VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2011 Bc. Michal Gábor

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro analýzu a vizualizaci dat motorového vozidla
System for Analysis and Visualisation of Car Data

2011 Bc. Michal Gábor

Prehlásenie

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne …....

Touto cestou by som chcel poďakovať Ing. Michalovi Krumniklovi za trpezlivosť, podporu a dobré rady pri tvorbe tejto diplomovej práce.

Abstrakt

V dnešnej dobe sa nestretávame s pojmom počítač len v klasickom poňatí osobného počítača. Vyskytujú sa všade okolo nás či už ako mobilné telefóny, GPS prijímače, prenosné mobilné zariadenia či dokonca riadiace jednotky v motorových vozidlách. Cieľom tejto diplomovej práce je poukázať na možnosť použitia mobilného telefónu ako prostriedku, pre zobrazovanie a spracovanie údajov z riadiacej jednotky motorového vozidla. Toto riešenie pozostáva z viacerých hardwarových a softwarových súčastí: mobilný telefón, GPS prijímač, prevodník ELM, server. Umožňuje tak užívateľovi zobraziť a spracovať aktuálne údaje o polohe, rýchlosti z GPS a hodnoty senzorov, ktoré vyhodnocuje riadiaca jednotka motorového vozidla. Ďalej je možné tieto údaje odoslať na server a následne zobraziť v prostredí Google Earth. Systém podporuje autentifikáciu, autorizáciu a evidenciu automobilov a užívateľov. Klientská časť je implementovaná v prostredí .NET Compact framework, serverová časť v ASP .NET resp. framework MVC 2.0.

Kľúčové slová

OBD, autodiagnostika, záznam jazdy, monitorovanie vozidla, CAN zbernica, GPS, NETCF, ASP .NET, MVC

Abstract

Today's computers are more than the classical personal computers from years ago. In fact, we find them all around us: in smart phones, GPS navigations, mobile devices, and even in cars operating as engine control units. The purpose of this diploma thesis is to explore the possibilities of using a mobile phone as a device for displaying and processing data from the engine control unit. This solution is composed of multiple hardware and software parts: mobile phone, GPS unit, ELM converter and server. The user can access all information about the current position, speed and sensors values, gathered from the engine control unit. In addition, the data can be sent to a server and displayed in the Google Earth environment. The system supports authentication and authorization of users and vehicles. The client part is implemented in .NET Compact framework, the server in ASP .NET, MVC 2.0 Framework.

Keywords

OBD, vehicle diagnostic, driving report, vehicle monitoring, CAN bus, GPS, NETCF, ASP .NET, MVC

Zoznam významných symbolov a skratiek

OBD	On-board diagnostics
CAN	Controller Area Network
GPS	Global Positioning System
NMEA	National Marine Electronics Association
PID	Parameter identification number
DTC	Diagnostic trouble code
KML	Keyhole Markup Language
DLL	Dynamic Link Library
API	Application Programming Interface
CRC	Cyclic redundancy check
IIS	Internet Information Services
LINQ	Language-Integrated Query
ORM	Object relational mapper

Obsah

1. Úvod 1

2. On-board diagnostics 2

2.1. Základná charakteristika 2

2.2. Verzie protokolov OBD 2

3. Existujúce riešenia v oblasti OBD software 3

3.1. OBD Gauge 3

3.2. ProScan 4

3.3. pyOBD 5

3.4. TouchScan 6

3.5. Zhrnutie 6

4. Problematika navigačných systémov a možnosti spracovania GPS signálu 8

4.1. Družicové navigačné systémy 8

4.2. Global Positioning System 9

4.2.1. Štruktúra a princíp fungovania GPS 10

4.2.1.1.Kozmický segment 10

4.2.1.2.Riadiaci segment 10

4.2.1.3.Užívateľský segment 11

4.3. Komunikačný protokol NMEA 0183 11

4.3.1. Spracovanie prijatých dát 11

5. Štandard OBD II 13

5.1. OBD II konektor a fyzická vrstva 13

5.1.1. Komunikačné protokoly 13

5.2. Protokol SAE J1979 14

5.2.1. PIDs 15

5.2.1.1.Skupiny PIDs 15

5.2.1.2.Zoznam PIDs 15

5.2.1.3.Počiatočné preverovanie podporovaných PIDs 16

5.2.1.4.Proces získavania dát 17

5.3. Prevodník ELM 327 17

5.3.1. Komunikácia s ELM 327 18

5.3.2. Nastavenie a výber komunikačného protokolu 19

6. Využitie Google Maps a Google Earth pre zobrazenie dát 20

6.1. Google Maps API 20

6.1.1. API pre JavaScript 21
6.1.2. Google Earth API 22
6.1.2.1.KML 23
7. Vlastná realizácia 24
7.1. Klientská časť 24
7.1.1. Stručný náčrt funkcionality 24
7.1.2. Voľba platformy 25
7.1.3. GUI 26
7.1.3.1.Zobrazenie aktuálnych dát 26
7.1.3.2.Správanie aplikácie pri rôznych rozlíšeniach obrazoviek 27
7.1.4. Nastavenie aplikácie a ukladanie nastavení 27
7.1.5. Komunikácia prostredníctvom sériového portu 28
7.1.6. Spracovanie dát z GPS prijímača 29
7.1.6.1.NMEA parser 29
7.1.6.2.Autodetekcia sériového portu 30
7.1.7. Spracovanie dát z OBD 30
7.1.7.1.Inicializácia 30
7.1.7.2.OBD Parser 31
7.1.8. Sieťová komunikácia a spracovanie dát 32
7.1.8.1.Komunikačný protokol 32
7.1.8.2.Odosielanie nameraných dát na server 33
7.1.8.3.Práca pri slabej konektivite 34
7.1.8.4.Práca v úplne odpojenom režime 35
7.2. Serverová časť 35
7.2.1. Stručný náčrt funkcionality 35
7.2.2. Výber platformy 35
7.2.2.1.Vzor MVC 36
7.2.3. Komunikácia s klientskými aplikáciami 38
7.2.3.1.Spracovanie prihlasovacích údajov 39
7.2.3.2.Prístup k evidencií vozidiel 40
7.2.3.3.Príjem dát 40
7.2.4. Správa a riadenie užívateľských účtov 40
7.2.4.1.Autentifikácia a autorizácia 42
7.2.4.2.Evidencia užívateľov a definovanie užívateľských rolí 43
7.2.5. Ukladanie dát a práca s databázou 44
7.2.5.1.Implementácia LINQ to SQL do projektu 45
7.2.6. Vizualizácia dát a zobrazovanie jázd 46
7.2.6.1.Evidencia jázd 47

7.2.6.2.Vytvorenie KML súborov a formy zobrazenia dát 47
7.2.6.3.Odosielanie KML súborov 50
8. Záver a zhodnotenie práce 51
9. Literatúra 52
10. Prílohy 53

Zoznam obrázkov

Obrázok 1: Ukážka aplikácie OBD Gauge....3
Obrázok 2: Ukážka aplikácie OBD Graph for Windows....4
Obrázok 3: Ukážka rozhrania aplikácie ProScan....5
Obrázok 4: Ukážka rozhrania aplikácie pyOBD....5
Obrázok 5: Ukážka rozhrania aplikácie TouchScan....6
Obrázok 6: Chyba v GPS signále pred a po vypnutí SA....9
Obrázok 7: Konektor OBD II....13
Obrázok 8: Prevodník ELM 327 18
Obrázok 9: Pohľad na moduly realizácie....24
Obrázok 10: Architektúra .NET Framework....36
Obrázok 11: Architektonický vzor MVC....36
Obrázok 12: Provider model....41
Obrázok 13: Tabuľky v databáze....45
Obrázok 14: Klientská aplikácia: úvodná obrazovka....53
Obrázok 15: Klientská aplikácia: nastavenia....54
Obrázok 16: Klientská aplikácia: meranie dát....54
Obrázok 17: Klientská aplikácia: meranie dát nastavenie senzorov....55
Obrázok 18: Serverová aplikácia: úvodná sekcia....56
Obrázok 19: Serverová aplikácia: obrazovka po prihlásení administrátora do systému....56
Obrázok 20: Serverová aplikácia: evidencia vozidiel....57
Obrázok 21: Serverová aplikácia: evidencia užívateľov....57
Obrázok 22: Serverová aplikácia: nastavenie TCP servera....58
Obrázok 23: Serverová aplikácia: zoznam spojení od klientských aplikácií....58
Obrázok 24: Serverová aplikácia: zoznam jázd....59
Obrázok 25: Serverová aplikácia: jazdy, výber zobrazovanej veličiny....59
Obrázok 26: Serverová aplikácia: jazdy, legenda....59
Obrázok 27: Zobrazenie meranej veličiny v základnom móde....60
Obrázok 28: Zobrazenie meranej veličiny interpretované ako nadmorská výška....60
Obrázok 29: Zobrazenie jazdy pomocou simulátora....61

Zoznam tabuliek

Tabuľka 1: Vlastnosti vybraných OBD II aplikácii.....7
Tabuľka 2: Úloha pinov v konektore OBD II pri rôznych protokoloch.....14
Tabuľka 3: Skupiny PIDs [1].....15
Tabuľka 4: PIDs použité v tomto projekte[1].....16
Tabuľka 5: PIDs dotazy na zistenie podpory[1].....16
Tabuľka 6:Podporované protokoly v ELM 327 pre komunikáciu s riadiacou jednotkou....19

Zoznam zdrojových kódov

Zdrojový kód 1: Vloženie API kľúča....21
Zdrojový kód 2: Vytvorenie mapy....21
Zdrojový kód 3: Vytvorenie Google Earth mapy....22
Zdrojový kód 4: Príklad štruktúry KML súbora....23
Zdrojový kód 5: Metódy pre vykresľovanie grafiky v UserControll....27
Zdrojový kód 6: Vytvorenie nového fontu....27
Zdrojový kód 7: Hlavičky metód pre prácu s registrami....27
Zdrojový kód 8: Metódy pre prácu a nestavania sériového portu....28
Zdrojový kód 9: Získanie zoznamu sériových portov....28
Zdrojový kód 10: Odosielanie prihlasovacích informácií....33
Zdrojový kód 11: Inicializácia jazdy....34
Zdrojový kód 12: Odosielanie nameraných dát....34
Zdrojový kód 13: Ukončenie jazdy....34
Zdrojový kód 14: Základné metódy pre spracovanie užívateľských účtov....42
Zdrojový kód 15: Oddelenie časti View pre vybraných užívateľov....42
Zdrojový kód 16: Autorizácia na strane akčnej metódy....43
Zdrojový kód 17: Hlavička akčnej metódy pre operácie nad užívateľskými účtami....43
Zdrojový kód 18: Spôsob pridávania a odoberania užívateľa do role....43
Zdrojový kód 19: Reset užívateľského hesla....43
Zdrojový kód 20: Vloženie objektu typu Car do tabuľku....46
Zdrojový kód 21: Spracovanie validačných správ....46
Zdrojový kód 22: Akčná metóda pre spracovanie operácií nad záznamov jazdy....47
Zdrojový kód 23: Príklad štruktúry deklarácie štýlu v KML súbore....48
Zdrojový kód 24: Príklad štruktúry deklarácie úsečky v KML súbore....48
Zdrojový kód 25: Príklad štruktúry deklarácie počiatočnej pozície v KML súbore....49
Zdrojový kód 26: Metódy pre vytváranie KML súborov....50
Zdrojový kód 27: Odosielanie KML súborov....50

Zoznam vybraných triednych diagramov

Triedny diagram 1: Klientská aplikácia: SerApi.....62
Triedny diagram 2: Klientská aplikácia: AquaGauge.....62
Triedny diagram 3: Klientská aplikácia: NmeaParser.....62
Triedny diagram 4: Klientská aplikácia: Tools.....63
Triedny diagram 5: Klientská aplikácia: ConnSettings.....63
Triedny diagram 6: Klientská aplikácia: ObdParser.....63
Triedny diagram 7: Klientská aplikácia: Client.....63
Triedny diagram 8: Serverová aplikácia: Server.....64
Triedny diagram 9: Serverová aplikácia: ConnectionHandler.....64
Triedny diagram 10: Serverová aplikácia: MapsParser.....64
Triedny diagram 11: Serverová aplikácia: Tools.....64

1. Úvod

V dnešnej dobe zažívajú mobilné zariadenia veľký rozmach. Mobilné telefóny sa postupom času menia na prenosné počítače s výpočtovým výkonom, ktorý by im v nedávnej minulosti záviděli aj stolové počítače. Tak isto záber využitia výpočtovej techniky je v súčastnosti rôznorodý a rozsiahly. Čím ďalej, tým viac sa ich pôsobenie približuje do každodenného života ľudí. Výpočtová technika postupne našla uplatnenie aj v automobilovom priemysle. V súčastnosti je srdcom každého vyrobeného vozidla riadiaca jednotka, ktorá má za úlohu hlavne monitorovať, riadiť a pokiaľ je to možné zabezpečiť bezproblémovú prevádzku vozidla.

Cieľom tejto diplomovej práce je vytvoriť systém, ktorý bude získavať vybrané dáta z motorových vozidiel a následne ich spracovávať a zobrazovať. Celé riešenie sa skladá z dvoch častí: klientskej a serverovej. Aplikácie medzi sebou komunikujú prostredníctvom sieťového rozhrania.

Klientská časť má za úlohu hlavne obstarať dáta získavané z riadiacej jednotky vozidla prostredníctvom protokolu OBD II a dáta z navigačného systému GPS, teda údaje o polohe, rýchlosti a čase. Aplikácie je určená pre mobilné zariadenie, ktoré je umiestené priamo vo vozidle. Poskytuje užívateľovi určitú mieru nastavení, jedná sa hlavne o možnosť výberu zobrazovaných a odosielaných dát. Údaje, ktoré aktuálne aplikácia spracúva sú v prehľadnej forme zobrazované. Ďalšia časť mobilnej, respektíve klientskej aplikácie sa stará o odosielanie nameraných dát na serverovú časť aplikácie. Serverová časť obsahuje evidenciu užívateľov a motorových vozidiel. Klientská aplikácia ich bude spätne využívať. Systém teda podporuje autentifikáciu a autorizáciu viacerých užívateľov. Táto časť aplikácie rieši aj problémy so slabým alebo prerušovaným pripojením. Klientská aplikácia dokáže pracovať aj v úplne odpojenom režime a dáta neskôr vo vhodnej chvíli odoslať na server.

Serverová časť slúži v najväčšej miere na vyhodnotenie a zobrazenie dát, ktoré boli získané od klientov. Okrem toho obsahuje už spomínané evidencie, nad ktorými umožňuje prevádzať štandardné operácie. Hlavnou časťou tejto aplikácie je evidencia jázd, ktorá obsahuje jednotlivé záznamy spolu z nameranými dátami. Dáta je možné vizualizovať v prostrední Google Earth. Jazda je zobrazovaná ako postupnosť úsečiek, ktorých hodnota je vyjadrená odlíšením farby, respektíve výškou, vbb keďže sa jedná o zobrazenie v 3D. Aplikácia umožňuje aj vizualizáciu jazdy prostredníctvom simulátora. Tento krát sa jedná o pohybujúci sa model vozidla, opäť v prostrední Google Earth. Prístup do aplikácie je riešený prostredníctvom webového informačného systému. Súčasťou je aj TCP server, ktorý obstaráva komunikáciu s klientskými aplikáciami.

Prvé časti tejto diplomovej práce sú skôr teoretického charakteru. Objasnená je problematika diagnostiky vozidiel a obzvlášť protokolu OBD II. Ďalšie časti obsahujú stručný prierez existujúcich riešení založených na OBD II protokole. Tak isto je objasnená problematika navigačných systémov a možnosti spracovanie GPS signálu. S tým je spojené zobrazovanie dát v prostredí Google Maps a Google Earth.

V ďalších častiach sa táto práca snaží o objasnenie vlastného riešenia, ktoré bolo predmetom zadania tejto diplomovej práce. Priblížené je fungovanie jednotlivých modulov a súčasti implementácie či už na klientskej alebo serverovej aplikácií.

Súčasťou je aj užívateľská príručka, ktorá má za úlohu objasniť fungovanie aplikácie z pohľadu užívateľa. K dispozícií sú aj nafotené obrazovky jednotlivých častí.

2. On-board diagnostics

2.1. Základná charakteristika

Význam pojmu OBD[1] môžeme definovať ako palubná diagnostika automobilu. Jedná sa o súbor špecifikovaných prostriedkov, ktoré umožňujú spracovávať stav, respektíve hodnoty rôznych podsystémov a prevádzkových parametrov v motorových vozidlách. Najčastejšie sa používa pri zisťovaní informácií za účelom servisu alebo odstraňovania chyby automobilu. Servisnému technikovi poskytuje za relatívne krátky čas pohľad na fungovanie a správanie systémov v automobile. Ďalším aspektom využitia môže byť spracovanie informácií pre samotného užívateľa motorového vozidla, ktorý má záujem bližšie sledovať správanie vozidla a nestačia mu základné dostupné informácie. V neposlednom rade treba spomenúť aj využite za účelom dohľadu nad vozidlom. Jedná sa hlavne kontrolu spotreby, stav palivovej nádrže a iné dôležité parametre, ktoré sú dôležité napríklad pre vedenie prepravnej spoločnosti či majiteľov priemyselných mechanizmov.

Pre spracovanie, respektíve zobrazenie údajov je potrebný software, ktorý funkcie obstará. Komunikácia s riadiacou jednotkou automobilu je štandardizovaná. Bližšie informácie o protokole sú uvedené v nasledujúcich kapitole 5.Štandard OBD II .

2.2. Verzie protokolov OBD

- Protokol OBD I: bol primárne určený pre meranie emisií.
- Protokol OBD II: zavedený v roku 1996 v USA. Cieľom bolo zjednotenie systému emisnej diagnostiky[2]. Zjednotenie spočíva v zachovaní kompatibility na rôznych automobiloch a aj pri rôznych výrobcoch.
- Protokol EOBD: jedná sa o európsku verziu OBD II. V podstate je identický s OBD II, rozdiel je len v dátumoch zavedenia:
  - zvedenie pre osobné automobily s benzínovým motorom: rok 2001,
  - zvedenie pre osobné automobily s dieslovým motorom: rok 2004,
  - zvedenie pre nákladné automobily a autobusy: rok 2005 [1].

Podpora protokolov v automobiloch vyrobených pred oficiálnym zavedením protokolov je voliteľná.

1 On-board diagnostics
2 Systém emisnej diagnostiky – systém, ktorý ovplyvňuje množstvo emisií vyprodukovaných motorovým vozidlom.

3. Existujúce riešenia v oblasti OBD software

V predchádzajúcej kapitole je uvedený význam OBD. K samotnému fungovaniu je však potrebný obslužný software, ktorý bude údaje spracovávať, poprípade vizualizovať. Tento software môže pracovať na rôznych mobilných platformách alebo platformách určených pre pevné počítače. Tak isto sa môže líšiť licencia pod ktorou bude aplikácia distribuovaná a miera funkcionality, ktorou aplikácia disponuje.

3.1. OBD Gauge

Freeware software, ktorý je dostupný pre rôzne platformy.

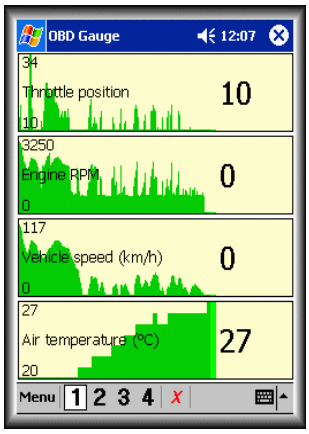
OBD Gauge pre PalmOS a OBD Gauge pre PocketPC

Podporuje čítanie dát zo snímačov, čítanie diagnostických kódov, odstránenie „check engine“ upozornenia, uchovávanie údajov a zobrazenie v grafe. Podmienkou funkčnosti je sériový port na zariadení. Pri PacketPC je kompatibilita zaručená od verzie PC 2002 alebo 2003.

Aktuálne je software vo verzii:

- OBD Gauge 1.9 for Palm
- OBD Gauge 1.3 for Pocket PC

K dispozícii sú aj zdrojové kódy aj dokumentácia pre všetky uvedené platformy [2].

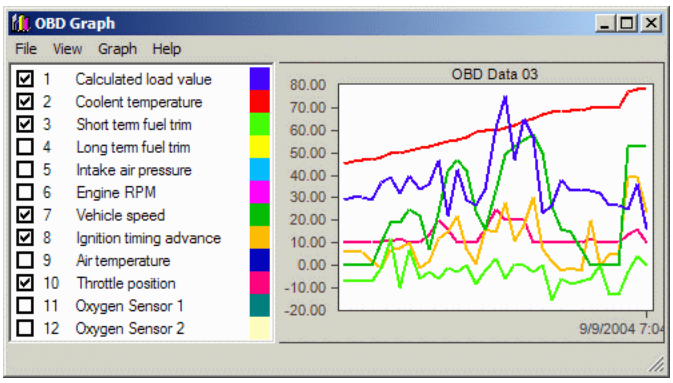


Obrázok 1: Ukážka aplikácie OBD Gauge

OBD Graph for Windows

Zobrazuje údaje zo senzorov, ktoré boli nahrané aplikáciu OBD Gauge. Kreslí grafy a tabuľky. Umožňuje export do CSV[3] súboru[2]. Systém OBD Gauge sa teda skladá z 2 častí. Časť, ktorá zbiera údaje a druhá časť, ktorá ich zobrazuje a vyhodnocuje. Každá časť aplikácie je implementovaná pre iné prostredie a operačný systém. Aktuálna verzia softwaru je: OBD Graph 1.1 for Windows.

3 Comma-separated value – súbor vhodný pre ukladanie tabuľkových dát oddelených čiarkami.



Obrázok 2: Ukážka aplikácie OBD Graph for Windows

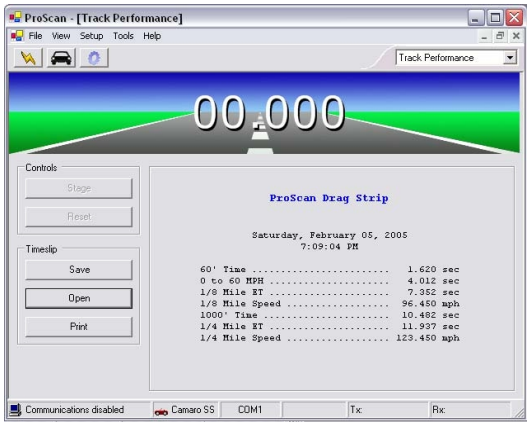
3.2. ProScan

Jedná sa o software určený pre operačné systémy Windows. Konkrétne sa o verzie od Windows 98 až po Windows 7. Nutná podmienka je prítomnosť USB portu alebo Bluetooth. Produkt teda nie je primárne určený pre mobilné zariadenia. Je distribuovaný pod licenciou shareware. Výrobca ho ponúka ako kompozíciu hardware a software. Tento balíček pozostáva z software, prevodníku OBD II, potrebných pripojovacích káblov a užívateľskej príručky. Ak zákazník disponuje kompatibilným hardwarom, tak je možné dokúpiť len software. Podmienkou fungovania je protokol OBD II vo vozidle. Zákazník si môže vybrať z dvoch variánt produktu. Rozdiel vo variantách spočíva v spôsobe pripojenia prevodníka na počítač. V základnej verzií ponúka pripojenie prostredníctvom USB, v rozšírenej bezdrôtovo, prostredníctvom Bluetooth. Cena balíčkov je uvedená na stránkach produktu. Aktuálne je 180 $, respektíve 240 $. Výrobca pri zakúpení ponúka doživotne bezplatný update aplikácie [3].

