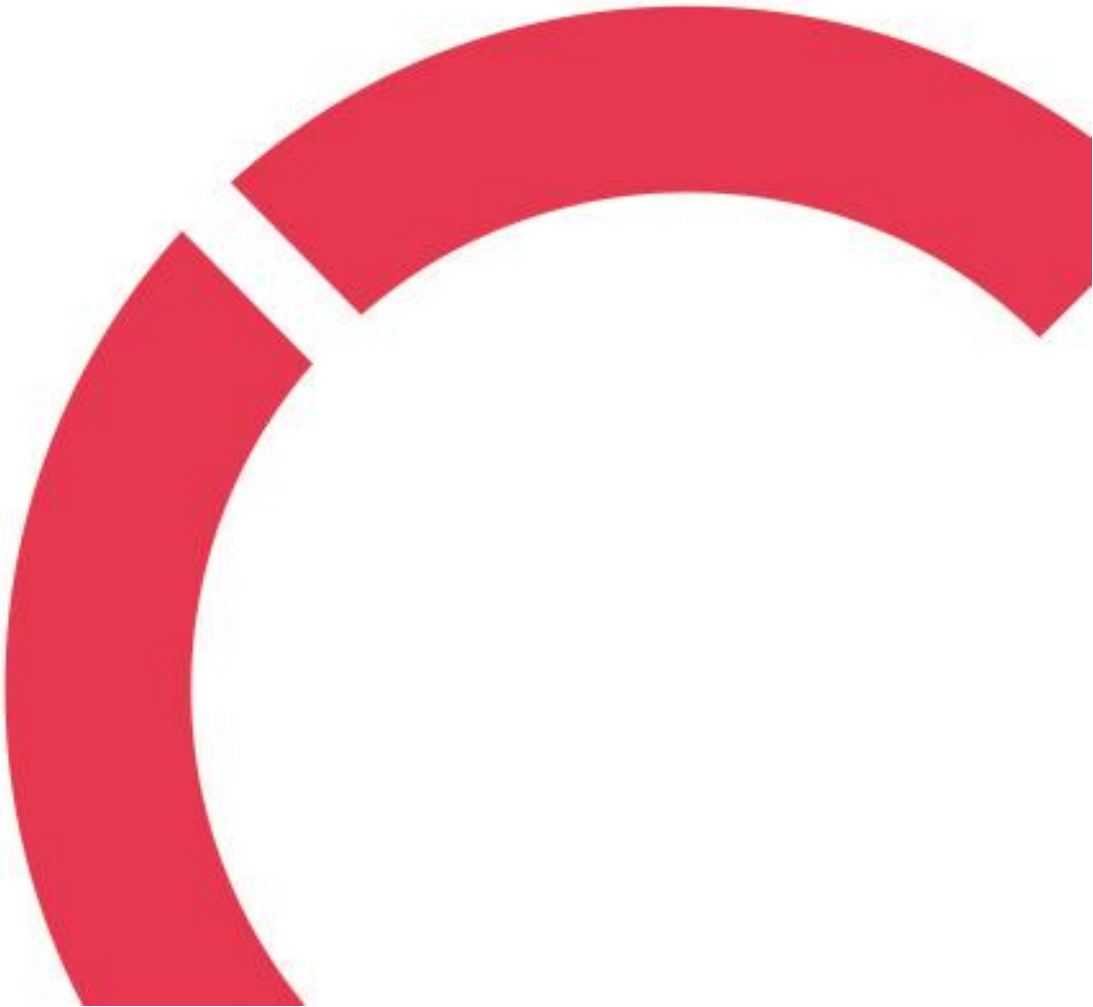
Topi Miettinen

ETÄASIAKASPALVELUJÄRJESTELMÄ

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Insinööri (amk), tieto- ja viestintätekniikka
Toukokuu 2022



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Toukokuu 2022	**Tekijä/tekijät** Topi Miettinen
Koulutus Insinööri (amk), tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi Etäasiakaspalvelujärjestelmä		
Työn ohjaaja Jari Isohanni		**Sivumäärä** 25 + 3
Työelämäohjaaja		

Työn aiheena oli selvittää ja kehittää järjestelmä, jonka avulla pystyttäisiin antamaan asiakaspalvelua etänä siten, että asiakkaalle palvelu on täysin automatisoitu. Asiakkaan ei siis tarvitse tehdä muuta, kuin saapua järjestelmän eteen merkitylle alueelle. Asiakas saisi täten etäyhteyden asiakaspalvelijaan, joka näkisi ja kuulisi asiakkaan; asiakas vain kuulisi asiakaspalvelijan. Järjestelmässä käytettiin automaation ja ohjelmoinnin kombinaatiota. Järjestelmä oli prototyyppi, eli järjestelmää ei jalostettu käyttöönotto-valmiiksi, mutta järjestelmä oli silti pääpirtein toimiva. Järjestelmä koostui Raspberry Pistä, kamerasta, kaiuttimista, mikrofonista, infrapuna-anturista, Chrome-selaimesta sekä internet-yhteydestä. Ohjelmointi tapahtui Python-ohjelmointikielellä.

Työn tavoite oli saada aikaan prototyyppi, jotta voitaisiin todistaa, että konsepti on mahdollinen. Työ onnistui tavoitteessaan hyvin, vaikka työ oli suhteellisen haastava.

Asiasanat
Automaattinen, Asiakaspalvelu, Chrome, Etä, Järjestelmä, Prototyyppi, Python, Raspberry Pi

ABSTRACT

Centria University of Applied Sciences	**Date** May 2022	**Author** Topi Miettinen
Degree programme Engineer, information and communication		
Name of thesis Remote customer service system		
Centria supervisor Jari Isohanni		**Pages** 25 + 3
Instructor representing commissioning institution or company		

The subject of the work was to research and engineer a system that would be able provide customer service remotely in a way that is completely automated for the customer. The customer therefore is only required to arrive on the designated area. The customer thus got remote access to a customer service representative who saw and heard the customer; the customer only heard the customer service representative. A combination of automation and programming was used on the system. The system was a prototype, meaning that the system was not fully developed, but was largely operational. The system consisted of a Raspberry Pi, a camera, a speaker, a microphone, an infrared sensor, Chrome-browser and an internet connection. The programming language was python.

The goal of the work was to engineer a working prototype of the system, in order to prove that the concept is possible. The work was successful in its goal. The work was relatively challenging. In addition to me, there was another student working on the project, who is a student on automation side

Key words
Automatic, Chrome, Customer support, Prototype, Python, Raspberry Pi, Remote, System.

KÄSITTEIDEN MÄÄRITTELY

Bugi
Bugi tarkoittaa jossain ohjelmassa syntyvää ratkaisematonta ongelmaa, joka aiheuttaa ohjelman tai ohjelmien virheellisen toiminnan tai pysähtymisen.

End-to-End
End-to-End tietotekniikassa tarkoittaa kahden laitteen välistä toimintaa, eli kahden päätelaitteen, kuten tietokoneen, puhelimen tai tabletin, välistä kommunikointia.

Import
Ohjelmointikielissä käytettävä tuontikomento, jolla tuodaan ohjelmaan kirjasto, ohjelma tai muu kokonaisuus, joka ei lähtökohtaisesti sisälly ohjelman käyttämään tulkitsijaan tai ohjelmaan.

Open source
Open source tarkoittaa avointa lähdettä. Open source ohjelmat ovat vapaasti käytettävissä ja ohjelman lähdekoodi kaikkien nähtävillä ja muokattavissa.

Prototyyppi
Versio laitteesta tai järjestelmästä, joka ei ole lopullinen versio. Prototyypillä monesti pyritään osoittamaan jokin mahdolliseksi. Se versoisista usein ensimmäinen.

Python
Ohjelmointikieli, jonka avuilla voidaan luoda ohjelmia tai komentoja.

Module
Module, eli moduuli on tiedosto, joka sisältää koodikirjastoja tai funktiota. Näitä moduuleita sitten lisätään toiseen ohjelmiin, jossa kyseisen moduulin sisältöä halutaan käyttää.

Raspberry Pi
Minikokoinen tietokone. Valmistaja on Raspberry Pi.

Syntax
Ohjelmoinnin peruspilari, jonka mukaan ohjelmointia tehdään; syntax määrittelee miten ja missä järjestyksessä mitäkin kirjoitetaan, yleiskielen vastine voisi olla ”kielioppi” ohjelmoinnille. Eri ohjelmointikielessä on eri syntaxit.

1 JOHDANTO

Tämä opinnäyte perustuu ammattiharjoittelussa tehtyyn etäasiakaspalvelujärjestelmään. Varsinainen työ tehtiin ammattiharjoitteluna Centria-ammattikorkeakoulun DUDE-hankkeessa. DUDE-hanke, jossa Centria on mukana, on ympäristö, jossa opiskelijat tai työttömät voivat edistää opintojaan tai työelämään pääsemistä. Työn taustalla oli halu kokeilla, miten onnistuisi antaa asiakkaille asiakaspalvelua etänä. Projektin päämääränä ei ollut saattaa järjestelmää käyttöönottovalmiiksi vaan kokeilla, onnistuisiko se ylipäätään ja millä tavalla.

Pääpaino projektissa oli siinä, että asiakaspalvelun saaminen etänä olisi asiakkaalle mahdollisimman helppoa ja täysin automatisoitua. Asiakkaan ei tarvitsisi kuin saapua etäasiakaspalvelulle merkitylle paikalle. Suurin ongelma olikin se, että asiakaspalvelun pitää olla täysin automaattista asiakkaalle; asiakkaan ei tarvitse painaa mitään tai tehdä mitään muuta kuin saapua merkitylle paikalle. Ongelmaksi muodostui myös se, miten yhteys asiakaspalvelijaan saadaan päättymään asiakkaan lähdettyä, koska ei haluttu, että yhteys olisi koko ajan auki.

Järjestelmä on suhteellisen laaja kokonaisuus niin automaatiota kuin ohjelmointia. Pääpaino on kuitenkin ohjelmoinnissa. Monen asian tässä järjestelmässä pystynee tekemään useallakin eri tavalla. Koska tarkoituksena oli saada prototyyppi aikaiseksi, kaikkia mahdollisuuksia ei käyty läpi. Lähdemateriaalina on käytetty eri foorumeilta löytyneitä Python-ohjelmia ja komentoja, sekä Linux-ohjekirjaa ja Pythonin eri syntaksien kehittäjien omia ohjemanuaaleja. Järjestelmä on kasattu olemassa olevista palasista halutuksi kokonaisuudeksi. Näistä syistä järjestelmää pystyy jatkojalostamaan osissa. Järjestelmästä on tehty dokumentaatio, minkä vuoksi kaikesta tästä on myös tehty käyttöönottodokumentti. Dokumentin sekä käytettyjen tiedostojen avulla pitäisi pystyä rakentaa prototyyppi uudelleen, sekä sitten halutessa jatkojalostaa sitä. Ensin työssä esitellään mitä asiakaspalvelu on ja miten etäasiakaspalvelu eroaa siitä. Tämän jälkeen esitellään järjestelmän yleisesti ja lopuksi erittelen tarkemmin yksittäisiä osia ja niiden toimintaa tarkemmin. Myös käytettyjen tekniikoiden teoriaa esittelen hieman. Teoriaa tässä opinnäytteessä on suhteellisen vähän, koska tämä on konkreettinen järjestelmä, joka perustuu pitkälti ohjelmointiin sekä jo olemassa oleviin alustoihin.

2 ETÄASIAKASPALVELU

Etäasiakaspalvelulla tarkoitetaan asiakaspalvelun antamista etänä ja virtuaalisesti tietotekniikan avulla. Perinteisestä asiakaspalvelusta etäasiakaspalvelu eroaa siten, että asiakas ja asiakaspalvelija eivät ole samassa tilassa, eivätkä yleensä edes samassa rakennuksessa. Vuoropuhelu tapahtuu yleensä tietokoneen, mikrofonin, kaiuttimen, kameran ja internetin välityksellä. Asiakaspalveluun yleisesti voi sisältyä muutakin kuin vuoropuhelua. Tässä opinnäytetyössä ei keskitytä kuin siihen, miten tämä vuoropuhelu saadaan käyntiin ja lopetettua asiakkaalle täysin automaattisesti.

2.1 Asiakaspalvelu yrityksissä

Yleisesti asiakaspalvelu on yritykselle tärkeää. Se on näkyvä osa yritystä. Se voi olla myös markkinointikeino, jolla yritys pyrkii saamaan lisää asiakkaita; tieto hyvästä asiakaspalvelusta kulkee asiakkaalta toiselle. Hyvä asiakaspalvelu myös tuo yritykselle joustavuutta toimintaan. Asiakkaan tulee pystyä jopa tuohtuneena tuomaan ongelmansa tai kysymyksensä ilmi, koska se lisää mahdollisuutta, että asiakas jatkossa käyttää yrityksen palveluita tai suosittelee niitä muille. (Routamaa-Päiviö 2021.)

2.2 Asiakaspalvelu tässä järjestelmässä

Vaatimukset asiakaspalvelun suhteen tälle projektille olivat suhteellisen yksinkertaiset, ja ne keskittyivät ainoastaan tietotekniikan alueelle: tämän vuoropuhelun aloittamisen pitää olla täysin automaattinen asiakkaalle. Asiakaan tarvitsee vaan saapua paikalle ja kertoa asiansa. Vaatimuksena tekniselle puolelle oli myös se, että asiakas pystytään näkemään ja kuulemaan, kun taas asiakkaan tarvitsee vain kuulla asiakaspalvelija. Pääpaino tässä järjestelmässä olivatkin juuri asiakkaat ja heidän reaktionsa etäasiakaspalveluun. Äänen ja kuvan laatuun ei kiinnitetty huomiota. Yhteyden tulee avautua ja sulkeutua asiakkaalle automaattisesti.

3 ETÄASIAKAPALVELUN TEKNINEN SUUNNITTELU

Järjestelmän teknisessä suunnittelussa vastattiin kysymyksiin, jotka liittyivät siihen, miten ja millä saadaan järjestelmän vaatimukset täytettyä. Näitä kysymyksiä olivat esimerkiksi: ”Mikä päätelaite on sopiva tähän tarkoitukseen?”, ”Mikä arkkitehtuuri tai sovellutus on sopiva audiovisuaaliseen yhteyteen asiakkaan ja asiakaspalvelija välillä”, ”Miten audiovisuaalinen yhteys saadaan avattua ja suljettua?”. Suunnitellussa edettiin yksi askel kerrallaan vaatimuksien täyttämistä kohti. Erilaisia tapoja kokeiltiin ennen lopullista versiota prototyypistä. Internettiä tutkimalla löysin eri tapoja, joilla saavuttaa haluttu päämäärä ja vastata kysymyksiin. Tämän jälkeen ohjelmoitiin toteutus, jonka jälkeen toimivuutta testattiin fyysisesti. Mikäli toteutus toimi halutusti, siirryttiin seuraavaan tavoitteeseen. Tarvittaessa myös palattiin myöhemmin takaisin, mikäli myöhemmin jokin asia ei onnistunut aiemmat toimintatavan takia. Näin edettiin, kunnes suhteellisen toimiva prototyyppi oli valmis.

Järjestelmän tietoteknisessä suunnittelussa oli vapaa valinta tekniikoihin ja arkkitehtuureihin, koska kyseessä oli prototyyppi ja järjestelmän tilaaja ei määritellyt muuta kuin halutut tavoitteet. Usein teknisessä suunnitellussa on käytössä tietyt arkkitehtuurit ja ohjelmat, mutta tässä projektissa niin ei ollut.

4 TILAT

Vaatimuksia siitä, minkälainen tila järjestelmälle pitäisi olla, ei ollut. Tätä järjestelmää voidaan periaatteessa käyttää missä tilassa tahansa. Tämä projektin vaatimukset liittyivät vain järjestelmän kokonaisuuteen.

Akustiikkaan vaikuttavat monet asiat kuten:
- Huoneen rakennusmateriaali
- Huoneen muoto ja mittasuhteet
- Huoneen koko
- Akustiikka-paneelit
- Äänilähteiden sijainti
(Gaggia. 2022.)

Huoneen, eli tilan, koko vaikuttaa ääniaaltojen heijastumiseen, huoneen äänieristyksen määrään sekä jälkikaiunnan määrään. Huoneen muoto ja mittasuhteet vaikuttavat ääniaaltojen resonanssitaajuuksiin ja diffuusioon (äänienergian tasaiseen leviämiseen). Huoneen rakennusmateriaali vaikuttaa äänienergian imeytymiseen riippuen ääniaaltojen taajuudesta. Akustiikka-paneelit lisäävät äänieristystä. Äänilähteiden sijainti vaikuttaa itse ääniaaltoihin ja niiden energioihin. (Gaggia. 2022.)

Optimaalinen tila olisi äänieristetty pienehkö huone. Huoneen tulisi olla hyvin valaistu. Huoneessa pitäisi olla pistorasioita. Tilan olisi hyvä olla äänieristetty siitä syystä, että puhe mikrofoniin olisi mahdollisimman selkeä. Myös tulevan äänen kuulisi selkeämmin, kun ulkoapäin ei tulisi ääniä. (Gaggia. 2022.)

Myös olisi hyvä, että järjestelmä sijoitettaisiin sellaiseen paikkaan, jossa järjestelmän eteen fyysisesti ei pääse ylimääräisiä ihmisiä tai eläimiä. Syy tähän on se, että järjestelmä ottaa asiakaspalvelijaan yhteyden aina kun sensori havaitsee infrapuna-aaltoja (mikäli järjestelmä ei ole jo käytössä); vältyttäisiin vahingollisilta yhteydenotoilta. Lisäksi huoneen pitää olla hyvin valaistu, koska järjestelmässä on webbikamera. Webbikamera vaatii hyvän valaistuksen, jotta siitä saadaan selvää kuvaa.

5 KÄYTETYT ARKKITEHTUURIT JA OHJELMISTOT

Järjestelmässä hyödynnetään useampaa valmista arkkitehtuuria tai ohjelmistoa; tämä nopeutti ja helpotti tavoitteisiin pääsemistä. Tämä oli mahdollista, koska järjestelmälle ei asetettu vaatimuksia arkkitehtuurien tai tekniikoiden suhteen. Järjestelmässä käytettiin pääpiirtein vain ilmaisia arkkitehtuureja ja tekniikoita.

5.1 Python

Pääohjelmointikieleksi valikoitui python. Syy siihen oli, että python on erittäin laaja ja monipuolinen ohjelmointikieli. Siihen oli tuhansia eri kirjastoja ja laajennusmoduuleita. (Python Software Foundation 2022a.)

Sillä oli myös mahdollista ja suhteellisen yksinkertaista suoraan käyttää tai ohjata toisia ohjelmia, kuten järjestelmässä käytettävää Chrome-selainta. Pythonin kyky ja helppous automatisoida asioita oli iso osa valintaprosessia; python tukee esimerkiksi Selenium- moduulia, jota käytin projektissa. (Python Software Foundation 2022b.)

