UNIVERZITA PARDUBICE

Dopravní fakulta Jana Pernera

Dispečerský monitorovací systém pro automobilovou amatérskou rally

Pierre Litvák

Diplomová práce

2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 26. 5. 2017 Pierre Litvák

Poděkování

V první řadě bych rád poděkoval rodičům mého bratrance Martina Hrabě, i jemu samotnému, jejichž iniciativa mi pomohla *rozhodnout se pro studium na Univerzitě Pardubice. Poděkování patří rodičům a celé mé rodině za podporu během studia, zaměstnancům KEEZ DFJP a všem spolužákům.*

Zvláštní poděkování patří spolužákům, bez kterých by se projekt tvorby monitorovacího systému amatérské rally neuskutečnil, Ing. Petru Bílkovi a Tomáši Heringovi.

Anotace

Společně s prací Ing. Petra Bílka (Návrh vozidlové jednotky pro monitorovací systém využitelný v podmínkách amatérské automobilové rally) je tato práce součástí projektu na vývoj monitorovacího systému pro amatérskou rally. Práce Ing. Petra Bílka je zaměřena na hardware a tato práce na software.

V práci jsou rozebírány zejména tyto kapitoly:

- systémy, které jsou schopny vykonávat zadanou úlohu a jsou k dispozici na trhu
- analýza způsobů přenosu dat
- popis softwaru (uživatelská a programátorská příručka)

Výstupem této práce je aplikace, ve které může uživatel dohlížet na průběh závodu. Uživatel může sledovat vybrané vozidlo, nahrané do aplikace, na mapových podkladech nebo může pozorovat další informace o vozidle jako např. rychlost, status, průběžný čas…

Klíčová slova

LabVIEW, rally, monitorovací, sledovací, systém, software, SQL, databáze

Title

Dispatching monitoring system for amateur car rally

Annotation

Together with Ing. Petr Bílek's diploma thesis (Design of car unit for the monitoring system applicable for amateur rally) is this thesis a part of project for amateur rally monitoring system development. Ing. Petr Bílek's thesis is focused for hardware and following thesis for software.

Mainly there are discussed following topics in this thesis:

- systems you can already buy to perform the task
- analysis best way to transmit important data
- software documentation (user and programmer manuals)

Output of this thesis is an application where can user supervise over situation of the race. User can watch chosen car, loaded into application, on maps or view other information about the car like speed, status current time.

Keywords

LabVIEW, rally, monitoring, system, software, SQL, database

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pierre Litvák**
Osobní číslo:	**D14002**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Elektrotechnické a elektronické systémy v dopravě**
Název tématu:	**Dispečerský monitorovací systém pro automobilovou amatérskou rallye**
Zadávající katedra:	**Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě**

Zásady pro vypracování:

Cílem práce je navrhnout a vytvořit dispečerskou nadstavbu nad vozidlový monitorovací systém pro amatérskou rallye. Tato nadstavba bude tvořena systémem sběru aktuálních dat z jednotlivých vozidel závodu a jejich grafickou vizualizaci.
Práce musí obsahovat:
1) Analýzu současné situace nabízených analogických komerčních řešení monitorovacích systémů amatérských závodů
2) Analýza vhodných bezdrátových přenosových prostředků využitelný v podmínkách rallye závodu
3) Analýza vhodného řešení grafické vizualizace jízd jednotlivých vozidel a jejich sledování z hlediska amatérské rallye
4) Návrh dílčích jednotek systému dispečerské nástavby monitorovacího systému
5) Realizace dispečerského monitorovacího systému
6) Praktické ověření navrženého systému

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

[1] Kring, Jim;Travis, Jeffrey, Labview for everyone, Pearson Education (US), 2015, ISBN: 9780131856721
[2] Pedro Ponce-Cruz, Fernando D. Ramirez-Figueroa , Intelligent Control Systems with LabVIEW, Springer Verlag, 2009, ISBN: 9781848826830
[3] Database Connectivity Toolkit User Manual, National Instruments, 2008. PDF dostupné na http://www.ni.com/pdf/manuals/371525a.pdf
[4] http://www.w3schools.com/sql/
[5] http://www.mysql.com/

Vedoucí diplomové práce: **Ing. Václav Lenoch, Ph.D.**

Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě

Datum zadání diplomové práce: **30. listopadu 2016**

Termín odevzdání diplomové práce: **26. května 2017**

doc. Ing. Libor Švadlenka, Ph.D.
děkan

L.S.

Ing. Dušan Čermák, Ph.D.
vedoucí katedry

V Pardubicích dne 20. února 2017

Obsah

1. Úvod

Přestože vývojem zabezpečovacích a monitorovacích systémů pro závody automobilů se po světě zabývá mnoho firem, například BOSCH, Motorola, jsou tyto systémy stále velmi drahé, a proto jsou využívány převážně na profesionální úrovni. Stejnou potřebu zajistit bezpečnost na závodech cítí i pořadatelé amatérských soutěží. Dosavadní systém ONI však na dnešní poměry působí trochu těžkopádně a zastarale. Na profesionální systém však v této sféře nejsou peníze. Proto vznikl logický požadavek na vytvoření relativně levného systému, který zjednoduší výměnu informací a zrychlí reakci pořadatelů při vzniku krizových situací, jakými jsou například havárie vozidla. [1]

Cílem je navrhnout systém, který zjednoduší práci organizátorů závodu a omezí stereotypní činnosti pro člověka, jako je například zaznamenávání průjezdů vozidel kolem kontrolních stanovišť, dále pak zvýší kvalitu zjišťování aktuálního stavu vozidla, tzn., zda je vozidlo v pořádku nebo zda došlo k havárii.

Tvorba systému je popsána ve dvou samostatných diplomových pracích. Návrhem hardwarové části a ovládacího softwaru jednotlivých zařízení se zabýval ve své diplomové práci, nesoucí název Návrh vozidlové jednotky pro monitorovací systém využitelný v podmínkách amatérské automobilové rally, Ing. Petr Bílek.

Tato práce se zabývá tvorbou softwaru pro dispečerské stanoviště, kde bude probíhat zpracování zpráv od všech zařízení a jejich přehledná interpretace obsluze.

2. Rozbor situace před zpracováním

2.1 Současný systém

Pozn.: text kapitoly 2.1 je převzat z obdrženého zadání pro nový systém od pořadatelů závodů

- Bezpečnostní ONI systém, který vozy sleduje, je pouze pro dispečink a má výstup dat pro společnost, která provozuje web s výsledky.
- Na startu se hlásí pro všechny radiobody a cíl, že odstartoval vůz č. X v čase hh:mm.
- Jednotlivé radiobody si v papírech odškrtávají průjezd vozu.
- V cíli se zaznamenává čas a hlásí se pro všechny, že vůz č. X projel cílem.
- Pokud nedojede k dalšímu radiobodu, tak se začíná shánět systémem. Příslušný radiobod hlásí vedoucímu RZ, že auto neprojelo. Vedoucí RZ potom zjišťuje dotazem u předchozích radiobodů, zda u nich auto projelo.
- Komunikace pořadatelů se uskutečňuje pronajatými radiostanicemi a komunikace s vozem neexistuje.
- V případě nehody sice ONI systém vyhlásí poplach, ale není možno zjistit, o co jde, jak je to vážné.

2.1.1 Shrnutí zadání pro návrh nového systému

Shrnutí všech podstatných bodů z obdrženého zadání pro vypracování nového systému.

A) Časomíra

- Každé vozidlo musí být systémem jednoznačně identifikováno.
- K měření času bude použit časový údaj z GPS.
- Kontrola předčasného odstartování vozidla (brát v úvahu i reakční dobu člověka).
- Zaznamenávání časů z průjezdů následujících stanovišť: start, průjezdní body (většinou 2-3 na jedné RZ), cíl, příjezd a odjezd ze servisní zóny. Tyto časy bude možné použít jako mezičasy.
- Informace z jednotlivých stanovišť (čas vozidla, případně rychlost vozidla) budou přenášeny k vedoucímu RZ a do dispečinku pomocí radiového přenosu.
- Možnost využití systému pro online sledování vozu v závodě.

B) Bezpečnost

- Vyhodnocování stavu vozidla na základě čidla nárazu. Obsluha tlačítka, pokud je posádka v pořádku, jinak je vyhodnocena nehoda a je vyhlášen poplach.
- Trvalé sledování vozidla – rozlišení polohy přejezd/RZ/servis. Při přejezdu mezi RZ zejména sledovat překročení povolené rychlosti.
- V případě poplachu odeslat varovný signál do vozidel, za zdrojem poplachu – nebezpečí na trati. Stejný signál zaslat i dispečinku a vedoucímu RZ pro dočasné zakázání dalších startů.
- Hlasová komunikace s jednotlivými vozy – propojení do interkomu vozu/hlasitý reproduktor ve voze.
- V případě nehody odeslat GPS polohu pro záchranné složky.

C) Komunikace

- Systém je možné použít i pro komunikaci pořadatelů.
- Systém umožňuje kompletní pokrytí signálem celé rally.
- Dispečink, každá RZ a každá servisní zóna by byly jedním samostatným systémem, které by byly vzájemně propojeny s ostatními systémy pomocí satelitního spojení.
- Možnost komunikace s jednotlivci, skupinami nebo se všemi v systému.

D) Technické zabezpečení

- Systém bude postavený na bázi digitálního radiového přenosu s několika podsystémy vzájemně propojenými pomocí satelitních spojů s jedním dispečerským pracovištěm v dispečinku rally a jedním dohledovým centrem.
- Podsystémy budou tvořeny mobilními základnovými stanicemi, které se budou přesouvat mezi jednotlivými RZ.
- V jednotlivých vozech budou radiostanice s připojením k elektronickému boxu a k interkomu vozu.
- Pořadatelé budou vybaveni přenosnými radiostanicemi pro komunikaci.
- Systém bude soběstačný, nepotřebuje pro svoji činnost žádné externí komerční sítě.

E) Další OPTION (služba pro týmy za poplatek)

- Displej s časomírou ve vozidle zobrazující průběžný čas, jednotlivé mezičasy, celkový čas v cíli, případně odstup od dalších posádek. Mimo RZ by mohl displej zobrazovat další časy z časových kontrol.

2.2 Průzkum komerčně nabízených systémů a modulů

Trh nabízí celou řadu zařízení logujících množství údajů o vozidle v průběhu závodu, časoměřičů, čidel nárazu. Některé firmy nabízejí dokonce i kompletní systémy pro sledování celého závodu. Následuje přehled některých firem a jejich výrobků na trhu se stručným přehledem vlastností.

2.2.1 Produky fy Alfano

Základní jednotka PRO III EVO

- zaznamenává základní veličiny z vozu – otáčky motoru, 2x teplotu, rychlost vozidla
- možnost připojení čidla (IR, magnetické) pro určení počátku měření času s přesností na 1/100 s
- operace s nashromážděnými daty (min, max, nejlepší čas atd.)
- částečná možnost úpravy – nové způsoby zobrazení dat, možnost připojení k GSM
- rozhraní USB
- IP56; obrazovka 75 x 45 mm; velikost 133,5 x 90 x 28 mm; váha 434 g, zdroj napětí – 2x AA baterie
- možnost rozšíření o modul GPS1 nebo GPS4
- aplikace pro Android/iOS/PC, ve které lze také analyzovat získaná data
- cena 295 € (září 2016)

Obr. č. 1 – zařízení PRO III EVO [2]

PRO III EVO s modulem GPS1

- sledování rychlosti a dráhy vozidla pomocí GPS
- rozhraní bluetooth
- možnost vytvoření okruhu pomocí souřadnic GPS + počátek IR/magneticky nebo pouze pomocí GPS souřadnic
- cena 145 € (září 2016)

Obr. č. 2 – modul GPS1 [2]

PRO III EVO s modulem GPS4

- stejné jako modul GPS2
- tříosový akcelerometr
- možnost dalších 2 snímačů teploty
- cena 220 € (září 2016)

Obr. č. 3 – modul GPS4 [2]

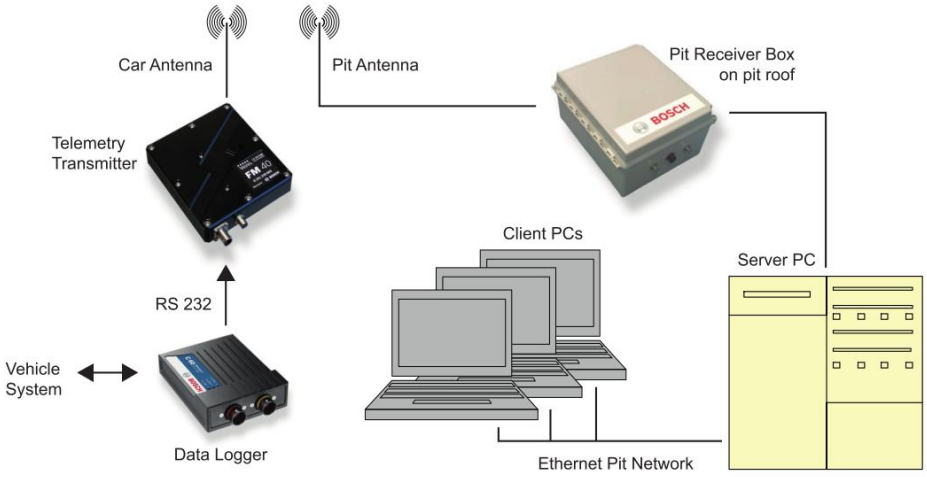
2.2.2 Bosch Online Telemetry Systém

Umožňuje sledování vozidel v reálném čase. Data z vozidla jsou nashromážděna pomocí data loggeru a standardem RS 232 odeslána do vysílače FM40, který z dat vytvoří rámce a přidá nadbytečné informace pro snížení chybovosti při příjmu.

Parametry FM40

- vysílací výkon 1 – 10W
- vysílací frekvence 430 – 470 MHz

V centrále je umístěn přijímací box, který filtruje a zesiluje vstupní signál, klíčuje datové rámce a odesílá je přes rozhraní ethernet do PC.

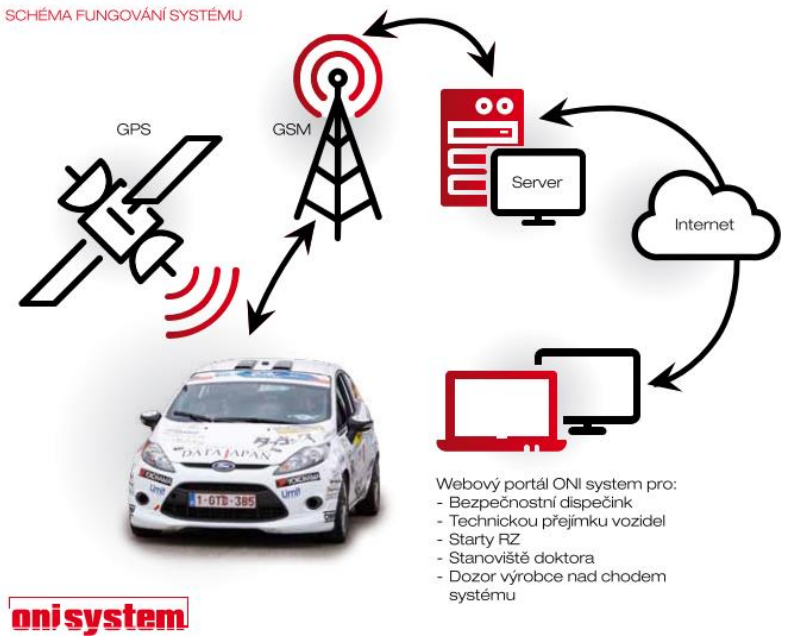


Obr. č. 4 – blokové schéma systému Bosch [3]

Cena celého systému se pohybuje nad 100 000 Kč.

Detailněji se tomuto systému věnuje Ing. Petr Bílek v přidružené diplomové práci (název viz úvod).

2.2.3 ONI system – rally monitoring



Obr. č. 5 – schéma fungování systému ONI [4]

Profesionální systém, který téměř dokonale splňuje celé zadání. Mapové podklady dobře znázorňují pozice jednotlivých skupin zainteresovaných v závodu (závodní vozy, pohotovostní vozy, záchranné složky…)

Celá koncepce systému je téměř identická s navrhovaným systém tohoto projektu. Nutno však podotknout, že tento projekt byl vypracován nezávisle na tomto systému. Informace o existenci systému ONI, specializujícího se pouze na rally, byla získána až po vytvoření celého konceptu. Systém ONI byl celou dobu návrhu systému, který je předmětem této DP, považován pouze za monitorovací systém firemních vozidel.

