Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Řídicí modul pro systém na monitorování polohy mobilních objektů

Diplomová práce

Bc. Lukáš Valenta

Vedoucí práce: Ing. Jan Fesl

České Budějovice 2016

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL MAGISTERSKÉ PRÁCE

Student: **Lukáš Valenta, Bc.**
(jméno, příjmení, tituly)

Obor – zaměření studia: .Aplikovaná informatika.....

Katedra: UAI PRF JU.....

Školitel: Jan Fesl, Ing.....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma magisterské práce: .....
Řídicí modul pro systém na monitorování polohy mobilních objektů
.....

Cíle práce :

Prvním cílem této diplomové práce je výběr a zprovoznění (nainstalování OS Linux) vhodné hardwarové ebedded platformy pro realizaci systému na monitorování mobilních objektů. Základním kritériem výběru je pořizovací cena, rozšiřitelnost systému, dostatek vstupně výstupních portů a možnost portace operačního systému Linux. Dalším cílem této práce jest návrh protokolu pro vzdálenou komunikaci mezi systémem a distribuovanou řídicí aplikací. Vlastní realizace komunikačního protokolu bude realizována na bázi multiprocesovým tcp-démona, který se stane permanentní částí spuštěného operačního systému. Vlastní realizace démona bude vytvořena v programovacím jazyce C/C++.

Základní doporučená literatura :

[1] http://www.root.cz/serialy/sokety-a-cc/ , online

[2] Linux – dokumentační projekt, Kolektiv autorů, 4. vydání

Financování práce :.....

Vedoucí práce :.....podpis : .....

U externích vedoucích fakultní garant práce.....podpis : .....

Garant oboru mag. studia ..... podpis : .....

Vedoucí katedry ..... podpis .....

Případný souhlas vedoucího ústavu AV .....podpis : .....

V Českých Budějovicích dne 19. 2. 2013

Převzal/a dne 19. 2. 2013 podpis : .....

Bibliografické údaje

Valenta Lukáš, 2016: Řídicí modul pro systém na monitorování polohy mobilních objektů. [The control module for a system monitoring the location of mobile objects. Mgr.. Thesis, in Czech.] – 64 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Diplomová práce se zaměřuje na volbu hardwarové embedded platformy pro realizaci systému monitorování mobilních objektů MoTracker a následné zprovoznění systémové části. Dále na komunikační protokol zajišťující komunikaci mezi aplikací MoTracker a řídícím serverem. Součástí komunikačního protokolu je kontrolér zajištující předávání požadavků do správného modulu. V závěru této práce je popsáno ovládání aplikace MoTracker kterou vyvíjeli všichni členové podílející se na projektu.

Anotation:

This thesis focuses on the choice of embedded hardware platform for the implementation of the monitoring of mobile objects MoTracker and subsequent commissioning of the system. Further it focuses on the communication protocol that ennables the communication between MoTracker and a control server. Part of the communication protocol is a controller that ensures that the requests are passed over to the right modul. The conclusion of the thesis includes the controls of the MoTracker application that have been developed by all members of the project.

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 20. 4. 2016 ……

Valenta Lukáš

Poděkování

Chtěl bych poděkovat svému vedoucímu diplomové práce panu Ing. Janu Feslovi za možnost účastnit se tohoto projektu, cenné odborné rady, věcné připomínky, vstřícnost při konzultacích, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval.

Obsah

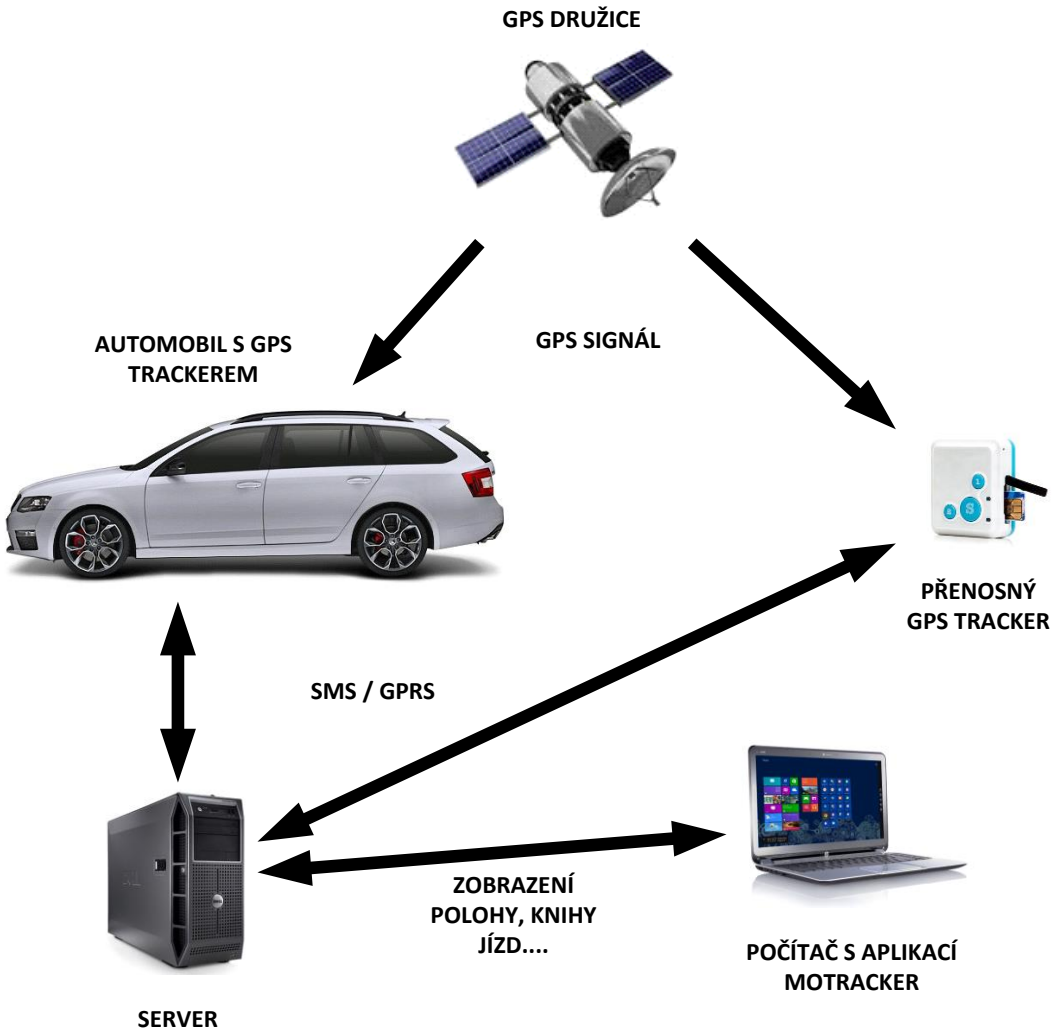
Úvod ... 3
Cíle práce ... 5
Teoretická část ... 6
Volba hardwarové platformy ... 6
Volba a instalace operačního systému ... 8
Konfigurace DHCP ... 10
Konfigurace TFTP ... 10
PXE (Preboot execution environment) ... 12
Princip komunikace ... 12
Zavaděče s podporou PXE ... 13
Nasazení serveru ... 13
Nasazení klientů ... 14
GPS-základní princip ... 14
GPS tracker ... 16
Přenos informací SMS /GPRS ... 18
BSD sockets ... 19
Hlavičkové soubory ... 19
Funkce socket API ... 20
DLL ... 22
Implementace ... 23
Controller ... 24
Komunikační protokol pro aplikaci MoTracker ... 26
Serverová část ... 27
Klientská část ... 27
Databázový modul ... 29
UPS modul ... 32
GPRS modul ... 33
MoTracker ... 35

Poloha vozidel 36
Kniha jízd 39
Rezervace vozidel 41
Nastavení systému 43
Migrace systému 49
Závěr 51
Seznam použitých zkratek 52
Přílohy 56
Obsah přiloženého CD: 56
Seznam obrázků 57
Seznam použité literatury 58

Úvod

Monitorování polohy objektů (např. automobilů), případně zvířat je poslední dobou stále častější. V případě monitorování automobilů, na které je celý projekt zaměřen, se menší firmy mohou potýkat s relativně vysokou pořizovací cenou celého systému nebo vysokým měsíčním paušálem za poskytování služeb. Projekt MoTracker (motracker.prf.jcu.cz) by měl být východiskem pro firmy, které nechtějí investovat velké částky na provoz profesionálního komerčního systému a zároveň chtějí mít absolutní přehled o pohybu svých vozidel. Veškeré zdrojové kódy budou proto volně dostupné a jediná investice je tracker, který je nutné namontovat do vozidla a jednodeskový počítač případně jiný fyzický či virtuální počítač, na kterém bude server aplikace. Systém umožnuje sledování polohy vozidel, spravování požadavků rezervovaných vozidel a pro mnohé firmy důležitou knihu jízd pro dané vozidlo. Proto jedním z aspektů byla pořizovací cena celé sestavy. Nicméně systém lze použít i na monitorování například chodců, případně zvířat, které budou monitorovány prostřednictvím přenosných kapesních trackerů, což mohou využít i jiné katedry na Přírodovědecké fakultě.

Systém je navržen tak, že tracker umístěný v sledovaném objektu zasílá prostřednictvím GPRS nebo SMS aktuální polohu určenou pomocí GPS signálu a jiné informace do serveru viz Obrázek 2 Blokové schéma. Server tyto údaje zpracuje a uloží do databáze. Uživatel si zobrazí požadovaná data pomocí aplikace MoTracker, kterou má nainstalovanou na svém počítači.



Obrázek 1 Blokové schéma

Práce je rozdělena na 3 části, v teoretické části se čtenář dozví základní informace o použitém hardwaru a jeho základní funkčnosti spolu s výhodami a nevýhodami, jak nainstalovat operační systém na deskový počítač bez možnosti připojení monitoru, základní vlastnosti GPS trackerů a princip GPS. V druhé část se řeší samotná implementace, tzn. jak celý systém funguje a jak je navržen a řešen. Poslední část popisuje možnosti a ovládání aplikace což může sloužit jako jednoduchý návod.

Cíle práce

Prvním cílem této diplomové práce je výběr a zprovoznění (nainstalování operačního systému) vhodné hardwarové embedded platformy pro realizaci systému na monitorování mobilních objektů. Základním kritériem výběru je pořizovací cena, rozšiřitelnost systému, dostatek vstupně výstupních portů a možnost portace operačního systému. Dalším cílem této práce je návrh protokolu pro vzdálenou komunikaci mezi systémem a distribuovanou řídicí aplikací. Vlastní realizace komunikačního protokolu bude realizována na bázi multiprocesorového tcp-démona, který se stane permanentní částí spuštěného operačního systému. Vlastní realizace démona bude vytvořena v programovacím jazyce C/C++. Posledním úkolem bylo vytvořit kontrolér, který všechny příchozí požadavky zpracuje a předá určenému modulu pro zpracování.

Teoretická část

V této kapitole jsou popsány základní principy fungování použitého hardwaru i softwaru. Kapitola se prvně zaměřuje na použitý hardware pro serverovou část aplikace, následně volbu a instalací operačního systému za použití protokolu PXE. Poté jsou rozepsány základní principy GPS lokace, funkce trackeru a možnosti přenášení informací z trackerů do řídícího serveru. V poslední části popisuji BSD sockety, které jsou základním komunikačním prvkem celého projektu.

Volba hardwarové platformy

Hlavní požadavky pro výběr hardwarové platformy byla rozšiřitelnost, dostatečný počet vstupně výstupních portů, výkon, spotřeba samotného zařízení pro případné napájení z externího zdroje elektrické energie (např. externí bateriový systém dobíjený pomocí solárních panelů pro případné používání mimo dosah elektrické sítě) a v neposlední řadě cena. Prvotní myšlenka byla celý systém zprovoznit na běžném počítači, který splňuje takřka všechny požadavky. Rozšiřitelnost je možná pomocí přidání dalších periferií prostřednictvím USB portů, případně rozšiřujících karet do PCI, což zajistí i dostatečný počet vstupně výstupních portů. Výkon se odvíjí od konfigurace daného počítače, s čímž souvisí pořizovací cena. Dalším negativním parametrem je velikost počítače a jeho spotřeba. Sice v dnešní době můžeme pořídit počítač typu SFF, případně USFF s velice malou spotřebou, ale i tak jsou zde jiné možnosti výběru hardwarové platformy. Další možností je zařízení Raspberry Pi. Raspberry Pi je, co se týče velikosti a spotřeby naprosto ideální zařízení. V době realizace diplomové práce mělo Raspberry nedostačující počet portů a jeho rozšiřitelnost nebyla tak veliká, avšak výkon byl dostačující. Poslední model Raspberry je pravděpodobně výkonnější než celá řada počítačů s datem výroby starším než sedm let a má dostatečný počet USB portů pro připojení potřebných periferií. Další výhodou je 40 pinů, které se dají připojit například k malému LCD panelu či jinak naprogramovat dle požadavků uživatele. V případě realizace v tomto roce by Raspberry Pi 2 byl velký kandidát při volbě

platformy a nejspíše bych jej zvolil. Cílem bylo najít zařízení, které by splňovalo veškeré požadavky, jaké byly zmíněny. Nejlépe všem požadavkům vyhovuje deska od firmy ALIX, a to konkrétně model ALIX.2D13.

Obrázek 2: Základní deska ALIX.2D13, převzato z[1]

Parametry desky ALIX.2D13 jsou následující [2]:

- CPU: 500MHz AMD Geode Lx800
- DRAM: 256Mb DDR DRAM
- CompactFlash socket. 44pin
- miniPCI slot

[1] http://www.luxus.cz/obchod_pic/alix2d13.jpg

- Napájení: DC jack nebo pomocí POE, rozsah vstupního napájecího napětí 7-20V
- 3 x RJ45 (Via VT6105M 10/100)
- Sériový port
- 2 x USB 2.0
- Rozměry 152,4 x 152,4 mm

K zařízení bylo nutné dodat úložiště v podobě 32GB CompactFlash karty, která by měla stačit na pokrytí všech potřebných dat což se následně potvrdilo po bez mála půl ročním testování. Konkrétně se jedná o Kingston Compact Flash 32GB 600x Ultimate a externí univerzální napájecí zdroj s rozsahem výstupního napětí 7 – 20 V. Pro zlepšení vizuální stránky a především kvůli zvýšení mechanické odolnosti a ochrany desky, byla deska s úložištěm uložena do casu, který je pro tento model běžně dostupný.

Celkové pořizovací náklady za hardwarovu část činí celkem 5500 Kč (počítáno s drobnou rezervou a cenami v roce 2013), položkový rozpočet:

- 3200 Kč s DPH - ALIX.2D13
- 1500 Kč s DPH - Kingston Compact Flash 32GB 600x Ultimate
- 300 Kč s DPH – Napájecí adaptér
- 300 Kč s DPH – Case

Volba a instalace operačního systému

Správná volba operačního systému je další klíčový bod pro správnou funkčnost systému. Jako první se nabízí některý produkt z řady Microsoft Windows. Zde je nutno zakoupit licenci, což navýší pořizovací náklady na celou sestavu. Dalším problémem je, že režie na běh operačního systému je příliš vysoká a aplikace na monitorování pohyblivých objektů by se mohla zasekávat, být extrémně pomalá, případně by mohla nestíhat zaznamenávat získané hodnoty. Problém by mohl být se získáním vhodných ovladačů pro zvolený operační systém z řady Microsoft Windows. Vzhledem k tomu, že celý monitorovací systém by měl být pokud možno co nejlevnější a nejflexibilnější, je Linux nejlepší volbou. O tom jaká je nejlepší

distribuce, by se mohly vést dlouhé debaty. Vybral jsem distribuci Debianu, jelikož není nutná kompilace jádra pro ALIX, protože existuje distribuce Debianu, která je přímo připravená pro základní desku ALIX.