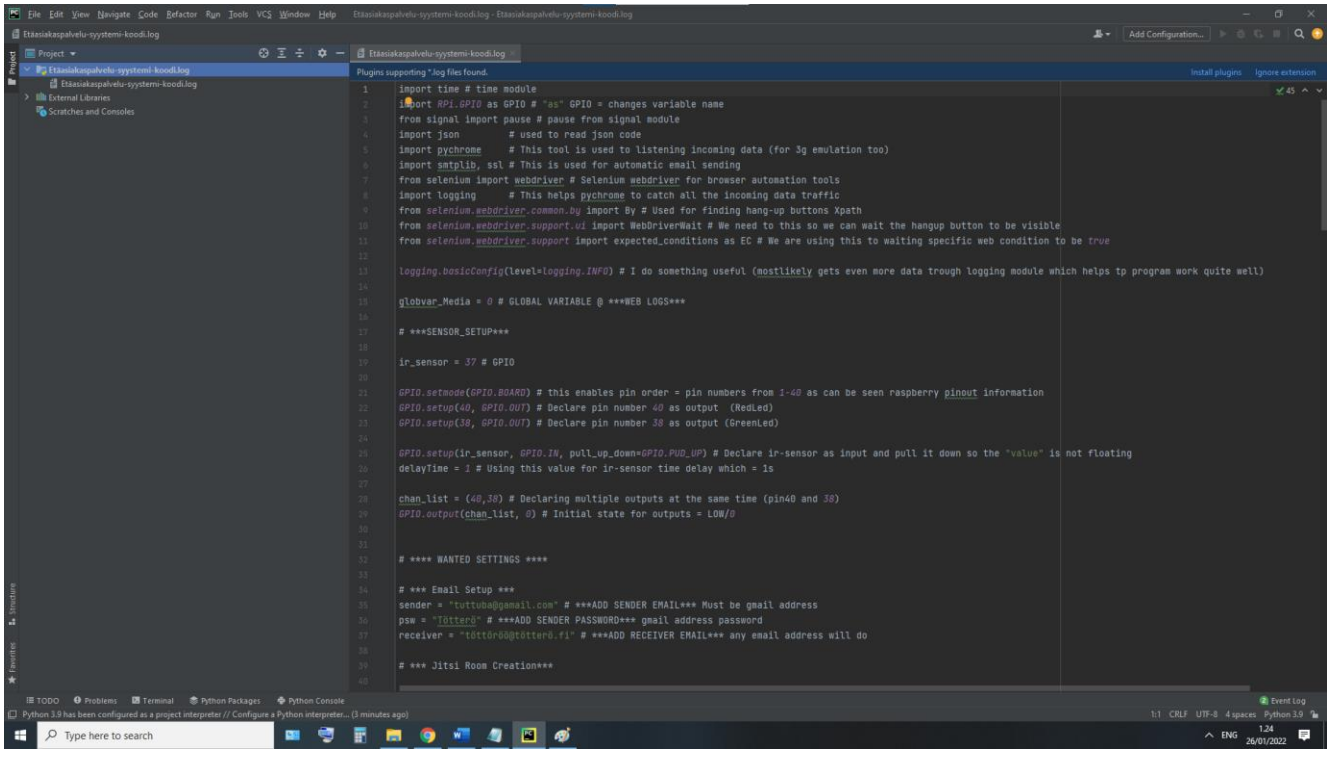
Python ohjelmointikielenä on yksinkertainen ja voimakas. Siinä on tehokkaat datarakenteet ja olio-ohjelmointi. Pythonin dokumentointi on erittäin laajaa. Pythonin dokumentit esittelevät tarkoituksensa ja toimintatapansa lukijalle, mikä tekee Pythonista helppokäyttöisen. (Python Software Foundation 2021a.)

Python-ohjelmia kehittäessä ei tarvitse käyttää useaa eri kirjastoa, koska pythonin oma kirjasto sisältää monta eri komponenttia. Pythonin kirjasto sisältää tietotyyppejä, jotka monissa kielessä kuuluvat kyseisten kielien perustaan, kuten esimerkiksi numerot ja luettelot. Näissä tapauksissa python määrittää literaalien muodon ja asettaa rajoituksia niiden semantiikkaan; python ei kuitenkaan määrittele täysin semantiikkaa. Pythonin kieliydin määrittelee syntaksien ominaisuudet. Lisäksi python-kirjasto sisältää useita eri sisäänrakennettuja funktiota ja poikkeuksia, joita voidaan käyttää vapaasti ilman erillistä importoida. Koska pääosa pythonin kirjastosta koostuu erinäistä moduuleista, niin useimmat niistä pitää importoida erikseen. (Python Software Foundation 2021b.)

5.2 Pycharm

Ohjelmointiympäristönä projektissa käytettiin Pycharm-nimistä ohjelmaa. Pycharm on ilmainen (siitä on olemassa myös maksullinen versio) ja vapaakäyttöinen. Sillä se on isohko ohjelma, jolla pystyy ohjelmoimaan useita eri ohjelmointikieliä, kuten Python, Javascript, Coffeescript, Typescript, HTLM, AngularJS, NodeJS ja CSS. Pycharm sisältää eri tulkitsijoita, kuten Python 3.9, joka tulkitsee syötetyn tekstin Python-ohjelmaksi. Pycharmiin pystyy asentamaan erittäin paljon lisäosia eli erillisiä lisäohjelmia. Ne toimivat sitten Pycharmissa. Pycharmin IDE (Integrated Development Enviroment, eli ohjelman kehitysympäristö) sisältää koodauksessa hyödyllisiä elementtejä kuten virheiden huomaaminen, tekstin ennakoinnin sekä koodin korjausehdotuksia. Lisäksi ohjelmassa on tekstinhaku-mahdollisuus, jolla pystyy hakemaan tekstistä tiettyä sanaa ja sen paikkaa tekstissä. Pycharmissa on myös live-edit-mahdollisuus, jonka avulla pystyy näkemään muutokset reaaliajassa. (Jetbrains 2022.)

Pycharm, Pythonin lisäksi tarjoaa ensiluokkaisen tuen webbikehitykselle; tästä syystä valikoitui Pycharm projektini ohjelmointiohjelmaksi. Pycharmin tuki muun muassa Chromedriver-lisäohjelmalle oli hyvin oleellinen tässä projektissa, koska sen avulla saadaan Chrome-selaimella audiovisuaalinen yhteys automaattisesti auki.



KUVA 1. Pycharmin ulkoasu omassa projektissani.

5.3 Jitsi Meet

Äänen ja kuvan välitykseen käytettiin Jitsi Meet alustaa. Se on käytettävissä Jitsin omilla verkkosivuilla: https://jitsi.org/jitsi-meet/. Se on ilmainen ja täysin vapaasti käytettävissä sekä muokattavissa. Sen käyttöön ei tarvitse tiliä, mutta Google- tai Microsoft-tilin voi yhdistää Jitsi Meetin käyttöön. Ilmaisessa Jitsi Meetissä on seuraavat ominaisuudet: HD-tasoinen ääni ja video, 100 samanaikaisen käyttäjän tapaaminen, käyttäjien välinen salaus, usean käyttäjän samanaikainen näytönjako, osallistujien tietokoneiden etäkäyttö sekä mahdollisuus liittää Google- ja Microsoft-tilit Jitsi Meetiin. (8x8, Inc. 2022.)

Jitsillä on myös maksullinen versio alustasta, sen nimi on 8x8 Meet. 8x8 Meet:issä on, Jitsi Meetin perusominaisuuksien lisäksi lisäominaisuuksina soittamisen mahdollisuus, lokitiedot, Youtube-live streamaus, automaattinen englanninkielinen tekstitys ja sääntöjen asettaminen. Tätä versiota ei testattu tässä projektissa. (8x8, Inc. 2022.)

Jitsi on ilmainen kokoelma avoimen lähdekoodin alustoja. Näiden alustojen, projektien, avulla pystytään rakentamaan ja ottamaan käyttöön videoneuvotteluratkaisuja. (8x8, Inc. 2022.)

Jitsi-projektin sai alkunsa 2003, kun Emil Ivanov teki SIP Communicatorin. Vuonna 2005 SIP Communicator-arkkitehtuuri uusittiin täysin, uudella Open Service Gateway Initiativ-mallilla, jotta liitännäisiä olisi helpompi lisätä alustaan. 2007 SIP Communicator sai ensimmäisen Wikipedia-sivunsa. Vuonna 2008 SIP Communicatoriin lisättiin End-to-End-salaus ZRTP:n avulla. Vuonna 2009 Emil Ivanov ja Yana Stamcheva perustivat BlueJimp-yrityksen, joka työllistää Jitsin pääedistäjät. Vuonna 2011 SIP Communicator-nimi vaihdettiin Jitsiin (”Jitsi” bulgaariaksi tarkoittaa ”johdot). 2012 Jitsi lisäsi mahdollisuuden videoneuvotteluihin, jotka perustuvat videostriimien reitittämiseen. Vuonna 2013 Jitsi kehitti Jitsi Videobridgen, jonka avulla yhdestä paikasta voi löytää tuhansia videostriimejä. Vuonna 2014 Jitsi kehitti Jitsi Meetin, joka internetin välityksellä toimiva videoneuvottelualusta. Vuonna 2015 Atlassian hankki BlueJimpin, ja teki pitkäaikaisen investoinnin pitääkseen Jitsin täysin ilmaisena ja open sourcena. Vuonna 2018 8x8 hankki Jitsin. 2020 Jitsillä on kuukausittain yli 20 miljoonaa aktiivista käyttäjää, ja Jitsiä tarjotaan yhtenä 8x8:n palveluratkaisuna. (8x8, Inc. 2022.)

6 LAITTEISTOT

Tässä kappaleessa kerron käytetyistä hardwareista, eli fyysisistä laitteista, jota järjestelmässä käytetään sekä niiden asennuksista. Lisäksi kerron yleistä teoriaa näihin liittyen.

6.1 Raspberry Pi

Raspberry Pi on minikokoinen tietokone. Raspberry Pistä on useampia eri versiota. Kaikki koostuvat yhdestä piirilevystä. Niiden pääkäyttöinen käyttökohde on harrastukset sekä opetuskäyttö, johtuen niiden pienikokoisuudesta, yksinkertaisuudesta sekä hinnasta. Raspberry Pin hinta, riippuen mallista ja varustuksesta, on nykypäivänä yksityiselle käyttäjälle noin 30–100 euroa.

Esimerkiksi Raspberry Pi 3 model A+ sisältää seuraavat komponentit:

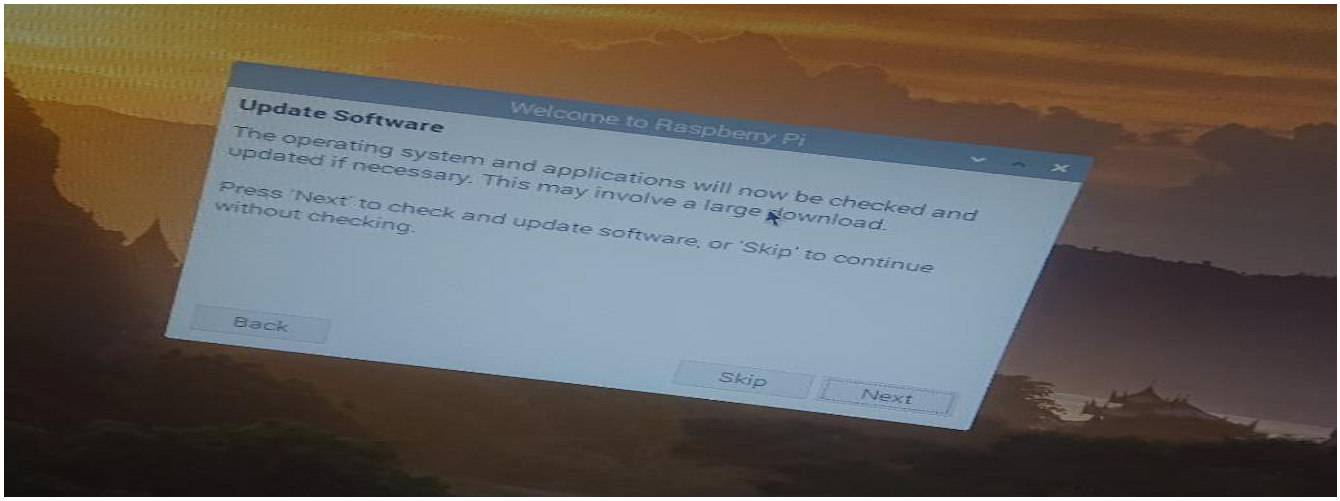
- 1,4 GHz Cortex-A53 64-bit prosessori
- 512 MB SDRAM sisäistä muistia
- Bluetooth vastaanotin
- Langattoman verkon vastaanotin
- 40-pin GPIO header
- Usb-portteja
- CSI-kameraportti
- DSI-näyttöportti
- HDMI-portti
- Composite video-port
- 4-Pole stereo output
- Micro Sd-portti
- 5V/2,5A DC-virtaliitin

(Raspberry Pi Foundation 2021a.)

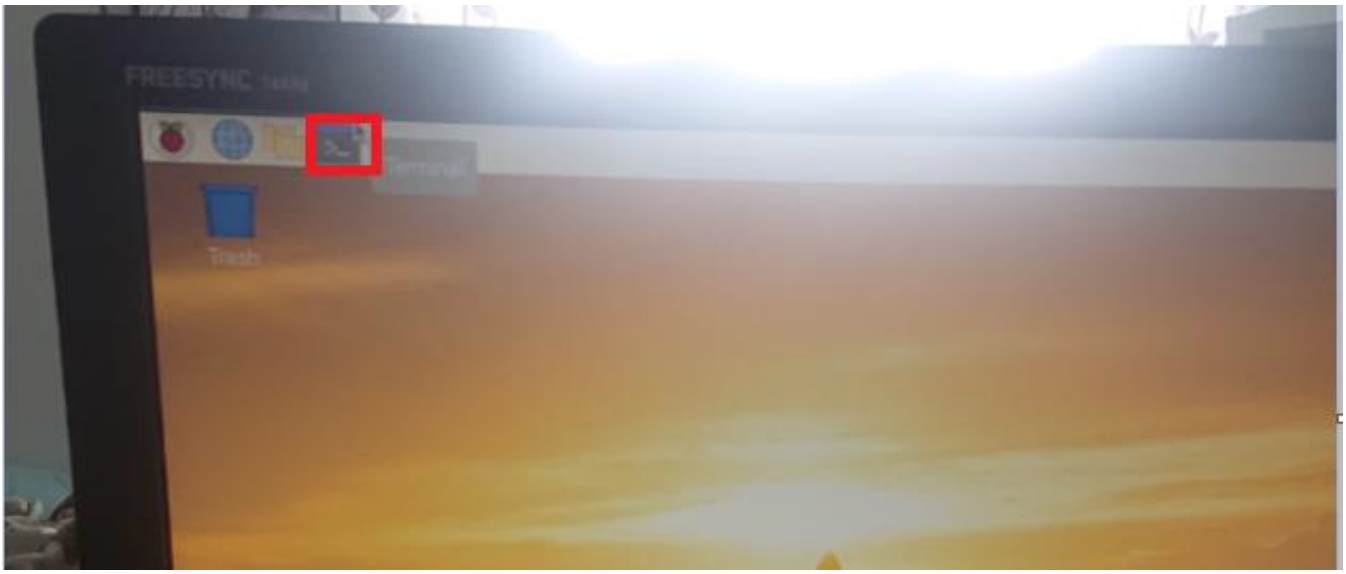
6.1.1 Asennus

Normaaleiden tietokoneiden tapaa pitää Raspberry Pihin, asentaa jokin käyttöjärjestelmä ennen kuin sitä pystyy normaalisti käyttämään. Raspberry Pihin on hyvä laittaa näyttö, näppäimistö sekä hiiri kiinni tässä kohtaa. Ensimmäiseksi pitää ladata käyttöjärjestelmä, esimerkiksi Raspberry os (operating system), SD-kortille. Ohjeet tämän vaiheen suorittamiseen löytyvät Raspberryn Pin omilta kotisivuilta: https://www.raspberrypi.org/documentation/installation/installing-images/. Tämän vaiheen jälkeen pitää valita, millä metodilla Raspberry Pihin tuodaan internet-yhteys, jotta saadaan asennettua päivitykset käyttöjärjestelmän asennuksen jälkeen.

Tässä vaiheessa seurataan ohjeita ja suoritetaan ensiasennukset (KUVA 2) sekä käyttöjärjestelmän asennus. Kuvaohjeet ovat kokonaisuudessaan liitteessä 1. Tämän jälkeen avataan Raspberry Pin terminaali (KUVA 3).



KUVA 2. Raspberry Pi:n ensiasennuksen aloitus.



KUVA 3. Raspberry Pi:n terminaalin avaaminen.

Seuraavaksi tehdään näppäimistön asetukset. Ohjeet tähän ovat liitteessä 1. (Liite 1).

