Sveučilište u Zagrebu

Geodetski fakultet

Frane Glasinović

Android aplikacija kao GNSS kontroler

Diplomski rad

6. rujna 2012.

Sažetak

GNSS RTK uređaj sastoji se od dvije komponente, antene i kontrolera. GNSS antena ima ulogu prijamnika koji obrađuje signale satelita dok kontroler je taj koji obrađuje podatke poslane iz antene te ih "pretvara" u informacije pogodne za daljnju obradu.

Što kada bismo ulogu kontrolera koji rade specifično samo sa antenama istog proizvođača zamjenili Android aplikacijom na mobilnim ili tablet računalima? S ovakvim pristupom imali bismo jedan uređaj koji bi zamjenio kontroler koji bi radio sa svim GNSS antenama.

Tehnologija koja se koristi za komunikaciju između antene i kontrolera, a i sa referentnim sustavom za korekcije, odavno postoji na mobilnim uređajima koji mogu stati u bok sa kontrolerima što se tiče veličine, snage za složenije izračune a i pohrane ogromne količine podataka.

Softver je taj koji bi uz pomoć komponenti mobilnih uređaja komunicirao sa GNSS antenama i referentnim sustavom za dobivanje korekcija u svrhu točnijih mjerenja.

Ključne riječi: Android, GNSS kontroler, NTRIP, NMEA, Bluetooth

Apstract

GNSS RTK device consists of two components, antenna and controller. GNSS antenna acts as the receiver that processes satellite signals, while the controller handles the data sent from the antenna and "transforms" it into information suitable for further processing.

What if we would replace controllers that work specifically with antennas of the one manufacturer with Android apps on mobile or tablet computers? With this approach, we would have one device that would replace the controller and that would work with all GNSS antennas.

The technology used for communication between the antenna and controller, and with the reference system for corrections, has existed for a long time on mobile devices that fit with controllers in terms of size, power for complex calculations and storage for huge amounts of data.

The software would, with the help of components of mobile devices, communicate with GNSS antennas and reference system for receiving corrections for the purpose of obtaining more accurate measurements.

Keywords: Android, GNSS controller, NTRIP, NMEA, Bluetooth

Sadržaj

Popis slika

Popis tablica

1 Uvod

Otkako su mobiteli postali sve napredniji i močniji, dogodio se pravi skok u mobilnoj telefoniji. Mobiteli su počeli biti sve sličniji računalima koje koristimo u svojim domovima. Prva tvrtka koja je napravila veliki značajni korak u mobilnoj komunikaciji, bila je kompanija Apple. Lansiranjem njihovog poznatog proizvoda iPhone, mobilna telefonija kakvu smo do tada znali se potpuno promijenila. Svi su ga htjeli imati, ne samo radi izgleda već i kasnije radi mnoštva aplikacija koja su se svakim danom povećavale. Mobilni uređaj je postao minijaturno računalo na kojem su korisnici željeli imati što više aplikacija koje su sve više sličila onima za stolna računala. Imati računalo bilo gdje sa sobom, usto u bilo kojem trenutku imati pristup Internet-u, otvorilo je vrata mnogim razvijateljima aplikacija. Uvidjevši veliki potencijal u svemu tome, brzo je stigao odgovor iz Google a zvao se Android. Android nije mobilni telefon kao što je to slučaj kod iPhone, to je operacijski sustav koji se može instalirati na skoro pa svaki mobilni uređaj što mu daje veliku prednost pred iOS-om (operacijski sustav na iPhone-u) koji se može instalirati samo na uređaje proizvedene od strane Apple-a.

Dolaskom iPhone i Android uređaja na tržište 2007. godine, promijenjena je uporaba mobitela u svakodnevnom životu. Svakim danom sve više i više aplikacija stizalo je na tržište. Do današnjeg dana na tržištu postoji više 500 000 aplikacija kako za iPhone tako i za Android uređaje. Tržište mobilnih aplikacija postalo je najbrže rastuće tržište u računalnoj industriji, a kako stvari stoje, ono će samo još i više napredovati kako će uređaji svakim danom postajati sve bolji i brži.

Uvidjevši rast ovog segmenta tržišta, odjednom su svi željeli biti uključeni u taj trend. Poglavito su se tu bili zainteresirali geoinformatičari koji su uvidjeli veliki potencijal za prostorne informacije koje su se već do sada veli-

kom brzinom raširile u sve proizvode. Kako geoinformatika po svojoj prirodi ima jako raširenо područje djelovanja, od daljinskih istraživanja, satelitske navigacije pa sve do hidrografskog mjerenja, u svim tim područjima postoji mjesto na kojem možemo dodatno obogatiti mobilnom aplikacijom koja bi nam uvelike mogla pomoći zbog njezinih velikih mogućnosti. Na nekim područjima, jedna jedina aplikacija bi mogla zamijeniti cijeli set uređaja koji bi se sada mogao odvijati na mobilnom uređaju. To je jedna stvarno koju moramo prihvatiti i koja bi se mogla dogoditi danas sutra. Imajući takve stvari u vidu, potrebno se što više prilagoditi tržištu kako bi bili u korak sa konkurencijom i mogli što bolje ponuditi proizvode na tržištu te približiti se sve većem broju korisnika.

2 Android

Google Android je prvi otvoreni operacijski sustav za mobilne uređaje (mobilni telefoni, tableti, netbook računala, Google TV) pokrenut od strane Google Inc.® i vođen od strane Open Handset Alliance - grupe koja danas broji preko 80 tehnoloških kompanija između kojih se nalaze T-Mobile, HTC, Intel, Motorola, Qualcomm, i drugi, čiji je cilj ubrzati inovacije na području mobilnih operacijskih sustava, a samim time ponuditi krajnjim kupcima bogatije, jeftinije i bolje iskustvo korištenja. Android je modularan i prilagodljiv pa

Slika 1: Android logo. Google [2012]

tako postoje slučajevi njegovog prenošenja (portanja) na razne uređaje kao što su čitači elektronskih knjiga, mobilni telefoni, prijenosnici, te multimedijski playeri.

2.1 Povijest

Android Inc. su osnovali Andy Rubin, Rich Miner, Nick Sears i Chris White u listopadu 2003. godine kako bi razvijali programe za pametne mobilne uređaje koji bi uzimali u obzir korisničke postavke te njegovu lokaciju. Nakon dvije godine gotovo tajnog rada (jedino što je bilo poznato bilo je da se radi o softveru za mobitele), Google je odlučio kupiti Android te počinju spekulacije o ulasku Googlea na tržište pametnih telefona. Osnivači i ključni programeri, osnaženi Googleovim programerima, na tržište donose mobilnu platformu temeljenu na linuxovom kernelu koja bi trebala biti potpuno prilagodljiva zahtjevima korisnika.

U studenome 2007. godine osnovana je OHA[1] s ciljem stvaranja javnog standarda za mobilne uređaje. Glavni inicijator i ovoga puta bio je Google koji je okupio 34 tvrtke iz različitih domena mobilne industrije poput proizvođača mobilnih telefona, programera aplikacija, mobilnih operatera i sličnih.

Istoga dana, 5. studenog 2007. godine, OHA otkriva mobilnu platformu otvorenog koda baziranu na Linux kernelu – Android. Ovo je prvo javno predstavljanje Androida kao operativnog sustava, a prvi komercijalni uređaj u koji je bio ugrađen Android OS bio je T-Mobile G1, tajvanskog proizvođača pametnih telefona HTC (poznat i pod nazivom HTC Dream). Osnivanjem ovog saveza, Android je bačen u utrku sa ostalim mobilnim platformama na tržištu: iOS (Apple), Windows Phone (Microsoft), Symbian (Nokia, Sony Ericsson), Palm (HP), Bada (Samsung).

[1] Open Handset Alliance

2.2 Značajke

Android platforma je prilagođena za uporabu na uređajima sa većim zaslonima poput pametnih telefona. Za pohranu podataka koristi relacijsku bazu podataka SQLite koja nije pohranjena u Androidu kao zasebni proces, već je integrirana u samu aplikaciju. Što se tiče povezivanja sa drugim uređajima, koriste se standardne tehnologije koje dolaze skoro sa svakim mobilnim uređajem, GSM/EDGE/CDMA, UMTS, Bluetooth, WiFi, LTE[2] i NFC[3]. Bluetooth podržava sučelje za upravljanje drugim uređajima (primjerice televizija, radio) te prijenos audio zapisa sa jednog uređaja na drugi. Od Android verzije 3.0 postoji podrška za spajanje uređaja za upravljanje (tipkovnica, miš, igraće palice), dok na prijašnjim verzijama podrška je bila napravljena od samih proizvođača takvih uređaja. Za prijenos poruka koriste se SMS i MMS servisi sa podržanim prikazom razgovora. Internet preglednik je temeljen na WebKit-ovom engine-u uparenom sa Googleovim Chrome V8 JavaScript engine.

Iako je većina aplikacija pisana u Java programskom jeziku, na Android uređajima ne postoji Java Virtual Machine pa tako nije moguće izvršavati Java byte kod. Za pokretanje Java aplikacija, Android koristi Dalvik virtualni stroj.

Od podrške za dodatne hardvere, Android ima ugrađenu podršku za ekran osjetljiv na dodir, GPS, akcelerometar, žiroskop, magnetometre, igraće palice, senzore osjetljive na dodir i blizinu, termometar i grafičku 3D akceleraciju. Također postoji podrška za multi-touch.

[2]Long Term Evolution, 4G
[3]Near field communication

2.3 Arhitektura

Android je baziran na Linux 2.6 kernelu napisanom u C/C++ programskom jeziku. Obzirom na otvorenost koda, aplikacije putem posrednika (eng. middleware) imaju mogućnost komuniciranja i pokretanja drugih aplikacija primjerice za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Iako su C i C++ programski jezici korišteni za radno okružje (eng. framework), većina aplikacija pisana je u Java programskom jeziku koristeći Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u C/C++ programskom jeziku, no tada se koristi Android Native Code Development Kit (NDK). Ovakvim postupkom omogućuje se bolje raspolaganje resursima i korištenje knjižnica iz kernela i radnog okružja. Ovakvim postupkom aplikacije se ubrzavaju i do 10 puta, no pisanje samog koda je složenije.

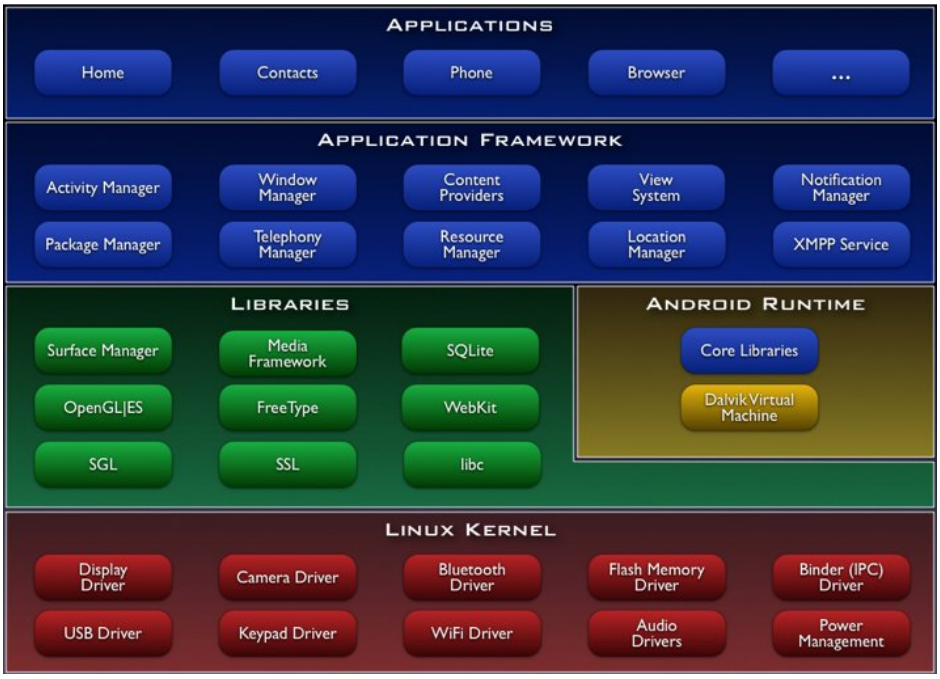
Na dnu stoga (Slika 2) nalazi se Linux 2.6 kernel koji sadrži upravljačke programe od kojih su najvažniji programi za međuprocesnu komunikaciju[4] koji služi za izmjenu podataka između različitih procesa ili niti unutar istog procesa te upravljački programi za upravljanje napajanjem (Power Managment).

Iznad kernela nalaze se knjižnice koje su pisane u C/C++ programskom jeziku:

- Surface Manager – knjižnica koja nadzire renderiranje grafičkog sučelja
- OpenGL | ES – knjižnica za hardversku 3D akceleraciju (ukoliko je moguća) te za visoko optimiziranu 3D softversku rasterizaciju
- SGL – 2D knjižnica korištena za većinu aplikacija

[4]IPC - Inter-process communication

- Media Framework – knjižnica temeljena na OpenCORE koja podržava snimanje i reproduciranje poznatih audio/video formata
- FreeType – knjižnica namijenjena renderiranju fontova
- SSL (Secure Sockets Layer) - knjižnica za sigurnosnu komunikaciju putem interneta
- SQLite – knjižnica za upravljanje bazama podataka dostupna svim aplikacijama
- WebKit – engine za web preglednike
- Sistemska C knjižnica prilagođena za ugradbene sustave bazirane na Linux OS-u



Slika 2: Arhitektura Android operacijskog sustava

Slijedi Android Runtime odnosno sloj koji služi pokretanju aplikacija. Sastoji se od dvije važne komponente. Prva su tzv. "Core libraries" odnosno knjižnice koje sadrže većinu jezgrenih knjižnica programskog jezika Java. Druga komponenta je Dalvik Virtual Machine koji pokreće aplikacije kao zasebne procese odnosno kao instance virtualnog stroja.

Nakon knjižnica dolazi aplikacijski okvir[5] koji se sastoji od mehanizama koji pomažu pisanje aplikacija. Applikacijski okvir dozvoljava upotrebu svih API-ja[6] koji su korišteni za bazne aplikacije. Tako je omogućeno upravljanje programskim paketima, aktivnostima aplikacije (odnosi se na životni ciklus aplikacije), pozivima, prozorima, resursima (pohrana komponenti aplikacija koje nisu sami kôd, primjerice slike), korištenje podataka od više različitih aplikacija, dohvaćanje i korištenje trenutne lokacije korisnika, prikaz obavijesti te baza pogleda i objekata koji mogu biti korišteni za dizajn aplikacije.

2.4 Google Play

Google Play je Googleov online dućan aplikacija. Predstavljen je u kolovozu 2008. godine, a njegovo korištenje počinje od listopada 2008. godine. Pojava prvih komercijalnih aplikacija započinje od strane britanskih i američkih programera od veljače 2009. U ožujku 2009. godine trgovina broji oko 2 300 aplikacija, a do kraja godine ta brojka raste do 20 000. Također, popis zemalja sa mogućnošću objavljivanja komercijalnih aplikacija doseže brojku 29. Do kraja 2010. godine broj aplikacija raste na 100 000 sa gotovo milijardu skinutih aplikacija. U sljedećih pola godine zabilježen je još rapidniji rast aplikacija kojih danas u trgovini aplikacija ima prema službenim procjenama preko 250 000 sa preko 3 milijarde skinutih aplikacija, dok se ukupan broj

[5]eng. Application Framework
[6]Application Programming Interface

Android aplikacija, uključujući neslužbena "prodajna" mjesta kreće oko 530 000, od čega preko polovice čine besplatne aplikacije (57%).

Za objavljivanje aplikacija programeri moraju posjedovati Google korisnički račun te kod prve prijave potrebno je uplatiti iznos od 25 američkih dolara na ime registracije. Google na stranicama za programere nudi čitavi niz uputa i zadataka koje treba obaviti prije objavljivanja same aplikacije. Prodaja komercijalnih aplikacija zasada je dostupna iz 29 zemalja u 13 različitih valuta. Cijene aplikacija kreću se od 99 centi do 200 američkih dolara, a prodavatelji dobivaju 70% cijene aplikacija. Sa ostalih 30% financiraju se pružatelji usluga za online plaćanje, dok Google ne uzima svoj udio od prodaje aplikacija.

3 Android programiranje

3.1 Programski jezik Java

Java je objektno-orjentirani programski jezik razvijen u timu predvođenim James Gosling-om u kompaniji Sun Microsystems početkom 1990-tih. Ideja je bila da se stvori programski jezik koji bi bio nezavisan od operativnog sistema, baziran na C++-u, ali sa pojednostavljenom sintaksom i pojednostavljenom kontrolom memorije.

Slika 3: Java logo, Java [2010]

Činjenica koja nam otežava jednostavno definiranje ovog programskog jezika je ta da je Java u biti puno

različitih stvari. Osim toga, pravi potencijal Jave u mnogome ovisi o kojoj Javi u biti govorimo.

Za Javu danas možemo reći da je:

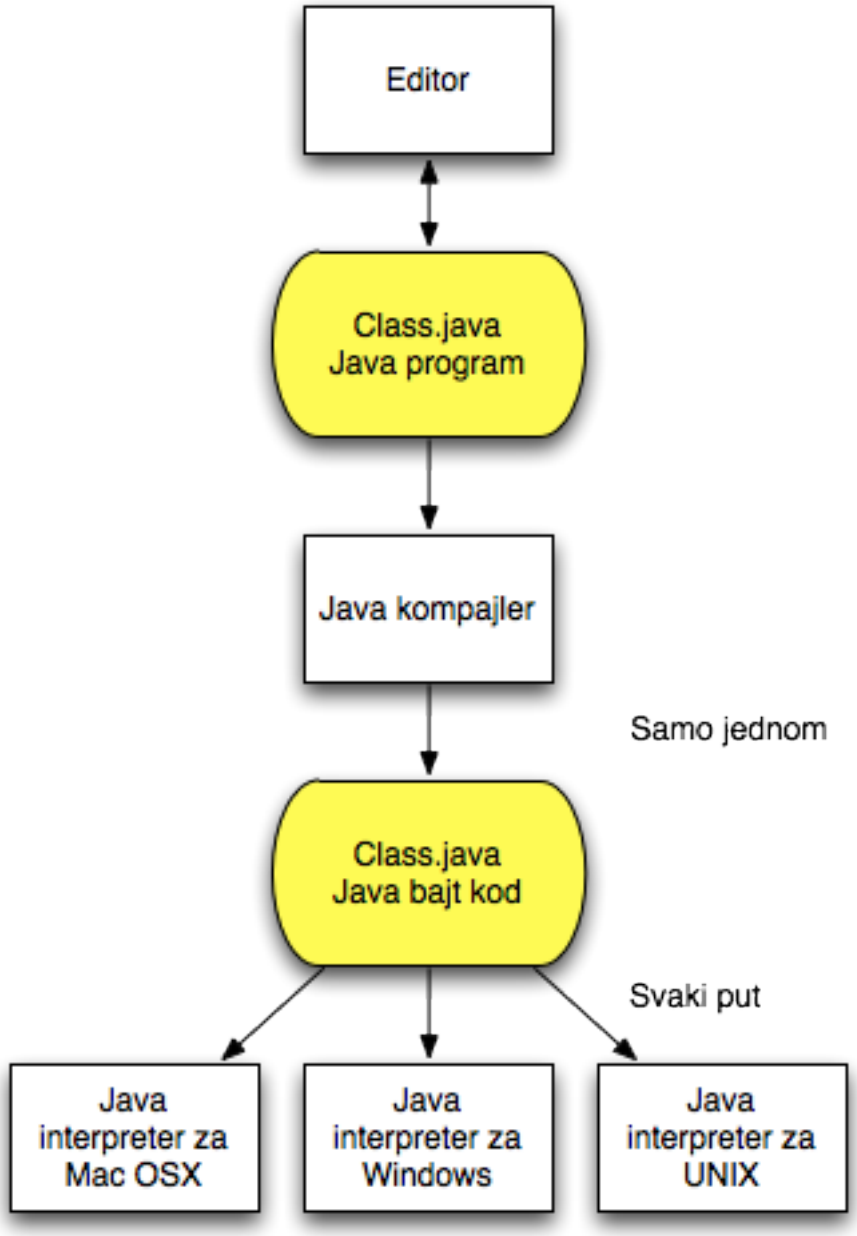
- specifikacija programskog jezika i standardni zbir klasa
- implementacija navedenog programskog jezika i njegovih pratećih datoteka (eng. libraries) u okolici prevođenja i izvođenja (eng. compile and run time enviroment) za izradu i izvršavanje aplikacija
- implementacija navedenog programskog jezika kao podskup ugrađenog koda u HTML stranicama (applet)
- implementacija navedenog programskog jezika kao dodatak animaciji i interakciji kod 3D objekata i scena (VRML 2.0)

3.1.1 Java Virtual Machine - JVM

Projektanti Jave su se odlučili na korištenje kombinacije kompilacije i interpretiranja. Programi pisani u Javi se prevode u strojni jezik, ali u strojni jezik računala koji zapravo ne postoji. Ovo takozvano prividno (eng. virtual) računalo se zove *Java Virtual Machine*. Strojni jezik za Java Virtual Machine se zove Java bytecode. Nema razloga zbog kojega Java bajt kod ne bi mogao biti korišten kao strojni jezik i nekog stvarnog računala, osim ovog prividnog. Zapravo, Sun Mycrosystems, začetnik Jave, razvio je CPU-ove koji izvršavaju Java bajt kod kao svoj strojni jezik.

