Tampereen ammattikorkeakoulu

KUVIEN LÄHETYS DROONILTA PALVELIMELLE

Ville Helin

OPINNÄYTETYÖ
Toukokuu 2021

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

HELIN, VILLE:
Kuvien lähetys droonilta palvelimelle

Opinnäytetyö 23 sivua
Toukokuu 2021

Opinnäytetyössä tutustuttiin drooniin ja siihen tehtävään sovelluskehitykseen. Sovelluskehityksen tarkoituksena oli luoda toimiva Android-sovellus, jolla voidaan lähettää ilmassa olevalta droonilta kuvia palvelimelle. Sovellukselta vaadittiin, että se toimisi Android-käyttöjärjestelmää käyttävällä puhelimella ja hyödyntäisi Tampereen ammattikorkeakoulun käytössä olevaa droonia.

Opinnäytetyössä perehdyttiin droonin toimintaan. Lisäksi käsiteltiin työn Android sovelluskehitykseen hyödynnettyjä osia ja tehtiin Java-ohjelmointikieltä hyödyntäen Android-sovellus, jolla kuvien lähetys palvelimelle onnistuu. Työssä tarkoituksena oli hyödyntää Android-kehityksessä DJI Mobile SDK:ta, mutta se hylättiin sen toimimattomuuden vuoksi.

Opinnäytetyö tehtiin yhteistyössä Tampereen ammattikorkeakoulun ja SURE-projektin kanssa.

Asiasanat: drooni, sovelluskehitys, ilmakuvaus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

HELIN, VILLE:
Sending Images from a Drone to a Server

Bachelor's thesis 23 pages
May 2021

The purpose of this thesis was to create a working application for devices that use Android operating system. The goal of the application is to send images taken with a drone to a server in real time. It was required that the application work on Android operating system and make use of drones that Tampere University of Applied Sciences have.

The theoretical part of the study explored the history of drones and their current development. Technical specification and capabilities of drones were considered. The theoretical part also explained tools that were used in making the application and presented their capabilities. The technical part of the study presented the most important parts of the program made in the work. The main idea of the program was that it would be able to send the pictures taken with the drone to a server while the drone was still flying. At the beginning of the work it was prescribed to use DJI Mobile SDK but it was canceled due to it not being able to work like it was intended.

The work was conducted in cooperation with Tampere University of Applied Sciences and SURE project.

Key words: drone, software development, aerial photography

SISÄLLYS

LYHENTEET JA TERMIT

DJI	Kiinalainen drooneja valmistava yritys
SDK	Ohjelmistokehityspaketti
GPS	Maailmanlaajuinen paikannusjärjestelmä
Android	Google kehittämä käyttöjärjestelmä
iOS	Applen kehittämä käyttöjärjestelmä
retrofit	Androidille suunniteltu kirjasto http kutsuja varten
id	Tässä työssä droonien tunnistukseen käytetty arvo

1 JOHDANTO

Erilaiset pienet kuvaskopterit eli droonit ovat yleistyneet paljon viimevuosina. Drooneilla saa otettua laadukkaita ilmakuvia helpommin kuin koskaan aikaisemmin ja pystyy näkemään maailmaa linnun silmin. Droonit ovat tulleet tätä myöten varmasti tutuksi monille, mutta silti ei vielä tiedetä mihin kaikkeen drooneja voidaan hyödyntää tai käyttää apuna. Tässä työssä pienestä kuvauskopterista käytetään nimitystä drooni.

Opinnäytetyö tehtiin yhteistyössä Tampereen ammattikorkeakoulun ja SURE-projektin kanssa. SURE- projekti on Tampereen kaupungin ja monen Tampereella toimivan yrityksen yhteinen projekti, jonka ideana on luoda kaupungin turvallisuutta lisääviä älykkäitä ratkaisuita.

Tässä opinnäytetyössä on tarkoitus tehdä mobiilisovellus Android-käyttöjärjestelmälle. Sovelluksella voidaan lähettää lentävältä droonilta sillä otettuja valokuvia palvelimelle reaaliajassa ja silloin kun drooni on vielä ilmassa. Tämänkaltaista sovellusta ei ole vielä markkinoilla ja sovelluksella tullaan lisäämään esimerkiksi kaupungin turvallisuutta Tampereen kaupungin alueella tapahtuvissa tapahtumissa. Ideana on karkeasti arvioida tapahtumiin osallistuvien ihmisten määrää.

Opinnäytetyössä käydään läpi työssä käytettyjä tekniikoita, tutustutaan apuna käytettyyn drooniin ja lopuksi avataan tehdyn sovelluksen toimintaa ja koodia helpommin ymmärrettäväksi.

2 SURE

Lyhenne SURE tulee englannin kielen sanoista Smart Urban Security and Event Resilience ja se on hanke, jonka tavoitteena on luoda kaupungin ja kaupunkiympäristöstä turvallisempi sen kaupunkilaisille ja kaupungissa vieraileville. Hankkeen tarkoituksena on kehittää ja testata erilaisia kaupungin turvallisuuteen liittyviä älykkäitä ratkaisuja (SURE – Smart Urban... n.d.).

