MJERENJE ODMAKA POZICIJE U MEMS INERCIJSKOJ NAVIGACIJI TEMELJENOJ NA INTEGRIRANJU

Horvat, Josip

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

Mjerenje odmaka pozicije u MEMS inercijskoj navigaciji temeljenoj na integriranju

Josip Horvat

Zagreb, veljača 2020.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 10.2.2020.

Josip Horvat

Predgovor

Zahvaljujem svom mentoru, profesoru Goranu Đambiću, koji me usmjerio te pomogao mi pri izradi ovog završnog rada. Osim toga, htio bih zahvaliti Visokom učilištu Algebra za svo preneseno znanje i sve prilike koje mi je pružilo. Također bih htio zahvaliti svim zaposlenicima na ljubaznosti, dostupnosti i pomoći koju su mi pružili.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Navigacija postaje dio svakodnevice te sve više sustava koristi neku vrstu navigacije. Najčešći tip navigacije je GPS (engl. *Global positioning system*) pomoću kojeg se određuje lokacija nekog objekta. Navigacija za rute je došla do impresivne razine da možemo bez problema otići na putovanje u nama nepoznato područje te se snaći i odraditi što smo došli odraditi. To vrijedi za putovanje pješke, automobilom, biciklom, vlakom i ostalim prijevoznim sredstvima. Područje gdje je takva navigacija nepouzdana su zgrade. Iz tog razloga će ovaj rad napraviti analizu očitavanja lokacije pomoću senzora koji su danas uključeni u skoro svaki pametni telefon te ju usporediti s očitavanjima lokacije dobivene pomoću GNSS-a (engl. *Global Navigation Satellite System*). Očitavanja će se raditi u zatvorenim prostorima i vani te će se analizirati efikasnost navigacijskih sustava u određenim okolinama.

Ključne riječi: Android, senzori, akcelerometar, žiroskop, GNSS.

Abstract

Navigation is becoming a part of our everyday lives and more and more systems are using some type of navigation. The most common type of navigation is GPS (Global positioning system) which determines the location of an object. The route navigation has reached an impressive level so that we can easily go on a journey to an unfamiliar area and manage and do what we came to do. This applies to walking, driving using a car, bike, train or by any other means of transport. The area where such navigation is unreliable is buildings. For this reason, this paper will analyze the location readings using the sensors that are included in almost every smartphone today and compare them with the location readings provided by GNSS (Global Navigation Satellite System). The readings will be done indoors and outdoors and will analyze the efficiency of navigation systems in specific environments.

Keywords: Android, sensors, accelerometer, gyroscope, GNSS.

Sadržaj

1. Uvod

Tema ovog rada je analiza podataka dobivenih pomoću senzora uključenih u većinu današnjih pametnih telefona. Analiza će se odrađivati na četiri različita skupa podataka: GNSS očitavanja lokacije u zatvorenom prostoru, GNSS očitavanja lokacije vani, lokacija dobivena uz pomoć senzora u zatvorenom prostoru, lokacija dobivena uz pomoć senzora vani.

Cilj ovog rada je otkriti odmak koji se dešava pri svakom od navedenih očitavanja, a primarni fokus je analiza odmaka inercijskog navigacijskog sustava s protekom vremena.

Analiza će se odvijati putem mobilne aplikacije napravljene za Android operativni sustav koja će biti temeljni praktični doprinos ovog rada. Aplikacija će omogućiti korisniku da ucrta rutu kojom će proći te da nakon toga prijeđe tu rutu s omogućenim inercijskim navigacijskim sustavom i GNSS-om. Stvarna ruta i očitane rute će biti vizualno prikazane korisniku putem Google karata. Nakon što korisnik potvrdi da je put prijeđen, bit će dostupna detaljnija analiza s prikazanim odmacima.

Drugo poglavlje sadrži detaljni opis projekta kojim se ovaj rad bavi. Treće poglavlje opisuje inercijske navigacijske sustave, što su oni, koje vrste postoje i njihove referentne okvire. U četvrtom poglavlju se analizira izračun pozicije korištenjem inercijskih senzora te odmak koji se pri tom dobiva. Peto poglavlje opisuje arhitekturu aplikacije i objašnjava kako su određeni dijelovi izvedeni. Šesto poglavlje analizira razlike pri dobivanju pozicije koristeći GNSS i INS te odmak koji se stvara u tom procesu. U sedmom poglavlju se nalazi zaključak ovog rada.

2. Projektni zadatak i hipoteza

Cilj ovog rada je razviti aplikaciju kojom će se moći odrediti odmak pozicije dobivene pomoću očitavanja MEMS inercijske navigacije temeljene na integriranju. Aplikacija će također uključiti očitavanja pozicije GNSS-a, u svrhu bolje analize podataka i odstupanja.

Inercijski navigacijski sustavi (engl. *Inertial Navigation Systems*, skraćeno INS) su se inicijalno proizveli za rad s raketama tijekom drugog svjetskog rata. Pomoću dva žiroskopa i jednog akcelerometra se računao azimut rakete pri letu. Pedesetih godina prošlog stoljeća su se počeli koristiti za izradu sustava za navođenje. [7] Šezdesetih i sedamdesetih godina, koristili su se u navođenju i navigaciji aviona koristeći žiroskope za određivanje trenutne pozicije koristeći prethodno očitanu poziciju. [8]

U današnje doba su akcelerometri, žiroskopi, magnetometri i slični senzori ugrađeni u brojne pametne telefone. Stariji pametni telefoni možda nemaju uvijek sve navedene senzore, no može se pretpostaviti da imaju ugrađen barem akcelerometar. Sam akcelerometar prikazuje akceleraciju u trodimenzionalnom prostoru, no može se koristiti za izračun brzine kretanja i pozicije. Problem koji se pojavljuje kod senzora koji su ugrađeni u pametne telefone je što nisu vrlo precizni te svaki od njih proizvodi određeni odmak (engl. *drift*) pri očitavanju vrijednosti.

Iako je GNSS pouzdaniji i češće korišten danas, on je vrlo neprecizan u zatvorenom prostoru, ponekad i potpuno neupotrebljiv. U ovakvim situacijama INS postaje bolja opcija. Možda ne kod pametnih telefona jer se s vremenom podaci postaju vrlo nepouzdani radi odmaka. Ukoliko se koriste precizni senzori, INS može biti idealna opcija u zatvorenom prostoru. Ovaj rad će se fokusirati na očitavanje lokacije koristeći pametne telefone te su očekivani rezultati sljedeći:

Tablica 2.1 Očekivani rezultati očitavanja lokacije

Okolina / Izvor lokacije	GNSS	MEMS inercijska navigacija
Zatvoreni prostor	Veliki odmak	Veliki odmak
Vani	Mali odmak	Veliki odmak

Mjerenja će biti obavljena s tri moderna pametna telefona različitih proizvođača kako bismo mogli dodatno usporediti potencijalne razlike u rezultatima s obzirom na potencijalno različite senzore koji se nalaze u tim pametnim telefonima.

Dodatni praktični doprinos rada će biti i zaključak koliko je zaista velik odmak inercijskog navigacijskog sustava s protekom vremena te će dati zaključak može li se takav sustav na modernim pametnim uređajima uspješno koristiti za navigaciju.

Najčešća upotreba INS-a je za mobilne igrice. Tržište mobilnih igrica je poraslo s godinama te se može uočiti da ih mnogi igraju, a najčešće su to djeca. Čak se auto industrija prilagođava modernim trendovima te ugrađuju sve više utora za punjenje pametnih uređaja jer pretpostavljaju da će ih djeca u prijevozu koristiti. Kod igrica se INS najčešće koristi za upravljanje određenim sustavom kao što je, na primjer, automobil. Naginjanjem mobitela lijevo, automobil skreće ulijevo, a naginjanjem desno skreće udesno.

3. Inercijski navigacijski sustavi

Inercijski navigacijski sustavi su sustavi navigacije kod kojih se koristi tehnika mjerenja koristeći senzore akcelerometar, magnetometar i žiroskop. Podatke koje nam navedeni senzori daju su pozicija i orijentacija objekta relativno poznatoj početnoj točki, orijentaciji i brzini. Od navedenih senzora, može se koristiti samo akcelerometar što može biti posljedica odluke da se smanji potrošnja baterije ili ukoliko mobitel ne posjeduje ostale senzore. Magnetometar i žiroskop služe kao pomoćni alati za dobivanje preciznijih mjera. U ovom radu žiroskop nije presudan jer se orijentacija može odrediti pomoću očitavanja dobivenih iz magnetometra.

3.1. Vrste i principi rada

Svaki inercijski navigacijski sustav se temelji na nekoj metodi ili tehnici, pri čemu se na najvišoj razini one dijele na tehnike temeljene na elektromagnetskim valovima i metode procesiranja slika.

3.1.1. Tehnike temeljene na elektromagnetskim valovima

Tehnike koje su temeljene na elektromagnetskim valovima koji su zapravo vidljiva ili nevidljiva svjetlost. Tehnike temeljene na radijima procjenjuju lokaciju u odnosu na poznate lokacije radio odašiljača poput satelita. Dijele se na tehnike temeljene na satelitima i radio metode.

Kod tehnika kojima se dohvaća lokacija i orijentacija korištenjem satelitskih signala može biti problematično ako se objekt nalazi u zatvorenom prostoru jer su satelitskim signalima krov i zidovi prepreka. Pod ove tehnike spada danas najpopularniji način dobivanja pozicije i orijentacije – globalni pozicijski sustav (GPS). Osim GPS-a, postoje još dva relativno poznata satelitska navigacijska sustava – Galileo i GLONASS.

Kod radio metoda se lokacija određuje mjerenjem jačine dobivenog signala i usporedbom istog s jačinom signala emitiranog referentnog signala. U ovoj kategoriji su poznate tehnologija RFID (identifikacija pomoću radio frekvencije) te sljedeće mrežne komunikacijske tehnologije:

- WiFi

- Bluetooth
- ZigBee

3.1.2. Metode procesiranja slika

Metode procesiranja slika čine metode koje lokaciju određuju procesiranjem informacija dobivenih iz slike. Dvije glavne podvrste su metoda otkrivanja oznaka i metoda otkrivanja značajki.

Metoda otkrivanje oznaka zahtijeva bazu podataka u kojoj se nalazi skup identifikacija oznaka s pripadnom geografskom lokacijom. Time se omogućuje da korisnik ove metode pronađe i skenira određenu oznaku (npr. QR kod ili barkod) te se pomoću nje iz baze podataka dohvaća lokacija.