Ipak, jedna od glavnih prednosti Jave je da zapravo može biti korištena na bilo kojem računalu. Sve što je na tom računalu potrebno je interpreter za Java bajt kod. Takav interpreter oponaša Java virtual machine na isti način kao što prividno računalo oponaša osobno računalo.

Naravno, Java interpreter je potreban za svaku vrstu računala, ali nakon



Slika 4: Slijed jednog java programa

što računalo dobije Java bajt kod interpreter, može izvršavati bilo koji Java bajt kod program. A isti taj Java bajt kod program može biti izvršen na bilo kojem računalu koje ima takav interpreter. Ovo je jedna od glavnih osobina Jave: isti kompajlirani program se može izvršavati na više različitih vrsta računala.

3.1.2 Osnovni Java program

Program je niz naredbi koje računalo izvršava da bi obavilo određenu zadaću. Da bi računalo moglo izvršavati naredbe, one moraju biti pisane na računalu na razumljiv način, tj. u programskom jeziku. Programski jezici se razlikuju od ljudskog po svojoj jasnoći i strogosti što je u programu dozvoljeno, a što nije. Pravila koja određuju što je dozvoljeno zove se sintaksa jezika. Sintaksna pravila određuju osnovni rječnik jezika i način na koji se programi mogu stvarati koristeći petlje, grananja i podprograme. Sintaksno ispravan program je onaj koji je moguće kompilirati ili izvršiti. Programi koji sadržavaju sintaksne greške će biti odbačeni uz poruku o grešci. Dakle, za postati uspješan programer potrebno je detaljno poznavati sintaksu korištenog programskog jezika.

Primjer jednostavnog "Hello World" programa napisanog u Java programskom jeziku:

```
1 public class HelloWorldApp {
2         public static void main(String[] args) {
3                 System.out.println("Hello World!");
4         }
5 }
```

Kada pokrenete ovaj program poruka "Hello world!" (bez navodnika) će biti ispisana na ekranu.

3.2 Android razvojni alati

Razvojni alati (eng. development kit) su skup alata koji omogućuju kreiranje aplikacija za određene skupine programa, programskog okruženja (eng. software framework), hardvera, operacijski sustava i sličnih platformi.

Razvojni alati se najčešće sastoje od skupine gotovih programskih rješenja koji su u mogućnosti komunicirati sa specifičnim programom ili platformom, ali uz to dolazi i hrpa alata za lakše otklanjanje grešaka u programu, dizajniranje sučelja te drugih sličnih alata koja pomažu programerima u pisanju što kvalitetnijih aplikacija.

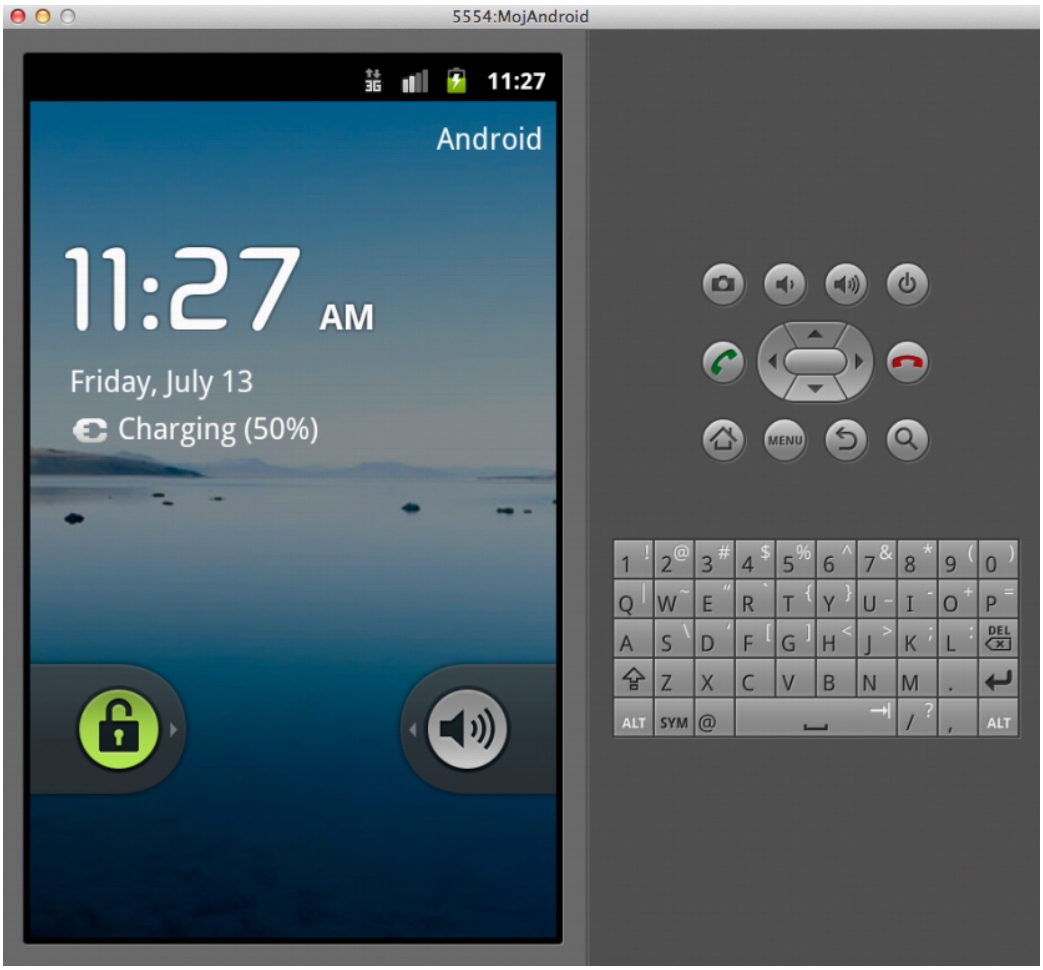
3.2.1 Android SDK

Android SDK[7] uključuje opsežan skup alata za programiranje aplikacija na Android mobilnoj platformi. U alate je uključen programa za uklanjanje grešaka (eng. debugger), skup biblioteka (eng. libraries), emulator (slika 5), dokumentacija, primjeri kodova te dokumentacija za učenje. Android SDK radi na svim operacijskim sustavima, Linuxu (svim vrstama distrubicija), Mac OS X 10.5.9 pa nadalje te Windowsima XP na više. Razvojno okruženje koje Google preporučuje za programiranje na Android-u te ga službeno i podržava je Eclipse (slika 6). Za Eclipse je Google razvio posebni dodatak (eng. plugin) koji omogućuje lakše kreiranje projekata te izradu i manipuliranje emulatorima. S ovim potezom programeri nisu ograničeni programiranjem samo u Eclipse razvojnom okruženju. Pisanje aplikacija za Android moguće je u bilo kojem tekstualnom i XML editoru, te kompajliranjem izvornog koda u komandnom okruženju.

Praktična strana Android SDK-a je što možemo birati verziju operativnog sustava te tako ciljati na veći broj korisnika koji ima i niže verzije softvera,

[7]Software Developmet Kit

sve te mogućnosti mogu se testirati u emulatoru tako da nije potrebno imati mobitele sa različitim verzijama sustava radi testiranja. Jednom kada se iz-



Slika 5: Android emulator

vorni kod kompajlira, Android SDK ga pakira u obliku .apk datoteke koja sadrži izvršni kod kojeg čita Android-ov JVM, Dalvik, te ostale datoteke potrebne za rad aplikacije kao što su slike, stilovi, teme i sl.

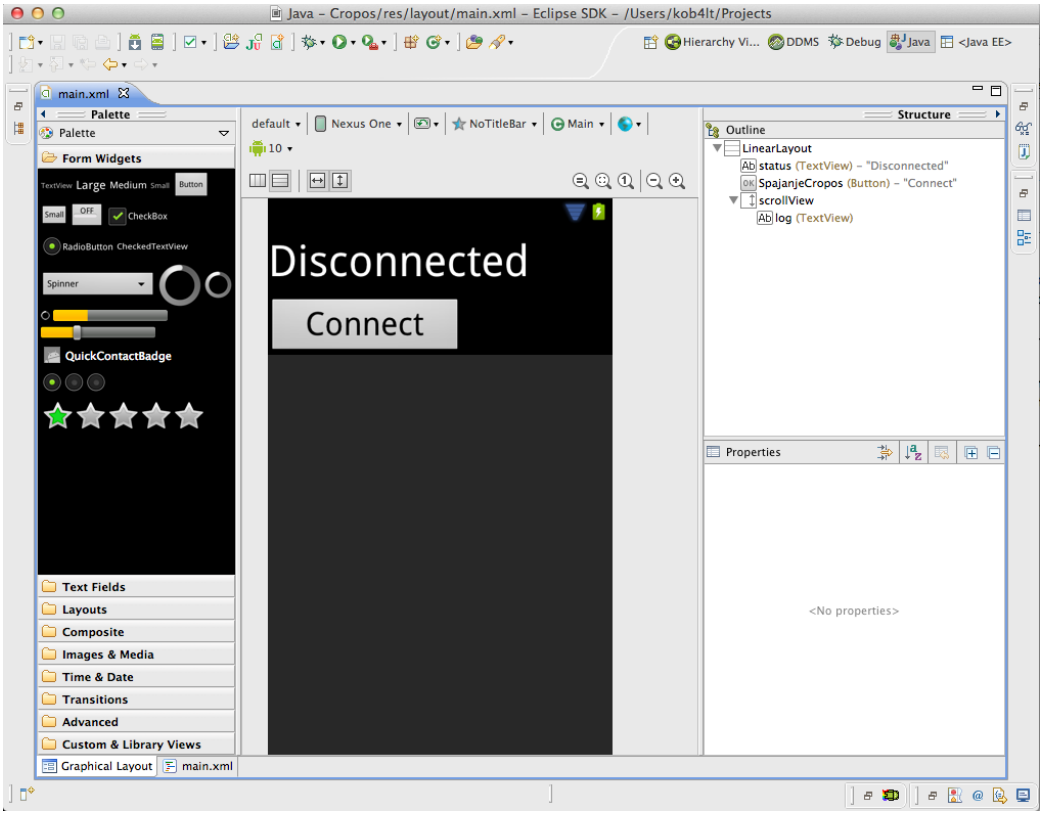
3.2.2 Android NDK

Android NDK (eng. Native Development Kit) je, kao što samo ime kaže, skup alata za programiranje Android aplikacije kompajliranih u izvorni kod koji razumije procesor, ARM ili x86. Prednost ovakvog načina programiranja je bolje upravljanje memorijskih resursima zbog direktne povezanosti sa uređajem dok je kod SDK Dalvik JVM posrednik preko kojeg se kod izvršava. Google službeno ne izdaje nikakav dodatak za Eclipse koji pomaže programiranju u NDK te smo tako osuđeni na kompajliranje, testiranje i pronalaženje grešaka preko programa u komandnoj liniji.

3.3 Razvojno okruženje - Eclipse

Eclipse je tzv. integrirano razvojno okruženje (eng. Integrated Development Environment, IDE) koje omogućava razvoj aplikacija u programskom jeziku Java. Snaga Eclipse programskog rješenja nalazi se u njegovoj proširivosti. Različiti dodaci mogu potpuno prilagoditi sučelje prema vrsti projekta. Dodaci omogućavaju da Eclipse postane razvojno rješenje ne samo za Javu, već i za Adu, C, C++, COBOL, Haskell, Perl, PHP, Python, R, Ruby i mnoge druge programske jezike.

Google je prepoznao tu prilagodljivost Eclipse-a te je baš zbog toga odabrao Eclipse kao službeno okruženje za izradu aplikacija baziranih na Android operacijskom sustavu.

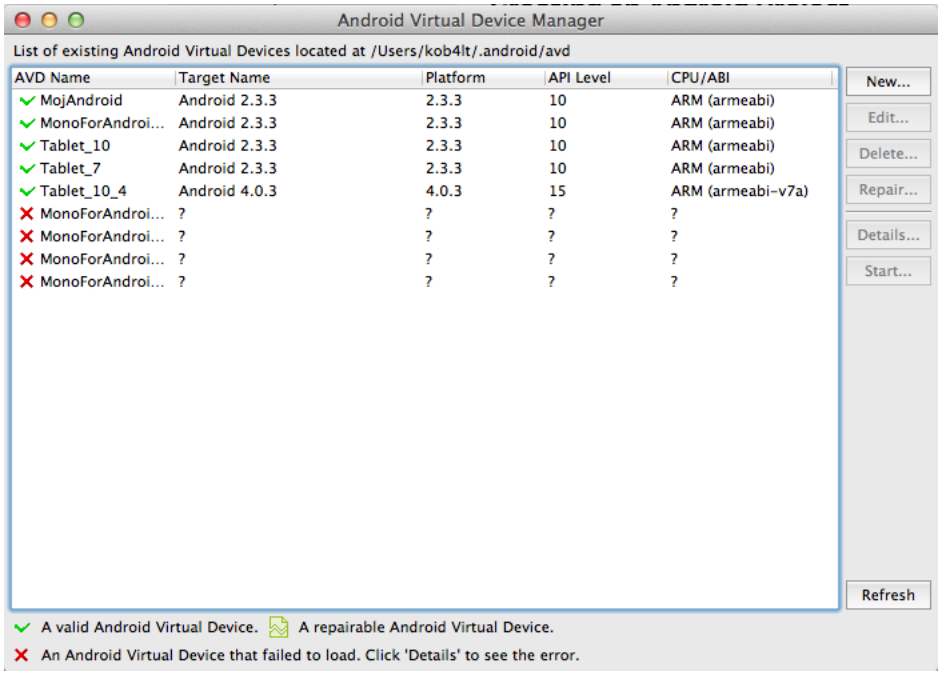


Slika 6: Razvojno okruženje Eclipse

3.4 Android projekt unutar Eclipse-a

Kao što smo prethodno rekli, Google službeno podržava i razvija alate za razvijanje Android programa unutar razvojnog okruženja Eclipse. Prije nego li krenemo izrađivati aplikacije za Android, potrebno je sa Internet-a skinuti službeni razvojni paket Android SDK te ADT (Android Development Tools) alate za Eclipse kao nadogradnju programa. Osim samih biblioteka i programa za kompajliranje izvornog koda Android projekata, u ADT-u dolaze dva alata: AVD (Android Virtual Device) i Android SDK Manager.

AVD je program koji služi kako bi mogli kreirati virtualne Android uređaje. Pogodnost ovog programa je testiranje aplikacija na različitim virtualnim uređajima koji mogu imati različite postavke, verzije operativnog sustava, veličine ekrana, memorije, brzine procesora itd. (slika 7). Kada ne

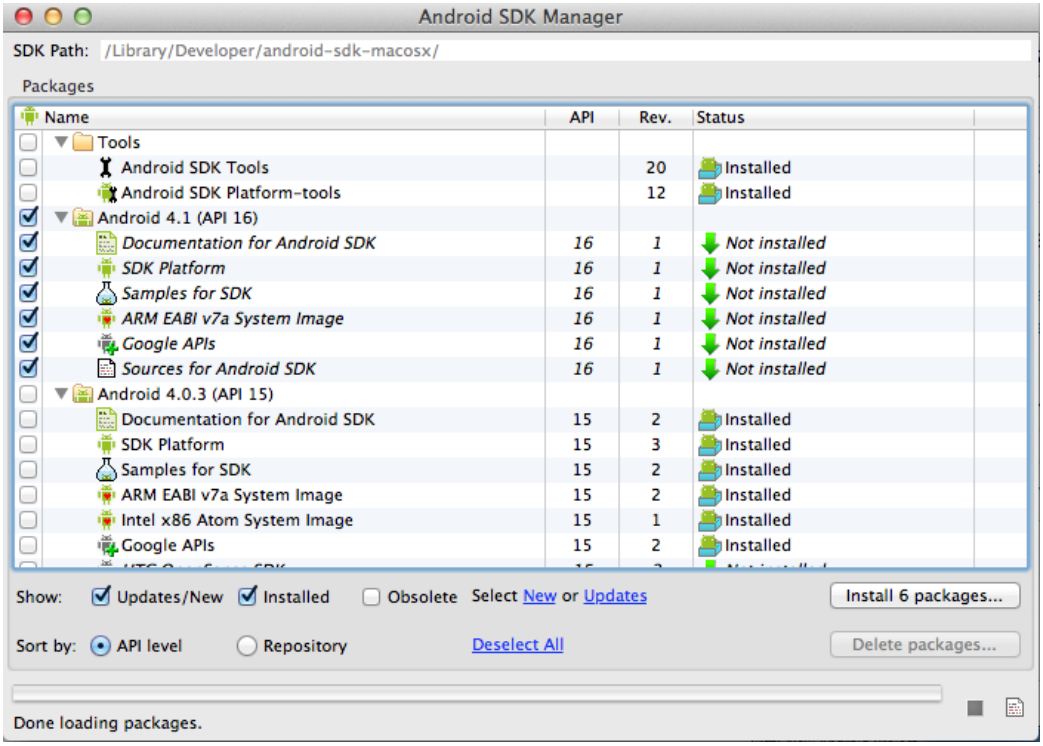


Slika 7: Android Virtual Device Manager

bi smo imali AVD, u svrhu testiranja morali bi smo imati hrpu fizičkih mo-

bilnih uređaja koji bi nam služili za testiranje aplikacije. Jedini nedostatak virtualnih uređaja je taj što oni ne predstavljaju stvarni prikaz brzine aplikacije koja se vrti na virtualnom uređaju i na fizičkom mobilnom uređaju. Osjećaj korištenja aplikacije na stvarnom uređaju nikada ne može biti isti kao i na virtualnom uređaju.

Android SDK Manager je kao što samo ime kaže, upravljački program preko kojeg instaliravamo nove verzije softvera, skidamo zakrpe ili brišemo dijelove koji nam nisu potrebni (slika 8). Imajući Android SDK paket instali-



Slika 8: Android SDK Manager

ran na računalo, možemo krenuti programirati za Android. U paketu dolaze i predlošci za kreiranje Android projekta koji nam omogućavaju lakše kreiranje svih potrebnih datoteka koji su potrebni za izradu Android aplikacija.

Praktična stvar kod programiranja za Android uređaj je ta što program

kojeg smo prethodno kompajlirali možemo direktno instalirati na sami uređaj te odmah ga isprobati. Jednom kompajliran program možemo bez ograničenja instaliravati na druge uređaje, što kod drugih mobilnih platformi nije moguće. Da bi smo instalirali program na iPhone ili Windows Phone uređaj, potrebno je aplikaciju prvo postaviti u trgovinu (eng. market) sa ostalim aplikacijama te prethodno registrirati se i platiti članarinu u određenom iznosu.

3.4.1 Struktura datoteka

Svaki Android projekt sadržava osnovni set direktorija i datoteka potrebnih za rad aplikacije. Osnovna struktura projekta čine ovi direktoriji:

- src
- gen
- Android *verzija*
- assets
- res
  - drawable-*
  - layout
  - values

Ovo nije potpuna lista direktorija koja su moguća u jednom Android projektu, ali su standardna koja dolaze sa predloškom kada kreiramo novi projekt u Eclipse-u. Postoje još par direktorija koji su skriveni od strane programera. To su direktoriji: *bin, libs i referirane datoteke.* Referirane datoteke se prikažu tek kada sa projektom spojimo neke vanjske poveznice te koristimo

njihov API u projektu. Postoji još dvije datoteke koje treba spomenuti, Android.manifest i default.properties. Krenimo redom.

Src

Direktorij u kojem se nalaze izvorni kodovi pisani u Javi potrebni za aplikaciju.