Tampereen kaupunki on saanut projektille rahoituksen Euroopan unionilta ja sen toteutuksessa ovat mukana muun muassa Tampereen kaupunki, Tampereen yliopisto, Tampereen ammattikorkeakoulu, Business Tampere, Nokia, Securitas, Insta ja Intopalo (Kaupunkiturvallisuutta kehittämässä – SURE! 2019).

SURE tuo myös yrityksille mahdollisuuden testata ja kehittää erilaisia kaupunginturvallisuuteen liittyviä tuotteitaan ja palveluita keskeisten kaupunkialueiden ja kaupungin ympäristössä (SURE – Smart Urban... n.d.).

Vuoden 2021 alussa voimaan tullut Euroopan unionin asettama laki määrää rekisteröitäväksi kaikki yli 250 grammaa painavat tai kaikki kameralliset droonit. Lisäksi droonin käyttäjän pitää perehtyä droonien lentämiseen koskeviin sääntöihin ja suorittaa niistä koe (Uusi dronelaki voimaan… 2020).

Opinnäytetyöllä oli tavoitteena saada lentävältä droonilta sen ottama kuva palvelimelle analysoitavaksi mahdollisimman reaaliajassa. Sovellukselta vaadittiin, että se toimisi Android puhelimella ja käyttäisi mahdollisesti DJI:n Mobile SDK:ta.

Sovellusta on tarkoitus käyttää SURE-projektissa laskemaan ihmismääriä erilaisista tapahtumista Tampereen alueella. Tässä opinnäytetyössä tutustaan kuinka droonilta saadaan lähetettyä kuvat palvelimelle.

3 TEKNIIKOIDEN KUVAUS

3.1 DJI Yritys

DJI on kiinalainen yritys, joka on perustettu vuonna 2006. DJI on maailmanlaajuisesti tunnettu sen valmista drooneista ja niissä käytetyistä kamera tekniikasta. Yritys on erityisesti tunnettu sen valmistamasta Phantom -droonista. Vuonna 2017 yritys hallitsi noin 70 % kaupallisista droonimarkkinoista. (From Startup to Empire… 2018.)

3.1.1 DJI Mavic Air -drooni

Tässä opinnäytetyössä käytettiin sovelluskehityksen ja sovellustestauksen apuna DJI:n valmistamaa DJI Mavic Air -droonia. Tehdyn sovelluksen pitäisi silti toimi kaikilla DJI:n drooneilla, joita voi ohjata käyttämällä DJI GO 4 -sovellusta. Kuvassa 1 näkyy drooni vasemalla ja oikealla droonin kauko-ohjain.

KUVA 1. Mavic air -drooni ja sen kauko-ohjain

Käyttäjäkokemusta parantaa huomattavasti DJI:n drooneihin sisään rakennettu kompassi ja GPS-tekniikka. Nämä helpottavat käyttäjää paikantamaan drooni ilmassa ja saamaan droonilta tämän sijainti tietoja. GPS-tekniikan avulla drooni

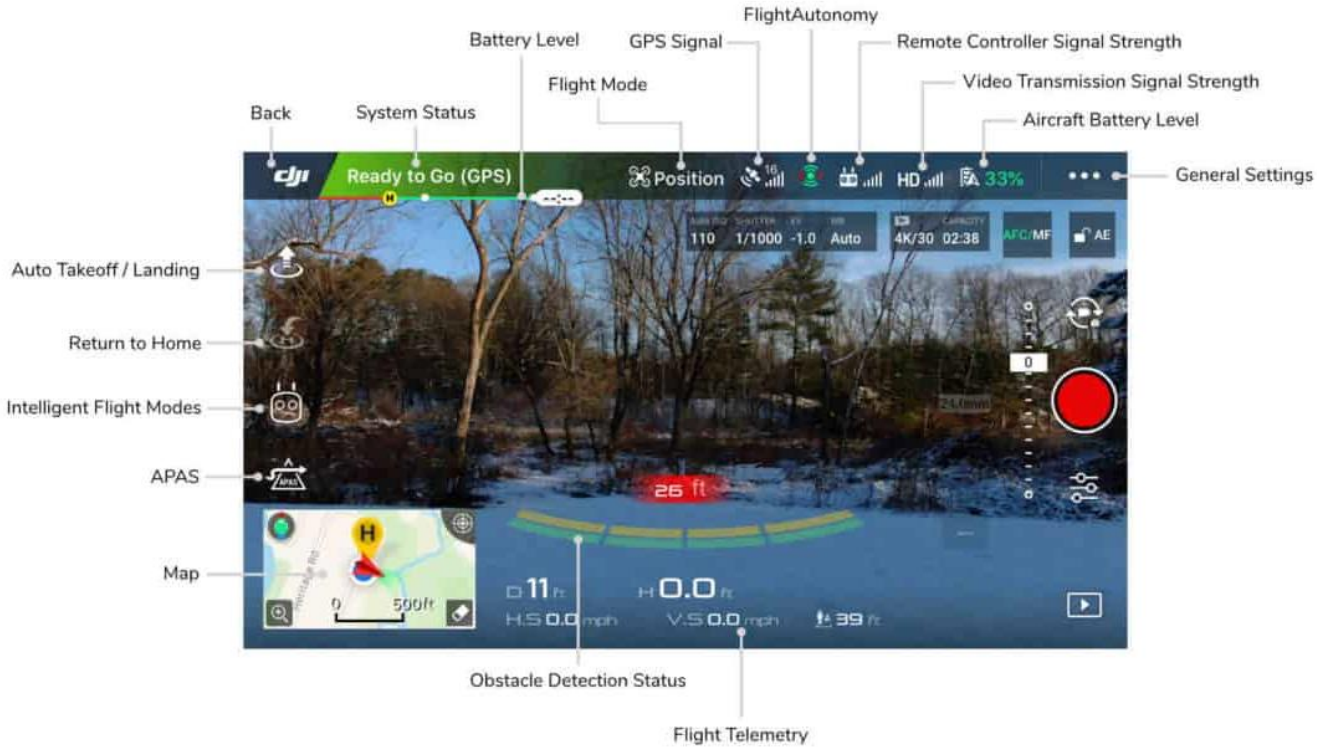
voi leijua täysin paikallaan ilman, että sen käyttäjä tekee mitään kauko-ohjaimella, joka tekee lentämisestä helpompaa ja turvallisempaa. Lightbrigde-tekniikan avulla käyttäjä pystyy katselemaan drooni live-videota suoraan älypuhelimelta ja tallentamaan kuvia ja videoita suoraan droonilta, jopa pitkiäkin matkoja, joka ei ollut mahdollista ennen mahdollista. (From Startup to Empire… 2018.)