Cena ONI system – rally monitoring není veřejně dostupná.

2.2.4 Závěr z průzkumu trhu

Ačkoliv by některá zařízení byla vhodná pro použití v novém systému, jejich cena je natolik vysoká, že nejsou cenově dostupná pro účely amatérské rally. Dalším problémem je, že software, který tato zařízení ovládá, je dále neupravitelný a nelze proto zařízení přizpůsobit přímo potřebám amatérské rally.

Z výše uvedených důvodů vyplývá, že bude mnohem efektivnější vytvořit vlastní zařízení s vlastním softwarem. Návrh vlastního zařízení také umožní integrovat všechny podstatné části do jednoho boxu, což usnadní následnou manipulaci se zařízením a jeho instalaci do vozidla.

3. Analýza možností provedení důležitých součástí systému

3.1 Bezdrátové přenosové prostředky

Kapitola pojednává o evoluci návrhů přenosu dat mezi mobilní částí a dispečerským pracovištěm. Zatím poslední verze je rozebrána v podkapitole 3.1.2 Komerční mobilní síť - bod D.

3.1.1 Vlastní nezávislá síť

Původním záměrem bylo vytvořit vlastní nezávislou síť, po které by probíhala veškerá komunikace. Uvažovalo se zejména o APRS, což je systém pro radioamatéry (otevřené frekvenční pásmo), kde spolu komunikuje libovolný počet stanic v reálném čase. Datová propustnost APRS je však poměrně nízká, 1200 nebo 9600 baud, a existuje reálná šance, že pokud budou komunikovat dvě vozidla zároveň, pak se může některá ze zpráv ztratit.

V ideálním případě by tedy bylo nutné vytvořit vlastní systém na koupeném frekvenčním pásmu, kde by byla jistota, že komunikace nebude rušena z jiných zdrojů, např. od radioamatérů. Takový systém by byl však velmi drahý, a proto bylo od vlastní nezávislé sítě úplně upuštěno.

3.1.2 Komerční mobilní síť

Využití mobilní datové sítě GSM se zdá být jako nejvhodnější alternativa z několika důvodů:

- Síť je dostupná téměř po celé ČR, pokrytí zhruba 92% území [5].
- Kapacita a dostupnost sítě se neustále zvyšuje.
- Mobilní data se zlevňují.

A) Přenos přímo TCP/IP, UDP

Cílem bylo vytvořit přímé internetové spojení mezi dispečerským pracovištěm a mobilním zařízením. Provedlo se několik pokusů o navázání této komunikace a na lokální síti mezi dvěma počítači fungovala komunikace velmi dobře. Pokus o komunikaci po internetu se nezdařil. Problém se pravděpodobné skrývá v jednom či více z následujících bodů:

- Příjemce zprávy není vidět za routerem.

- Zpráva jde přes množství zabezpečených kanálů.
- Problém je v používání NAT, který nějakým způsobem mění IP adresu, a tím nevychází checksum ve zprávě UDP/TCP.

I kdyby se nám nakonec přes internet komunikovat podařilo, nebylo by možné zjistit, jakou IP adresu aktuálně každé zařízení má, pokud by každé zařízení nemělo předem pevně přidělenou vlastní IP adresu.

B) Přenos dat pomocí sdílených tabulek Google spreadsheet

Důvody pro zvažování tohoto způsobu řešení byly následující:

- tabulky pracují online,
- je možné do nich přistupovat více uživateli najednou,
- každá změna se okamžitě ukládá,
- poskytují možnosti pro případné rozšíření aplikace,
- byl by použit hotový, spolehlivý a v praxi odzkoušený systém.

Pro nasazení takového způsobu komunikace bylo nutné vyřešit následující problémy:

- Jak přistoupit přímo do tabulky?
- Je možné tuto službu využívat legálně pro naše potřeby?
- Nebude Google omezovat zápis neustálého proudu drobných dat?

Během řešení těchto problémů se vyskytnul nový nápad, který byl velmi podobný systému od Googlu, vytvořit vlastní databázi SQL.

C) Vlastní databáze SQL

Vytvoření vlastní databáze v prostředí MySQL Workbench nepředstavovalo příliš velký posun, jelikož zde vyvstaly podobné problémy jako při snaze komunikovat přímo přes protokoly TCP/IP, případně UDP.

D) SQL databáze na externím serveru

Na externím serveru (sql2.webzdarma.cz) jsou vytvořeny 3 tabulky pro předávání dat. Tabulka s názvem *car* slouží pro předávání dat mezi dispečerským stanovištěm a jednotlivými vozidly. Druhá tabulka, nesoucí název *checkpoints,* slouží k předávání informací mezi

startem, kontrolními stanovišti a cílem. Prozatím poslední tabulka – *caution* obsahuje všechna vozidla, která se nacházejí v okruhu menším než 300 metrů od nehody jiného vozidla.

Ukázka rozvržení DB tabulky *car*

Položka	row	ID_car	speed [km/h]	status	latitude	longitude
Datový typ	unsigned INT(10)	unsigned TINYINT(3)	unsigned TINYINT(3)	unsigned TINYINT(3)	DOUBLE	DOUBLE

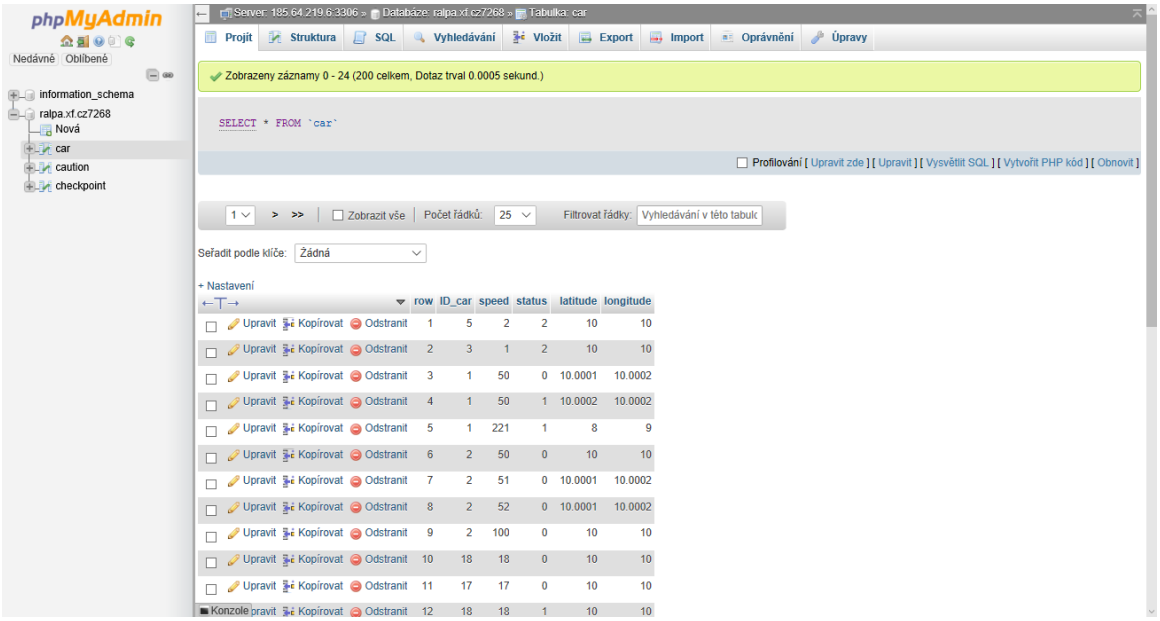
Ukázka rozvržení DB tabulky *checkpoints*

Položka	row	ID_checkpoint	ID_car	time [ms]	status
Datový typ	unsigned INT(10)	unsigned TINYINT(3)	unsigned TINYINT(3)	unsigned INT(10)	unsigned TINYINT(3)

Ukázka rozvržení DB tabulky *caution*

Položka	row	ID_car
Datový typ	unsigned INT(10)	unsigned TINYINT(3)

Pozn.: DATOVYTYP(maximální počet cifer)



Obr. č. 6 – ukázka prostředí na serveru sql2.webzdarma.cz

Ke všem tabulkám se přistupuje přes připravené skripty umístěné přímo na serveru sql2.webzdarma.cz a napsané v jazyce php. Skripty určené pro vkládání mají vytvořený kontrolní kód pro zamezení vkládání zpráv od neoprávněných softwarů/uživatelů. Čtení zpráv není nikterak ošetřeno, kdokoliv tedy zná internetový odkaz na skript, může si přečíst libovolný zápis v databázi.

Přehled skriptů používaných pro komunikaci s DB

Výběr z tabulky checkpoint
Odkaz: ralpa.xf.cz/checkpoint/select.php?row=[row]

- odpověď
  - [row],[ID_checkpoint],[ID_car],[time],[status]
  - [row],!

Vložení do tabulky checkpoint
Odkaz:
ralpa.xf.cz/checkpoint/add.php?a=[checkpoint]&b=[car]&c=[code]&d=[time]&e=[status]

- odpověď
  - ok
  - fool

Výběr řádku z tabulky car
Odkaz: ralpa.xf.cz/car/select.php?row=[row]

- odpověď
  - [row],[ID_car],[speed],[status],[longitude],[latitude]
  - [row],!

Výběr vozidla z tabulky car
Odkaz: ralpa.xf.cz/car/select_car.php?car=[ID_car]

- odpověď
  - [row1],[ID_car],[speed1],[status1],[longitude1],[latitude1]
    [row2],[ID_car],[speed2],[status2],[longitude2],[latitude2]
    …
  - [car],!

Přidání zprávy do tabulky car
Odkaz: ralpa.xf.cz/car/add.php?a=[car]&b=[speed]&c=[code]&d=[status]&e=[long]&f=[lat]

- odpověď
  - ok
  - fool
  - code = (součet délek řetězců všech argumentů) XOR (délka řetězce [speed])
    - řetězcová interpretace všech čísel s plovoucí čárkou je počítána s 6 desetinnými místy

Přidání vozidla do tabulky caution
Odkaz: ralpa.xf.cz/caution/add.php?a=[car]&c=[code]

- odpověď
  - ok
  - fool
  - code = předchozí prvočíslo od [car] * následující prvočíslo od [car] + [car]
    - prvočíslo je bráno matematicky přesně (první je 2),
    - předchozí < [car], následující > [car]
    - pokud není předchozí nalezeno, bere se místo něj následující

Smazání vozidla z tabulky caution
Odkaz: ralpa.xf.cz/caution/delete.php?a=[car]&c=[code]

- odpověď
  - ok
  - fool
  - code = stejný jako pro vkládání

Vyhledání vozidla v tabulce caution
Odkaz: ralpa.xf.cz/caution/find_car.php?car=[car]

- odpověď
  - [car]
  - [car],!

Testování propustnosti

Simulace ukázala, že je možné do externí databáze nahrávat zprávy od 40 jednotek minimálně s periodou 5 s. Každá zpráva byla posílána samostatně, jako při reálném nasazení. Zároveň se ukázalo, že systém bez problému dokáže toto množství zpráv plynule vyčítat, což je umožněno tím, že se z DB vždy stahují všechny dosud nepřečtené zprávy. Pokud by k vyčítání docházelo po jedné zprávě, pak by se pokaždé musela otevírat komunikace s databází a celý proces by nabíral značné zpoždění.

Software pro generování zpráv

Projekt je pojmenován *CarMsgSim.lvproj*, programovaný ve vývojovém prostředí LabVIEW, ve kterém jde v relativně krátkém čase vytvořit požadovanou aplikaci. Architektura kódu je typu QSM – PC, což je architektura, kdy jedna while smyčka vykonává většinu výpočetních operací („consumer“) a minimálně další jedna while smyčka vytváří požadavky na vykonání určitých operací („producer“), typicky obsluhou uživatelského prostředí. Příkazy se předávají přes tzv. fronty, kde se řadí příkazy za sebou podle toho, jak byly generovány, případně jakou měly prioritu.

Po spuštění programu uživatel přidává ID jednotlivých vozidel (celkový počet není omezen). Ihned po přidání se zaznamená hodnota čítače v procesoru a podle této hodnoty se pravidelně kontroluje, zda je aktuální hodnota čítače větší než poslední uložená pro konkrétní ID. Pokud ano, vygeneruje se zpráva, odešle se do databáze a hodnoty zprávy se zapíší do tabulky *lastMsg* v uživatelském rozhraní.

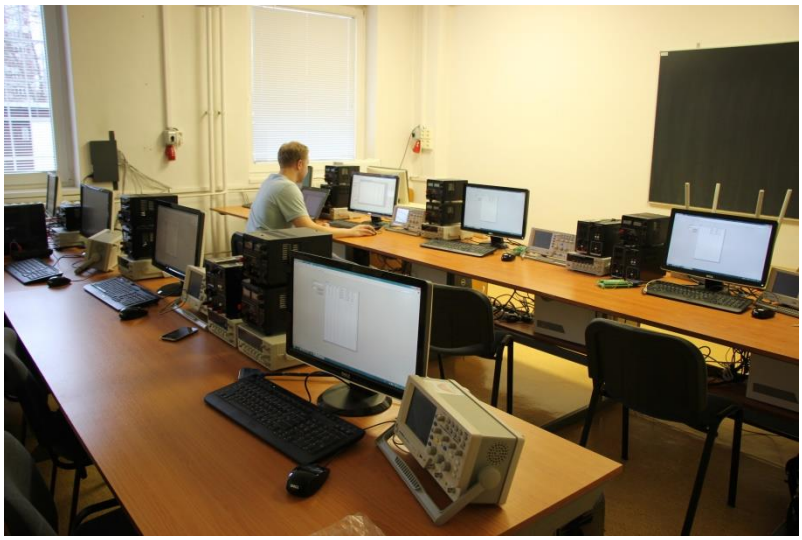
Pro provedení simulace propustnosti bylo vytvořeno náhodné generování hodnot s dodatečným ošetřením pro více realistické hodnoty.

Záznam o provedené simulaci

8x PC
Počet zpráv na 1 PC: 5 vozidel
Perioda odesílání: 5s/zpráva
Datum simulace: 13. 1. 2017
Doba trvání: ~39 minut
Start: 13:42 Konec: 14:21
Pozorované chyby: žádné
Celkem vytvořeno záznamů v DB: 18138
Celkem přijato záznamů z DB: 18138

Tab. č. 1 – zprávy odeslané z osmi PC

Car ID	Počet zpráv	Car ID	Počet zpráv	Car ID	Počet zpráv	Car ID	Počet zpráv
1	446	11	452	21	459	31	463
2	445	12	451	22	458	32	462
3	445	13	451	23	450	33	458
4	445	14	451	24	450	34	458
5	444	15	450	25	449	35	456
6	448	16	455	26	460	36	463
7	448	17	454	27	460	37	462
8	448	18	454	28	452	38	462
9	448	19	454	29	452	39	462
10	447	20	453	30	452	40	461
celkem	4464		4525		4542		4607
	18138						



Obr. č. 7 – fotografie z testování propustnosti [6]

V pozdější fázi vývoje byl software *CarMsgSim* rozšířen o několik dalších možností, aby bylo možné efektivněji testovat software, který je předmětem této práce.

Ve zdokonalené verzi softwaru *CarMsgSim* je přidaná možnost volby mezi náhodným generováním zpráv od vozů a závislým generováním s ohledem na polohu vozidla na trati, kdy se po informaci o projetém startu začne simulovat pohyb vozidla přičítáním konstanty k GPS poloze v každé nové zprávě (vozidlo se pohybuje po přímce). Dále je možné poslat libovolnou vozidlovou zprávu, což je vhodné pro testování mimořádných situací. Poslední přidanou funkcí je posílání zpráv z checkpointů.

Jelikož se jedná o software pro simulaci, není dostatečně robustní vůči neznalé osobě. Například není schopen zabránit uživateli, aby odeslal několik zpráv o tom, že vozidlo projelo startem nebo, že vozidlo projíždí checkpointy ve správném pořadí. Ačkoliv takto fungující program není zcela podle programátorských zásad, není účelem vytvářet dokonalý simulační program, ke kterému se neznalá osoba v reálném nasazení nikdy nedostane a nejsou proto nijak ohrožena data ze závodu.

Program *CarMsgSim* je nahraný na přiloženém disku ve verzi LabVIEW 2011.

