UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jernej Gorički

Strojna in programska oprema za aktivno sledenje z navigacijskim sistemom GPS

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Uroš Lotrič

Ljubljana, 2011

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01746/2011
Datum: 01.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **JERNEJ GORIČKI**

Naslov: **STROJNA IN PROGRAMSKA OPREMA ZA AKTIVNO SLEDENJE Z NAVIGACIJSKIM SISTEMOM GPS**

DEVELOPMENT OF HARDWARE AND SOFTWARE EQUIPEMENT FOR REAL TIME GPS TRACKING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Zasnujte celostni sistem za sledenje osebam ali živalim v realnem času z uporabo navigacijskega sistema GPS. Rešitev naj vključuje strojno vezje in programske rešitve za krmiljenje namenskih naprav, njihovo komunikacijo s strežnikom, izdelavo samega strežniškega sistema in odjemalca. Strojno opremo izberite tako, da bo možna njena integracija v zapestnico ali ovratnico.

Mentor:
prof. dr. Uroš Lotrič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Jernej Gorički,

z vpisno številko 63030085,

sem avtor/-ica diplomskega dela z naslovom:

Strojna in programska oprema za aktivno sledenje z navigacijskim sistemom GPS

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom prof. dr. Uroš Lotrič
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se staršem, ki so mi v času študija stali ob strani, mentorju prof. dr. Urošu Lotriču ter zaposlenim v podjetju Evo-Teh, ki so z mano delili svoje praktične izkušnje.

Kazalo

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic in simbolov

GSM - Global System for Mobile communications

GNSS - Global Navigation Satellite Systems

GPRS - General Packet Radio Service

GPS - Global Positioning System

LBS - Location Based Services

LBIS - Location Based Information System

NMEA - National Marine Electronics Association

REST - Representational state transfer

URL - Uniform Resource Locators

WCF - Windows Communication Foundation

IIS - Internet Information Services

PIP - Point in Polygon

Povzetek

Diplomska naloga obravnava izdelavo informacijskega sistema namenjenega aktivnemu sledenju objektov z uporabo sistema GPS. Glavna gradnika sistema sta strojno vezje (namenska naprava) zmožno določitve položaja GPS in brezžične komunikacije GPRS ter informacijski sistem. Diplomska naloga opisuje izdelavo programske opreme za krmiljenje namenske naprave, kot tudi programsko opremo potrebno za nadzorni (strežniški) del in uporabniški del sistema.

Osrednji del diplomske naloge je tako razdeljen na strojni ter programski del. Glavni sestavni del namenske naprave predstavlja Telitov modul GSM, katerega nalogi sta pridobitev podatka o zemljepisni dolžini in širini, ter vzpostavitev povezave GPRS z oddaljenim strežnikom. Vsa programska oprema na strani namenske naprave je napisana v višje-nivojskem jeziku Python. Na drugi strani sistema se nahaja strežniška aplikacija, ki prejete podatke shranjuje v podatkovni strežnik Microsoft SQL Express ter spletna stran, ki s pomočjo kontrolnika Bing Maps podatke prikazuje v grafični obliki. Na strežniški strani smo kot razvojno platformo izbrali ogrodje Microsoft .NET in jezik C#, del sistema, ki skrbi za grafični prikaz podatkov, pa je izdelan v programskem jeziku Java Script.

Ključne besede: GPS, sledenje, Telit GE 865 Quad

Abstract

This thesis deals with the development of an information system intended for real-time tracking of people or objects using GPS satellite system positioning. Two main parts of the system are a dedicated device, capable of obtaining the location and communicating with other components over GPRS, and an information system. The thesis presents the development of software used by the dedicated device and also the software that manages the remote application server.

The main part of the thesis is divided into the hardware and the software section. The core part of the dedicated device is a Telit GSM module, whose task is to obtain information about longitude, latitude and transfer data to the remote application server. All software on the dedicated device is written in the higher-level language Python. On the other side of the system is a server application that receives data through a REST interface and stores it using Microsoft SQL Server Express. It also provides a user interface that displays visual representation of the whereabouts in real time using Bing Maps. From the software point of view the server-based applications are programmed using .Net platform and C# language, while the part of the system that handles information visualization is done with Java Script.

Keywords: GPS, tracking, Telit GE 865 Quad

1 Uvod

Pogoj za uspešno delovanje sledilnega sistema je vedno prisotna namenska naprava, katere fizične lastnosti ne smejo, oziroma morajo v čim manjši meri, vplivati na delovanje interesnih objektov. Idealna naprava torej predstavlja nevsiljiv dodatek, ki je zmožen zlitja z okolico, v kateri interesni objekt deluje. Fizične lastnosti takšnih naprav so razlog, da je storitev sledenja trenutno najbolj razširjena na področju prometa in prevoznih sredstev. Avtomatizacijo in optimizacijo logističnih procesov, sledenje in upravljanje voznega parka v podjetju, zaščito vozila proti kraji in podobno omogočajo naprave, ki nameščene v vozilo skoraj niso opazne.

Napredek v razvoju vedno manjših elektronskih naprav, razcvet mobilne telefonije ter elektronski elementi, kot so merilni in izvršni členi ter procesne enote, ki so dosegljivi po vedno bolj dostopnih cenah, so dejavniki, ki povzročajo nenehne spremembe in hiter razvoj področja lokacijskih storitev. Miniaturizacija in vedno manjša poraba električne energije pa sta glavna razloga, da lahko storitev sledenja uporabimo tudi za druge interesne skupine, kot so na primer starostniki, otroci ali male živali. Ljudje smo v vsakdanjem življenju vedno bolj odvisni od sodobne tehnologije, zato od nje zahtevamo, da nam je na voljo vedno in povsod. Vizija prihodnosti postavlja človeka v središče okolja prepredenega z inteligentnimi uporabniškimi vmesniki, podprtimi z informacijskimi in komunikacijskimi tehnologijami, ki bodo vgrajeni v vsakdanje predmete.

Diplomsko delo opisuje programsko ter strojno opreme, ki je potrebna za izdelavo informacijskega sistema namenjenega aktivnemu sledenju objektov, oziroma sledenju objektov v realnem času. Z besedo aktivno označujemo, da gre za sistem, kjer namenska naprava podatek o poziciji objekta, običajno preko omrežja GSM (ang. Global System for Mobile communications), sproti pošilja v nadzorni center, kar je v nasprotju s pasivnimi sistemi, kjer se podatki o poziciji, hitrosti ter smeri vedno obdelujejo za že pretečeno časovno obdobje.

Funkcijo lokacijske enote v diplomi opravlja namenska naprava zmožna določevanja pozicije, sprejemanja ukazov ter pošiljanja podatkov na oddaljeni strežnik. Majhne fizične ter ustrezne delovne lastnosti izbranih elementov so primerne za integracijo namenske naprave v ovratnico za male živali ali pa osebno sledilno napravo namenjeno varovanju otrok ali starostnikov. To je tudi glavni razlog, da smo se odločili za uporabo arhitekture brez zunanjega mikrokrmilnika.

Velik izziv pri razvoju programske opreme, ki se izvaja na namenski napravi, je kar najbolj podaljšati avtonomijo baterije. V primeru, da se objekt ne giblje, oziroma, da se giblje na nekem v naprej predvidenem območju, lahko energijo privarčujemo na račun manjšega števila prenesenih podatkov na oddaljeni strežnik, ne da bi pri tem vplivali na kvaliteto delovanja celotnega sistema.

Obveščanje uporabnika bi lahko v celoti izvedli s pošiljanjem kratkih sporočil SMS ali preko elektronske pošte. A vendar predstavlja uporaba oddaljenega strežnika iz stališča razširljivosti sistema ter cenovnega vidika, bolj smiselno izbiro. Oddaljeni strežnik sestavljajo relacijska podatkovna baza, vmesnik REST za dostop do podatkov ter administrativna in uporabniška spletna stran. Uporabniška spletna stran izvaja prikaz podatkov v grafični obliki s pomočjo storitve Bing Maps [14]. Poleg spremljanja lokacije lahko na zemljevidu določimo navidezno ograjo (angl. GeoFence), kar omogoča proženje alarmov v primeru, da opazovani objekt zapusti ali vstopi v navidezno ograjeno območje.

Prvi cilj diplomskega dela je vzpostavitev razvojnega okolja, v katerem bomo lahko testirali programsko opremo namenjeno izvajanju na namenski napravi, katere osrčje predstavlja modul GSM z mikrokrmilniškimi lastnostmi Telit GE 865. Drugi cilj je izdelava programske opreme tako na nivoju namenske naprave kot tudi na oddaljenem računalniku oziroma strežniku. Naloge programske opreme želenega sistema so pridobitev podatkov o poziciji, storitev virtualne ograje (ang. GeoFence), komunikacija z oddaljenim strežnikom, spletna aplikacija za sprejem podatkov, podatkovna baza in predstavitev informacij.

V naslednjem poglavju bodo najprej opisani sestavni deli ciljnega sistema, v tretjem poglavju pa bodo opisani navigacijski satelitski sistemi. Jedro diplomske naloge v obliki strojnega in programskega dela sistema je predstavljeno v četrtem oziroma v petem poglavju. Najpomembnejše ugotovitve so zbrane v zaključku.

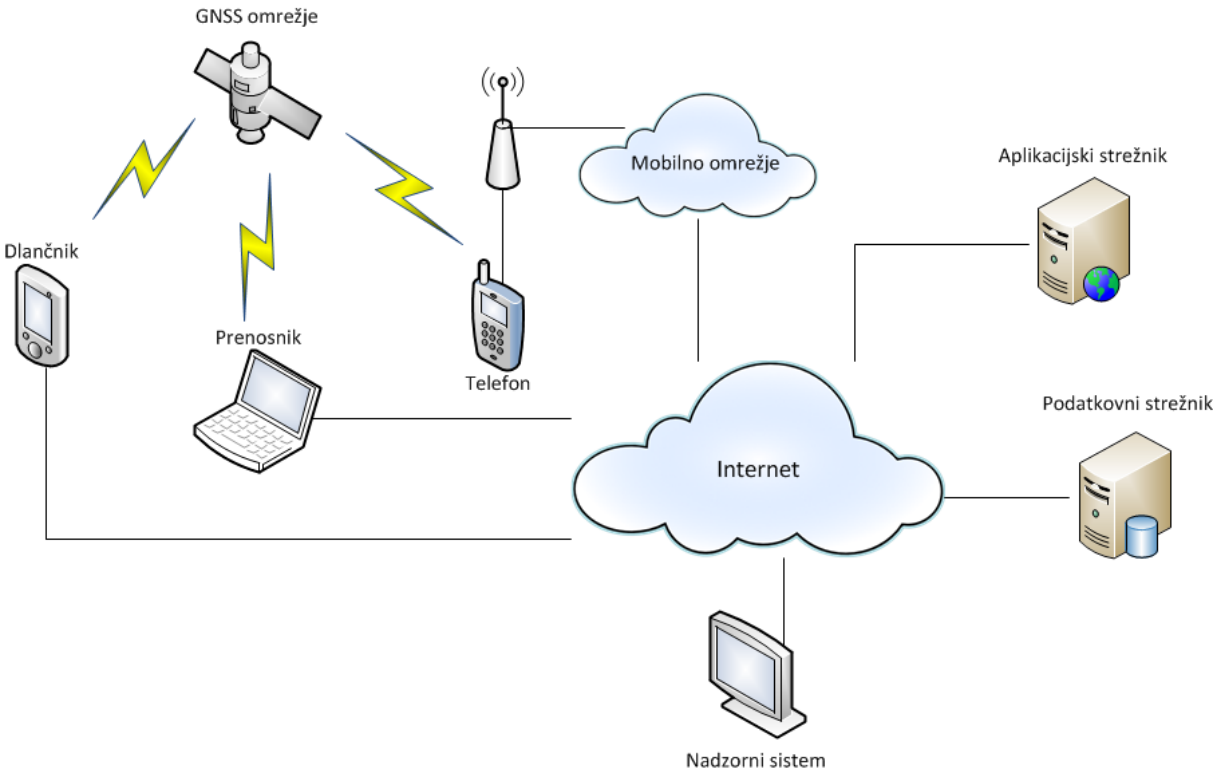
2 Lokacijsko podprte storitve

Storitve sledenja uvrščamo med lokacijsko podprte storitve (angl. Location Based Services oziroma LBS). Diplomsko delo [2] lokacijsko podprte storitve definira kot kontekstno zavedne storitve, ki skrbijo za dostavo podatkov in informacij, katerih vsebina je prirejena trenutni ali neki projektirani lokaciji in kontekstu mobilnega uporabnika. Med najbolj popularne tovrstne storitve spadajo [3]: navigacija GPS, vremenska napoved, prometne informacije, iskanje interesnih točk (trgovine, restavracije, bankomati) ter mobilno oglaševanje.