DJI Mavic Air on yksi DJI:n valmistamista pienimmistä ja kevyimmistä drooneista. Se julkaistiin tammikuussa 2018 ja siinä oli silloin ainutlaatuisia ja uusia ominaisuuksia. Mavic air -droonissa on erityisen hyvä kamera, jolla pystyy ottamaan jopa 12 megapikselin kuvia tai 120 fps:n videokuva. Droonin kamera käyttää gimbal-tekniikka, joka tasapainoitaa kameran kolmea akselia hyödyntäen niin, että sillä voi ottaa vakaita kuvia, vaikka drooni heiluisikin ilmassa. Drooni painaa noin 400 grammaa, sen maksimi lentoaika on 21 minuuttia ja sen huippunopeus voi olla jopa 68 km/h. (Corrigan, F. 2020.)

Droonissa on sensorit edessä, takana ja alhaalla, jolla pystytään havaitsemaan mahdollisia esteitä, joita lentäessä voi tulla eteen. Droonin havaitessa este voi se yrittää kiertää estettä tai mahdollisesti pysähtyä enne osumaa, joka tekee lentämisestä huomattavasti turvallisempaa. Asetuksista voidaan myös ottaa tämänkaltaiset automaattiset avustukset pois päältä.

3.1.2 Dji Go 4 -sovellus

DJI GO 4 sovellus on DJI:n tekemä sovellus, joka toimii niin Android kuin iOS laitteilla. Sovelluksella pystytään hallitsemaan droonia liittämällä laite, johon sovellus on asennettu droonin kauko-ohjaimeen. Sovellus on ladattavissa puhelimen sovelluskaupasta tai DJI:n verkkosivuilta ilmaiseksi. Sovelluksella nähdään droonin kameran kuvaa reaaliajassa ja voidaan muuttaa droonin asetuksia. Voisi sanoa, että sovellus on pakollinen osa droonilla lentämistä. Sovellus on hyvin laaja ja se on jatkuvasti kehityksen alla. Kuva 2 näyttää miltä DJI GO 4 näyttää kun on DJI GO 4 -sovellus on lentonäkymässä kuvassa myös selitykset tärkeimmille elementeille sovelluksessa.



KUVA 2. DJI GO 4 sovelluksen näkymä selitettynä (Doggett, S. n.d.)

Sovellus on hyvin helppo käyttää ja se lisää huomattavasti käyttäjän mukavuutta droonilla lentämiseen. Sovellus pitää hyvin droonissa olevan käyttöjärjerstemän ajantassalla ja kertoo käyttäjälle signaalin tason kauko-ohjaimelle, joka tekee lentämisestä turvallisempaa.

3.1.3 DJI Mobile SDK

DJI Mobile SDK on DJI:n valmistama ohjelmistokehityspaketti Android ja iOS laitteille. Paketilla voidaan tehdä sovelluksia laitteille käyttämällä droonin kaikki mahdollisia ominaisuuksia hyödyksi yhdessä sovelluksessa kuten esimerkiksi DJI Go 4 - sovellus. DJI:n ohjelmistokehityspaketti tarvitsee Android sovelluksessa asennettavaksi Dji:n kirjastoja ja lisäksi verkkosivuilla projektin luomisen, joka antaa sovellukseen käyttöön api key:n.

DJI Mobile SDK on siis eräänlainen ohjelmistokehityspaketti, jolla voidaan yhdistää droonin toiminnot suoraan tehtävään sovellukseen androidin kehitysvaiheessa. Tällä pitäisi saada tehtyä samankaltainen sovellus minkälainen DJI Go

4 -sovellus on. Eli voitaisiin käyttää droonin kameraa ja saada droonilta tietoja sen paikasta ja nopeudesta suoraan yhteen sovellukseen.