3.2 Grafická vizualizace jízd vozidel

Vzhledem k tomu, že bylo upuštěno od návrhu využívat systém APRS, který dokáže zobrazovat jednotlivé radiobody na mapě, a přešlo se na systém s použitím komerční mobilní sítě, bylo nutné vyřešit, zda zobrazovat polohu vozidel přímo na mapách na internetu nebo se budou stahovat mapové podklady a na ně se poté bude poloha vykreslovat.

Nakonec byla zvolena první metoda, tedy přímé zobrazování na internetu. Výhodou takového řešení je, že uživatel nemusí stahovat žádné mapové podklady, ani cokoliv nastavovat. Rovněž po stránce programátorské je tento způsob řešení velmi výhodný.

Pokud je vybráno některé vozidlo ke sledování a zároveň je-li uživatel na záložce *Mapy* – uživatelské prostředí obslužného programu, pak každá příchozí zpráva od sledovaného vozidla vygeneruje nový internetový odkaz na server mapy.cz s aktuální polohou.

Podrobnější popis této funkce je rozebrán dále v kapitole 4.1 po uživatelské stránce a v kapitole 4.2 po programátorské stránce.

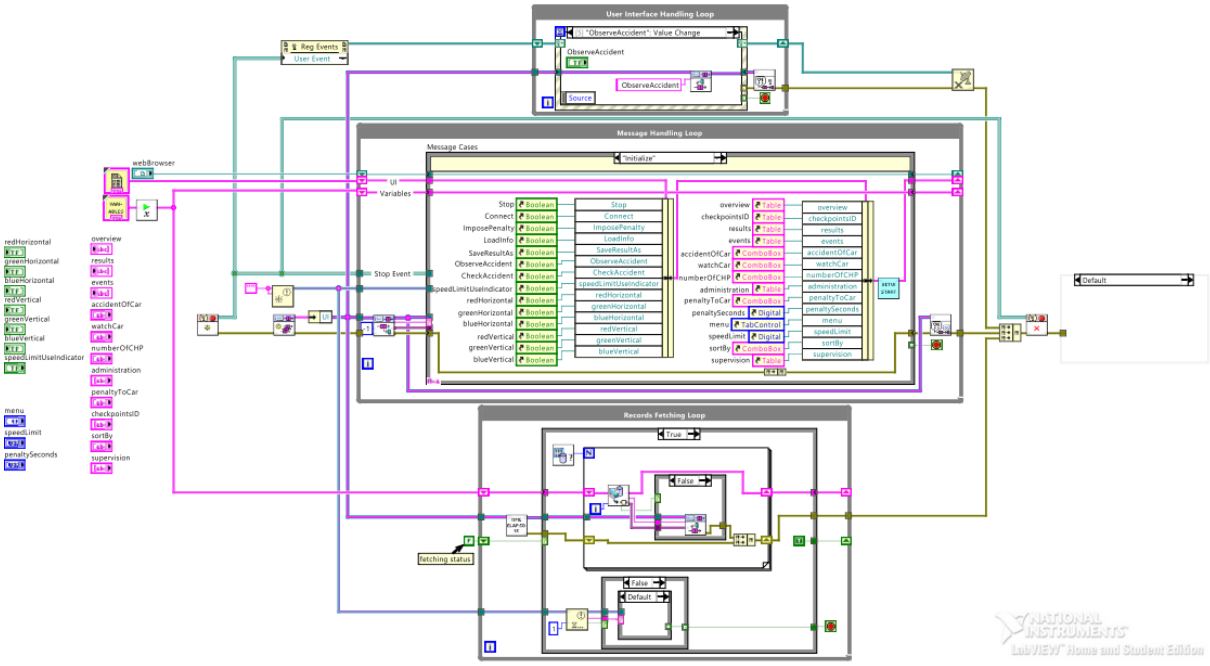
4. Návrh a popis monitorovacího softwaru

Kapitola obsahuje stručný popis celého programu, zjednodušené blokové schéma, dále uživatelskou příručku – popis ovládání programu a jeho chování z hlediska uživatele, programátorskou příručku – rozbor jednotlivých funkcí, části zdrojového kódu, zjednodušené vývojové diagramy.

Vývojový diagram se nachází v příloze A.

4.1 Stručný popis programu

Obslužný program dispečerského stanoviště je psán ve vývojovém prostředí LabVIEW od společnosti National Instruments.



Obr. č. 8 – ukázka architektury kódu

Architektura kódu je opět typu QSM-PC (Querry State Machine – Producer Consumer).

Výsledný program je tvořen dvěma while smyčkami „producer“. Jedna sleduje pokyny z uživatelského prostředí a druhá neustále vyčítá nové zprávy z databáze. While smyčka „consumer“, fungující jako stavový automat, příkazy jednotlivě zpracovává. Po zpracování se vždy ihned vyzvedává další příkaz z fronty. V některých případech je výstupem zpracování příkazu vygenerování dalšího příkazu přímo v „consumer“ smyčce, který se zařadí do fronty.

Consumer obsahuje case strukturu, která má definované následující stavy:

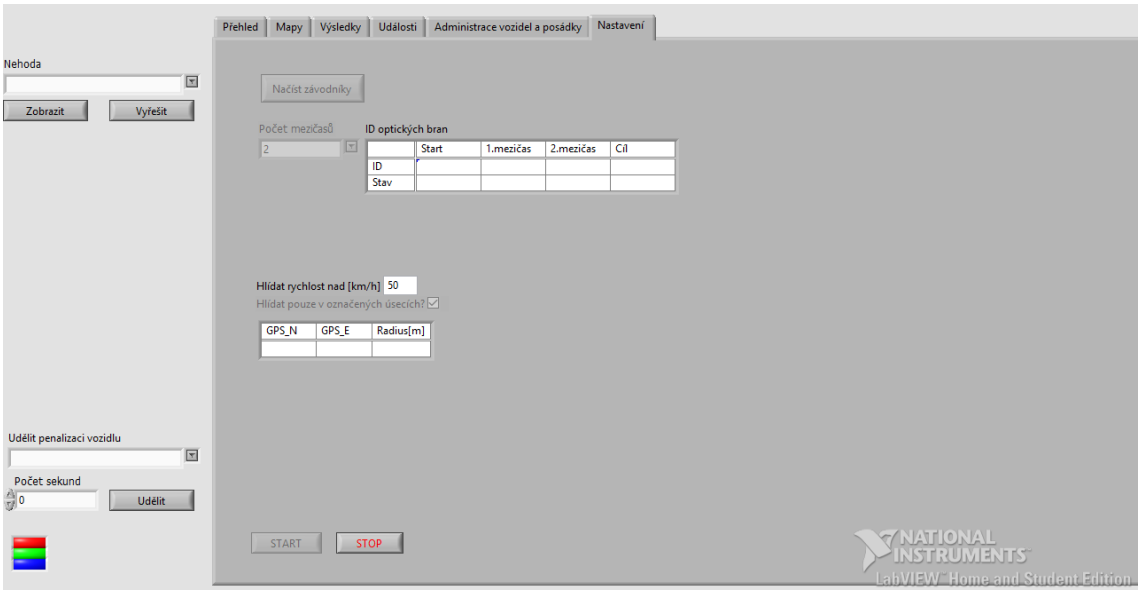
- **Initialize** – předání referencí prvků z uživatelského rozhraní do clusteru s názvem UI_ref, což následně umožňuje přistupovat k hodnotám prvků uživatelského rozhraní přes tento cluster
- **LoadInfo** – načtení informací o závodnících z externího textového souboru ve specifickém formátu
- **SetCHP** – nastavení některých tabulek na základě počtu zadaných kontrolních stanovišť, tzv. checkpointů
- **Connect** – započne se vyčítání zpráv z databáze, program se přepne z úvodních nastavení do pracovního režimu
- **ImposePenalty** – dle zadaných hodnot se udělí penalizace
- **penalty** – změní stav ovládacích prvků pro udílení penalizace
- **ObserveAccident** – otevře mapu se zobrazeným místem zvolené nehody
- **CheckAccident** – po provedení této funkce se nehoda považuje za vyřešenou a zmizí ze seznamu nehod k zobrazení
- **car** – zpracování zprávy od jednotek z vozidel
- **checkpoint** – zpracování zprávy od jednotek checkpointů
- **time** – čítač sekund orientačního času, času od poslední zprávy, času k udělení penalizace
- **currentTime** – výpis aktuálních orientačních časů do tabulky *Přehled*
- **SaveResultAs** – uloží výsledky závodu do externího dokumentu
- **ChangeVisibility** – mění stav indikátoru „zamrznutí“ programu
- **AdaptSupervisionTable** – mění velikost tabulky obsahující úseky, kde se bude kontrolovat rychlost vozidla
- **Error** – vyhodnocení chyb
- **ConfirmQuit** – potvrzení ukončení programu
- **Exit** – ukončení programu
- **Default** – stav pro podchycení nedefinovaných stavů

4.2 Uživatelská příručka

Podrobný popis obsluhy programu uživatelem.

Barevný čtverec v levém dolním rohu slouží jako indikátor stavu programu. Normálně mění svůj stav každou sekundu. Pokud dojde ke zpomalení nebo výraznějšímu narušení periody otáčení čtverce, pak je program zahlcen příchozími zprávami, případně jinými operacemi. Zastavení čtverce značí tzv. zamrznutí programu, a ten pravděpodobně bude nutné restartovat. Všechny zprávy jsou však uložené na serveru. Program by měl být schopný je všechny zpracovat během několika minut (v závislosti na počtu záznamů), nicméně po opětovném spuštění pravděpodobně nebudou sedět orientační časy, časy událostí atp. Je také možné, že po opětovném spuštění systém chybně vyhodnotí polohu vozidla na trati/mimo trať a dojde k vypsání události o překročení rychlosti. Měření skutečného času průjezdu závodních vozů tratí zůstane správné.

4.2.1 Nastavení



Obr. č. 9 – ukázka uživatelského rozhraní (karta Nastavení)

Ihned po startu programu jsou povoleny úpravy pouze na záložkách *Administrace vozidel a posádky* a *Nastavení*. V prvně jmenované lze editovat tabulku s informacemi o závodnících. Pokud je připravený dokument s údaji o jednotlivých účastnících, pak lze na záložce *Nastavení* tento dokument načíst přes tlačítko *Načíst závodníky*. Důležité je zachovat správný formát tohoto dokumentu. Pořadí jednotlivých sloupců musí odpovídat tabulce na záložce *Administrace vozidel a posádky*:

Číslo zařízení	Startovní číslo	Označení vozu	Posádka	Plánovaný start

Dokument je možné přehledně vytvořit například v tabulkovém procesoru Microsoft Excel. Následně je nutné jej uložit jako soubor s koncovkou *.txt a jako oddělovač zvolit tabulátor. Teprve až soubor s příponou *.txt je možné načíst. Po vybrání dokumentu *.txt se objeví dotaz, zda dokument obsahuje záhlaví, což znamená, že na prvním řádku se nachází cokoliv jiného než informace o závodnících.

Jelikož software je schopen zajistit kontrolu rychlosti vozidel mimo závodní trať, a to s využitím údajů z GPS, je vhodné provést nastavení limitu rychlosti a úseky, kde kontrola bude probíhat. V základním stavu se rychlost kontroluje všude mimo trať s výjimkou krátkého časového úseku za cílem, kde vozidla většinou dobrzďují z vysoké rychlosti. Čas potřebný pro zpomalení se vypočte jako:

$$t = \frac{v_{max} - v_{limit}}{a}$$

kde: v_{max} … 56 m/s (uvažovaná maximální rychlost vozidla – cca 200 km/h)
v_{limit} … zvolená limitní rychlost
a … 2 m/s^2 (uvažované konstantní zpomalení vozidla)

Zaškrtnutím políčka *Kontrolovat jen na vybraných úsecích* se zobrazí tabulka se třemi sloupci: GPS_N (severní šířka), GPS_E (východní délka) a okruh působnosti (poloměr) v metrech. Pozice GPS postačuje zapsat na 6 desetinných míst. Pokud všechny 3 údaje budou čísla, tabulka se automaticky zvětší o jeden řádek až do limitu 10 řádků, poté se zobrazí vertikální posuvník. Je tedy možné vytvořit libovolný počet kontrolních lokalit. Mimo tyto lokality se rychlost nekontroluje. Každá zpráva od vozidla, která nese informaci o překročení rychlosti o více než 20 % z udaného limitu, se zapíše do tabulky událostí. Vysoká tolerance (20 %) je vhodná, jednak z hlediska ne zcela přesného systému GPS, kde skutečně ujetá dráha za jednotku času se může lišit od dráhy změřené pomocí GPS. Změřenou rychlost je proto nutné brát pouze orientačně a vždy s určitou tolerancí. Zároveň jistě není účelem vzbuzovat v závodnících pocit, že jsou extrémně sledováni a budou napomínáni nebo trestáni za překročení rychlosti o jednotky km/h.

Kontrolu rychlosti mimo závodní trať lze deaktivovat zadáním rychlostního limitu 0 km/h.

Na záložce *Nastavení* je také nutné zvolit počet checkpointů (start a cíl nejsou počítány jako checkpointy). Podle počtu checkpointů se nastaví tabulka ID optických bran. Do té je potřeba správně zadat ID zařízení na jednotlivých stanovištích.

Všechny popsané úkony na záložce *Nastavení* lze provádět v libovolném pořadí.

Po ukončení procesu nastavování se kliknutím na tlačítko *Připojit* zkontroluje, zda se v tabulce *Administrace vozidel a posádky* nenacházejí duplicity startovních čísel nebo zařízení, případně zda není zcela prázdná. Pokud je vše v pořádku, uzamkne se většina možností nastavení. Pouze v případě, že byla zvolena možnost sledovat rychlost pouze ve vyznačených oblastech, lze tuto tabulku i nadále editovat v průběhu chodu programu. Zároveň se do tabulky ID optických bran přidá řádek zobrazující status jednotlivých zařízení. Stavy jsou následující:

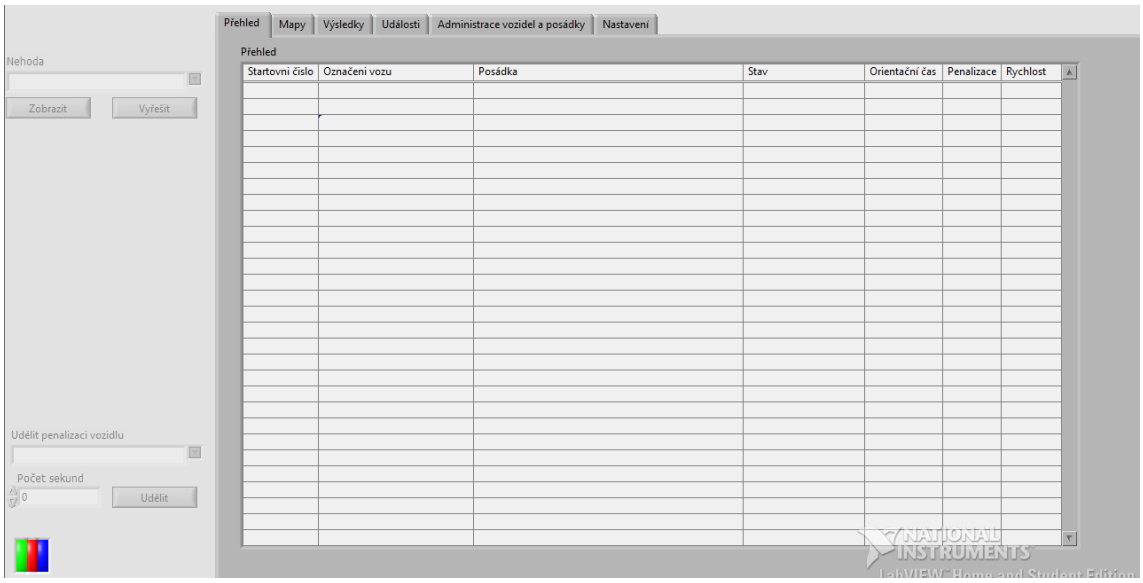
- beze zprávy – zařízení ještě nezačalo komunikovat. Možné příčiny:
    - čeká se na synchronizaci času s GPS,
    - ID bylo zadáno nesprávně a zařízení není v provozu.
- ok – všechny důležité funkce pracují správně
- !detekce – do zařízení se nevrací vysílaný optický signál:
    - paprsku v cestě stojí nějaká překážka
    - paprsek je odchýlen ze správného směru

  V tomto stavu zařízení není schopné rozpoznat průjezd vozidla.
- bez GPS – zařízení ztratilo signál GPS, čas se dopočítává přímo v zařízení, není potřeba dělat žádná opatření, nicméně při dlouhodobé ztrátě signálu dojde k nepřesnosti v čase

Tlačítko STOP, vytvoří požadavek na ukončení programu. Po potvrzení ve vyskakovacím okně se program ukončí.