Razvoj interneta, brezžičnih tehnologij ter mobilnih naprav je omogočil idealno tehnološko podlago, ki jo potrebujejo lokacijsko podprte storitve za uspešno izvajanje. Pod pojmom lokacijsko podprti informacijski sistem (angl. Location Based Information System ali LBIS) tako razumemo celotno infrastrukturo potrebno za izvajanje lokacijsko podprtih storitev. Takšen sistem sestavljajo [1]:

- sistem določanja položaja,
- namenska naprava,
- komunikacijsko omrežje,
- glavni nadzorni sistem,
- strežniška infrastruktura.

Slika 1 prikazuje primer enostavnega LBIS sistema za sledenje.



Slika 1: Infrastruktura LBIS za sledenje.

2.1 Sistemi določanja položaja

Ključno vlogo pri storitvah LBS igra podatek o poziciji. Na podlagi tega kdo posreduje podatek izvajalcem LBS, lahko določevanje pozicije razdelimo na 3 različne arhitekture [1].

- Pri določanju v omrežju (angl. Network-based location provider) podatke o poziciji izračunava ter hrani omrežje. Eno izmed takih je telekomunikacijsko omrežje GSM, ki pozicijo izračunava s pomočjo podatkov iz baznih postaj, na katere je uporabnik v določenem trenutku povezan. Storitev je običajno zaračunana iz strani ponudnika mobilne telefonije. Primer je Mobitelova storitev "Kje sem"[1]. Natančnost določanja pozicije je odvisna od lastnosti ter števila baznih postaj. Natančnost v najboljših primerih znaša 50 metrov.

- Pri določanju na napravi (angl. Mobile-based location provider) je naprava zmožna samostojno določiti pozicijo, najpogosteje ji to omogoča vgrajeni sprejemnik GPS, lahko pa si pomaga tudi s podatki ponudnika mobilne telefonije.

- Pri določanju preko ponudnika (angl. location provider-based) so podatki o poziciji posredovani tretji stranki, ta pa jih nato dostavi več izvajalcem storitev LBS. Takšen način rešuje problem večkratnega pošiljanja istih informacij iz naprave v primeru, da se na napravi izvaja več kot ena LBS aplikacija.

Nalogo določevanja pozicije v diplomski nalogi opravlja namenska naprava opremljena z sprejemnikom GPS.

2.2 Namenska naprava

Tipični predstavnik je mobilni telefon, katerega največji prednosti sta prilagodljivost oziroma izvajanje lastnih aplikacij ter neprestana povezanost v omrežje. Lahko gre tudi za dlančnik, zapestnico GPS za starejše, ovratnico za male živali oziroma katerokoli napravo, ki je zmožna določiti pozicijo ter komunicirati z ostalimi deli sistema LBIS.

2.3 Transportno omrežje

Omrežje preko katerega med seboj komunicirajo različne komponente sistema LBIS. Prenosne naprave običajno komunicirajo preko mobilnega podatkovnega omrežja s storitvijo GPRS, ostali deli pa so v splet povezani preko žičnega ali brezžičnega omrežja po protokolu TCP/IP.

[1] http://www.mobitel.si/storitve/kje-sem.aspx

2.4 Nadzorni sistem

Izvaja funkcijo upravljanja s sistemom in omogoča vizualno predstavitev lokacije v realnem času. Zraven opravlja tudi administratorska opravila dodajanje novih uporabnikov, brisanje uporabnikov, urejanje informacij ter komunicira z namensko napravo.

2.5 Strežniška infrastruktura

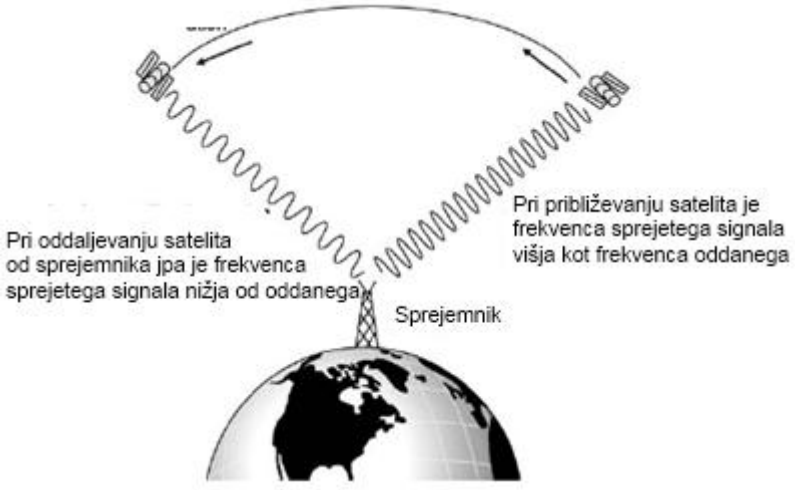
Podatkovni ter aplikacijski strežniki omogočajo centraliziran način procesiranja ter shranjevanja podatkov, ki jih pošiljajo namenske naprave.

3 Sistem GPS

Podatke o poziciji GSM modulu posredujemo preko sprejemnika GPS (ang. Global Positioning System). V tem poglavju predstavimo zgodovino, glavne segmente ter naravne zakonitosti navigacijskih satelitskih sistemov GNSS (Global Navigation Satellite Systems) med katere trenutno uvrščamo ameriški GPS (NAVSTAR), ruski GLONASS, evropski Galileo ter kitajski Beidou. Podrobnejši vpogled v delovanje sistema GNSS nam omogoča izbiro primernega sprejemnika za izgradnjo naše namenske naprave. Poudarek je na ameriškem sistemu GPS, ki je edini operativen sistem na globalni ravni. Sledi mu ruski GLONASS, katerega sprejemniki si že počasi vtirajo pot na globalni trg navigacijskih naprav, saj že zagotavlja pokritost celotnega ruskega ozemlja, do leta 2012 pa naj bi dosegel polno operativno zmogljivost [9]. Evropska in kitajska sistema sta trenutno še v povojih in načrtujeta operativno delovanje do leta 2020. Kratica GPS v tej nalogi označuje GNSS, s katerim upravljajo Združene države Amerike, prav tako bo sprejemnik GPS označeval napravo, ki deluje znotraj ameriškega sistema GNSS.

3.1 Zgodovina

Prvi so idejo za izgradnjo sistema GNSS dobili Američani leta 1957 [9]. Prek spremljanja signalov, ki jih je na zemljo pošiljal prvi umetni satelit ruski Sputnik 1, so ugotovili, da je frekvenca signala višja, ko se satelit sprejemni postaji na zemlji približuje ter nižja, ko se le ta od nje oddaljuje (slika 2). Dopplerjev pojav, ki je razlog za spremembo valovne dolžine oziroma frekvence signala, je tako postal osnova za delovanje prvih sistemov GNSS kot sta bila ameriški Transit in ruski Tsyklon. Za določitev lokacije je bil potreben signal iz enega satelita, ki je vseboval informacije o času ter lokaciji satelita, sprejemnik pa je nato preko Dopplerjevega premika frekvence določil razdaljo do 500 metrov natančno. Za doseganje večje natančnosti je ameriški sistem oddajal signal v dveh frekvencah, kar je omogočalo natančnost do 25 metrov.



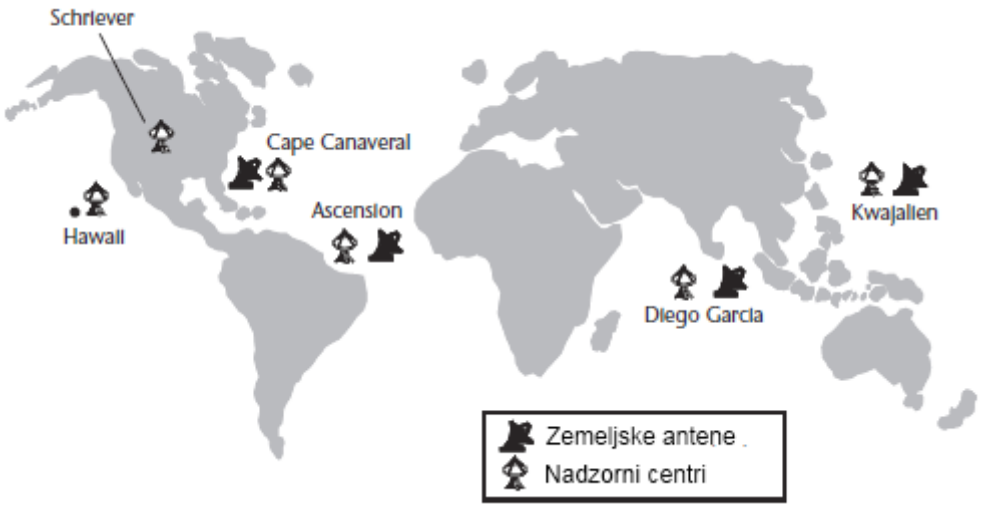
Slika 2: Valovna dolžina med približevanjem in oddaljevanjem satelita.

Sistemi GNSS, ki so računali lokacijo z uporabo Dopplerjevega premika, so bili predhodniki sistemov kot jih poznamo danes. Največji pomanjkljivosti sistema Transit sta bili poleg nenatančnosti še nizka orbita satelitov, kar je posledično pomenilo omejeno dosegljivost storitve.

3.2 Segmenti sistema GPS

Leta 1973 je bila Ameriški letalskim silam dodeljena skrb za nov sistem po imenu NAVSTAR (Navigation Satelite Timing And Ranging), ki predstavlja današnji GPS. Prvi satelit sistema NAVSTAR je bil izstreljen leta 1978, danes pa sistem sestavlja že 28 satelitov od katerih so štirje rezervni. Sateliti so razdeljeni v šest orbit. V vsaki orbiti se nahajajo štirje sateliti, ki krožijo na višini 20.200 kilometrov in so glede na ekvatorialno ravnino nagnjeni 55 stopinj. Takšna razporeditev satelitov omogoča, da je iz katerekoli točke na zemlji neprestano vidnih od štiri do dvanajst satelitov. Najpomembnejša komponenta GPS satelita je atomska ura, ki igra ključno vlogo pri določevanju psevdorazdalje med satelitom in sprejemnikom. Sateliti oziroma vesoljski del tako predstavlja enega izmed treh segmentov sistema GPS. Poleg vesoljskega segmenta sta tu še kontrolni del, ki s sateliti upravlja ter uporabniški del, ki ga predstavlja tržišče sprejemnikov GPS.

Kontrolni del je odgovoren za pravilno delovanje celotnega omrežja GPS. Vzdržuje ga z nadzorovanjem delovanja satelitov, ohranjanjem pravilnih orbit, aktivacijo rezervnih satelitov ter generiranjem in distribucijo navigacijskega sporočila. Podatki iz satelitov se preko šestih nadzornih centrov razkropljenih po celem svetu, skupaj z meteorološkimi podatki, pošiljajo v glavno nadzorno postajo locirano v vojaški bazi Schriever Air Force Base v zvezni državi Colorado. Popravki izračunani v glavni nadzorni postaji se nato preko štirih zemeljskih anten približno vsaki dve uri pošljejo nazaj satelitom. Slika Slika 3 prikazuje razporeditev objektov kontrolnega dela sistema GPS.



Slika 3: Kontrolni del sistema GPS.