Metodom otkrivanja značajki se procesira slika iz koje se izvlače određene značajke. Značajka može biti bilo što, no skoro uvijek je povezana s problemom koji rješava program koji koristi ovu metodu. Može biti vrlo procesno zahtjevna, ali dobiveni rezultat može dovesti do vrlo visoke preciznosti određivanja lokacije.

3.2. Referentni okviri

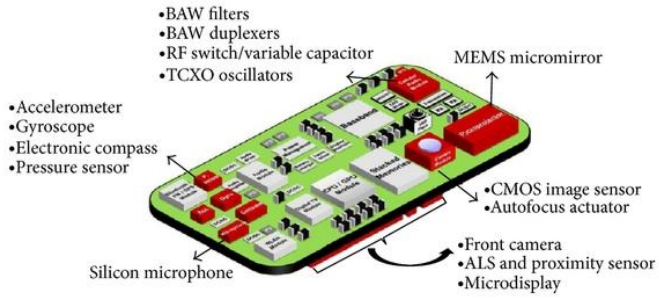
Referentni okviri se dijele na dva glavna tipa – inercijski i ne inercijski. Inercija je svojstvo materije u kojoj objekt koji se nalazi u mirovanju želi ostati u mirovanju, a objekt koji se kreće želi se nastaviti kretati ravno, osim ako na njega djeluje neka druga sila. Inercijski referentni okvir je referentni okvir u kojem objekt ili ostaje u mirovanju ili se kreće konstantnom brzinom, osim ako na njega djeluje neka druga sila. Kad tijelo ostavlja dojam da ne djeluje u skladu s inercijom, nalazi se u ne inercijskom referentnom okviru, tj. ubrzava.

Za lakše shvaćanje samih pojmova, promotrimo ih kroz primjer. Osoba se sama vozi u tramvaju te ništa ne vidi niti čuje. Ukoliko tramvaj vozi ravnom linijom i ne ubrzava, tj. ima konstantnu brzinu, osoba je u inercijskom referentnom okviru. Tek kada tramvaj prestane voziti konstantnom brzinom, osoba ulazi u ne inercijski referentni okvir.

3.3. MEMS senzori

Mikroelektromehanički sustavi (engl. *Microelectromechanical systems*, skraćeno MEMS) su sustavi karakteristični po maloj veličini i načinu na koji su napravljeni. Veličine

komponenti koje se koriste u takvim sustavima su veličine 1-100 mikrometara. Iako su vrlo mali, vrlo su kompleksni i često imaju pokretne dijelove kojima upravlja integrirana mikroelektronika. [5] Najpopularniji senzori danas su dostupni u pametnim telefonima koji uključuju mnogo korisnih senzora za geomatske aplikacije koje se baziraju na određivanju i procesiranju lokacija. U pametne telefone su ugrađeni raznovrsni unutarnji senzori poput digitalnih fotoaparata, GNSS prijemnika, inercijalnih platformi zasnovanih na žiroskopima, akcelerometrima, magnetometrima i RFID sustavima. [1]



Slika 3.1 Senzori unutar pametnog telefona

3.3.1. Linearni akcelerometar

Linearni akcelerometar mjeri akceleraciju u odnosu na jednu os inercijskog referentnog okvira. S obzirom da se mjeri akceleracija samo na jednoj osi, često se u praksi koriste tri takva akcelerometra kako bi se dobila akceleracija u trodimenzionalnom prostoru. Bazira se na drugom Newtonovom zakonu koji kaže da je akceleracija tijela proporcionalna sili koja djeluje na to tijelo i obrnuto proporcionalna masi tog tijela.

Akcelerometar se može koristiti u mnogo svrha, na primjer:

- Auto – mjerenje brzine kretanja
- Igrice za pametne uređaje – upravljanje kretanjem lika u igrici

3.3.2. Žiroskop

U današnjim osobnim komunikacijskim uređajima se koriste žiroskopi bazirani na *Coriolisovom* efektu kojeg mjere pomoću vibracijskog elementa koji je sadržan u samom žiroskopu. Objekt mase m koji se kreće brzinom v u referentnom okviru koji se okreće kutnom brzinom ω podliježe *Coriolisovoj* sili. Formula se može vidjeti u izrazu (1). [3]

$$F_c = -2m\,(\omega \cdot v) \quad (1)$$

Kad se koordinatni sustav žiroskopa okrene, *Coriolisova* sila inducira sekundarnu vibraciju na os okomitu na mjernu os.

Žiroskop se najčešće koristi za određivanje orijentacije nekog tijela poput aviona ili raketa.

3.3.3. Magnetometar

Magnetometar je senzor koji se koristi za određivanje snage magnetskog polja. Kod INS-a se koristi za smanjenje odmaka te kao pomoć pri određivanju orijentacije jer se pomoću njega može odrediti gdje se nalazi sjever. Problem kod magnetometara je što ih ostali magnetizirani objekti čine manje preciznima. [2]

4. Izračun pozicije korištenjem inercijskih senzora

4.1. Metoda dvostruke integracije

Metoda dvostruke integracije služi za izračunavanje pozicije uređaja korištenjem dvostruke integracije vrijednosti dobivenih iz akcelerometara. Osim toga, potrebno je uzeti u obzir gravitaciju prije nego što se krene s integriranjem. Dakle, kad se dobiju vrijednosti iz akcelerometara, vrijednosti se ispravljaju s obzirom na gravitaciju. Dobivene vrijednosti se integriraju čime ćemo dobiti brzinu. Nakon toga, još jednim integriranjem, dobivamo poziciju. Primjer dobivanja pomaka, brzine i akceleracije odrađen je kroz Kôd 5.2.

Kako bismo odredili i orijentaciju, moramo koristiti žiroskop ili kombinaciju magnetometra i akcelerometra. U ovom primjeru se koristi kombinacija magnetometra i akcelerometra jer je to preporučeni način rada prema Android dokumentaciji. Prvi korak je određivanje matrice rotacije, za što postoji ugrađena metoda na razredu `SensorManager`:

```
SensorManager.getRotationMatrix(RT, null,
accelerometerReadings, magnetometerReadings);
```

Gdje se koriste sljedeće varijable:

- *RT* – matrica rotacije (polje od 9 decimalnih (*float*) brojeva) u svjetskom koordinatnom sustavu koji je definiran kao izravna ortonormalna osnova:
    - X je definiran kao vektorski produkt Y i Z vrijednosti (tangencijalan za tlo na trenutnom položaju uređaja i otprilike usmjerava prema istoku)
    - Y je tangencijalan tlu na trenutnom položaju uređaja i usmjerava prema magnetskom Sjevernom polu
    - Z je usmjeren prema nebu i okomit je na tlo
- *I* – matrica nagiba (nije korištena – proslijeđena *null* vrijednost)
- *accelerometerReadings* (polje od 3 *float* broja) – vrijednosti koje smo očitali koristeći akcelerometar
- *magnetometerReadings* (polje od 3 *float* broja) – vrijednosti koje smo očitali koristeći magnetometar

Nakon toga koristimo ugrađenu metodu za dobivanje orijentacije:

```
SensorManager.getOrientation(RT, orientation);
```

Gdje se koriste sljedeće varijable:

- *RT* – matrica rotacije (polje od 9 decimalnih (*float*) brojeva) dobivena iz prethodnog koraka
- *orientation* (polje od 3 *float* broja) – orijentacija u svjetskom koordinatnom sustavu, brojevi koje dobijemo su sljedeći:
    - Azimut (rotacija oko Z osi)
    - *Pitch* (rotacija oko X osi)
    - *Roll* (rotacija oko Y osi)

4.2. Odmak

Pri određivanju pozicije dolazi do određenog odmaka radi nesavršenosti akcelerometra. Postoji nekoliko faktora koji dovode do greške koje ćemo analizirati u sljedećim odlomcima.

4.2.1. Konstantno odstupanje

Konstantno odstupanje akcelerometra je odmak izlaznog signala od prave vrijednosti izražene u $\frac{m}{s^2}$. Označava se s ε te zbog duple integracije izaziva četverostruku grešku koja raste četverostruko s obzirom na vrijeme. Akumulirana greška u poziciji je definirana formulom (2) u kojoj t označava vrijeme integracije, a s poziciju. [2]

$$s(t) = \varepsilon \cdot \frac{t^2}{2} \tag{2}$$

Kako bi se odstupanje moglo procijeniti, može se izračunati prosjek čitanja akcelerometra koji nije pod utjecajem akceleracije kroz duži period vremena. Kako bi se dobila ta vrijednost, potrebno je postaviti pametni telefon na stol, početi očitavati s njega vrijednosti akceleracije određen broj puta te nakon toga uzeti aritmetičku sredinu svih vrijednosti.

4.2.2. Termo-mehanički bijeli šum / slučajni hod brzine

Rezultati dobiveni iz akcelerometra odudaraju zbog sekvence bijelog šuma koji generira slučajni hod brzine. Slučajni hod brzine predstavlja proces određivanja brzine na temelju kretnji te vrijednosti očitanih za vrijeme kretanja. Efekt bijelog šuma na izračunatu poziciju možemo analizirati duplom integracijom vrijednosti dobivenih iz akcelerometra. Rezultat duple integracije bijelog šuma signala ε(t) kroz vremenski raspon $t = n \cdot \delta t$, gdje je n broj

vrijednosti dobivenog iz uređaja duž tog perioda, a δt je vrijeme između sukcesivnih vrijednosti, je definiran formulom definiranom izrazima (3) i (4).

$$\int_0^t \int_0^t \varepsilon(\tau)\, d\tau d\tau = \delta t \sum_{i=1}^{n} \delta t \sum_{j=1}^{i} N_j \tag{3}$$

$$= \delta t^2 \sum_{i=1}^{n} (n - i + 1) N_i \tag{4}$$

Pogreška u poziciji se pretpostavlja kao što je vidljivo u izrazima (5) i (6).

$$E\left(\int_0^t \int_0^t \varepsilon(\tau)\, d\tau d\tau\right) = \delta t^2 \sum_{i=1}^{n} (n - i + 1) E(N_i) \tag{5}$$

$$= 0 \tag{6}$$

Varijanca je definirana u izrazima (7) i (8), a njezina aproksimacija u izrazu (9) gdje se pretpostavlja da je δt mala vrijednost jer je za moderne MEMS akcelerometre učestalost uzorkovanja velika.

$$Var\left(\int_0^t \int_0^t \varepsilon(\tau)\, d\tau d\tau\right) = \delta t^4 \sum_{i=1}^{n} (n - i + 1)^2 Var(N_i) \tag{7}$$

$$= \frac{\delta t^4 n(n+1)(2n+1)}{6} Var(N) \tag{8}$$

$$\approx \frac{1}{3} \cdot \delta t \cdot t^3 \cdot \sigma^2 \tag{9}$$

Zaključak analize je da bijeli šum akcelerometra stvara slučajni hod brzine drugog reda sa standardnom devijacijom definiranom izrazom (10) koji raste proporcionalno s $t^{\frac{3}{2}}$.

$$\sigma_s(t) = \sigma \, t^{\frac{3}{2}} \cdot \sqrt{\frac{\delta t}{3}} \tag{10}$$

4.2.3. Šum treperenja, utjecaj temperature, pogreške pri kalibriranju

Ostali faktori koji utječu na odmak od stvarnih vrijednosti su:

- Šum treperenja – uzrokuje variranje odstupanja kroz vrijeme
- Utjecaj temperature – iako ovaj faktor ovisi o uređaju, linearnost odstupanja je vrlo niska. Može se ispraviti, ali to zahtjeva temperaturne senzore te korekciju na temelju očitanih vrijednosti.
- Pogreške pri kalibriranju – pogreške u faktorima razmjera, poravnanja i izlaznih linearnosti

5. Arhitektura i izvedba aplikacije

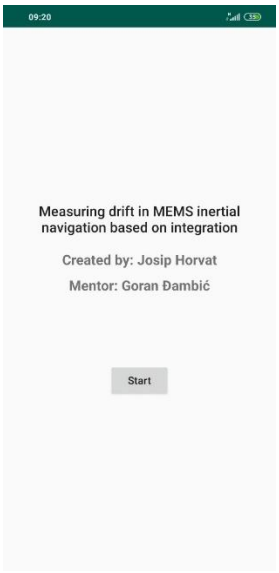
Aplikacija je napisana za Android operativni sustav koristeći programski jezik Java.

