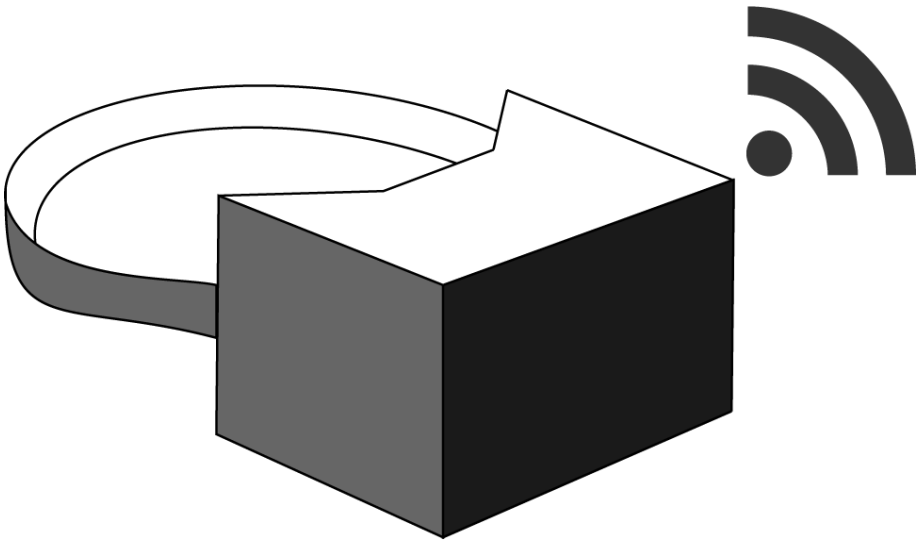
Nieminen Joonas

VR-laseilla etäohjattu kameralaite



Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2019

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Nieminen Joonas

Työn nimi: VR-laseilla etäohjattu kameralaite

Tutkintonimike: Insinööri (AMK), Tieto- ja viestintätekniikka

Asiasanat: VR-lasit, etäohjattu, kameralaite, mobiililaite

Opinnäytetyön toimeksiantajana toimi Kajaanin ammattikorkeakoulu Oy ja työssä tehtävä laite meni Kajaanin ammattikorkeakoululle esittely käyttöön. Idea työhön tuli omasta tarpeesta sekä mielenkiinnosta VR-laseja kohtaan. Työ oli myös uudelleen kehitystä aikaisemmasta opiskeluaikaisesta projektista.

Opinnäytetyön tavoitteena oli toteuttaa prototyyppi virtuaalilaseilla etäohjattavasta kamerasta, jonka datansiirto tapahtui internetin välityksellä laitteelta toiselle. Kameralaite koostui kahdesta servomoottorista, USB-liityntäisestä webkamerasta, tietokoneesta sekä Arduino uno -kehitysalustasta. Kameralaitteen runko suunniteltiin ja tulostettiin Kajaanin ammattikorkeakoulun 3D-tulostimella. Virtuaalilaseina toimi Android-puhelin, jonka Android-versio on 7.0 tai suurempi. Mobiililaitteen ja kamelaitteen välinen videoliikenne tehtiin käyttämällä Googlen tarjoamaa vapaanlähdekoodin omaavaa WebRTC-ohjelmointirajapintaa. VR-laseja käyttävän henkilön pään liikkeet tunnistivat mobiililaitteen gyroskooppi ja kiihtyvyysanturi. Näiden anturien data siirrettiin kameralaitteelle käyttäen Googlen Firebase-pilvipalvelua. Laitteiden välinen yhteys muodostettiin signaaliserverin kautta, jonka välityksellä mobiililaite ja kameralaite loivat yhteyden käyttäen samaa ennalta määritettyä salasanaa. Signaaliserverikoneena toimi Raspberry Pi -tietokone, joka oli yhteydessä internettiin.
Lopputulokseksi tuli kohtalaisen pienellä viiveellä toimiva 180 asteen kääntösäteellä oleva kameralaite, sekä mobiilisovellus, jolla kameralaitetta voi liikutella etänä sekä nähdä reaaliaikaista kuvaa kameralaitteen kamerasta. Kameralaite vaati verkkovirtaa sekä hyvän internet yhteyden toimiakseen.

KAMK • University of Applied Sciences

Abstract

Author(s): Nieminen Joonas

Title of the Publication: Remote controlled camera device by VR glasses

Degree Title: e.g. Bachelor of Engineering, ICT

Keywords: VR goggles, remote controlled, camera, mobile device

The Bachelor's thesis was commissioned by Kajaani University of Applied Sciences and the outcome is going to be in showcase use. The idea for the work came from the author's personal need and interest in VR glasses. The work was also a re-development of a previous study project.

The aim of the thesis was to make a prototype of a camera device which is controlled by VR glasses, the data transfer is done over the Internet from one device to another. The camera device included two servomotors, a USB-enabled web camera, a computer and an Arduino Uno development platform. The camera device frame was designed and printed with the Kajaani University of Applied Sciences 3D printer. An Android phone with Android version 7.0 or higher was working as virtual glasses. Video stream between the mobile device and the camera device was made using the open source WebRTC programming interface provided by Google. User head movement was recognized with the gyroscope and accelerometer of the mobile device. The data from these sensors was transferred to the camera device using Google's Firebase cloud service. The connection between the devices was established via a signal server and the mobile device and the camera device established a connection using the same predefined password. The signal server was a Raspberry Pi computer which was connected to the Internet.

The result was a moderately low delayed camera with a 180-degree turning radius and a mobile application for moving the camera device remotely and seeing a real-time view of the web camera. The camera device needs mains power and a good internet connection.

SISÄLLYS

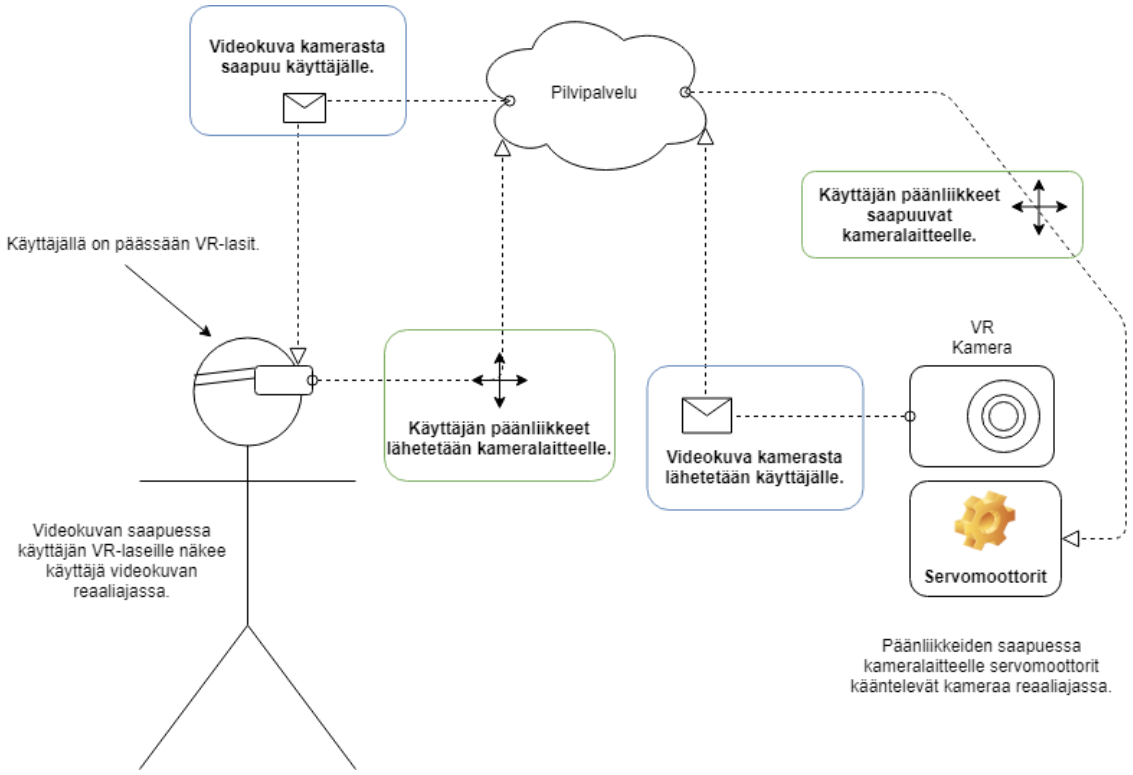
Symboliluettelo

ANDROID	Mobiililaitteilla toimiva käyttöjärjestelmä
C#	Ohjelmointikieli
FIREBASE	Reaaliaikainen datasäilö
GYROSKOOPPI	Kallistusanturi
JAVASCRIPT	Ohjelmointikieli
NODE.JS	Avoimen lähdekoodin alustariippumaton JavaScript runtime-ympäristö
UNITY 3D	Pelimoottori
WI-FI	Wireless local area network, langaton lähiverkkotekniikka
WEBRTC	Web Real-Time Communication -ohjelmointirajapinta
3G/4G	Langaton tiedonsiirtoteknologia

1 JOHDANTO

Idea työlle syntyi omasta tarpeesta ja mielenkiinnosta virtuaalilaseja kohtaan. Lisäkannustuksena tämän aiheen valintaan toimi se, että Kajaanin ammattikorkeakoulu oli halukas ottamaan tässä työssä tehdyn laitteen demokäyttöön erinäisissä tapahtumissa.