Uporabniški segment predstavlja končne uporabnike storitve sistema GPS. To so nabori čipov GPS, ki se vgrajujejo v mobilne telefone, dlančnike, navigacijske naprave itd. Sistemi GNSS so pasivni sistemi, kar pomeni, da signal vedno potuje le v smeri satelit – sprejemnik. Pasivnost omogoča, da število uporabnikov takšnega sistema ni številsko omejeno. Kljub temu da izdelovalci čipov GPS dandanes skoraj na vsakodnevni ravni objavljajo nove izdelke, ki obljubljajo najbolj zmogljiv, najmanjši ter najbolj nizko porabni nabor, se le ti pri testiranju vsi obnesejo skoraj enako dobro. Osrčje najbolj popularnih GPS sprejemnikov kot so Garmin in TomTom predstavljajo čipi podjetij MediaTek in CSR (bivši SiRF).

3.3 Struktura signala GPS

Strukturo GPS signala tvorita psevdo-naključni niz (angl. PRN – PseudoRaNdom code) ter navigacijsko sporočilo. Niz PRN služi za identifikacijo satelita ter kalkulacijo psevdorazdalje [9]. Navigacijsko sporočilo tvorijo trije pomembnejši sklopi informacij. V prvem delu se nahajajo informacije o trenutnem stanju satelita ter njegovi uri. Drugi del vsebuje efemeride[2], ki vsebujejo podatke potrebne za izračun trenutne lokacije satelita. Tretji del, imenovan almanah, pa predstavlja imenik vseh operativnih GPS satelitov ter njihovih identifikacijskih številk. Navigacijsko sporočilo sestavljajo logične celote podatkov imenovani okvirji. Velikost enega okvirja je 1.500 bitov. Okvir je sestavljen iz petih pod-okvirjev, ki se oddajajo s hitrostjo 50 bit/s, kar pomeni, da prenos okvirja traja 30 sekund.

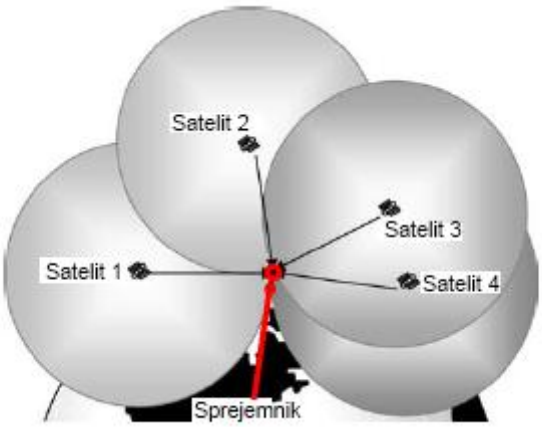
Sateliti oddajajo 2 vrsti niza PRN: C/A (angl. Coarse Acquisitionj) in P (Precision) na dveh različnih frekvencah. Niz C/A se oddaja na frekvenčnem pasu L1 (1.575,42 MHz), niz P pa na frekvenčnih pasovih L1 ter L2 (1.227,60 MHz). Dva signala obstajata zaradi dveh nivojev uslug. Storitev SPS (angl. Standard Positioning Service) uporablja niz C/A in je namenjen civilni uporabi. Storitev PPS (angl. Precise Positioning Service) pa uporablja niz PRN tipa P,

[2] Efemeride (astr.): periodična publikacija s podatki o legah nebesnih teles; termin GPS-efemeride uporabljamo

ki je dodatno zaščitena, natančnejša ter na voljo samo pooblaščenim uporabnikom – ameriški vojski.

3.4 Določanje položaja

Sistem GPS oziroma vsi današnji sistemi GNSS delujejo po principu trilateracije, kar pomeni, da lahko vsakemu sprejemniku na zemlji določimo položaj, če poznamo trenutni položaj ter oddaljenost vsaj štirih referenčnih točk oziroma satelitov. Lokacije satelitov so zapisane v efemeridnih podatkih, ki jih sprejemnik prejme preko navigacijskega sporočila. Razdaljo med sprejemnikom in satelitom imenujemo psevdorazdalja, določimo pa jo tako, da izmerimo čas, ki ga je signal porabil za potovanje med satelitom in sprejemnikom. Satelit proizvaja niz PRN s pomočjo algoritma, ki je časovno sinhroniziran z algoritmom na strani sprejemnika. Sprejemnik nato določi čas potovanja signala tako, da poišče kdaj je satelit ustvaril isti niz PRN, kot ga je on sam. Razliko v času množimo s hitrostjo potovanja signala, ki je enaka hitrosti svetlobe, in dobimo psevdorazdaljo. Težava je v tem, da ima sprejemnik uro slabše kakovosti tako, da ni čisto sinhronizirana s satelitom. Psevdorazdalja se imenuje, ker je izračunana na podlagi časovne razlike med dvema neusklajenima urama. Popravki se izračunavajo tako, da sprejemnik neprenehoma popravlja notranjo uro na podlagi dejstva, da mora obstajati med štirimi sferami natanko eno presečišče.



Slika 4: Navidezne sfere s polmerom enakim razdalji.

Slika 4 prikazuje sprejemnik, ki se nahaja v presečišču štirih navideznih sfer s premerom enakim razdalji med sprejemnikom in satelitom. Razdaljo do treh satelitov potrebujemo za določitev zemljepisne širine in dolžine, četrti satelit pa za popravek ure ter določitev višine.

3.5 Viri napak

Vsak izmed treh segmentov sistema GPS je ranljiv na svoj način. Vesoljski del z občutljivimi atomskimi urami, kjer lahko napaka samo ene mikrosekunde povzroči odstopanje do 300 metrov. Kontrolni del, z napačnim izračunavanjem popravkom ter uporabniški del, kjer je zaradi šibkosti signala otežen oziroma skoraj nemogoč sprejem pod vodo, zemljo ali v zaprtih prostorih. Poleg tega je signal lahko podvržen še namernim interferenčnim motnjam (angl. jamming) in potvarjanju signala (angl. spoofing).

Ključnega pomena, pri izračunu psevdorazdalje, je, kot smo že omenili, časovna razlika med oddanim in prejetim signalom. Pri tem upoštevamo, da je hitrost signala enaka hitrosti potovanja svetlobe, kar pa ne velja celotno pot potovanja. Pri potovanju skozi ionosfero (80 – 400 kilometrov višine) in troposfero (do 12 kilometrov višine) se hitrost signala upočasni. Pri troposferi vplivata na hitrost vlažnost ter spremenljiva temperatura. V ionosferi pa povzročajo oviranje signalov elektroni in ioni. Manj problematične so napake v troposferi, saj jih lahko s pomočjo predvidljivih vremenskih razmer in matematičnih modelov lažje odpravimo, medtem ko so napake v ionosferi, ki jih povzročajo nepredvidljive razmere v vesolju (aktivnosti sonca) trši oreh.

4 Strojna oprema

Namen tega poglavja je predstaviti komponente razvojnega okolja v katerem je bila napisana ter preizkušena programska oprema napisana za mikrokrmilniški vmesnik GSM Telit GE 865 Quad. Omenjeni modul GSM predstavlja osrčje namenske naprave, katere glavne naloge so GPS določitev položaja preko ločenega sprejemnika GPS ter komunikacija z oddaljenim strežnikom preko povezave GPRS.

Razvojno okolje smo pripravili s pomočjo treh glavnih komponent:

- razvojne enote Telit EVK2 Evaluation kit,
- mikrokrmilniškega vmesnika Telit GE 865 Quad in
- sprejemnik GPS z oznako FGPMMOPA6B.

4.1 Razvojna enota Telit EVK2 evaluation kit

Razvojna enota je namenjena hitremu razvoju in testiranju programske opreme za celotno paleto Telitovih mikrokrmilniških modulov GSM. Sam modul se nahaja na ločenem vmesniku, ki se na razvojno enoto priključi preko dveh 40 - pinskih priključkov. Takojšnji pričetek s programiranjem naprave nam omogočajo tipke za vklop ter ponovni zagon mikrokrmilnika, priključka USB/DB9 za povezavo z osebnim računalnikom ter ustrezna logika za transformacijo nivojev primernih za RS232 (pretvorba iz TTL napetostnih nivojev v RS232 nivoje in obratno). Slika 5 prikazuje razvojno enoto z modulom, pritrjenim prek ločenega vmesnika.

Slika 5: Razvojna enota in vmesnik Telit GE 865 Quad.

4.2 Vmesnik Telit GE 865 QUAD

Vmesnik s pritrjenim GE 865 GSM/GPRS modulom, nosilcem za SIM kartico, SMA (SubMiniature version A) priključkom za GSM anteno. Prav tako se na vmesniku nahajajo nožice za testiranje GPIO (angl. General Purpose Input Output) ter vmesnikov A/D in D/A. Modul ima namreč deset vhodov/izhodov tipa GPIO, ter dva pretvornika A/D in en pretvornik D/A. Vmesnik lahko priključimo na osebni računalnik tudi brez razvojne enote, vendar moramo v tem primeru sami poskrbeti za tipke, transformacijo signalov serijskega vmesnika ter statusne diode LED.

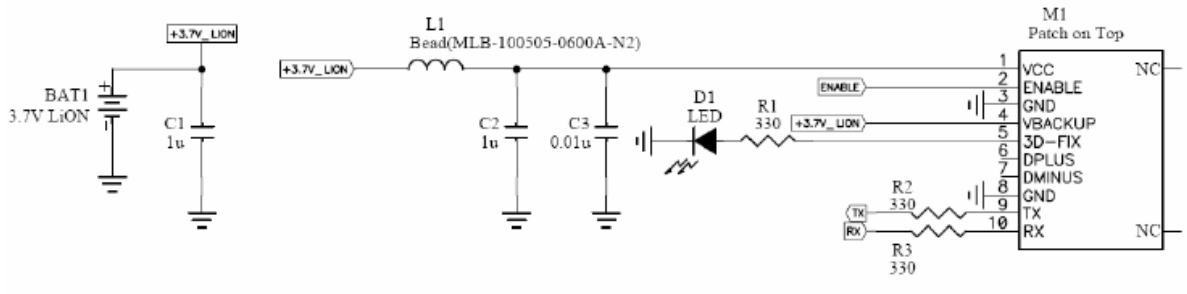
4.3 Sprejemnik GPS FGPMMOPA6B

Preden smo se odločili za izbiro GPS modula smo raziskali, kako različne lastnosti vplivajo na delovanje modula in s tem posredno tudi na našo namensko napravo. Lastnosti so predstavljene v tabeli Tabela 1.

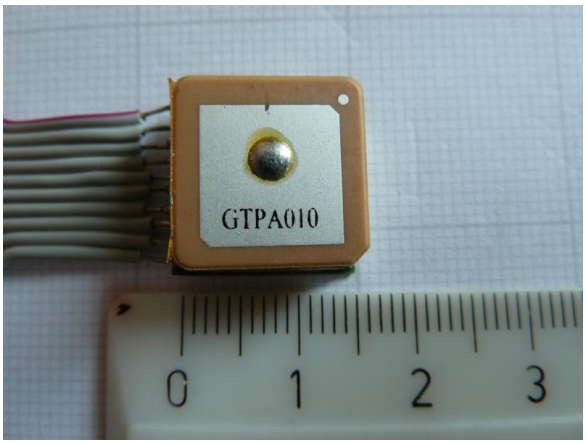
Tabela 1: Lastnosti sprejemnika GPS.