Gen

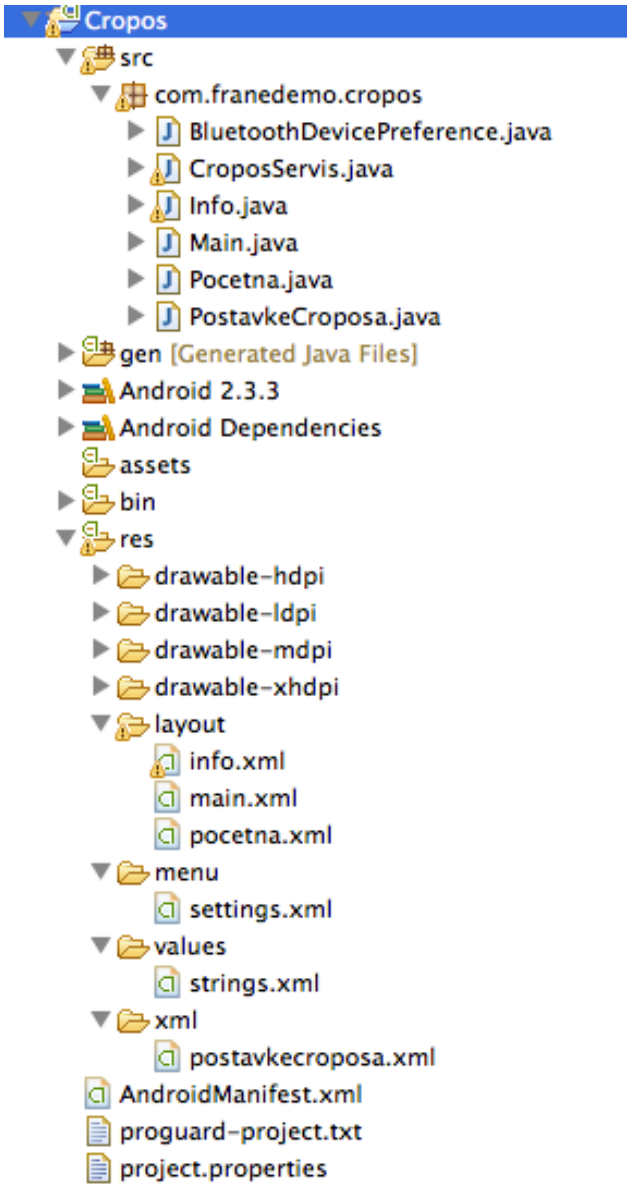
Ovaj direktorij sadrži Java datoteke generirane od strane ADT-a. Datoteke sadrže reference na različite resurse, od kontrola do zvučnih i slikovnih datoteka. Direktorij sadrži specijalnu klasu "R" koja sadrži sve te reference na datoteke. Uzmimo kao primjer jednu sliku koja se nalazi u *drawable* direktoriju. Da bi smo se referencirali na tu sliku, samo moramo pozvati referencu *R.drawable.ime_slike* u izvornom kodu. Klasa "R" se automatski generira i osviježava indeksirajući sve datoteke u projektu.

Asset

Kada tek kreiramo projekt, primjetiti ćemo da je *asset* direktorij prazan. Svrha *asset* direktorija je slična *res* direktoriju. Sadržaj tog direktorija ćemo popunit isto datotekama kao što su slike, zvukovi, fontovi i sl. Glavna razlika između ta dva direktorija je ta što se *asset* direktorij ponaša više kao datotečni sustav te ga možemo organizirati po našoj volji dok je struktura *res* direktorija strogo određena. Datoteke iz ovog direktorija dohvaćamo kao bilo koju datoteku u Java programskom jeziku.

Res

Kao što je prethodno objašnjeno, u ovaj direktorij ćemo smještati slike, zvu-



Slika 9: Struktura CROPOS projekta

kove i različite xml datoteke. Svaka specifična datoteka ima svoj pripadajući direktorij. Ako govorimo o slikama, na izbor imamo 4 direktorija. U koji od 4 direktorija ćemo koju sliku stavit ovisiti će o rezoluciji za koju je slika napravljena. Tako jedan slika može biti više prilagođena uređajima sa visokom rezolucijom te ćemo nju postaviti u *drawable-xdpi* direktorij. Najčešće se slike smještaju u *drawable-hdpi* (eng. *hgdpi, high definition dots per inch*) direktorij. Od slikovnih formata, Android podržava PNG i JPEG datoteke. Što se tiče xml datoteka, ovisno o njihovoj namjeni oni se postavljaju u odgovarajući direktorij. Datoteke koje definiraju raspored i izgled prikaza ekrana se postavljaju u direktorij *res/layout* direktorij. Datoteke koje definiraju menije smještaju se u direktorij *res/menu* itd.

AndroidManifest.xml

Ova datoteka je neizostavna u svakom Android projektu. U njoj su definirane sve postavke aplikacije, kakvu će ikonu imati, koja dopuštenja su potrebna za rad aplikacije, koje servise koristi aplikacije, prikazi koji će se koristiti i druge važne postavke za rad aplikacije.

default.properties

U njoj su najčešće opisane minimalne verzije softvera koji su potrebne kako bi uopće pokrenuli aplikaciju. Datoteka se automatski generira od strane alat te nju ne uređujemo direktno.

4 Bluetooth

Bluetooth je način bežične razmjene podataka između dva ili više uređaja. Većina današnjih modernih računala, mobitela, digitalnih kamera i audio

uređaja imaju mogućnost slanja podataka pomoću Bluetooth-a. Veza se uspostavlja putem radio valova u frekvencijskom području od 2,4 do 2,48 GHz. Zbog korištenja radio veze uređaji koji se povezuju ne moraju biti u optičkoj vidljivosti kao niti međusobno usmjereni a veza se može ostvariti u promjeru od otprilike 10 metara oko uređaja. Osnovna inačica Bluetooth-a omogućava prijenos podataka do 1 Mbit/s. Prije puštanja u uporabu proizvodi s bluetooth tehnologijom moraju biti kvalificirani i proći ispitivanje frekvencijskog međudjelovanja.

4.1 Koncept rada

Bluetooth tehnologija dijeli puno sličnosti sa ostalim tehnologijama kao što su WiFi, IRDA, USB ili Ethernet tehnologija. Svi oni dijele jednu stvar zajedničku, svi su bazirani na TCP/IP protokolu. TCP/IP protokol je osnovni protokol Internet-a. U protokolu je definirano na koji način dva uređaja komuniciraju preko različitih medija, bilo to preko kabela ili bežično. Objasnimo li princip rada TCP/IP protokola, objasnili smo veliki dio rada Bluetooth protokola.

Postoji par stavki po čemu se Bluetooth razlikuje od ostalih tehnologija:

- Kod Bluetooth-a moramo odabrati uređaj prema kojem ćemo uspostaviti komunikaciju
- Moramo otkriti kako ćemo zapravo komunicirati sa uređajem, na koji način
- Znati uspostaviti komunikaciju za slanje podataka
- Moći prihvatiti dopuštenje za primanje podataka od drugih uređaja
- Uspostaviti obostranu komunikaciju sa uređajem

Sve ove stavke ne znače da vrijede uvijek. Može se dogoditi u nekom slučaju da je već u programu pred definirano na koji će se uređaj Bluetooth spojit ili možda sigurnosna provjera za spajanje na drugi uređaj će biti preskočena zbog nekog razloga.

4.1.1 Identifikacija uređaja

Svaki Bluetooth uređaj ikad proizveden dobiva svoj identifikacijski broj na globalnoj razini u obliku 48 bitne adrese koju zovemo Bluetooth adresa ili jednostavnije, adresa uređaja. Ta adresa identična je MAC adresi mrežne kartice koja ima oblik XX:XX:XX:XX:XX:XX gdje X reprezentira 16 bitni broj. Bluetooth adresa se dodijeljuje još u proizvodnji i ona ostaje unikatna cijelo svoje vrijeme. S tim brojem se jedan uređaj razlikuje od svih drugih uređaja u svijetu te predstavlja temelj pri identifikaciji uređaja kod povezivanja.

Da bi se uopće mogla uspostaviti komunikacija između dva uređaja, Bluetooth na određeni način mora otkriti adresu od drugog uređaja. Kod TCP/IP protokola. što se tiče niže razine komunikacije, komunikacija se uspostavlja preko MAC adresa uređaja, ali to je dosta rijedak slučaj. Najčešće se komunikacija uspostavlja na višoj razini, preko IP adrese.

Kako smo do sad cijelo vrijeme pratili princip TCP/IP protokola, tako se i kod Bluetooth protokola ova komunikacija pojednostavljuje tako da se spajanje ne odvija preko adrese uređaja nego preko imena kojeg svaki korisnik može proizvoljno zadat za svoj uređaj. S time je dosta pojednostavljena identifikacija uređaja prilikom spajanja za krajnje korisnike jer ne moramo pamtit cijeli onaj set brojeva.

Svaki korisnik može proizvoljno zadati ime svom uređaju. U mnogim slučajevima, to ime neće biti unikatno kao što je to slučaj kod adrese uređaja,

te se može desiti da će kod pretraživanja uređaja u blizini biti više uređaja sa istim imenom. To može stvarati problem kod korisnika. ali u krajnjem slučaju, sva komunikacija se uvijek odvija preko Bluetooth adrese uređaja.

4.1.2 Traženje uređaja

Prije nego uopće uspostavimo bilo kakvu komunikaciju sa drugim uređajem, potrebno je uređaj naći i upariti sa našim uređajem. Traženje uređaja ponekad može biti prilično dug i iritantan posao. Vrijeme koje je potrebno da se dva uređaja koju su postavljenja jedan do drugoga nađu može trajati i do desetak sekundi. Razlog tome je što prilikom pretraživanja Bluetooth uređaj u samo jednoj sekundi promjeni više od 1000 različitih frekvencija dok ne nađen onu na kojoj osluškuje drugi uređaj.

4.2 Bluetooth protokoli

Nakon što smo otkrili uređaj i adresu na koju ćemo se spojiti, moramo odrediti na koji način ćemo spajanje uspostaviti. Kod Internet komunikacije, postoje dva načina spajanja tj. dva protokola, TCP i UDP protokoli. TCP protokol pruža nešto sigurniji način slanja podataka dok kod UDP-a to nije slučaj. Kod TCP-a možemo biti sigurni da je druga strana komunikacije primila paket dok kod UDP-a cijelo vrijeme šaljemo pakete fiksne veličine ali ne možemo biti sigurni da je druga strana primila paket podataka.

Iako po tom pitanju Bluetooth nema protokole, ima nešto slično što se može smatrati protokolima.

4.2.1 RFCOMM

RFCOMM protokol je po specifikaciji vrlo sličan TCP protokolu iako po specifikaciji striktno piše da je dizajniran kako bi oponašao RS-232 serijski port

računala. Razlog tome je taj da bi proizvođači koji u svojim uređajima već imaju implementirano sučelje za povezivanje sa drugim uređajima pomoću serijskog porta, lakše implementiraju Bluetooth sučelje za taj isti uređaj.

Najveća razlika između TCP i RFCOMM protokola je izbor broja portova. TCP protokol podržava 65 535 otvorenih portova na jednom računalu, dok je kod RFCOMM protokola taj broj ograničen na samo 30.

4.2.2 L2CAP

UDP protokol omogućuje slanje kratkih poruka (eng. datagrams) između aplikacija na umreženim računalima. UDP nema mogućnost provjere primitka poruke jer ne čuva informaciju o stanju veze (tj. radi na principu pošalji i zaboravi). Zbog toga se koristi kada je bitnija brzina i efikasnost od pouzdanosti, npr. za prijenos govora u realnom vremenu (VoIP telefonija), a također i kada je potrebno slanje iste poruke na više odredišta.

Sve što vrijedi za RFCOMM protokol, vrijedi i za L2CAP. Specifikacije UDP primijenjene su i za ovaj protokol radi dodatnog ne kompliciranja stvari kad je u pitanje komunikacija između dva uređaja.

4.3 Service Discovery Protocol (SDP)

Zadnji korak spajanja pomoću Bluetooth-a je odrediti na koji ćemo se port spojiti. Mogućnost spajanja na više portova omogućuje da se više uređaja ili aplikacija može simultano spojiti na uređaj.

Ako smo se kojim slučajem odlučili za RFCOMM protokol, onda možemo birati između 30 portova, a ako smo se odlučili za L2CAP protokol, izbor je nešto veći. L2CAP raspolaže portovima od 1 do 32 765 ali ograničenim samo na neparne brojeve. Za razliku od RFCOMM protokola gdje su svi portovi slobodni za uporabu, kod L2CAP se taj raspon proteže od 1024 do 32 765.

Portovi od 1 do 1023 služe za unaprijed predodređene aplikacije, oni su tzv. rezervirani portovi. Sažetak portova i protokola možemo pogledati u tablici 1.

Rezervirani portovi su u jednu ruku praktična stvar, u svakom trenutku

Tablica 1: Portovi i njihova terminologija za različite protokole

Protokol	**Termin**	**Rezervirani portovi**	**Dinamički dodjeljeni portovi**
TCP	port	1 – 1024	1025 – 65535
UDP	port	1 – 1024	1025 – 65535
RFCOMM	kanal	nema	1 – 30
L2CAP	PSM	neparni brojevi 1 – 4095	neparni brojevi 4097 – 32765

možemo bez nekakvih dodatnih analiziranja portova odredit koji port se za šta koristi. Ali što ako želimo imati više programa koji inače osluškuju na predefiniranom portu, npr. FTP poslužitelj na portu 21. U tom slučaju ćemo svaki sljedeći rezervirani port premjestiti na neki ne rezervirani, a aplikacija koja će se spajati na taj poslužitelj unaprijed će morati znati broj porta jer se više neće nalaziti na standardnom portu.

Bluetooth protokoli imaju puno manje slobodnih portova na koje se mogu aplikacije spojit te zbog toga može doći do problema ako sami unaprijed definiramo port prilikom spajanja. L2CAP protokol ovim problemom nije toliko zahvaćen koliko RFCOMM koji ima samo 30 slobodnih portova. Nakon sedme aplikacije koja se spoji na neki Bluetooth uređaj, postoji 50% šanse da dođe do kolizije portova. U tom slučaju, programeri aplikacije ne bi trebali proizvoljno zadavati broj porta spajanja. Rješenje tih problema leži u *Service Discovery Protocol-u.*

Umjesto da portove unaprijed definiramo već u kodu, Bluetooth dodjeljuje portove u tijeku spajanja programa na uređaj. To nam omogućuje poslužiteljska aplikacija *Service Discovery Protocol* koja kada se aplikacija prijavi na nju dinamički dodjeljuje broj porta na temelju opisa usluge koju zatražuje.

4.3.1 Service ID

Svaka usluga koja postoji na Bluetooth uređajima opisana je jedinstvenim identifikatorom kojeg zovemo *Service ID*. Identifikator *Service ID* najčešće se nalazi u obliku 128 bitnog *Universally Unique Identifier* (UUID) identifikatora koji ima oblik

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

gdje svaki "X" predstavlja heksadecimalni broj.

Service Discovery Protocol sadrži bazu svih usluga koje postoje na uređaju omogućavajući pri tome da uređaj, ako to zatraži, može dobiti na uvid popis usluga. Na temelju popisa usluga kojeg dobijemo od SDP-a, mi možemo odrediti da li uređaj na koji se spajamo podržava uslugu koju mi želimo.

Može se dogoditi slučaj da je proizvođač uređaja eksplicitno zabranio pojedinačnu uslugu na uređaju te radi toga moramo tražiti alternativne načine spajanja ili možda i kupiti baš njihov proizvod koji koristi tu uslugu. Pravi primjer je iPhone uređaj. Kod tih uređaja nije moguće ostvariti prijenos podataka sa jednog uređaja na drugo pomoću Bluetooth-a, pa tako ni u našem slučaju, nije moguće spojiti iPhone uređaj na GNSS antenu kako bi mogli dobivati podatke iz nje.

4.4 Komunikacija preko *socket-a*

Sudeći prema tekstu prije, ispada da najteži dio komunikacije preko Bluetooth sučelja otpada na traženje uređaja, pronalaženju porta i protokola preko kojeg će se uređaj spojiti. Kada su te predradnje uspješno izvršene, ostaje nam komunikacija između dva uređaja. Kako smo prije spomenuli da se Bluetooth protokol najviše temelji na TCP/IP protokolu, to vrijedi i za komunikaciju, tj. prijenos podataka. Komunikacija se bazira na tzv. *socket-ima*.

Socket[8] predstavlja krajnju točku u komunikaciji. Kada gledamo sa aplikacijske točke gledišta, svaki prijenos podataka koji dolazi ili odlazi mora proći kroz *socket*.

Kada želimo uspostaviti Bluetooth komunikaciju, prvo što moramo napraviti je stvoriti *socket* koja će se koristiti kao krajnja točka u komunikaciji. Najčešće to radimo pomoću ključne riječi *create*. Neposredno nakon što smo kreirali *socket*, moramo odrediti koju vrstu komunikacije (protokol) ćemo koristiti, L2CAP ili RFCOMM. Koji protokol ćemo koristiti ovisi o vrsti prijenosa podataka koja nam je potrebna. Jedina stvar koju još moramo specificirati prije nego li uopće možemo ostvariti bilo kakvu komunikaciju je odrediti u kojem načinu rada će se ta krajnja točka koristiti, da li će služit kao sučelje na koje će se uređaji spajati ili će se taj *socket* spajati na neki drugi uređaj. Ako će se uređaj koristiti u svrhe posluživanja podataka, komunikacija mora biti ostvarena na način da uređaj osluškuje upite koji dolaze prema njemu te sukladno tome uspostaviti komunikaciju.

[8] hrv. utičnica

4.4.1 Klijentski *socket*

Klijentski *socket* je prilično jednostavan za korištenje. Nakon što smo kreirali *socket*, ostaje nam jedino da uspostavimo komunikaciju pomoću *connect* naredbe, pritom specificirajući na koji uređaj ćemo se spojit i na koji port. Operativni sustav se brine o detaljima niske razine (eng. low level) kao što je dodjeljivanje memorije Bluetooth sučelju, pronalasku uređaja, formiranju komunikacije između uređaja i uspostavljanju konekcije. Jednom kada je *socket* kreiran, može se koristiti za prijenos podataka.

4.4.2 Poslužiteljski *socket*

Poslužiteljski *socket* zahtjeva dodatne korake za uspostavljanje komunikacije. Prvo je potrebno povezati (eng. bind) *socket* sa Bluetooth sučeljem specificirajući koji adapter će se koristiti i na kojem portu će poslužitelj osluškivati komunikaciju. Drugo, *socket* se mora postaviti u način rada koju osluškuje nadolazeće konekcije (eng. listening mode). I na kraju, da bi uopće mogao sustav prihvatiti bilo kakvu dolazeću konekciju, mora se pozvati *accept* naredba.

Glavna razlika između klijentskog i poslužiteljskog *socket-a* je ta što *socket* od poslužitelja se nikad ne može koristiti za bilo kakvu vrstu prijenosa podataka. On se isključivo koristi samo za spajanje, a u trenutku kada se drugi uređaj spoji na taj *socket*, poslužitelj kreira novi *socket* koji se koristi za prijenos podataka, a prvobitni *socket* ostaje slobodan za osluškivanje novih konekcija.

5 GNSS

Globalni navigacijski satelitski sustavi (GNSS), standardni generički termin za satelitske navigacijske sustave (Sat Nav) koji pružaju autonomno geoprostorno pozicioniranje s globalnom pokrivenošću. GNSS omogućuje malim elektroničkim prijamnicima determinaciju njihove lokacije (longitude,latitude i altitude) s odmakom od samo nekoliko metara. Prijamnici računaju precizno vrijeme i poziciju.

Jedini potpuno operativni GNSS je američki NAVSTAR Global Positioning System (GPS, hrv. Globalni pozicijski sustav). Ruski GLONASS jest GNSS u procesu pripreme za punu operativnost. Pozicijski sustav Galileo Europske unije nalazi se u inicijalnoj fazi implementacije s planiranom operativnošću do 2014. godine. Narodna Republika Kina naznačila je kako će proširiti svoj regionalni navigacijski sustav Beidou u globalni navigacijski sustav Compass do 2015. godine.

5.1 GPS

NAVSTAR globalni sustav pozicioniranja (GPS) je svevremenski, u svemiru stacionirani sustav razvijen od strane Ministarstva obrane SAD (DoD) s ciljem da zadovolji potrebe vojnih snaga da precizno odrede svoju poziciju, brzinu i vrijeme u jedinstvenom referentnom sustavu, bilo gdje na Zemlji ili blizu zemljine površine na permanentnoj osnovi. Razvoj sustava je započet 1973. godine, dok je civilna uporaba sustava odobrena 1983/84. godine. Initial Operational Capability (IOC) je postignut u srpnju 1993. godine, kada se u orbitama našlo 24 funkcionirajuća satelita, što je službeno proglašeno u prosincu 1993. godine. Full Operational Capability (FOC) je postignut u ožujku 1994. godine, kada je u orbitama bilo 24 Blok II/IIA satelita, koji su

testirani za operativnu vojnu službu, što je službeno proglašeno 17. srpnja 1995. godine. GPS je prvenstveno vojni sustav, za koji je odgovoran UsAF dok operativno sustavom rukovodi JPO (Joint program Office).

