Sustav za vizualno praćenje bespilotnih letjelica koristeći proširenu stvarnost

Bakale, Toni

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Toni Bakale

SUSTAV ZA VIZUALNO PRAĆENJE BESPILOTNIH LETJELICA KORISTEĆI PROŠIRENU STVARNOST

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Toni Bakale

Matični broj: 0016135426

Studij: Poslovni sustavi

SUSTAV ZA VIZUALNO PRAĆENJE BESPILOTNIH LETJELICA KORISTEĆI PROŠIRENU STVARNOST

ZAVRŠNI RAD

Mentor :

Doc. dr. sc. Boris Tomaš

Varaždin, rujan 2021.

Toni Bakale

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad će se baviti razvojem aplikacije koje će korištenjem AR-a (eng. *augmented reality*, proširene stvarnosti) omogućiti osiguranje VLOS-a (eng. *visual lign of sight*, optička vidljivost) sa udaljenim dronom. Naime za letenje drona su potrebne najmanje dvije osobe, jedna koja će upravljati samom letjelicom i druga koja će pratiti letjelicu kako se ona ne bi izgubila iz VLOS-a, naravno ponekad se VLOS nad letjelicom izgubi i tad uskače aplikacija razvijena u sklopu ovog rada koja nam pomoću proširene stvarnosti pomaže u pronalaženju drona i njegovom daljnjem praćenju.

Ključne riječi: VLOS; Dron; AR; GPS; Android; Unity; Firebase

Sadržaj

1. Uvod

Proširena stvarnost (eng. *Augmented reality*) je tehnologija koja vuće svoje korijene još iz pedesetih godina prošlog stoljeća, ali njezini potencijali su se uvelike počeli prepoznavati u posljednjih nekoliko godina, aplikacije poput Pokemon GO-a su imale veliki utjecaj u upoznavanju javnosti sa samom tehnologijom i pridonijele njezinoj popularnosti, a u zadnje dvije godine se tehnologija počela još više razvijati zbog pandemije COVID-19 jer daje mogućnosti rada na daljinu u mnogim područjima gdje to inače ne bi bilo moguće poput operacijskih zahvata. Taj "AR boom" nije bio samo u domeni područja gdje omogućava rad na daljinu nego se širio i u druge domene pa i u domenu ovog rada, a to su bespilotne letjelice.

Ideja ovog rada je razviti aplikaciju koja će olakšati kopilotu drona praćenje dronove lokacije i njegovo pronalaženje ako izgubi vidljivost nad samim dronom. Uloga kopilota u kontekstu upravljanja dronovima je uloga promatrača (eng. *spotter*), njegov je glavni zadatak biti "drugi par očiju" pilotu, prateći dron u letu. Uz tu aplikaciju potrebno je razviti i posebnu aplikaciju za upravljanje samim dronom kako bi aplikacija za AR prikaz mogla primati podatke o lokaciji drona u stvarnom vremenu. Aplikacija za upravljanje dronom biti će modificirana open source kontroler aplikacija i biti će izrađena u alatu Android Studio za Android mobilne uređaje, aplikacija za AR prikaz biti će izrađena u alatu Unity također za Android mobilne uređaje, a komunikacija između kontrolera i aplikacije za AR prikaz biti će realizirana pomoću Firebase-a.

Rad će se sastojati od teoretske analize u kojoj će biti pojašnjene tehnologije koje će biti korištene u samom radu, to uključuje bespilotne letjelice, GPS i AR. Nakon što će osnovni koncepti vezani uz tehnologije biti objašnjeni prijeći će se na sam razvoj aplikacija, prvo će biti govora o alatima u kojima će biti razvijane same aplikacije, zatim opisi rada aplikacija, pojašnjenja nekih ključnih metoda i nakon toga prikaz rezultata testiranja.

2. Pregled literature

U ovom poglavlju se obrađuje teoretska analiza tehnologija koje obuhvaća tema rada, počevši od samih bespilotnih letjelica, detaljnije o specifičnim vrstama letjelica za koje će se raditi same aplikacije, o geolokacije pomoću GPS-a te o proširenoj stvarnosti i njezinoj primjeni.

2.1. Bespilotne letjelice

Bespilotna letjelica (eng. *Unmanned Aerial Vehicle*) je kao što joj i ime govori letjelica koja u sebi nema pilota. Bespilotne letjelice može upravljati pilot ručno koristeći kontroler ili njihov let može biti automatiziran pomoću isprogramiranih misija letova (TheUAV.com, 2021). Pojam koji se također koristi za bespilotne letjelice je dron (eng. *drone*) i biti će korišten u ostatku rada.

Dronovi se mogu dijeliti na više tipova, vojne dronove koji se koriste u oružanim sukobima, dronove za prikupljanje informacija i dronove koji služe kao mamci (isto u oružanim sukobima), zatim istraživačke dronove koji se koriste za daljnji razvoj novih tehnologija koje bi se mogle koristit u budućim dronovima i još postoje civilni i komercijalni dronovi koji će biti dalje razmatrani i korišteni u ovom radu (TheUAV.com, 2021), to su dronovi koji su javno dostupni te ih poduzeća i pojedinci mogu nabavljati i koristiti u vlastite svrhe, naravno uz praćenje svih regulativa u letenju dronova za područje gdje se nalazimo.

Konkretni dronovi za koje će aplikacije biti razvijene su ručno upravljani civilni i komercijalni dronovi, specifično dronovi proizvođača DJI. Razlog odabira njihovih dronove biti će objašnjen u sljedećem potpoglavlju.

2.1.1. DJI

DJI (eng. *Da-Jiang Inovations*) je kinesko poduzeće koje je vodeći svjetski proizvođač dronova, ima 70 posto udjela na svjetskom tržištu dronova. Sjedište im je locirano u Shenzenu, takozvanoj "Silicijskoj dolini Kine", gdje je i 2006. godine osnovano. (Xu F, Muneyoshi H, 2017)

DJI nudi velik izbor različitih modela dronova, koji su klasificirani u različite serije ovisno o njihovoj namjeni, to su Air, Phantom, FPV, Mavic serije. U ovom radu fokus će biti na Mavic seriji, specifično na dron Mavic 2 Enterprise Dual, na njemu će aplikacija biti testirana. DJI ima svoj SDK koji je javno dostupan i omogućuje razvoj aplikacija koje rade na velikoj većini DJI-evih dronova. (dji.com, 2021)

Gore navedene prednosti imale su veliki utjecaj u odabiru DJI-evih dronova kao dronova za koje će se razvijati aplikacije, budući da je 70 posto svih komercijalnih dronova DJI i njihov SDK omogućuje rad aplikacije na više različitih modela dronova bili su očiti izbor.

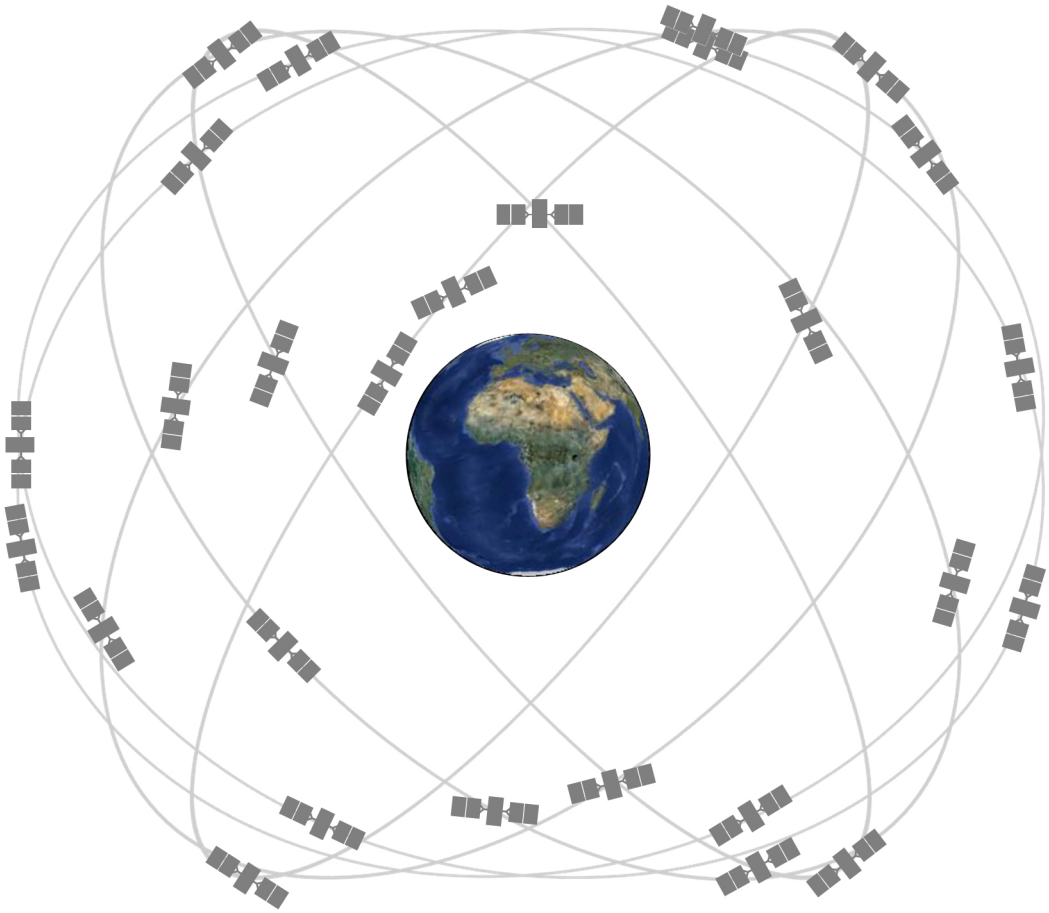
DJI Mavic 2 Enterprise Dual je dron koji će biti korišten u testiranju aplikacija, dron je opremljen sa izuzetno velikim brojem senzora, od kolizijskih senzora sa svih strana, termalne kamere... Uglavnom je namijenjen za misije spašavanja zbog toplinske kamere, dodataka po-

put zvučnika i rasvjete, no za potrebe ovog rada nam je bitan njegov sustav za pozicioniranje, tj. podaci o njegovoj lokaciji pomoću GPS-a i njegove relativne visine.