Velikost	Skoraj vsi današnji moduli so velikosti nekaj centimetrov. V primeru, da ima modul že vgrajeno anteno, ga to naredi malenkost višjega.
Frekvenca osveževanja	Frekvenca označuje kako pogosto modul na svojem izhodu zapiše podatek o lokaciji. Hitreje kot se premikamo, višjo frekvenco potrebujemo, za natančno določitev pozicije. Večina modulov deluje s frekvenco 1 Hz, kar je v našem primeru zadovoljivo.
Napajanje	Povprečna poraba okoli 30 mA pri 3,3 V, napajanje pa potrebuje tudi antena ki lahko porabi med 20 mA in 30 mA.
Število kanalov	Več kot ima modul kanalov, več satelitov lahko ta naenkrat spremlja, hitrejši je čas fiksacije pozicije. Ker pa je maksimalno število satelitov, ki jih iz katerekoli točke na zemlji lahko vidimo naenkrat samo 12 (v zelo redkih primerih) je večje število kanalov nepotrebno in predstavlja prodajni trik proizvajalcev čipov GPS.
Antena	V primeru, da ima modul že vgrajeno keramično anteno, ki je pritrjena na vrhu, ne potrebujemo dodatne antene. Vendar so takšni moduli bolj občutljivi, saj njihova lastna elektronika proizvaja motnje pri sprejemu. V primeru keramične antene moramo poskrbeti, da je modul obrnjen z anteno navzgor.
VBATT priključek	Nekateri sprejemniki vsebujejo dodatni delovni spomin namenjen shranjevanju efemeridnih ter almanah podatkov. VBATT zvežemo na vir napajanja. S tem dosežemo, da se omenjeni podatki ohranijo tudi ob izklopu modula. V nasprotnem primeru bo podatke potrebno ponovno prenašati ob vsakem vklopu sprejemnika. S shranjevanjem efemeridnih ter almanah podatkov se znatno zmanjša čas nadaljnjih vzpostavitev TTSF (angl. Time To Subsequent Fix) medtem, ko lahko čas do prve vzpostavitve TTFF (angl. Time To First Fix) traja tudi do 15 minut.

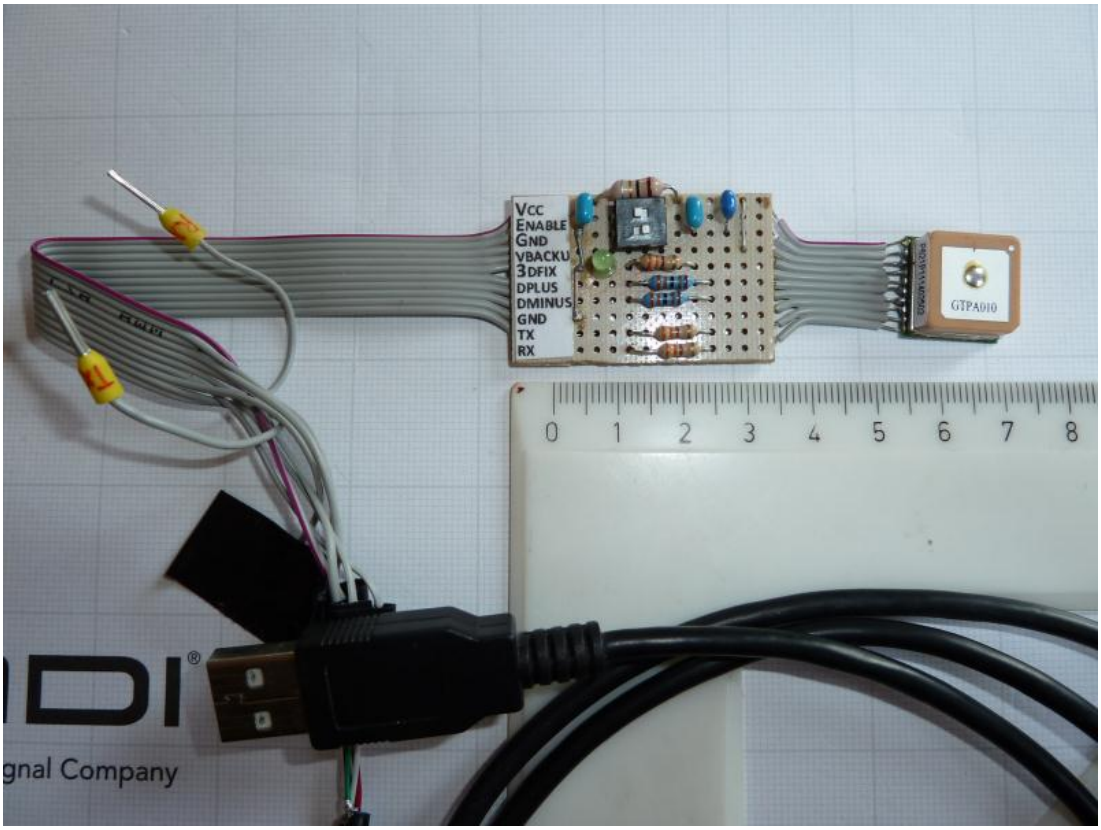
V diplomski nalogi smo uporabili sprejemnik GPS FGPMMOPA6B z že vgrajeno keramično anteno (slika 7) ter vgrajenim vmesnikom USB. Dodatna priključka D+ ter D- smo povezali neposredno z vodilom USB na računalniku. Enostaven priklop nam je omogočil, da smo programsko opremo ter delovanje sprejemnika najprej testirali na računalniku, serijsko vezavo na vmesnik Telit GE 865 Quad pa izvedli šele v naslednjem koraku. Modul, ki ima vgrajene čipe podjetja MediaTek, omogoča sprejem na frekvenci L1 (66 kanalov). Maksimalna frekvenca osveževanja znaša 10 Hz, povprečna poraba pri iskanju satelitov je 48 mA, pri sledenju pa 37 mA. Za potrebe testnega okolja smo na sprejemniku (slika 6 in slika 8) vezali priključne nožice VCC, GND ter D+ in D-.



Slika 6: Sprejemnik GPS - vezava priključkov.



Slika 7: Sprejemnik GPS s keramično anteno.



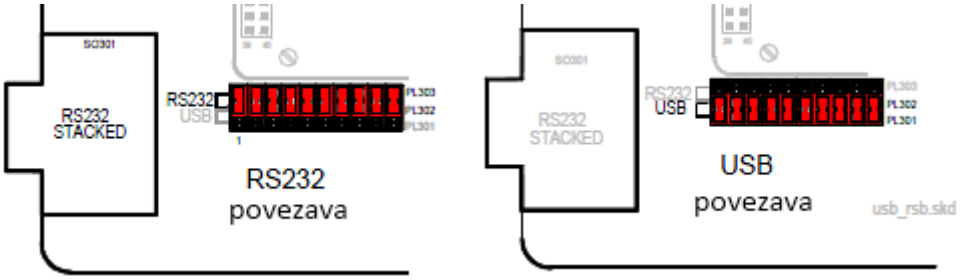
Slika 8: Pripravljen GPS za povezavo z osebnim računalnikom.

4.4 Priprava razvojnega okolja

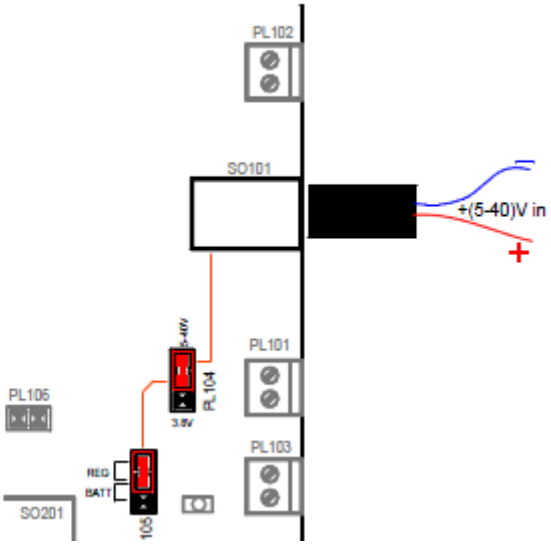
Za uspešno komunikacijo ter testiranje modula Telit GE 865 Quad potrebujemo:

- kabel USB ali serijski kabel, pri slednjem potrebujemo na kablu minimalno vezavo priključkov RX/TX,
- adapter,
- anteno GSM,
- gonilnike FTDI in
- kartico SIM.

V primeru serijske povezave med razvojno platformo in računalnikom ni potrebno namestiti dodatnih gonilnikov. Ker pa so serijski priključki COM dandanes že redkost je potrebno ob uporabi priključka USB namestiti gonilnike za navidezna vrata COM [6]. Pozorni moramo biti na pravilno nastavitev mostičkov na razvojni enoti v odvisnosti od realiziranega priključka USB/serijski (slika 9) in pa tudi vira napetosti. Za vzpostavitev delovanja v testnem okolju ga napajamo z enosmerno napetostjo v območju 5 – 40 V (slika 10). Upoštevati moramo tudi navodilo, da je potrebno modul najprej vklopiti in šele nato priključiti na računalnik, saj lahko pošiljanje podatkov na serijskem vodilu prvih 200 ms po vklopu modula privede do nepravilnega delovanja [12].



Slika 9: Lega mostičkov za priklop na vmesnik RS232 (levo) in za priklop na vmensik USB (desno).



Slika 10 Lega mostičkov za izbrano napajanje.

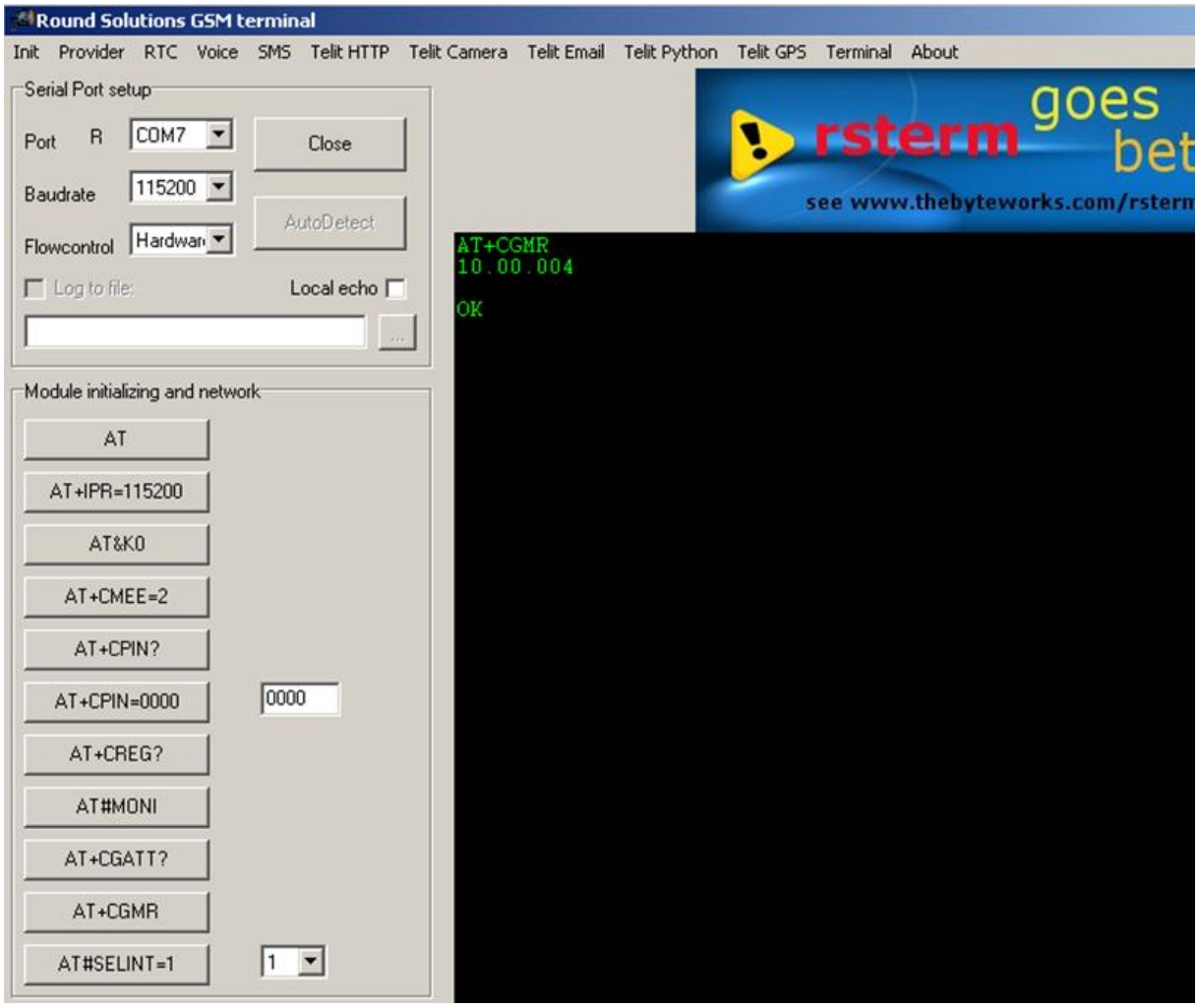
Po prižigu razvojne enote spremljamo statusno diodo. Če imamo vstavljeno kartico SIM in priključeno anteno GSM, bo v primeru uspešne registracije na omrežju GSM, začetno utripanje diode LED v eno-sekundnih intervalih po približno 15 sekundah nadomestilo utripanje v tri-sekundnih intervalih. To pomeni, da se je modul uspešno registriral v omrežju ponudnika mobilne telefonije. V primeru, da imamo kartico SIM zaščiteno s kodo PIN moramo le to vnesti z ukazom: AT+CPIN = XXXX.