Sitten voidaan poistua asetusvalikosta painalla Esc-näppäintä. Käynnistetään Raspberry Pi uudelleen kirjoittamalla ”sudo reboot” terminaaliin ja painetaan enteriä, jolloin saadaan uudet asetukset heti käyttöön uudelleenkäynnistyksen jälkeen. Seuraavaksi asennetaan projektiin tarvittavat lisäosat ja päivitykset terminaalin kautta. Asennettavat lisäosat ovat Selenium, Chromedriver ja Pychrome. Kuvallinen ohjeistus on liitteessä 2. (Liite 2)

6.1.2 Raspberry Pi etäyhteyden muodostaminen

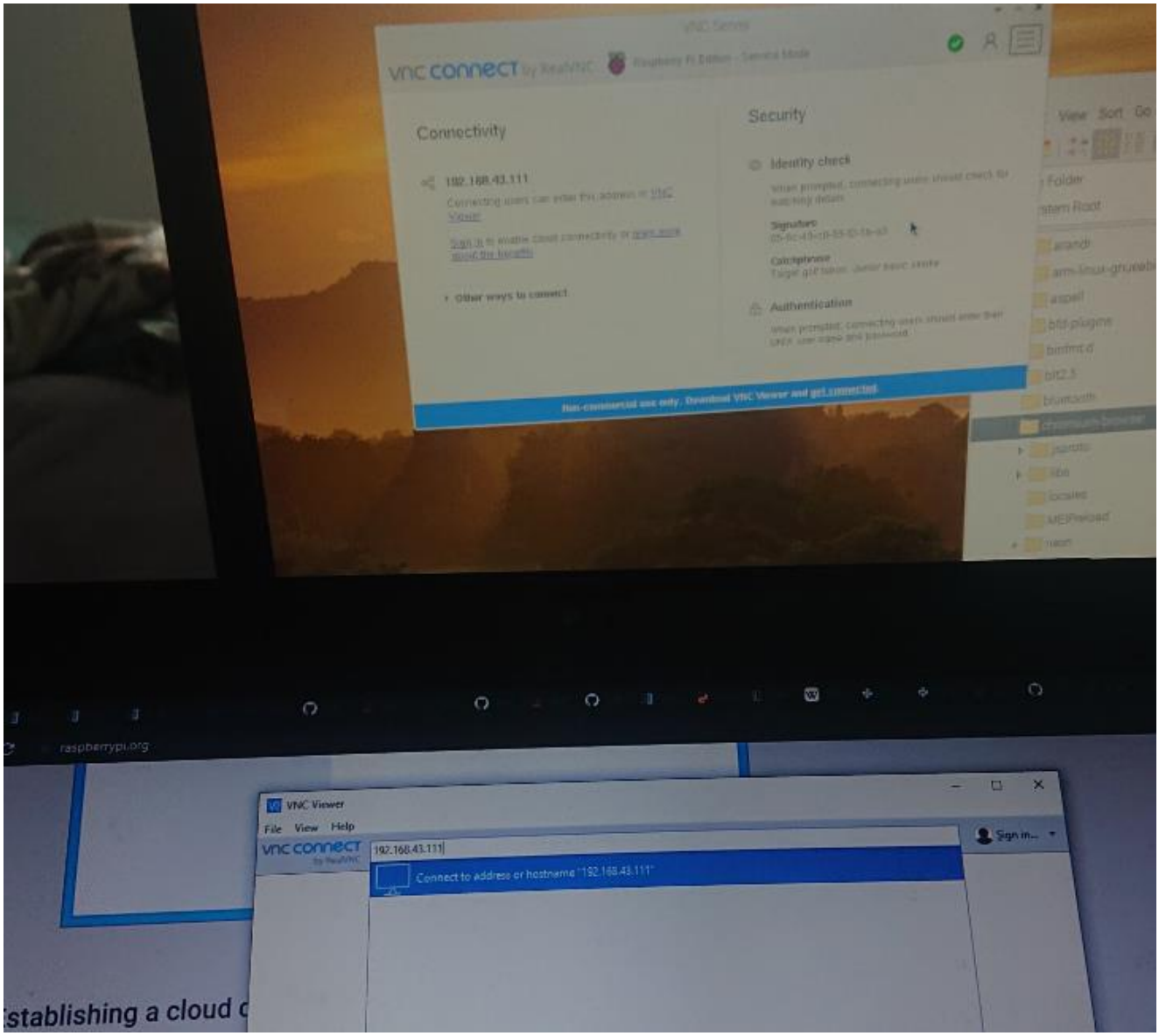
Vaatimukset Raspberry Piihin etäyhteyden muodostamiseksi ovat VNC Viewer-ohjelma etätietokoneella sekä VNC Server Raspberry Pissä. Lisäksi päätelaitteiden (Raspberry sekä etätietokone) pitää olla päällä sekä niissä pitää olla internetyhteys. Tarkat ohjeet asennuksiin ja käyttöönottoihin löytyvät sivulta https://help.realvnc.com/hc/en-us/articles/360002249917-VNC-Connect-and-Raspberry-Pi#setting-up-your-raspberry-pi-0-0. VNC on ilmainen opetustarkoituksessa ja yksityisessä käytössä.

VNC Serverin asennus Raspberry Pille tapahtuu avaamalla Raspberry Pin terminaali, ja kirjoittamalla sinne ”sudo apt-get update” sekä ”sudo apt-get install realvnc-vnc-server”, jonka jälkeen painetaan enter. Tämän jälkeen terminaaliin kirjoitetaan ”sudo raspi-config” ja painetaan enter; avautuneesta valikosta valitaan hiirellä klikkaamalla ”Interfacing options”, ”VNC” ja viimeisenä ”Yes”. Nyt VNC Server käynnistyy Raspberry Pissä aina kun Raspberry Pi käynnistyy. Ilman tätä ohjelmaa ei Raspberry Pihin saada VNC Viewerillä etäyhteyttä.

Kun etätietokoneelle on asennettu, avataan VNC Viewer. Sitten kirjoitetaan hakuriville Raspberryn Pin oma IP-osoite, sekä portti, johon yhteys halutaan muodostaa (oletusportti on 5900). Esimerkki kirjoitusasusta hakuriville: 99.20.236.9:5900. ”99.20.236.9” on IP osoite, ”5900” on portti, välissä on kaksoispiste.

Optionaalisesti olisi hyvä olla RealVNC-tili, joka sisältää sähköpostin ja salasanan. Tällöin VNC Viewerin avaamisen jälkeen tarvitsisi vain klikata sign in, syöttää tilin tiedot, ja paina sign in. Tällöin käy-

tetyt IP-osoitteet sekä portit jäävät varmasti muistiin ja olisivat kirjautumisen jälkeen heti käytettävissä. Lisäksi silloin voi tallentaa osoitteet sekä portin jollakin nimellä, kuten ”Esa Esimerkin Raspberry Pi”. Kuvassa 4 on kuva VNC Viewerin näkymästä.



KUVA 4. Lisätään Raspberryn oma IP-osoite VNC Vieweriin ja kokeillaan onnistuuko yhteyden muodostaminen. Toinen vaihtoehto on luoda tili, jolla luodaan pilvipalvelu yhteys Raspberryyn. Tämä takaa jatkuvasti staattisen (ei muuttuvan) IP-osoitteen.

Lisäksi joissakin tapauksissa pitää Raspberry Piin tulevat yhteydenotot VNC Serverin kautta hyväksyä erikseen. Tämä on tietoturvallisuuden kannalta hyvä ominaisuus. Tämän voi myös kytkeä pois päältä. Ohjeet tähän asiaan löytyvät tältä sivulta: https://help.realvnc.com/hc/en-us/articles/360002250457-Requiring-Connection-Approval-From-a-Remote-Computer-Owner-.

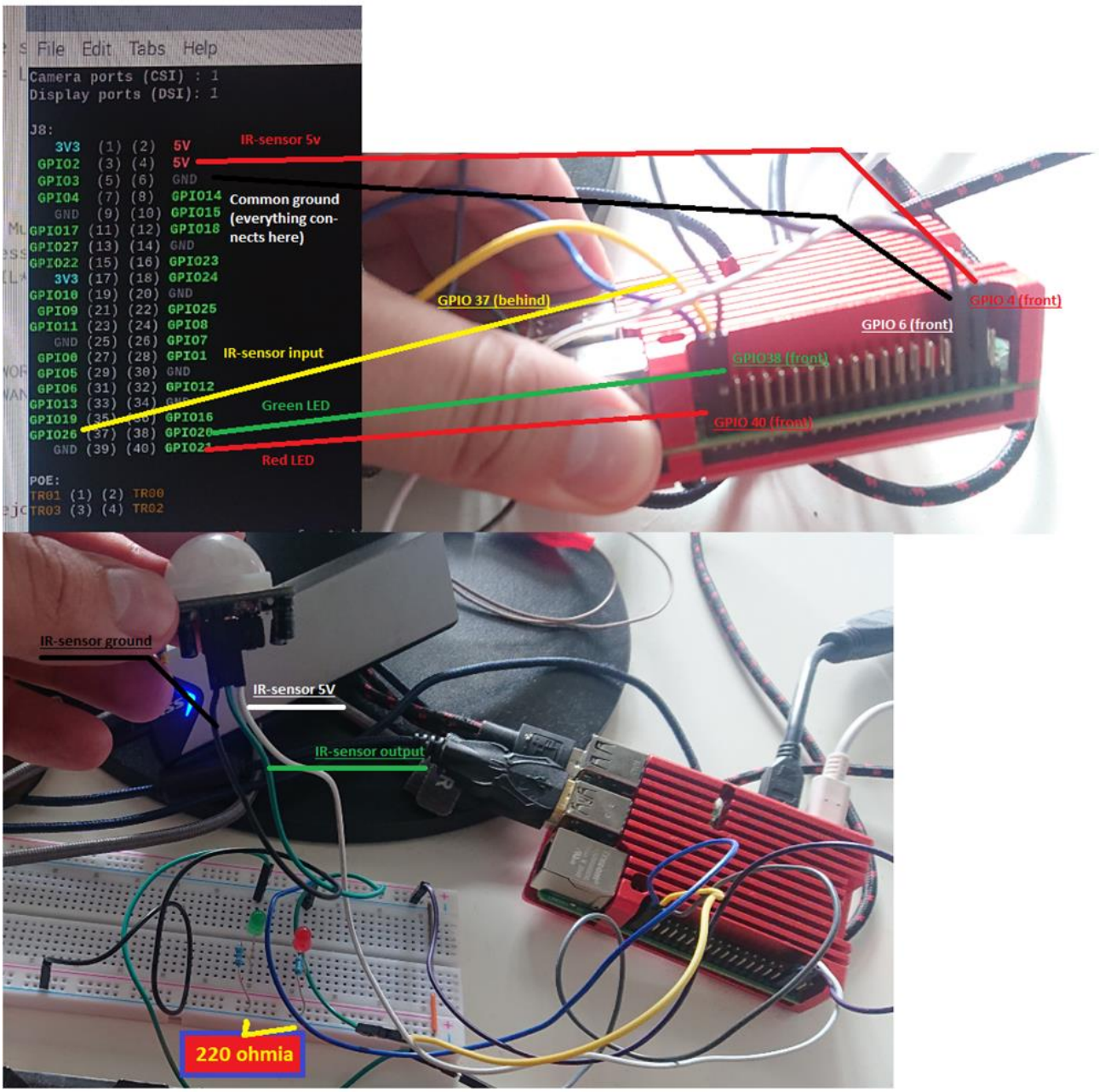
Etäyhteys helpottaa Raspberryn seuraamista bugien varalta ja mahdollistaa muokkauksien tekemistä internetyhteyden kautta. Ilman etäyhteyttä Raspberryn ei Raspberryn mitään asetuksia voida muuttaa etänä.

6.2 Infrapuna-anturi

Infrapuna-anturi on pieni elektroninen laite, joka havaitsee liikettä tai mittaa lämpötilaa. Infrapuna-antureita on kahdenlaisia: passiivisia ja aktiivisia. Passiiviset infrapuna-anturit eivät lähetä infrapuna-aaltoja, vaan pelkästään vastaanottavat niitä. Aktiiviset infrapuna-anturit puolestaan myös lähettävät aktiivisesti infrapuna-aaltoja. Molemman tyyliset anturit siis tulkitsevat niille tulevia infrapuna-aaltoja. (Jf-parede. 2022.)

Etuja infrapuna-anturissa ovat melu-, korroosio- ja hapetusresistanssi, ei tarvitse fyysistä kosketusta, vähäinen virrankulutus, ei tarvitse valoa. Haittoja puolestaan ovat pieni tiedonsiirtonopeus, suora näköyhteys vaaditaan, kantama on rajallinen. (Jf-parede. 2022.)

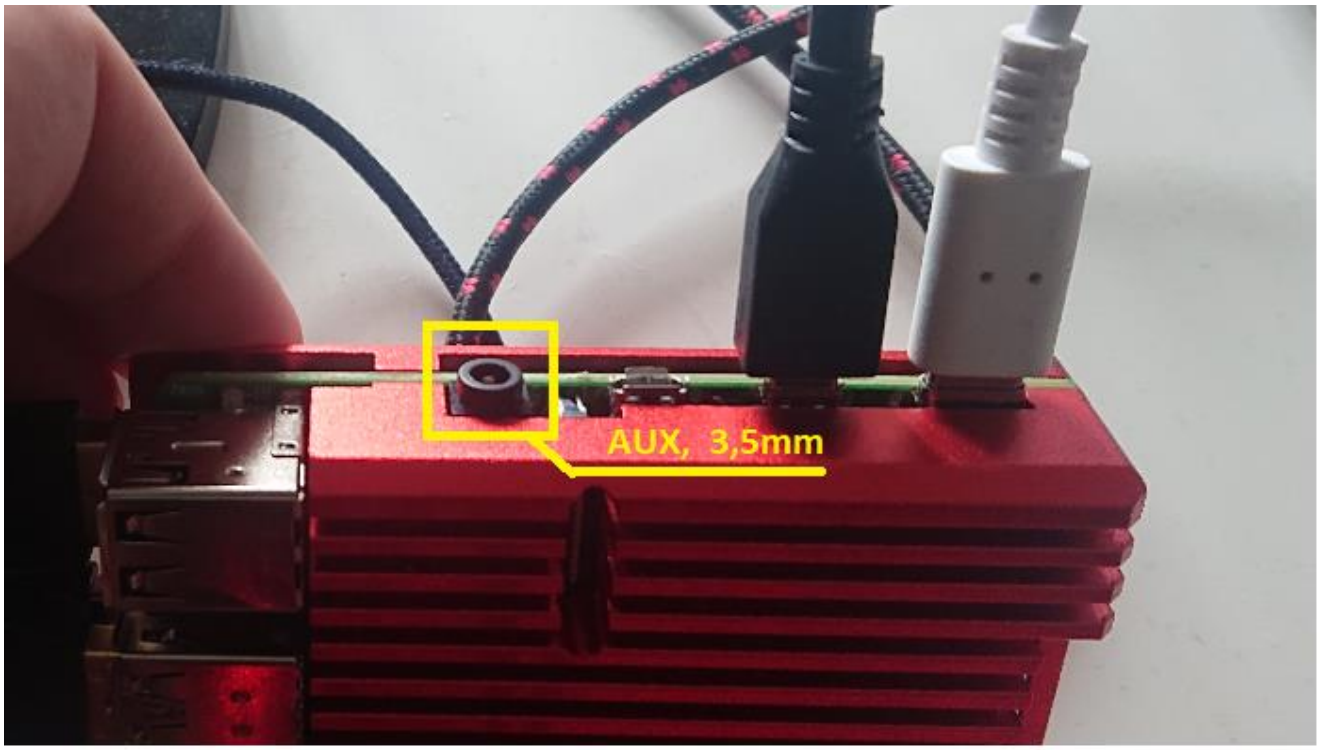
Infrapuna-anturin kytkettiin Raspberry Pihin sekä 220Ω-vastuksiin. Infrapuna-anturi oli 5V. Pinnien liitännät toimivat siten, että oikealla puolella olevat parilliset arvot ovat eturivillä ja parittomat arvot ovat takarivillä. Liitännät on merkitty mahdollisimman helppolukuisella tavalla, jotta vältytään vahingoilta (KUVA 5).



KUVA 5. Tässä kuvassa näkyy kytkentöjen sijainti sekä koodin nimike. Led-vastuksien arvot ovat myös nähtävillä.

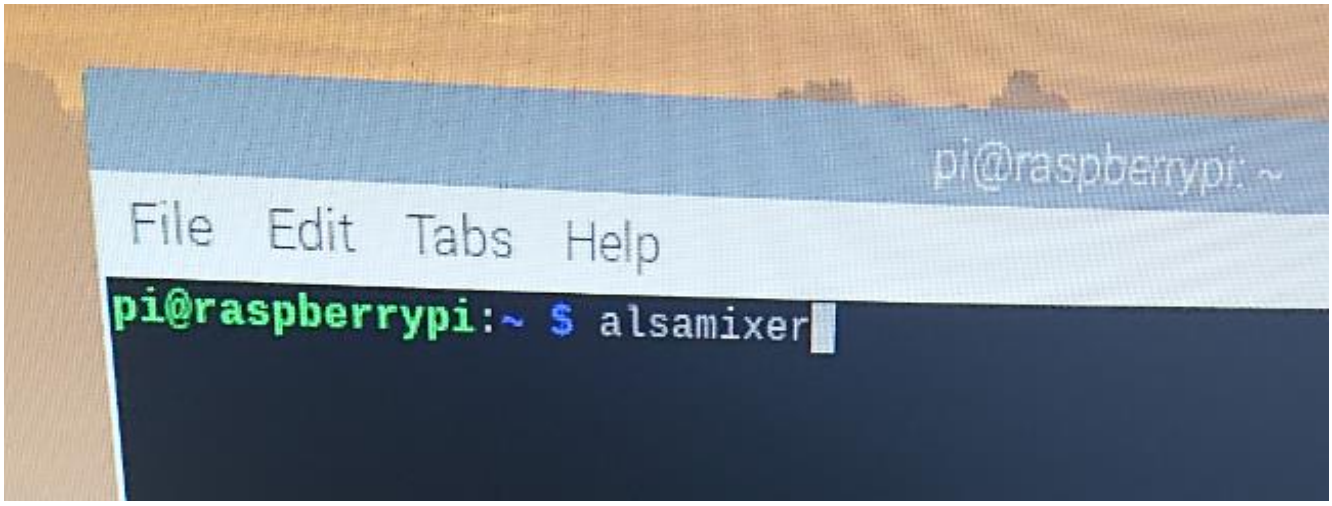
6.3 Äänentoisto

Prototyyppi-järjestelmässä äänentoistossa käytettiin kaiuttimia sekä mikrofonia. Kaiuttimet kytkettiin 3,5 mm AUX-päätteeseen. Mikrofoni oli Micro-USB-portissa (KUVA 6). Raspberry Pissä AUX-liitäntää ei voi käyttää sisääntulona, joten mikrofoni pitää liittää Micro-USB porttiin.

