ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

Bakalárska práca

Monitorovanie mestskej dopravy pomocou plávajúcich vozidiel

Praha, 2008

Autor: Marek Riedel

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Marek Riedel**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření

Název tématu: **Monitorování městské dopravy pomocí plovoucích vozidel**

Pokyny pro vypracování:

1. Seznamte se se zařízením GC 072 240F, umožňující měřit svojí GPS polohu a tuto odesílat pomocí GPRS na server v Internetu.
2. Navrhněte a implementujte program zajišťující komunikaci s uvedeným zařízením. Toto zařízení nastavte tak, aby periodicky měřilo svoji polohu, která bude pomocí vašeho programu s doplňkovými údaji ukládána do databáze.
3. Navrhněte a implementujte webové rozhraní pro prezentaci naměřených dat.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Michal Kutil

Platnost zadání: do konce zimního semestru 2008/2009

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 25. 2. 2008

Prehlásenie

Prehlasujem, že som svoju bakalársku prácu vypracoval samostatne a použil som iba podklady (literatúru, projekty, SW atď.) uvedené v priloženom zozname.

V Prahe, dňa ……. …….

Podpis

Anotácia

Zber GPS dát z dopravy pomocou plávajúcich vozidiel. Plávajúce vozidlá sú vozidlá, ktoré jazdia v mestskej doprave a v pravidelných intervaloch zaznamenávajú svoju GPS pozíciu. Takýto záznam a odoslanie GPS polohy umožňuje aj použité zariadenie POSITRA GC 072 240F firmy LEVELNA. Tieto namerané pozície sa odosielajú na server, kde sa uložia. Dáta sa následne vyhodnocujú a určuje sa z nich stav dopravy v meste. Moja práca sa zaoberá iba zberom a ukladaním nameraných dát.

Annotation

Summarising and monitoring GPS data information from public transport by floating vehicles. Floating or probe vehicles are all vehicles in public transport, equipped with special units able to periodically measure their location through GPS. All the information about the GPS location of vehicle could be sent for example by POSITRA GC 072 240F equipment produced by LEVELNA. All the information are sent and stored on the server. Data information are analysed and they show the public transport vehicle position. My diploma paper is dealing with summarising and storing of already measured data information.

Obsah

1. Úvod 3

2. Popis komunikátora 4

2.1. Popis hardvéru 4

2.2. Spôsoby konfigurácie komunikátora 6

2.2.1. SMS konfigurácia 6

2.2.2. USB konfigurácia 7

2.2.3. Zaslanie konfigurácie prostredníctvom GPRS 8

2.2.4. Základná konfigurácia zariadenia 8

2.3. Používané funkcie zariadenia 9

3. Serverová služba 10

3.1. UDP server 10

3.1.1. Príjem prvých dát a oživenie komunikácie so zariadením 11

3.1.2. Efektivita prijímania dát 12

3.2. Návrh používaných funkcií 16

3.2.1. Konfigurácia aplikácie a používané konštanty 16

3.2.2. Práca so súbormi 16

3.2.3. Metódy pre komunikáciu so zariadením 17

3.3. Návrh databázy 18

3.3.1. Oddelenie databázovej vrstvy od aplikačnej 21

3.3.2. MS SQL databázové procedúry 22

3.4. Viacvláknový UDP server 22

3.4.1. Pomocné vlákno ConfigCron 24
3.4.2. Inicializácia obslužného vlákna 25
3.4.3. Opakované odoslanie správy 25
3.4.4. Spracovanie odpovede na vyžiadanú službu 26
3.4.5. Ukončenie obslužného vlákna 26
3.4.6. Vzdialená konfigurácia komunikátora 26
3.4.7. Postup odosielania vzdialenej konfigurácie 27
4. Webové rozhranie 28
4.1. Rozloženie stránky 28
4.2. Prihlásenie 29
4.3. Správa zariadení 32
4.4. Zobrazenie prijatých dát na mape 32
4.5. Výber zobrazovaných dát 32
4.6. Zobrazenie vybraných dát 34
4.6.1. Výber dát na strane databázového servera 34
4.6.2. Spracovanie vrátených dát 34
4.6.3. Vykreslenie dát 35
4.7. Tabuľkové zobrazenie prijatých dát 36
5. Záver 39
6. Použitá literatúra a nástroje 40
Príloha 40
CD 40

1. Úvod

Táto práca vznikla ako súčasť projektu Lomon, ktorý sa zaoberá zberom GPS dát z dopravy pomocou plávajúcich vozidiel. Plávajúce vozidlá sú vozidlá, ktoré jazdia v mestskej doprave a v pravidelných intervaloch zaznamenávajú svoju GPS pozíciu. Tieto namerané pozície sa odosielajú na server, kde sa uložia. Dáta sa následne vyhodnocujú a určuje sa z nich stav dopravy v meste.

Pre zber a odosielanie GPS dát boli zvolené zariadenia POSITRA GC 072 240F firmy LEVELNA. Cieľom mojej práce bolo zoznámiť sa s týmto komunikátorom a navrhnúť aplikáciu, ktorá bude so zariadeniami komunikovať, prijímať a ukladať dáta pre ich neskoršie spracovanie a navrhnúť webové rozhranie pre prezentáciu nameraných dát.

Medzi základné požiadavky mojej práce patrí minimalizácia nákladov na príjem užitočných dát, šifrované zasielanie dát a možnosť vzdialenej zmeny konfigurácie zariadení. Aplikácia komunikujúca so zariadeniami má byť spustiteľná ako na pozadí bežiaca služba na serveri s operačným systémom Microsoft Windows Server 2003 a dáta sa majú ukladať na databázový server MS SQL 2005.

Navrhnutá aplikácia slúži ako podklad pre nadväzujúcu diplomovú prácu Bc. Milana Janečka, ktorá sa zaoberá spracovaním a analýzou dát. V rámci jeho práce boli zároveň samostatne riešené problémy súvisiace so šifrovaním a zabezpečením dát medzi serverom a komunikátorom. Výsledkom sú navrhnuté metódy, ktoré využívam v tejto práci.

2. Popis komunikátora

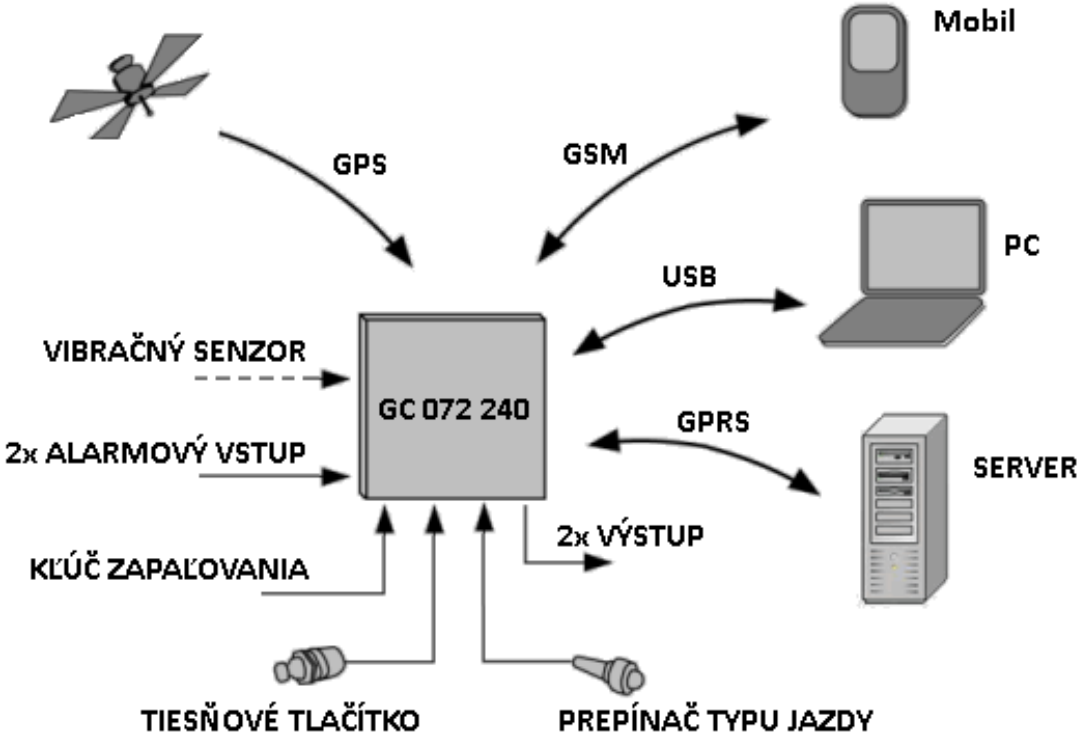
Obr. 2.1 Komunikátor POSITRA GC 072 240F[1]

2.1. Popis hardvéru

Komunikátor POSITRA GC 072 240F (obr. 2.1) je vybavený modulom GSM/GPRS, GPS modulom uBlox SIRF, 5 binárnymi vstupmi, vnútorným vibračným senzorom, 2 binárnymi výstupmi a rozhraním USB. Význam vstupov a výstupov zobrazuje obrázok 2.2. Napájacie napätie komunikátora je 10 – 15V.

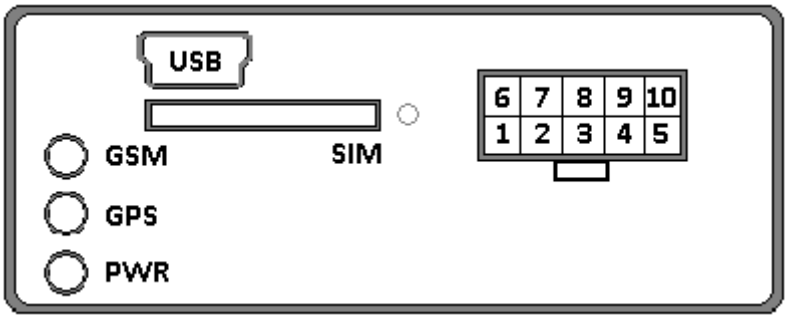
Vnútorná EEPROM pamäť o veľkosti 256 kB umožňuje zaznamenať 5 000 (10 000 pri kompresii) pozícií alebo 2000 udalosti.

[1] Obrázok prevzatý z dokumentu Manual GC072 240F verze 1.0 CZ.pdf od firmy LEVELNA



Obr. 2.2 Vstupy a výstupy komunikátora

Po pripojení napájacieho napätia sa zariadenie v priebehu 30 sekúnd uvedie do činnosti. Zariadenie pracuje samostatne bez nutnosti pripojenia ďalších vstupov/výstupov. GSM/GPRS modul sa uvedie do činnosti až po vložení SIM karty. SIM karta musí mať dostatočný kredit, vymazané SMS a aktivované dátové služby GPRS. Na prednom paneli sa nachádzajú stavové LED diódy, Mini USB konektor, priestor karty SIM a konektor, ktorého význam pinov je popísaný v tabuľke 2-1.



Obr. 2.3 Predný panel komunikátora

PIN 1	Vstup z kľúča od zapaľovania +12 V
PIN 2	Vstup tlačidla tiesňového volania. Reaguje na spojenie s +12 V
PIN 3	Alarmový vstup č. 1. Reaguje na spojenie s +12 V
PIN 4	Alarmový vstup č. 2. Reaguje na spojení so zemou
PIN 5	Zem
PIN 6	Napájanie +12 V
PIN 7	Záložné napájanie +12 V (zariadenie nedobíja)
PIN 8	Prepínač typu jazdy. Rozpojené = služobná jazda Spojené so zemou = súkromná jazda
PIN 9	Výstup OUT1
PIN 10	Výstup OUT2

Tabuľka 2-1 Zapojenie konektora

2.2. Spôsoby konfigurácie komunikátora

Konfiguráciu zariadenia je možné meniť prostredníctvom:

- konfiguračných SMS príkazov
- prostredníctvom PC cez USB rozhranie
- vzdialeným zaslaním konfigurácie prostredníctvom Internetu (GPRS)

2.2.1. SMS konfigurácia

Konfigurácia a obsluha zariadenia sa vykonáva prostredníctvom SMS správ odoslaných na telefónne číslo SIM karty umiestenej v zariadení. Do jednej SMS správy je možné vložiť až 4 príkazy s parametrami. Komunikátor informuje spätnou SMS

správou o vykonaní požadovaných povelov. Zariadenie nerozlišuje malé a veľké písmena.