2.2. GPS

GPS (eng. *Global Positioning System*) je američki sustav za globalno pozicioniranje. Sastoji se od tri segmenta: svemirskog, kontrolnog i korisničkog. (gps.gov, 2021)

Svemirski segment čini konstelacija satelita koji šalju radio signale korisnicima na Zemlji. U Zemljinoj orbiti se nalazi 31 GPS satelit od kojih 24 trebaju biti gotovo uvijek dostupna kako bi sustav radio istovremeno bilo gdje na Zemlji, kao što je prikazano na slici dolje. (gps.gov, 2021)



Slika 1: Prikaz položaja GPS satelita u Zemljinoj atmosferi
(Izvor: https://www.gps.gov/multimedia/images/constellation.jpg)

Ovakva raspodjela omogućuje spajanje na minimalno četiri satelita neovisno o tome gdje se na Zemlji nalazimo. (gps.gov, 2021)

Sljedeći segment je kontrolni segment i njega čini mreža ustanova na Zemlji koje prate i analiziraju performanse samih satelita i šalju im naredbe i podatke. Ustanove su rasprostranjene diljem Zemlje kako bi u svakom trenutku mogli pratit sve satelite. (gps.gov, 2021)

Korisnički segment sastoji se od GPS prijemnika koji primaju signale od satelita i na

temelju njih računaju korisnikovu lokaciju. Besplatan je i javno dostupan svima, prijemnici se nalaze u raznim uređajima od kojih su za temu ovog rada bitni pametni telefoni i dronovi, koji naravno u sebi imaju GPS prijemnike. (gps.gov, 2021)

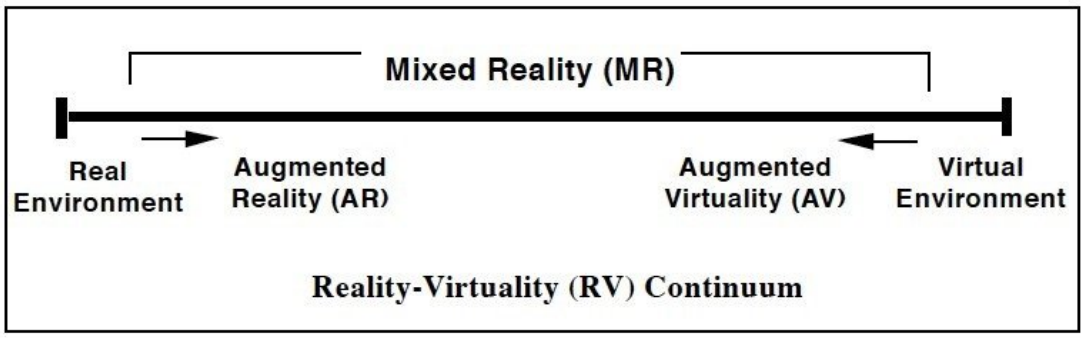
Sumirano na primjeru, naš dron u sebi ima GPS prijemnik koji prima signale od satelita, oni šalju podatke o svojoj lokaciji, statusu i preciznom vremenu kad je podatak poslan, podaci se šalju gotovo brzinom svjetlosti. Ako GPS ima podatke o lokaciji barem četiri satelita, računa svoju udaljenost od njih i na temelju toga, koristeći triangulaciju određuje svoju lokaciju na Zemlji. (gps.gov, 2021) Ta lokacija je prikazana u obliku geografskih koordinata. One su prikazane kao dva broja, jedan je geografska dužina (eng. *latitude*) koja predstavlja udaljenost neke točke od ekvatora u smjeru sjever-jug, vrijednosti se kreću između 0 i +/-90 (- za jug, + za sjever) stupnjeva, drugi je broj geografska širina (eng. *longitude*) koja predstavlja udaljenost neke točke od nultog meridijana u smjeru istok-zapad, vrijednosti se kreću između 0 i +/-180 (- za zapad, + za istok) stupnjeva. Kad korisnik ima koordinate drona i njegovu visinu, koju dron također mjeri i koordinate uređaja (visina se gleda kao relativna, tj. da je korisnik na 0, a sam uređaj na visini od 1.4 metra, u nekoj prosječnoj razini prsa) s kojim će prikazati lokaciju drona u AR-u ima sve potrebne podatke kako bi mogao prikazati približnu lokaciju drona. (ibm.com, 2019.)

2.3. Proširena stvarnost

Proširena stvarnost (eng. *Augmented reality*) može se definirati kao direktan ili indirektan pogled stvarnog fizičkog svijeta koji je proširen virtualnim računalom generiranim informacijama u stvarnom vremenu. Ne samo da omogućuje prikaz 3D i 2D virtualnih objekata nego daje i mogućnost interakcije s njima, bilo to preko npr. zaslona pametnog telefona ili kretanjem u stvarnom svijetu na mjesto prošireno virtualnim objektima. (Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M, 2011.)

Cilj proširene stvarnosti je pojednostaviti korisnikov život tako da mu pruža potrebne virtualne informacije u njegovom vizualnom okruženju te na taj način poboljšava njegovu percepciju i interakciju sa stvarnim svijetom. Najbolji primjer praktične primjene proširene stvarnosti je u raznim proizvodnim industrijama gdje radnici imaju AR naočale koje ih vode u njihovom poslu, npr. u sastavljanju automobila im označavaju gdje koji dio treba biti stavljen ili u poslu sa elektronikom daju informacije čemu koja žica služi. Također važno je reći da proširena stvarnost nije samo ograničena vizualnim aspektom kako se uglavnom koristi, ona može "proširiti" bilo koje osjetilo ili u situaciji kada neka osoba nema jedno od osjetila, npr. slijepoj osobi proširiti sluh tako da joj AR aplikacija daje zvučne signale o njihovom okruženju, ako su blizu cesta ili neke prepreke. (Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M, 2011.)

Takvim promatranjem proširene stvarnosti moguća je pomisao da je to relativno nova tehnologija, no ona vuče svoje korijene još iz pedesetih godina prošlog stoljeća kada je kinematograf Morton Heiling napravio svoj model mješovite stvarnosti koja bi bila primjenjivana u kinima budućnost. (Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M, 2011.)



Slika 2: Prikaz Heilingovog modela proširene stvarnosti

(Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M, 2011.)

Nakon njega slijedi Ivan Suthreland koji je izumio prvi uređaj za videoprikaz koji se nosi na glavi te je to zapravo bio prvi AR sustav, napravljen još 1968. godine. Tehnologija se dalje razvijala i 2000. godine je izumljena prva AR videoigra, to je bio ARQuake (Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M, 2011.), i tako sve do danas gdje postoje popularne aplikacije poput raznih videoigara koje koriste proširenu stvarnost (Ingress, Pokemon GO, Jurassic World Alive), a i sama tehnologija je integrirana u razne aplikacije poput društvenih mreža (filteri na Snapchat-u i Instagram-u). Pandemija COVID-19 je imala ogroman utjecaj u daljnjem razvoju AR tehnologije, a i samih aplikacija jer je potreba za njima postala sve veća, što u obrazovnom sustavu, što u industriji, što u medicini. Pa tako što ima mjesta za AR u svim gore navedenim industrijama ima i mjesta u upravljanju bespilotnih letjelica što je tema ovog rada. Danas postoje javno dostupni besplatni AR SDK-evi (eng. *Software development kit*) kao što su ARCore, ARKit, Vufioria, Wikitude. U aplikaciji za prikaz drona u AR-u koristit će se ARCore u kombinaciji sa AR Foundation i AR+GPS plugin-ovima u alatu Unity kako bi prikazali približnu lokaciju drona.

3. Aplikacije

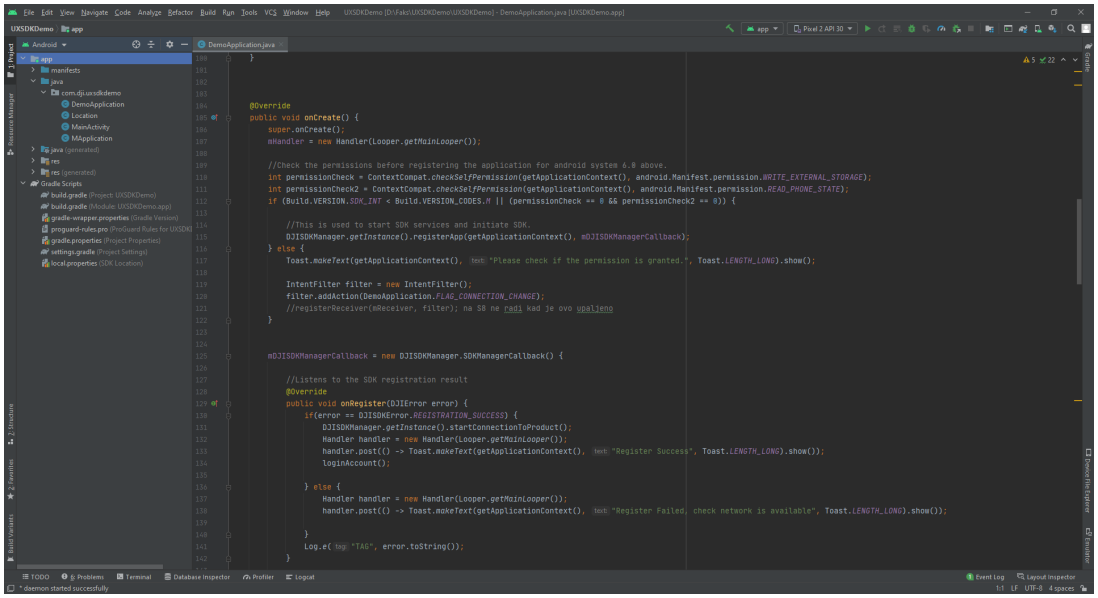
U ovom poglavlju će prvobitno biti objašnjeni alati u kojima će aplikacije biti izrađene. To su alati Android Studio i Unity. Nakon toga će biti objašnjen Firebase koji će biti korišten za komunikaciju i zatim rad samih aplikacija i njihove implementacije, koje su tehnologije bile korištene unutar samih alata (SDK-evi, pluginovi), analiza koda bitnih metoda i prikaz dizajna i rada samih aplikacija.