4.2.2 Popis prvků a jejich obsluhy po spuštění čtení z databáze

A) Přehled



Obr. č. 10 – ukázka uživatelského rozhraní (karta Přehled)

Do této tabulky jsou nakopírovány podstatné údaje o závodnících přesně ve stejném pořadí jako v tabulce *Administrace vozidel a posádky*, zároveň je zde možné sledovat:

informace o stavu vozidla

- ok
- nehoda – vždy, když je u nějakého vozu zjištěna zpráva o nehodě, program uživatele automaticky přepne na záložku *Přehled*. Nehoda se zapisuje do tabulky událostí a do rozbalovacího menu v levém horním rohu. Popis rozbalovacího menu se nachází v bodě E) této kapitoly. Zároveň jsou automaticky varována vozidla, která jsou před místem nehody méně než 300 metrů. Na zařízení ve vozidle se rozsvítí světlo indikující výstrahu. Jakmile se vozidla od místa nehody vzdálí, výstraha se ruší.
- nehoda,bezGPS – provede se stejný proces jako u popisu nehody, ale za místo nehody se považuje poslední známá poloha vozidla, jelikož aktuální poloha není známá

U stavu vozidla se zároveň zobrazuje, jak je údaj aktuální: <10s, <60s, >60.

Orientační čas

Tento čas je skutečně pouze orientační a může se lišit od přesného času i o několik sekund. Čas se spouští až ve chvíli, kdy program stáhne z databáze informaci o tom, že vozidlo projelo startem. Po projetí cílem se čas zastaví. Orientační čas se zobrazuje ve formátu hh:mm:ss.

Penalizace

Pokud je udělena penalizace nějakému vozidlu, pak se v tomto sloupci na dobu 10 s zobrazí +XXs. Až po odpočítání 10 s se údaj o penalizaci smaže a připočítá se k orientačnímu času.

Rychlost

Zobrazuje poslední známou rychlost pomocí GPS. Nemusí se jednat o zcela přesný údaj, jak již bylo zmíněno výše v kapitole 4.2.1 Nastavení.

B) Mapy

Záložka obsahuje dva prvky: rozbalovací menu *Sledovat vozidlo* a okno prohlížeče. Vybráním vozidla z menu se začne automaticky zobrazovat každá nová poloha vozidla, která se liší od předchozí polohy o více než 20 metrů. Mapy se odkazují přímo na server mapy.cz pomocí vygenerovaného odkazu ze souřadnic GPS.

C) Výsledky

Tabulka opět přebírá základní údaje o závodnících z tabulky *Administrace vozidel a posádky*. Počet sloupců tabulky se odvíjí od nastaveného počtu průjezdních bodů (checkpointů) na kartě *Nastavení*.

Tato tabulka zobrazuje přesné časy v každém úseku s přesností na milisekundy. Vozidla je možné řadit podle zadaného filtru. Uplatnit lze vždy pouze jeden filtr (výsledný čas, 1. mezičas, 2. mezičas atd.). Výběr filtru se provádí v rozbalovacím menu *Řadit podle*. V případě, že vozidlo ještě neprojelo úsekem, podle kterého se řadí (čas, podle kterého se řadí, neexistuje), pak je umístěno do spodní části tabulky.

Sloupec *penalizace* ukazuje celkový počet sekund penalizace, která už je započítaná ve výsledném čase.

Poslední obslužný prvek na kartě *Výsledky* je tlačítko *Uložit jako…* Po jeho stisknutí se zobrazí nabídka, kam dokument uložit. Stejně jako při načítání informací o závodnících, i dokument s výsledky má příponu *.txt a jako oddělovač je zvolen tabulátor. Oproti samotné tabulce *Výsledky* jsou do dokumentu přidány některé další sloupečky, které s ostatními poskytují kompletní informaci o vozidle a jeho časech. Nezávisle na zvoleném filtru řazení se výsledky v dokumentu řadí vždy podle výsledného cílového času, kde na prvním řádku je nejrychlejší čas. Na dalších řádcích je pouze ztráta na prvního s přesností na milisekundy.

D) Události

Každý řádek nese jednu událost a zobrazuje systémový čas zjištění události, stručný popis události, startovní číslo a posádku vozidla.

Jsou vytvořeny 3 typy událostí:

- **Nehoda** – číselně označená v pořadí, v jakém byla zaznamenána
- **Penalizace** – vygeneruje se vždy při udělení jakékoliv penalizace (včetně udělení 0 s)
- **Překročení rychlosti v kontrolovaném úseku** – jako událost o překročení rychlosti se vypíše každá zpráva od vozidla, která nese informaci o překročení rychlosti v kontrolovaném úseku o více než 20 %.

E) Vnější ovládací prvky

Rozbalovací menu nehod a k němu přidružená tlačítka

Každá přijatá zpráva o nehodě se zobrazí v rozbalovacím menu *Nehoda*. Pokud se jedná o zprávu o nehodě od stejného vozidla, pak se GPS poloha musí lišit minimálně o 20 metrů, aby byla vytvořena nová událost o nehodě. Na místo nehody lze nahlížet pomocí tlačítka *Zobrazit*, což odkáže na místo nehody podle souřadnic GPS na server mapy.cz. V případě nehody, kde není validní údaj z GPS, se zobrazí jako místo nehody poslední známá poloha vozidla.

Tlačítkem *Vyřešit* se nehoda odstraní z rozbalovacího menu a zůstane po ní záznam pouze v tabulce událostí.

Jak již bylo zmíněno v bodě A), jsou varována vozidla v blízkosti 300 metrů od nehody. V případě odstranění nehody tlačítkem *Vyřešit* se zajistí zrušení výstrahy všem vozidlům v okolí nehody. Výstraha se zruší rovněž i za situace, že se vozidlo, které hlásilo nehodu,

začne pohybovat (například když vozidlo narazí, ale je schopné normálně pokračovat v závodě). Nehoda se však nepovažuje za vyřešenou, a proto je třeba, aby jí byla věnovaná pozornost. Je tedy nutné takovou nehodu vyřešit stejně jako všechny ostatní.

Nehoda bez validního GPS signálu má pravidlo o zrušení výstrahy při započetí pohybu zablokované. Za místo nehody je totiž v tomto případě považována poslední známá poloha. V případě opětovného načtení signálu GPS po nehodě by mohl být rozdíl poloh větší než 20 metrů, systém by vyhodnotil pohyb vozidla a výstraha by byla zrušena, přestože nehoda možná stále trvá.

Penalizace

Pomocí rozbalovacího menu se provede výběr vozidla, kterému má být udělena penalizace. V tu chvíli se odblokuje tlačítko *Udělit*. Ačkoliv je tlačítko *Udělit* možné stisknout, je třeba pamatovat na fakt, že v kolonce *Počet sekund penalizace* je stále nula.

Stisknutím tlačítka *Udělit* se vykonají následující procesy:

- znemožní se opětovné stisknutí tlačítka udělit penalizaci (je nutné provést opětovný výběr vozidla),
- kolonka počet sekund se smaže na nulu,
- rozbalovací menu se vrátí do základního stavu,
- penalizace se připočítá k celkové penalizaci v tabulce *Výsledky,*
- v tabulce *Přehled* se zobrazí na dobu 10 s, po odpočítání této doby se připočítá k orientačnímu času,
- vytvoří se událost popisující udělení penalizace.

4.3 Programátorská příručka

Podrobný popis použitých proměnných, funkcí a částí kódu. U popisu každé funkce je přiložen malý obrázek (přímo z prostředí LabVIEW), který dobře zachycuje vstupy a výstupy dané funkce.

4.3.1 Přehled proměnných

Výčet všech proměnných použitých v programu.

A) UI_ref (cluster of 28 elements)

Connect (boolean refnum) – reference na tlačítko na kartě *Nastavení*. Po jeho obsloužení dojde k aktivaci komunikace s databází, program přejde z inicializačního stavu na provozní.

ImposePenalty (boolean refnum) – reference na tlačítko v levém postranním panelu. Obsluhou se udělí penalizace nastavenému vozidlu.

LoadInfo (boolean refnum) – reference na tlačítko na kartě *Nastavení*. Obsluhou vyskočí dotaz na adresu k souboru obsahujícímu údaje o posádkách.

ObserveAccident (boolean refnum) – reference na tlačítko v levém postranním panelu. Zobrazí zvolenou nehodu na mapě.

overview (table refnum) – reference na tabulku na kartě *Přehled*. Zachycuje aktuálně obdržené informace o vozidlech.

results (table refnum) – reference na tabulku na kartě *Výsledky*. Zobrazuje jednotlivé mezičasy a výsledný čas posádek.

watchCar (combo box refnum) – reference na rozbalovací menu na kartě *Mapy*. Volbou vozidla z menu započne sledování vozidla na mapě.

numberOfCHP (combo box refnum) – reference na rozbalovací menu na kartě *Nastavení*. Výběrem se zvolí počet mezičasů.

administration (table refnum) – reference na tabulku na kartě *Administrace vozidel a posádky*. Výchozí tabulka pro ostatní tabulky (*overview*, *results*, *events*).

penaltyToCar (combo box refnum) – reference na rozbalovací menu v levém postranním sloupci. Obsahuje vozidla, kterým lze udělit penalizaci.

penaltySeconds (digital numeric refnum) – reference na číselný prvek v levém postranním sloupci. Zapisuje se do něj počet sekund penalizace.

CheckAccident (boolean refnum) – reference na tlačítko v levém postranním sloupci. Obsluhou se programově vyřeší zvolená nehoda, tzn., zruší se výstraha vozům.

accidentOfCar (combo box refnum) – reference na rozbalovací menu v levém postranním sloupci. Obsahuje všechny neobsloužené nehody.

menu (tab control refnum) – reference na záložkové menu.

events (table refnum) – reference na tabulku na kartě *Události*. Obsahuje události týkající se vozidel - nehody, překročení rychlosti, udělené penalizace.

checkpointsID – (table refnum) – reference na tabulku na kartě *Nastavení*. Slouží pro přiřazení ID zařízení k checkpointům. Zároveň zobrazuje status checkpointů.

sortBy – (combo box refnum) – reference na rozbalovací menu na kartě *Výsledky*. Volbou možnosti se vytvoří příkaz pro seřazení tabulky *Výsledky* dle vybrané možnosti.

speedLimitUseIndicator (boolean refnum) – reference na zaškrtávací políčko na kartě *Nastavení*. Svojí hodnotou určuje, zda bude kontrola rychlosti prováděna všude mimo závodní trať nebo pouze v zadaných úsecích.

speedLimit (digital numeric refnum) – reference na číselný prvek na kartě *Nastavení*. Hodnota určuje rychlostní limit pro kontrolu. 0 má význam deaktivace kontroly systému monitorujícího rychlost.

redHorizontal, **greenHorizontal**, **blueHorizontal**, **redVertical**, **greenVertical**, **blueVertical** (boolean refnum) – reference na indikátory v levém postranním panelu. Nemají žádný vstup ovlivňující jejich hodnotu. Je ovlivňována pouze jejich viditelnost. Střídavě se přepíná viditelnost horizontálních a vertikálních indikátorů.

supervison (table refnum) – reference na tabulku na kartě *Nastavení*. V základním stavu skrytá. Zviditelňuje ji prvek *speedLimitUseIndicator*. Slouží pro zapsání míst, kde bude probíhat kontrola rychlosti vozidel.

Stop (boolean refnum) – reference na tlačítko na kartě *Nastavení*. Obsluhou se vytvoří vyskakovací okno. Jeho potvrzením se program ukončí.

SaveResultAs (boolean refnum) – reference na tlačítko na kartě *Výsledky*. Obsluhou se vytvoří okno s dotazem na cestu k uložení souboru s výsledky.

B) Variables (cluster of 7 elements)

tableList (1D array of strings) – v poli jsou definované názvy tabulek v databázi, ze kterých se za chodu programu neustále čte.

lastRec (1D array of unsigned long) – číselná hodnota řádku v DB, který se bude aktuálně číst.

CarMsgContent (cluster of 5 elements) – obsahuje prvky odpovídající formátu zprávy od vozidel.

carID (unsigned long) – identifikační číslo zařízení ve vozidle

velocity (unsigned long) – rychlost vozidla v km/h měřená pomocí GPS

status (unsigned long) – číselné vyjádření stavu vozidla (každý bit má svůj význam):

0…ok

1…vozidlo obdrželo informaci o tom, že se nachází v blízkosti nehody

2…vysoké přetížení, možná nehoda

4…tlačítko „crash button“ bylo stisknuto

8…GPS není validní

16, 32, 64, 128…rezerva

GPS_N (single – 6 digits precision) – souřadnice severní šířky

GPS_E (single – 6 digits precision) – souřadnice východní délky

ChpMsgContent (cluster of 4 elements) – obsahuje prvky odpovídající formátu zprávy od checkpointů.

checkpointID (unsigned long) – identifikační číslo checkpointu

carID (unsigned long) – identifikační číslo projíždějícího vozidla

time (unsigned long) – aktuální čas z GPS při průjezdu vozidla v milisekundách

status (unsigned long) – číselné vyjádření stavu zařízení checkpointu (každý bit má svůj význam):

1…detekován IR kód

2…detekováno přerušení světelného paprsku

4…ztráta odrazu světelného paprsku (světelný paprsek je odchýlen od polarizační odrazky nebo je trvale zastíněn)

8…ztráta signálu GPS – neprobíhá synchronizace času (zpráva o ztrátě nebo zisku signálu GPS není samostatně odesílána)

16…ochranná doba proti zahlcení – byla-li protnuta brána bez IR kódu po dobu 100 ms od první registrace protnutí, nebude další takový průjezd zpracován (konec ochranné doby není odesílán jako samostatná zpráva)

32, 64, 128…rezervováno

Pozn.: Ukázkový průjezd má hodnotu 3 – detekován IR kód + přerušení světelného paprsku.

eventCounter (unsigned long) – čítač událostí

carData (1D array of clusters) – každý element pole obsahuje informace o jednom vozidle

CarData (cluster of 5 elements)

carID (unsigned long) – identifikační číslo zařízení ve vozidle

trackPosition (unsigned long) – číselné vyjádření, za jakým checkpointem se vozidlo nachází (např.: 0 … vozidlo je před startem, 1 … vozidlo projelo startem)

LastCarMsg (cluster of 4 elements) – obsahuje poslední obdrženou zprávu z vozidla

velocity (unsigned long)

state (unsigned long)

GPS_N (single – 6 digits precision)

GPS_E (single – 6 digits precision)

events (1D array of clusters) – pole obsahuje informace o událostech

eventRecord (cluster of 9 elements)

velocity (unsigned long)

state (unsigned long)

GPS_N (single – 6 digits precision)

GPS_E (single - 6 digits precision)

description (string) – jednoduchý popis události

systemTime (time stamp) – systémový čas v počítači při zjištění události

checked? (boolean) – příznak, zda uživatel potvrdil, že událost viděl, a není třeba ji dále zobrazovat

EventID (cluster of 2 elements) – identifikační čísla události

eventNr (unsigned long) – pořadí vzniku události

eventType (unsigned long) – typ události

accident – nehoda

penalty – udělená penalizace

speedLimit – překročení povolené rychlosti

withoutGPS (boolean) – příznak, zda událost vznikla, když vozidlo nemělo validní signál z GPS

Time (cluster of 15 elements)

Checkpoints (1D array of unsigned long) – aktuální časy z GPS v milisekundách při průjezdech kolem jednotlivých checkpointů

lastMsg (unsigned long) – čas od poslední přijaté zprávy v ms počítaný z uživatelského PC

currentT (unsigned long) – průběžný čas v ms počítaný v PC v tabulce *Přehled* se zobrazuje ve sloupci orientační čas

finishT (unsigned long) – čas z GPS v ms získaný při průjezdu vozidla kolem cíle

startT (unsigned long) – čas z GPS v ms získaný při průjezdu vozidla kolem startu

penalty (unsigned long) – obsahuje součet všech udělených penalizací (v milisekundách)

timeToImposePen (unsigned long) – čas, který zbývá do připsání penalizace k průběžnému času v sekundách (zpoždění 10 s, pro vyšší přehlednost prováděných úkonů)

decTimeToPen (boolean) – příznak, zda se bude odčítat (-1s) čas do udělení penalizace

incLastMsg (boolean) – příznak, zda se bude přičítat (+1000ms) čas od poslední obdržené zprávy

incCurrenT (boolean) – příznak, zda se bude přičítat (+1000ms) průběžný/orientační čas

RelevantTimes (cluster of 3 elements) – stavy určující, zda časy, obsažené v proměnných se shodným pojmenováním, jsou platné

startT (boolean)

finistT (boolean)

checkpoints (1D array of boolean)

servisT (unsigned int) – v programu nepoužito, připraveno pro případný požadavek na kontrolu času stráveného v servisní zóně

incServisT (boolean) – v programu nepoužito

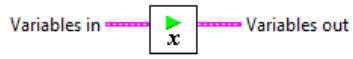
decelerationT – obsahuje čas potřebný na zpomalení za cílem

decDecelerationT – příznak, zda se odečítá čas potřebný na zpomalení vozidla za cílem (sleduje se rychlost vozidla mimo závodní trať, a proto je nutné vozidlu umožnit zpomalit za cílem, aniž by se zobrazila událost o porušení rychlosti)

primeNumbers (1D array od unsigned long) – pole obsahuje všechna prvočísla menší než 500, prvočísla jsou použita pro generování bezpečnostního kódu pro vkládání dat do DB

4.3.2 Přehled funkcí

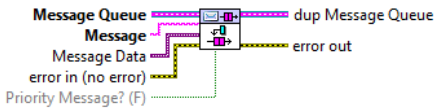
InitSupportVar.vi



Funkce inicializuje cluster proměnných *SupportVar.ctl*. Nulují se počítadla (*lastRecord*, *eventCounter*), generují se prvočísla nižší než 500 (nutné k přístupu do databáze – tabulka *caution*).