Instalaci operačního systému jsem prováděl po síti za pomocí PXE protokolu, princip tohoto protokolu bude vysvětlen v následující kapitole, jelikož deska ALIX nemá žádný grafický výstup, na který by se dal připojit monitor a provést běžnou instalaci pomocí připojené externí USB CD/DVD mechaniky. Zde by bylo zařízení Raspberry Pi 2 velkou výhodou jelikož by bylo možné připojit monitor a provádět běžnou instalaci. Aby bylo možné provést síťovou instalaci, je nutné mít další počítač, na kterém bude zprovozněna služba DHCP a TFTP server s podporou PXE. Pro operační systém Microsoft Windows existuje TFTP32/TFTP64 a pro Linux balíček TFTPD. Použil jsem opět distribuci Debianu, jelikož jsem předpokládal, že celý proces bude jednodušší než instalace Linuxu pomocí Microsoft Windows. Na pomocném počítači je nutné nastavit DHCP server a TFTP server, který lze nainstalovat z repozitáře. Dále budeme potřebovat stáhnout jádro operačního systému, v našem případě kernel Debianu určeného pro ALIX a zavaděč jádra (initrd). Jelikož se jedná o síťovou instalaci, musíme zajistit, aby pomocný počítač s ALIXem mohl komunikovat. V našem případě měl pomocný počítač pouze jednu síťovou kartu, která byla přímo připojena do desky ALIX. Toto zapojení je možné použít za předpokladu, že si do pomocného počítače nejprve uložíme veškeré potřebné soubory. V případě, že bychom chtěli mít pomocný počítač připojený do internetové sítě, museli bychom použít připojení přes switch.

Konfigurace DHCP

Jak jsem již zmiňoval, je nutné nainstalovat a nastavit DHCP server, který zajistí, aby cílové zařízení dostalo potřebné síťové nastavení a bylo schopné stáhnout potřebné soubory nutné k vzdálené instalaci. Nastavení DHCP serveru se provádí pomocí konfiguračního souboru dhcpd.conf, který nalezneme v adresáři /etc/dhcp*/. V uvedeném souboru je nutné nastavit minimálně tyto parametry:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.20;
  option routers 192.168.1.250;
  next-server 192.168.1.250;
  filename "pxelinux.0";
}
```

Uváděné hodnoty jsou funkční pro naše zapojení a pro jiné topologie a síťové podmínky se mohou lišit.

Konfigurace TFTP

TFTP server zprostředkuje úložiště pro potřebné soubory, které budou nutné pro vzdálenou instalaci operačního systému. Nastavení TFTP serveru se provádí v konfiguračním souboru TFTP, který vytvoříme v adresáři /etc/xinetd.d/. V uvedeném souboru je nutné nastavit minimálně následující hodnoty:

```
service tftp
{
   socket_type = dgram
   protocol = udp
   wait = yes
   user = root
   server = /usr/sbin/in.tftpd
   server_args = -s /tftpboot
}
```

Následně vytvoříme adresář /tftpboot, který bude obsahovat všechny datové a konfigurační soubory. Do tohoto adresáře přesuneme jádro operačního systému

(kernel) a zavaděč jádra (initrd), který umožňuje nahrát předstartovní RAM disk. V tomto adresáři vytvoříme složku pxelinux.cfg, ve které vytvoříme soubor s názvem default. Soubor default obsahuje následující nastavení:

```
default bootup
  label bootup
   kernel linux
   append vga=normal initrd=initrd.gz console=ttyS0,38400n8
```

Nyní se stačí připojit pomocí sériového kabelu případně redukce k desce ALIX a za pomocí terminálu (např. Tera Term) povolit bootování ze sítě pokud není aktivní. Do BIOSu se dostaneme pomocí klávesy S v průběhu POSTu, následně povolíme PXE bootování stisknutím klávesy E. Díky protokolu PXE začne bootování po síti. Základní funkce protokolu je popsána v následujících kapitolách

PXE (Preboot execution environment)

Je v informatice označení technologie pro bootování počítačů z počítačové sítě. Využívá se pro tenké klienty, které nemají pevný disk, hardwarovou a softwarovou diagnostiku, automatické instalace operačních systémů a podobně (např. informační kiosky, počítače v knihovně, instalaci systému do počítače bez optické mechaniky).

PXE je standard zavedený firmou Intel v září roku 1999. Nahradil a sjednotil předchozí technologie pro start počítače za pomocí počítačové sítě (např. metodu využívající BOOTP nebo server Novell NetWare a protokol NCP přes IPX/SPX). Stejně jako předchozí technologie rozšiřuje PXE možnosti startu počítače tím, že je k síťové kartě dodána flash paměť s kódem rozšiřujícím schopnosti BIOSu o zavedení operačního systému z počítačové sítě. Dnes je PXE standardní součástí BIOSu na základních deskách, které mají integrovánu síťovou kartu, avšak PXE najdeme i jako rozšíření BIOSu na samostatných síťových kartách, které jsou zasouvány do slotů sběrnice (např. PCI nebo PCI-Express). PXE je nezávislé jak na použitém hardwaru, tak na použitém operačním systému. [1]

Princip komunikace

Síťové bootování pomocí PXE využívá Internet Protocol (IP) z rodiny protokolů TCP/IP. Základní postup komunikace je následující:

Po inicializaci požádá počítač pomocí DHCP o základní údaje, nutné pro komunikaci v počítačové síti (IP adresu, masku sítě, výchozí bránu a DNS server), využívající rodinu protokolů TCP/IP. Klient vyšle broadcastovou zprávu typu DHCPDISCOVER s nastaveným PXE příznakem v UDP datagramu. Na něj odpoví pouze takový DHCP server, který je nastaven tak, aby odpovídal stanicím žádajícím síťové bootování (v Linuxu je v konfiguračním souboru dhcpd.conf ISC DHCP serveru nastaveno allow booting a allow bootp). Ostatní DHCP servery, které podporu startu bezdiskové stanice nemají nastavenu, takový požadavek ignorují. DHCP server odešle klientovi DHCPOFFER se základními údaji: IP adresa, maska sítě, default gateway a adresa TFTP serveru.

Pokud byl v první odpovědi DHCP serveru určen TFTP server, PXE ho

kontaktuje podruhé pomocí DHCPREQUEST (multicast či unicast) se žádostí o sdělení kompletní cesty k síťovému bootstrapping programu (tzv. NBP, zaváděcí program stažený přes síť). DHCP server odpoví pomocí DHCPACK.

Nyní dojde ke stažení bootstrapping programu (NBP) do operační paměti RAM. Přenos NBP je z TFTP serveru proveden pomocí UDP protokolu. Umístění NBP v paměti je závislé na architektuře klienta. Může též proběhnout kontrola autenticity, kdy si klient vyžádá soubor s kontrolním součtem a porovná ho s kontrolním součtem staženého bootstrapping programu.

NBP je síťovou obdobou zavaděče jádra operačního systému (je běžně umístěn v boot sektoru). Stažený NBP je spuštěn. Další síťovou komunikaci obstarává sám NBP, může zobrazit menu nebo zavést do paměti RAM jádro operačního systému a aktivovat ho. [1]

Zavaděče s podporou PXE

Bootstrap program může obsahovat libovolný kód, nicméně pro spouštění konkrétních operačních systémů je třeba zvolit kompatibilní zavaděč s podporou síťového bootování přes PXE, což umožňuje např. BootX (pro Mac OS X), GRUB (který následně volá proprietární zavaděče jednotlivých systémů) či RedBoot. Pro systémy UNIX je asi nejznámější SYSLINUX, resp. jeho derivát PXELINUX. Microsoft Windows lze nainstalovat pomocí služby Remote Installation Service (RIS), kde se do cílové operační paměti načte RAM disk pro spuštění Windows PE (předinstalační prostředí). Nástupcem RIS je ve Windows systémech verze vyšší, než Vista, resp. Windows Server 2008 technologie Windows Deployment Services (WDS), která přináší zároveň nadstavbu pro využití dalších protokolů a služeb zejména pro hromadné síťové instalace. [1]

Nasazení serveru

Na serveru musí být k dispozici služby, které jsou odpovědné za poskytnutí přesměrování klienta na vhodný zaváděcí server. Jsou dva způsoby jak tyto služby přesměrovat:

1. Kombinace standardního DHCP a přesměrování služby: DHCP server, který

dodává IP adresy klientům je upraven (případně nahrazen) tak, aby funkci PXE Boot přesměroval na potřebný zaváděcí server.

2. Samostatný standardní DHCP server a přesměrování služby: PXE přesměrovávající servery (proxy DHXP) jsou přidány do stávající infrastruktury. Reagují pouze na PXE požadavky klientů a poskytují pouze přesměrování na zaváděcí server.

Každý PXE zaváděcí server musí mít minimálně jeden nebo více spustitelných souborů vhodné pro klienty.

Nasazení klientů

PXE nenastavuje detaily a funkce NBP, pouze jej obdrží od serveru. Smyslem je, že spuštěním tohoto spustitelného souboru bude mít uživatel systém připravený pro použití. Při nejmenším to znamená instalaci operačního systému, potřebných ovladačů a software určený pro hardwarovou konfiguraci klienta. Zároveň může obsahovat specifické nastavení a software uživatele.

PXE specifikuje protokoly, o které klient žádá a soubory pro spuštění zaváděcího obrazu. Pokud jsou splněny minimální požadavky pro spuštění klienta, je obraz stažen.

GPS-základní princip

Abychom byli schopni pokud možno, co nejpřesněji určit polohu sledovaného objektu využijeme globální polohovací systém (Global Positioning Systém) spíše známy jako GPS. Tato kapitola by měla nastínit základní funkčnost tohoto systému a ne podrobné vysvětlení všech jeho částí

Jako mnoho jiných projektů i GPS vznikl ve vojenském odvětví a provozuje ho Ministerstvo obrany Spojených států jak pro vojenské tak i pro veřejné účely. Celý systém je rozdělen na 3 základní části:

Kosmická část obsahuje družice, které obíhají v určité výšce nad Zemí v 6 kruhových drahách, které jsou vůči sobě posunuty. Na každé dráze se nachází

několik družic v takřka pravidelných rozestupech. Poloha, rozestup, rychlost a jiné parametry jsou nastaveny tak aby nedošlo k vzájemné kolizi družic a zároveň aby bylo možné určit polohu kdekoliv na Zemi. Pro základní určení polohy je nutné mít signál alespoň od 3 satelitů, pochopitelně čím více satelitů může lokalizovat polohu, tím bude výsledná poloha přesnější. Reálně tak získáváme signál od 5- 12 satelitů současně (nejčastější počet satelitů zasílajících signál je 8-9).

Kontrolní a řídící část zajišťuje obsluhu družic, což znamená, že na družicích provádí údržbu, provádí manévry, aby nedošlo ke kolizím nebo aby se dala lépe lokalizovat poloha. Systém se v případě vyřazení všech kontrolních středisek na zemi přepne do automatického stavu, ve kterém družice vydrží půl roku bez vnějšího zásahu. K tomuto stavu zatím nedošlo, a proto nejsou známy výsledky testů.

Poslední je uživatelská část kde uživatelé pomocí vhodného pasivního GPS přístroje (přístroj není schopen odeslat data do družice ale pouze je přijímat) přijímají signály od dostupných družic.

GPS systém je velice složitý a zahrnuje mnoho částí, které řeší například zpoždění signálu, přesností měření, šifrování dat a mnohé další.

GPS tracker

GPS tracker je zařízení, které umí zjistit aktuální polohu a dále s ní pracovat dle nastavení. Zařízení využívá GSM/GPRS (850/900/1800/1900Mhz) sítě a GPS satelitní poziční systém. Některá zařízení mají integrované GPS a GSM antény, jiné mají konektory pro připojení. Výjimkou není ani integrovaný, popřípadě externí mikrofon pro případný záznam zvuku, SOS tlačítko, které z pravidla vyžaduje další nastavení (co se má po stisknutí stát), slot pro SD kartu, nebo konektor pro připojení k řídící jednotce automobilu (v tomto případě se jedná o trackery určeny pro použití v automobilech).

Tracker lze používat v několika základních režimech:

Real-time sledování: název je trochu zavádějící, jelikož nedostáváme data v reálném čase například každou sekundu, ale spodní hranice většiny trackerů se pohybují v rozsahu 5-20 sekund. Tato hodnota je plně dostačující pro přesný záznam pohybu chodců i automobilů.

Poloha na požádání: v tomto režimu tracker zašle aktuální polohu například pokud zavoláme na telefonní číslo trackeru (hovor je trackerem odmítnut) nebo pokud pošleme SMS v požadovaném tvaru. Opět záleží na daném výrobci.

SOS funkce: některé trackery jsou vybaveny SOS tlačítkem, které si můžeme různě nastavit. Většinou se po stisknutí tlačítka odešle SMS zpráva přednastaveným uživatelům a u modelů vybavených mikrofonem se mohou po zavolání na telefonní číslo trackeru spustit nahrávací případně monitorovací funkce v okolí trackeru.

Geofencing: tento režim hlídá vymezenou oblast, a v případě, že ji tracker opustí, vykoná přednastavený příkaz. Většinou odeslání SMS na přednastavené číslo(a).

Režim snížené spotřeby elektrické energie: do tohoto režimu se zařízení přepne, pokud je v nečinnosti nastavenou dobu. Z režimu spánku se automaticky probudí, pokud je požadována aktuální poloha, některé modely se automaticky probouzí například každých 10 minut a ověřují svojí aktuální polohu.

Další režimy se liší dle jednotlivých výrobců a předpokládaného použití trackeru. U trackerů určených do automobilů není problém nastartovat vzdáleně auto, v případě

krádeže vypnout motor nebo připojit vibrační senzor a tracker použít jako alarm.

Samotné nastavení trackeru se většinou provádí pomocí SMS zprávy v daném konfiguračním tvaru. Například pro autorizaci, která telefonní čísla, mohou měnit nastavení zašleme SMS zprávu ve tvaru „*admin+heslo+mezera+číslo_mobilního_telefonu*“ (*admin123456 123456789*). Pokud nastavení proběhlo v pořádku, měla by se vrátit SMS ve tvaru „admin ok“. Konfigurační SMS se lišší dle daného výrobce, princip a syntaxe jsou většinou velice podobné.

Informace získané od trackeru pomocí SMS zprávy jsou například v následujícím tvaru:

Lat: 22.566901 long: 114.051258 speed: 0.00 14/08/14 06.54 bat:F Signal:F help me imei:354776031555474

Lat:(zem.šířka) 22.566901 long:(zem.délka) 114.051258= souřadnice
Speed 0.00 = rychlost
14/08/09 06.54= datum a čas
bat:F = baterie je nabitá, když bude symbol ‘L’ znamená to, že baterie, je slabá
Signal:F = plný GPS signal ‘L’ žádný GPS signal
help me = zpráva SOS
imei:354776031555474= IMEI

Pokud používáme GPRS přenosové protokoly, zprávy zasílané z trackeru na nastavenou IP adresu budou například v následujícím formátu:

Pořadové číslo+autorizované číslo+GPRMC+GPS signálu Indikátor+command+IMEI+kontrolní součet CRC16

Konkrétní příklad:

*090907070718,13145826175, GPRMC, 070718,000,, 2234,0228, N, 11403.0764, E, 0,00 ,, 070909,,, A * 73, F, helpme, IMEI: 354776030042714,132,40512*

090907070718 = pořadové číslo (včetně data a času)
13145826175 = autorizované číslo telefonu

GPRMC, 070718.000, A, 2234.0228, N, 11403.0764, E, 0,00,, 070909,,, A * 73, =
GPS modulu Original GPRMC věty s uvedením pozice
F = Plný GPS signál L = žádný GPS signál
Help mi = pomozte mi = zpráva SOS
IMEI: 354776030042714 = IMEI číslo trackeru
132 = délka řetězce GPRS
40512 = CRC16 kontrolní součtu

Přenos informací SMS /GPRS

Získaná data potřebujeme nějakým způsobem doručit do řídicího systému pro další zpracování a zobrazení. Předpokladem je že na straně vysílače (GPS trackeru) a přijímače (řídící systém – Alix) je funkční SIM karta nebo jiné připojení do internetové sítě.

Data můžeme posílat například pomocí běžné SMS zprávy ve tvaru který byl zobrazen v předcházející kapitole, nebo pomocí GPRS. Zasílání informací o poloze objektu prostřednictvím SMS zprávy se jeví jako výhodnější za předpokladu, že nepotřebujeme data zasílat příliš často a že nám nevadí případné zpoždění doručené zprávy, což by mohlo být například u sledování spíše statických nebo pomalu se pohybujících objektů. Z dané zprávy je třeba pomocí vhodného parseru vybrat potřebná data a uložit je do databáze a pochopitelně nastavit požadovaný interval zasílání informací o poloze.

Pokud potřebujeme zasílat informace o poloze častěji, je vhodnější využít přenos dat pomocí GPRS. Dnes se nemusíme obávat o to, že by GSM síť nebyl rozšířen o paketovou část která umožnuje GPRS přenos, nebo byl až na výjimky nedostatečný signál. GPRS je dále výhodnější, protože se jedná o nepřetržité spojení a lze data posílat kontinuálně, rychleji a pokud by to bylo nutné i ve větších objemech. Na první pohled se přenos dat může jevit dražší, ale objem poslaných zpráv není nikterak velký a při správné volbě datového tarifu či využití podnikového tarifu je částka za měsíční provoz v řádu desítek korun maximálně několika stokorun (při používání SIM karet v majetku PřF JU, roaming celkovou částku nepatrně navýší).

BSD sockets

BSD sockets se poprvé objevily v operačním systému Unix BSD verze 4.2, který byl uvolněn v roce 1983. Aktuálně je implementace BSD sockets dostupná v každém moderním operačním systému, jedná se tak o standard v rámci připojování k Internetu a jsou široce používány v síťových aplikacích. Určitou alternativou jsou Transport Layer Interfaces (TLI), které ovšem nejsou tak rozšířené jako BSD sockets. API BSD sosckets je psáno v jazyce C, ale dostupné je i v dalších jazycích, nicméně i tyto implementace vycházejí většinou z implementace pro jazyk C. [3]

Hlavičkové soubory

Rozhraní Berkeley sockets je definováno v několika hlavičkových souborech. Jména a obsah souborů se může mírně lišit dle implementace, obecně se jedná o následující soubory:

<sys/socket.h>

Základní funkce a datové struktury BSD sockets

<netinet/in.h>

Adresy AF INET a AF INET6 a k nim protokoly PF_INET and PF_INET6 obsahují IP adresy a čísla portů TCP a UDP

<sys/un.h>

Skupina adres PF_UNIX/PF_LOCAL se používá pro lokální komunikaci v rámci jednoho stroje.

<arpa/inet.h>

Funkce pro manipulaci s číselnými IP adresami

<netdb.h>

Funkce pro překlad jmen protokolů a jmen hostů do jejich číselné podoby. Využívá k tomu lokální data i DNS. [6]

Funkce socket API

Následující seznam vypisuje nejdůležitější funkce, které API BSD sockets nabízí.

socket

Funkce socket() je volána jako první a je určena k vytvoření nového socketu daného typu. Jako parametr se předává požadovaný protokol obsahující doménu, typ socketu specifikující požadovanou službu a protokol v rámci rodiny protokolů. Za poslední parametr se většinou dosazuje nula, což zapříčiní automatický výběr vhodného protokolu pro požadovaný typ služby.

Většina implementací umožňuje specifikovat alespoň rodiny protokolů PF_UNIX pro lokální meziprocesovou komunikaci, prostřednictvím souborového systému a PF_INET pro komunikační doménu Internetu. Z typů služeb je vždy k dispozici datagramová služba (SOCK_DGRAM) a virtuální spojení (SOCK_STREAM). [2, 3]

int socket (int pf, int type, int protocol)

bind

K přidělení lokální adresy socketu, který dosud tuto adresu přidělenou nemá, se používá funkce bind(). Socket pro přiřazení adresy je identifikován deskriptorem sock a adresa je uložena ve struktuře odkazované pointrem name. Parametr namelen definuje délku struktury name. Velice často se používá výraz sizeof(sockaddr_in).

V některých případech pak nelze přiřadit ani adresu typu loopback(127.0.0.1). V tomto případě je nutné stanici IP adresu zajistit a pak ji přiřadit příslušnému socketu, nebo použít konstanty INETADDR_ANY. [2]

int bind (int sock, struct sockaddr name, int namelen)*

connect

Funkce connect() se liší podle typu socketu, na který je aplikována. V případě socketu typu SOCK_STREAM, zřizuje virtuální spojení se vzdálenou stanicí identifikovanou adresou name. Pokud je aplikována na socket typu SOCK_DGRAM, specifikuje se tím adresa, na kterou budou všechny další datagramy z tohoto socketu odesílány a ze které budou přijímány. Proces navazování spojení pokračuje paralelně

a je na aplikaci aby detekovala okamžik, kdy spojení bylo navázáno, nebo kdy transportní vrstva rozhodla o nedostupnosti partnera. Navázání spojení, resp. informaci o nedostupnosti volaného, lze detekovat jako asynchronní událost s využitím signálů. Toto nastane tehdy, když pokus o navázání spojení nebyl úspěšný. [2, 4]

int connect (int sock, struct sockaddr name, int namelen)*

send, recv