5.1. Arhitektura aplikacije

Aplikacija je napisana po preporučenim praksama pisanja Android aplikacija. Sučelje (ekrani) i logika koja se događa u pozadini su odvojeni, a unutar aktivnosti (engl. *Activity*) se ne piše nikakva logika osim postavljanja komponenti (koje se prikazuju korisniku, npr. tekst ili Google karta) da rade ono za što su predviđene. Logičke cjeline su odvojene u zasebne razrede te omogućuju lagano snalaženje u kodu. Razredi su grupirani u pakete koji imaju naziv kojim se lagano može odrediti čemu služe. Na primjer, sve aktivnosti se nalaze u paketu `com.horvat.bthesis.activities`, razred za rad s GNSS-om se je definiran kao `com.horvat.bthesis.data.gps.GpsManager`, a razred za rad s podacima koji su dobiveni iz senzora je definiran kao `com.horvat.bthesis.data.sensors.DataManager`.

5.1.1. Naslovni ekran

Naslovni ekran sadrži naslov rada, ime autora i mentora te gumb koji služi za prijelaz na sljedeći ekran.

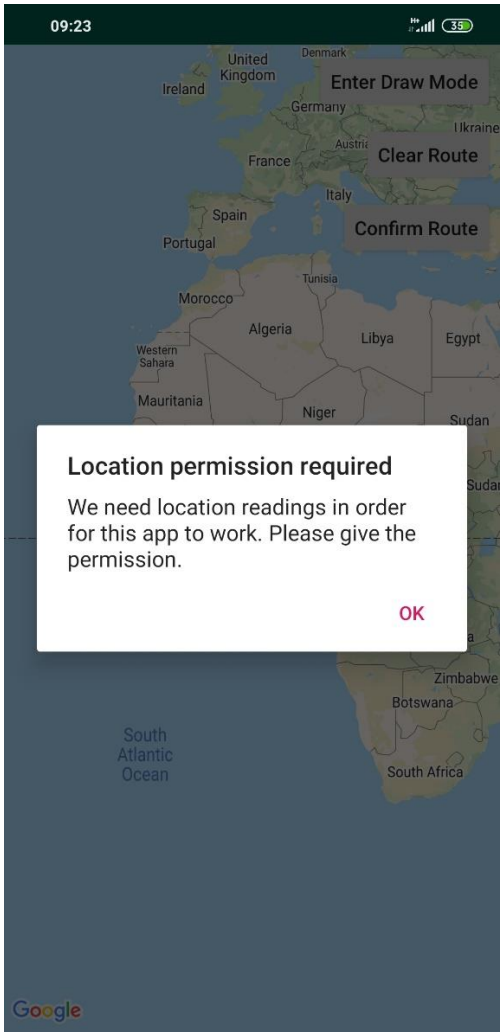


Slika 5.1 Naslovni ekran

5.1.2. Ekran za crtanje rute

Kako bi se odredila ruta koju će aplikacija koristiti kao točnu / referentnu rutu, korisniku se prikazuje mapa na kojoj može odrediti danu rutu.

Kada korisnik prvi puta dođe na ekran, ako nema upaljen GPS servis, aplikacija će ga zamoliti da ga upali (Slika 5.2) te će time olakšati daljnje korištenje. Prvo time što će se odmah zumirati na korisnikovu trenutnu lokaciju, a nakon toga time što neće morati opet pitati za lokaciju na sljedećem ekranu, na kojem će korisnik prelaziti rutu koju ucrta na trenutnom ekranu.

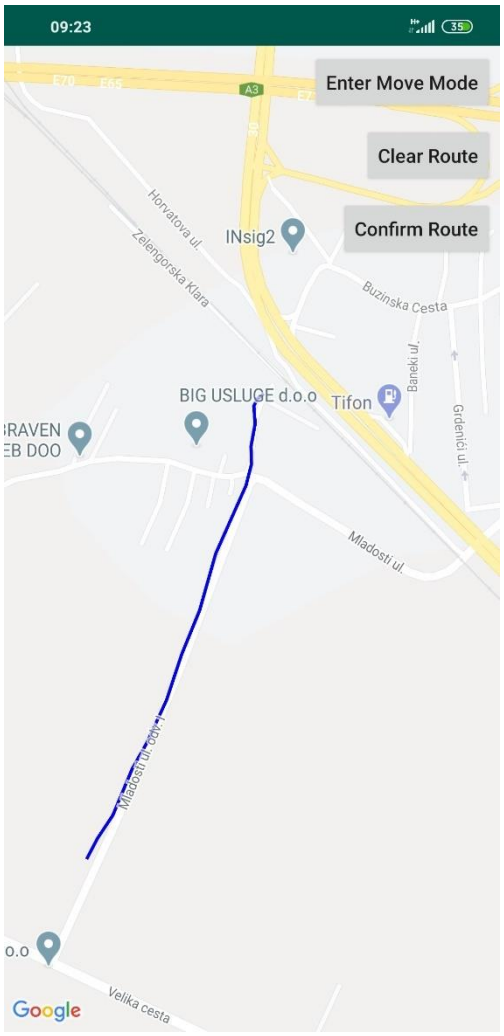


Slika 5.2 Zahtjev za dozvolu pristupa lokaciji mobilnog uređaja

Ekran nudi dva načina rada na mapi. Prvi način rada, ujedno i predefinirani, je onaj u kojemu korisnik može manipulirati mapom (pomicanje, okretanje). Drugi način rada je onaj u kojemu korisnik može crtati rutu na mapi.

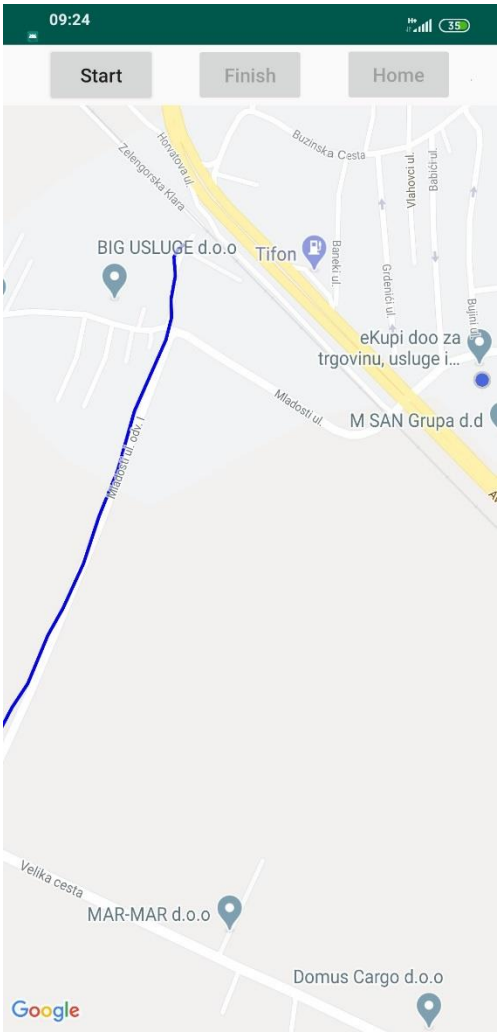
Slika 5.3 Ekran za crtanje rute

Ekran sadrži tri gumba koji se nalaze u gornjem desnom kutu, kao što je moguće vidjeti na slici (Slika 5.3). Prvi gumb služi za prebacivanje između prethodno definiranih načina rada. Drugi gumb služi za brisanje ucrtane rute kako bi mogao popraviti pogrešno ucrtanu rutu. Treći gumb je potvrdni te služi za prijelaz na sljedeći ekran. Ukoliko ruta nije ucrtana, korisnik će biti upozoren o toj činjenici te aplikacija neće prijeći na sljedeći ekran.



Slika 5.4 Crtanje vlastite rute

5.1.3. Ekran za praćenje rute pomoću inercijskih navigacijskih sustava



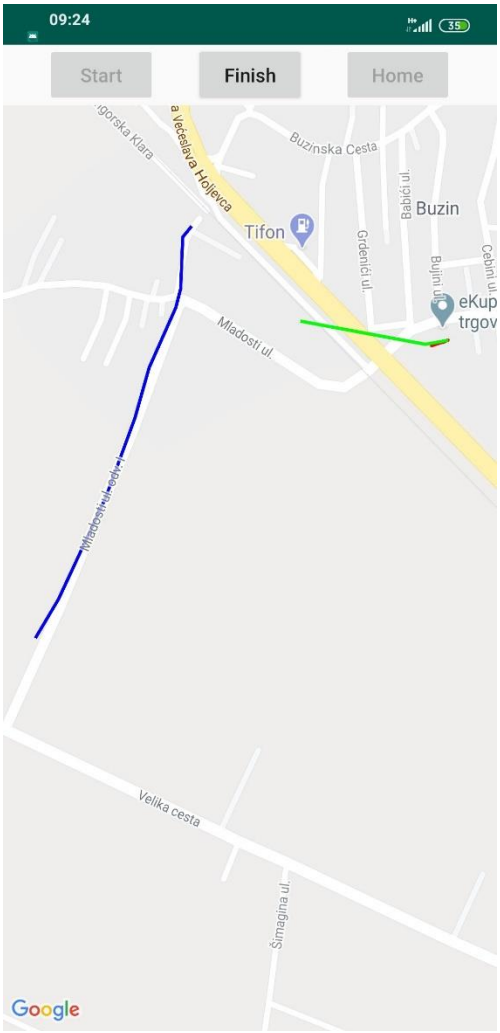
Slika 5.5 Ekran za praćenje rute

Ekran omogućuje korisniku da započne pratiti rutu ucrtanu na prethodnom ekranu. Kako bi započeo praćenje rute, korisnik treba stisnuti gumb s oznakom *Start*. Preporuka je da korisnik pritisne gumb tek onda kad se nalazi na početnoj točki ucrtane rute. Plava točka obrubljena sivom kružnicom, vidljiva na slici (Slika 5.5), označava trenutnu lokaciju te olakšava pri lociranju početne točke ucrtane rute.

