Primjena senzora u razvoju aplikacija za mobilne uređaje

Cerovac, Matija

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Preddiplomski studij informatike

Matija Cerovac

Primjena senzora u razvoju aplikacija za mobilne uređaje

Završni rad

Mentor: izv. prof. dr. sc. Božidar Kovačić

Rijeka, rujan 2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, datum

Zadatak za završni rad

Pristupnik: Matija Cerovac

Naziv završnog rada: Primjena senzora u razvoju aplikacija za mobilne uređaje

Naziv završnog rada na eng. jeziku: Application of sensors in the development of mobile applications

Sadržaj zadatka:

Potrebno je objasniti rad senzora u mobilnim aplikacijama za tri glavne vrste senzora: senzore pokreta, senzore okoliša te senzore položaja. Za svaki senzor je potrebno objasniti njegovu funkcionalnost, način rada te navesti primjer primjene svakih od senzora u aplikacijama. Zatim slijedi opis prototipova jednostavnih aplikacija koje pokazuje rad triju glavnih vrsta senzora.

Mentor

dr.sc. Božidar Kovačić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 05.05.2021.

(potpis pristupnika)

Cilj rada

U ovom završnom radu biti će objašnjen i demonstriran rad senzora u mobilnim aplikacijama. Senzori su podijeljeni u tri glavne vrste: senzore pokreta, senzore okoliša te senzore položaja. Za svaki senzor je objašnjena njegova funkcionalnost, način rada te su također dani primjer primjene svakih od senzora u aplikacijma. Naposlijetku, napravljeni su prototipi jednostavnih aplikacija koje pokazuje rad tri glavnih vrsta senzora.

Ključne riječi: Senzori, Mobilne Aplikacije, Senzori Pokreta, Senzori Okoliša, Senzori Položaja, Android Studio

Sadržaj

1. Uvod 4
2. Senzori u aplikacijama 5
2.1. Podjela i vrsta senzora 5
2.1.1. Senzori pokreta 5
2.1.2. Senzori okoliša 6
2.1.3. Senzori položaja 7
2.2. Komuninkacija aplikacija i senzora na nivou operativnog sustava Android 8
2.3. Primjena senzora u različitim aplikacijama 9
2.3.1. Primjeri upotrebe senzora pokreta 9
2.3.2. Primjeri upotrebe senzora okoliša 10
2.3.3. Primjeri upotrebe senzora položaja 11
3. Prototip aplikacija sa senzorima 14
3.1. Aplikacija očitanja pokreta i temperature 14
3.1.1. Očitanje temperature 14
3.1.2. Očitanje pokreta 17
3.2. Aplikacija očitanja položaja (kompas) 23
4. Zaključak 28
5. Popis slika 29
6. Literatura 30

1. Uvod

Mobilna aplikacija ili skraćeno app je računalni program ili softverska aplikacija dizajnirana za rad na mobilnom uređaju kao što je pametni telefon, tablet ili sat. Mobilne aplikacije često su u suprotnosti s desktop aplikacijama koje su dizajnirane za rad na stolnim računalima i web aplikacijama koje se izvode u mobilnim web preglednicima, a ne izravno na mobilnom uređaju.

Aplikacije su izvorno bile namijenjene za pomoć produktivnosti kao što su e-mail, kalendar i baze podataka kontakata, ali javna potražnja za aplikacijama uzrokovala je brzo širenje na druga područja kao što su mobilne igre, tvornička automatizacija, GPS i usluge temeljene na lokaciji, praćenje narudžbi i karte kupnje, tako da su sada dostupni milijuni aplikacija. Upravo zbog toga danas mnogi ljudi koriste različite vrste apliakcija u svakodnevnom životu te su one znantno olakšale živote ljudi diljem svijeta. Iz toga se može vidjeti potreba za aplikacijama koje će očitavati informacije u okolini uređaja te ih pretvarati u funkcionalnosti za upotrebu od strane korisnika. Za to su potrebni senzori. Senzor može biti uređaj, modul, stroj ili podsustav koji otkriva događaje ili promjene u svojoj okolini i šalje informacije drugoj elektronici, često računalnom procesoru te se oni gotovo uvijek uvijek koriste s drugom elektronikom. Sam Android operativni sustav ima izvrsnu podršku za raznolike senzore te se oni mogu pristupiti preko okvira senzora koji dolazi u sklopu hardverskog paketa u Android-u. Za potrebe ovog završnog rada biti će korištena Android platforma te njegov okvir senzora.

Okvir senzora za Android omogućuje pristup mnogim vrstama senzora. Neki od tih senzora temelje se na hardveru, a neki na softveru. Senzori temeljeni na hardveru fizičke su komponente ugrađene u slušalicu ili tablet uređaj. Svoje podatke dobivaju izravnim mjerenjem specifičnih svojstava okoliša, poput ubrzanja, jakosti geomagnetskog polja ili kutne promjene. Senzori temeljeni na softveru nisu fizički uređaji, iako oponašaju senzore temeljene na hardveru. Softverski bazirani senzori izvode svoje podatke iz jednog ili više hardverski baziranih senzora i ponekad se nazivaju virtualni senzori. Senzor linearnog ubrzanja i senzor gravitacije primjeri su softverski baziranih senzora.

Jedna zanimljivost je što samo nekoliko uređaja sa sustavom Android ima sve vrste senzora. Na primjer, većina mobilnih uređaja i tableta ima akcelerometar i magnetometar, ali manje uređaja ima barometre ili termometre. Također, uređaj može imati više od jednog senzora određene vrste. Na primjer, uređaj može imati dva senzora gravitacije, od kojih svaki ima drugačiji domet i primjenu.

2. Senzori u aplikacijama

Senzori se u kontekstu mobilnih uređaja koriste u brojnim mobilnim aplikacijama sa različitim namjenama. Neki senzori služe samo da bi uljepšali rad samog android uređaja, primjerice okretanje ekrana ocisno o rotaciji uređaja, drugi postoje radi povećanja zabave poput video igara, dok neki postoje kako bi pomogli korinsiku u svakodnevnom životu poput mapi ili brojača koraka. Stoga možemo vidjeti da postoje mnoge vrste senzora koji se koriste u različitim vrstama aplikacija, dok se čak i za jednu vrstu aplikaciju mogu koristiti različite vrste senzora.

2.1. Podjela i vrsta senzora

Platforma Android podržava tri široke kategorije senzora:

- Senzori pokreta
- Senzori okoliša
- Senzori položaja

Svaka od ove tri kategorije pod sobom ima veći broj senzora koje zajedno kategorizira slična funkcionalnost, ali se njihova implementacija razlikuje.

2.1.1. Senzori pokreta

Senzori pokreta mjere sile ubrzanja i rotacijske sile duž tri osi. Ova kategorija uključuje senzore poput akcelerometara, gravitacijskih senzora, žiroskopa i rotacijskih vektorskih senzora. Platforma Android nudi nekoliko senzora koji vam omogućuju praćenje kretanja uređaja. Same arhitekture senzora razlikuju se ovisno o vrsti senzora. Primjerice senzori gravitacije, linearnog ubrzanja, vektora rotacije, značajnog gibanja, brojača koraka i detektora koraka temelje se na hardveru ili softveru, dok se senzori akcelerometra i žiroskopa uvijek temelje na hardveru.

Naziv senzora	**Tip senzora**	**Opis**	**Uobičajena primjena**
ACCELEROMETER SENSOR	Hardverski	Mjeri silu ubrzanja u m/s2 koja se primjenjuje na uređaj na sve tri fizičke osi (X, Y i Z), uključujući silu gravitacije.	Detekcija pokreta (tresenje, naginjanje, itd.).
GRAVITY SENSOR	Hardverski ili Softverski	Mjeri silu gravitacije u m/s2 koja djeluje na uređaj na sve tri fizičke osi (X, Y i Z).	Detekcija pokreta (tresenje, naginjanje, itd.).
GYROSCOPE SENSOR	Hardverski	Mjeri brzinu rotacije uređaja u rad/s oko svake od tri fizičke osi (X, Y i Z).	Detekcija rotacije (vrtnja, okret, itd.).
LINEAR ACCELERATION SENSOR	Hardverski ili Softverski	Mjeri silu ubrzanja u m/s2 koja se primjenjuje na uređaj na sve tri fizičke osi (X, Y i Z), isključujući silu gravitacije.	Praćenje ubrzanja duž jedne osi.

ROTATION VECTOR SENSOR	Hardverski ili Softverski	Mjeri orijentaciju uređaja pružajući tri elementa vektora rotacije uređaja.	Detekcija pokreta i detekcija rotacije.
SIGNIFICANT MOTION SENSOR	Hardverski	Mjeri naglu promjenu u krenji uređaja.	Mjerenje i detekcija promjena brzina.
STEP COUNTER SENSOR	Hardverski	Broj koraka koje je korisnik napravio od zadnjeg ponovnog pokretanja dok je senzor bio aktiviran.	Brojanje koraka.
STEP DETECTOR SENSOR	Hardverski	Prepoznaje kada korinik napravi jedan korak u pokretu.	Brojanje koraka.

Figure 1 Tablica senzora pokreta

Većina današnjih uređaja sa sustavom Android ima akcelerometar, a mnogi sada uključuju i žiroskop. Dostupnost senzora koji se temelje na softveru više je varijabilna jer se oni često oslanjaju na jedan ili više hardverskih senzora za izvođenje svojih podataka. Ovisno o uređaju, ovi softverski senzori mogu izvući svoje podatke iz akcelerometra i magnetometra ili iz žiroskopa.

Senzori pokreta korisni su za praćenje kretanja uređaja, poput naginjanja, trešnje, rotacije ili njihanja. Kretanje je obično odraz izravnog korisničkog unosa (na primjer, korisnik koji upravlja automobilom u igri ili korisnik koji kontrolira loptu u igri), ali može biti i odraz fizičkog okruženja u kojem se uređaj nalazi (na primjer, kretanje s vama dok vozite svoj automobil). U prvom slučaju, nadzire se kretanje u odnosu na referentni okvir uređaja ili referentni okvir aplikacije, u drugom slučaju pratite kretanje u odnosu na referentni okvir svijeta. Sami senzori kretanja obično se ne koriste za nadzor položaja uređaja, ali se mogu koristiti s drugim senzorima, kao što je senzor geomagnetskog polja, za određivanje položaja uređaja u odnosu na referentni okvir svijeta oko uređaja.

Svi senzori pokreta vraćaju višedimenzionalne nizove vrijednosti senzora za svaki „SensorEvent“. Na primjer, tijekom događaja jednog senzora akcelerometar vraća podatke o sili ubrzanja za tri koordinatne osi, a žiroskop vraća podatke o brzini rotacije za tri koordinatne osi. Ove vrijednosti podataka vraćaju se u nizu (vrijednosti) zajedno s drugim parametrima „SensorEvent“.

2.1.2. Senzori okoliša

Ovi senzori mjere različite parametre okoliša, kao što su temperatura i tlak okolnog zraka, osvjetljenje i vlažnost. Ova kategorija uključuje barometre, fotometre i termometre.