Funkce send() a recv() je obdobou read() a write() s tím, že dovolují specifikovat přídavný parametr, který má význam jen v souvislosti s rozhraním Sockets. Tento parametr zavádí možnost předání příznaků, které v současné verzi (1) identifikují urgentní data a (2) umožňují obejít standardní mechanismus routingu. Použitelný je ještě (3) příznak MSG_PEEK, který dovoluje předečtení určitého množství dat, aniž by při tom byla data odstraněna ze vstupní fronty. [2,3]

int send(int sock, char buf, int buflen, int flags);*
int recv(int sock, char buf, int buflen, int flags);*

sendto, recvfrom

Využití těchto funkcí je obdobné, jako funkce send() a recv() s tím rozdílem, že adresa vzdálené stanice je při volání vždy zadávána explicitně.

Při vysílání datagramů ze socketu typu SOCK_DGRAM, nesmí být překročena maximální délka IP paketu podporovaná síťovou vrstvou pro použitý interface. Jestliže k překročení dojde, žádná data vyslána nebudou a funkce se vrací s chybovým stavem EMSGSIZE.

int sendto (int sock, char buf, int len, int flags, struct sockaddr* to, int tolen);*

int recvfrom (int sock, char buf, int len, int flags, struct sockaddr* from, int* fromlen);*

listen

Funkce listen() je určena nastavení socketu do stavu listening, tedy k jeho přípravě k přebírání žádosti o zřízení spojení. Parametr backlog specifikuje délku fronty, do níž se budou ukládat žádosti, které není možné vyřídit okamžitě (maximální hodnota parametru backlog je zpravidla 5). Žádosti, které není možné umístit do fronty, jsou automaticky odmítány.

int listen(int sock, int backlog);

accept

Funkce accept() slouží k selekci požadavků o spojení z fronty socketu, který byl předtím uveden do stavu listening. Pokud je takový požadavek k dispozici, vytvoří funkce nový socket se stejnými parametry, jako má socket, z jehož fronty byl požadavek na spojení selektován. Pokud je accept() volán v situaci, že žádný požadavek o spojení není k dispozici, bude podle režimu socketu proces zablokován, nebo se funkce vrátí s indikací chyby EWOULDBLOCK. [2, 3, 4]

int accept (int sock, struct sockaddr addr, int* addrlen);*

close

Funkcí close() uzavíráme socket v případě, že již nebude dál využíván. Úspěšné dokončení této operace vede k uvolnění přiřazených systémových prostředků. V případě socketu typu SOCK_STREAM, mu samozřejmě předchází soubor akcí vedoucích ke korektnímu zrušení virtuálního spojení.

int close (int sock);

DLL

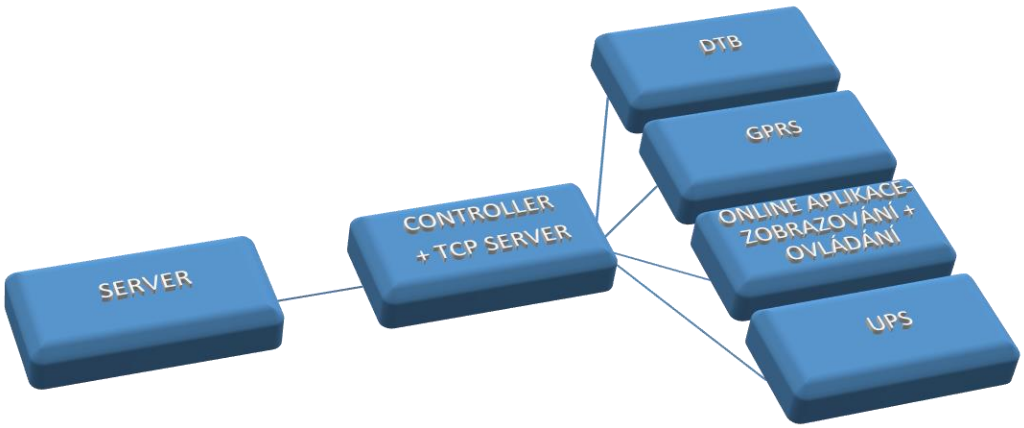
Dynamicky linkované knihovny obsahují data a kód, který lze použít ve více programech současně a nedochází tak ke zdvojení kódu, nebo k jeho velkému snížení. Výrazně se tím sníží vliv na výkon nejen samotného programu ale i ostatních programů, které jsou spuštěny. Každý program může použít funkce obsažené v této knihovně což umožnuje opětovné použití kódu a lepší využití paměti. DLL knihovny fungují jako moduly, tzn., že pokud nějaká část programu je řešena pomocí knihovny a je třeba ji změnit, stačí změnu aplikovat na danou knihovnu, případně lze knihovny přidávat nebo odebírat za běhu a není třeba celou aplikaci instalovat znova. Využívá-li více programů stejnou knihovnu, projeví se úprava knihovny ve všech programech, které jí používají, což zjednoduší správu aplikací. Případné knihovny jsou odděleny, tudíž pokud při úpravě v knihovně dojde k chybě, neovlivní to ostatní knihovny. DLL knihovny jsou spuštěny probíhajícím programem a sdílí s ním přístupová práva a paměťový prostor.

Implementace

Celý systém můžeme pomyslně rozdělit na 3 část. První část se stará o lokalizaci sledovaného objektu a přenesení dat k řídící aplikaci, druhá část je server (řídící aplikace), která zajišťuje sběr dat, dekódování příchozích zpráv a ukládání dat. Poslední částí je aplikace pro zobrazení již získaných a zpracovaných hodnot o poloze.

Princip lokalizace sledovaného objektu byl již zmíněn v teoretické části, jakmile tracker má polohu odešle informace o poloze a další potřebná data (přesný datum a čas, EMAI, telefonní číslo a jiné upřesňující informace v závislosti na výrobci daného trackeru) buď prostřednictvím SMS zprávy nebo pomocí GPRS.

Jakmile TCP server, popřípadě SIM karta v serveru, obdrží data, začne controller na základě EMAI obsaženého na začátku příchozí zprávy data dekódovat. Jelikož každý výrobce posílá data v rozdílném tvaru, není možné aplikovat stejný parser. Po určení výrobce předá controller zprávu příslušnému modulu k dalšímu zpracování. Serverová část aplikace obsahuje 5 základních částí: databázi, GPRS modul, UPS modul, napojení na uživatelskou aplikaci a controller s TCP serverem viz Obrázek 3 Blokové schéma serveru. Jednotlivé části budou popsány v následujících kapitolách.



Obrázek 3 Blokové schéma serveru

Poslední částí je aplikace MoTracker, která zobrazuje potřebná data uživatelům. Skrz tuto aplikaci si zobrazí potřebné informace. Aplikace MoTracker je určena pro operační systémy z řady Microsoft jelikož většina uživatelů používá právě některý z operačních systémů této společnosti. Jelikož zobrazení dat v samotné aplikaci není pro další zpracování zcela vhodné, je požadováno, aby aplikace spolupracovala i s jiným, vhodnějším softwarem, konkrétně s MS Office. Data je tak možné exportovat do MS Excelu a následně zpracovávat dle vlastního uvážení a potřeb. Jelikož je v databázi spousta záznamů a je třeba rychlost nativní aplikace nebylo výhodné řešit zobrazovací aplikaci přes webové rozhraní. Pokud by uživatelé uvítali pouze základní zobrazení jako například aktuální polohu vozidla, bylo by možné to vyřešit právě pomocí webového rozhraní případně přes aplikaci pro mobilní zařízení.

Controller

K správné funkčnosti aplikace bylo nutné vytvořit controller, který by zpracovával požadavky od síťové části aplikace (GPRS modul a komunikační protokol aplikace MoTracker), která zajišťuje komunikaci uživatelské aplikace a trackerů se serverem, a předal je ke zpracování do ostatních částí, např. databáze či GPRS/GSM modulu. V praxi to funguje tak, že síťová část (TCP server) přijímá požadavky od všech trackerů, které směřují k serveru, následně dle typu požadavku předá controller požadavek příslušnému modulu, který jej zpracuje a vykoná například zápis do databáze. Obdobné je to i při komunikace uživatelské aplikace se serverem. Zdrojové kódy controlleru včetně podrobných komentářů jsou na přiloženém CD ve složce controller.

Samotný controller je dělaný jako dinamicky linkovaná knihovna, jejíž výhody byly popsány v teoretické části, z které se exportují dvě funkce, které se následně používají. A to funkce createController(), která vytvoří instanci controlleru a vrátí odkaz na controller. A funkce executeCommand(), která na konkrétní instanci controlleru spustí konkrétní příkaz

Controller má definované následující třídy:

*database *dbs;*

Třída sloužící pro přístup k databázi obsahující například seznam uživatelů, seznam trackerů a aut, ve kterých jsou umístěny, polohu vozidel a jiné

vector<user> users;

Třída user typu vektor je třída šablony sekvence kontejnerů, která upořádá prvky daného typu v lineárním us pořádání a umožnují rychlý a náhodný přístup k libovolnému prvku. V našem případě se jedná o seznam všech uživatelů (uživatelské jméno, heslo, id uživatele…)

vector<tracker> trackers;

Třída tracker je obdobou třídy user, jen s tím rozdílem že se nejedná o seznam uživatelů, ale o seznam trackerů, který obsahuje následující záznamy: id trackeru, jméno, telefonní číslo, model a imei.

vector<car> cars;

Třída car obsahuje seznam automobilů případně jiných objektů, které mají instalovaný tracker. Seznam obsahuje následující záznamy: id automobilu (objektu), jméno automobilu (objektu), id instalovaného trackeru a registrační značku vozidla.

Realizace controlleru tak, aby byl schopný obsluhovat více požadavků najednou, má dvě možnosti řešení. Jedním z řešení je klonování procesů pomocí fork. Při použití fork se celý proces duplikuje s jiným procesem ID, s tím, že kopíruje celý obsah paměti a paměťové prostory jsou pak oddělené. Druhou možností je využití threadu. Výhoda je, že zůstane stejný proces ID, jen se mění thread ID a využívá stejnou alokovanou paměť. Tudíž tento způsob nezabírá tolik místa jako klonování procesů pomocí fork

Aby bylo možné z aplikace zobrazit požadované hodnoty, musí se spojit s řídící aplikací na serveru pomocí vhodného komunikačního protokolu, pomocí kterého se přenášejí požadované hodnoty například pro vykreslení trasy. Popis komunikačního protokolu je uveden v následující kapitole.

Komunikační protokol pro aplikaci MoTracker

Dalším cílem této práce je návrh protokolu pro vzdálenou komunikaci mezi systémem a distribuovanou řídicí aplikací. Vlastní realizace komunikačního protokolu je realizována na bázi multiprocesového tcp-démona, který se stane permanentní částí spuštěného operačního systému. Realizace démona bude vytvořena v programovacím jazyce C/C++ za využití Berkeley sockets, spíše známé jako BSD sockets. Zdrojové kódy komunikačního protokolu včetně komentářů jsou na přiloženém CD ve složce remoteAdmin.

Komunikační protokol se skládá z dvou základních částí a to klientské části, která je určena pro aplikaci MoTracker v prostředí Windows a serverová část, která obsluhuje požadavky od všech klientů na linuxovém serveru. Celá komunikace je z důvodu bezpečnosti šifrována pomocí SSL.

Zdrojový kod remoteAdmin_server obsahuje následující třídy:

bool sslOn; ověřuje, zda je SSL šifrování zapnuté

*SSL_CTX *ctx;* načtení všech potřených SSL knihoven

*remoteAdminLogger *ral;* slouží pro vytváření logů

*remoteAdminSettings *ras;* slouží k základnímu nastavení systému

string ip; nastavuje, na které IP adrese bude server naslouchat

int port; nastavje na kterém portu bude server naslouchat

vector <thread> threads*; je pole threadu, které budou obsluhovat uživatele

map<int, thread> connections;* slouží k namapování connectionHandle na thread

Serverová část

Program serveru běží v následujících krocích:

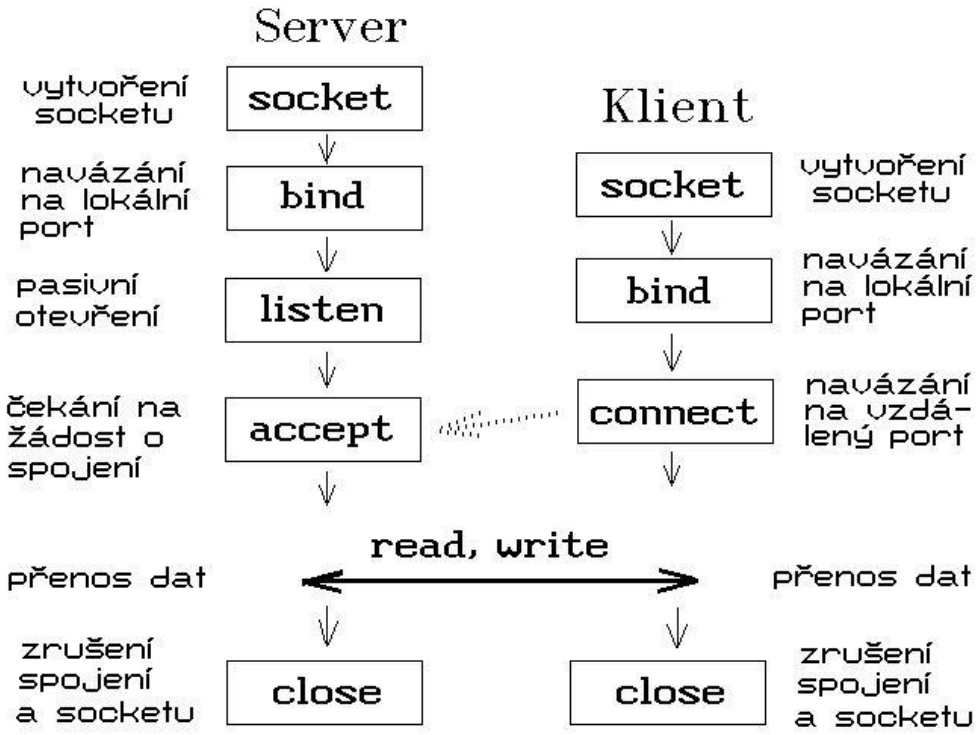
Zavolá se funkce socket(), ta vytvoří socket typu SOCK_STREAM, na kterém budou přijímány žádosti o spojení ze strany klientů. Funkcí bind(), se tomuto socketu přiřadí lokální adresa. Následně se socket přivede do stavu pasivního otevření listen(), a od této chvíle se požadavky o spojení ukládají do fronty daného socketu. Následuje volání funkce accept(), která čeká na žádost o spojení. V tomto místě se může proces rozdělit pomocí funkce fork() tak, že starý proces pokračuje v čekání na další žádosti o spojení a nový zajistí obsluhu klienta. Na socketu z funkce accept() může nyní probíhat datová komunikace nejčastěji za využití služeb send() a recv(), popřípadě read(), write(). Uzavření spojení může být iniciováno jak serverem, tak klientem. Druhý případ je rozšířenější. Stanice se o rozpojení spojení druhou stranou dozví při volání funkce recv() nebo některé se shodnou funkčností tak, že je vrácen nulový počet přečtených bajtů. V takovémto případě se volá funkce close() pro zrušení socketu a uvolnění systémových zdrojů. Server může rovněž volat funkci close() jako první, pokud se rozhodne spojení aktivně ukončit.

Klientská část

Implementace klienta pro virtuální spojení je podstatně snazší. Je třeba provést následující operace:

Funkcí socket(), se vytvořit nový socket typu SOCK_STREAM, určený pro napojení na server. Funkcí bind() se tomuto socketu přiřadí lokální adresa. Voláním connect() se naváže virtuální spojení se serverem. Po úspěšném zřízení spojení může po neomezenou dobu probíhat datová komunikace prostřednictvím funkcí send() a recv(), popřípadě read() a write(). Pokud chceme zrušit spojení, zavoláme funkci close().

Následující obrázek zobrazuje příklad komunikace mezi klientem a serverem.



Obrázek 4 Příklad ustavení spojení s využitím BSD sockets, převzato z[2]

[2] http://www.earchiv.cz/a93/a315c110.php3

Databázový modul

Databázový modul slouží ke zpracování příchozích dat a následnému uložení do databáze, která obsahuje tabulky pro záznam souřadnic, skupin, trackerů, aut, lidí (přenosné trackery) rezervací, jízd, objektů.

Databáze má následující tabulky:

Coordinations slouží k ukládání souřadnic polohy objektu a data a času kdy byly souřadnice poslány

Objects je asociační tabulka která spojuje tabulky souřadnic, skupin, jízd a trackeru

Groups je tabulka monitorované skupiny dle kritérií (auta, lidé…)

Trackers slouží pro seznam trackerů spolu se jménem, telefonním číslem atd.

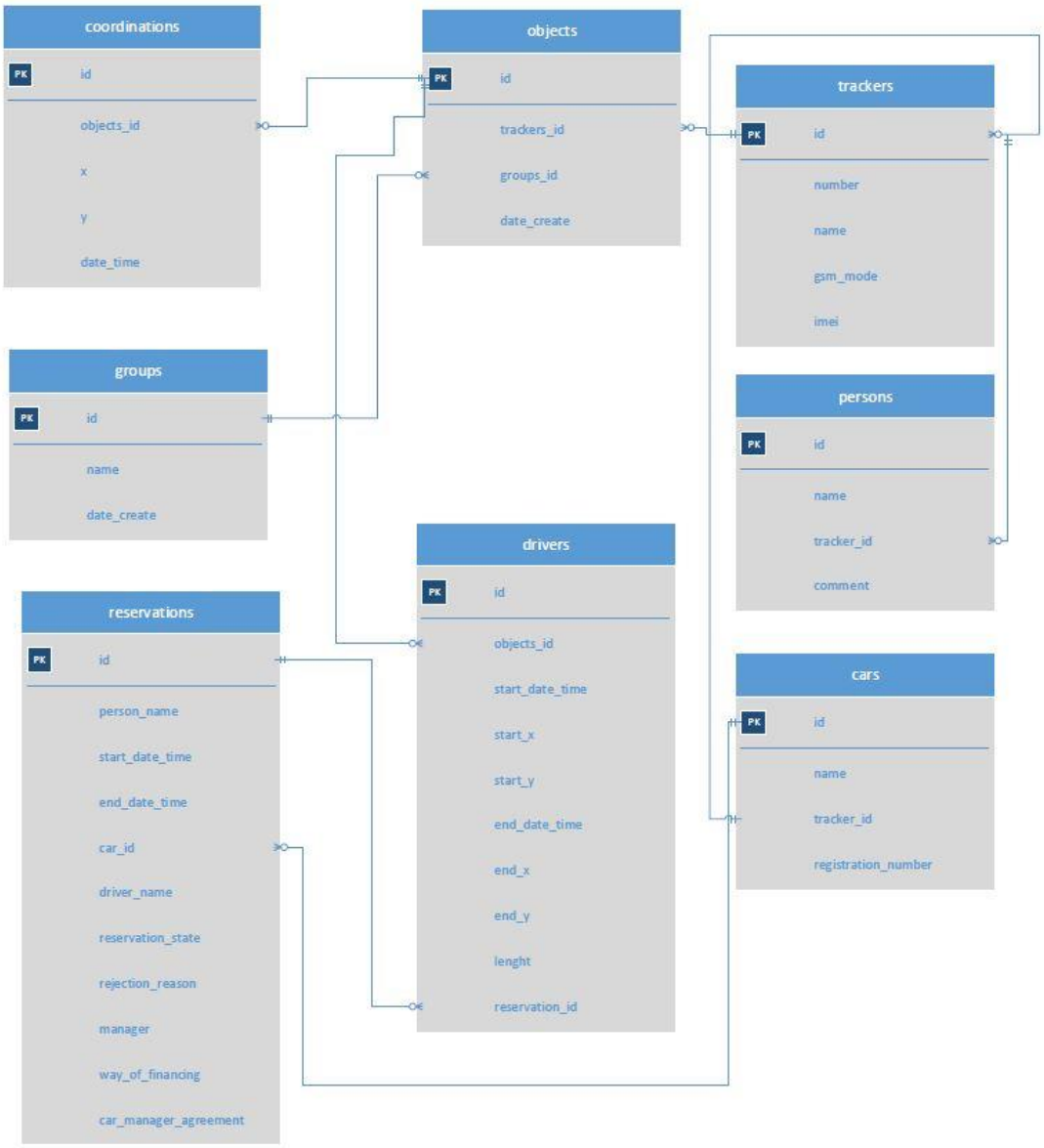
Tabulka cars obsahuje seznam automobilů kde je instalovaný tracker

Persnos specifikuje jaká osoba je sledována

Drivers specifikuje konkrétní jízdu (pohyb) sledovaného objektu, hlavně souřadnice a datum počátku a konce.

Reservations obsahuje informace z rezervačního systému

Následující obrázek zobrazuje schéma databáze



Obrázek 5 Schéma databáze

Databáze má následující typy funkcí:

Trackery:

*int getTrackers(vector<tracker> *trackers);* pro získání informací o trackeru

*int addTracker(tracker *newTracker);* pro přidání nového trackeru

int delTracker(string trackerName); pro smazání trackeru

*int editTracker(tracker *editedTracker);* pro editaci záznamu trackeru

Objekty:

*int getCars(vector<car> *cars);* pro získání informací o objektu

*int addObject(object *newObject);* pro přidání objektu

int delObject(string objectName); pro smazání objektu

*int editObject(object *editedObject);* pro editaci objektu

Auta:

*int getLatestCarPosition(string carId, timeCoordinate *tc);* pro získání informací o poslední poloze auta

*int getCarTrace(string carId, string startDateTime, string endDateTime, vector<timeCoordinate> *timeCoordinates);* pro monitorování polohy auta

*int insertCarPosition(string carId, coordinate *c, string *dateTime);* pro uložení polohy vozidla

Jízdy:

int detectAndStoreNewDrives(); nalezení nových jízd všech vozidel a uložení do databáze

*int getCarDrives(string carId, string startDateTime, string endDateTime, vector<drive> *drives);* vrátí informace o jízdě v určitém období

Rezervace

*int getNewReservations1(vector<reservation> *reservations);* zjištění nových žádostí o rezervaci vozidel pro správce 1. úrovně

*int getNewReservations2(vector<reservation> *reservations);* zjištění nových žádostí o rezervaci vozidel pro správce 2. úrovně

Zjištění rezervace v archivu

*int getArchivedReservations(string startDateTime, string endDateTime, vector<reservation> *reservations);*

Nastavení stavu souhlasu správce vozového parku - schválení/ zamítnutí 1.úroveň schválení

int setReservationStatus1(string reservationId, string status, string rejectionReason, string manager);

Nastavení stavu rezervace – schválení/zamítnutí 2.úroven schválení

int setReservationStatus2(string reservationId, string status, string rejectionReason, string manager);

UPS modul

UPS modul není pro chod aplikace zcela klíčový, jedná spíše o systémový nástroj, který zabrání nesprávnému vypnutí serveru a tím případné ztrátě dat a jeho použití se předpokládá v případě napájení pomocí UPS dobíjené například přes solární panel. Jedná se o démona *apcupsd* který lze nainstalovat z běžného repozitáře systému. Ke správné funkčnosti je třeba mít propojený PC s UPS, která umožnuje napojení pomocí nějakého kabelu (USB, RS232, RJ45). Modul si může správce aplikace kdykoliv nainstalovat a nastavit dle potřeb. V případě virtuálních strojů by démon neměl smysl za předpokladu že hypervizor vypne server sám.

GPRS modul

Gprs modul se stará o přijímání dat od všech trackerů. Jakmile přijde nové spojní tak systém vytvoří nový thread s novou BSD socketou pro daného klienta. Jelikož ne všichni výrobci zasílají zprávy ve stejném tvaru je nutné pro konkrétního výrobce trackeru zvolit příslušný způsob dekódování zprávy. V databázi máme uložený seznam EMAI, který je spárovaný se způsobem jak komunikaci od daného výrobce dekódovat. Toto je možné, jelikož první informace příchozí zprávy jsou právě EMAI.

GPRS modul obsahuje třídu gprsModulServer, která obsluhuje všechny vzdáleně připojené trackery. Tato třída obsahuje další třídy pro logování, spuštění a zastavení obsluhy spojení a třídu pro nastavení IP adresy a portu, na kterém bude server naslouchat viz. následující výpis hlavičkového souboru. Zdrojové kódy gprsModulu jsou na přiloženém CD ve složce gprsModul.

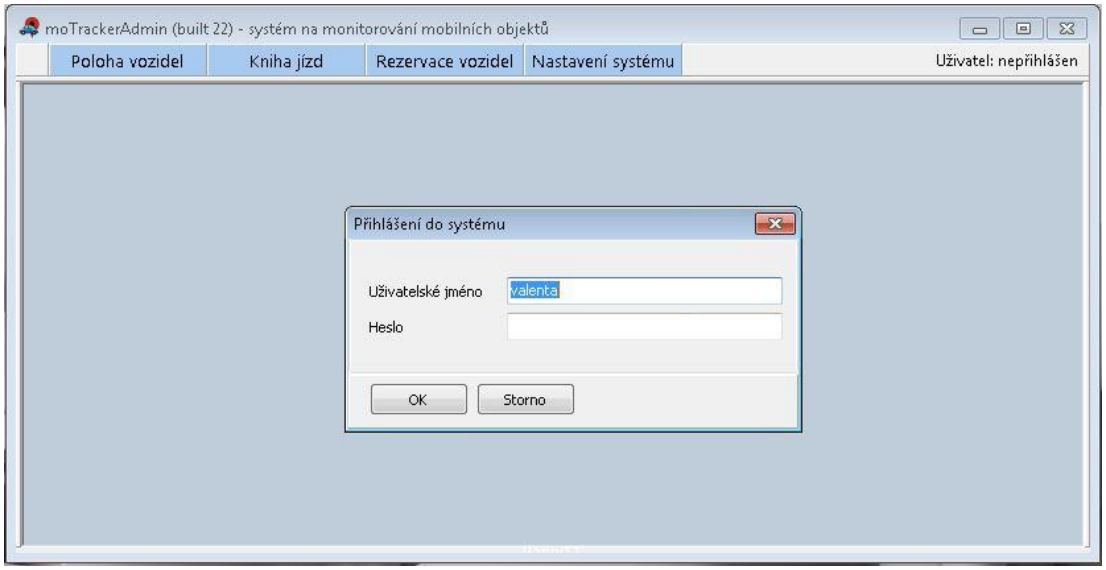
```
class gprsModulServer
{
public:
   gprsModulLogger *gml;
   virtual int start() = 0;
   virtual void stop() = 0;

