Analiza troškova voznog parka zasnovana na obradi GPS podataka

Vogel, Alan

SVEUČILIŠTE U ZAGREBU FAKULTET
PROMETNIH ZNANOSTI

Alan Vogel

ANALIZA TROŠKOVA VOZNOG PARKA ZASNOVANA NA OBRADI GPS PODATAKA

ZAVRŠNI RAD

Mentor: prof. dr. sc. Tonči Carić

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU FAKULTET
PROMETNIH ZNANOSTI ODBOR ZA
ZAVRŠNI RAD

Zagreb, 24. travnja 2017.

Zavod: Zavod za inteligentne transportne sustave
Predmet: Baze podataka

ZAVRSNI ZADATAK br. 4040

Pristupnik: Alan Vogel (0135237989)
Studij: Inteligentni transportni sustavi i logistika
Smjer: Inteligentni transportni sustavi

Zadatak: Analiza troškova voznog parka zasnovana na obradi GPS podataka

Opis zadatka:

Vozila tvrtke koja se prate pomoću GPS uređaja generiraju veliku količinu podataka koje je potrebno obraditi i izračunati troškove korištenja vozila. Potrebno je izraditi aplikaciju koja će omogućiti analizu troškova temeljem putnih naloga, GPS tragova vozila i pravila tvrtke o obračunu troškova i upotrebi vozila za službene ili osobne potrebe. Aplikacija mora uvažiti geografska i vremenska pravila tvrtke o obračunu troškova, kao što su izuzimanje vožnji van grada ili županije ili vožnje neradnim danom i praznikom.

Zadatak uručen pristupniku: 28. travnja 2017.

Mentor: Predsjednik povjerenstva za završni ispit:

_____ _____

prof. dr. sc. Tonči Carić

SVEUČILIŠTE U ZAGREBU FAKULTET PROMETNIH ZNANOSTI

ZAVRŠNI RAD

ANALIZA TROŠKOVA VOZNOG PARKA ZASNOVANA NA OBRADI GPS PODATAKA

FLEET COST ANALYSIS BASED ON GPS DATA PROCESSING

Mentor: prof. dr. sc. Tonči Carić

Student: Alan Vogel
JMBAG: 0135237989

Zagreb, rujan 2017.

ANALIZA TROŠKOVA VOZNOG PARKA ZASNOVANA NA OBRADI GPS PODATAKA

SAŽETAK

Analiza troškova voznog parka zasnovana na obradi GPS podacima

U ovome završnom radu prikazan je način pohranjivanja velike količine GPS podataka u Microsoft SQL server, izračun troškova i udaljenosti prijeđenog puta vozila te prikaz ruta u Google Earth-u. Za potrebe izračuna troškova izrađena je aplikacija pomoću Microsoft Visual Studio 2015 koja se povezuje sa SQL serverom te stvara kml datoteku koja služi za prikaz rute vozila u Google Earth-u na području Zagreba i okolice.

KLJUČNE RIJEČI: troškovi voznog parka; SQL; Visual Studio; Google Earth; rute vozila

SUMMARY

In this work has been shown how to save and processing large quantities of data to Microsoft SQL server, calculate the cost and kilometrage of vehicles. Data of the vehicle routes is display in Google Earth using application that was made in Microsoft Visual Studio 2015. For the purpose of this work the application can connect to SQL server where it stores large data based on GPS tracking vehicles and create a kml file which is used to display vehicle routes in Google Earth on areas of Zagreb and surroundings.

KEYWORDS: vehicle fleet cost; SQL;Google Earth;Miscrosoft Visual Studio; vehicle routes

SADRŽAJ

1. UVOD

Zahvaljujući globalizaciji ukinuta su mnoga ograničenja između država čime je bitno postignut lakši prijevoz robe, ljudi i informacija. Zahvaljujući tome povećana je robna razmjena i suradnja država na svim područjima. Kako se je tržište povećavalo stupanjem država u mnoge saveze, slobodnom tržištu te deregulaciji tako se počeo mijenjati i odnos ponude i potražnje gdje je za raznim uslugama bila velika potražnja, a došlo je i do kvalitetnih pružanja usluga i roba. Problem koji se javlja kod velikih tržišta je kako prenijeti uslugu (robu) do cilja, a da minimiziramo troškove. Troškovi voznog parka ovise o raznima elementima kao što su ljetni i zimski mjeseci, gusto naseljena mjesta i velike udaljenosti.

Tijekom ljetnih i zimskih mjeseci vidljivi su problemi koji nastaju sa slabim protokom vozila na određenim dionicama kada je prometni volumen na cesti veći od kapaciteta koji prometnica nudi. Taj problem je dosta velik kod velikih gradova zbog guste naseljenosti, a vidljiv je i u srednjim gradovima poput grada Zagreba. Iako su se ispitivanja vršila unutar Zagreba i njegovoj bliži također postoji problem kod velikih udaljenosti zbog odabira raznih prometnih modova i puta kako bi troškovi bili minimalni.

Praćenjem vozila i prikupljanjem GPS podataka moguća su mnoga ispitivanja u kojima se mogu odrediti vremenski period putovanja, udaljenosti koje je jedno ili više vozila obavilo od točke A do točke B. Također mogu se vršiti ispitivanja o zagušenju pojedine prometnice preko pada ili rasta brzine što uvelike pomaže kod rješavanja problema sa zagušivanjem u velikim urbanim sredinama. Beneficije u praćenju vozila su velika jer bitno smanjuju troškove zbog koji raste profit tvrtkama, pa s time raste i kvaliteta pružanja usluge i robe. Najvažnije beneficiju kod upravljanja voznim parkom su povećanje efikasnosti i sigurnost radnika, smanjenje goriva, pa s time se smanjuje zagađenje okoliša, povećanje produktivnosti i vijek trajanja voznog parka.

Najveći nedostatak u gradu Zagrebu, općenito Republici Hrvatskoj je nedostatak sredstva za ulaganje u nove tehnologije i interesom za prometnim centrom u gradu Zagrebu čime bi došlo do velikog poboljšanja i efikasnosti u upravljanju prometom i sigurnosti sudionika na prometnim cestama.

Za potrebe ovog rada izrađena je aplikacija u Microsoft Visual Studio, programski jezik C# koja služi za pohranu velikih količina podataka koji su prikupljeni nad praćenim vozilima u SQL relacijsku bazu. Također računa troškove vozila, tvrtke, vozača i udaljenosti koje su vozila obavila pri obavljanju svoje rute. Uz to aplikacija stvara kml datoteku za geografski prikaz podataka n Google Earth-u kako bi korisnik mogao vidjeti gdje se i kada vozilo kretalo.

Aplikacija ima mogućnost povezivanja Visual Studija sa SQL-om te uz to stvara kml datoteku koju sprema u debug folder.

Rad je podijeljen u 7 cjelina:

1) Uvod
2) Obrada GPS podataka dobivenih iz uređaja za praćenje vozila
3) Pohrana podataka u relacijsku bazu podataka
4) Analiza kretanja vozila temeljen GPS tragova
5) Izračun troškova voznog parka tijekom službenog radnog vremena
6) Prikaz ruta i troškova
7) Zaključak

U prvom poglavlju definirani su beneficije upravljanja prometom, praćenje vozila te nedostaci. Objašnjeni su neki od problema troškova u urbanim sredinama i pružanju usluge na velikim udaljenostima. Također je objašnjena struktura završnog rada.

U drugom poglavlju se prikazuje i objašnjava funkcioniranje GPS sustava te se daje uvid nad prikupljenim podacima vozila i njihova obrada. Također se objašnjava određivanje pozicije vozila.

U trećem poglavlju se objašnjava postupak stvaranja relacijske baze podataka. Prikazan je prikaz ER modela i njegove pretvorbe u relacijski model. Svi dijelovi i cijeli postupak pretvorbe iz ER modela u relacijski model je prikazan i objašnjen. Prikazan je i kod SQL skripta pomoću koje su rađene tablice i njeni atributi koje su potrebni za pohranjivanje podataka.

U četvrtom poglavlju se opisuje kako izgledaju GPS segmenti na prometnici te objašnjava njihova netočna pozicija na digitalnoj karti. Opisuje se rad map matching algoritma kao rješenje nepreciznosti GPS sustava i ispravljanja to jest postavljanja pozicije vozila na prometnu mrežu čime se postiže točni prikaz na digitalnoj karti.

U petom poglavlju su objašnjene sve formule koje su bile potrebne za izračun troškova voznog parka i prijeđenog puta vozila za jednu ili više ruta. Također su prikazani kodovi koji su rađeni u programskom jeziku C# za potrebe izračuna. Kodovi su objašnjeni kako funkcioniraju i računaju troškove i prijeđeni put vozila.

U šestom poglavlju su prikazani svi osnovni troškovi (dnevni, mjesečni i godišnji), troškovi za vrijeme službenog radnog vremena (troškovi tvrtke i vozača). Troškovi su prikazani grafikonima u kojima su uzimati pojedini podaci kako bi se mogla vršiti analiziranja. Također je objašnjeno na koji način možemo prikazati rutu vozila te su prikazane slike kako izgledaju na Google Earth-u.

U sedmom poglavlju dan je zaključak ovoga rada te osvrt na buduća istraživanja. Predloženi su prijedlozi za proširenje funkcionalnosti aplikacije i raznih mogućnosti koji se mogu istražiti preko dobivenih podataka.

2. OBRADA GPS PODATAKA DOBIVENIH IZ UREĐAJA ZA PRAĆENJE VOZILA

Danas postoje mnogo informacijskih-komunikacijskih uređaja pomoću kojih možemo skupljati velike količine podataka. Takvi uređaji se mogu nalaziti unutar nekog prijevoznog sredstva ili izvan njega. Kada govorimo o prikupljanju podataka o vozilima tada uređaje odnosno senzore koje se nalaze izvan vozila možemo podijeliti na:

a) Induktivna petlja
b) Radarski senzor
c) Ultrazvučni senzor
d) Infracrveni senzor
e) Magnetski senzor
f) Video kamera

Bluetooth, GPS prijemnik i RFID su jedna od vrsta uređaja koje se mogu nalaziti u vozilu te služe također za prikupljanje podataka. Pomoću mobilnih uređaja također se mogu prikupljati podaci o brzini vozila, vremenu putovanja ili lokacije preko metode FCD. Podaci koji se prikupljaju su od izuzetne važnosti za daljnji razvoj ITS-a pomoću kojeg se razvija i poboljšava klasični promet na neku višu apstraktnu razinu.

Za potrebe ovoga rada i izrade aplikacije korišteni su GPS podaci o vozilima koje je ustupila tvrtka Mireo d.d. Fakultetu prometnih znanosti u svrhu izrade zajedničkog projekta SORDITO. Projekt SORDITO trajao je u razdoblju od kraja 2014. do početka 2016. godine (16 mjeseci) s ciljem smanjenja troškova dostave.