Tavoitteena oli tehdä virtuaalilaseilla etäohjattava kameralaite, jonka data liikkuu internetin välityksellä käyttäen WiFi-, 3G/4G-yhteyksiä. Laseina toimi käyttäjän mobiililaite, johon asennettiin Unity3D:llä tehty Android-sovellus. Kameralaitteena toimi tietokone, johon oli kytketty Arduino Uno -kehitysalusta. Arduinoon oli kytketty kolme servomoottoria, jotka liikuttavat kameraa mobiililaitetta käyttävän henkilön pään liikkeiden mukaan. Kuvassa 1 laitteen toiminta yksinkertaistetusti.



Kuva 1. Toimintakaavio

2 KÄYTETTÄVÄ LAITTEISTO

Työn vaatima laitteisto valittiin työn kehittyessä, jolloin saatiin valittua hyvin toimivat laitteet. Myös kustannukset pyrittiin pitämään mahdollisimman pieninä.

2.1 Mobiililaite

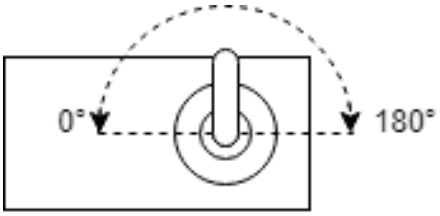
Mobiililaitteena toimi Samsung Galaxy S9+ Android-älypuhelin. Mobiililaitteena voivat toimia kaikki Android-laitteet, joiden Android-versio on yli 7.0.

Älypuhelimen työlle olennaiset tekniset tiedot [1, s. 1]:

- Samsung Exynos 9810
- 8-ytiminen suoritin
- 4x Exynos M3 @ 2,7 GHz + 4x ARM Cortex-A55 @ 1,7 GHz
- ARM Mali-G72 MP18 -grafiikkasuoritin
- 4G LTE -mobiiliverkkoyhteys 1 gigabitin latausnopeuksiin asti
- Wi-Fi
    - Wi-Fi on paljon käytetty langaton verkkoteknologia, joka käyttää radioaaltoja nopeaan tiedonsiirtoon.
- Android 8.0 Oreo-käyttöjärjestelmäversio ja Samsung Experience 9.0 -käyttöliittymä
- Kiihtyvyysanturi
    - Kiihtyvyysanturi on anturi, joka löytyy melkein kaikista nykyisistä älypuhelimista. Kiihtyvyysanturilla mitataan laitteen kiihtyvyyttä ja voidaan laitteen paikoillaan ollessa päätellä painovoiman avulla, missä asennossa puhelin on.
- Gyroskooppi
    - Gyroskooppi on anturi, joka mittaa laitteen kulmanopeutta.

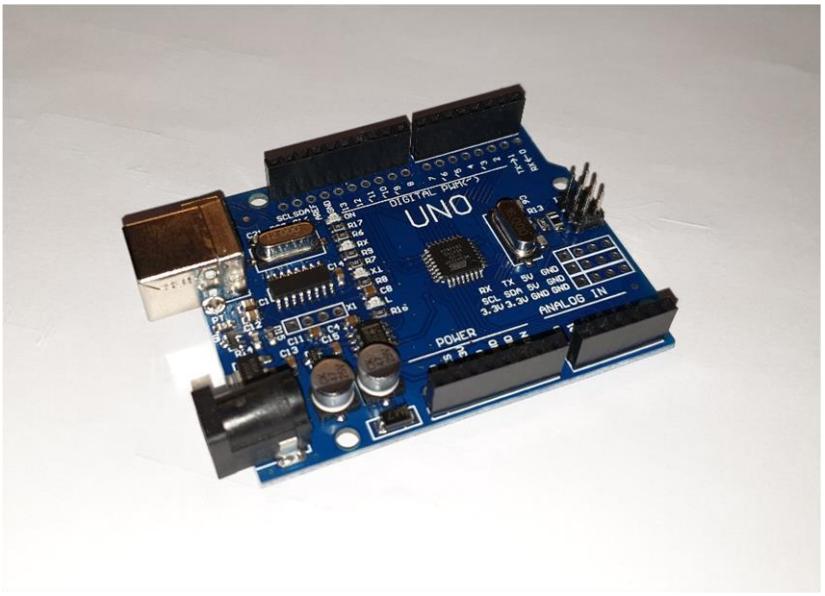
2.2 Muu laitteisto

- Windows-tietokone
    - Työssä käytettiin Windows-käyttöjärjestelmällä toimivaa tietokonetta, koska Unity 3D:llä tehdyt sovellukset eivät toimi Raspberry Pi -tietokoneella.
    - Tietokoneelta vaaditaan kaksi kappaletta USB-portteja, yksi kameralle ja yksi Arduino Uno -kehitysalustalle.
- Kaksi kappaletta servomoottoreita
    - Servomoottorin ohjaus tapahtui PWM-ohjauksella, jossa servoa ohjataan eripituisilla PWM-pulsseilla. Jos pulssin pituus on 1,5 millisekuntia, on servo nollakohdassa. Jos pulssin pituus on 1 millisekuntia, kääntyy servomoottorin akseli vasemmalle ja 2 millisekunnin pulssilla servomoottorin akseli kääntyy oikealle. Kuvassa 2 esimerkki servon kääntösäteestä.



Kuva 2. Servon kääntösäde

- Arduino Uno -kehitysalusta
    - Arduino Uno on pieni mikrokontrolleripohjainen kehitysalusta, jossa on 8-bittinen RISK-pohjainen ATmega328P-mikrokontrolleri. Kehitysalustassa on 14 digitaalista tuloa/lähtöä, joista kuutta voi käyttää PWM-lähtöinä, sekä 6 analogista sisääntuloa. Kehitysalustaa ohjelmoidaan tietokoneelle asennettavalla Arduino IDE-sovelluksella, joka voidaan ladata valmistajan kotisivuilta ilmaiseksi. Kehitysalusta ei välttämättä tarvitse ulkoista virtalähdettä; se ottaa tarvittavan käyttöjännitteen USB-ohjelmointikaapelin kautta tietokoneesta. [12, s. 1.] Arduino Uno -kortti kuvassa 3.



Kuva 3. Arduino Uno -kehitysalusta

- Kamera
    - Kamerana toimi Logitech Pro 9000 -webkamera, joka on kuvassa 4. Kameran maksimiresoluutio on 1280x720. Resoluutio pyrittiin pitämään kohtalaisen pienenä videonsiirron viiveen minimoimiseksi.
    - Kamerana voi toimia mikä tahansa USB-liityntäinen, joka toimii webkamerana tietokoneessa.

Kuva 4. Logitech Pro 9000

2.3 Laitteiston rajoitteet

Alla on listattu laitteiston asettamat rajoitteet.

- Servomoottoreiden kääntösäde on 180 astetta, joten vaikka käyttäjä pyörähtäisi täyden ympyrän, kameralaite ei käänny kuin 180 astetta.
- Käytettäväksi valitun kameran resoluutio rajoitti maksimiresoluutiota.
- Arduino Unossa ei ole WiFi-yhteyttä, jolloin kehitysalustan voisi yhdistää suoraan itsenäisesti Firebase-pilvipalveluun.
- Työhön käytetty Windows-tietokone vaati verkkovirran toimiakseen.

3 KÄYTETYT SOVELLUKSET

Työhön käytetyt sovellukset valittiin jo kohtalaisen tutuiksi tulleista sovelluksista, jolloin uusien sovellusten opetteluun ei kulu liikaa aikaa, vaan pystyin keskittymään itse prototyypin kehittämiseen.

3.1 Unity 3D -pelimoottori

Unity 3D on Unity Technologiesin kehittämä monialustainen pelimoottori, jolla voi kehittää 2D- tai 3D-pelejä eri alustoille. Pelimoottorista on saatavilla ilmainen sekä maksullinen versio. Nämä versiot eroavat toisistaan vain ulkonäöllisesti ja siinä, että ilmaisversiossa on pakotettu Unity 3D-logo sovelluksen käynnistyessä. Maksullisen version joutuu ostamaan, jos yrityksen tuotto on yli 100 000 dollaria. Työssä käytetään Unity3D-ilmaisversiota 2018.1.6.f1, joka on mahdollista siksi, koska kyseessä ei ole kaupallinen tuote vaan prototyyppi. [2, s. 1.]

Unity 3D:ssä pelien ja sovellusten ohjelmointiin voi käyttää joko UnityScriptiä tai C#-kieltä. UnityScript muistuttaa paljon JavaScriptiä, ja sitä kutsutaan usein JavaScriptiksi, vaikka ne ovat kaksi eri kieltä. C# on kuin hybridikieli, eli Javan ja C-kielien sekoitus. Työssä käytetään C#-kieltä sen tehokkuuden takia verrattuna UnityScriptiin, jonka takia se onkin yleisin Unityllä käytetty kieli.

3.2 Visual Studio 2017

Microsoft Visual Studio on Microsoftin ohjelmankehitysympäristö, jolla voi ohjelmoida useita eri ohjelmointikieliä, kuten Visual Basiciä, C++:aa, C#:a ja F#:a. Ohjelmalla voidaan tehdä esimerkiksi Windows-, web- ja mobiilisovelluksia. Visual Studioon saa ladattua erilaisia lisäosia, joilla ohjelman käytettävyyttä voidaan parantaa. [3, s. 1.]