protected:
   string ip;
   int port;
};
class tcpGprsModulServer : public gprsModulServer
{
   vector <thread*> threads;
   map<int, thread*> connections;
```

```
    public:
    tcpGprsModulServer(string ip, int port, gprsModulLogger *gml);
    void removeThread(int connectionHandle);
    friend void serveTracker(int connectionHandle, gprsModulServer *gmsrv);
    int start();
    void stop();
};
```

MoTracker

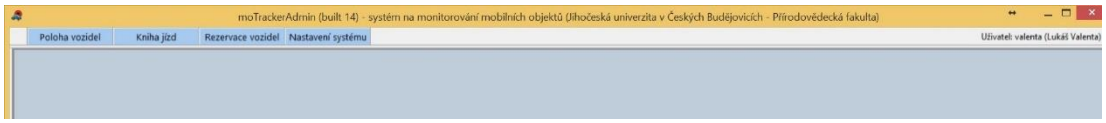
MoTracker je uživatelská aplikace (rozhraní), která umožnuje sledování polohy vozidel ve kterých je GPS tracker instalován, přenosných GPS trackerů, či mobilních telefonů vybaveným GPS. Komunikace aplikace se serverem je zajištěna pomocí komunikačního protokolu (remoteadmin) který na straně serveru naslouchá na určeném portu a v případě požadavku se naváže spojení a zaslané požadavky z aplikace se předají k dalšímu zpracování controlleru, který následně kontaktuje potřebný modul a vrátí zpět požadovanou hodnotu.

Po spuštění instalačního souboru se otevře průvodce instalací, který v jednotlivých krocích nabízí umístění instalace, název položky aplikace v nabídce Start, umístění zástupce na ploše a následnou instalaci. Přihlašovací obrazovku vidíme na Obrázku 5, pro používání aplikace nám nyní postačí uživatelské jméno a heslo, které obdržíme od administrátora aplikace. Po úspěšném přihlášení můžeme využívat všechny možnosti, které aplikace nabízí. V následujících kapitolách budou tyto funkce popsány.



Obrázek 6 Přihlašovací obrazovka

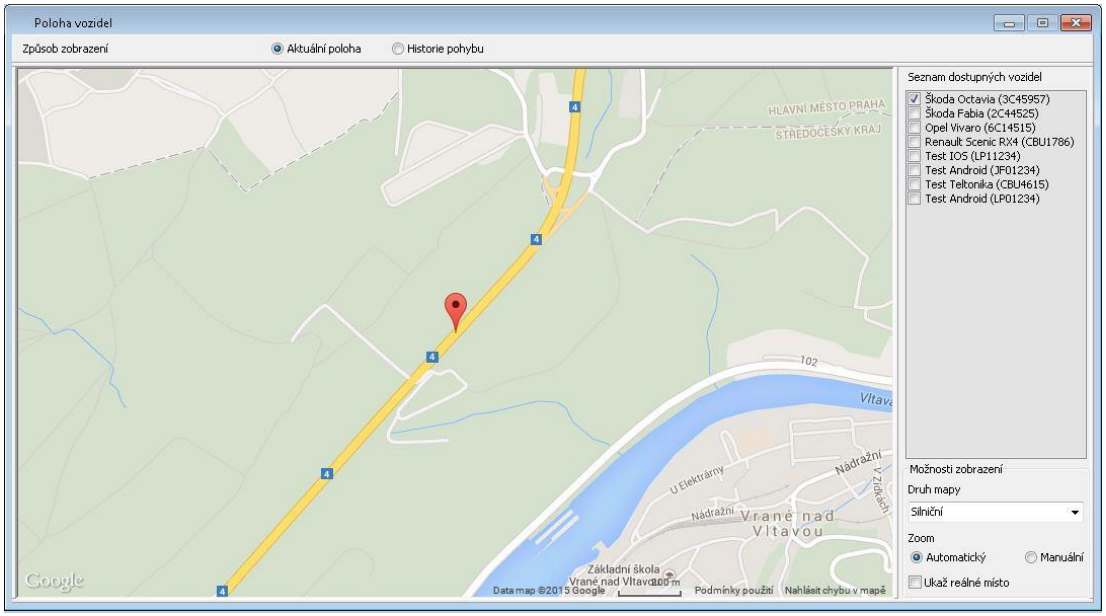
Po přihlášení se zobrazí úvodní obrazovka se základním navigačním menu viz Obrázek 6.



Obrázek 7 Úvodní obrazovka

Poloha vozidel

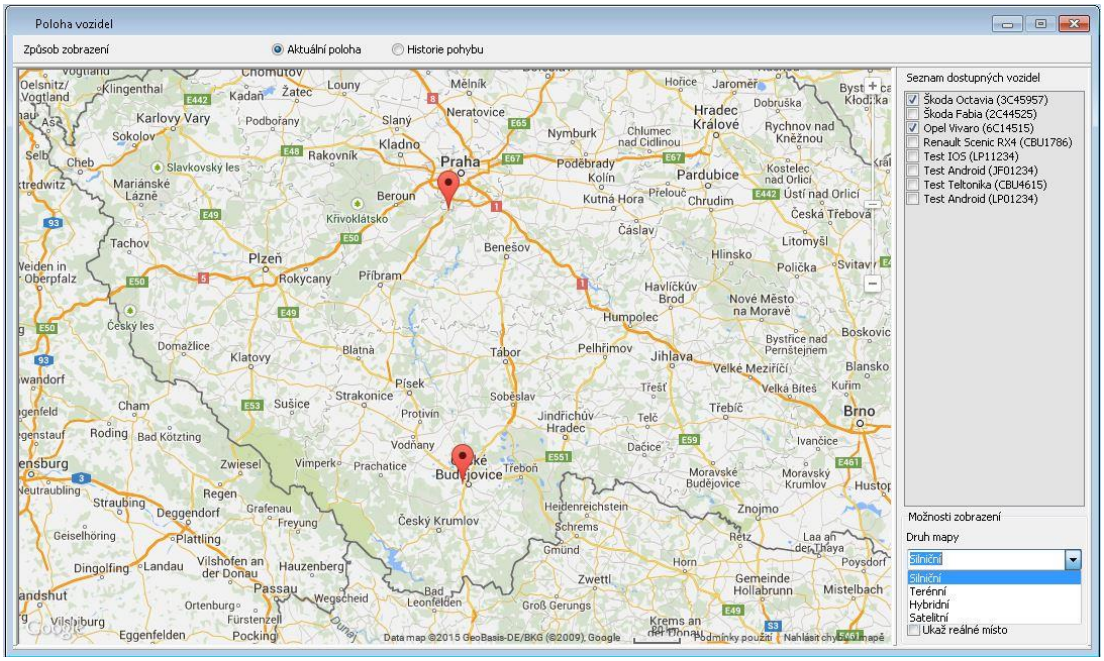
Při zvolení možnosti *Poloha vozidel* se otevře nové okno, ve kterém se zobrazí náhled silniční mapy (pokud ve výchozím nastavení není uvedeno jinak, tato volba bude zmíněna později). V pravé části se nachází *Seznam dostupných vozidel* respektive sledovaných objektů jak znázorňuje Obrázek 7.



Obrázek 8 Poloha vozidla

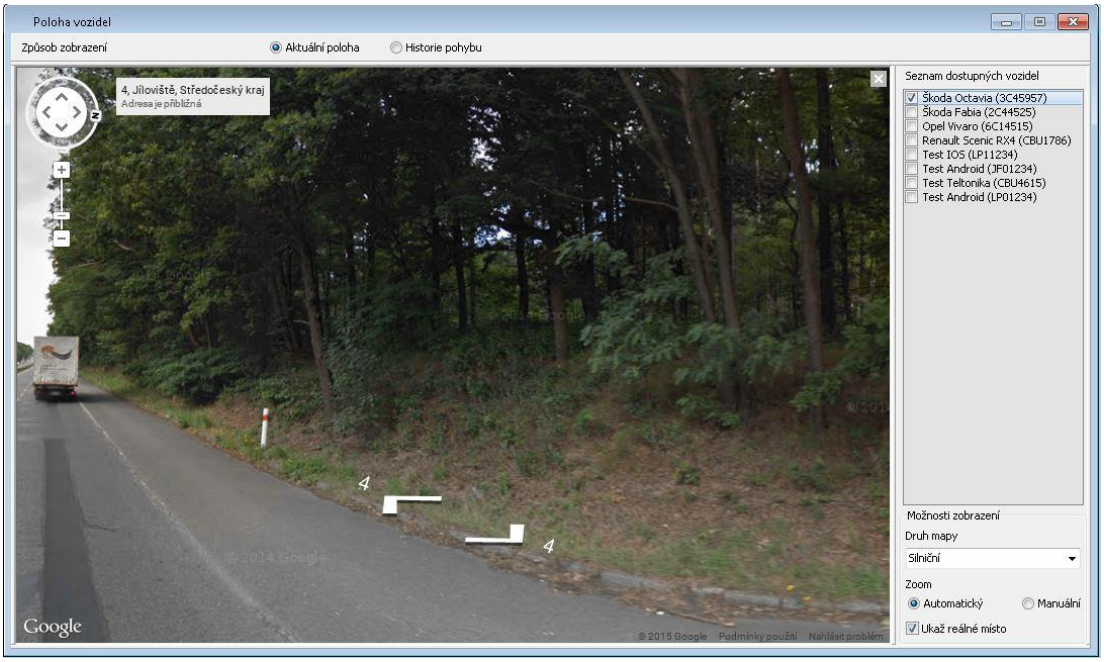
Po zaškrtnutí jednoho či více vozidel (objektů), se na mapě ukáže aktuální poloha daného vozidla (vozidel, objektů). V možnostech zobrazení lze zvolit druh mapy, na které chceme polohu zobrazit (*Silniční, Terénní*…). Velikost přiblížení (*Zoom*) lze nastavit automaticky nebo manuálně. Při automatické volbě se velikost přiblížení mapy nastaví tak, aby poloha vozidla byla ideálně vidět. Při manuálním nastavením lze ručně pohybovat mapou a pomocí ovládacích prvků na pravé straně

mapu přiblížit či oddálit viz následující obrázek.



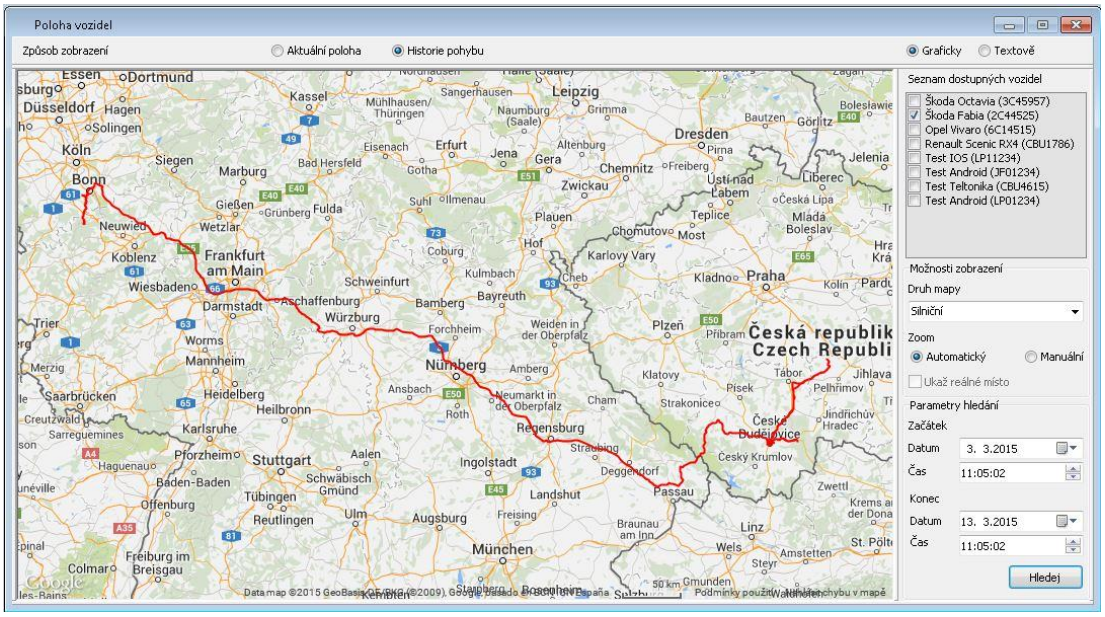
Obrázek 9 Poloha vozidel, druh mapy

Pokud zaškrtneme volbu *Ukaž reálné místo*, zobrazí se náhled aktuálního místa vozidla pomocí Google Street View (pokud je tato volba dostupná). Obraz reálného místa se změní vždy, po zaslání aktuální polohy (každých 20 vteřin) viz Obrázek 9



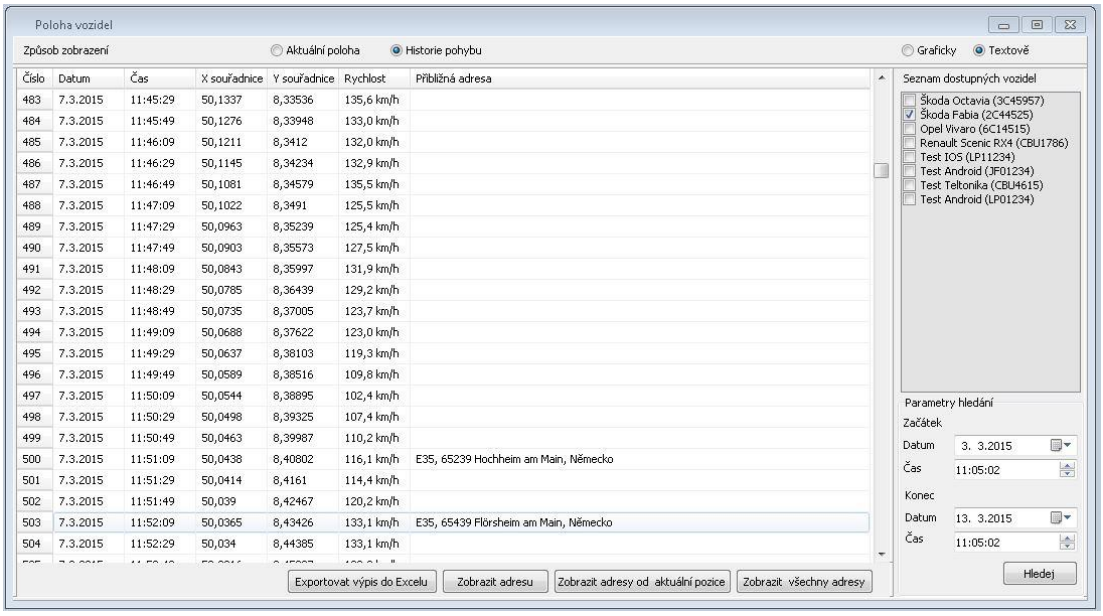
Obrázek 10 Poloha vozidel – reálné místo

Ve volbě *Historie pohybu* si můžete nechat vykreslit trasu vozidla za určité období, které nastavíte pomocí kalendáře v části *Parametry hledání*. Při výběru delšího časového období může vykreslení chvilku trvat a systém se může jevit jako nečinný.



Obrázek 11 Historie pohybu - graficky

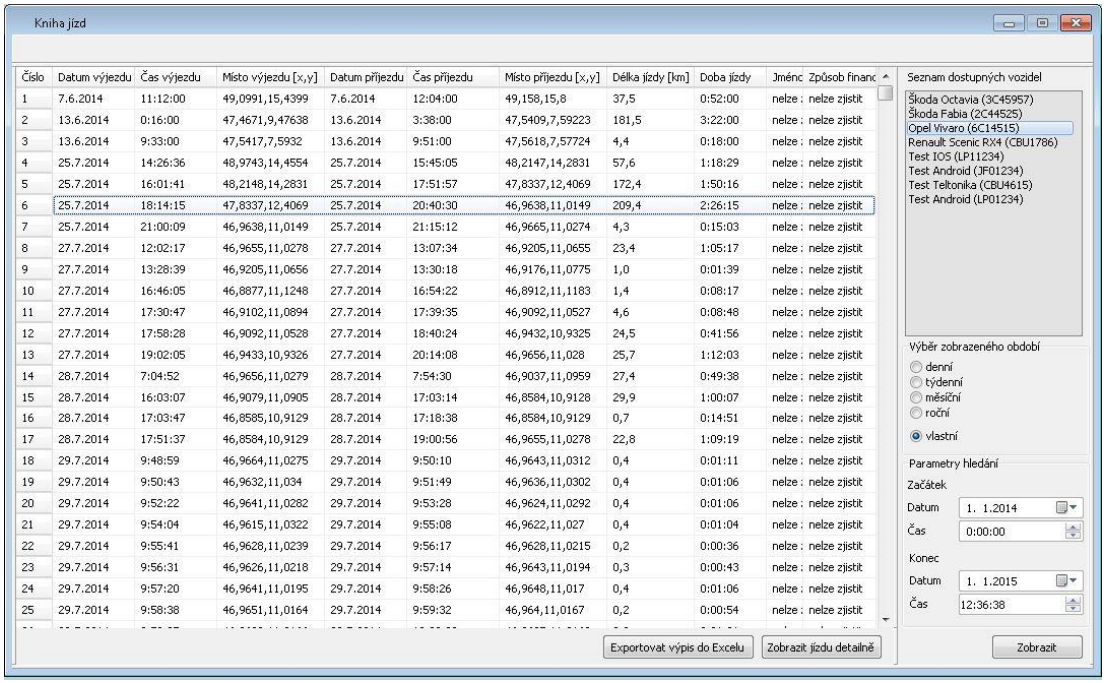
V případě, že preferujeme textový výpis, zvolíme jako způsob zobrazení *Textově.* V tomto případě se zobrazí soupis všech naměřených hodnot, které obsahují *Datum*, *Čas, souřadnice X a Y*, *Rychlost* v měřeném okamžiku. Poslední sloupec je defaultně prázdný a je určen pro zobrazení *Přibližné adresy*. Pokud chceme vypsat konkrétní adresy nebo ji kontinuálně vypisovat od určité pozice, případně zobrazit všechny adresy, použijeme jedno z tlačítek umístěných ve spodní části výpisu (*Zobrazit adresu, Zobrazit adresy od aktuální pozice, Zobrazit všechny adresy*). Celou tabulku můžeme exportovat do Excelu pro případné další zpracování prostřednictvím tlačítka *Exportovat výpis do Excelu*.



Obrázek 12 Historie pohybu - textově

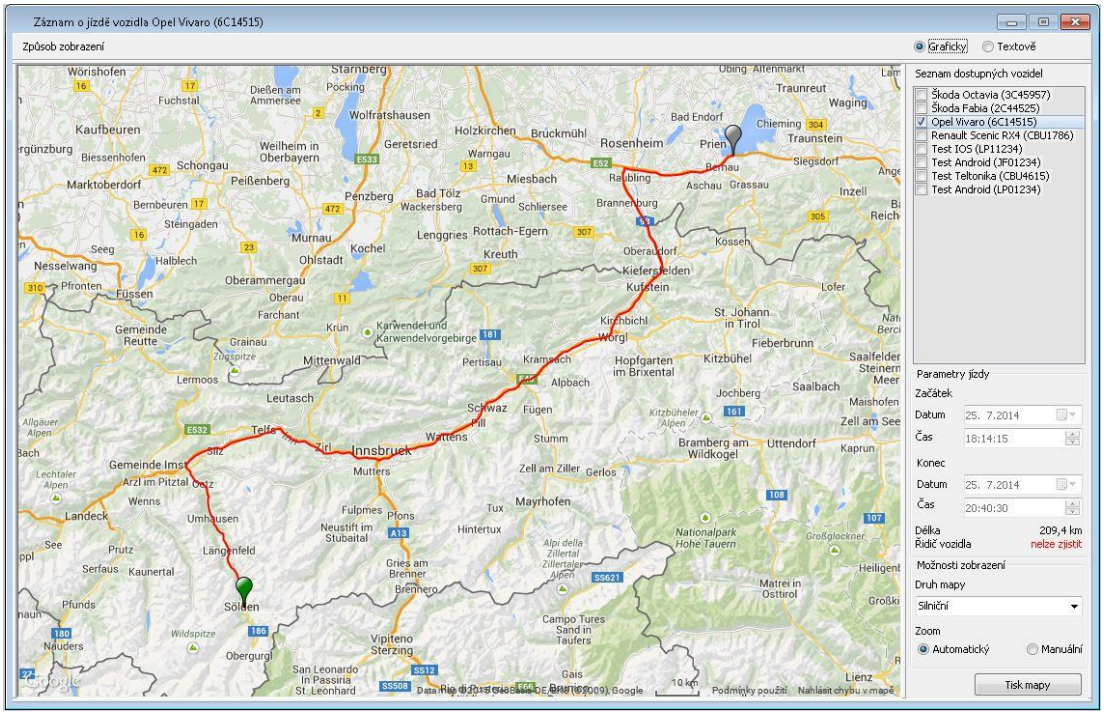
Kniha jízd

Kniha jízd by měla sloužit k zobrazení všech jízd jednotlivých vozidel za dané časové období. V první řadě musíme vybrat, u kterého vozidla chceme zobrazit knihu jízd a následně zvolit období, které chceme zobrazit. Období lze vybrat buď pomocí kalendáře v části *Parametr hledání*, nebo pomocí přednastavené volby v části *Výběr zobrazeného období (denní, týdenní…)*. Po kliknutí na tlačítko *Zobrazit* se vypíše podrobná kniha jízd daného vozidla. U každé jízdy jsou zaznamenávány následující informace: *Datum výjezdu, Čas výjezdu, Místo výjezdu, Datum příjezdu, Čas příjezdu, Místo příjezdu, Délka jízdy, Doba jízdy, Jméno řidiče* a *Způsob financování*. Tuto knihu jízd lze exportovat do Excelu a dále zpracovávat například pro měsíční či roční uzávěrky, pomocí tlačítka *Exportovat výpis do Excelu* viz Obrázek 12 Kniha jízd.



Obrázek 13 Kniha jízd

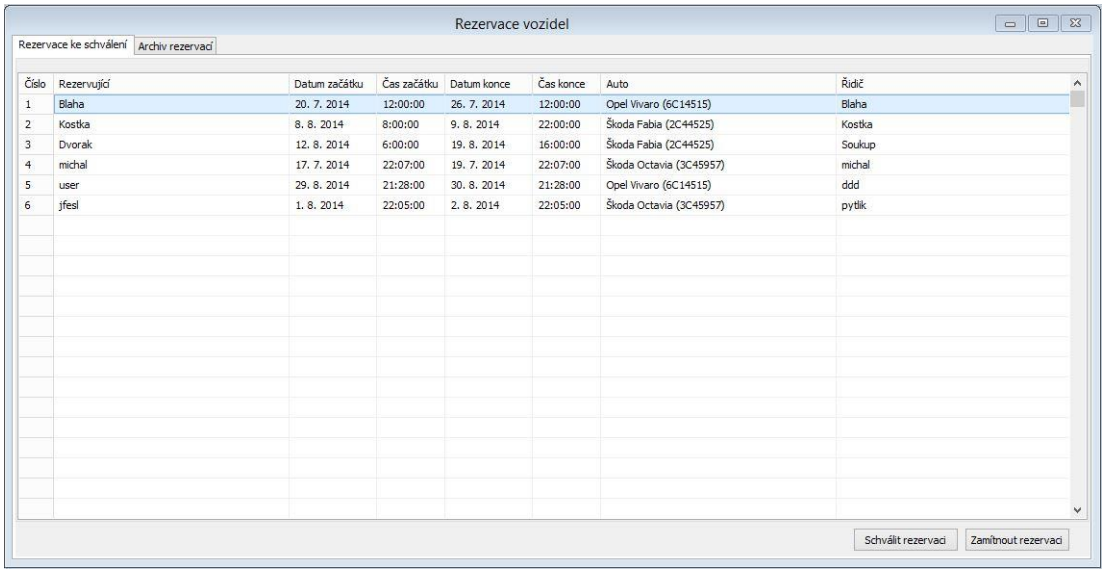
Pokud nás zajímá konkrétní jízda detailněji, stačí označit danou jízdu a stisknou tlačítko *Zobrazit jízdu detailně*. Otevře se nové okno s vykresleným pohybem vozidla v dané jízdě. Opět si můžeme zvolit druh mapy a nastavení zoomu. V části *Parametry jízdy* je zobrazena délka trasy a jméno řidiče. Jízdu si opět můžeme vypsat graficky nebo textově stejně jako u polohy vozidel. Tuto mapu je možné vytisknout pomocí zlatíčka *Tisk mapy*.



Obrázek 14 Kniha jízd - detail

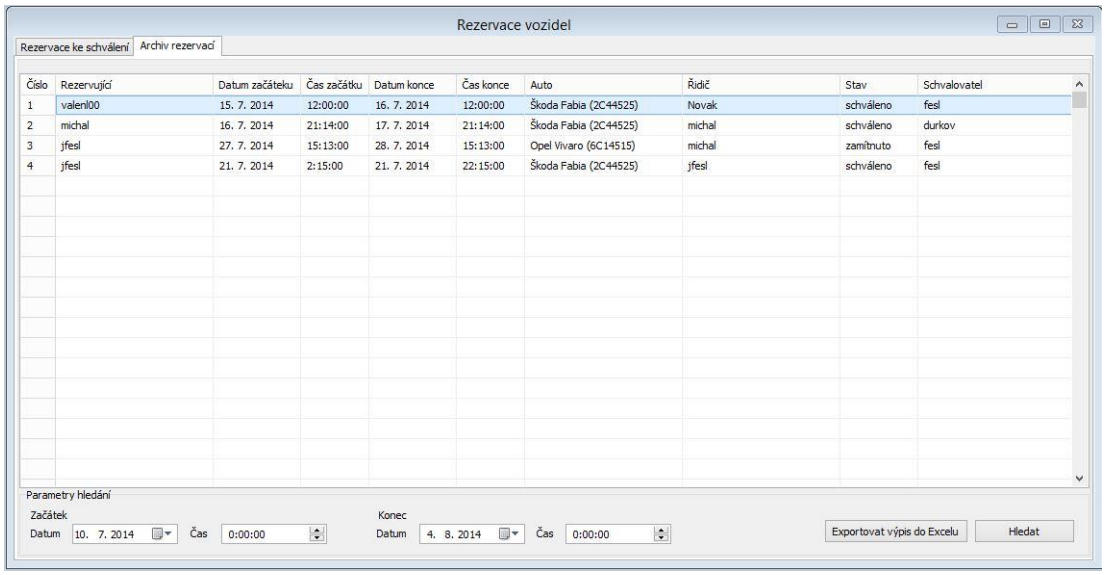