ProScan podporuje nasledujúcu funkcionalitu:

- zobrazenie tzv. „Freeze“ údajov. Sú to dáta, ktoré boli zaznamenané práve v tom čase, kedy došlo vo vozidle k poruche alebo nastala chyba,
- mazanie chybových kódov a Freeze údajov,
- odstránenie „check engine“ upozornenia,
- monitorovanie hodnôt z kyslíkových senzorov,
- čítanie hodnôt až z 169 senzorov vo vozidle, záleží na samotnom vozidle, ktoré senzory budú podporované,
- zobrazenie dát v reálnom čase v rôznych formátoch (tabuľky, grafy),
- nahrávanie, prehrávanie dát,
- výpočet nákladov na cestu podľa priemernej spotreby a vzdialenosti k cieľu,
- odhad výkonu a krútiaceho momentu motora,
- export dát do formy vhodnej pre tlač,

- prepínanie medzi anglickými a metrickými mierami [3].



Obrázok 3: Ukážka rozhrania aplikácie ProScan

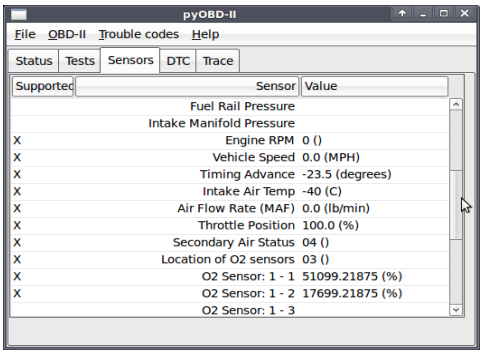
3.3. pyOBD

Tento diagnostický software je šírený pod licenciou GPL[4]. Pre komunikáciu s riadiacou jednotkou využíva prevodník ELM 32X. Podporuje základné funkcie ako zobrazenie štandartných OBD hodnôt, zobrazenie chybových kódov[4].

Aplikácia je výlučne napísaná v jazyku Python. Výrobca udáva testovanie a bezproblémovú funkčnosť na operačných systémoch: Windows, Debian, Ubuntu, Mac OSX 10.3 pri splnení nasledujúcich požiadaviek:

- OBD II protokol v automobile,
- prevodník ELM 32x,
- sériový port[4].

Výhodou tohoto software je prenositeľnosť medzi rôznymi operačnými systémami.



Obrázok 4: Ukážka rozhrania aplikácie pyOBD

4 General Public License – licencia pre slobodný software.

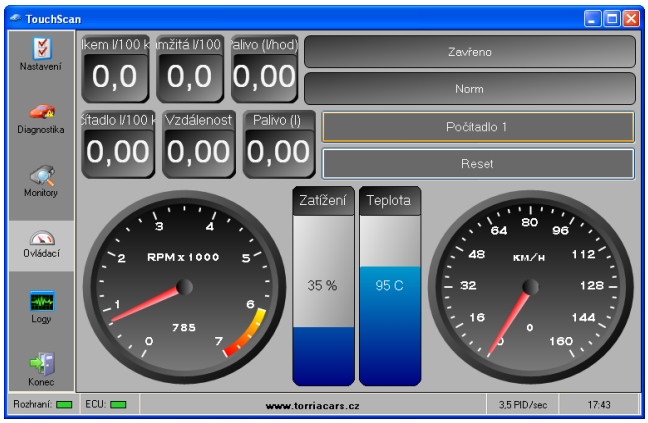
3.4. TouchScan

Tento OBD software zaujme najmä tým, že je plne lokalizovaný do češtiny. Vyvíjaný je americkou firmou OCTech, LCC. Je šírený pod licenciou shareware. K testovaniu je k dispozícií aj Trial verzia [5].

TochScan štandardne využíva na komunikáciu prevodník ELM 327. Výrobca udáva nasledujúcu podporovanú funkcionalitu:

- čítanie a mazanie chybových kódov motora,
- livedata – štandardné hodnoty dostupné v protokole OBD II,
- zobrazenie tzv. „Freeze“ údajov. Sú to dáta, ktoré boli zaznamenané práve v tom čase, kedy došlo vo vozidle k poruche alebo nastala chyba,
- informácie z palubnej dosky,
- test monitor jednotlivých systémov pre splnenie emisnej normy,
- informácie o vozidle (vin číslo...),
- výpočet aktuálnej spotreby vozidla (užitočné pri vozidlách bez palubného počítača) [5].

Aplikácie je určená pre operačné systémy Windows. Podporované sú verzie: Windows 2000, Windows Vista, Windows XP, Windows 7 [5].



Obrázok 5: Ukážka rozhrania aplikácie TouchScan

3.5. Zhrnutie

Dostupných aplikácií pre spracovanie údajov z OBD existuje mnoho. Táto kapitola bližšie popísala pár výraznejších zástupcov. Hlavné rozdiely medzi aplikáciami môžeme hľadať najmä v platformách, pre ktoré sú určené, funkcionalitu, licenciu a podobne. Záleží na požiadavkách užívateľa, pre ktorú sa rozhodne. Nasledujúca tabuľka ukazuje na základné vlastnosti a charakteristiky vybraných OBD II aplikácií.

Software	Zobrazenie Freeze dát	Mazanie chyb. kódov	Mobilná platforma	Ukladanie dát	Freeware	Grafy a tab.	Výpočty spotreby...
OBD gauge	●	●	●	●	●	●	
ProScan	●	●		●		●	●
pyObd					●		
TouchScan	●	●					●
Digimoto lite		●		●	●	●	
EasyObd II	●	●					
Obd logger	●			●	●		
Obd 2 crazy		●		●	●		
Obd2 Spy	●	●		●		●	

Tabuľka 1: Vlastnosti vybraných OBD II aplikácii

Pravidlom býva, že takýto software bude na komunikáciu využívať prevodník ELM 32x, komunikuje väčšinou prostredníctvom Bluetooth, USB alebo sériového portu. Využíva komunikačný protokol OBD II, čo v podstatne umožňuje sledovať základné parametre vozidla. Túto funkcionalitu podporuje v určitej miere každý dostupný software. Pri platených verziách sa môže jednať o rozšírenie aplikácie, ktorá vykonáva rôzne výpočty (odhad spotreby paliva na danú vzdialenosť, výpočet aktuálnej spotreby, cena za najazdenú vzdialenosť a podobne), poprípade exportuje údaje do podoby vhodnej pre tlač alebo vykresľuje grafy a štatistiky. Pravidlom býva aj dobrá dokumentácia a technická podpora. OBD software, ktorý spadá pod licenciu „Freeware“ resp. „Open Source“ , môže dovoľovať aj úpravu zdrojového kódu. Užívateľ, ktorý sa v potrebnej problematike vyzná, si môže tak do aplikácie dorobiť vlastnú funkcionalitu, poprípade upraviť existujúcu podľa svojich potrieb.

4. Problematika navigačných systémov a možnosti spracovania GPS signálu

Už od nepamäti bolo nutné nejakým spôsobom vedieť určiť polohu na zemskom povrchu. V minulosti sa jednalo hlavne o určovanie polohy a navigovanie lodí. Pri absencií dnešnej modernej techniky sa uplatňovala navigácia podľa nebeských telies.

Hlavná myšlienka v tomto type navigácie spočíva v meraní uhla medzi viditeľným nebeským telesom a horizontom. Touto metódou je možne zistiť polohu na mori aj na súši. V danom časovom okamžiku sa nachádza dané nebeské teleso presne nad jedným konkrétnym geografickým miestom. Toto miesto je definované podľa zemepisnej šírky a dĺžky. Všetky telesá, pomocou ktorých je možne určovať polohu, boli zaznamenané v námornom almanachu. V tomto almanachu sú uvedené aj presné rozloženia telies na daný kalendárny rok [6].

V dnešnej dobe sa navigačné systémy využívajú v značne vyššej miere ako v minulosti. Už asi len málokto bude využívať navigáciu pomocou nebeských telies. V súčastnosti sa najčastejšie používajú družicové navigačné systémy.

4.1. Družicové navigačné systémy

Tento systém pracuje na princípe prijímania signálu z umelej družice. Prijatá informácia okrem iného obsahuje časové razítko a identifikátor družice. Prijímacie zariadenie obsahuje zoznam družíc a tzv. pozičných súradníc družíc. Na základe signálu z viacerých družíc je možné prepočtom času prijatia signálu a časového razítka v signále, ako aj pomocou pozičných súradníc, určiť polohu prijímacieho prístroja s presnosťou niekoľko metrov.

Na základe vyhodnocovania polohy môžeme prijímače rozdeliť do týchto skupín:

- autonómny družicový prijímač (pre určenie polohy prijíma signál z družíc, presnosť merania je daná počtom zachytených družíc),
- diferenciálny družicový prijímač (sústava dvoch alebo viacerých prijímačov, jeden z nich je umiestený na pevnej, respektíve dobre známej pozícií a slúži ako základňa pre ostatné prijímače).

Známe družicové systémy:

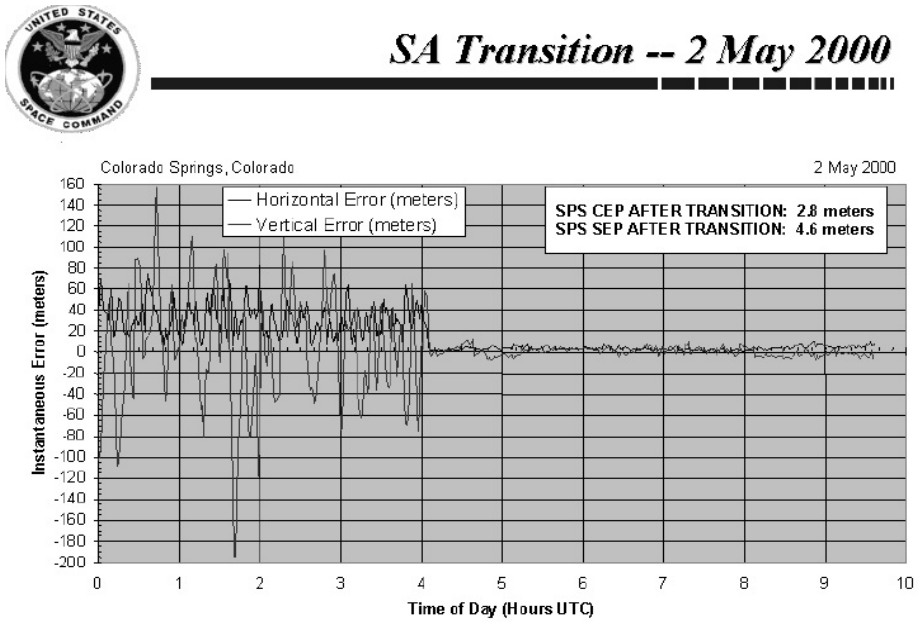
- Transit: prvý satelitný navigačný systém vyvinutý v roku 1958. Výpočet polohy prebieha prostredníctvom Dopplerovského javu[5].
- Navstar GPS: navigačný systém vyvíjaný armádou USA. Prvý krát bol spustený v roku 1973 a funguje dodnes.
- Glonass: ruský navigačný systém. Výpočet polohy je podobný systému GPS. Prvá družica bola vypustená v októbri 1982.
- Galileo: v súčasnej dobre ešte stále vyvíjaný a nie plne funkčný navigačný systém. Má slúžiť pre potreby EU. Systém má byť podobný GPS. Má sa stať jeho alternatívou.

5 Dopplerovský jav – popisuje zmenu frekvencie a vlnovej dĺžky medzi vysielaným a prijímaným signálom, spôsobenú nenulovou vzájomnou rýchlosťou vysielača a prijímača.

4.2. Global Positioning System

Systém má oficiálne pomenovanie NAVSTAR - GPS (Navigation Satellite Timing and Ranging Global Positioning System). Počiatky siahajú do roku 1973, kedy boli vypustené prvé štyri družice. Postupne na obežnú dráhu putovali ďalšie a ďalšie družice. V roku 1993 bolo prvý krát dosiahnuté trojrozmerné zameriavanie v a roku 1995 došlo o oficiálnemu vyhláseniu plnej operačnej spôsobilosti systému - FOC[6] [6].

Primárne bol GPS vyvíjaný na vojenské účely. Jednalo sa o lokalizáciu a navigáciu vojenských jednotiek, navádzanie rakiet, zameriavanie cieľu a podobne. V 80. rokoch 20. storočia americká vláda rozhodla o umožnení využívania GPS pre civilný sektor. Navigačný systém sa rýchlo rozšíril a našiel uplatnenie v mnohých sférach. V roku 1990 však prichádza k aktivácií mechanizmu, ktorý zámerne do signálu určeného pre civilné účely vnáša chybu. Nazýva sa selektívna dostupnosť resp. Selective Availability – SA. Dôvodom zavedenia SA bolo zaistenie bezpečnosti USA a zabránenie zneužívaniu navigačného systému teroristickým skupinám alebo vtedajším nepriateľom USA. Spôsobuje chybu určovania polohy až 100m. SA bola deaktivovaná v máji roku 2000. Od tohto obdobia sa GPS intenzívne využíva aj v civilnej oblasti. Na GPS je dnes závislých mnoho systémov. Treba si však uvedomiť, že k opätovnému spusteniu SA môže dôjsť aj v tomto období. Zmenu chyby pred a po vypnutí SA v signále vykresľuje nasledujúci graf [7]:



Obrázok 6: Chyba v GPS signále pred a po vypnutí SA

6 FOC - Full Operational Capability – plná operačná spôsobilosť.

4.2.1. Štruktúra a princíp fungovania GPS

Celý systém je tvorený tromi základnými segmentami:

- kozmický,
- riadiaci,
- užívateľský [7].

4.2.1.1. Kozmický segment

Tento segment je tvorený sústavou umelých družíc systematicky rozložených na obežných dráhach. Týchto kruhových dráh je 6 a každá z dráh má 5 pozíc pre umiestenie družice. Z toho vyplýva, že maximálny počet družíc je 30. Pozícia č. 5 je na každej dráhe určená ako záložná. K dosiahnutiu FOC je teda nutných minimálne 24 družíc. Družice sú od zemského povrchu vzdialené 20190km a pohybujú sa rýchlosťou 11300km/h. Sklon obežných dráh je 55° vzhľadom k rovníku. Jeden obeh okolo zeme trvá 11 hodín a 58 minút. Dá sa povedať, že družica je každý deň na tom istom mieste o 4 minúty skôr [6].

Na každom mieste zeme by mal byť dostupný signál minimálne zo 4 družíc. Väčšinou to však býva viac. Rovnaká konštalácia družíc sa nad daným miestom zopakuje približne čo 24 hodín. Kozmický sektor je veľmi stabilný, preto je možné dobre modelovať a predvídať jeho chovanie, resp. určovať presné parametre obežných dráh [7].

Každá družica obsahuje veľmi presné atómove hodiny. Starajú sa o frekvenčnú stabilitu frekvenčného signálu. Vytvorené sú 2 základné nosné signály:

- L1 = 1 575,42 MHz ,
- L2 = 1 227,60 MHz .

Každý vysielaný signál je kombináciou nosnej frekvencie, diaľkomerného kódu a navigačnej správy. Frekvencia L1 je modulovaná dvoma diaľkomernými kódmi:

- P - kód (presný kód, môže slúžiť pre vojenské účely, šifrovaná varianta sa nazýva Y- kód),
- C/A - kód (dostupný kód určený pre civilné využitie).

Frekvencia L2 je modulovaná len P – kódom alebo jeho šifrovanou variantou Y – kódom. Civilné prijímače spracúvajú frekvenciu L1 a C/A kód. Tieto spracovávané signály sa označujú ako štandardné polohovacie služby. Príjem dvojice frekvencií L1 a L2 sa označuje ako presná polohová služba. Využívajú ju špeciálne vybavené vojenské prijímače [7].

4.2.1.2. Riadiaci segment

Tento segment zodpovedá za riadenie celého navigačného systému. Pozostáva z týchto častí:

- 5 pozemných monitorovacích staníc,
- 1 hlavná riadiaca stanica (1 záložná riadiaca stanica),
- 3 stanice pre komunikáciu s družicami.

Pozemné monitorovacie stanice sú umiestené vo veľkých vojenských a leteckých základniach Havaj, Kwajalein, Diego García, Ascension a Colorado Springs. Tieto stanice sú bezobslužné, riadi ich hlavná riadiaca stanica. V podstate sa jedná o veľmi kvalitné a presné GPS prijímače, ktoré naviac obsahujú atómové hodiny. Sú schopné sledovať všetky aktuálne viditeľné GPS družice. Neprevádzajú žiadne spracovanie dát. Zachytené údaje sa prenášajú do hlavnej riadiacej stanice. Ich úloha by sa dala definovať ako sledovanie kozmického segmentu [7].

Hlavná riadiaca stanica sa nachádza v Colorade na leteckej základni Schriver (Schriver Air Force Base). Je situovaná v podzemí a chránená proti jadrovým hrozbám. Stanica na základe dát z monitorovacích staníc prepočítava presné údaje obežných dráh a korekcie atómových hodín pre jednotlivé stanice. Jej úloha spočíva aj v riadení kozmického segmentu. Potrebné údaje sú prenášané na stanice komunikujúce s družicami [7].

Úlohou staníc pre komunikáciu s družicami je odosielanie údajov družiciam o ich obežných dráhach, korekcia hodín a aktualizácia navigačných správ. Umiestnené sú na vojenských základniach Kwajalein, Diego García a Ascension [7].

4.2.1.3. Užívateľský segment

Užívateľský segment sa skladá z GPS prijímačov jednotlivých užívateľov. Tieto prijímače na základe prijatých signálov z družíc prevádzajú výpočet polohy a času. Aby bol výpočet všetkých súradníc (x,y,z,t) korektný, je potrebné prijímať signál aspoň zo 4 družíc. GPS prijímače nachádzajú najväčšie využitie v určovaní polohy, zameriavaní plochy, navádzaní na cieľ, zisťovaní presného času a veľa iných.

4.3. Komunikačný protokol NMEA 0183

NMEA[7] 0183 je štandard, ktorý popisuje a špecifikuje komunikáciu hlavne medzi námornými zariadeniami ako sonar, gyroskop, anemometer, GPS prijímače a mnoho iných zariadení. Informácie prenáša prostredníctvom jednoduchého ASCII protokolu. NMEA je určená pre jednosmerný sériový prenos. Zbernica komunikuje rýchlosťou 4800 buadov. Každá prenášaná správa začína výlučne znakom $. Samotná informačná časť správy obsahuje numerické aj alfanumerické údaje. Údaje sú oddelené čiarkou. Na konci správy obvykle nájdeme hviezdičku a kontrolný súčet. Správy od seba oddelené znakom konca riadku [8].

4.3.1. Spracovanie prijatých dát

Všeobecný tvar prijatej správy má formu: `$ttsss,d1,d2,....<CR><LF>`

Prvé dve písmená definujú vysielajúceho (talker identifier). Ďalšie tri písmená identifikujú správu (sentence identifier). Pre GPS prijímače je identifikácia zariadenia určená znakmi GP. Príklad MNEA vety:

```
$GPGSA,A,3,29,26,22,09,07,05,04,,,,,,1.7,1.0,1.4*30
```

Základné MMEA správy pre GPS navigácie sú:

- Veta RMC – minimálna doporučená správa pre navigáciu,
- Veta GSV – informácie o družiciach,
- Veta GGA – zemepisná. šírka a dĺžka, geodetická výška, čas určenia súradníc.

7 National Marine Electronics Association

Software, ktorý na vstupe očakáva GPS dáta, musí potrebné údaje vyextrahovať priamo zo správ prichádzajúcich z GPS prijímača. Väčšinou sa tu využíva skutočnosti, že sú dáta v správach oddelené čiarkami a jednotlivé dáta majú v správach presne definovanú pozíciu. Na základe týchto informácií sa zostaví parser, ktorý hodnoty predá aplikácií pre ďalšie spracovanie.

Pre určité programovacie jazyky existujú aj hotové knižnice, ktoré parsovanie vykonávajú. Programátor sa tak nemusí starať o detaily parsovania dát. Príkladom takýchto knižníc sú:

- pre jazyk C++ – NMEA Library
- pre jazyk Java – NMEA Java Library
- pre jazyk C# – GpsTalk

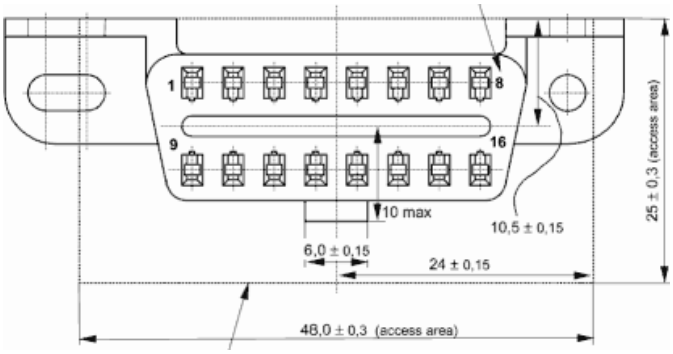
Faktické údaje z kapitoly 4.3.1. čerpajú zo zdroja [8].

5. Štandard OBD II

Kapitola 2. informuje o základných vlastnostiach používaných diagnostických systémov a protokolov v automobiloch. V tejto časti bude bližšie priblížené fungovanie štandardu OBD II. Ako už bolo uvedené, v EU je povinnosť podporovať štandard OBD II od roku 2001 resp. 2004 u dieslových motorov.

5.1. OBD II konektor a fyzická vrstva

Pre komunikáciu s riadiacou jednotkou vozidlá kompatibilné s normou OBD II využívajú štandardizovaný 16 pinový konektor (štandard SAE J1962).



Obrázok 7: Konektor OBD II

Konektor býva zvyčajne umiestený tak, aby bol prístupný vodičovi. Najďalej 50 cm od volantu. Najtypickejšie umiestenia sú:

- pod palubnou doskou medzi stredovou konzolou a vodičovými dverami,
- pod označeným krytom,
- pod alebo za popolníkom alebo blízko zapaľovača na stredovej konzole.

Treba podotknúť, že prítomnosť tohto konektora vo vozidle automaticky neznamená podporu OBD II štandardu. Napríklad tento konektor bol montovaný už v Škode Felícia, no štandard OBD II nepodporovala ešte ani prvá generácia Octávie [1].

5.1.1. Komunikačné protokoly

Pre jednotnosť komunikácie boli definované komunikačné protokoly. Tie špecifikujú formát komunikácie, rýchlosť linky, napäťové úrovne a podobne. Komunikačný protokol sa využíva na komunikáciu prevodníka OBD a riadiacej jednotky. Dá sa teda povedať, že prítomnosť určitého protokolu nezaručuje v každom prípade aj prítomnosť štandardu OBD II. Medzi najznámejšie protokoly patria:

- ISO 9141-2
- ISO 14230-4 (KWP2000)
- SAE J1850 PWM

- SAE J1850 VPW
- ISO 15765-4 CAN
- SAE J2248 CAN

Pre každý typ protokolu je definované iné využite pinov konektora OBD II.

PIN	ISO	SAE PWM	SAE VPW	ISO / SAE CAN
2		bus +	bus +	
4	tienenie	tienenie	tienenie	tienenie
5	vbat-	vbat-	vbat-	vbat-
6				can- h
7	k- line			
10		bus -		
14				can- l
15	l- line			
16	vbat+	vbat+	vbat+	vbat+

Tabuľka 2: Úloha pinov v konektore OBD II pri rôznych protokoloch

Je zrejmé, že v každom protokole je totožné rozloženie napájacích pinov (označených vbat+, vbat-, tienenie). Ako napájacie napätie je použité priamo napätie z batérie vozidla (12V) [1].

5.2. Protokol SAE[8] J1979

OBD II využíva pre komunikáciu správy, ktoré boli štandardizované podľa normy SAE J1979. Tento štandard definuje spôsob a formát zasielania žiadostí a formát odpovedí. SAE J1979 definuje žiadosť často označovanú ako PID[9]. Existuje kompletný zoznam PIDs. Výrobcovia nemusia podporovať všetky PID a môže implementovať aj svoje vlastné [1].

