LAPIN AMK
Lapland University of Applied Sciences

TEKOÄLYN KÄYTTÖ KULUNVALVONNASSA

Eloranta Ville

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2020

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Ville Eloranta	Vuosi	2020
Ohjaaja	Tauno Tepsa		
Toimeksiantaja	Lapin ammattikorkeakoulu, TEQU		
Työn nimi	Tekoälyn käyttö kulunvalvonnassa		
Sivumäärä	32		

Opinnäytetyön tarkoituksena oli toteuttaa Lapin ammattikorkeakoulun TKI-yksikkö TEQUlle älykäs ovivahtijärjestelmä. Opinnäytetyön päätutkimusongelmina tutkittiin kasvojen- ja puheentunnistusta. Näiden lisäksi järjestelmä oli tarkoitus kytkeä sähkölukkoon, jolloin muovisia kulkukortteja ei olisi enää tarvittu.

Järjestelmän toteuttamiseen käytettiin Jetson Nanoa, joka on NVIDIAn valmistama tekoälykehitysalusta. Ohjelmointi suoritettiin Python3-ohjelmointikielellä käyttäen kasvojen- ja puheentunnistukseen soveltuvia kirjastoja. Kasvojentunnistukseen käytettiin HOG-menetelmää, jossa alkuperäinen kuva muutetaan valoisuudenmuutossuuntakuvaksi. Puheentunnistukseen sekä järjestelmän äänien luomiseen käytettiin Googlen ilmaisia verkkorajapintoja.

Suurin ongelma järjestelmässä oli sen ominaisuus tunnistaa henkilöitä valokuvista. Toinen ongelma oli, että Googlen rajapinnat tarvitsivat jatkuvan internetyhteyden. Ongelmista huolimatta työn tuloksena syntyi järjestelmä, joka tervehti nimellä TEQUn toimiston ovella kävijöitä.

Avainsanat kasvojentunnistus, koneoppiminen, puheentunnistus, tekoäly

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information
and Communication Technology
Bachelor of Engineering

Author Ville Eloranta Year 2020
Supervisor Tauno Tepsa
Commissioned by Lapland University of Applied Sciences, TEQU
Subject of thesis Usage of Artificial Intelligence in Access Control
Number of pages 32

The purpose of this thesis was to create a door control system for the Lapland University of Applied Sciences development environment TEQU. The thesis utilized a research about facial recognition and speech recognition. The system was to be connected to an electronic lock which would eliminate the need to use the plastic ID cards.

The system was built on an artificial intelligence development board Jetson Nano. It was programmed with Python3 using libraries capable of performing facial and speech recognition tasks. The used methods included the Histogram of Oriented Gradients method which transforms an image into a simple brightness gradient image. The speech recognition and text-to-speech functions were implemented using Google's web APIs.

The main problem with the system was its ability to recognize faces from photos. A minor problem was that the Google's APIs required an Internet connection. Apart from those the system fulfilled its purpose on keeping an eye on the TEQU's door and greeting the known and unknown people in front of it.

Key words artificial intelligence, facial recognition, machine learning, speech recognition

SISÄLLYS

KÄYTETYT MERKIT JA LYHENTEET

API	Application Programming Interface, ohjelmointirajapinta, jonka avulla kaksi sovellusta voivat keskustella toistensa kanssa (MuleSoft 2020)
CSI	Camera Serial Interface, liitäntätyyppi, jolla voidaan yhdistää CSI-kamera ja isäntälaite
CUDA	Compute Unified Device Architecture, NVIDIAn kehittämä rinnakkaislaskentamalli, jolla laskentatehoa voidaan ohjata keskussuorittimelta grafiikkasuorittimelle (NVIDIA 2020)
FFmpeg	Fast Forward Moving Picture Experts Group, multimediakehys kaikelle olemassa olevalle äänelle ja videolle (FFmpeg 2020)
FLAC	Free Lossless Audio Codec, pakattu häviötön äänitiedostomuoto (FLAC 2020)
GPIO	General Purpose Input Output, pinnijärjestelmä, jonka toimintaa voidaan muokata omaan tarkoitukseen sopivaksi (Barnes 2015, 32–33)
HDMI	High Definition Multimedia Interface, kuvan ja äänen siirrossa käytetty liitäntä
HOG	Histogram of Oriented Gradients, algoritmi, jolla kuva esitetään värisävyn muutoksena (Dalal & Triggs 2005)
PHP	PHP: Hypertext Preprocessor, webohjelmointikieli