Rezervace vozidel

Na kartě rezervace vozidel nalezneme seznam uživatelů, kteří žádají o rezervaci vozidla. Zobrazeny jsou základní informace o dané rezervaci tj. *Rezervující, Datum začátku rezervace, Čas začátku rezervace, Datum a Čas konce rezervace*, *Auto* které má být rezervováno a v neposlední řadě *Řidič* vozidla. Schválení či zamítnutí rezervace je možné pomocí dvou tlačítek *Schválit rezervaci* a *Zamítnout rezervaci*. V případě zamítnutí rezervace lze uvést i důvod zamítnutí. V obou případech systém vyžaduje potvrzení akce.



Obrázek 15 Rezervace vozidel

Na kartě *Archiv* rezervací si můžeme zobrazit veškeré rezervace (schválené a zamítnuté) za zvolené časové období, které se nastavuje v parametru hledání stejným způsobem jako v jiných částech systému. Výpis můžete následně exportovat do Excelu pomocí tlačítka *Exportovat výpis do Excelu*.



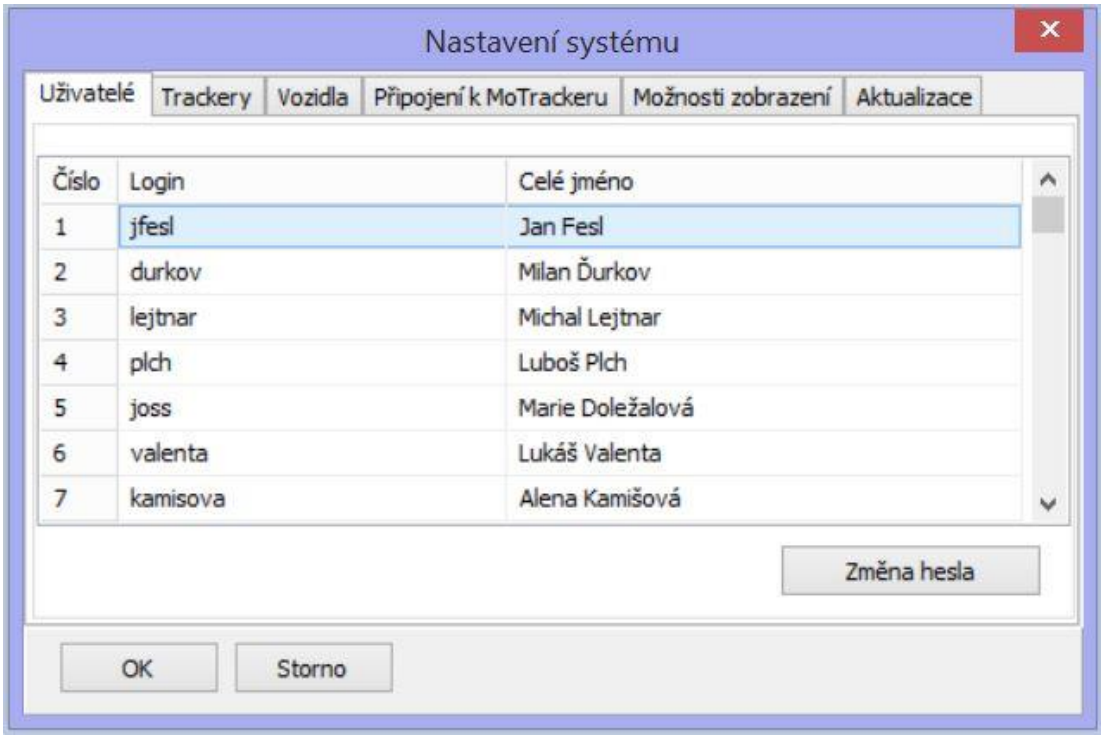
Obrázek 16 Archiv rezervací

Nastavení systému

V této části se nastavuje přístup k aplikaci a aplikace samotná. Některé karty vyžadují vyšší oprávnění pro provedení změny nastavení. Nastavení systému obsahuje následující záložky:

Uživatelé

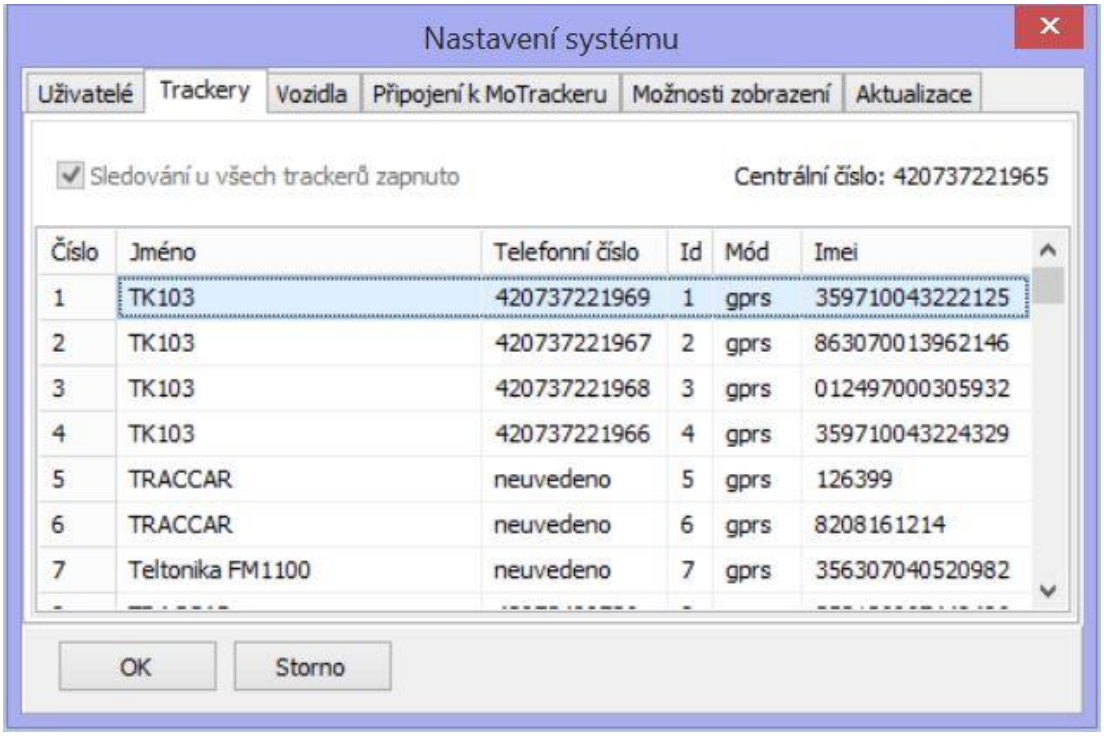
Seznam uživatelů včetně uživatelského jména. Každý uživatel si zde může změnit své heslo. Změnu hesla u jiného uživatele může provádět pouze uživatel s administrátorským oprávněním viz následující obrázek.



Obrázek 17 Nastavení systému – uživatelé

Trackery

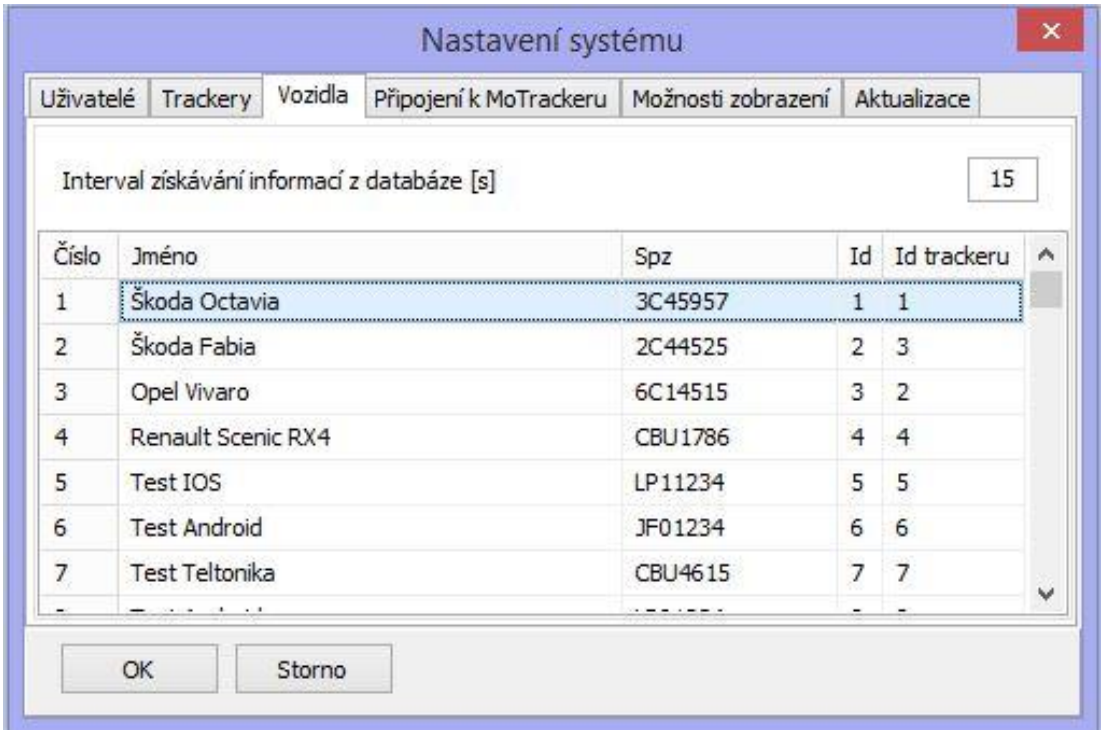
Seznam všech sledovacích zařízení (přenosná, namontovaná v automobilu nebo mobilní telefon) s telefonním číslem SIM karty, umístěné v zařízení včetně módu pro přenos informací, ID zařízení a IMEI.



Obrázek 18 Nastavení systému - Trackery

Vozidla

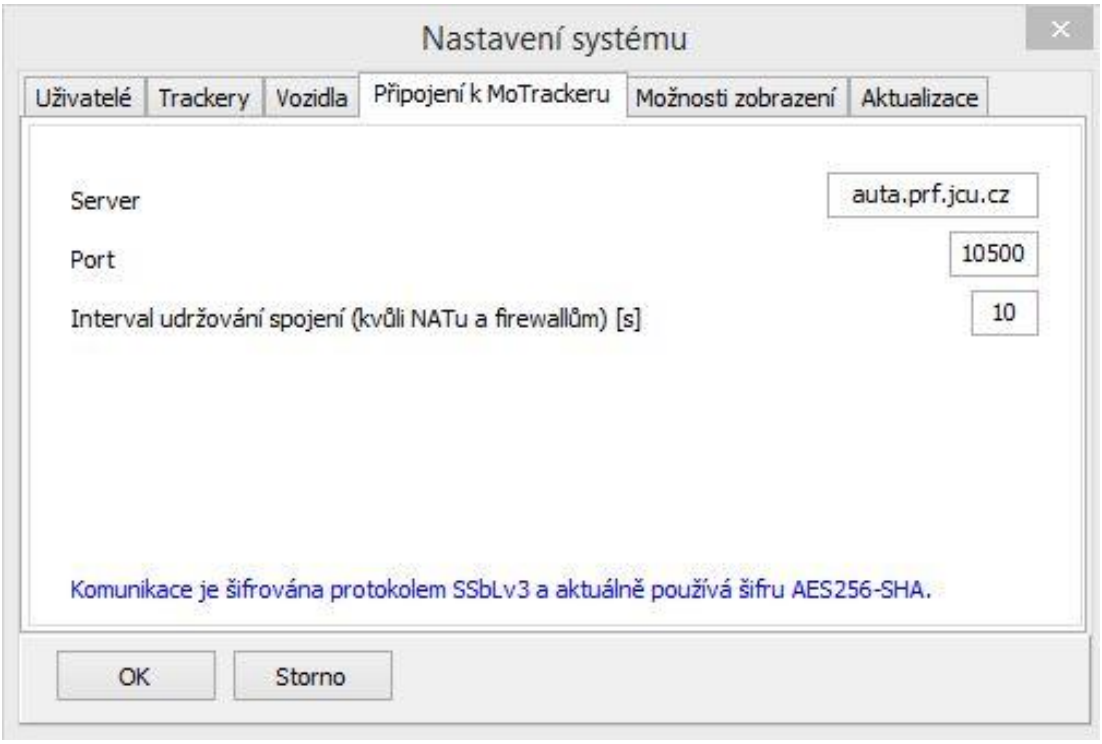
Typy vozidel a přenosná zařízení, ve kterých jsou trackery umístěny. U každého vozidla a zařízení je uvedena státní poznávací značka a ID trackeru, podle kterého lze najít používané telefonní číslo ve vozidle.



Obrázek 19 Nastavení systému - Vozidla

Připojení k MoTrackeru

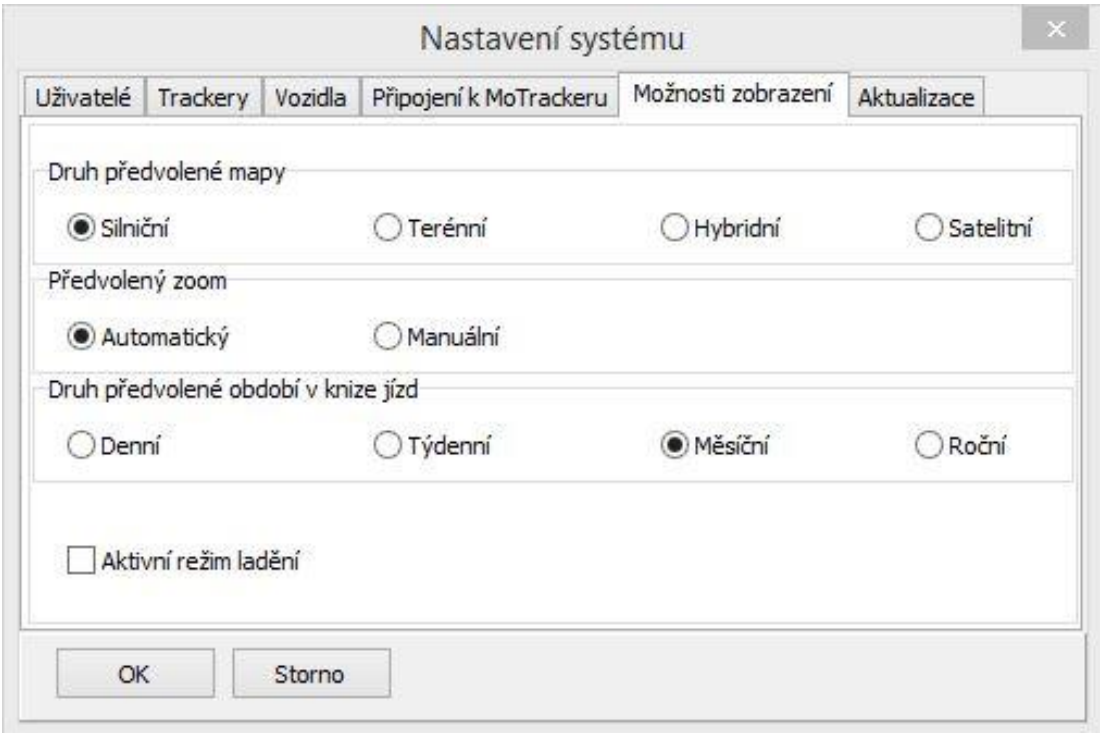
Do pole *Server* vkládáme název serveru, na kterém jsou veškerá data o záznamu pohybu vozidel a přenosných zařízení. Pole *Port* obsahuje číslo portu, na kterém klient komunikuje se serverem. *Interval udržování spojení* je časový interval pro udržení spojení se serverem viz Obrázek 19.



Obrázek 20 Nastavení systému – Připojení k MoTracker

Možnosti zobrazení

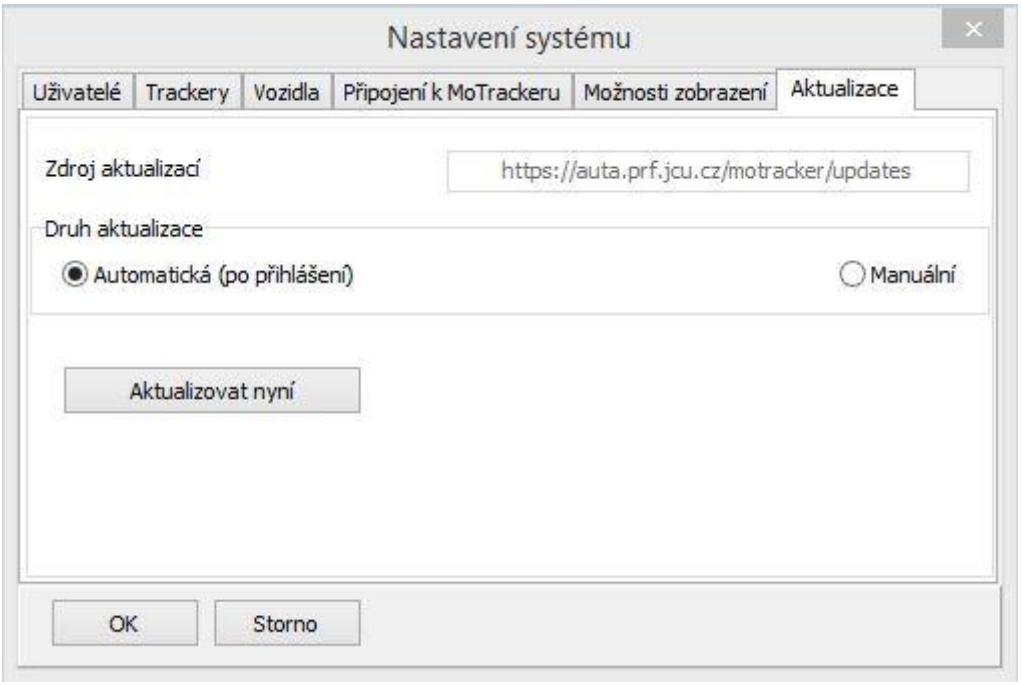
Zde si můžete nastavit preferované výchozí zobrazení mapy, přiblížení a období v knize jízd.



Obrázek 21 Nastavení systému – Možnosti zobrazení

Aktualizace

V této kartě si nastavíte, zda chcete aplikaci automaticky aktualizovat (výchozí nastavení) při každém přihlášení, nebo budete-li aktualizovat aplikaci manuálně (nedoporučuje se). Tlačítkem *Aktualizovat* nyní provedete kontrolu případně instalaci nové verze.



Obrázek 22 Nastavení systému – Aktualizace

Migrace systému

Systém pro monitorování mobilních systémů na stávající hardwarové platformě fungoval, ale vzhledem k stále rostoucí databázi a častějšímu přístupu k aplikaci od více uživatelů, je systém vytížen na 95% svého výkonu. Nejslabším článkem je procesor, následně disk a operační paměť. Problém by se dal vyřešit zálohováním a následným smazáním starších záznamů databáze v pravidelných intervalech, což pro uživatele není příliš přívětivé. Pro menší firmy, kde k aplikaci bude přistupovat z pravidla jedna osoba, maximálně dvě, výkon postačí. Nicméně instituce větších rozměrů, kde může být přihlášeno několik uživatelů najednou a všichni mohou mít požadavky na záznam polohy vozidel se ukázalo, že stávající výkon je nedostačující pro plynulý chod aplikace. Aplikace se díky nedostatečnému výkonu občas zasekne, nebo výpis dat trvá nepřiměřeně dlouhou dobu. Z tohoto důvodu jsem se rozhodl přesunout celou aplikaci na výkonnější hardware. Bylo by možné provést novou instalaci operačního systému a následné nastavení aplikace (což jsem nakonec musel beztak udělat), ale chtěl jsem přesun udělat co možná nejjednodušší. Rozhodl jsem se o přesunutí pomocí bitové kopie zdrojového disku.

Samotný přesun byl proveden na PC s čtečkou paměťových karet. Paměťová karta se připojila pomocí příkazu *mount*. Následně byl použit příkaz *dd*, který slouží k zálohování jednotlivé partition případně celého disku. Parametry příkazu jsou *if* – input file tzn. cesta k adresáři, který chceme zálohovat, *of* – output file tzn. cesta k adresáři případně souboru, kam chceme zálohovat, *bs* – blocks size, nebo-li velikost bloku při vytváření zálohy. Abychom zjistili správný název zdrojového disku, použijeme příkaz *fdisk -l* . V mém případě vypadal příkaz následovně *dd if=/dev/sdc1 of=lukas.img bs=1M.* Příkaz *dd* vytvoří bitovou kopii celého zdrojového disku bez ohledu na obsazení, proto je nutné počítat s tím, že vytváření daného img souboru nějakou dobu potrvá. Flash disk ze kterého jsem dělal zálohu má kapacitu 32 GB a záloha trvala zhruba 20 minut. Vytvořený soubor jsem uložil na flash disk, který poslouží jako zdroj pro obnovu na novém hardwaru.

Nyní když máme vytvořenou bitovou kopii, je nutné ji obnovit na silnějším hardwaru. Nový hardware je starší běžné PC s dvoujádrovým procesorem, který by měl výkonově postačit a s 2 GB operační paměti. Na PC je nutné nabootovat z live