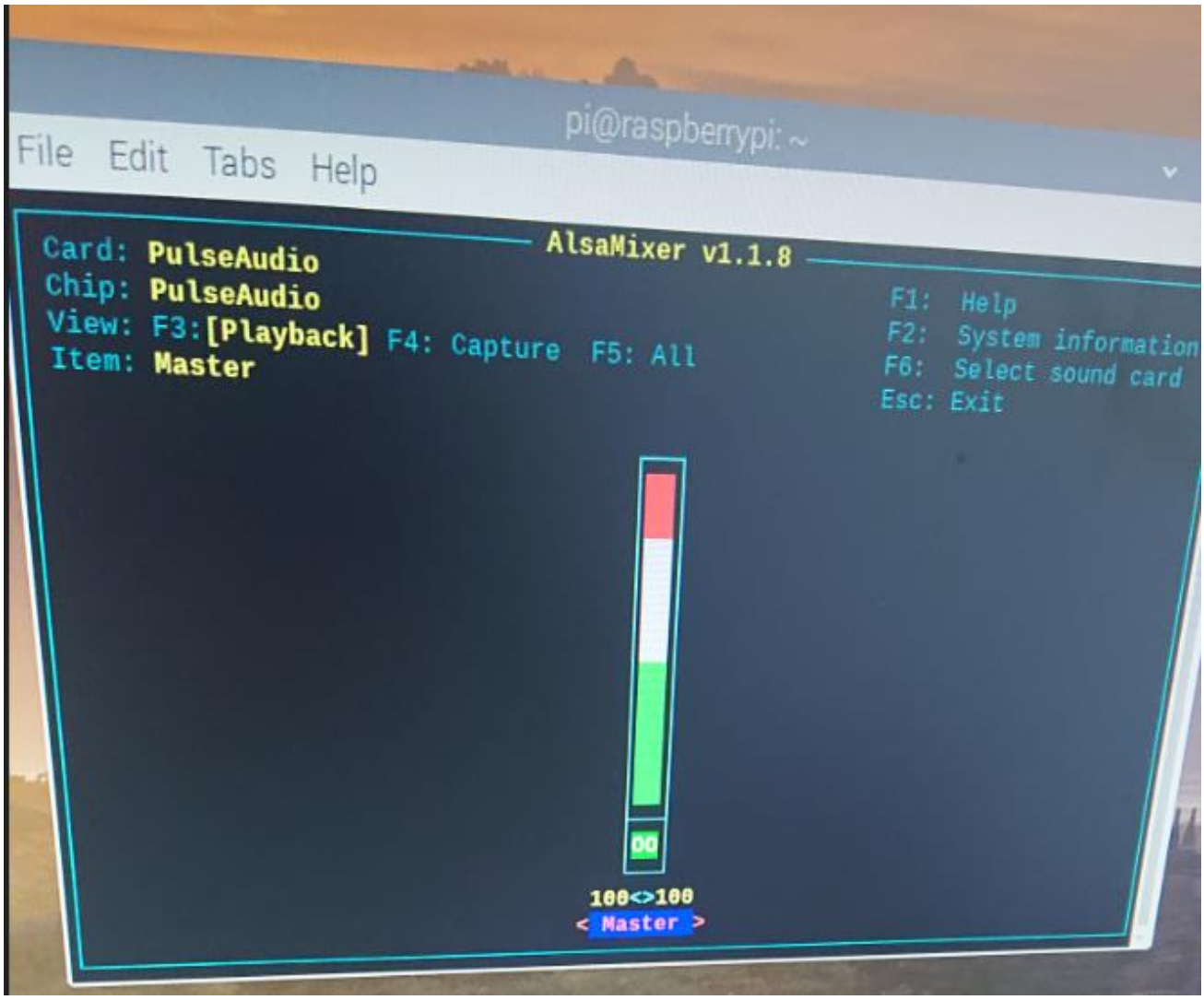


KUVA 6. Tähän saadaan liitettyä AUX-kaapelin kautta toimivat kaiuttimet, koko 3,5 mm. Mikro-USB portissa oli näppäimistö ja mikrofoni.

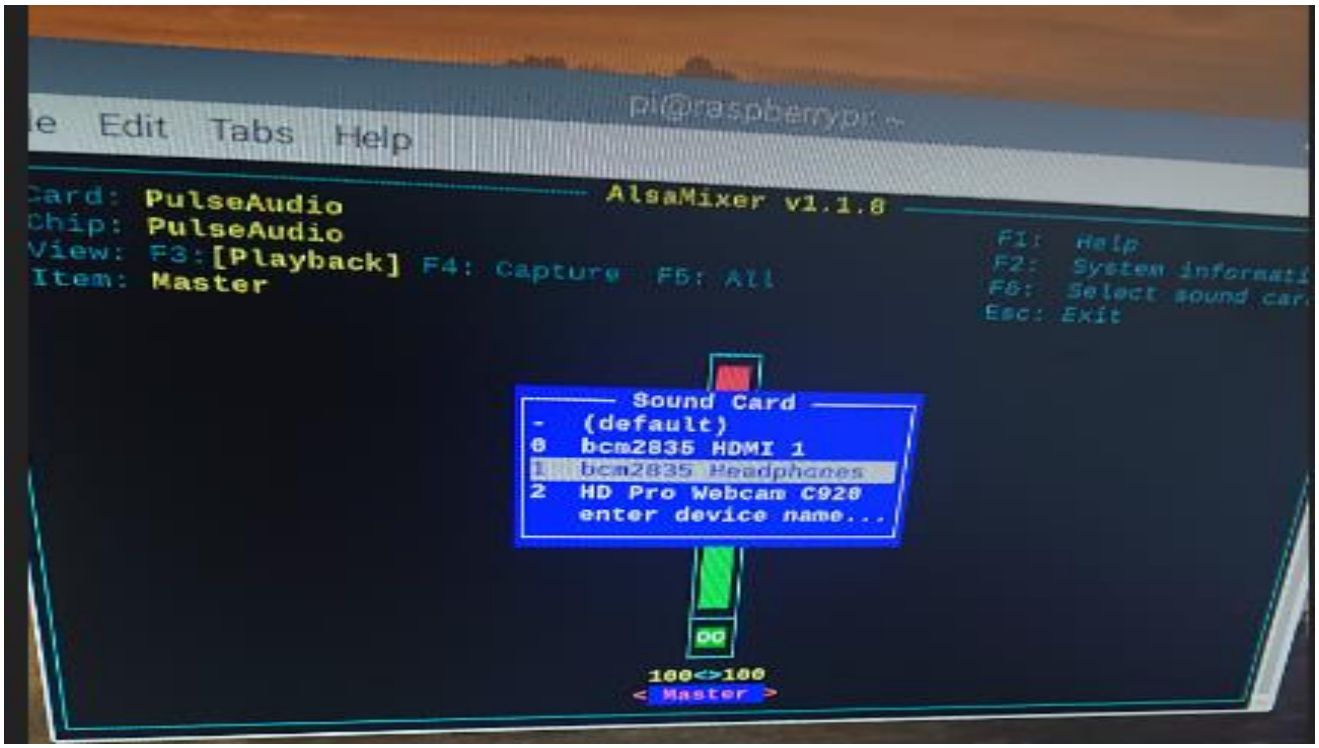
Äänen asetuksia voidaan säätää seuraavien kuvien 7–9 osoittamalla tavalla Raspberry Pissä. Näitä asetuksia ei tarvitse tehdä, mikäli äänentoisto on halutulla tasolla.



KUVA 7. Kirjoitetaan terminaaliin ”alsamixer” ja painetaan enteriä.



KUVA 8. Painetaan F6-näppäintä näppäimistössä.



KUVA 9. Liikutaan nuolinäppäimillä kohtaan ” 1 bcm2835 Headphones” (äänentoistolaitteen nimi voi olla eri riippuen käytetystä laitteesta) ja painetaan enteriä.

Tämän jälkeen lisätään äänenvoimakkuutta, kunnes haluttu äänenvoimakkuuden taso on löydetty ja poistutaan esc-näppäimellä. Äänentaso riippuu käytetystä laitteesta.

6.4 Led-lamput

Järjestelmässä oli kaksi led-lamppua punainen sekä vihreä. Niiden tarkoituksen oli osoittaa etäasiakas-palvelujärjestelmän tilaa, oliko se käytössä vai ei. Punainen led-lamppu indikoi tilan olevan pois päältä, kun taas vihreä led-lamppu indikoi tilan olevan päällä. Nämä lamput haluttiin järjestelmään, jotta olisi helppo tietää onko järjestelmä päällä vai ei.

7 OHJELMAN TOIMINTA

Tässä luvussa kerrotaan ohjelman, ja täten järjestelmän, toiminnasta yleisesti. Kun hardware- ja software-asennukset on tehty kaikkien laitteiden osalta, voidaan järjestelmän toiminta aloittaa. Järjestelmän käynnistäminen tapahtuu laittamalla kaikki laitteet päälle, ja käynnistämällä Python-etäasiakaspalveluohjelma Raspberry Pissä. Etäasiakaspalvelun käyttämän ohjelman Python-koodi kokonaisuudessaan on liitteessä 3.

7.1 Aloitus

Järjestelmän aloitus tapahtuu, kun etäasiakaspalvelu-ohjelma käynnistetään. Tämän jälkeen Python-ohjelma hallinnoi Raspberry Pitä. Kun infrapuna-anturi havaitsee liikettä lähettää se signaalin Raspberry Pille, joka käynnistää Chrome-selaimessa Jitsi Meet-Chat-huoneen Chromedrive-moduulia käyttäen, sekä lähettää automaattisen sähköpostinviestin asiakaspalvelijalle (smtplib-moduulin avulla). Mikrofonin sekä kaiuttimien käyttöönotto tapahtuu automaattisesti ohjelman avulla; tämä tapahtuu myös Chromedriver-moduulin avulla.

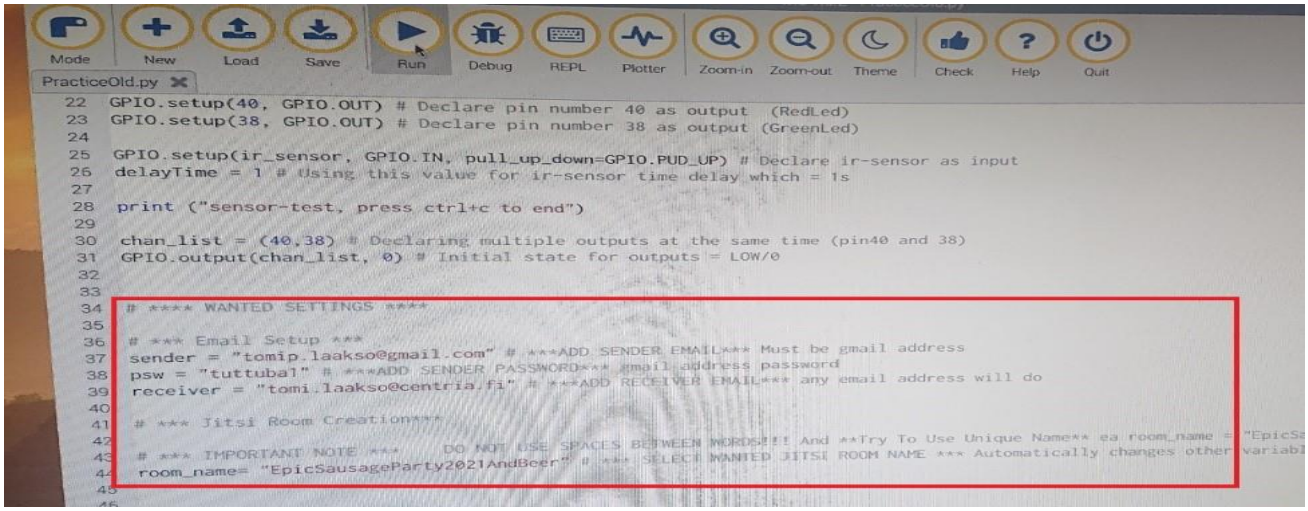
7.2 Lopetus

Kun asiakaspalvelu on saatu päätökseen, sulkee asiakaspalvelija Jitsi Meet-Chat-huoneen; tällöin sulkeutuu myös Rasperry Pin Chrome-selain, ja täten myös yhteys Jitsi Meet-Chat-huoneen. Tämä vaihe vaatii sen, että asiakaspalvelija sulkee manuaalisesti yhteyden kyseiseen Jitsi Meet-Chat-huoneeseen; muutoin Raspberry Pissä jää kaikki auki, mikä aiheuttaa ongelmia (kuten kameran ylikuumenemisen jossain kohtaa). Jotta havaittaisiin, milloin asiakaspalvelija sulkee yhteyden Jitsi Meet- Chat-huoneeseen, käytetään aktiivista tiedonharavointia Raspberry Pissä Chat-huoneen olleessa auki. Kun tämä vaihe on tehty, ollaan valmiita taas uuden asiakkaan palvelemiseen.

8 AUTOMAATTISEN SÄHKÖPOSTIN SELOSTUS JA ASETUKSET

Järjestelmän ohjelmassa on käytössä automaattisen sähköpostin lähetys, kun asiakas saapuu järjestelmän eteen. Sähköposti lähetään halutulta sähköpostiosoitteesta haluttuun sähköpostiosoitteeseen, halutulla viestillä; näitä kaikkia pystyy muokkaamaan ohjelman koodista (KUVA 10). Lisäksi sähköpostissa on linkki Chat-huoneeseen. Tämä vaatii sen, että lähettäjän sähköpostiosoite ja sen salasana tiedetään.

Tämä ominaisuus haluttiin sen takia, että etäasiakaspalvelija saisi tiedon, että asiakas haluaa palvelua tietyssä Chat-huoneessa. Tämän ominaisuuden ansioista asiakaspalvelijan ei tarvitse aktiivisesti valvoa Chat-huonetta, eikä näin ollen tarvitse pitää kameraansa tai yhteyttä Chat-huoneeseen koko ajan auki.



KUVA 10. ”Wanted Settings” kohtaan laitetaan lähettäjän sähköposti (gmail) ja kohtaan ”psw” tulee sähköpostin salasana, vastaanottajaksi merkitään vastaaanottajan sähköposti.

Chat-huoneen nimeä vaihtamalla saadaan uusi huone aikaiseksi ja kaikki vaadittavat parametrit muuttuvat automaattisesti vastaamaan uutta huoneen nimeä (KUVA 11). Kohdassa ”message” voi valita millaisen sähköpostin haluaa lähettää, kun asiakas on odottamassa (KUVA 12).

```
room_name= "EpicSausageParty2021AndBeer" # *** SELECT WANTED JITSI ROOM NAME *** Automatically changes other variables to match selected room name
```

KUVA 11. Koodin kohta, josta voidaan vaihtaa Chat-huoneen nimeä vaihtamalla kohtaan ’EpicSausageParty2021AndBeer’ jotain muuta.

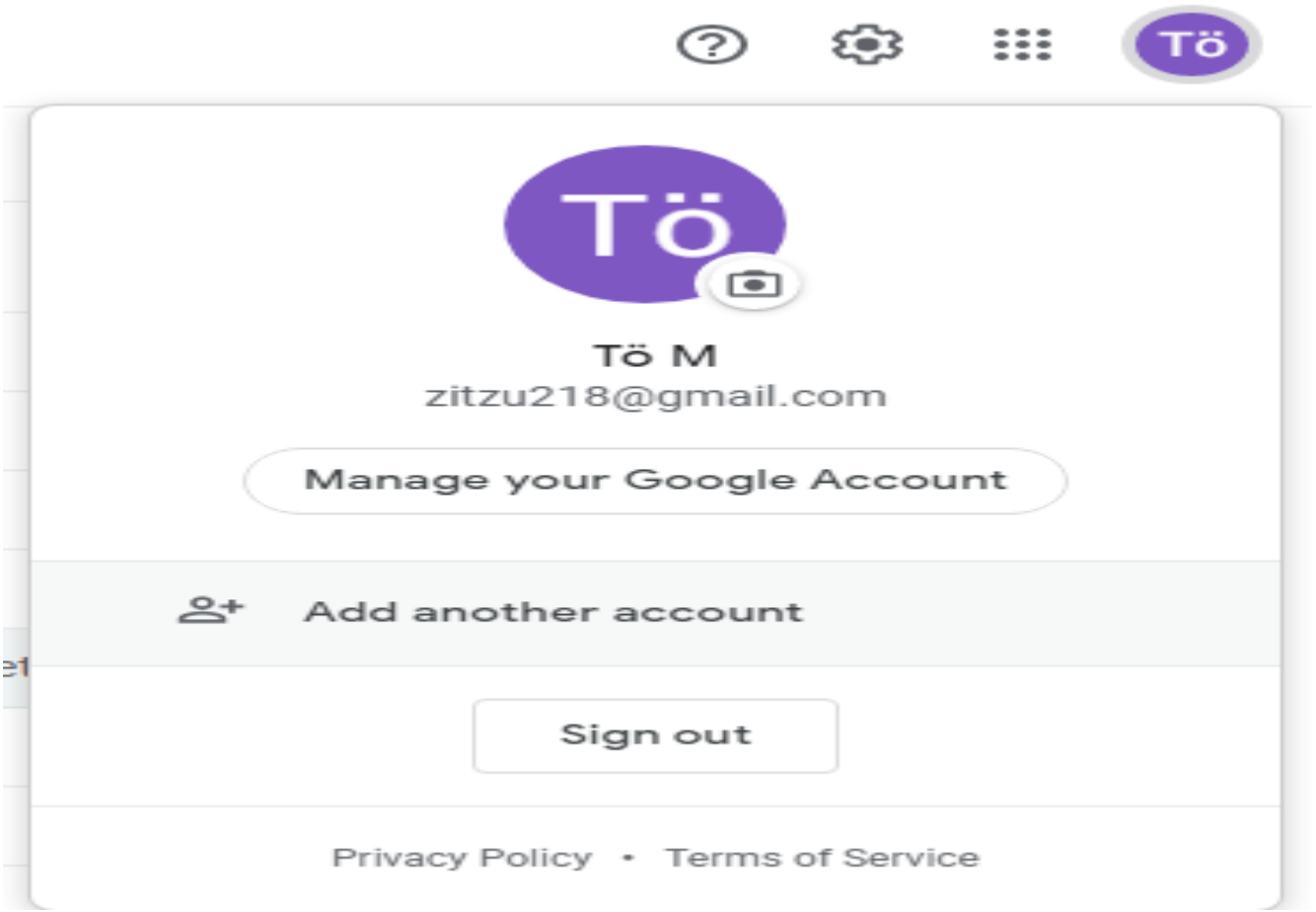
```
message = header + f'\n There is a customer waiting at https://meet.jit.si/{room_name} \n\n'
```

KUVA 12. Automaattisen sähköpostin viestin sisällön voi muuttaa vaihtamalla kohtaan ’\n There is a customer waiting at https://meet.jit.si/{room_name} \n\n’ haluttu teksti.

