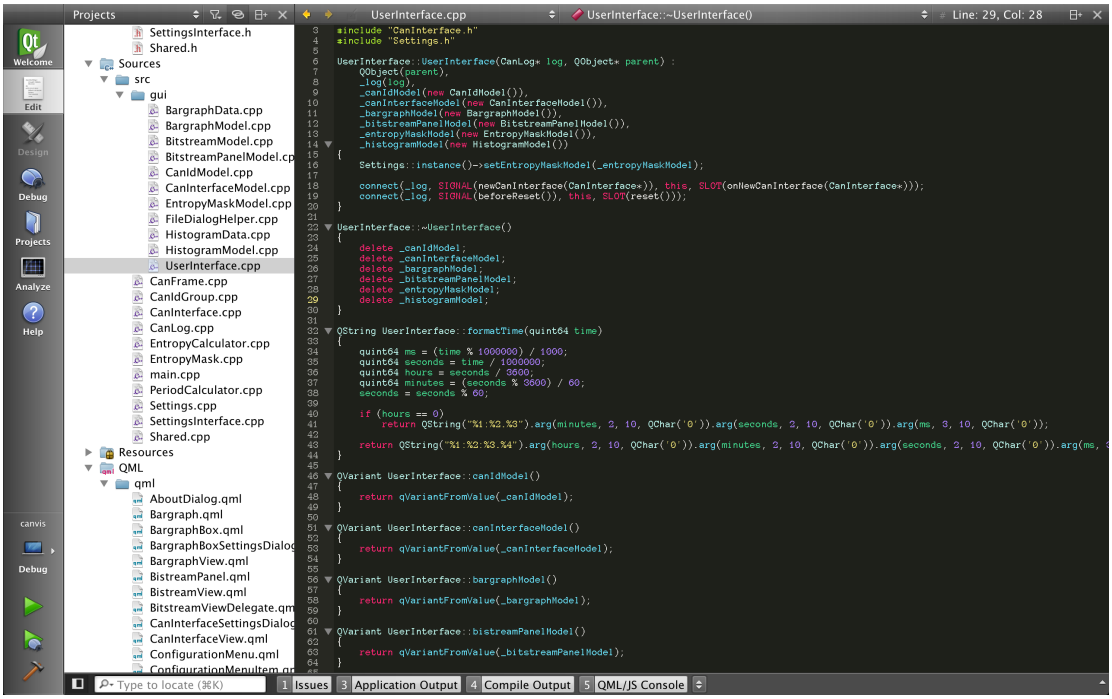
4.2.2 Postup kompilace

Po splnění závislostí, které byly uvedeny v předchozím kroku je dalším krokem kompilace získání zdrojového kódu, například z GIT repozitáře, a spuštění terminálu v adresáři se zdrojovým kódem.

Dále je nutné vygenerovat Makefile, to je možné pomocí příkazu qmake, který jej vygeneruje podle souboru s pojektem. qmake spustíme v adresáři se zdrojovými kódy takto: `qmake CONFIG+=release`. Po úspěšném dokončení příkazu by měl být vygenerován Makefile.

Nyní je možné spustit vlastní kompilaci příkazem `make`, který zkompiluje aplikaci podle vygenerovaného Makefile z předchozího kroku. Po úspěšném ukončení příkazu make je kompilace dokončena a program může být spuštěn vygenerovaným spustitelným souborem.

4.3 QtCreator projekt



Obrázek 4.4. Vývojové prostředí Qt Creator.

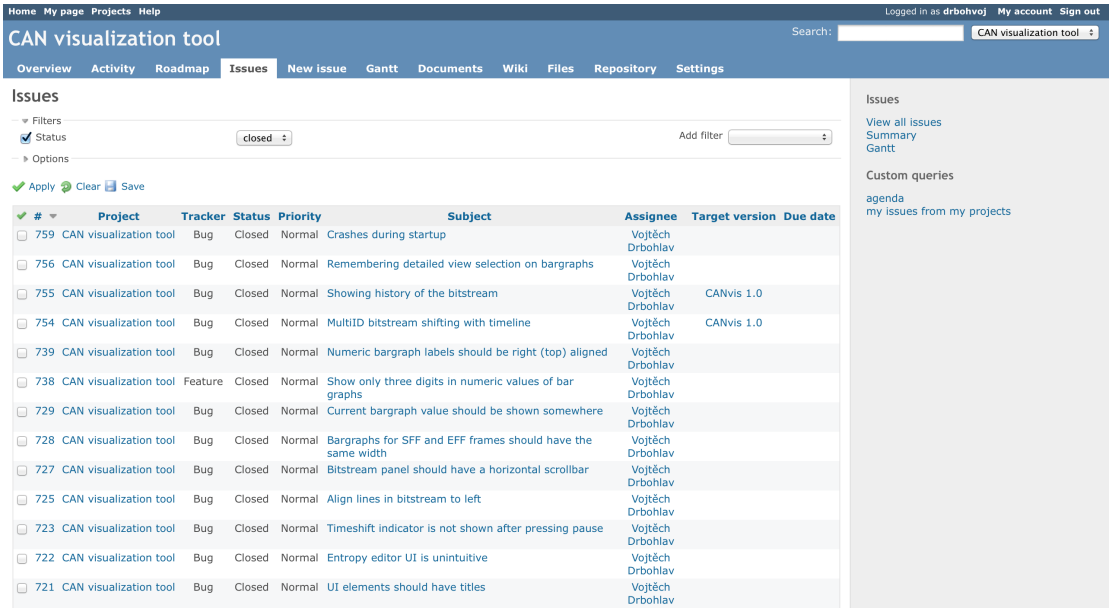
Jak jsem již uvedl pro vývoj aplikace jsem používal IDE Qt Creator. Jedná se o vývojové prostředí primárně určené pro vývoj aplikací v C++ s využitím knihovny Qt.