distribuce Linuxu, kterou je například dobře známé SystemRescueCd. Na lokálním disku je třeba vytvořit oddíl, kam budeme zálohu obnovovat. Obnovu provedeme stejným způsobem, jakým jsme vytvářeli zálohu. Použijeme opět příkaz *dd* jen s tím, že parametr *if* bude odkazovat na vytvořený img soubor, který máme uložený na flash disku. Parametr *of* bude odkazovat na lokální disk. Dalším důležitým krokem je instalace a aktualizace zavaděče. Dále úprava souboru */etc/inittab*. Po restartu systém naběhne ve stejné podobě jako na předchozím hardwaru. Posledním krokem je aktualizace a kontrola systému. Po kontrole a aktualizacích je systém plně funkční.

Jelikož se jednalo o ořezanou verzi Debianu, při upgradu na novější verzi došlo k potížím se systémem. Aplikace přestala z neznámého důvodu pracovat a nepodařilo se my systém přivést do funkčního stavu. Proto jsem raději celý systém nainstaloval znova a vše šlo poté bez problémů. Při upgradu na čisté instalaci se problém již neprojevil.

Při testování, zda se chod aplikace zlepšil, jsem zjistil, že databáze vytěžuje procesor zhruba na 60 – 80% výkonu. Při výpisu stejné jízdy je rychlost zobrazení znatelně rychlejší a systém se nezasekává. Pochopitelně, že při výpisu velkého časového období je prodleva delší, ale v porovnání s původním měřením znatelně kratší. Musíme brát v potaz, že každé auto zaznamenává svou polohu relativně často a tudíž při požadavku na výpis pohybu vozidla za období několika měsíců je dat opravdu hodně. Výkon by se dal bezpochyby navýšit ale otázkou je, zda by to po finanční stránce bylo výhodné. Přeci jen výpisy za dlouhé období nejsou očekávány každý den a při nutnosti zobrazení takového výpisu bude uživatel seznámen s drobnou prodlevou.

Závěr

Cílem této diplomové práce bylo vybrat a zprovoznit vhodnou embedded platformu, na které bude celá řídící aplikace včetně databáze a všech potřebných součástí bezproblémově fungovat. K tomu patří volba a následná instalace vhodného operačního systému. Základním kritériem výběru je pořizovací cena, rozšiřitelnost systému, dostatek vstupně výstupních portů a možnost portace operačního systému.

Dalším cílem této práce bylo navrhnout komunikační protokol pro vzdálenou komunikaci mezi řídící aplikací a klientskou aplikací, kterou bude mít uživatel na svém PC. K tomu bezpochyby patří vytvoření controlleru, který zajišťuje zpracování a předání požadavků ze sítového protokolu do ostatních částí aplikace jako například databáze a GPRS modul. Vlastní realizace komunikačního protokolu je realizována na bázi multiprocesového tcp-démona, který je částí spuštěného operačního systému.

Poslední část této práce se zabývá přenesením systému na jiné hardwarové zařízení. Cílem bylo ověřit možnost portace, což je jeden z přidružených cílů. Přenesení systému se po menších problémech povedlo.

Projekt MoTracker by neměl zaniknout po ukončení studia a bude dále vyvíjen a zlepšován.

Z výše uvedených poznatků považuji veškeré cíle této práce za úspěšně splněné. Některé úkoly, např. aplikace MoTracker byly provedeny nad rámec samotného zadání, nicméně byly nutné k úspěšnému provozu celého projektu.

Seznam použitých zkratek

API: Application Programming Interface označuje v informatice rozhraní pro programování aplikací.

BIOS: Basic Input/Output Setup - implementuje základní vstupně-výstupní funkce pro počítače a představuje firmware pro osobní počítače. V současné době se BIOS používá hlavně při startu počítače pro inicializaci a konfiguraci připojených hardwarových zařízení a následnému spuštění operačního systému, kterému je pak předáno další řízení počítače.

BOOTP: Bootstrap Protocol (zkratka BOOTP) je v informatice síťový protokol, který se používal v počítačových sítích pro nastavení parametrů pro stanice používající TCP/IP. BOOTP je definován v RFC 951 z roku 1985. V současné době je nahrazen protokolem DHCP.

CPU: Central Processor Unit (centrální procesorová jednotka) - je základní součást každého počítače. CPU je nejmenší součást počítače schopná samostatně provádět strojový kód programu. V současné době bývá CPU implementována jako součást mikroprocesoru nebo vícejádrového procesoru.

DC: Direct Current (stejnosměrná elektrický proud) - je elektrický proud, který protéká obvodem stále stejným směrem, na rozdíl od proudu střídavého.

DNS: Domain Name Systém - je hierarchický systém doménových jmen, který je realizován servery DNS a protokolem stejného jména, kterým si vyměňují informace. Jeho hlavním úkolem a příčinou vzniku jsou vzájemné převody doménových jmen a IP adres uzlů sítě.

DRAM: Dynamic Random Access Memory - je druh počítačové paměti, která uchovává data v podobě elektrického náboje v kondenzátoru, který odpovídá

parazitní kapacitě řídící elektrody tranzistoru typu MOS. Tento tranzistor současně slouží i jako čítací prvek paměťové buňky - bitu. V praxi byly běžné typy DRAM nahrazeny modernějšími synchronními typy SDRAM a DDR SDRAM.

DHCP: Dynamic Host Configuration Protocol - protokol z rodiny TCP/IP nebo označení odpovídajícího DHCP serveru či klienta. Používá se pro automatickou konfiguraci počítačů připojených do počítačové sítě. DHCP server přiděluje počítačům pomocí DHCP protokolu zejména IP adresu, masku sítě, implicitní bránu a adresu DNS serveru.

GPRS: General Packet Radio Service (GPRS) je služba umožňující uživatelům mobilních telefonů GSM přenos dat a připojení k Internetu (případně k jiným sítím).

GSM: Globální Systém pro Mobilní komunikaci je nejrozšířenější standard pro mobilní telefony na světě.

IPIX: Internetwork Packet Exchange (IPX) je protokol síťové vrstvy v sadě protokolů IPX/SPX. Může fungovat i jako nespojovaný protokol transportní vrstvy. IPX je odvozen od svého protějšku Internet Datagram Protocol (IDP) v sadě protokolů Xerox Network Services (XNS) firmy Xerox.