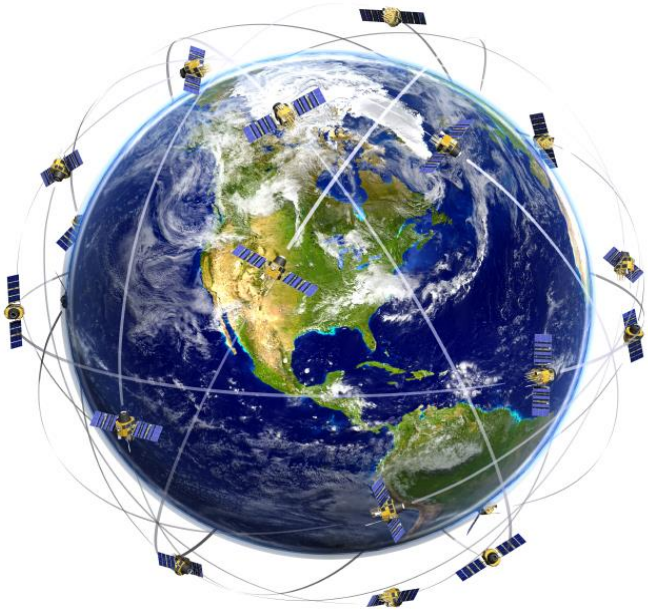
GPS podaci sadržavaju informacije o 4200 vozilima koja su praćena od 2009. do 2014. godine što iznosi oko 7 milijardi zapisa. Aplikacija je izrađena u Microsoft Visual Studio 2015 koja računa troškove voznog parka, udaljenosti rute koje je vozilo prošlo od točke A do točke B te stvara datoteku kml preko koje korisnik na Google Earth-u može vidjeti prikaz rute na području grada Zagreba i Zagrebačke županije. Aplikacija također omogućava prikaz podataka u My SQL serveru. Troškovi voznog parka su podijeljeni na:

a) Dnevni troškovi
b) Mjesečni troškovi
c) Godišnji troškovi

Izračunati su također troškovi radnog vremena gdje su prikazani troškovi tvrtke i vozača ovisno o vremenu obavljene rute. Radno vrijeme izračun troškova za vozača je od 08:00-16:00, dok se ostatak dodjeljuje troškovima tvrtke.

2.1 GPS SUSTAV

GPS (slika 1) je globalni navigacijski satelitski sustav koji služi za određivanje pozicije korisnika te se sastoji od prostornog, nadzorno-upravljačkog te korisničkog segmenta.



Slika 1: GPS sustav, [1]

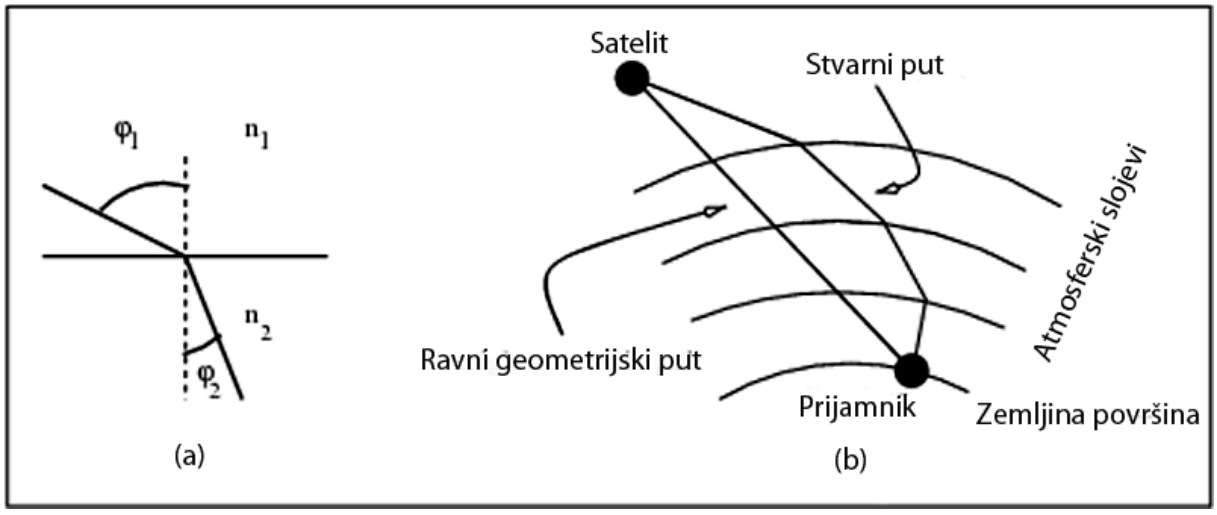
Sustav se temelji na računanju vremena koje je potrebno za propagaciju signala od satelita do prijemnika. Raspoloživost sustava je stalna i globalna, a nakon uvođenja civilne frekvencije točnost mu je uvećana na 8,5 metara. Prostorni segment sastoji se od 32 satelita koji se gibaju oko Zemlje u kružnim orbitama na 20 000 km. Raspored satelita je raspoređen unutar 6 orbitalnih ravnina tako da se u svakome trenutku iznad horizonta nalazi najmanje 5 ili više satelita čime se postiže globalna pokrivenost. Kontrolni segment služi za nadzor i upravljanje sustavom. Stanice su raspoređene po čitavoj Zemlji te prate sve GPS satelite te prosljeđuju primljene signale u glavnu kontrolnu postaju na obradu. Korisnički segment se dijeli na autorizirani (američka vojska i držane službe) i neautorizirani (korisnici širom svijeta), [2].

U današnje vrijeme velika je upotreba GPS sustava koji se koristi za civilne usluge. Iako se koristi za navigaciju zrakoplova u zračnom prometu ili navigaciju brodova u vodnom prometu najrasprostiranija upotreba je ipak u cestovnom prometu to jest na kopnu. Zbog velikog interesa kod civilnih korisnika došlo je do razvoja raznih aplikacija za lociranje i prikaz najbližih ruta i mjesta na digitalnoj mapi ili kameri na mobilnom uređaju. Isto tako omogućene su brojne aplikacije kod korištenja osobnih vozila. Veliki tehnološki pomak je i kod autonomnih vozila koje koriste brojne senzore za interakciju s prometom koje ne bi bilo moguće bez navigacijskog sustava.

2.2 ODREĐIVANJE POZICIJE

Određivanje pozicije se svodi na vremenu potrebnog da signal krene iz satelita te dođe do prijemnika koji se nalazi na nekom zemljinom području. Signali se emitiraju dvjema frekvencijama f1 (L1) i f2 (L2) te sadržavaju navigacijske poruke u kojima se nalaze korekcijski podaci. Korekcijski podaci služe kako bi GPS prijemnik mogao izračunati vlastitu poziciju na temelju pozicije satelita koja je uvijek poznata [3].

U korekcijskim podacima također postoji i zapis odstupanja atomskih oscilatora zbog Zemljinog atmosferskog utjecaja na širenje signala koja djeluje na takav način da ovisno o sloju atmosfere dolazi do promjene indeksa loma kako je prikazano slikom 2. Atomski oscilatori moraju biti usklađeni i stabilni kako bi sustav bio u mogućnosti normalno funkcionirati te u tom pogledu korekcijske poruke ispravljaju odstupanja satova između satelita i prijemnika.



Slika 2: Utjecaj zemljine atmosfere na signal, [4]

Kako atomski oscilatori nisu usklađeni koriste se četiri mjerenja:

a) Geografska širina
b) Geografska duljina
c) Nadmorska visina
d) Odstupanje sata prijemnika

Za izračunavanje pozicije korisnika i razmaka je moguće ako je iznad horizonta najmanje 4 satelita koliko nam je potrebno za trodimenzionalno računanje. Računanje se svodi prema formuli:

$$\rho_i = \left| s_i - u \right| + c * t_u \qquad (1)$$

gdje je :

- ρ_i - pseudorazmak
- c - brzina svjetlosti
- t_u - offset (razmak satelitskog i sata u prijemniku)
- s_i - pozicija satelita
- u - pozicija prijemnika

Za izračun vlastite pozicije GPS prijemnik dobiva navigacijske podatke od satelita koje sadržavaju poziciju i korekciju pogrešaka zbog utjecaja zemljine atmosfere na medij kojim se širi signal, te mora izračunati udaljenost do satelita koja se računa prema formuli.

$$R = c * \Delta t = c * (ts - tp) \qquad (2)$$

Gdje je:

- R - udaljenost prijemnika do satelita
- c - brzina svjetlosti
- ts - pokazivanje satelitskog sata
- tp - pokazivanje sata u prijemniku

Pomoću map matching algoritma dobivamo optimalne informacije i prikaz puta jednog vozila što je jako bitno u računanju troškova vozila i njegovu udaljenost koju je prešao od polazišta do odredišta. Bez ispravljanja pogreške ne bi bilo moguće dobiti točne podatke niti prikazati rutu koju je vozilo obavilo.

2.3 PODACI PRAĆENIH VOZILA

Na početku 2 poglavlja objašnjeno je kako su se i u kojem periodu podaci prikupljali, dok će se u ovome pod poglavlju objasniti što podaci sadržavaju te kako su pohranjeni u direktoriju. Svi podaci za jedan uređaj koji su dobiveni su pohranjeni u jednom direktoriju, na primjer:

\0-99\23\354330030921731\

gdje se prve dvije razine direktorija mogu ignorirati (0-99 i 23). Zadnji broj direktorija predstavlja ime (354330030921731) te je jedinstveni indikator za praćenje uređaja. Najčešće to je 15-znamenkasti IMEI i može biti bilo kakav nit znakova.

Za svaki mjesec pozicije su u odvojenoj .hs1 datoteci tako da datoteka ima sljedeći zapis: 2013_4.hs1 u kojoj prvi broj označuje godine, a sljedeći mjesece u kojima se vršilo praćenje vozila. U formatu .hs1 datoteke prva 4 bajta su header, a nakon toga slijede pozicije kod kojih je svaka velika točno 64 bajta.

Kraj niza pozicija označen je posebnom dummy pozicijom u kojoj su prva četiri bajta nule, a sve vrijednosti formata pozicije su little-endian. Kako se kroz godine format pozicije mijenjao, tako se provjera tip pozicije (predzadnji bajt) i u ovisnosti o njemu interpretira sve ostalo. Ako je tip pozicije < 20, poziciju treba ignorirati jer je format prestar, dok je ostala pozicija prikazana u tablici 1.

Tablica 1: Format pozicije za tip pozicije < 20

OFFSET	DULJINA	ZNAČENJE
0	4	Vrijeme pozicije
4	4	X koordinata
8	4	Y koordinata
49	1	Oznaka
50	1	Map matching: brzina
51	1	Map matching: kurs
52	2	Map matching: udaljenost
54	4	Map matching: X koordinata
58	4	Map matching: Y koordinata
62	1	Tip pozicije

Dodatna pojašnjenja tablice 1:

- Vrijeme pozicije je broj sekundi nakon 1. 1. 1970. 0:00 UTC
- X koordinata – zapis u int
- Y koordinata – zapis u int
- Oznaka - najniža tri bita ovog bajta su oznaka koju je dodijelio heuristički filter pozicija. Ako su jednaka 4, pozicija je vožnja. Ako su 5, pozicija je stajanje. Inače poziciju treba odbaciti (pozicija nema smisla ili je neprecizna)
- Map matching: brzina – ispravljena brzina u km/h
- Map matching: kurs – ispravljeni kurs podijeljen s 2 ili 255 ako nije poznat
- Map matching: udaljenost – u metrima, po cesti od prethodne matchirane točke koja je vožnja ili stajanje. 65535 ako je nepoznata.
- Map matching: X koordinata – zapis u int

- Map matching: Y koordinata – zapis u int
- Kako su x i y koordinate zapisane u cijelim brojevima potrebna je njihova konverzija prema sljedećoj formuli:

$$longitude = X / ((double)0x08000000) * 180.0 \quad (3)$$

$$latitude = (2*atan(exp((Y / ((double)0x08000000) * PI))) - PI/2) * 180.0 / PI \quad (4)$$

Kodiranje brzine i kursa se radi na dva različita načina, ovisno tome da li je tip < 50 ili je tip >= 50, a podaci su prikazani tablicom 2.