Po otevření projektu Qt Creator zobrazuje strom projektu s jeho zdrojovými kódy, ukázka s otevřeným projektem je vidět na obrázku 4.4. Vpravo je vidět editor zdrojového kódu. Qt Creator také umožňuje projekt zkompilovat a spustit, během jeho běhu je vidět standardní výstup i standardní chybový výstup aplikace.

IDE také obsahuje uživatelské rozhraní pro GNU Project Debugger (GDB), které se poměrně snadno používá. Dá se pomocí něho ladit vyvíjená aplikace, podporuje i breakpointy a krokování běhu aplikace.

Kapitola 5
Testování

Během vývoje aplikace probíhalo intenzivní testování aplikace. Testovali jsme ji já i můj vedoucí a samozřejmě také technici z firmy Volkswagen. Komunikovali jsme spolu pomocí ticketovacího systému Redmine (viz obrázek 5.1), který celou komunikaci usnadnil. Vyřešené připomínky jsme označovali jako uzavřené, aby komunikace zůstávala přehledná.



Obrázek 5.1. Seznam uzavřených připomínek v ticketovacím systému Redmine.

Připomínky, které během testování aplikace vznikaly se týkaly jak vzhledu uživatelského rozhraní, tak i funkčnosti aplikace. Během vývoje jsme postupně testovali i několik způsobů výpočtu entropie a různé konstanty zapomínání pro výpočet klouzavého průměru, který se využívá pro výpočet hodnot periody a entropie zpráv. Konstanty jsme museli nastavit tak, aby byly dobře vidět anomálie v komunikaci na sběrnici.

Já jsem své testování během vývoje prováděl na operačním systému Gentoo GNU/Linux. Analýzu log souboru z nástroje candump jsem testoval na souboru dodaném vedoucím práce a na souborech, které jsem si ručně vytvořil pomocí nástroje candump z balíčku can-utils. Analýzu online provozu na sběrnici jsem mohl testovat díky jadernému modulu vcan, který umožňuje vytváření lokálních virtuálních CAN sběrnic na počítači. Data pro tyto sběrnice jsem generoval nástrojem cangen z již zmíněného balíčku can-utils. Balíček také obsahuje nástroj canreplay, který umožňuje přehrávat log soubor uložený nástrojem candump na virtuální CAN sběrnici.

Virtuální sběrnice typu vcan jsem přidával do systému nástrojem ip takto: `ip link add type vcan`. Takto přidané sběrnice jsou automaticky pojmenované vcan0, vcan1 atd., před jejich použitím je nutné je aktivovat příkazem `ip link set dev vcan0 up`. Po vytvoření a aktivování sběrnic jsem na nich nástrojem cangen generoval provoz

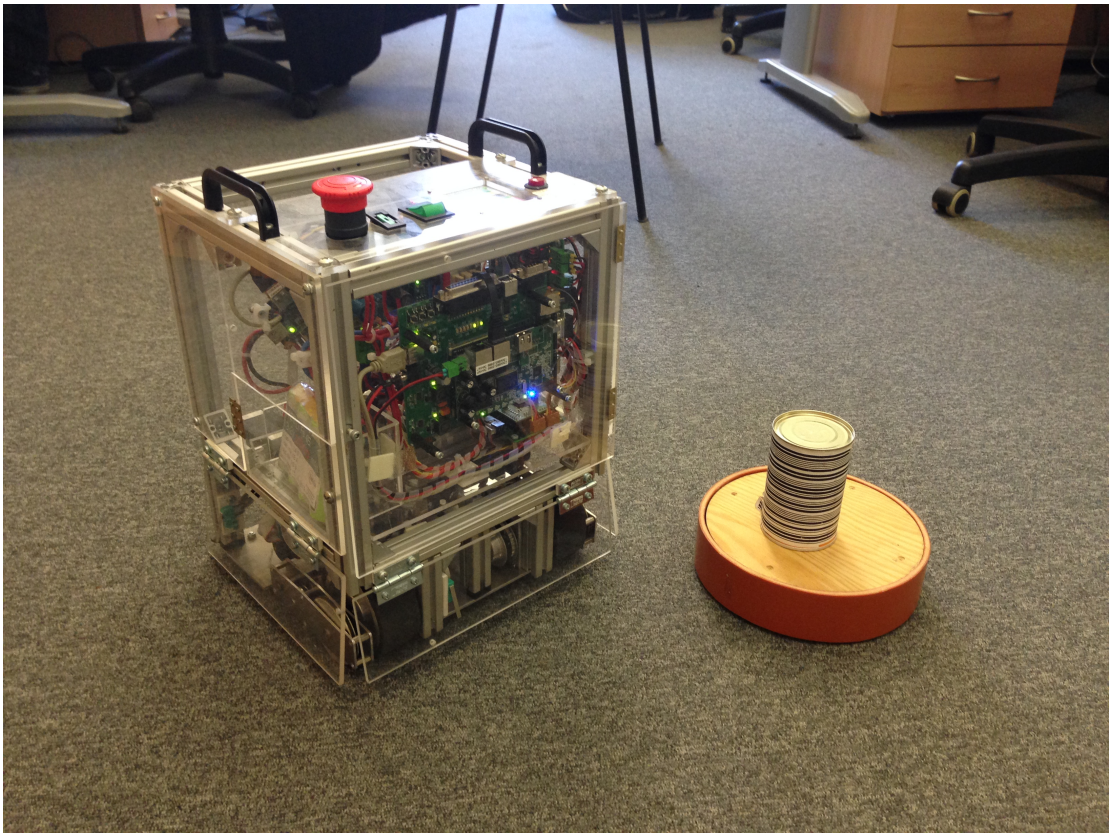
a nechal ho analyzovat vyvíjenou aplikací. Pomocí parametrů nástroje cangen jsem otestoval chování aplikace při nízkých i vysokých periodách i při různých entropiích zpráv.