Celý postup komunikácie sa dá všeobecne popísať takto:

- zadanie PID,
- skenovacie zariadenie odošle informáciu na zbernicu vozidla (CAN, PWM, VPW, ISO, KWP),
- zariadenie na zbernici určí, že PID je podporované a výslednú hodnotu zapíše na zbernicu,
- skenovacie zariadenie vyhodnotí a spracuje hodnotu na výstupe[1].

8 Surface Vehicle standard – SAE J1979 – Ekvivalent k štandardu ISO 15031-5 (Road vehicles- Communication between vehicle and external equipment for emissions-related diagnostics – Part 5: Emissions-related diagnostic services).

9 Parameter identification numbers – pomocou PID sú adresované dotazy na riadiacu jednotku.

5.2.1. PIDs

5.2.1.1. Skupiny PIDs

Podľa štandardu SAE je definovaných desať prevádzkových režimov resp. skupín, do ktorých sa jednotlivé PID zaraďujú. Sú označované prefixom „0x“ a za ním nasleduje poradové číslo v hexa tvare napr.: 0x05 [1].

Označenie	Popis
0x01	Čítanie aktuálnych dát.
0x02	Čítanie tzv. „Freeze“ údajov. Sú to dáta, ktoré boli zaznamenané práve v tom čase, kedy došlo vo vozidle k poruche alebo nastala chyba.
0x03	Čítanie uložených DTC[10] kódov.
0x04	Mazanie DTC a uložených hodnôt.
0x05	Výsledky monitorovania testov na kyslíkových senzoroch (neplatí pre zbernicu CAN).
0x06	Výsledky monitorovania testov pre iné systémy alebo systémový monitoring (pre zbernicu CAN zahŕňa testy kyslíkových senzorov).
0x07	Čítanie nevyriešených DTC.
0x08	Ovládanie palubných komponentov a systémov.
0x09	Čítanie informačných údajov o vozidle.
0x0A	Čítanie permanentných resp. vymazaných DTC.

Tabuľka 3: Skupiny PIDs [1]

5.2.1.2. Zoznam PIDs

Nasledujúca tabuľka uvádza PIDs, ktoré boli vybrané a implementované v tomto projekte. Kompletný zoznam (neuvádzaný, kvôli veľkej obsahovosti) s podrobným vysvetlením je možné nájsť v literárnom zdroji [1].

Mode	PID	Bajty v odpovedi	Popis PID	Min	Max	Jedn.	Prevodový vzorec
01	04	1	Calculated engine load value	0	100	%	A*100/255
01	05	1	Engine coolant temperature	-40	215	°C	A-40
01	06	1	Short term fuel % trim- Bank 1	-100	99,2	%	(A-128) * 100/128
01	07	1	Long term fuel % trim- Bank 1	-100	99,2	%	(A-128) * 100/128
01	0A	1	Fuel pressure	0	765	kPa	A*3
01	0B	1	Intake manifold absolute	0	255	kPa	A

10 Diagnostic trouble code – kód chyby diagnostického systému napr.: P0130 - Sensor Circuit Malfunction.

			pressure				
01	0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
01	0D	1	Vehicle speed	0	255	km/h	A
01	0E	1	Timing advance	-64	63,5	°	A/2 - 64
01	0F	1	Intake air temperature	-40	215	°C	A-40
01	10	2	MAF air flow rate	0	633,35	g/s	((A*256)+B) / 100
01	11	1	Throttle position	0	100	%	A*100/255
01	23	2	Fuel Rail Pressure (diesel)	0	655350	kPa	((A*256)+B) * 10
01	2C	1	Commanded EGR	0	100	%	100*A/255
01	2F	1	Fuel Level Input	0	100	%	100*A/255
01	46	1	Ambient air temperature	-40	215	°C	A-40

Tabuľka 4: PIDs použité v tomto projekte[1]

Odpovede na dotazy môže obsahovať 1 až 4 bajty. Tie sú označené do blokov a, b, c. Tieto premenné sa použijú na výpočet hodnoty a je ich možné vidieť v časti tabuľky prevodový vzorec.

5.2.1.3. Počiatočné preverovanie podporovaných PIDs

Pomocou jedného dotazu je možné zistiť podporu pre 32 po sebe idúcich PIDs. Pri poslaní požiadavky je vždy počet vrátených bajtov 4. Nasledujúca tabuľka ukazuje priradenie skupín PID k dotazom na zistenie podpory.

Dotaz	**Poradie podporovaných PIDs (v hex)**
0100	0 až 20
0120	21 až 40
0140	41 až 60

Tabuľka 5: PIDs dotazy na zistenie podpory[1]

Napríklad na dotaz `0100` môže prísť odpoveď v tvare `41 00 BE 3E A8 11`

Bajt `41` hovorí, že sa jedná o odpoveď. Konkrétne hodnota `4` znamená odpoveď a hodnota `1` identifikuje skupinu, tak ako sú uvedené v Tabuľka 3. Ďalšie 4 bajty nesú informáciu o podporovaných PIDs. Postup je nasledovný:

- prevedieme informačné bajty, ktoré sú v tvare hex na binárny tvar, v našom prípade `BE 3E A8 11`, čoho binárny ekvivalent je: `10111110001111101010100000010001`,
- binárny formát ma 32 pozícií s hodnou 0 alebo 1. Pozícia v binárnom reťazci je zhodná s pozíciu PIDs v kompletnom zozname. Hodnota 1 znamená, že PID je podporované, naopak hodnota 0 značí, že PID podporované nie je.

5.2.1.4. Proces získavania dát

Po zistení podporovaných PIDs sa môže pristúpiť k posielaniu dotazov. Zadávajú sa v tvare Mode + PID tak ako je to naznačené v Tabuľka 4 ako jedena celistvá sekvencia znakov. Po zadaní dotazov môžeme očakávať odpoveď v nasledujúcom tvare:

- prvý bajt – identifikácia odpovede,
- druhý bajt – identifikácia dotazu,
- ostatné bajty – užitočné dáta.

Napríklad po zadaní dotazu: `0105` môžeme očakávať odpoveď v tvare : `41 05 63`. Bajt `41` hovorí, že sa jedná o odpoveď. Konkrétne hodnota `4` znamená odpoveď a hodnota `1` identifikuje skupinu, tak ako sú uvedené v Tabuľka 3.

Druhý prijatý bajt `05` jednoznačne identifikuje dotaz, pre ktorý vznikla táto odpoveď (`0105`). Z Tabuľka 4 je zrejmé, že na tento dotaz sa prijíma užitočná odpoveď, ktorá má 1 bajt. Keďže je odpoveď jednobajtová, bude označená ako formula A. Pri viacbajtových odpovediach sa pokračuje označovaním B, C. Výpočet požadovanej hodnoty prebieha nasledovným spôsobom:

- z Tabuľka 4 sa vyberie konkrétny PID (pomocou druhého bajtu v odpovedi),
- určia sa formule podľa počtu bajtov A, B, C,
- jednotlivé formule sa prevedú na decimálny tvar,
- výpočet hodnoty prebieha podľa vzorca uvedeného v Tabuľka 4.

Pre uvedený príklad je hex hodnota informačného bajtu `63` z toho sa vypočíta decimálna hodnota, čo je `99`. Dá sa povedať, že formula A má hodnotu `99`. Vzorec uvedený pri tomto PID je: `A- 40`. Z toho vyplýva, že výsledná hodnota je 59. Teda teplota chladiacej kvapaliny bola pri tomto meraní 59°C.

5.3. Prevodník ELM 327

Keďže rozhranie pre pripojenie diagnostických zariadení v motorových vozidlách nie je priamo kompatibilné so známymi rozhrania v počítači, musia sa pre prenos dát používať prevodníky. Prevodník značky ELM prešiel mnohými vývojovými zmenami, aktuálna verzia je ELM 327.

Tento prevodník je navrhnutý tak, aby vytváral premostenie medzi OBD rozhraním v motorovom vozidle a štandardným sériovým portom v počítači. K dispozícií je vyhotovenie so štandardným sériovým portom alebo Bluetooth.

Hlavné rysy prevodníka sú:

- práca v pohotovostnom režime,
- rýchlosť sériovej linky 500 kbps,
- automatické vyhľadanie použitého protokolu,
- konfigurácia pomocou AT príkazov,
- nízka spotreba energia vďaka CMOS čipu.

Obrázok 8: Prevodník ELM 327

5.3.1. Komunikácia s ELM 327

Pre správne fungovanie komunikácie je nutné nastaviť sériový port nasledovne:

- rýchlosť prenosu 9600 alebo 38400 buadov,
- 8 dátových bitov,
- žiadna parita,
- 1 stop bit.

Pre preverenie komunikácie sa môže použiť štandardná aplikácia, ktorá umožňuje textovú komunikáciu po sériovom porte, napríklad Hyperterminal alebo Putty. Ak je zariadenie pripojené správne, zobrazí sa znak > . Znamená ta, že prevodník je v kľudovom stave a je pripravený na komunikáciu. Prevodník nerozlišuje medzi veľkými a malými písmenami (nie je case sensitive) a ignoruje medzery a riadiace znaky.

Príkazy určené pre konfiguráciu samotného prevodníka začínajú sekvenciu znakov `AT` . Ukážkou jednoduchého AT príkazu môže byť `AT RV`. Tento dotaz vráti hodnotu vstupného napätia na prevodníku. Môže sa očakávať výsledok typu: `12.6V` .

Príkazy, ktoré sú určené pre OBD sa konštruujú tak, ako je to uvedené v kapitole 5.2.1.4.Proces získavania dát. Prevodník ich identifikuje tak, že nezačínajú sekvenciu AT. Najskôr prebehne overovanie platnosti zadaného príkazu a potom vloženie do dátových paketov. Niektoré protokoly vyžadujú aj kontrolný súčet. To zabezpečí prevodník automaticky. Väčšina OBD dotazov má dĺžku jeden alebo dva bajty, môže ich byť aj viac. Prevodník dovolí poslať dotaz o dĺžke maximálne 7 bajtov. Ak bude dotaz väčší, bude ho ignorovať a vráti znak „`?`“ . Po odoslaní dotazu bude prevodník čakať na zbernici a po načítaní vyhovujúcej hodnoty je naspäť odošle na sériový port. Ak je dotaz nepodporovaný, bude ho ignorovať a vráti reťazec „`NO DATA`“ . Prevodník podporuje všetky skupiny PIDs podľa štandardu SAE J1979 tak ako je uvedené v Tabuľka 3. Zariadenie je možné uviesť na továrenské nastavenia pomocou príkazu: „`AT Z`“ .

5.3.2. Nastavenie a výber komunikačného protokolu

Prevodník ELM 327 podporuje niekoľko rôznych typov protokolov pre komunikáciu s riadiacou jednotkou vozidla. Zariadenie je továrensky nastavené na automatický výber. Protokoly sú uvedené v nasledujúcej tabuľke.

Protokol	**Popis**
0	Automatický výber
1	SAE J1850 PWM (41.6 kbaud)
2	SAE J1850 VPW (10.4 kbaud)
3	ISO 9141-2 (5 baud init)
4	ISO 14230-4 KWP (5 baud init)
5	ISO 14230-4 KWP (fast init)
6	ISO 15765-4 CAN (11 bit ID, 500 kbaud)
7	ISO 15765-4 CAN (29 bit ID, 500 kbaud)
8	ISO 15765-4 CAN (11 bit ID, 250 kbaud)
9	ISO 15765-4 CAN (29 bit ID, 250 kbaud)
A	SAE J1939 CAN (29 bit ID, 250 kbaud)
B	User1 CAN (11 bit ID, 125* kbaud)
C	User2 CAN (11 bit ID, 50* kbaud)

Tabuľka 6:Podporované protokoly v ELM 327 pre komunikáciu s riadiacou jednotkou

Výpis aktuálne nastaveného protokolu sa prevádza pomocou príkazu: „`AT DP`“ . Odpoveď bude názov protokolu alebo pri automatickom výbere „`auto`“ .

Nastavenie protokolu prebieha prostredníctvom príkazu: AT SP + označenie protokolu v tabuľke, napríklad: „`AT AP 0`“ nastaví výber na automatický. Je možné nastaviť aj 2 preferované protokoly napríklad: „`AT AP A0`“ kedy bude nastavený automatický výber aj protokol SAE J1939 CAN. Odpoveďou na nastavenie protokolu je sekvencia znakov „`OK`“ . Ručné nastavenie môže byť výhodné, ak je prevodník stále používaný v jednom vozidle (protokol sa nemení). Ušetrí sa tým čas inicializácie, respektíve vyhľadávania podporovaného protokolu pri automatickom výbere.

Kompletná dokumentácia k prevodníku ELM 327 je dostupná ako literárny zdroj [9]. Z tohto zdroja čerpá aj celá kapitola 5.3.Prevodník ELM 327.

6. Využitie Google Maps a Google Earth pre zobrazenie dát

V súčastnosti existuje viac mapových systémov, ktoré poskytujú online mapové podklady oblasti, pohľad na terén, letecké snímky či snímky ulíc. Patrí k nim samozrejme aj Google Maps.

Google Maps je technológia mapových služieb od spoločnosti Google. Je k dispozícií zdarma pre nekomerčné použitie. Poskytuje prepracované a intuitívne užívateľské rozhranie. Medzi bežné využívané funkcie systému patrí hlavne detailné zobrazenie cestnej infraštruktúry, plánovanie cesty medzi dvoma bodmi či už pešo alebo pomocou dopravných prostriedkov, zobrazenie terénu krajiny, zobrazenie satelitných snímkov [10].

Google Earth sa dá popísať ako virtuálny glóbus. Pôvodný tvorca software je spoločnosť Keyhole, Inc. V roku 2004 ho odkúpila spoločnosť Google. V súčasnej dobe je dostupný pod dvoma rôznymi licenciami:

- Google Earth – bezplatná verzia s obmedzenou funkčnosťou,
- Google Earth Pro – platený variant pre komerčné využitie.

Aplikácia Google Earth je dostupná pre všetky v súčastnosti najviac využívané desktopové platformy ako systémy typu Windows, MacOS, Linux, FreeBSD. Dostupná je aj pre mobilné platformy ako iPhone, Blackberry a Android. Veľkou výhodou je dostupnosť zásuvného modulu webové prehliadače. Zásuvný modul je dostupný od mája roku 2008. Zobrazovanie máp prostredníctvom zásuvného modulu v prehliadači je použité aj v tomto projekte.

6.1. Google Maps API

Google Maps disponuje širokou škálou rozhraní API. Pomocou týchto rozhraní je možné vo všeobecnosti spracovávať dáta a tie následne zobraziť vo webovej aplikácií alebo stránke. Google rozdeľuje API do piatich základných vrstiev:

- **API pre JavaScript** – riešenie vkladania mapy do webovej aplikácie prostredníctvom JavaScriptu. Aktuálna verzia tohto API je 3. Je navrhnuté tak, aby pracoval korektne aj na mobilných platformách. Poskytuje základné operácie nad mapami ako vyznačenie miesta, vkladanie bublinových textov, vykresľovanie geometrických tvarov, vyznačovanie ciest a podobne.
- **API pre Flash** – API je podobné predchádzajúcemu. Je však určené pre webové aplikácie a stránky využívajúce technológiu Flash.
- **Google Earth API** – umožňuje pracovať so skutočným 3D modelom zeme a ten umiestniť na webové stránky. Spracovanie prebieha prostredníctvom JavaScriptu. Je možné pracovať so súbormi typu KML[11] a zobrazovať dáta rôzneho typu.
- **API pre statické mapy** – jedná sa o rýchlu a výkonovo nenáročnú implementáciu mapy. Nie je použitý žiaden skript pre zobrazenie dát. Mapa je kompletne statická, teda neexistuje žiadne dynamické načítavanie. Interpretácia dát je riešená len za pomoci URL adresy. Je možné vybrať druh mapy, štýl mapy, veľkosť, označiť miesto na mape, vyznačiť cestu podľa súradníc a podobne.

11 Keyhole Markup Language – formát súboru, ktorý slúži na zobrazenie geografických dát v aplikácií Google Earth.

- **API pre webové služby** – Google Maps API poskytuje rozhranie pre webové služby. Webové služby využívajú HTTP požiadavky na konkrétne URL adresy, prostredníctvom ktorých sa odovzdávajú argumenty služby [11].

6.1.1. API pre JavaScript

Toto API využíva tzv. API kľúč. Kľúč je možné zdarma získať prostredníctvom webovej stránky code.google.com/apis/maps/signup.html. Podmienkou získania kľúča je užívateľský účet v systéme Google. Pri generovaní je nutné zadať doménu, na ktorej bude mapa umiestená. Je možné kľúč generovať aj pre lokálne účely resp. lokálne domény. Príklad vkladanie API kľúča je v uvedený v Zdrojový kód 1: Vloženie API kľúča.

```
<script type="text/javascript" src="http://www.google.com/jsapi?
key=API_KLUC"></script>
```

Zdrojový kód 1: Vloženie API kľúča

Vykreslenie jednoduchej mapy do webovej stránky môže prebiehať nasledujúcim spôsobom:

Na začiatku musí byť vložený kľúč s API. V ďalšom kroku musí byť načítané Google API: `google.load("maps", "2");` Metóda `load` má dva parametre. Prvý parameter je názov API a druhý je jeho verzia. Voliteľne je možné použiť parametre pre nastavenie lokalizácie a podobne.

Ďalej nasleduje funkcia, ktorá tvorí samotnú mapu. Vytvorí sa objekt mapy. Ďalej sa nastaví stred a priblíženie na mape. V ďalšej časti sa predchádzajúca funkcia zavolá a ukončí sa časť JavaScriptu.

```
function GoogleMAPA() {
var mapa = new
google.maps.Map2(document.getElementById("mapa"));
mapa.setCenter(new google.maps.LatLng(49.79545,15.732422), 7);
}
google.setOnLoadCallback(GoogleMAPA);
```

Zdrojový kód 2: Vytvorenie mapy

Mapa sa v časti HTML kódu vkladá do blokového elementu DIV, kde je možné nastaviť veľkosť mapy. Predchádzajúci príklad poukazoval na najjednoduchšie vykreslenie mapy vo webovej stránke. Samotné Google API poskytuje široké možnosti, ako napr. zisťovanie kompatibility prehliadača, zmena typu zobrazovanej mapy (klasická, satelitná, kombinácia klasickej a satelitnej, terén). API umožňuje do mapy vkladať aj okno s HTML kódom.

K dispozícii je značné množstvo informačných a koordinačných metód ako funkcie na získavanie aktuálne zobrazených častí mapy, zisťovanie priblíženia, vystredenie mapy na zadané súradnice a pod.

Na mape je možné odchytávať udalosti. Zväčša sa môže jednať o klik myšou alebo stisk tlačítka. Následne k týmto udalostiam priradiť funkcie, ktoré budú na udalosti reagovať.

Pre lepšiu manipuláciu s mapou môžu byť pridané ovládacie prvky:

- panel pre ovládanie mapy (posun a priblíženie mapy),
- tlačítka pre ovládanie mapy (posun a priblíženie mapy),

- tlačidlá + a – pre nastavenia úrovne priblíženia,
- nastavenie mierky mapy,
- pridanie zmenšeniny mapy do rohu mapy,
- tlačidlá pre prepínanie typu mapy (klasická, satelitná, kombinácia klasickej a satelitnej, terén).

Informácie o implementácií vyššie uvedených alebo iných funkcií či rôznych prispôsobení máp je možné nájsť v literárnom zdroji [12], odkiaľ aj čerpajú údaje z tejto kapitoly.

6.1.2. Google Earth API

Google Earth API je ovládané podobne ako API pre JavaScript. Veľký význam tohto API sa nesie s príchodom Google Earth pluginu pre webové prehliadače. To rozširuje možnosti 3D modelu zeme nie len na použitie v desktopovej aplikácií, ale posúva sa smerom k webovým aplikáciam a stránkam.

API poskytuje minimálne také možnosti ako API pre JavaScript, respektíve vkladať do mapy rôzne značky, kresliť čiary a geometrické tvary, vkladať 3D modely. Značné využitie má práca z KML súbormi.

Vloženie jednoduchej mapy typu Google Earth je podobné ako pri Google Maps. Pred samotným používaním Google Earth vo webovej aplikácií je nutné nainštalovať zásuvný modul pre webové prehliadače. Modul je dostupný na stránkach projektu. Na začiatok je opäť nutné vložiť skript, ktorý definuje API kľúč tak ako to popisuje Zdrojový kód 1: Vloženie API kľúča.

V ďalšom kroku musí byť načítané Google API: `google.load("earth", "1");` Metóda `load` má dva parametre. Prvý parameter je názov API a druhý je jeho verzia. Voliteľne je možné použiť parametre pre napr. pre nastavenie lokalizácie a podobne.

Základný rozdiel v implementácií oproti JavaScript API je definovanie inicializačných funkcií. Je vytvorená metóda, ktorá má za úlohu vytvoriť inštanciu pluginu. Ďalej sú definované metódy, ktoré sú spúšťané, ak sa inštancia pluginu podarí alebo naopak. V ďalšej časti sa predchádzajúca funkcia `init` zavolá a ukončí sa časť JavaScriptu. Funkcie popisuje nasledujúci zdrojový kód.

```
function init() {
   google.earth.createInstance('map3d', initCB, failureCB);
}
function initCB(instance) {
   ge = instance;
   ge.getWindow().setVisibility(true);
}

function failureCB(errorCode) {
}
google.setOnLoadCallback(init);
```

Zdrojový kód 3: Vytvorenie Google Earth mapy

Mapa sa v časti HTML kódu vkladá do blokového elementu DIV, kde je možné nastaviť veľkosť mapy. Google Earth API má podobné široké možnosti ako API pre JavaScript.

Informácie o implementácií vyššie uvedených alebo iných funkcií či rôznych prispôsobení máp je možné nájsť v literárnom zdroji [13], odkiaľ čerpajú aj údaje z tejto kapitoly.

6.1.2.1. KML

Jedná sa o typ súboru, prostredníctvom ktorého sa v systéme Google Earth zobrazujú geografické dáta. Aktuálna verzia je 2.2. KML bol vyvinutý spoločnosťou Keyhole. Formát súboru používa štruktúru založenú na značkách s vnorenými prvkami a atribútami. Vychádza zo štandardu XML. Google Earth spracúva KML súbor podobne ako webový prehliadač zdrojový kód z webovej stránky.

Z KML súbora je možné do mapy načítavať bodky, čiary, mnohouholníky, obrázky, modely a podobne. S použitím KML súborov sa môže zdieľať resp. prenášať informácie o rôznych miestach, poprípade dátach k nim pridruženým. Prostredníctvom KML sa v praxi vizualizujú rôzne typy dát. Ako príklad sa dá uviesť vizualizácia počasia (predpovedné modely), pokrytie územia signálom z vysielača a podobne.

KML súbory sú často distribuované ako KMZ súbory. Jedná sa v podstate o typ ZIP archívu s príponou KMZ. V archíve sa nachádza samotný KML súbor a dodatočné multimediálne súbory napr. obrázky a podobne. Nasledujúci zdrojový kód ukazuje príklad štruktúry jednoduchého KML súbora.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>
      Attached to the ground. Intelligently places itself at the
      height of the underlying terrain.
    </description>
    <Point>
      <coordinates>-
122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Zdrojový kód 4: Príklad štruktúry KML súbora

Pomocou tohto súbora sa na mape vykreslí jeden bod zo zadanými súradnicami, ktorý naviac obsahuje text v „bublinovej“ podobe. Kompletné informácie o možnostiach vyžitia KML súbora s prehľadnými príkladmi sú dostupné v literárnom zdroji [14], odkiaľ aj čerpajú informácie v tejto kapitole.