Platforma Android nudi četiri senzora koji omogućavaju praćenje različitih svojstava okoliša. Ovi senzori mogu se koristiti za praćenje relativne vlažnosti okoline, osvijetljenosti, tlaka okoline i temperature okoline u blizini uređaja. Prije je postojao senzor naziva „TYPE_TEMPERATURE“, no nakon što je izašao API Level 14 njega je zamijenio senzor „TYPE_AMBIENT_TEMPERATURE“ koji nešto robusniji i efikasniji od svog prethodnika.

Naziv senzora	**Tip senzora**	**Opis**	**Uobičajena primjena**

TAMBIENT TEMPERATURE SENSOR	Hardverski	Mjeri sobnu temperaturu u stupnjevima Celzija (°C). Pogledajte napomenu u nastavku.	Praćenje temperature zraka.
LIGHT SENSOR	Hardverski	Mjeri razinu ambijentalnog osvjetljenja (osvjetljenje) u lx-ima.	Kontrola svjetline zaslona.
PRESSURE SENSOR	Hardverski	Mjeri tlak okolnog zraka u hPa ili mbar.	Praćenje promjena tlaka zraka.
RELATIVE HUMIDITY SENSOR	Hardverski	Mjeri relativnu vlažnost okoline u postocima (%).	Praćenje rosišta, apsolutne i relativne vlažnosti.

Figure 2 Tablica senzora okoliša

Svih četiri senzora okoliša temelje se na hardveru i dostupni su samo ako ih je proizvođač uređaja ugradio u uređaj. S iznimkom senzora za svjetlo, koji većina proizvođača uređaja koristi za kontrolu svjetline zaslona, senzori za okoliš nisu uvijek dostupni na uređajima. Zbog toga je posebno važno da se tijekom izvođenja provjeri postoji li senzor okoline prije nego što se pokuša prikupiti podatke s njega. Za razliku od većine senzora kretanja i senzora položaja, koji vraćaju višedimenzionalni niz vrijednosti senzora za svaki SensorEvent, senzori okoline vraćaju jednu vrijednost senzora za svaki podatkovni događaj. Na primjer, temperatura u °C ili tlak u hPa. Također, za razliku od senzora pokreta i senzora položaja, koji često zahtijevaju visokopropusno ili niskopropusno filtriranje, senzori okoline obično ne zahtijevaju nikakvo filtriranje podataka ili ikakvu obradu podataka.

2.1.3. Senzori položaja

Ovi senzori mjere fizički položaj uređaja. Ova kategorija uključuje senzore orijentacije i magnetometre. Platforma Android nudi dva senzora koji omogućuju određivanje položaja uređaja: senzor geomagnetskog polja i akcelerometar. Platforma Android također nudi senzor koji omogućuje da se odredi koliko je lice uređaja blizu objekta, poznato kao senzor blizine.

Naziv senzora	**Tip senzora**	**Opis**	**Uobičajena primjena**
MAGNETIC FIELD SENSOR	Hardverski	Mjeri geomagnetsko polje okoline za sve tri fizičke osi (x, y, z) u µT.	Stvaranje kompasa.
ORIENTATION SENSOR	Softverski	Mjeri stupnjeve rotacije koje uređaj čini oko sve tri fizičke osi (x, y, z).	Određivanje položaja uređaja.
PROXIMITY SENSOR	Hardverski	Mjeri blizinu objekta u cm u odnosu na zaslon uređaja.	Položaj telefona tijekom poziva.

Figure 3 Tablica senzora položaja

Senzor geomagnetskog polja i senzor blizine temeljeni su na hardveru. Većina proizvođača mobilnih telefona i tableta uključuje senzor geomagnetskog polja. Isto tako, proizvođači slušalica obično uključuju senzor blizine za određivanje kada se slušalica drži blizu lica korisnika (na primjer, tijekom telefonskog poziva). Za određivanje orijentacije uređaja može se koristiti očitanja s akcelerometra uređaja i senzora geomagnetskog polja.

Senzori položaja korisni su za određivanje fizičkog položaja uređaja u referentnom okviru svijeta. Na primjer, može se koristiti senzor magnetskog polja u kombinaciji s akcelerometrom za određivanje položaja uređaja u odnosu na sjeverni magnetski pol. Također možete koristiti ove senzore za određivanje orijentacije uređaja u referentnom okviru vaše aplikacije. Na primjer, senzor geomagnetskog polja daje vrijednosti jakosti geomagnetskog polja za svaku od tri koordinatne osi tijekom jednog događaja senzora.

2.2. Komuninkacija aplikacija i senzora na nivou operativnog sustava Android

Pristupiti senzorima dostupnim na uređaju platforme Android i prikupljanje neobrađenih podataka senzora se omogućuje pomoću okvira senzora Android. Okvir senzora dio je android.hardware paketa i uključuje sljedeće klase i sučelja:

- SensorManager

  Ova se klass koristi za stvaranje instance usluge senzora. Ova klasa pruža različite metode za pristup i popis senzora, registraciju i deregistraciju slušatelja događaja senzora i dobivanje informacija o orijentaciji. Ova klasa također nudi nekoliko konstanti senzora koje se koriste za izvješćivanje o točnosti senzora, postavljanje stopa prikupljanja podataka i kalibraciju senzora.

- Senzor

  Ova se klas koristi za stvaranje instance određenog senzora. Ova klasa pruža različite metode koje vam omogućuju određivanje mogućnosti senzora.

- SensorEvent

  Sustav koristi ovu klasu za stvaranje objekta događaja senzora, koji pruža informacije o događaju senzora. Objekt događaja senzora uključuje sljedeće informacije: neobrađene podatke senzora, vrstu senzora koji je generirao događaj, točnost podataka i vremensku oznaku događaja.

- SensorEventListener

  Ovo sučelje možete koristiti za stvaranje dvije metode povratnog poziva koje primaju obavijesti (događaje senzora) kada se promijene vrijednosti senzora ili kada se promijeni točnost senzora.

Kombiniranjem ovih klasa i sučelja android uređaj može koristiti senzore i primjenjivati u različitim kontekstima. U tipičnoj primjeni ove API-je povezane sa senzorom se može koristiti za obavljanje dva osnovna zadatka:

1. Identificiranje senzora i mogućnosti senzora

Identificiranje senzora i mogućnosti senzora tijekom rada korisno je ako aplikacija ima značajke koje se oslanjaju na specifične vrste senzora ili mogućnosti. Na primjer, možda ćete htjeti identificirati sve senzore koji su prisutni na uređaju i onemogućiti sve značajke aplikacije koje se oslanjaju na senzore koji nisu prisutni. Isto tako, mogu se identificirati svi senzori određene vrste kako bi se mogla odabrati implementacija senzora koja ima optimalne performanse za određenu aplikaciju.

2. Praćenje događaja senzora

Praćenje događaja senzora je način na koji se dobivaju neobrađeni podaci senzora. Događaj senzora događa se svaki put kada senzor detektira promjenu u parametrima koje mjeri. Događaj senzora pruža četiri informacije: naziv senzora koji je pokrenuo događaj, vremensku oznaku događaja, točnost događaja i neobrađene podatke senzora koji su pokrenuli događaj.

Dostupnost senzora se razlikuje od uređaja do uređaja, može se razlikovati i među verzijama Androida. To je zato što su Android senzori uvedeni tijekom nekoliko izdanja platforme. Na primjer, mnogi senzori predstavljeni su u Androidu 1.5 (API razina 3), ali neki nisu implementirani i nisu bili dostupni za upotrebu sve do Androida 2.3 (API razina 9). Isto tako, nekoliko senzora predstavljeno je u Androidu 2.3 (API razina 9) i Android 4.0 (API razina 14). Dva senzora su zastarjela i zamijenjena novijim, boljim senzorima.

Okvir Android senzora pruža nekoliko metoda koje olakšavaju određivanje koji se senzori nalaze na uređaju tijekom izvođenja. API također nudi metode koje omogućuju određivanje mogućnosti svakog senzora, kao što je njegov maksimalni domet, njegova razlučivost, njegovi zahtjevi za napajanje itd.

2.3. Primjena senzora u različitim aplikacijama

Senzori se u današnje doba primjenjuju u različitm apliakcijama, toliko da ponekad ljudi i ne primjete da se za neku funkcionalnost koriste senzori, makar oni bili hardverski, softverski ili kombinacija od oboje.

2.3.1. Primjeri upotrebe senzora pokreta

Žiroskopski senzor u telefonu može se koristiti u foto i video aplikacijama za stabilizaciju slike. One prima vrijednosti o rotaciji uređaja preko hardverskog dijela te zatim preko softvera pokuša što više smanjiti tresanje leće kako bi se dobila čišća slike odnosno video. Na slici 1 je prikazan primjer izgleda slike. prije i poslije primjene stabilizacije slike Ova metoda postoji na mnogim digitalnim kamerama i fotoaparatima no u proteklih pet godina sve više mobilnih uređaja je dobilo podršku za ovu značajku.

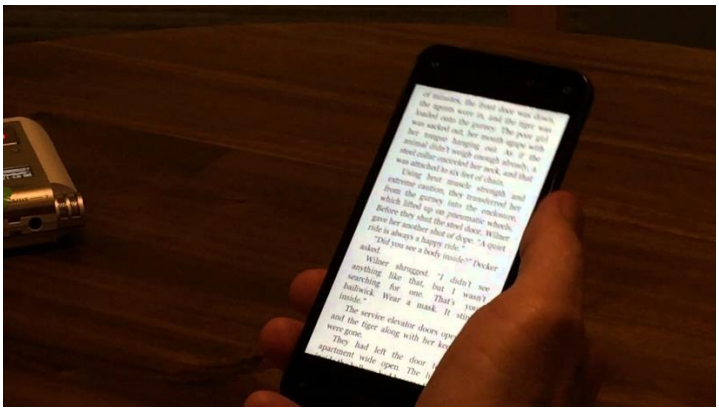
Slika 1 Prikaz stabilizacije slike

Žiroskop također mogu koristiti uređaji za praćenje fitnessa u kombinaciji s brzinomjerom i optičkim senzorom koji mjeri puls. Preko senzora žiroskopa i brzinomjera se može izračunati brzina kretanja ali također i vrsta kretanja što koriniku pomaže u vježbanju. Takoer preko optičkog senzora moguće je izračuanti korisnikov puls što dalje omogućava računanje potrošenih kalorija i drugih podatak tijela korisnika uređaja. Ova funkcionalnost je često primjenjena u pametnim satovima, na slici 2 je vidljiv primjer jednih od takvih pametnih satova.

Slika 2 Prikaz apliakcije za mjerenje pulsa

2.3.2. Primjeri upotrebe senzora okoliša

Senzor ambijentalnog svjetla u pametnim telefonima mogu koristiti aplikacije za čitanje. Senzor hvata podatke iz okoline o količini svjetlosti koju prepoznaje te zatim očitava razinu osvjetljenja ekrana te ju prilagođuje ovisno o osvjetljenju prostorije u kojoj se uređaj nalazi. Ova funkcionalnost senzora se može uočiti na gotovo svim modernim pametnim uređajima preko samog operativnog sustava no moguće je ovu funkcionalnost implementirati i kontrolirati preko samostalnih aplikacija. Ovaj senzor može korisničko iskustvo učiniti ugodnijim automatskim podešavanjem svjetline zaslona ovisno o uvjetima čitanja, slika 3 prikazuje primjer podešavanja svjetline ekrana tijekom čitanja u zatamnjenom prostoru .