5.1 Log soubory z firmy Volkswagen

Technici firmy Volkswagen nám poskytli k testování kompletní záznam komunikace všech sběrnic v automobilu zachycený během jízdy na testovacím okruhu. Tento automobil je upraven pro účely testování, takže má všechny sběrnice připravené pro připojení k počítači a během jízdy je možné ovlivňovat komunikaci na jednotlivých sběrnicích, například ji přerušit. Díky tomu je možné simulovat různé útoky a sledovat, jak se mění parametry komunikace.

V tomto log souboru byla zaznamenaná komunikace na šesti sběrnicích, jeho délka byla 1 minutu a 40 vteřin. Log soubor obsahoval velké množství zpráv s periodou od 10 ms až do několika vteřin. Některé zprávy se periodicky opakovaly, jiné byly velmi nepravidelné a měly i různou velikost dat. Entropie zpráv se také velmi lišila. Log obsahoval velké množství zpráv s různými parametry, takže to byl výborný vzorek pro testování.

5.2 Robot



Obrázek 5.2. Demonstrační robot využitý při testování [13].

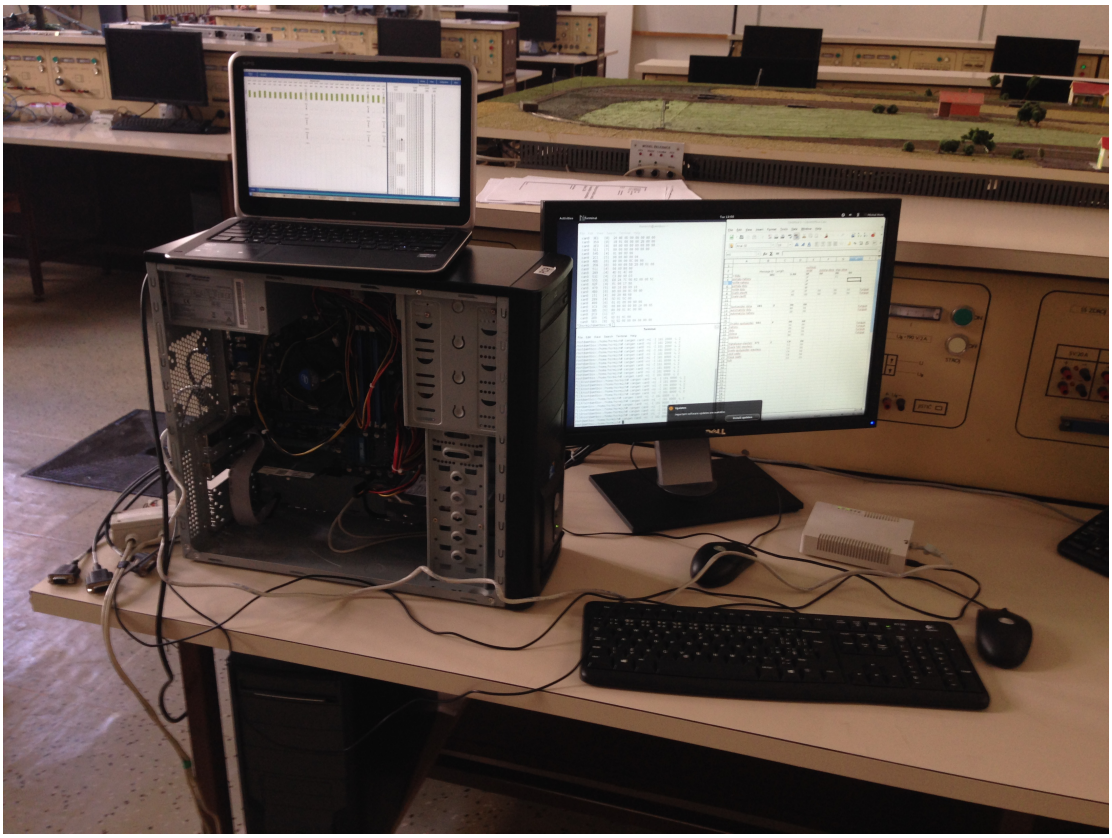
Jako další zdroj testovacího log souboru jsme využili demonstračního robota (viz obrázek 5.2), kterého v rámci své diplomové práce vytvořil Ing. Michal Vokáč [13]. Robot používá několik CAN sběrnic ke komunikaci periferií podobně jako jsou využity