Formát SMS správy:

HESLO PRÍKAZ1 PARAMETER1 ... PRÍKAZ4 PARAMETER4

HESLO heslo do zariadenia, maximálne 8 znakov A-Z a 0-9.

PRÍKAZ názov príkazu pre požadovanú konfiguráciu

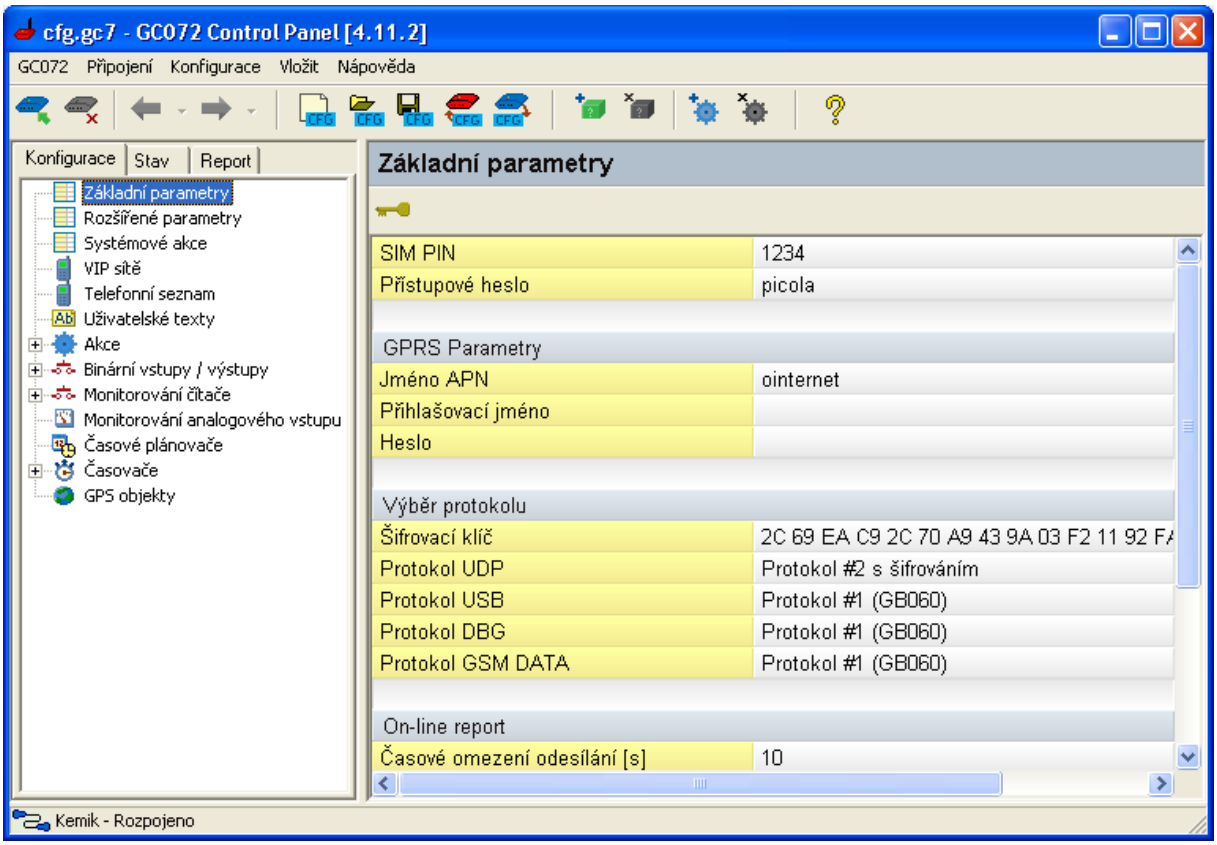
PARAMETER hodnota, ? pre dotaz, alebo . (bodka) pre vymazanie hodnoty

2.2.2. USB konfigurácia

Pred pripojením komunikátora k PC prostredníctvom USB rozhrania je nutné nainštalovať dodávaný ovládač pre operačný systém Microsoft Windows Xp spustením súboru "Driver GC 072 Win NT XP.exe". Po úspešnom nainštalovaní a pripojení komunikátora k PC systém Windows automaticky rozpozná pripojené zariadenie a nainštaluje ho do systému ako Level USB GSM Modem 1.1.

Následná konfigurácia komunikátora sa vykonáva prostredníctvom aplikácie GC 072 Control Panel. Pre komunikáciu so zariadením je nutné najskôr zistiť, cez ktorý sériový port je komunikátor pripojený k PC a nastaviť ho v aplikácii.

Aplikácia umožňuje nielen upravovať konfiguráciu komunikátora, ale aj sledovať jeho stav a obsah reportu v pamäti. Nastavenú konfiguráciu je možné uložiť do súboru s príponou *.gc7.



Obr. 2.4 GC 072 Control Panel

2.2.3. Zaslanie konfigurácie prostredníctvom GPRS

Pre online odoslanie konfigurácie zariadeniu je nutné mať k dispozícii konfiguráciu vo formáte súboru *.gc7. Tento spôsob konfigurácie neumožňuje meniť položky jednotlivo, a preto sa pri každej zmene konfigurácie musí zaslať úplná konfigurácia. Je výhodný v prípade, ak nie je možné pripojiť zariadenie k PC prostredníctvom USB rozhrania a potrebujeme meniť nastavenia, ktoré nie je možné zmeniť prostredníctvom SMS konfigurácie alebo by bola SMS konfigurácia veľmi zdĺhavá a nákladná.

2.2.4. Základná konfigurácia zariadenia

Na začiatku mojej práce som mal k dispozícii iba dve možnosti konfigurácie zariadenia. SMS a USB konfiguráciu. Overil som, že SMS konfigurácia skutočne

funguje, ale z užívateľského hľadiska je veľmi nepohodlná a nákladná. Preto som tento spôsob už viac nepoužíval.

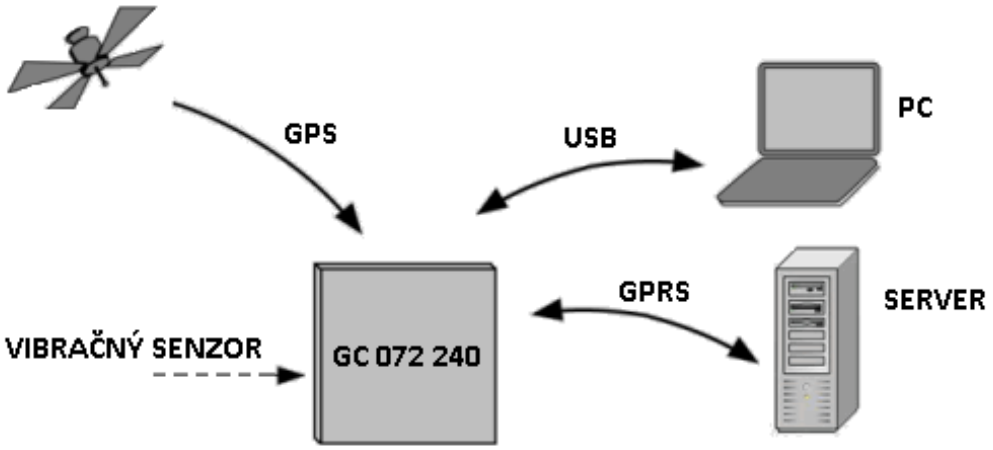
Oproti tomu USB konfigurácia prostredníctvom aplikácie GC 072 Control Panel je veľmi rýchla a prehľadná. Pomocou tejto aplikácie som vytvoril základný konfiguračný súbor, ktorý obsahuje základné nastavenia zariadenia používané pri práci. K dispozícii som mal SIM kartu s predplateným kreditom mobilného operátora Vodafone, preto GPRS nastavenia zodpovedajú nastaveniam tohto operátora.

Základná konfigurácia obsahuje nastavenia:

- PIN kód použitej SIM karty.
- Prístupové heslo do zariadenia.
- GPRS parametre (meno APN, prihlasovacie meno, heslo).
- Použité protokoly pre komunikáciu prostredníctvom UDP a USB protokolu. V súlade s požiadavkami som použil šifrovaný protokol #2 pre komunikáciu s UDP serverom.
- Šifrovací 16 bajtový kľúč používaný šifrovaným protokolom #2.
- Zisťovanie aktuálneho stavu kreditu prostredníctvom služby USSD
- IP adresu a port UDP servera, s ktorým bude zariadenie komunikovať

2.3. Používané funkcie zariadenia

Pre účely tejto práce nebolo nutné zapájať žiadne externé vstupy a výstupy zariadenia. Využíval som iba vstavaný GPS a GSM modul, vstavaný vibračný senzor a USB rozhranie pre účely konfigurácie (obr. 2.5).



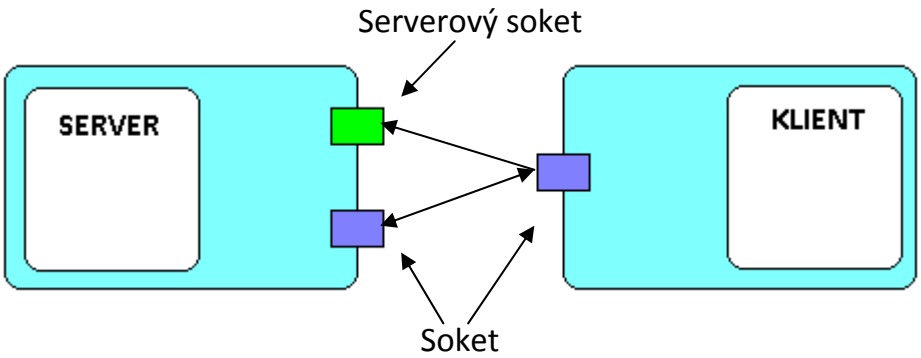
Obr. 2.5 Používané vstupy a výstupy pri práci

3. Serverová služba

3.1. UDP server

Zariadenie využíva na prenos dát v sieti Internet UDP (User Datagram Protocol) protokol. UDP protokol prenáša datagramy medzi zariadeniami v sieti, ale na rozdiel od TCP protokolu nezaručuje, že prenášaný paket sa nestratí, že sa nezmení poradie paketov, ani že sa niektorý paket nedoručí viackrát. Výhodou sú však nižšie režijné náklady na prenos dát a vyššia rýchlosť.

Pre účely zoznámenia sa a testovania zariadenia GC 072 240F som navrhol jednoduchý UDP server ako konzolovú aplikáciu v jazyku C#. Tento jazyk bol zvolený preto, aby neskôr mohla aplikácia bežať ako služba na serveri s operačným systémom Microsoft Windows Server 2003. Úlohou UDP servera je vytvoriť serverový soket na zvolenom porte, na ktorom čaká na pripojenie niektorého zo zariadení. Po pripojení zariadenia sa vytvorí nový (dočasný) soket, na ktorom server komunikuje so zariadením, takže serverový soket môže opäť čakať na ďalšie spojenie. Tento princíp ilustruje obrázok 3.1.