Kad praćenje započne, korisnik bi trebao krenuti hodati ucrtanom rutom te će se očitavati i prikazivati lokacija kojom on prolazi. Na mapi će se nalaziti tri rute označene različitim bojama:

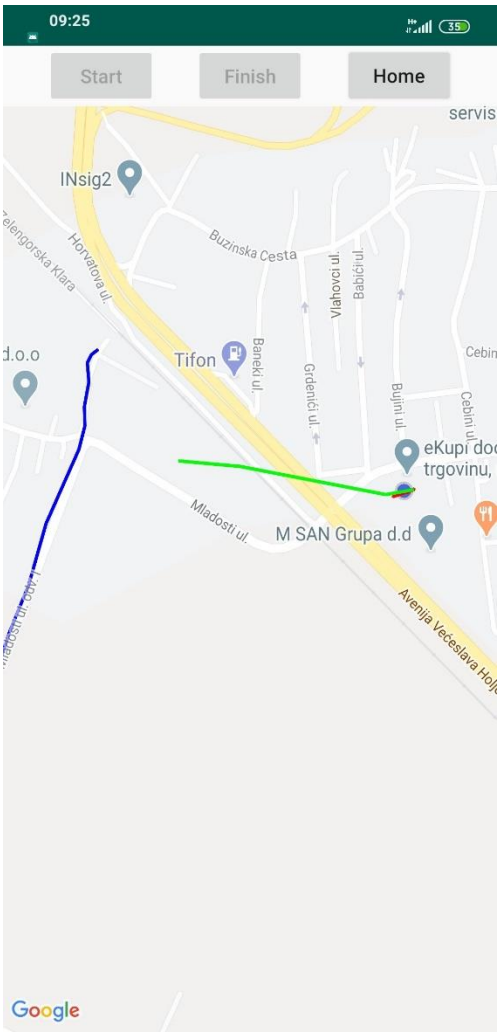
1. Ucrtana ruta, označena plavom bojom
2. Ruta koju očitava GNSS, označena crvenom bojom

3. Ruta koju očitavaju MEMS senzori, označena zelenom bojom



Slika 5.6 Praćenje rute pomoću inercijskih navigacijskih sustava

Kad korisnik dođe do kraja rute, treba pritisnuti gumb s oznakom *Finish* čime će aplikaciji dati do znanja da je došao do kraja rute. Ovdje se može analizirati razlika između INS-a, GNSS-a i stvarne rute koja je trebala biti prijeđena. Klikom na gumb *Home* je moguće vratiti se na početni ekran.



Slika 5.7 Analiza prijeđene rute

5.2. Komunikacija sa senzorima

Komunikacija sa senzorima je ostvarena kroz Android operativni sustav koji izlaže pristup raznim senzorima kako bismo lagano došli do njihovih očitavanja.

`SensorManager` je razred koji nam omogućuje pristup senzorima. On se smatra sistemskim servisom te ga možemo pozvati korištenjem metode `getSystemService` definirane na razredu `Context` koji je bazni razred bilo koje aktivnosti (ekrana). Metodi `getSystemService` trebamo proslijediti naziv senzora kojeg želimo dohvatiti. Popis naziva senzora je dostupan u razredu `Context` putem konstanti tipa `String` što nam omogućuje da, unutar razreda koji nasljeđuje razred `Activity` koji interno nasljeđuje razred `Context`, napišemo sljedeće kako bismo došli do servisa odgovornog za upravljanje senzorima:

```
SensorManager sensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
```

Kako bismo došli do senzora, trebamo pozvati metodu `getDefaultSensor` definiranu unutar `SensorManager` razreda koja prima tip senzora koji želimo dohvatiti. Tip senzora je definiran kao cijeli broj. Tipovi senzora su popisani unutar razreda `Sensor` putem konstanti tipa `int` što nam omogućuje da dohvaćamo senzore na sljedeći način:

```
Sensor accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Kako bismo mogli dobivati očitavanja senzora, moramo implementirati sučelje `SensorEventListener` koje definira metodu `onSensorChanged` koja kao argument prima instancu razreda `SensorEvent` u kojoj se nalaze podaci očitani od strane senzora. Polja na razredu `SensorEvent` potrebna za obradu podataka su:

- *sensor* – instanca senzora koji je inicirao događaj (poziv metode)
- *values* – skup podataka koji su očitani iz danog senzora

Primjer implementacije metode `onSensorChanged`:

```
@Override
public void onSensorChanged(SensorEvent event) {
if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
onAccelerationChanged(event.values[0], event.values[1],
event.values[2]);
}
}

private void onAccelerationChanged(float accelerationX,
float accelerationY, float accelerationZ) {
System.out.print(MessageFormat.format(
"Acceleration: (X: {0}, Y: {1}, Z: {2})",
accelerationX,
accelerationY,
accelerationZ
));
```

```
}
```

Kôd 5.1 Primjer implementacije metode `onSensorChanged`

5.3. Izračun pozicije

Kako bismo došli do pozicije koristeći ugrađene senzore, koristimo metodu dvostruke integracije. Ukoliko pretpostavimo da ćemo krenuti iz stanja mirovanja, možemo doći do trodimenzionalne pozicije koristeći sljedeći isječak kôda:

```
static final float NS2S = 1.0f / 1000000000.0f;
float[] last_values = null;
float[] velocity = null;
float[] position = null;
long last_timestamp = 0;

@Override
public void onSensorChanged(SensorEvent event) {
    if(last_values != null){
        float dt = (event.timestamp - last_timestamp) * NS2S;