Tablica 2: Format brzine i kursa

ZA TIP < 50		
OFFSET	DULJINA	ZNAČENJE
16	2	Brzina
18	2	Kurs
ZA TIP >= 50		
OFFSET	DULJINA	ZNAČENJE
16	1	Brzina
18	1	Kurs

Dodatna pojašnjenja tablice 2:

1) za tip pozicije koji je manji od 50:
- Brzina – izražena u km/h ili je negativan broj ako nije poznata
- Kurs – izražen u stupnjevima ili je negativan broj ako nije poznata
2) Za tip pozicije veći ili jednak 50:
- Brzina – izražena u km/h ili 255 ako nije poznata. Ako je veća od 254 km/h, tada je zapis 254
- Kurs – zapis je podijeljen s 2 ili 255 ako kurs nije poznat. Moguće vrijednosti su od 0 do 179 i 255.

Pozicije nisu sortirane kronološki, nego redom kojim su primane od uređaja za praćenje te je stoga moguća pojava duplikata. Kurs pri stajanju vozila najčešće nije poznat, ali mnoge vrste uređaja greškom šalju kurs 0 u tim situacijama. Windows Form aplikacija za računanje troškova voznog parka pisana je u Microsoft Visual Studio-ju C# jezikom.

3. POHRANA PODATAKA U RELACIJSKU BAZU PODATAKA

Podaci koji su očitani s datoteke hs1 se prebacuju u relacijsku bazu podataka kako bi korisnik mogao dobiti uvid u podatke s kojima raspolaže. Prebacivanje podataka u SQL je opisano u poglavlju 3.2.

Bazu podataka promatramo kao zbirku podatkovnih zapisa pohranjenih na računalu koja je lako dostupna korisnicima i aplikacijama. Sastoji se od skupa međusobno povezanih podataka, pohranjenih zajedno, bez štetne ili nepotrebne zalihosti. Podaci su u bazi pohranjeni u obliku neovisnom o aplikacijama koje ih koriste, a rukovanje podacima izvodi se isključivo kroz zajedničko i nadzirano sučelje. Sustav za upravljanje bazom podataka DBMS je programska podrška koja izvodi sve operacije nad bazom podataka. Primjeri operacija nad bazom podataka su kreiranje strukture, brisanje, mijenjanje i dohvaćanje podataka, administracija i dr. Sustav za upravljanje bazom podataka brine se o fizičkom smještaju podataka, administraciji sustava i obnovi podataka nakon rušenja baze podataka, [5].

Kako bi se uspješno prebacili podaci u bazu podataka potrebno je prvo odrediti korisničke zahtjeve. Korisnički zahtjevi su jako bitni kako bi znali odrediti kako će tablice i veza između njih biti, a da bismo uspješno napravili bazu treba se prvo napraviti ER model. ER model radi se na principu korisničkih zahtjeva te služi kako bi korisnik bolje razumio model podataka.

Model podataka je sustav koji se sastoji od skupa objekata, operacija i pravila, a glavni zadaća je definiranje logičke strukture baze podataka. U logičkoj strukturi opisuje se procesi i tokovi informacija koji su spremljeni u tablice.

Postoji više vrsta modela podataka kao što su:

a) Mrežni
b) Hijerarhijski
c) Dimenzijski

ER modeliranje rađeno je s alatom yEd -Graph Editor u kojem postoje elementi ER modela. Alat služi za izradu raznih dijagrama kao što su mentalne mape, ER modeli, dijagrami toka itd. te je jako poželjan kod izrade baza podataka jer kod izrade velikih količina podataka za bazu olakšava njenu izradu.

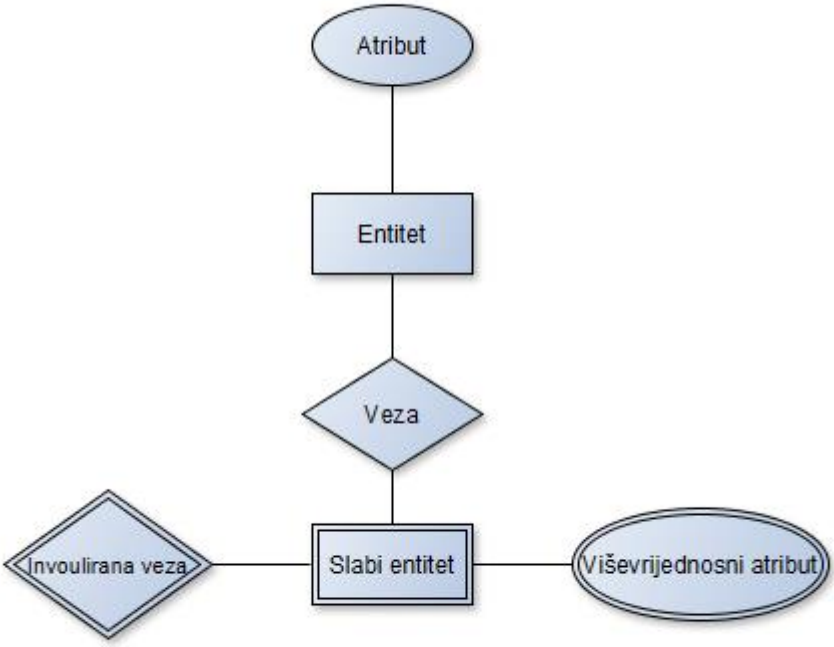
Baze podataka su višekorisnički sustavi u kojima je nužno osigurati dodjeljivanje i poštivanje ovlasti pristupu, mijenjaju i brisanju podataka. Administriranje korisničkih ovlasti i njihovo provođenje jedna je od bitnih zadaća SUBP-a. Pojedini SUBP-ovi razlikuju se u načinu provođenja i upravljanja ovlastima korištenja podataka, a najčešće se implementacija svodi na korištenje unaprijed određenih uloga i prava pojedinih korisnika, [5].

3.1 ER MODEL

ER model je mnogo precizniji od korisničkih zahtjeva, a namjena mu je da objekte iz stvarnog svijeta prikaže u jednom od modela. Elementi ER modela su:

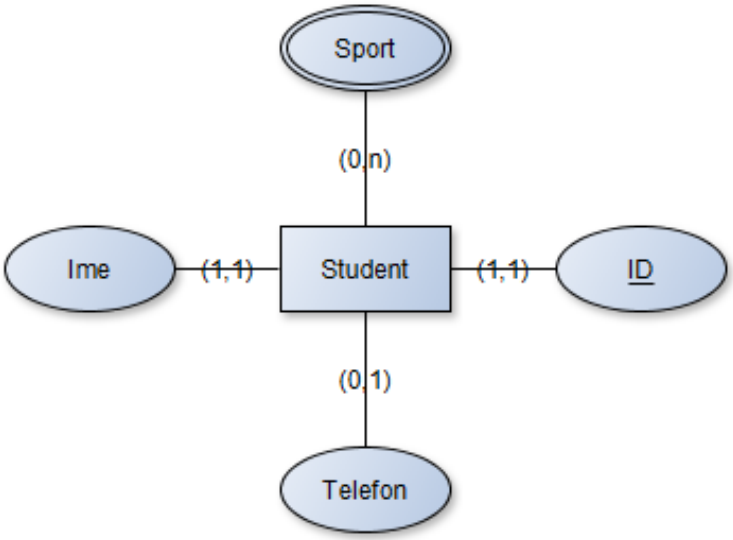
a) Entitet
b) Atribut
c) Veza
d) Slabi entitet
e) Više vrijednosni atribut
f) Involuirana veza

Entitet je neki objekt iz stvarnog svijeta kojeg želimo opisati. Atribut pripada entitetu to jest opisuje ga. Veze služe za povezivanje entiteta te pomoću njih možemo prikazati ER dijagram. Entiteti, veze i atributi prikazani su slikom 3.



Slika 3: Elementi ER modela

ER model sastoji se od dijagrama entiteta i ER dijagrama. Dijagram entiteta prikazuje odnose između entiteta i njegovih atributa koji ga opisuju. Uz navedene elemente dijagrama također je važno odrediti kardinalitet atributa te identifikacijski atribut bez kojih nije moguće složiti dobru bazu podataka. Kardinalitet atributa nam govori koliko opisuje entitet kojem je pridijeljen. Iznos se zapisuje pomoću brojeva 0 i 1 te slova n ovisno koliko daje za opis entiteta. Slikom 4 su prikazani na koji način se zapisuje kardinalitet.



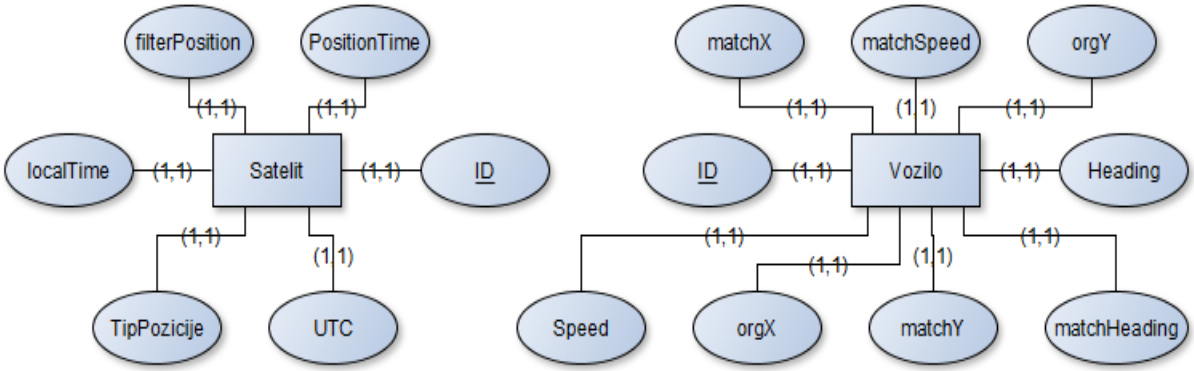
Slika 4: Primjer dijagrama entiteta

Izvor: [6]

Na slici 5 je prikazan jedan primjer dijagrama entiteta koji se sastoji od jednog entiteta (Student) i četiri atributa (ID, Ime, Telefon i Sport) s time da atribut Sport je viševrijednosni atribut. Kako su ID i Ime obavezni za unos kardinalitet će iznositi (1,1), dok Telefon i Sport nisu obavezni za unos pa stoga iznose (0,1) odnosno (0,n). Prvi broj nam kazuje da li je unos obavezan ili nije, dok drugi broj označava da li unosimo jednu ili više informacija koje unosimo.

U ovome slučaju kod atributa Ime i ID vidimo da je unos obavezan i da se unosi samo jedna informacija (kod imena postoji slučaj s dva imena ali u ovome slučaju smo uzeli unos samo studenata s jednim imenom). Atribut Telefon nije obavezan za unos te unosimo jedan iznos. Viševrijednosni atribut Sport nije obavezan za unos međutim kako se student može baviti više sportova mora na drugom mjestu ići „n“. Preko korisničkih zahtjeva se određuje kardinalitet atributa.

ER dijagram pokazuje odnose između entiteta bez njegovih atributa te služi kako bismo odredili veze između entiteta. Pomoću veza može odrediti u koju tablicu se stavlja strani ključ kako bismo povezali tablice. Na slici 5 je prikazan dijagram entiteta dobiven podacima za praćenje vozila.



Slika 5: Dijagram entiteta vozila i prijemnika

Kako u datoteci hs1 ima jako puno podataka uzeti su samo oni s kojima su se računali troškovi voznog parka.

Veze između entiteta mogu se podijeliti na, [7]:

- Jednostavne veze (binarne veze):

a) 1 : 1 - Svaki element skupa R može biti povezan samo s jednim elementom skupa S. Isto vrijedi za elemente skupa S.
b) 1 : N - Svaki element skupa R može biti povezan s više elemenata skupa S, ali ne nužno. Dok svaki element skupa S može biti povezan sa samo jednim elementom skupa R.
c) N : 1- Svaki element skupa R može biti povezan s jednim elementom skupa S. Dok svaki element skupa S može biti povezan s više elemenata skupa R.
d) M : N- Svaki element skupa R može biti povezan s više elemenata skupa S, ali i ne mora. Isto vrijedi za elemente skupa S.