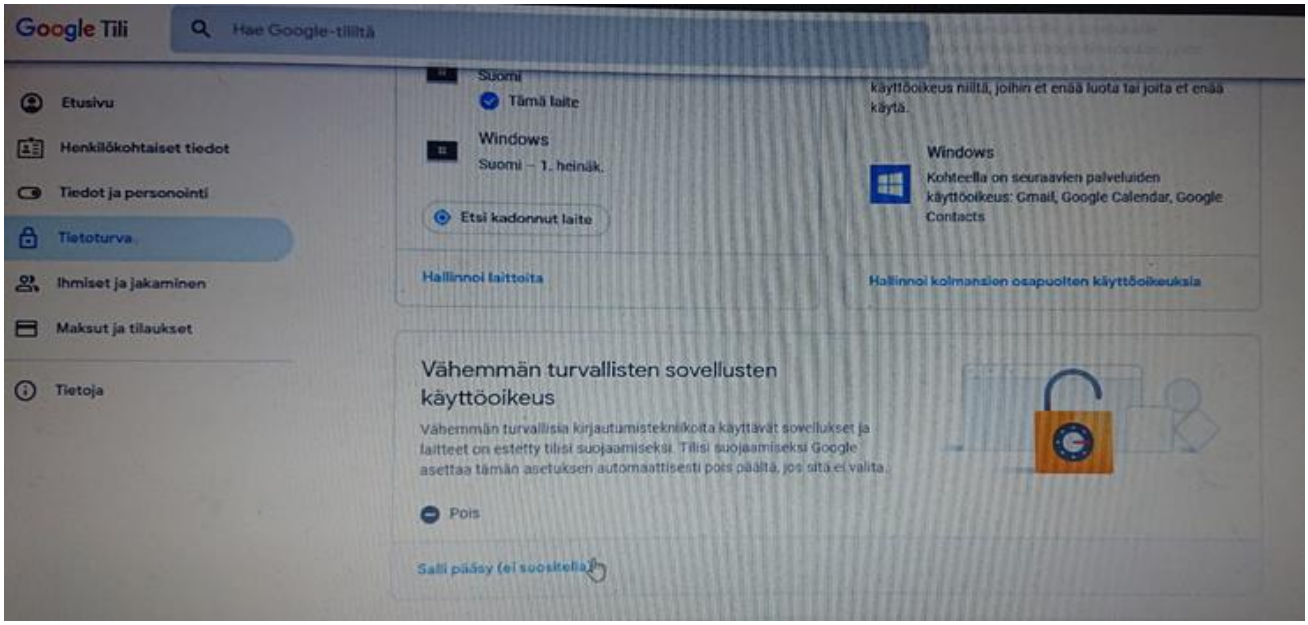
8.1 Gmail

Seuraavaksi katsotaan mitä asetuksia pitää laittaa itse sähköpostiasetuksissa, jotta viestit saadaan lähetettyä ja vastaanotettua Gmail-tilillä. Tämä vaihe täytyy tehdä, koska tietoturvallisista syistä yhtiöt eivät halua, että kaikki viestit pääsevät automaattisesti läpi.

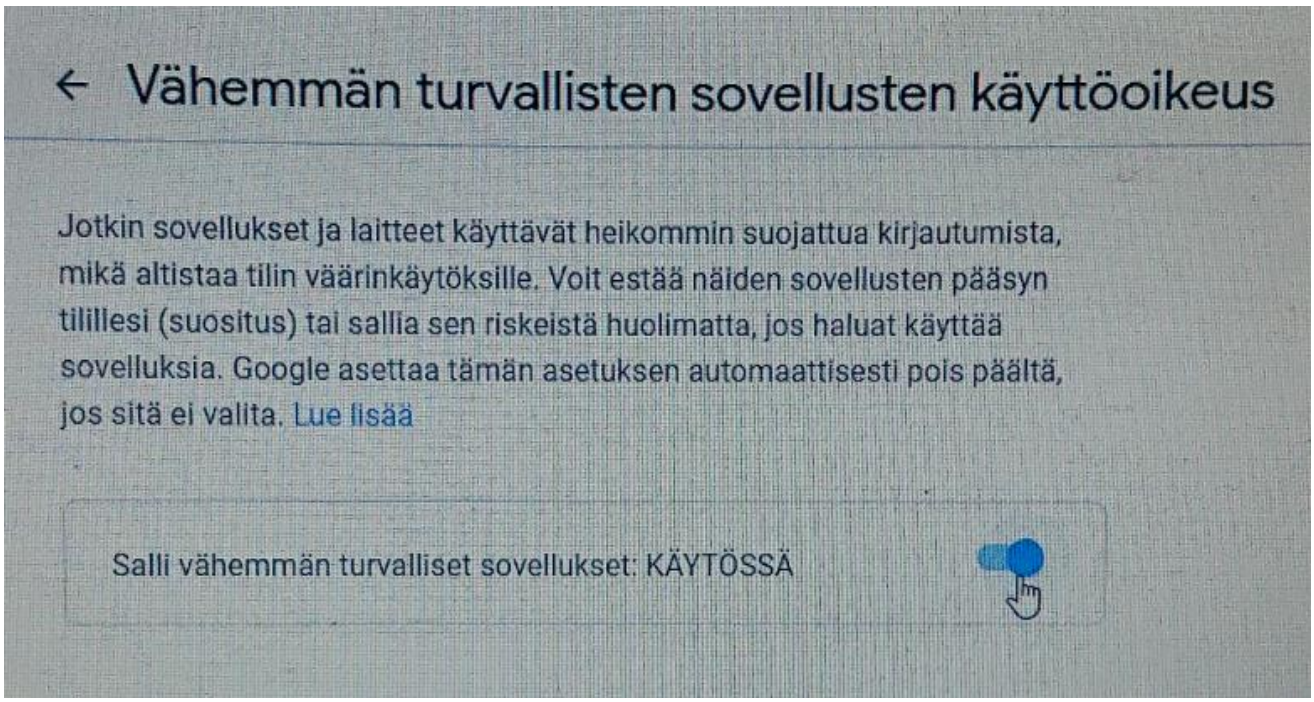
Täysin uusi Gmail-tili voidaan luoda sähköpostiviestien lähettämistä varten. Kirjaudutaan Gmail-tilille, mennään ”Tilin määritykset”-valikkoon (KUVA 13) ja etsitään kohdasta tietoturva ”Vähemmän turvallisten sovellusten käyttöoikeus” (KUVA 14) ja klikataan ”salli vähemmän turvalliset sovellutukset: käytössä” hiirellä (KUVA 15). Valikkojen nimet riippuvat käytetystä kielestä.



KUVA 13. Klikataan hiirellä oikeassa ylänurkassa olevaa kaksikirjaimista painiketta, jonka jälkeen klikataan ”Manage your Google Account”. Täten päästään kyseisen Gmail-tilin asetusvalikkoon.



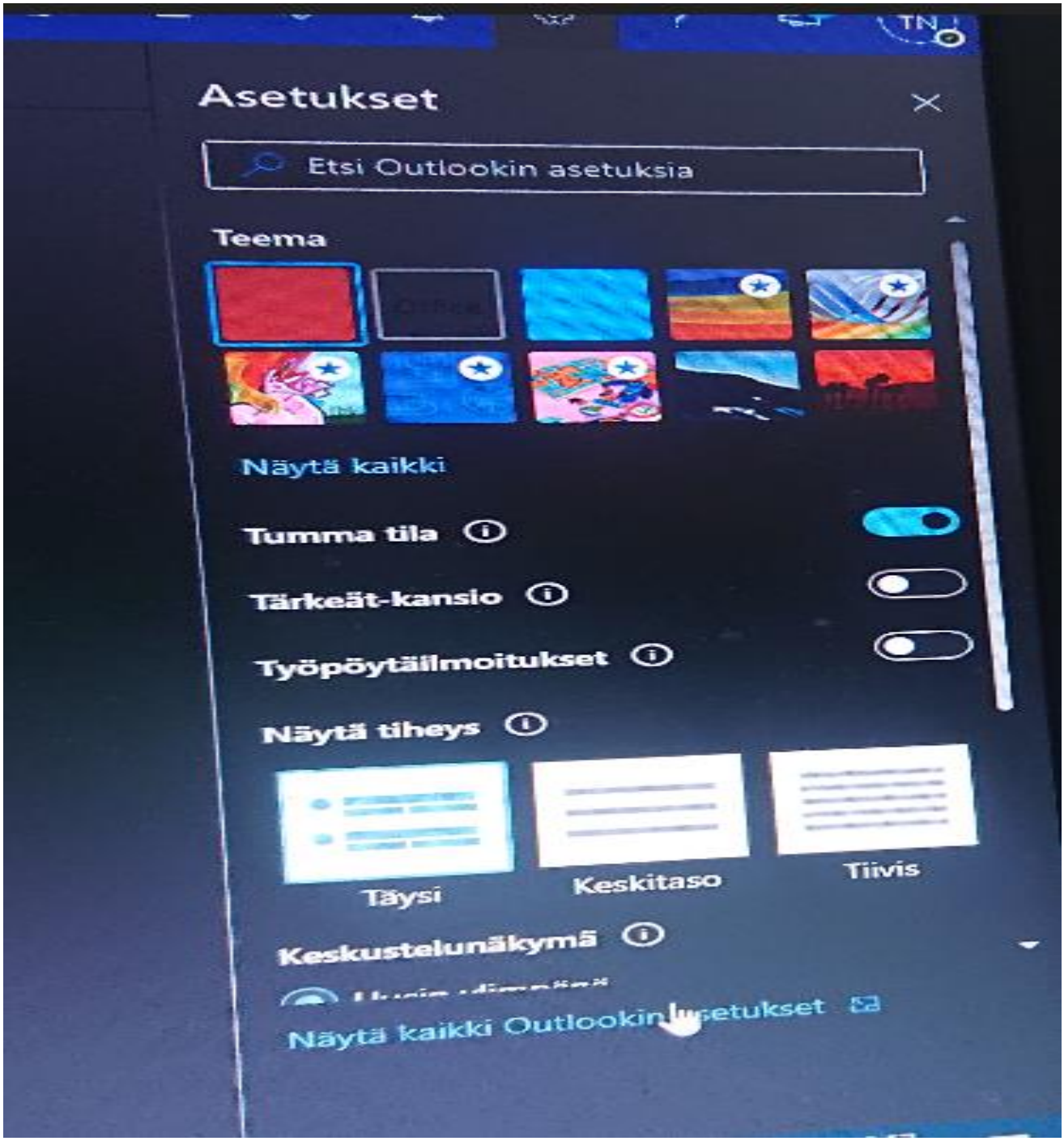
KUVA 14. Gmail-tilin asetusvalikko, josta valittuna ”Tietoturva”.



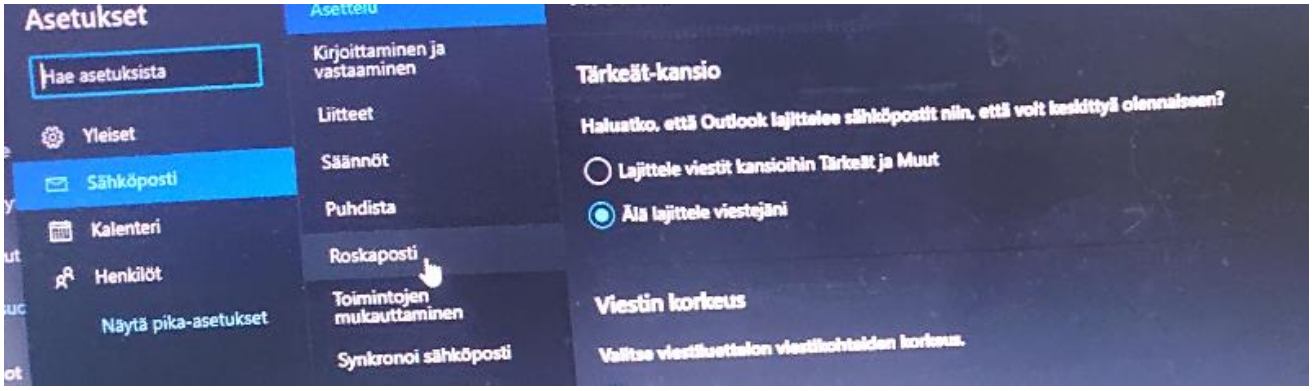
KUVA 15. Klikataan hiirellä salli vähemmän turvallisten sovellukset käyttöoikeudet päälle. Muuten automaattinen sähköposti ei toimi, koska Google estää automaattisten viestien lähetyksen ja vastaanoton.

8.2 O365

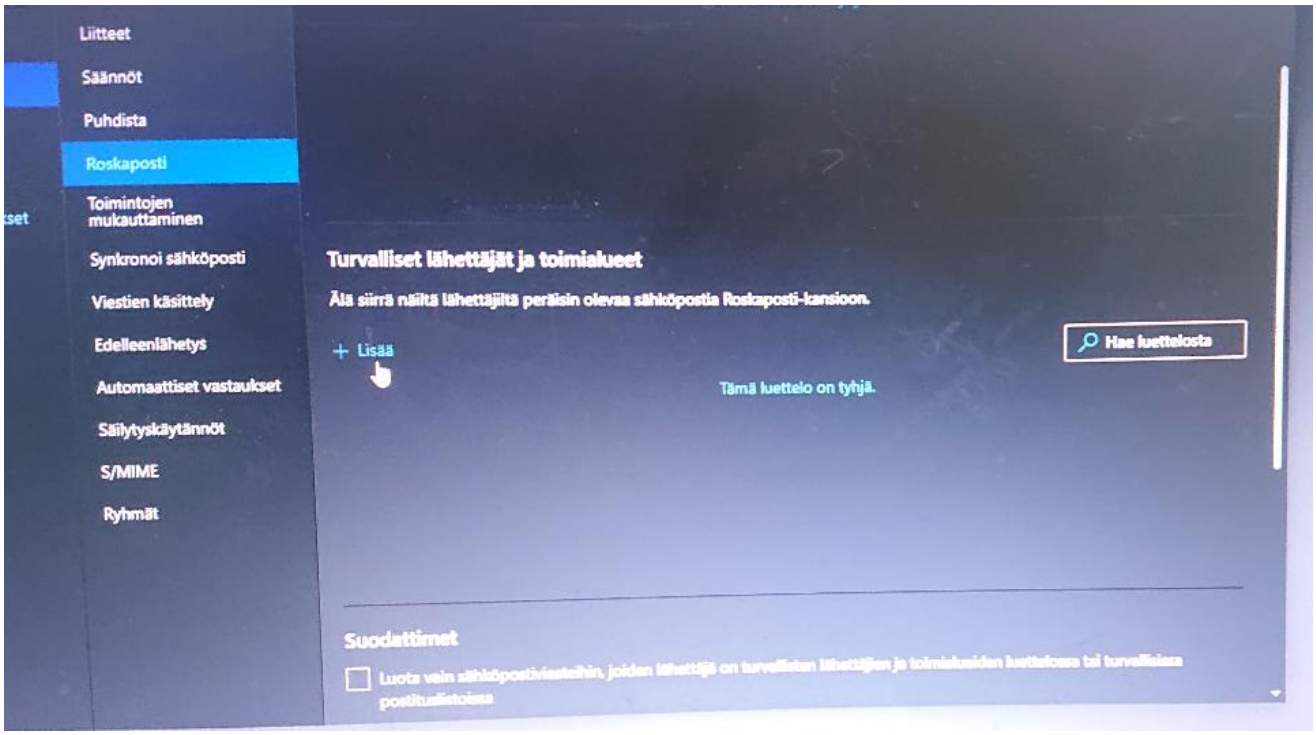
O365-sähköpostilla, eli Microsoftin Outlook (Hotmail)-sähköpostilla automaattisen sähköpostin lähetyksen sekä vastaanoton asetukset tehdään alle olevien ohjeiden mukaisesti. Ensin kirjaudutaan selaimella Outlook-sähköpostitilille. Kuvat 16-19 osoittavat mitä sitten tulee tehdä.



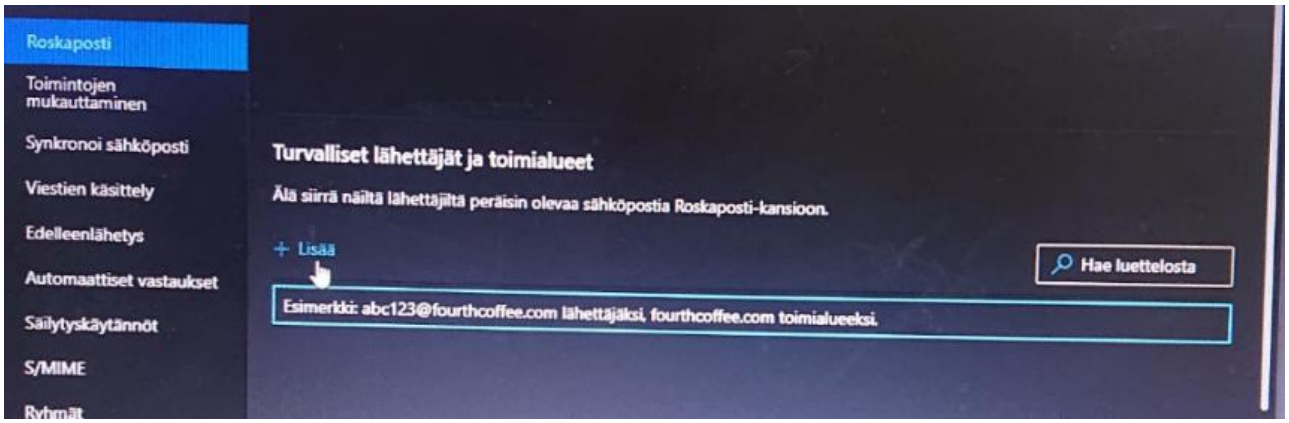
KUVA 16. Valitaan asetukset oikeasta yläkulmasta, jonka jälkeen valitaan pudotusvalikosta ”näytä kaikki Outlookin asetukset” ja klikataan sitä hiirellä.



KUVA 17. Valitaan kohdasta sähköposti ”roskaposti” ja klikataan hiirellä.



KUVA 18. Lisätään automaattisen sähköpostin lähettäjän osoite painamalla ”lisää” painiketta.



KUVA 19. Lähettäjän sähköpostiosoitteen lisäämisen jälkeen tallennetaan muutokset.

9 JOHTOPÄÄTÖKSET

Projektin alkuperäiseen kysymykseen “Voiko asiakaspalvelua antaa etänä?” pystyttiin vastaamaan “Kyllä”. Projekti onnistui siis varsin hyvin. Tämän jälkeen pohdittiin varsinaisia tapoja antaa asiakaspalvelua etänä. Projekti oli hyvin avonainen tapojen suhteen, sillä mitään tarkkoja parametrejä ei ollut; oleellisinta oli saada tehtyä prototyyppi, jolla todistetaan, että etäasiakaspalvelujärjestelmä on mahdollinen. Projektin edessä ilmeni mahdollisia ongelmia, jotka onnistuttiin kuitenkin ratkomaan. Yhteys saatiin lopetettua täysin automaattisesti asiakkaan puolesta, asiakaspalvelijan pitää lopettaa yhteys.

Projektin aihealue oli erittäin ajankohtainen sen tekohetkellä, kesällä 2021, kun COVID-19 aiheutti ongelmia maailmassa; monia eri palveluita pyritään toimittamaan etänä. Toisaalta myös muutenkin palveluita on jo aiemmin pyritty antamaan etänä.

Projektissa yhdisteltiin automaatiota sekä ohjelmointia, mikä teki projektista haastavan mutta mielenkiintoisen. Haastavuutta lisäsi se, että projektissa ei ollut mukana ammattilaisia, joilta olisi voinut kysyä käytännöntason kysymyksiä, koska projektia ei tehty yrityksen työntekijänä. Toisaalta ammattilaisten puuttuminen lisäsi innovaatiota sekä antoi vapaammat kädet tehdä omalla tyylillä.