Visual Studio 2017 julkaistiin 7. maaliskuuta 2017 [3, s. 1].

Unity 3D pystyy käyttämään Visual Studiota ohjelmointiympäristönä ja tukee Visual Studion ominaisuuksia kuten virheenetsintää. Tämä helpottaa koodissa olevien virheiden löytymistä huomattavasti.

3.3 Node.js

Node.Js on JavaScript runtime -ympäristö, joka on alustariippumaton. Sen avulla JavaScript-kielellä kirjoitetut ohjelmat voi suorittaa suoraan palvelimella, jonka jälkeen verkkosivu välitetään käyttäjälle. JavaScriptiä on pääsääntöisesti käytetty osana nettisivujen koodia ja scriptit suoritetaan käyttäjän koneella JavaScript-moottorilla. JavaScript ja Node.Js molemmat suoritetaan Googlen Chrome V8 JavaScript-moottorilla. [4, s. 1; 5, s. 1.]

4 OHJELMOINTIKIELET

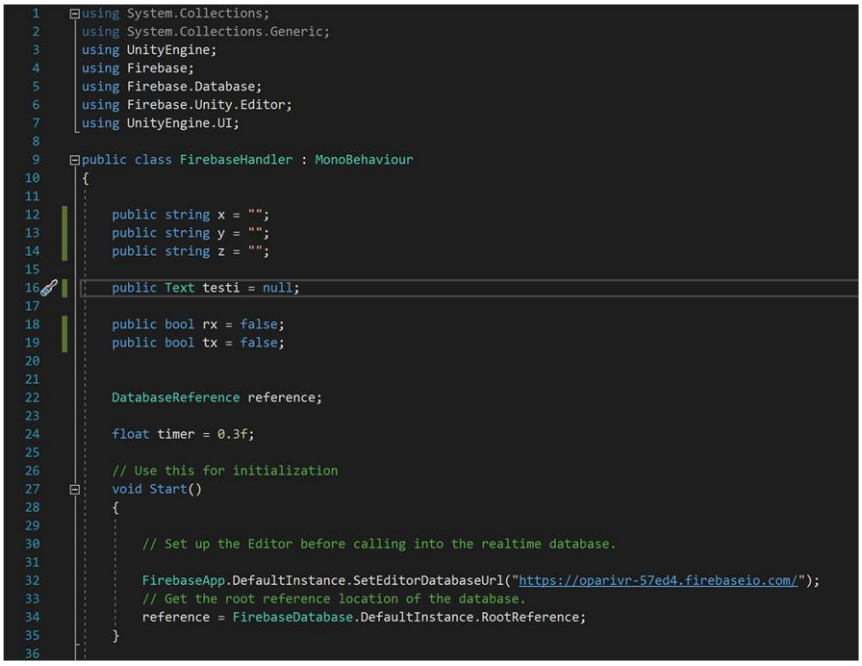
Työssä käytettävät ohjelmointikielet tulivat käytettyjen ohjelmien tukemien kielien perusteella.

4.1 C#-ohjelmointi

C# eli C Sharp on Microsoftin kehittämä olio-ohjelmointikieli, joka muistuttaa C++ ja Java-kielen sekoitusta. C# kieli on yleistymään päin sen helppouden takia verrattuna C++ tai C-kieleen. Tämän helppouden takia sitä käytetään useasti pelikehityksessä, mikäli vain pelimoottori tukee sitä. Sen syntaksin ymmärtäminen käy jo muita kieliä osaavalle helposti. C#-kielen esimerkki kuvassa 5. [6, s. 1.]

C# kielen ominaisuuksia:

- Case sensitivity – syntaksi eli isoilla ja pienillä kirjaimilla on merkitystä
- Automaattinen roskien keruu. (Engl. Garbage collection).
- Muuttujat on alustettava ennen käyttöä.
- C# käyttää pelkkiä olioita.
- Nimeäminen: kaikkien sanojen ensimmäinen kirjain kirjoitetaan kapitaaleilla, esim. ”*GameObject.CreatePrimitive(PrimitiveType.Cube);*”.
- C#-kieli tukee .NET Framework -kirjastoja vain Windows-ympäristöissä.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Firebase;
using Firebase.Database;
using Firebase.Unity.Editor;
using UnityEngine.UI;

public class FirebaseHandler : MonoBehaviour
{

    public string x = "";
    public string y = "";
    public string z = "";

    public Text testi = null;

    public bool rx = false;
    public bool tx = false;


    DatabaseReference reference;

    float timer = 0.3f;

    // Use this for initialization
    void Start()
    {

        // Set up the Editor before calling into the realtime database.

        FirebaseApp.DefaultInstance.SetEditorDatabaseUrl("https://oparivr-57ed4.firebaseio.com/");
        // Get the root reference location of the database.
        reference = FirebaseDatabase.DefaultInstance.RootReference;
    }

```

Kuva 5. Esimerkki C# -koodista

4.2 JavaScript-ohjelmointi

Nimestään huolimatta Java ja JavaScript ei ole sama asia. Molemmissa kielissä on sama syntaksi, mutta muita yhtäläisyyksiä ei juurikaan ole.

JavaScriptissä käytetään olioita, ja se on alustariippumaton kieli.

JavaScriptiä käytetään paljon web-kehityksessä. Monen nettisivun toiminnallisuudet on tehty käyttäen JavaScriptiä. Esimerkiksi esiin ponnahtavat mainokset on tehty JavaScriptillä. HTML-esimerkki kuvassa 6. [7, s. 1; 8, s. 1; 9, s. 1.]

JavaScriptin ominaisuuksia:

- Case sensitivity – syntaksi eli isoilla ja pienillä kirjaimilla on merkitystä
- Merkistönä käytetään Unicodea.
- Muuttaa automaattisesti muuttujan tyyppiä tilanteen mukaan. Muuttuja on ”var”, joka tulee sanasta variable.

```
<script type="text/javascript">

// Tulostetaan tekstiä selaimeen:
document.write("Hello World!");