3.1. Razvojna okruženja

Budući da su određene tehnologije specifične za pojedino razvojno okruženje, jedna je aplikacija napravljena u Android Studiju dok je druga napravljena u Unitiy-ju, obje aplikacije su i dalje nativne Android aplikacije što znači da nema utjecaja na same korisnike aplikacija, ali su načini izrade samih aplikacija različiti zbog samih razvojnih okruženja pa će u ovom poglavlju ta okruženja biti opisana.

3.1.1. Android Studio

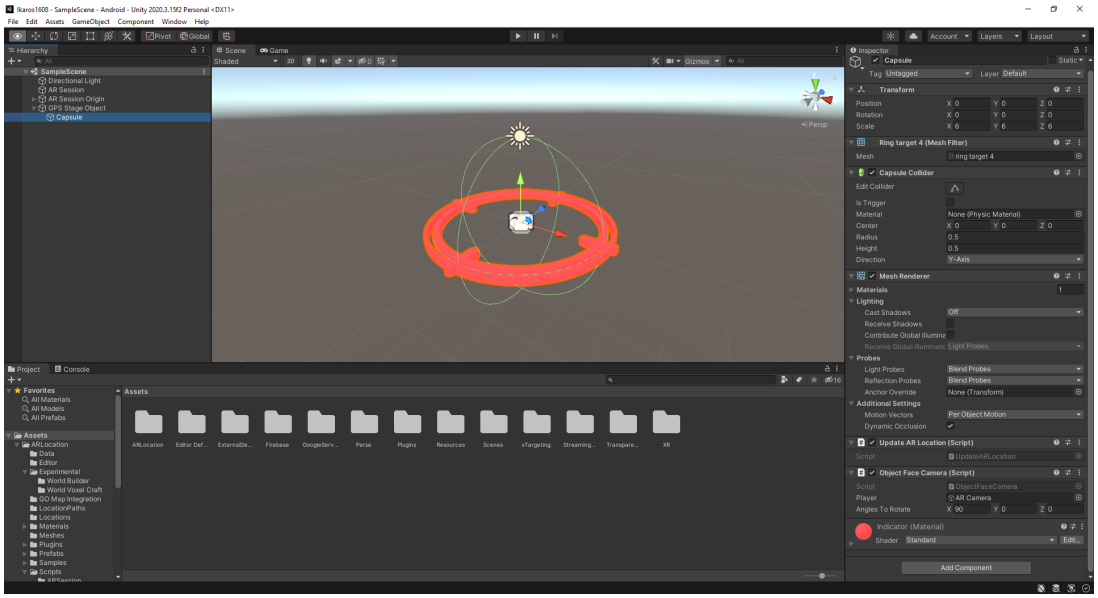
Android Studio je Google-ovo službeno integrirano razvojno okruženje koje koristi IntelliJ IDEA i IntelliJ editor koda za aplikacije namijenjene uređajima koji koriste Android operacijski sustav. Programski jezik koji primarno koristi je Java. Android Studio je najpopularniji IDE za razvoj Android aplikacija i velik broj Android aplikacija je razvijen u njemu pa tako i DJI-evi open source programi od kojih je jedan aplikacija za upravljanje samim dronom koja će biti modificirana kako bi mogla komunicirati sa aplikacijom za AR prikaz približne lokacije drona. (developer.android.com, 2021.)



Slika 3: Prikaz izgleda alata Android Studio

3.1.2. Unity

Unity je 3D/2D game engine primarno namijenjen razvoju video igara za velik broj platformi: mobilne (iOS, Android), desktop (Windows, Mac, Linxu), web, konzole (PlayStation, Xbox, Nintedno Switch) i VR/AR (Oculus, Playstation VR, ARCore, ARKit, Windows Mixed Reality). Iz navedenih primjera može se vidjeti jako širok spektar platformi za koje mogu biti razvijane aplikacije pa je tako naveden Android i ARCore, koji su ključni u razvoju same aplikacije za AR prikaz približne lokacije drona. Budući da AR koristi razne 3D modele koje zatim prikazuje pomoću kamere Unity je odličan izbor za razvoj AR aplikacija jer u sebi ima 3D model editor pomoću kojeg je jako lako uređivati same 3D objekte, također za razliku od Android Studija koji je baziran na Javi, Unity koristi C# za pisanje skripti koje se zatim dodjeljuju samim 3D objektima. (unity.com, 2021) Također ima velik broj plugin-ova koji pojednostavljuju rad u alatu pa tako i imaju plugin-ove za rad sa AR-om. Plugin-ovi koji će se koristit u AR aplikaciji su AR Foundation i AR+GPS.



Slika 4: Prikaz izgleda alata Unity

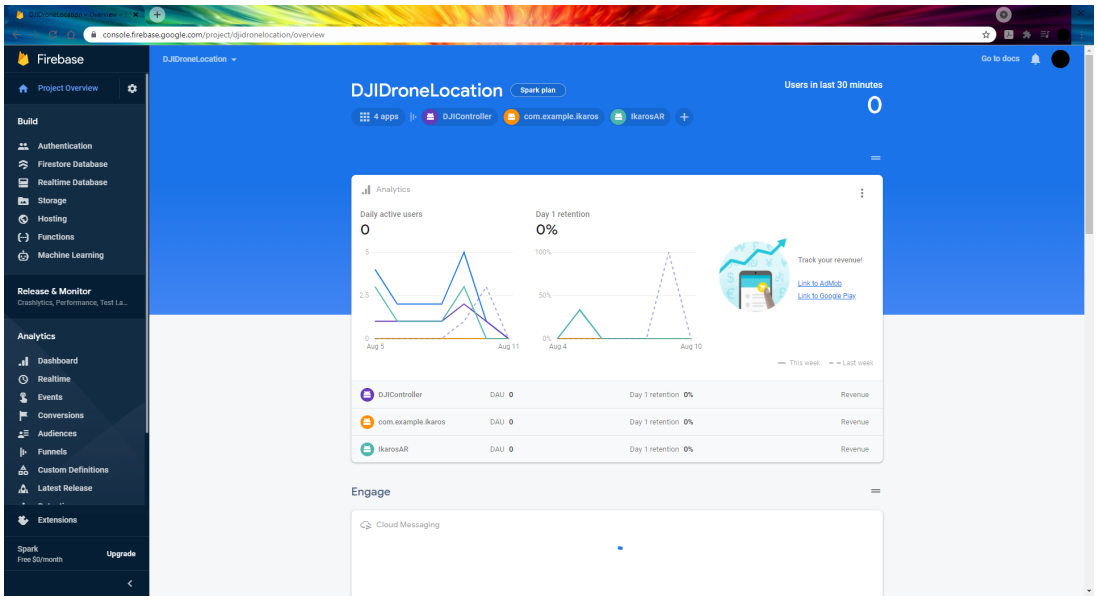
3.2. Komunikacija između aplikacija

Jedna aplikacija prikuplja podatke, a druga ih prima i obrađuje stoga je potrebno uspostaviti komunikaciju između aplikacija. Klasične relacijske baze podataka nisu najbolji izbor za ovakvu razmjenu podataka jer je ključna brzina. Također sami podaci koji se trebaju slati su relativno jednostavni, 3 decimalna broja pa je zbog ta dva faktora odlučeno da će se koristit Real Time Database Firebase-a za komunikaciju između aplikacija.

3.2.1. Firebase

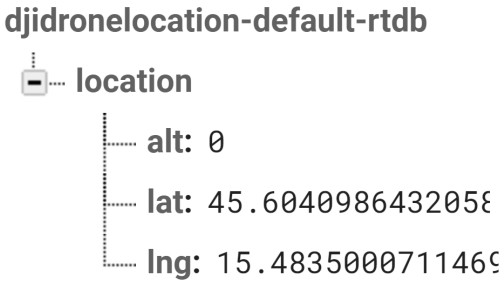
Firebase je platforma koja pomaže u razvoju mobilnih i web aplikacija. Pomoću Firebase-ove web konzole je moguće stvoriti Firebase projekt koji nudi niz usluga, primarno vezanih uz

pohranu podataka od kojih je za ovaj rad najbitniji Realtime Database. Na Firebase projekt se može spojiti niz aplikacija sa različitih platformi (Android, iOS, Web i Unity), pa će se tako spojiti i obje aplikacije i komunicirati pomoću Firebase-a. Za razliku od klasičnih relacijskih baza podataka Firebase ne koristi SQL, već podatke sprema u takozvanim JSON datotekama. (firebase.google.com, 2021.)



Slika 5: Prikaz izgleda Firebase konzole

Ovdje je prikazana struktura podataka u Realtime Database-u koju koriste obje aplikacije za komunikaciju.



Slika 6: Prikaz izgleda strukture podataka korištene u aplikacijama

Kao što se može vidjeti na slici, struktura je poprilično jednostavna i intuitivna. Firebase koristi kolekcije kako bi organizirao podatke u JSON-u, u njoj s nalazi kolekcija location u kojoj su spremljeni podaci o visini drona, njegovoj geografskoj širini i dužini. Svaki put kad se lokacija drona promijeni struktura se ne briše nego se promijene vrijednosti podataka u njoj.

3.3. Implementacija

U ovom će poglavlju biti opisan rad samih aplikacija, njihova implementacija u odgovarajućim okruženjima, analiza ključnih metoda i prikaz dizajna. Prvo će biti govora o DJI

Controller-u, modificiranoj aplikaciji za upravljanje dronom a zatim o Ikaros-u, aplikaciji za AR prikaz približne lokacije drona.