PoE	Power over Ethernet, virransyöttö ja verkkoyhteys samassa kaapelissa
RAM	Random-access Memory, tietokoneen työmuisti
SADP	Search Active Devices Protocol, protokolla, jolla verkosta etsitään aktiivisia laitteita (Hikvision 2020)
SPI	Serial Peripheral Interface, protokolla, jolla laitteet voivat kommunikoida keskenään isäntä-loinen-periaatteella.
SQL	Structured Query Language, tietokantakieli
SWIG	Simplified Wrapper and Interface Generator, mahdollistaa C- ja C++-kielisten ohjelmien käytön muissa kielissä (SWIG 2019)
TKI	Tutkimus-, kehittämis- ja innovaatiotoiminta, ammattikorkeakoulujen harjoittama avoin tutkimustoiminta (Lapin AMK 2020)
TPU	Tensor Processing Unit, Googlen kehittämä tekoälysuoritin (Google 2020)
USB	Universal Serial Bus, yleisliitäntä tietokoneisiin liitettäville laitteille.

1 JOHDANTO

Tekoäly on mukana arjessa joka päivä. Jokainen on ollut sen kanssa tekemisissä, vaikkei olisi sitä itse huomannutkaan. Yleisin arkipäivän tekoälykokemus on Google-haku, jossa Googlen tekoäly suosittelee verkkosivuja käyttäjän antamien ehtojen perusteella. Kun pelataan tietokonetta vastaan videopeliä, vastustajana on tekoäly. Lisäksi arjessa on mukana lukematon määrä erilaisia puheohjattavia avustajia. Nämä kaikki ovat tekoälyä käyttäviä sovelluksia ja laitteita.

Tämän opinnäytetyön tarkoituksena oli kehittää Lapin ammattikorkeakoulun TKI-yksikkö TEQUa asiakasystävällisemmäksi ja tehdä sitä tunnetummaksi opiskelijoille. Toimeksiantona oli toteuttaa prototyyppi ovivahtijärjestelmästä, joka valvoisi TEQUn toimiston ovea. Järjestelmän tuli sisältää kamera, joka tunnistaisi toimiston ovelle tulijan ja tietäisi, onko tällä oikeus tulla sisään. Siinä täytyi myös olla mikrofoni ja kaiutin, joiden kautta järjestelmä pystyisi tervehtimään tunnettuja tulijoita ja tulija voisi antaa äänikomentoja. Lisäksi järjestelmä oli tarkoitus kytkeä sähkölukkoon, joka avattaisiin tuntemattomille tulijoille vain sallittuina kellonaikoina. TEQUn omat työntekijät pääsisivät sisään myös muina aikoina.

Opinnäytetyön tärkeimpinä tutkimusongelmina tutkittiin kasvojen- ja puheentunnistusta. Kasvojentunnistus oli järjestelmän ydin, jonka ympärille muut ominaisuudet toteutettiin.

Työn käsittely etenee teorialuvusta käytäntölukuun. Ensin on kerrottu tekoälystä yleisesti. Sen jälkeen on kerrottu, kuinka kasvojen- ja puheentunnistus toimivat. Näiden jälkeen on selostettu, kuinka järjestelmä toteutettiin. Viimeisenä on esitetty johtopäätöksiä ja tuloksia työstä.

2 TEKOÄLYN ESITTELY

2.1 Tekoäly arkielämässä

2.1.1 Tekoälyn käyttökohteet

Tekoäly on käytössä monissa arkipäivän asioissa. Lähes kaikki uudet puhelimet tarjoavat käyttäjälleen mahdollisuuden avata puhelimen lukituksen kasvojentunnistusta käyttäen (La 2018). Turvallisuusviranomaiset tunnistavat vaarallisia ihmisiä kaduilta kasvojen perusteella (Keränen 2017), ja oppilaitokset käyttävät kasvojentunnistusta opiskelijoiden läsnäolon seurantaan (Uusiteknologia 2019). Lisäksi monista autoista löytyy nopeusrajoituksia seuraava kamera, joka kuvaa liikennemerkkejä ja kertoo kuljettajalle, paljonko nopeusrajoitus on kyseisellä tieosuudella.