v automobilech. Na sběrnicích probíhá například výměna informací o rychlosti nebo aktuální poloze. Robot je možné ovládat mobilní aplikací z tabletu s operačním systémem Android nebo spustit automatický režim, kdy se robot snaží najít objekt označený čárovým kódem.

Log soubory z demonstračního robota jsme vygenerovali vzdáleně přes Secure Shell (SSH). Vygenerovali jsme je jak v režimu ovládání mobilní aplikací, tak i v automatickém režimu robota, kdy robot hledal překážku. Tyto logy jsou k nahlédnutí na přiloženém CD.

Po načtení log souborů do aplikace bylo vidět, že v robotu je několik sběrnic na kterých probíhá výměna informací. Nejvýraznější byly zprávy s CAN ID obsahující informace o aktuální poloze robota. Při prudkém zrychlení perioda i entropie zpráv stoupaly, protože se poloha aktualizovala častěji. Naopak při zpomalení bylo vidět, jak se odchylky period i entropie dostávají do záporných čísel.

5.3 Škoda Octavia



Obrázek 5.3. Zapojení pro testování na automobilu Škoda Octavia.

Testování analýzy online provozu jsme prováděli na automobilu značky Škoda Octavia, který je v laboratoři na ČVUT FEL na Karlově náměstí. Ten má pro připojení k počítači připravenu jednu sběrnici CAN, na které se posílají informace například o požadavku na stažení okénka nebo informace o tom, zda jsou otevřené nebo zavřené dveře.

Abychom mohli otestovat i dotykové ovládání aplikace, tak jsme sběrnici z automobilu připojili přes CAN-PCI adaptér ke stolnímu počítači, protože notebook s dotykovou obrazovkou nemá kartu pro připojení CAN sběrnice. Na notebooku, kde jsme chtěli

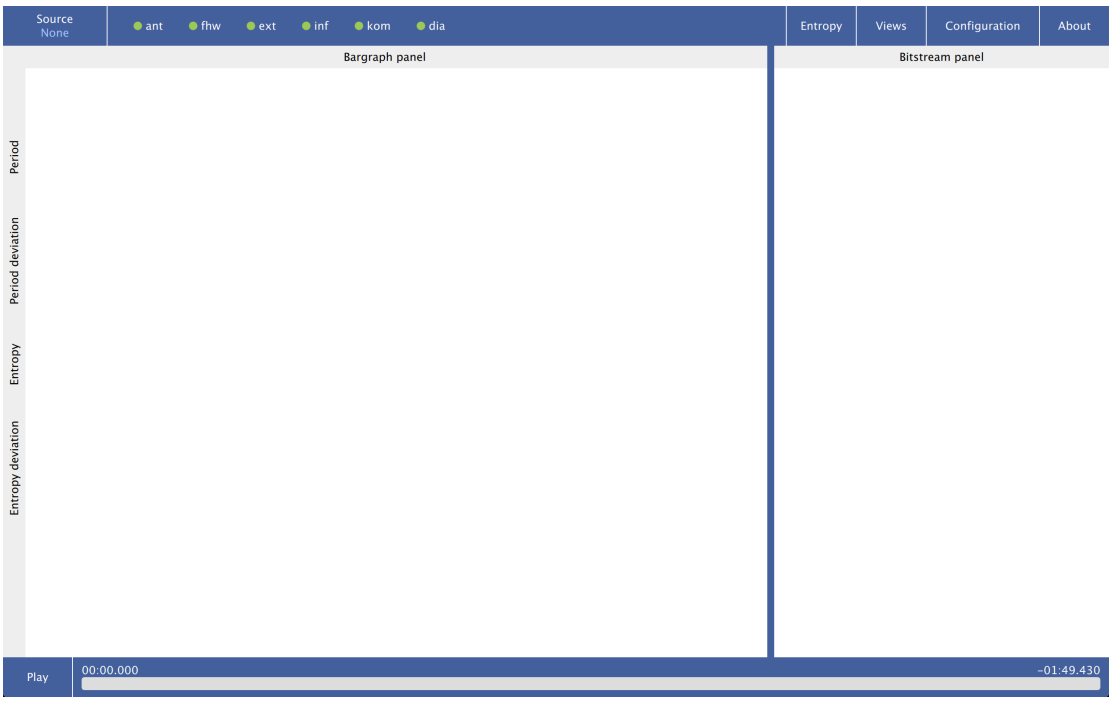
otestovat analýzu komunikace jsme vytvořili virtuální CAN sběrnici pomocí jaderného modulu *vcan*. Přes SSH jsme na stolním počítači spustili *candump* a na notebooku *canreplay*, který data získaná přes *candump* přes SSH přeposílal na místní virtuální CAN sběrnici. Díky tomu se komunikace ze sběrnice dostala přes stolní počítač z automobilu do notebooku s dotykovou obrazovkou, kde běžela naše aplikace. Fotografie zapojení je na obrázku 5.3.