        for(int index = 0; index < 3;++index){
            velocity[index] += (event.values[index] +
last_values[index])/2 * dt;
            position[index] += velocity[index] * dt;
        }
    }
    else{
        last_values = new float[3];
        velocity = new float[3];
        position = new float[3];
        velocity[0] = velocity[1] = velocity[2] = 0f;
        position[0] = position[1] = position[2] = 0f;
    }
    System.arraycopy(event.values, 0, last_values, 0, 3);
    last_timestamp = event.timestamp;
}
```

Kôd 5.2 Primjer očitavanja trodimenzionalne pozicije.

Kako bi ovaj isječak radio, potrebno je postaviti slušanje promjena vrijednosti senzora kako je objašnjeno u sekciji 5.2. i to specifično sa linearnim akcelerometrom, do kojeg se može

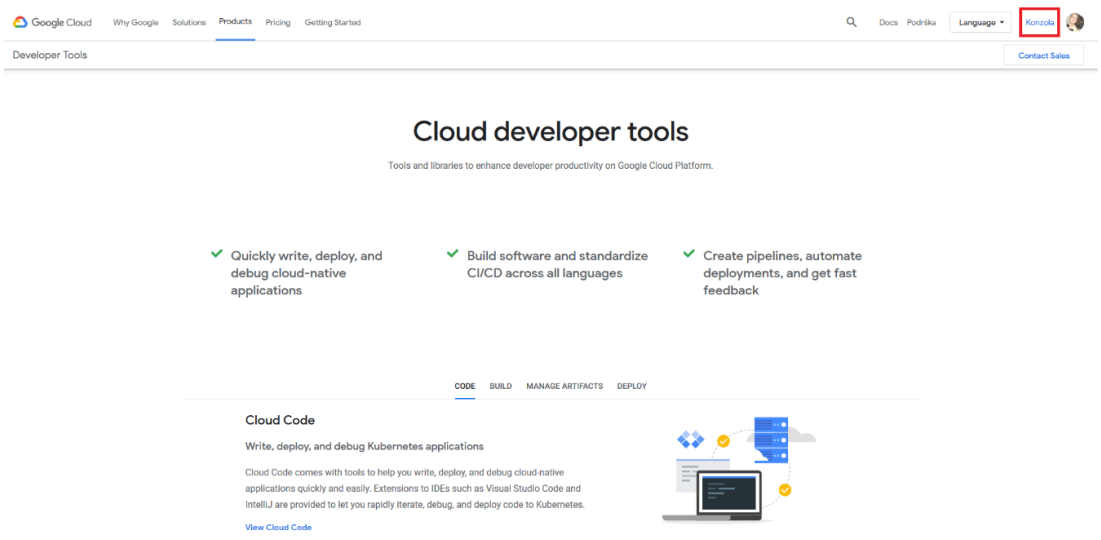
doći tako da se proslijedi konstanta `Sensor.TYPE_LINEAR_ACCELERATION` unutar metode `getDefaultSensor` instance razreda `SensorManager`.

5.4. Komunikacija sa GNSS

Komunikacija sa GNSS je izvedena korištenjem Google-ovog aplikacijskog programskog sučelja (engl. *Application programming interface*, skraćeno API) za Google karte.

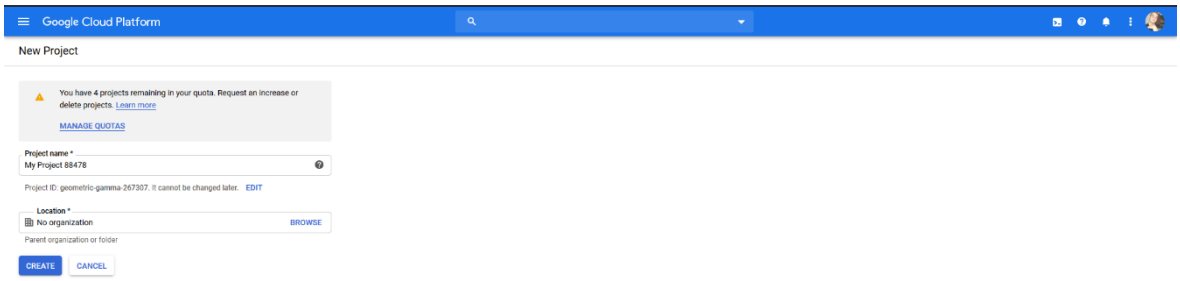
5.4.1. Pristup aplikacijskom programskog sučelju Google karata

Kako bismo mogli pristupiti aplikacijskom programskom sučelju Google karata, prvo moramo napraviti ključ za isto. Prvi korak je prijava u Google razvojne alate [6], što zahtjeva posjedovanje Google računa. Na početnoj stranici Google razvojnih alata je potrebno kliknuti na vezu *Konzola* (Slika 5.8).



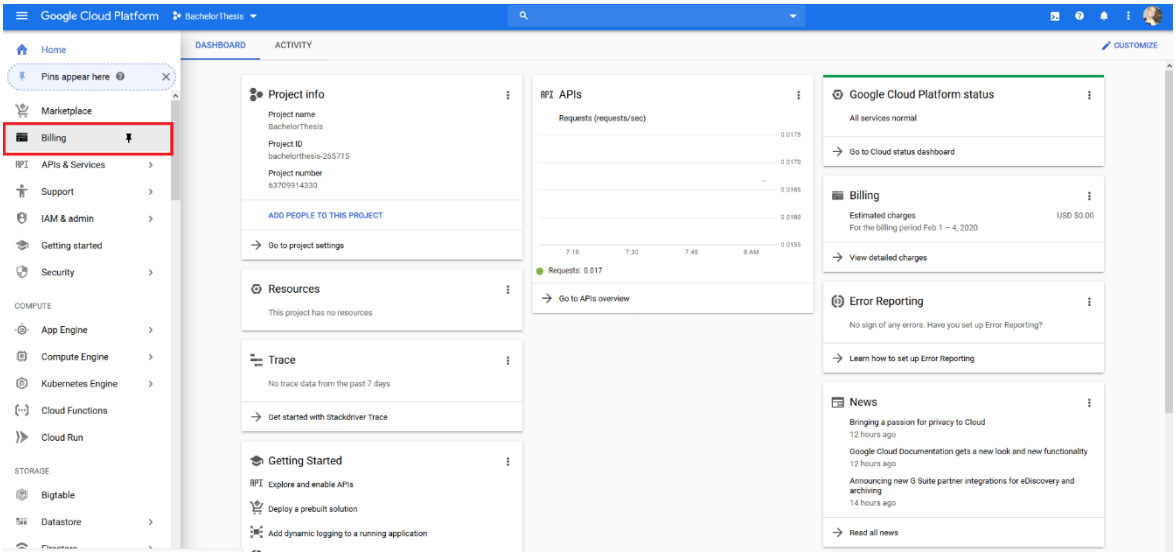
Slika 5.8 Google razvojni alati – odabir konzole

Ako račun nema projekata, na početku će se pojaviti prozor vidljiv na slici (Slika 5.9) u kojemu je potrebno popuniti podatke za izradu novog projekta.



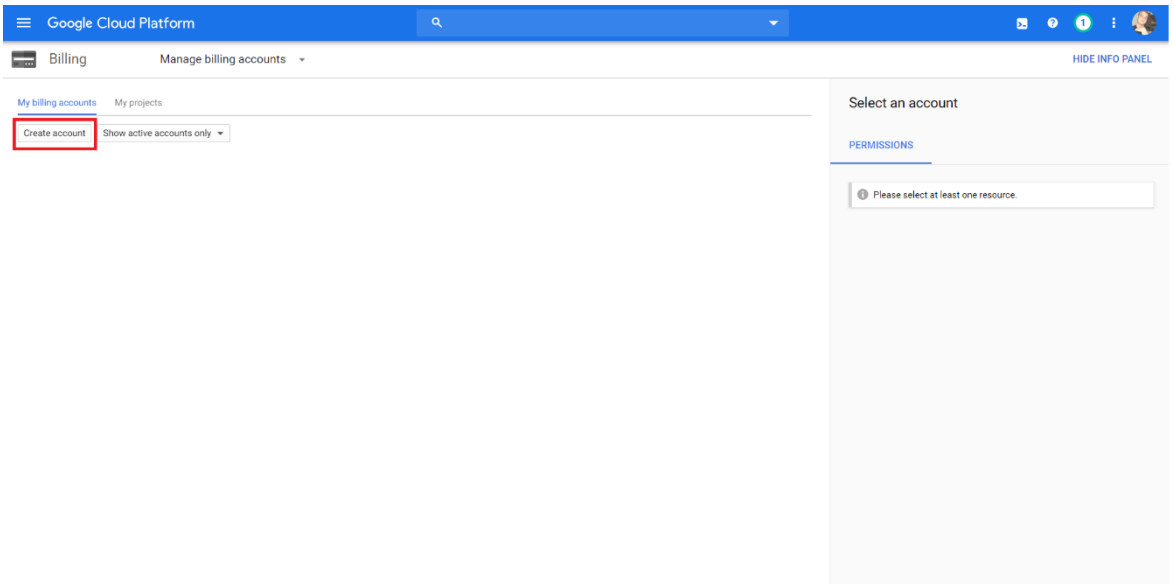
Slika 5.9 Google razvojni alati – kreiranje novog projekta

Za projekt je potrebno dodati plaćanje što se radi klikom na gumb uokviren na slici (Slika 5.10).



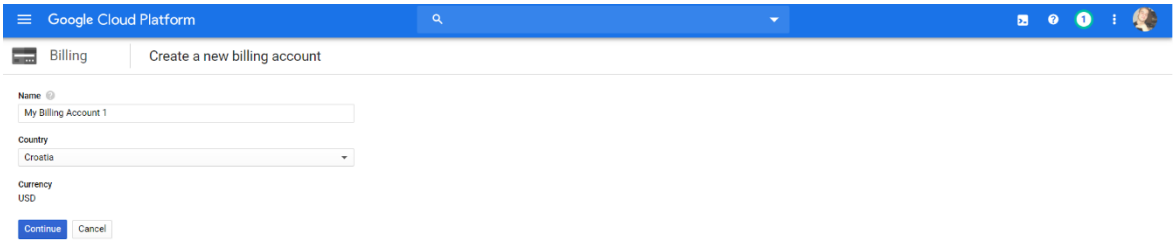
Slika 5.10 Google razvojni alati – odabir izbornika za plaćanje

Nakon toga, pod upravljanjem računima za plaćanje, potrebno je kreirati novi račun za plaćanje klikom na gumb uokviren na slici (Slika 5.11).



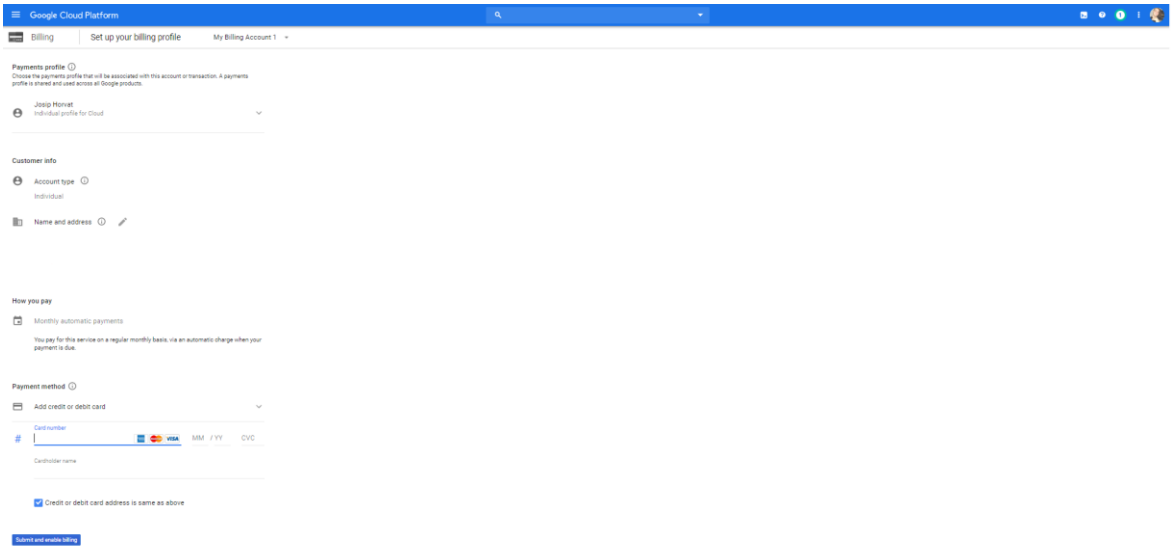
Slika 5.11 Gumb za kreiranje novog računa za plaćanje

Zatim će se pojaviti ekran vidljiv na slici (Slika 5.12) u kojem treba popuniti osnovne podatke o računu za naplatu.



Slika 5.12 Forma za unos osnovnih podataka za račun za naplatu

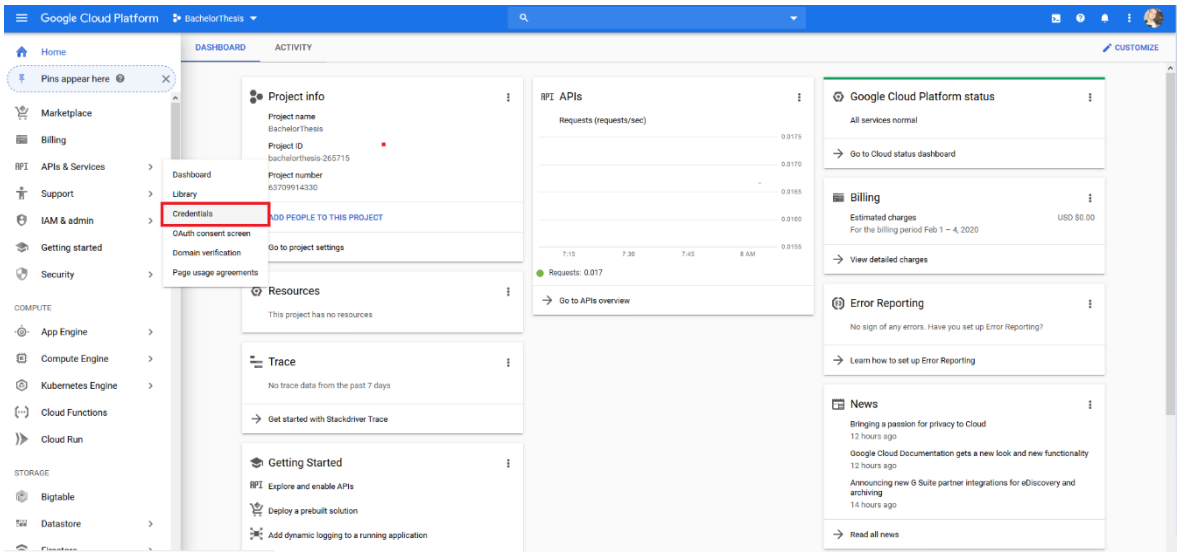
Nakon popunjavanja osnovnih podataka (Slika 5.12), potrebno je postavljanje profila za naplatu popunjavanjem podataka i unosom kartice, što se radi na sljedećem ekranu (Slika 5.13) do kojeg se dolazi klikom na gumb označen *Continue* (Slika 5.12).



Slika 5.13 Forma za unos preostalih podataka veznih za račun za naplatu

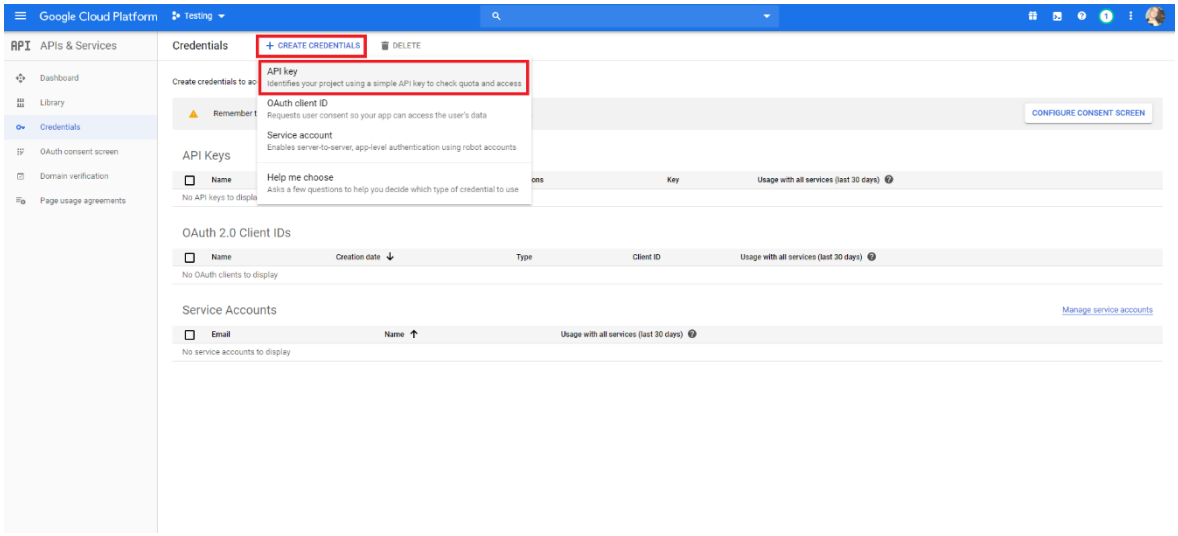
Nakon što su podaci ispunjeni i forma predana, profil za naplatu je napravljen te će biti dostupan za korištenje aplikacijskog programskog sučelja Google karata. Usluge dostupne unutar Google razvojnih alata se plaćaju, uključujući upotrebu aplikacijskog programskog sučelja Google karata. Svaka usluga ima svoj način naplate te se obično naplaćuje mjesečno. Google svakom računu daje svaki mjesec 200 američkih dolara vrijednosti korištenja usluga. Ukoliko se koristi samo aplikacijsko programsko sučelje Google karata do određenog broja zahtjeva dnevno, usluga je efektivno besplatna.

Sljedeći korak je kreiranje identifikacijskog ključa koji će nam omogućiti pristup aplikacijskom programskom sučelju Google karata. Kako bismo to napravili, prvo moramo unutar Google razvojnih alata s lijeve strane odabrati opciju uokvirenu na slici (Slika 5.14).



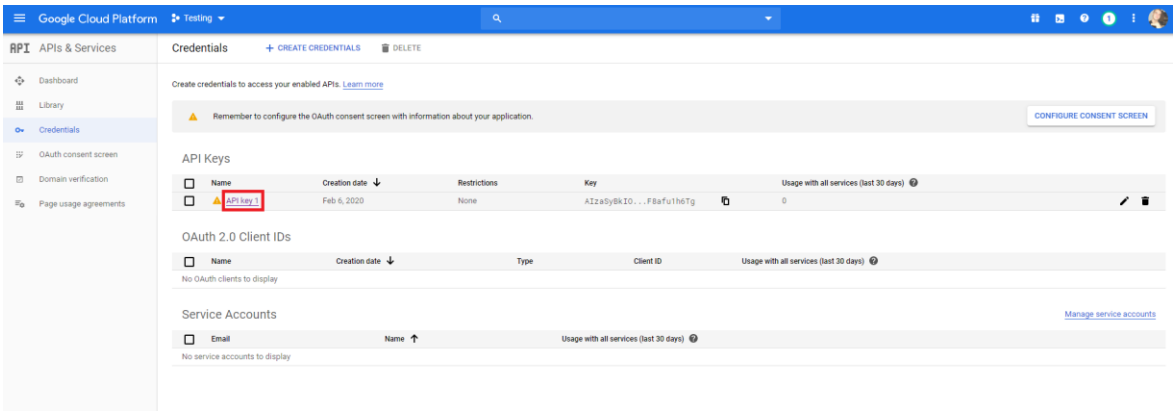
Slika 5.14 Google razvojni alati – odabir izbornika za vjerodajnice

Nakon toga, trebamo kreirati prethodno spomenuti ključ tako da odaberemo opciju uokvirenu na slici ().



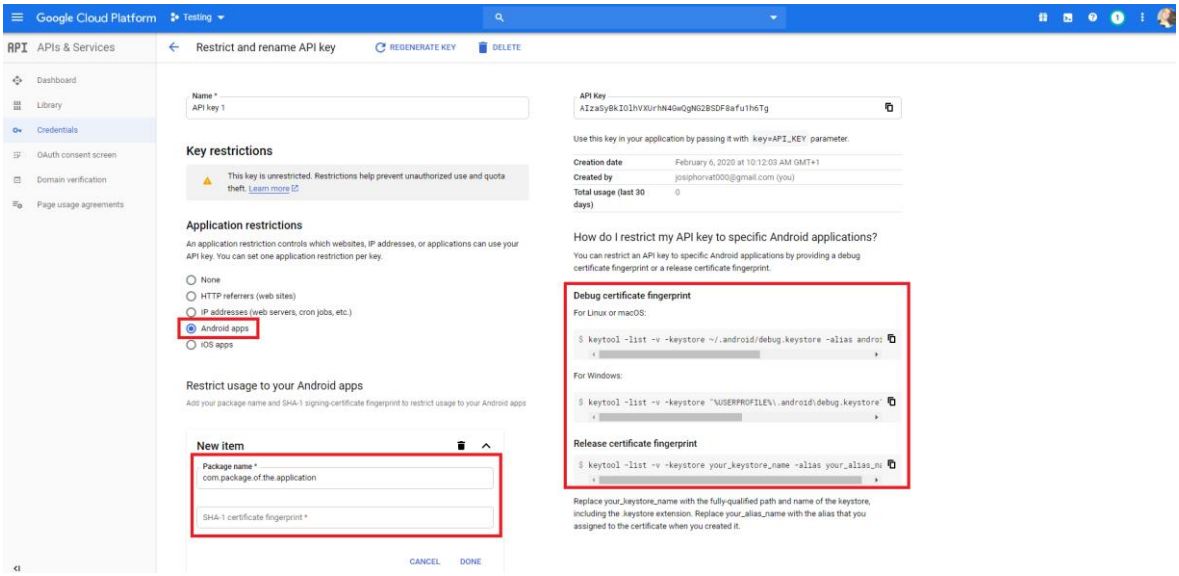
Slika 5.15 Izrada vjerodajnica za aplikacijsko programsko sučelje Google karte

Time se kreirao ključ. Sljedeći korak je restrikcija istog. Prvo trebamo kliknuti na naziv ključa (Slika 5.16).



Slika 5.16 Odabir identifikacijskog ključa

To će nas dovesti na ekran vidljiv na slici Slika 5.17. Ovdje (opcionalno) ograničavamo mogućnosti pristupa odabranog identifikacijskog ključa. Preporučeno je napraviti korake uokvirene na slici Slika 5.17. Prvo, odabrati tip aplikacije, u ovom slučaju odabiremo Android aplikaciju kao opciju. Nakon toga, pod *Package name* stavljamo naziv paketa u kojem se nalazi naša aplikacija. *SHA-1 certificate fingerprint* dohvaćamo koristeći komande dostupne s desne strane. Potrebno je kopirati (engl. *copy*) komandu ovisno o operativnom sustavu u kojem se aplikacija razvija i ovisno o vrsti distribucije aplikacije. Nakon što je kopirana, zalijepimo (engl. *paste*) je u naredbeni redak (na Windows operativnom sustavu) ili u terminal (u Linux ili macOS operativnom sustavu) te pokrenemo. Dobivenu vrijednost zalijepimo kao otisak prsta generiran pomoću SHA1 algoritma za sažimanje.



Slika 5.17 Ograničavanje pristupa identifikacijskom ključu

5.4.2. Postavljanje ovisnosti za aplikacijsko programsko sučelje Google karata unutar Android projekta

Nakon što je dodano plaćanje i ključ za pristup aplikacijskom programskom sučelju Google karata, potrebno je kopirati ključ i zalijepiti ga kao niz znakova unutar datoteke *google_maps_api.xml* pod ključem s nazivom *google_maps_key*:

```
<resources>
    <string name="google_maps_key"
templateMergeStrategy="preserve"
translatable="false">API_KEY_HERE</string>
</resources>
```

Nakon toga, u datoteku *AndroidManifest.xml* potrebno je dodati sljedeću stavku:

```
<meta-data
android:name="com.google.android.geo.API_KEY"
android:value="@string/google_maps_key" />
```

Kako bi bio omogućen pristup razredima i sučeljima potrebnima za rad s Google kartama, u datoteku *build.gradle,* unutar sekcije *dependencies* je potrebno dodati sljedeće dvije ovisnosti:

```
implementation 'com.google.android.gms:play-services-
maps:16.1.0'
implementation 'com.google.android.gms:play-services-
location:17.0.0'
```

5.4.3. Slušanje promjena lokacije

Zadnji korak je postavljanje usluge kojom ćemo raditi upite prema aplikacijskom programskom sučelju Google karata. Razredi koji su na potrebni za to su sljedeći:

- *FusedLocationProviderClient* – Pruža uslugu dohvaćanja lokacije
- *LocationCallback* – definira što se treba dogoditi kad se promijeni lokacija
- *LocationRequest* – definira koliko često bi se trebala dohvaćati lokacija

Pri postavljanju navedenih razreda, prvo je potrebno inicijalizirati instancu razreda *FusedLocationProviderClient*:

```
FusedLocationProviderClient locationProvider =
LocationServices.getFusedLocationProviderClient(this);
```

Nakon toga, trebamo definirati zahtjev za lokaciju, npr.:

```
LocationRequest locationRequest = new LocationRequest()
.setInterval(5000)
.setFastestInterval(10000)
.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

Sljedeće što trebamo je definirati kako ćemo procesirati dobivenu lokaciju, što radimo kroz instancu razreda `LocationCallback`, npr:

```
LocationCallback locationCallback = new LocationCallback() {
      @Override
      public void onLocationResult(LocationResult
locationResult) {
            super.onLocationResult(locationResult);
            Location lastLocation =
locationResult.getLastLocation();
            System.out.println("Last location: " +
lastLocation);
      }
};
```

Zadnje što trebamo napraviti je pokrenuti slušanje promjena lokacije, npr.:

```
locationProvider.requestLocationUpdates(locationRequest,
locationCallback, Looper.myLooper());
```

6. Analiza odmaka i validacija hipoteze

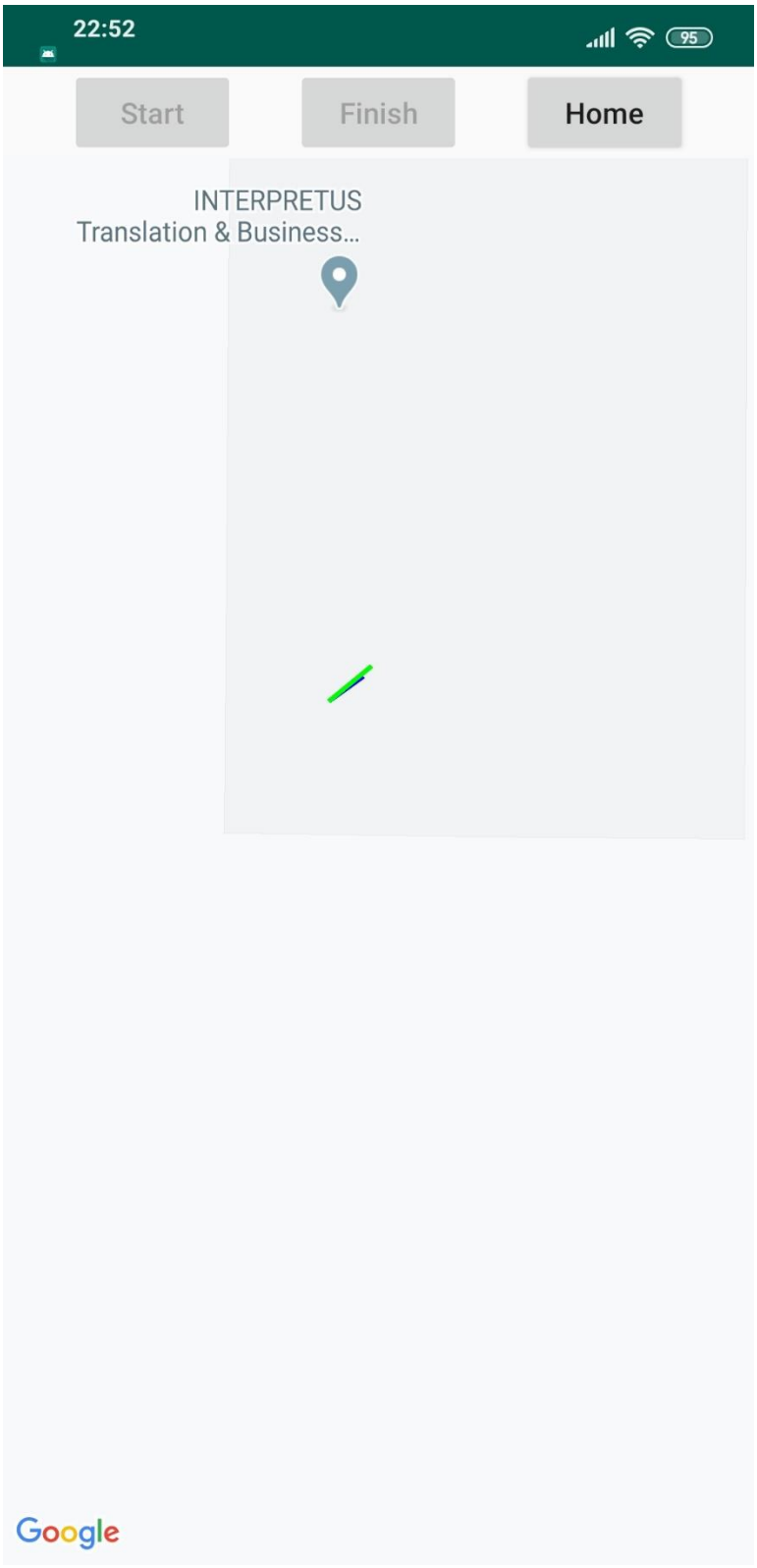
6.1. Analiza

Za analizu točnosti očitavanja senzora možemo usporediti rezultate sljedećih ruta:

- Ruta koju smo odredili da ćemo proći njome
- Ruta koju je očitao GNSS
- Ruta koju su očitali MEMS senzori

Kako bismo pokrili mjerenja odmaka za razne situacije, trebamo pokriti slučajeve navedene u tablici Tablica 2.1.

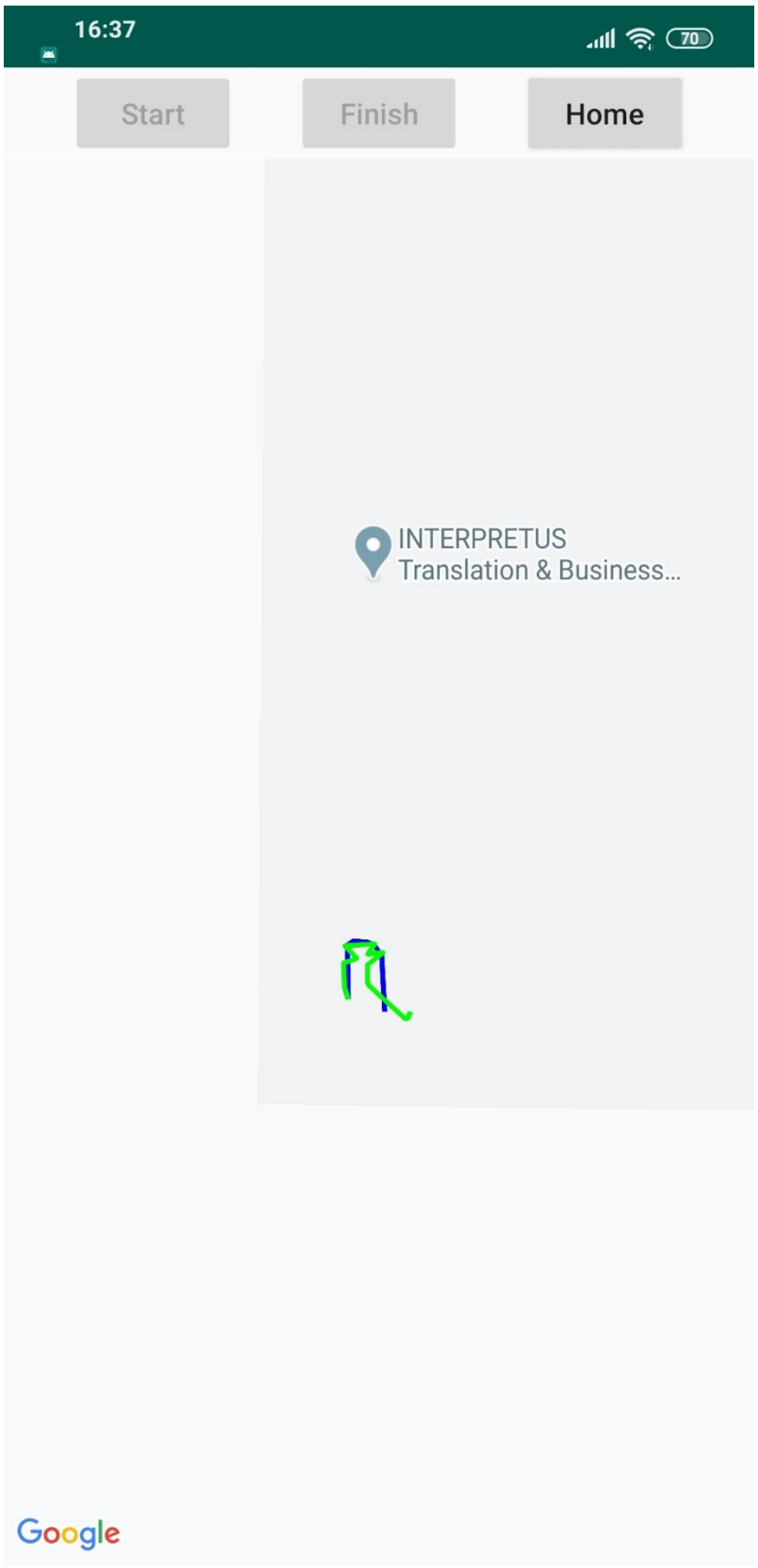
Prvo što ćemo mjeriti je prijelaz jednog metra bez skretanja. Rezultat je vidljiv na slici (Slika 6.1). Iako se radi samo o jednom metru, odmah se može uočiti da se već stvorila određena pogreška, kako smo i očekivali s obzirom na analizu odmaka napravljenu u poglavlju 4.2 Odmak.



Slika 6.1 Mjerenje – prijelaz jednog metra bez skretanja

Sljedeće što je bitno izmjeriti je preciznost orijentacije. Kako bismo to izmjerili, odredimo rutu u obliku luka. Rezultat je vidljiv na slici Slika 6.2. Odstupanja su uglavnom prisutna zbog pogreške koja se pojavljuje u žiroskopu i magnetometru, ali osim toga se pojavljuje

pogreška pri orijentiranju mobitela jer mobitel nije pomaknut koristeći određeno pomoćno sredstvo već ju određena osoba nosi u ruci i prelazi definirani put.