Slika 3 Prikaz osvjetljenog ekrana mobilnog uređaja

Temperatura okoline i barometar mogu se koristiti u vremenskim aplikacijama za slučajeve kada korisnici žele saznati ne samo vrijeme na određenoj lokaciji već i trenutne vremenske uvjete s visokom razinom točnosti, na slici 4 se nalazi primjer aplikacije za vrijeme. Također, ove senzore mogu koristiti rudarski radnici za prepoznavanje potencijalno opasnih promjena unutar rudnika. Za ovakve senzore je potrebno imati hardversku implementaciju njih u mobilnom uređaju, no moguće je da se mobilni uređaj spoji na drugi uređaj koji ima potrebne senzore te mu tako javlja trenutno stanje okoline.

Slika 4 Prikaz aplikacije za vrijeme

2.3.3. Primjeri upotrebe senzora položaja

Senzor magnetskog polja, magnetometar, povezan sa senzorom brzinomjera može stvoriti kompas u mobilnim telefonima koji se mogu koristiti za navigaciju u prostoru, slika 5 prikazuje primjer aplikacije kompas. Sam magnetometar ne korsti stvarne magnete u uređaju kako bi to oštetilo uređaj, većina uređaja koristi magnetno-induktivnu tehnologiju koja očitava razlike u magnetskom polju zemlje te se time dobi funkcionalnost digitalnog kompasa.

Slika 5 Prikaz aplikacije Kompas

GPS senzori naširoko se koriste na pametnim telefonima, pametnim satovima i drugim uređajima poput rješenja za praćenje lokacije, planiranje rute i optimizaciju te geofencing. Za implementaciju ove funkcionlnosti također je potrebno i detaljno mapiranje terena kako bi senzor dao što točniju poziciju te je također potrebna dostupnost interneta. U zadnje vrijeme se također pojavljuje tehnologija koja omogućuje spajanje mobilnog uređaja sa satelitom što omogućava da se funkcionalnost GPS koristi čak i u kontekstu gdje nema dostupnog interneta. Na slici 6 je vidljiv primjer aplikacije za navigacije, točnije aplikacije naziva Google Maps.

Slika 6 Prikaz aplikacije Google Maps

GPS u kombinaciji sa beacon tehnologijom mogu se koristiti za slanje obavijesti kada je neki uređaj u pokretu što je primjenjeno u aplikacijama za dostavu hrane ili taxi službi kako bi korisnici vidjeli udaljenost naručene službe, na slici 7 je dan primjer praćenja lokacije narudžbe u aplikaciji Glovo.

Slika 7 Prikaz aplikacije Glovo

3. Prototip aplikacija sa senzorima

Za demonstraciju rada senzora u mobilnim aplikacijama izrađene su dvije aplikacije koje prikazuju rad tri senzora:

- Brzinomjer (TYPE_ACCELEROMETER) i žiroskop (TYPE_GYROSCOPE) za prikaz rada senzora pokreta
- Termometar (TYPE_AMBIENT_TEMPERATURE) za prikaz rada senzora okoliša te
- Magnetometar (TYPE_MAGNETIC_FIELD) za prikaz rada senzora položaja

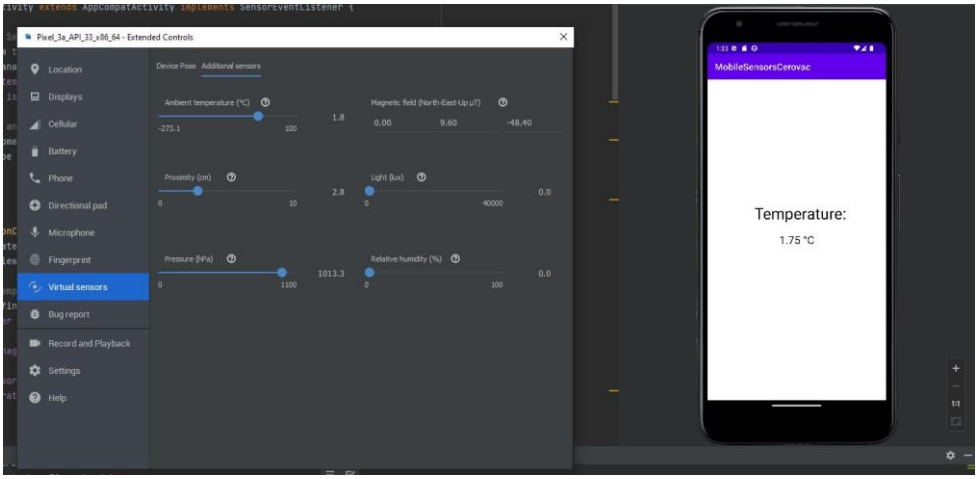
Prva aplikacija prikazuje rad brzinomjera, žiroskopa i termometra, dok druga prikazuje rad magnetometra u obliku kompasa. Aplikacije su rađene za Android platformu pomoću Android Studio razvojnog okruženja te je pisan u programskom jeziku Java na operativnom sistemu Windows. Prikaz apliakcije je odrađen preko emulatora kako oni daju pristup svim senzorima za razliku od stvarnih pametnih telefona koji nemaju sve hardverske senzore. Izrađene mobilne aplikacije se mogu pokretati i na emulatorima i na stvarnim Android uređajima

3.1. Aplikacija očitanja pokreta i temperature

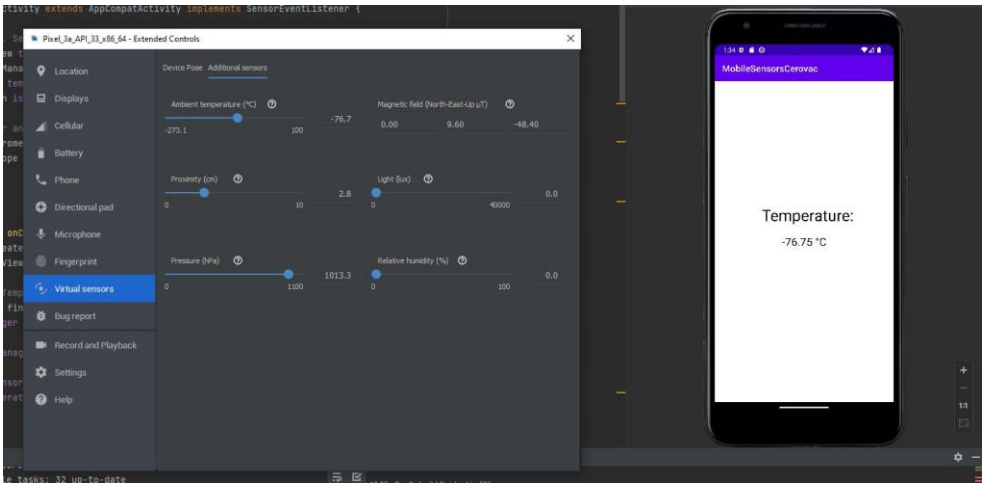
Prva aplikacija prikazuje rad senzora brzinomjera i senzora termometra. Aplikacija je jednostavnog koncepta. Na ekranu se mjenjaju boje ovisno da li se uređaj pomaknuo u lijevo ili u desno ili ako se zarotirao. Na ekranu se također ispisuje trenutna temperatura sobe u kojoj se uređaj nalazi.

3.1.1. Očitanje temperature

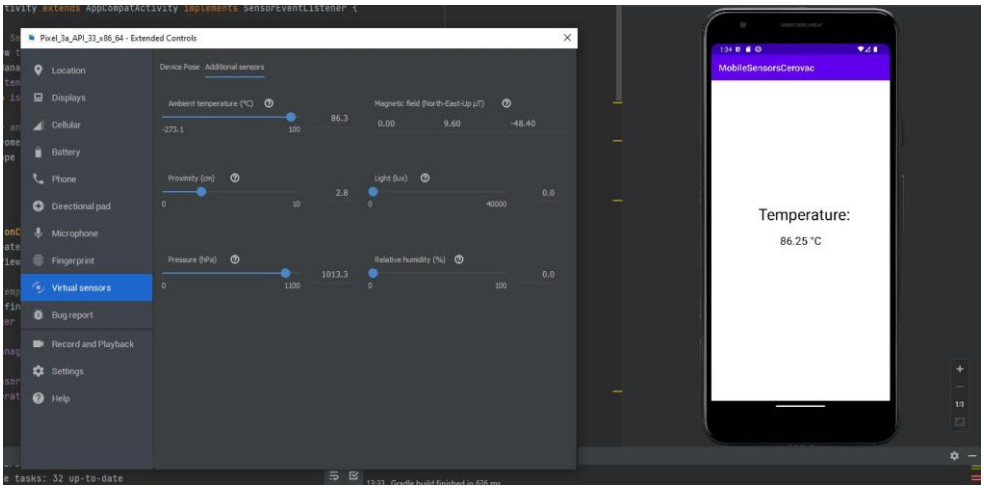
Očitanje temperature je jednostavnog principa. Na ekranu se ispisuje trenutna ambijentalna temperatura prostorije u kojoj se uređaj nalazi. Na sljedećim slikama (slike 8-10) je prikazano kako se ispis temperature na ekranu mjenja ovisno o promjeni trenutne temperature u senzorima u emulatru.



Slika 8 Aplikacija za očitanje temperature (1/3)



Slika 9 Aplikacija za očitanje temperature (2/3)



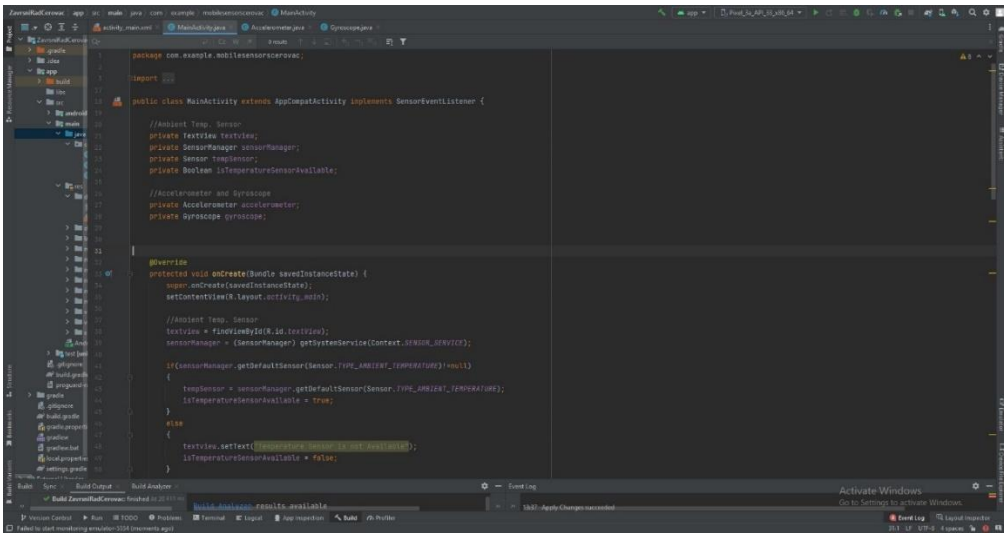
Slika 10 Aplikacija za očitanje temperature (3/3)

Konstanta za senzor će biti „TYPE_AMBIENT_TEMPERATURE“ te će se jedinica mjere biti u °C. Vrsta senzora u uređaju je hardverski senzor što znači da će podaci koje senzor da Android sustavu biti sirovi te će se ti sirovi podaci ispisivati na ekran. Za izradu aplikacije termometar potrebno je u „activity_main.xml“ datoteci dodati dva TextView-a. TextView-ovi služe za ispis teksta na ekran u aplikaciji. Prvi će imati statičan tekst koji će glasiti „Temperature:“, dok će drugi imati promjenjivi tekst na kojemu će se ispisivati trenutna ambijentalna temperatura te je drugom promjenjivom tekstu potrebno definirati ID, u ovom slučaju mu je dan ID „textView“.