Delovanje modula lahko preverimo s programi, ki omogočajo pošiljanje ukazov prek serijskega vmesnika RS232. V primeru da imamo na razvojnem računalniku nameščen operacijski sistem Windows XP, lahko uporabimo kar vgrajeni program namenjen diagnosticiranju modemov Hyper Terminal (hypertrm.exe), lahko pa uporabimo brezplačni program namenjen prav testiranju in diagnosticiranju Telitovih modulov rsterm.exe (slika 11) dosegljiv na spletnem naslovu http://www.roundsolutions.com.

V primeru pravilnega delovanja lahko modem preizkusimo z ukazi predstavljenimi v tabeli Tabela 2.

Tabela 2: Osnovni ukazi AT.

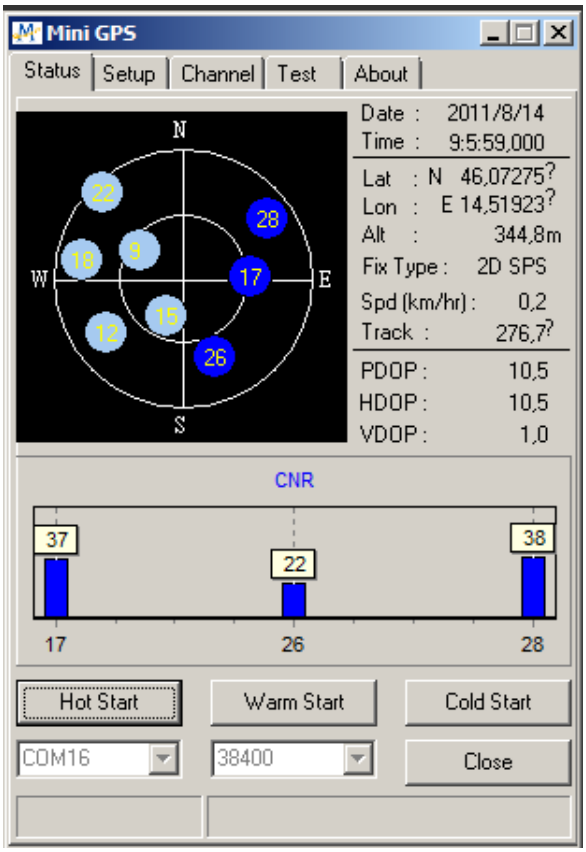
Ukaz	Odgovor	Opis
AT	OK	Ping
AT+CGMR	10.00.004 OK	Verzija strojne programske opreme
AT+CGMM	GM862-QUAD OK	Ime modula
AT+CGSN	523456786543212	Identifikacijska številka naprave



Slika 11: Testiranje modula GSM z orodjem rsterm.exe.

4.5 Komunikacija osebni računalnik – sprejemnik GPS

Za potrebe testiranja smo sprejemnik GPS najprej preko vodila USB povezali z osebnim računalnikom. Ob uspešnem priklopu na računalnik postane sprejemnik GPS viden preko virtualnih vrat COM, na katera se povežemo z brezplačno aplikacijo Mini GPS (slika 12) dostopno na naslovu http://minigps.software.informer.com/1.4/.



Slika 12: Statusni podatki sprejemnika GPS.

Namen uporabe aplikacije Mini GPS je:

- testiranje delovanja vezja GPS,
- spreminjanje parametrov sprejemnika GPS,
- spoznavanje komunikacijskega protokola NMEA (angl. National Marine Electronics Association).

4.6 Komunikacija vmesnik Telit GE 865 Quad – sprejemnik GPS

Tekom razvoja programske opreme za namensko napravo sprejemnika GPS nismo priključili neposredno na razvojno platformo, saj je serijsko povezavo že zasedala povezava z osebnim računalnikom. Ker smo sprejemnik GPS že imeli povezan z osebnim računalnikom smo to izkoristili ter preusmerili podatke NMEA iz sprejemnika GPS na serijska vrata razvojne platforme, nato pa jih prebrali iz modula s pomočjo knjižnice za branje in pisanje podatkov na serijski vmesnik SER. Slednji je bolj podrobno opisan v poglavju 5.1.1.

5 Programska oprema

Delovanje sistema omogoča več-nivojska programska oprema, katere glavne naloge so:

- pridobitev podatkov o položaju,
- storitev virtualne ograje (GeoFence),
- komunikacija z oddaljenim strežnikom,
- spletna aplikacija za sprejem podatkov,
- podatkovna baza,
- predstavitev informacij.

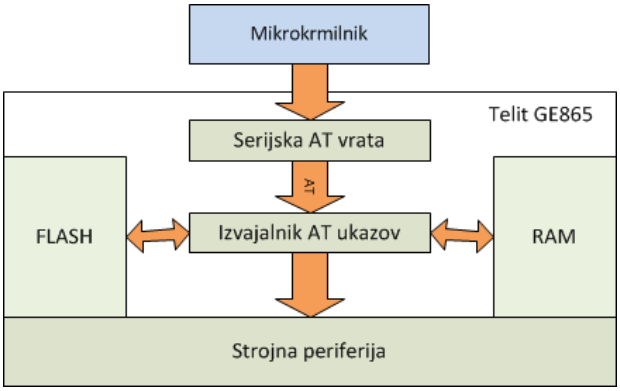
Procesiranje podatkov GPS, algoritem virtualne ograje ter komunikacija z oddaljenim strežnikom so naloge namenske naprave. Razvoj te funkcionalnosti je bil izveden v integriranem razvojnem okolju Telit PythonWin 1.5.2 ter testiran na modulu GSM Telit GE 865 Quad. Ostali nivoji sistema se izvajajo na oddaljenem strežniku in so bili realizirani v ogrodju .NET, kar pomeni, da smo za razvoj uporabili orodje Visual Studio 2010 ter jezik C#. Za podatkovno bazo smo izbrali SQL Server Express. Sprejem podatkov pa poteka preko spletne storitve izdelane na podlagi REST modela, ki smo ga izdelali s pomočjo metodologije WCF.

5.1 Namenska naprava

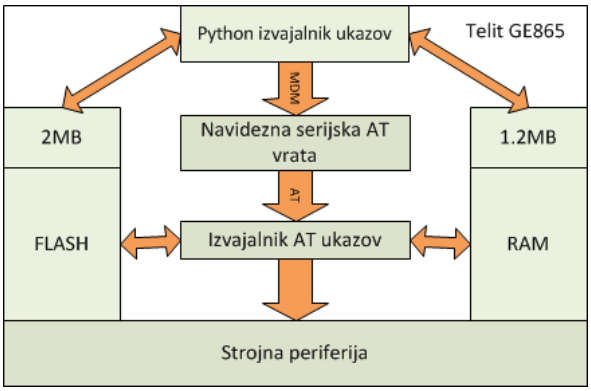
Klasična arhitektura (slika 13) takšne naprave bi normalno vsebovala mikrokrmilnik, ki preko zaporednih vrat upravlja z GSM modulom tako, da mu preko vmesnika AT pošilja ukaze. Easy Python Script extension (slika 14) omogoča, da odpravimo mikrokrmilnik in s tem poenostavimo razvoj naprave. Glavne značilnosti modula so:

- Pythonov prevajalnik verzije 1.5.2,
- datotečni sistem z 2 MB spomina,
- 1.2 MB delovnega spomina ,
- velikost spremenljivk do 16 KB.

Naenkrat se lahko izvaja ena skripta, ki se izvaja z najnižjo prioriteto zato, da ne moti izvajanja GSM/GPRS operacij. Podatkovne tipe, kontrolne stavke ter ostale podprte elemente izvajalnika Python so podrobneje opredeljeni v [7].



Slika 13: Klasična arhitektura.



Slika 14: Easy Python Script extension.

Pri upravljanju modula GSM s pomočjo programskih skript napisanih v jeziku Python so najbolj pomembne knjižnice:

- MDM, MDM2 za pošiljanje ukazov AT ter sprejemanje odgovora,
- SER, SER2 za branje/pisanje iz/na serijski vmesnik,
- GPIO za upravljanje z vhodno/izhodnimi nožicami,
- MOD z zbirko uporabnih funkcij,
- IIC za uporabo vodila IIC,
- SPI za uporabo serijskega perifernega vmesnika in
- GPS[3] za delo z vgrajenim vmesnikom GPS.

Knjižnice vključimo v program z ukazom *import*.

[3] Če ima modul že vgrajeni GPS sprejemnik

5.1.1 Izvajanje kode Python

Izvorno kodo (.py) ali pa že prevedeno kodo (.pyo) je potrebno prenesti na modul ter jo omogočiti. Prenos datoteke izvedemo z ukazom AT#WSCRIPT v naslednji obliki:

AT#WSCRIPT = "PozdravljenSvet.pyo", 232 .

Prvi parameter ukaza predstavlja ime datoteke, drugi pa velikost datoteke v bajtih. Uspešnost operacije nato preverimo z ukazom AT#LSCRIPT, ki izpiše vse datoteke, ki se nahajajo na modulu, ter njihove velikosti.

V drugem koraku je potrebno izmed vseh datotek naloženih na modul, izbrati tisto, ki želimo, da se ob zagonu modula izvede. V primeru, da program sestavlja več datotek oziroma modulov, izberemo tisto datoteko, kjer se nahaja glavni del. Glavno izvršilno datoteko določimo z ukazom AT#ESCRIPT

AT#ESCRIPT = "PozdravljenSvet.pyo" .

Uspešnost operacije preverimo z ukazom AT#ESCRIPT?.

Izbrano skripto Python lahko nato izvedemo z ukazom AT#EXECSCR ali pa preko ukaza AT#STARTMODESCR, kjer določimo v katerih pogojih se skripta zažene ob ponovnem zagonu modula. Možnosti so predstavljene v tabeli .

Tabela 3 STARTMODESCR ukaz

AT#STARTMODESCR = 0	Skripta se ob ponovnem zagonu modula izvede le v primeru nizkega DTR (Data Terminal Ready) signala, torej samo takrat ko serijski vmesnik na modulu ni aktiven.
AT#STARTMODESCR = 1, xx	Skripta se izvede v primeru, da v času, ki ga določa drugi parameter, ni bil na serijski vmesnik poslan noben AT ukaz.
AT#STARTMODESCR = 2	Skripta se izvede v vsakem primeru, ne glede na signal DTR ter ukaze poslane na serijski vmesnik.

5.1.2 Python in virtualna vrata

V načinu delovanja, ko razvojno okolje priklopimo na osebni računalnik in na modulu GSM ne izvajamo skripte Python, predstavljajo fizična serijska vrata privzeti kanal za upravljanje z modulom preko ukazov AT. V trenutku, ko na modulu omogočimo izvajanje kode Python, pa pride v načinu delovanja do sledečih sprememb:

- fizična zaporedna vrata se rezervirajo za komunikacijo z zunanjimi napravami preko knjižnice SER,

- upravljanje z modulom se omogoči preko internih zaporednih virtualnih vrat, ki predstavljajo programski most med knjižnico MDM in enem izmed treh razčlenjevalnikov ukazov AT.