/* Kommentointi tapahtuu samallatavalla kuin C# kielessä*/

</script>

<noscript>Selaimesi ei tue JavaScriptiä!</noscript>
```

Kuva 6. Esimerkki HTML-kieleen upotetusta JavaScriptistä

5 FIREBASE-pilvipalvelu

Firebase on Googlen tarjoama palvelu, joka mahdollistaa käyttäjälle analytiikan, pilvipalvelut, viestinnän ja virheraportit käytettäväksi projekteissa pienemmällä työmäärällä. Firebase on rakennettu Googlen palvelimille ja skaalautuu isompiinkin käyttötarkoituksiin. Kaikkien näiden ominaisuuksien hallinta tapahtuu Firebase-konsolin kautta.

Firebase toimii usealla eri alustalla: iOS, Android, Web, Unity 3D, C++.

5.1 Firebase Realtime database

Firebase Realtime Database on NoSQL-pohjainen pilvipalvelu, joka mahdollistaa käyttäjän tallettaa sekä hakea dataa pilvestä reaaliajassa pienellä viiveellä.

Firebase-pilvipalvelu mahdollistaa myös käyttäjäpohjaisen tietoturvan, joka perustuu luotuihin käyttäjätileihin Firebase-pilvipalvelutilille. Tällä tavalla voidaan määrittää, kenellä on oikeus tallentaa tai hakea dataa pilvipalvelusta.

Mikäli käyttäjä ei ole yhteydessä internetiin, voidaan data tallentaa välimuistiin, josta data siirtyy pilvipalveluun automaattisesti, kun yhteys internetiin on luotu.

Pilvipalvelua voidaan myös hyödyntää serverin tavoin eli pyörittää siellä ohjelmia joita voi hyödyntää, vaikka yhteyden luomisessa videopuhelua varten [10, s. 1].

5.2 Käyttäjätili ja sen hallinta

Käyttäjätilin hallinta tehtiin käyttäen Firebasen käyttäjän tunnistusta.

Firebaseen on olemassa ”admin”-tunnukset, jonka alle voitiin luoda useita käyttäjiä. Tässä tapauksessa käyttäjiä tarvitsi olla vain yksi. Käyttäjän rekisteröinti tapahtui Firebase-konsolissa olemassa olevan sähköpostin avulla, sekä valittiin käyttäjälle salasana.

Kaikki muutokset käyttäjätiliin tehtiin ”admin”-tunnusten avulla. Mobiililaitteen ohjelmistoon tuli kirjautumisikkuna, johon luodut tunnukset voitiin syöttää, jolloin päästiin luomaan yhteys kameralaitteeseen. Salasanan resetointia ei tässä tapauksessa tarvittu, koska ”admin”-tunnukset Firebase-konsoliin oli tiedossa ja ”admin”-tunnukset voi tarvittaessa resetoida.

6 WEBRTC-ohjelmointirajapinta

WebRTC on avoin ohjelmointirajapinta, joka mahdollistaa reaaliaikaisen kommunikaation selainpohjaisesti. WebRTC sisältää valmiit komponentit datan, äänen ja videon siirrolle. Näitä komponentteja voidaan käyttää JavaScript API:n avulla. [11, s. 1.]

WebRTC:n etuja:

- Googlen tarjoama täysin ilmainen ohjelmistorajapinta.
- Muita tämän kaltaisia ei ole.
- Datansiirto tapahtuu erittäin nopeasti.
- Peer2Peer (p2p) eli vertaisverkko. Toisin kuin tavallisessa verkossa, jossa asiakkaat lataavat tietoja palvelimelta, vertaisverkossa kaikki asiakkaat toimivat myös palvelimina, jolloin ne lähettävät sekä vastaanottavat tietoja. [13, s. 1.]
- WEBRTC-esimerkkejä löytyy paljon internetistä.

6.1 WebRTC mobiililaitteella

Mobiililaitteella WebRTC:tä käytettiin Unity3D:llä tehdyllä sovelluksella. Tähän on olemassa Unity3D-lisäosapalvelussa valmis paketti, jossa on hyvät esimerkit, joiden pohjalta tai niitä muokkaamalla saa erittäin hyvin tehtyä toimivan sovelluksen videon-, datan- ja äänensiirtoon.

7 SIGNAALISERVERI

Signaaliserveri hoitaa P2P-yhteyden muodostamisen kahden laitteen välille eli tässä tapauksessa Windows-tietokoneen ja mobiililaitteen välille.

7.1 Signaaliserverin suunnittelu

Suunnitteluvaiheessa tarkoituksena oli tehdä Googlen tarjoamaan pilvipalveluun virtuaalikone, jossa signaaliserveri pyörii. Google Cloud -palvelusta on mahdollista ottaa yhden vuoden kokeiluversio käyttöön, mutta silti käyttäjätilin luominen on pakollinen. Tämä tarkoittaa sitä, että tilin omistajan on pakko syöttää maksuvälinetiedot palveluun, joten tämä ei tullut kysymykseen, sillä konetta pitää pystyä tarvittaessa käyttämään muutkin kuin tilin omistaja.

Tässä tapauksessa, koska virtuaalikone ei ollut mahdollinen vaihtoehto, suunnitelmia piti muuttaa ja toisena vaihtoehtona oli paikallinen Linux-käyttöjärjestelmällinen kone, jossa signaaliserveri ajetaan.

7.2 Signaaliserverin toteutus

Toteutuksessa otettiin huomioon suunnitteluvaiheessa ilmenneet ongelmat Googlen tarjoaman palvelun kanssa, joten paikalliseksi serverikoneeksi valittiin Raspberry Pi 3 model B -yhden piirilevyn tietokone, joka esitelty kuvassa 7. Raspberry Pi -tietokoneeseen tuli käyttöjärjestelmäksi Ubuntu Mate -käyttöjärjestelmä, jonka saa ladattua Raspberry Pi:n kotisivuilta.

Kuva 7. Raspberry Pi 3 Model B -tietokone

Ubuntu Mate -levykuva ladattiin 16gb SD-muistikortille käyttäen balenaEtcher-ohjelmistoa. Kun ohjelmiston lataus oli valmis, käyttöjärjestelmän sisältävä kortti oli valmis käytettäväksi Raspberry Pi -tietokoneessa.

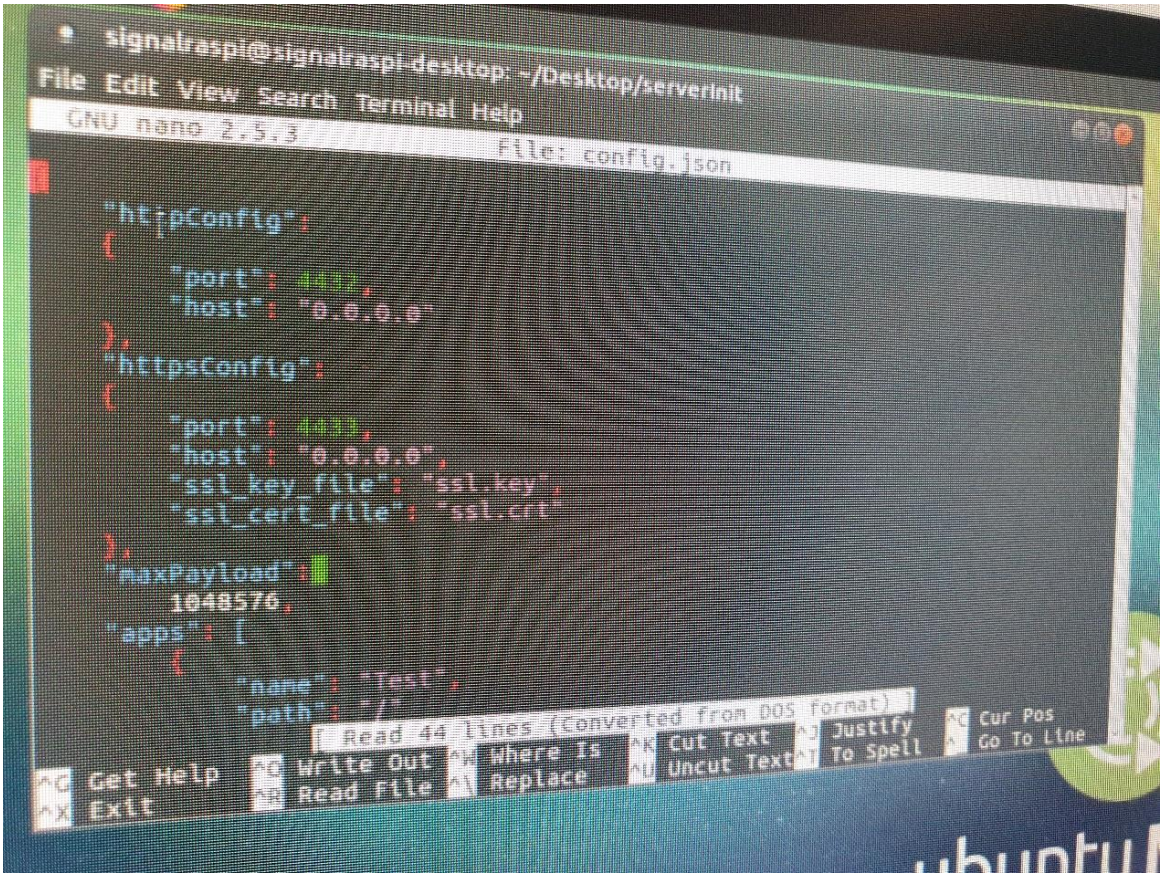
Ubuntu Mate -käyttöjärjestelmän käynnistyttyä ajettiin terminaalin kautta uusimmat päivityksen koneelle komennolla ”*sudo apt-get update”* ja ”*sudo apt-get upgrade”.*

Signaaliserverin asennuspaketti tuli WebRTC-lisäosan mukana. Paketti purettiin Raspberry Pi-tietokoneelle haluttuun polkuun, jonka jälkeen terminaalissa mentiin kansion juureen, jossa serveri oli eli tässä tapauksessa ”/Home/Desktop/InitServer”.