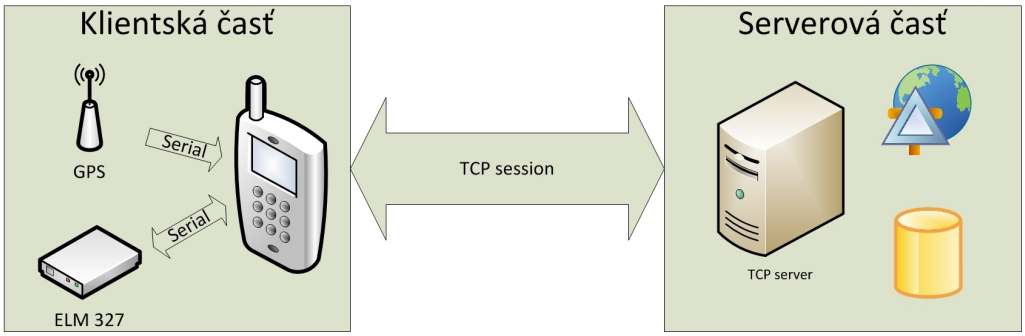
7. Vlastná realizácia

Vlastná realizácia sa skladá z dvoch vzájomne komunikujúcich častí, klient a server. Každý z nich plní svoju úlohu, má inú funkcionalitu, systémy sú na sebe nezávislé, implementované pre iné prostredia a komunikujú spolu cez sieťové rozhranie.

Zber a zobrazovanie aktuálnych dát, komunikáciu s GPS prijímačom a prevodníkom ELM 327 realizuje klientská časť. Je možné vybrať rôzne kombinácie zobrazovania a odosielania dát. Pri režime nahrávania je možné dáta ukladať offline do súbora a neskôr na server odosielať.

Server sa stará u ukladanie dát, vizualizáciu, autorizáciu konfiguráciu a podobne. Vizualizácia jazdy a údajov z vozidla je riešená v prostredí Google Maps a Google Earth. Užívateľské rozhranie je riešené prostredníctvom webového informačného systému. Komunikácia s klientskými aplikáciami je riešená prostredníctvom TCP socket servera.

Nasledujúce podkapitoly detailnejšie popisujú implementáciu a funkčnosť jednotlivých modulov a častí celého projektu.



Obrázok 9: Pohľad na moduly realizácie

7.1. Klientská časť

7.1.1. Stručný náčrt funkcionality

Aplikácia je určená pre mobilné zariadenie. Pre dosiahnutie plnej funkcionality je nutné, aby zariadenie, na ktorom aplikácia beží obsahovalo GPS prijímač a správne komunikovalo s prevodníkom ELM 237.

Komunikuje s GPS prijímačom, načítava správy NMEA, parsuje a spracúva ich. Konkrétne sa jedná o nadmorskú výsku, zemepisnú šírku, výšku a rýchlosť. Tieto údaje môžu byť zobrazené na display mobilného zariadenia alebo lokálne ukladané a odosielané na server na účelom ďalšieho spracovania.

Umožňuje zistiť zoznam podporovaných PID v danom vozidle. Z toho zoznamu dovoľuje užívateľovi vybrať konkrétne PID a následne zobraziť alebo odoslať na server podobne ako dáta z GPS.

Poskytuje užívateľovi určitú mieru nastavení. Jedná sa hlavne o prihlasovacie údaje, potrebné pre komunikáciu so serverovou časťou a nastavenie sériových portov pre komunikáciu s perifernými zariadeniami.

Pri odosielaní dát na server aplikácia dokáže pracovať aj v úplne odpojenom režime (dáta nahráva do súboru) alebo pri slabej konektivite.

7.1.2. Voľba platformy

Je zrejmé, že pre túto klientskú aplikáciu bude výhodné použiť niektorú z mobilných platforiem. Hlavné nároky pri výbere platformy, boli zariadenia, ktoré majú integrovaný GPS prijímač, umožňujú komunikovať na sériovom porte alebo prostredníctvom Bluetooth. Platforma by mala mať dobrú podporu ovládacích prvkov, rozšírenosť medzi užívateľmi. V neposlednom rade platforma musela umožniť nejakým spôsobom vykresľovať grafické elementy. Tie majú byť použité pre vykresľovanie aktuálnych údajov na display. Samozrejme programovací jazyk, v ktorom sa na tejto platforme vyvíja, by mal byť pre programátora príjemný a mal by riešiť potrebnú funkcionalitu rozumným spôsobom.

Rozhodol som sa pre platformu Windows Mobile vo verzií 6.5. Základ tohto systému tvorí Windows CE. Je to operačný systém určený pre zariadenia s obmedzeným výkonom a zdrojmi. Používa podmnožinu Win32 API. Je spustiteľný na procesoroch: Intel x86, MISP, ARM, Hitachi Super H. Pre Windows CE je možné použiť nasledovné programovacie prostredia a jazyky:

- vývoj natívnych aplikácií v jazyku C++,
- použitie platformy .NET Compact Framework (vývoj aplikácií pre túto platformu v programovacích jazykoch C# a Visual Basic),
- vývoj v jazyku JavaME (obmedzené funkcie a GUI).

Klientská aplikácia tohto projektu je implementovaná pre platformu .NET Compact Framework, konkrétne je napísaná v programovacom jazyku C#. Implantácia na tejto platforme má nasledovné rysy:

- relatívne jednoduchý vývoj,
- široké možnosti jazyka,
- veľká dostupnosť užívateľských ovládacích prvkov, podobne ako je to v desktopovej verzií,
- využite knižnice GDI+ pre prácu s grafikou – priamy prístup aplikácie k grafickým ovládačom zariadenia (využité v tomto projekte),
- aplikácia môže byť spúšťaná aj v prostredí Windows CE s nainštalovaným balíčkom .NET Compact framework. Nemusí sa teda jednať výlučne o mobilný telefón, ale môžu byť použité i rôzne priemyselné či embedded zariadenia ako napríklad GPS navigácie a iné,
- dobrá dokumentácia, veľká programátorská komunita.

Vývoj tejto časti projektu prebiehal vo vývojovom prostredí MS Visual Studio 2008. Najnovšia verzia MS Visual Studio 2010 vývoj aplikácie pre packet PC nepodporuje.

7.1.3. GUI

Celá aplikácia sa skladá z viacerých formulárov. GUI je tvorené hlavne z komponentov dostupných v .NET Compact Framework. Jedná sa hlavne o `Label`, `Button`, `TextBox`, `CheckBox` a podobne. Tieto komponenty sú použité prevažne v nastavovacích častiach aplikácie. Využívané sú aj vlastné komponenty `UserControl`, pomocou ktorých sú vyzualizované aktuálne namerané hodnoty.

7.1.3.1. Zobrazenie aktuálnych dát

V tomto projekte bolo nutné vybrať spôsob akým sa budú zobrazovať hodnoty či už z GPS prijímača alebo hodnoty z riadiacej jednotky vozidla. Rozhodol som sa pre štýl dobre známy z automobilov. Je to pozadie s číselníkom a ručička, ktorá ukazuje na aktuálnu hodnotu, teda „budíky“. Tie sú riešené ako `UserControl`. Dá sa povedať, že sa jedná o komponentu, ktorá je viacnásobne použitá a je možné ju prispôsobovať pre dané využitie. Základné vlastnosti a charakteristika by sa dala definovať ako:

- jednoduché použitie a nastavovanie parametrov,
- zamedzenie opakovanie kódu v projekte,
- export do DLL[12] knižnice a použitie komponenty aj v iných projektoch.

`UserControl` obsahuje Properties[13], tie slúžia na už spomínané prispôsobovanie komponenty pre požadované využitie. V tomto prípade sa hlavne jedná o atribúty veľkosť, rozsah hodnôt, aktuálna hodnota, jednotka, názov meranej veličiny, a podobne.

Komponenta sa skladá z viacerých častí:

- pozadie (číselník),
- názov meranej veličiny,
- ručička,
- centrálny bod,
- hodnota veličiny.

Ukážka komponenty vykreslenej vo formulári je dostupná v kapitole Prílohy ako Obrázok 16: Klientská aplikácia: meranie dát a Obrázok 17: Klientská aplikácia: meranie dát nastavenie senzorov.

Pre odstránenie nepríjemného blikania (tzv. flicking) pri prekresľovaní obrazovky je nutné využiť tzv. „double buffering[14]“. Princíp spočíva v tom, že obraz sa začne vykresľovať až keď je úplne kompletný. Kompletnosťou mám na mysli všetky súčasti komponenty, ktoré sú uvedené vyššie. Tento obraz potom nahradí predchádzajúci obraz ako celok. Keby sa double buffering nepoužil, dochádzalo by k blikaniu pri prekresľovaní ručičky, čo by bolo naozaj nepríjemné, pretože k zmenám zobrazovaných hodnôt môže dochádzať relatívne často, napr. pri rýchlosti či

12 Dynamic Link Library – interpretácia konceptu zdieľaných knižníc od spoločnosti Microsoft.
13 Vlastnosti sa v jazyku C# chovajú ako verejné položky tried, interne sú tvorené množinou prístupových metód, ktoré realizujú zápis alebo čítanie z premenných sústredených v tejto triede.
14 Vykresľovanie s pomocou 2 vyrovnávacích pamätí (virtuálna vyrovnávacia pamäť v zobrazovacej jednotke a virtuálna pamäť v pamäti počítača).

RPM. UserControl obsahuje pre prácu s grafikou dve kľúčové metódy.

- `OnPaint (PaintEventArgs e)`
- `OnPaintBackgroung (PaintEventArgs e)`

Zdrojový kód 5: Metódy pre vykresľovanie grafiky v UserControll

Obe metódy slúžia na vykresľovanie grafiky. Sú zavolané pomocou metódy `Refresh ()` , ktorá je volaná pri zmene vlastnosti `Value` . Dá sa povedať, že k prekresľovaniu „budíku“ dochádza pri zmene hodnoty, ktorú sám zobrazuje.

K textovému zobrazeniu meraných hodnôt je použité 7- segmentový font s názvom NI7SEG, ktorý nie je štandardnou súčasťou Windows CE. Pre využívanie fontu je ho nutné nakopírovať do cieľového adresára: `\Windows\Fonts` . Následne je tento font možné v aplikácii použiť:

```
  Font font = new
System.Drawing.Font("NI7SEG",System.Drawing.FontStyle.Regular);
```

Zdrojový kód 6: Vytvorenie nového fontu

Systém ho automaticky vyhľadá v hore uvedenom priečinku. Tento font je možné stiahnuť zo stránky, ktorá je uvedená v literárnom zdroji [15]. Triedny diagram je k dispozícií v časti Prílohy ako Triedny diagram 3: Klientská aplikácia: NmeaParser.

7.1.3.2. Správanie aplikácie pri rôznych rozlíšeniach obrazoviek

Pri mobilných zariadeniach, pre ktoré je klientská aplikácia určená, existujú rôznorodé rozlíšenia obrazovky. GUI je potrebné nastaviť tak, aby nevznikal problém so zobrazovaním. Ovládacie prvky sú volené tak, aby sa dobre zobrazovali aj na nižších rozlíšeniach. Aplikácia sa zobrazuje bez posuvníkov na rozlíšení 260 x 340 a vyššom. Najlepšie pracuje na rozlíšení 480 x 800, pre ktoré bola primárne vyvíjaná. Pri nižšom rozlíšení, ako je najmenšie definované, sa aplikácia zobrazuje s posuvníkmi. Ak by mala byť aplikácia optimalizovaná na nízke rozlíšenie, bolo by nutné zmenšiť obrázky, ktoré sú určené ako pozadie pre „budíky“, veľkosť a rozloženie ostatných komponentov by sa mohlo dynamicky prepočítavať.

7.1.4. Nastavenie aplikácie a ukladanie nastavení

Aplikácia umožňuje užívateľovi nastavovať základné parametre. Zobrazenie a spracúvanie nastavení sprostredkúva formulár `Settings.cs` . Je možné nastaviť údaje pre pripojenie na serverovú časť ako adresa, port, login a heslo. Dáta sú ukladané v registroch[15] systému Windows. Zdrojový kód 7 ukazuje na hlavičky metód, ktoré sa používajú na prácu s registrami. Nachádzajú sa v triede `Tools.` Triedny diagram je k dispozícií v časti Prílohy ako Triedny diagram 4: Klientská aplikácia: Tools

- `public void StoreConnSettings(ConnSettings connSettings)`
- `public ConnSettings LoadConnSettings()`

Zdrojový kód 7: Hlavičky metód pre prácu s registrami

15 Hierarchická databáza v systéme Windows, kde si aplikácie ukladajú nastavenia. Náhrada INI súborov.

Po zavolaní týchto metód je možné uložiť, upraviť alebo načítať aktuálne nastavenia aplikácie. Metódy pracujú s objektom typu `ConnSettings`. Objekt tohto typu obsahuje potrebné Properties pre ukladanie nastavení. Triedny diagram je k dispozícií v časti Prílohy ako Triedny diagram 5: Klientská aplikácia: ConnSettings

Cesta k záznamom aplikácie v registroch je:

```
HKEY_CURRENT_USER\Software\Car client mobile\
```

7.1.5. Komunikácia prostredníctvom sériového portu

Aplikácia s perifernými zariadeniami ako je GPS prijímač a prevodník ELM 327 komunikuje prostredníctvom sériového portu či už priamo alebo prostredníctvom Bluetooth. Pre prístup k sériovým rozhraniam je použitá trieda `SerApi.cs` . Tá využíva systémovú knižnicu `core.dll` aby mohla pracovať so sériovým portom. Táto knižnica je súčasťou Windows CE a väčšina Win32 API[16] je sústredených práve v tejto knižnici. Samotný .NET Compact Framework poskytuje prístup k sériovým portom. Riešenie s knižniciu `core.dll` by malo priniesť väčšiu kompatibilitu s rôznymi zariadeniami.

Táto trieda umožňuje aj základné nastavenia komunikácie sériového portu ako:

- prenosová rýchlosť,
- počet dátových bitov,
- počet stop bitov,
- parita,
- timeout.

Tieto parametre sú predvolene nastavené na štandardné hodnoty. Okrem Win32 API metód a inicializačných parametrov obsahuje trieda `SerApi.cs` aj metódy, ktoré sú použité pre textovú komunikáciu s perifernými zariadeniami. Triedny diagram je k dispozícií v časti Prílohy ako Triedny diagram 1: Klientská aplikácia: SerApi.

- `public bool PortOpen(string port)`
- `public void PortClose()`
- `public byte PortRead()`
- `public void PortWrite(byte[] WriteBytes)`
- `public string PortLineRead()`

Zdrojový kód 8: Metódy pre prácu a nestavania sériového portu

Textový zoznam dostupných sériových portom je možné prostredníctvom:

```
System.IO.Ports.SerialPort.GetPortNames()
```

Zdrojový kód 9: Získanie zoznamu sériových portov

16 Application Programming Interface – zbierka procedúr, funkcií alebo tried a metód určitej knižnice, ktoré môže programátor využívatať.

SerApi teda umožňuje otvoriť sériový port špecifikovaný jeho názvom, port zavrieť, zapísať pole bajtov a čítať po bajtoch alebo po riadkoch. Metóda `PortLineRead()` využíva čítanie po bajtoch a po narazení na špeciálny znak konca riadku `'\r'` alebo `'\n'` , alebo ak dĺžka presiahne 1024 bajtoch je buffer prekonvertovaný na reťazec a použitý ako návratová hodnota. Táto metóda v projekte dobre poslúži napríklad pri čítaní NMEA viet alebo čítaní údajov z prevodníka ELM 327.

7.1.6. Spracovanie dát z GPS prijímača

Aplikácia získava potrebné informácie o polohe a rýchlosti z navigačného systému GPS. Bližšie fungovanie tohoto systému popisuje kapitola 4.Problematika navigačných systémov a možnosti spracovania GPS signálu.

Pripojenie aplikácie na GPS prijímač je realizované prostredníctvom sériového portu, tak ako objasňuje kapitola 7.1.5.Komunikácia prostredníctvom sériového portu.

7.1.6.1. NMEA parser

NMEA parser je spúšťaní z formulára `SelectForm.cs` . Po vytvorení inštancie triedy `NmeaParser` s potrebným parametrom - názov sériového portu GPS prijímača, sa vytvorí nové vlákno, v ktorom bude celý parser pracovať. Vlákno parsera je vytvorené či už len pri prehliadaní údajov ako aj pri kombinovanom prehliadaní a nahrávaní.

V každom načítanom riadku je vypočítané CRC[17]. Ak sa vypočítaná hodnota rovná hodnote, ktorá je zapísaná na danej pozícií v NMEA vete (viz. kapitola 4.3.Komunikačný protokol NMEA 0183), môže sa s analýzou vety pokračovať ďalej. Parser zisťuje či prijatý riadok začína sekvenciou znakov špecifických pre protokol NMEA 0183: `$GPRMC`, `$GPGGA`, `$GPVTG`, `$GPRMA`. Z NMEA viet tento parser získava nasledovné dáta:

- presný čas,
- zemepisná šírka,
- indikátor sever / juh,
- zemepisná dĺžka,
- indikátor východ / západ
- vodorovná rýchlosť.

Všetky dáta, ktoré sú vyextrahované z danej NMEA vety sú vložené do inštancie triedy `Data`. Tá je následne vložená do triedy `AppProperties.cs` . Na zmenu hodnoty premennej `Data` sú prostredníctvom udalostí a delegátov naviazané metódy pre obnovu a načítanie aktuálnych dát v grafickom rozhraní. Dá sa povedať, že ak dôjde k zmene GPS údajov, tak sú údaje zapísané ako premenná typu `Data` a následne vypísané resp. upravené na zobrazovaciu jednotku klientského zariadenia. Môže sa jednať o výpis textovej hodnoty (dĺžka, šírka, výška) alebo rýchlosť (pohyb ručičky na budíku).

17 Cyklická redundantná kontrola – mechanizmus, ktorý sa používa pre zistenie či prijaté dáta obsahujú chybu.

Variantu “prezeranie a nahrávanie dát” parser identifikuje pomocou premenenej typu `bool CaptureMode,` ktorá sa nachádza v triede `AppProperties`. Ak sa jedná o tento mód, tak premennú typu `Data` uloží do fronty, z ktorej bude neskôr odobraná a odoslaná prostredníctvom siete na server. Fronta je typu `Queue`, má názov `DataList` a je uložená v triede `AppProperties.` Pred odosielaním na frontu sú k údajom z GPS pripojené ešte aj údaje z OBD. O odosielaní údajov na server informuje kapitola 7.1.8.Sieťová komunikácia a spracovanie dát. Ukončenie parsera je realizované prostredníctvom metódy `Close ()`, ktorá následné zruší vlákno, v ktorom je parser spustený. Triedny diagram parsera je k dispozícií v časti Prílohy ako Triedny diagram 3: Klientská aplikácia: NmeaParser.

7.1.6.2. Autodetekcia sériového portu

Vo formulári `Settings` je možné nastaviť aj názov sériového portu pre komunikáciu z GPS prijímačom. S využitím toho, že na začiatku NMEA vety sa musí nachádzať príznačná sekvencia znakov, bolo možné automaticky detekovať, na ktorom sériovom porte sa GPS prijímač nachádza.

Prehľadávanie sériových portov má na starosti metóda `ScanGpsPort ()` , ktorá sa nachádza v triede `Tools`. Táto získa zoznam sériových portov a postupne každý zo zoznamu otvorí. Ďalej sa snaží prečítať jeden riadok a zisťuje či sa v ňom na začiatočnej pozícií nachádzajú sekvencie typické pre protokol NMEA 0183. Po zistení správneho sériového portu pre komunikáciu s GPS sa jeho názov zapíše do premennej `GPS` , ktorá sa nachádza v triede `AppProperties`.

7.1.7. Spracovanie dát z OBD

Dáta z riadiacej jednotky sa do aplikácie dostávajú prostredníctvom prevodníku ELM 237 (viz. kapitola 5.3.Prevodník ELM 327). V tomto riešení prevodník komunikuje s mobilným zariadením cez Bluetooth. V operačnom systéme je pre tento prevodník vyhradený sériový port. Prostredníctvom tohto portu prebieha celá komunikácia aplikácie s motorovým vozidlom, respektíve jeho riadiacou jednotkou. Komunikáciu aplikácie so sériovým portom objasňuje kapitola 7.1.5.Komunikácia prostredníctvom sériového portu. O možnostiach a rozmanitosti protokolu OBD II hovorí kapitola 5.2.Protokol SAE8 J1979.

7.1.7.1. Inicializácia

Pre spracovanie OBD dát musel byť opäť navrhnutý parser. Podobne ako GPS parser je OBD parser spúštaní z formulára `SelectForm.cs`. V tomto prípade je parser prvý krát použitý pri zisťovaní podporovaných PIDs. Táto operácia netrvá veľmi dlho a preto ju nie je nutné spúštať v samostatnom vlákne. Všeobecné informácie o princípe zisťovania podporovaných PIDs je možné nájsť v kapitole 5.2.1.3.Počiatočné preverovanie podporovaných PIDs . Metóda, ktorá zabezpečuje zisťovanie podporovaných PIDs má pomenovanie `GetSupportedPids(string mode, string group).` Táto metóda vracia binárny reťazec, podľa ktorého je možné v ďalších krokoch určiť podporované PIDs.

Každý PID je opísaný objektom typu `Senzor`, ktorý má tieto premenné:

- `string pidID` (identifikácia podľa tabuľky dostupných PIDs),
- `int ticket` (počet tiketov – určuje prioritu pri odosielaní daného požiadavku),

- `float value` (hodnota daného PID),
- `bool supported` (určuje či je daný PID v tejto konfigurácií podporovaný),
- `bool view` (určuje, či sa tento PID použije pri zobrazovaní a teda aj meraní),
- `bool capture` (určuje, či sa tento PID použije pri nahrávaní a teda aj meraní a zobrazovaní).

Celý zoznam PIDs, s ktorými aplikácia vie pracovať je umiestnený opäť v triede `AppProperies`, jedná sa v podstate o `List`, ktorý obsahuje položky typu `Senzor`. Pri zisťovaní podporovaných PIDs sa tento list prechádza a v jednotlivých objektoch sa menia vlastnosti `supported`.

Vo formulári `SelectForm` sú zobrazené `CheckBoxy` a podľa dostupnosti PID je im nastavená `property Enabled` na hodnotu `true` alebo `false`.

7.1.7.2. OBD Parser

OBD parser je spúšťaný z formulára `SelectForm.cs` . Po vytvorení inštancie triedy `OBDParser` s potrebným parametrom - názov sériového portu prevodníka ELM a referencie na triedu `SelectForm`, sa vytvorí nové vlákno, v ktorom bude celý parser pracovať. Vlákno parsera je vytvorené či už len pri prehliadaní údajov ako aj pri kombinovanom prehliadaní a nahrávaní.

Na začiatku sa vyberú PIDs, ktoré sú podporované v aktuálnej konfigurácií a naviac boli označené na prehliadanie alebo na prehliadanie + nahrávanie.

V ďalšom kroku je na tento zoznam zvolaná metóda `int GetTotalTickets ()`. Keďže vybrané objekty v zozname sú typu Senzor, je im možné priradiť určitý počet Tiketov. To bude neskôr určovať prioritu pri zasielaní požiadaviek na sériový port prevodníka. Metóda `GetTotalTickets ()` vypočíta súčet všetkých tiketov.

V ďalšom kroku metóda `int RandomNumber(int max)` vráti náhodné číslo od 0 po súčet všetkých tiketov, ktorý dostane ako parameter.

Metóda `string SenzorFromRandomNumber(int randomNumber, List<Senzor> parseSenzor)` na základe náhodne vygenerovaného čísla a zoznamu všetkých aktuálne používaných PIDs, určí ktorému PID patrí číslo a to nasledovaným spôsobom:

- Každý PID má definované hranice na základe počtu tiketov. Napr. prvý PID má rozsah hodnôt 0 až jeho počet tiketov. Druhý PID začína na počte tiketov prvého +1 a končí na hodnote prvého +1 + svoj počet tiketov.
- Náhodne vygenerované číslo teda vždy spadne do oblasti rozsahu hodnôt tiketov nejakého PID.
- Nakoniec názov zodpovedajúceho PID vráti ako reťazec.