Puheentunnistus on käytössä muun muassa Amazonin Alexassa ja Applen Sirissä. Ne tunnistavat käyttäjänsä puhetta ja pystyvät sen perusteella suorittamaan toimintoja, kuten musiikin toistoa tai puheluiden soittamista. Google raportoi vuonna 2018, että 27 prosenttia kaikista puhelimella Google-hakuja tekevistä ihmisistä käytti hakemiseen puheentunnistusta (Google 2018).

2.1.2 Tekoälyn haasteet

Kasvojentunnistustekniikka löytyy useista puhelimista, mutta tietyt mallit ovat luotettavampia kuin toiset. Kun tekniikka alkoi yleistyä vuonna 2011, ensimmäiset sitä käyttäneet Android-puhelimet oli mahdollista huijata auki esimerkiksi valokuvaa käyttämällä (La 2018). Keväällä 2019 ilmestyneen tutkimuksen mukaan vieläkin yli 40 prosenttia testatuista puhelimista oli mahdollista huijata auki valokuvalla (Kulche 2019).

Kehittyneempi kasvojentunnistus, jota ei voi huijata valokuvalla, perustuu kasvonmuotojen kartoittamiseen infrapunavalolla. Muun muassa Apple käyttää Face ID nimistä tekniikkaa iPhone X -sarjassaan. Puhelimien etuosassa on TrueDepth-kamera, joka sisältää valokuvakameran lisäksi 3D-näkemisen mahdollistavia komponentteja. Näiden avulla kamera kuvaa 30 000 kasvonpistettä ja luo niistä

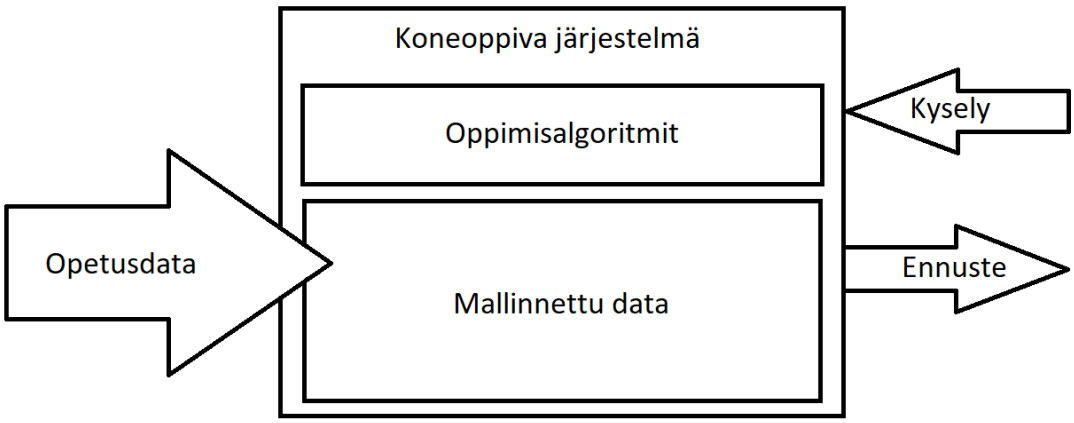
3D-kuvan, jonka jälkeen kuvaa verrataan tallennettuihin kasvoihin. Applen mukaan tekniikka tunnistaa henkilön meikkien läpi sekä suuren parran ajamisen jälkeen. Lisäksi aurinkolasit ja muut asusteet eivät ole sille ongelma ja se toimii myös täysin pimeässä. (Apple 2019.)

Puheentunnistus ei ole niin turvallinen ominaisuus kuin voisi luulla. Esimerkiksi puhelimen käyttö on kielletty autossa, mutta puhuminen ei. Kuljettajalle voi tulla houkutus käyttää puhelinta puheentunnistuksella ja soittaa kaverille. Puhelin voi arvaamattomasti puhelun sijaan alkaa soittamaan kaverin nimistä kappaletta, jolloin kuljettajan huomio keskittyy väärään kappaleeseen ajamisen sijaan. (Van der Velde 2019.)