Satelit odašilje noseći val koji može imati frekvenciju L1 (1,57542 GHz), L2 (1,2276 GHz) ili L5 (1,17645 GHz). Svaki Blok II i sljedeća generacija satelita je opremljena s dva rubidijumska i cezijumska atomska sata. Dugo periodična stabilnost frekvencije tih satova iznosi 10^{-18} do 10^{-14} u razdoblju od jednog dana. Atomski satovi generiraju fundamentalnu frekvenciju od 10.23 MHz, L – područje (LBand). Na noseće valove modulirana su dva pseudo slučajna "šumna" koda (eng. pseudorandom noise code): C/A kod (Coarse/Acquisition code – 1,023 MHz) također označen kao Standard Positioning Service (SPS), moduliran samo na L1, P kod (Precise code – 10,23 MHz) također označen kao Precise Positioning Service (PPS), moduliran na oba nosača L1 i L2, W-kod (0,5115 MHz) se koristi za šifriranje P – koda u Y-kod koji je nedostupan ne autoriziranim korisnicima i D kod (Data code – 50 Hz) na kojem je modulirana poruka s podacima (eng. broadcast message) koja sadrži efemeride satelita, pogreške vremenskog sustava i sata satelita i drugo.

Segmenti GPS-a su: svemirski (sateliti), kontrolni (stanice za praćenje, kontrolu i upravljanje) i korisnički (prijamnici, permanentne mreže, servisi).

5.1.1 Svemirski segment

GPS sateliti se gibaju u skoro kružnoj orbiti na visini od 20 200 km iznad Zemlje s periodom rotacije od 12 zvjezdanih sati. Raspored i broj satelita (prvotno zamišljenih 24 satelita u tri ravnine s inklinacijom od 63°; sada 24 satelita, raspoređena u 6 ravnina s inklinacijom od 55°). U orbiti se nalazi 27 aktivnih satelita (3 prekobrojna, rezervni aktivni sateliti). S punom konstela-

cijom satelita, svemirski segment jamči globalnu pokrivenost od 4 do 8 satelita koji se mogu simultano opažati pri elevaciji većoj od 15° iznad horizonta. Ukoliko se elevacijska maska reducira na 10° ili 5° moguće je povremeno opažati i do 10 ili 12 satelita. Zadatak satelita je odašiljanje radio signala pomoću kojih se udaljenosti između satelita i prijamnika mogu mjeriti. Zato nam koriste radio odašiljači, atomski satovi, računalo i dr. Poruke koje satelit odašilje služe za određivanje prostornog položaja satelita (signali za mjerenje pseudo-udaljenosti). Identifikacija satelita se određuje pomoću pripadnog pseudoslučajnog koda, broja orbitalne pozicije, kataloškog broja NASA-e i međunarodnom oznakom. Imamo pet klasa satelita. Blok I sateliti: ukupno 11, lansirani od 1978. – 1985. godine iz zračne luke Vanderberg, Kalifornija pomoću AtlasF raketa. Prvo lansiranje je bilo neuspješno. Predviđeni rok trajanja je bio 4,5 godina. 1995. godine je isključen posljednji, iako je još bio ispravan. Inklinacija iznosi 63°. Blok II sateliti: lansirani 1989. i 1990. godine sa inklinacijom od 55°, predviđeni rok trajanja 7,5 godina, povećana težina: 1500kg, promjenjen pristup sigurnosti sustava (neki signali nedostupni za neautorizirane korisnike). Blok II Advanced: lansirani između 1990. i 1997. godine, te su opremljeni mogućnošću uzajamnog komuniciranja. Neki nose retroreflektore za praćenje laserima sa Zemlje (precizno pozicioniranje). Svih 19 je lansirano DeltaII raketom. Blok II Replacement: namjenjeni zamjeni Blok II i Blok IIA satelita, bili su pla-

Slika 10: Orbite satelita

nirani za lansiranje 1995. godien. Prvo lansiranje je ipak bilo 1997.godine i to neuspješno. Ukupno je lansirano 12 ovakvih satelita. Poboljšana oprema za komunikaciju, povećanu autonomnost (samostalno određivanje orbite u dužem razdoblju), pojačana zaštita od radijacije. Blok IIFollow on (trebali biti od 2001. do 2010.) Trebali sadržavati daljnja poboljšanja koja bi trebala omogućiti još veću autonomnost satelita i dodatne mogućnosti odašiljanja signala.

5.1.2 Kontrolni segment

Kontrolni segmnet služi za praćenje satelita u svrhu određivanja orbita i vremena, sinkronizacija vremena satelita, odašiljanje poruka s neophodnim informacijama satelita. Operational Control System određuje i dostupnu razinu točnosti sustava, priprema i lansiranje satelita. Glavna kontrolna stanica je: Consolidate Space Operations Center u Falconu, Colorado Springs, Colorado, prije je bila smještena u zračnoj luci Vanderberg, Californija. U glavnoj kontrolnoj stanici skupljaju se podaci s opažačkih stanica, računaju putanje satelita, parametri sustava te se podaci prosljeđuju jednoj od tri zemaljske stanice zbog eventualnog slanja prema satelitima.

Kontrola satelita i kompletna operacionalizacija sustava odvija se preko opažačkih stanica: Hawaii, Colorado Spings, Ascenscion u južnom Atlantiku, Diego Garcia u Indijskom oceanu i Kwajalein u sjevernom Pacifiku. Opremljene su cezijumskim satovima i neprekidno mjere pseudoudaljenoosti do svih satelita na horizontu. Registriraju mjerenja svakih 1,5 minuta, a u 15 minuta filtriraju podatke koje šalju glavnoj kontrolnoj stanici. Filtrirani podaci služe za određivanje Broadcast efemerida.

Zemaljske kontrolne stanice Ascenscion, Kwajalein i Diego Garcia služe za odašiljanje poruka satelitima o efemeridama i satovima satelita izraču-

natih u glavnoj kontrolnoj stanici. Sastoje se od velikih antena s pratećom opremom. Prije su se podaci slani svakih 8 h, danas 1 – 2x dnevno. Ako se izgubi kontakt s kontrolnom stanicom, sateliti su u stanju zadržati svoju funkciju: Blok I – 4 dana; II – 14 dana; IIA – 180 dana; IIR – 180 dana. Sa svrhom zaštite nacionalnih interesa, SAD su uvele za sve ne autorizirane korisnike ograničenje točnosti i pristupa. Postoje dvije metode pomoću kojih se to ograničenje provodi: Selektivna dostupnost (SA – Selective Availability) i Anti-spoofing (A – S). Kreatori sustava očekivali su točnost C/A koda od 400 m, no prvi testovi u praksi pokazali su točnost od 15 – 40 m. Svrha SA je da onemogući tu visoku navigacijsku točnost s C/A kodom. SA djeluje na dva načina: manipulacijom (namjernim drhtanjem) sata satelita (δ – proces) i manipulacijom (umanjenjem točnosti) efemerida satelita (ε – proces). Kada je SA uključen, navigacijska točnost pada na cca 120 m, pri čemu su kodne i fazne pseudoudaljenosti pogođene na isti način. Utjecaj SA može se zaobići kao autorizirani korisnik ili uporabom dvaju prijamnika. Utjecaj smetnji jednak je na oba prijamnika te se primjenom diferenciranja opažanja između prijamnika utjecaj SA može potpuno zanemariti. SA je bio uključen od uspostave FOC do 2.5.2000. godine. Sa A – S metodom se P – kod "isključuje" za ne autorizirane korisnike, P – kod se kombinira s tajnim W – kodom i rezultat je šifrirani Y-kod dostupan samo autoriziranim korisnicima. Time se DoD želi zaštiti u slučajevima ratne opasnosti od pokušaja neprijatelja da ometaju signal satelita ili sustava u cjelini. AS je neprekidno uključen od 31.1.1994. Utjecaj A – S se ne može zaobići. Utjecaj je moguće umanjiti rekonstrukcijom P – koda u prijamniku obzirom da osnovna struktura Y – koda slijedi strukturu P-koda. Rekonstruirani P – kod ima točnost C/A koda. Prednost rekonstrukcije P – koda leži u činjenici da je moduliran na oba nosača.

5.1.3 Korisnički segment

Imamo dvije kategorije korisnika: vojni korisnici (autorizirani) i civilni korisnici (neautorizirani). Što se tiče prijamnika tu imamo više vrsta koje ćemo opisati u nastavku. C/A kodni prijamnici koji mjere samo pseudoudaljenosti očitavaju samo C/A kod, najčešće služe za navigaciju jer ne pružaju visoku točnost, malog su formata te imaju mogućnost ispisa trodimenzionalnih koordinata. C/A kodni prijamnici za fazna mjerenja očitavaju C/A koda i L1 fazu te ne mogu opažati L2 frekvenciju. P-kodni prijamnici očitavaju P-kod na oba nosača kada je A – S isključen.

5.2 GLONASS

GLONASS je kratica za "Globalnaja navigacionaja sputnikovaja sistema". GLONASS je bio odgovor SSSR-a na razvoj GPS-a, njegova je uspostava započeta 1976. GLONASS je također prvenstveno vojni sustav. Sustav se sastoji iz tri komponente: svemirske (sateliti), kontrolne (kontrolna stanica i stanice za telemetriju) i korisničke (GLONASS prijamnici). Sustav se sastoji iz minimalno 24 satelita u orbiti (21 + 3). Sateliti kruže na visini od oko 19 130 km iznad Zemlje u 3 orbite koje su skoro potpuno kružne. Inklinacija orbita iznosi 64,8°, a period rotacije satelita je 11h 15m 40s. GLONASS sateliti emitiraju signale na 2 frekvencije: L1 (1598,0625 – 1607,0625 MHz) i L2 (1242,9375 – 1249,9375 MHz). Kontrolni segment se sastoji iz: kontrolnog centra sustava (Moskva) i 5 stanica za telemetriju, praćenje i kontrolu (Moskva + St. Petersburg, Ternopol, Jenisejsk, Komsomolsk na Amuru). Nedostatak je što se sve stanice nalaze na teritoriju bivšeg SSSR-a te stoga postoji razdoblje kada se ne može provoditi praćenje i ažuriranje satelita podacima. Rusija se, iako već nekoliko godina najavljuje punu operativnu

sposobnost sustava (24 operativna satelita), nikako ne uspijeva niti približiti tom stanju. Najveći problem GLONASS-a je i dalje kvaliteta satelita. Za razliku od GPS satelita koji predugo traju, GLONASS sateliti traju prekratko. Stoga im i broj satelita opada. O izvornim GLONASS prijemnicima malo se zna ili ih ima jako malo, dok su GPS-GLONASS prijamnici dostupni na otvorenom tržištu (Ashtech, Topcon, Javad, Trimble, ...).

5.3 Princip rada GNSS-a

Svaki satelit koji se nalazi u orbiti iznad Zemlje konstantno odašilje određeni set informacija: jedinstveni identifikacijski kod satelita (PRN kod), svoju trenutnu poziciju, vrijeme odašiljanja vala i pozicije ostalih satelita (almanah). Almanah je potreban GNSS prijamniku kako bi lakše i brže pronašao satelite u orbiti. Može se dogoditi slučaj ako prijamnik nije duže vrijeme bio u funkciji, vrijeme potrebno za inicijalizaciju i početak rada prijamnika će biti nešto duže nego inače baš zbog razloga što će prijamnik zahtijevati osvježenu datoteku almanaha.

Jednom kada je prijamnik prikupio dovoljan broj informacija iz signala satelita, može pristupiti određivanju udaljenosti između prijamnika i satelita. Pošto znamo da se svaki elektromagnetski val širi brzinom svjetlosti, udaljenost između jednog satelita i prijamnika dobiti ćemo množenjem vremena potrebnog da signal stigne od satelita do prijamnika pri brzini svjetlosti koja iznosi 299 792 458 m/s.

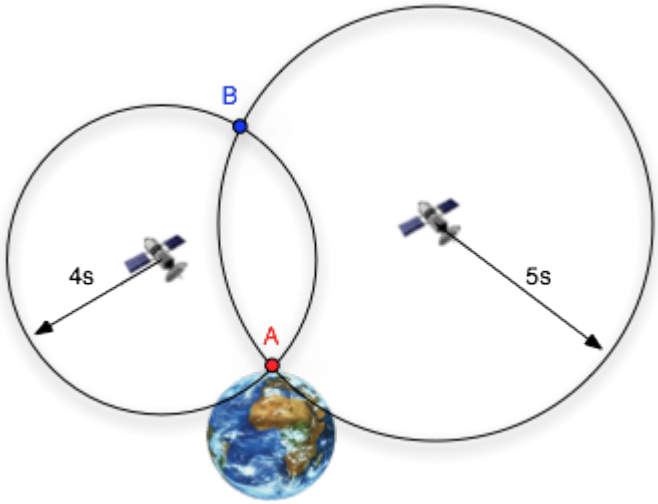
$$s = v * t$$

Uzmemo li udaljenost od još dva satelita, ukupno 3, možemo pomoću izračuna trialteracije dobiti lokaciju prijamnika u 2D-u. Visina prijamnika bi u tom slučaju trebala biti aproksimirana na površinu Zemlje. Kako znamo

da Zemlja nije savršeno tijelo, da bi smo dobili pravu visinu prijamnika, potrebna nam je udaljenost od još jednog satelita. Tek pomoću 4 satelita možemo dobiti lokaciju u tri dimenzije.

U teoriji ova stvar funkcionira, ali u stvarnosti postoji jedan problem. Satovi koji se nalaze u satelitima imaju točnost od $2 * 10^{-13}$ sekundi na dan, dok kvarcni satovi koji se ugrađuju u GNSS prijamnike su daleko jeftiniji i netočniji. Baš zbog te razlike u točnosti i neusklađenosti može doći do velikih nesuglasica u izračunavanju udaljenosti.

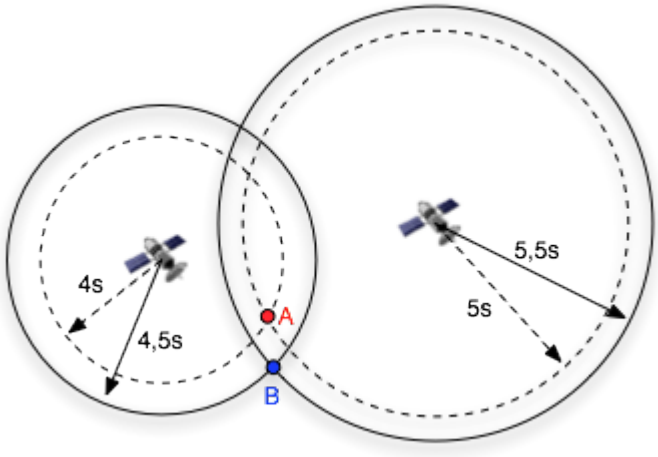
Radi jednostavnosti crtanja i vizualiziranja, princip rada ćemo objasniti u 2D-u. Neka se u orbiti nalaze dva satelita koji konstantno odašilju signale. Prijamnik koji se nalazi na Zemlji prima signale i na temelju izračunate udaljenosti presjekom kružnica dobiva dva rješenja (slika 11). Rješenje B može



Slika 11: Lokacija određena na temelju dva satelita (u 2D prostoru)

se izbaciti kao moguće riješenje zbog njegove velike udaljenosti od središta Zemlje te prihvatiti A kao konačno. Da bi izbjegli takve nesuglasice, uvodi se još jedan dodatni satelit. Zbog pogreške satova koje smo prethodno spo-

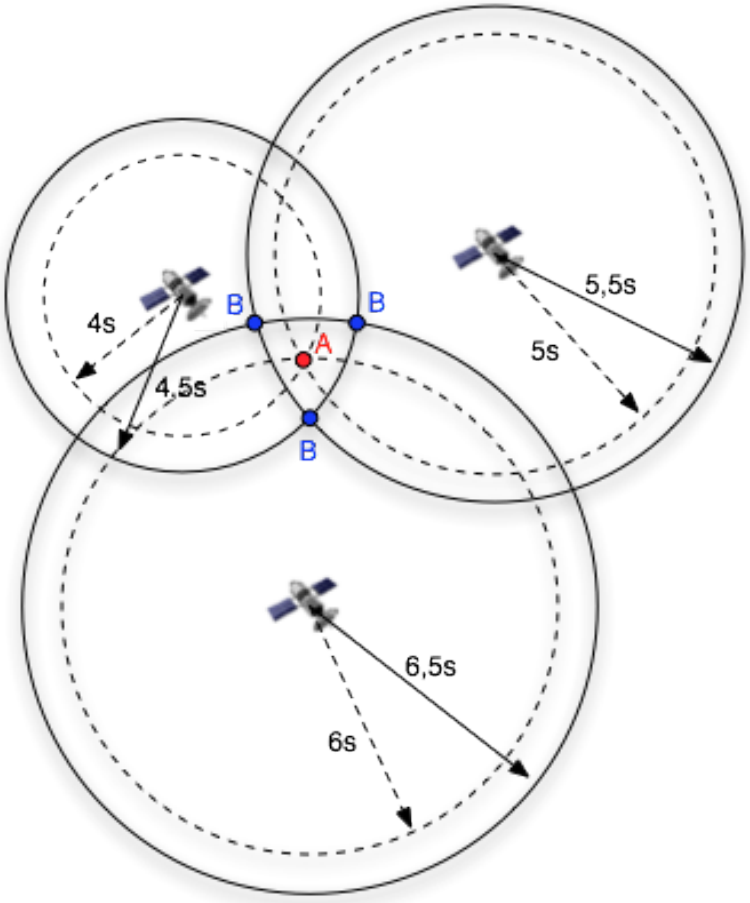
minjali dogodit će se sljedeći slučaj (slika 12). Pretpostavimo da sat koji se



Slika 12: 2D položaj prijamnika određen na temelju dva satelita i pogreške satova

nalazi u prijamnicima rani u odnosu na satove u satelitima (iscrtkana kružnica). Zbog takve pogreške dobiti ćemo da je izračunata lokacija B a ne A koja i je pravo rješenje. Baš zbog takve greške u nesinhroniziranosti satova, uvodi se još jedan satelit koji će pomoći odrediti točno rješenje (slika 13). Dodavanjem trećeg satelita i presjekom svih kružnica zapravo ćemo dobiti tri moguća rješenja (rješenja opterećena pogreškom satova). Jednolikim povećanjem ili smanjenjem (u našem slučaju) u jednom trenutku će se sva rješenja poklopiti u jednoj točci, točki A. S ovakvim principom određivanja položaja prijamnika, sinkronizirali smo satove prijamnika i satelita te dobili lokaciju u 3D koordinatnom sustavu.

No kako naš svijet nije dvodimenzionalan kao u primjeru već trodimenzionalan, dodavanjem treće dimenzije moramo dodati i još jedan sateliti, 4. satelit kao što je rečeno na početku.



Slika 13: 3D položaj prijamnika određen na temelju tri satelita i pogreške satova

5.4 NMEA protokol

Protokol je uspostavljen od strane National Marine Electroics Association (NMEA) s ciljem razmjene podataka između različitih sustava unutar nacionalnih pomorskih tvrtki na području USA.

NMEA 0183 (kako se zapravo zove standard) uređaji su programirani tako da ili osluškuju poruke ili ih odašilju (postoje uređaji koji mogu raditi u oba načina rada). Uređaj kao što su GNSS antene su uređaji koji odašilju NMEA poruke, a uređaji poput GNSS kontrolera su uređaji koji osluškuju poruke i analiziraju ih. Kako bi kontroler uspješno osluškivao NMEA protokol, on mora biti pravilno podešen. Postavke osluškivanja NMEA protokola:

- Baud rate: 4800
- Number of data bits : 8
- Stop bits: 1
- Parity: none
- Handshake: none

5.4.1 Formati poruka

Svi podaci koji se šalju preko NMEA protokola, šalju se u obliku rečenica. Koristi se samo ASCII tekst zajedno sa posebnim znakovima CR[9] i LF[10]. Svaka rečenica počinje znakom "$" i završava sa posebnim znakovima $< CR >< LF >$. U NMEA protokolu postoje tri vrste formata poruke: *talker sentence*, *proprietary sentence* i *query sentence*.

Talker sentence je poruka koja sadrži informacije odvojene zarezom u obliku:

[9] eng. carriage return
[10] eng. line feed

$$\$ttsss, d1, d2, ... < CR >< LF >$$

Prva dva znaka nakon $ znaka su tzv. *talker* identifikatori (tt). Sljedeća tri znaka (sss) su tzv. identifikatori poruke, nakon kojih slijede podaci odvojeni zarezom. Svaka rečenica završava proizvoljnim provjernim brojem (eng. *checksum*) te posebnim znakovima.

Proprietary sentence je format poruke koji je posebno prilagođen svakom proizvođaču da on proizvoljno definira informacije specifične za njegove proizvode. Ovaj oblik poruke počinje znakovima "$P" nakon kojih slijede 3 znaka koji označavaju identifikacijskih broj proizvođača te proizvoljan broj informacija odvojenih zarezom.