3.2 Android

Android on yksi suosituimmista käyttöjärjestelmistä erilaisille kosketusnäyttöä hyödyntäville laitteille. Alun perin Android on Android Inc. kehittämä käyttöjärjestelmä, mutta vuonna 2005 google osti käyttöjärjestelmän. Nykyään Android on pääosin googlen kehittämä avoimen lähdekoodin käyttöjärjestelmä. (Callaham, J. 2020.)

Avoimella lähdekoodilla tarkoitetaan, että käyttöjärjestelmän koodi on vapaasti kaikkien nähtävillä, käytettävissä ja muokattavissa. Androidille voidaan kirjoittaa sovelluksia Java tai Kotlin ohjelmointikielillä ja sovellukset asennetaan Android laitteisiin käyttäen APK-asennuspakettitiedostoja.

3.2.1 Retrofit

Retrofit on square Inc. kehittämä kirjasto Androidille, Javalle ja Kotlinille. Kirjaston tarkoituksena on tarjota nopea ja helppo yhteys erilaisiin http verkkopyyntöjen lähettämiseen okHttp:n avulla. Tyypillisesti http kutsuja käytettään erilaisen json muotoisten kutsujen lähettämiseen. (A type-safe HTTP client for… n.d.)

3.2.2 FileObserver

FileObserver on Androidin sisäinen luokka, jolla voidaan hallita ja tarkkailla tiedostojen muutoksia ja lisäystä ennalta määritetyissä poluissa. Muutoksen havaittua pystytään tekemään haluttuja operaatioita koodissa. Kuvassa 3 on lista mitä muutoksia fileObserver pystyy tunnistaa. Listasta voidaan määrittää luokalle erilaisia muutoksia mitkä käynnistävät sen toiminnan. Listassa on myös mahdollisuus valita ALL_EVENTS, joka käynnistää toiminnan kaikista mahdollisista muutoksista mihin fileObservver voi reagoida. Jokainen tilanne antaa oman koodin, josta pystytään tunnistamaan mikä tapahtuma sen laukaisi ja näin voidaan myös määrittää tapahtumille erilaisia toimintoja. Yksi fileObserver instanssi voi käsitellä montaa tiedostoa tai alikansiota. (FileObserver. n.d.)

ACCESS Event type: Data was read from a file
ALL_EVENTS Event mask: All valid event types, combined Value is either **0** or a combination of **ACCESS**, **MODIFY**, **ATTRIB**, **CLOSE_WRITE**, **CLOSE_NOWRITE**, **OPEN**, **MOVED_FROM**, **MOVED_TO**, **CREATE**, **DELETE**, **DELETE_SELF**, and **MOVE_SELF**
ATTRIB Event type: Metadata (permissions, owner, timestamp) was changed explicitly
CLOSE_NOWRITE Event type: Someone had a file or directory open read-only, and closed it
CLOSE_WRITE Event type: Someone had a file or directory open for writing, and closed it
CREATE Event type: A new file or subdirectory was created under the monitored directory
DELETE Event type: A file was deleted from the monitored directory
DELETE_SELF Event type: The monitored file or directory was deleted; monitoring effectively stops
MODIFY Event type: Data was written to a file
MOVED_FROM Event type: A file or subdirectory was moved from the monitored directory
MOVED_TO Event type: A file or subdirectory was moved to the monitored directory
MOVE_SELF Event type: The monitored file or directory was moved; monitoring continues
OPEN Event type: A file or directory was opened

KUVA 3. Tapahtumat, joita fileObserver voi hallita (FileObserver. n.d.)