Obr. 3.1 Komunikácia zariadenia so serverom

Server používa synchrónny serverový soket, ktorý sa vyznačuje tým, že v čase prijímania dát je pozastavené vykonávanie celého aktuálneho vlákna. Nie je vhodný pre aplikácie s veľkou náročnosťou na dátový prenos, avšak pre moju úlohu je dostačujúci. Objektový návrh aplikácie umožňuje veľmi rýchlu náhradu asynchronným serverovým soketom v prípade nutnosti.

Na začiatku mojej práce fungoval tento server len ako jednovláknová aplikácia, ktorá prijaté dáta vypisovala na konzolu, prípadne ukladala do súboru. Primárnym cieľom bolo oboznámenie sa s komunikátorom a jeho konfiguráciou.

3.1.1. Príjem prvých dát a oživenie komunikácie so zariadením

Pre správnu funkciu aplikácie je nutné zabezpečiť, aby prijímajúci UDP server mal verejnú IP adresu a správne nakonfigurovať zariadenie tak, aby bolo schopné zasielať dáta prostredníctvom GPRS na server.

Pre účely testovania príjmu dát na server som nastavil komunikátor tak, aby mi v krátkych časových intervaloch zasielal svoju aktuálnu GPS polohu. V prvých prijatých dátach som overil, či sa ich formát zhoduje s formátom udávaným výrobcom, konkrétne minimálnu dĺžku prijatej správy a či obsahuje identifikačné číslo zariadenia.

Pri overovaní prijatých dát som zároveň zistil, že jedno a to isté zariadenie má približne v minútových časových intervaloch pridelený nový UDP port pri nezmenenej

IP adrese. Túto skutočnosť dokladá tabuľka 3-1 obsahujúca krátky záznam prijatých pripojení.

Čas	IP	Port
13:22:53	217.77.165.37	39715
13:34:17	217.77.165.37	10327
14:00:15	217.77.165.37	40049
14:00:28	217.77.165.37	40049
14:00:48	217.77.165.37	40049
14:01:50	217.77.165.37	40057
14:04:16	217.77.165.37	40082
14:19:16	217.77.165.37	10820

Tabuľka 3-1

Táto neustála zmena portu zariadenia znemožňuje kontaktovať zariadenie v ľubovoľnom čase prostredníctvom internetu. Komunikácia smerom k zariadeniu je preto možná iba v krátkom čase bezprostredne po pripojení zariadenia na server.

3.1.2. Efektivita prijímania dát

Dôležitým cieľom tejto práce je získať GPS dáta zo zariadenia za čo najnižšiu cenu bez nutnosti pripojenia k PC prostredníctvom USB káblu, to znamená použitím GSM/GPRS modulu. Pre dosiahnutie tohto cieľa bolo nutné stanoviť, ktoré dáta sa budú posielať, ich formát a zároveň nájsť najefektívnejší spôsob ich odosielania.

Z veľkého množstva dát, ktoré je možné nastaviť, aby zariadenie zaznamenávalo, som zvolil záznam iba 3 základných údajov. Zemepisnú šírku, zemepisnú dĺžku, čas merania tejto polohy. Tieto dáta komunikátor dokáže zaznamenať vo viacerých formátoch. Zvolil som ten najkratší možný formát, v ktorom tieto dáta zaberajú dohromady 8 bajtov.

Zaznamenané dáta je možné okamžite odoslať na server alebo uložiť do reportu do cyklickej vnútornej pamäte a odoslať naraz v nastavenom časovom intervale.

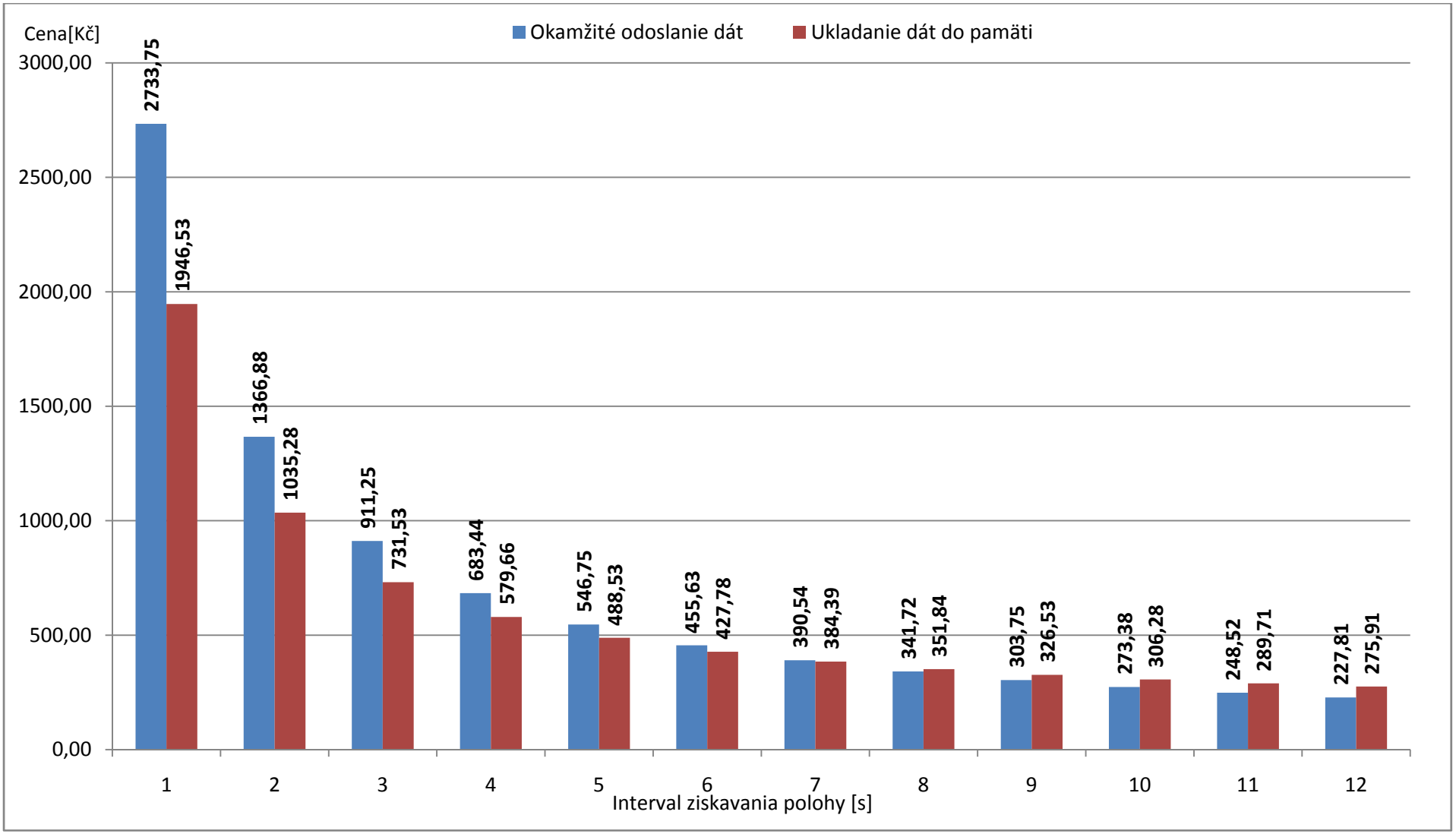
Pri okamžitom odosielaní dát má správa vždy konštantnú dĺžku 18 bajtov. Správu tvorí hlavička a dátová časť správy.

Správa odosielaná z vnútornej pamäte má rôznu dĺžku v závislosti na obsahu reportu. Opäť sa skladá z hlavičky a dátovej časti. Do dátovej časti sa postupne vkladajú jednotlivé záznamy reportu spolu s obslužnými bajtmi, ktoré umožňujú rozlíšiť jednotlivé záznamy. Do tohto reportu sa ukladajú aj krátke systémové správy, ktoré sa odosielajú naraz so všetkými dátami. V prípade, že report obsahuje veľké množstvo dát, dáta sa rozdelia do viacerých správ.

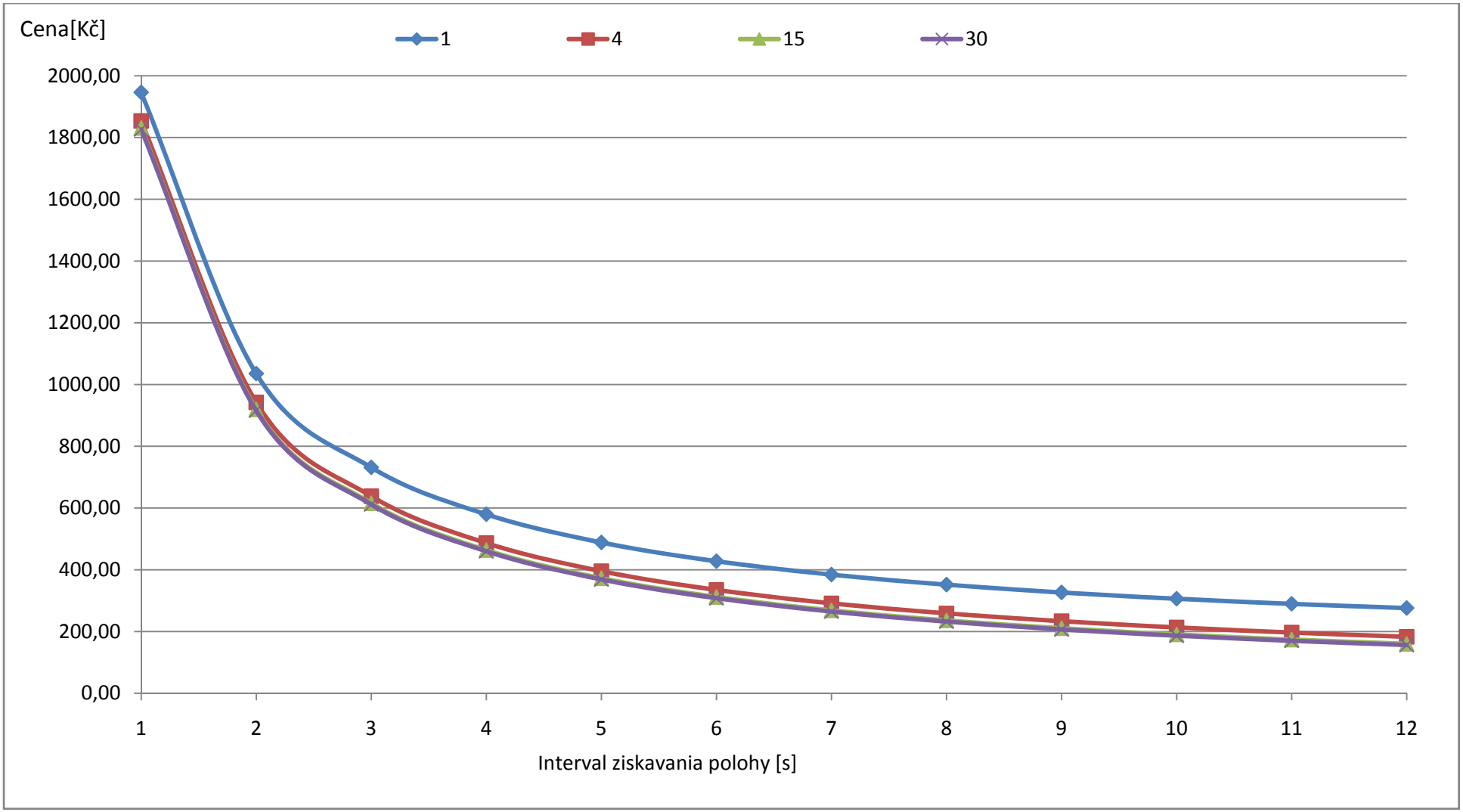
Názorné porovnanie týchto dvoch spôsobov zobrazuje graf 1, v ktorom som na vodorovnú os vyniesol interval zberu a zároveň aj okamžitého odoslania dát a na zvislú os cenu za prenesené dáta pre daný spôsob odosielania. Množstvo prenesených dát je vypočítané pre 24 hodinový nepretržitý prenos po dobu 30 dní, hromadné odosielanie dát z pamäti je každých 15 minút. Získané množstvo dát som násobil cenou 0,06 Kč/kB, čo je aktuálna priemerná cena mobilných operátorov. Do porovnania som zahrnul aj priemernú veľkosť systémových správ komunikátora.