Tässä vaiheessa tapahtui Node.js:n asennus komennolla ”*sudo apt install nodejs”* sekä ”*sudo apt install npm”*. Asennuksen lopuksi voitiin käynnistää signaaliserveri komennolla ”*node server.js”*.

Tämän jälkeen muodostuivat tietoliikenne ongelmat eli koneeseen piti päästä kiinni paikallisen verkon ulkopuolelta käyttäen httptä. Signaaliserverin ”config”-tiedostoon täytyi käydä muutta-

massa sellaiset portit, joiden kautta haluttiin sallia ulkopuolinen liikenne. Tässä tapauksessa porteiksi valittiin http-yhteyttä käyttäen 4432 ja turvallisempaa https-yhteyttä käyttäen 4433. Kuvassa 8 on serverin config-tiedosto avattuna tekstinkäsittelyohjelmaan.



Kuva 8. Config-tiedosto

Kun portit oli määritetty, tehtiin vielä porttien avaus terminaalissa seuraavilla komennoilla:

Sallitaan portin reititys:

”sysctl net.ipv4.ip_forward=1”

Lisätään reitityssääntö:

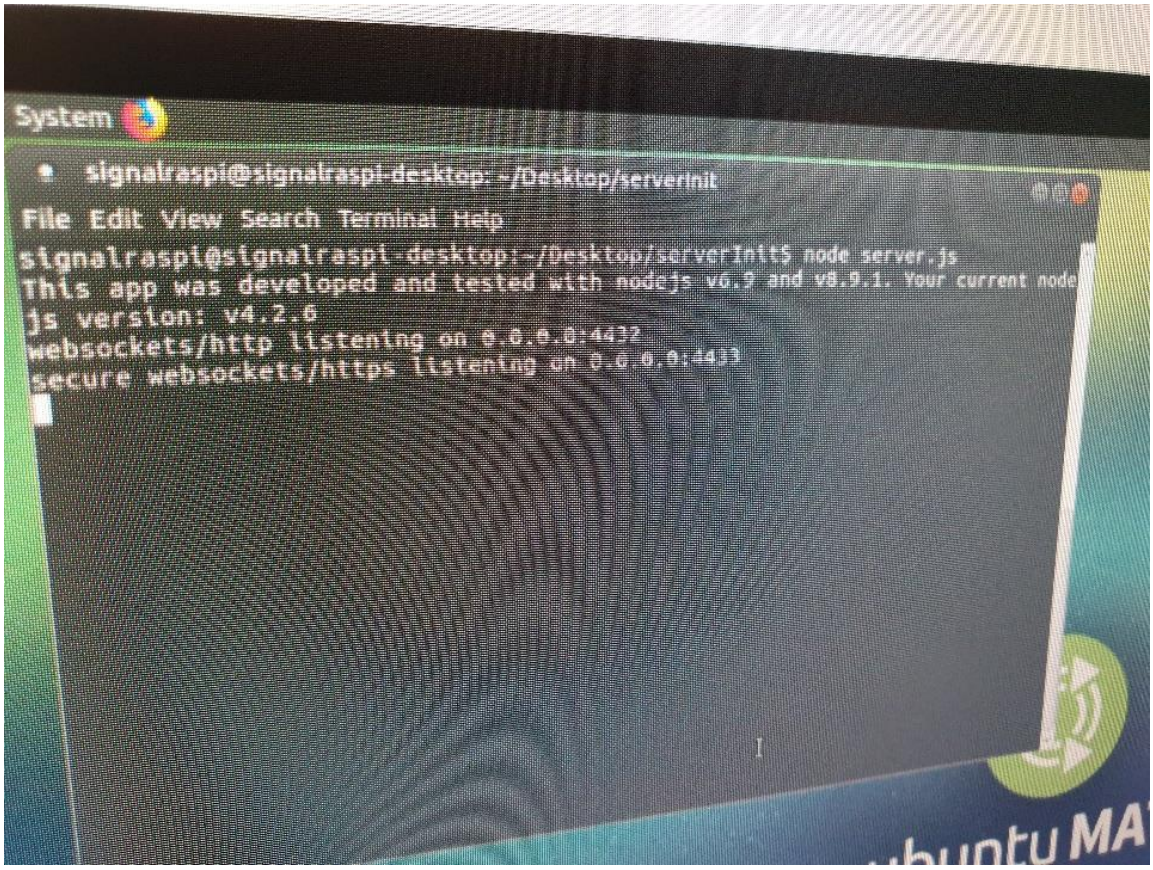
“iptables -t nat -A PREROUTING -p tcp -d n.n.n.n --dport portti -j DNAT --to-destination n.n.n.n:portti”

Iptable maski:

“iptables -t nat -A POSTROUTING -j MASQUERADE”

Portin reitityksen jälkeen tehtiin käyttäjätili osoitteeseen No-Ip.com, mihin määritettiin dynaaminen DNS-osoite Raspberry Pi -tietokoneen ipv4-osoitetta käyttäen.

Näiden määritysten jälkeen Raspberry Pi -tietokone käynnistettiin uudelleen ja käynnistettiin signaaliserveri uudestaan komennolla *”node server.js”* signaaliserveri hakemistosta. Serverin toimivuus varmistettiin kirjoittamalla selaimen osoiteriville dynaaminen DNS-osoite sekä haluttu portti 4432. Selaimeen ilmestyi teksti *”Running...”,* joka tarkoitti, että serveri on käynnissä. Kuvassa 9 terminaali, jossa serveri käynnistettiin.



Kuva 9. Signaaliserveri käynnissä Raspberry Pi:llä

8 KAMERALAITE

Kameralaite on helposti liikuteltava laite, jonka voi viedä minne tahansa, missä on saatavilla verkkovirtaa sekä hyvä nettiyhteys.

8.1 Laitteen suunnittelu

Projektisuunnitelmasta poiketen kameralaitetta ei voitu tehdä käyttäen Raspberry Pi -tietokonetta, vaan käytettiin Windows-tietokonetta, johon on USB-kaapelilla kytketty Arduino uno -kehitysalusta. Tämä siksi, koska tietokoneelle olemassa oleva WebRTC-lisäosa on todella hyvä ja tällä tavalla käyttökokemuksesta saadaan mahdollisimman reaaliaikainen.

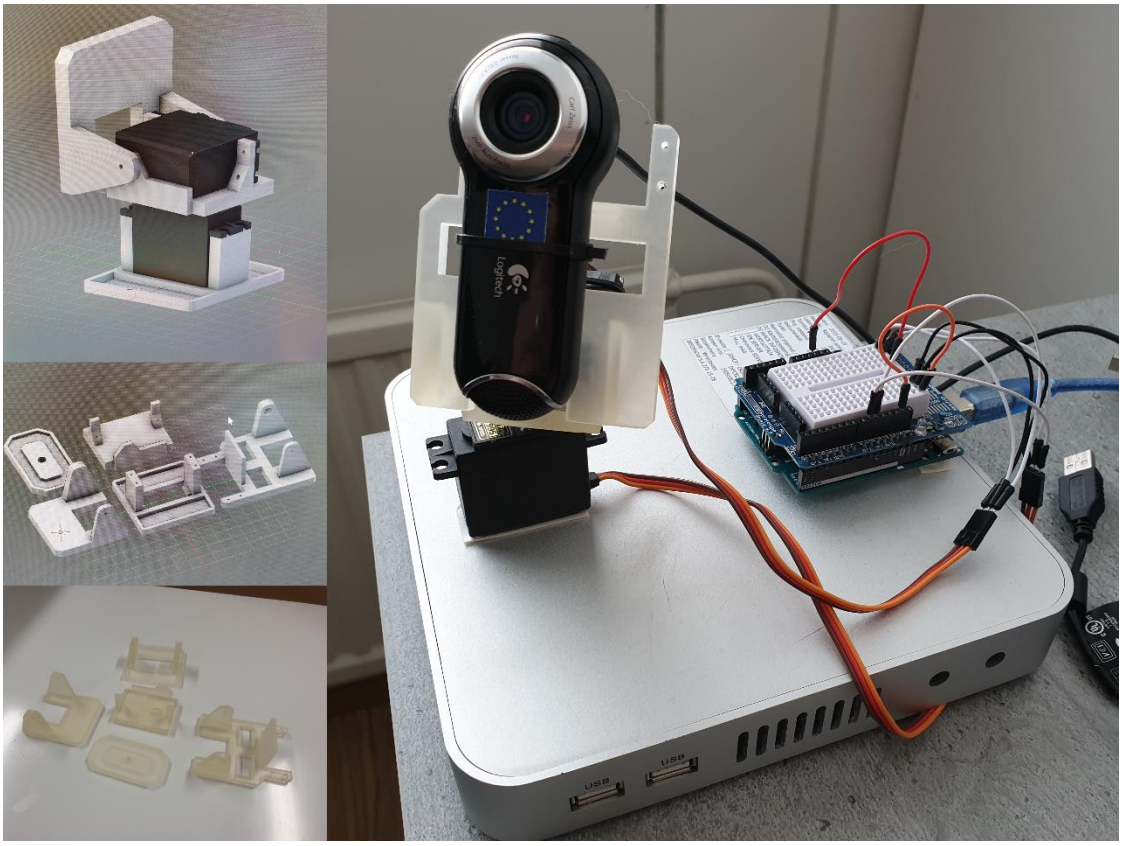
Kamerana suunniteltiin käytettäväksi webkameraa/webkameroita.

Tiedonsiirron tietokoneen ja Arduino unon välillä suunniteltiin käytettäväksi sarjaliikenne RS232 -tiedonsiirtoprotokollaa.

Laitteeseen suunniteltiin kytkettävän kolme kappaletta pieniä servomoottoreita, joita ohjataan sarjaportista tulevalla datalla kameran liikuttamiseksi. Servojen liikettä voitaisiin rajoittaa ohjelmallisesti, mikäli tarvetta, sekä saataisiin tehtyä ohjelmaan myös kameran rotaation resetointi. Kameralaitteen kokoamiseen tarvittavat osat suunniteltiin ja tulostettaan käyttäen 3d-tulostinta.

8.2 Laitteen toteutus

Kameralaitteen toteutuksessa lähdettiin liikkeelle siitä, että mallinnettiin kokoamiseen tarvittavat osat Blender3D -suunnitteluohjelmalla, jonka jälkeen mallit tallennettiin 3d-tulostimen tukemaan tiedostomuotoon STL. Mallit siirrettiin muistitikulle, josta edelleen tulostuskoneelle. Tarvittavien osien tulostusaika oli noin 12h. Tulostuksen jälkeen laite kasattiin laitteen tietokonekotelon päälle hyvän liikuteltavuuden takia. Kamerana toimi Logitech Pro 9000 -webkamera. Kuvassa 10 näkyy vaiheita kameralaitteen tekemisestä.



Kuva 10. Kameralaite

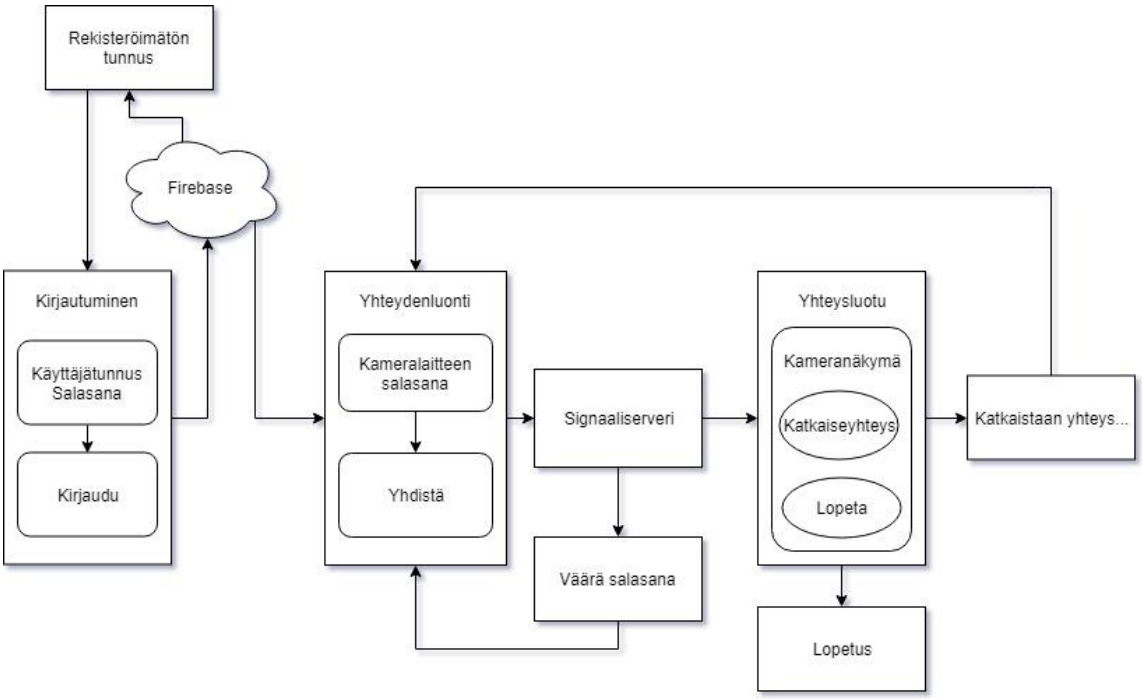
9 MOBIILISOVELLUS

9.1 Mobiilisovelluksen suunnittelu

Mobiilisovellus suunniteltiin toteutettavaksi käyttäen Unity3D-pelimoottoria ja siihen olemassa olevaa WebRTC-lisäosaa muokkaamalla työn tarpeisiin sopivaksi. Myös Firebase SDK:ta tarvittiin sovelluksen kehittämiseen.

Mobiilisovellukseen suunniteltiin yksinkertainen kirjautumisikkuna, jossa käyttäjä syöttää valmiiksi luodut tunnukset, jotta voi ottaa yhteyden kameralaitteeseen ja saada sieltä livekuvaa sekä käännellä kameraa.

Yhteydenluonti suunniteltiin tapahtuvaksi painamalla ”Connect”-nappia, jolloin laite odottaa signaaliserveriltä vastausta ennalta määrätyllä salasanalla, esim. ”Huone658”. Kameralaitteita voi olla useampi, mutta jokaiselle kameralaitteelle pitää määritellä oma salasana, jotta käyttäjä tietää, mihin kameralaitteeseen ottaa yhteyden. Kuvassa 11 on esitelty ohjelman toiminta kaaviona.



Kuva 11. Ohjelman toimintakaavio

9.2 Mobiilisovelluksen toteutus

Mobiilisovellus toteutettiin suunnitelman mukaan Unity 3D -pelimoottoria käyttäen.

Luotiin Unity 3D:hen uusi projekti, johon ladattiin tarvittavat lisäosat; WebRTC, Firebase SDK sekä Ardunity.

Firebase SDK määritykset tapahtuivat kirjautumalla projektin Firebase-tunnuksella konsoliin, josta ladattiin Firebase-määritystiedosto. Tiedosto sijoitettiin Unity 3D -projektin sisälle, ”Assets” kansion juureen. Määrityksen jälkeen voitiin aloittaa datansiirtoon tarvittavan scriptin ohjelmointi. Luotiin uusi script-tiedosto nimellä ”*FirebaseHandler.cs*”.

Ensiksi määritettiin tarvittavat muuttujat niin kuin kuvassa 12:

```
// Optional account verification
public InputField email;
public InputField password;

// CameraDevice Password for WebRTC connection
public InputField cameraPassword;

//For Debug use only!
public bool noLogin;

//Login panel
public GameObject LoginPanel = null;

//Strings which are for Firebase data
public string x = "";
public string y = "";
public string z = "";

//Sender or receiver build?
public bool rx = false;
public bool tx = false;

//WebRTC CallAppUi
private CallAppUi callAppUi;

//Firebase variables
private DatabaseReference reference;
private Firebase.Auth.FirebaseAuth auth;
private Firebase.Auth.FirebaseUser newUser;

//Delay timer for sending data to firebase
private float timer = 0.1f;
```

Kuva 12. Mobiilisovelluksen muuttujat

Muuttujien määrityksen jälkeen ohjelmoitiin ohjelma ottamaan käynnistyksessä yhteys Firebasen sekä tarkistamaan, onko kyseessä Debug-tila vai ei. Mikäli ei olla Debug-tilassa, tehtiin ohjelmassa tarvittavat Firebase-määritykset. Tämän jälkeen ohjelmoitiin ohjelma lähettämään

Unity-kameran rotaatiot(x,y,z), 100 ms viiveellä Firebase-pilvipalveluun. Lähettämiseen lisättiin viivettä sen takia, jotta dataväylä ei tukkeudu, koska muuten arvot lähtisivät pilveen joka kerran, kun ruutua päivitetään. Kuvassa 13 on esitelty, miten tämä on toteutettu. Unity-kamera saa rotaationsa Unityn integroidun Google Cardboard SDK:n kautta, joka käyttää puhelimen gyroskooppia.

```
timer -= Time.deltaTime;

if (timer <= 0)
{
    if (tx)
    {
        ChangeX();
        ChangeY();
        ChangeZ();
    }

    timer = 0.1f;
```

```
public void ChangeX()
{
    reference.Child("x").SetValueAsync(Camera.main.gameObject.transform.rotation.eulerAngles.x);
}

public void ChangeY()
{
    reference.Child("y").SetValueAsync(Camera.main.gameObject.transform.rotation.eulerAngles.y);
}

public void ChangeZ()
{
    reference.Child("z").SetValueAsync(Camera.main.gameObject.transform.rotation.eulerAngles.z);
}
```

Kuva 13. Arvojen lähetys Firebase-pilvipalveluun (Merkattu keltaisella)

Tässä vaiheessa tarkistettiin, että arvot menivät Firebase-pilvipalveluun. Arvot eivät ensin menneet suoraan palveluun, koska Firebase-konsolista piti määrittää pääsy datoihin muuttamalla kuvan 14 mukaisesti ”read = true” ja ”write = true”.

```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Kuva 14. Pääsyn salliminen Firebase-konsolissa

Sovellukseen voi lisätä tarvittaessa kaksivaiheisen käyttäjätarkistuksen, eli ensin kirjaudutaan sähköpostilla ja salasanalla, jonka jälkeen käyttäjän täytyy vielä syöttää kameralaitteen salasana. Tässä tapauksessa sähköpostikirjautuminen jätettiin vielä pois, mutta on helposti otettavissa käyttöön. Ohjelman käyttöliittymä pyrittiin pitämään mahdollisimman yksinkertaisena, kuten kuvasta 15 näkyy, jotta käyttö olisi helppoa.

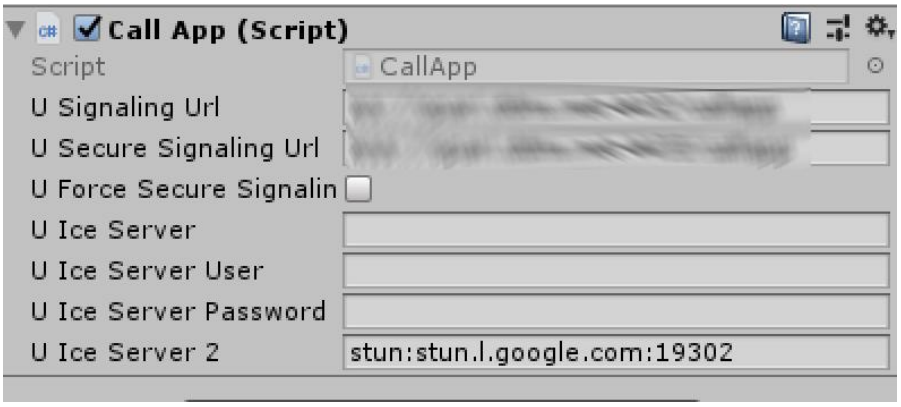
Kuva 15. Mobiilisovelluksen käyttöliittymä

Itse WebRTC-yhteys luotiin ”CallApp.cs”- ja ”CallAppUi.cs”-scripteissä.
”CallAppUi.cs” scriptissä määritettiin alussa tarvittavat asetukset; videokuvanresoluutio, päivitystaajuus, sekä muut tarvittavat asetukset kuten kuvasta 16 on näytetty. Nämä kaikki asetukset kameran salasanaa lukuun ottamatta oli esimääritetty scriptissä.

```
uAudioToggle.isOn = PlayerPrefsGetBool(mPrefix + PREF_AUDIO, true);
uVideoToggle.isOn = PlayerPrefsGetBool(mPrefix + PREF_VIDEO, true);
mStoredVideoDevice = PlayerPrefs.GetString(mPrefix + PREF_VIDEODEVICE, null);
uRoomNameInputField.text = roomNameFromJson; //PlayerPrefs.GetString(mPrefix + PREF_ROOMNAME, uRoomNameInputField.text);
uIdealWidth.text = PlayerPrefs.GetString(mPrefix + PREF_IDEALWIDTH, "640");
uIdealHeight.text = PlayerPrefs.GetString(mPrefix + PREF_IDEALHEIGHT, "480");
uIdealFps.text = PlayerPrefs.GetString(mPrefix + PREF_IDEALFPS, "30");
uRejoinToggle.isOn = PlayerPrefsGetBool(mPrefix + PREF_REJOIN, false);
uLocalVideoToggle.isOn = PlayerPrefsGetBool(mPrefix + PREF_LOCALVIDEO, true);
```

Kuva 16. WebRTC-asetukset

Unity3D -editorin puolella määritettiin, mihin signaaliserveriin ”CallApp.cs” yrittää ottaa yhteyttä. Kuvan 17 kohtiin ”U Signaling Url” ja ”U Secure Signaling Url” kirjoitettiin jo aikaisemmin käynnistetyn signaaliserverin No-Ip -osoite.