3.3.1. DJI Controller

DJI Controller je nativna Android aplikacija koja omogućuje spajanje Android mobilnog uređaja USB konekcijom na kontroler drona te na taj način omogućuje: pogled iz dronove kamere, provjere baze podataka o područjima u kojima je zabranjeno letenje, upozorenja o obližnjim letjelicama, kalibraciju GPS-a i IMU-a, automatsko polijetanje i slijetanje te niz drugih funkcionalnosti koje su dostupne u aplikacijama poput DJI Pilot ili DJI GO. Te aplikacije su temelj prema kojima je i napravljen UXSDKDemo, open source aplikacija čiji je kod javno dostupan na DJI-evom Github repozitoriju dostupnom na sljedećoj poveznici:

https://github.com/DJI-Mobile-SDK-Tutorials/Android-UXSDKDemo

UXSDKDemo pruža sve gore navedene funkcionalnosti i potpuno je opremljena aplikacija za let DJI dronovima. To daje veliku prednost u razvoju DJI Controller-a jer aplikacija ne mora biti razvijana iz početka već je dovoljno modificirati kod UXSDKDemo-a.

3.3.1.1. Analiza koda

Kod će biti modificiran na 2 ključna načina. Prva modifikacija je preuzimanje podataka o lokaciji drona u stvarnom vremenu. Naime, DJI-ev SDK omogućuje prikazivanje, preuzimanje i manipuliranje raznim podatcima o dronovom statusu pa tako i o njegovoj lokaciji. Ta lokacija podrazumijeva dronove 3D koordinate koje uključuju njegovu geografsku dužinu, širinu i visinu. Geografske koordinate su već bile objašnjene u poglavlju o GPS-u ali visina nije, ona se dobiva na 2 načina, koristeći geolokacijske satelite i koristeći visinske senzore u dronu, oboje se koristi pri određivanju visine ali ona ima veliki problem, generalno se gleda relativna visina u smislu 0 je visina s koje je dron poletio, što je problem u konkretnoj aplikaciji jer se može dogoditi teoretska situacija gdje će dron poletjeti sa veće ili manje nadmorske visine od lokacije kopilota, a kopilotova AR aplikacija će također uzeti mjesto gdje on stoji kao nulu i doći će do nepoklapanja visina pa to treba imati na umu pri testiranju aplikacija.

Biti će napravljena metoda za preuzimanje podataka iz drona i nova klasa lokacija koja će u sebi imati definirane podatke o dronovoj lokaciji. Sljedeća modifikacija je slanje samih podataka u Firebase, potrebno je stvoriti konekciju na Firebase i napraviti konstantan output podataka o lokaciji drona u stvarnom vremenu u Firebase. Sve to će se događati u pozadini tako da sam korisnik neće direktno vidjeti ili primijetiti razliku u radu UXSDKDemo i DJI Controller aplikacija.

Preuzimanje podataka o lokaciji drona je implementirano na sljedeći način:

```
private FlightController mFlightController;

private double droneLocationLat = 50;
private double droneLocationLng = 130;
private double droneLocationAlt = 20;


public void initFlightController(){


        BaseProduct product = DemoApplication.getProductInstance();
        if(product != null && product.isConnected()){
                if(product instanceof Aircraft){
                        mFlightController = ((Aircraft) product).
                            getFlightController();
                }
        }

        if(mFlightController!=null){
          mFlightController.setStateCallback(
                djiFlightControllerCurrentState -> {
                        droneLocationLat = djiFlightControllerCurrentState.
                            getAircraftLocation().getLatitude();
                        droneLocationLng = djiFlightControllerCurrentState.
                            getAircraftLocation().getLongitude();
                        droneLocationAlt = djiFlightControllerCurrentState.
                            getAircraftLocation().getAltitude();
                        updateLocation();
                }
          );
        }
}
```

Prvo je potrebna metoda initFlightController(), u njoj se zapravo dolazi do podataka o lokaciji drona. Prvo treba stvoriti instancu proizvoda (drona). Zatim provjeriti uspješnost tako da se testira je li vrijednost product-a null i je li proizvod spojen, ako je onda se instancira mFlightController kako bi se moglo doći do podataka o dronu koje dron šalje kontroleru. Nakon toga se provjerava je li mController uspješno spojen tako da se provjerava je li njegova vrijednost različita od null, ako je onda se pomoću djiFlightControllerCurrentState-a preuzimaju trenutni podatci o dronu iz kontrolera te se dobivaju željeni podatci pozivanjem odgovarajućih metoda djiFlightControllerCurrentState-a (npr. getAltitude() za visinu) i pohranjuju u prije definirane varijable. Zatim pozivamo metodu updateLocation() koja komunicira sa Firebase-om. Ona će biti objašnjena sljedeća.

```
    public static boolean checkGpsCoordinates(double latitude, double longitude){
        return (latitude > -90 && latitude < 90 && longitude > -180 && longitude <
            180) && (latitude != 0f && longitude != 0f);
        }

public void updateLocation(){

        if (checkGpsCoordinates(droneLocationLat, droneLocationLng)) {

                Location location = new Location(droneLocationLat,droneLocationLng,
                    droneLocationAlt);

                myRef.child("location").setValue(location);

        }
}
```

Metoda updateLocation() komunicira sa Firebase-om i pohranjuje podatke u bazu, prvo provjerava ispravnost podataka, geografska duljina i širina imaju spektar vrijednosti koje mogu poprimiti i metoda checkGpsCoordinates() provjerava njihovu ispravnost, ako su ispravni stvara se novi objekt lokacije sa podacima dobivenim u initFlightController() i nakon toga se u bazu spremaju podaci o lokaciji pomoću setValue() u kolekciju location o kojoj je već bilo riječi u poglavlju o Firebase-u. Također je bitno istaknuti da je myRef instanca Firebase baze, ona je instancirana na kraju onCreate() metode na sljedeći način:

```
myRef = FirebaseDatabase.getInstance("https://djidronelocation-default-rtdb.europe-
    west1.firebasedatabase.app").getReference();
```

Dohvaća se referenca na bazu pomoću getReference(), u ovom specifičnom programu se spajanje na Firebase nije moglo napraviti preko Android Studijevih čarobnjaka već je konekcija trebala biti uspostavljena ručno pa je točna adresa baze vidljiva kao ulazni argument metode, a inače to Android Studio radi automatski.

Struktura klase Location:

```
package com.dji.uxsdkdemo;

public class Location {
        double lat, lng, alt;

        public Location(double lat, double lng, double alt) {
                this.lat = lat;
                this.lng = lng;
                this.alt = alt;
        }

        public Location(){

        }
```

```
        public double getLat() {
                return lat;
        }

        public void setLat(double lat) {
                this.lat = lat;
        }

        public double getLng() {
                return lng;
        }

        public void setLng(double lng) {
                this.lng = lng;
        }

        public double getAlt() {
                return alt;
        }

        public void setAlt(double alt) {
                this.alt = alt;
        }
}
```

Preostali korak je slanje podataka u bazu u stvarnom vremenu, to je ostvareno pomoću Handlera čiji je kod opisan dolje.

```
private final Handler locationHandler = new Handler();
private final Runnable locationRunnable = new Runnable() {
        @Override
        public void run() {

                initFlightController();
                locationHandler.postDelayed(this, 100);
        }
};
```

Ovdje postavljamo Handler koji svakih 100 milisekundi poziva initFlightController(), te na taj način svakih 100 milisekundi šalje dronovu lokaciju Firebase-u.

3.3.1.2. Dizajn aplikacije

Dizajn same aplikacije je vrlo sličan dizajnu DJI Pilot aplikacije, Na slikama dolje može se vidjeti jednu ispod druge sliku DJI Pilota i DJI Controllera.

Slika 7: DJI Pilot

Slika 8: DJI Controller

Kao što se može vidjeti UI aplikacija je vrlo sličan te su razlike vrlo malene, najveća je razlika u tome što DJI Pilot u doljnjem lijevom kutu ima minimapu dok DJI Controller ima samo prikaz pomoću kompasa i DJI Pilot zauzima prostor cijelog ekrana dok je DJI Controller kompaktniji.

Time je izrađena funkcionalna aplikacija za upravljanje dronom koja u stvarnom vremenu šalje podatke o njegovoj lokaciji. Sljedeći korak je implementacija aplikacije za AR prikaz lokacije dobivene DJI Controller-om.