Z grafu 1 vyplýva, že pri intervale zberu dát do 7 sekúnd je efektívnejší spôsob hromadného odosielania dát formou reportu. V intervale väčšom ako 7 sekúnd je však výhodnejšie dáta okamžite odosielať. V praxi je však zariadenie nastavené tak, aby nepoužívalo pravidelný zber dát, ale dynamický v závislosti na rýchlosti a vzdialenosti od predošlej meranej pozície.

Graf 2 porovnáva intervaly hromadného odosielania dát. Na vodorovnej osi je interval zberu dát a na zvislej osi cena za prenesené dáta. Jednotlivé priebehy predstavujú cenu hromadného odoslania dát v danom minútovom intervale. Z grafu je vidieť, že ak zvolím interval hromadného odosielania väčší ako 4 minúty, cena odoslaných dát sa takmer nemení. Je to spôsobené tým, že veľkosť hlavičky správy reportu je zanedbateľne malá oproti dátovému obsahu správy.



Graf 1 Porovnanie spôsobu odosielania dát



Graf 2 Porovnanie intervalov hromadného odosielania

3.2. Návrh používaných funkcií

3.2.1. Konfigurácia aplikácie a používané konštanty

Aby sa nemusel server kvôli zmene konfigurácie neustále reštartovať, je možné aplikácii za behu meniť niektoré konfiguračné parametre prostredníctvom vytvoreného konfiguračného XML súboru. Táto konfigurácia sa v pravidelných intervaloch načíta. V prípade, že by nejaký konfiguračný parameter chýbal v XML súbore, použije sa namiesto neho prednastavená hodnota uložená v triede Constants.

O preklad konfiguračného súboru a uloženie nastavených hodnôt sa stará trieda Config. Konfiguračné premenné sú prístupné ako verejné statické inštančné premenné.

Dáta získané zo zariadenia obsahujú okrem dátovej časti aj číselné konštanty, ktoré identifikujú napr. typ správy. Pretože pre programátora nie je v ľudských silách pamätať si význam jednotlivých číselných hodnôt, vytvoril som triedu Constants, v ktorej sú tieto konštanty uložené ako statické premenné s názvom, ktorý vypovedá o ich obsahu. Okrem konštánt zariadenia sú tu uložené aj počiatočné nastavenia konfiguračných parametrov aplikácie napr. číslo portu, použitá IP adresa , časové oneskorenia, názov konfiguračného súboru a pod.

3.2.2. Práca so súbormi

O načítanie a ukládanie súborov sa stará trieda SaveFile. Statické metódy tejto triedy sú volané napr. z triedy dbApi po zlyhaní komunikácie s databázou alebo sa používajú pre načítanie konfiguračného súboru zariadenia. Všetky vytvárané súbory majú do názvu pridávanú predponu aktuálneho dátumu podľa stredoeurópskeho času, aby bola zjednodušená orientácia v súboroch a zároveň ich obsah rozdelený do menších súborov. Názvy súborov sú prednastavené v konštantách pre prípad, že by neboli definované v konfiguračnom XML súbore.

3.2.3. Metódy pre komunikáciu so zariadením

Testovací jednovláknový UDP server slúžil ako základný nástroj pre zoznámenie sa s komunikačným protokolom používaným komunikátorom. Bolo nutné vytvoriť obslužné funkcie, ktoré sa starajú o preklad prijatých a odosielaných dát. Táto úloha už je mimo rámec mojej práce a bola riešená samostatne v rámci diplomovej práce Bc. Milana Janečka. Naznačím však dôležité body, ktoré sa museli vyriešiť v týchto triedach:

- Dešifrovanie prijatej správy
- Overenie správnosti dát kontrolným súčtom CRC
- Preloženie dát podľa protokolu definovaným výrobcom
- Vytvorenie odosielanej správy zariadeniu
- Vytvorenie kontrolného súčtu CRC pre odosielanú správu zariadeniu
- Zašifrovanie odosielanej správy

Z vývojového hľadiska sú najstaršie funkcie pre preklad prijatých dát uložených v triede GPSData. Tejto triede sa do konštruktoru vložia prijaté dáta a príslušný šifrovací kľúč, podľa ktorého sa dáta majú dešifrovať. V prvom kroku sa dáta dešifrujú a overí sa ich platnosť pomocou kontrolného súčtu CRC. Ak sú dáta platné, preložia sa v závislosti na ich type definovanom v protokole výrobcu a uložia sa do lokálnych premenných triedy. Tieto hodnoty sú následne prístupné pomocou verejných triednych metód.

V triede SendData sú uložené funkcie generujúce dáta posielané zariadeniu. Táto trieda je úzko spojená s triedou GPSData, pretože napríklad v prípade žiadosti o vymazanie časti reportu je nutné zadať rozsah dát na zmazanie, ktorý je možný určiť len z prijatých dát. Preto je trieda SendData predkom triedy GPSData s deklarovanými chránenými premennými, ktoré následne dedí trieda GPSData, aby do nich mohla zapisovať.

Funkcie starajúce sa o výpočet a kontrolu kontrolného súčtu CRC sú uložené v triede Crc16Ccitt.

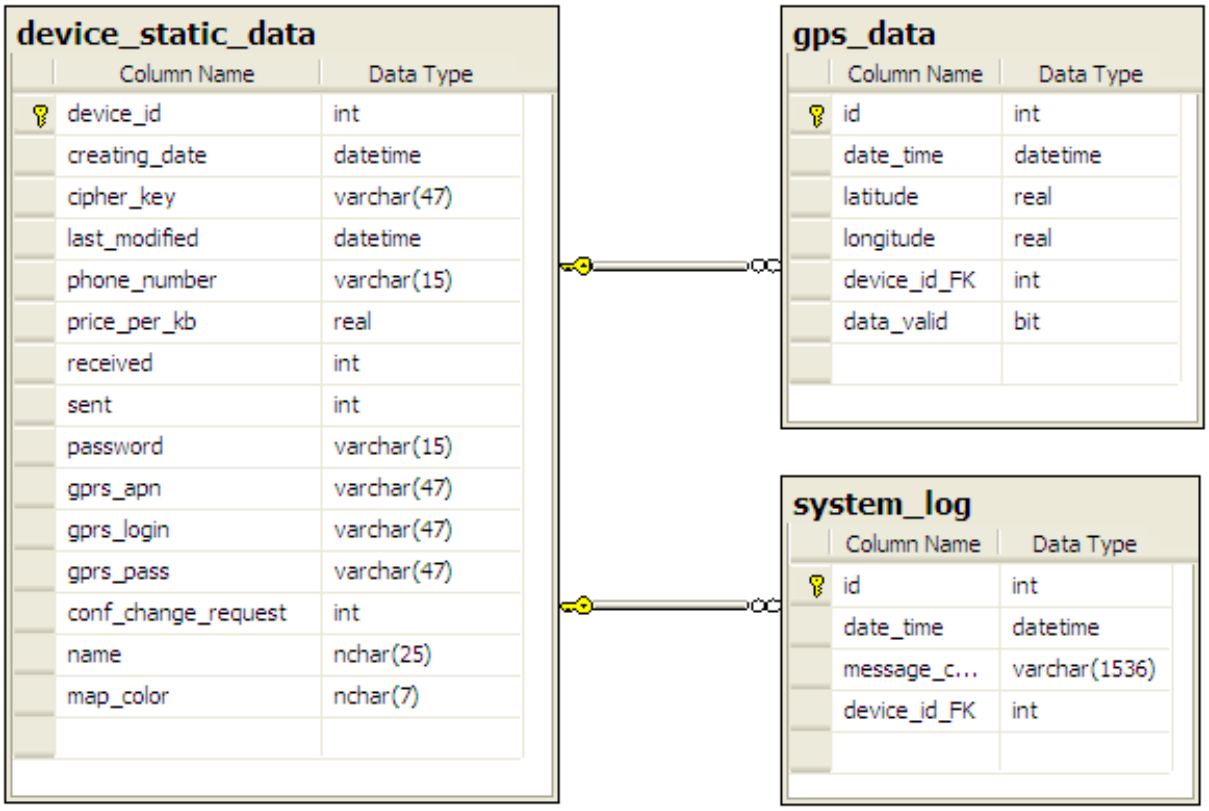
3.3. Návrh databázy

Návrh databázy vychádza z potreby uchovávania nastavení pre jednotlivé zariadenia a od nich prijatých dát. Ako databázový server je použitý MS SQL Server 2005. Časť ER-diagramu navrhnutej databázy zobrazuje obrázok 3.2.

V tabuľke device_static_data sa uchováva statická konfigurácia jednotlivých zariadení. Primárnym kľúčom je ID zariadenia, pretože každé zariadenie má od výrobcu pridelené jedinečné číslo ID. Statickú konfiguráciu tvorí šifrovací kľúč, GPRS nastavenia mobilného operátora a prístupové heslo do zariadenia. Okrem týchto údajov sa v tabuľke nachádzajú údaje používané pre ľahšiu užívateľskú identifikáciu zariadenia prostredníctvom webového rozhrania.

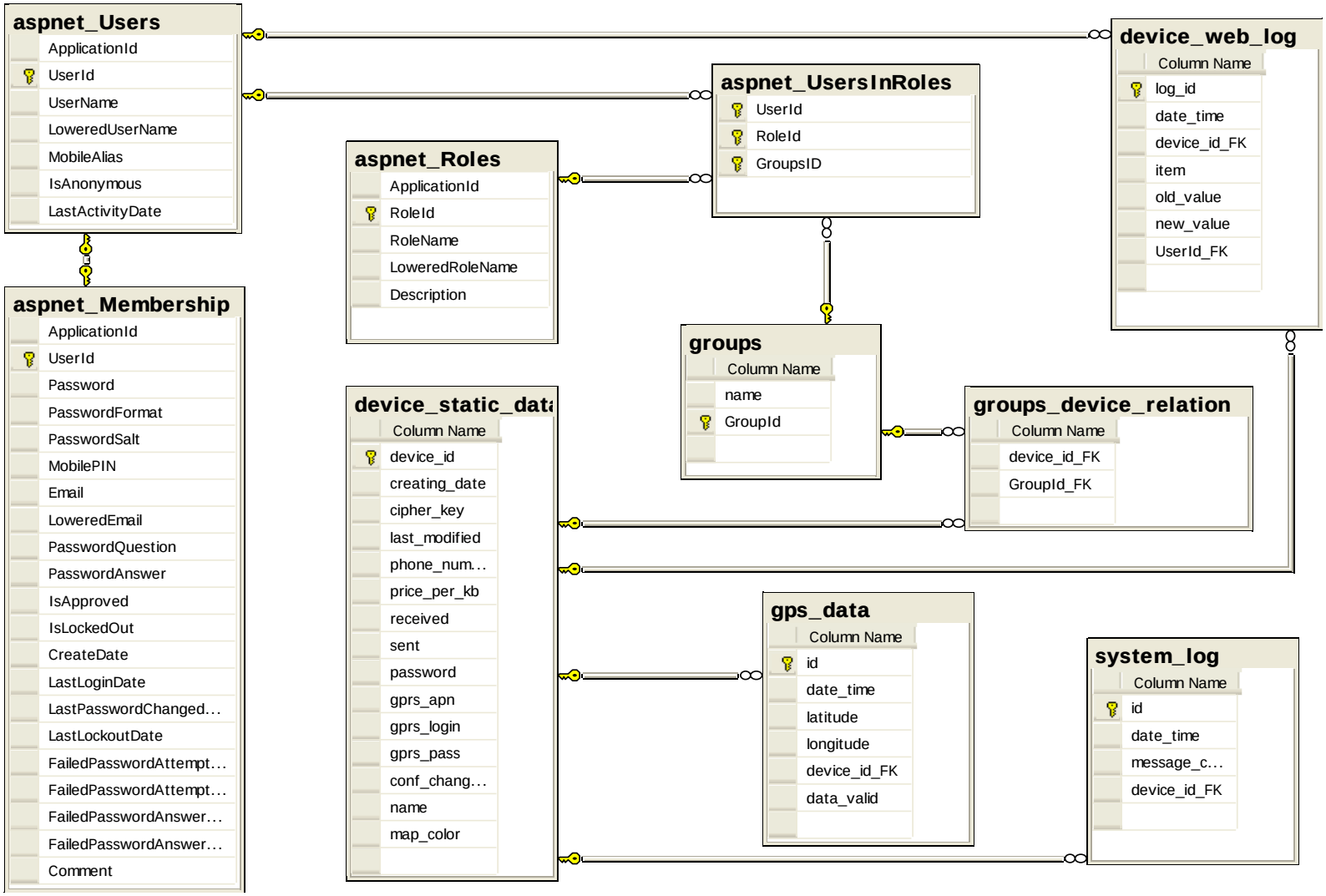
Do tabuľky gps_data sa ukladajú všetky GPS dáta získané od zariadení. GPS dáta obsahujú zemepisnú šírku, zemepisnú dĺžku a čas odmeru. Spolu s nimi sa zároveň ukladá aj ID zariadenia, ktoré dáta odoslalo a nastavuje sa bit, či ide o platné alebo neplatné dáta.