A) User Interface Handling Loop

Enqueue Message (single).vi (přejato z příkladu v LabVIEW)



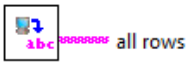
Funkce zajišťuje přidání jakékoliv zprávy (vstup *Message Data*) do fronty s libovolným příznakem typu string (vstup *Message*).

B) Message Handling Loop

SetUIStart.vi

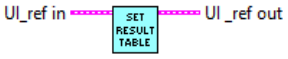
Funkce přistupuje k prvkům uživatelského rozhraní a nastavuje jejich vlastnosti (viditelnost, editovatelnost, velikost atd.) po startu programu.

LoadData.vi



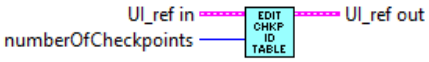
Funkce umožňuje načtení informací o závodnících do 2D pole řetězců *administration* z externího souboru se známým formátem a s příponou *.txt.

SetResultTable.vi



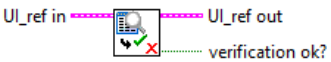
Funkce upravuje velikost tabulky *results* na kartě *výsledky* na základě zvoleného počtu mezičasů.

EditCheckpointIDTable.vi



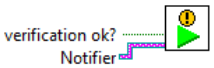
Funkce upravuje velikost a záhlaví tabulky *checkpointsID* na kartě *Nastavení* na základě zadaného zvoleného počtu mezičasů.

VerifyAdminTable.vi



Funkce ověří, že v 2D poli *administration* nejsou duplicitní startovní čísla a čísla zařízení a zároveň, že se v těchto sloupcích nachází pouze číslice. Pokud je pole *administration* úplně prázdné, upozorní na tento fakt uživatele.

ActivateDataMining.vi

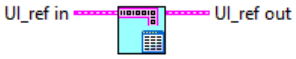


Na základě správnosti údajů v 2D poli *administration* funkce odešle do notifikátoru zprávu, ze které se v *Records Fetching Loop* vyhodnotí případná změna hodnoty proměnné *fetching status*.

SetUIConnect.vi

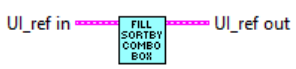
Přistupuje k prvkům uživatelského rozhraní a nastavuje jejich vlastnosti (viditelnost, editovatelnost, velikost atd.) po stisknutí tlačítka *Connect* uživatelem.

CopyInputIntoTable.vi



Funkce převádí údaje z 2D pole řetězců *administration* do 2D polí řetězců *overview* a *results*.

FillSortBy.vi

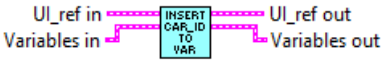


Funkce zapíše hodnoty do combo boxu *SortBy*, podle kterých bude možné následně řadit tabulku *results* na kartě *Výsledky*. Řadit lze podle výsledného času a jednotlivých mezičasů.

WithdrawOccupData.vi

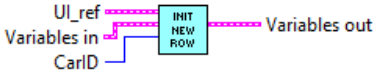
Funkce ze vstupního řádku 2D pole *administration* vybere hodnoty ze sloupce *deviceNr*, *startNr* a *crew*. Následně jsou tato data v nadřazené funkci *CopyInputIntoTable* vložena do combo boxů *watchCar* a *penaltyToCar*.

InsertCarIDToVariables.vi



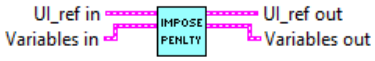
Funkce vybere postupně každé číslo zařízení z pole *administration* a inicializuje ho do clusteru proměnných *Variables*.

InitNewRow.vi



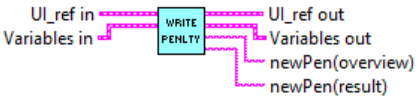
Funkce inicializuje proměnné, které souvisí s konkrétním číslem zařízení.

ImposePenalty.vi



Funkce zajišťuje zapsání času udělené penalizace na patřičné pozice (dle čísla vozu) do polí *overview* a *results*. Zároveň přes vnořenou funkci *WritePenalty* vytvoří událost o udělení penalizace.

WritePenalty.vi



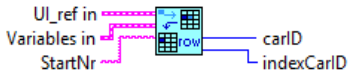
Z hodnoty v combo boxu *penaltyToCar* si funkce zjistí startovní číslo vozidla, kterému se uděluje penalizace, funkce *LinkStartNrWithRowIndex* zjistí, jaké číslo zařízení odpovídá startovnímu číslu, jelikož veškeré informace o vozidle jsou vedeny pod číslem zařízení. Funkce *LinkDeviceNrWithRowIndex* zjistí řádek, na kterém je umístěné číslo zařízení v poli *overview* a *results*. Následně se funkcí *ReadPenalty* zjišťuje, zda v poli *overview* není aktuálně postihovanému vozu udělena již jiná penalizace. Pokud ano, pak se k předchozí penalizaci přičte nově udělená a ze součtu se vytvoří nová hodnota pro zapsání do pole *overview*. Poté se do clusteru *CarData*, proměnné *Time.penalty* zapíše nová celková hodnota penalizace (v ms), tato hodnota v sekundách je zároveň novou hodnotou určenou pro zápis do pole *results*. *Time.timeToImposePenalty* se zapíše hodnota 10 (10 sekund se čeká, než se hodnota penalizace v poli *overview* přičte k orientačnímu času).

Součástí funkce *WritePenalty* je také funkce *GeneratePenaltyEvent*, která zajišťuje vytvoření události o penalizaci.

GetStartNr.vi

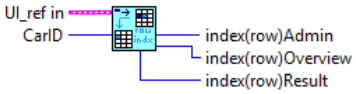
Funkce vyčte hodnotu proměnné *penaltyToCar* a oddělí z ní startovní číslo zvoleného vozu.

LinkStartNrWithRowIndex.vi



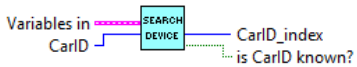
Funkce vybere z pole *administartion* sloupec obsahující pouze startovní čísla a integrovanou funkcí LabVIEW, *Search 1D*, nalezne pozici řádku, na kterém se startovní číslo nachází. Z tohoto řádku se pak vybere číslo zařízení. Dále se prohledává cluster *Variables*, aby se zjistilo, na které pozici se nachází konkrétní zařízení.

LinkDeviceNrWithRowIndex.vi



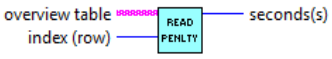
Funkce izoluje z každého pole (*administration*, *overview*, *results*) sloupec čísla zařízení a aplikuje na něj funkci *Search 1D*. Výsledkem jsou indexy řádku, kde se aktuálně nachází konkrétní číslo zařízení.

SearchDevice.vi

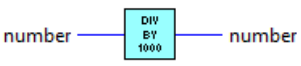


Funkce v cyklu prohledává pole *CarData*. Při nalezené shodě je cyklus for zastaven a příznak *is CarID known?* má hodnotu true. Poslední hodnota indexu cyklu for je zároveň indexem, kde se nachází hledané číslo zařízení. Pokud celý cyklus doběhne bez shody, *is CarID known?* má hodnotu false a *CarID_index* obsahuje nesprávnou hodnotu.

ReadPenalty.vi



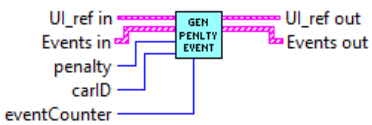
Funkce vyčte z pole *overview* hodnotu penalizace v sekundách na konkrétním řádku.



DivBy1000.vi

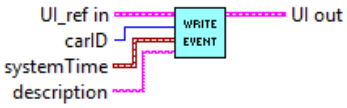
Funkce vydělí vstupní číslo tisícem.

GeneratePenaltyEvent.vi



Funkce zaznamená důležité informace o udělené penalizaci do clusteru *Events* – identifikátoru události (složený z čísla události a typu události), stručný popis události, systémový čas, a v tomto případě nepodstatný příznak *checked?* (ihned nastaven na true), nicméně v rámci obecnosti mají všechny události stejný formát informace.

WriteEvent.vi



Funkce upraví vstupní data a zapíše je do připravených sloupců do pole *events*.

FormatSystemTime.vi

Funkce převede systémový čas v datovém typu TimeStamp do datového typu string a ve formátu hh:mm:ss.

CreateAccidentURL.vi

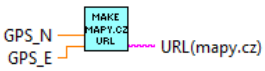
Z proměnné *accidentOfCar* (combo box, ze kterého uživatel vybral nehodu) funkce vyčte číslo události. Dále pak funkce prohledá všechny události. Nalezená data o události jsou vstupem funkce *CreateGPS_URL* a výstupem z ní je následně připravený internetový odkaz.

Funkce zároveň přepíše hodnotu proměnné *watchCar* na prázdný string, aby nedocházelo ke sledování vozidla, když uživatel požaduje zobrazení místa nehody.

CreateGPS_URL.vi

Funkce z dat o události vyčte pozici GPS, která je vstupem funkce *MakeMapyczURL*.

MakeMapyczURL.vi

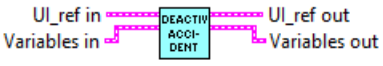


Funkce z pozice GPS vygeneruje odkaz na server mapy.cz ve formátu https://mapy.cz/zakladni?x=GPS_E&y=GPS_N&z=16&source=coor&id=GPS_E%2CGPS_N, kde:

z=XX … podrobnost zobrazeného mapového podkladu (nižší číslo = větší měřítko)

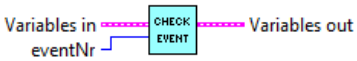
id= … souřadnice pro zobrazení značky

DeactivateAccident.vi



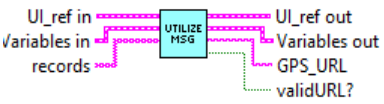
Funkce odstraní nehodu z combo boxu *accidentOfCar*, vyčte číslo aktuálně smazané události, které přebere funkce *CheckEvent*.

CheckEvent.vi



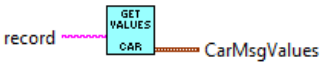
Funkce podle čísla události vyhledá odpovídající data události a změní hodnotu příznaku *checked?* na true.

UtilizeMsg.vi



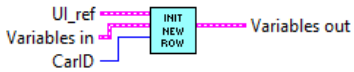
Celá funkce běží v cyklu for. Počet opakování je řízen aktuálním počtem vyčtených řádků z databáze (*records*). Každou jednotlivou zprávu převede pomocí funkce *GetValuesCar* z jednoho dlouhého řetězce na jednotlivé položky ve zprávě. Funkcí *ResetLastMsgTimer* se resetuje čas od poslední přijaté zprávy. Dále se pokračuje do funkce *AssignCarState*, kde se nad přijatou zprávou provedou patřičné operace (viz *AssignCarState*).

GetValuesCar.vi



Funkce převádí zprávu z dlouhého řetězce o známém formátu do clusteru, který obsahuje proměnné, do kterých je zpráva roztřízena.

ResetlastMsgTimer.vi

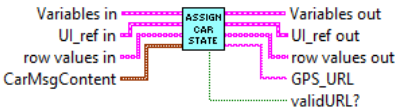


Funkce vyhledá cluster *CarData* konkrétního vozidla a do proměnné *Time.lastMsg* zapíše hodnotu 0 a do proměnné *Time.inclastMsg* hodnotu true. Čas od poslední přijaté zprávy je poté pro uživatele vidět v tabulce *overview*.

UnbundleOverview.vi

Funkce pouze z clusteru referencí *UI_ref* rozbalí hodnoty pole *overview*. Opačný postup pak provede funkce *BundleOverview*.

AssignCarState.vi



Hned zpočátku si funkce vyčte status, který se aktuálně nachází v poli *overview* u konkrétního vozidla. Zároveň se vyčte nový status z aktuálně příchozí zprávy, ten určuje case, který bude vybrán. Case pro ok je vybrán v případě, že status je 0, 1, 8, 9 (stavy popsány v kapitole 4.3.1 - B). Case accident je určen stavy 2 až 5, 10 až 13 je accident without relevant GPS.

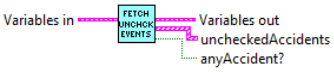
Case ok pokračuje výběrem všech neodbavených nehod pomocí funkce *FetchUncheckedEvents*. Následně se volá funkce *HandleAccidentArea*, která zjišťuje, zda se vozidlo aktuálně nachází v blízkosti nehody. Uvnitř case pro status ok se nachází další case, který dále dělí status na ok s GPS (status 0, 1) a na ok bez GPS (status 8, 9). V případě statusu 0 a 1 se provede funkce *CheckSpeedLimit*, která rozhodne, zda se má kontrolovat rychlost.

Pokud se vozidlo nachází mimo závodní trať, volá se funkce *MonitorSpeedLimit*, ve které se zkontroluje, zda vozidlo nepřekročilo rychlost v místě, kde by nemělo. Dále se funkcí *CheckAccidentStatus* kontroluje, zda vozidlo, jehož zpráva se aktuálně zpracovává, nemělo v předchozí zprávě nehodu a nyní se změnil status na ok. Další funkcí *CreateURLForWatchCar* se vytvoří internetový odkaz pro sledování vozidla na mapě. V případě statusu 8 a 9 se neprovádí žádná operace.

V case accident se funkcí *CheckForExistingEvent* ověřuje, zda aktuální vozidlo nehlásí opět už jednou hlášenou stejnou nehodu, díky čemuž se funkcí *WriteAccidentEvent* vytvoří nebo nevytvoří nová událost – nehoda. Funkce *FillCautionTable* odešle čísla zařízení všech vozidel, která se nacházejí v blízkosti některé z nehody do tabulky *caution* v databázi. Na základě přijatého stavu se také zapíše do příslušné buňky v poli *overview* buď *Nehoda* (status 2, 3) nebo *CrashButton* (4, 5).

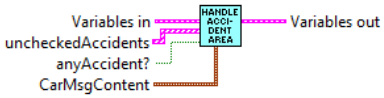
V case accident without relevant GPS se nahradí přijaté GPS souřadnice posledními validními souřadnicemi a pokračuje se stejně jako u předchozího casu accident. Do příslušné buňky v poli *overview* se zapíše *Nehoda,bezGPS*.

FetchUncheckedEvents.vi



Funkce prohledá všechna data událostí a vždy, když narazí na událost, která má příznak *checked?* nastaven na false, přidá data této události do výstupního pole. Příznak *anyAccident?* je vytvořen jako negovaný logický součet všech zkoumaných příznaků *checked?*.