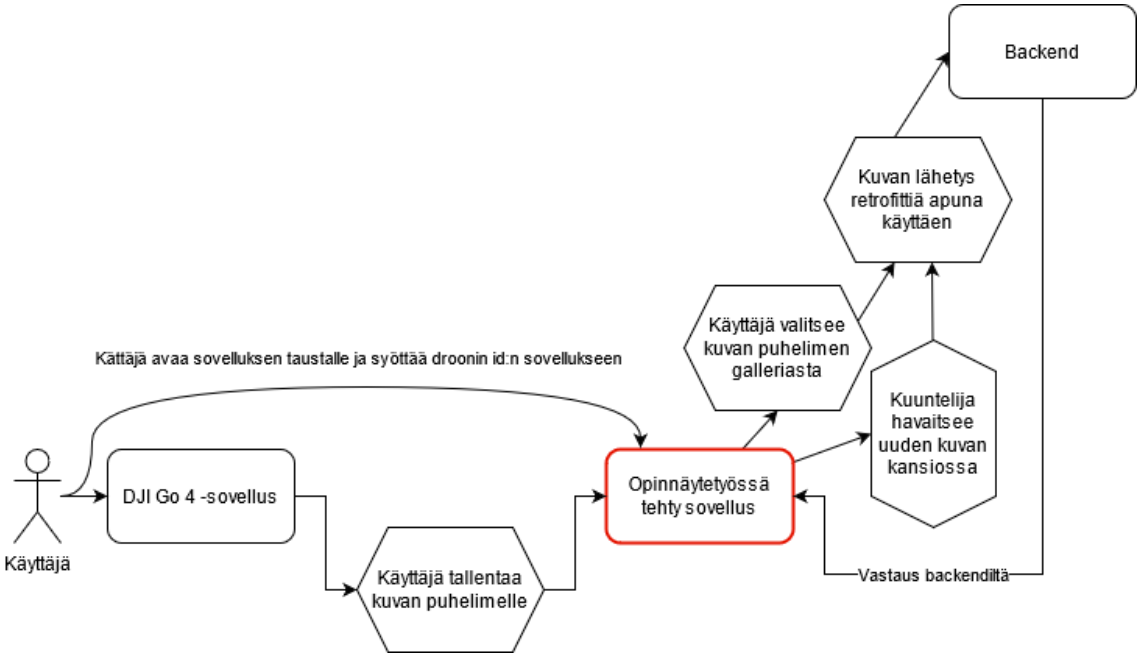
FileObserveria voidaan käyttää sovelluksissa, jotka tarvitsevat jonkin kuvan kolme mukaisen toiminnon laukaisemaan tapahtumia. Toiminnot liittyvät pääasiassa tiedostojen siirtoon tai muokkaukseen. Kun tapahtuma on laukaistu pystytään esimerkiksi kyseiselle tiedostolle tekemään jotain kuten tässä opinnäytetyössä kyseinen kuva lähetettään palvelimelle.

4 SOVELLUS

4.1 Esittely

Sovelluksen tarkoituksena on saada lähetettyä droonilla otettu kuva palvelimelle. droonia operoidaan normaalisti käyttäen sen kauko-ohjainta. Kauko-ohjain on nähtävissä kuvasta 1. Kauko-ohjaimeen liitetään puhelin, joka käyttää DJI Go 4 -sovellusta ja johon on asennettu opinnäytetyössä tehty sovellus kuvien lähetykseen.

Käyttäjän aloittaessa kuvien lähetys tulee hänen aluksi avata opinnäytetyössä tehty sovellus laittaa droonin_id tekstikenttään saamansa id ja käynnistää kuuntelija. Tämän jälkeen sovellus voidaan laittaa taustalle ja operoida droonia normaalisti käyttäen DJI Go 4 -sovellusta. Kuvan tai kuvien oton jälkeen voi kuvat tallentaa puhelimelle, jolloin sovellus huomaa ne ja lähettää palvelimelle. Sovelluksella voidaan lähettää kuvia suoraan puhelimen galleriasta valitsemalla ne sieltä ja lähettämällä sovelluksen avulla. Sovelluksen toiminta on havainnollistettu kuvaan 4.



KUVA 4. Sovelluksen arkkitehtuuri

4.2 Sovelluksen käyttöoikeudet

Sovellusta avattaessa ensimmäistä kertaa tai jos sovelluksella ei ole oikeutta hallita puhelimen muistia näytetään käyttäjälle kuvan viisi mukainen ilmoitus. ilmoitus pyytää käyttäjältä lupaa sovelluksen hallita laitteen muistiin kirjoitettuja tiedostoja. Sovellus tarvitsee toimiakseen oikeuden käyttää laitteen muistia. Androidissa on käyttäjän turvallisuuden takaamiseksi estetty sovellusten pääsy laitteen muistiin ilman, että käyttäjä antaa siihen luvan. Joten tähän pyydetään käyttäjältä lupa heti sovellusta avattaessa.

KUVA 5. Käyttäjän lupa hallita laitteen muistia

Kuvan kuusi koodi esittää miten kuvan viisi kaltainen lupa kysytään käyttäjältä. Koodissa listataan permissiot mitä halutaan kysyä listaan ja kutsumalla checkPermissions funktiota käydään tämä lista läpi ja tarkistetaan, että androidissa on lupa käyttää näitä permissioneita. Tätä funktiota kutsutaan activiteetin käynnistyessä.

```
String[] permissions = new String[] {
  Manifest.permission.READ_EXTERNAL_STORAGE,
  Manifest.permission.WRITE_EXTERNAL_STORAGE,
};

private boolean checkPermissions() {
  int result;
  List<String> listPermissionsNeeded = new ArrayList<>();
  for (String p : permissions) {
    result = ContextCompat.checkSelfPermission(this, p);
    if (result != PackageManager.PERMISSION_GRANTED) {
      listPermissionsNeeded.add(p);
    }
  }
  if (!listPermissionsNeeded.isEmpty()) {
    ActivityCompat.requestPermissions(
      this,
      listPermissionsNeeded.toArray(new String[listPermissionsNeeded.size()]),
      100
    );
    return false;
  }
  return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
  if (requestCode == 100) {
    if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
      // do something
    }
    return;
  }
}
```

KUVA 6. Koodi luvan pyytämiseen

Androidissa on niin sanottuja normaaleja ja vaarallisia käyttöoikeuksia. Vaarallisiin käyttöoikeuksiin luokitellaan kaikki, jotka voivat vaarantaa käyttäjän identiteettiä tai aiheuttaa kuluja käyttäjälle kuten soittaa puheluita (Permissions on Android n.d.).