Zkoušeli jsme, zda bude v aplikaci vidět, když se budou zprávy posílané automobilem měnit. Při otevření dveří řidiče se ve zprávách s CAN ID změnil poslední bit zprávy. Když byly dveře řidiče zavřené, tak poslední bit zprávy byl 0, při jejich otevření se změnil na 1. Zprávy o stavu otevření dveří posílá automobil periodicky s periodou 100 ms. Dále jsme zkoušeli, jaké zprávy se na sběrnici objeví při stisku tlačítka na ovládání okénka spolujezdce. Při stisku tlačítka na stažení okénka se na sběrnici periodicky posílaly zprávy o délce 2 byty s různým obsahem, podle toho zda se jednalo o stisk tlačítka pro stažení nebo vytažení okénka. Pokud jsme tlačítko přidrželi zprávy se opakovaly.

Kapitola 6
Uživatelská dokumentace

V této kapitole popíšu, jak je možné spustit analýzu komunikace na sběrnici a představím, jaké má aplikace funkce. Názvy prvků uživatelského rozhraní jsem nechal v anglickém jazyce, ve kterém je napsané celé uživatelské rozhraní aplikace.

6.1 Spuštění



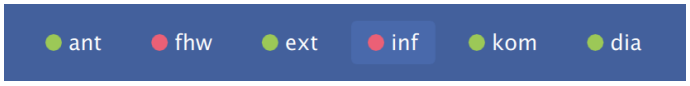
Obrázek 6.1. Hlavní okno aplikace.

Spuštěná aplikace je vidět na obrázku 6.1, lze ji spustit dvěma způsoby. Buď bez parametrů, poté je nutné z menu Source vybrat log soubor, který se bude analyzovat, a spustit jeho analýzu tlačítkem Play. V menu Source je také možné vybrat analýzu online provozu na sběrnici, která automaticky detekuje dostupné CAN sběrnice a spustí analýzu provozu.

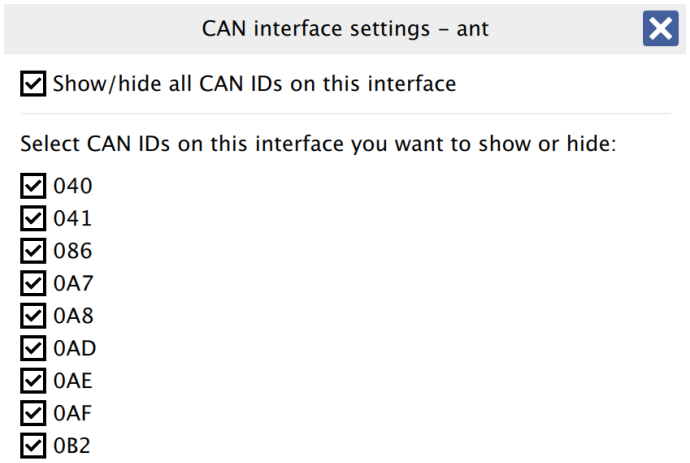
Druhým způsobem spuštění aplikace je spuštění z terminálu, kdy jako první parametr je možné aplikaci předat cestu k log souboru, který bude analyzován.

6.2 Ovládání aplikace

Hlavní okno aplikace je na obrázku 6.1. Po načtení log souboru z disku je možné spustit analýzu komunikace tlačítkem Play ve spodní části aplikace. Pokud uživatel vybral možnost Live, tak se automaticky spustí analýza online komunikace.



Obrázek 6.2. Seznam CAN sběrnic v aplikaci.



Obrázek 6.3. Dialog pro nastavení CAN sběrnice.

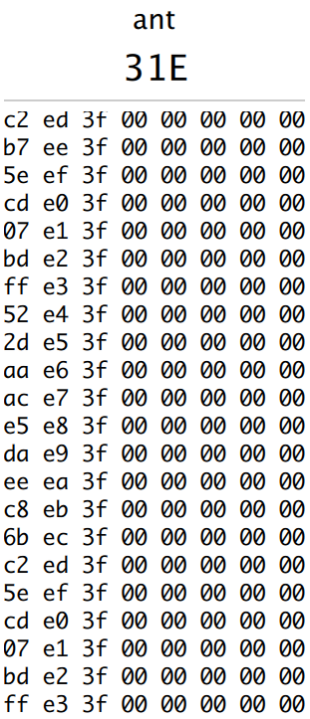
Obrázek 6.4. Bargraph panel obsahující sloupcové grafy pro jednotlivá CAN ID.