3.3.2. Ikaros - Aplikacija za AR prikaz lokacije drona

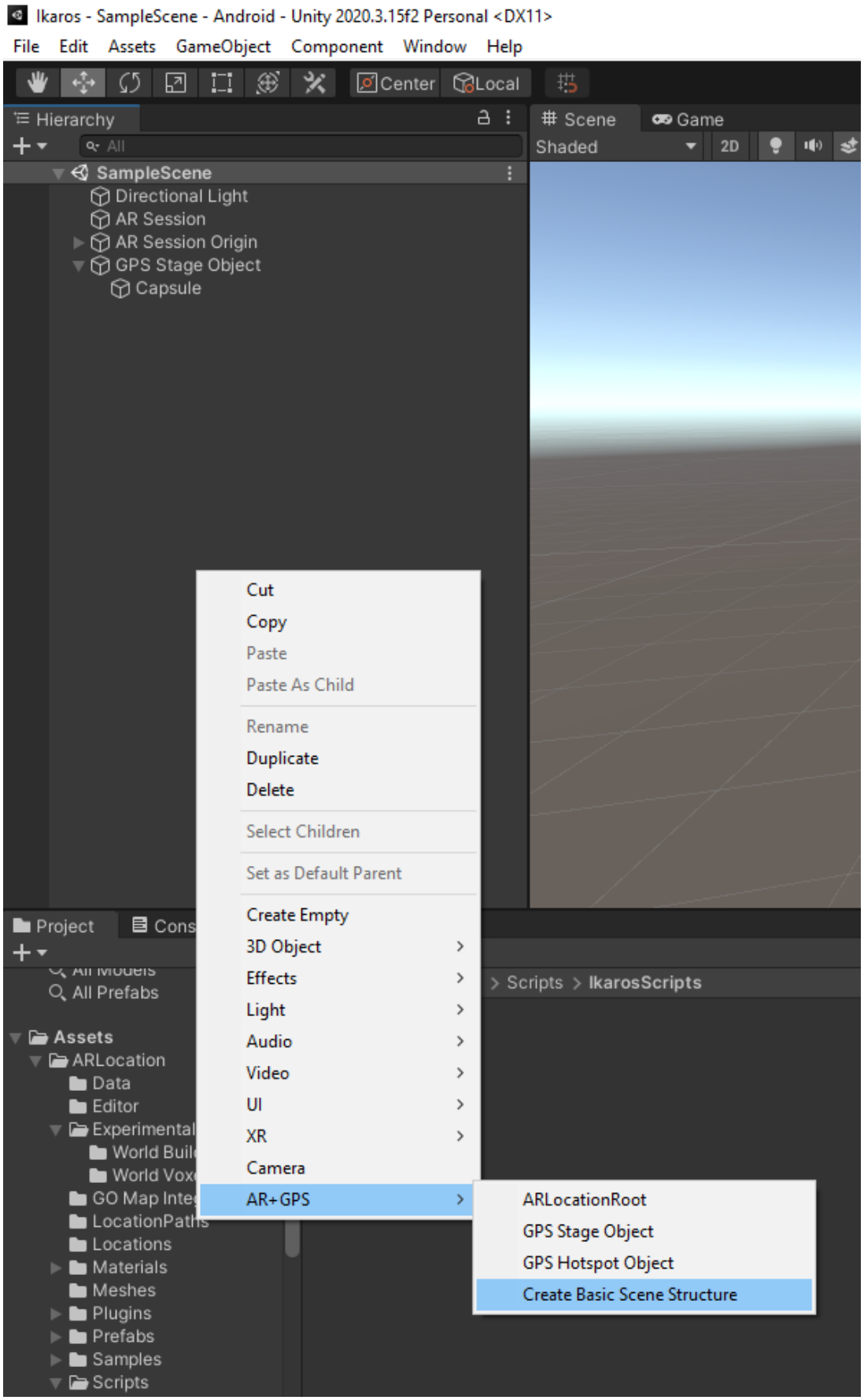
Ikaros je nativna Android aplikacija koja omogućuje prikaz približne lokacije drona koristeći proširenu stvarnost. Aplikacija se spaja na Firebase i u stvarnom vremenu preuzima podatke o lokaciji drona i pomoću njih i lokacije samog Android uređaja koji se koristi za AR prikaz računa i prikazuje lokaciju drona pomoću 3D objekta koji predstavlja dronovu lokaciju. Uređaj ima upaljenu stražnju kameru i ovisno o prije navedenim parametrima i orijentaciji ka-

mere program računa približnu lokaciju drona i prikazuje ju pomoću crvenog nišana u čijem se okruženju nalazi dron.

Aplikacija je izrađena u alatu Unity koji pruža niz tehnologija za razvoj AR aplikacija. Dvije glavne tehnologije korištene u ovom radu su AR Foundation i AR+GPS plugin-ovi koji u sebi imaju implementirane metode za crtanje AR objekata pomoću podataka o njihovim geografskim dužinama, širinama i relativnoj visini. Doduše one su namijenjene primarno za prikazivanje stacionarnih objekata čija je lokacija fiksna ili objekata koji se kreću, ali po zadanoj ruti. Ikaros treba prikazivati te podatke u stvarnom vremenu i uz promjenu lokacije samog drona i uređaja koji prikazuje AR prikaz, stoga je potrebno napraviti posebnu skriptu koja prikuplja podatke o lokaciji drona iz Firebase-a u stvarnom vremenu i osvježava lokaciju AR prikaza drona. Tu skriptu zatim treba dodijeliti odgovarajućem 3D objektu za prikaz približne lokacije drona. Dizajn samog objekta za prikaz lokacije drona je napravljen u samom Unity editoru dok je skripta, za razliku od koda DJI Controller-a koji je napisan u Javi, napisana u C#-u koristeći Visual Studio jer Unity nema svoj editor već koristi Visual Studio.

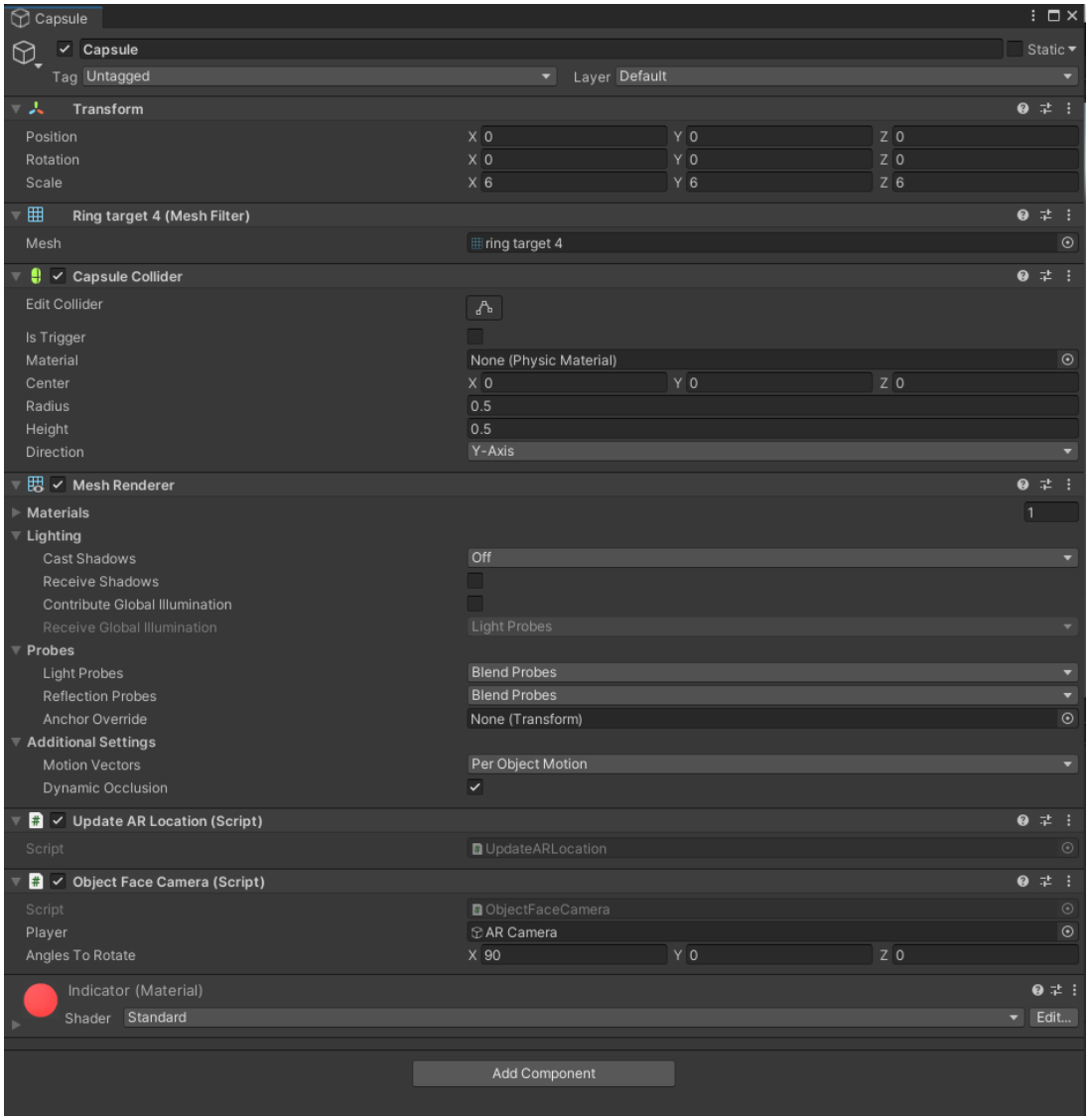
3.3.2.1. Analiza koda

Prvi korak kod izrade aplikacije je dodati sve potrebne plugin-ove u Unity projekt, ručno preko Assets > Import Package > Custom Package ili preko Window > Package Manager. Nakon što su AR Foundation, AR Core XR Plugin (budući da se razvija aplikacija za Android, kad bi se aplikacija razvijala za iOS sve bi bilo isto, samo bi se koristio AR Kit XR Plugin) i AR+GPS dodani potrebno je ispraviti greške tako da se promijeni API razina i minimalna verzija Androida na minimalno Android 7 jer prijašnje verzije ne podržavaju AR Core. Nakon što je sve postavljeno u hijerarhiji je potrebno obrisati Main Camera i dodati Basic Scene Structure desnim klikom kao na slici dolje. (docs.unity-ar-gps-location.com, 2020)



Slika 9: Dodavanje bazične AR+GPS strukture

Time su osnovne stvari spremne i može se mijenjati dizajn 3D objekta tako da se pod GPS Stage Object > Capsule klikne desni klik > Properties i mijenjaju parametri poput veličine, teksture modela, skripti koje se odnose na njega. Prikazano dolje.