HandleAccidentArea.vi



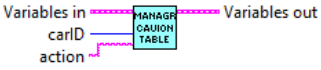
V případě, že žádná nehoda neexistuje, spustí se smyčka for, počet opakování závisí na velikosti pole *carData* a funkce *EvaluateCautionState* postupně zjišťuje poslední status každého vozidla. Pokud poslední status obsahoval informaci o tom, že vozidlo se nachází v blízkosti nehody, pak se přes funkci *ManageCautionTable* odešle příkaz o smazání konkrétního vozidla z databáze, tabulky *caution*. Pokud nějaká nehoda existuje, pak se nejprve vyhodnocuje, zda se vozidlo nachází v oblasti nehody (funkce *CompareGPS*). Zároveň je nutné vyhodnotit, zda byla součástí předchozího statusu informace o tom, že se vozidlo nachází v oblasti nehody. Kombinací těchto dvou booleovských hodnot mohou nastat 4 stavy:

- Vozidlo se nachází v oblasti nehody nyní i v předchozí zprávě. V takovém případě se neprovádí žádná operace.
- Vozidlo se nachází v oblasti nehody, ale v předchozím stavu se nenacházelo. Funkcí *ManageCautionTable* se odešle číslo tohoto zařízení do databáze.
- Vozidlo se nenachází v oblasti nehody a ani v předchozí zprávě se nenacházelo. Žádná další operace.
- Vozidlo se nenachází v oblasti nehody, ale v předchozí zprávě se nacházelo. Funkcí *ManageCautionTable* se odešle příkaz ke smazání vozidla z databázové tabulky *caution*.

EvaluateCautionState.vi

Funkce vyhodnocuje, zda status obsahuje informaci o tom, že zařízení je informováno, že se nachází v oblasti nehody.

ManageCautionTable.vi

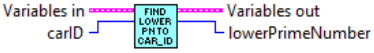


Funkce vytvoří odkaz podle známého formátu a odešle jej přes funkci *GET*, která je součástí LabVIEW. Hodnota code v odkazu se vyhodnocuje funkcí *GenerateCautionTableCode*.

GenerateCautionTableCode.vi

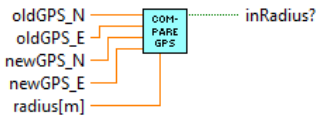
Funkce vyhledá první nižší prvočíslo (funkce *FindLowerPrimeNrToCarID*) než je hodnota čísla zařízení a první vyšší prvočíslo (funkce *FindHigherPrimeNrToCarID*) než je hodnota čísla zařízení. Pokud nižší prvočíslo neexistuje, je uvažováno první vyšší. Následně je nižší prvočíslo vynásobeno vyšším prvočíslem a výsledek sečten s číslem zařízení.

FindLowerPrimeNrToCarID.vi



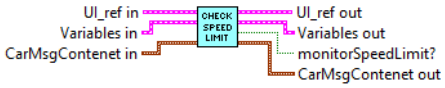
Hledání probíhá ve while smyčce. Od čísla zařízení se každým průchodem decrementuje o 1 a následně se hledá v připraveném poli prvočísel, zda takové prvočíslo existuje. Stejně pracuje funkce *FindLowerPrimeNrToCarID*, ovšem s tím rozdílem, že číslo zařízení je každým průchodem incrementováno.

CompareGPS.vi



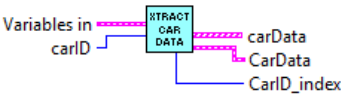
Funkce vypočítá vzdálenost mezi dvěma body GPS a vyhodnotí, jestli jsou od sebe tyto body vzdáleny méně, než je zadaný okruh.

CheckSpeedLimit.vi



Jelikož je nutné vozidlu po projetí cílem poskytnout čas na dobrzdění, aniž by bylo kontrolováno na rychlost, byla pro tyto účely vytvořena ochranná doba. Tato funkce zjišťuje, zda se vozidlo nachází v této ochranné době a podle toho nastavuje výstup *monitorSpeedLimit?*. Na false bude výstup rovněž nastaven v případě, že kontrola rychlosti není požadována (číslo 0 v proměnné *speedLimit*).

ExtractCarData.vi

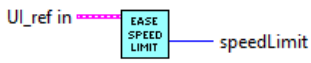


Funkce rozbalí z clusteru Variables pole *carData*, cluster *CarData* pro konkrétní číslo zařízení. Index je zjištěn pomocí funkce *SearchDevice*.

MonitorSpeedLimit.vi

Funkce nejprve vyhodnocuje, zda se vozidlo nachází na trati nebo mimo ni. Pokud se vozidlo nachází na trati, pak se kontrola rychlosti neuplatňuje. Pokud se rozhodne, že vozidlo je mimo trať, uplatní se funkce *EaseSpeedLimit*, která hodnotu z proměnné *speedLimit* zvýší o 20 %. Následně se rozhoduje, zda aktuální rychlost vozidla je vyšší než *speedLimit* zvýšený o 20 %. Dále se zjišťuje, zda se má rychlost kontrolovat pouze ve vypsaných úsecích (hodnota proměnné *speedLimitUseIndiator*), pokud má kontrola probíhat pouze ve vyznačených úsecích, zjišťuje se funkcí *CheckCarInSupervisionArea*, zda se v některé z oblastí nachází a pokud ano a zároveň je překročena rychlost, volá se funkce *WriteOverSpeedLimit*, která vytvoří událost o překročení rychlosti. Obdobně se postupuje i v případě, že sledování rychlosti má probíhat plošně, s tím rozdílem, že se vynechává funkce *CheckCarInSupervisionArea*.

EaseSpeedLimit.vi

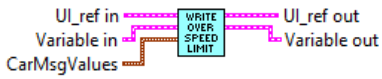


Funkce vynásobí hodnotu v proměnné *speedLimit* hodnotou 1,2 (20 % tolerance rychlosti – ne vždy zcela přesná interpretace polohy GPS může způsobit nepřesnosti v rychlosti.)

CheckCarInSupervisionArea.vi

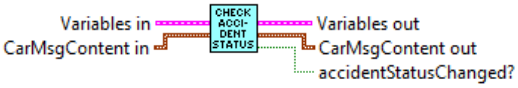
Funkce prověří, zda se vozidlo aktuálně nenachází v jedné z oblastí, ve kterých je kontrolována rychlost, použitím funkce *CompareGPS*.

WriteOverSpeedLimit.vi



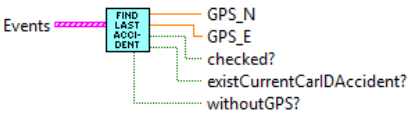
Funkce do clusteru událostí konkrétního vozidla zapíše informace o stavu vozidla, rychlosti, GPS poloze, dále systémový čas, stručný popis události a identifikátor události (složený z čísla události a typu události). Následně se událost pomocí funkce *WriteEvent* zapíše do pole *events*.

CheckAccidentStatus.vi



Funkce přepíše předchozí GPS souřadnice novými GPS údaji a poté zjišťuje, zda se vozidlo nenachází v místě své vlastní nehody, jelikož stav o nehodě se odesílá pouze jednou a následně si vozidlo myslí, že je vše zase v pořádku. Tato funkce ošetřuje, aby nedošlo k přepsání informace o případné nehodě v tabulce *overview*.

FindLastAccident.vi



Funkce prohledá v cyklu for pole clusteru *Events* konkrétního vozidla pro poslední nehodu.

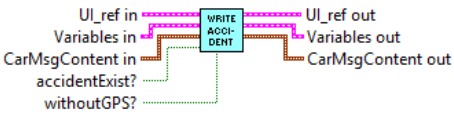
CreateURLForWatchCar.vi

Funkce prověří, jaké vozidlo je v proměnné *watchCar* vybráno a podle poslední zobrazené polohy vygeneruje nový URL odkaz. Pokud je nová poloha dále než 20 metrů od předchozí, pak se příznak *validURL?* nastaví na true. Pokud není nová poloha dále jak 20 m, pak bude příznak *validURL?* false, což je informace pro nadřazené funkce, že se žádný nový odkaz nemá odesílat.

CheckForExistingEvent.vi

Funkce zjišťuje, zda hlášení o nehodě na tomto místě a tohoto vozidla již jednou nebylo zpracováno. Pokud ano, příznak *accidentExist?* bude nastaven na true.

WriteAccidentEvent.vi

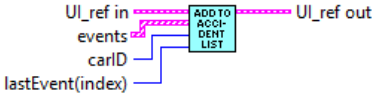


Funkce do clusteru událostí konkrétního vozidla zapíše informace o stavu vozidla, rychlosti, GPS poloze, dále systémový čas, stručný popis události a identifikátor události (složený z čísla události a typu události). Následně se událost pomocí funkce *WriteEvent* zapíše do pole *events*. Dále se pomocí funkce *SetUIaccident* nastaví uživatelské prostředí a funkce *AddToAccidentList* nehodu zapíše do combo boxu *accidentOfCar*.

SetUIaccident.vi

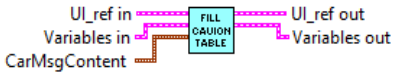
Funkce odblokuje tlačítka *ObserveAccident* a *CheckAccident* a přepne menu na kartu *Události*.

AddToAccidentList.vi



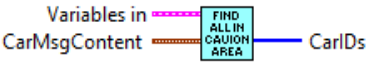
Funkce zapíše informace o nehodě do combo boxu *accidentOfCar* ve formátu *systémový čas – startovní číslo, posádka*.

FillCautionTable.vi



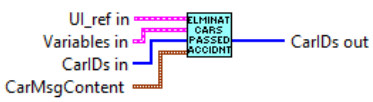
Pomocí funkce *FindAllInCautionArea* se nejprve vytvoří pole všech vozidel, která se nacházejí uvnitř oblasti nehody. Poté funkce *EliminateCarsPassedAccident* odstraní z pole všechna vozidla, která jela před vozidlem, jež způsobilo nehodu. Poslední funkce (*InsertCarIntoCautionTable*) postupně odešle všechna vozidla do databázové tabulky *caution*.

FindAllInCautionArea.vi



Funkce porovná poslední GPS polohu všech vozidel s polohou nehody. Pomocí funkce *CompareGPS* se zjišťuje, zda se vozidlo nachází v okruhu 300 metrů od nehody. Výstupem jsou všechna vozidla, která se nacházejí uvnitř této oblasti.

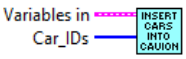
EliminateCarsPassedAccident.vi



Nejprve se vyloučí všechna vozidla, která mají hodnotu proměnné *trackPosition* větší než vozidlo, které hlásí nehodu (jsou tedy v závodě dále). Následně se zjišťuje časová značka zaznamenaná u průjezdu posledního kontrolního bodu. Pokud je tato hodnota aktuálně porovnávaného vozidla nižší

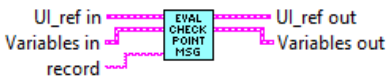
než u vozidla, které hlásí nehodu, předpokládá se, že je před nehodou a není ho třeba zahrnovat.

InsertCarsIntoCaution.vi



Ve funkci je spuštěn cyklus for, jehož počet opakování závisí na velikosti pole vozidel určených k zapsání do tabulky *caution*. Při každém průchodu se volá funkce *ManageCautionTable*, která zajistí vložení vozidla do tabulky *caution*.

EvalCHPMsg.vi



Řetězec, který obsahuje zprávu z checkpointu se funkcí *GetValuesCheckpoint* rozklíčuje do clusteru s jednotlivými položkami zprávy. Na základě informací obsažených v tabulce *checkpointsID* se pomocí funkce *ClasifyCheckpoint* přiřadí přijatý identifikátor k pořadí checkpointu na trati. Zároveň se převede hodnota statusu na booleovské pole (každý bit ve statusu má svůj význam). Následně se zjišťuje, zda stále není načtena GPS. Pokud stále není načtena, volá se funkce *ChangeReadiness*, která přepíše status konkrétního checkpointu v tabulce *checkpointsID* na stav *bezGPS*. Pokud už načtená je, pak se dále zjišťuje, zda byl zaznamenán IR kód a zároveň přerušení optického paprsku. Pokud tato podmínka splněna není, volá se funkce *CheckpointReadiness*, která přepíše stav konkrétního checkpointu v tabulce *checkpointsID* na !detekce nebo ok a případně upozorní uživatele. Pokud podmínka splněna je, pak se volá funkce *ChngeReadines*, která přepíše stav na ok, dále se volá funkce *AssignTrackPosition*, která vozidlu, které právě minulo checkpoint zvýší pozici na trati o 1 a následně se podle polohy checkpointu na trati aktivuje jeden z následujících case:

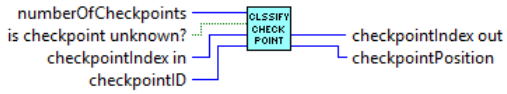
- Start
- Checkpoint
- Finish

V každém z těchto casu probíhají obdobné operace. Přiřazení času do clusteru *Variables* pomocí jedné z funkcí *InitRunT-AddStartT, AddCheckpointT, StopRunT-AddFinishT*, zapsání času do tabulky *results* v případě zapisování mezičasů a cílového času a zapsání 00:00:00 do tabulky *overview* v případě startu.

GetValuesCheckpoint.vi

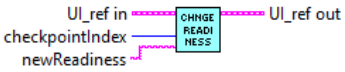
Funkce převede string o známém formátu na jednotlivé proměnné a zabalí je do clusteru.

ClasifyCheckpoint.vi



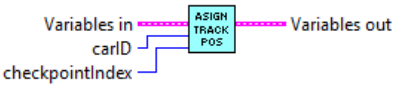
Funkce přiřadí checkpointu enum podle toho v jakém pořadí se nachází. Pokud je index checkpoint 0, pak je mu přiřazen *start*, pokud je index poslední, je mu přiřazen *finish*. Vše mezi tím má přiřazeno *checkpoint*.

ChangeReadiness.vi



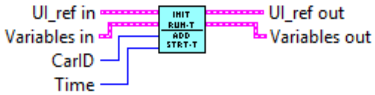
Funkce přepíše status v tabulce *checkpointsID* na novou hodnotu *newReadiness*.

AssignTrackPosition.vi



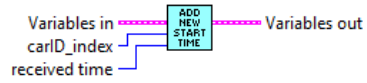
Funkce zvýší vozidlu, které aktuálně projelo okolo checkpointu, *trackPosition* o 1. Hodnota *trackPosition* začíná před startem na 0, po projetí startem je 1 atd.

InitRunT-AddStartT.vi



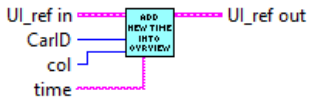
Funkce ověří, že číslo zařízení se nachází v poli *carData*. Pokud ne, funkcí *InitNewRow* toto zařízení inicializuje a následně funkce *AddNewStartTime* zapíše čas při startu k datům konkrétního vozidla. Pokud číslo zařízení již bylo inicializováno, uplatní se pouze funkce *AddNewStartTime*.

AddNewStartTime.vi



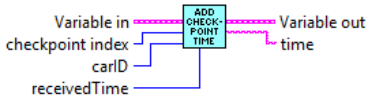
Funkce k datům konkrétního vozidla (proměnná *Time.startT*) zapíše hodnotu přijatého času z checkpointu (synchronizovaného s GPS). Do proměnné *Time.currentT* zapíše 0, aby mohlo začít nové čítání. Zároveň přepíše proměnnou *Time.incCurrentT* na true, z čehož část kódu v case *time* (*message handling loop*) určí, že má probíhat čítání orientačního času pro dané vozidlo.

AddNewTimeIntoOverview.vi



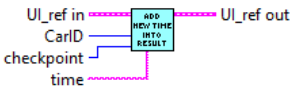
Funkce zapíše do buňky pro průběžný čas v tabulce *overview* hodnotu 00:00:00 (formát hh:mm:ss).

AddCheckpointT.vi



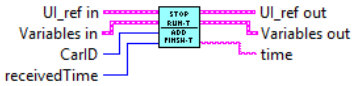
Funkce odečte *startT* od přijatého času a ten zapíše na základě vstupu *checkpointIndex* k patřičnému checkpointu do proměnné *Time.checkpoints*.

AddNewTimeIntoResult.vi



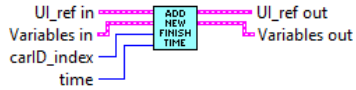
Funkce zapíše do buňky pro checkpoint dle vstupu změřený čas ve formátu mm:ss:msmsms.