NBP: Network Bootstrap Program - vzdálený zaváděcí obraz stažený pomocí PXE přes TFTP

NCP: NetWare Core Protocol (NCP) je síťový protokol používaný pro komunikaci klientských stanic se servery v síťovém operačním systému Novell NetWare.

PCI: Peripheral Component Interconnect - sběrnice používaná pro připojování přídavných karet k základní desce. Pomocí PCI sběrnice je možné připojit například grafickou kartu, síťovou kartu, zvukovou kartu, TV karty apod.

POE: Power over Ethernet - je napájení po datovém síťovém kabelu, bez nutnosti přivést napájecí napětí k přístroji dalším samostatným kabelem. Často využívané například pro IP kamery a Wi–Fi přístupové body.

POST: Power On Self Test - je diagnostický program, který kontroluje hardware v zařízení a zároveň i jejich součinnost. Spouští se automaticky po startu přístroje (počítač, router nebo i tiskárna).

PXE: Pre-boot eXecution Environment - označení technologie pro bootování (tj. start) počítačů z počítačové sítě

RJ45: Koncovka RJ-45 je dnes nejčastěji používaný typ zapojení síťových UTP kabelů.

SFF: Small From Factor – jedná se o počítač resp. počítačovou skříň, která je navržena tak aby byla znatelně menší než běžná počítačová skříň, při zachování stejné funkčnosti a srovnatelného výkonu jako běžná počítačová skříň. Vyrábí se v různých tvarech a velikostech, nejčastěji však v rozměrech 34 x 38 x 10 cm (Š x H x V)

SMS: Služba krátkých textových zpráv (z anglického Short message service) je název pro službu dostupnou na většině digitálních mobilních telefonů.

SSL: Secure Sockets Layer, SSL (doslova vrstva bezpečných socketů) je protokol, resp. vrstva vložená mezi vrstvu transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran. Následovníkem SSL je protokol Transport Layer Security (TLS).

TCP/IP: je sada protokolů, sloužící ke komunikaci v počítačové síti a propojení dvou počítačů. Základ tohoto modelu tvoří protokoly TCP (Transmision Control Protocol) a IP (Internet Protocol). Model TCP/IP poskytuje propojení mezi dvěma

koncovými uzly a specifikuje, jak budou data rozděleny na pakety, označeny adresami, přenášeny, směrovány a přijímány v cílovém bodě.

TFTP: Trivial File Transfer Protocol - velice jednoduchý protokol pro přenos souborů, obsahující jen základní funkce protokolu FTP. TFTP je určen pro přenos souborů v případech, kdy je běžný protokol FTP nevhodný pro svou komplikovanost. Typickým případem je bootování bezdiskových počítačů ze sítě, kdy se celý přenosový protokol musí vejít do omezeného množství paměti, která je k dispozici na bezdiskovém stroji.

UDP: User Datagram Protocol - jeden ze sady protokolů internetu též známý jako nespolehlivý. Na rozdíl od protokolu TCP totiž nezaručuje, zda se přenášený datagram neztratí, zda se nezmění pořadí doručených datagramů, nebo zda některý datagram nebude doručen vícekrát.

USB: Universal Serial Bus - univerzální sériová sběrnice, moderní způsob připojení periferií k počítači. Nahrazuje dříve používané způsoby připojení (sériový a paralelní port, PS/2, Gameport apod.) pro běžné druhy periférií – tiskárny, myši, klávesnice, joysticky, fotoaparáty, atd., ale i pro přenos dat z videokamer, čteček paměťových karet, MP3 přehrávačů, externích pevných disků a externích optických mechanik.

USFF: Ultra Small From Factor: obdoba SFF nejčastěji v rozměrech 7 x 24 x 24 cm (Š x H x V)

Přílohy

Obsah přiloženého CD:

Na přiloženém cd se nacházejí následují přílohy:

controller- zdrojové kódy

remoteAdmin- zdrojové kódy

database-zdrojové kódy

gprsModul-zdrojové kódy

Diplomová práce

Seznam obrázků

Obrázek 1 Blokové schéma 4
Obrázek 2: Základní deska ALIX.2D13, převzato z 7
Obrázek 3 Blokové schéma serveru 23
Obrázek 4 Příklad ustavení spojení s využitím BSD sockets, převzato z 28
Obrázek 5 Schéma databáze 30
Obrázek 6 Přihlašovací obrazovka 35
Obrázek 7 Úvodní obrazovka 36
Obrázek 8 Poloha vozidla 36
Obrázek 9 Poloha vozidel, druh mapy 37
Obrázek 10 Poloha vozidel – reálné místo 37
Obrázek 11 Historie pohybu - graficky 38
Obrázek 12 Historie pohybu - textově 39
Obrázek 13 Kniha jízd 40
Obrázek 14 Kniha jízd - detail 41
Obrázek 15 Rezervace vozidel 42
Obrázek 16 Archiv rezervací 42
Obrázek 17 Nastavení systému – uživatelé 43
Obrázek 18 Nastavení systému - Trackery 44
Obrázek 19 Nastavení systému - Vozidla 45
Obrázek 20 Nastavení systému – Připojení k MoTracker 46
Obrázek 21 Nastavení systému – Možnosti zobrazení 47
Obrázek 22 Nastavení systému – Aktualizace 48

Seznam použité literatury

[1] Preboot Execution Environment. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001. [cit. 2014-04-19]. Dostupné z: http://cs.wikipedia.org/wiki/Preboot_Execution_Environment.

[2] Preboot Execution Environment (PXE) Specification [online]. [cit. 2014-07-15]. Dostupné z: http://www.pix.net/software/pxeboot/archive/pxespec.pdf.

[3] MAREK, Vlastimil. *Něco v síti: fejetony, které vycházely od roku 1997 na internetu na adrese http://svet.namodro.cz*. 1. vyd. Praha: Dharma Gaia, 1999, 172 s. ISBN 808601357x.

[4] Peterka, Jiří. *Když se řekne počítačová síť* [online]. [cit. 2015-01-25]. Dostupné z: http://www.earchiv.cz/a91/a139c110.php3

[5] Dostál, Radim. *Seriál Sokety a C/C++* [online]. [cit. 2015-03-19]. Dostupné z: http://www.root.cz/serialy/sokety-a-cc/.

[6] Berkeley sockets. In: *Wikipedia: the free encyclopedia* [online]. [cit. 2014-04-19]. San Francisco (CA): Wikimedia Foundation, 2001. Dostupné z: http://cs.wikipedia.org/wiki/Berkeley_sockets.

[7] Linux Programmers Manual [online]. Dostupné z: http://manned.org/socket