- Složene veze:

a) Involuirane veze - Involuirana veza povezuje neki entitet sa samim sobom. Dakle riječ je o binarnoj vezi između entiteta istog tipa.

b) Podtip veze - Podtip veze su veze u kojima je entitet E1 podtip entiteta E2. E1 ima sve atribute od E2 i svoje dodatne, a ostvaruje se vezom 1:1. Ova veza je reprezentacija nasljeđivanja u objektno orijentiranoj paradigmi programiranja.
c) Ternarne veze -Ternarna veza je ona veza koja u sebi sadrži tri različita tipa entiteta. Ternarna veza se uvodi onda kada vezu nije moguće rastaviti na binarne veze. Ternarna veza može se ostvariti na sljedeće načine N:M:P, 1:N:M, 1:1:M ili 1:1:1.

Veza između satelita i vozila je tipa N:1 i prikazana je slikom 6.

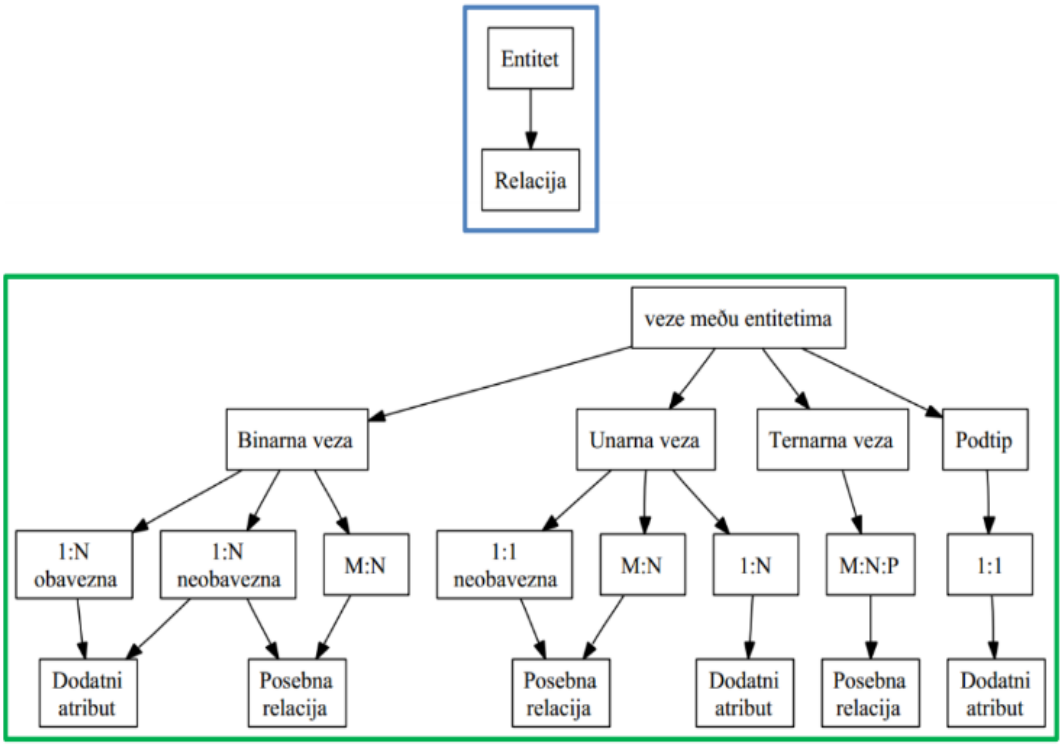
Slika 6: ER dijagram vozila i satelita

3.2 RELACIJSKI MODEL

Relacijska baza podataka služi za pohranu podataka u tablice koje se potom mogu povezivati s ostalim tablicama ovisno o njihovoj vezi. Tablice predstavljaju entitet dok su podaci kojim se pune vrijednosti atributa. Prije samog kreiranja relacijske baze potrebno je odrediti entitete, atribute i njihove veze (Poglavlje 3.1) kako bi se ispunili korisnički zahtjevi i kako bi baza funkcionirala kako treba.

Relacijski model je bio teorijski zasnovan još krajem 60-tih u radovima E.F.Codd-a. Dugo vremena se je pojavljivao samo u akademskim raspravama i knjigama. Prve realizacije na računalu bile su suviše spore i neefikasne. Ali zahvaljujući istraživanjima i razvoju računala efikasnost relacijskih baza se postepeno poboljšavala. Tako sredinom 80-tih relacijski model postaje prevladavajući. Većina suvremenih DBMS koristi relacijski model, [8]

Pravila koja se poštuju kod pretvorbe iz ER dijagrama u relacijsku shemu prikazana su slikom 7. Slika 7 prikazuje moguće veze između entiteta u kojima se ovisno o vezi sprema strani ključ. Postoje primarni i strani ključ. Primarni ključ se uvijek stavlja pod ID, dok strani ključ se može staviti pod glavni entitet, a ako se radi primjerice o M:N vezi tada se radi nova tablica koja povezuje entiteta (ostale tablice).



Slika 7: Transformacijska pravila, [9]

Primarni ključ je atribut koji identificira svaki redak u tablici, dok stani ključ predstavlja primarni ključ glavne tablice, a spremljen je u drugoj kako bi se mogla ostvariti veza.

Postoji 8 transformacijskih pravila koje se moraju poštovati, [10]:

1) Dijagram entiteta ER modela preslikava se u relacijski shemu na sljedeći način:
   a) Svaki entitet će postati jedna tablica - ime tablice jednako je nazivu entiteta
   b) Svaki atribut će postati jedan stupac tablice
   c) Za svaku tablicu potrebno je odabrati primarni ključ i pri tome treba paziti na pravila jedinstvenosti, minimalnosti i integriteta

2) Prikaz veze 1:1 ostvaruje se na tri načina ovisno o članstvu u vezi:
a) ako su članstva za E1 i E2 obavezna
b) način ako je članstvo za samo jedan od entiteta obavezno
c) način ako su oba članstva za E1 i E2 neobavezna

3) Pretvaranje binarnih veza 1:N
   a) Ako su E1 i E2 u vezi 1:N =>E1 ne utječe na vezu dok se E2 proširuje s dodatni atributom (stranim ključem) koji je primarni ključ u E1
   b) Ako je članstvo E1 neobavezno strani ključ u E2 se postavi da može postati NULL tip

4) Pretvaranje binarnih veza M:N
   a) Za transformaciju veze M:N uvijek se uvodi nova tablica koja se sastoji od primarnih ključeva entiteta E1 i E2 koji zajedno čine složeni primarni ključ nove tablice
   b) Ako veza ima neke dodatne atribute i oni se uključuju u novu tablicu

5) Transformacijsko pravilo: Ako entitet E1 ima više vrijednosni atribut transformacija se radi tako da se taj atribut prikazuje u posebnoj tablici i promatra se funkcionalnost veze između nove tablice i stare te se već poznatim pravila izradi relacijski model

6) Pretvaranje involuirane veze se radi tako da se prepozna veza (1:1, 1:N i M:N) entiteta sa samim sobom zatim se radi pretvaranje kao kod jednostavnih veza

7) Pretvaranje podskup veze se radi tako da svi podskupovi u vezi kreiraju u posebne tablice i sadrže samo svoje specifične atribute te strani ključ na nadskup tablicu

8) Ternarna veza se u relacijski model transformira tako što:
   a) Svaki od entiteta se prikazuje posebnom tablicom
   b) Za povezivanje se uvodi nova tablica koja sadrži primarne ključeve od sva tri entiteta
   c) Primarni ključ nove tablice mogu biti ta tri strana ključa ili se može uvesti generički primarni ključ ID sto se najčešće i radi pogotovo ako postoje još dodatni atributi koji opisuju vezu

Osim transformacijskih veza potrebno je odrediti podatkovne tipove za atribute. Vrste i opis nekih podatkovnih tipova prikazana je tablicom 3.

Tablica 3: Podatkovni tipovi atributa, [11]

Kategorija	*Tip*	*Opis*
Egzaktni/brojčani	Bit	Cijeli broj (1,0 ili NULL)
	Int	Egzaktni cjelobrojni iznos
	Money	Podatkovni tip
	Decimal	Brojčani tip u kojem navodimo koliko decimala želimo
Aproksimirani/brojčani	Float	koristi plivajući zarez
Unicode/tekstualni	Nchar	UNICODE reprezentacija teksta fiksne veličine
	Nvarchar	UNICODE reprezentacija teksta varijabilne veličine
Tekstualni	Char	reprezentacija teksta fiksne veličine
	Varchar	reprezentacija teksta varijabilne veličine
Datumski i vremenski	Date	Datum u formatu „YYYY-MM-DD“
	Datetime	Datum i vrijeme u formatu „YYYY-MM-DD hh:mm:ss“
	Time	Vrijeme u formatu hh:mm:ss

Izvor: [11]

3.3 PREBACIVANJE PODATAKA U SQL TABLICU

Kreiranje baze podataka u SQL-u može se postići na dva načina. Prvi način je stvaranje baze preko GUI-a tako da napravimo novu tablicu te u nju ubacujemo atribute, njezine podatkovne tipove i označimo da li je unos obavezan ili ne.

Obavezni unos označimo s kvačicom ili je ostavimo praznom ako unos nije obavezan. Drugi način kreiranje baze podataka je preko SQL skripte u kojem kreiramo bazu, a zatim tablice. Unutar svake tablice se unosi zapis ime atributa, primarni ključ, podatkovni tip, NULL/NOT NULL vrijednosti koja označava da li je unos obavezan ili ne, te strani ključ ovisno o vezi.

U ovome radu baza podataka je kreirana preko SQL skripte preko koje su kreirane dvije tablice s pripadajućim vrijednostima kako je prikazano na slici 8.