Je zrejmé, že PID s najväčším počtom tiketov bude vyberané najčastejšie. Názov PID, ktorý je výsledkom predchádzajúcich výpočtov, je odoslaný na sériový port, na ktorom je pripojený prevodník ELM 327. Komunikačný protokol OBD II, prostredníctvom ktorého prebieha komunikácia medzi prevodníkom a aplikáciou je popísaný v kapitole 5.2.1.4.Proces získavania dát.

Po odoslaní dotazu parser prechádza na časť čítania dát. Podľa toho ako je uvedené v kapitole zaoberajúcej sa OBD II protokolom, prijaté dáta budú obsahovať identifikáciu, že sa jedná o odpoveď, dotaz, na ktorý je odpoveď naviazaná a samotné dáta potrebné k výpočtu reálnej hodnoty daného PID. Na základe týchto skutočností je zostavená prijímacia časť parsera. Pracuje nasledovaným spôsobom:

- v prvom rade určí či sa jedná naozaj o korektnú odpoveď,
- vyextrahuje údaje o dotaze, na ktorý je generovaná samotná odpoveď,
- na základe vyextrahovaného počiatočného dotazu vypočíta hodnotu podľa vzorca, ktorý prislúcha práve tomuto dotazu,
- aktualizuje hodnotu daného PID v zozname,
- do vlastnosti `ObdValueChangedProp` uloží označenie dotazu, aby bolo možné identifikovať PID, ktorého hodnota sa zmenila. Na zmenu tejto vlastnosti sú naviazané ďalšie metódy v `SelectForm`, ktoré zabezpečia aktualizáciu grafických zobrazovacích prvkov.

Po vykonaní týchto operácií sa parser opäť vráti na začiatok a zasa náhodne vyberie PID, ktoré odošle a následne spracuje podľa uvedených bodov. Celý cyklus udalostí parsera sa v rýchlom slede stále opakuje dookola.

Ukončenie parsera je realizované prostredníctvom metódy `Close ()`, ktorá následné zruší vlákno, v ktorom je parser spustený. Triedny diagram parsera je k dispozícií v časti Prílohy ako Triedny diagram 6: Klientská aplikácia: ObdParser.

7.1.8. Sieťová komunikácia a spracovanie dát

7.1.8.1. Komunikačný protokol

Aplikácia na komunikáciu so serverovou časťou využíva socketové[18] pripojenie. Konkrétne je využitý TCP socket. Jedná sa teda o spojovo orientované socketové spojenie.

Trieda, ktorá celé spojenie riadi má názov `Client`. Po zavolaní jej konštruktora `public Client(string host, int port)` sa socket vytvorí, nastaví a pripojí.

Nakoniec je socket predaný triede `DataRW`. Tá implementuje metódy:

- `string Load ()` - načítanie jedného riadka zo streamu,
- `void Strore (string line)` - uloženie jedného riadka do streamu,
- `void Close ()` - uzatvorenie sieťových streamov.

Na prenos dát medzi serverom a klientom sa používa jednoduchý textový protokol. Údaje sa posielajú po riadkoch a jednotlivé dáta sú od seba oddelené znakom = . Za prvým „=“ je spravidla uvedený identifikátor dát, o ktoré sa bude jednať. Za ďalšími znakmi „=“ sa už nachádzajú samotné dáta v poradí a formáte v akom je aj časť servera a klienta.

18 Jedná sa o abstraktné pojmy, ktoré umožňujú programátorom využívať sieťové spojenie pri tvorbe aplikácii. Porty socketu umožňujú prepojiť počítače alebo zariadenia na logickej úrovni. Je identifikovaný číslom z rozsahu od 1 po 65535.

Klient po úspešnom pripojení na server odosiela prihlasovacie údaje aby mohol prejsť do stavu „logged“ a vykonávať tak ďalšie operácie. Formát odosielania prihlasovacích informácií popisuje Zdrojový kód 10.

```
StringBuilder line = new StringBuilder("=");

line.Append ("DriverAccount" + "=");
line.Append ( Login + "=" );
line.Append ( Password );
```

Zdrojový kód 10: Odosielanie prihlasovacích informácií

Server podľa prvého parametra rozozná, že sa jedná o prihlasovaciu správu, ktorú ďalej spracuje a odošle klientovi odpoveď :

- `LoginOK`,
- `NotAutorized`,
- `LoginFailed`.

Ďalším príkladom môže byť načítanie všetkých ŠPZ, ktoré sa nachádzajú v systéme za účelom zobrazenia v comboboxe a následného vybratia a pridruženia k danej jazde.

`"=GetCars"`

Odpoveďou na tento dotaz budú ŠPZ všetkých vozidiel registrovaných v systéme oddelených znakom „=“ . Diagram triedy Client je dostupný v časti prílohy ako Triedny diagram 7: Klientská aplikácia: Client.

7.1.8.2. Odosielanie nameraných dát na server

Ako už bolo uvedené v kapitole 7.1.6.1.NMEA parser, ak sa v danej konfigurácií jedná aj o nahrávanie dát, sú tieto objekty ukladané na frontu. Objekt typu `Data` obsahuje aj metódu `void AddObdData(ObdData obdData)`, ktorá slúži na pridanie dát z OBD do tohto objektu. Táto metóda je spustená vždy pri pridávaní objektu na frontu. Ukladajú sa teda kompletné údaje o polohe, rýchlosti z GPS prijímača ako aj údaje z motorového vozidla, získané prostredníctvom protokolu OBD II.

Samotné odosielanie dát má na starosti trieda `DataSender`. Jej inštancia je vytvorená pri spustení nahrávania dát. Ako parameter je jej predaná inštancia triedy `DataRW`, ktorá obsahuje metódy potrebné pre zapisovanie a čítanie textových dát v sieti (`Load ()`, `Store (string line)`). Konštruktor triedy `DataSender` spustí metódu `void Send ()` v novom, samostatnom vlákne. Táto metóda postupne vyberá objekty z fronty a odosiela do sieťového streamu. Ak je frotna úplne prázdna, vlákno sa pozdrží na 2000 ms a potom sa znova spustí.

Objekty typu `Data`, ktoré sú určené na odosielanie do sieťového streamu obsahujú preťaženú metódu `string ToString ()`. Táto metóda pripraví všetky dáta, na formu vhodnú pre odoslanie na server. Forma má rovnaké princípy, ako napr. pri odosielaní prihlasovacích údajov.

Kvôli potrebám servera je však nutné ešte pred začatím odosielania samotných dát celú reláciu inicializovať. Inicializácia prebieha v odoslaní ŠPZ a času kedy jazda začína. Táto správa je označená kľúčovým slovom `StartCapture`. Ukážku implementácia popisuje Zdrojový kód 11.

```
StringBuilder line = new StringBuilder("=");

line.Append ("StartCapture" + "=");
line.Append ( numberPlate + "=" );
line.Append ( startTime.ToString() );
```

Zdrojový kód 11: Inicializácia jazdy

Po tejto správe nasledujú správy s dátami. Správa je označená kľúčovým slovom `Data` a má nasledovný tvar:

```
StringBuilder line = new StringBuilder("=");

line.Append ("Data" + "=");
line.Append(Lat.ToString() + "=");
line.Append(Lng.ToString() + "=");
line.Append(Alt.ToString() + "=");
line.Append(Speed.ToString() + "=");
line.Append(TimeOfFix.ToString() + "=");
line.Append(Obd010D.ToString() + "=");
…
…
line.Append(Obd010C.ToString());
```

Zdrojový kód 12: Odosielanie nameraných dát

Ak je už jazda ukončená a sú odoslané všetky dáta, pristupuje sa k vloženiu správy, ktorá serveru identifikuje ukončenie odosielania a zároveň aj ukončenie celej jazdy. Je označená kľúčovým slovom `EndCapture` a okrem toho odosiela aktuálny čas ukončenia jazdy. Opäť pre potreby servera. Má nasledovný tvar:

```
StringBuilder line = new StringBuilder("=");

line.Append ("EndCapture" + "=");
line.Append (endTime.ToString());
```

Zdrojový kód 13: Ukončenie jazdy

7.1.8.3. Práca pri slabej konektivite

Slabou konektivitou sa myslí stav sieťového pripojenia klientského zariadenia, kedy je veľmi obmedzená šírka pásma, dochádza k veľkej chybovosti prenášaných paketov alebo k občasnému odpájaniu a pripájaniu. K takýmto stavom môže pri charaktere používania tejto aplikácie dochádzať pomerne často. Ak zoberieme do úvahy, že zariadenie sa bude nachádzať väčšinou v pohybujúcom sa automobile, výpadky či vysoké časové odozvy napr. pri GPRS alebo WiFi môžu byť relatívne častým javom.

Aplikácia je na javy podobného typu pripravená a vie s nimi pracovať. Najväčšiu šírku pásma spotrebuje prenos nameraných dát na server. Nasledujúce príklady zotavenia sa budú zameriavať práve na tento prenos.

Pri veľmi malej šírke pásma začne dochádzať k plneniu fronty, ktorá slúži na dočasné ukladanie dát. Aj keď sa budú z fronty odoberať objekty len veľmi pomaly, nakoniec sa odošlú všetky. Pri zmene konektivity na štandardnú sa môžu objekty z fronty veľmi rýchlo odosielať a vykryť tak predchádzajúce spomalenie. Ak je konektivita permanentne slabá, k vyprázdňovaniu fronty bude dochádzať aj po ukončení jazdy, až kým nebude fronta úplne prázdna.

Ak počas prenosu dôjde k úplnému odpojeniu od siete, objekty sa neprestanú na frontu ukladať. Metóda `void CheckServerReachable()` spustená v samostatnom vlákne v triede `Tools` bude periodicky každých 5 sekúnd zisťovať dostupnosť servera. Po kladnom zistení sa spustí metóda `Client CreateConnection(bool reconnect)` s parametrom `true`. Opäť prebehnú prihlasovacie operácie a proces odosielania dát z fronty bude obnovený.

7.1.8.4. Práca v úplne odpojenom režime

Vo formulári `Settings` je možné nastaviť, že aplikácia bude pracovať v úplne odpojenom režime. Rozdiel oproti režimu slabej konektivity je v tom, že sa dáta neodosielajú na frontu a potom do sieťového streamu ale rovno sa zachytávajú do textového súbora.

Samotné odosielanie dát má na starosti trieda `DataSender`. Jej inštancia je vytvorená pri spustení nahrávania dát. V tomto prípade sa volá konštruktor bez parametra. Tento konšruktor nastaví parametre výstupného súbora, do ktorého sa budú ukladať jednotlivé dáta. Konštruktoru triedy `DataSender` spustí metódu `void Send ()` v novom, samostatnom vlákne.

7.2. Serverová časť

7.2.1. Stručný náčrt funkcionality

Aplikáciu na strane servera tvorí webový informačný systém implementovaný v prostredí ASP. NET, MS SQL databáza a TCP server určený pre komunikáciu s klientskými aplikáciami.

Informačný systém má databázu užívateľov. Overovanie prihlasovaných údajov a oprávnení prebieha pri práci so systémom prostredníctvom webového rozhrania ako aj pri pripájaní klientských aplikácií.

Okrem databázy užívateľských účtov aplikácia obsahuje aj evidenciu motorových vozidiel a samotných jázd vytvorených klientskými aplikáciami. Evidencie je možné editovať, vkladať nové dáta prípadne nepotrebné mazať.

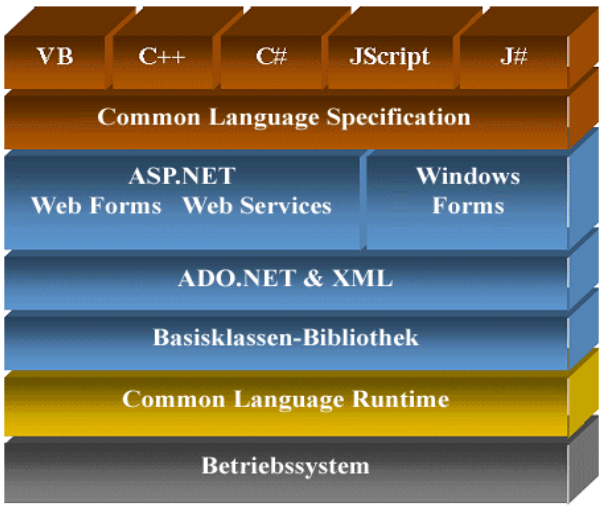
TCP server očakáva spojenia od klientov na definovanom porte. Server umožňuje obsluhovať viac klientov paralelne.

Klient si po úspešnom pripojení na server môže vyžiadať zoznam automobilov. Primárna funkcionalita komunikácie medzi serverom a klientom je v odosielaní dát získaných z jazdy. Server po začatí jazdy vytvorí databázový údaj, ktorému sa pridružia prijaté dáta z klientskej aplikácie. Vo webovom rozhraní aplikácie je po dokončení jazdy možné vizualizovať trasu v prostredí Google Earth spolu s údajmi s riadiacej jednotky automobilu. Trasu je možné zobraziť aj pomocou Driving simulátora.

7.2.2. Výber platformy

Pre tento typ aplikácie sa najviac hodí webový informačný systém, ktorý bude pracovať s databázou. Rozhodol som sa vybrať ako platformu pre serverovú časť ASP. NET, konkrétne framework MVC 2.0. ASP. NET je súčasť .NET frameworku.

Slúži na tvorbu webových aplikácií a webových služieb. Je nástupcom staršej technológie ASP. Keďže je ASP .NET súčasťou .NET Frameworku, je možné pre túto platformu vyvíjať v jazykoch zdieľaných .NET Frameworkom ako napr. Visual Basic alebo C#.



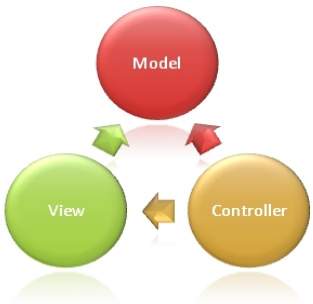
Obrázok 10: Architektúra .NET Framework

Implementácia v ASP. NET je vo veľa prípadoch podobná vývoju desktopových aplikácií pre Windows. Obsiahnuté sú podobné ovládacie prvky ako Buttons, Label a pod. Umožňuje tak programátorom plynule prechádzať z vývoja desktopových aplikácií k webovým. ASP. NET. Aplikácie sú primárne určené pre webový server IIS[19], bežiaci pod systémami Windows. IIS je aktuálne vo verzií 7.0 a radí sa medzi najpoužívanejšie webové servery. Dôvodom je práve podpora ASP. NET. Je možné použiť aj webový server Apache v kombinácií s implementáciu .NETu Monem.

7.2.2.1. Vzor MVC[20]

MVC je všeobecný architektonický vzor, ktorý môže byť použitý pri budovaní software. Oddeľuje vývoj softwaru do vrstiev resp. nezávislých komponent užívateľského rozhrania, aplikačnej logiky a dátového modelu. Všeobecne sa komponenty označujú takto:

- model (reprezentácia dát, s ktorými aplikácia pracuje),
- view (prevádzanie dát na podobu vhodnú pre prácu užívateľa – typicky HTML stránka),
- controller (reakcia na udalosti).



Obrázok 11: Architektonický vzor MVC

19 Internet Information Services – webový server od spoločnosti Microsoft.
20 Model–view–controller

Priebeh akcie prostredníctvom MVC môže všeobecne prebiehať takto:

- vyvolanie akcie v užívateľskom rozhraní (stisk tlačítka),
- riadenie prevezme controller, ktorý presne identifikuje akciu z užívateľského rozhrania,
- controller pristúpi k modelu a ten môže dáta aktualizovať,
- view použije zaktualizovaný model k zobrazeniu dát [16].

MVC vzory sú dostupné pre rôzne programovacie jazyky a prostredia. Medzi významné a najviac používané MVC frameworky patrí JavaServer Faces a Jakarta Struc 2 pre jazyk Java, Zend Framework a Cake PHP pre jazyk PHP a ASP MVC pre platformu .NET [16].

V súčastnosti sa MVC používa ako architektúra pre zložitejšie webové aplikácie, kde táto architektúra zabezpečuje flexibilitu a spoľahlivosť.

ASP MVC

ASP MVC je teda implementácia MVC pre ASP. NET. Ide o alternatívu pre ASP .NET WebForms[21], nie je však koncipovaný ako jeho náhrada. Aktuálne je vo verzií 3.0.

Výhody ASP MVC

- oddelenie logiky od prezentačnej časti (umožňuje lepšie rozdelenie práce medzi tím vývojárov a dizajnérov),
- zmena niektorej komponenty priamo neovplyvňuje fungovanie ostatných komponent,
- kontrola nad generovaným kódom v prezentačnej časti,
- dobrá integrácia s AJAX[22] a jQuery[23],
- jednoduchšie testovanie,
- URL adresy vhodné pre SEO[24].

Nevýhody ASP MVC

- zložitejšia implementácia,
- previazanosť vrstiev view a controller na modeli (zmena rozhrania modelu indikuje zásah do controllera a v niektorých prípadoch aj do view, zložitosť zásahov do kódu sa zvýši)[16].

MVC projekt

Aktuálnu verziu MVC 3.0 je možné stiahnuť z webových stránok projektu. MS Visual Studio 2010 MVC projekt obsahuje verziu 2.0. Staršie verzie Visual studia tento projekt neobsahujú, je teda nutné ho vždy dodatočne doinštalovať.

---

21 Technológia pre tvorbu dynamických webových stránok od spoločnosti Microsoft.

22 Asynchronous JavaScript and XML – technológia interaktívneho vývoja webových aplikácií, ktorá umožňuje meniť obsah webovej stránky bez nutnosti znovu načítania.

23 Javascriptová knižnica kladie dôraz na interakciu medzi Javascriptom a HTML, slúži hlavne na navigáciu, vytváranie animácií, spracovanie udalostí atď.

24 Search engine optimization – súbor techník na optimalizáciu webovej stránky za účelom zlepšenia pozície pri vyhľadávaní stránky vo vyhľadávačoch

Základné adresáre a súbory automaticky generované v MVC projekte sú:

- App_Data – v tomto adresári sú uložené databázové súbory (súbory .mdf alebo .xml),
- Content – obsahuje css súbory s kaskádovými štýlmi a ďalšie súbory potrebné pre grafickú časť aplikácie ako napr. obrázky,
- Controllers – adresár pre ukladanie controllerov,
- Models – adresár pre ukladanie modelov,
- View – adresár pre ukladanie view v podobe apsx súborov
- Scripts – v tomto adresári sú umiestnené scripty spúšťané na strane klienta.

Ďalšie adresáre vytvorené špeciálne pre tento projekt:

- Parsers – obsahuje parsery,
- TCPserver – obsahuje triedy, ktoré sú zodpovedné za fungovanie TCP servera pre komunikáciu s klientskými aplikáciami.

7.2.3. Komunikácia s klientskými aplikáciami

Komunikácia je zabezpečená prostredníctvom TCP servera, založená na socketoch. Zdrojové kódy k serveru sú umiestené v adresári TCPserver.

Zachytávanie nových spojení je riešené v triede `Server`, kontorétne metóda `public void AccConnection()`. Dá sa povedať, že spustením tejto metódy začne server odpovedať na žiadosti o spojenia. Metóda a teda aj celý TCP server je spúšťaní z triedy `Tools` pomocou statickej metódy `public static void StartTcpServer(int port, int numConnection)`. Metóda ešte pred spustením samotného servera prevedie inicializáciu prostredníctvom metódy `public void Init(int port, int numConnection)`. Inicializáciou sa v tomto prípade myslí definovanie TCP portu, na ktorom bude server očakávať spojenia a definovanie maximálneho počtu pripojených klientských aplikácií. Nastavenie týchto parametrov je možné meniť v užívateľskom rozhraní aplikácie. Na tom isto mieste je možné server zapnúť alebo vypnúť. Nastavenie servera sa ukladá do tabuľky v MS SQL databáze. Server môže byť automaticky spustený pri štarte systému. Záleží na užívateľskom nastavení. Pri spustení systému je zavolaná metóda `protected void Application_Start()` v triede `Global.asax`. V tejto metóde sa vykoná spúšťanie servera, ak sú kritéria na autoštart splnené. TCP sa spúšťa v novom samostatnom vlákne.

Po akceptovaní socketu pri pripojení od klientskej aplikácie metóda `AccConnection()` vytvorí inštanciu triedy `ConnectionHandler`. Táto inštancia v podstate reprezentuje jedno spojenie od klientskej aplikácie. Následne sa v novom vlákne spustí metóda `public void InitTransfer(Object clientSocket)`. Ďalšie akcie, spojené s obsluhou klienta prebiehajú už len prostredníctvom danej inštancie a daného vlákna triedy `ConnectionHandler`.

Metóda `public void InitTransfer(Object clientSocket)` ďalej vytvorí inštanciu na triedu `DataRW` a predá jej socket, s ktorým pracuje. Táto trieda obsahuje metódy `public void Store(string data)` a `public string Load()`, ktoré slúžia na

príjem a odosielanie textových reťazcov medzi serverom a klientom.

V ďalšom kroku sa volá metóda `public void ReceiverHandling()`, ktorá v nekonečnom cykle spracováva požiadavky od klientskej aplikácie. Komunikačný protokol medzi serverom a klientom popisuje kapitola 7.1.8.1.Komunikačný protokol. Metóda teda sleduje o akú požiadavku klientská aplikácia žiada, respektíve zasiela. Na základe toho spustí odpovedajúce metódy a spracuje, a odošle dáta klientskej aplikácií. Jedná sa o akcie, ktoré popisujú nasledujúce podkapitoly. Po odpojení klienta server reaguje zrušením vlákna, v ktorom spojenie prebiehalo.

Diagram triedy `Server` je dostupný v časti Prílohy ako Triedny diagram 8: Serverová aplikácia: Server, diagram triedy `ConnectionHandler` ako Triedny diagram 9: Serverová aplikácia: ConnectionHandler a diagram triedy `Tools` ako Triedny diagram 11: Serverová aplikácia: Tools.

7.2.3.1. Spracovanie prihlasovacích údajov

Na spracovanie prihlasovacích údajov slúži metóda `private void HandleAccount(DriverAccount driverAccount).` Táto metóda je volaná, pokiaľ bude prijatá správa identifikovaná ako žiadosť o prihlásenie. Identifikácia tejto správy je na základe označenia: `DriverAccount`. Tak ako popisuje 7.1.8.1.Komunikačný protokol, ďalšie časti správy obsahujú prihlasovacie informácie. Tie sú vyextrahované, následne vložené do objektu typu `DriverAccount`.

Overovanie užívateľských účtov voči databáze je riešené prostredníctvom triedy `AccountMembershipService` a jej metódy `ValidateUser(username, password)`. Inštancia tejto triedy je umiestená v triede `Tools` a sprístupnená pomocou Properties.

Preveruje sa aj to či užívateľ patrí do skupiny užívateľov Driver, teda či sa jedná o vodiča. Preverovanie je riešené pomocou metódy `public static bool IsInRole(string userName, string roleName)` v triede `Tools.`

Po preverení prihlasovacích údajov môže metóda `private void HandleAccount(DriverAccount driverAccount)` zaslať klientskej aplikácií nasledovné odpovede:

- `LoginOK,`
- `NotAutorized,`
- `LoginFailed.`

V systéme existuje zoznam, ktorý obsahuje informácie o jednotlivých spojeniach klientských aplikácií. Jedná sa o `ArrayList`, ktorý je umiestený v triede `ClientConnection`. Obsahuje aj všetky potrebné metódy pre vkladanie nových spojení či úpravu už existujúcich spojení. Jedná sa hlavne o úpravu Loginu pri prihlasovaní alebo úpravu stavu spojenia. Celý zoznam spojení je možné v užívateľskom rozhraní aplikácie zobraziť.