StopRunT-AddFinishT.vi



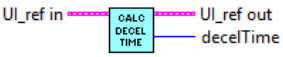
Funkce odečte *startT* od přijatého času. Ten jednak sečte s penalizací a převede jej na datový typ string (výstup *time*) a jednak odešle jako vstup do funkce *AddNewFinishTime*, kde se zapíše do proměnné *Time.finishT*.

AddNewFinishTime.vi



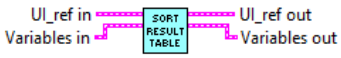
Funkce zapisuje vstup *time* do proměnné *Time.finishT*, nastavuje proměnnou *Time.incCurrentT* na false, volá funkci *CalcDecelTime* a zapisuje její výstup do proměnné *Time.decelerationT*.

CalcDecelTime.vi



Funkce vypočítá předpokládaný čas na zpomalení z rychlosti 200 km/h na rychlost v proměnné *speedLimit* s uvažovanou konstantní zápornou akcelerací 2 m/s. Po dobu trvání tohoto času nebude aktivní systém na kontrolu rychlosti mimo závodní trať.

SortResultTable.vi

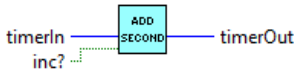


Funkce seřadí tabulku *result* podle vybraného sloupce v combo boxu *sortBy*. Řazení je vzestupné.

SwitchFreezeIndicator.vi

Funkce neguje stav viditelnosti prvků barevného čtverce v levém spodním rohu uživatelského prostředí. Celkem je součástí tohoto obrazce 6 pruhů a viditelné jsou vždy jen 3 (horizontální nebo vertikální).

AddSeconds.vi

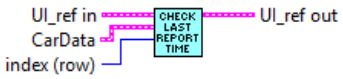


Pokud vstup *inc?* má hodnotu true, pak funkce ke vstupní hodnotě *timerIn* přičte hodnotu 1000 (ms) a novou hodnotu vrátí na výstupu *timerOut*. Obdobně pracuje funkce *SubtractSeconds*, která hodnotu 1000 (ms) odečítá.

UpdateTimeOverview.vi

Funkce v cyklu for prochází *CarData* všech vozidel a u každého, u kterého je nastaveno, že přičítání orientačního času je aktivní, aktualizuje orientační čas na novou hodnotu. Formát času je hh:mm:ss. Funkce zároveň sleduje, zda není aktivní odečítání času do připsání penalizace k orientačnímu času. Pokud aktivní je a zároveň čas do připsání penalizace je 0, přičte zadanou penalizaci k orientačnímu času. Jako poslední se uplatní u každého vozu funkce *ChckLastReportTime*, která kontroluje čas uplynulý od poslední zprávy přijaté od vozidla.

ChkLastReportTime.vi



Funkce vyčte aktuální hodnotu času od poslední zprávy, vydělí ji 1000 (převod z ms do s). Tato hodnota následně určuje case: 1, 10, 60, default a pomocí funkce *UpdateTSinceLastReport* se zapíše hodnota (<10s) nebo (<60s) nebo (>60s) nebo (neprovede se žádná změna) ke statusu konkrétního vozidla do tabulky *overview*.

UpdateTSinceLastReport.vi

Funkce nejprve pomocí funkce *GetStrBeforeSpace* vyjme status vozidla a za něj přidá novou informaci o délce trvání neaktivity vozidla.

GetStrBeforeSpace.vi

Funkce prohledá vstupní string a vyjme vše před prvním znakem mezery.

SaveResult.vi

Funkce nejprve seřadí pole *results* podle výsledného času pomocí funkce *Sort2D*, díky čemuž se na prvním řádku objeví startovní číslo nejrychlejší posádky. Na základě startovního čísla se vyhledá cílový čas vozidla v jeho clusteru *CarData*. Následuje funkce *PrepareResultDataForSave*, která nachystá potřebná data k uložení do souboru.

Sort2D.vi

Funkce seřadí 2D pole řetězců podle zvoleného sloupce vzestupně.

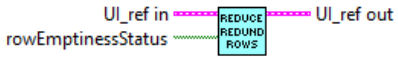
PrepareResultDataForSave.vi

Funkce připraví data jednotlivých vozů pro uložení takto: pořadí, startovní číslo, posádka, celkový čas, ztráta na prvního, mezičasy, penalizace, číslo zařízení.

AdaptSupervisionTable.vi

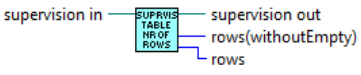
Funkce projde řádek po řádku, prvek po prvku, celé pole *supervision* a vyhodnotí u každé buňky, zda je prázdná nebo zda obsahuje číslo. Pokud jsou alespoň 2 řádky prázdné, funkce z tabulky oba řádky odstraní pomocí funkce *ReduceRedundantRows* a následně je pomocí funkce *ExpandSupervisionTable* jeden řádek přidán na konec tabulky. V případě, že všechny řádky a všechny buňky obsahují číslo, je přidán jeden řádek pomocí funkce *ExpandSupervisionTable*. Pokud je v jakékoliv buňce nalezen jiný znak než číselný (mimo tečky nebo čárky) je uživatel upozorněn na chybu v konkrétním řádku. Pokud je nalezena jakákoliv prázdná buňka, není provedena žádná operace (vyjma případu 2 a více prázdných řádků). Ani uživatel v takové situaci není upozorněn, když se na řádku, který obsahuje prázdnou buňku, nachází i jiný znak než pro číslo.

ReduceRedundantRows.vi



Jakékoliv true v poli *rowEmptinessStatus* má za následek vymazání řádku se stejným indexem jako toto true v poli *rowEmptinessStatus* z tabulky *supervision*. V případě, že při odmazání řádků dojde ke snížení celkového počtu řádků na 10 a méně, funkce odstraní vertikální posuvník. Počet řádků zjišťuje funkce *numberOfRowsSupervision*.

numberOfRowsSupervision.vi



Funkce zjistí počet řádků pole *supervision*.

ExpandSupervisionTable.vi

Funkce volá funkci *InitNewRow(supervision)*, která zajistí zvětšení tabulky o jeden řádek (nový řádek je plněn prázdným řetězcem).

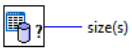
Následně se zjišťuje, zda při inicializaci nového řádku nedošlo ke zvýšení celkového počtu řádků na 11 a více. Pokud ano, je zviditelněn vertikální posuvník.

C) Records Fetching Loop

elapsedTime.vi

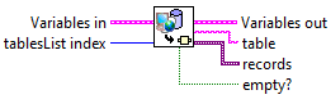
Funkce každou sekundu vygeneruje zprávu do fronty s hlavičkou *time*.

numberOfDBTables.vi



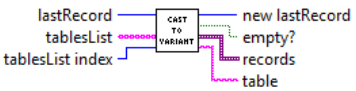
Funkce získá počet tabulek v databázi odsouzených k neustálému vyčítání.

GetNewRecords.vi



Funkce přebírá cluster *Variables*, ze kterého získá hodnotu řádku v databázi, od něhož bude vyčítat další hodnoty. Společně s hodnotami specifikujícími tabulku jsou předány do funkce *CastRecordsToVariant*. Výstupem je poté nový údaj o posledním řádku, tabulka, ze které se četlo, vyčtená data a booleovská hodnota, určující zda je pole vyčtených dat prázdné.

CastRecordsToVariant.vi



Funkce převádí data z výstupu funkce *FetchRecord* na univerzální datový typ variant. Pokud je vyčtené pole neprázdné, zvětší hodnotu posledního vyčítaného řádku o počet prvků ve vyčteném poli (tedy počet vyčtených řádků).

FetchRecord.vi

Na základě vstupních hodnot řádku a tabulky, ze které má probíhat čtení, funkce přistoupí k databázi pomocí funkce *GET* (součástí knihoven LabVIEW) známým skriptem na vyčtení všech následujících řádků od zvoleného. Výstupem funkce *GET* je dlouhý řetězec, kde je každý řádek oddělen znakem \n, díky čemuž tento řetězec může funkce *ObtainMsg* rozklíčovat do pole, kde každý element obsahuje jeden řádek. Pokud je z funkce GET získaný řetězec ve tvaru X,!, kde X je zadané číslo řádku, hodnota *empty?* se přepíše na *true* a následující funkce jsou takto informovány, že v DB nejsou žádné další nepřečtené řádky.

ObtainMsg.vi

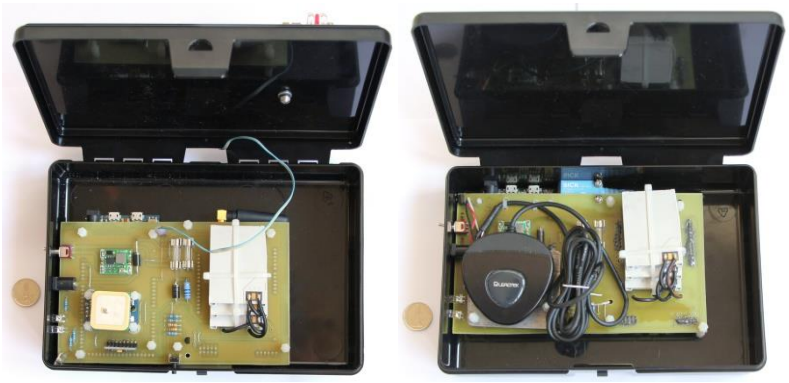
Funkce převádí dlouhý řetězec, ve kterém je oddělovač \n do pole. Každý element pole byl v řetězci oddělen znakem nového řádku \n.

5. Poznatky z praktického nasazení

Hned zpočátku je nutné poznamenat, že systém nebyl testován v ostrém provozu přímo na závodu rally. K takovému testování nedošlo z následujících důvodů:

- Z cenových důvodů byly vytvořeny pouze 3 zařízení (2 slouží jako checkpointy, 1 jako zřízení určené pro vozidlo), s takovým vybavením je skutečně celkem zbytečné testovat v ostrém provozu na závodě rally.
- Systém není zcela odladěný a občas se chová nepředvídatelně.

Proběhla tedy alespoň zkouška, při které byla mobilní část zařízení umístěna ve vozidle a projíždělo se okolo optické brány (checkpointu). Ukázalo se, že systém jako celek v dostatečně kvalitních podmínkách (vhodná poloha IR vysílače/přijímače, GPS senzor pod otevřeným nebem, mobilní signál) funguje. Dobře se osvědčil systém přenosu zpráv prostřednictvím databáze.



Obr. č. 11 – vlevo traťový modul (checkpoint), vpravo mobilní modul [7]

6. Závěr

Po několika návrzích na přenos informací mezi vozidlem a dispečerským stanovištěm, které se ukázaly jako ne zcela vhodné (kde kritérium vhodnosti je zejména cena a jednoduchost), se podařilo vyvinout systém, který má potenciál nahradit stávající, téměř neautomatizovaný, systém pro monitorování amatérské rally.

Prvním návrhem bylo vytvoření vlastní nezávislé sítě. V úvahu připadalo pouze volné frekvenční pásmo. Radioamatérský systém APRS se zdál být velice vhodný. Později se však projevily i některé nedostatky, jako neuspokojivá datová propustnost a možnost ztráty některých zpráv při přenosu.

Jako další návrh připadalo v úvahu využití již zbudované komerční sítě, mobilní datové služby, čímž se otevřelo velké množství možností pro výměnu dat. V rámci alespoň částečného splnění zadání, tedy vyvarovat se externím společnostem při přenosu dat, byla zkoušena komunikace přímo mezi jednotlivými zařízeními pomocí protokolu TCP/IP. Takové spojení však bylo velmi složité navázat (viz kapitola 3.1.2, bod A). Přešlo se proto na využívání databáze na externím serveru.

Aktuální systém využívá databázového serveru sql2.webzdarma.cz, který nabízí zdarma prostor 20 MB (4. 5. 2017), kde jsou vytvořeny 3 tabulky nesoucí informace. Operace s daty jsou zajištěny pomocí odkazů na skripty vytvořených v jazyce php.

Dispečerský software je naprogramován ve vývojovém prostředí LabVIEW. Uživateli nabízí přehledné sledování informací o jednotlivých vozidlech: status (v pořádku, nehoda), aktuální rychlost, výsledné časy (včetně mezičasů) – výsledky je možné řadit dle zvoleného kritéria, dále pak je možné sledovat aktuální polohu vozidla na mapovém podkladu. Obsluha má možnost udělit vozidlu penalizaci, nechat si zobrazit místo nehody na mapě (existuje-li nějaká nehoda), nastavit rozsah sledování rychlosti vozidla mimo závodní trať.

Společně s prací Ing. Petra Bílka vznikl systém, který plní většinu zadaných bodů, tzn. systém je více automatizovaný, stačí zadat vstupní data a o zbytek (kontrola průjezdů kolem kontrolních bodů, měření času, vyhodnocení nehody…) se stará technika. Není splněn bod o nepoužití komerční sítě. Dále není vytvořena hlasová komunikace a měření času vozidel stráveného v servisní zóně a není vytvořena žádná služba za poplatek pro týmy.

Seznam použitých zkratek

RZ	rychlostní zkouška
APRS	Automatic Position Reporting System
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
QSM-PC	Querry State Machine – Producer Consumer
DB	Database
URL	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

Seznam použité literatury a internetových zdrojů

[1] P. Litvák, „LabVIEW systém pro přenost telematických dat,“ Univerzita Pardubice, Pardubice, 2015.

[2] Alfano, „alfano.com,“ Alfano, 2015. [Online]. Available: http://www.alfano.com/wp-content/uploads/2013/12/PRO3_EVOv4.2.5_EN.pdf. [Přístup získán 7 9 2016].

[3] Bosch Motorsport, „bosch-motorsport.de,“ 23 10 2013. [Online]. Available: http://www.bosch-motorsport.de/media/catalog_resources/Telemetry_System_FM_40pdf~1.pdf. [Přístup získán 9 5 2017].

[4] NAM system, a.s., „onisystem.cz,“ [Online]. Available: https://www.onisystem.cz/wp-content/uploads/2017/04/ONI-system-RALLY-CZ.pdf. [Přístup získán 9 5 2017].

[5] „ctu.cz,“ ČTÚ, 12 5 2017. [Online]. Available: http://lte.ctu.cz/pokryti/. [Přístup získán 18 5 2017].

[6] P. Bílek, Artist, *testování propustnosti.* [Art]. ©pinckemposvete.com, 2017.

[7] P. Bílek, Návrh vozidlové jednotky pro monitorovací systém využitelný v podmínkách amatérské automobilové rallye, Pardubice: Univerzita Pardubice, 2016.

[8] J. Kring a J. Travis, LabVIEW for everyone, Pearson Education (US), 2015.

[9] P. Ponce-Cruz a F. D. Ramirez-Figueora, Inteligent Control System with LabVIEW, Springer Velag, 2009.

[10] National Instruments, „ni.com,“ National Instruments, 6 2008. [Online]. Available: http://www.ni.com/pdf/manuals/371525a.pdf. [Přístup získán 14 9 2016].

[11] Bosch Motorsport, „bosch-motorsport.de,“ 13 3 2017. [Online]. Available: http://www.bosch-motorsport.de/media/catalog_resources/Online_Telemetry_System_Overview_Datasheet_51_en_2784086923pdf.pdf. [Přístup získán 9 5 2017].

[12] „mysql.com,“ Oracle, [Online]. Available: http://www.mysql.com. [Přístup získán 2016].