Käyttöoikeuksien kysyminen on tehty turvaamaan käyttäjää ja lisäämään turvallisuutta, jotta sovellukset eivät ota puhelinta haltuunsa ilman, että käyttäjä on antanut tähän luvan. Internetin käyttöoikeus luokitellaan normaaliksi käyttöoikeudeksi ja siihen ei tarvita käyttäjän lupaa, mutta se listataan kaikkien käyttöoikeuksien joukkoon sovelluksen manifest -tiedostoon.

Androidissa on manifest -tiedosto, johon on listattu kaikki sovelluksen käyttöoikeudet, jotta käyttäjä voi saada kuvan siitä mitä sovelluksessa voidaan tehdä. Nämä käyttöoikeudet on listattu muun muassa puhelimen sovelluskaupassa so-

vellusta ladattaessa. Kuvassa 7 on tässä sovelluksessa käytetty manifest -tiedosto. Sovellus tarvitsee siis internet yhteyttä, jotta kuvat voidaan lähettää palvelimelle.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.example.DroneImageUpload">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">

        <activity android:name=".MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

KUVA 7. Sovelluksen manifest -tiedosto

4.3 Droonien tunnistus ja erittely

Kuvien eteenpäin lähetettäessä tarvitaan tieto millä droonilta kyseinen kuva on otettu. Tämä mahdollistaa, että droonilta saatu kuva voidaan liittää tiettyyn droo-niin, joka helpottaa droonien paikantamista ja aluetta mistä kuvat tulevat. Sovellus ei anna lähettää kuvia tai käynnistää kuuntelijaa enne kuin sille on annettu droonin id. Droonin id on tekstikenttä, joka ottaa syötteenä numeroita ja lähetetään kuvan lähetyksen kanssa palvelimelle. Kun droonin id on syötetty ja kuuntelija käynnistetty ei tekstikenttää pystytä enää muuttamaan ilman että kuuntelija sammutetaan väliaikaisesti. Tämä on tarkoitettu vähentämään käyttäjän tuottamia virheitä sovellusta ajettaessa.

4.4 Kuvien lähetys galleriasta

Sovelluksella kuvia voidaan lähettää myös ilman automaattilähetystä. Kuvat voidaan valita suoraan puhelimen kuva kirjastosta. Painamalla nappia valitse kuva avaa sovellus puhelimen kuva kirjaston ja pystyy käyttäjä valitsemaan kuvan puhelimelta. Käyttäjän valittua haluamansa kuva tulee se näkyviin sovellukseen. Lähetä kuva nappia painettaessa kuva lähetetään palvelimelle ja näytetään siitä palvelimelta tuleva vastaus näytöllä. Lähetä kuvaa painettaessa ilman että droonille on id asetettu antaa sovellus siitä virhe ilmoituksen käyttäjälle eikä sovellus lähetä kuvaa.

4.5 Kuuntelija ja sen toiminta

Kuvasta yhdeksän näkyvän koodin funktiota startWatching kutsutaan painettaessa sovelluksen käynnistä kuuntelija painiketta. Funktiossa alustetaan sijainti, jota halutaan tarkastella tässä tilanteessa sijainti on /DJI/dji.go.v4/DJI Album/. Tästä näytetään sovelluksessa toast ilmoitus käyttäjälle, jossa ilmoitetaan kuunneltava sijainti. Määritellään fileObserver tyyppinen kuuntelija tähän sijaintii ja tehdään se regoimaan moved_to tyyppisiin tapahtumiin. Kun kuuntelija havaitsee tapahtuman käynnistää se onEvent funktion. Koodin lopussa kuuntelija käynnistetään kutsumalla sen luokan metodia startWatching ja sovelluksen aktiiteettiin näytetään tekstikenttä, että kuuntelija on päällä. Kuuntelijan voidaan sammuttaa kutsumalla luokan metodia stopWatching. Kuvien lähetyksen koodi sijoitetaan onEvent funktion sisälle tässä tapauksessa se on multipartImageUpload ennen sen kutsumista tarkitetaan, että tiedosto on olemassa ja tiedosto ei ole jo lähetettyjen kuvien juokossa, jonka tarkoituksena on estää saman kuvan lähettäminen monta kertaa. Jos kuva on olemassa ja sen nimistä ei ole lähetty aikaisemmin niin lähetetään kuva palvelimelle.

```
public static FileObserver observer;

String fileUri;
ArrayList<String> uploadedFiles = new ArrayList<>();

private void startWatching() {

  final String pathToWatch =
    android.os.Environment.getExternalStorageDirectory().toString() + "/DJI/dji.go.v4/DJI Album/";

  Toast.makeText(
    this,
    "Kuuntelija käynnistetty. Kuunneellaan " + pathToWatch,
    Toast.LENGTH_LONG
  ).show();

  observer = new FileObserver(pathToWatch, FileObserver.MOVED_TO) {
    @Override
    public void onEvent(int event, final String file) {
      if (file != null) {
        if (!(uploadedFiles.contains(file))) {
          uploadedFiles.add(file);
          fileUri = pathToWatch + file;
          multipartImageUpload();
        }
      }
    }
  };
  observer.startWatching();
  observerTextView.setTextColor(Color.BLUE);
  observerTextView.setVisibility(View.VISIBLE);
}
```