Dalje se treba prijeći u datoteku „MainActivity.java“ u koju će cijela implementacija termometra biti smještena. Prvotno treba definirati varijable, imena varijabli nisu bitna, ali sam tip varijabi je. Imati ćemo četiri varijable, prva je tipa „TextView“ i nju ćemo spojiti sa TextView-om koji ispisuje trenutnu temperaturu na ekran te se u ovom slučaju naziva „textview“, druga je tipa „SensorManager“ koji će služiti za upravljenje senzorom, treća je varijabla tipa „Sensor“ koja nam služi za samo pozivanje senzora te je četvrta varijabla tipa „boolean“ naziva „isTemperatureSensorAvailable“ koja će služiti za provjeru da li uređaj ima fizički senzor za ambijentalnu temperaturu, što je bitno jer mnogi uređaji nemaju ugrađen taj hardverski senzor. Također, u liniju gdje je definirana glavna klasa

„MainActivity“ treba dodati „implements“ te implementirati „SensorEventListener“ što će automatski dodati dvije „@Override“ funkcije tipa „public void“ i naziva „onSensorChanged“ i „onAccuracyChanged“. One su bitne jer se preko njih prate promjene u senzoru i iste ispisuju na ekran uređaja.

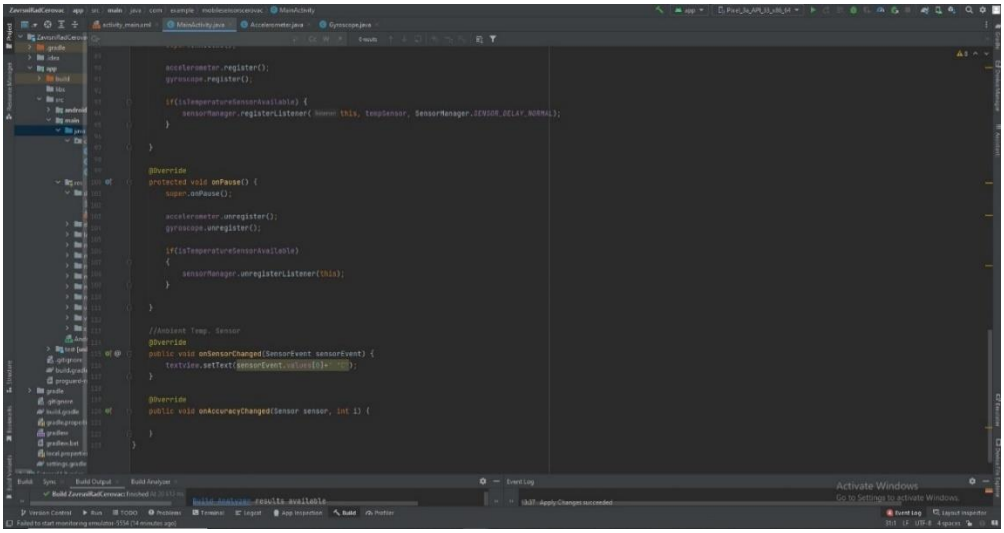
U funkciji „onCreate“ se definiraju vrjednosti varijabli te također tip senzora koji se koristi. Prvo treba varijabli tipa „TextView“ dati ID od TextView-a za ispis na ekran, to se radi preko metode „findViewId“ te se metodi proslijedi „R.id.textView“, „textView“ je u ovom slučaju ID TextView-a za ispis na ekran. Dalje treba varijabli tipa „SensorManager“ dati konetkst senzor servisa. To se radi preko metode „getSystemService“ te metodi proslijedi „Context.SENSOR_SERVICE“. Nakon toga se vrši provjera da li uređaj posjeduje senzor za ambijentalnu temperaturu preko if naredbe. U samoj if naredbi varijabli tipa „SensorManager“, u ovom slučaju naziva „sensorManager“ treba definirati tip senzora preko metode „getDefaultSensor“. Toj metodi treba proslijediti „Sensor.TYPE_AMBIENT_SENSOR“ te dodati provjeru „ != null“, što znači da će se ostatak koda nastaviti odvijati ako postoji fizički senzor. U protivnom u else naredbi se ispisuje poruka koja javlja da senzor nije pronađen te se varijabla tipa „boolean“ u ovom slučaju naziva „isTemperatureSensorAvailable“ postavlja na „false“. Ako postoji senzor „isTemperatureAvailable“ se postavlja na „true“ te se u varijablu tipa „Sensor“, u ovom slučaju naziva „tempSensor“ sprema prije postavljena varijbala „sensorManager“ te je time tip senzora postavljen na senzor za temperaturu ako on postoji.



Slika 11 Kod aplikacije za mjerenje temperature (1/2)

Sada treba dodati dvije nove „@Override“ funkcije, to se radi pritiskom tipki ctrl+o te u novo prozoru se biraju funkcije „onResume“ te „onPause“, obije su tipa „public void“. U „onResume“ funkciji treba registrirati listener-a koji prati da li je senzor u funkciji. Prvo se u if naredbi provjerava da li je „isTemperatureSensorAvailable“ jednak „true“ te ako je onda se na varijablu „sensorManager“ poziva metoda „registerListener“ te se njoj prosljeđuje referenca „this“ te „tempSensor“ i „SensorManager.SENSOR_DELAY_NORMAL“. Time je registriran senzor sa normalnim odgodom. U „onPause“ funkciji se također provjerava da li postoji senzor za temperaturu te se suprotno od „onResume“ funkcije ovdje odjavljuje listener za senzor. To se vrši metodom „unRegisterListener“ kojoj se prosljeđuje samo „this“. Na poslijetku treba još samo u „onSensorChanged“ funkciji napraviti varijabla tipa „SensorEvent“, u ovom slučaju naziva „sensorEvent“, koja sluša kada se deogodila promjena na senzoru te se preko metode „setText“ na varijabli „textview“ sprema trenutno stanje

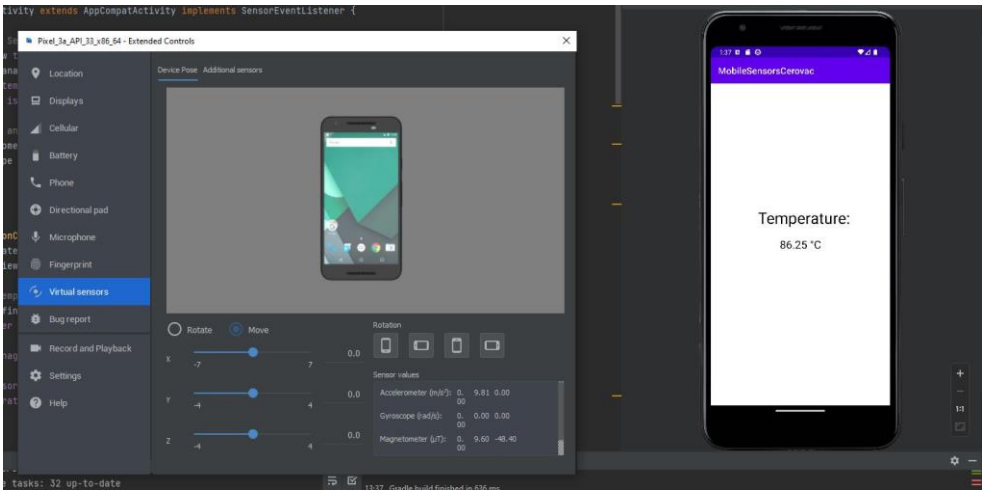
senzora i ispisuje na ekran uređaja. Time je implementacija funkcionalnosti očitanja temperature dovršena (cjelokupan kod aplikacije za mjerenje temperature je vidljiv na slikama 11 i 12).



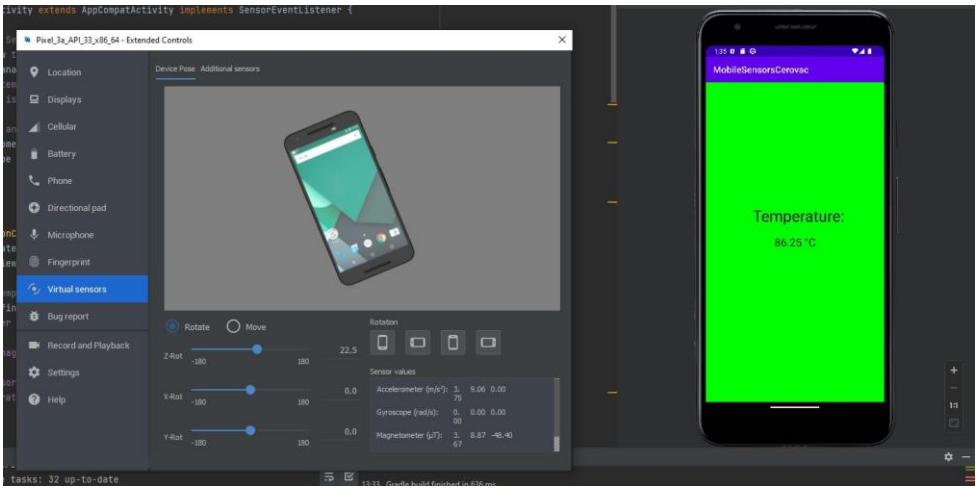
Slika 12 Kod aplikacije za mjerenje temperature (2/2)

3.1.2. Očitanje pokreta

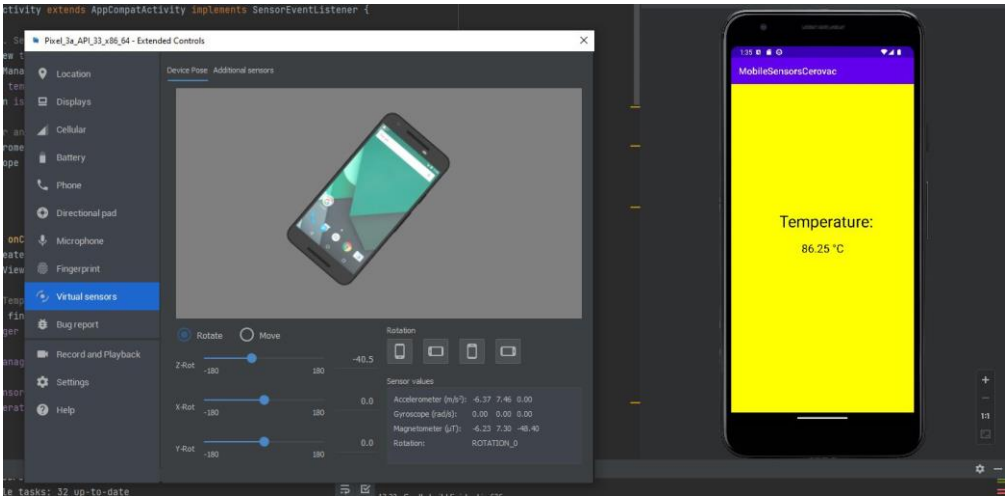
Svrha ove funkcionalnosti aplikacije je mjenjanje boje ekrana ovisno o položaju uređaja. Očitanje pokreta se sastoji od dva senzora: brzinomjera i žiroskopa. Razlog tome je što ova aplikacija mjenja boje ovisno da li se uređaj kreće linearno (za to je potreban brzinomjer) i ako se rotira (za što je potreban žiroskop). Prije nego što se uređaj pomakne njegov ekran je bijele boje (slika 13). Ako se zarotira u lijevo ekran će promjeniti boju u zelenu (slika 14), ako se zarotira u desno ekran će promjeniti boju u žutu (slika 15), u obje instance će se cijeli ekran uređaja zarotirati iz portret pregleda u landscape pregled (što je također funkcionalnost žiroskopa).