Query sentence je posebna vrsta poruke koja služi za slanje upita uređajima kako bi zatražili informacije od njih u obliku NMEA protokola. Format poruke:

$$\$ttllQ, sss, < CR >< LF >$$

Prva dva znaka (tt) označavaju identifikacijski broj uređaja koji zatražuje upit, sljedeća dva znaka (ll) označavaju identifikacijski broj uređaja kojemu je upućen upit nakon kojeg slijedi oznaka Q koja označava da se radi o upitu (eng. *query*). Znakovi "sss" označavaju vrstu poruke koju uređaj zatražuje. Primjer *query sentence* upita:

$$\$CCGPQ, GGA < CR >< LF >$$

Od svih ovih vrsta poruka, mi ćemo se ovdje fokusirati na *talker sentence* poruke koje počinju znakovima "GP" zato jer one sadrže prostorne informacije i informacije o satelitima.

5.4.2 $GPxxx vrsta poruke

$GPxxx vrste poruka su poruke poslane sa GNSS uređaja. Postoji 9 vrsta $GP poruka od kojih su nama najzanimljivije:

Tablica 2: Vrste poruka poslane sa GPS uređaja

Vrsta poruke	Opis
GGA	Vrijeme, pozicija i kvaliteta podataka
GLL	Geografska širina i dužina, UTC vrijeme, status
GSA	Statusne poruke satelita
GSV	Broj satelita sa elevaciom, azimutom i SNR podacima
MSS	SNR omjer, jačina signala i frekvencija
RMC	Vrijeme, datum, lokacija, smjer i brzina kretanja
VTG	Smjer i brzina kretanja u odnosu na površinu

5.4.3 GGA poruka

Nama najzanimljivija poruka je, dakako, GGA poruka. GGA poruka sadrži položaj uređaja u obliku geografskih koordinata u WGS84 koordinatnom sustavu. Primjer jedne GGA poruke:

$$\$GPGGA, 000019.00, 4158.8435628, N, 09147.4425035, W, 4, 16, 0.7,$$
$$257.280, M, -32.00, M, 01, 0602 * 64$$

U tablici 6 ćemo pobliže objasniti svaki segment poruke.

Tablica 3: Segmenti GGA poruke

Ime	Primjer	Opis
Vrsta $GP poruke	$GPGGA	Zaglavlje poruke
UTC vrijeme	000019.00	hhmmss.ss
Geografska širina	4158.8435	ddmm.mmmm
N/S indikator	N	N=sjever, S=jug
Geografska dužina	09147.4425	hhmm.mmmm
E/W indikator	W	E=istok, W=zapad
Indikator GPS kvalitete	4	Vidi tablicu 4
Korišteno satelita	16	Broj satelita od 0 do 12
HDOP	0.7	*Horizontal Dilution of Precision*
Ortometrijska visina	257.280	
Jedinice	M	
Geoidna razlika	-32.00	
Jedinice	M	
Starost dif. korekcija	01	Prazno kada se ne koristi DGPS
Broj provjere	0602*64	
$< CR >< LF >$		Kraj poruke

Tablica 4: Indikatori kvalitete GPS signala

Vrijednost	Opis
0	Nije dostupan
1	GPS SPS mod, fiksna vrijednost
2	DGPS. SPS mod, fiksna vrijednost
3	RTK, fiksno
4	RTK, float

5.5 NTRIP protokol

Network Transport of RTCM via Internet Protocol je protokol koji podržava prijenos GNSS podataka preko interneta. NTRIP protokol se zasniva na dobro poznatom protokolu za razmjenu informacija na internetu, *HyperText Transfer Protocol*, HTTP/1.1. Proširivanjem HTTP protokola, on postaje nosioc GNSS podataka za prijenos RTCM korekcija preko mreže.

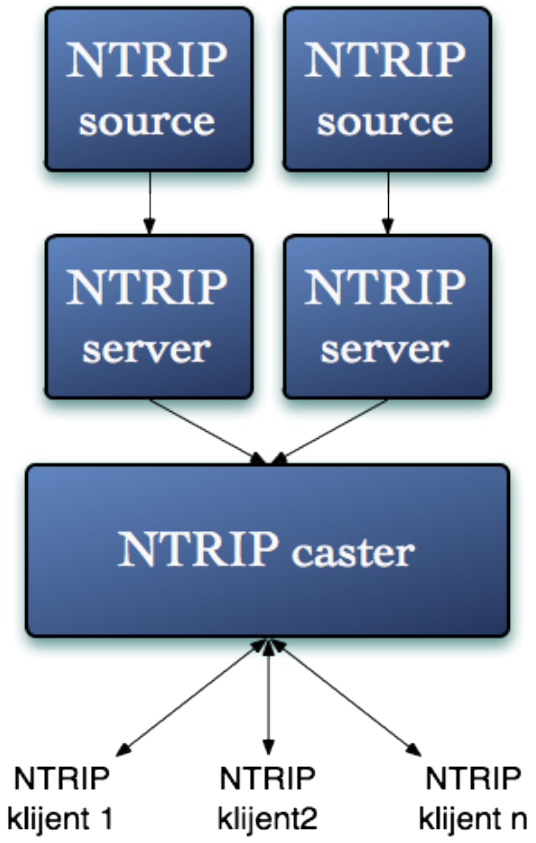
NTRIP je prvenstveno bio projektiran za prijenos diferencijalnih korekcijskih parametara i ostalih vrsta GNSS podataka prema mobilnim korisnicima i poslužiteljima preko Internet-a, omogućavajući simultanu konekciju između korisnika.

NTRIP se sastoji od tri komponente: NTRIP klijenta, NTRIP servera i NTRIP *Caster-a*. NTRIP *Caster* je poslužiteljsko računalo koje se ponaša kao HTTP poslužitelj dok su NTRIP klijent i NTRIP server HTTP klijenti (Vidi sliku 14). Glavne karakteristike NTRIP protokola su:

- Baziran je na dobro poznatom HTTP standardu za prijenos informacija preko Internet-a, te ga je lako za implementirati
- Ne postoji ograničenje u vrsti podataka koja se može prenositi u ovom

standardu

- Pogodan je za masovnu uporabu, što znači da protokol može podnijeti simultani prijenos podataka na više od tisuću korisnika istovremeno
- Što se tiče sigurnosne strane, protok informacija između klijenta i poslužitelja ne mora biti direktan, te protokol nije podložan blokiranju od strane vatrozida ili posrednih računala (eng. proxy servers)
- Protokol podržava sve vrste mreža jer se bazira na Internet standardu za prijenos informacija, TCP/IP standardu



Slika 14: Komponente i princip NTRIP protokola

NTRIP protokol se najčešće koristi u diferencijalnom načinu rada DGNSS (eng. Differencial Global Navigation Satellite System). DGNSS referentne stanice činu GNSS prijamnici koji se nalaze na lokacijama sa poznatim koordinatama. Na temelju podataka iz satelita, referentne stanice u svakom trenutku mogu izračunati gdje se koji satelit nalazi u orbiti i na temelju toga izračunati teoretsku udaljenost između prijamnika i satelita. Razlika između teorijske udaljenosti i udaljenosti dobivene preko signala satelita, izračunavaju se korekcije u datom trenutku. Dostupnost tih korekcija u bilo kojem trenutku za mobilne korisnike, glavna je svrha NTRIP protokola. Protokol se može koristiti i za prijenos ostalih vrsta GNSS podataka.

5.5.1 Komponente NTRIP protokola

Iz prethodne slike vidljivo je da se NTRIP protokol sastoji od 4 komponente, NTRIP Caster-a, NTRIP Server-a, NTRIP Source-a i NTRIP Klienta. Komunikacija između te tri komponente se odvija pomoću HTTP 1.1 verzije protokola u kojem su definirana specifična zaglavlja poruke koja definiraju vrstu komunikacije.

NTRIP Source se sastoji od grupe računala koja generiraju GNSS podatke u realnom vremenu (npr. RTCM-104 korekcije) te ih prosljeđuju dalje prema NTRIP Serveru. Svaki NTRIP Source predstavlja GNSS podatke koji su specifični samo za jednu lokaciju.

NTRIP Server je komponenta kojoj je zadaća da prenosi GNSS podatke (korekcije) sa NTRIP Source-a do NTRIP Caster-a. Prije nego proces slanja putem TCP/IP konekcije počne, NTRIP Server šalje *točku montiranja* (eng. mountpoint) prema NTRIP Caster-u. Točka montiranja je unikatni

identifikator koji se definira na NTRIP Source-u koji predstavlja servis sa kojeg se isporučuju korekcije. Određeni servisi daju određene vrste informacija i točnosti u obliku RTCM korekcija.

NTRIP protokol može služiti za prijenos RTCM korekcija za tzv. VRS[11] koncept. Napomenimo da je na tom istom konceptu baziran princip rada CROPOS sustava. Putem NTRIP protokola šalju se RTCM korekcije sa virtualne točke na terenu koja je generirana na temelju približne lokacije rovera. Korekcije poslane za tu virtualnu referentnu stanicu predstavljaju jedan NTRIP Source poslan od strane NTRIP Server-a.

NTRIP Caster je HTTP server koji podržava određeni broj zahtjeva/odaziva upućenih prema poslužitelju. Poruke koje dolaze prema NTRIP Caster-u mogu biti upućene bilo sa NTRIP Server-a bilo sa NTRIP Klijenta te na temelju toga poslužitelj određuje da li poruku mora proslijediti ili obraditi.

NTRIP Klijent je taj koji šalje podatke prema NTRIP Caster-u te ih prima i obrađuje ukoliko su poruke koje su poslane strukturirane prema pravilima NTRIP protokola. NTRIP Klijent kada šalje podatke, najčešće šalje svoju približnu lokaciju u obliku NMEA poruke na temelju koje NTRIP Caster odgovara sa RTCM korekcijama specifično za tu lokaciju. Ukoliko je klijent krivo definirao točku montiranja ili je uopće nije definirao, NTRIP Caster u tom slučaju odgovara sa kolekcijom montiranih točaka koji su definirane u sustavu. Na temelju ažuriranih podataka, klijent se može spojiti na dostupnu točku montiranja te početi dobivati podatke od tog servisa.

[11] eng. Virtual Reference Station

5.5.2 Komunikacija klijenta

Kao što je prije pisano, sva komunikacija odvija se preko HTTP protokola. Da bi klijent uputio bilo kakav zahtjev prema sustavu preko NTRIP protokola mora znati IP adresu poslužitelja, *port* i točku montiranja. Neki sustavi zahtijevaju korisničko ime i zaporku kako bi se mogli prijaviti u sustav, tako da i to treba uzeti u obzir.

Kada šaljemo poruku prema NTRIP Caster-u kako bi započeli komunikaciju i preuzimanje podataka od sustava, HTTP GET zahtjev moramo struktirirati prema određenim pravilima. Primjer jednog takvog zatjeva:

```
GET /*MountPoint* HTTP/1.0\r\n
User-Agent: NTRIP ProizvoljniNaziv\r\n
Accept: */*\r\n
Connection: close\r\n
Authorization: Basic *KorisnickoIme*:*Zaporka*\r\n
```

Napomena: MountPoint, KorisnickoIme i Zaporka treba zamjeniti pravim vrijednostima.

Pošto se u ovakvoj poruci šalju povjerljivi podaci poput korisničkog imena i zaporke, NTRIP protokol je uveo enkripciju tih osjetljivih dijelova, tako da se ta dva parametra trebaju kodirati pomoću Base64 enkripcije.

5.5.3 NTRIP Sourcetable

Svaki NTRIP Caster sadrži tzv. *source-table* u kojem su sadržane sve informacije o dostupnim *NTRIP Source-ovima* te dodatne informacije o svakom posebno. Da bi dobili Source-table od nekog sustava, dovoljno je poslati obični HTTP zahtjev preko internet pretraživača. U našem slučaju, poslat

ćemo takav zahtjev prema CROPOS sustavu tako što ćemo u polje za internet adresu upisati: *http://195.29.118.122:2101/*. Poruka koju smo dobili kao odgovor na zahtjev izgleda ovako:

```
SOURCETABLE 200 OK
Server: NTRIP Trimble NTRIP Caster
Content-Type: text/plain
Content-Length: 1092
Date: 18/Jun/2012:14:48:33 UTC

STR;CROPOS_VRS_RTCM23;CROPOS_VRS_RTCM23;
    RTCM 2.3;1(1),3(10),18(1),19(1);2;GPS+GLONASS;CROPOS;
    HRV;0;0;1;1;Trimble GPSNet;None;B;Y;0;;
STR;CROPOS_VRS_RTCM31;CROPOS_VRS_RTCM31;RTCM 3;
    1004(1),1005/1007(5),PBS(10);2;GPS+GLONASS;CROPOS;
    HRV;0;0;1;1;Trimble GPSNet;None;B;Y;0;;
STR;CROPOS_VRS_DGNSS;CROPOS_VRS_DGNSS;RTCM 2.3;
    1(1),3(10);0;GPS+GLONASS;CROPOS;HRV;0;0;1;1;
    Trimble GPSNet;None;B;Y;0;;
STR;CROPOS_VRS_HTRS96;CROPOS_VRS_HTRS96;RTCM 3;
     1004(1),1005/1007(5),
     PBS(10);2;GPS+GLONASS;CROPOS;HRV;0;0;1;1;T
     rimble GPSNet;None;B;Y;0;;
STR;CROPOS_VRS_HDKS;CROPOS_VRS_HDKS;RTCM 3;
     1004(1),1005/1007(5),
     1014(1, 1 msgs),1015(1, all msgs),1016(1, all msgs);2;
     GPS+GLONASS;CROPOS;HRV;0;0;1;1;Trimble GPSNet;
     None;B;Y;0;;
```

```
STR;CROPOS_VRS_HDKS_NW;CROPOS_VRS_HDKS_NW;
      RTCM 3;1004(1),1005/1007(5),1014(1, 1 msgs),1015
      (1, all msgs),1016(1, all msgs);2;GPS+GLONASS;CROPOS;
      HRV;0;0;1;1;Trimble GPSNet;None;B;Y;0;;
STR;CROPOS_VRS_HDKS_NE;CROPOS_VRS_HDKS_NE;
      RTCM 3;1004(1),1005/1007(5),1014(1, 1 msgs),1015
      (1, all msgs),1016(1, all msgs);2;GPS+GLONASS;CROPOS;
      HRV;0;0;1;1;Trimble GPSNet;None;B;Y;0;;
ENDSOURCETABLE
```

Pogledamo li malo strukturu zapisa uočit ćemo da je svaki podatak zapisa odvojen točka zarezom, ";". Broj polja nije striktno definiran tako da može sadržavati dodatne podatke, a cijela poruka mora početi sa zapisom *SOURCETABLE 200 OK* te završit zapisom *ENDSOURCETABLE* kako bi se mogao razlučiti početak i kraj poruke.

Izdvojimo sada jedan zapis i analizirajmo ga polje po polje:

```
STR;CROPOS_VRS_RTCM23;CROPOS_VRS_RTCM23;
    RTCM 2.3;1(1),3(10),18(1),19(1);2;GPS+GLONASS;CROPOS;
    HRV;0;0;1;1;Trimble GPSNet;None;B;Y;0;;
```

Tablica 5: Struktura jednog zapisa u Source-table poruci

Zapis	Opis
STR	Opisuje vrstu podatkovnog prijenosa
CROPOS_VRS_RTCM23	Ime točke montiranja
CROPOS_VRS_RTCM23	Identifikacijski zapis
RTCM 2.3	Format zapisa
1(1),3(10),18(1),19(1)	Opis RTCM formata poruke
2	Informacije sa L1 i L2 nosača
GPS+GLONASS	Podržani sustavi
CROPOS	Ime sustava
HRV	Ime države
0	Geografska širina
0	Geografska dužina
1	Klijent mora prvo poslati približnu lokaciju rovera u obliku NMEA poruke
1	Podaci su generirani na temelju mreže stanica
Trimble GPSNet	Opis software-a koji generira podatke
None	Algoritam enkripcije
B	Vrsta enkripcije za korisničko ime i zaporku
Y	Sustav se naplaćuje
0	Brzina prijenosa podataka

5.6 RTCM

Tehnička komisija (Radio Tehnical Commission for Martime Services) iz Virginia, USA, kreirala je široko rasprostranjeni radio-prijenosni format za distribuciju podataka, koji ima ustaljeni skraćeni naziv među korisnicima RTCM. Ovaj format prihvaćen je kao standard za sve GNNS aplikacije, koje omogućuju slanje diferencijalnih korekcije prema pokretnim uređajima (roverima). RTCM format se koristi za prijenos GNSS korekcija kada se koristi DGPS/RTK metoda mjerenja i najčešće dolazi u binarnom obliku zbog veće brzine obrade podataka u realnom vremenu. Baš zbog toga nismo u mogućnosti pročitati RTCM korekcije koje izvorno dolaze putem NTRIP protokola već ih je potrebno pretvoriti u format razumljiv čovjeku.

Tijekom vremena, RTCM format je doživio nekoliko promjena kako su se događale promjene u GNSS svijetu. Izdvojimo promjene po verzijama:

- RTCM 2.0 - Prilagođen DGPS mjerenju
- RTCM 2.1 - Dodana fazna korekcija za RTK mjerenja
- RTCM 2.2 - Dodana podrška za GLONASS
- RTCM 2.3 - Dodana podrška za definiranje GPS antene
- RTCM 3.0 - Dodana podrška za *Network RTK* i GNSS

Zbog sve veće zahtjevane točnosti i brzine inicijalizacije, integriran je signal GPS i GLONASS sustava. Iz prethodnog kao i zbog novih L2C i L5 signala nužno je bilo modificirati RTCM 2.3 poruke u novi RTCM 3.0.

Tablica 6: Prikaz moguće kreiranih poruka u RTCM 2.3 formatu

RTCM#	Status	Opis
1	Stalan	Diferencijalna GPS korekcija
2	Stalan	Delta diferencijalna GPS korekcija
3	Stalan	Parametri referentne GPS stanice
4	Probni	Datum referentne stanice
5	Stalan	Stanje GPS satelita
6	Stalan	Nulti GPS okvir
7	Stalan	DGPS almanah
8	Probni	Almanah pseudolita
9	Stalan	Parcijalni set GPS korekcija
10	Nedovršen	P-Code diferencijalna korekcija
11	Nedovršen	C/A-Code L1, L2 delta korekcija
12	Nedovršen	Pseudolitni parametri stanice
13	Probni	Parametri odašiljača
14	Stalan	GPS vrijeme u GOS tjednu
15	Stalan	Poruka o Ionosferi
16	Stalan	Specijalna GPS poruka
...		
63	Nedovršen	Višenamjensko korištenje

6 CROPOS

CROPOS je državna mreža referentnih GNSS stanica Republike Hrvatske. Svrha sustava je omogućiti određivanje položaja u realnom vremenu s točnošću od 2 cm u horizontalnom smislu te 4 cm u vertikalnom smislu na čitavom području države.
Značajke sustava su:

- Prikupljanje podataka mjerenja sa 30 referentnih GNSS stanica (Slika 15) pravilno raspoređenih na cijelom teritoriju Republike Hrvatske
- Umreženje referentnih GNSS stanica i računanje korekcijskih parametara u realnom vremenu
- Distribucija podataka mjerenja i korekcijskih parametara u realnom vremenu korisnicima
- Nadzor rada sustava i podrška korisnicima
- Raspoloživost usluga sustava 24\7

CROPOS sustav radi po principu umreženih referentnih stanica koje se nalaze na području cijele države međusobno razmaknutih oko 70 kilometara. Baš zbog tih referentnih stanica više nije potrebno imati baznu stanicu tijekom mjerenja što uvelike smanjuje broj potrebnih prijamnika za mjerenje. Zbog CROPOS-a sada je potrebno ponijeti svega jedan RTK prijamnik kako bi se vršilo mjerenje terena, a s time je skraćeno terensko mjerenje jer nema postavljanja i konfiguriranja baznog uređaja. Ostale prednosti su:

- Nema ograničenja zbog dosega radio uređaja za prijenos korekcijskih parametara
- Znatno kraće vrijeme inicijalizacije

Slika 15: Lokacije CROPOS stanica u Republici Hrvatskoj

- Homogenost mjerenja na cijelom području države
- Povećana točnost mjerenja

6.1 Komponente sustava

Sustav se sastoji od dvije komponente:

- Referentne GNSS stanice
  Sustav se sastoji od 30 referentnih stanica jednoliko raspoređenih po teritoriju države. Stanice primaju GPS i GLONASS signale, a u budućnosti će primati i GALILEO signal. Stanice su definirane koordinatama u dva referentna okvira, ITRF2005 i ETRF00. Od 30 stanica, dvije se koriste kao kontrolne, stanice Šibenik i Senj, koje služe kao kontrola kvalitete rada sustava.
- Kontrolni centar
  Služi za upravljanje i održavanje rada sustava, obradu podataka i izjednačenje mreže
  - Komunikacijska oprema
    Povezivanje pojedinih komponenti sustava
  - Oprema za distribuciju podataka
    Distribucija RTCM i VRS RTCM korekcijskih parametara, distribucija RINEX i Virtual RINEX datoteka

6.2 CROPOS usluge

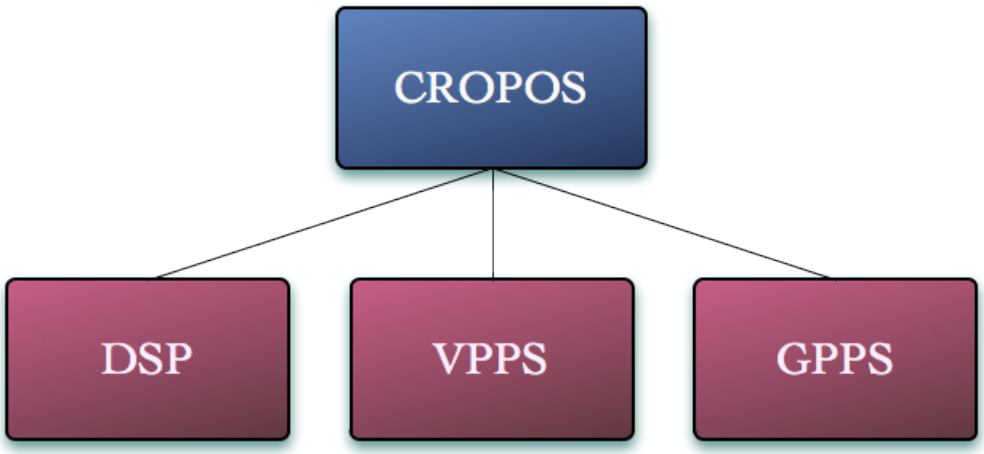
CROPOS servis nudi tri različite vrste usluga: DSP, VPPS i GPPS (Slika 16). Razlika između ove tri usluge je prvenstveno u točnosti mjerenja koje možemo dobiti preko sustava. Koju ćemo od uslugu od ove tri koristiti, ovisi

o vrsti mjerenja i potrebnoj točnosti.