2.2 Koneoppiminen

2.2.1 Koneoppimisen toiminta

Koneoppimisen tarkoituksena ei ole ohjelmoida järjestelmää toimimaan yhdellä tietyllä tavalla, vaan koneelle syötetään suuri määrä raakadataa (Kuvio 1). Tästä raakadatasta kone oppii itsenäisesti tiettyjen algoritmien avulla. Esimerkiksi koneelle voidaan antaa useita kuvia kissoista ja koirista ja kertoa, mitkä kuvat esittävät kissaa ja mitkä koiraa. Tällöin kone oppii erottamaan kissat ja koirat myös uusista kuvista, joita se ei ole ennen nähnyt. (Pietikäinen & Silvén 2019, 70.)



Kuvio 1. Koneoppimisen toimintaperiaate (Pietikäinen & Silvén 2019, 70)

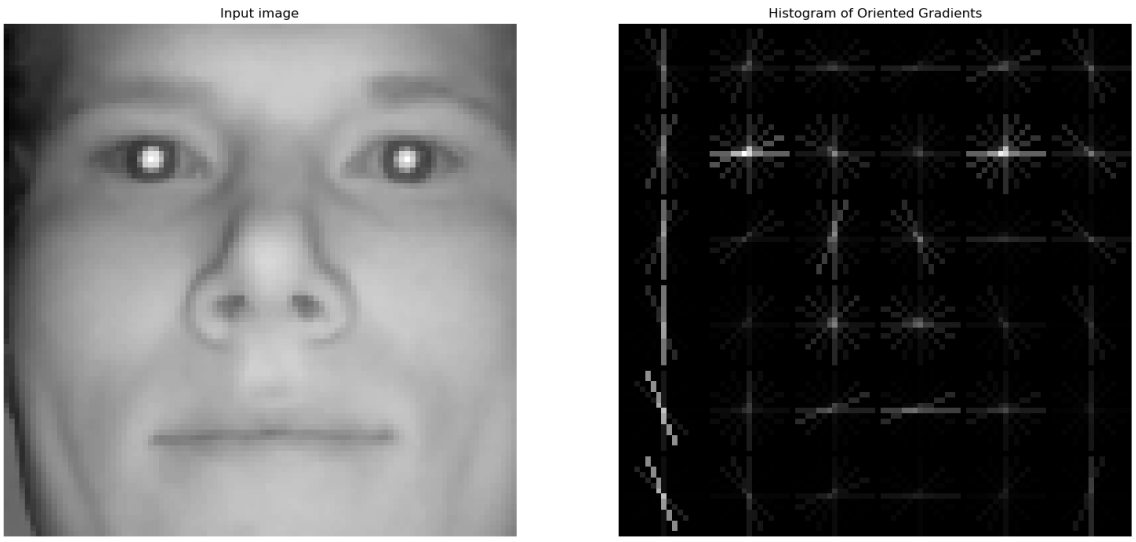
Kuviossa on esitetty, kuinka koneoppiminen toimii. Vasemmalla on kuvattuna isolla nuolella raakadata, jonka määrä lisää ennusteen luotettavuutta. Oikealla on

kuvattuna, kuinka järjestelmältä kysytään, onko kyseessä esimerkiksi kissan kuva. Ennuste tarkoittaa tulosta, jonka järjestelmä antaa ulos. Se on sitä tarkempi, mitä enemmän raakadataa järjestelmä on saanut käsiteltäväkseen. Ennuste voi esimerkiksi olla, että kyselykuva on sataprosenttisesti kissa, 60-prosenttisesti kissa tai ei kissa ollenkaan.

2.2.2 Kasvojentunnistuksen toiminta

Kasvojentunnistus on joukko koneoppimisen algoritmeja. Ensimmäisenä vaiheena on etsiä kuvasta kasvot käyttämällä esimerkiksi HOG-algoritmia. Siinä kuva muutetaan ensin mustavalkoiseksi, jonka jälkeen jokaista pikseliä verrataan viereisiin pikseleihin. Pikseli korvataan tämän jälkeen nuolella, joka osoittaa kuvan värisävyn muutoksen. Koska kuva on mustavalkoinen, nuoli on aina samansuuntainen riippumatta kuvan valotuksesta. (Geitgey 2016.)