Slika 13 Aplikacija za očitanje pokreta i rotacije (1/5)

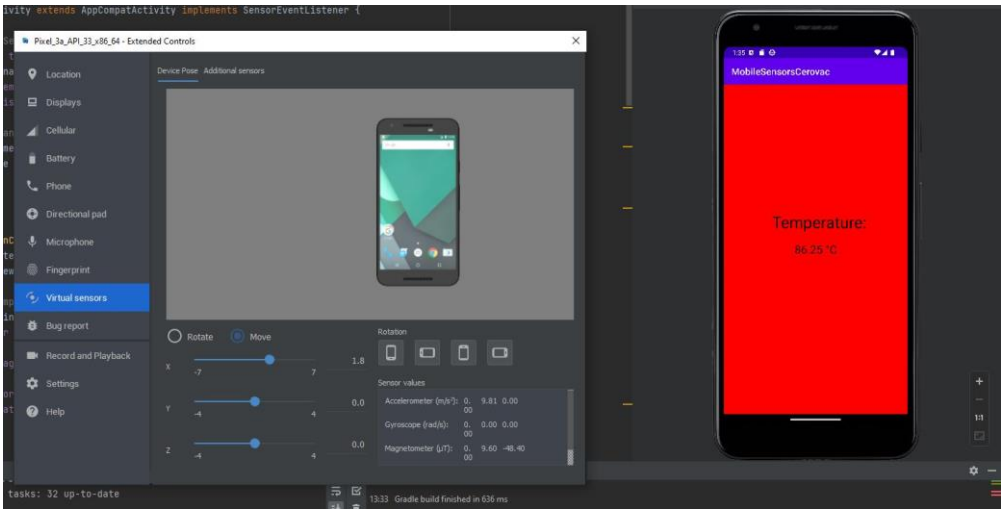


Slika 14 Aplikacija za očitanje pokreta i rotacije (2/5)

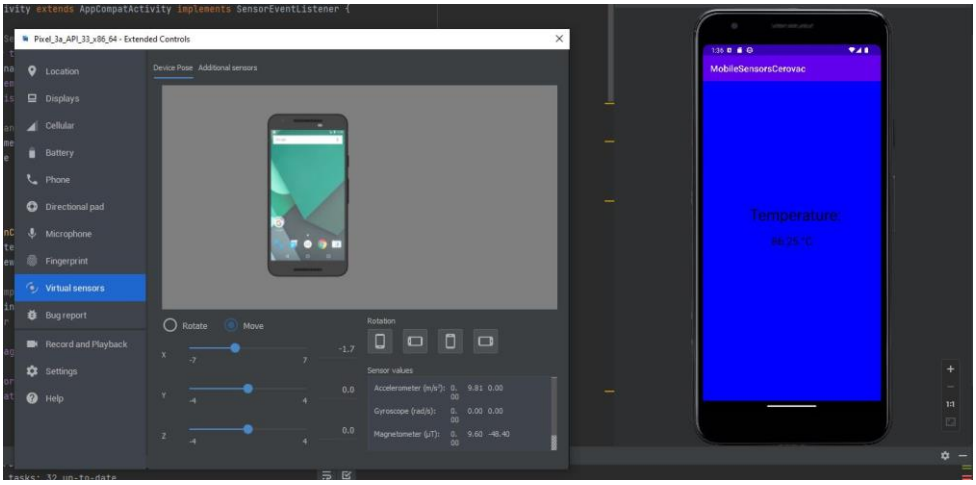


Slika 15 Aplikacija za očitanje pokreta i rotacije (3/5)

Ako se cijeli uređaj pomakne u desno ekran će promjeniti boju u crvenu (slika 16) te ako se cijeli uređaj pomakne u lijevo ekran će promjeniti boju u plavu (slika 17). Ako se uređaj pomiče kroz cijeli prostor ekran će mjenjati boje ovisno o trenutnom smjeru kretanja.



Slika 16 Aplikacija za očitanje pokreta i rotacije (4/5)



Slika 17 Aplikacija za očitanje pokreta i rotacije (5/5)

Dio koda će biti izvršen u već napravljenoj datoteci „MainActivity.java“ (za detaljan prikaz te datoteke treba se referirati na poglavlje 3.1.1. ovog završnog rada). Uz tu datoteku kreirati će se dvije dodatne dadoteke: „Accelerometer.java“ u kojoj će biti implementirana funkcionalnost brzinomjera i „Gyroscope.java“ u kojoj će biti implementirana funkcionalnost žiroskopa.

U datoteci „Accelerometer.java“ unutar klase „Accelerometer“ treba definirati „public interface“ koji u sebi ima funkciju tipa „void“ i naziva „onTranslation“ te su u njoj proslijeđene varijable tipa „float“ i naziva „tx“, „ty“ i „tz“. Zatim treba definirati varijablu tipa „Listener“, u ovom slučaju naziva „listener“. U sljedećoj funkciji tipa „void“ i naziva „setListener“ se stvara nova varijabla tipa „Listener“ i naziva „l“ te se ona u samoj funkciji poistovjećuje sa prije napravljenom varijablom „listener“. Time je postavljeno da senzor sluša kada je u pokretu, tj. kada je aktivan. Dalje je potrebno definirati tri varijable, prva je tipa „SensorManager“ i u ovom slučaju se zove „sensorManager“, druga je tipa „Sensor“ i u ovom se slučaju zove „sensor“ te posljednja je tipa „SensorEventListener“ i u ovom se slučaju zove „sensorEventListener“.

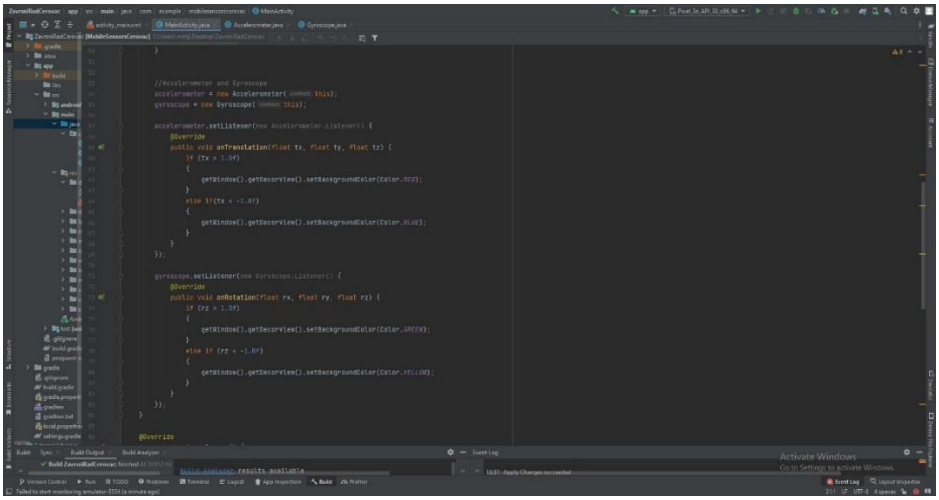
Sljedeće treba napraviti konstruktor, u ovom slučaju se zove „Accelerometer“, te mu se prosljeđuje varijabla tipa „Context“ i naziva „context“ kao argument. Unutar konstruktora treba varijabli „sensorManager“ postaviti kontekst i „SENSOR_SERVICE“ to se radi metodom „getSystemService“ te

se metodi prosljeđuje „Context.SENSOR_SERVICE“. Zatim treba varijabli „sensor“ postaviti tip senzora na brzinomjer. Prvo se „sensor“ poistovjećuje sa „sensorManager“ varijablom, zatim treba pozvati metodu „getDefaultSensor“ kojoj se prosljeđuje „Sensor.TYPE_LINEAR_ACCELERATION“ i time je tip senzora postavljen na brzinomjer. Uz to treba varijabli „sensorEventListener“ se treba pozvati even listener metoda. To se radi tako da varijablu „sensorEvenListener“ samo treba poistovjetiti sa „new SensorEvenListener ()“ što će automatski otvoriti vitičaste zagrade i unutar njih staviti dvije nove funkcije koje su tipa „@Override“ i nazivaju se „onSensorChanged“ i „onAccuracyChanged“. Unutar funkcije „onSensorChanged“ se stvara varijabla tipa „SensorEvent“, koja se u ovom slučaju zove „sensorEvent“ te se također događa if provjera koja gleda da „listener“ nije „null“. Ako je to istina onda se nad „listener“ varijabli poziva prije definirana funkcija „onTranslation“ kojoj se ovdje prosljeđuju kordinate X, Y i Z osi te se te varijable spremaju unutar „onTranslation“ funkcije u varijable „tx“, „ty“ i „tz“. To se izvodi tako da se uzima „sensorEvent“ varijabla koja prati promjene senzora te se pomoću „.values[]“ proširenja hvata dio polja koji odgovara osima, „.values[0]“ odgovara X osi, „.values[1]“ odgovara X osi te „.values[2]“ odgovara z osi. U funkciju naziva „onAccuracyChanged“ ne treba ništa dodavati. Izvan konstruktora „Accelerometer“ treba dodati dvije nove funkcije tipa „public void“, one su: „register ()“ i „unregister()“. Unutar njih je potrebno registrirati i odjaviti rad sensora. Za registraciju treba u „register()“ funkciji pozvati metodu „registerListener“ nad varijablom „sensorManager“ te proslijediti varijable „sensorEventListener“, „sensor“ i „SensorManager.SENSOR_DELAY_NORMAL“. Za odjavu senzora u funkciji „unregister()“ potrebno je samo pozvati metodu „unregisterListener“ i nad varijablom „sensorManager“ te metodi proslijediti varijablu „sensorEventListener“. Time je završen rad u datoteci „Accelerometer.java“.