Mireo_Podaci.sql -...reo_Podaci (sa (52))

```
create database Mireo_Podaci

create table Vozilo(
ID int not null primary key identity (1,1),
matchX float not null,
matchY float not null,
matchSpeed int not null,
matchHeading int not null,
orgX float not null,
orgY float not null,
Speed int not null,
Heading int not null,
)

create table Satelit(
ID int not null primary key identity (1,1),
UTC int not null,
filterPosition int not null,
localTime nvarchar(50) not null,
TipPozicije int not null,
PositionTime int not null,
VoziloID int not null foreign key references Vozilo(ID)
)
```

Slika 8: SQL skripta

Nakon što se napravi skripta za kreiranje tablica tada se preko Microsoft Visual Studio c# prebacuju učitani podaci u tablice. Kako bismo uspješno prebacili podatke potrebno je prvo povezati SQL i Visual Studio. Povezivanje se postiže sljedećim kodom koji je prikazan na slici 9:

```
private void btnConnect_Click(object sender, EventArgs e)
        {
            string Connectthestring = null;


            1) //Connectthestring = "Server= DESKTOP-6VUB19G\\SQLEXPRESS; Database=
               Mireo_Podaci;User ID=sa;Password=p@ssword13";

            2) //Connectthestring = "Server= DESKTOP-6VUB19G\\SQLEXPRESS; Database=
               Mireo_Podaci; Integrated Security=True";



            Connectthestring = "Server= DESKTOP-6VUB19G\\SQLEXPRESS; Database=
Mireo_Podaci;User ID=sa;Password=Baze123";
            connection = new SqlConnection(Connectthestring);
            try
            {
                connection.Open();
                MessageBox.Show("Connected!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Slika 9: Povezivanje SQL-a i Microsoft Visual Studia

Kod SQL postoji dvije verzije povezivanja: SQL i Windows autentikacija. Razlika je u tome što Windows autentikacija pruža veću sigurnost zbog različitih sigurnih protokola koji štite korisnički *account* bolje od SQL-ove autentikacije. Ako koristimo SQL autentikaciju tada moramo koristiti kod pod rednim brojem 1, a ako se koristi Windows onda pod rednim brojem 2.

Nakon povezivanja SQL-a i Visual Studija vrši se prebacivanje podataka u relacijsku bazu. Bitno je odrediti u koju tablicu spremamo podatke kako ne bi došlo do pogreške zbog toga što svaki podatak pripada određenom atributu koji smo realizirali kod SQL kripte te se ne može mijenjati u C#. Postupak prebacivanja podataka iz datoteke hs1 u relacijsku bazu prikazan je jednim dijelom sljedećeg koda na slici 10:

```
List<HS1Data> listData = readHSdataFile(fi.FullName);
                    SqlCommand command;
                    string sql = null;
                    string sql2 = null;
                    foreach (HS1Data d in listData)
                    {

                        SqlDataReader dataReader;
                        try
                        {
                            sql = ("INSERT INTO dbo.Vozilo
(matchX,matchY,matchSpeed,matchHeading,orgX,orgY,Speed,Heading)VALUES (" +
Convert.ToString(d.matchX).Replace(",", ".") + "," +
Convert.ToString(d.matchY).Replace(",", ".") + "," + d.matchSpeed + "," +
d.macthHeading + "," + Convert.ToString(d.orgX).Replace(",", ".") + "," +
Convert.ToString(d.orgY).Replace(",", ".") + "," + d.Speed + "," + d.Heading + ");");

                            command = new SqlCommand(sql, connection);
                            dataReader = command.ExecuteReader();
                            dataReader.Close();

                            sql2 = "INSERT INTO
dbo.Satelit(UTC,filterPosition,localTime,TipPozicije,PositionTime)VALUES(" + d.UTC +
"," + d.filterPosition + ",'" + d.localTime.Year + "-" + d.localTime.Month + "-" +
d.localTime.Day + "'," + d.TipPozicije + "," + d.PositionTime + ")";

                        command = new SqlCommand(sql2, connection);
                        dataReader = command.ExecuteReader();
                        dataReader.Close();

                         KML.Add(d);
                        }
```

Slika 10: Ubacivanje podataka iz hs1 datoteke u SQL

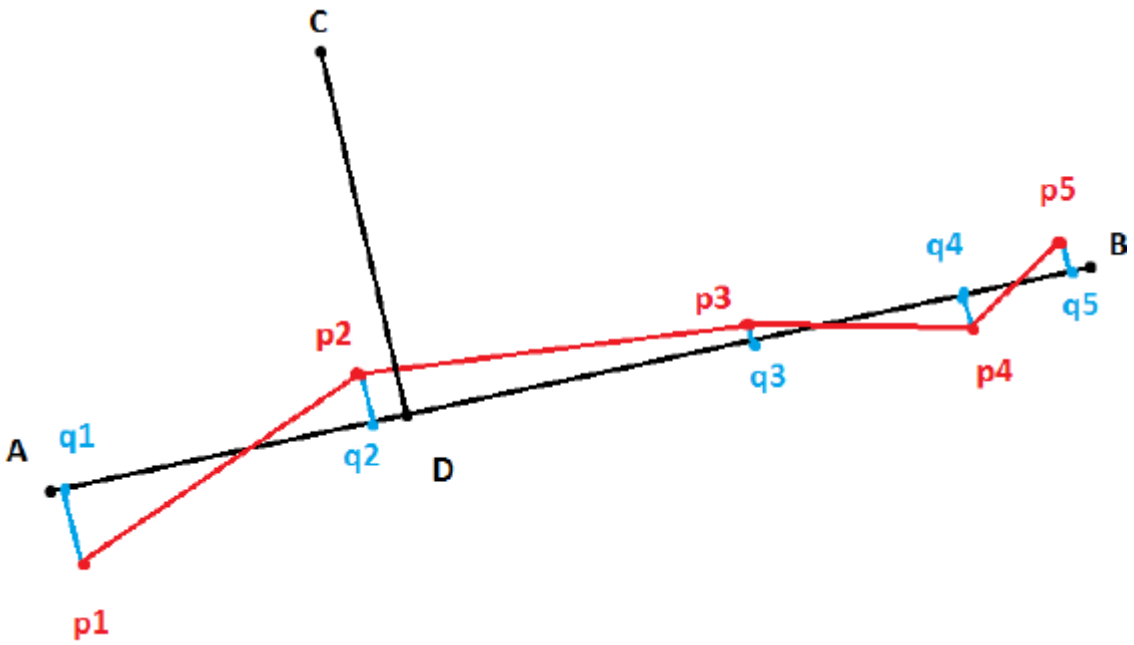
Na kraju koda imamo „kml.add(d)“ koji služi za spremanje svih pročitanih podataka u globalnu listu koja se nalazi u public Form1. Lista kml služi za pohranu i čitanje podataka kako bismo mogli stvoriti kml datoteku i prikazati na Google Earth-u rutu vozila. Rute vozila su prikazane u poglavlju 6.

4. ANALIZA KRETANJA VOZILA TEMELJEN GPS TRAGOVA

Uloga GPS-a postaje sve više zastupljenija u prikupljanju prometnih podataka kretanja vozila. Pomoću podataka moguće je odrediti troškove vozila, udaljenosti između polazišta i konačnog cilja te određivanje najkraćih udaljenosti pomoću koji se ti troškovi smanjuju.

Osim smanjenja troškova također se može postići smanjuje zagađenje okoliša i onečišćavanje okoline, smanjenje potrošnje goriva, smanjenje vremena putovanja i prometnih zagušenja. Putovanja od točke A do točke B određena su bridovima, vrhovima i GPS segmentima.

GPS segmenti su isječci na jednom bridu koji daje podatke kao što su lokacija vozila, udaljenost od prijašnjeg segmenta, brzinu itd. Problem koji nastaje je prikaz krive lokacije na digitalnoj karti koji se rješava map matching algoritmom. Slika 11 prikazuje rutu na kojoj su iscrtani segmenti i koordinate.



Slika 11: Prikaz rute, GPS segmenata i koordinata, [12]

Na slici su prikazani segmenti AB i CD, dok su p1,p2,p3 i p4 koordinate koje imaju grešku zbog nepreciznosti GPS sustava. Preko map matching algoritma dobije se precizniji prikaz te su označeni q1,q2,q3,q4 i q5.

Algoritmi mogu raditi na dva načina:

a) Rad u realnom vremenu – vrši istovremeno ispravljanje grešaka pozicioniranja nad dobivenim podacima
b) Rad nad prikupljenim podacima – vrši obradu podataka nakon što su podaci prikupljeni

Kod predviđanja pozicije imamo različite pristupe, [13]:

a) Poludeterministički pristup
b) Probabilistički pristup
c) Neizrazita logika

Da bismo odredili udaljenosti između segmenata potrebna nam je pozicija početne i konačne točke koja ima svoje koordinate. Udaljenost između dvije točke određuje se prema sljedećoj formuli:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (5)$$

gdje je :

d(A,B) – udaljenost od točke A do točke B

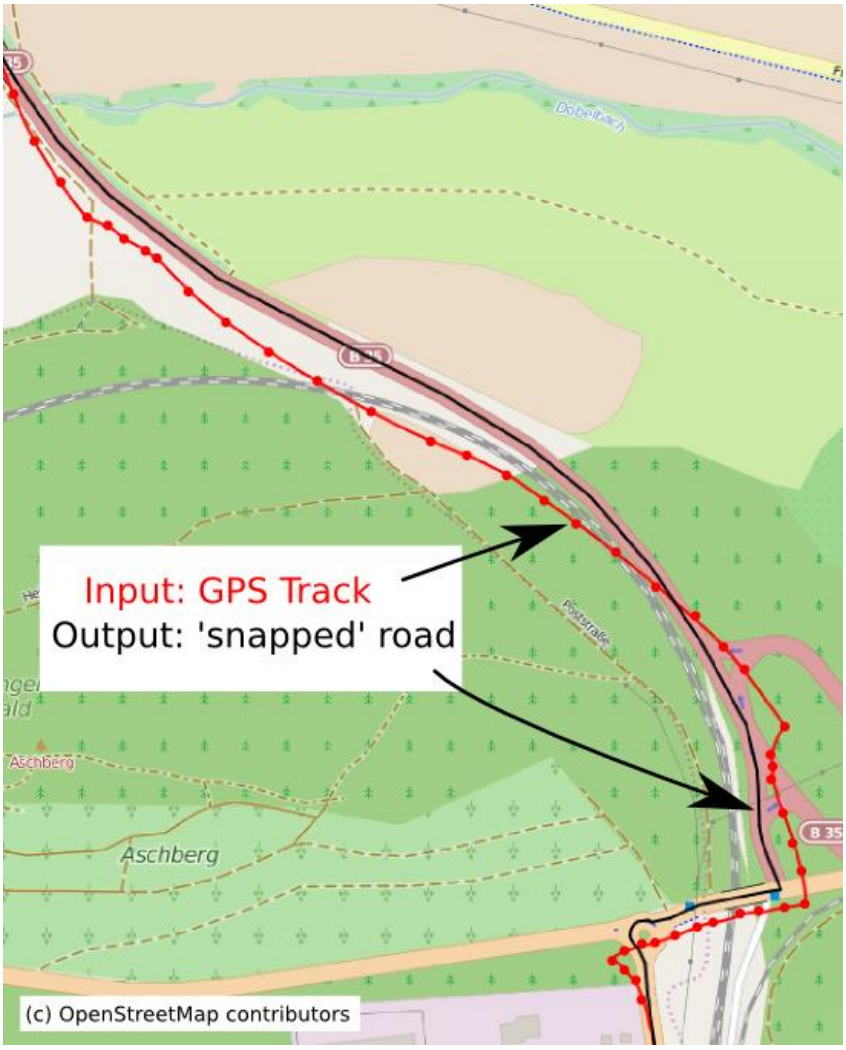
(x1,y1) – koordinate točke A

(x2,y2) – koordinate točke B

Map matching je algoritam koji služi kako bi smjestio podatak o lokaciji vozila na pripadajuće mjesto (u ovome slučaju cestu) te tako prikazao točni prikaz rute na digitalnoj karti. Razlog ispravljanja pogreške kod lociranja je u tome što GPS sustav ima pogrešku lociranja od 10 metara, pa zbog toga pozicija vozila se može nalaziti na pogrešnom mjestu.

Pozicija vozila određena je x-y koordinatama što na digitalnoj karti predstavlja zapravo točku, no kada sve te točke povežemo dobijemo jedan GPS segment koji se sastoji od pozicije A i B. U ovome radu koriste se podaci koji su dobiveni iz uređaja za praćenje vozila. Podaci koje koristimo su x-y koordinata i njihovih matchirani podaci MatchX i MatchY također matchDistance, matchSpeed i matchCourse.

Kako su se ispravljale koordinate tako je došlo do promijene vrijednosti udaljenosti, brzine i kursa. Slika 12 opisuje kako izgleda pogreška pozicioniranja vozila zbog GPS sustava i točno pozicioniranje vozila. Crnom bojom su označeni podaci dobiveni iz GPS uređaja, dok crvena boja pokazuje ispravljen zapis pomoću algoritma.



Slika 12: map matching algoritam, [14]

Pomoću map matching algoritma dobivamo optimalne informacije i prikaz puta jednog vozila što je jako bitno u računanju troškova vozila i njegovu udaljenost koju je prešao od polazišta do odredišta. Bez ispravljanja pogreške ne bi bilo moguće dobiti točne podatke niti prikazati rutu koju je vozilo obavilo.

5. IZRAČUN TROŠKOVA VOZNOG PARKA TIJEKOM SLUŽBENOG RADNOG VREMENA

U današnje vrijeme kada tržište postaje sve veće i kada su smanjena ograničenja roba, ljudi, usluga i informacija troškovi postaju sve veći problem. Postoje više načina reduciranja troškova kao što su smanjenje voznog parka ili njegova zamjena novi vozilima većeg kapaciteta, smanjenje vremena putovanja odabirom kraćih ruta, smanjenje potrošnje goriva ili prijeđenog puta. Za potrebe izračuna troškova voznog parka izračunate su udaljenosti rute vozila koja su praćena 5 godina od tvrtke Mireo d.d.

Osnovni troškovi koji su izračunati dijele se na:

a) Dnevni troškovi
b) Mjesečni troškovi
c) Godišnji troškovi

Troškovi tijekom službenog radnog vremena su:

a) Troškovi tvrtke
b) Troškovi vozača

U jednome danu je moguće da je snimljeno više od jednog vozila ili da je jedno vozilo praćeno kako obavlja dvije ili više ruta u jednom danu. Za potrebe ovog rada uzeti su podaci za 2013 i 2014 godinu. Mjeseci koje ulaze pod 2013 godinu su od travnja do prosinca, dok za 2014 podaci kreću od siječnja do kolovoza. Troškove koji su u ovome radu računati možemo podijeliti na osnovne troškove i troškove tijekom radnog vremena.

Osnovni troškovi su računati prema sljedećoj formuli:

$$T = \frac{td}{1000} * 2 \qquad (6)$$

gdje je:

T [HRK] – trošak

td [km] – udaljenost

Match distance u kojem je pohranjen podatak prijeđenog puta je izražen u kilometrima zbog čega je potrebna pretvorba u metrima, pa se zbog toga dijeli s 1000. Razlog zbog čega se uzima metar, a ne kilometar je u tome što je ispitivanje vršeno tako da potrošnja iznosi 2 kn/km. Mjerna jedinica je HRK (kuna).

Kod troškova radnog vremena imamo ograničenja koja vozač mora poštovati jer u protivnom slučaju mu se pripisuju troškovi ali samo oni izvan vremenskog okvira. Na slici 13 je prikazan kod u C# na koji način se prema formuli [6] računaju troškovi tvrtke i vozača.