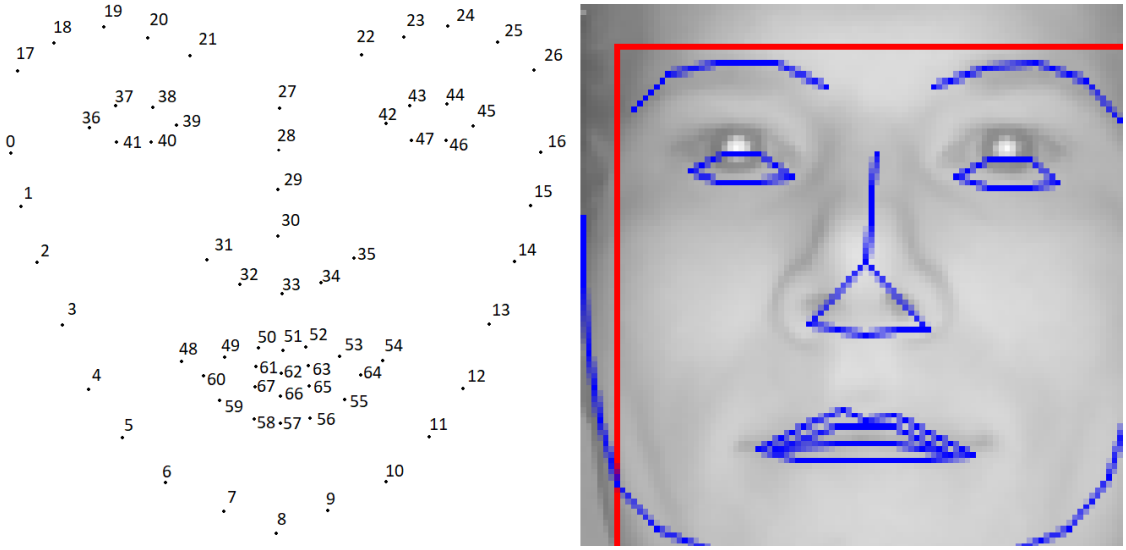
Kaikkia pikseleitä ei kannata käyttää kasvojen etsimisessä, koska tietomäärä olisi liian suuri. Järkevämpää on käyttää 16 x 16 -kokoista aluetta kuvasta ja määrittää tämän suuremman alueen yhteisnuolen suunnaksi se, jota alueella esiintyy eniten (Kuvio 2). Näin syntynyttä HOG-kuvaa on sen jälkeen mahdollista etsiä muista kuvista. (Geitgey 2016.)



Kuvio 2. Mustavalkokuvan muutos HOG-kuvaksi

Kuviosta nähdään, kuinka vasemmanpuoleinen mustavalkoinen kasvokuva on jaettu 16 x 16 pikselin kokoisiin alueisiin. Jokaisella alueella on itsenäinen värisävynmuutossuunta, ja kuva on hyvin pelkistetty. Tietokoneen on helpompi ymmärtää tätä värisävynmuutosmallia.

Seuraavana vaiheena on etsiä kasvoista leuka, silmät, kulmakarvat, nenä ja suu, jotta voidaan tietää, missä asennossa kasvot ovat. Tähän käytetään algoritmia, jolla kasvot esitetään 68:na eri pisteenä (Kuvio 3). Etsimällä nämä pisteet kasvokuva voidaan suunnata aina suoraan, ja henkilö voidaan tunnistaa myös sivukuvasta. (Geitgey 2016.)



Kuvio 3. Kasvonmuotojen etsiminen peruspisteillä (Amos, Ludwiczuk & Satyanarayanan 2016, 8)

Kuviossa on vasemmalla puolella esitetty 68 kasvopisteen malli. Oikealla puolella näkyy, kuinka tekoäly on löytänyt kasvonpiirteet 68 peruspisteen avulla ja merkinnyt ne kuvaan.