V tabuľke system_log sa ukladajú všetky systémové správy komunikátora, ktoré umožňujú analyzovanie jeho stavu a činnosti. Opäť sa tu ukladá ID zariadenia, čas záznamu a samotná systémová správa.



Obr. 3.2 Časť ER-diagramu databázy

ER-diagram z obrázku 3.2 je ešte rozšírený o ďalšie tabuľky. Pre samotný UDP server nemajú praktický význam. Používajú sa iba pre webové rozhranie. Ich štruktúra je automaticky vygenerovaná SQL skriptami dodanými spolu s vývojovým nástrojom Microsoft Visual Studio 2008. Slúžia pre správu užívateľských účtov a ich prístupových práv. Vygenerované tabuľky sú doplnené ešte o tabuľku device_web_log, groups a groups_device_relation. Tabuľka groups a groups_device_relation bude podľa návrhu slúžiť pre priradenie zariadení do skupín. Skupinu budú tvoriť užívatelia, ktorým patrí určitý počet zariadení. Zmeny nastavení zariadení sú zaznamenávané v tabuľke device_web_log, kde sa okrem zmenenej položky a času zmeny ukladá aj ID užívateľa, ktorý túto zmenu vykonal.



Obr. 3.3 Rozšírený ER-diagram databázy

3.3.1. Oddelenie databázovej vrstvy od aplikačnej

Aby bola aplikácia nezávislá na použitej databáze, vytvoril som triedu s názvom dbApi, ktorá oddeľuje databázovú vrstvu od aplikačnej. Táto trieda v sebe implementuje správu pripojenia k databáze a vykonáva samotné SQL dotazy do databázy. Aplikáciu teda nemusí zaujímať, čo sa deje v tejto vrstve, vie len formát dát, ktoré ma do nej poslať a aký formát dát z nej dostane. Výhodou teda je, že ak sa vymení databáza, nie je nutné nijak meniť samotný kód aplikácie, stačí iba prepísať túto triedu.

V prípade zlyhania zápisu dát do databázy trieda dbApi zabezpečuje uloženie dát do textového súboru, aby nemohlo dôjsť ku strate dát. Názov záložného súboru sa definuje v konfiguračnom súbore UDP servera. V dobe vývoja a testovania k tomuto zlyhaniu zápisu nikdy nedošlo, iba pri zámernom umelom vyvolaní tohto zlyhania.

Trieda dbApi je vytvorená podľa návrhového vzoru singleton, to znamená v celej aplikácii existuje jediná inštancia tejto triedy. Singleton sa vytvára tak, že sa v triede vytvorí privátny konštruktor a vnútorná premenná inštancie. Inštancia tejto triedy sa vráti ako návratová hodnota factory metódy, ktorá pri každom volaní skontroluje, či už bola inštancia vytvorená. Ak áno, vráti túto inštanciu. V prípade, že nebola vytvorená, vytvorí sa pomocou privátneho konštruktora nová inštancia a vráti sa ako návratová hodnota.

Pri každom volaní metód triedy dbApi sa vytvára vždy nové pripojenie do databázy. Tento spôsob pripájania do databázy je pomalší oproti trvalému pripojeniu, ale pre túto aplikáciu plne postačuje, pretože do databázy pristupuje iba pri prvom spustení samotnej aplikácie a potom už len pri ukladaní dát získaných zo zariadení a prípadne dopĺňaní chýbajúcich informácii pre chod aplikácie. V žiadnom prípade to nespomaľuje aplikáciu, pretože pre každý prístup do databázy je vytvorené samostatné vlákno.

3.3.2. MS SQL databázové procedúry

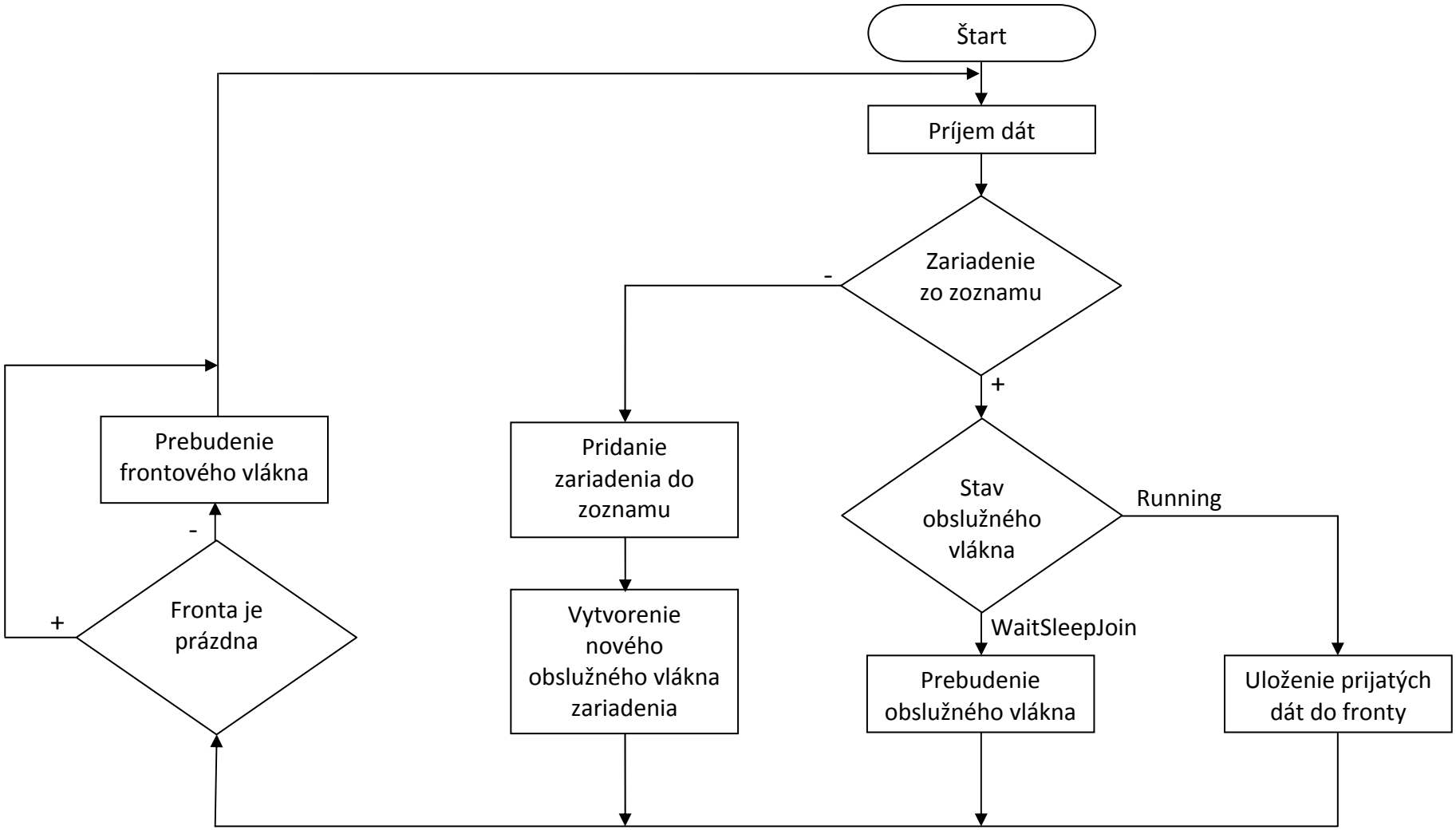
Microsoft SQL Server umožňuje vytvárať takzvané uložené databázové procedúry, ktoré zjednodušujú vývoj databázových aplikácii tým, že zoskupujú SQL dotazy do blokov, ktoré sa naraz v jednom volaní vykonajú. Tieto bloky SQL dotazov sú uložené priamo na strane databázového servera. Ich volanie vyžaduje iba zaslanie potrebných dát bez nutnosti zasielania celého SQL dotazu, čo výrazne znižuje objem prenesených dát medzi databázou a aplikáciou. Ďalšou výhodou je predkompilovanie týchto procedúr na strane databázového servera a následné rýchlejšie vykonanie po zavolaní.

3.4. Viacvláknový UDP server

Pretože testovací jednovláknový server umožňuje obsluhu iba jedného zariadenia v jednom svojom cykle, bolo nutné tento server rozšíriť do viacerých vlákien tak, aby aplikácia zvládla paralelne obslúžiť všetky pripojené zariadenia. Zjednodušený vývojový diagram hlavného vlákna je na obr. 3.4.

Po spustení aplikácie sa spustí hlavné vlákno, ktoré vytvorí serverový soket na sieťovom rozhraní s definovanou IP adresou a číslom portu definovanom v konfiguračnom súbore alebo v preddefinovaných konštantách. Na tomto serverovom sokete čaká na pripojenie zariadenia.

Po pripojení komunikátora na server a prijatí dát sa z nich určí ID zariadenia, pomocou ktorého sa overí, či už aktuálne pripojené zariadenie existuje v zozname posledných známych zariadení tvorenom hash tabuľkou. Tento zoznam obsahuje inštancie triedy DeviceThreadData. V tomto objekte sú vždy uložené prijaté dáta, dočasný soket, na ktorom sa so zariadením komunikuje, odkaz na vlákno, ktoré daný komunikátor obsluhuje, ID zariadenia, šifrovací kľúč platný v dobe prijatia dát a ďalšie dáta slúžiace pre obsluhu zariadenia.



Obr. 3.4 Vývojový diagram hlavného vlákna

V prípade, že sa pripojilo zariadenie, ktoré sa nenachádza v zozname, vytvorí sa a vyplní nová inštancia triedy DeviceThreadData a vytvorí sa nové obslužné vlákno, ktoré spracuje prijaté dáta.