7.2.3.2. Prístup k evidencií vozidiel

Klientská aplikácia môže generovať požiadavku na zoznam vozidiel, za účelom priradenia vozidla k danej jazde. Server odosiela zoznam ŠPZ, podľa ktorých sa dá identifikovať jednoznačne každé vozidlo v evidencií.

Identifikácia tejto správy od klienta je `GetCars`. Po narazení na túto správu je automaticky zavolaná metóda `private void HandleCarList()`. Tu je pomocou `CarController` a jeho metódy `public List<Car> GetCarList()` vytvorený zoznam ŠPZ a následne odoslaný klientskej aplikácií.

7.2.3.3. Príjem dát

Klient je navrhnutý tak, aby pred začatím odosielania dát celú akciu inicializoval. Server tak môže previesť úvodnú inicializáciu dát v databáze a pripraviť sa na samotný príjem dát. Pre tieto potreby je v databáze definovaná tabuľka `Driving`.

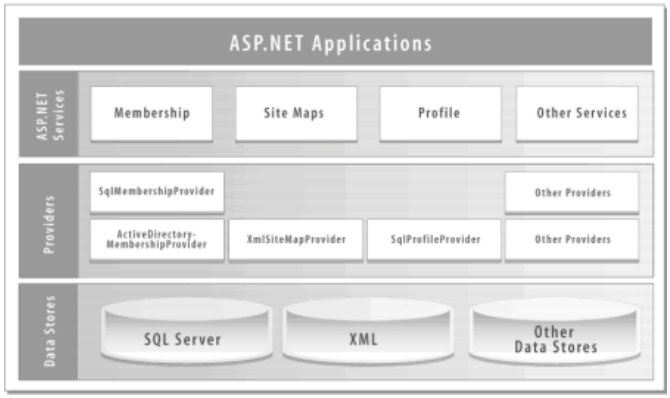
Inicializačná správa má označenie `StartCapture`. Táto správa nesie informáciu o ŠPZ vozidla priradenú k tejto jazde a čas začatia jazdy. Spracúva ju metóda `private void HandleStartCapture(string numberPlate, string startTime)`. Táto metóda vytvorí nový záznam v tabuľke `Driving` a pridá potrebné informácie. Ďalej sa vytvorí nový textový súbor, do ktorého budú dáta ukladané. Cestu k súboru je možné nastaviť v užívateľskom rozhraní aplikácie. Názov súbora je odvodený od primárneho kľúča záznamu v tabuľke `Driving`, teda atribút DrivingID. Na základe tohto identifikátora je súbor a záznam v tabuľke jázd previazaný.

Správa, ktorá obsahuje samotné dáta je označená identifikátorom `Data`. Obsluhu tejto správy má na starosti metóda `private void HandleData(string[] buffer)`. Funkcionalita spočíva v ukladaní správ v tom istom tvare ako prichádzajú z klientskej aplikácie do textového súbora. Vynecháva sa akurát identifikátor správy. Jednotlivé riadky v textovom súbore nájdu využitie v ďalších častiach projektu.

Ak už klient nehodlá k danej jazde posielať dáta, musí zaslať serveru správu, ktorá identifikuje ukončenie jazdy. Je označená identifikátorom `EndCapture` a obsluhuje ju metóda `private void HandleEndCapture(string endTime)`. Táto správa obsahuje čas ukončenia jazdy, tak ako to bolo inicializované na strane klienta. Tento čas sa teda neurčuje na strane servera. Správa totiž môže byť doručená v značnom rozdielom čase, ako bola jazda reálne ukončená. Funkcionalita metódy spočíva v úprave, respektíve doplnení údaju o ukončení jazdy do záznamu v databázovej tabuľke `Driving` a ukončení súborových streamov, ktoré zabezpečovali zapisovanie dát do pridruženého súbora.

7.2.4. Správa a riadenie užívateľských účtov

APS. NET na prácu s užívateľskými účtami obsahuje tzv. ASP .NET Provider model. K dispozícií je od verzie ASP .NET 2.0. Vývojári tak nemusia vyvíjať autorizačné mechanizmy pre webové aplikácie. Hlavnou úlohou je poskytovanie autorizačných a autentifikačných služieb pre ASP .NET aplikácie. V mnoho prípadoch je potrebné, aby mala aplikácia schopnosť rozlíšiť jednotlivých užívateľov a riadiť prístup k zdrojom.



Obrázok 12: Provider model

Provider model obsahuje 3 všeobecné základné moduly:

- Membership,
- Role,
- Profile.

Membership modul sprostredkúva riešenie pre tvorbu a správu užívateľov. Dá sa povedať, že sa jedná o model členstva. Obsahuje metódy na pridanie užívateľa, editáciu, reset hesla, zmazanie užívateľa, hlavne však overovanie pravosti prihlasovacích údajov a pod.

Modul role poskytuje mechanizmus pre tvorbu a správu jednotlivých rolí v systéme. V rámci aplikácie je možné nadefinovať viac rolí a týmto roliam je potom možné priradiť užívateľov. Jedna rola môže obsahovať viac užívateľov a jeden užívateľ môže byť členom viacerých rolí. Definovanie užívateľských rolí a priradenie užívateľov umožňuje používať autorizáciu, resp. overovať či má na danú akciu užívateľ oprávnenie.

Modul profil sa používa pre spracovanie dodatočných informácií o užívateľovi. Aplikácia ho môže využívať na rôznorodé dodatočné vlastnosti ako meno, priezvisko, bydlisko atď.

Tieto moduly sú všeobecné, jednotlivé implementácie sa môžu líšiť. Metódy definované v moduloch sú len určitou formou abstrakcie. Rozdiely v implementácií providra sú hlavne v tom, kam jednotlivé dáta ukladá. Môže sa jednať o SQL databázu, XML súbory, overovanie voči Active Directory a pod. Z tohto dôvodu je možné si provider napísať podľa svojich potrieb. Je len nutné dodržať štruktúru ako je definovaná v jednotlivých modeloch.

V tomto riešení je použitý AspNetSqlProvider. Teda provider, ktorý je priamo definovaný v ASP .NET a využíva ukladanie dát v MS SQL databázi. To, ktorý provider sa bude v aplikácií používať je zapísané v konfiguračnom súbore aplikácie `Web.config`.

Databázový súbor je umiestený v priečinku `App_Data` a má názov `ASPNETDB.MDF`. Databáza obsahuje väčší počet tabuliek, ktoré sú potrebné na spracovanie autorizácie a autentifikácie. Prístup k dátam je sprostredkovaný prostredníctvom triedy `AccountModels.cs`. Pre ilustráciu sa hlavne jedná o metódy ako:

- `public bool ValidateUser(string userName, string password)`

- `public MembershipCreateStatus CreateUser(string userName, string password, string email)`
- `public bool ChangePassword(string userName, string oldPassword, string newPassword)`

Zdrojový kód 14: Základné metódy pre spracovanie užívateľských účtov

Text z tejto kapitoly čerpá z literárneho zdroja [17].

7.2.4.1. Autentifikácia a autorizácia

Aplikácia tohto typu potrebuje, aby s ňou mohli pracovať užívatelia, ktorí sa preukážu platnými prihlasovacími údajmi (autentifikácia). Naviac je nutné rozdeliť užívateľov do rolí a podľa toho im povoľovať prevádzať v systéme dané operácie a akcie. V tomto projekte sú definované nasledovné užívateľské role: admin a driver.

Na overovanie užívateľských rolí je implementovaná statická metóda `public static bool IsInRole(string userName, string roleName)` v triede `Tools`. Pracuje na základe vstupných parametrov užívateľského mena a role. Vracia kladný výsledok ak je užívateľ členom danej role. Táto metóda je hlavne využívaná pri autorizácií jednotlivých operácií systému.

Logika systému, ktorá zaobstaráva prihlasovanie do aplikácie bola vytvorená spolu z novým projektom. Vytvorený bol `AccountController` a šablóny v adresári `Views\Account`. Je teda možné prevádzať štandardné operácie ako prihlásenie či registrácia. Do tejto aplikácie boli postupne pridané ďalšie operácie. Najskôr bolo však nutné správne implementovať správne autentifikačné a autorizačné mechanizmy do jednotlivých sekcií alebo na určité akcie.

Implementácia autentifikácie a autorizácie

Po spustený užívateľského rozhrania systému resp. webovej stránky je užívateľ definovaný ako neprihlásený. Musí mať teda obmedzené zdroje na minimum a jediné, čo mu bude z dostupné funkcionality systému je prihlásenie alebo registrácia. Zdrojový kód 15 ukazuje možnosť riadenia zobrazenia na strane na základe užívateľských rolí.

```
<%
if (ISCarData.Tools.IsInRole((Page.User.Identity.Name), "driver"))
{ %>
<!-- kód, ktorý je určený pre užívateľské role driver -->
<%} %>
```

Zdrojový kód 15: Oddelenie časti View pre vybraných užívateľov

Používa sa vyššie popísaná statická metóda `public static bool IsInRole(string userName, string roleName)`. Podobným spôsobom je definované zobrazovanie pre užívateľské role admin. Je zrejmé, že časť, ktorá je umiestená v tomto bloku bude prístupná len užívateľom, ktorý sú členmi definovanej role. Kód, ktorý nebude takto definovaný sa zobrazí aj neprihláseným užívateľom, poprípade užívateľom, ktorý nie sú členmi žiadnej role.

Ďalším zabezpečením aplikácie je na strane akčných metód v controlleroch pomocou atribútov metódy:

- `[Authorize]` – zabezpečenie pred neprihláseným užívateľom,
- `[Authorize(Roles = "admin")]` – zabezpečenie metódy proti volaniu iných

užívateľov ako tých, ktorý sú členmi role admin,

- `[Authorize(Roles = "driver")]` – zabezpečenie metódy proti volaniu iných užívateľov ako tých, ktorý sú členmi role driver.

Takto zabezpečená akčná metóda môže byť implementovaná nasledovne:

```
[Authorize(Roles = "admin")]
  public ActionResult Create()
   {
      Car car = new Car();
      return View(car);
   }
```

Zdrojový kód 16: Autorizácia na strane akčnej metódy

Ak sa pokúsi túto metódu zavolať užívateľ, ktorý nepatrí do role admin, akcia nebude vykonaná.

7.2.4.2. Evidencia užívateľov a definovanie užívateľských rolí

Pri registrácií je od užívateľa štandardne požadované meno, heslo a email. Administrátorovi systému je umožnené pracovať s evidenciou užívateľov. Podporu akcií zabezpečuje `AccountController`. Po zavolaní akcie List je administrátorovi ponúknutá tabuľka všetkých užívateľov.

Užívateľ je v systéme definovaný ako `MembershipUser`. Zobrazené sú vybrané Properties ako: `UserName`, `Email`, `CreationDate`, `LastLoginDate`, `IsOnline`. Okrem týchto informácií je pomocou `checkboxov` a metódy `bool IsInRole(string userName, string roleName)` zobrazené do akej role užívateľ patrí. Po registrácií nie je priradený do žiadnej role. Túto operáciu musí vykonať administrátor systému. Po zaškrtnutí požadovaného checkboxu je automaticky potvrdený formulár a zavolaná akčná metóda ktorej hlavička je popísaná v Zdrojový kód 17. Metódy, pomocou ktorých sa užívateľ do role pridáva alebo odoberá je uvedené v Zdrojový kód 18.

```
[Authorize(Roles = "admin")]
public ActionResult MembershipAction(bool adminCheckBox, bool driverCheckBox, string userName, string deleteButton, string resetButton)
```

Zdrojový kód 17: Hlavička akčnej metódy pre operácie nad užívateľskými účtami

- `Roles.AddUserToRole(userName, "admin");`
- `Roles.RemoveUserFromRole(userName, "admin");`

Zdrojový kód 18: Spôsob pridávania a odoberania užívateľa do role

Akčná metóda `MembershipAction` je volaná aj pri ostatných operáciách nad užívateľskými účtami ako mazanie užívateľa či reset hesla. Reset hesla je prevádzaný prostredníctvom nasledujúceho kódu:

```
string newPass = Membership.GetUser(userName).ResetPassword();
```

Zdrojový kód 19: Reset užívateľského hesla

Reťazec s novým heslom je následne odosielaný na View a administrátorovi zobrazený. Navrhnutý modul pre správu účtov administrátorovi ponúka kompletnú manipuláciu a riedenie účtov jednotlivých užívateľov.

7.2.5. Ukladanie dát a práca s databázou

Tento projekt na ukladanie dát využíva MS SQL databázu. Databázový súbor s názvom `DBcarAnalysis.mdf` je umiestený v adresári `App_Data`. Ako technológia pre prístup k dátam bola vybraná LINQ[25] to SQL. Okrem tejto technológie ASP .NET podporuje mnoho ďalších ako napr. ADO. NET, Entity Framework a podobne.

LINQ to SQL je v podstate ORM[26]. Súčasťou .NET frameworku je od verzie 3.5. Umožňuje mapovať jednotlivé tabuľky v databáze na štandardné triedy a pomocou nich spracovávať dáta. Verejné vlastnosti týchto tried odpovedajú jednotlivým atribútom v tabuľke a jedna inštancia odpovedá jednému riadku v tabuľke.

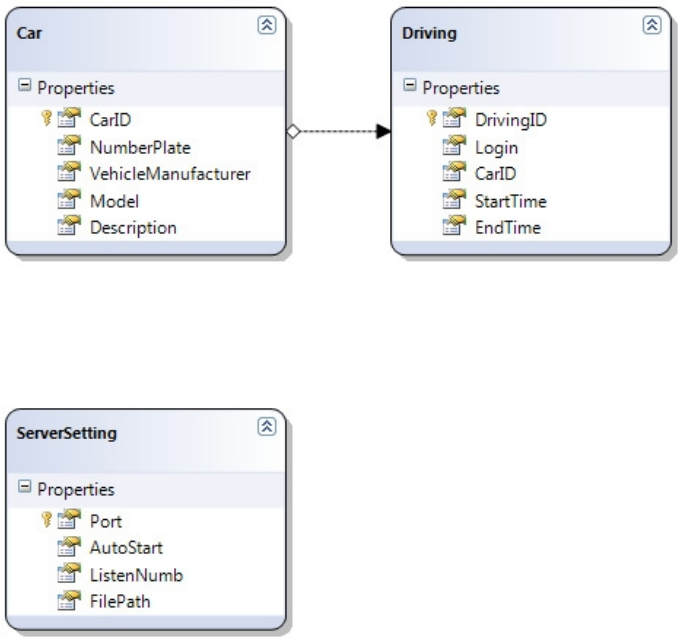
Táto aplikácia obsahuje jednu databázu a niekoľko tabuliek. Jedná sa hlavne o evidencie potrebných dát. Pre evidenciu užívateľov sa využíva Membership Provider model, ktorý je súčasťou ASP .NET projektu. Prácu s týmto modelom popisuje kapitola 7.2.4.Správa a riadenie užívateľských účtov.

Ďalšiu tabuľku tvorí evidencia vozidiel s názvom `Car`. Administrátorovi systému je umožnené túto tabuľku editovať. Jednotlivé atribúty tabuľky sú zobrazené v Obrázok 13: Tabuľky v databáze.

Aplikácia si v databáze ukladá konfiguračné dáta pre fungovanie TCP servera. Jedná sa o tabuľky `ServerSettings`. Štruktúra je opäť popísaná v Obrázok 13: Tabuľky v databáze.

Tabuľka `Driving` sa používa na ukladanie dát o jednotlivých jazdách. Obsahuje cudzí kľúč na tabuľku `Car`, teda ku každej jazde musí byť priradené vozidlo. Ďalej sa do tejto tabuľky priraďuje informácia o užívateľovi. Konkrétne sa jedná o atribút `Login`. Nový riadok v tabuľke `Driving` sa tvorí po zaslaní správy `StartCapture`, ktorú zasiela klientská aplikácia serverovej. O tejto komunikácií hovorí kapitola 7.1.8.2.Odosielanie nameraných dát na server . Treba podotknúť, že atribút `Login` je v podstate užívateľské meno, ktoré je odosielané z klientskej aplikácie, samozrejme následne overené. Užívateľ musí patriť do role Driver. Ďalší atribút, ktorý sa do riadku tabuľky vloží je `StartTime`. Tento atribút je získaní pri inicializácií jazdy na klientskej strane. Ďalej sa vytvorí súbor, do ktorého sú ukladané dáta. Po ukončení jazdy sa do riadku tabuľky vloží atribút `EndTime` opäť generovaný klientskou aplikáciou. Detailnejší popis akcií pri vytváraní jazdy a prijímaní dát z klientskej aplikácie poskytuje kapitola 7.2.3.3.Príjem dát.

25 Language-Integrated Query
26 Object relational mapper



Obrázok 13: Tabuľky v databáze

7.2.5.1. Implementácia LINQ to SQL do projektu

Súbory zdrojových kódov sú prehľadne rozdelené do adresárov podľa úlohy (Models, Views, Controllers). Pre správne fungovanie LINQ to SQL musí byť vytvorená tzv. LINQ to SQL trieda. Táto trieda je umiestená práve v adresári `Models`. Má názov `CarDataAnalysis.dbml`. Po vytvorení tohto súbora je otvorený „desinger“, do ktorého sa prostredníctvom „drag-and-drop“ postupne prenášajú databázové tabuľky. Systém automaticky vygeneroval triedy, ktoré sú reprezentáciou databázových tabuliek. Tak isto určil aj reláciu medzi tabuľkami `Car` a `Driving` ako „one-to-many“. Obrázok 13: Tabuľky v databáze je v podstate ukážka „desingera“ triedy `CarDataAnalysis.dbml`.

Spolu s triedami reprezentujúcimi databázové tabuľky sa vytvorí aj tzv. DataContext. Táto trieda predstavuje akýsi „medzikus“ medzi databázou a triedami modelu. Jej názov je generovaný automatiky a je závislý na pomenovaní projektu. Keďže tento projekt má názov `CarDataAnalysis`, DataContext je pomenovaný ako `CarDataAnalysisDataContext`. Dôležité je, že trieda obsahuje vlastnosti `Cars` a `Drivings`, pomocou ktorých sa s tabuľkami ďalej pracuje.

Pre komunikáciu controllera s DataContextom je použitý návrhový vzor Repository. LINQ dotazy nie sú uvedené v cotrolleroch. V adresári `Models` sú vytvorené triedy – `repository` všetkých databázových tabuliek. Tieto repository obsahujú metódy pre operácie nad tabuľkami a pristupujú na ne jednotlivé controllery. Každý repositor musí obsahovať inštanciu na DataContext resp. triedu `CarDataAnalysisDataContext`.

Jednotlivé metódy v repositoroch obsahujú LINQ dotazy. Dotazy sú automaticky premapované do príslušných SQL dotazov. S premapovaním je spojená určitá výkonnostná réžia. Tá je však skoro zanedbateľná. Príklad metódy v repositoroch s LINQ dotazom:

```
public void Add(Car car)
  {
    db.Cars.InsertOnSubmit(car);
  }
```

Zdrojový kód 20: Vloženie objektu typu Car do tabuľku

Validácia vstupu

Po nedefinovaní modelových tried prostredníctvom desingera majú vlastnosti týchto tried totožné dátové typy ako v databázových tabuľkách. Pri pokuse o prácu s nezodpovedajúcim dátovým typom aplikácia vyhlási výnimku.

Riešenie validácie v tomto projekte je rozšírené o ďalšiu funkcionalitu. Jedná sa hlavne o ošetrenie nevyplnených povinných častí formulárov. Implementácia spočíva v doplnení tried do adresára model. Jedná sa o triedy, ktoré budú rozširovať modelové triedy vytvorené desingerom. Pri deklarácií je nutné použiť kľúčové slovo `partial` (doplnenie implementácie triedy vytvorenej generátorom). Táto trieda obsahuje vlastnosť `public bool IsValid` a metódu `public Ienumerable<RuleViolation> GetRuleViolations()`. Pre potrebu tejto metódy je definovaná trieda `RuleViolation`, ktorá reprezentuje vlastnosti ako `ErrorMessage`, `PropertyName` a umožňuje tak niesť dodatočné informácie o porušení validity. Popis fungovania validátora je nasledovný:

- metóda `partial void OnValidate(ChangeAction action)` je volaná v momente požiadavky na uloženie dát do tabuľky databázi. V tele tejto metódy je overovaná vlastnosť `bool IsValid`,
- trieda `public Ienumerable<RuleViolation> GetRuleViolations()` na základe vnútornej implementácie vracia kolekciu typu `RuleViolation`. Tá je vytvorená na základe porušenia jednotlivých validačných pravidiel,
- vlastnosť `bool IsValid` vracia hodnotu `true` ak je veľkosť kolekcie, ktorú vracia `GetRuleViolations()` nulová.

Ak dáta nespĺňajú validačné pravidlá, formulár nebude spracovaný a na View budú zobrazené chybové hlášky, ktoré sú nastavené v metóde `public Ienumerable <RuleViolation> GetRuleViolations()`. Nasledujúci zdrojový kód ukazuje na spracovanie validačných správ.

```
<%= Html.ValidationMessage("NumberPlate", "*")%>
```

Zdrojový kód 21: Spracovanie validačných správ

Premenená `NumberPlate` je v podstate `PropertyName` z triedy `RuleViolation`. Namiesto tejto premennej bude na View zobrazená validačná správa.

7.2.6. Vizualizácia dát a zobrazovanie jázd

Serverová aplikácia umožňuje vizualizovať dáta, ktoré sú výsledkom zberu na strane klienta v prostrední Google Earth. Využíva na to Google Earth API dostupné prostredníctvom KML súborov. O možnostiach Google Earth API hovorí kapitola 6.1.2.Google Earth API.

7.2.6.1. Evidencia jázd

Záznamy v databázovej tabuľke `Driving` sú tvorené tak, ako to popisuje kapitola 7.2.3.3.Príjem dát. Užívateľovi sú tak na základe vlastnosti `Page.User.Identity.Name` poskytnuté údaje o jeho jazdách.

Nad tabuľkou `Driving` sú implementované základné operácie. Pridávanie dát je riešené požiadavkami z klientskej časti aplikácie. Operácie nad jazdami spracúva nasledujúca akčná metóda kontrolera DrivingsController.

```
public ActionResult ItemAction(string deleteButton, string simulatorButton, string earthButton, string earthButton2, string selectSenzor, int drivingID)
```

Zdrojový kód 22: Akčná metóda pre spracovanie operácií nad záznamov jazdy

Po odoslaní formulára v časti View sa prostredníctvom metódy Post odošlú parametre tejto metódy. Ovládacie prvky sú riešené ako elementy `Input` typu `image`. Akčná metóda tak vie rozhodnúť o akú akciu sa jedná pomocou ich hodnoty. Operáciu spracuje a vráti odpovedajúci View. Pri zobrazení evidencie je pri každej jazde vytvorený náhľad, vytvorený prostredníctvom PrieviewParsera. Použitá je technológia statickej Google mapy. Bližší pohľad na implementáciu je obsiahnutý v nasledujúcej kapitole.

7.2.6.2. Vytvorenie KML súborov a formy zobrazenia dát

KML súbory sú vytvárané z dátových podkladov, ktoré sú obsiahnuté v textových súboroch pridružených k danej jazde. Princíp tvorby textových súborov je obsiahnutý v kapitole 7.2.3.3.Príjem dát.

Implementovaná je sada parserov, ktoré KML súbory vytvárajú. Triedy sú umiestnené v adresári Parsers. Celkovo sú použité tri parsery:

- MapsParser,
- PreviewParser,
- SimulatorParser.

MapsParser

Tento parser je volaný z `DrivingsControllera`. Ako parametre sa predávajú cesta k zdrojovému textovému súboru a veličina, ktorá je požadovaná na vizualizáciu.