V horní části aplikace se načte seznam CAN sběrnic (viz obrázek 6.2), které je možné jednoduchým kliknutím myší povolit nebo zakázat. Dlouhým stiskem tlačítka s názvem CAN sběrnice se otevře její nastavení, viz obrázek 6.3. V nastavení je možné povolit nebo zakázat jednotlivé CAN ID, která se v komunikaci objevují, případně povolit nebo zakázat celou sběrnici.

Po spuštění analýzy logu tlačítkem Play se v části bargraph panel zobrazí sloupcové grafy zobrazující hodnoty period, entropií a jejich odchylek pro všechna povolená CAN

ID, obrázek 6.4. Analýzu je možné pozastavit stejným tlačítkem, kterým byla spuštěna. Při kliknutí na časovou osu vedle tlačítka pro spuštění, nebo pozastavení analýzy, se posune analýza do odpovídajícího času. Při pozastavení online analýzy provozu na sběrnici se budou stále sbírat nová data ze sběrnice, ale nebude probíhat jejich analýza. Při znovuspuštění online analýzy v režimu zpožděné analýzy komunikace se časový posun zruší a přejde se zpět do módu online analýzy provozu.

Pro každé CAN ID na každé CAN sběrnici je zobrazen jeden bargraph box obsahující 4 sloupcové grafy. Každý sloupcový graf znázorňuje periodu nebo entropii zpráv nebo jejich odchylky. CAN ID a sběrnice jsou vypsány v horní části bargraph boxu. Kliknutím na bargraph box se zobrazí nebo skryjí číselné hodnoty sloupcových grafů. Bargraph boxy je možné přetahovat a řadit. Dlouhým stistknutím myši nebo podržením prstu nad bargraph boxem se box aktivuje a potom je možné ho přesouvat.

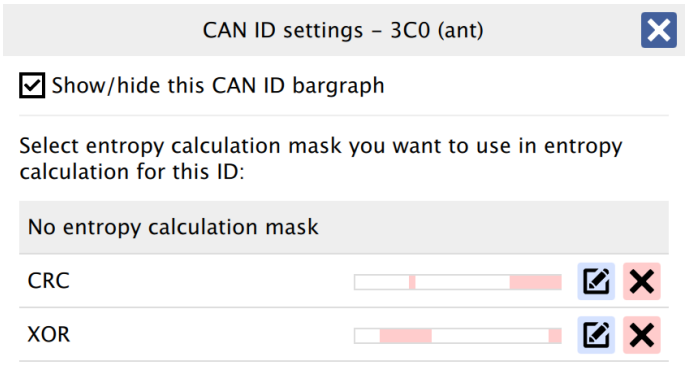


Obrázek 6.5. Bitstream s jedním CAN ID.

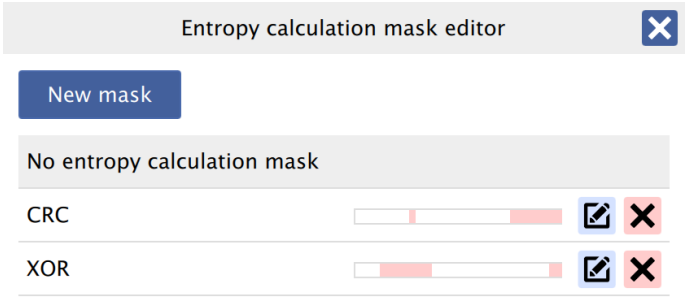
Pokud uživatel přetáhne bargraph box do bitstream panelu na prázdné místo, tak se začnou objevovat zprávy s CAN ID, ke kterému patřil přetažený bargraph box. Takto zobrazované zprávy jsou vidět na obrázku 6.5. Pokud uživatel přetáhne bargraph box na CAN ID, které v bitstream panelu již je, tak se zobrazí zprávy obou nebo více CAN ID spolu v čase tak, jak byly přijímány, obrázek 6.6. Při dlouhém stisknutí některého CAN ID nebo jejich skupiny v bitstream panelu a přetažení mimo něj se daná položka odebere.

ant 31E	ant 117
	75 0c 00 40 68 00 00 ff
da e9 3f 00 00 00 00 00	
	11 0d 00 40 68 00 00 ff
	bd 0e 00 40 68 00 00 ff
	d9 0f 00 40 68 00 00 ff
ee ea 3f 00 00 00 00 00	
	9b 00 00 40 68 00 00 ff
	ff 01 00 40 68 00 00 ff
c8 eb 3f 00 00 00 00 00	
	53 02 00 40 68 00 00 ff
	37 03 00 40 68 00 00 ff
	24 04 00 40 68 00 00 ff
6b ec 3f 00 00 00 00 00	
	40 05 00 40 68 00 00 ff
	ec 06 00 40 68 00 00 ff
c2 ed 3f 00 00 00 00 00	
	88 07 00 40 68 00 00 ff
	ca 08 00 40 68 00 00 ff
	ae 09 00 40 68 00 00 ff
b7 ee 3f 00 00 00 00 00	
	02 0a 00 40 68 00 00 ff
	66 0b 00 40 68 00 00 ff
5e ef 3f 00 00 00 00 00	
	75 0c 00 40 68 00 00 ff
	11 0d 00 40 68 00 00 ff
	bd 0e 00 40 68 00 00 ff
cd e0 3f 00 00 00 00 00	