Lähteet:

8x8, Inc. 2022. Internet-sivu.
Saatavissa: https://meet.jit.si/. Viitattu 1.3.2022

Gaggia, Michele. 2022. Recording studio design. Www-dokumentti.
Saatavissa: https://www.digitalnaturalsound.com/images/stories/fh_mma_courses/pdf/mg_studio_design.pdf. Viitattu 3.2.2022

Jetbrains. 2022. Internet-sivu.
Saatavissa: https://www.jetbrains.com/pycharm/features/. Viitattu 26.1.2022

Jf-parede. 2022. Mikä on IR-anturi: piirikaavio ja sen toiminta. Internet-sivu.
Saatavissa: https://fi.jf-parede.pt/what-is-an-ir-sensor. Viitattu 15.4.2022

Python Software Foundation. 2021a. Internet-sivu.
Saatavissa: https://docs.python.org/3/tutorial/index.html. Viitattu 21.12.2021

Python Software Foundation. 2021b. Internet-sivu.
Saatavissa: https://docs.python.org/3/library/intro.html. Viitattu 21.12.2021

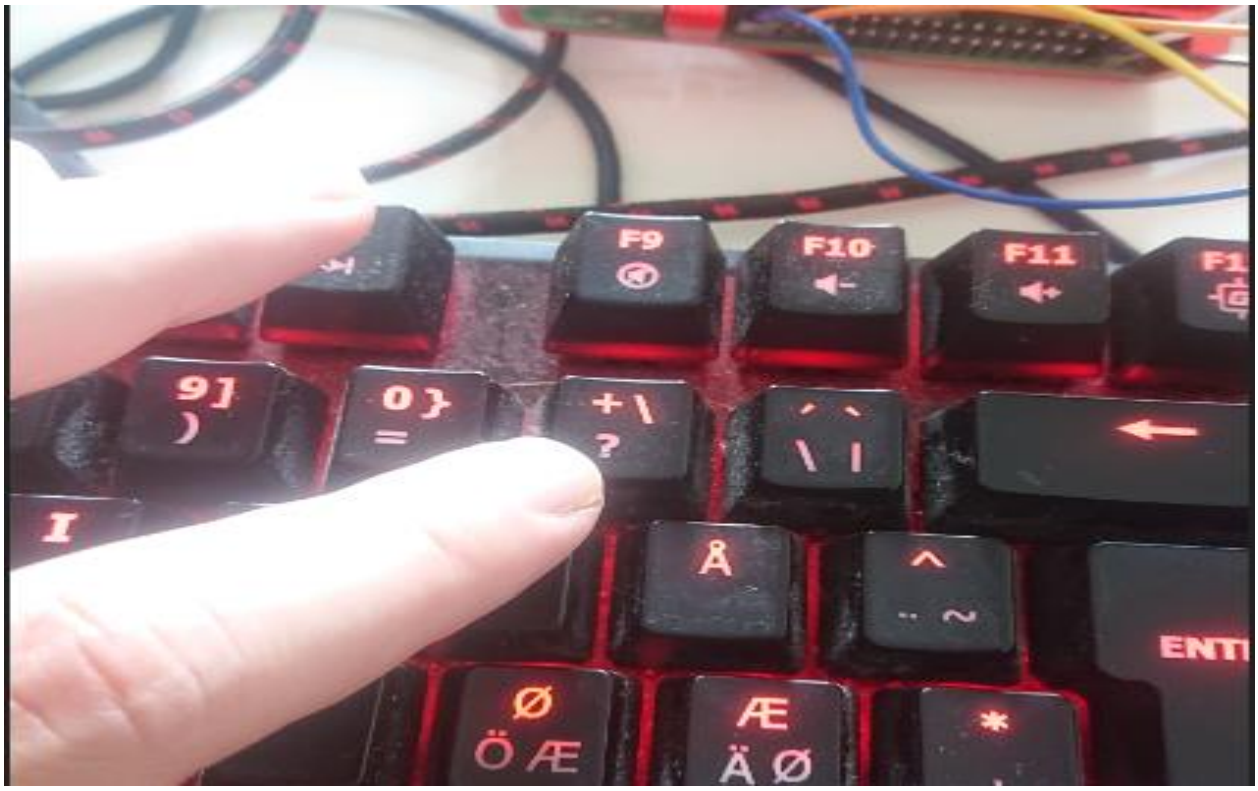
Python Software Foundation. 2022a. Internet-sivu.
Saatavissa: https://docs.python.org/3/library/webbrowser.html. Viitattu 3.2.2022

Python Software Foundation. 2022b. Internet-sivu.
Saatavissa: https://pypi.org/project/selenium/ . Viitattu 3.2.2022

Raspberry Pi Foundation 2021. Raspberry Pi 3 Model A+. Internet-sivu.
Saatavissa: https://www.raspberrypi.com/products/raspberry-pi-3-model-a-plus/. Viitattu 11.1.2022

Routamaa-Päiviö, Nina. 2021. Asiakaspalvelun ABC ravintola- ja cateringalalla. Saatavissa: https://peda.net/ahtari/lukio/luokat-oppiaineet/kmt1-kurssi/ap:file/download/85bd3767828663e6126f1638c64bbb8616848086/Asiakaspalvelun_perusteet_oppimateriaali.pdf.
Viitattu 15.12.2021

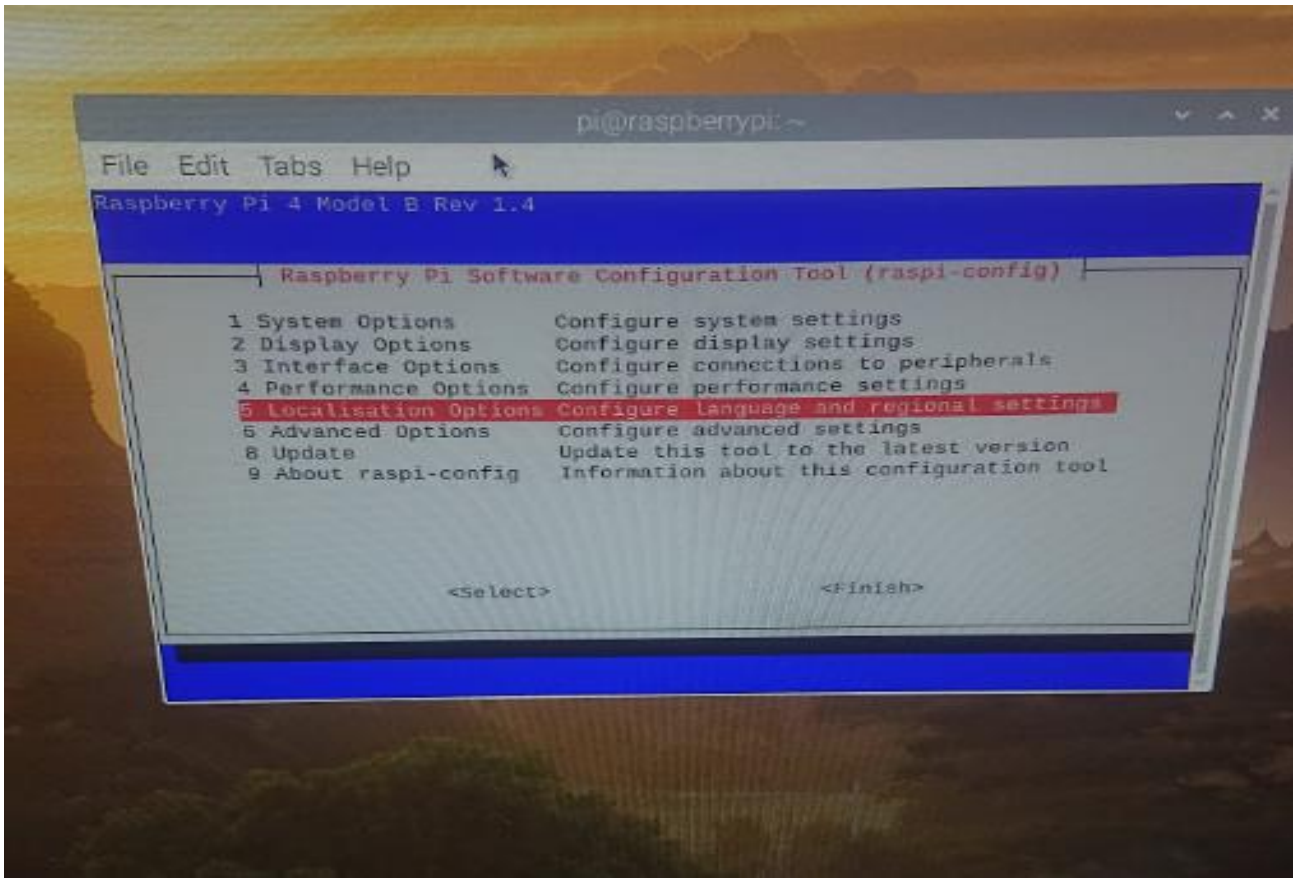
Käyttöjärjestelmän asennus Raspberry Pihin



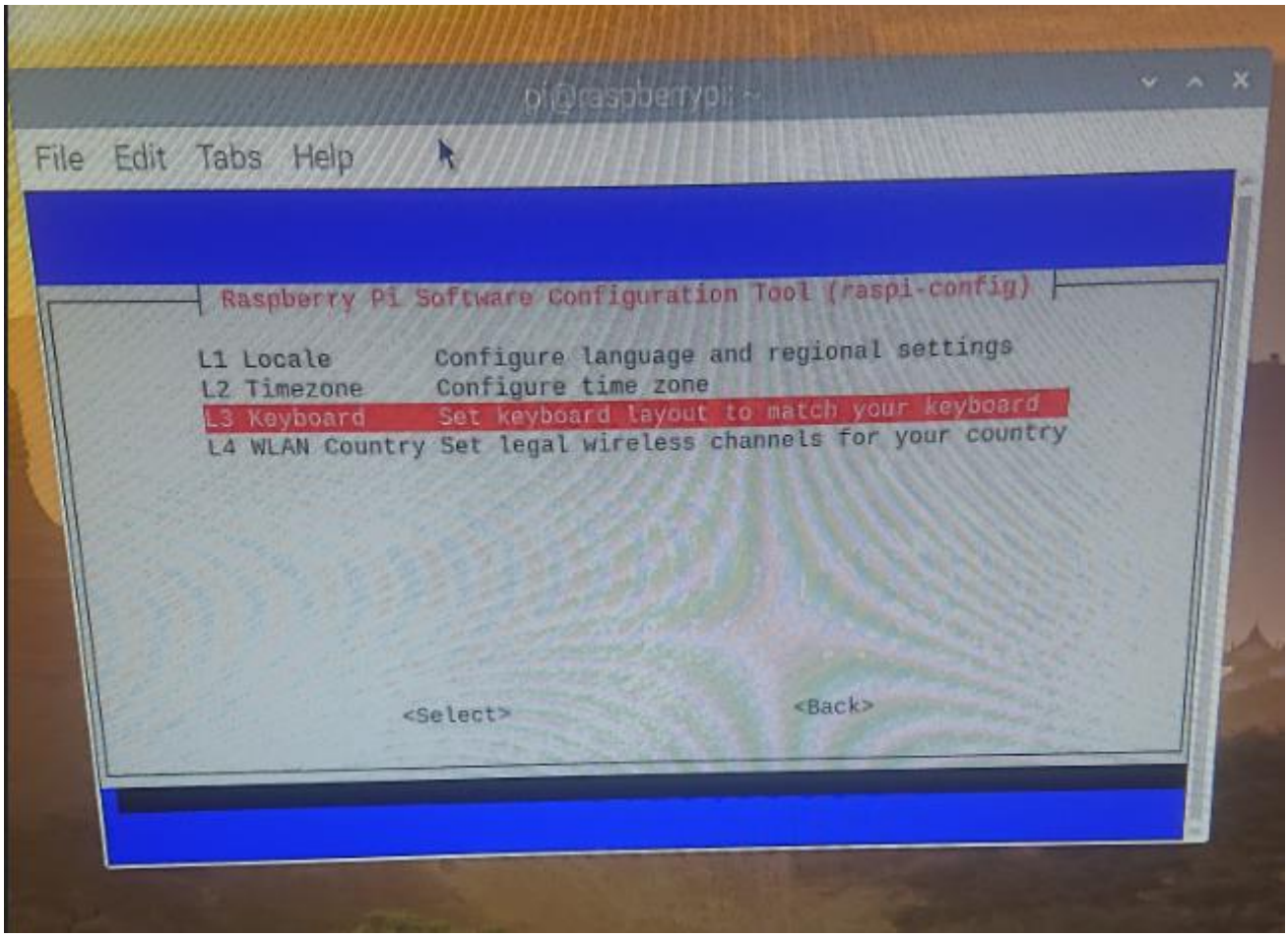
Suomalaisessa näppäimistössä +-merkki vastaan US-näppäimistössä väliviivaa.

Raspberry Pin terminaalissa asetusvalikon avaus.

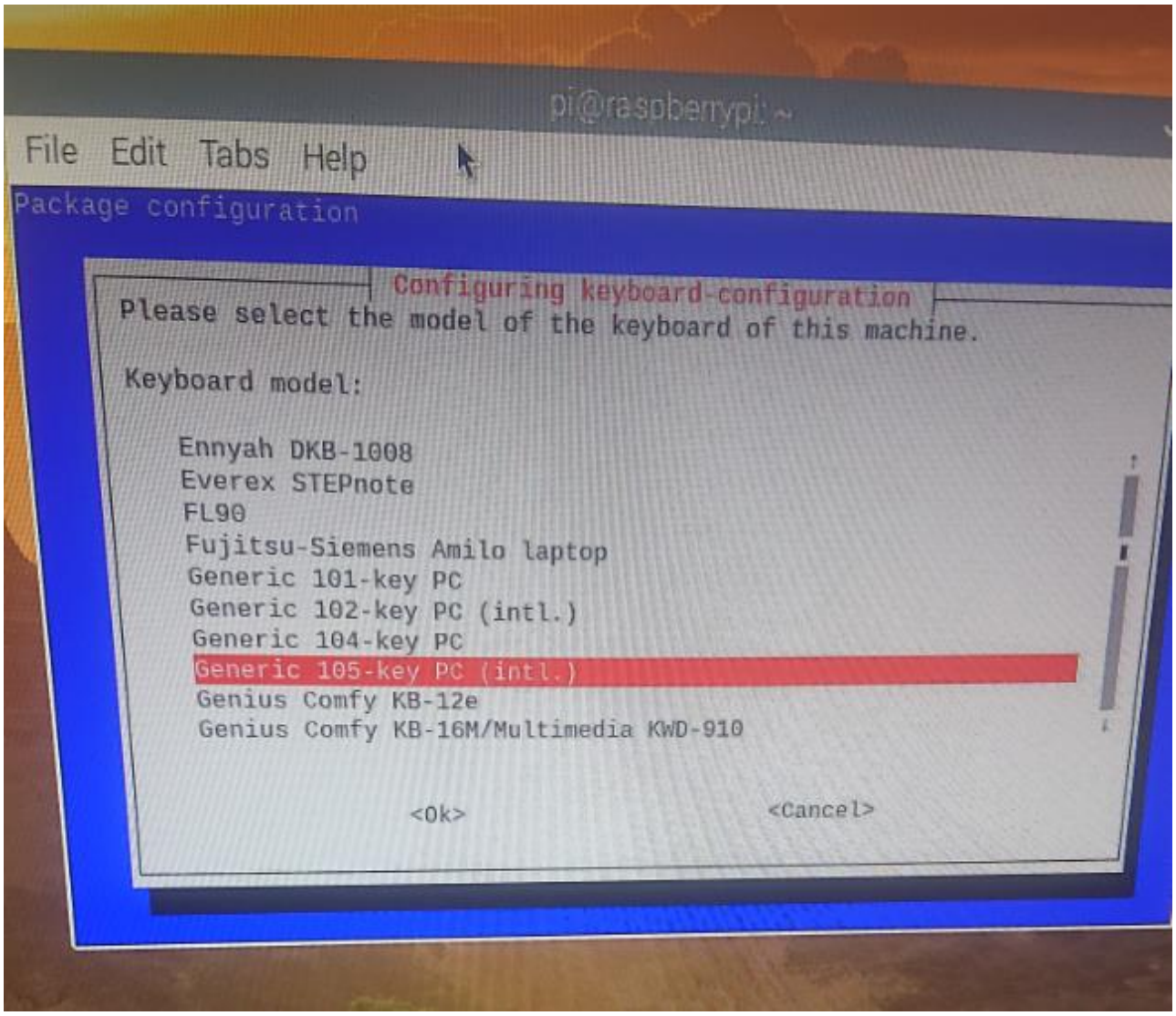
Seuraavaksi muutetaan näppäimistön asetukset skandinaaviseen muotoon ja alustetaan etäyhteys. Vaiheiden eteneminen seuraavissa kuvissa.



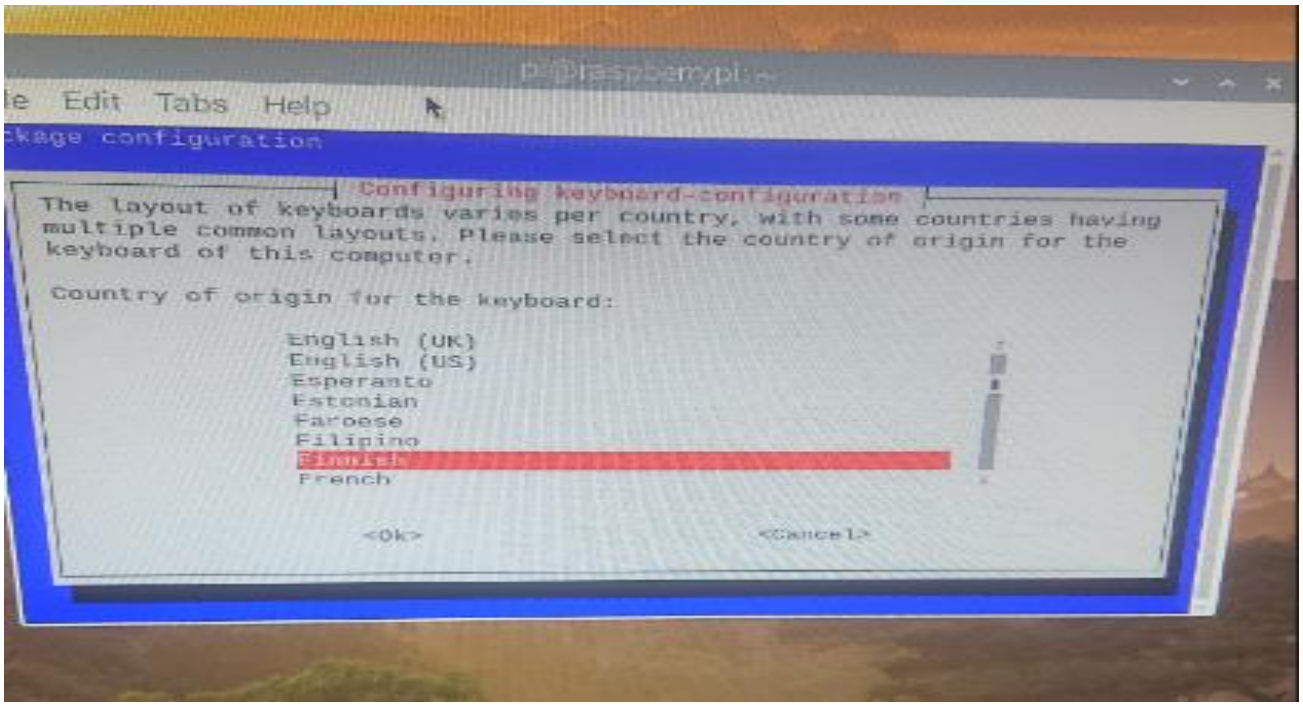
Raspberry Pin asetusvalikko. Valitaan ”Localisation Options” ja painetaan enteriä.