KUVA 9. Kuuntelijan käynnistykseen käytetty koodi

4.6 Kuvan lähetys palvelimelle

Kuvan lähetykseen käytetään apuna retrofit kirjastoa. Kuvan lähetyksen lisäksi samanaikaisesti palvelimelle lähetetään leveysaste, pituusaste, droonin id ja token. Kuvan lähetystä kutsutaan, kun kuva on valittu galleriasta tai jos kuuntelija on havainnut kuvan lisäyksen tarkkailemassaan kansiossa. Kuvasta 10 on nähtävissä koodi, jolla kuva lähetettään palvelimelle.

```
interface ApiService {
    @Multipart
    @POST("/upload")
    Call<ResponseBody> postImage(
            @Part MultipartBody.Part image,
            @Part("lat") RequestBody lat,
            @Part("lon") RequestBody lon,
            @Part("drone_id") RequestBody drone_id,
            @Part("token") RequestBody token
    );
}

String fileUri;

private void multipartImageUpload() {
        Toast.makeText(this, "Kuvaa lähetetään..", Toast.LENGTH_SHORT).show();

        drone_id = droneIdEditText.getText().toString();

        File file = new File(fileUri);

        RequestBody reqFile = RequestBody.create(MediaType.parse("image/*"), file);
        MultipartBody.Part imageBody = MultipartBody.Part.createFormData(
          "image",
          file.getName(),
          reqFile
        );
        RequestBody latBody = RequestBody.create(MediaType.parse("text/plain"), "61.50353");
        RequestBody lonBody = RequestBody.create(MediaType.parse("text/plain"), "23.80842");
        RequestBody drone_idBody = RequestBody.create(MediaType.parse("text/plain"), drone_id);
        RequestBody tokenBody = RequestBody.create(MediaType.parse("text/plain"), "***TOKEN HERE***");

        Call<ResponseBody> req = apiService.postImage(
          imageBody,
          latBody,
          lonBody,
          drone_idBody,
          tokenBody
        );

        req.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

                if (response.code() == 200) {
                    responseTextView.setText("Lähetys onnistui: " + response.code());
                    responseTextView.setTextColor(Color.BLUE);
                } else
                    responseTextView.setText(response.code());
            }

            @Override
            public void onFailure(Call<ResponseBody> call, Throwable t) {
                responseTextView.setText("Lähetys epäonnistui");
                responseTextView.setTextColor(Color.RED);
                t.printStackTrace();
            }
        });
    }
```

KUVA 10. Kuvien lähetykseen käytetty koodi

Leveys- ja pituusaste on tallennettu kuvan tietoihin droonin toimesta kuvanotto-hetkellä, josta palvelin hakee tiedot saatuaan kuvan. Leveys- ja pituusaste ovat kuitenkin lähetyksessä pakollisia kenttiä, joten niihin on kova koodattu TAMKin koordinaatit. Tämä myös varmistaa sen, jos paikkatiedot hukkuvat tai niitä ei ole saatavilla kuvasta. Droonin id muuttujaksi luetaan käyttäjän syöttämä arvo tekstikentästä. Tokenilla varmistetaan palvelimelle, että lähetys tulee tunnetusta lähteestä.

Kuvan 10 koodin alussa annetaan käyttäjälle ilmoitus, että kuvaa ollaan lähettämässä ja otetaan droonin id:lle arvo tekstikentästä ja valmistellaan tiedosto sen sijainnin perusteella. Nyt ollaan valmiita viimeistelemään lähetys palvelimelle. Otetaan kaikki lähetettävät arvot, valmistellaan ne lähetystä varten ja kutsutaan lähetystä. Lähetyksestä saadaan vastaus palvelimelta ja tämä näytettään sovelluksen näkymässä.

5 POHDINTA

Opinnäytetyössä tehty sovellus onnistui hyvin. Siihen sisältyi pääasiassa Android kehitystä, mutta jouduttiin tutustumaan drooniin ja sen toimintaan. Itse en ollut aikaisemmin päässyt käsiksi drooneihin, joten kaikki oli uutta. Alkuluuloista huolimatta päästiin kehittämään sovellusta, jossa voitiin soveltaa aikaisemmin opittuja asioita. Lisäksi tehdylle sovellukselle oli oikea tarve ja vastaavanlaista sovellusta ei ole tarjolla muualta. Lopputulokseksi saatiin toimiva sovellus, jolla voidaan lähettää kuvia droonilta palvelimelle.