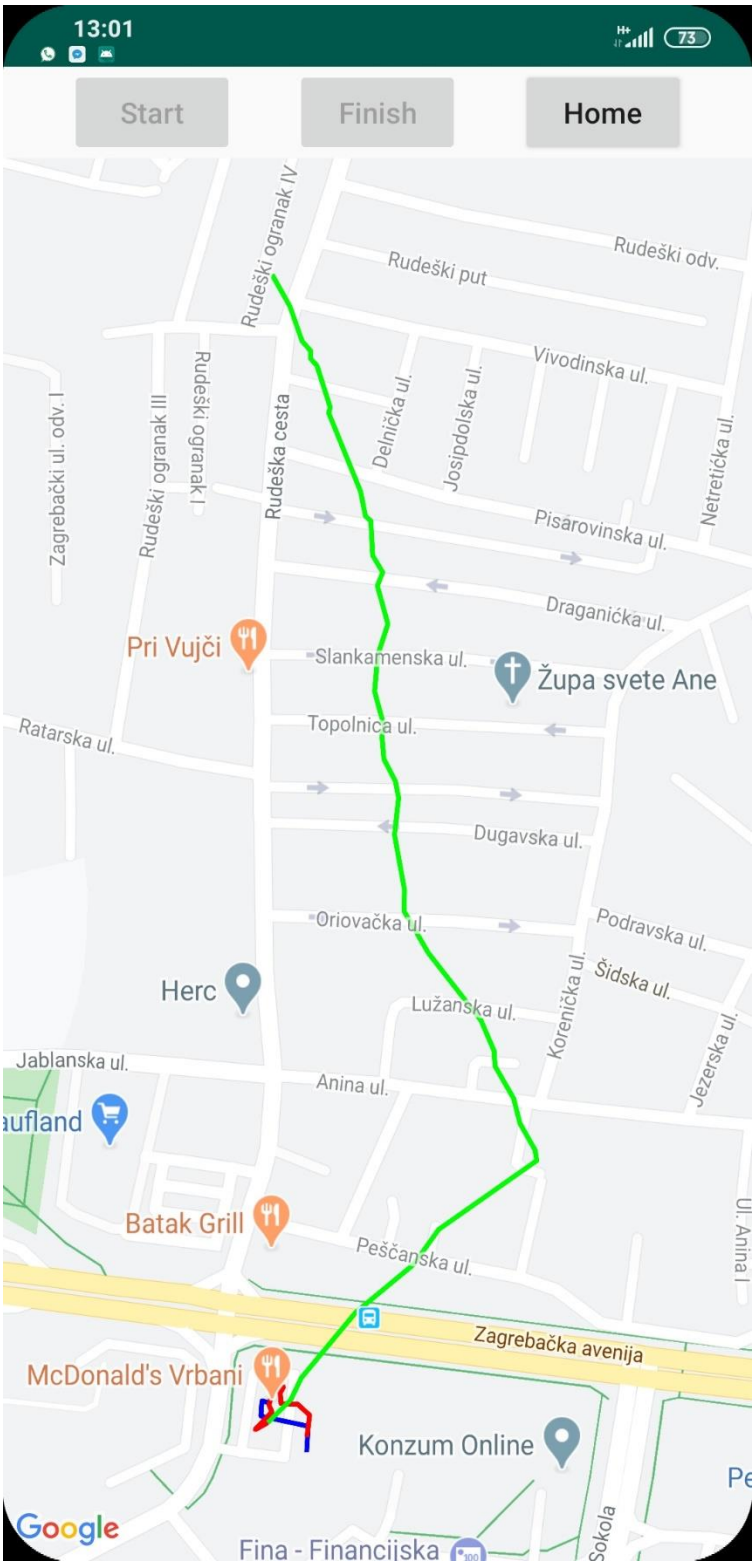
Slika 6.2 Mjerenje – prijelaz rute u obliku luka

Nakon što smo analizirali pogrešku u orijentaciji, trebamo malo detaljnije pogledati što se dešava na dužim relacijama. U mjerenju, čiji rezultat se može vidjeti na slici Slika 6.3, se prelazi nešto duži put od jednog metra (oko 7 metara) te se može vidjeti koliko je veći odmak čak i na toj relativno maloj distanci.

Slika 6.3 Mjerenje – prijelaz nešto duže rute bez skretanja

Sljedeće što je potrebno analizirati je usporedba navigacije koristeći GNSS i koristeći MEMS senzore. Na slici Slika 6.4 je moguće vidjeti koliko se greška može akumulirati kod navigacije koristeći MEMS senzore. Osim što se dogodila ekstremno velika greška u udaljenosti, orijentacija je također potpuno neispravna, do čega može doći ukoliko se

mobitel ne okreće u smjeru u kojem se osoba kreće već je mobitel u nasumičnoj poziciji koja nema veze s definiranom rutom. GNSS je, prema očekivanom, neispravno pokazivao rutu pri kretanju u zatvorenom prostoru, no čim je uređaj napustio zatvoreni prostor, GNSS je vrlo precizno očitavao rutu.



Slika 6.4 Mjerenje – kompleksna ruta

Napokon, ukoliko pokušamo na dugoj relaciji koristiti MEMS senzore kao alat za navigaciju, greška će se akumulirati do razine gdje nema smisla uopće je uzimati u obzir. GNSS je ponovno očitavao rutu ispravno na otvorenom, no čim je uređaj ušao u zatvoreni prostor, lokacija se očitavala neispravno.

Slika 6.5 Mjerenje – kompleksna duga ruta

6.2. Prijedlog mogućih metoda smanjenja utjecaja odmaka

S obzirom na veliki odmak koji stvaraju senzori, smišljeno je nekoliko metoda kojima ga možemo smanjiti [4]:

- Niskopropusni filter (engl. *Low pass filter*) – Uzima signal dobiven od senzora te ga ispravlja, ali kroz vrijeme što znači da se povećava vrijeme između svakog novog očitavanja.
- Visokopropusni filter (engl. *High pass filter*) – odašilje signale s frekvencijom višom od određene frekvencije isključivanja i prigušuje signale s frekvencijama nižim od granične frekvencije.
- Kalman filter – Izbacuje vrijednosti koje nisu specifične za model koji se koristi. Primjer česte primjene takvog filtera je kretanje automobila. Za automobil se zna da ne može ubrzati prema gore pa se te vrijednosti odbacuju.
- Algoritam pedometra – algoritam koji se temelji na principu hodanja.

7. Zaključak

Ovaj rad omogućuje bilo kome tko je zainteresiran u područje inercijskih navigacijskih sustava da prouči što su oni točno, kako nam mogu pomoći i jesu li doista praktični za uporabu. Analiza koja se vršila u ovom radu je bazirana na MEMS senzorima koji se nalaze u pametnim telefonima te će možda potaknuti još nekoga da krene proučavati tematiku u detalje. Što tehnologija više napreduje, to će se više stvari moći napraviti koristeći sustave poput INS-a. Najbitnija stvar koja se može izvući iz ovog projekta je da se pomoću kombinacije nekoliko senzora može napraviti mnogo vrlo zanimljivih projekata koji će pomoći ljudima u svakodnevnom životu ili da zabavno provedu svoje vrijeme.

Rezultati trenutno odstupaju podosta od stvarnih vrijednosti, no primjenom određenih metoda se očitane vrijednosti mogu pretvoriti u takve koje se koriste u razne svrhe. Navigacija u zatvorenom prostoru je sve traženija te je cilj omogućiti korisnicima razinu optimiziranosti navigacije u zatvorenom prostoru kao što je ima navigacija vani koristeći GNSS. INS navigacija koristeći MEMS senzore je dobar korak u tom smjeru, no još nije dovoljno dobro rješenje primjenjivo na pametne telefone koji su danas uobičajeni uređaji za navigaciju. Ukoliko se u budućnosti razviju optimiziraniji senzori koji neće toliko odstupati, INS navigacija pomoću MEMS senzora u zatvorenom prostoru bi mogla zaživjeti. Dok se to ne dogodi, navigacija u zatvorenom će morati omogućavati precizna očitanja koristeći druge metode.

U budućnosti bi se moglo proširiti ono što je napravljeno kroz ovaj rad te vidjeti hoće li tada biti odmak senzora smanjen i čini li ih to korisnima za izradu sustava koji nam pomažu u svakodnevnom životu.

Popis kratica

API	*Application programming interface*	Aplikacijsko programsko sučelje
SHA	*Secure Hash Algorithm*	Sigurni algoritam sažimanja
GPS	*Global Positioning System*	Globalni položajni sustav
GNSS	*Global Navigation Satellite System*	Globalni navigacijski satelitski sustav
INS	*Inertial Navigation System*	Inercijski navigacijski sustav
RFID	*Radio-Frequency Identification*	Identifikacija radio frekvencijom
MEMS	*Microelectromechanical Systems*	Mikroelektromehanički sustavi

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] P. DABOVE, G. GHINAMO AND A. M. LINGUA: *Inertial sensors for smartphones navigation*, SpringerPlus 2015, 30 December 2015, https://springerplus.springeropen.com/articles/10.1186/s40064-015-1572-8.

[2] OLIVER J. WOODMAN: *An introduction to inertial navigation*, August 2007, University of Cambridge, https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf.

[3] MICHAL HOLCÍK: *Indoor Navigation for Android*, Masaryk University Faculty of Informatics, Master's thesis, Brno, spring 2012, https://is.muni.cz/th/o4tfv/holcik_thesis.pdf.

[4] DAVID SACHS: Sensor Fusion on Android Devices: A Revolution in Motion Processing, https://www.youtube.com/watch?v=C7JQ7Rpwn2k, veljača 2020.

[5] Eletronicsforu, What are MEMS Sensors? What are the Types Available in the Market?, https://www.electronicsforu.com/resources/learn-electronics/mems-sensors-available-market, veljača 2020.

[6] Android Developers: Get an API Key, https://developers.google.com/maps/documentation/android-sdk/get-api-key, veljača 2020.

[7] Wikipedia: Inertial navigation system, https://en.wikipedia.org/wiki/Inertial_navigation_system, veljača 2020.

[8] Skybrary: Inertial navigation system, https://www.skybrary.aero/index.php/Inertial_Navigation_System_(INS), veljača 2020.

Prilog

Završni rad može imati priloge, ali se oni ne prilažu uz pisanu verziju završnog rada već se mogu priložiti na završnom ispitu ukoliko povjerenstvo na završnom ispitu tako odluči. Važno je čuvati svu poratnu dokumentaciju koja je nastala pri izradi završnog rada.

S unutarnje strane na zadnjim koricama originala, kao i svake kopije završnog rada, pričvršćuje se CD s kompletnim završnim radom u izvornom formatu (npr. .doc) i .pdf formatu sa svom popratnom dokumentacijom i programima. Pri čemu je obvezno da na tom CD- u postoji i dokument koji opisuje kako se rezultat njegova diplomskog rada (softver ili hardver) koristi (ili kako se npr. izvode mjerenja koja je opisao u radu). Ako se radi o softveru nužno je opisati i kako se programska podrška instalira.

ALGEBRA
VISOKO UČILIŠTE

Mjerenje odmaka pozicije u MEMS inercijskoj navigaciji temeljenoj na integriranju

Pristupnik: Josip Horvat, 0321006163

Mentor: prof. dr. sc. Goran Đambić