V prípade, že sa pripojilo zariadenie, ktoré sa už nachádza v zozname, znamená to, že sa prijal väčší objem dát rozložený do viacerých správ alebo odpoveď na niektorú z požiadaviek zaslanú serverom. Tieto správy chodia za sebou vo veľmi krátkych intervaloch, preto sa môže stať, že obslužné vlákno ešte spracováva predošlú správu. Pre tento prípad sa testuje, v akom stave sa nachádza obslužné vlákno. Ak je v stave wait, sleep alebo join, znamená to, že obslužné vlákno je pripravené prijať ďalšiu správu, a preto sa mu pošle prijatá správa.

Ak je vlákno v aktívnom stave, nie je schopné spracovať aktuálne prijatú správu. Preto sa táto správa uloží do fronty správ. Spracovanie fronty správ sa spustí na konci každého cyklu hlavného vlákna v samostatnom vlákne, ak fronta obsahuje nejaké dáta. Spracovanie správ je pasívne, to znamená, že sa dáta iba preložia a uložia podľa ich typu do databázy. Týmto spôsobom je zaručené, že sa žiadne dáta nestratia.

Tento spôsob spracovávania bude nutné prepracovať, pretože sa vo fronte na obsah prijatých dát nijak nereaguje narozdiel od spracovávania obslužným vláknom. V dobe návrhu tohto spôsobu spracovania ešte nebol úplne správne pochopený význam jednotlivých typov služieb hlavne pre prípad rozdeleného väčšieho reportu.

3.4.1. Pomocné vlákno ConfigCron

Účelom pomocného vlákna je v pravidelných intervaloch načítať konfiguračný XML súbor. Interval načítania je definovaný konštantou s hodnotou 5 sekúnd. Zároveň vlákno kontroluje databázu, či niektorému zo zariadení nebola zmenená konfigurácia. Ak bola, zapíše sa to do hash tabuľky, ktorá obsahuje zoznam zariadení, ktorým sa pri najbližšom prihlásení zašle nová konfigurácia.

3.4.2. Inicializácia obslužného vlákna

Obslužné vlákno tvorí spolu s hlavným aplikačným vláknom jadro celého servera. V obslužnom vlákne sa spracovávajú takmer všetky správy prijaté serverom. Len malá časť sa spracuje vo frontovom vlákne.

Pri prvom vytvorení obslužného vlákna sa nainicializujú premenné a spustí sa nekonečná slučka, ktorá sa ukončuje až po obslúžení pripojeného komunikátora. Pri inicializácii premenných sa nastaví premenná requestedService (vyžadovaná služba) na definovanú hodnotu, ktorá označuje, že ide o prvé spustenie vlákna a neočakáva sa žiadna odpoveď komunikátora na správu odoslanú zariadeniu zo servera.

V každom cykle vlákna sa najskôr testuje, či sú dáta platné, to znamená, či sedí kontrolný súčet CRC. Zároveň sa testuje, či už nebol komunikátor úplne obslúžený a či existuje požiadavka na odoslanie novej konfigurácie pre zariadenie. Na začiatok predpokladajme, že ide o prvé spustenie a prijaté dáta sú platné. Z preložených dát sa určí, aký typ správy bol prijatý. Ak sa prijala nejaká systémová správa alebo neznáma správa, ktorej server nerozumie, uloží sa o tom záznam do databázy do tabuľky system_log. Na systémové správy server neodosiela žiadnu spätnú správu do zariadenia. V prípade správ s GPS súradnicami sa tieto dáta uložia do databázy a vyžiada sa ich zmazanie z pamäti komunikátora. Na každú žiadosť sa očakáva zo zariadenia odpoveď s definovaným typom služby. Hodnota tejto služby sa nastaví do premennej requestedService. Obslužné vlákno sa „uspí“ na dobu, pokým nepríde z rovnakého zariadenia ďalšia správa.

3.4.3. Opakované odoslanie správy

Pre každú odosielanú správu do pripojeného zariadenia sa vytvorí samostatné vlákno nazvané RepeatedSending. Jeho úlohou je v pravidelných intervaloch opakovane odosielať správu do komunikátora. Interval je definovaný prostredníctvom konfiguračného súboru. Maximálny počet opakovaní je opäť

definovaný v konfiguračnom súbore. V prípade, že príde odpoveď od zariadenia skôr než sa dosiahne tento maximálny počet opakovaní, je toto vlákno z obslužného vlákna zrušené. Ak sa dosiahne maximálny počet opakovaní, je prebudené obslužné vlákno tak, aby zrušilo RepeatedSending vlákno, zapíše sa správa o neúspešnom odoslaní správy a zavolá sa metóda na zrušenie samotného obslužného vlákna, pretože už nie je možné naďalej so zariadením komunikovať.

3.4.4. Spracovanie odpovede na vyžiadanú službu

Ak premenná requestedService má hodnotu, ktorá je rôzna od inicializačnej hodnoty, znamená to, že je očakávaná odpoveď na odoslanú požiadavku. V prípade, že je prijatá správa s iným typom služby, než sa očakáva, celá správa sa uloží do fronty pre neskoršie spracovanie frontovým vláknom. Ak správa obsahuje očakávanú službu, spracuje sa podobne ako pri prvom prihlásení zariadenia, čo znamená, že systémové správy sa uložia do databázy do tabuľky system_log, GPS súradnice sa uložia do tabuľky gps_data a vyšle sa žiadosť na ich zmazanie.

3.4.5. Ukončenie obslužného vlákna

O ukončenie obslužného vlákna sa stará metóda DestroyHandlingThread. Na začiatku tejto metódy sa overí, že či sa nepožaduje zmena konfigurácie zariadenia. Komunikátor je totiž obslúžený a je stále pripojený k serveru a ak je požadovaná zmena konfigurácie, obslužné vlákno sa neukončí. Nastaví sa premenná indikujúca, že v nasledujúcich cykloch obslužného vlákna sa bude odosielať vzdialená konfigurácia.

Ak nie je požiadavka na vzdialenú konfiguráciu, zo zoznamu známych zariadení tvoreného hash tabuľkou sa odoberie objekt vytvorený pre aktuálne pripojené zariadenie a ukončí sa jeho obslužné vlákno.

3.4.6. Vzdialená konfigurácia komunikátora

Vzdialená konfigurácia komunikátora neumožňuje zmenu jednotlivých nastavení samostatne, ale musí sa odoslať celá konfigurácia naraz. Formát

konfigurácie je rovnaký s formátom súboru generovaného v aplikácii GC Control Panel spoločnosti Levelna. Spôsob, ktorým je tento súbor vytváraný je firemným tajomstvom spoločnosti Levelna a nebol prezradený. Poskytla však protokol, ktorým sa odosiela konfigurácia do zariadenia. Tento protokol má tvar technickej správy bez vysvetliviek, preto hľadanie spôsobu odosielania vzdialenej konfigurácie do zariadenia bolo založené na systéme pokus-omyl. Tento protokol nie je bežne poskytovaný zákazníkom a bol som požiadaný, aby som ho nezverejňoval. Dokument tvorí súčasť projektovej dokumentácie, ktorá nie je práve z tohto dôvodu verejne prístupná.

Aj keď formát konfiguračného súboru nepoznám, viem, že sa skladá zo statickej a dynamickej časti. Dynamickú časť nie som schopný programovo meniť, pretože jej veľkosť a aj obsah sa pri každej zmene konfigurácie mení. Statická časť je vždy pevná a obsahuje informácie ako PIN kód SIM karty, GPRS nastavenia mobilného operátora, prístupové heslo do zariadenia, šifrovací kľúč a ďalšie. Pretože má táto konfigurácia stále pevnú veľkosť, je možné ju programovo meniť a práve to je využívané pre vytváranie a zasielanie konfigurácie zariadeniam.

3.4.7. Postup odosielania vzdialenej konfigurácie

Ak existuje požiadavka na zmenu konfigurácie, načíta sa súbor so spoločnou konfiguráciou a upraví sa statická časť tejto konfigurácie v závislosti na parametroch uložených v databáze. V databáze sú uložené špecifické parametre pre prihlásené zariadenie spolu so zmenenými parametrami. Takto upravená konfigurácia sa uloží do súboru s názvom obsahujúcim ID zariadenia, aby bolo možné túto konfiguráciu napr. analyzovať v prípade problémov.

Keď už je špecifická konfigurácia pripravená, musí sa konfigurácia rozdeliť do blokov, pretože konfigurácia je väčšia než najväčší povolený prenášaný blok dát prenášaný zariadením. K jednotlivým blokom sa pridávajú obslužné bajty dát a následne sa šifrujú. O prípravu dát sa starajú metódy v triede SendData.

Takto pripravené bloky dát sa posielajú zariadeniu, ktoré po prijatí každého bloku vráti odpoveď s rovnakým obsahom dát, aby bolo možné overiť, že sa konfigurácia odoslala správne. V prípade, že sa počas posielania konfigurácie preruší spojenie a neodošle sa celá konfigurácia, zariadenie prestane komunikovať. V tomto prípade je nutný hardvérový reštart. Na serveri je kvôli tomuto prípadu zlyhania nastavené, aby sa po zlyhaní odosielala správa o ukončení odosielania konfigurácie. Zariadenie totiž po získaní tejto správy zloží prijaté bloky konfigurácie a overí, či má platný formát. Ak nemá, používa naďalej starú konfiguráciu, v prípade platnej konfigurácie sa nahrá nová konfigurácia a zariadenie sa reštartuje.

4. Webové rozhranie

Cieľom webového rozhrania je sprístupnenie a možnosť prezerania nameraných dát, ich prezentácia na mape a jednoduchá správa komunikátorov. Zároveň slúži ako stručná prezentácia projektu LOMON, pretože spustiteľný exe súbor UDP servera, prípadne holé zariadenie nezainteresovanému divákovi nič nepovie. Pretože je prostredníctvom tohto rozhrania možné meniť aj konfiguráciu zariadení, je prístup k týmto stránkam chránený užívateľským menom a heslom. Rozhranie je vyvinuté v ASP.NET frameworku v jazyku C# a spolupracuje so spoločnou MS SQL databázou pre UDP server.

4.1. Rozloženie stránky

Na obr. 4.1 je úvodná stránka webového rozhrania. Dizajn stránky pochádza z voľne dostupnej webovej šablóny, ktorú som upravil tak, aby zodpovedala normám W3C a zároveň vyhovovala potrebám projektu. V hornej časti stránky sa nachádza hlavné menu spolu s možnosťou zmeny jazykovej verzie stránky. Hlavné menu sa dynamicky kopíruje aj do päty stránky, pretože využívajú spoločný znovu

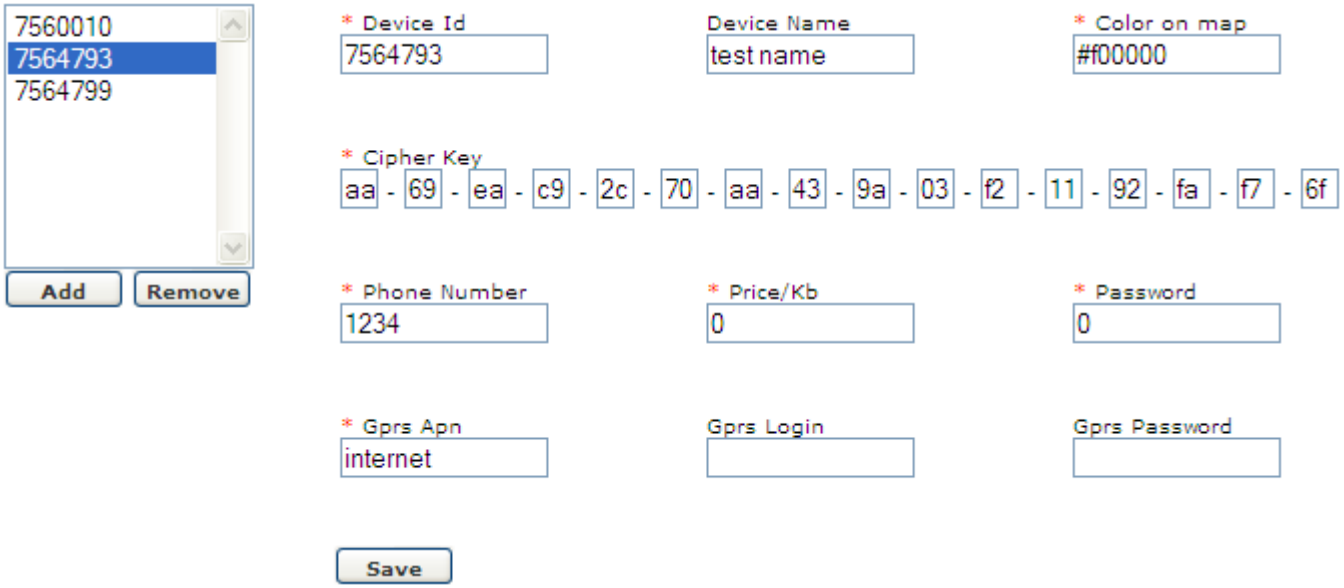
použiteľný fragment. V ľavej časti stránky pod hlavným menu je dynamicky sa meniace menu a odkazy na súvisiace stránky. Pravá časť tvorí hlavnú časť stránky.