Opinnäytetyössä olisi alun perin ollut tarkoituksena hyödyntää DJI:n luomaa ohjelmistokehityspakettia eli DJI Mobile SDK:ta, mutta sen hankalan toiminnan ja toimimattouuden takia unohdettiin sen käyttö ja siirryttiin toiseen mahdolliseen ratkaisuun. Jossa käytettään apuna, jo valmista DJI Go 4 -sovellusta. Tällä operoidaan droonia normaalisti ja voidaan tallentaa kuvat puhelimen muistiin jo valmiiksi tehdyillä ominaisuuksilla. Joten opinnäytetyössä tarvitsee vain keskittyä Android kehitykseen ja tehdään DJI Go 4 – sovelluksen rinnalle toinen sovellus, joka havaitsee kuvan tallentamisen ja lähettää sen palvelimelle.

Sovellusta voisi helposti jatkokehittää myös liittämällä tehdyn sovelluksen DJI Go 4 -sovelluksen kanssa ja tehdä näistä yhtenäisen sovelluksen varsinkin, kun kaikki kuvan lähetykseen tarvittavat toiminnot on tehty. Lisäksi itse tehtävää sovellusta pystyttäisiin jatkokehittämään, jos niin haluttaisiin. Esimerkiksi droonin id on käyttäjän itse syöttämä kenttä. Voitaisiin tälle kentälle tehdä pudotusvalikko, jossa oli kaikki vapaat droonin id ja käyttäjä voisi valita niistä mitä käyttää. Tällä voitaisiin vähentää huomattavasti käyttäjän luomia virheellisiä syötteitä.

Tehtyä sovellusta voidaan tulla käyttämään apuna luomaan Tampereen alueella tapahtuvista tapahtumista turvallisempia. Sovelluksella pystytään lähettämään droonilla otettuja kuvia palvelimelle, jossa kuvia voidaan analysoida tarkemmin. Kuvista voidaan esimerkiksi tunnistaa objekteja ja laskea erilaisten objektien lukumäärää. Tehty sovellus toimii puhelimella pääasiassa taustalla, kun droonin ohjauksen apuna ja kuvien ottamiseen käytetään DJI Go 4 – sovellusta. Tehtyyn sovellukseen syötetään droonin id ja painetaan sovelluksen käynnistä kuuntelija

nappulaa. Kuuntelijan ollessa päällä huomaa sovellus, kun kuvat tallennetaan DJI Go 4- sovelluksesta ja lähettää ne sen jälkeen palvelimelle.

Uuden drooneihin kohdistuvan lain takia ei sovelluksen toimivuutta päästy testaamaan opinnäytetyön aikana kunnolla. Mikä olisi voinut olla mielenkiitoinen osa sovelluskehitystä, kun olisi nähnyt tekemän sovelluksen paremmin itse tarkoituksessaan ja nähnyt sovelluksessa olevia bugeja ja niihin liittyviä bugikorjauksia.

Android -kehitys onnistui opinnäytetyössä hyvin eikä se juuri tuottanut ongelmia opinnäytetyö prosessissa. Tarvitsi vain tutustua ja löytää itselleni uudet palikat, joilla tämänkaltainen sovellus voidaan tuottaa. DJI Mobile SDK ei taas halunnut lähteä toimimaan millään enkä sitä lopuksi saanutkaan toimimaan. Tämä vei suuren osan ajasta, jota ei voitu edes käyttää hyödyksi opinnäytetyössä ja tuotti paljon päänvaivaa opinnäytetyö prosessin alussa, koska ideana oli tehdä kyseinen sovellus hyödyntäen tätä SDK:ta.

LÄHTEET

A type-safe HTTP client for Android and Java. Luettu 11.3.2021. https://square.github.io/retrofit/

Callaham, J. 2020. Google made its best acquisition 15 years ago: Can you guess what it was?. Android Authority 11.6.2020. Luettu 19.3.2021. https://www.androidauthority.com/google-android-acquisition-884194/

Corrigan, F. 2020. DJI Mavic Air Features Review, Specifications and FAQs. Dronezon 13.4.2020. Luettu 19.3.2021. https://www.dronezon.com/drone-reviews/dji-mavic-air-review-features-specifications-faqs-answered/

Doggett, S. The Ultimate DJI Go 4 Tutorial. Dronegenuity. Luettu 9.3.2021. https://www.dronegenuity.com/dji-go-4-app-tutorial/

FileObserver. Luettu 11.3.2021. https://developer.android.com/reference/android/os/FileObserver

From Startup to Empire: The Evolution of DJI Drones. Dronelife 14.5.2018. Luettu 31.3.2021. https://dronelife.com/2018/05/14/from-startup-to-empire-the-evolution-of-dji-drones/

Kaupunkiturvallisuutta kehittämässä – SURE! 2019. Tampereen yliopisto 9.8.2019. Luettu 1.3.2021 https://www.tuni.fi/fi/ajankohtaista/kaupunkiturvallisuutta-kehittamassa-sure

Uusi dronelaki voimaan vuodenvaihteessa - mikä muuttuu? Traficom 3.12.2020. Luettu 6.4.2021. https://www.traficom.fi/fi/ajankohtaista/uusi-dronelaki-voimaan-vuodenvaihteessa-mika-muuttuu

Permissions on Android. Luettu 12.4.2021. https://developer.android.com/guide/topics/permissions/overview

SURE – Smart Urban Security and Event Resilience. Luettu 1.3.2021 http://www.baltic.org/project/sure-smart-urban-security-and-event-resilience/