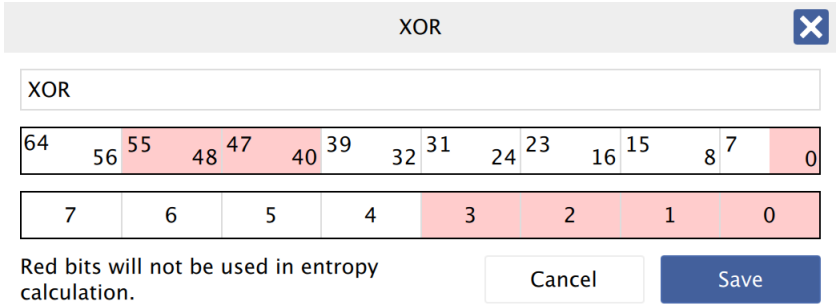
Obrázek 6.6. Bitstream s dvěma CAN ID.



Obrázek 6.7. Dialog pro nastavení CAN ID.



Obrázek 6.8. Dialog pro správu masek pro výpočet entropie.



Obrázek 6.9. Dialog pro úpravu masky pro výpočet entropie.

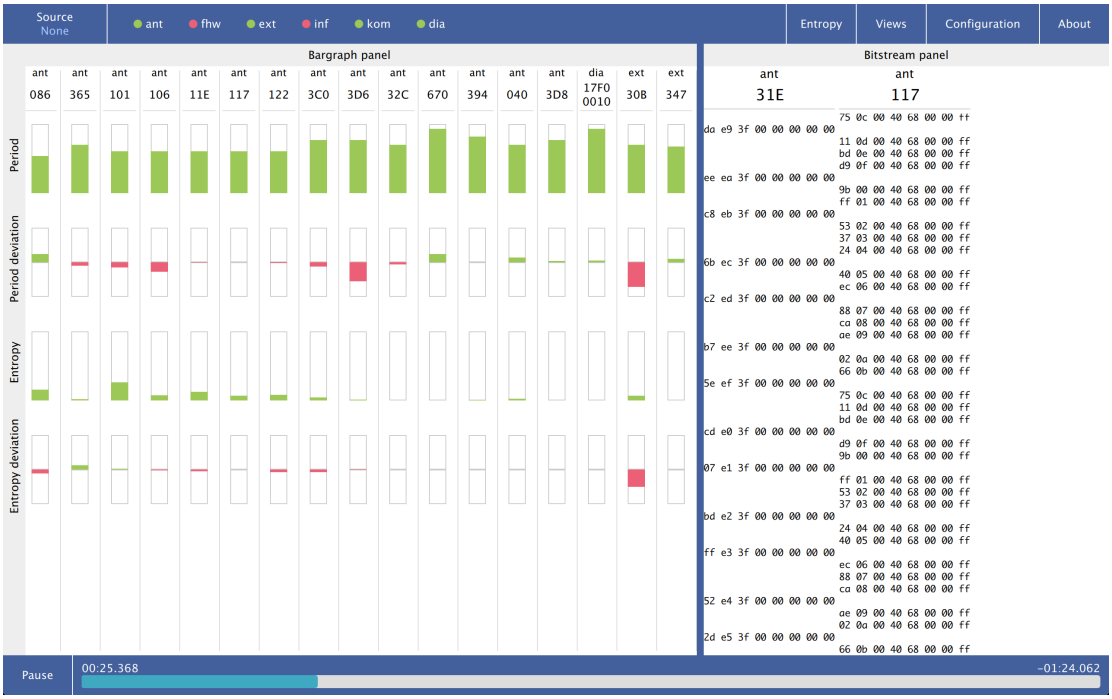
Pokud uživatel dlouze stistkne bargraph box, ale nikam ho nepřesune, tak se otevře nastavení CAN ID, ke kterému bargraph box patří. Dialog nastavení je vidět na obrázku 6.7. V něm je možné CAN ID zakázat, to poté nebude vidět v bargraph panelu. Dále je možné vybrat masku entropie, která se použije pro výpočet entropie zpráv a bude znázorněna i v bitstream panelu ve zprávách s tímto CAN ID.

Masky pro výpočet entropie je možné vytvářet a měnit v editoru masek, který se otevře po kliknutí na tlačítko Entropy masks v horním panelu aplikace. V tomto dialogu, který je na obrázku 6.8, je možné masky přidávat, případně editovat nebo mazat. Při přidávání nebo editování masky se zadává její název a červeně se označují bity, které se mají při výpočtu entropie s danou maskou ignorovat. Dialog pro přidávání a úpravu masek je vidět na obrázku 6.9.