Kuva 17. Signaaliserverin osoitteen määritys

Tässä vaiheessa sovelluksen ensimmäinen versio oli valmis testattavaksi Android-mobiililaitteella. Aivan suoraan sovellus ei toiminut, koska Google Cardboard SDK muutti mobiililaitteen näytön kahteen osaan, jolloin käyttöliittymäkin näkyi kahtena. Asia korjattiin asettamalla Unity 3D:n asetuksista Cardboard SDK:n rinnalle ”None”, joka mahdollisti sen, että scriptissä voitiin muuttaa ajon aikana näytön piirtotapaa. Ohjelmaan täytyi lisätä funktio, joka käynnistyksessä kytki Cardbord SDK:n pois päältä ja asetti sen taas päälle, kun kirjautuminen oli tapahtunut. Funktion toiminta on esitetty kuvassa 18.

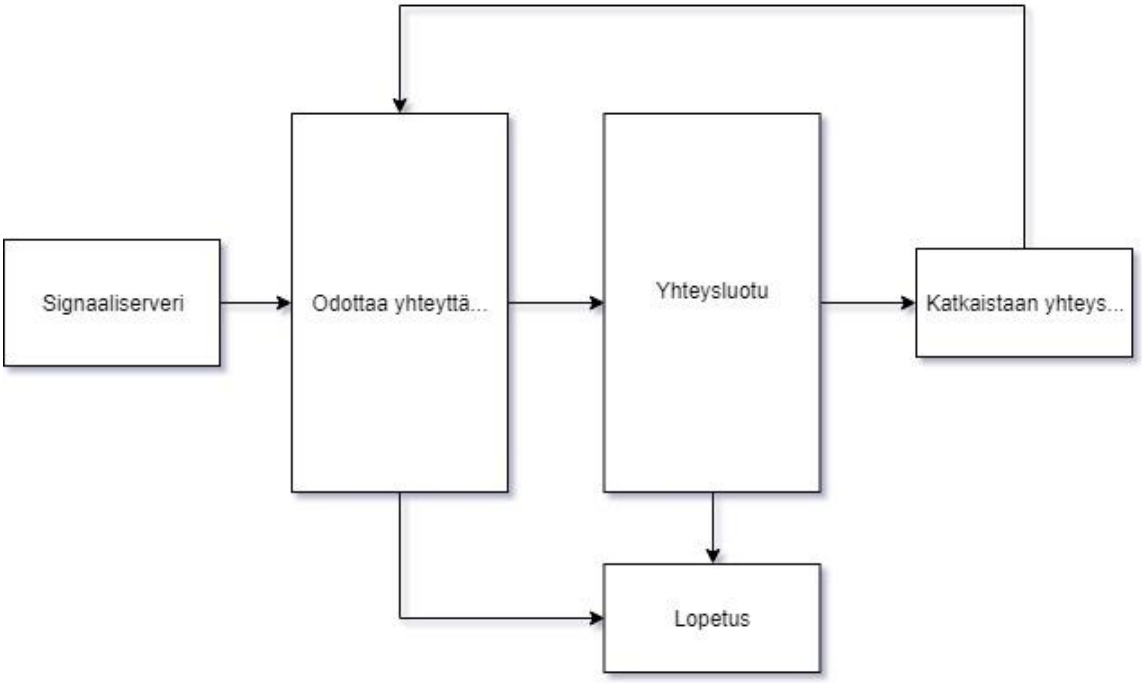
```
IEnumerator SwitchToVR()
{
    string desiredDevice = "cardboard";

    if (String.Compare(XRSettings.loadedDeviceName, desiredDevice, true) != 0)
    {
        XRSettings.LoadDeviceByName(desiredDevice);

        yield return null;
    }

    XRSettings.enabled = true;
}
```

Kuva 18. Näyttötilan muuttava funktio

10 KAMERALAITTEEN OHJELMOINTI

10.1 Ohjelmiston suunnittelu

Kameralaitteen ohjelmiston suunniteltiin käyttävän samaa pohjaa kuin itse mobiililaitteessa. Toisin kuin mobiililaitteessa, tähän ei suunniteltu kirjautumisruutua, vaan käynnistyessä laite aloittaa välittömästi yhteyden etsimisen ennalta määrätyllä salasanalla, esim. ”Huone658”. Kameralaitteen ohjelmiston suunniteltu toimintakaavio on kuvassa 19.



Kuva 19. Kameralaitteen ohjelmiston toimintakaavio

10.2 Ohjelmiston toteutus

Kamerasovellus toteutettiin samassa Unity 3D-projektissa kuin mobiilisovellus, mutta tällä kertaa sovellus tehtiin Windows-alustalle. Asetettiin ”FirebaseHandler.cs”-scriptistä muuttujat, ”Rx = true” ja ”Tx = false”. Tällä tavalla oli vain käytössä funktiosta ne, jotka olivat ehtolausekkeen ”Rx = true” alla. Tämä sama valinta kytki käyttöliittymän pois päältä.

Kamerasovellukseen ei tullut minkäänlaista käyttöliittymää vaan kameran salasana määritettiin, ”data.json” tiedostoon, josta se haettiin sovelluksen käynnistyksessä. Kun kameran salasana oli haettu, otti sovellus automaattisesti yhteyttä signaaliserveriin ja odottaa mobiililaitetta, joka yrittää muodostaa yhteyttä samalla salasanalla.

Firebasen määritys tapahtui sovelluksen käynnistyksessä samalla tavalla kuin mobiilisovelluksessakin.

Datan haku Firebase-palvelusta tapahtui 100 ms viiveellä, jotta dataväylä ei tukkeudu, sillä tässäkin tapauksessa dataa yritettiin muuten hakea joka näytön päivityskerralla. Jotta saatiin haettua oikea data oikeaan paikkaan, Firebase-pilvipalveluun oli määritetty etukäteen referenssit ”x,y,z”. Näillä määreillä saatiin scriptissä haettua oikea data oikeaan muuttujaan.

Kuvassa 20 tarkistettiin 100 ms välein, oliko data muuttunut pilvessä, ja jos oli, niin haettiin se sieltä ja asetettiin se funktion sisällä muuttujiin ”x,y,z”.

```
timer -= Time.deltaTime;

if (timer <= 0)
{
    if (rx)
    {
        FirebaseDatabase.DefaultInstance
          .GetReference("x")
          .ValueChanged += HandleValueChangedx;

        FirebaseDatabase.DefaultInstance
        .GetReference("y")
        .ValueChanged += HandleValueChangedy;

        FirebaseDatabase.DefaultInstance
        .GetReference("z")
        .ValueChanged += HandleValueChangedz;
    }

    timer = 0.1f;
}
```

Kuva 20. Datan haku Firebase-pilvipalvelusta

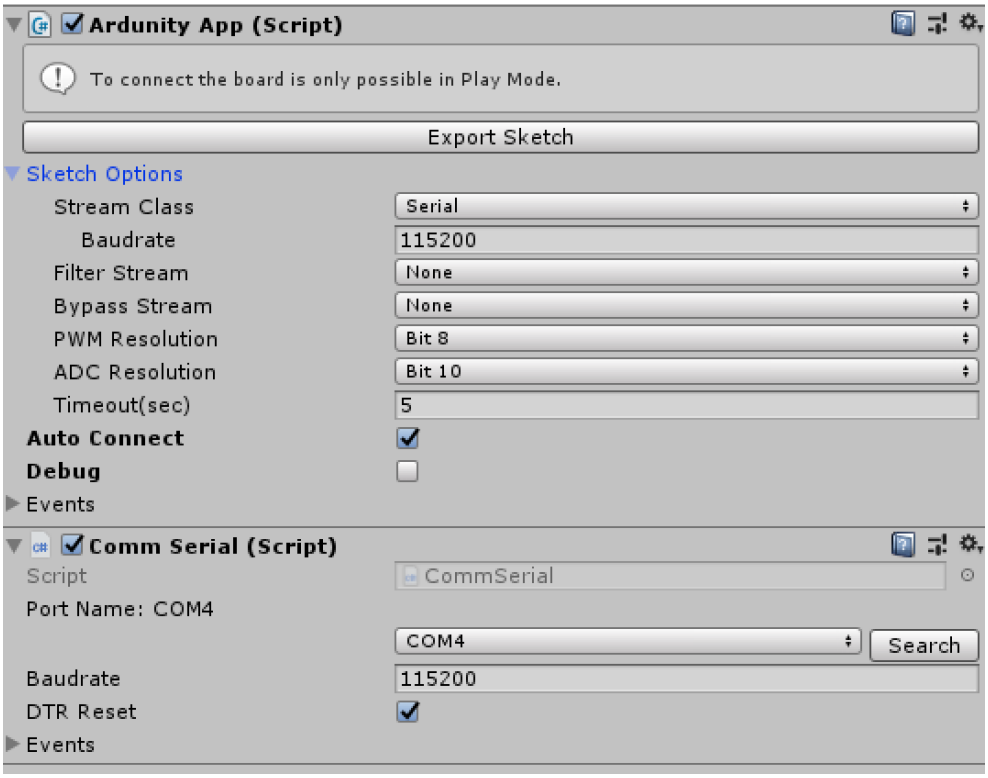
Tässä vaiheessa loin uuden scriptin nimeltä ”DummyRotator.cs”, joka käytännössä muutti Firebasesta tulleet arvot jonkin ”Dummy”-peliobjektin rotaatioksi. Tämä tapahtui joka kuvanpäivityskerralla ”Update()”-funktiossa. Koska pilvestä tulevat arvot olivat string-arvoja, jouduttiin ne parsimaan float-arvoiksi. Kuvassa 21 funktion toiminta.

```
void Update () {

    transform.rotation = Quaternion.Euler(float.Parse(fh.x), float.Parse(fh.y), float.Parse(fh.z));
}
```