Knjižnica MDM je prav tako odgovorna za pošiljanje ali sprejemanje podatkov med vzpostavljeno povezavo GPRS. Kot smo že omenili razpolaga modul s tremi razčlenjevalniki ukazov AT, kar pomeni, da v primeru, ko smo s knjižnico MDM vzpostavili povezavo GPRS, upravljamo z modulom preko knjižnice MDM2, ki preko internih zaporednih virtualnih vrat vzpostavi povezavo s sekundarnim razčlenjevalnikom ukazov AT.

5.1.3 Komunikacija vmesnik Telit GE 865 Quad – sprejemnik GPS

Izhodni signal sprejemnika GPS je združljiv z komunikacijskim protokolom NMEA, ki predpisuje hitrost 4800 b/s in prenos po 8 znakov brez paritete. Začetek okvirja NMEA označuje znak "$" , konec pa znak "*", ki mu sledi heksadecimalna kontrolna vsota ter znaka CR in LF, ki označujeta novo vrsto [11].

V privzetem načinu delovanja, ko je hitrost osveževanja 1 Hz, se tako vsako sekundo na izhodu sprejemnika GPS FGPMMOPA6B pojavijo naslednji okvirji NMEA: GGA, GSA, GSV, RMC, VTG. Ime okvirja podrobneje določa njegovo vsebino, informacije znotraj okvirjev pa so ločene z vejicami. Ime vsakega okvirja ima predpono GP, kar pomeni, da okvir vsebuje navigacijske podatke. Natančno določen vrsti red informacij znotraj vsakega okvirja določa standard NMEA, definicijo strukture okvirja NMEA pa si lahko preberemo tudi v navodilih sprejemnika GPS [10].

Primer podatkov na izhodnem signalu sprejemnika GPS tekom 1 sekunde prikazuje slika Slika 15:

```
1  $GPVTG,84.71,T,,M,0.17,N,0.32,K,A*00
2  $GPGGA,082951.000,4604.3722,N,01431.1382,E,1,4,2.82,306.2,M,44.4,M,,*59
3  $GPGSA,A,3,08,17,28,26,,,,,,,,,2.98,2.82,0.97*07
4  $GPRMC,082951.000,A,4604.3722,N,01431.1382,E,0.33,84.71,140811,,,A*53
```

Slika 15: Izhodni podatki sprejemnika GPS

Najbolj pogosti okvir NMEA je okvir GPRMC (angl. Recomended Minimum Navigation Information), ki v skrajšani obliki vsebuje vse potrebne informacije za določanje položaja, hitrosti, smeri gibanja.

Sledi programska koda (slika 16), ki prepozna okvirje GPRMC, preveri njihovo veljavnost s primerjanjem podane ter izračunane kontrolne vsote ter pretvori podatke o zemljepisni dolžini/širini iz oblike po standardu NMEA, pri akterem je zemljepisna širina podana kot ddmm.mmm, zemljepisna dolžina pa kot dddmm.mmm v decimalno obliko.

```
import string

def parseGPSData(GPSData) :
    gpsSentence = GPSData.split("\r\n")
    result = {}
    result["FIX"] = 0
    for item in gpsSentence :
        if  isValid(item) == 0 :
            continue
        words = item.split(",")
        if words[0] == "$GPRMC" :
            if words[2] == "A" :
                result["GPRMC"] = parseGPRMC(words)
                result["FIX"]   = 1
            else :
                break
    return result

def parseGPRMC(GPRMCdata):
    latitude_in = float(GPRMCdata[3])
    longitude_in = float(GPRMCdata[5])

    if GPRMCdata[4] == "S" :
        latitude_in = -latitude_in
    if GPRMCdata[6] == "W" :
        longitude_in = -longitude_in;

    latitude_degrees = int(latitude_in/100)
    latitude_minutes = latitude_in - latitude_degrees*100
    longitude_degrees = int(longitude_in/100)
    longitude_minutes = longitude_in - longitude_degrees*100
    GPRMCdata[3] = latitude_degrees + (latitude_minutes/60)
    GPRMCdata[5] = longitude_degrees + (longitude_minutes/60)

    return GPRMCdata

def getCheckSum(gpsSentence) :
    checksum = 0
    gpsSentence = gpsSentence[gpsSentence.find('$') +1 : gpsSentence.rfind('*')]

    for item in gpsSentence :
        checksum = checksum ^ ord(item)

    return "%x" % checksum

def isValid(gpsSentence) :
    return   int(getCheckSum(gpsSentence),16) == int(gpsSentence[gpsSentence.rfind('*')+1:len(gpsSentence)]
```

Slika 16: Procesiranje okvirja GPRMC.

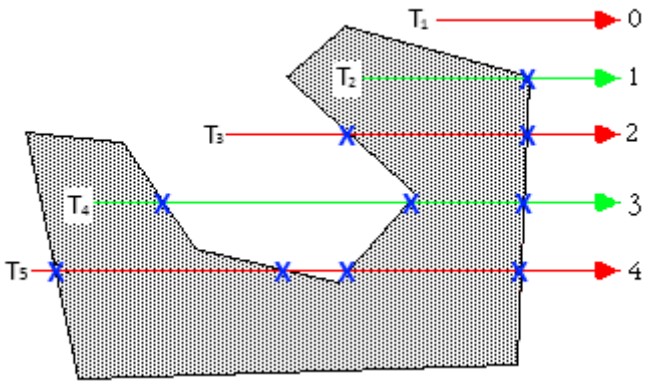
5.1.4 Procesiranje podatkov

Dodano vrednost lokacijskih sistemov predstavlja informacija o lokaciji, ki je pridobljena iz dodatno obdelanega podatka o položaju [2], ki nam ga posreduje sprejemnik GPS. Preračunavanje lahko poteka na strani namenske naprave ali pa na strani oddaljenega strežnika, kateremu se podatki posredujejo. Kje se preračunavanje odvija je odvisno od aplikacije, storitve ali virov potrebnih za procesiranje. Vsak način ima dobre in slabe strani. Lokalno procesiranje omogoča takojšnji odziv a večjo porabo virov ter manjšo elastičnost sistema, medtem ko procesiranje na strežniku omogoča večjo fleksibilnost a zahteva bolj pogosto komunikacijo med namensko napravo in oddaljenim strežnikom.

Z mislijo o prenosni napravi, ki ji želimo omogočiti čim daljšo avtonomijo baterije, smo želeli čim bolj zmanjšati komunikacijo z oddaljenim strežnikom oziroma vzpostavljanje povezave GPRS. Tako smo v programsko opremo namenske naprave vključili algoritem, ki ga intervalno izvajamo in vzpostavi povezavo s strežnikom samo v primeru, ko naprava zapusti določeno v naprej definirano območje, v nasprotnem primeru pa s klicem funkcije *MOD.powerSaving (interval)* za določeno obdobje preide v varčevalni način delovanja.

5.1.4.1 Geometrično poizvedovanje

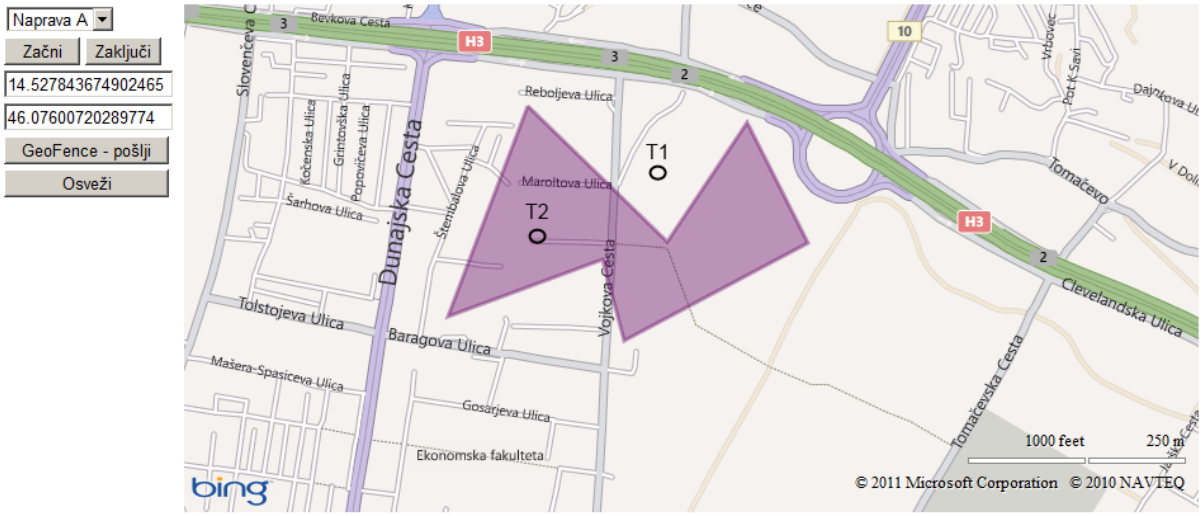
Algoritem za določanje položaja točke T(x,y) znotraj zaključenega dvo-dimenzionalnega poligona (angl. Point in Polygon, PIP) šteje presečišča linije, vzporedne z abscisno osjo, ki poteka skozi izbrano točko T(x,y), in poljubno točko izven poligona. Število presečišč linije in stranic poligona določa ali je izbrana točka T(x,y) vsebovana v poligonu ali ne. V primeru da je število presečišč liho, je točka T(x,y) vsebovana, v primeru sodega števila presečišč, pa točka leži zunaj poligona.



Slika 17: Algoritem PIP.

Slika 17 prikazuje presečišča ravnih linij s stranicami poligona. Točke T1, T3 in T5 imajo sodo število presečišč, zato ležijo zunaj poligona. Število presečišč linij skozi točke T2 in T4 pa je liho, saj točki ležita znotraj poligona.

Uporabniški vmesnik na sliki 18 prikazuje izbrano točko T1, ki se nahaja zunaj definiranega območja, ter točko T2, ki se nahaja znotraj meja večkotnika. Uporaba vmesnika je podrobneje opisana v poglavju 5.2.3, tukaj pa ga omenjamo samo zaradi konkretnih podatkov uporabljenih pri izvorni kodi predstavljeni na sliki 19, kjer prikazujemo delovanje PIP algoritma, ter njegov rezultat na izbranih točkah. Tako spletna storitev za prikaz zemljevida Bing Maps, kot sistem GPS uporabljata referenčni koordinatni sistem WGS 84 (angl. World Geodetic System), kar pomeni, da lahko uporabljamo podatke sprejemnika GPS brez dodatnih pretvorb.



Slika 18: Večkotnik ter dve izbrani točki za prikaz PIP.

```
def pip(x,y):
    c = False

    #OGLJIŠČA VEČKOTNIKA
    xp=[46.081960574760345,46.07701897958296,46.07835861236232,46.07645334708897,
        46.0787460231859,46.08157401017149,46.0787757922248]
    yp=[14.5169223564864,14.514219123561535,14.519454795558605,14.520184356410656,
        14.526427468541282,14.524367532017845,14.521663865330833]

    j = len(xp) - 1

    for i in range (0,len(xp),1):
        if ((((yp[i] <= y) and (y < yp[j])) or ((yp[j] <= y) and (y < yp[i])))) :
            if  x < (xp[j] - xp[i]) * float((y - yp[i])) / float((yp[j] - yp[i])) + xp[i]   :
                c = not c
        j = i
    return c


print pip(46.08017491893684, 14.521148881200695)  #T1 ... vrne FALSE
print pip(46.079907003802035, 14.516986092809741) #T2 ... vrne TRUE
```

Slika 19: Izvorna koda algoritma PIP.

5.1.5 Komunikacija z oddaljenim strežnikom

Namenska naprava komunicira z oddaljenim strežnikom s pomočjo storitve GPRS. Celotna komunikacija temelji na arhitekturnem stilu REST (angl. Representational State Transfer), kar pomeni, da se za prenos podatkov na/iz strežnika, tako kot v primeru spletnih brskalnikov, generirata zahtevka GET in POST.