DSP ili diferencijalni servis pozicioniranja u realnom vremenu je servis koji koristi umreženo rješenje kodnih mjerenja u realnom vremenu. Tehnologija koja se koristi za spajanje na sustav je pomoću bežičnog interneta i to se isključivo koristi NTRIP (eng. Network Transfer RTCM over IP) protokol. Točnost koju ovaj servis pruža je od 0.3 do 0.5 metara. Ova usluga koristi RTCM korekcijske parametre. DSP servis se najčešće koristi prilikom izrade geoinformacijskih sustava gdje nije potrebna velika točnost, u navigaciji, upravljanju prometom, zaštiti okoliša, poljoprivredi, šumarstvu itd.

VPPS ili visokoprecizni servis pozicioniranja u realnom vremenu koji koristi umreženo rješenje faznih mjerenja u realnom vremenu. VPPS servis također koristi bežični internet za spajanje na sustav preko NTRIP protokola, samo što se kod ovog sustava može spajati i preko GSM modema. Usluga pruža točnost od 0.02 metra u 2D koordinatnom sustavu te do 0.04 metra u 3D sustavu. Servis omogućava izbor između dvije vrste verzija RTCM korekcija, 2.3 i 3.1. Najčešća uporaba ovog servisa je u osnovni geodetskim radovima, katastru, inženjerskoj geodeziji, izmjeri državne granice, topografske izmjere i sl.

GPPS ili geodetski precizni servis pozicioniranja je najpreciziji servis. Servis se najčešće koristi za mjerenja gdje je potrebna najveća točnost kao što su mjerenja geodetske osnove, određivanja referentnih sustava, znanstvena i geodinamička istraživanja itd. Ovaj servis je poseban po tome što ne daje podatke u realnom vremenu već je potrebno pomoću Internet-a preuzeti korekcije u obliku RINEX ili VRS RINEX datoteka. Korištenjem ove usluge možemo dobiti subcentimetarsku točnost (< 0.01 metar).

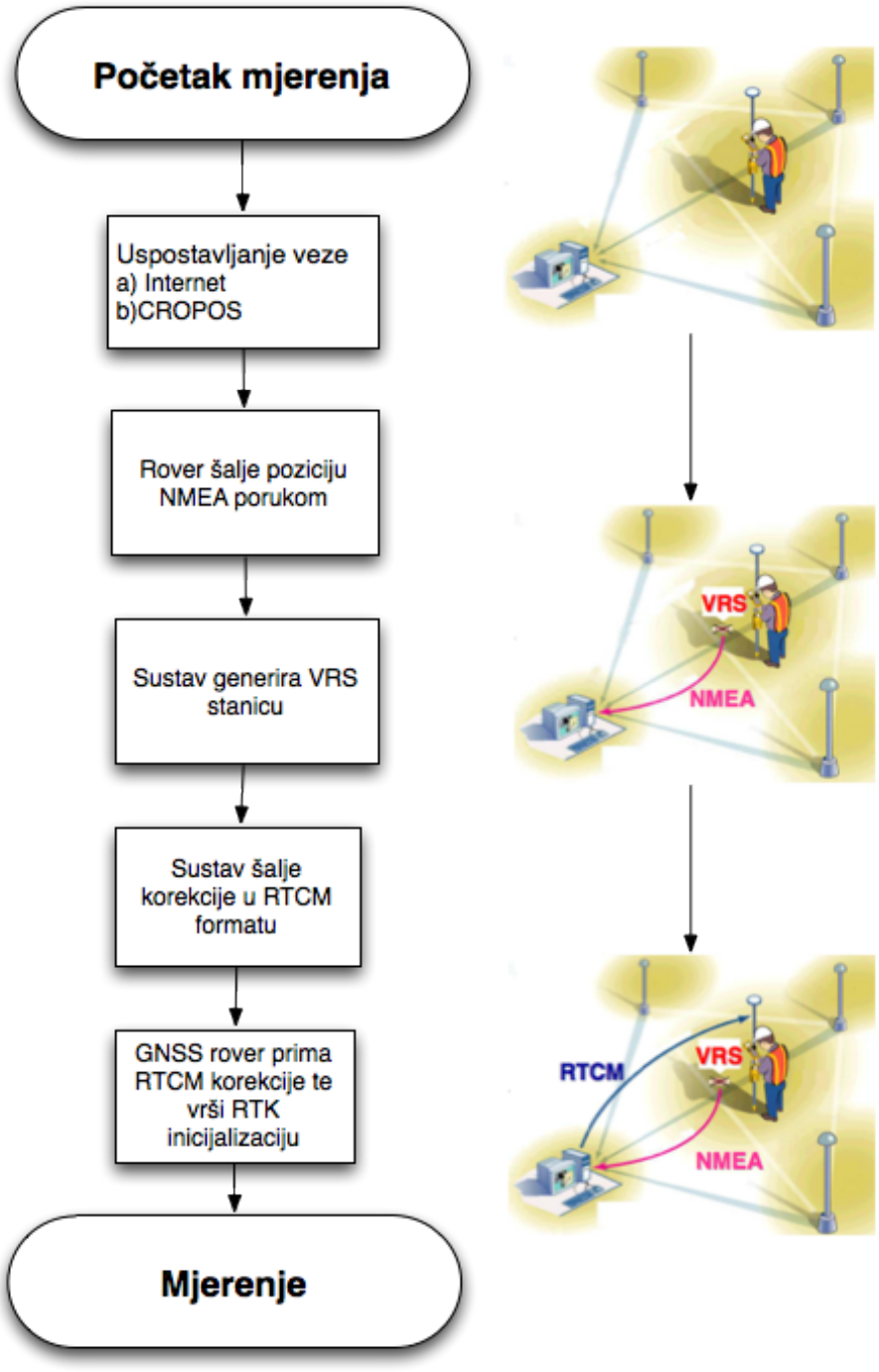


Slika 16: CROPOS servisi

6.3 Princip rada CROPOS-a

Svaki korisnik CROPOS servisa mora posjedovati korisničko ime i zaporku kako bi se koristio uslugama. Osim korisničkih podataka, potrebno je upisati internet adresu servisa na koji se spajamo te odrediti koju vrstu usluge želimo koristiti, npr. CROPOS_VRS_RTCM23, CROPOS_VRS_RTCM31, CROPOS_VRS_HDKS ili CROPOS_VRS_HTRS96. Nakon što smo upisali sve potrebne podatke za korištenje CROPOS-a, tek sada se možemo koristiti sa njime.

Sam princip mjerenja uz pomoću CROPOS-a je prilično jednostavan. Nakon izvršenog spajanja na sustav, potrebno je poslati koordinate rovera u obliku NMEA poruke CROPOS sustavu koji će na temelju dobivene lokacije moći izgenerirati virtualnu referentnu stanicu (VRS) koju će nam natrag poslat u obliku RTCM korekcijskih parametara. Nakon promjene položaja lokacije rovera, cijeli postupak se ponavlja. Shematski prikaz rada sa CROPOS sustava objašnjen je na slici 17.



Slika 17: Shematski prikaz rada sa CROPOS sustavom

7 Android aplikacija

Kao što samo ime "Geoinformatika" znači spoj između informatike i geodezije, s tom pomisao došli smo na ideju spajanja aktualnih trendova u svijetu informatike sa jednim specifičnim područjem geodezije, točnije spojiti mobilni uređaj sa GNSS antenom.

7.1 Opis aplikacije

GNSS antena ima samo jednu funkciju, obrađivanje signala koje prima od satelita. Nakon obrade signala, antena preko Bluetooth veze ili kabelom šalje podatke prema GNSS kontroleru koji je spojen na antenu. Podaci koje GNSS antena pošalje kontroleru su u obliku ranije spomenute NMEA poruke. Jednom kada poruka stigne u kontroler, mogućnosti su beskonačne. Što kada

Slika 18: Zamjena kontrolera za Android aplikaciju, Geomatika [2012]

bi smo svrhu kontrolera mogli instalirati u obliku aplikacije na "bilo koji"[12] Android uređaj. U suštini, kontroler jer uređaj koji "zna" komunicirati sa

[12]Mora sadržavati Bluetooth modul te vezu sa Internetom

GNSS antenom preko Bluetooth veze te izvući potrebne podatke u svrhu daljnjeg računanja. Današnji mobiteli imaju već sve potrebne karakteristike (snagu, radnu memoriju) te mogućnosti (Bluetooth modul) da ih se koristi kao zamjena ili pomoćni uređaj prilikom snimanja GNSS/RTK metodom.

Osim same komunikacije kontrolera sa GNSS antenom, pojedini kontroleri imaju mogućnost rada u DGPS-u, tj. spajanje na referentni GNSS sustav koji šalje korekcije mjerenja. S time se dobiva povećana točnost samog mjerenja u ovisnosti o usluzi i točnosti koju taj sustav pruža.

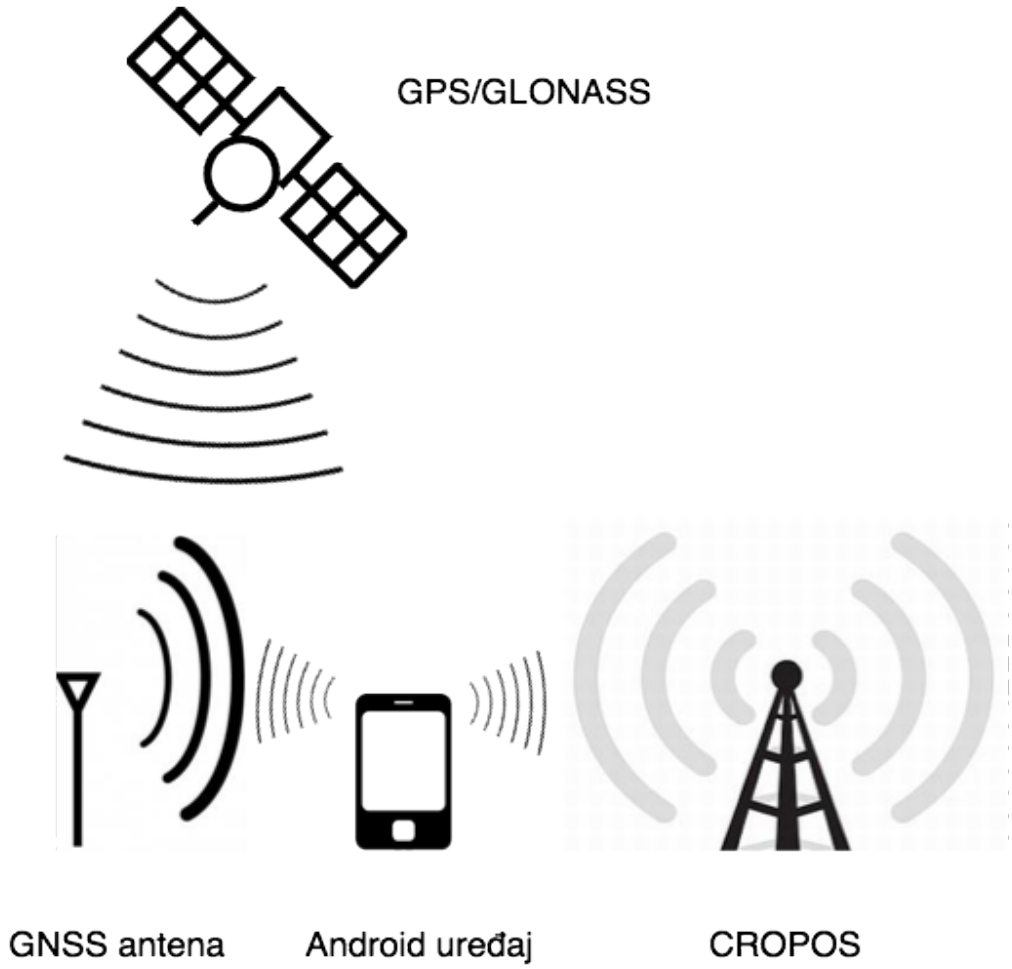
Upravo je to i svrha ove aplikacije, hardver koji postoji na GNSS kontrolerima iskoristiti kako bi zamijenio kontroler sa Android uređajem sa adekvatnim softverom sličnim na službenim kontrolerima. Time bi se dobila mogućnost odabira "kontrolera" pri kupovini GNSS kompleta te u određenim situacijama jednostavnost što se tiče kompatibilnosti sa ostalim antenama.

7.2 Infrastruktura

Za rad jedne ovakve aplikacije, potrebne su određene komponente sustava. Prvenstveno nam je potrebno:

- GNSS antena koja ima mogućnost korekcije podataka uz pomoć RTCM korekcijskih parametara. Sa ovakvom antenom postižemo povećanu točnost u odnosu na točnost deklariranu od strane samog proizvođača
- Minimalno 4 vidljiva GPS/GLONASS satelita
- Operabilan i funkcionalan referentni sustav za korekciju koordinata, u našem slučaju se to odnosi na CROPOS sustav (treba napomenuti da je za rad sa ovakvim sustavom ponekad potrebna autorizacija korisnika)
- Mobilni uređaj sa predinstaliranim Android operacijskim sustavom koji

u sebi ima mogućnost spajanja na Internet (zbog korekcijskih podataka) te Bluetooth modulom za komunikaciju sa antenom

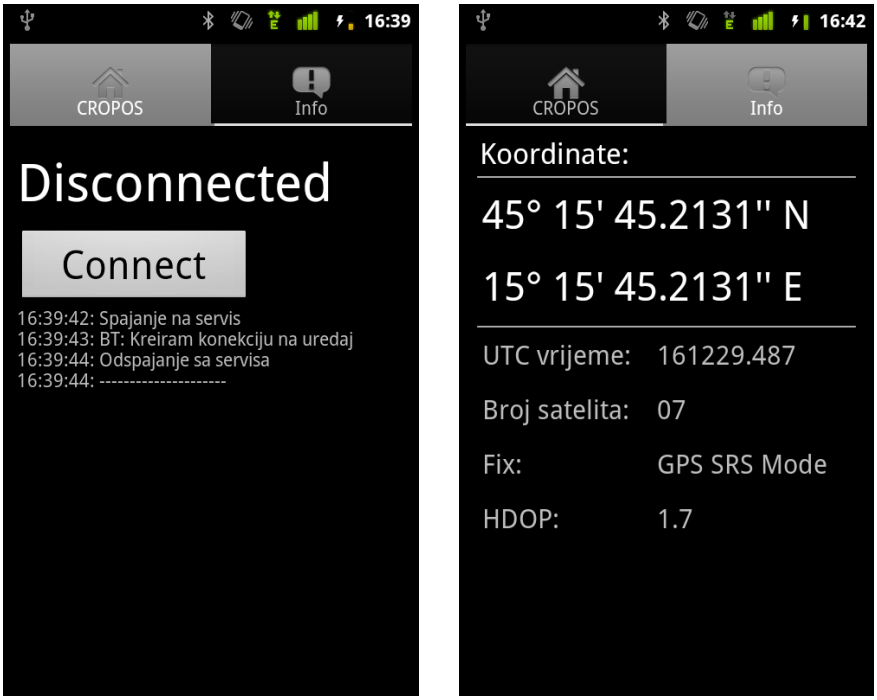


Slika 19: Infrastruktura potrebna za rad aplikacije

Treba napomenuti da aplikacije može raditi i sa antenama koji nemaju mogućnost korekcije podataka te se onda nije potrebno povezati na referentni sustav za korekcije. U tom slučaju bi se aplikacija koristila samo za očitavanje i spremanje podataka za daljnju obradu.

7.3 Komponente aplikacije

Vidljivi dio aplikacije sastoji se od dva prozora ili *Activity-a* (slika 20) ugnježdenih u jedan tzv. TabHost. TabHost je vrsta komponente koja služi za ugnježđivanje *Activity-a* u obliku tabova. Pritiskom na *Menu* gumb do-

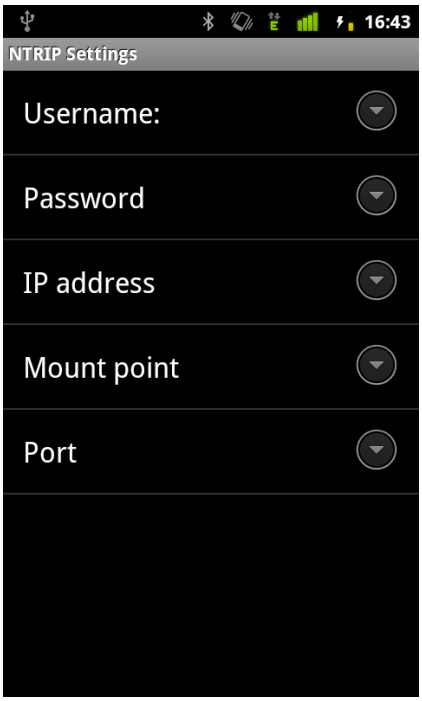


(a) Početni *Activity* (b) *Activity* sa podacima

Slika 20: Izgled aplikacije

lazimo do posebnog *Activity-a*, postavke (slika 21). U postavkama postoje dvije grupacije, jedna grupacija odnosi se na postavke vezane za povezivanje na referentni sustav za korekcije dok je druga grupacija vezana za komunikaciju sa GNSS antenom (odabir Bluetooth uređaja na koji ćemo se spojit). Prije svakog spajanja ili rada sa aplikacijom potrebno je upariti uređaj sa antenom. Postupak uparivanja se radi preko generalnih postavki samog uređaja. Jednom kada je uređaj uparen, u aplikaciji moramo izabrati taj isti uređaj na koji ćemo se spojiti preko postavki u aplikaciji.

Aplikaciju osim vizualnih dijelova sačinjavaju klase koje se brinu za rad

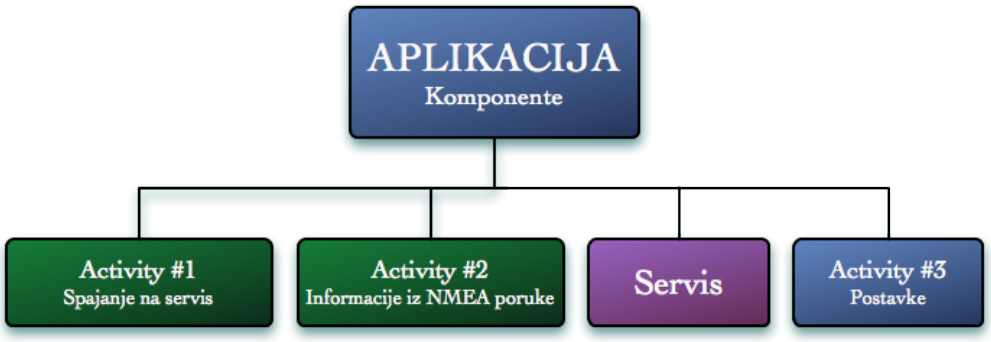


Slika 21: Postavke potrebne za rad sa CROPOS sustavom

same aplikacije. Klasa pod imenom *BluetoothDevicePreference.java* brine se za izlistavanje uparenih Bluetooth uređaja u postavkama aplikacije. Najvažnija klasa cijele aplikacije naziva se *CroposServis.java*. Kao što samo ime kaže radi se o pozadinskom servisu koja upravlja zadacima od uparivanje uređaja, uspostavljanja komunikacije sa antenom, prijenos podataka iz i u antenu, spajanje i komunikacija sa CROPOS servisom.