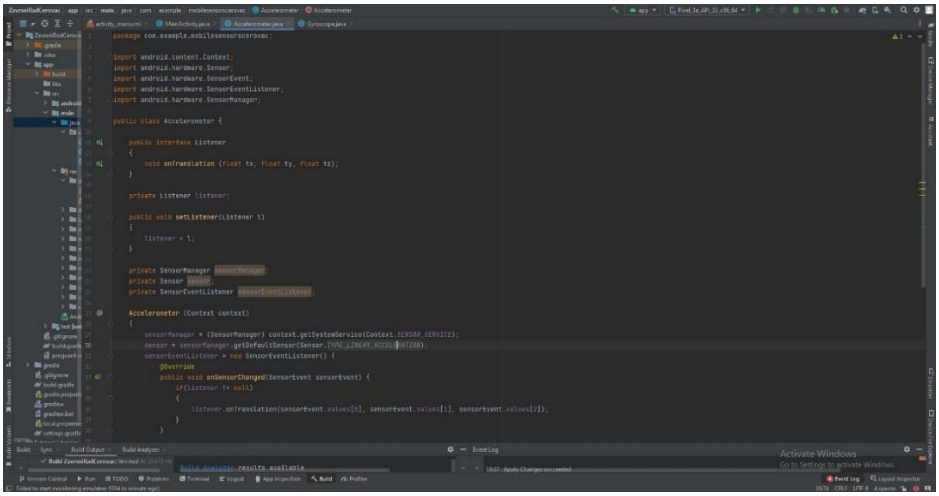
U datoteci „Gyroscope.java“ treba implementirati funkcionalnost žiroskopa. Postupak je gotovo identičan kao i kod stvaranje brzinomjera, jedina je razlika da kod postavljanja tip senzora varijabli „sensor“ umjeto tipa „TYPE_LINEAR_ACCELERATION“ treba postaviti tip na „TYPE_GYROSCOPE“. Uz to kod stvaranja interface-a treba napraviti funkciju „onRotation“ umjesto „onTranslation“ koja će gledati promjenu u rotaciji, ona također ima tri varijable tipa „float“ i nazivaju se „rx“, „ry“ i „rz“.

Sada se je potrebno vratiti u datoteku „MainActivity.java“ i u njoj stvoriti dvije nove varijable, jedna tipa „Accelerometer“ i naziva „accelerometer“, a druga tipa „Gyroscope“ i naziva „gyroscope“ što će povezati datoteke „Accelerometer.java“ i „Gyroscope.java“ sa „MainActivity.java“ datotekom. U „onCreate“ funkciji potrebno je pozvati nove instance brzinomjera i žiroskopa. To se radi tako da se pozovu varijable „accelerometer“ i „gyroscope“ te da one pozivaju same sebe sa „new Accelerometer(this)“, odnosno „new Gyroscope (this)“ linijama koda. Sada je potrebno postaviti even listenere na brzinomjer i žiroskop, koji će pratiti rad senzora te izvršiti mjenjanje boje ekrana. Prvo se pozove instanca „accelerometer“ te joj se da metoda „setEventListener“ koja ima proslijeđen „new Accelerometer.Listener()“ u sebi. Unutar nje se napravi funkcija tipa „public void“ i naziva „onTranslation“ kojoj se prosljeđuju varijable koje prate pokret u tri osi. Zatim se vrši provjera po X osi (može se napraviti provjera za bilo koje druge osi, ali za potrebe ovoga rada uzeta je samo X os) i preko metode „getWindow().getDecorView().setBackgroundColor()“ se ovisno o željenoj poziciji boja ekrana mjenja u crvenu ili plavu. Postupak je isti za žiroskop samo što se njemu poziva funkcija „onRotation“ umjesto „onTranslation“ te se boja ekrana ovisno o smjeru rotacije mjenja u žutu ili zelenu. Na samom kraju u funkciji „onResume“ je potrebno registrirati listener za brzinomjer i žiroskop preko metode „.register()“ te ih odjaviti u funkciji „onPause“ preko metode „.unregister()“. Time je funkcionalnost mjenaje boje ekrana ovisno o položaju i rotaciji uređaja uspješno implementirana te je

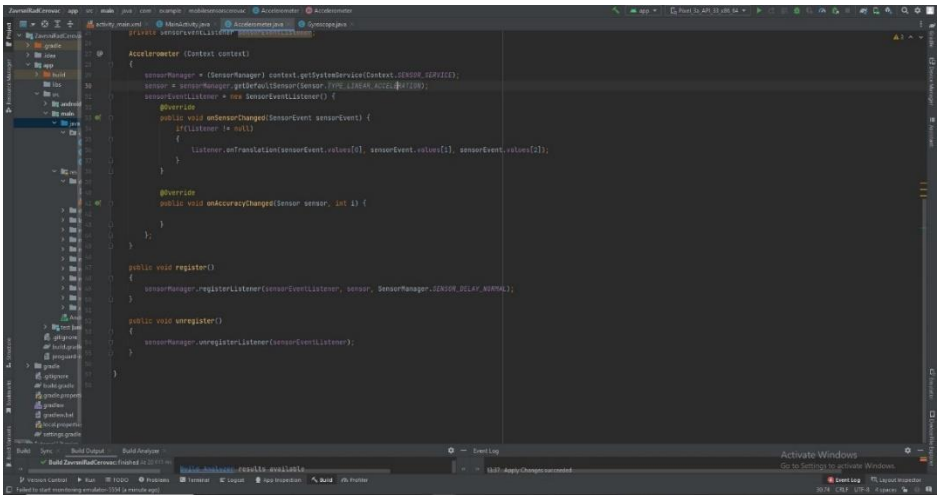
i sama aplikacija očitanja promjene pokreta i temperature završena (cjelokupan kod aplikacije za očitanje pokreta i rotacije je vidljiv na slikama od 18 do 22).



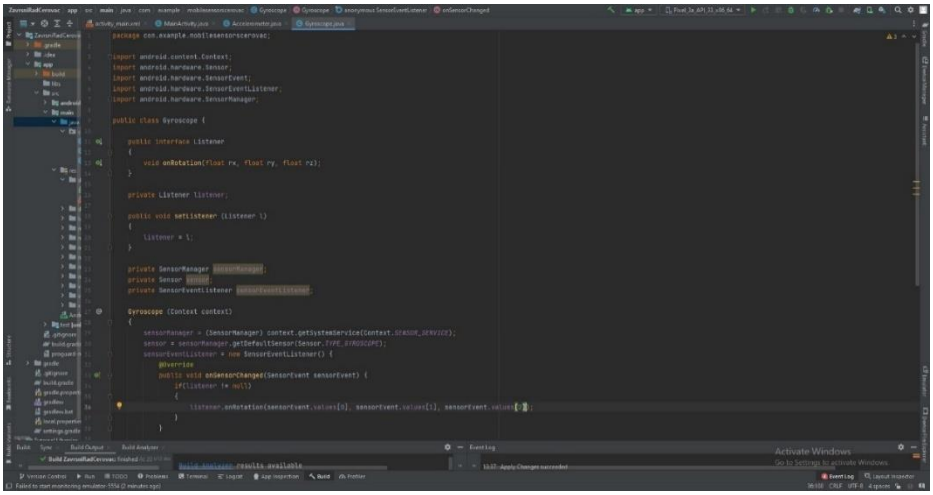
Slika 18 Kod aplikacija za očitanje pokreta i rotacije (mjenjanje boja)



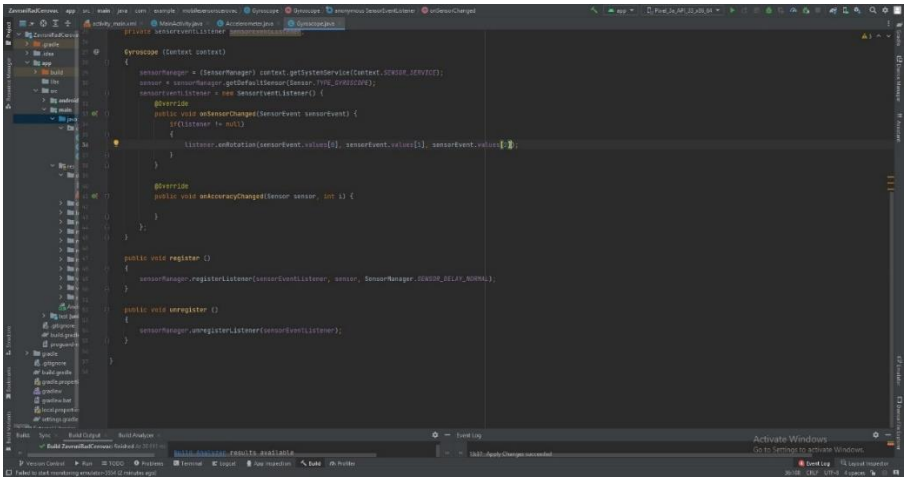
Slika 19 Kod aplikacije za očitanje pokreta (1/2)



Slika 20 Kod aplikacije za očitanje pokreta (2/2)



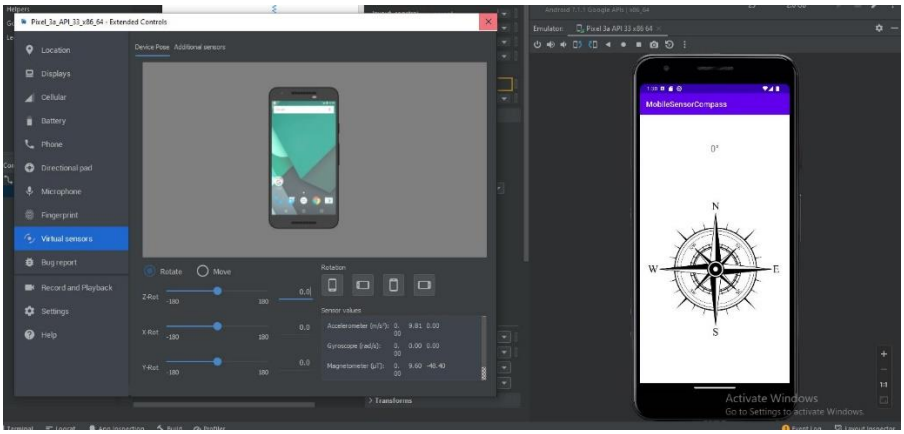
Slika 21 Kod aplikacije za očitanje rotacije (1/2)



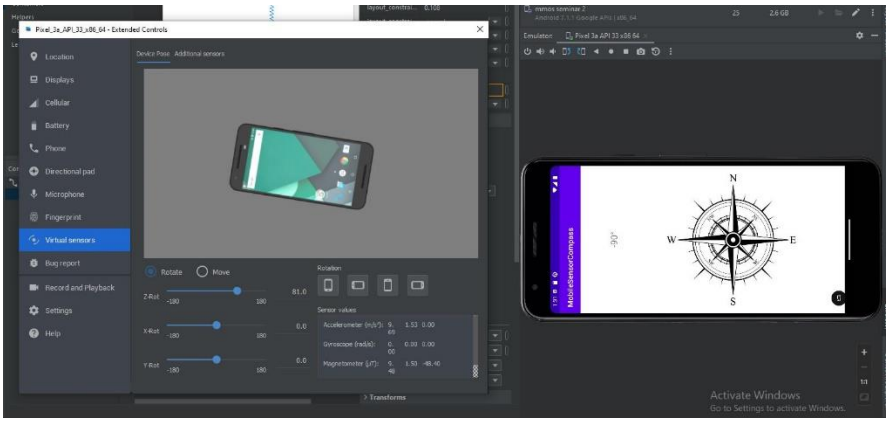
Slika 22 Kod aplikacije za očitanje rotacije (2/2)

3.2. Aplikacija očitanja položaja (kompas)

Za prikaz rada senzora položaja napravljena je aplikacija koja ima funkcionalnost kompasa. Na ekranu uređaja se nalazi slika kompasa koja se rotira ovisno o poziciji sjevera te se također ispisuju stupnjevi koji ukazju odmak uređaja od sjevera u stupnjevima. Aplikacija radi i na emulatoru i na stvarnom uređaju, doduše na stvarnom uređaju aplikacija je puno preciznija (slike 23 i 24 prikazuju rad aplikacije u emulatoru, dok slika 25 prikazuje rad aplikacije na stvarnom uređaju).