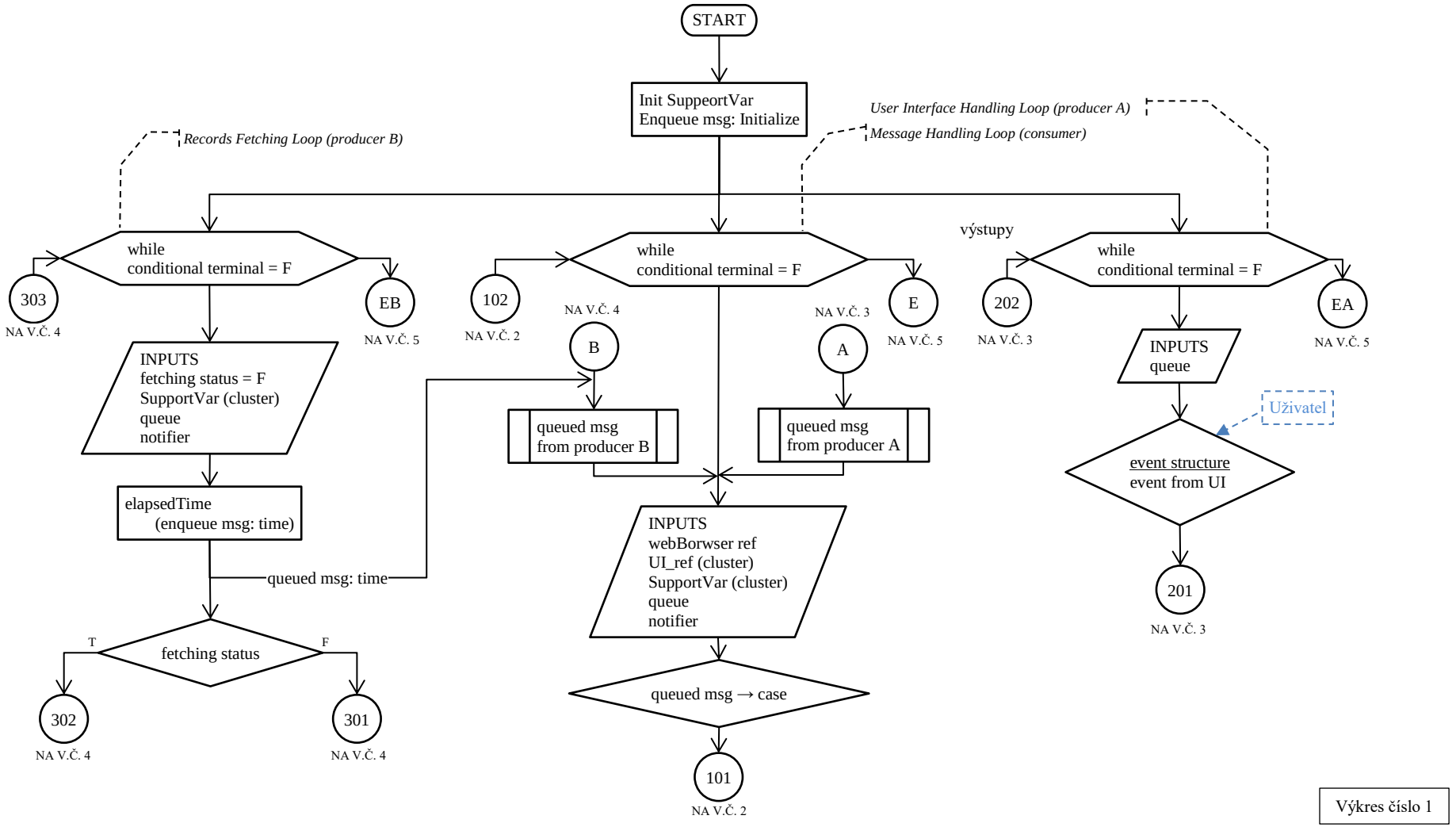
[13] „sql2.webzdarma.cz,“ phpAdmin, [Online]. Available: https://sql2.webzdarma.cz/sql.php?db=ralpa.xf.cz7268. [Přístup získán 2017].

[14] „w3schools.com,“ Refsnes Data, [Online]. Available: http://www.w3schools.com/sql. [Přístup získán 2016].

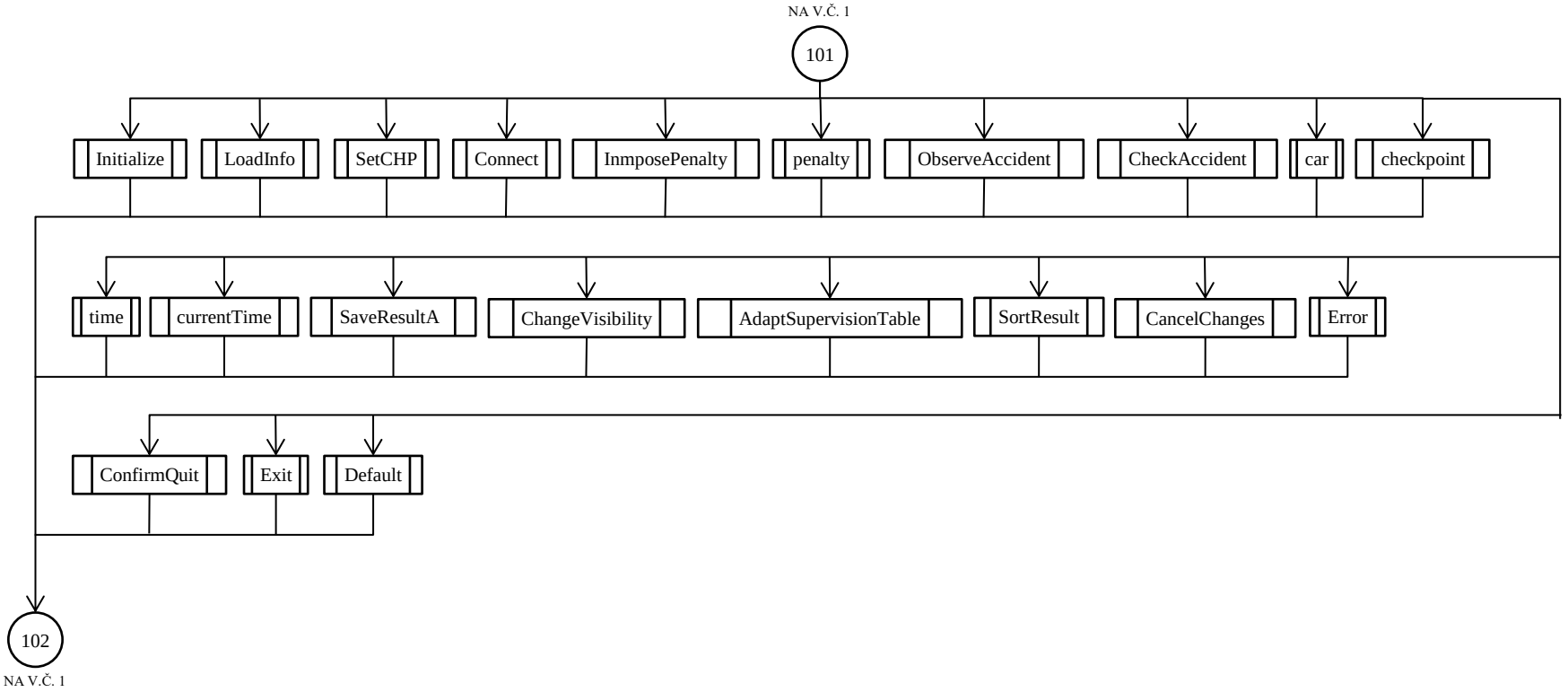
[15] „wikipedia.org,“ Wikipedia Foundation, [Online]. Available: https://en.wikipedia.org/wiki/Network_address_translation. [Přístup získán 2 2016].

[16] „stackoverflow.com,“ Stack Exchange Inc., 24 6 2012. [Online]. Available: http://stackoverflow.com/questions/11180035/how-to-set-up-a-udp-connection-between-two-computers-over-the-internet. [Přístup získán 2 2016].

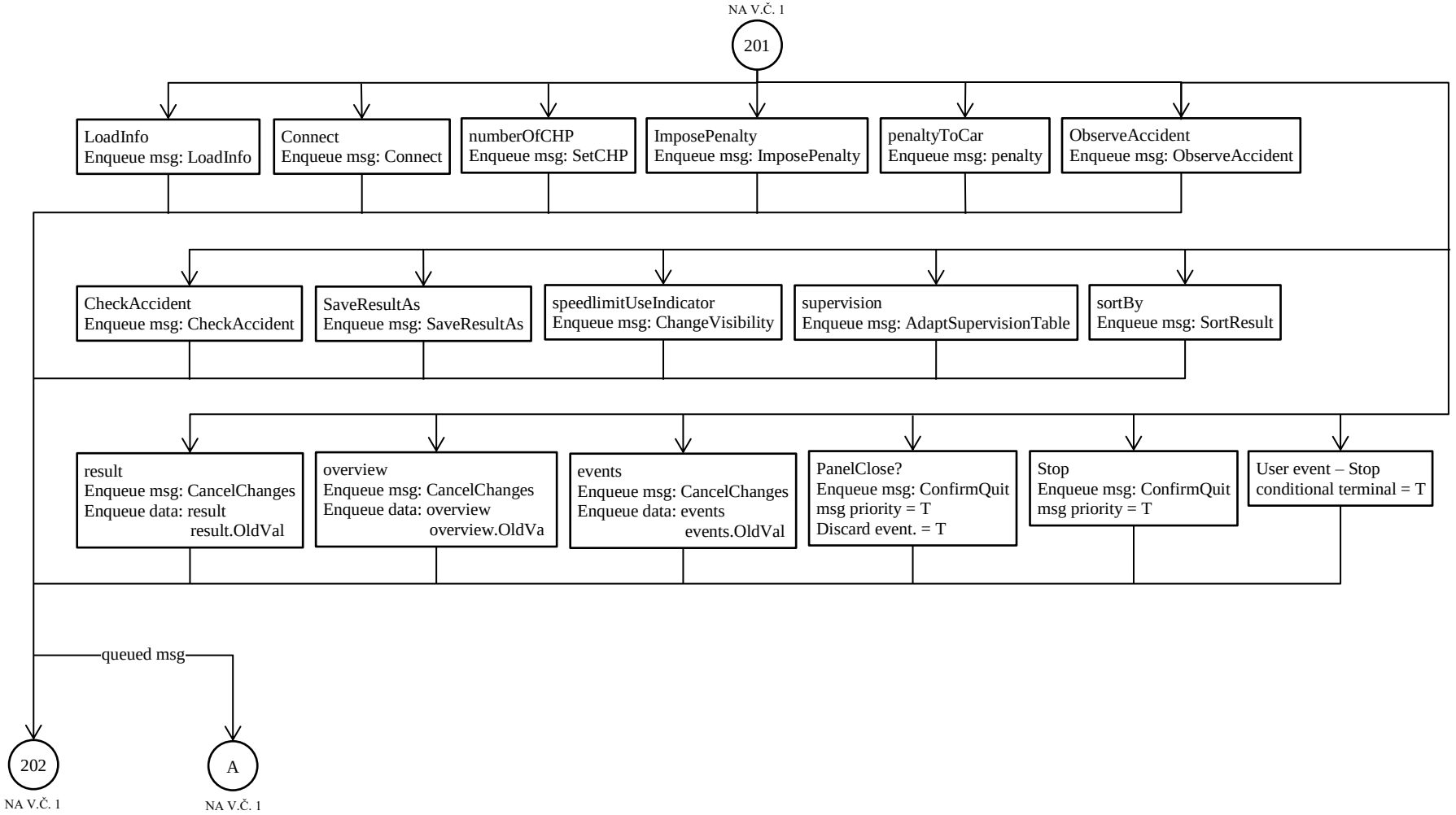
Příloha A – vývojový diagram



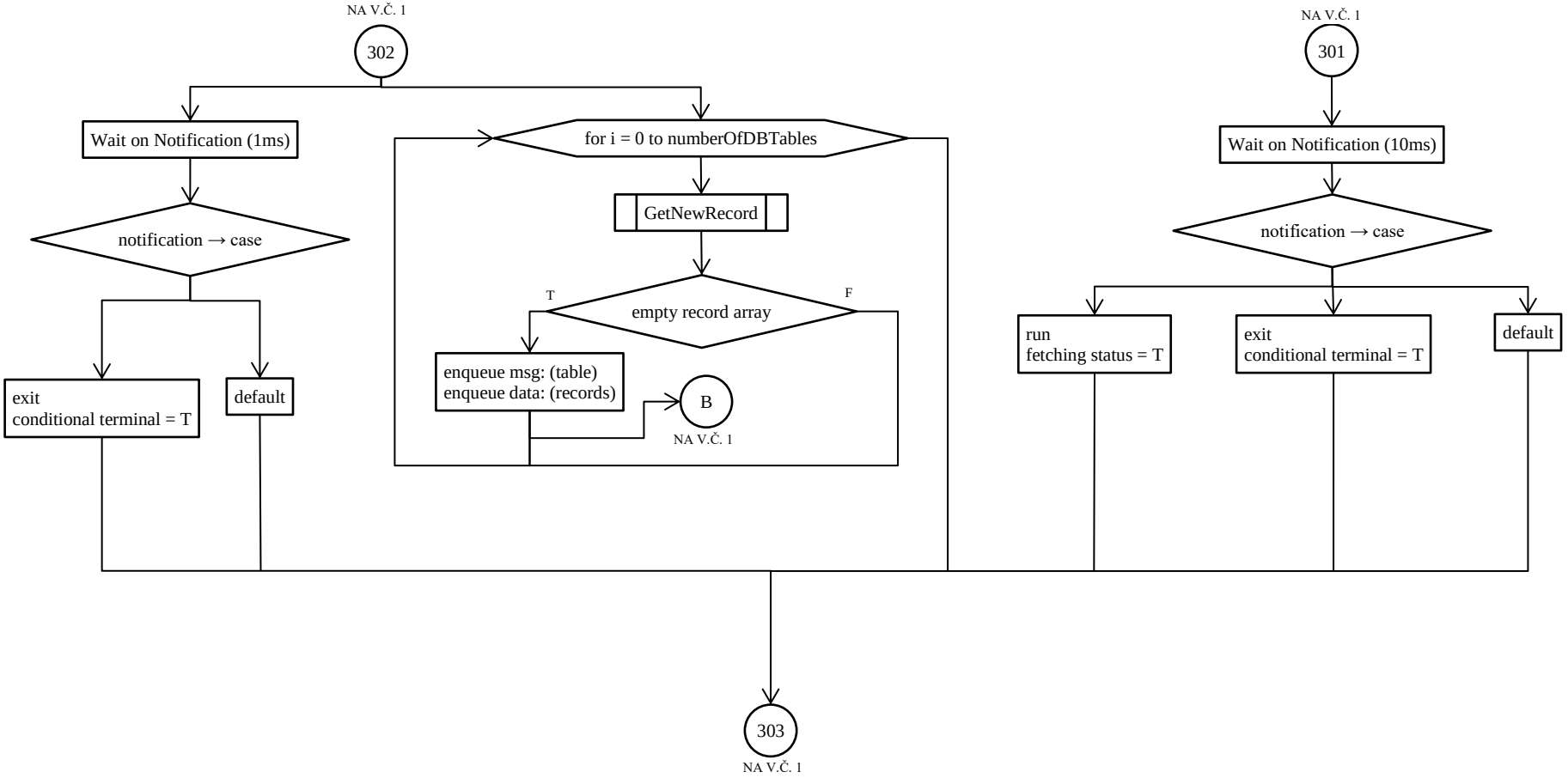
Výkres číslo 1



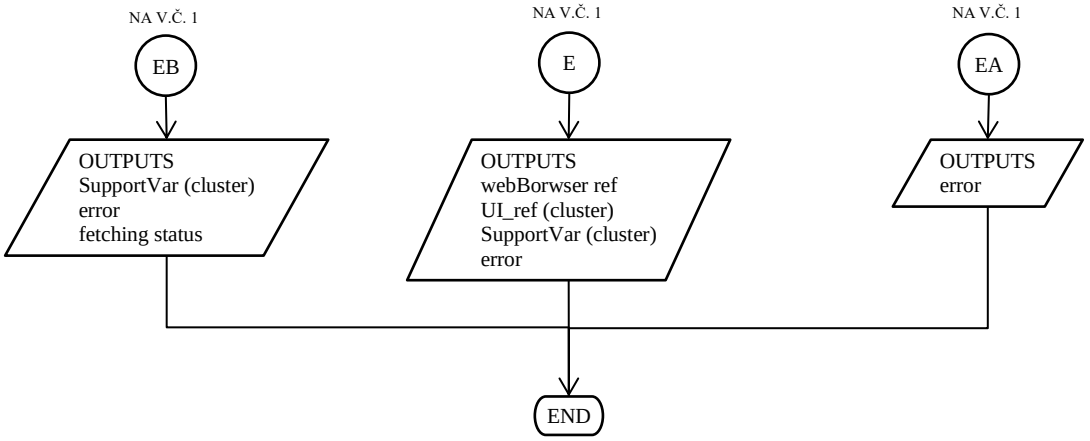
Výkres číslo 2



Výkres číslo 3



Výkres číslo 4



Výkres číslo 5

Příloha B

Disk obsahující pdf soubor s diplomovou prací, program *CarMsgSim* ve verzi LabVIEW 2011, hlavní program této DP opět ve verzi LabVIEW 2011. Dále pak některé další pdf soubory, které sloužily jako zdroje.