Pre jednotný vzhľad stránky a zároveň, aby nedochádzalo k opakovaniu rovnakých časti kódu, sa používa jedna hlavná stránka - Master Page. Je tvorená horným hlavným, ľavým a dolným menu, pričom obsah pravej časti sa dynamicky mení.

4.2. Prihlásenie

ASP.NET framework v sebe obsahuje ovládacie prvky pre registráciu, prihlasovanie a overenie identity užívateľa. Aby ich bolo možné používať, musí sa vytvoriť v databáze potrebná štruktúra tabuliek z dodaného SQL skriptu. Pôvodný návrh počítal aj so správou prístupových práv a užívateľských rolí užívateľov. Vzhľadom na náročnosť implementácie som sa rozhodol od tohto zámeru upustiť. Vyžadovalo by to prepísanie implementácie väčšiny automaticky generovaných metód slúžiacich pre správu užívateľov. Užívatelia majú teraz rovnaké prístupové práva a ich správa nie je umožnená prostredníctvom webového rozhrania.

Obr. 4.1 Rozloženie webovej stránky



4.2 Správa zariadení

4.3. Správa zariadení

Stránka so správou zariadení na obrázku 4.2 umožňuje pridávať nové zariadenia a upravovať existujúce. Mazanie existujúcich zariadení je taktiež implementované, ale nie je úplne vyriešená synchronizácia s UDP serverom.

Pre každé zariadenie sa nastavuje číslo zariadenia, voliteľný názov zariadenia, hexadecimálny kód farby, ktorou sa zobrazí zariadenie na mape, šifrovací kľúč, GPRS nastavenia mobilného operátora, cena za 1KB prenesených dát, prístupové heslo do zariadenia.

Vstupné polia sa opakujú, preto som vytvoril fragment pre jedno vstupné pole, ktorému je možné nastaviť nadpis, prednastavenú hodnotu a používanie validácii. Rovnako pre šifrovací kľúč som vytvoril samostatný fragment s validáciami, ktorého výstupom je textový reťazec so šifrovacím kľúčom v definovanom formáte.

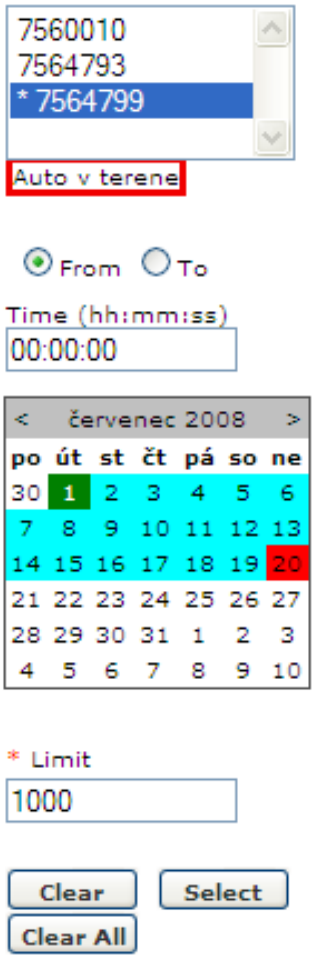
4.4. Zobrazenie prijatých dát na mape

Uložené dáta v databáze sa zobrazujú na voľne dostupnej mape Google Maps prostredníctvom Google Map Api. Na mape sa zobrazujú iba namerané body s príznakom platných dát. Dáta spracúvajú prostredníctvom JavaScriptu na strane klienta čo znamená, že so stúpajúcim počtom zobrazovaných bodov na mape rastú aj nároky na výkon počítača u klienta. Zistil som, že ak je počet bodov na mape vyšší ako hodnota približne 1000, je práca s mapou veľmi pomalá a na výkon procesora náročná. Rovnako som zistil, že pre prácu s mapou Google Maps je z hľadiska rýchlosti najvhodnejší webový prehliadač Mozilla Firefox.

4.5. Výber zobrazovaných dát

Pretože v praxi nechceme zobrazovať všetky dostupné dáta v databáze, ale iba ich časť, navrhol som ovládací prvok na obr. 4.3, ktorým je možné vymedziť rozsah zobrazovaných dát. V hornej časti je výber zariadenia, pre ktoré sa majú nastaviť parametre výberu. Zoznam obsahuje identifikačné čísla dostupných

zariadení. Po označení zariadenia sa pod zoznamom zobrazí názov zariadenia spolu s definovanou farbou, ktorou sa zariadenie zobrazí na mape. Voliteľným nastavením je filter rozsahu dátumu a času nameraných dát. Je možné ho nechať nevyplnený, alebo nastaviť iba počiatočný dátum od, alebo iba koncový dátum do, prípadne nastaviť rozsah od do, kde je možné zvoliť ako najmenší časový rozdiel 1 sekundu. Jediným povinným parametrom je limit, jeho rozsah hodnôt je od 1 do 1000. Význam parametru bude popísaný neskôr. Všetky zmeny v parametroch sa ukladajú do globálnej premennej prostredníctvom AJAXu bežiacom na pozadí. Výhoda globálnej premennej je tá, že prepínaním medzi jednotlivými zariadeniami sa jednotlivé nastavenia nestratia. Preto je možné mať súčasne vybrané viac než jedno zariadenie a každé má samostatné nastavenie.



Obr. 4.3 Spoločný fragment pre výber dát

4.6. Zobrazenie vybraných dát

Pri každom pohybe s mapou sa spustí na pozadí AJAX požiadavka, ktorá zavolá na pozadí stránku GetGPSData, ktorá získa dáta z databázy podľa zvolených užívateľových nastavení. O výber správnych dát sa stará databázová procedúra, do ktorej sa tieto nastavenia odosielajú. Okrem zvolených užívateľových nastavení sa do databázy zasielajú aj súradnice aktuálneho zobrazeného okna mapy, to znamená súradnice ľavého horného a pravého dolného rohu. Definovaním okna sa zúži výber dát na aktuálne okno a tým aj zníži množstvo prenášaných dát. Aby po posunutí mapou nebolo vidieť koniec predošlých dát, súradnice rohov sú 1,5-násobne zväčšené na všetky strany. Týmto sa dosahuje akési predspracovanie okolia.

4.6.1. Výber dát na strane databázového servera

V databáze sa z dát vyhovujúcich zadaným kritériám vytvorí dočasná tabuľka. Z nej sa následne cyklicky vyhodia body, ktoré by sa pri zvolenom priblížení mapy zobrazili ako jeden bod. Pri každom priblížení mapy viem určiť minimálnu vzdialenosť bodov, ktorú musia mať, aby sa nezobrazili ako jeden bod. Pretože sú GPS súradnice definované v sférických súradniciach, vzťah pre výpočet vzdialenosti 2 bodov je plný goniometrických funkcií, čo spôsobuje veľkú časovú náročnosť na výpočet. Preto sa pred použitím tohto algoritmu najskôr vyhodí každý n-tý záznam z dočasnej tabuľky tak, aby tabuľka obsahovala iba trikrát väčší počet bodov než je požadovaný nastavený limit. Parameter limit predstavuje maximálny počet zobrazených záznamov v okne mapy. Po odstránení bodov, ktoré by sa zobrazili na mape ako jediný bod sa spočíta počet zvyšných bodov. V prípade, že je vyšší než nastavený limit, zopakuje sa opäť rovnaký algoritmus, pri ktorom sa vyhodí každý n-tý bod tak, aby výsledok obsahoval najviac toľko bodov, koľko je nastavený limit.

4.6.2. Spracovanie vrátených dát

Návratová hodnota AJAX požiadavky je textový reťazec, ktorý obsahuje vždy identifikačné číslo zariadenia, farbu, ktorou sa majú body zariadenia vykresliť

a súradnice bodov s ich ID záznamu, ktoré patria k danému zariadeniu. Tento reťazec sa následne prostredníctvom JavaScriptu na strane klienta spracováva. Najskôr sa dáta rozdelia do blokov podľa ID zariadenia, v každom z týchto blokov sa získa farba, ktorou sa dané zariadenie vykresľuje. Následne sa v každom bloku spracujú a uložia súradnice bodov do poľa, ktorého index tvorí ich ID záznamu súradnice. Tým je zaručené, že sa ten istý bod nebude v poli opakovať viackrát.

4.6.3. Vykreslenie dát

Po spracovaní prijatých dát nasleduje ich vykreslenie. Najskôr sa z predošlých prijatých dát vyhodia tie, ktoré nepatria do okna mapy zväčšeného 1,5 násobne do všetkých strán. K polohám, ktoré ostali sa pridajú nové prijaté dáta. Pre každé ID zariadenia je tento proces vykonaný samostatne.

Prvý návrh vykresľovania dát pracoval s vykresľovaním jednotlivých bodov samostatne. Tento spôsob sa ukázal byť veľmi pomalý a na procesor veľmi náročný už pri malom počte bodov. Bolo to spôsobené tým, že každý bod vystupuje ako samostatný objekt, ktorý môže obsahovať množstvo doplnkových informácií, čo spôsobuje spomalenie pri spracovaní. Ukázalo sa, že ak spojím body do čiary – polyline, vystupuje táto čiara ako jeden samostatný objekt, čo výrazne urýchlilo prácu.

Čiara - polyline je vždy tvorená minimálne 2 bodmi. JavaScript u klienta preto musí rozhodnúť, ktoré 2 po sebe nasledujúce body má spojiť. Následnosť bodov neurčujem z času, ale z ich ID záznamu, pretože viem, že sa body ukladajú do databázy v chronologickom poradí so zväčšujúcim sa ID o jedničku. Ak by som miesto ID záznamu zasielal čas a dátum záznamu, znamenalo by to výrazne spomalenie spracovávania na strane klienta. Uvažoval som aj nad zasielaním času v tvare počtu sekúnd od zvoleného základného dátumu. Problém bol však vo veľkosti čísla, ktoré takto vznikalo a s narastajúcim objemom posielaných dát.