Za uspešno vzpostavitev povezave GPRS ter pošiljanje zahtevka GET iz okolja Python, moramo modulu GSM poslati ukaze AT prikazane na sliki Slika 20:

```
1  AT+CGDCONT = 1,"IP","internet","0.0.0.0",0,0
2  AT#SCFG=1,1,300,90,600,50
3  AT#SGACT=1,1,"mobitel","internet"
4  AT#SD=1,0,80,"89.212.233.86",0,0
```

Slika 20: Vzpostavitev povezave GPRS

pri čemer so uporabniško ime, geslo ter dostopna točka del splošnih nastavitev ponudnika Telekom. V primeru uspešno izvedenega ukaza SGACT (angl. Context Activation) nam mikrokrmilnik odgovori z dodeljeno številko IP, na primer: #SGACT: 193.199.234.255. Z zadnjim ukazom definiramo protokol transportnega sloja (TCP), vrata vtičnice (80) ter cilj zahtevka. Podrobnejši opis ostalih parametrov si lahko bralec prebere v [13]. Sedaj je vse pripravljeno, da na vmesnik MDM zapišemo zahtevek GET/POST v obliki besedila, ki ga ustrezno opremimo z oznakami za pomik v novo vrsto:

GET / HTTP/1.1<cr><lf>
Host: www.google.com.com<cr><lf>
Connection: keep-alive<cr><lf>
<cr><lf>

Slika 21 prikazuje programsko kodo, ki s pomočjo pravkar opisanih ukazov preko funkcije *send_data* na oddaljeni strežnik pošlje podatke poslane preko določila "*data*", ki ga podaja URL.

```
import MDM
import MOD
import urllib

def define_connection():
    MDM.send('AT#GPRS=0\r', 2)
    MDM.receive(5)
    MDM.send('AT+CGDCONT = 1,"IP","internet","0.0.0.0",0,0\r', 2)
    MDM.receive(5)
    MDM.send('AT#SCFG=1,1,300,90,600,50\r', 2)
    MDM.receive(5)

def request_ip():
    while 1:
        MDM.send('AT#SGACT=1,1,"mobitel","internet"\r',2)
        MOD.sleep(20)
        a = MDM.receive(5)
        if a.find('SGACT') != -1:
            #print a
            return

def open_socket():
    while 1:
        MDM.send('AT#SD=1,0,80,"89.212.233.86",0,0\r', 2)
        MOD.sleep(20)
        if MDM.receive(5).find('CONNECT') != -1:
            #print 'Socket connected'
            return

def send_data(data):
    define_connection()
    request_ip()
    open_socket()
    str_to_send = 'GET / HTTP/1.0\r\nHost: 89.212.233.86/TrackingInfo?Data='
                + urllib.quote(data)
                + '\r\nConnection: keep-alive\r\n\r\n'

    MDM.send(str_to_send, 10)
```

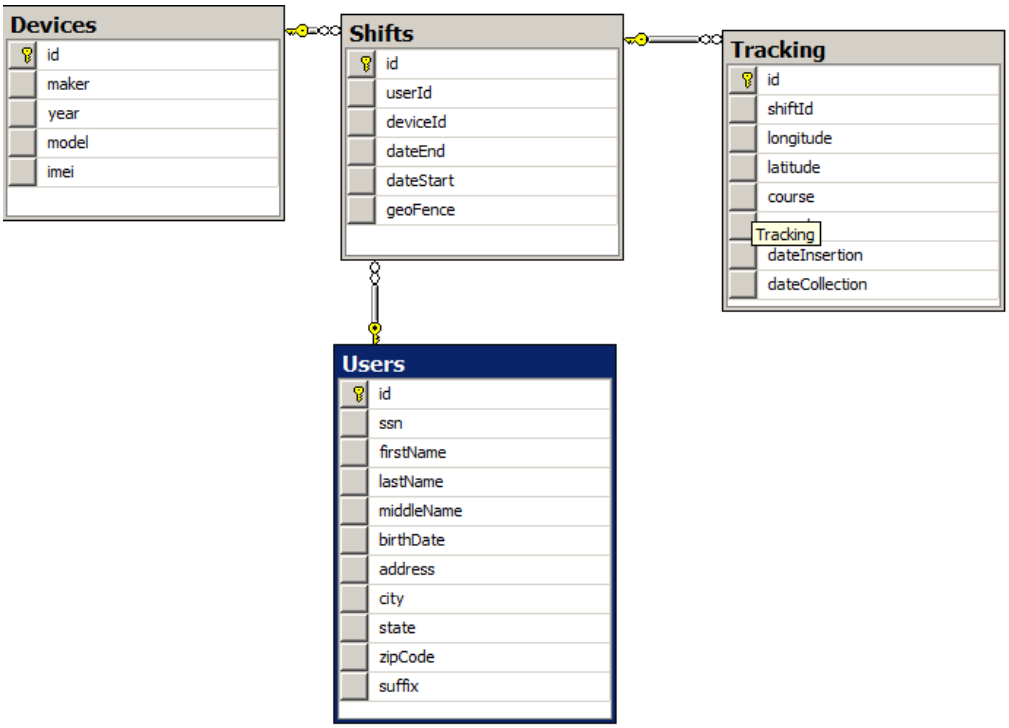
Slika 21: Vzpostavitev povezave GPRS in pošiljanje zahtevka GET.

5.2 Oddaljeni strežnik

Razvoj strežniške aplikacije je potekal v okolju Windows z nameščenim spletnim strežnikom IIS 7.5 (angl. Internet Information Services). Poleg spletnega strežnika smo namestili tudi podatkovno bazo Microsoft SQL Server Express Edition. Za preslikavo relacijskega podatkovnega modela v objektnega in avtomatično generacijo za to potrebnih poizvedb SQL smo uporabili ORM (angl. Object Relational Mapping) orodje ADO.NET Entity Framework.

5.2.1 Podatkovna baza

Fizični model podatkovne baze smo osnovali na podlagi modela predlaganega v viru [1], ki pa smo ga dodatno priredili našim potrebam. Osnovna naloga takšne zasnove podatkovne baze prikazane na sliki Slika 22 je hranjenje podatkov poslanih iz namenskih naprav. Tabela Tabela 1 prikazuje podrobne opise stolpcev podatkovne baze.



Slika 22: Fizični model podatkovne baze.

Tabela 4: Tabele v podatkovni bazi.

Tabela	Opis
Users	Seznam uporabnikov sistema.
Devices	Seznam namenskih naprav
Shifts	Vmesna tabela, ki predstavlja povezavo med uporabnikom in namensko napravo. Naprava tako ni zaklenjena na lastništvo enega uporabnika.
Tracking	Zgodovina gibanja določene naprave

Tabela *Tracking* vsebuje podatke, ki jih namenska naprava pridobi preko sprejemnika GPS ter posreduje na oddaljeni strežnik. Podrobnejši pregled ter razčlemba podatkov, ki jih pridobi sprejemnik GPS je obravnavana v poglavju 5.1.3.

Tabela 5: Stolpci tabele Tracking.

Stolpec	Opis
shiftId	Povezava z uporabnikom ter napravo.
Longitude	Zemljepisna dolžina.
Latitude	Zemljepisna širina.
Course	Smer.
Speed	Hitrost.
dateInsertion	Čas zapisa podatka na oddaljenem strežniku.
dateCollection	Čas meritve podatka na namenski napravi.

5.2.2 Sprejem podatkov

Za izgradnjo porazdeljenih aplikacij je Microsoft pripravil objektno usmerjeni pristop WCF (angl. Windows Communication Foundation). Eden izmed načinov komunikacije, ki jih podpira pristop WCF je tudi izgradnja storitev na podlagi modela REST. Kot smo že omenili, poteka klicanje oddaljene metode v našem primeru preko izgradnje zahtevkov GET/POST, ki jih posredujemo na naslov spletne aplikacije, ki se izvaja v okviru spletnega strežnika IIS.

Izgradnja storitve s pomočjo WCF poteka v 3 korakih :

1. definicija pogodb: z uporabo atributov WCF ter definicijo vmesnikov dosežemo ločitev poslovne logike ter izbranega načina komunikacije (slika Slika 24),
2. implementacija storitve: v skladu s pogodbami iz točke 1.,
3. konfiguracija: ustrezna prilagoditev datoteke app.config (slika 23), kjer definiramo končne točke, preko katerih dostopamo do storitve, določimo lokacijo storitve ter uporabljen transportni protokol (http).

```
<service name="lbisREST.DataService">
  <endpoint address="http://89.212.233.86/LBIS"
            binding="webHttpBinding"
            contract="lbisREST.ILbis" />
</service>
```

Slika 23: Konfiguracija storitve REST.

Vmesnik storitve v našem primeru sestavljata dve metodi. Metoda *GetSettings* na podlagi številke IMEI napravi vrne podatke povezane z virtualno ograjo. Podatki se vračajo v obliki JSON (angl. JavaScript Object Notation). Metoda *CreateTrackingInfo* pa skrbi za zapis podatkov o poziciji v podatkovno bazo.

```
[ServiceContract(Name = "LBIS")]
public interface ILbis
{
    [OperationContract]
    [WebGet(UriTemplate = "TrackingInfo?Data={data}")]
    void CreateTrackingInfo(string data);

    [OperationContract]
    [WebGet(UriTemplate = "ClientSettings?Imei={imei}", ResponseFormat = WebMessageFormat.Json)]
    string GetSettings(string imei);
}
```

Slika 24: Definicija pogodbe.

Vmesnik, ki definira storitev označimo z atributom *ServiceContract*, prav tako moramo z atributom *OperationContract* označiti vse metode, za katere želimo, da so dosegljive preko končne točke.

Izvorna koda na sliki 25 prikazuje implementacijo storitve po omenjeni pogodbi. To narekuje ukaz *Ilbis* v glavi definicije *DataService* razreda. Parametri URL se pri tem avtomatično dekodirajo in prenesejo v istoimenske parametre funkcije, kar je razvidno v definiciji pogodbe.

Razred *LBISEntities* predstavlja objektni model naše podatkovne baze. Tabele ter njihovi stolpci so zastopane kot istoimenski objekti .NET s pripadajočimi lastnostmi. Generiranje nove vrstice v tabeli je tako sestavljeno iz inicializacije želenega objekta, definicije vrednosti njegovih lastnosti ter ukazov *AddToImeTabele* ter *SaveChanges*. Prvi ukaz doda objekt v lokalni kontekst, drugi pa ga zapiše v podatkovno bazo v obliki nove vrstice.

```
public class DataService : ILbis
{
    public void CreateTrackingInfo(string data)
    {
        string imei = data.Split(',')[0];

        string[] nmeaData = data.Split(',');
        using (LBISEntities ctx = new LBISEntities())
        {
            var tracking            = new Tracking();
            tracking.latitude       = nmeaData[3];
            tracking.longitude      = nmeaData[5];
            tracking.speed          = nmeaData[7];
            tracking.course         = nmeaData[8];
            tracking.shiftId        = GetShiftByImei(imei).id;
            tracking.dateCollection = string2date(nmeaData[1]);
            tracking.dateInsertion  = DateTime.Now;
            ctx.AddToTracking(tracking);
            ctx.SaveChanges();
        }
    }
    public string GetSettings(string imei)
    {
        return GetShiftByImei(imei).geoFence;
    }
    public Shifts GetShiftByImei(string imei)
    {
        Shifts result = null;
        using (LBISEntities ctx = new LBISEntities())
        {
             result = ctx.Shifts.Where(s =>
                ctx.Devices.Where(d => d.imei == imei).First<Devices>().id == s.deviceId &&
                s.userId == SessionUserId &&
                DateTime.Now <= s.dateEnd).FirstOrDefault<Shifts>();
        }
        return result;
    }
    /// <summary>
    /// </summary>
    /// <param name="time">hhmmss.sss</param>
    /// <returns></returns>
    private DateTime string2date(string time)
    {
        DateTime result;
        int h = int.Parse(time.Substring(0,2));
        int m = int.Parse(time.Substring(2,2));
        int s = int.Parse(time.Substring(4,2));
        result = new DateTime(DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day, h, m, s);
        return result;
    }
}
```

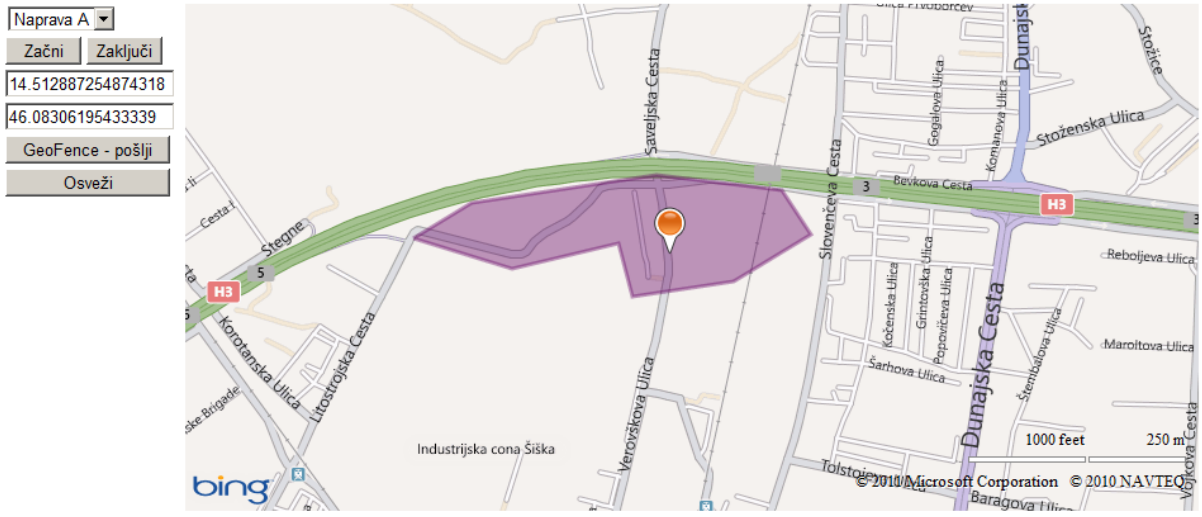
Slika 25: Implementacija storitve.