```
for (int i = 0; i < KML.Count; i++)
            {
                int indGodina = KML[i].localTime.Year - 2009;
                int indMjesec = KML[i].localTime.Month - 1;
                int indDana = KML[i].localTime.Day - 1;

                if (KML[i].localTime.Hour > 6 && KML[i].localTime.Hour < 17)
                {

troskoviUnutarRVPoGodinama[indGodina].dodajTrosak(KML[i].matchDistance, indMjesec,
indDana);
                }
                else
                {

troskoviUnutarIZVPoGodinama[indGodina].dodajTrosak(KML[i].matchDistance, indMjesec,
indDana);
                }
            }


            //Trosak Vozaca
            for (int i = 0; i < troskoviUnutarIZVPoGodinama.Count; i++)
            {
                TrosakGodina l = troskoviUnutarIZVPoGodinama[i];
                double td = l.ukupniTrosakGodina;
                string godina = (i + 2009) + "godine";
                dataGridView6.Rows.Add(godina, Convert.ToString((td / 1000) * 2));
            }
            //Trosak Tvrtke
            for (int i = 0; i < troskoviUnutarRVPoGodinama.Count; i++)
            {
                TrosakGodina l = troskoviUnutarRVPoGodinama[i];
                double td = l.ukupniTrosakGodina;
                string godina = (i + 2009) + "godine";
                dataGridView7.Rows.Add(godina, Convert.ToString((td / 1000) * 2));
            }
```

Slika 13: Kod za računanje troškova tijekom službenog vremena

Kako su svi podaci spremljeni u listu KML potrebno je preko for petlje proći po svim podacima. U posebnoj klasi *TrosakGodina* su definirane godine unutar kojih se nalazi „troskoviMjeseci“. „troskoviMjesec“ su definirani kao polje u kojima su spremljeni dani te za svaki dan njegova udaljenost koje je vozilo obavilo tijekom svog puta i označena je s *td*.

Kako su u pitanju polja tako je svaki prvi redak označen s 0 te se zbog toga mjeseci i dani moraju oduzimati s jedan. Razlog je u tome što ako prvi član u polju počinje s poljem [0], a zadnji završava na [12] dobijemo rezultat da jedna godina sadržava 13 mjeseci. Isto tako problem se javlja u danima, pa je potrebno istom mjerom oduzeti polje sa -1.

Godine se oduzimaju s 2009 kako bi program shvatio da se radi o pet godina posto stvara 2015 polja, a u ovome slučaju polje [0] predstavlja 2009. godinu, dok polje [5] predstavlja 2014. godinu. If naredba služi kako bi se postavio nekakav uvjet ili ograničenja, a unutar blok naredbi koje će se izvršiti. Kako slika 13 prikazuje postavljena su vremenska ograničenja u kojima ako je u rasponu od 7 – 16 sati zadovoljen uvjet tada se izvršava naredba.

Unutar naredbe se pod „troskoviUnutarRVPoGodinama“ spremaju podaci kao što su udaljenost, indeks mjesec i indeks dan. Ako ograničenje nije zadovoljeno onda se izvršava *else* naredba u kojoj se spremaju podaci u „troskoviUnutarRVPoGodinama“. Troškovi vozača se dobivaju tako da se preko for petlje prolazi po svim podacima koji su spremljeni unutar „troskoviUnutarRVPoGodinama“ te se spremaju pod „TrosakGodina l“.

Varijabla l sadržava ukupnu udaljenost tako da se na kraju izvršavanja programa mora podijeliti s 1000, te pomnožiti s dva kako je zadano prema formuli (6). Rješenje se prikazuje u tablicu s imenom „DataGridView 6“ u kojoj se nalazi dva stupca. U prvi stupac se ispisuju datumi kada je trošak nastao, a u drugi stupac se upisuju troškovi.

Ista je stvar kod računanja troškova tvrtke samo što se unutar njezine for petlje vrti varijabla „troskoviUnutarIZVPoGodinama“, a formula za računanje troškova ostake ista kao i u prethodnom rješavanju.

Osim izračunatih troškova u ovome radu su također izračunate udaljenosti koje je vozilo obavilo tijekom svoga puta. Put koje je vozilo obavilo najčešće započinje svojim polazištem (točka A) i odredištem (točka B), međutim kod prikaza ruta može se vidjeti da to i nije tako čest slučaj. U ovome slučaju polazište započinje početkom praćenja vozila i završava njenim prestankom. Udaljenost se računa na taj način da se udaljenost podijeli sa 1000 iz razloga što su vrijednosti matchDistance izražene u kilometrima.

Udaljenosti su izračunate za područje unutar i izvan grada Zagreba kako bi se moglo odrediti gdje su nastali veći troškovi i gdje su se vozila više kretala. Na slici 14 prikazan je kod u C# kako su ograničenja postavljena ovisno o kojem se području vozilo kretalo.

```
for (int i = 0; i < KML.Count; i++)
            {
                int indGodina = KML[i].localTime.Year - 2009;
                int indMjesec = KML[i].localTime.Month - 1;
                int indDana = KML[i].localTime.Day - 1;