Slika 10: Uređivanje AR objekta

Nakon što je uređivanje AR objekta gotovo potrebno je spojiti Unity projekt sa Firebase-om, slično kao i kod DJI Controller-a, potrebno je dodati aplikaciju putem konzole i pratiti upute za dodavanje preostalih resursa u Unity projekt, to uključuje preuzimanje google-services.json-a i dodavanje njega u resurse projekta te instaliranje Firebase plugin-a, kao što je prije instalirani AR+GPS Plugin. Ostatak implementacije Firebase-a je sličan kao i u DJI Controller-u i biti će objašnjen u kodu skripte, koji je prikazan dolje.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ARLocation;
using Firebase;
using Firebase.Database;
using Firebase.Extensions;

public class UpdateARLocation : MonoBehaviour
{

        double lat = 0;
        double lng = 0;
        double alt = 0;

        void Start()
        {
                var loc = new Location()
                {
                        Latitude = 50,
                        Longitude = 70,
                        Altitude = 2,
                        AltitudeMode = AltitudeMode.GroundRelative
                };

                var opts = new PlaceAtLocation.PlaceAtOptions()
                {
                        HideObjectUntilItIsPlaced = true,
                        MaxNumberOfLocationUpdates = 0,
                        MovementSmoothing = 0.1f,
                        UseMovingAverage = false
                };

                PlaceAtLocation.AddPlaceAtComponent(gameObject, loc, opts);

                DatabaseReference reference = FirebaseDatabase.DefaultInstance.
                    RootReference;
        }

        void Update()
        {
                GetData();
        }

        void LocationUpdate(double lat,double lng, double alt)
        {
                var newLocation = new Location()
                {
                        Latitude = lat,
                        Longitude = lng,
                        Altitude = alt,
                        AltitudeMode = AltitudeMode.GroundRelative
```

```
            };

            var placeAtLocation = GetComponent<PlaceAtLocation>();
            placeAtLocation.Location = newLocation;
        }



        void GetData()
        {
            FirebaseDatabase.DefaultInstance.GetReference("location").
                GetValueAsync().ContinueWithOnMainThread(task => {
                    if (task.IsFaulted)
                    {

                    }
                    else if (task.IsCompleted)
                    {
                        DataSnapshot snapshot = task.Result;

                        lat = double.Parse(snapshot.Child("lat").Value.
                            ToString());
                        lng = double.Parse(snapshot.Child("lng").Value.
                            ToString());
                        alt = double.Parse(snapshot.Child("alt").Value.
                            ToString());

                        LocationUpdate(lat, lng, alt);


                    }
            });
        }
}
```

C# skripte u Unitiy-ju imaju dvije glavne metode, jedna je Start() koja se izvršava pri pokretanju skripte i druga je Update() koja se poziva u svakom frame-u, u ovoj skripti su nam obje metode bitne i biti će korištene. Biblioteke koje su korištene i koje je važno istaknuti su ARLocation iz kojeg se zovu metode za manipuliranje AR objektom i Firebase koje služe za komunikaciju sa bazom.

Prva stvar koja se događa u Start() metodi je stvaranje nove lokacije AR objekta loc, koji predstavlja novu lokaciju, njezini atributi su namjerno, u svrhu testiranja, postavljeni tako da lokacija bude "na drugom kraju svijeta" kako bi sa sigurnošću bilo provjereno uspostavljanje konekcije sa bazom jer će se kasnije ta lokacija osvježiti i biti u vidokrugu kamere. Zatim opts predstavlja opcije samog AR objekta, jako je bitno napomenuti da je atribut MaxNumberOfLocationUpdates postavljen na 0, kad je postavljen na 0 to znači da će se lokacija AR objekta moći osvježiti neograničen broj puta što je jako bitno jer se lokacija našeg AR objekta konstantno mijenja.

Nakon što su loc i opts postavljeni poziva se PlaceAtLocation.AddPlaceAtComponent(),

čiji su argumenti gameObject - koji predstavlja objekt na kojem je postavljena skripta, loc i opts i stavlja objekt na definiranu lokaciju.

Sljedeća stvar koja se izvršava je instanciranje reference na Firebase kako bi se mogla uspostaviti komunikacija sa Firebase bazom.

Nakon toga se prelazi na Update() metodu u kojoj se poziva metoda GetData(). GetData() se spaja na Firebase, ako je uspješno spojena vraća DataSnapshot trenutnog stanja u bazi, koji se onda može pretraživati i parsirati dok se ne dobiju željeni podatci i zatim se spremaju u globalne varijable lat, lng i alt te se zatim poziva metoda LocationUpdate() čiji su ulazni argumenti te varijable.

LocationUpdate() prvo stvara novu lokaciju, poput loc-a koji je bio objašnjen ranije sa dobivenim podatcima o lokaciji iz baze. Nakon toga se dohvaća trenutna lokacija AR objekta i osvježava mu se lokacija (opts se ne dira, on je isti kao prije).

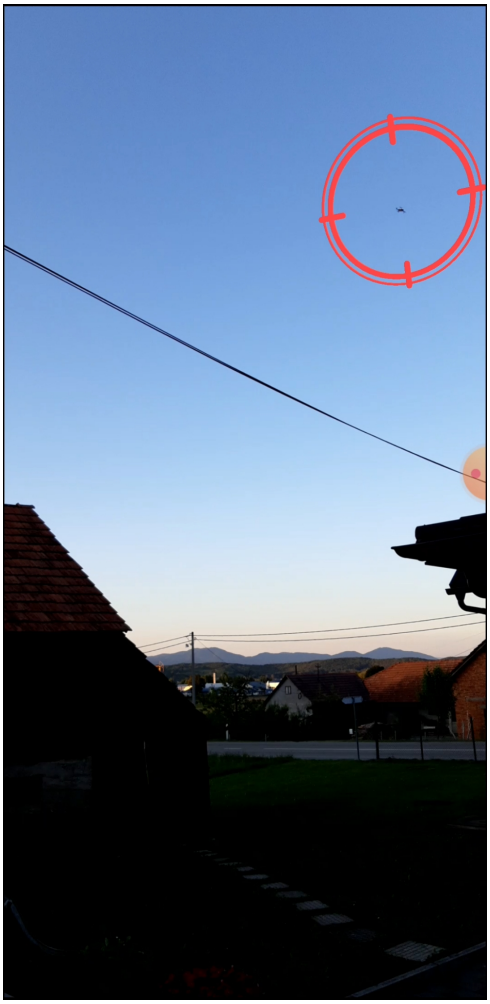
Budući da aplikacija koristi nišan za prikaz približne lokacije drona potrebno ga je rotirati tako da uvijek gleda u kameru pravom stranom. To je ostvareno u skripti čiji je kod prikazan dolje.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObjectFaceCamera : MonoBehaviour
{
        public GameObject player;


        public Vector3 anglesToRotate;

        void Start()
        {

        }


        void Update()
        {
                transform.LookAt(player.transform);
                this.transform.Rotate(anglesToRotate);

        }
}
```

Potrebno je instancirati GameObject koji ne predstavlja nišan već predstavlja objekt oko kojeg će se nišan rotirati, a to je u ovom slučaju AR Camera, budući da je atribut public objekt se može odrediti preko Unity editora. Zatim se definira Vector3 koji će biti korišten za rotaciju nišana oko svoje osi. U Update() metodi se poziva metoda LookAt() koja rotira nišan u smjeru kamere, no ta rotacija nije idealna pa se još metodom Rotate() korigira orijentacija ovisno o podacima postavljenim u Unity editoru.

3.3.2.2. Dizajn aplikacije

Što se tiče dizajna aplikacije, vrlo je jednostavan, pri paljenju aplikacije se pali kamera i AR objekt se prikazuje na približnoj lokaciji drona. Prikazano na slici dolje.



Slika 11: Prikaz rada Ikarosa

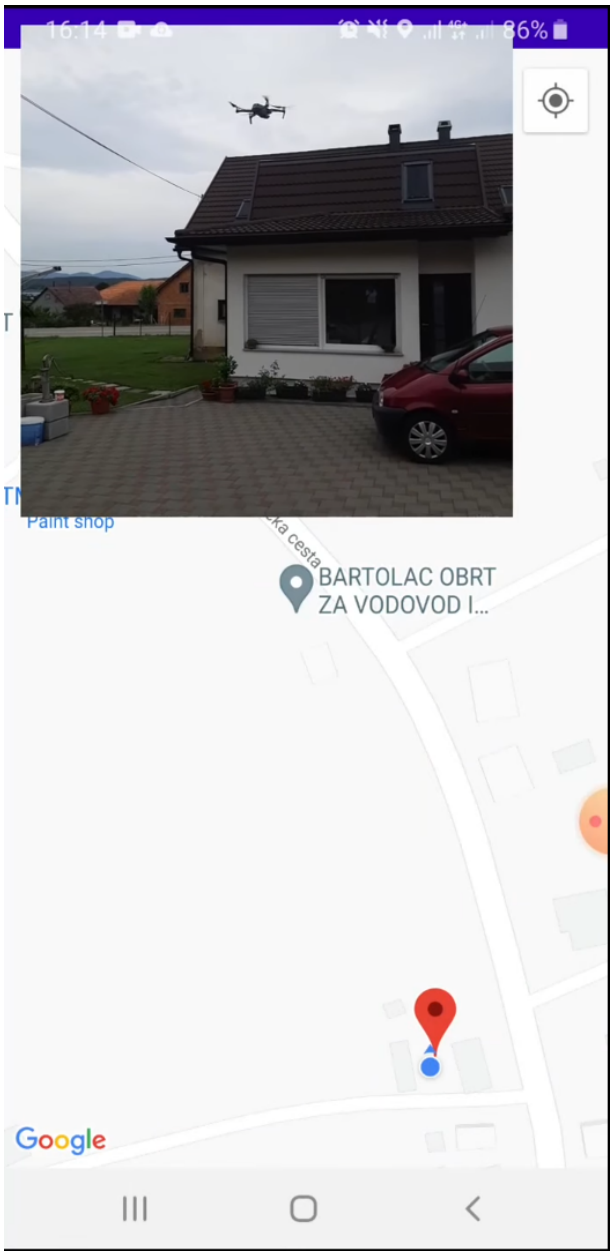
4. Testiranje

Sad kad je rad aplikacija detaljno objašnjen možemo govoriti o procesu testiranja aplikacija. Najzahtjevniji dio testiranja aplikacija je bio taj što se nisu mogle testirati u emulatorima, nego vani na stvarnim uređajima, to je naravno i otežalo sam proces debuggiranja ali je i potaknulo kreativnost u rješavanju problema, koristite ono što vam je dano.

Prvo testiranje je bilo testiranje UXSDKDemo aplikacije, trebalo je provjeriti radi li open source DJI kontroler dobro kao i službeni kontroleri, način na koji je to bilo testirano je paralelnim radom u USDKDemo aplikaciji i DJI Pilot aplikaciji tako što su provedene provjere GPS-a, IMU-a u DJI Pilot aplikaciji, zatim je napravljen test ručnog polijetanja i slijetanja koristeći DJI Pilot aplikaciju i kada je bilo ustanovljena uspješnost, isto testiranje je provedeno i na UXSDKDemo-u. Nakon što je rad UXSDKDemo-a bio uspješno provjeren sljedeći korak je bio prikaz podataka o lokaciji drona, najlakši način prikaza je bio preko Toast poruka, to su pop-up poruke na Android uređajima, provedeni su testni letovi u kojima bi unutar Toast poruka podaci o geografskoj širini, dužini i visini drona bili prikazivani te bi se pratile njihove promjene. Kada je točnost podataka bila provjerena bilo je potrebno prijeći na sljedeći korak, a to je uspostavljanje komunikacije.