5.2.3 Uporabniški vmesnik

Spremljanje podatkov o opazovanih objektih je omogočeno preko storitve Bing Maps oziroma Bing Maps AJAX Control 7 (slika 26) gradnika Java Script, ki je uporabniku dostopen preko spletnega portala. Diploma izgradnje spletnega portala ne obravnava, saj gre za standardni sistem za upravljanje vsebine (angl. Content Management System), ki omogoča urejanje vsebine, avtentikacijo in avtorizacijo uporabnikov ter ostalo funkcionalnost za upravljanje s spletišči.

Naloga prikazovalnika je, da prikaže trenutno lokacijo vseh naprav povezanih z uporabnikom. Poleg prikaza omogoča tudi izgradnjo virtualne ograje, katere definicija se prenese na oddaljeni strežnik, od tam pa ob ponovnem zagonu na namensko napravo.

Prikazovanje ažurnih oziroma zadnjih poslanih podatkov iz namenske naprave, zahteva v ozadju sinhronizacijo podatkov med gradnikom Java Script in oddaljenim strežnikom oziroma podatkovno bazo. Kljub temu, da se v našem primeru spletni portal ter podatkovna baza nahajata na istem strežniku, smo zavoljo boljše uporabniške izkušnje, podobno kot v primeru komunikacije med namensko napravo in oddaljenim strežnikom, izdelali storitev REST, preko katere s pomočjo knjižnice Java Script JQuery osvežujemo uporabniški vmesnik, ne da bi bilo uporabniku potrebno osvežiti celotno spletno stran. Na takšen način omogočimo komunikacijo med zemljevidom Java Script, ki se izvaja v kontekstu uporabnikovega spletnega brskalnika ter podatki na našem spletnem strežniku.



Slika 26: Bing Maps Ajax Control 7.

Preizkusni uporabniški vmesnik (slika 26) omogoča uporabniku izbiro z njim povezane namenske naprave, izdelavo virtualne ograje, zemljepisno širino ter dolžino trenutne lokacije namenske naprave v obliki decimalnega zapisa ter v obliki grafičnega simbola na zemljevidu.

Podrobnejšo predstavitev delovanja kontrolnika Bing Maps si lahko bralec prebere na naslovu [14], tukaj pa si bomo pogledali funkciji Java Script, ki s pomočjo najbolj popularne knjižnice JQuery oziroma funkcije *get* (slika 27), kjer nam za klic storitev REST na strežniku ni

potrebno ročno izdelati zahtevka GET, ampak samo podamo naslov storitve ter parametre, ki jih želimo prenesti. Funkcija *sendGeoData* na strežnik pošlje informacije o vrisani virtualni ograji. Poleg samih podatkov o ogliščih, kot parameter prenesemo tudi informacije o izbrani napravi. Podatkov funkcija ne vrača, tako da je zadnji parameter klica le še metoda, ki predstavlja povratni klic ob uspešnem zaključku funkcije.

V primeru klica funkcije *updateLocation*, ki vrača objekt JSON z informacijami o zemljepisni dolžini in širini, pa je potrebno informacije še pretvoriti v objekt Java Script s pomočjo funkcije *jQuery.parseJSON*. Ker gre za asinhroni klic funkcije *get*, moramo paziti tudi, da rezultate klica uporabimo šele, ko se klic uspešno zaključi in so podatki na voljo. Zato je vsa logika potrebna za grafični prikaz vsebovana v funkciji, ki predstavlja povratni klic ob uspešnem zaključku funkcije *get*.

```
function sendGeoData() {
    $.get("http://89.212.233.86/GeoFence", { data: points.toString(), imei: $('#selectedDevice').val() },
        function (data) { alert("Data sent!"); });
}

function updateLocation() {
    $.get("http://89.212.233.86/UserData", { imei: $('#selectedDevice').val() },
        function (data) {
            var obj = jQuery.parseJSON(data);
            var loc = new Microsoft.Maps.Location(obj.lat, obj.lon);
            var pushpin = new Microsoft.Maps.Pushpin(loc, null);
            map.entities.push(pushpin);
        });
}
```

Slika 27: Generiranje zahtevka GET s knjižnico jQuery.

```
[ServiceContract(Name = "UserData")]
public interface IUserData
{
    [OperationContract]
    [WebGet(UriTemplate = "UserData?Imei={imei}", ResponseFormat = WebMessageFormat.Json)]
    string GetPosition(string imei);

    [OperationContract]
    [WebGet(UriTemplate = "GeoFence?Imei={imei}&Data={data}")]
    void SaveGeoFence(string imei, string data );
}
```

Slika 28: Pogodba za storitev REST.

```
public class UserDataService : IUserData
{

    public string GetPosition(string imei)
    {
        Tracking trackingInfo = null;
        using (var ctx = new LBISEntities())
        {
             trackingInfo = ctx.Tracking.Where(t=>ctx.Shifts.Where(s =>
                ctx.Devices.Where(d => d.imei == imei).FirstOrDefault<Devices>().id == s.id &&
                s.userId == 1 && DateTime.Now <= s.dateEnd).FirstOrDefault<Shifts>().id == t.shiftId)
                .OrderByDescending(t => t.dateCollection).FirstOrDefault<Tracking>();
        }
        JavaScriptSerializer oSerializer = new JavaScriptSerializer();
        return oSerializer.Serialize(new {lon = trackingInfo.longitude,lat = trackingInfo.latitude });
    }

    public void SaveGeoFence(string imei, string data)
    {
        using (var ctx = new LBISEntities())
        {
            var shift = ctx.Shifts.Where(s => ctx.Devices.Where(d => d.imei == imei).
                FirstOrDefault<Devices>().id == s.deviceId && s.userId == SessionUserId &&
                DateTime.Now < s.dateEnd).FirstOrDefault<Shifts>();

            shift.geoFence = data;
            ctx.SaveChanges();
        }
    }
}
```

Slika 29: Implementacija storitve.

6 Zaključek

Uporabljena programska oprema temelji na operacijskem sistemu Windows ter razvojni platformi .NET Framework 4.0. Prav tako bi lahko sistem temeljil na odprtokodnem skupku programske opreme LAMP, akronimom za katerim stojijo operacijski sistem Linux, spletni strežnik Apache, podatkovna baza MySQL ter programski jezik Pearl/PHP.

Izbira vmesnika GSM Telit GE865 Quad z mikrokrmilniškimi lastnostmi ter izvajalnikom programske kode Python je skrajšala čas potreben za izdelavo aplikacije, ki poganja namensko napravo v našem primeru. Uporaba višje nivojskega jezika ter odprava potrebe po uporabi dodatnega mikrokrmilnika sta glavna razloga, da smo se odločili za uporabo omenjenega modula. Pri pogovoru z izkušenimi načrtovalci sistemov, so le ti izrazili dvom v stabilnost naprave, ki jo ne sestavlja kateri izmed klasičnih mikrokrmilnikov uveljavljenih proizvajalcev, saj se modul brez nadzornega vezja lahko obesi, kljub omogočenim internim stražnim mehanizmom. Vendar se nam pri testiranju ni zgodilo, da bi naprava postala neodzivna. Prednost dodatnega nizkoporabnega mikrokrmilnika bi bila tudi možnost popolnega izklopa GSM vezja s čimer bi podaljšali čas delovanja naprave.

Šibko točko programske opreme, prikazane v diplomski nalogi, predstavlja osveževanje uporabniškega vmesnika, ki prikazuje informacijo o trenutni lokaciji namenske naprave. Trenutna implementacija funkcije *updateLocation* omogoča ročno osveževanje, oziroma osveževanje na podlagi zahtevka GET poslanega iz smeri uporabnika sistema, kar je seveda normalno, saj smo omenili, da koncept REST temelji na protokolu HTTP, ki narekuje, da so kakršnekoli informacije poslane iz strežnika rezultat predhodnega odjemalčevega povpraševanja. Klasični spletni model nam tako onemogoča, da bi v trenutku, ko pride do spremembe lokacije namenske naprave, strežnik o tem avtomatično obvestil odjemalca, ki bi poskrbel za takojšnjo posodobitev prikaza. Takšen način komunikacije, bi preprečil nepotrebno obremenjevanje strežnika ter naredil sistem bolj razširljiv oziroma pripravljen za delo z večjim številom uporabnikov. Ena od možnosti razširitve diplomske naloge bi tako bila uvedba tehnologije, ki omogoča potisni način komunikacije preko standardnega spletnega modela.

7 Viri, literatura

[1] M. A. Labrador, A.J. Pérez, P. M. Wightman: Location-Based Information Systems Developing Real-Time Tracking Applications, Chapman and Hall, 2010

[2] T. Nedeljko: Pregled in tehnološka zasnova lokacijsko podprtih storitev na mobilnih napravah, diplomsko delo, Maribor: Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2011

[3] Location Based Services Usage and Perceptions Survey Presentation, dostopno na: http://www.microsoft.com/download/en/details.aspx?id=3250, jan. 2011

[4] A. Brimicombe, C. Li: Location-Based Services and Geo-Information Engineering, West Sussex: Wiley-Blackwell, 2009

[5] P. P. Prešeren, B. Stopar: Izračun položaja GPS-satelita iz podatkov oddanih efemerid, Geodetski vestnik 48, 2004, str.151 – 166.

[6] EVK2 Drivers Installation Guide, dostopno na: http://www.telit.com/module/infopool/download.php?id=662

[7] Easy Script in Python, dostopno na http://www.telit.com/module/infopool/download.php?id=617

[8] E. D. Kaplan, C. J. Hegarty: Understanding GPS: Principles and Applications, Second Edition, Norwood, Artech House, 2005.

[9] A. K. Maini, V. Agrawal: Satellite Technology Principles And Applications,John Wiley & Sons, West Sussex, 2011

[10] 66-channel GPS Engine Board Antenna Module with MTK Chipset FGPMMOPA6B, dostopno na: http://www.svet-el.si/download/FGPMMOPA6B.pdf

[11] M. Vidmar, Uporaba modulov za GPS sprejemnike, Svet elektronike, 12 (12), 30-38, 2005

[12] Telit EVK2 User Guide, dostopno na: http://www.telit.com/module/infopool/download.php?id=554

[13] AT Commands Reference Guide, dostopno na: http://www.telit.com/module/infopool/download.php?id=542

[14] Bing Maps AJAX Control 7.0 ISDK, dostopno na: http://www.bingmapsportal.com/isdk/ajaxv7#CreateMap1