Po vytvorení potrebných streamov na textový súbor je následne zavolaná metóda `public double[] BoundaryValues()`, ktorá má za úlohu vypočítať hraničné hodnoty veličiny. Rozlíšenie zobrazovanej hodnoty v prostredí Google Earth je riešené prostredníctvom farby vykresľovanej úsečky. K dispozícií je 5 typov farieb, ktorými môže byť úsečka vykreslená. Metóda `BoundaryValues()` v konečnom dôsledku určí rozsah hodnôt a rozdelí ich do 5 rovnakých dielčích rozsahov. Vypočítané hraničné hodnoty sú odoslané do View a zobrazené v legende s príslušnými farbami, aby bolo možné jednoducho určiť, do akého rozsahu hodnota patrí.

V ďalšom kroku je zavolaná metóda `public void CreateKML(int mode)`, ktorá samotný KML súbory vytvorí. Súbor je vytvorený za pomoci triedy `XmlDocument`. Parser je

navrhnutý tak, aby dokázal tvoriť KML súbory dvoch rôznych typov. V prvom prípade je požadovaná veličina pri vizualizácií odlíšená len farbou úsečiek, v druhom prípade je použité odlíšenie farbou a naviac je hodnota veličiny premietnutá na mape ako parameter nadmorskej výšky. V tomto prípade sa dajú lepšie identifikovať menšie zmeny, ku ktorým v meranej veličine dochádzalo. Identifikácia varianty je riešená podľa parametra metódy `int mode`.

Samotný KML súbor určený pre tento typ zobrazovania musí po úvodných hlavičkách súboru obsahovať deklarácie jednotlivých farebných štýlov. Definuje sa farba a šírka úsečky. Príklad deklarácie štýlu

```
<Style id="1">
    <LineStyle>
       <color>FFFF0000</color>
       <width>5</width>
    </LineStyle>
</Style>
```

Zdrojový kód 23: Príklad štruktúry deklarácie štýlu v KML súbore

V ďalšej časti KML súbora sa nachádzajú tzv. Placemarky. Tie majú definovaný farebný štýl a ďalej obsahujú koordinanty, respektíve súradnice, podľa ktorých sa daná úsečka vykreslí. Paraser pri vytváraní KML súbora sleduje odchýlku hodnoty od prechádzajúcej. Ak spadá do iného rozsahu, potom vytvorí nový Placemark s iným farebným štýlom. Príklad jedného Placemarku môže vyzerať nasledovne:

```
<Placemark>
      <styleUrl>#2</styleUrl>
      <LineString>
          <coordinates>
                28.140055, 36.4161683333333, 0.5
                28.1402033333333, 36.416205, 1.7
                28.1402033333333, 36.416205, 1.7
                28.1403483333333, 36.416245, -0.8
          </coordinates>
      </LineString>
</Placemark>
```

Zdrojový kód 24: Príklad štruktúry deklarácie úsečky v KML súbore

V koordinantoch je umiestnená informácia o zemepisnej šírke a výške a nadmorskej výške. Vo variante s vizualizáciu meranej veličiny ako nadmorskej výšky je jednoducho výška nahradená meranou veličinou. Zobrazenie mapy v základnej a rozšírenej variante je možné vidieť v časti Prílohy na Obrázok 27: Zobrazenie meranej veličiny v základnom móde a Obrázok 28: Zobrazenie meranej veličiny interpretované ako nadmorská výška. Diagram triedy MaspParser je dostupný v časti Prílohy ako Triedny diagram 10: Serverová aplikácia: MapsParser.

PreviewParser

Tento parser je využívaný pri vytváraní jednoduchých náhľadov jázd pri zobrazovaní celej evidencie. Plní dve základné operácie:

- zistenie použitých PIDs, pre zobrazenie výberu meranej veličiny,
- vykreslenie jednoduchého náhľadu jazdy.

Zisťovanie použitých PIDs prebieha na základe analýzy zdrojového textového súbora. Parser pozná jednotlivé umiestnenia PIDs v riadkov súbora a na základe toho či obsahujú hodnotu určí či je PID použité. Do View sa tento údaj vracia za pomoci metódy `public List<string> GetSensorsUsed()` ako zoznam reťazcov.

Tvorbu statickej Google mapy zabezpečuje metóda `public string Parse()`. Statická mapa sa k danému využitiu hodí. Pri náhľade nie sú nutné detailné informácie a možnosť ovládania mapy. Spotrebovaný výkon na vykreslenie je podstatne menší ako pri použití dynamickej mapy generovanej na základe KML súbora.

Zobrazenie mapy je riešené veľmi jednoducho: vložením obrázka s generovanou URL adresou. Tvar URL adresy je nasledovný:

```
http://maps.google.com/maps/api/staticmap?path=color:0xFF0000|
weight:3
```

Definovaná je farba, pomocou ktorej je jazda zobrazená a veľkosť čiary. Za URL adresou nasleduje rad informácií o zemepisnej dĺžke a šírke. Parser je navrhnutý tak, aby vybral len význačné informácie a tak zabránil príliš dlhej URL adrese. Rozlíšenie vkladanej mapy je 280 x 120 pixelov. Umiestnenie mapy je možné vidieť v časti Prílohy na Obrázok 24: Serverová aplikácia: zoznam jázd.

SimulatorParser

Tento parser vytvára KML súbory pre DriveSimulator. Funguje na podobnom princípe ako MapsParser, akurát všetky koordinanty sa nachádzajú v jednom bloku Placemark. DriveSimulator dokáže simulovať rýchlosť vozidla. Tento údaj je do KML súbora vložený namiesto parametra nadmorskej výšky.

Na začiatku KML súbora je definovaná počiatočná pozícia. Na túto pozíciu sa umiestni vozidlo pred samotným štartom simulátora.

```
<LooAt>
    <longitude>28.0889866666667</longitude>
    <latitude>36.4016033333333</latitude>
    <altitude>0</altitude>
    <range>500</range>
    <tilt>45</tilt>
    <heading>36.4016033333333</heading>
</LooAt>
```

Zdrojový kód 25: Príklad štruktúry deklarácie počiatočnej pozície v KML súbore

Nakoniec nasledujú jednotlivé koordinanty, po ktorých sa bude vozidlo pohybovať. Simulátor najlepšie pracuje ak majú každé dva po sebe idúce koordinanty rôzne súradnice. Aplikácia to pri vytváraní KML súbora zohľadňuje a prípadným rovnakým súradniciam zabráni vloženie. Z toho vyplýva jednoznačné obmedzenie, že drive simulátor nedokáže simulovať vozidlo, ktoré sa nepohybuje. Ak reálne pri jazde došlo k zastaveniu vozidla, simulátor túto skutočnosť bude ignorovať a bude pokračovať na ďalšej zmenenej súradnici jazdy.

Simulátor vychádza z hotového riešenia dostupného na umiestnení, ktoré uvádza literárny zdroj [18]. Jedná sa o Open Source riešenie, teda je možné zdrojový kód aplikácie upraviť a použiť. Aplikácia využíva JavaScriptové knižnice z ukážkového riešenia od spoločnosti Google, ktoré je dostupné ako umiestnenie literárneho zdroja [19].

Aplikácia pracuje tak, že po načítaní stránky je možné spustiť parsovanie KML súbora. Pre zobrazenie vozidla na mape sa využíva súbor typu KMZ, ktorý vo vnútri archívu obsahuje grafiku vozidla.

Do tohto riešenia teda boli vložené potrebné JavaScriptové knižnice. Nachádzajú sa v adresári `Scripts` spolu s ostatnými knižnicami. HTML časti boli presunuté na korešpondujúce View. Časti kódu, ktoré sa starajú o načítavanie KML a KMZ súborov boli upravené tak, aby o súbory žiadali dynamicky na základe požiadavky zobrazovanej jazdy.

Odosielanie súborov na webový prehliadač je riešené podobne ako pri predchádzajúcich módoch zobrazenia. Rieši to `DownloadController`. Informuje o tom nasledujúca kapitola. Vizuálny motív mapy so simulátorom jazdy je možné vidieť v časti Prílohy na Obrázok 29: Zobrazenie jazdy pomocou simulátora.

7.2.6.3. Odosielanie KML súborov

Kapitola 6.1.2.1.KML hovorí o možnosti využitia súborov tohto typu pre vizualizáciu dát v prostredí Google Earth. Táto aplikácia vytvára príslušné KML súbory na požiadavky, ktoré prichádzajú z webových prehliadačov. Samotné súbory sú teda spracované na strane webového prehliadača. Súbory teda musia byť na prehliadač odoslané.

Príslušný KML súbor je vytvorený po zavolaní akčnej metódy `ActionResult`. Funkcionalitu odosielania súborov pokrýva `DownloadController`. Konkrétne sa jedná o metódy:

- `public ActionResult FileKml(int id),`
- `public ActionResult FileKmlMode2(int id),`
- `public ActionResult FileSimulatorKml(int id).`

Zdrojový kód 26: Metódy pre vytváranie KML súborov

Vyhľadanie súborov je riešené podľa vopred definovanej cesty. Parametrom metód je ID danej jazdy, tak ako je uložená v databázovej tabuľke. Pomocou ID sa vyberie a identifikuje príslušný súbor. Sťahovanie KML súbora je inicializované z JavaScriptu. Konkrétne sa jedná o kód funkcie `function createNetworkLink()`. K URL adrese, ktorú Javascript volá bolo nutné pripojiť časové razítko pomocou `new Date().getTime()`. Riešené je to metódou typu GET. Dôvod pridávania spočíva vo funkcií cache pamäti prehliadačov. Keby bola URL adresa stále rovnaká, prehliadač by použil KML súbor z pamäti cache, čo by zabraňovalo zmene zobrazovaných dát na základe výberu zobrazovanej veličiny. Nasledujúci kód popisuje odosielanie KML súborov.

```
Response.ContentType = "APPLICATION/OCTET-STREAM";
System.String disHeader = "Attachment; Filename=\"" + filename +
                "\"";
Response.AppendHeader("Content-Disposition", disHeader);
FileInfo fileToDownload = new FileInfo(filePath);
Response.WriteFile(fileToDownload.FullName);
Response.End();
Response.Cache.SetCacheability(HttpCacheability.NoCache);
```

Zdrojový kód 27: Odosielanie KML súborov

8. Záver a zhodnotenie práce

Cieľom tejto diplomovej práce bolo vytvoriť systém, ktorý umožňuje vizualizáciu vybraných dát z motorových vozidiel. Zber dát vo vozidle vykonáva aplikácia v mobilnom zariadení. Údaje sú zasielané na server, ktorý dáta vizualizuje.

Pred začatím samotnej realizácie bolo nutné zmapovať súčasný software, ktorý podobné funkcie zabezpečuje. Jedná sa o software, ktorý spracovával a zobrazoval dáta z riadiacej jednotky motorového vozidla prostredníctvom protokolu OBD II.

Po ozrejmení základných náležitostí som mohol pristúpiť k samotnej implementácií tohto projektu. Tá pozostávala z dvoch základných častí: serverovej a klientskej aplikácie. Pri implementovaní som si musel ozrejmiť určité technológie a postupy, ktoré sú nutné pri tvorbe projektu tohto typu. V klientskej časti sa jednalo hlavne o implementáciu aplikácie pre mobilné zariadenia v .NET Compact Framework. Komunikácia a spracovanie dát z GPS prijímača a spracovanie dát z prevodníka ELM. V neposlednom rade bolo nutné vytvoriť vlastný element typu UserControll, ktorý slúžil na zobrazovanie aktuálnych dát. Pre možné výpadky sieťového pripojenia bolo nutné upraviť aplikáciu pre prácu v slabej konektivite alebo v úplne odpojenom režime. V serverovej časti sa jednalo hlavne o vytvorenie jednoduchého informačného systému a zvládnutie vykresľovania dát v prostredí Google Earth.

Pri nasadení tohto systému do praxe by celá aplikácia musela prejsť dôkladným testovaním, ktoré by určite odhalilo radu problémov. Vhodné by bolo doimplementovať rôzne rozširujúce funkcie, ako napríklad výpočet aktuálnej spotreby, odhadovanie času dojazdu do cieľa, mazanie chybových hlásení vozidla a rozšírenie počtu podporovaných senzorov. Výhodné by bolo naimplementovať klientské aplikácie pre rôzne platformy a operačné systémy.

Dá sa povedať, že zadanie tejto diplomovej práce som splnil v celom rozsahu. Po celkovej realizácií sa stala táto práca pre mňa jednoznačne prínosom z hľadiska ozrejmenia rôznych technológií a v poukázaní na možnosti využitia výpočtovej techniky v motorových vozidlách.

9. Literatúra

[1] Surface Vehicle standard (SAE J1979) - *Equivalent to ISO/DIS 15031-5:April 30, 2002*

[2] QContinuum Freeware, *OBD Gauge* [online]. [cit 25. 3. 2011]. Dostupné z:http://www.qcontinuum.org/obdgauge

[3] My scan tools-*ProScan* [online]. [cit 27. 3. 2011]. Dostupné z: http://www.myscantool.com/

[4] OBDTester.com - *pyOBD - Open-source OBD-II diagnostics* [online]. [cit 28. 3. 2011]. Dostupné z: http://www.myscantool.com/

[5] OBDsoftware.net - *TochScan* [online]. [cit ň3. 4. 2011]. Dostupné z: http://www.obdsoftware.net/TouchScanInfo.aspx

[6] Ahmed El- Rabanny - *Introduction to GPS*, Artech House Inc. 2002, ISBN 1- 58053- 138- 0

[7] Petr Rapant, VŠB-TU Ostrava, Institut geoinformatiky - *Základy geoinformatiky XIII Geoinformační technologie Systém GPS* [online]. [cit 29. 3. 2011]. Dostupné z: http://gis.vsb.cz/Rapant/vyukove_materialy/ZS/ZGI/Prezentace/Z_GInf_2009_13.pdf

[8] Changu Thota – *Programming MapPoint in .NET, O´REAILLY 2006,* ISBN 0- 596- 00906- 2

[9] ELM 327 datasheet - *OBD to RS232 Interpreter* [online]. [cit 6. 4. 2011]. Dostupné z: http://elmelectronics.com/DSheets/ELM327DS.pdf

[10] Geospatial Training Services - *Mashup Mania with Google Maps, Version 5: Updated January 2009*

[11] *Google Maps API Family* [online]. [cit 26. 4. 2011]. Dostupné z: *http://code.google.com/intl/sk/apis/maps/index.html*

[12] Gabriel Svennerberg – *Beginning Google Maps API 3,* Apress 2010, ISBN 978-1-4302-2803-5

[13] *Google Earth API Reference* [online]. [cit 27. 4. 2011]. Dostupné z: http://code.google.com/intl/sk/apis/earth/documentation/reference/index.html

[14] Sterling Udell- *Beginning Google Maps mashups with mapplets, KML and GeoRSS: from novice to Professional*, Apress 2008, ISBN 978-1-4302-1621-6

[15] *How do I Create a 7-Segment (LED type) Numeric Display in ComponentWorks?*[online]. [cit 12. 4. 2011]. Dostupné z: http://digital.ni.com/public.nsf/allkb/5117E438FDF05A48862564EA00798D3F

[16] Staven Sanderson – *Pro ASP. NET MVC 2 framework*, second edition, Apress 2010, ISBN 978-1-4302-2886-8

[17] Simone Chiaretta, Keyvan Nayyeri - *Beginning ASP.NET MVC 1.0,* John Wiley & Sons, 2009, ISBN 9780470433997

[18] *Real-time Path Touring with the Google Earth Plug* [online]. [cit 28. 4. 2011]. Dostupné z: http://www.thekmz.co.uk/2009/02/real-time-path-touring-with-the-google-earth-plugin/

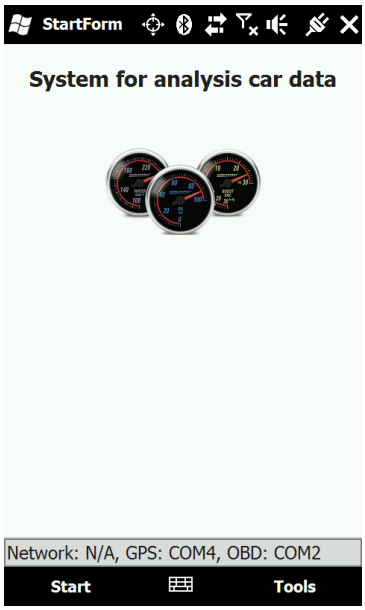
[19] Driving simulator svn - earth-api-samples - *Revision 120: /trunk/demos/drive-simulator* [online]. [cit 28. 4. 2011]. Dostupné z: http://earth-api-samples.googlecode.com/svn/trunk/demos/drive-simulator/

10. Prílohy

A) Užívateľská príručka, časť klient

Aplikácia sa do mobilného telefónu inštaluje prostredníctvom CAB súbora štandardným spôsobom. Po inštalácií sa do menu zariadenia pridá položka Car Client Mobile. Týmto odkazom sa celá aplikácia spúšťa.

Po spustení aplikácie sa zobrazí úvodná obrazovka. V spodnej časti je zobrazený panel, v ktorom sú vypisované základné operačné parametre aplikácie. Jedná sa o stav sieťového pripojenia a sériové porty, prostredníctvom ktorých komunikuje GPS prijímač a prevodník ELM 327.



Obrázok 14: Klientská aplikácia: úvodná obrazovka

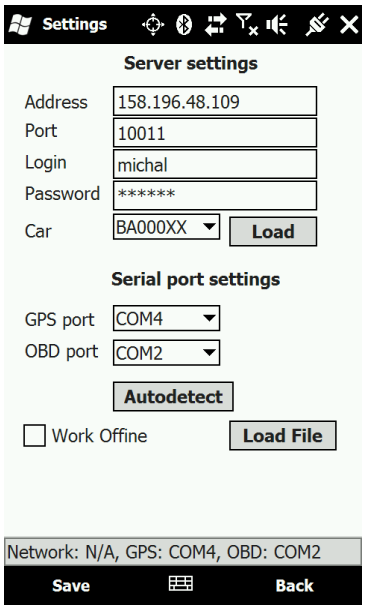
Pri prvotnom spustení aplikácie je nutné vykonať počiatočné nastavenia. Prístup k obrazovke s nastaveniami je prostredníctvom Menu -> Tools -> Settings. Pre komunikáciu so serverom je nutné nastaviť cieľovú IP adresu, TCP port, prihlasovacie meno a heslo.

Po kliknutí na tlačidlo load je možné načítať zoznam vozidiel zo servera. Samozrejmosťou sú korektne vyplnené údaje pre prístup k serveru. Pri nekorektne vyplnených alebo neplatných prihlasovacích údajoch je na obrazovku vypísaná informačná hláška: Connection problem- Unable connect or login to server.

V ďalšej časti je možné nastaviť sériové komunikačné porty pre periférne zariadenia. Konkrétne sa jedná o GPS prijímač a prevodník ELM 327. Je možné použiť aj autodetekciu, kedy sa aplikácia pokúsi sama určiť, na ktorých sériových portoch periférne zariadenia komunikujú.

Ďalej je možné nastaviť či má aplikácia pracovať v stave Offline. Ak bude tento stav vybraný, dáta sa budú ukladať do textového súbora, ktorý bude možno po ukončení jazdy vyhľadať a v pripojenom režime na server odoslať. Pre výber súboru slúži tlačidlo Load File. Súbory sú pomenované podľa času a dátumu začiatku jazdy. Pre odosielaní súborov je opäť nutné, aby boli správne vyplnené údaje pre prístup k serveru.

Po zadaní platných údajov je možné celé nastavenia aplikácie uložiť. Slúži na to tlačidlo Save v menu aplikácie. Pri ukladaní nekorektného alebo chýbajúceho záznamu, aplikácia vypíše chybovú hlášku. V nej bude uvedené, ktorý údaj nie je v poriadku. Pre návrat na predchádzajúci formulár sa používa tlačidlo Back.



Obrázok 15: Klientská aplikácia: nastavenia

Ak je aplikácia správne nastavená, môže prejsť do stavu spracovávania dát. Cesta do tejto sekcie vedie cez Menu -> Start. Nasledujúce okno obsahuje viac tabov. Tie slúžia na vizualizáciu jednotlivých meraných hodnôt a nastavenia nahrávania a zobrazenia.

Obrázok 16: Klientská aplikácia: meranie dát

Obrázok 17: Klientská aplikácia: meranie dát nastavenie senzorov

Pred začatím merania je nutné previesť skenovanie podporovaných PID vo vozidle. Skenovanie sa prevádza v tabe Setup po stlačení tlačidla Scan na konci formulára. Skenovanie by nemalo trvať viac ako 5 sekúnd. Po skončení operácie je zobrazená hláška, ktorá buď informuje o úspešnom zistení senzorov alebo oznamuje problém. Ďalej je možné zaškrtnúť požadované senzory či už pre prezeranie alebo aj pre nahrávanie a určiť im prioritu merania pomocou počtu tiketov. Čím väčší počet tiketov má daný senzor, tým častejšie bude prevádzané jeho meranie. Na konci je potrebné celé nastavenie uložiť pomocou tlačidla Save na konci tohto formulára.

Meranie môže byť spustené v dvoch módoch:

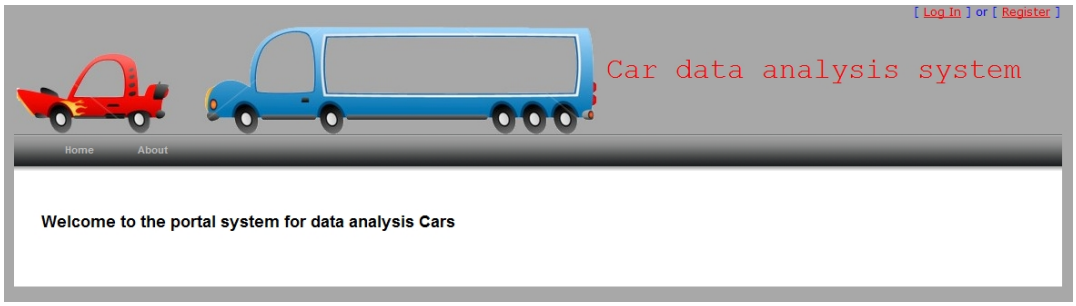
- zobrazovanie,
- zobrazovanie a nahrávanie.

Tieto akcie sa spúšťajú prostredníctvom Menu -> Start -> View alebo Menu -> Start -> View and Capture. Ukončenie zobrazovania a nahrávania je riešené cez Menu -> Start -> Stop.

B) Užívateľská príručka, časť server

Pred prvým spustením servera je nutné aby bol v IIS nastavený AspNetSqlRoleProvider. Ďalej musia byť definované role admin a driver, musí byť vytvorený jeden užívateľ, ktorý bude patriť do role admin. Bez týchto náležitostí nebude možné s aplikáciou plne pracovať.

Aplikácia štartuje v úvodnej sekcií, kde je možné sa zaregistrovať alebo sa prihlásiť už existujúcimi prihlasovacími údajmi. Slúžia na to odkazy Log In a Register v hornej pravej časti obrazovky.