Valitaan vaihtoehto ”Keyboard” ja painetaan enteriä.



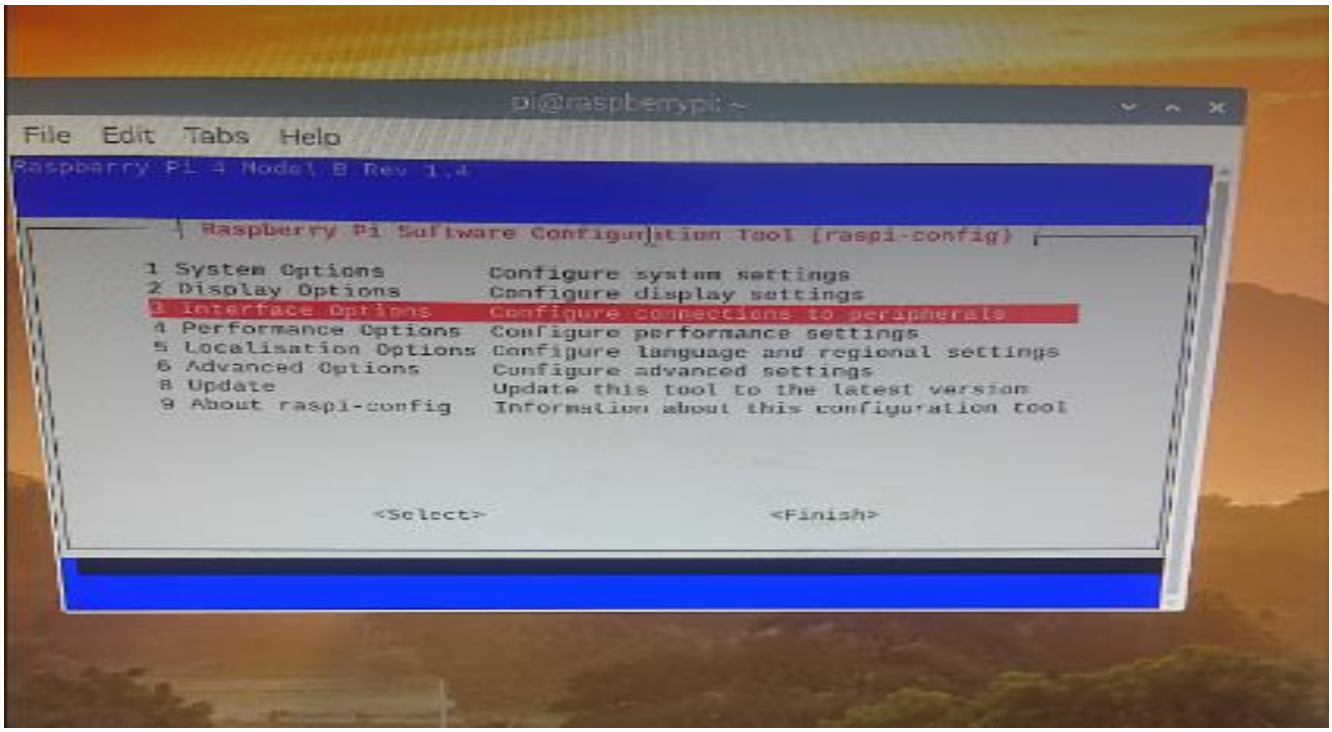
Valitaan ”Generic 105-key PC” ja painetaan enteriä.



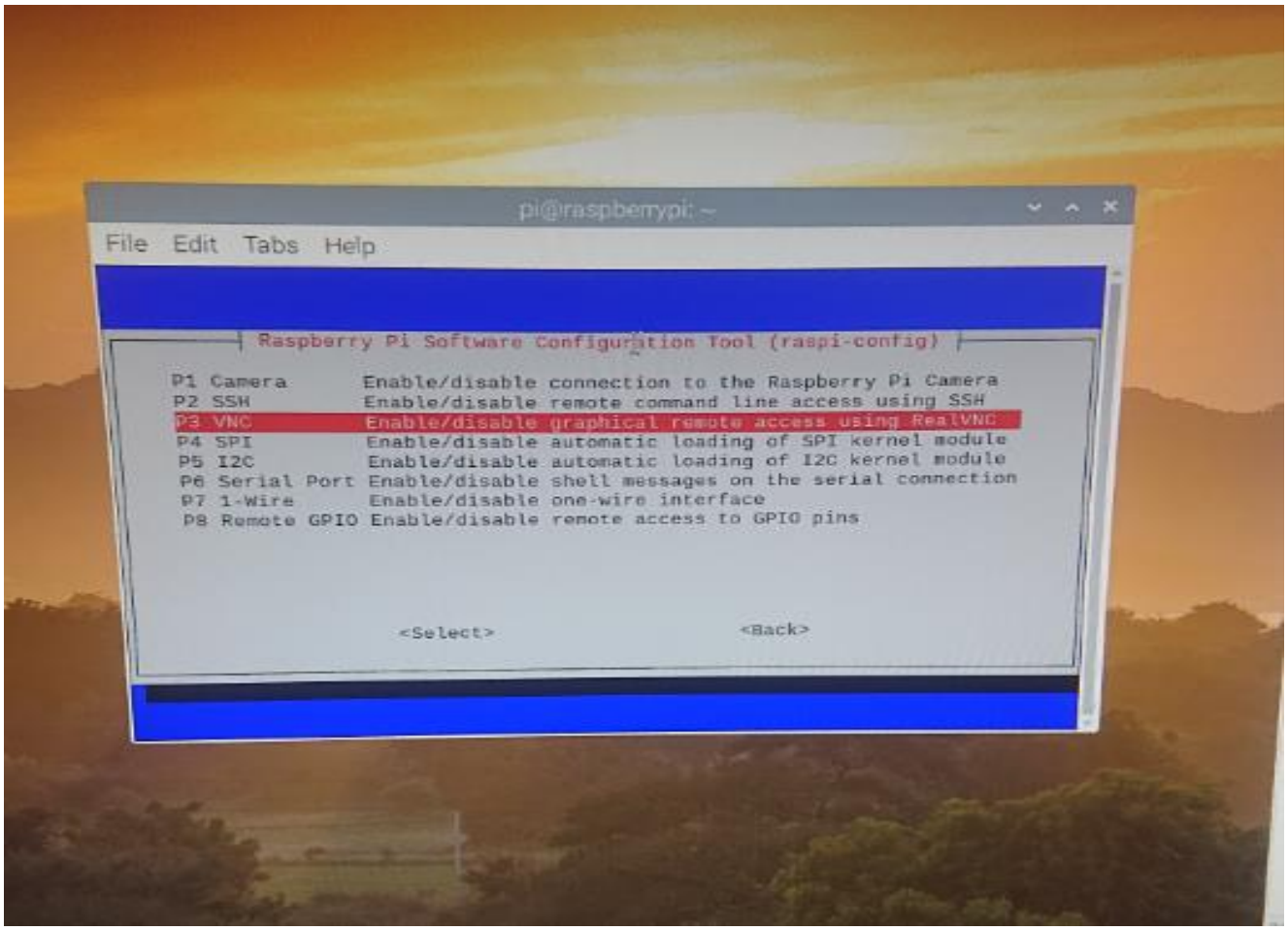
Valitaan ”Finnish” ja painetaan enteriä.

Tämän jälkeen voit painaa enteriä niin monta kertaa, kunnes päästään takaisin asetusvalikkoon (eli kuvan 5 näkymään).

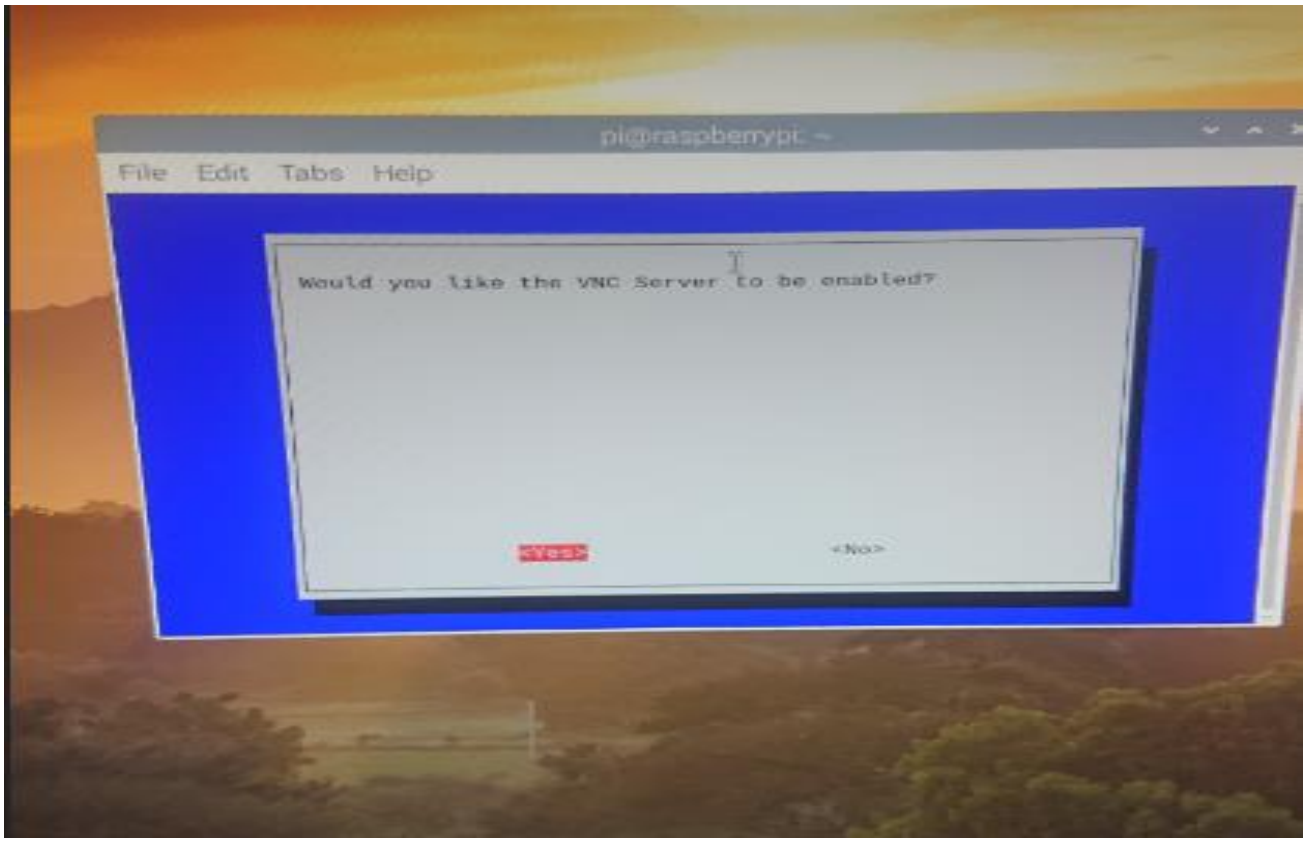
Seuraavaksi alustetaan VNC-serveri Raspberryn etäyhteyttä varten. Alustuksen vaiheet tulevat seuraavissa kuvissa ilmi.



Valitaan ”Interface Options” ja painetaan enteriä.

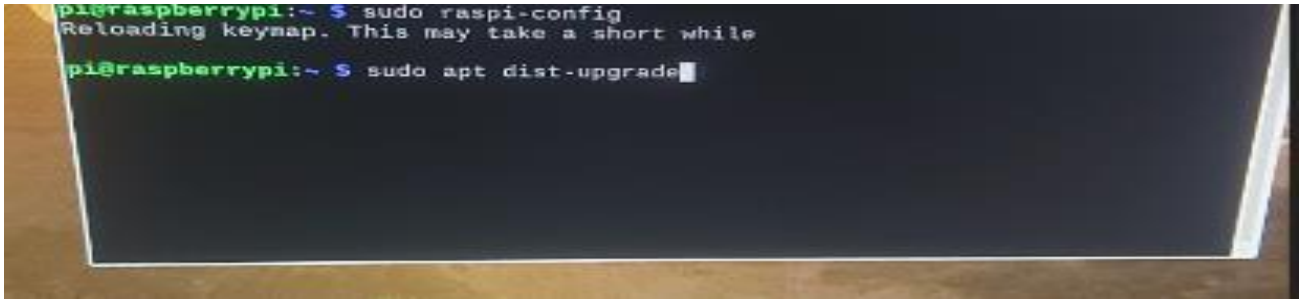


Valitaan ”VNC” ja painetaan enteriä.

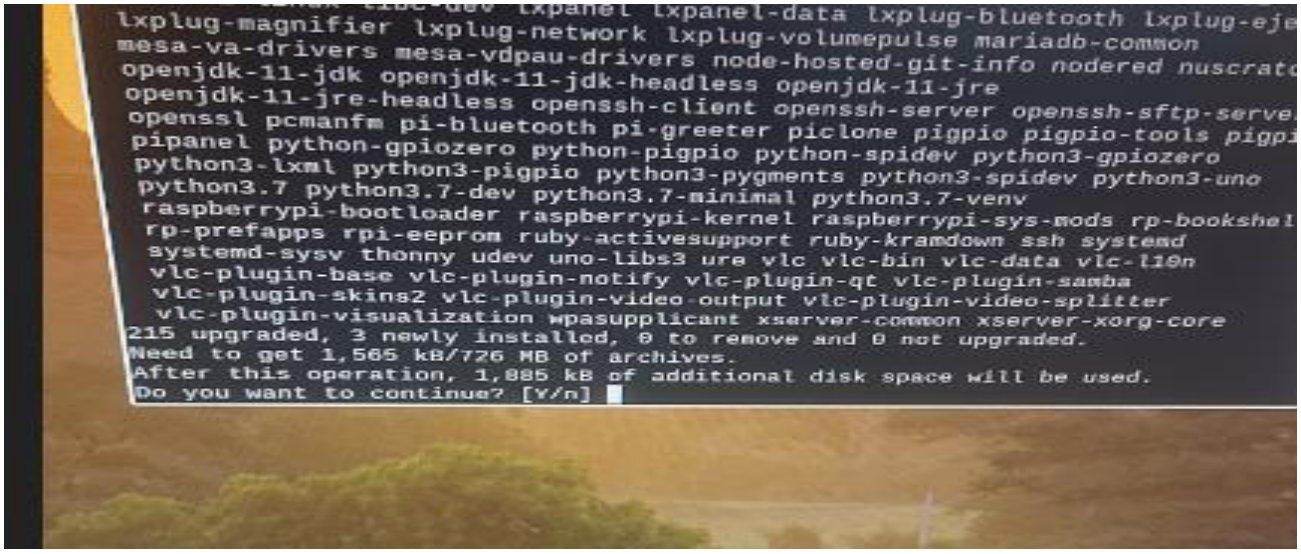


Valitaan” YES” ja painetaan enteriä.

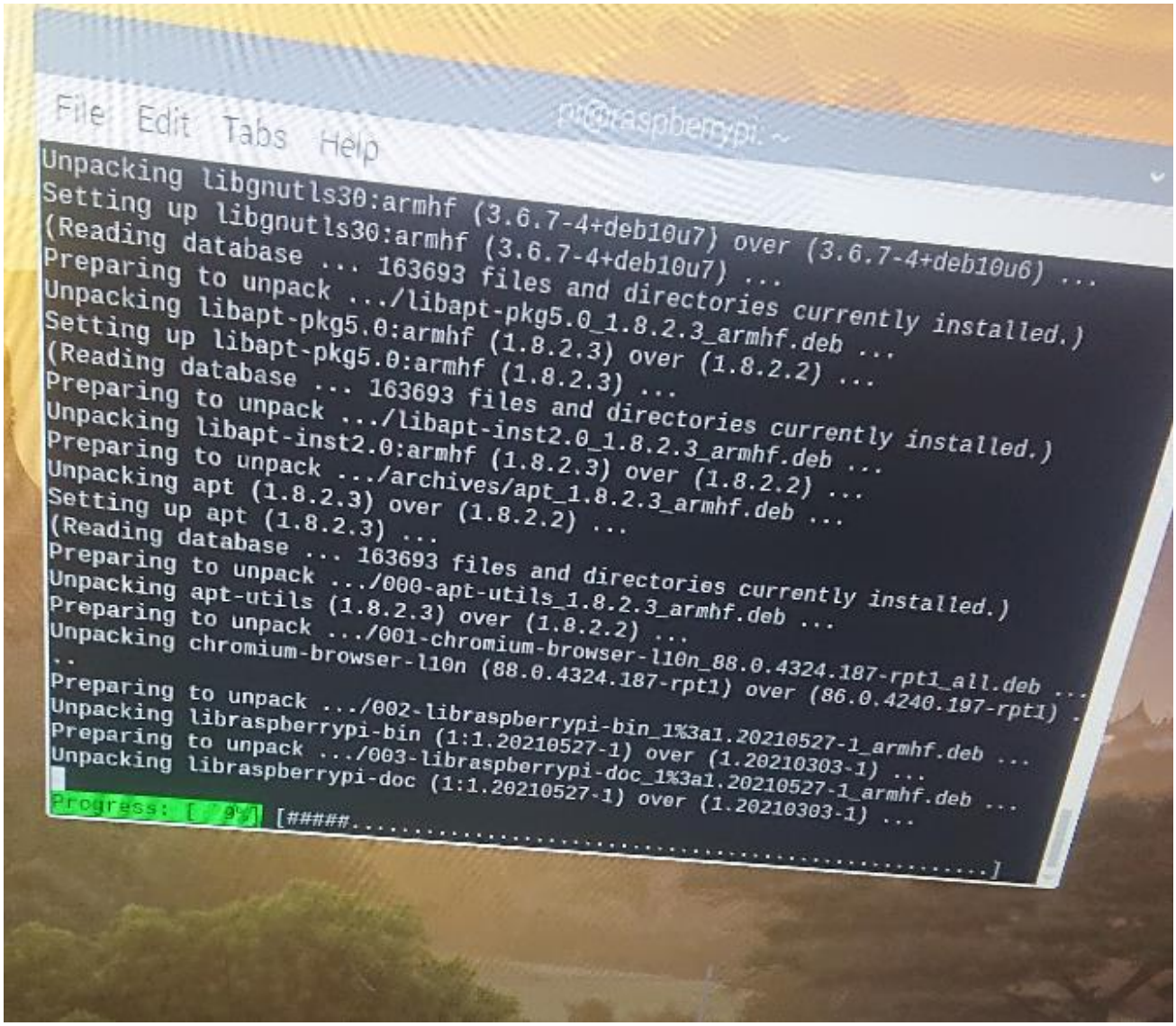
Päivitysten sekä lisäosien asennus



Kirjoitetaan ”sudo apt dist-upgrade” ja asennetaan kaikki päivitykset. Tämä vaihe vie noin 5–10 minuuttia riippuen internet yhteyden nopeudesta.