Slika 23 Aplikacija kompas na emulatoru (1/2)



Slika 24 Aplikacija kompas na emulatoru (2/2)

Slika 25 Aplikacija kompas na stvarnom uređaju

Za stvaranje aplikacije kompas potrebna su dva senzoa, a to su brzinomjer i magnetometar. Koristeći niz podataka koji stvara brzinomjer zajedno sa nizom podataka koji koristi magnetometar može se dobiti matrica rotacije. Zatim pomoću niza u matrici rotacije može se dobiti niz orijentacije te će prvi element u tom nizu biti stupanj po kojemu treba zarotirati sliku kompasa da uvijek pokazuje na sjever.

Prije implementacije koda treba poraditi na izgledu aplikacije. U „activity_main.xml“ datoteci potrebno je dodati TextView komponentu koja će ispisivati trenutnu vrjednost stupnja po kojemu uređaj odstupa od sjever te također ImageView koji će biti slika koja će se rotirati ovisno o stupnju rotacije. Obije komponente trebaju imati postavljen ID, u ovom slučaju ID od TextView komponente je „xTextView“ dok je ID od ImageView komponente „imageView“. Za ImageView je korištena besplatna slika kompasa preuzeta sa interneta u .PNG formatu, doduše bilo koja slika može biti korištena u ovom procesu.

Sljdeće se treba prebaciti u „MainActivity.java“ datoteku. Tu je potrebno definirati varijable koje će biti dalje korištene u kodu. Prvo je potrebno napraviti varijable za ImageView i TextView komponente, to se radi pozivom varijabli tipa „TextView“, u ovom slučaju naziva „textView“ za prikaz stupnja rotacije na ekranu i varijabli tipa „ImageView“, u ovom slučaju naziva „imageView“ za prikaz slike na ekranu uređaja. Dalje je potrebno pozvati varijablu tipa „SensorManager“, ovdje se naziva sensorManager koja će upravljati senzorima i dvije varijable tipa „Sensor“ koje se ovdje zovu „accelerometerSensor“ i „magnetometerSensor“ koje će pozivati senzore brzinomjera magnetometra. Nakon toga potrebno je definirati četiri niza koja će se koristiti za provjeru i očitanja rotacije. Oni su svi tipa „float[]“ i zovu se redom: „lastAccelerometer“, „lastMagentometer“, „rotationMatrix“ i „orientation“, veličina svih nizova je podešena na tri osim kod niza „rotationMatrix“ koji je podešen na devet. Sljedeće je potrebno definirati dvije varijable tipa „boolean“ i podesiti njihovu zadanu vrijednost na „false“, u ovom slučaju ove se varijable zovu „isLastAccelerometerArrayCopied“ i „isLastMagnetometerArrayCopied“. Na posljetku potrebno je definirati varijablu tipa „long“, ovdje se ona zove „lastUpdateTime“ i postaviti njenu vrijednost „0“ te također je potrebno definirati varijablu tipa „float“, ovdje se zove „currentDegree“ i postaviti njenu vrijednost na „0f“.

Unutar već postojeće funkcije „onCreate“ treba postaviti tipove senzora u brzinomjer i magnetometar te povezati komponente TextView i ImageView sa njihovim istoimenim varijablama. Za povezivanje tekste potrebno je u varijablu „textview“ preko metode „findViewById“ proslijediti

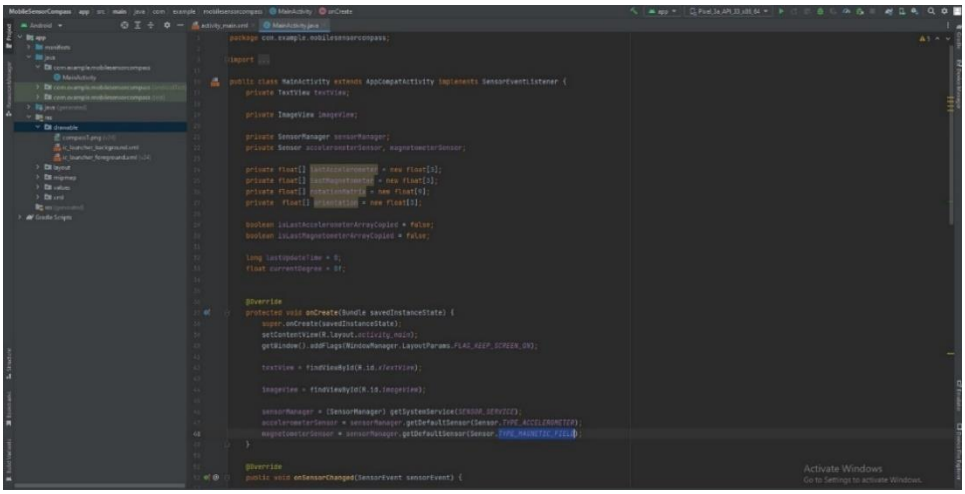
„R.id.xTextView“ gdje je u ovom slučaju „xTextView“ ID od TextView komponente u koju želimo ispisivati stupnje rotacije. Za ispis slike na ekran je veoma sličan postupak, u varijablu „imageView“ potrebno je ponovno preko metode „findViewById“ proslijediti „R.id.imageView“ gdje je u ovom slučaju „imageView“ ID od ImaageView komponente odnosno slike kompasa koja će se rotirati. Za registraciju senzora prvo je potrebno pozvati varijablu „sensorManager“ te nad njom pozvati metodu „getSystemService“ i metodu proslijediti „SENSOR_SERVICE“. Sljedeće treba pozvati varijablu „accelerometerSensor“ i nju poistovjetiti sa sensorManager varijablom, ali nad kojom je pozvana metoda „getDefaultSensor“ u koju je proslijeđena vrjednost „Sensor.TYPE_ACCELERATOR“ što definira varijablu „acceleratorSensor“ u tip senzora brzinomjer. Isti postupak se ponovi nad varijablem „magnetometerSensor“ jedino se njoj prosljeđuje „Sensor.TYPE_MAGNETIC_FIELD“ kako bi se registrirao tip senzora u magnetometar. Time su definirani brzinomjer i magnetometar te se mogu koristiti u ostatku koda.

Sljedeće je potrebno otići u liniji koda na početku gdje je definirana glavna klasa naziva „MainActivity“ i na nju dodati „implements“ te je potrebno implementirati „SensorEventListener“. Njegovom implementacijom će se stvoriti dvije nove „@Override“ funkcije naziva „onSensorChanged“ i „onAccuracyChanged“. Također je potrebno registrirati i odjaviti listener-a. To se radi pritiskom tipki ctrl+o na tipkovnici te zatim odabirom „onResume“ i „onPause“ funkcija te če time ove dvije funkcije biti dodane u kod, obije funkcije su tip „public void“. Registracija i odjava senzora je identična kod brzinomjera i magnetometra, jedina je razlika u nazivu varijabli koje su defnirane za svaki od dva senzora. Kod registracije potrebno je otići u funkciju „onResume“ te nad „sensorManager“ varijablom pozvati metodu „registerListener“ te njoj proslijediti „this“, „accelerometerSensor“ (za magnetometar potrebno je proslijediti varijablu „magnetometerSensor“) te „SensorManager.SENSOR_DELAY_NORMAL“, time je listener registriran na sensor manager-a koji je upravlja sa senzorima brzinomjera i magnetometra te im je postavljeno normalno zakašnjenje što znači da će se promjene u aplikaciji ovinso o promjenama senzora događati u stvarnome vremenu. Za odjavu listenera potrebno je otići u funkciju „onPause“, postupak je kao i kod registracije identičan kod odjave listener-a za oba senzora, jedina razlika je u imenima varijabli. Unutar „onPause“ funkcije nad varijablom „sensorManager“ potrebno je pozvati metodu „unregisterListener“ te joj proslijediti „this“ te „accelerometerSensor“ za brzinomjer odnosno „magnetometerSensor“ za magnetometar. Time su listener-i registrirani i odjavljeni za senzore brzinomjera i magnetometra što će značiti da će se ispis vrijednosti senzora mjenjati samo kad dođe do promjena u očitanju senzora.

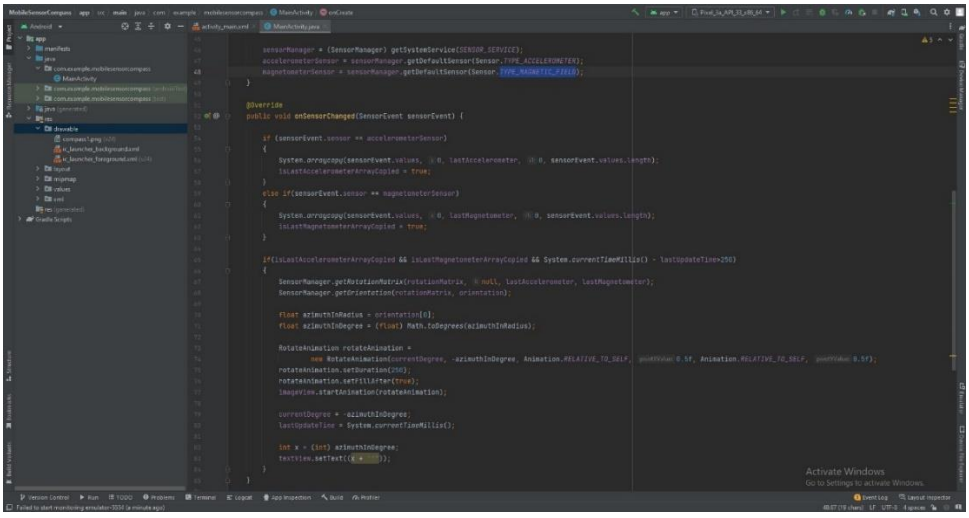
Na kraju je potrebno očitati trenutno stanje senzora, izračunati stupanj rotacije te animirati sliku i ispisati vrjednost stupnjeva rotacije na ekranu. Za to je potrebno otići u funkciju „onSenorChanged“. Prvo u njoj treba provjeriti da li je trenutni senzor od kojeg se hvataju očitanja brzinomjer ili magnetometar. Treba otvoriti novu if izjavu koja će pregledavati da li je trenutni senzor brzinomjer. To se radi preko uvjeta „if (sensorEvent.sensor == accelerometerSensor“ u protivnom u else if izjavi provjerava da li je sensor magnetometar preko uvjeta „else if(sensorEvent.sensor == magnetometerSensor)“. Unutar tih izjavi se kopira novo stanje niza zasebnih senzora. To se vrši u liniji „System.arraycopy(sensorEvent.values, 0, lastAcclerometer, 0, sensorEvent.values.length);“, ova linija ukazuje kako će se u „lastAccelerometer“ varijablu spremiti novi niz podataka očitan od strane senzora brzinomjera te je na kraju „boolean“ varijablu „isLastAccelerometerArrayCopied“ potrebno postaviti na „true“. Za očitanje stanja magnetometra postupak je isti, jedino treba zamijeniti varijablu „lastAccelerometer“ sa varijablom „lastMagentometer“ i varijablu „isLastAccelerometerArrayCopied“ zamjeniti sa varijablom „isLastMagnetometerArrayCopied“.