7.3.1 Pocetna.java

U datoteci *AndroidManifest.xml* možemo definirati koja od skupa klasa koja se nalaze u projektu će se pozivati kao početna klasa. U našem slučaju to je baš ova klasa u kojem je definirana prethodno spomenuta kontrola TabHost.



Slika 22: Komponente aplikacije

Klasu *Pocetna.java* nasljeđuje klasa *TabActivity* s čime smo usko definirali i proširili klasu. U metodi *onCreate()* koja se poziva tijekom prvog pozivanja te klase, definirali smo *Activity-e* koji će se prikazivati u tabovima. To su u našem slučaju klase *Main.java* i *Info.java*.

Primjer koda koji kreira tab objekte te ih dodjeljuje kontroli TabHost:

```
1 ...
2 intent = new Intent().setClass(this, Main.class);
3 spec = tabHost.newTabSpec("servis").setIndicator("
    CROPOS", res.getDrawable(R.drawable.ic_menu_home)).
    setContent(intent);
4 tabHost.addTab(spec);
5 ...
6 tabHost.setCurrentTab(0); //Postavlja pocetni tab
```

Da bi se u aplikaciji definiralo da je klasa *Pocetna.java* klasa koja se prva mora pozvati prilikom pokretanja aplikacije, potrebno je u datoteci *AndroidManifest.xml* definirati određene filtre.

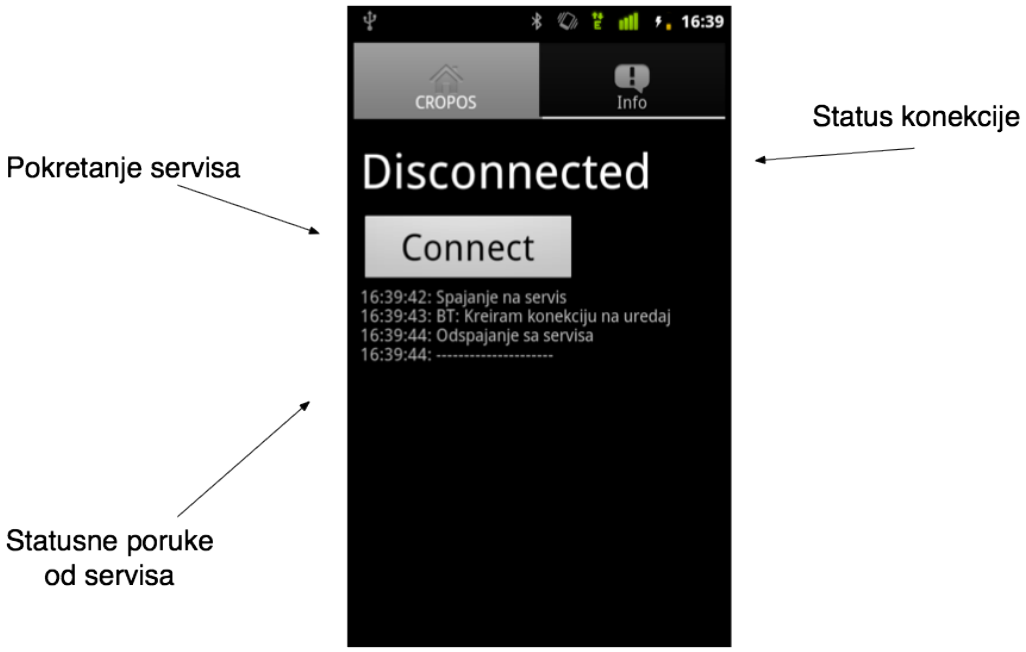
Isječak koda iz datoteke izgleda ovako:

```
1 <activity android:name=".Pocetna">
2   <intent-filter>
3     <action android:name="android.intent.action.MAIN"/>
4     <category android:name="android.intent.category.
        LAUNCHER" />
5   </intent-filter>
6 </activity>
```

Pomoću *indent-filter-a* definirali smo dodatne postavke za klasu ".Pocetna".

7.3.2 Main.java

Klasa *Main.java* je zapravo druga klasa koja se poziva prilikom pokretanja aplikacije jer je ona zapravo prvi tab koji se pokazuje u TabHost kontroli (slika 23). Na slici 23 prikazan je glavni *Activity* sa svim objašnjenjima. Kao

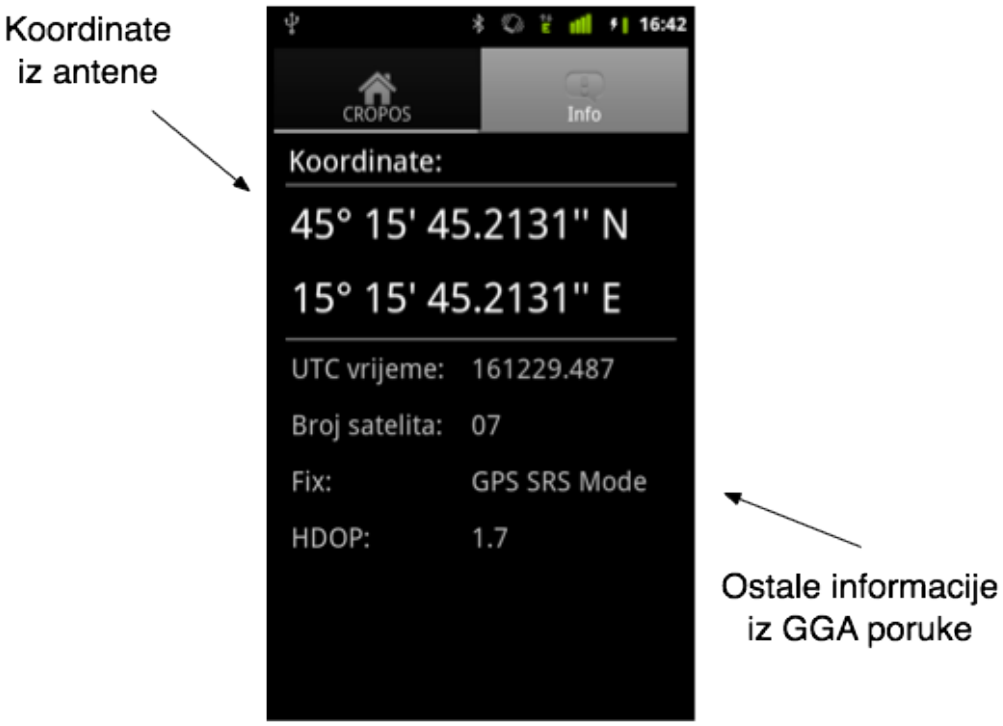


Slika 23: Tab koji se prvi pokazuje tijekom paljenja aplikacije

što možemo uočiti, radi se o prilično jednostavnom prikazu ekrana. Na vrhu se nalazi *TextView* kontrola koja ispisuje status pokrenute aplikacije. Ispod statusa konekcije nalazi se glavni gumb pomoću kojeg se cijeli servis pali ili gasi. Da bi smo znali što se zapravo događa u pozadini i u kojem dijelu izvršavanja se nalazi program, to sve možemo vidjeti u statusnim porukama koje se ispisuju ispod.

7.3.3 Info.java

Sljedeći tab pod nazivom "Info" zapravo je prezentacija *Info.java* klase (slika 24). Ovdje se isto radi o ugnježđenom *Activity-u* koji konstantno komunicira sa klasom "Main.java" kako se dogodi koje osvježenje podataka iz antene. *Activity* se sastoji od nekoliko *TextView-a*, svaki posebno za određenu infor-

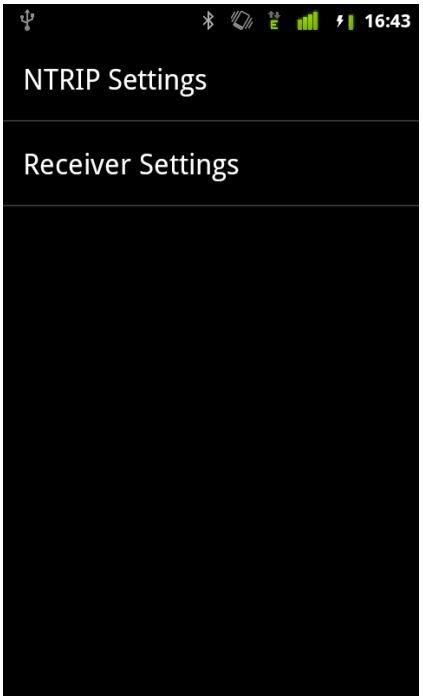


Slika 24: Tab koji prikazuje informacije

maciju.

7.3.4 PostavkeCroposa.java

Ako na glavnom *Activity-u* pritisnom hardversku tipku "Menu" izlistati će nam se menu izbornik sa jednom opcijom "Settings". Taj gumb zapravo poziva klasu "PostavkeCroposa.java". Pošto se ovdje radi o postavkama, Android za to ima poseban *Activity* za lakše kreiranje postavki a i koristi se iz razloga da sve postavke imaju unificiran dizajn diljem svih aplikacija na Android uređajima. *Interface* koji moramo naslijediti nad klasom *Activity* je *PreferenceActivity*. Da bi smo iskoristili interface, moramo "nadjačati" (eng. override) metodu *onCreate()*. Metoda *onCreate()*, kao što samo ime kaže, se poziva kada se prvi put pozove ta klasa. U toj metodi definiramo izgled samog prozora pozivanjem metode *addPreferencesFromResource()*.



Slika 25: Početni prozor postavki

```
1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     addPreferencesFromResource(R.xml.postavkecroposa);
```

Ono što metoda *addPreferencesFromResource()* zapravo radi jest da prikaz (eng. layout) prozora definira/iscrtava uz pomoć xml datoteke, što je u ovom slučaju datoteka "postavkecroposa.xml".

Isječak xml datoteke *postavkecroposa.xml*:

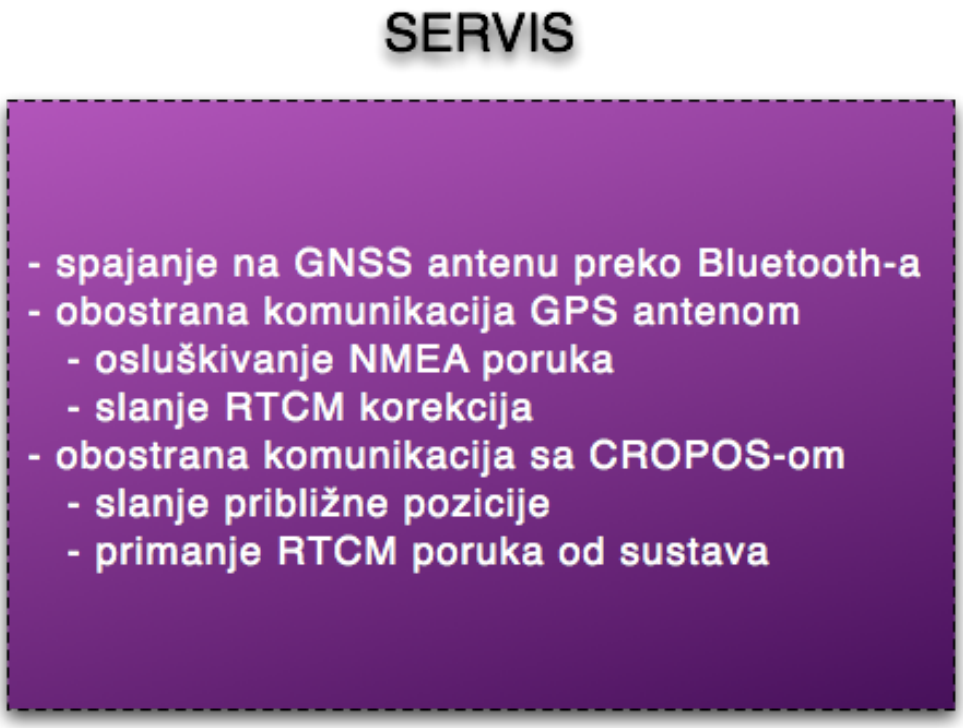
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen>
3    <PreferenceScreen
4       android:key="ntripsettings"
5       android:title="NTRIP Settings" >
6    <EditTextPreference
7        android:defaultValue="195.29.118.122"
8        android:dialogTitle="IP Address (x.x.x.x)"
9        android:key="ipaddress"
10       android:title="IP address" >
11   </EditTextPreference>
12 ...
```

Android preferira tzv. odvojeno programiranje (eng. code behind), što u ovom slučaju znači da je definiranje izgleda *Activity-a* izdvojeno od samoga koda u xml datotekama. *Code behind* programiranjem odvajamo dizajnere aplikacije od programera koda.

7.3.5 CroposServis.java

Ono što zapravo čini cijelu aplikaciju nalazi se u ovoj klasi. Kao što smo prije spomenuli, radi se o servisu. Servis je vrsta potprograma koji se izvršava u pozadini, najčešće u zasebnoj niti (eng. thread) kako ne bi opterećivala glavnu nit koja iscrtava korisničko sučelje. Prednost servisa je činjenica da se na njega mogu spojiti i ostali dijelovi aplikacije pa čak i neke odvojene

aplikacije instalirane na Android uređaju. S ovime dobivamo na modularnosti aplikacije, a i mogućnošću izrade drugih vrsta aplikacije koje bi koristile svrhu tog servisa.



Slika 26: Zadaci servisa

7.4 Princip rada aplikacije

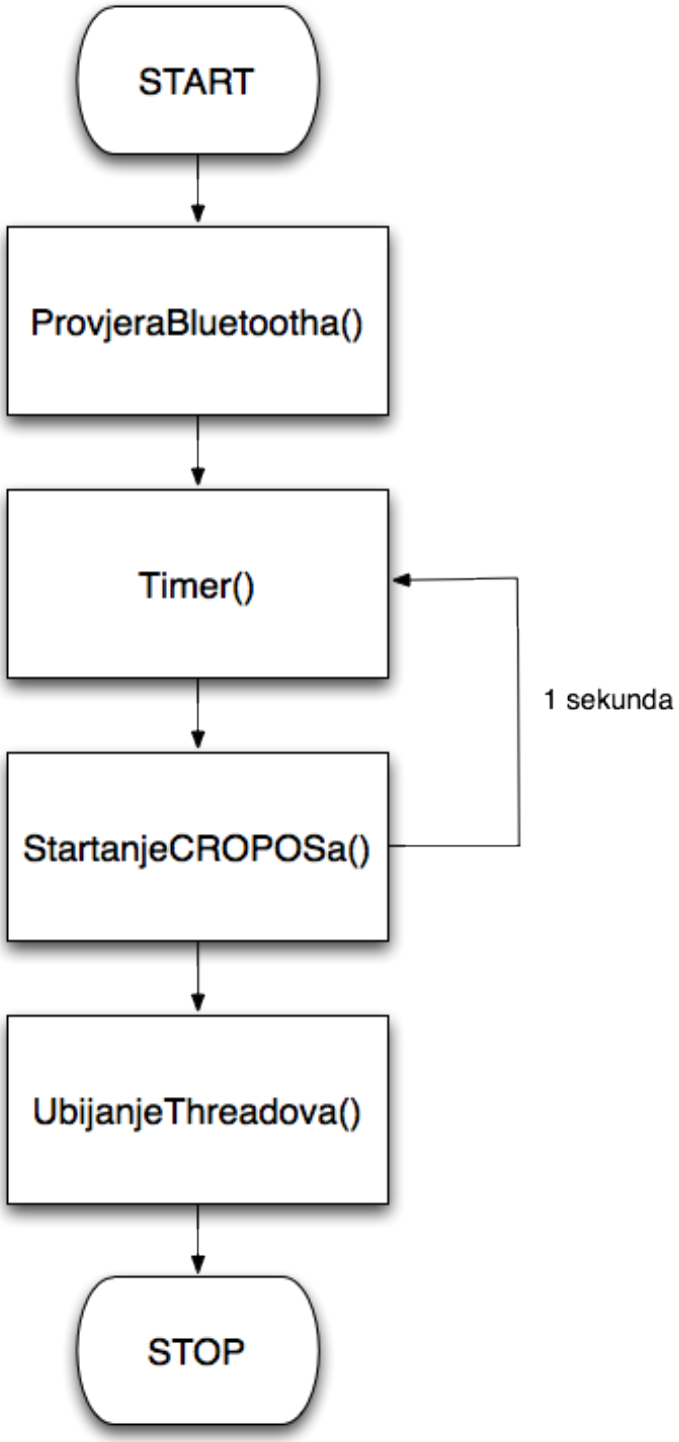
Sve počinje paljenjem aplikacije i prikazivanjem glavnog *Activity-a* (slika 23). Paljenje servisa izvršavamo pritiskom na glavni gumb (slika 27). Prije samog startanja servisa, izvršavaju se provjere koje su bitne za rad same aplikacije. To su:

- Provjera prisutnosti Bluetooth modula
- Provjera upaljenosti Bluetooth-a
- Uparenost uređaja sa GNSS antenom

Prilikom pokretanja aplikacije, servis je ugašen. Na glavnom gumbu vrši se kreiranje servisa te izvršava se postupak spajanja (eng. bind) kada su zadovoljeni prethodni uvjeti. Kreiranjem servisa poziva se metoda *onCreate* te se učitavaju sve postavke programa koje smo prethodno upisali i namjestili.

```
1 SharedPreferences preferences = PreferenceManager
2  .getDefaultSharedPreferences(getBaseContext());
3  pUsername = preferences.getString("username", "test");
4  pPassword = preferences.getString("password", "test");
5  pIpAddress = preferences.getString("ipaddress", "
       0.0.0.0");
6 ...
```

Prilikom spajanja na servis, servis šalje objekt tipa *IBinder* koji nam služi za komunikaciju sa različitim nitima programa. Kada smo primili objekt pomoću kojeg možemo komunicirati sa servisom, njemu sad isto prosljeđujemo objekt za komuniciranje, ali u drugom smjeru, ovaj put objekt vrste *Messenger*. Ovim postupkom smo osigurali da te dvije različite niti mogu komunicirati jedna sa drugom u oba smjera.



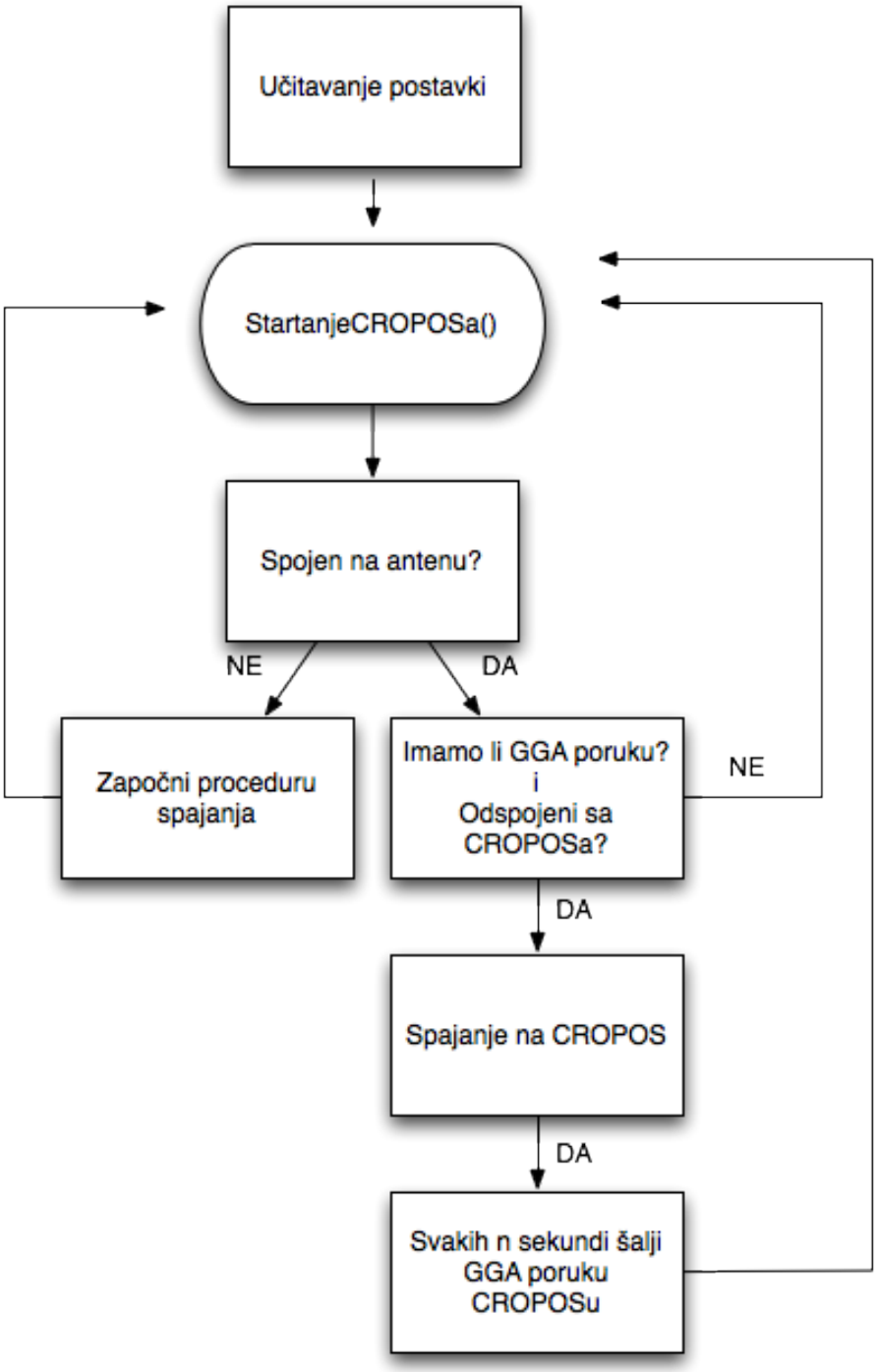
Slika 27: Diagram toka programa