                //Podrucje grada Zagreba
                if (KML[i].matchX > x1 && KML[i].matchX < x3 && KML[i].matchY < y1 &&
KML[i].matchY > y2)
                {
                    troskoviPoGodinama[indGodina].dodajTrosak(KML[i].matchDistance,
indMjesec, indDana);
                    troskoviPoGodinama3[indGodina].dodajTrosak(KML[i].matchDistance,
indMjesec, indDana);
                }
                //podrucje Zagrebacke zupanije
                else
                {
                    troskoviPoGodinama2[indGodina].dodajTrosak(KML[i].matchDistance,
indMjesec, indDana);
                    troskoviPoGodinama3[indGodina].dodajTrosak(KML[i].matchDistance,
indMjesec, indDana);
                }
            }
            //udaljenost grad Zagreb
            for (int i = 0; i < troskoviPoGodinama.Count; i++)
            {
                TrosakGodina l = troskoviPoGodinama[i];
                for (int j = 0; j < l.troskoviMjesec.Length; j++)
                {
                    TrosakMjesec tm = l.troskoviMjesec[j];
                    for (int k = 0; k < tm.troskoviDan.Length; k++)
                    {
                        double td = tm.troskoviDan[k];
                        if (td != 0)
                        {
                            string dan = (k + 1) + "." + (j + 1) + "." + (i + 2009);
                            dataGridView1.Rows.Add(dan, Convert.ToString(td / 1000));
                        }
                    }
                }
            }
            //udaljenost izvan Zagreba
            for (int i = 0; i < troskoviPoGodinama2.Count; i++)
            {
                TrosakGodina l = troskoviPoGodinama2[i];
                for (int j = 0; j < l.troskoviMjesec.Length; j++)
                {
                    TrosakMjesec tm = l.troskoviMjesec[j];
                    for (int k = 0; k < tm.troskoviDan.Length; k++)
                    {
                        double td = tm.troskoviDan[k];
                        if (td != 0)
                        {
                            string dan = (k + 1) + "." + (j + 1) + "." + (i + 2009);
                            dataGridView2.Rows.Add(dan, Convert.ToString(td / 1000));
                        }
                    }
                }
            }
```

Slika 14: Kod za izračun udaljenosti unutar i izvan grada Zagreba

For petlja vrši istu stvar kao i kod računanja troškova koja je prikazana na slici 13, dok je jedina promjena unutar if naredbe. Unutar naredbe if su ograničenja koja su uzeta iz Google Earth-a u kojem ako se uvjet ispuni udaljenosti koje je vozilo obavilo tijekom svoga puta su bile unutar grada Zagreba, a ako uvjet nije ispunjen tada je vozilo bilo izvan grada. Unutar naredbe if nalaze se tri varijable (troskoviPoGodinama, troskoviPoGodinama2 i troskoviPoGodinama3), te svaka od te tri varijable se prepisuju udaljenostima ili troškovima.

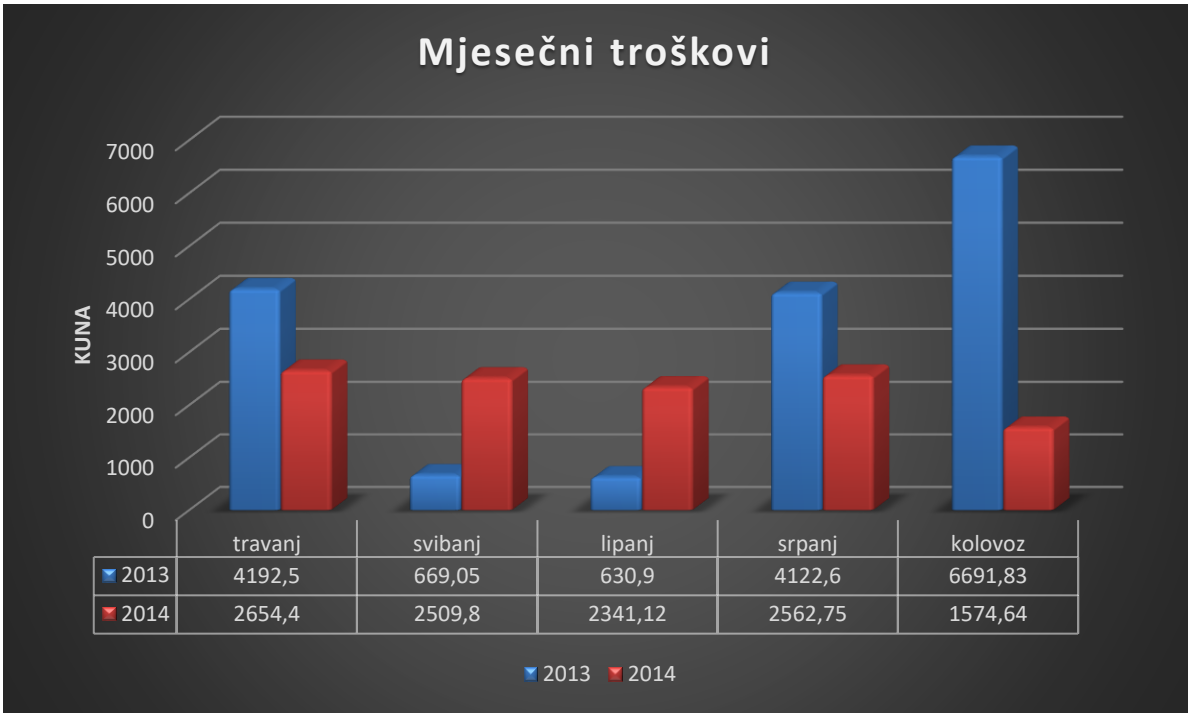
Ako je uvjet ispunjen u naredbi if tada se izvršava blok naredbe troskoviPoGodinama koje se prepisuju u tablicu dataGridView1 u kojoj su pohranjeni podaci udaljenosti unutar grada Zagreba i troskoviPoGodinama3 u kojoj se spremaju troškovi koje je vozilo obavilo unutar Zagreba. Ako prvi uvjet naredbe if nije ispunjen tada se vrši blok naredbi koji se nalaze unutar else. Unutar naredbe else podaci za udaljenosti izvan Zagreba se spremaju pod troskoviPoGodinama2, a troškovi pod troskoviPoGodinama3.

Kako se računaju ukupni troškovi tada na njih ne utječe uvjet gdje je vozilo bilo pozicionirano tijekom obavljana puta te se zbog toga naredba troskoviPoGodinama3 nalazi u unutar obje naredbe kako se ne bi izgubio podatak. Troškovi za vrijeme radnog vremena moraju ispunjavati uvjete, dok osnovni ne. Isto tako ukupna prijeđena udaljenost mora ispunjavati uvjet da li je unutar ili izvan Zagreba, pa zbog toga za prvi uvjet podatke spremamo pod troskoviPoGodinama, a za drugi troskoviPoGodinama2.

6. PRIKAZ RUTA I TROŠKOVA

Kako bi mogli usporedili troškove voznog parka uzeti su podaci od 1. travnja do 31. kolovoza 2013. i 2014. godine. Razlog zbog čega će se uzimati navedeni podaci su zbog usporedbe potrošnje između dvije godine. Mjesečni troškovi su prikazani grafikonom 1.

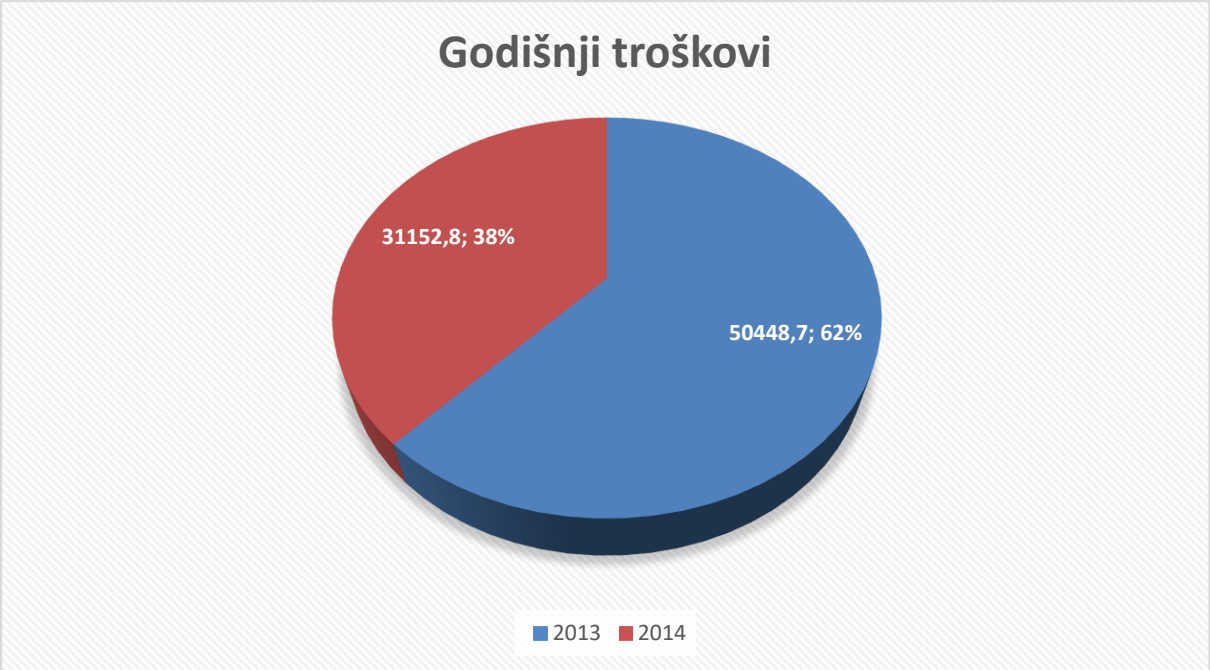
Grafikon 1: Mjesečni troškovi



Ukupna mjesečna potrošnja u 2013. godini iznosi 16.306,88 kn, dok za isti period u 2014 godini taj iznos je 11.642,71 kn što ukazuje da je potrošnja u 2014 godini smanjena za 4.664,17 kuna.

Kod godišnjih troškova uzeti su svi podaci koji su bili na raspolaganju. Za 2013. godinu uzeti su podaci u periodu od 1. travnja do 31. prosinca, a za 2014. godinu od 1. siječnja do 31. kolovoza. Godišnji troškovi prikazani su grafikonom 2

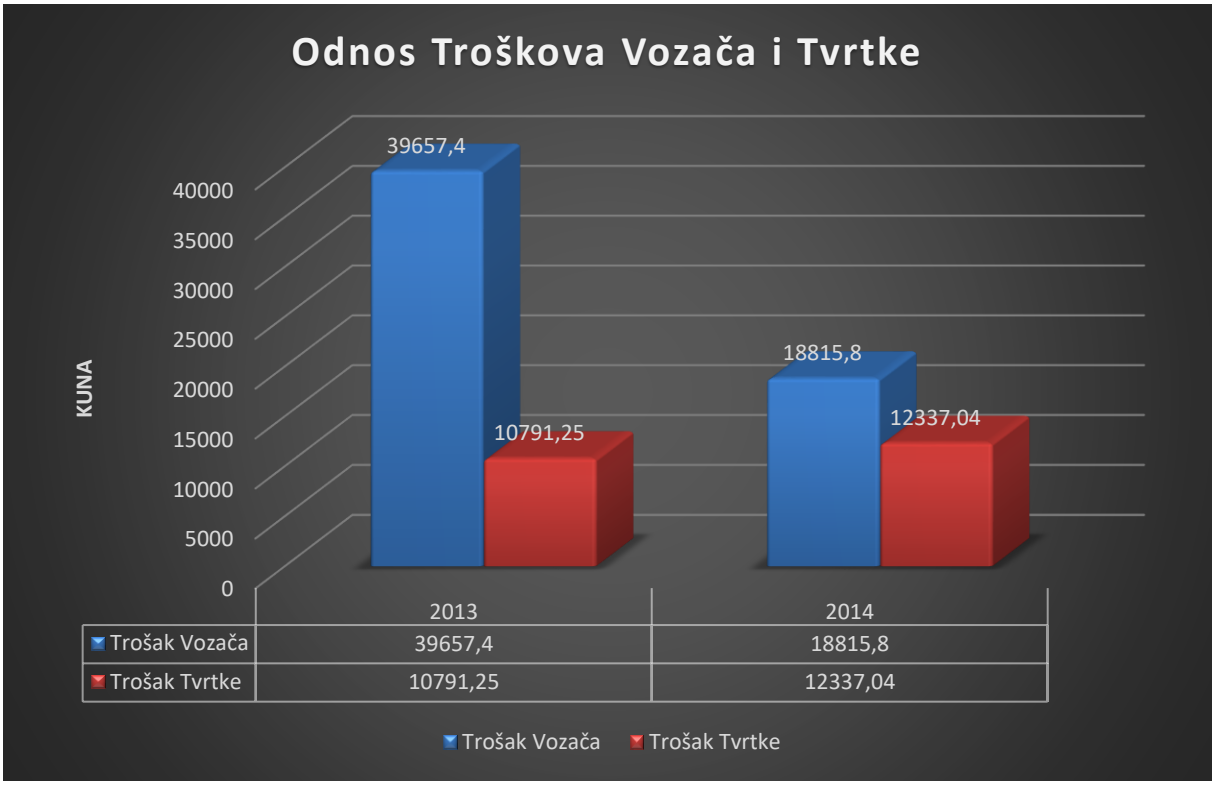
Grafikon 2: Godišnji troškovi



Ukupni godišnji troškovi za 2013. i 2014. godinu iznose 81.601,5 kuna. Troškovi su prema podacima koji su dobiveni za rad veći u 2013 godini. Odnos većih i manjih troškova ovisi o ljetnim i zimskim mjesecima te tadašnjim stanjima na cesti i mjerama štednje. Za detaljniji uvid u troškove potrebni su podaci za svih 5 godina praćenja vozila.

Troškove tijekom radnog vremena uključuje podjelu troškova ovisno da li je vozač obavio rutu u predviđeno radno vrijeme ili ne. Predviđeno radno vrijeme koje je potrebno za obavljanje dostave je od 7 sati ujutro do 17 popodne. Ako unutar radnog vremena vozač obavi rutu svi troškovi se prepisuju tvrtki, dok u slučaju prelaska granice troškovi se prepisuju vozaču, ali samo oni koji su izvan okvira obavljenog vremena. Godišnji troškovi vozača i tvrtke prikazani su grafikonom 3.