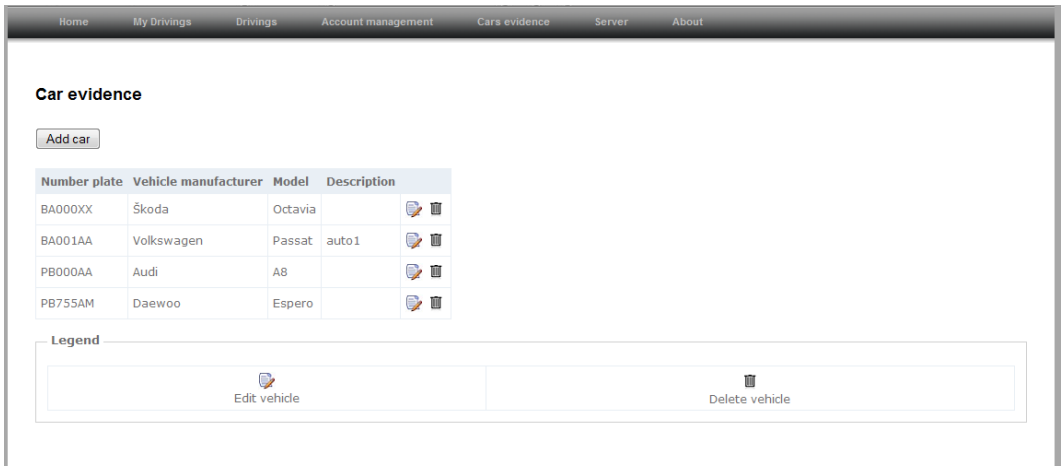


Obrázok 18: Serverová aplikácia: úvodná sekcia

Pri prihlásení užívateľa, ktorý patrí do role admin je k dispozícií kompletné riadenie aplikácie. Stav systému po prihlásení ukazuje nasledujúca obrazovka.

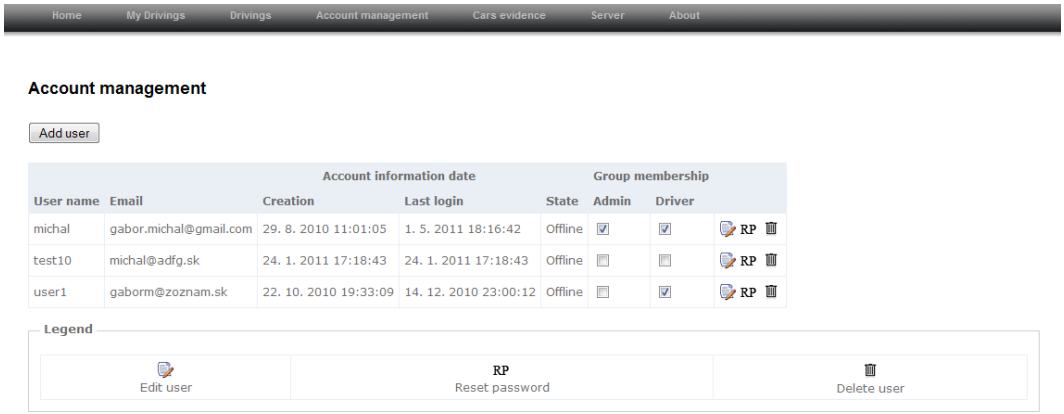
Obrázok 19: Serverová aplikácia: obrazovka po prihlásení administrátora do systému

Evidencia vozidiel je opäť prístupná len administrátorom systému. Po otvorení tejto sekcie je možné zobraziť aktuálne pridané vozidla v databáze, editovať, mazať či pridávať nové.



Obrázok 20: Serverová aplikácia: evidencia vozidiel

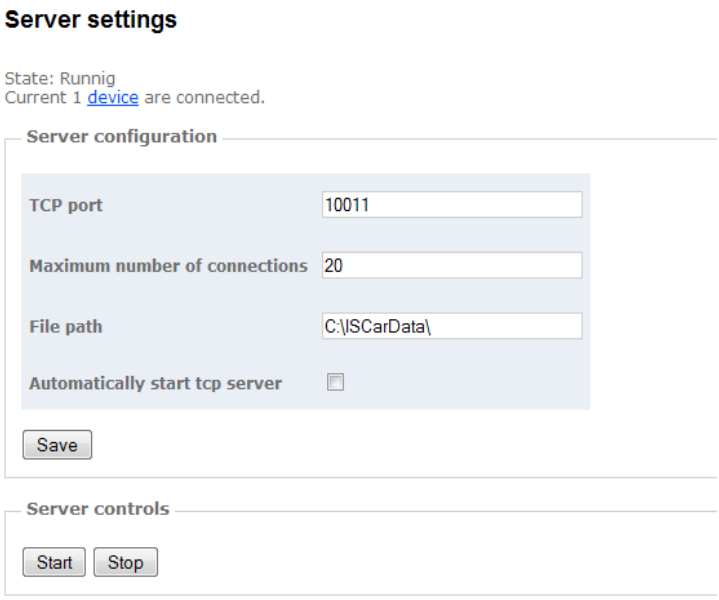
Ďalej môže administrátor spravovať užívateľské účty. Evidencia účtov je spustená po kliknutí na odkaz Account Management. Po otvorení je zobrazený zoznam užívateľov s vybranými atribútmi. Správca môže prideliť užívateľov do štandardných rolí, editovať, zmazať, či resetovať heslo.



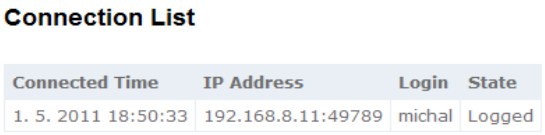
Obrázok 21: Serverová aplikácia: evidencia užívateľov

Ďalšia významná sekcia je Server. Je tu možné nastaviť základné parametre TCP servera, ktorý bude spracovávať spojenia od klientov. Je možné nastaviť TCP port, maximálny počet pripojených klientov, cestu k ukladaniu súborov. Na tomto mieste je možné server zapínať a vypínať. Obrazovka nastavenia servera je na Obrázok 22: Serverová aplikácia: nastavenie TCP servera.

V hornej ľavej časti tejto obrazovky je možné sledovať počet aktívnych spojení. Po kliknutí na toto na odkaz device sa zobrazia detaily o jednotlivých spojeniach. Obrazovka je na Obrázok 23: Serverová aplikácia: zoznam spojení od klientských aplikácií.



Obrázok 22: Serverová aplikácia: nastavenie TCP servera

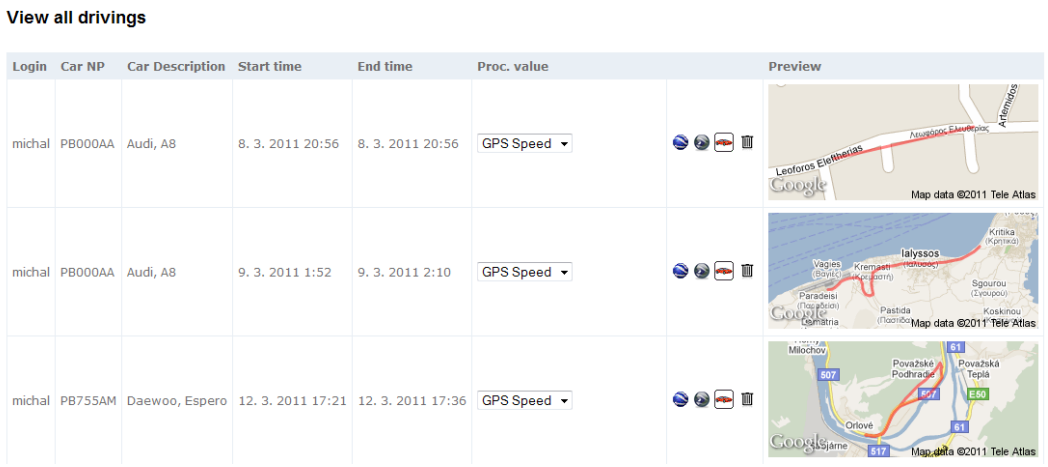


Obrázok 23: Serverová aplikácia: zoznam spojení od klientských aplikácií

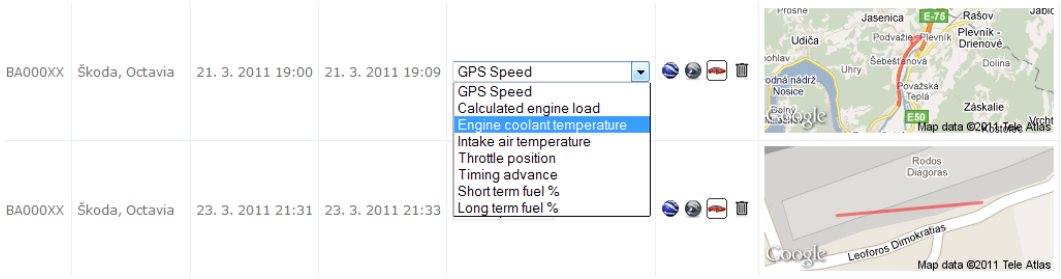
Najhlavnejšou časťou serverovej aplikácie je zobrazenie jázd a ich následná vizualizácia na mape. Prístup k evidencií jázd je možný prostredníctvom odkazov v menu My Drivings a Drivings. Administrátorovi systému je umožnené sledovať jazdy všetkých užívateľov. Bežnému užívateľovi sa zobrazuje len položka My Drivings, kde sú k dispozícií jazdy daného užívateľa.

Obrázok 24: Serverová aplikácia: zoznam jázd ukazuje ako vyzerá zoznam všetkých jázd. V tabuľke vidno aj dodatočné atribúty jázd. V pravej časti každej jazdy je zobrazený jednoduchý náhľad, riešený prostredníctvom statickej mapy.

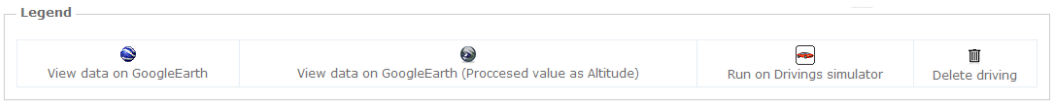
Obrázok 25: Serverová aplikácia: jazdy, výber zobrazovanej veličiny Zobrazuje možnosti výberu zobrazovanej veličiny. V tomto comboboxe je možné vybrať senzor, ktorého hodnota sa bude ďalej spracovávať, respektíve zobrazovať. Význam ovládacích prvkov je uvedený v Obrázok 26: Serverová aplikácia: jazdy, legenda.



Obrázok 24: Serverová aplikácia: zoznam jázd



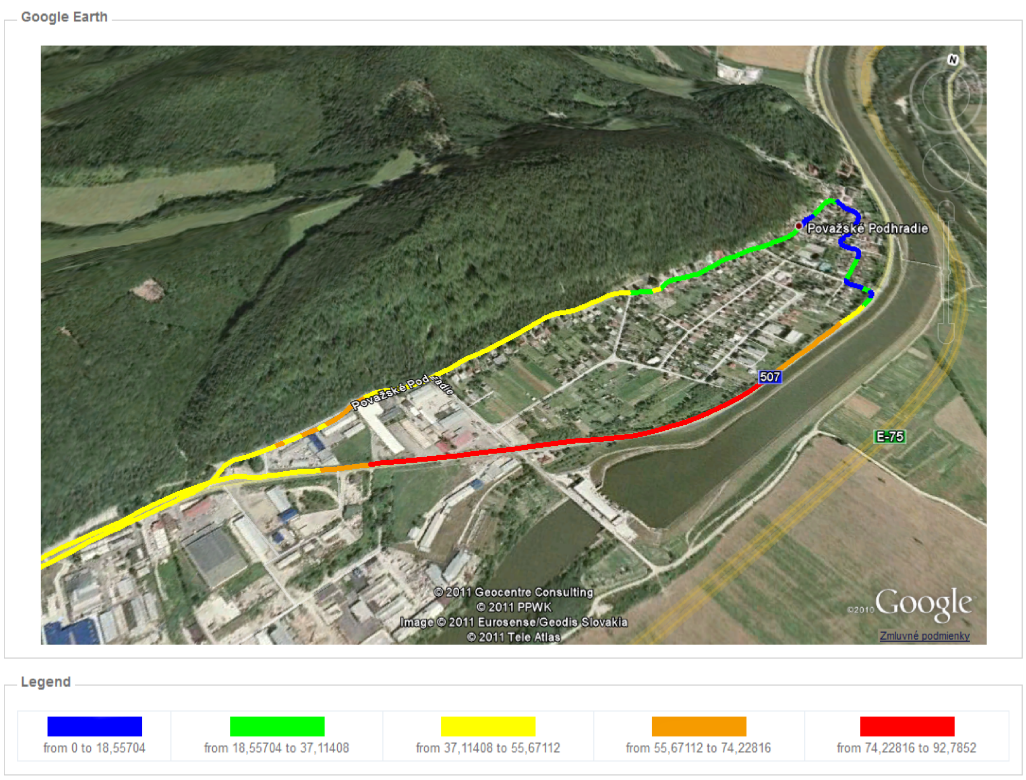
Obrázok 25: Serverová aplikácia: jazdy, výber zobrazovanej veličiny



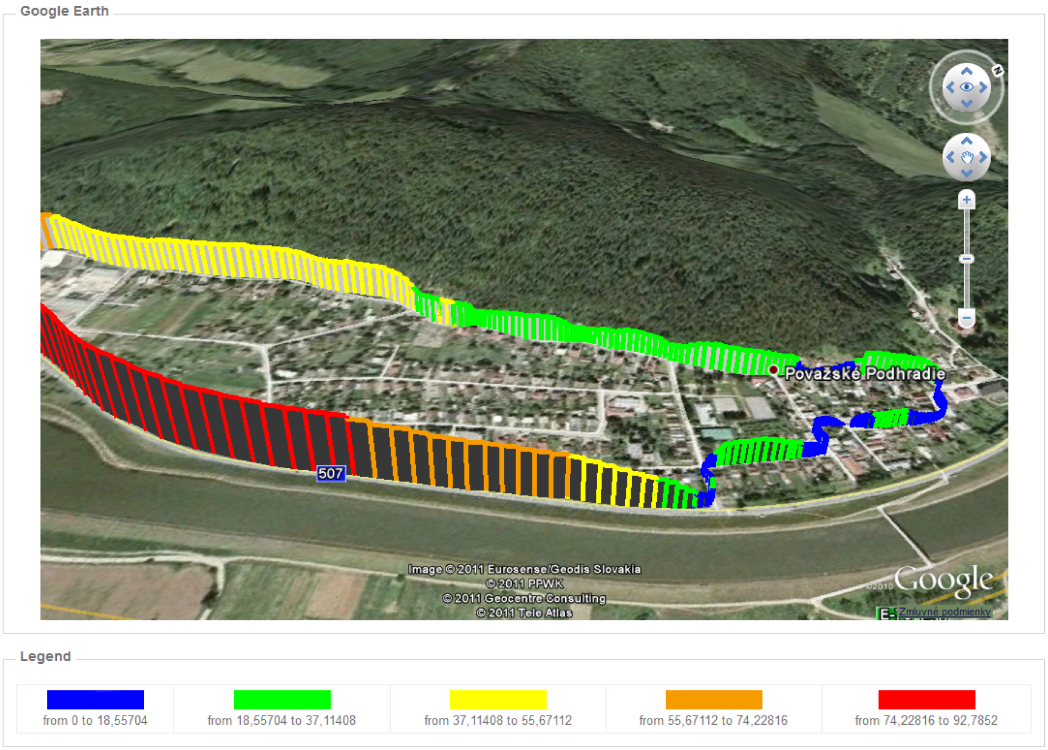
Obrázok 26: Serverová aplikácia: jazdy, legenda

Hodnota je zobrazená v základnom móde, kedy je celý rozsah rozdelený do piatich farebných segmentov alebo je možné vybrať mód, kedy je veľkosť hodnoty zobrazovaná pomocou nadmorskej výšky. Oba módy popisuje obrazovka Obrázok 27: Zobrazenie meranej veličiny v základnom móde a Obrázok 28: Zobrazenie meranej veličiny interpretované ako nadmorská výška.

Ďalšia možnosť zobrazenia je pomocou simulátora jazdy, tak ako ukazuje obrazovka Obrázok 29: Zobrazenie jazdy pomocou simulátora. Pre spustenie jazdy je nutné kliknúť na tlačidlo Initialise. Je možné nastaviť polohu kamery, zmeniť rýchlosť z reálnej na ručne definovanú a priblíženie k mape.



Obrázok 27: Zobrazenie meranej veličiny v základnom móde

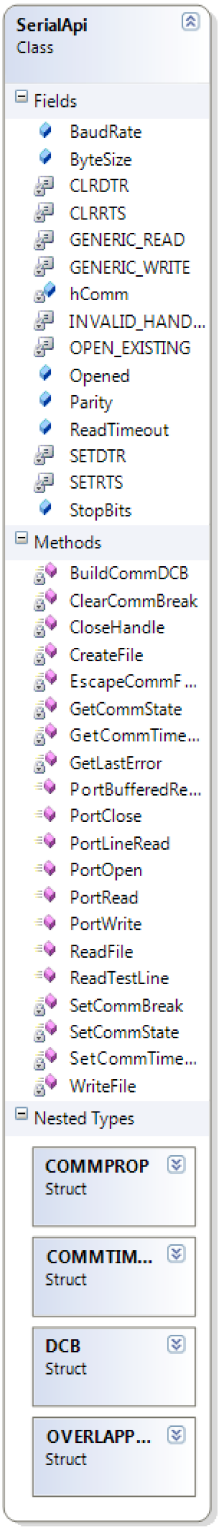


Obrázok 28: Zobrazenie meranej veličiny interpretované ako nadmorská výška

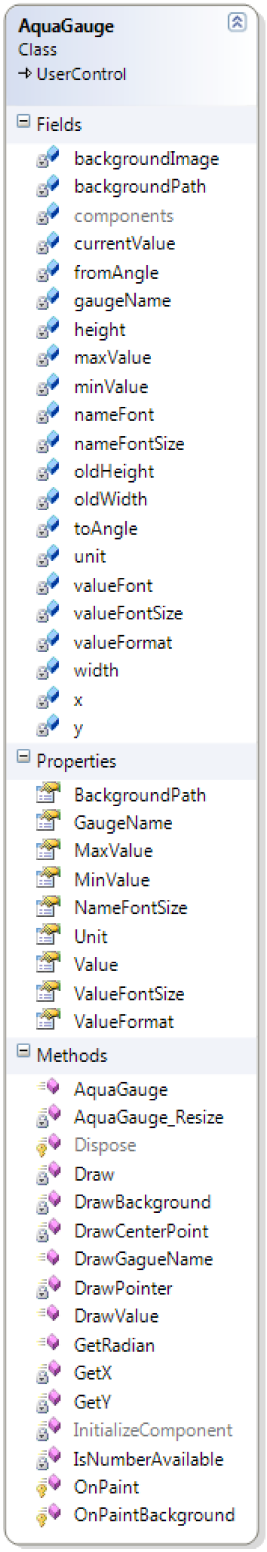
Obrázok 29: Zobrazenie jazdy pomocou simulátora

C) Diagramy vybraných tried

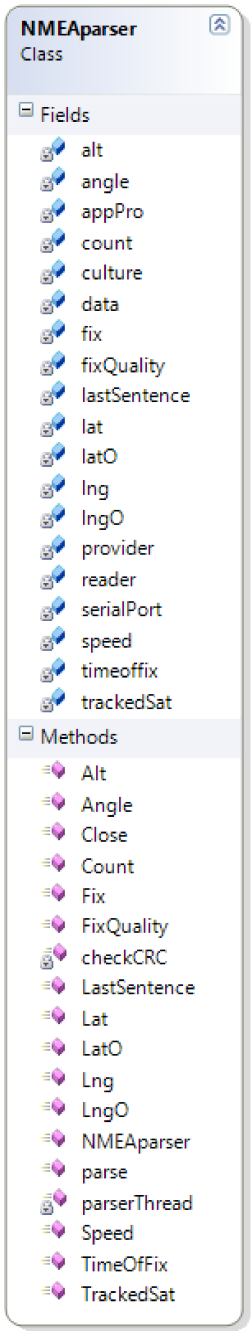
Táto sekcia obsahuje ukážky dôležitých triednych diagramov, na ktoré sa odkazujú kapitoly v praktickej časti. Sama o sebe neplní žiadny význam.



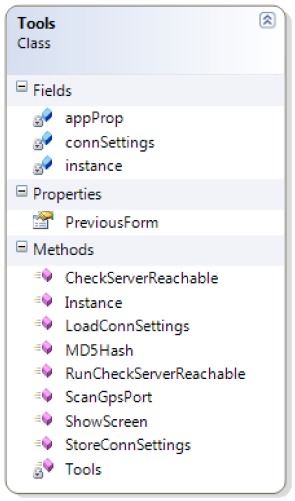
Triedny diagram 1: Klientská aplikácia: SerApi



Triedny diagram 2: Klientská aplikácia: AquaGauge

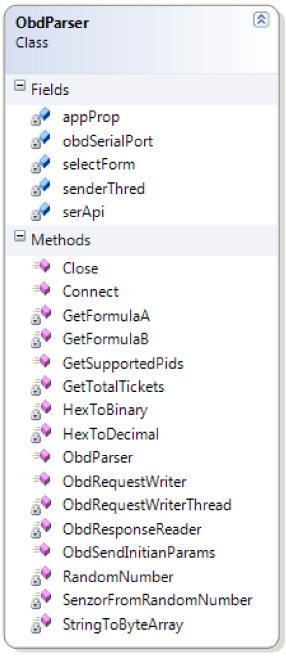


Triedny diagram 3: Klientská aplikácia: NmeaParser

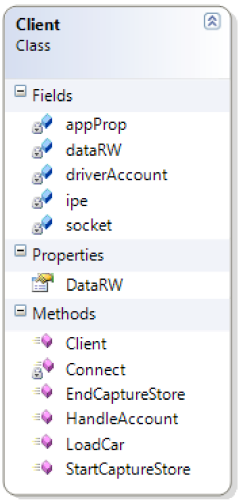


Triedny diagram 4: Klientská aplikácia: Tools

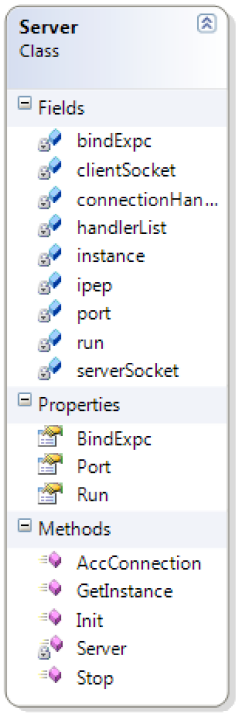
Triedny diagram 5: Klientská aplikácia: ConnSettings



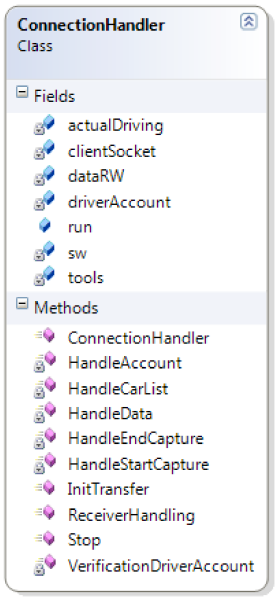
Triedny diagram 6: Klientská aplikácia: ObdParser



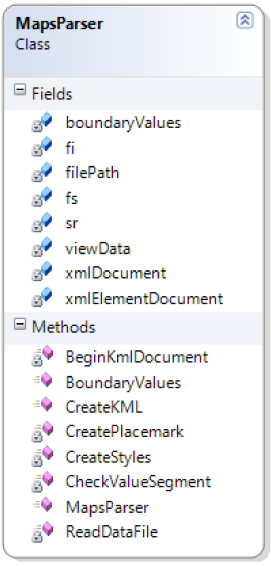
Triedny diagram 7: Klientská aplikácia: Client



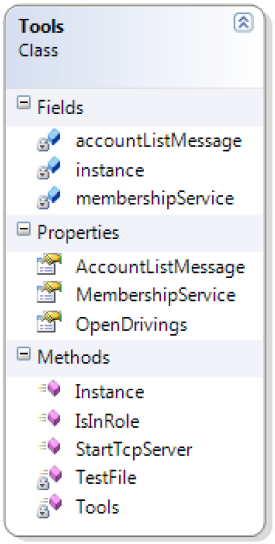
Triedny diagram 8: Serverová aplikácia: Server



Triedny diagram 9: Serverová aplikácia: ConnectionHandler



Triedny diagram 10: Serverová aplikácia: MapsParser



Triedny diagram 11: Serverová aplikácia: Tools