```
1 @Override
2  public void onServiceConnected(ComponentName name,
      IBinder service) {
3  mServiceMessenger = new Messenger(service);
4  try {
5      Message msg = Message.obtain(null,
6      CroposServis.MSG_REGISTRIRAN_CLIENT);
7      msg.replyTo = mMessenger;
8      mServiceMessenger.send(msg);
```

Sada kada imamo objekt pomoću kojeg možemo komunicirati sa servisom, pozivamo objekt vrste *Timer* koji je konfiguriran tako da konstantno poziva glavnu metodu *StartanjeCROPOSa()* (slika 29.) u intervalu od 1 sekunde. Razlog pozivanja metode svake sekunde leži u obostranoj komunikaciji sa GNSS antenom i CROPOS servisom. U toj jednoj sekundi, servis konstantno čita komunikacijske protokole te ih prosljeđuje dalje na obradu.
StartanjeCROPOSa() je metoda koja u sebi sadrži tri uvjeta:

- Da li je uspostavljena komunikacija sa antenom?
- Da li je primljena ispravna GGA poruka?
- Da li je proteklo n sekundi od spajanja na CROPOS servis?

Provjerom uvjeta, izvršavaju se odgovarajuće akcije. Prvim pokretanjem metode, pretpostavka je da ne postoji još komunikacija između uređaja i antene, stoga moramo pokrenuti odgovarajuću proceduru. U zasebnoj niti prvo se "otvara" RFCOMM *socket* za komunikaciju sa antenom, a nakon toga otvaraju se kanali za obostranu komunikaciju sa antenom. Obostrana komunikacija sa antenom nam je potrebna zbog toga što antena svaku sekundu



Slika 28: Objašnjenje toka servisa nakon paljenja

šalje podatke u obliku NMEA poruke, a mobilni uređaj će svaku sekundu slati korekcije koje će primati od CROPOS sustava prema anteni.

U sljedećem isječku koda pokazati ćemo kako se uspostavlja komunikacija sa antenom te slanje i primanje podataka:

```
1 ...
2 device = btAdapter.getRemoteDevice(macAddress);
3 UUID MY_UUID = UUID
4     .fromString("00001101-0000-1000-8000-00805F9B34FB");
5 tmpBluetoothSocket = device
6     .createRfcommSocketToServiceRecord(MY_UUID);
7 socket = tmpBluetoothSocket;
8 socket.connect();
```

U drugoj liniji koda možemo vidjeti da smo stvorili novi objekt *BluetoothDevice* tako što smo ga identificirali preko MAC adrese (MAC adresu smo dobili onda kada smo se uparili sa antenom). Prilikom kreiranja *socket-a*, moramo definirati vrstu protokola. Iz linije 5 vidljivo je da smo definirali RFCOMM protokol pomoću *Service ID-a*. I na kraju povezujemo uređaje naredbom *socket.connect()*.

```
1 ...
2 tmpInputStream = socket.getInputStream();
3 tmpOutputStream = socket.getOutputStream();
4 byte[] buffer = new byte[1024];
5 while (true) {
6   bytesread = inputStream.read(buffer);
7 ...
```

Komunikacijski kanal je uspostavljen. Podaci koji se prenose preko kanala, moramo proslijediti preko tzv. *Stream-a*. Podaci koji ulaze u uređaju će se nalaziti u *InputStream-u*, a podaci koje ćemo slat, prenositi će se preko *OutputStream-a*. Kao što je vidljivo iz koda, u beskonačnoj petlji osluškuje se dolazni podaci veličine 1024 bajtova. Ta petlja će se vrtjeti sve dok ne zaustavimo servis.

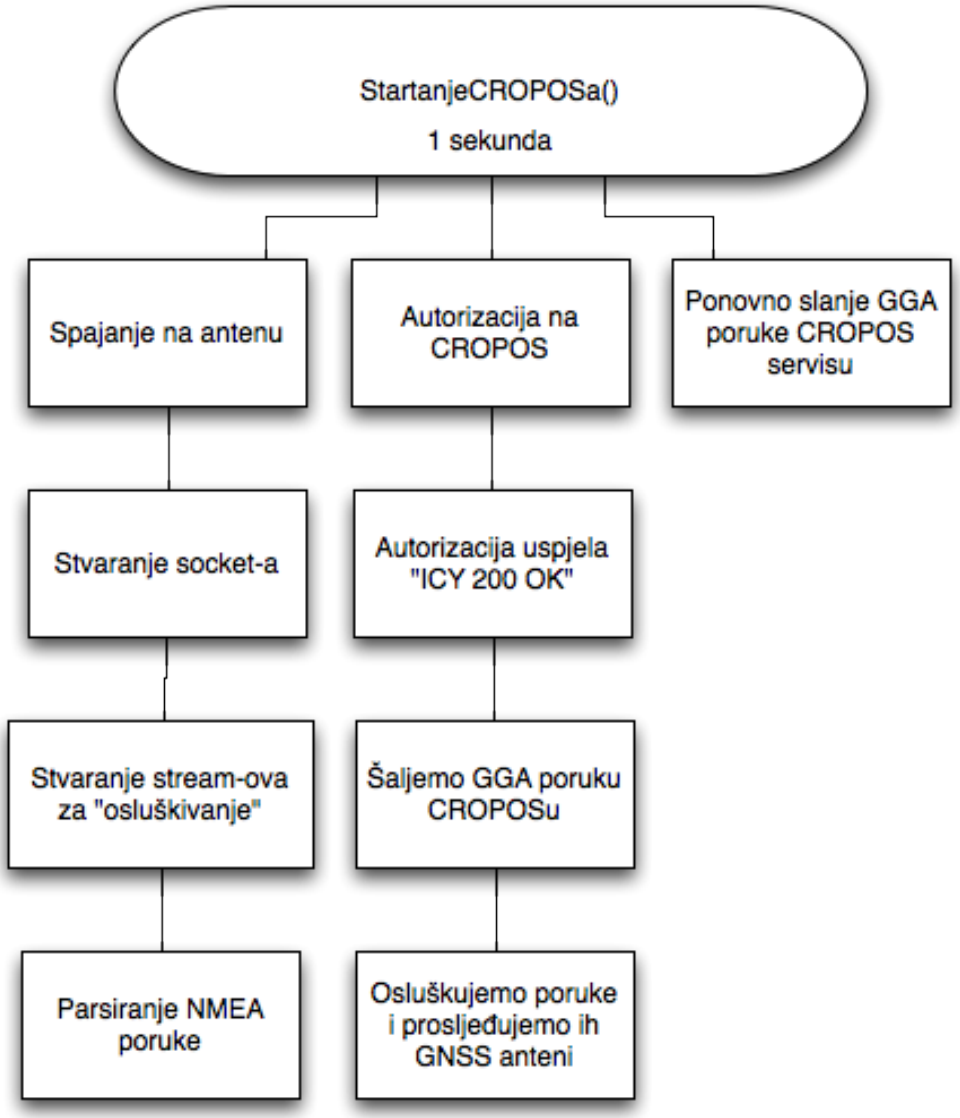
Što se tiče prijenosa podataka prema GNSS anteni, za to ćemo koristiti *OutputStream*:

```
1 ...
2 outputStream.write(buffer);
3 ...
```

S ovime je prvi uvjet u metodi *StartanjeCROPOSa()* ispunjen (slika 29.). Nakon svakog dobivenog podatka iz antene, on se šalje na daljnju obradu, točnije, na izdvajanje GGA linije iz NMEA poruke. Prisjetimo se da u NMEA poruci, osim GGA linije, postoji još nekolicina poruka koju šalje GNSS antena (GLL, GSA, GSV...). Svim tim poruka jedna stvar je zajednička, svi oni su odvojeni pomoću dva posebna znaka, "$\backslash r \backslash n$". Izdvajanje GGA linije od ostalih ćemo upravo pomoću tih znakova učiniti i provjeravati koji od izdvojenih linija počinje znakovima "GPGGA". Izdvojena GGA poruka uvjet je za pristupanje CROPOS servisu.

Za rad sa CROPOS servisom, potrebna je autentikacija u obliku korisničkog imena i zaporke koju smo prethodno unijeli u postavkama aplikacije. Osim toga, potrebna je IP adresa te port za spajanje na servis preko Internet-a. Postupak spajanja preko *socket-a* na servis identičan je postupku spajanja na Bluetooth uređaj.

```
1 ...
2 SocketAddress socketAddress = new
```



Slika 29: Detaljni opis metode *StartanjeCROPOSa()*

```
3   InetSocketAddress(ipAddress, port);
4 croposSocket = new Socket();
5 croposSocket.connect(socketAddress, 10 * 1000);
6 ...
7 croposInputStream = croposSocket.getInputStream();
8 croposOutputStream = croposSocket.getOutputStream();
```

U zaseban objekt, *SocketAddress*, kao parametre konstruktora unijeti ćemo IP adresu CROPOS servisa te port. Da bi komunikaciju upostavili, moramo kreirati objekt *Socket* što smo izvršili u liniji 4. Ako primjetimo, u metodi *connect()* (linija 5) kao drugi parametar definirali smo tzv. *timeout* od 10 sekundi (milisekunde su ulazni parametar). S time smo osigurali da prilikom uspostavljanja veze između mobilnog uređaja i servisa, ako se dogodi greška prilikom spajanja pričeka sljedećih 10 sekundi do sljedećeg pokušaja. Jednom kada je veza uspostavljena, proslijeđujemo objekte vrste *Stream* kako bi uspostavili obostranu komunikaciju sa servisom.

```
1 String requestString = "GET /" + mountPoint
2         + " HTTP/1.0\r\n";
3 requestString += "User-Agent: NTRIP CroposClientFrane\r
    \n";
4 requestString += "Accept: */*\r\n";
5 requestString += "Connection: close\r\n";
6 requestString += "Authorization: Basic "
7         + ToBase64(username + ":" + password);
8 requestString += "\r\n";
```

Nakon uspostavljene komunikacije sa servisom, moramo se autorizirati. U NTRIP protokolu definirana je autorizacijska poruka pomoću koje klijent

vrši autorizaciju. Izgled takve jedne poruke prikazan je u gornjem primjeru koda. *User-Agent* je proizvoljno ime klijenta koje se koristi za identifikaciju. Važnu stvar koju treba primjetiti u linij 7 je *ToBase64()* metoda. Kako se svi podaci šalju preko ne osigurane komunikacije, korisničko ime i zaporka se kriptiraju pomoću Base64 algoritma radi sigurnosti.

```
1 // Poslat preko stream-a
2 croposOutputStream.write(requestString.getBytes());
```

Autorizacijska poruka formatirana na ovaj način, šalje se preko *OutputStream-a* prema servisu.

```
1 byte[] buffer = new byte[4096];
2 int read = croposInputStream.read(buffer, 0, 4096);
3 while (read != -1 && runningThread) {
4   byte[] tempData = new byte[read];
5   System.arraycopy(buffer, 0, tempData, 0, read);
6   handler.sendMessage(handler.obtainMessage(
7     MSG_NETWORK_GOT_DATA, tempData));
8   read = croposInputStream.read(buffer, 0, 4096);
9 }
```

Poruka koju CROPOS servis mora poslati ako je autorizaciju uspješno izvršena počinje statusnim kodom "ICY 200 OK". Slično kao i kod komunikacije sa Bluetooth-om, veza će se održavati sve dok servis bude slao podatke u obliku bajtova. Dobiveni podaci se šalju na obradu kako bi se ustanovila vrsta povratne informacije. CROPOS kao povratnu informaciju može poslati 3 tipa: statusni kod, RTCM korekciju ili Sourcetable.

```
1 ...
2 // slanje poruke CROPOSu
```

```
3 croposOutputStream.write(msg.getBytes());
```

Nakon uspješno provedene autorizacije, sljedeće što servisu zapravo moramo poslati je naša lokacija kako bi servis na temelju toga mogao generirati VRS (eng. *Virtual Reference Station*) te poslati nam odgovarajuće korekcije. Pomoću *OutputStream-a* šaljemo mu ispravnu GGA poruku te se opet vraćamo u prethodnu petlju gdje osluškivamo poruke od servisa i čekamo RTCM korekcije. Korekcije koje dobijemo direktno prosljeđujemo u GNSS antenu bez dodatnog analiziranja.

```
1 byte[] buffer;
2 outputStream.write(buffer);
3 ...
```

U postavkama postoji jedna opcija kojom se definira vrijeme u sekunda koje mora proći kada ćemo ponovno poslati CROPOS-u našu lokaciju u obliku GGA poruke. Isječak iz metode *StartanjeCROPOSa()* koja prikazuje zadnji uvjet, prikazan je u kodu ispod.

```
1 if (NTRIPTicksSinceGGASent != 0 &&
2   NTRIPTicksSinceGGASent % pPonavljanjeSlanjaGGA == 0
3   && MostRecentGGA.length() > 5) {
4     SendDataToNetwork(MostRecentGGA);
5 }
```

8 Daljnji koraci

Opisana aplikacija neuspješno je testirana s dvjema Trimble GNSS R8 antenama koje mi je fakultet ustupio u svrhu diplomskog rada. Jedna antena sadrži stariju verziju softvera (eng. firmware) 2.135 koja se pokazala kao ne-

moguća za bilo kakvo testiranje zbog njezine nemogućnosti prihvaćanja bilo kakve komunikacije s ostalim uređajima koji nisu od tvrtke Trimble. Uređaj bi se nakon konfiguracije odmah vratio na početno stanje ukoliko bi se samo na trenutak izgubila Bluetooth veza. Kod antene sa novijim softverom (4.1), riješen je problem postavki, ali se javio jedan drugi relativno čudan problem. U trenutku slanja RTCM korekcija s Android uređaja prema anteni, na anteni se pokazuje indikacija (u obliku treptajuće LED diode) da prima korekcije, ali u GGA poruci ne pokazuje da ih koristi u svrhu korigiranja koordinata.

U trenutku pisanja diplomskog rada, na fakultetu ne postoji ni jedan drugi GNSS uređaj na kojem bi se moglo izvršiti testiranje aplikacije u potpunosti.

Ovakva aplikacija predstavlja osnovnu podlogu za bilo kakva daljnja računanja i obradu podataka iz GNSS antene. Aplikacijom je pokazano da Android platforma može biti dostojna zamjena dosadašnjim kontrolerima koji su bazirani na zastarjelim tehnologijama koje se više ne razvijaju. Pravi primjer su Trimble kontroleri koji se baziraju na Windows Mobile 5.x/6.0 operativnom sučelju. Prelaskom na Android bio bi logičan korak bas zbog svih prednosti koje ovaj sustav donosi. Jedna stvar koja bi uveliko povećala efikasnost prikupljanja, razmjenjivanja i razmjenu mjerenih podataka je spremanje podataka u oblak, tzv. *cloud storage*. Direktnim spremanjem opažanih mjerenja u cloud omogućila bi se lakša i trenutna razmjena podataka na terenu i uredu. Spremljenim podacima mogli bi se pristupiti bilo kada i bilo gdje da se nalazili. Još jedna prednost spremanja podataka u oblak je jedna vrsta sigurnosti. Bilo što da se dogodi uređajima na kojima obavljamo spremanje podataka, uvijek možemo biti sigurni da se podaci koje smo spremili nalaze na nekom udaljenom računalu.

Android operacijski sustav dolazi sa mnoštvo već ugrađenih funkcija i

aplikacija koje bi mogli iskoristiti u svrhu lakšeg pregledavanja podataka. Jedna od tih je zasigurno interaktivna karta koja bi se koristila kao podloga svim vrstama mjerenja u svrhu lakšeg snalaženja u prostoru, a i kasnije u uredu. Današnji prijenosni uređaji postali su toliko moćni da se sve više približavaju mogućnostima koje imamo na stolnim računalima. Skoro svaki mobilni uređaj omogućava spajanje na različite uređaje kao što su projektori, televizori, monitori i različiti periferni uređaji. Spajanjem Android uređaja na projektor mogli bi prezentirati nedavno snimljene podatke širem broju ljudi direktno sa mobilnog uređaja bez dodatnih obrada i prebacivanja podataka. Skoro svaki prijenosni uređaj danas posjeduje kameru koja bi na mogla poslužiti da zabilježimo stvari koje su nam od važnosti za određena mjerenja. Sve ovo je nešto što Android uređaji već posjeduju samo ih moramo iskoristiti u našu korist.

Što se tiče ostalih aplikacija, slične aplikacije već postoje na tržištu, primjerice poput *ESRI ArcGIS for iPhone*, ali niti jedna od trenutno dostupnih aplikacija nema mogućnost prikupljanja DGPS podataka tj. korištenja GPS korekcija.

9 Zaključak

Zbog svoje otvorenosti, Android je postao mobilna platforma sa velikim mogućnostima povezivanja sa ostalim uređajima, komunikacije sa različitim servisima te iskorištavanju različitih resursa što mobilni uređaji nude. Upravo radi tih karakteristika moguće je napraviti aplikaciju kao jednu vrstu NTRIP klijenta koji kasnije može poslužiti kao dostojna zamjena za GNSS kontroler koja će raditi na širokom spektru Android uređaja te sa različitim vrstama GNSS antena.

Literatura

Željko Bačić. *Satelitsko pozicioniranje.* Geodetski fakultet, Kačićeva 26, Zagreb, 2009a.

Željko Bačić. *Satelitsko pozicioniranje.* Geodetski fakultet, Kačićeva 26, Zagreb, Prosinac 2009b.

DGU. Sourcetable od cropos-a. http://195.29.118.122:2101/, Lipanj 2012.

FER. Programski jezik java. http://fly.cc.fer.hr/ zox/diplomski/Programski_Jezik_JAVA.html, Travanj 2010.

GDC. *Ntrip Protocol Version 1.0.* GNSS Data Center, Njemačka, Rujan 2004.

GDC. Gnss data center. http://igs.bkg.bund.de/, Svibanj 2012.

Geomatika. Trimble kontroler. http://www.geomatika-smolcak.hr/baza/smolcak/kat/a_363_TSC3%20studio%20front%20107.jpg, Svibanj 2012.

Google. Android logo. https://upload.wikimedia.org/wikipedia/commons/d/d7/Android_robot.s Svibanj 2012.

Java. Java logo. http://www.egilmela.com/wp-content/uploads/2010/10/322px-java_logosvg1.png, Listopad 2010.

Kowoma. Gps satellite orbits. http://www.kowoma.de/en/gps/orbits.htm, Travanj 2009.

Tino Krečak. *GPS - Galileo - Glonass.* Geodetski fakultet, Kačićeva 26, Zagreb, Prosinac 2010.

Miljenko Lapaine, Miroslava Lapaine, and Dražen Tutić. Gps za početnike. http://www.kartografija.hr/old_hkd/obrazovanje/prirucnici/gpspoc/gpspoc.htm, Svibanj 2012.

Wires Are Obsolete. The missing manual: Android bluetooth rfcomm. http://wiresareobsolete.com/wordpress/2010/11/android-bluetooth-rfcomm/, Studeni 2010.

Milan Rezo and Bačić Željko. *CROPOS – Kvaliteta sustava.* Geodetski fakultet, Kačićeva 26, Zagreb, Lipanj 2008.

Besplatni seminarski radovi. Programski jezik java. http://www.besplatniseminarskiradovi.com/Programiranje/ProgramskiJezikJava.htm, Prosinac 2011.

Wikipedia. Android operacijski sustav. http://hr.wikipedia.org/wiki/Android_(operacijski_sustav), Srpanj 2012a.

Wikipedia. Android programiranje. https://en.wikipedia.org/wiki/Android_software_development, Svibanj 2012b.

Wikipedia. Bluetooth. https://hr.wikipedia.org/wiki/Bluetooth, Lipanj 2012c.

Wikipedia. Eclipse programski paket. https://en.wikipedia.org/wiki/Eclipse_(software), Kolovoz 2012d.

Wikipedia. Google play. http://hr.wikipedia.org/wiki/Google_Play, Svibanj 2012e.

Wikipedia. Global positioning system. https://hr.wikipedia.org/wiki/Global_Positioning_System, Lipanj 2012f.