Sljedeća if izjava provjerava da li su kopirani nizovi za brzinomjer i magnetometar te ako jesu kreće računica stupnja rotacije i animacije slike. Uvjet za izvršavanje if izjave zahtjeva već spomenuto kopiranje niza oba senzora ali također preko uvjeta „System.currentTimeMillis() – lastUpdateTime>250) osigurava da se animacija slike neće ažurirati više od 4 puta u sekundi. Za dobivanje matrice rotacije nad „SensorManager“-u se koristi metoda „getRotationMatrix“ u koju su proslijeđeni varijabla „rotationMatrix“, „null“, „lastAccelerometer“ te „lastMagnetometer“ što u stvara matricu rotacije u „rotationMatrix“ varijabli. Za dobivanje orijentacije nad „SensorManager“-u se treba pozvati metode „getOrientation“ te u nju proslijediti „rotationMatrix“ te „orientation“ varijable. Time je dobivena trenutna orijentacija uređaja. Nakon toga treba pretvoriti trenutnu orijentaciju koja je u radijanima u stupnjeve. Prvo se definira varijabla tipa „float“, u ovom slučaju naziva „azimuthInRadius“ i u nju se sprema prvi element u nizu „orientation[0]“. Zatim se definira nova varijabla tipa „float“, u ovom slučaju imena „azimuthInDegree“ te se u nju radijani pretvaraju u stupnjeve preko metode „Math.toDegrees()“, gdje se metodi prosljeđuje „azimuthInRadius“.

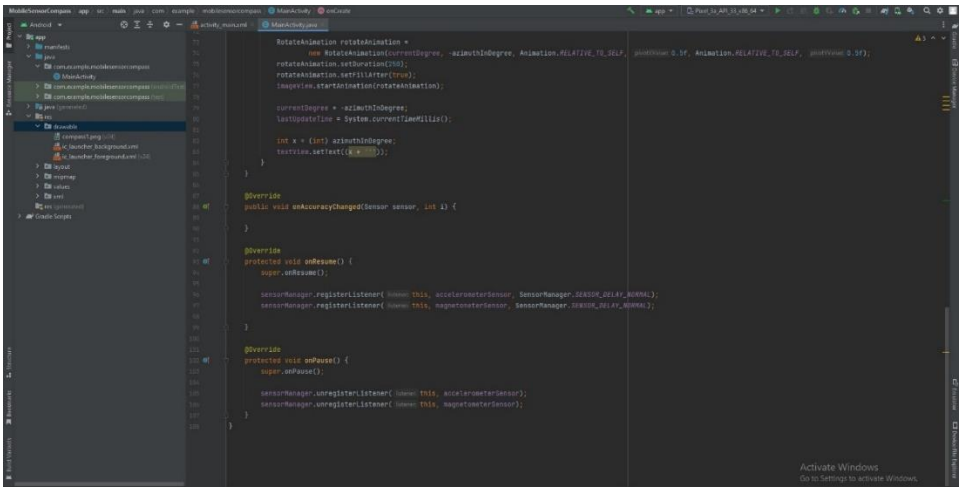
Sada je vrijme za animiranje slike. Prvo se pozove objekt „RotateAnimation“ gdje se u ovom slučaju instanca naziva rotateAnimation i u njoj se pozove nova instanca objekta u kojoj se spremi sljedeća vrijednost: „new RotateAnimation(currentDegree, -azimuthInDegree, Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);“. Ova vrijednost ukazuje da će objekt očitati trenutni stupanj rotacije, oduzeti novu vrijednost stupnja rotacije te relativno na samog sebe pokrenuti animaciju za vrjednost of „0.5f“ prvo po X osi, a zatim će to isto napraviti po Y osi. Dalje se brzina rotacije postavlja na 250ms preko linije: „rotateAnimation.setDuration(250)“ te se nova pozicija animacije primjenjuje kao trenutna preko linije: „rotateAnimation.setFillAfter(true);“. Sama animacija se pokreće linijom: „imageView.startAnimation(rotateAnimation);“. Time je animacija definirana i postavljena da se osvježuje četiri puta u sekundi, ako je došlo do promjene u očitanju senzora (ova vrijednost se može promijeniti ovisno o potrebama i ukusima). Još samo treba ažurirati stupnja rotacija i trenutno vrijeme. Prvo se u varijablu „currentDegree“ postavlja vrijednost „-azimuthInDegree“, ovdje je veoma bitan „-“ jer inače vrijednost rotacije neće biti konzistentan te se u varijablu „lastUpdateTime“ sprema trenutno vrijeme preko metode „System.currentTimeMillis()“. Na samom kraju je preostalo ispisati stupanj rotacije na ekran uređaja. Definira se varijabla „int x“ u koju se sprema vrijednsot varijable „azimuthInDegree“ te se onda preko „textView.setText“ metode trenutni stupnjevi ispisuju na ekran uređaja u °C putem varijable „textView“ koja je povezana sa komponentom TextView u „activity_main.xml“ datoteci. Sa time izrada aplikacije kompada je gotova i spremna za upotrebu na emulatoru i na stvarnom uređaju koji posjeduje senzore brzinomjera i magnetometra (cjelokupan kod aplikacije kompas je vidljiv na slikama 26, 27 i 28).



Slika 26 Kod aplikacije kompas (1/3)



Slika 27 Kod aplikacije kompas (2/3)



Slika 28 Kod aplikacije kompas (3/3)

4. Zaključak

Kako je rasla popularnost mobilnih uređaja tako se uz njih pojavila i potreba za mobilnim aplikacijama sa različitim primjenama. Senzori su omogućili mobilnim aplikacijama da se prilagode na različite okoliše te tako povećali robusnost i korisnost mobilnih apliakcija. Senzori u mobilnim aplikacijama su postali dio naših života u tolikoj mjeri da bi se teško priviknuli na život bez njih. To naravno uključuje sav zabavni sadržaj koji se rodio nakon implementacije senzora u mobilne uređaje, ali to nije sve. Implementacija senzora u mobilne uređaje donjela je brojne tehnologije u ruke ljudi diljem svijeta. Takva pristupačnost omogućuje ljudima da otkriju zanimanja za koja prije nikada ne bi imali mogućnost isprobati, primjerice poput fotografiranja. Također, senzori u mobilnim aplikacijama također mogu biti korišteni za spašavanje života. Na svjoj konferenciji u rujnu 2022. godine tvrtka Apple je predstavila nove pametne telefone serije iPhone koji pomoću različitih senzora mogu prepoznati kada se dogodila nesreća u prometu te automatski pozvati hitnu pomoć na scenu. Ovo su sve samo neki primjeri kako je implementacija senzora u mobilne uređaje promjenila naše živote i vjerojatno će ih promjeniti u još većoj mjeri u budućnosti.

5. Popis slika

Slika 1 Prikaz stabilizacije slike 10
Slika 2 Prikaz apliakcije za mjerenje pulsa 10
Slika 3 Prikaz osvjetljenog ekrana mobilnog uređaja 11
Slika 4 Prikaz aplikacije za vrijeme 11
Slika 5 Prikaz aplikacije Kompas 12
Slika 6 Prikaz aplikacije Google Maps 12
Slika 7 Prikaz aplikacije Glovo 13
Slika 8 Aplikacija za očitanje temperature (1/3) 14
Slika 9 Aplikacija za očitanje temperature (2/3) 15
Slika 10 Aplikacija za očitanje temperature (3/3) 15
Slika 11 Kod aplikacije za mjerenje temperature (1/2) 16
Slika 12 Kod aplikacije za mjerenje temperature (2/2) 17
Slika 13 Aplikacija za očitanje pokreta i rotacije (1/5) 17
Slika 14 Aplikacija za očitanje pokreta i rotacije (2/5) 18
Slika 15 Aplikacija za očitanje pokreta i rotacije (3/5) 18
Slika 16 Aplikacija za očitanje pokreta i rotacije (4/5) 19
Slika 17 Aplikacija za očitanje pokreta i rotacije (5/5) 19
Slika 18 Kod aplikacija za očitanje pokreta i rotacije (mjenjanje boja) 21
Slika 19 Kod aplikacije za očitanje pokreta (1/2) 21
Slika 20 Kod aplikacije za očitanje pokreta (2/2) 22
Slika 21 Kod aplikacije za očitanje rotacije (1/2) 22
Slika 22 Kod aplikacije za očitanje rotacije (2/2) 22
Slika 23 Aplikacija kompas na emulatoru (1/2) 23
Slika 24 Aplikacija kompas na emulatoru (2/2) 23
Slika 25 Aplikacija kompas na stvarnom uređaju 24
Slika 26 Kod aplikacije kompas (1/3) 27
Slika 27 Kod aplikacije kompas (2/3) 27
Slika 28 Kod aplikacije kompas (3/3) 27

6. Literatura

Mobile app. [Mrežni] pristup]. Dostupno: https://en.wikipedia.org/wiki/Mobile_app. [Pokušaj pristupa 10.9.2022.].

Sensor. [Mrežni pristup]. Dostupno: https://en.wikipedia.org/wiki/Sensor. [Pokušaj pristupa 10.9.2022.].

Android Sensor Tutorial. [Mrežni pristup]. Dostupno: https://www.javatpoint.com/android-sensor-tutorial. [Pokušaj pristupa 10.9.2022.].

Engineering Made Easy, Sensors in Mobile Phones Explained - Sensor Box for Android - Mobile Sensor App - Mobile Sensors. 8.1.2020. [Mrežni pristup]. Dostupno: https://www.youtube.com/watch?v=-siT5ipMm6E. [Pokušaj pristupa 10.9.2022].

Scientific American, What Sensors Are in a Smartphone?. 2.5.2018. [Mrežni pristup]. Dostupno: https://www.youtube.com/watch?v=CxC1KCoGbIM. [Pokušaj pristupa 11.9.2022].

Philipp Lackner, How to Use Device Sensors the Right Way in Android - Android Studio Tutorial. 8.5.2022. [Mrežni pristup]. Dostupno: https://www.youtube.com/watch?v=IU-EAtITRRM&t=441s. [Pokušaj pristupa 11.9.2022].

WsCube Tech, What are Sensors and How Sensor Works in Android - Android Sensor Programming. 5.3.2022. [Mrežni pristup]. Dostupno: https://www.youtube.com/watch?v=H0dxrwRT1aE. [Pokušaj pristupa 11.9.2022].

MindSea Team, Smartphone Sensors for Health and Wellness Mobile Apps: An Explainer. [Mrežni pristup]. Dostupno: https://mindsea.com/smartphone-sensors/. [Pokušaj pristupa 11.9.2022].

Silverblip Ltd - Sensor Technology and IPR Partner, Mobile Device Sensors. [Mrežni pristup] Dostupno: https://mobiledevicesensors.com/sensor-applications/. [Pokušaj pristupa 11.9.2022].

GSMARENA, Sensors – definition. [Mrežni pristup]. Dostupno: https://www.gsmarena.com/glossary.php3?term=sensors. [Pokušaj pristupa 12.9.2022].

Ivy Wigmore, smartphone sensor. 15.10.2014. [Mrežni pristup]. Dostupno: https://www.techtarget.com/whatis/definition/smartphone-sensor. [Pokušaj pristupa 12.9.2022.].