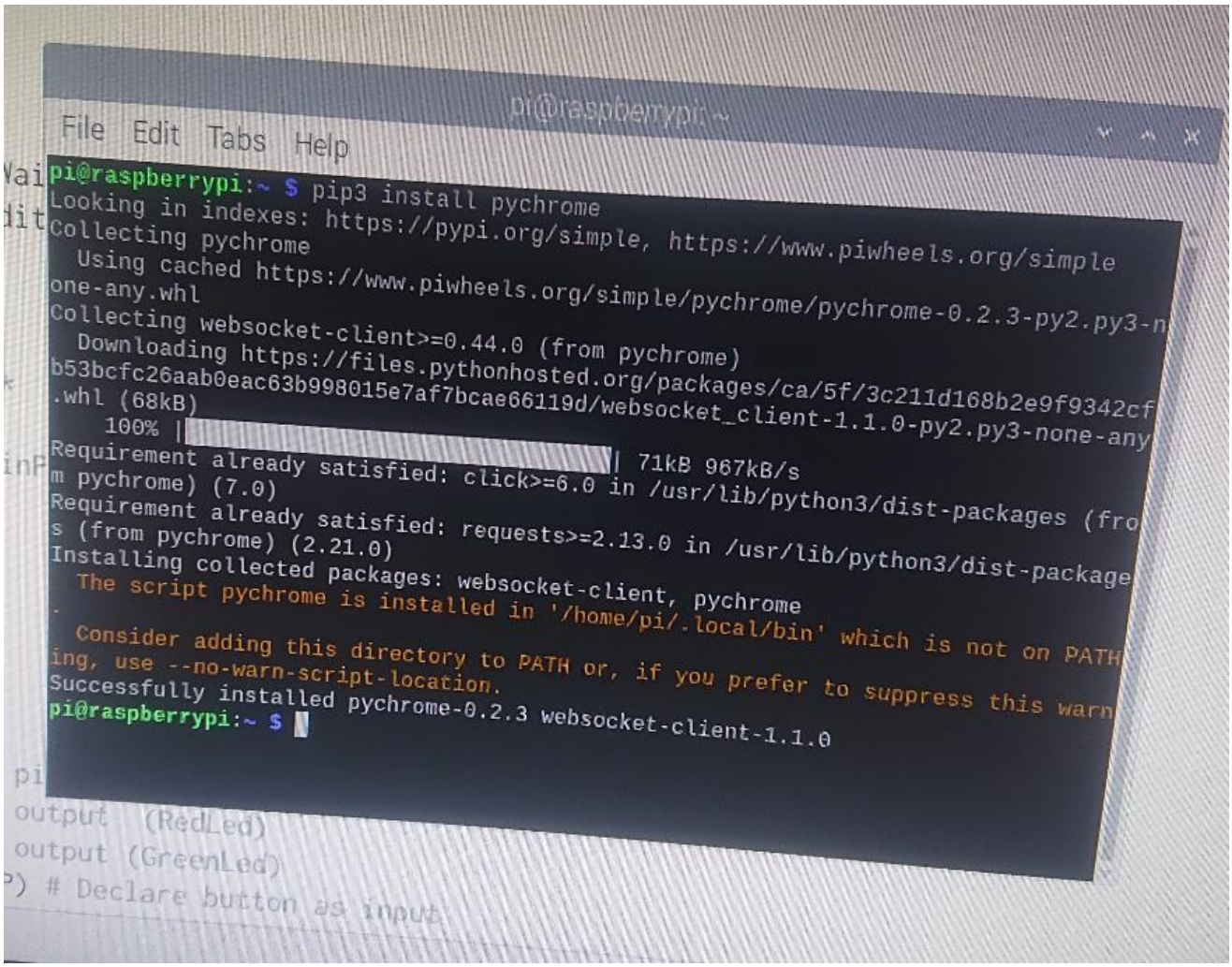


Hyväksytään kirjoittamalla ”y” ja painalla enteriä.

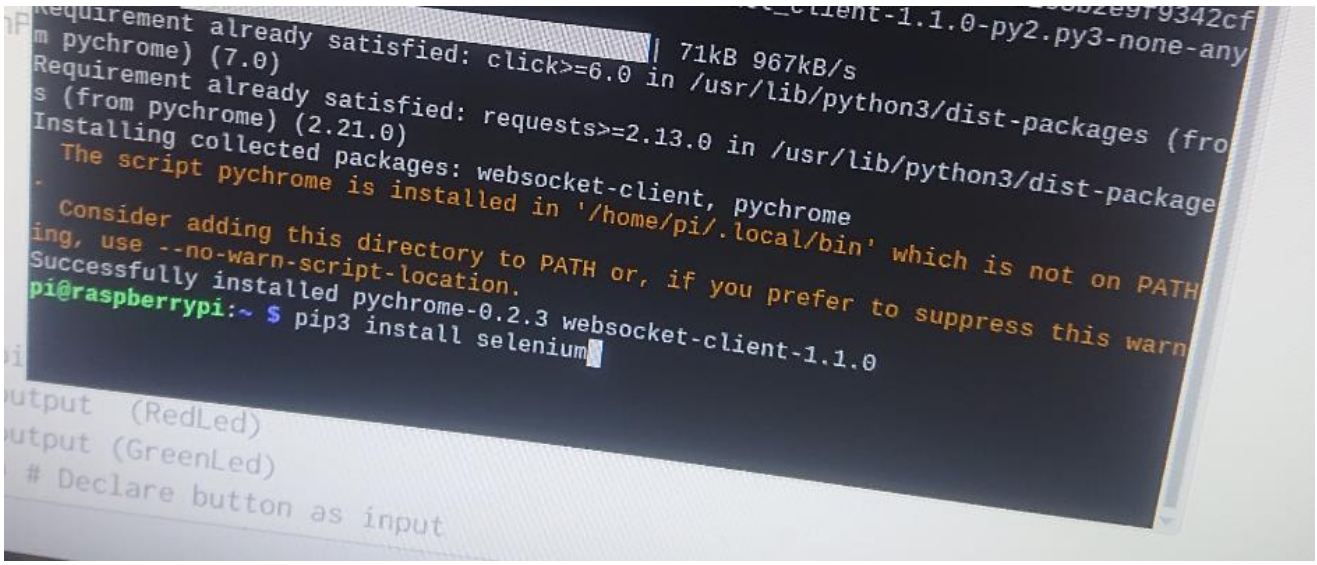


Päivitys-ikkunan näkymä, kun päivityksiä ladataan.

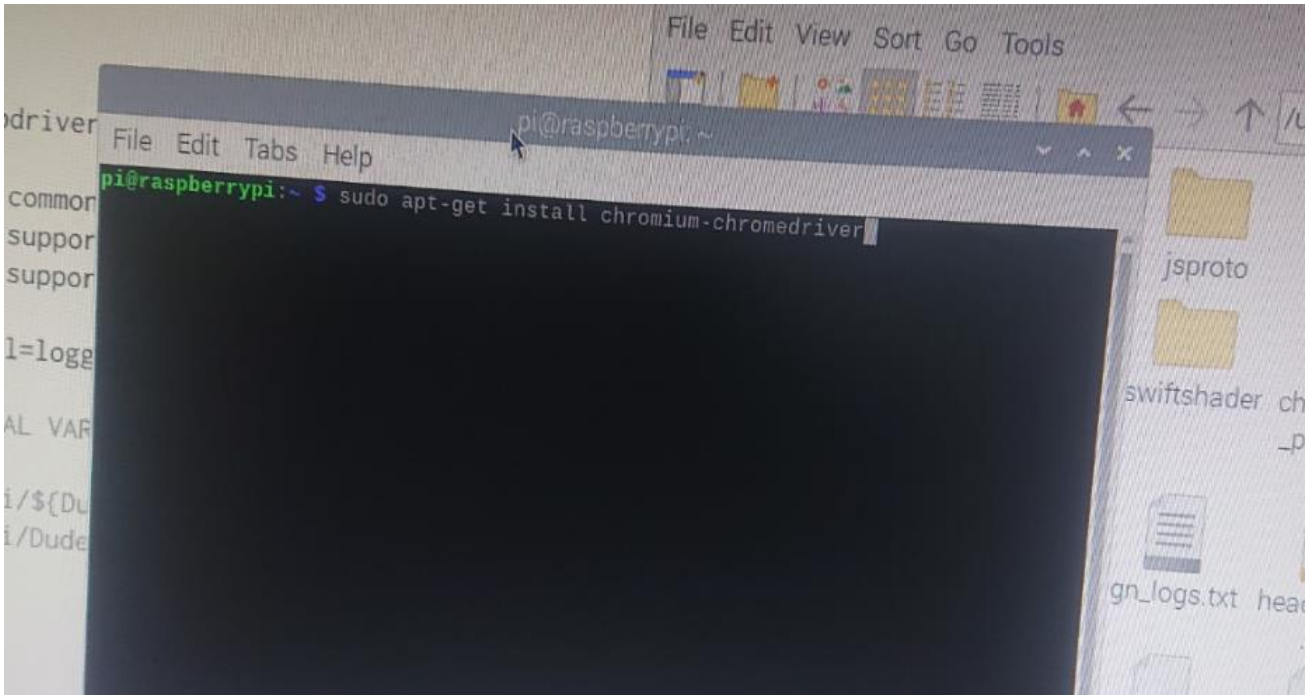
Onnistuneiden päivitysten jälkeen ladataan laajennukset.



Kirjoitetaan ”pip3 install pychrome” ja painetaan enteriä.



Kirjoitetaan ”pip3 install selenium” ja painetaan enteriä.



Kirjoitetaan ”sudo apt-get install chromium-chromedriver” ja paina enteriä.

Tämän asennuksen jälkeen kirjoitetaan vielä lopuksi ”sudo reboot” ja paina enteriä. Raspberry Pi käynnistyy uudelleen. Uudelleenkäynnistyksen myötä kaikki uudet valitut asetukset tulevat voimaan.

Uudelleenkäynnistuksen jälkeen koodin voi vielä lisätä mu-editoriin ja se alkaa toimimaan välittömästi. (Huom. ohjelma ei toimi ilman oikeita sähköpostiasetuksia, ohjeet löytyvät alempaa)

Etäasiakaspalvelu-järjestelmän Python ohjelman koodi kokonaisuudessaan

```
import time_# time module
import RPi.GPIO as GPIO_# "as" GPIO = changes variable name
from signal import pause_# pause from signal module
import json
import pychrome
import smtplib, ssl
from selenium import webdriver
import logging
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

logging.basicConfig(level=logging.INFO)_# I do something useful

globvar_Media = 0_# GLOBAL VARIABLE @ ***WEB LOGS***

#url="https://meet.jit.si/${Dudetest1}#config.prejoinPageEnabled=false"
#url="https://meet.jit.si/Dudetest1"

# ***SENSOR_SETUP***

ir_sensor = 37
button = 36

GPIO.setmode(GPIO.BOARD)_# this enables pin order = pit_n number 1-40
GPIO.setup(40, GPIO.OUT)_# Declare pin number 40 as output  (RedLed)
GPIO.setup(38, GPIO.OUT)_# Declare pin number 38 as output (GreenLed)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)_# Declare button as input

GPIO.setup(ir_sensor, GPIO.IN, pull_up_down=GPIO.PUD_UP)_# Declare ir-sensor as input
delayTime = 1_# Using this value for ir-sensor time delay which = 1s

print_("sensor-test, press ctrl+c to end")

chan_list = (40,38)_# Declaring multiple outputs at the same time (pin40 and 38)
GPIO.output(chan_list, 0)_# Initial state for outputs = LOW/0


# **** WANTED SETTINGS ****

# *** Email Setup ***
sender = "zitzu18@gmail.com"_# ***ADD SENDER EMAIL*** Must be gmail address
psw = "Dudetest192"_# ***ADD SENDER PASSWORD*** gmail address password
receiver = "tom1.laakso@centria.fi"_# ***ADD RECEIVER EMAIL*** any email address will do

# *** Jitsi Room Creation***

# *** IMPORTANT NOTE ***    DO NOT USE SPACES BETWEEN WORDS!!! And **Try To Use Unique Names** ea room_name = "EpicSausageParty2021AndBeer"
room_name= "EpicSausageParty2021AndBeer"_# *** SELECT WANTED JITSI ROOM NAME *** Automatically changes other variables to match selected room name


# ***AUTO_MAIL****

url_room= f"https://meet.jit.si/${room_name}#config.prejoinPageEnabled=false"

def email():
    port = 587  # For starttls
    smtp_server = "smtp.gmail.com"_# automatic
    sender_email = sender
    receiver_email = receiver
    password = psw
    header = 'To:' + receiver_email + '\n' + 'From: ' + sender_email + '\n' + 'Subject:ALERT!!! \n'
    #message = header + '\n There is a customer waiting at https://meet.jit.si/%7Bdudetest1%7D#config.prejoinPageEnabled=false \n\n'
    message = header + f'\n There is a customer waiting at https://meet.jit.si/{room_name} \n\n'
    context = ssl.create_default_context()
    with smtplib.SMTP(smtp_server, port) as server:
        server.ehlo()  # Can be omitted
        server.starttls(context=context)
        server.ehlo()  # Can be omitted
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, message)
        server.quit()


#***CHROME/CHROMIUM OPTIONS***

opt = webdriver.ChromeOptions()
opt.add_argument("--remote-debugging-port=9225")
opt.add_argument("--disable-infobars")
opt.add_argument("start-maximized")
opt.add_argument("--disable-extensions")
opt.add_experimental_option("detach", True)
opt.add_experimental_option("prefs", {
    "profile.default_content_setting_values.media_stream_mic": 1,
    "profile.default_content_setting_values.media_stream_camera": 1,
    "profile.default_content_setting_values.geolocation": 1,
    "profile.default_content_setting_values.notifications": 1
  })


# ****AUTO BROWSER IN AND OUT****


def AutoBrow_IN():
    driver = webdriver.Chrome(options=opt, executable_path='/usr/lib/chromium-browser/chromedriver')
    driver.get(url_room)
    wait = WebDriverWait(driver, 10)
    element_on = wait.until(EC.element_to_be_clickable((By.XPATH, '//*[@id="new-toolbox"]/div/div/div/div[2]/div/div[1]/div/div')))
    driver.quit()

def AutoBrow_OUT():
    driver = webdriver.Chrome(options=opt, executable_path='/usr/lib/chromium-browser/chromedriver')
    driver.get(url_room)
    wait = WebDriverWait(driver, 10)
    element_off = wait.until(EC.element_to_be_clickable((By.XPATH, '//*[@id="new-toolbox"]/div/div/div/div[9]/div/div'))).click()
    driver.quit()


# ****WEB LOGS****

def output_on_end(*args, **kwargs):
    y = ''
    new_dict = {}
    global globvar_Media
    globvar_Media
    print("ITEMS", kwargs.items())
    for type, value in kwargs.items():
        if value == 'Media':
            new_dict['Media'] = kwargs.get('response')
            y = json.dumps(new_dict, indent=4)
            if 'left.wav' in y:
                globvar_Media=True
                print("FINISHED ,globvar_Media")


# ***VIDEO CONNECTION***

def log():
    driver = webdriver.Chrome(options=opt, executable_path='/usr/lib/chromium-browser/chromedriver')
    driver.get(url_room)
    dev_tools = pychrome.Browser(url="http://localhost:9225")
    print("Page Title is : %s" %driver.title)
    tab = dev_tools.list_tab()[0]
    tab.start()
    tab.call_method("Network.emulateNetworkConditions",
                    offline=False,
                    latency=100,
                    downloadThroughput=9375,
                    uploadThroughput=3125,
                    connectionType="cellular3g")
    tab.call_method("Network.enable", _timeout=70)
    tab.set_listener("Network.responseReceived", output_on_end)

# ***COUNTERS***
varcounter = 0
counter = 0_# counts how many times IR-sensor have been activated
button_counter=0_# this is used for state initator for log() state condition
spy_counter=0

try:
    while True:

        if globvar_Media == True and GPIO.input(ir_sensor) == False:_# if button state is true execute this command
            globvar_Media = False
            varcounter = varcounter+1
            if varcounter == 1:
                GPIO.output(chan_list, (1,0))_# changes green led to "off" state and red led to "on" state
                time.sleep(delayTime)# Here we use the 1s delay and reset the current signal
                spy_counter=0
                AutoBrow_OUT()
                print("Button, reset")_# this used to see how push button behaves in console
            if varcounter == 2:_# I USED YO BE NUMMER TRE
                varcounter=0

        elif GPIO.input(ir_sensor) == True and spy_counter==0:_# if button is not pressed and ir sensor gets signal execute this command
            GPIO.output(chan_list, (0,1))_# change red led "off" and green led "on"
            time.sleep(delayTime)_# Here we use the 1s delay and reset the current signal
            spy_counter=spy_counter+1
            counter=counter+1
            print("Detection", counter)_# This is used to view detection and counter, and observe its behaviour at the console
            email()
            if button_counter==1:
                AutoBrow_IN()
            elif button_counter==0:_# if button counter value is 0 execute video() and add +1 to the button_counter
                button_counter=button_counter+1_# add value +1 every time when counter detects movement
                log()

        if GPIO.input(button):_# if button state is true execute this command
            time.sleep(delayTime)# Here we use the 1s delay and reset the current signal
            GPIO.output(chan_list, (1,0))
            print(globvar_Media)
            AutoBrow_OUT()
            spy_counter=0


finally:
  GPIO.cleanup()_# clean any signal "jitter" that could be present after latest program run
  pause()_# Stops signaling so we can avoid GPIO conflicts
```