Dalším tlačítkem v horním panelu je Views, které otevře menu, kde je možné skrývat jednotlivé panely aplikace a tlačítko About, které zobrazí informace o verzi a licenci.

Posledním tlačítkem je Configuration, které umožňuje uživateli explicitně načíst nebo uložit jinou konfiguraci, než je automaticky načtená konfigurace podle aktuálně probíhající analýzy.

Vzhled celé aplikace při probíhající analýze komunikace na sběrnici je vidět na obrázku 6.10.



Obrázek 6.10. Celá aplikace při probíhající analýze komunikace.

Kapitola 7
Závěr

Cílem mé bakalářské práce bylo vytvořit grafický nástroj pro vizualizaci vybraných dynamických parametrů v komunikaci na automobilové sběrnici CAN, který bude možné využít při hledání anomálií, když se útočník snaží ovlivnit komunikaci na sběrnici. Nástroj jsem navrhl a vytvořil podle požadavků vedoucího práce a techniků z firmy Volkswagen, pro kterou byla aplikace primárně vyvíjena. Všechny požadavky, které jsou uvedeny na začátku práce jsem splnil a aplikace je funkční. Zpracoval jsem i požadavky, které vnikaly testováním aplikace vedoucím práce a techniky firmy Volkswagen.

Dalšími funkcemi, které by se daly v práci implementovat by mohla být například historie hodnot period a entropie a jejich odchylek, která by byla znázorněna v přehledném grafu. I po odevzdání bakalářské práce budu rád dál zpracovávat dodatečné požadavky, které by mohly vzniknout dalším testováním aplikace. Zdrojový kód aplikace je dokumentovaný pomocí Doxygenu, takže by neměl být problém aby se v něm vyznal někdo další v případě, že by chtěl aplikaci dále vyvíjet.

Na základě požadavku firmy Volkswagen není aplikace veřejná a tudíž soubory se zdrojovým kódem nejsou na přiloženém CD. Aplikace by se totiž dala využít k reverznímu inženýrství automobilové komunikace a k přípravě útoků na automobily.

Literatura

[1] Miller, Charlie a Chris Valasek. *Adventures in Automotive Networks and Control Units*.
http://illmatics.com/car_hacking.pdf.

[2] Checkoway, Stephen, Damon McCoy, Brian Kantor a kol. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In: *USENIX Security Symposium*. USENIX Association Berkeley, 2011. s. 6.
http://www.autosec.org/pubs/cars-usenixsec2011.pdf.

[3] Koscher, Karl, Alexei Czeskis, Franziska Roesner a kol. Experimental Security Analysis of a Modern Automobile. *2010 IEEE Symposium on Security and Privacy*. 2010, s. 447–462. Dostupné na DOI 10.1109/SP.2010.34.
http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5504804.

[4] Scarfone, Karen a Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS). *NIST special publication*. 2007, ročník 800, č. 2007, s. 94.
http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf.

[5] Hartkopp, Oliver a Volkswagen AG. The CAN networking subsystem of the Linux kernel. In: *Proceedings of the 13th iCC*. 2012.
http://www.can-cia.org/fileadmin/cia/files/icc/13/hartkopp.pdf.

[6] Davis, Robert I., Alan Burns, Reinder J. Bril a kol. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*. Springer, 2007, ročník 35, č. 3, s. 239–272. Dostupné na DOI 10.1007/s11241-007-9012-7.
http://link.springer.com/article/10.1007%2Fs11241-007-9012-7.

[7] *Readme file for the Controller Area Network Protocol Family (aka SocketCAN)*. [vid. listopad 2013].
https://www.kernel.org/doc/Documentation/networking/can.txt.

[8] *CAN userspace utilities and tools*. [vid. listopad 2013].
https://gitorious.org/linux-can/can-utils.

[9] *Debian GNU/Linux*. [vid. únor 2014].
http://www.debian.org.

[10] *Qt Creator manual*. [vid. leden 2014].
http://qt-project.org/doc/qtcreator-3.1/index.html.

[11] Blanchette, J. a M. Summerfield. *C++ GUI programming with Qt 4*. Upper Saddle River, NJ: Prentice Hall in association with Trolltech Press, 2008. ISBN 978-0132354165.

[12] *Qt 5.2 documentation*. [vid. leden 2014].
http://qt-project.org/doc/qt-5/index.html.

[13] Vokáč, Michal. *Demonstrační robotická platforma*. České vysoké učení technické v Praze, 2012. Diplomová práce.
http://rtime.felk.cvut.cz/~sojka/students/Dp_2012_vokac_michal.pdf.

Příloha A
Zkratky

CAN	Control Area Network, sériová sběrnice využívaná v automobilové komunikaci
CAN ID	Číselný identifikátor CAN zprávy
IDS	Intrusion Detection System
NRZ	Non-Return-To-Zero kódování
IDE	Integrované vývojové prostředí (Integrated Development Environtment)
QML	deklarativní jazyk pro tvorbu uživatelského rozhraní implementovaný v knihovně Qt

Příloha B
Obsah přiloženého CD

/bp Text bakalářské práce ve formátu PDF.
/log Log soubory z demonstračního robota použité při testování aplikace.