Za to je odlučeno korištenje Firebase-a, ali spajanje na njega nije bilo najjednostavnije, naime UXSDKDemo u svojem kodu ima neka upozorenja, ona ne utječu na rad same aplikacije već samo upozoravaju da će neke stvari uskoro zastarjeti, nažalost Firebase to ne voli i ne želi se spojiti na klasičan način pomoću alata za spajanje koji su ugrađeni u Android Studio. Prvi pokušaj rješavanja tog probleme je bio riješiti upozorenja, ali UXSDKDemo nije radio bez njih pa je bilo potrebno uzeti u obzir alternative za komunikaciju, no prije toga je pokušano uspostaviti ručnu konekciju na Firebase i pokušaj je bio uspješan. Konekcija je bila testirana tako da je na jednom računalu bila otvorena Firebase konzola i snimana za vrijeme testnog leta. Nakon testa je bilo potrebno analizirati snimku i promjene unutar baze koje su na njoj bile prikazivane.

U tom trenutku tehnologija za AR prikaz nije bila odlučena pa je napravljena testna aplikacija. Koja radi sličnu stvar, ali umjesto AR prikaza prikazuje lokaciju drona na Google karti.



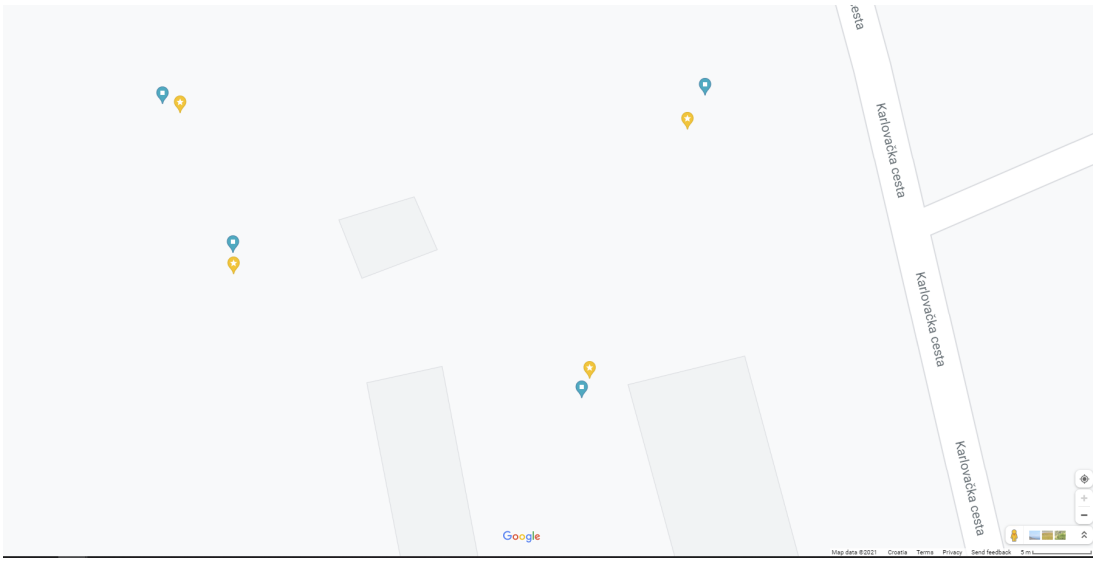
Slika 12: Prikaz testiranja komunikacije pomoću Google Maps-a

Plavi marker predstavlja Android uređaj dok crveni marker predstavlja lokaciju drona, crveni marker je postavljen na lokaciju drona i njegova lokacije se osvježava u stvarnom vremenu, isto tako se osvježava i lokacija Android uređaja koji prikazuje kartu. Test je bio uspješan, jedina mana ovakvog prikaza je ta da se ne može prikazati visina drona, stoga je sljedeći korak bio razvoj AR aplikacije.

Prvo testiranje AR prikaza je bilo sa ručno unesenim koordinatama, nakon što su testirani ručni prikazi aplikaciju je bilo potrebno spojiti na bazu i napraviti prvi testni let sa AR prikazom prikazan na slici dolje.

Slika 13: Prikaz prvog testiranja AR prikaza

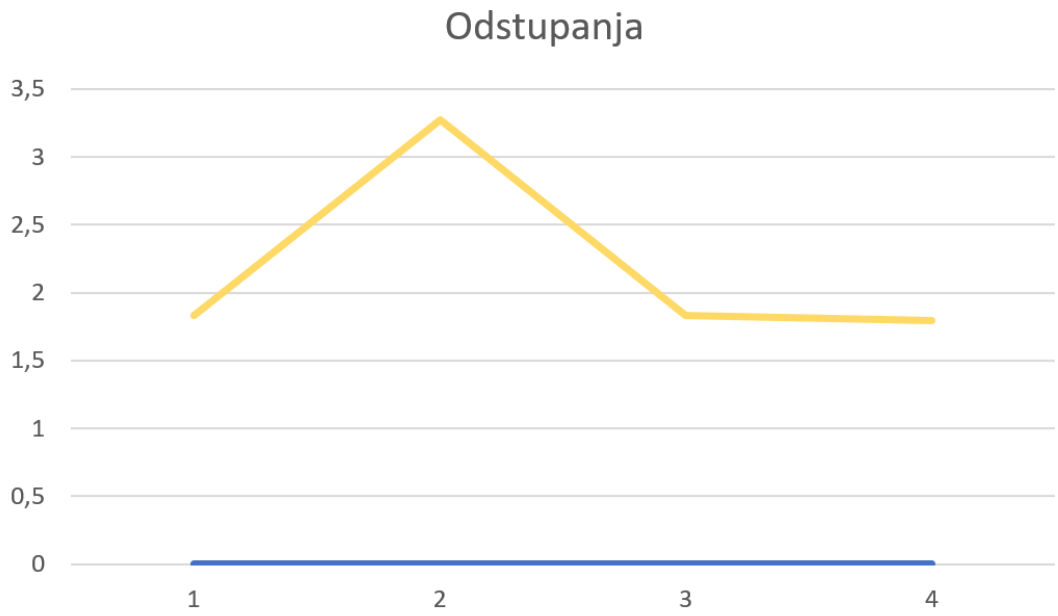
Rezultati testiranja su bili obećavajući ali se problem nepreciznosti GPS tehnologije i kalkuliranja približne lokacije drona očigledno vidio, optimiziranje je bilo potrebno, nažalost postoje tehničke limitacije. Provedeno je ručno testiranje odstupanja tako da su tri osobe testirale preciznost, jedan osoba je koristila mobitel za AR prikaz kako bi navodila drugu osobu gdje se treba stati, kad se druga osoba stala na lokaciju AR prikaza prva osoba bi joj rekla da stane i druga osoba bi zabilježila svoje GPS koordiante, a treća bi osoba napravila isto sa podatcima lokacije drona u bazi. Taj je proces ponovljen nekoliko puta i nakon toga su ti podaci bili ručno nacrtani kao markere na Google karti kao što možete vidjeti na sljedećoj slici.



Slika 14: Testiranje odstupanja

Mjerenje	1	2	3	4
Dron-LAT	45,6041411054224	45,6043334231526	45,6042217072224	45,6043454637738
Dron-LNG	15,4834945538975	15,4836025236777	15,4831013815186	15,4830431440745
AR-LAT	45,6041259	45,6043596	45,6042382	45,6043528
AR-LNG	15,4834856	15,4836218	15,4831012	15,4830226
Odstupanje	**1,829**	**3,274**	**1,834**	**1,794**
Smjer odstupanja	***sjeverozapadno***	***jugoistočno***	***sjeverno***	***sjeverozapadno***

Tablica 1: Tablični prikaz odstupanja



Graf 1: Grafički prikaz odstupanja

Plavi markeri prikazuju lokacije drona, a žuti lokacije AR prikaza. Također su u tablici ispod slike su brojčano prikazana odstupanja i nakon njih grafički pomoću grafa. Najveći problem

je u tome što su odstupanja svestrana, kad bi odstupanja uvijek bila u jednom smjeru, dijagonalna, mogla bi se jednostavno ručno korigirati, odstupanja su nakon doradivanja aplikacije u zadovoljavajućim granicama, u prosjeku oko 2 metra.

Sa tim rezultatima je postignut precizniji prikaz, koji se može vidjeti na slici primjera dizajna aplikacije i najbolje u videu. Link videa testiranja:

https://www.youtube.com/watch?v=-w096W7VNE0

Rad aplikacije u videu je puno bolji, no unaprijeđenja su uvijek moguća, pogotovo u preciznosti i brzini.

5. Budući rad

Kao što je u testiranju navedeno, uvijek postoji mjesta za napredak i aplikacija može biti usavršavana na razne načine.

Prvobitno bi trebalo poboljšati njezinu preciznost, prikaz ponekad zna biti dosta neprecizan, brzina je isto bitan faktor, sam AR prikaz zaostaje za dronom ako on putuje velikom brzinom. Potrebno je uspostaviti bržu komunikaciju između aplikacije, po mogućnosti direktnu, putem Bluetootha-a ili WiFi-a. Preciznost lokacije drona može biti poboljšana preuzimanjem podataka o IMU senzoru i prikazati kopilotu još neke korisne informacije poput smjera kretanja i upozorenja o prekidu VLOS-a nad samim dronom.

Uz samo poboljšavanje performansi aplikacije moguće ju je i proširiti dodavanjem mogućnosti praćenja više dronova od jednom, naravno to bi imalo utjecaj na samu brzinu prikaza pa bi se ona prvo trebala usavršiti. Uspješnom implementacijom gore navedenih funkcionalnosti aplikacija bi mogla biti spremna za svoju produkcijsku verziju. Također Ikaros ne treba nužno biti korišten samo za dronove. U teoriji on može biti korišten za AR praćenje bilo kojeg uređaja koji može slati podatke o svojoj geolokaciji i to otvara niz mogućnosti za slične aplikacije.