Kuva 21. Arvojen muunto rotaatioksi

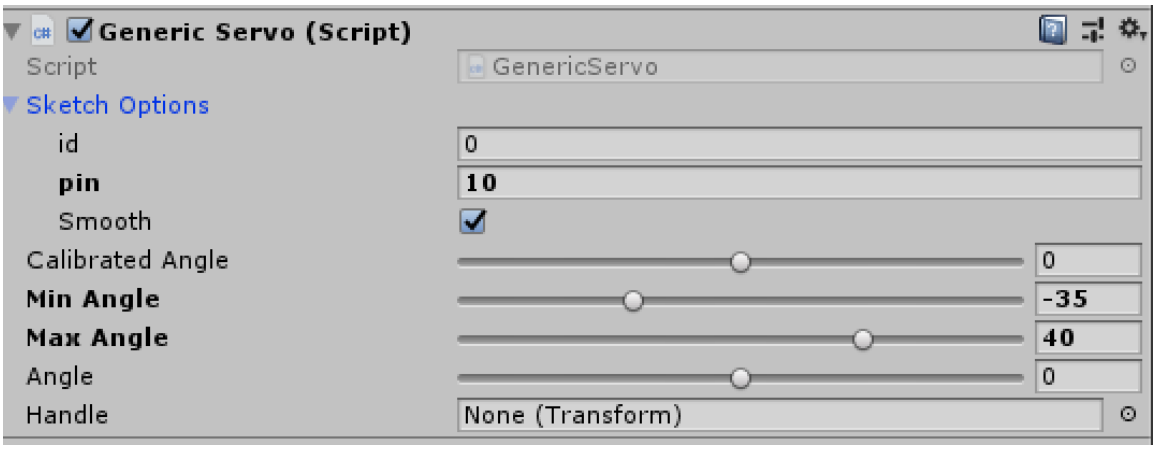
Testasin tässä vaiheessa, toimiiko yhteys ja datansiirto laitteelta toiselle käynnistämällä molemmat sovellukset. Pyörittelemällä puhelinta myös ”Dummy”-peliobjekti pyöri, eli yhteys toimi.

Työn tässä vaiheessa otettiin Ardunity-lisäosa käyttöön, jonka avulla saatiin serial portia pitkin USB-kaapelilla siirrettyä ”Dummy”-peliobjektin rotaatio Arduinolle, joka liikutti servomoottoreita. Laitettiin Unity 3D-editorissa tyhjään peliobjektiin, ”ArdunityApp.cs” niminen scripti sekä ”CommSerial.cs”-scripti (kuvassa 22), joissa määritettiin serial portin asetukset.



Kuva 22. Serial portin asetusten määrittäminen

Seuraavaksi piti määrittää servot, eli montako servoa on käytössä ja mitkä niiden raja-arvot olivat, sekä pinnien numerot Arduino Uno -kehitysalustalla. Kuvan 23 mukaiset asetukset tehtiin kolmelle eri servolle erikseen.



Kuva 23. Raja-arvojen ja pinnien määrittäminen

Kun määritykset oli tehty, saatiin Ardunity-lisäosasta tallennettua tietokoneelle ”.ino”-tiedosto Arduino-kehitysalustaa varten. Tämä tiedosto avattiin Arduino IDE -sovelluksella, joka käännettiin suoraan USB-kaapelin avulla tietokoneessa kiinni olevalle Arduinolle.

Testatessa laitetta ensimmäisen kerran havaittiin, että aikaisemmin määritettyjä raja-arvoja servojen liikkeille piti muuttaa, jotta mikään osa kameralaitteesta ei kääntymisen takia ota mihinkään kiinni ja estä servojen vapaata liikkumista.

11 LOPPUTULOS / JATKOKEHITYS

Tarkoituksena oli tehdä toimiva prototyyppi VR-laseilla ohjattavasta kameralaitteesta. Tarkoituksena oli myös se, että laite toimii 3G/4G-verkossa tai WiFi-yhteydellä. Molemmat näistä ehdoista täyttyvät, mikäli kameralaitteeseen on kytketty esimerkiksi 3G/4G-reititin tai vaikka puhelin, joka jakaa internetyhteyden laitteelle.

Prototyyppi toimi kohtalaisen hyvin ja videokuva sekä liike välittyy pienellä viiveellä, tietysti internet yhteyden nopeudesta riippuen. Joitakin muutoksia jouduin työn aikana tekemään aikarajojen sisällä pysymiseksi, eli stereokameraa en päässyt tähän työhön tekemään kuin kokeiluasteella, ja siinä olisi ollut liian paljon tekemistä tälle aikavälille. Mutta jatkokehitystä ajatellen pääsin toteamaan, että stereokamera on mahdollinen toteuttaa.

Päätin myös käyttää mobiililaitteen gyroskoopin tietojen välitykseen kameralaitteelle Firebase SDK:ta, vaikka luultavasti WebRTC-datakanavat olisivat voineet tarjota myös tämän saman toiminnon. Firebase mahdollistaa todella monipuolisen käyttötarkoituksen niin datansiirron kuin käyttäjähallintaakin, joten siksi se oli oiva valinta.

Totesin myös, että laite oli parempi tehdä kahdella servolla eli käyttäjän pään rotaation tulee huomioiduksi vain kahdelta akselilta. Tämä siksi, että mahdollisen viiveen takia kameralaitteen hallinta meni melko hankalaksi, jos pään kallistukset otettiin huomioon.

Kaiken kaikkiaan lopputulos on jatkokehittämisen arvoinen ja laitehan on ensimmäinen prototyyppi, joten olen tyytyväinen laitteen toimintaan puutteista huolimatta. Jatkokehitystä ajatellen stereokamera olisi erittäin hyvä ominaisuus ja laitetta pitäisi saada pienempään kokoon sekä hiljaisempaan muotoon.

LÄHTEET

1. mobiili.fi, Samsung Galaxy S9+. 2018. https://mobiili.fi/puhelin/samsung_galaxy_s9/ (Haettu 12.1.2019).

2. Wikipedia, Unity (Pelimoottori). 2018. https://fi.wikipedia.org/wiki/Unity_(pelimoottori) (Haettu 12.1.2019).

3. Wikipedia, Microsoft Visual Studio. 2018. https://fi.wikipedia.org/wiki/Microsoft_Visual_Studio (Haettu 12.1.2019).

4. Wikipedia, Node.js. 2019. https://fi.wikipedia.org/wiki/Node.js (Haettu 12.1.2019).

5. FreeCodeCamp.org. 2018. https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5 (Haettu 09.02.2019)

6. C# perusteet. 2016. http://appro.mit.jyu.fi/gko/luennot/luento1/#TOC1 (Haettu 09.02.2019)

7. JavaScript perusteet. 2018. http://appro.mit.jyu.fi/tiea2120/luennot/javascript_basics/#TOC5 (Haettu 09.02.2019)

8. MDN Web Docs. 2018. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction (Haettu 09.02.2019)

9. koulutus.fi, Ohjelmoinnin suosio kasvaa – mikä ohjelmointikieli sopii sinulle? 2018. https://www.koulutus.fi/oppaat/ohjelmoinnin-suosio-kasvaa-mika-ohjelmointikieli-sopii-sinulle-13947 (Haettu 09.02.2019)

10. Firebase.com, Realtime Database. 2019. https://firebase.google.com/products/realtime-database/ (Haettu 11.02.2019)

11. WebRTC.org, What Is WebRTC? 2029. https://webrtc.org/faq/#what-is-webrtc (Haettu 4.3.2019)

12. arduino.cc, Arduino Uno Overview. 2019. https://store.arduino.cc/arduino-uno-rev3 (Haettu 9.4.2019)

13. afterdawn.com, P2P. 2019. https://fin.afterdawn.com/sanasto/selitys.cfm/p2p (Haettu 16.04.2019)