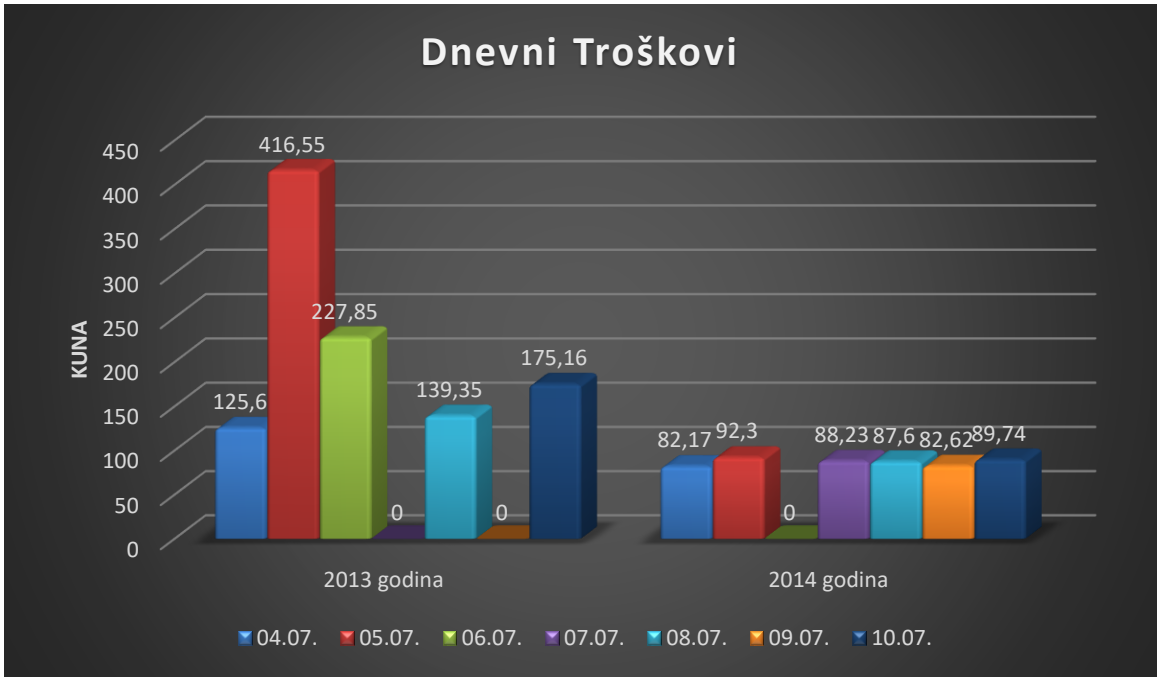
Grafikon 3: Troškovi vozača i tvrtke



Ukupni troškovi Vozača u 2013. i 2014. godini iznosi 58.473,18 kuna, dok trošak tvrtke iznosi 23.128,29 kuna. Iz grafikona je vidljivo da se odnos između vozača i tvrtke u 2014 godini dosta poboljšao ali još uvijek vozač ne može jednu rutu obaviti u određenome vremenu. Treba također uzeti u obzir da u jednome danu vozilo može obaviti više ruta i mogućnost obavljanja ruta s više vozila tako da je odnos ipak stabilniji.

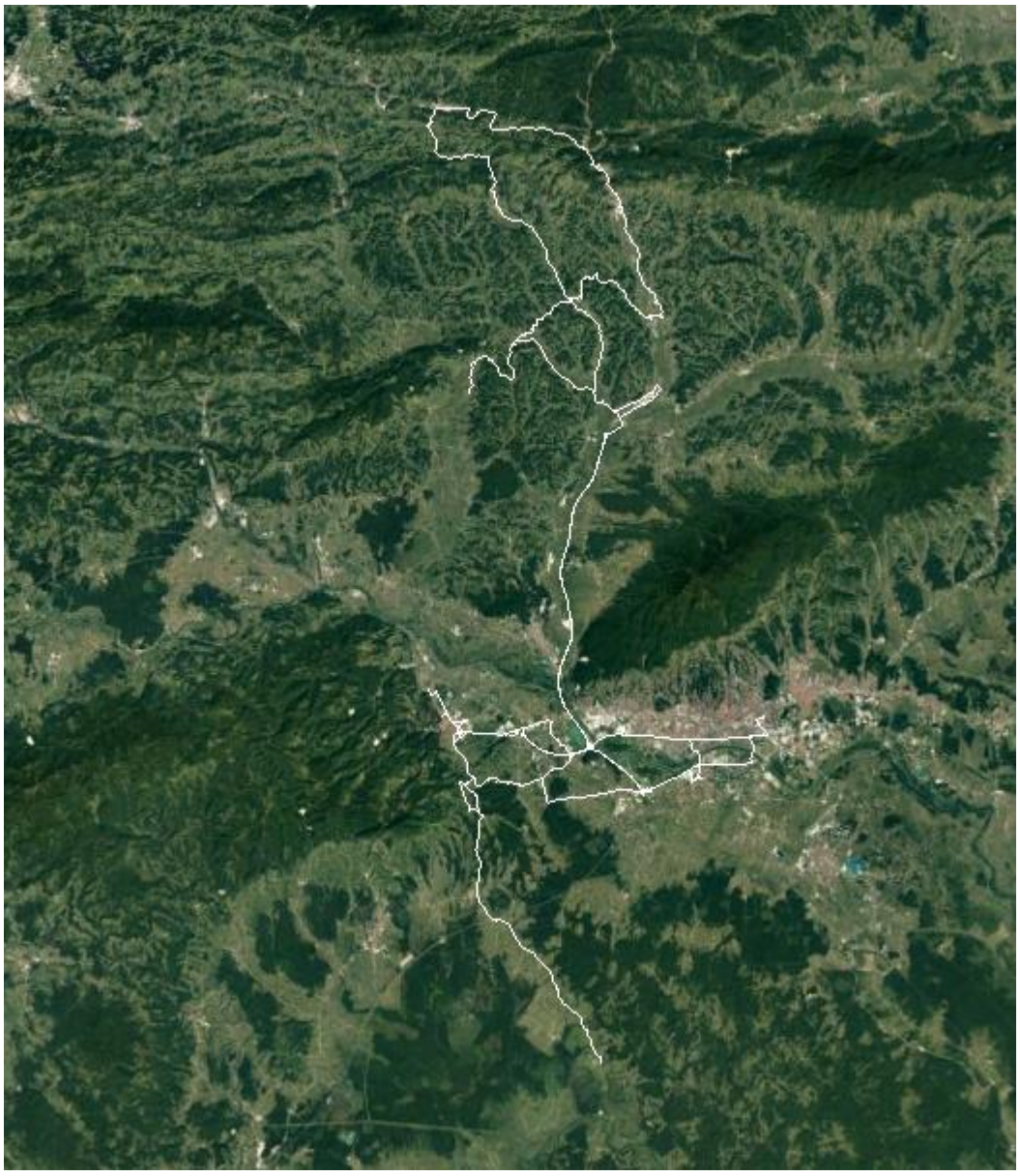
S obzirom na to da su se u ovome radu koristili podaci koji sadržavaju podatke o vozilima koji su praćeni kroz 17 mjeseci za dnevne troškove ima previše podataka koji se mogu prikazati. Zbog toga za prikaz dnevni troškova uzeti su podaci u ljetnim mjesecima (7 mjesec) za period od 5. srpnja do 10. srpnja za oba dvije godine i prikazani su za usporedbu na grafikonu 4.

Grafikon 4: Dnevni troškovi 2013/2014 godine



Iznosi na kojima nema prikazanih troškova je prikazan brojem 0 zbog toga što tih dana nisu obavljani nikakvi transportni poslovi. Iako je u 2013. godini dva dana vozilo mirovalo u odnosu jedan dan u 2014. godini, troškovi su ipak veći. Ukupni dnevni troškovi za godinu 2013. prema grafikonu 4 iznosi 1.084,5 kuna, dok za isti period u 2014. godini taj trošak iznosi samo 522,67 kuna.

Za prikaz ruta korišten je alat Google Earth u kojem se prikazuju sve rute koje je vozilo obavilo u jednom danu. Google Earth služi za prikaz zemljine površine i svemira u 3D prostoru te nudi velik broj alata za obradu podataka. Aplikacija pritiskom na gumb KML stvara kml datoteku pomoću koje možemo vidjeti mjesta na zemljinog površini gdje se vozilo kretalo. Kml datoteka služi za prikaz geografskih podataka na mapi kako je prikazano na slici 15.



Slika 15: Ruta vozila

Na slici su prikazane rute vozila koje je praćeno od svibnja do srpnja 2013. godine. Ruta je obavljena na području grada Zagreba, Zagrebačke županije te Krapinsko – zagorske županije.

7. ZAKLJUČAK

Iako danas postoje mnogi problemi pri transportu dobara iz jednog mjesta u drugo, ispitivanja koja su se vršila su pokazala da unatoč tome postoje bitni pomaci u smanjenju troškova. Postoje već mnoga rješenja koja se primjenjuju kako bi se poveća efikasnost voznog parka, a time smanjuju negativni učinci. U sklopu ispitivanja koja su se vršila za izračun potrošnje voznog parka dobiveni su podaci koji su pokazali da se svake druge godine troškovi smanjuju što pozitivno utječe na profit tvrtke ali isto tako i na smanjenje negativnih učinka na okoliš.

Za potrebe ovoga rada izrađena je aplikacija u programskom jeziku c# koja pohranjuje velike količine podataka u relacijsku bazu kako bi korisnik imao uvid na podatke koje koristi. Također pruža mogućnost izračuna troškova voznog parka i prijeđenog puta unutar grada Zagreba i okoline te izrađuje kml datoteku koja da je korisniku uvid na karti gdje se je vozilo kretalo i u kojem danu je vršeno putovanje.

Aplikacija ima dosta prostora za proširenje svoje funkcije i analiziranje drugih podataka. Moguće je preko podataka prikazati ovisnost brzine na pojedinim prometnicima ovisno o ljetnim i zimskim mjesecima. Ispitivanja bi se mogla vršiti analizirajući rast ili pad brzine te bi se daljnjim ispitivanjem tražio uzrok tome. Aplikaciju nudi dobru bazu za daljnju nadogradnju i ispitivanje novim podacima.

POPIS LITERATURE

[1] URL: http://fs-tech.ca/how-it-works/science-of-gps/ (pristupljeno: srpanj 2017.)

[2] URL: http://e-student.fpz.hr/Predmeti/L/Lokacijski_i_navigacijski_sustavi/Materijali/05-Satelitski_pozicijski_sustavi.pdf (pristupljeno: srpanj 2017.)

[3] Kaplan, E.D., Hegarty, C.J.: Understanding GPS principles and applications second edition

[4] URL: http://www.navipedia.net/index.php/Electromagnetic_Beam_Bending (pristupljeno: srpanj 2017.)

[5] URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (pristupljeno: srpanj 2017.)

[6] URL: http://moodle.srce.hr/2016-2017/pluginfile.php/801228/mod_resource/content/5/Vje%C5%BEba3.pdf (pristupljeno: srpanj 2017.)

[7] URL: http://moodle.srce.hr/2016-2017/pluginfile.php/801233/mod_resource/content/7/Vje%C5%BEba4.pdf (pristupljeno: srpanj 2017.)

[8] URL: http://grdelin.phy.hr/~ivo/Nastava/Baze_podataka/predavanja-2002/03_Relacijski-model.pdf (pristupljeno: srpanj 2017.)

[9] URL: http://moodle.srce.hr/2016-2017/pluginfile.php/801265/mod_resource/content/1/%C5%A0ali%C4%87.pdf (pristupljeno: srpanj 2017.)

[10] URL: http://moodle.srce.hr/2016-2017/pluginfile.php/801238/mod_resource/content/4/5.tjedan.pdf (pristupljeno: srpanj 2017.)

[11] URL: http://moodle.srce.hr/2016-2017/pluginfile.php/801240/mod_resource/content/7/Vje%C5%BEba5.pdf (pristupljeno: srpanj 2017.)

[12] URL: https://www.researchgate.net/figure/281458585_fig6_Figure-11-GPS-Map-Matching-AB-and-CD-are-road-segments-p1-p2-p3-p4-and-p5-are-raw (pristupljeno: srpanj 2017.)

[13] URL: http://e-student.fpz.hr/Predmeti/L/Lokacijski_i_navigacijski_sustavi/Materijali/06-Map_matching.pdf (pristupljeno: srpanj 2017.)

[14] URL: https://en.wikipedia.org/wiki/Map_matching (pristupljeno: srpanj 2017.)

POPIS ILUSTRACIJA

Popis Slika:

Popis tablica:

Popis grafikona:

Sveučilište u Zagrebu

Fakultet prometnih znanosti

10000 Zagreb

Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj završni rad

isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog rada

pod naslovom **Analiza troškova voznog parka zasnovana na obradi GPS podataka**

na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

Student/ica:

U Zagrebu, 4.9.2017

(potpis)