U sljedećim potpoglavljima će detaljnije biti opisane same funkcionalnosti koje bi mogle biti dodane u aplikaciju kako bi se njezin rad usavršio.

5.1. Prikaz više informacija o statusu drona

Aplikacija za sad ima vrlo jednostavan prikaz približne lokacije pomoću kamere i crvenog nišana koji prikazuje približnu lokaciju. Za njezinu osnovnu funkcionalnost to je dovoljno, no postoje informacije koje su kopilotu korisne, a ne mogu se vidjeti u trenutnom jednostavnom prikazu. One uključuju smjer kretanja drona, njegovu brzinu kretanja, orijentaciju. Sve te podatke moguće je prikazati na ekranu, naravno potrebno je uzeti u obzir da njihov prikaz ne narušava rad glavne funkcionalnosti AR praćenja drona tako da ti podaci zauzimaju puno prostora na samom ekranu. Taj je problem moguće riješiti na razne načine, dodavanje "Hide/Show UI" tipke koja bi omogućivala prebacivanje (eng. *toggle*) iz pojednostavljenog prikaza, kakav je trenutno dostupan u aplikaciji i proširenog prikaza sa svim dodatnim informacijama.

Osim tekstualnih podataka moguće je i vizualnim elementima oko samog AR prikaza prikazati podatke o kretanju drona i njegovoj orijentaciji dodavanje strelica izvan samog nišana.

5.2. Istovremeno praćenje više dronova

Aplikacija je trenutno napravljena na način da prikazuje prikaz samo jednog drona, taj prikaz nije ograničen na jedan specifičan dron, može biti bilo koji dron kojeg podržava DJI SDK, no ne može se istovremeno prikazati AR prikaz dva drona. Aplikacija je razvijana na taj način iz više razloga, primarno jer je jedan dron bio dostupan za testiranje i potreba optimizacije

kod praćenja većeg broja istovremenih uređaja, ali je moguće proširenje na više dronova od početka bilo uzimano u obzir.

Način na koji bi dodavanje dodatnih dronova bilo omogućeno je sljedeći. Prvi korak je promjena strukture podataka u Firebase-u, trenutna struktura je zamišljena na principu da se kod slanja podataka o dronovoj lokaciji podaci prepisuju jedni preko drugih, budući da se radi o jednom dronu u bazi nije bio potreban nikakav identifikator jer se uvijek radilo na istoj strukturi. Takva struktura ne može funkcionirati sa više dronova jer je potrebno ažuriranje podataka svakog drona na razini njihove strukture pa je u samu strukturu potrebno dodati njezin identifikator. U DJI Controller aplikaciji je potrebno dodjeljivanje identifikatora drona pri spajanju aplikacije na dron. Zatim je potrebno stvoriti novu strukturu sa tim identifikatorom i početi pohranjivati podatke te kada se uređaj odspoji izbrisati iz baze strukturu sa prije dodijeljenim identifikatorom. Sa druge strane u Ikarosu, aplikaciji za AR prikaz, je potrebno omogućiti stvaranje novih 3D objekata ovisno o broju struktura sa različitim identifikatorima i zatim ih pravilno ažurirati.

5.3. Upozoravanje o prekidu VLOS-a

Aplikaciju bi se također moglo proširiti dodatnim upozorenjima poput upozorenja o prekidu VLOS-a, na taj način bi se moglo upozoriti kopilota i pilota pomoću audio signala, vibracije i poruke da je dron izašao iz VLOS-a. Te poruke bi trebale dati signal pilotu i kopilotu da vrate dron u vidno polje.

Postoji više načina za realiziranje te funkcionalnosti, jedna je korištenje Google-ovog Elevation API-ja, on sadrži podatke o nadmorskoj visini na svim točkama Zemlje i pomoću toga bi se mogla povući dužina od točke drona do točke AR uređaja i preuzeti nadmorske visine uređaja i drona, na njih se zbroje njihove visine koje se koriste u AR prikazu i zatim se prolaze sve točke od točke drona i točke AR prikaza te ako se nalazi neka točka čija je nadmorska visina veća od zbroja nadmorske visine broja i relativne visine dobivene iz DJI API-ja korisnika se informira da nema VLOS-a. Naravno i tu postoje neke limitacije, Google Elevation API ne uzima u obzir drveće koje isto može prekinuti VLOS.

Drugi način realizacije je takav da se gleda boja u nišanu, dron bi primarno iz perspektive kopilota trebao biti na nebu stoga bi sve oko njega trebalo biti plavo, sivo ili bijelo, pa bi se pomoću tih podataka moglo javiti je li VLOS izgubljen. Naravno i taj način ima puno problema, što ako je objekt iza kojeg se skriva dron plave boje, ili je zalazak ili izlazak Sunca pa nebo poprima drugačiju boju.

Kombinacijom oba rješenja bi se moglo dobiti preciznije rješenje za upozoravanje o prekidu VLOS-a koje bi zasigurno obogatilo aplikaciju.

5.4. Produkcijska verzija aplikacije Ikaros

Nakon što su sve gore navedene funkcionalnosti implementirane aplikacija će biti spremna za sljedeću fazu svojeg razvoja, a to je razvoj produkcijske verzije. Verzije aplikacije koja će

biti javno dostupna svima i bit će redovito održavana. Za to je potrebno napraviti managment i user načine rada, managment za administratore i developere, i user za sve druge korisnike koji preuzmu aplikaciju. Također će biti potrebno dodati mogućnost uparivanja drona sa samom aplikacijom, to će omogućiti lakše spajanje drona, prikupljanje i prikazivanje statističkih podataka o letovima i naravno krajnji cilj imati suradnju sa DJI-em na način da korisnici ne trebaju koristit DJI Controller, već mogu koristiti službene aplikacije poput DJI GO i DJI Pilot.

5.5. Proširenje na druge vrste uređaja

Ikaros je aplikacija namijenjena za AR prikaz dronova, ali to ne znači da se ona ne bi mogla prilagoditi za prikaz drugih vrsta uređaja poput mobitela, automobila ili bilo koje vrste uređaja koja ima u sebi neku vrstu geolokacijskog prijemnika. Moguće je razviti posebne aplikacije za te uređaje ili u samoj Ikaros aplikaciji odabrati kakvu vrstu uređaja želimo pratiti. Jedina stvar koja bi se trebala mijenjati je izgled UI-a ovisno o tome kakav uređaj želimo pratiti, no sama funkcionalnost praćenja je u srži ista, bitno je samo da uređaj može slati svoju lokaciju u stvarnom vremenu.

6. Zaključak

Ovaj rad bavio se razvojem aplikacije za vizualno praćenje bespilotnih letjelica koristeći proširenu stvarnost. Prvobitno se bavio teoretskom analizom tehnologija gdje su detaljno objašnjene tehnologije poput bespilotnih letjelica, GPS-a i proširene stvarnosti, također su bili opisani alati u kojima su same aplikacije bile razvijane i Firebase koji je bio korišten za njihovu komunikaciju, a zatim samom izradom aplikacija potrebnih za AR prikaz lokacije drona i procesu testiranja kroz vrijeme gdje su bile opisane i prepreke koje su se pojavljivale kod samog razvoja aplikacija i na kraju su bila istaknuta područja u kojima je moguć napredak i proširenje.

Aplikacije koje su bile razvijane u radu su DJI Controller koja je izrađena u Android Studijiu u Javi, koja se koristi za upravljanje samim dronom i slanje podataka o njegovoj lokaciji na Real Time Database Firebase-u. Druga aplikacija koja je bila razvijana je Ikaros, aplikacija za AR prikaz približne lokacije drona u vizualnom prostoru. Izrađena je u Unitiy-ju u C#-u.

Obje aplikacije su uspješno napravljene i testirane, ali naravno nisu savršene i postoji mnogo prostora za unaprijeđenje. Ikaros, aplikacija za AR prikaz, ima veliki potencijal za proširenje na veći broj istovremenih uređaja i na druge vrste uređaja sa GPS prijemnicima.

Smatram da mi je ovaj rad bio dobar izazov kao bi naučio nove tehnologije i testirao sposobnosti snalaženja u nišnim područjima gdje nema velik broj sličnih aplikacija pa ujedno i materijala, gdje testiranje nije najjednostavnije i potrebno je smisliti kreativne načine za rješavanje problema.

7. Literatura

[1] TheUAV.com (2021) *The UAV – Unmanned Aerial Vechile*. Preuzeto 12.08.2021. s http://www.theuav.com

[2] Xu F, Muneyoshi H (2017) *A Case Study of DJI, the Top Drone Maker in the World*. Preuzeto 12.08.2021. s https://www.kindai.ac.jp/files/rd/research-center/management-innovation/kindai-management-review/vol5_6.pdf

[3] dji.com (2021) *DJI - Official Webiste*. Preuzeto 12.08.2021. s https://www.dji.com/hr

[4] gps.gov (2021) *GPS: The Global Positioning System*. Preuzeto 12.08.2021. s https://www.gps.gov/

[5] ibm.com (2021) *Geographic coordinate system*. Preuzeto 12.08.2021. s https://www.ibm.com/docs/en/informix-servers/12.10?topic=data-geographic-coordinate-system

[6] Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E, Ivkovic M (2011) *Augmented reality technologies, systems and applications*. Preuzeto 13.08.2021. s https://www.csd.uoc.gr / hy469/files/panels/Augmented_reality_technologies_systems_and_applications.pdf

[7] developer.android.com (2021) *Android for Developers*. Preuzeto 13.08.2021. s https://developer.android.com/studio/intro

[8] unity.com (2021) *Unity Real-Time Development Platform*. Preuzeto 13.08.2021. s https://unity.com/our-company

[9] firebase.google.com (2021) *Firebase* Preuzeto 13.08.2021. s https://firebase.google.com/docs

[10] https://docs.unity-ar-gps-location.com (2020) *Unity AR+GPS Location*. Preuzeto 14.08.2021. s https://docs.unity-ar-gps-location.com/