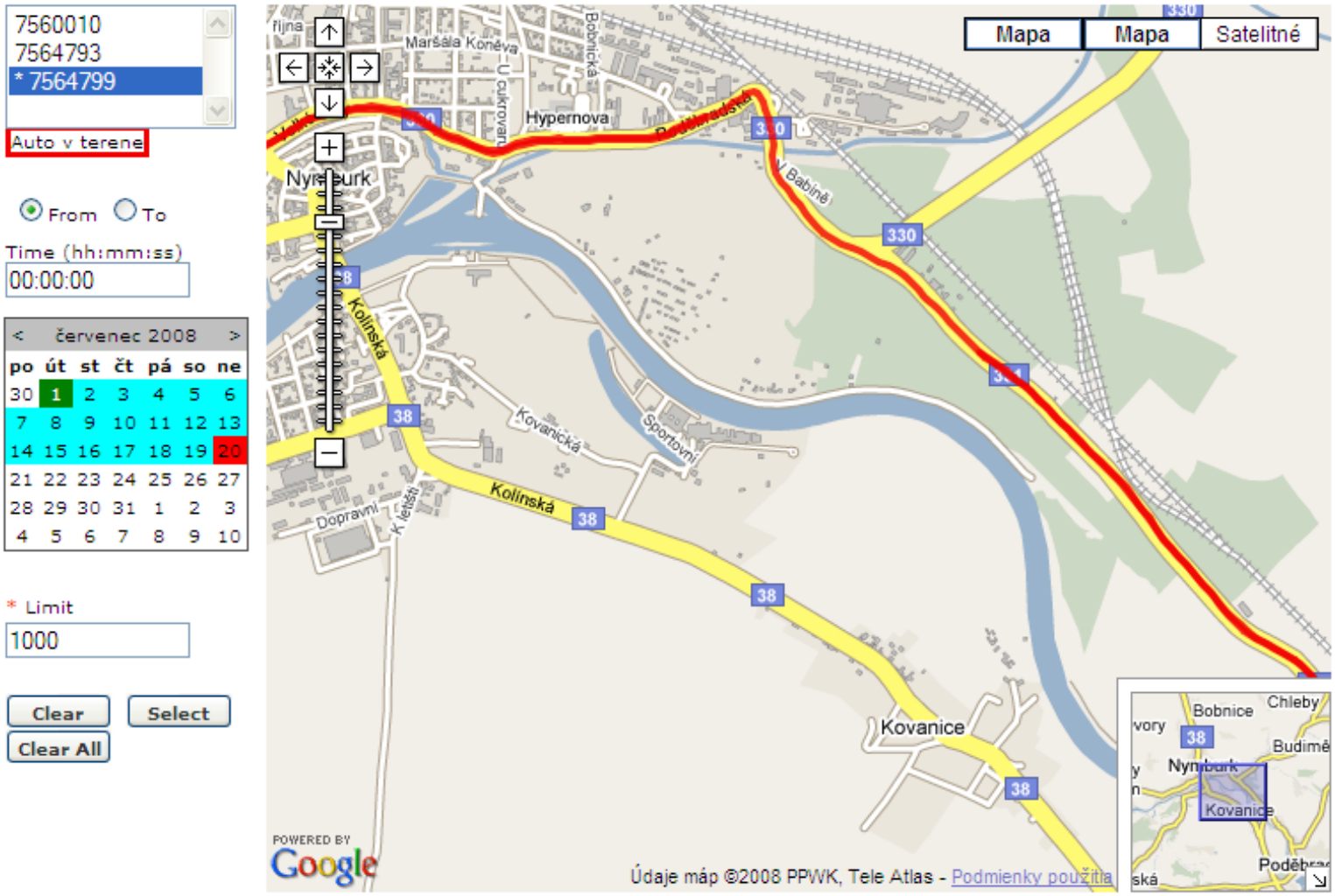
Pre každé 2 po sebe nasledujúce body sa určí ich vzájomná vzdialenosť. V prípade, že je ich vzdialenosť menšia ako nastavená konštanta 100m, vložia sa

súradnice bodov do poľa pre prvú čiaru - polyline. Ak je bod vzdialenejší ako 100m, vloží sa do druhého poľa pre druhú čiaru – polyline. Takto sa vytvoria jednotlivé čiary. V prípade, že pole obsahuje iba jediný bod, je nutné ho doplniť ešte jedným bodom, aby sa čiara vykreslila. Týmto druhým bodom je fiktívny bod, ktorý sa vypočíta pridaním malej odchýlky od prvého bodu. Po vytvorení všetkých čiar sa zavolá funkcia na vymazanie aktuálnych čiar a vykreslia sa nové. Príklad vykreslenia dát je na obrázku 4.4.

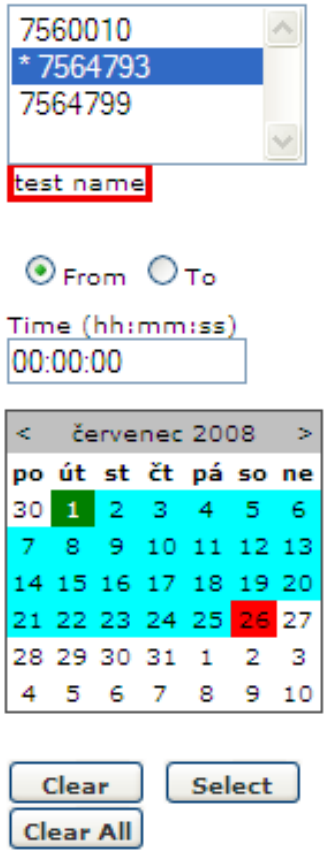
4.7. Tabuľkové zobrazenie prijatých dát

Na stránke GPS dáta (obr. 4.5) je možné zobraziť všetky prijaté GPS dáta v tabuľkovej podobe. Na rozdiel od zobrazenia na mape sa tu zobrazia aj GPS záznamy s príznakom neplatných dát spolu s dátumom a časom meraného záznamu. Pre výber a filtrovanie dát opäť slúži rovnaký fragment ako pre výber zobrazovaných dát na Google mape. Tento fragment však neobsahuje vstupné pole pre limit.

Stránkovanie dát je po 50 záznamoch, pre ich lepšiu čitateľnosť sa strieda farebné podfarbenie dvoch po sebe nasledujúcich záznamov. Zobrazené dáta je možné zoradiť podľa jedného z piatich stĺpcov a to času záznamu, ID zariadenia, zemepisnej šírky, dĺžky a príznaku platných dát.



Obr. 4.4 Zobrazenie dát na mape



Date	Device Id	Latitude	Longitude	GPS Valid
3.7.2008 21:10:41	7564793	0	0	False
3.7.2008 21:10:44	7564793	0	0	False
3.7.2008 21:10:47	7564793	0	0	False
3.7.2008 21:10:50	7564793	0	0	False
3.7.2008 21:10:53	7564793	0	0	False
4.7.2008 19:30:41	7564793	0	0	False
4.7.2008 19:31:41	7564793	0	0	False
4.7.2008 19:32:41	7564793	0	0	False
4.7.2008 19:32:54	7564793	49,85707	14,7956	True
4.7.2008 19:32:56	7564793	49,8567	14,79583	True
4.7.2008 19:32:58	7564793	49,85632	14,79605	True
4.7.2008 19:33:00	7564793	49,85595	14,79628	True
4.7.2008 19:33:02	7564793	49,85558	14,79645	True
4.7.2008 19:33:04	7564793	49,85518	14,79657	True
4.7.2008 19:33:06	7564793	49,85472	14,7969	True
4.7.2008 19:33:08	7564793	49,85432	14,7971	True
4.7.2008 19:33:10	7564793	49,85392	14,79718	True
4.7.2008 19:33:12	7564793	49,85352	14,79722	True
4.7.2008 19:33:14	7564793	49,85313	14,79727	True
4.7.2008 19:33:16	7564793	49,85273	14,79725	True
4.7.2008 19:33:17	7564793	49,85235	14,79722	True
4.7.2008 19:33:19	7564793	49,85195	14,7972	True
4.7.2008 19:33:21	7564793	49,85157	14,79715	True
4.7.2008 19:33:23	7564793	49,85117	14,7971	True
4.7.2008 19:33:25	7564793	49,8508	14,79705	True
4.7.2008 19:33:27	7564793	49,8504	14,797	True

Obr. 4.5 Tabuľkové zobrazenie prijatých dát

5. Záver

Cieľom mojej bakalárskej práce bolo zoznámiť sa s komunikátorom POSITRA GC 072 240F firmy LEVELNA. K tomuto účelu mi poslúžil stručný manuál dodávaný k zariadeniu spolu s aplikáciou GC Control Panel, ktorá poskytuje podrobnejší popis, než dodaný manuál od výrobcu. Napriek tomu, že v pomocníkovi k tejto aplikácii je toho popísane veľa, mnohé popisy nemajú veľmi veľkú výpovednú hodnotu, preto som sa veľmi často musel obrátiť s otázkami priamo na výrobcu komunikátora.

Pri overovaní jednotlivých funkcií komunikátora som narazil napríklad na chybu zariadenia, že ak karte SIM vyprší platnosť alebo je zo zariadenia vybraná v čase, keď pamäť komunikátora obsahuje nejaké záznamy, po zapnutí zariadenia sa tieto dáta bez varovania stratia. Výsledkom môjho neustáleho dotazovania a pripomienok k zariadeniu bolo vydanie nového firmwaru, ktorý tieto problémy odstránil. Novým firmwarom sa podarilo odstrániť niekoľko chýb, ktoré som hľadal v navrhnutej aplikácii a nemohol som ich nájsť.

Pri návrhu aplikácie som vychádzal z poskytnutej dokumentácie k zariadeniu, ktorá je dostupná iba na požiadanie od výrobcu. Zaviazal som sa, že túto dokumentáciu žiadnym spôsobom nezverejním. Dokumentácia obsahuje používaný protokol zariadenia. Žiaľ, opäť je písaná spôsobom, ktorý si vyžiadal častú komunikáciu s výrobcom kvôli ozrejmeniu mnohých nejasností. Na mnohé záležitosti sa mi podarilo prísť spôsobom pokus-omyl. Práve neprehľadná dokumentácia spôsobila nesprávne pochopenie zasielaných služieb a následne na to som navrhol nevhodný spôsob spracovávania prijatých dát. Preto som už navrhol nový spôsob spracovávania prijatých dát. Tento návrh sa bude implementovať v nasledujúcom kole prací projektu Lomon.

Webové rozhranie nemalo byť hlavnou prioritou mojej práce. Napriek tomu som nad ním strávil porovnateľné množstvo času ako na vývoji samotnej aplikácie. Bolo to spôsobené hlavne tým, že som nikdy predtým ešte nevyvíjal v ASP.NET frameworku. Tento framework je z môjho hľadiska príliš komplikovaný, napriek tomu sa mi nakoniec podarilo z veľkej časti splniť moje stanovené ciele. Jediným bodom, ktorý som navrhol a nesplnil, je správa užívateľov webového rozhrania. Napriek tomu, že správu užívateľov ASP.NET framework ponúka, môj návrh správy užívateľov a zariadení by si vyžiadal prepísanie veľkej časti vygenerovaného kódu.

6. Použitá literatúra a nástroje

[1] PROSISES, Jeff. Programování v Microsoft.NET Webové aplikace v.NET Framework, C# a ASP.NET. Brno: Computer Press, 2003.

[2] MACDONALD Matthew - SZPUSZTA Mario. ASP.NET 2.0 a C#, tvorba dynamických stránek profesionálně. Brno: ZONER software s.r.o., 2006

[3] Microsoft MSDN Library for Visual Studio 2008

[4] Manuály a dokumentácia komunikátora GC 072 240F

[5] Vývojový nástroj Microsoft Visual Studio 2008

[6] Konfiguračný nástroj GC 072 Control Panel firmy LEVELNA

Príloha

CD

[Drivers and Manuals]

[Application]

BP_2008_Marek_Riedel.pdf

readme.txt