Kun kasvonmuodot ovat selvillä, kasvot koodataan. Tätä varten tietokone luo 128 mittausta kasvojen eri kohdista. Näitä mittaustuloksia verrataan muiden tunnettujen kasvojen mittauksiin ja voidaan päätellä, kuka henkilö on kyseessä. Kun tietokoneen vertailtavaksi annetaan kaksi saman henkilön kasvokoodausta ja yhden eri henkilön kasvokoodaus, tietokone oppii, kuka henkilö on kyseessä ja kuka ei. Näin tietokone antaa jatkossa samankaltaisia mittaustuloksia samalle henkilölle. (Geitgey 2016.)

2.2.3 Puheentunnistuksen toiminta

Puheentunnistus on sarja koneoppimisen algoritmeja. Ensimmäisenä vaiheena on nauhoittaa analogista puhetta ja muuntaa se digitaaliseksi tiedoksi. Sen jälkeen nauhoite paloitellaan millisekunnin mittaisiin osiin, jolloin eri foneemit saadaan eroteltua. Foneemit ovat puhutun kielen pienimpiä osia, joita yhdistelemällä voidaan muodostaa eri sanoja. (Grabianowski 2020a.)

Kun eri foneemit on eroteltu, niitä on verrattava ympärillä oleviin foneemeihin kontekstin ymmärtämiseksi. Esimerkiksi */korilla/ ja /gorilla/* kuulostavat samalta, ja ihminen voi käyttää sanoja ristiin. Tämän takia sanaa on verrattava koko lauseeseen, jotta voidaan tietää, puhutaanko eläimestä vai esineestä. (Grabianowski 2020b.)

Puheentunnistuksesta saadaan tarkempi, jos sitä käyttää ainoastaan yksi henkilö. Silloin riittää, että koneelle opetetaan yksi puhetyyli. Jos puhujia on useampia, on koneelle opetettava jokaisen puhetyyli ja aksentti, mikä lisää virheiden määrää tuloksissa. (Van der Velde 2019.)

Taustamelu on puheentunnistuksen ongelma. Kone ei pysty erottamaan koiran haukkumista tai katujyrän melua puheesta. Häiriöiden poistamiseksi koneelle on opetettava, mitä taustamelu on. Sen jälkeen äänestä voidaan suodattaa pois tarpeeton osa, jolloin jäljelle jää pelkkää puhetta. (Van der Velde 2019.)

2.3 Heikko ja vahva tekoäly

Edellä mainitut tekoälyn käyttökohteet edustavat kaikki heikkoa tekoälyä. Siinä järjestelmä suorittaa ennalta ohjelmoituja ohjelmia, eikä periaatteessa ole itse älykäs. Esimerkiksi shakkiohjelma ei osaa pelata shakkia, vaan sen toiminta perustuu ennalta määritettyihin siirtokäskyihin, joita se suorittaa perustuen vastustajan siirtoihin. (Skycode Oy 2020.)

Vahva tekoäly olisi heikkoa älykkäämpi. Se kykenisi ajattelemaan itsenäisesti kuten ihminen ja ymmärtämään olemassaolonsa. Vahvaa tekoälyä ei kuitenkaan ole pystytty luomaan. (Skycode Oy 2020.)

3 JÄRJESTELMÄN TOTEUTTAMINEN

3.1 Alustan valinta

Ensimmäisenä ratkaistavana ongelmana oli käytettävän kehitysalustan valinta. Aluksi suunnitelmana oli käyttää Raspberry Pi -tietokonetta, koska se oli edullinen ja laajasti opetuskäytössä käytetty laite. Aiempiin Raspberry Piillä tehtyihin kasvojentunnistuskokeiluihin perustuen todettiin, ettei sen laskentateho ollut riittävä. Siihen olisi pitänyt liittää erillinen Intelin valmistama laskentatehoa lisäävä USB-tikku tai käyttää tehokkaampia Google Edge TPU Dev Board ja NVIDIA Jetson Nano -kehitysalustoja. (Tepsa 2019.)

Valmistajan tekemän vertailun perusteella Jetson Nano oli suorituskykyisempi kuin vastaavat laitteet (Taulukko 1). Ainoastaan Google Edge TPU oli saanut kahdessa testissä paremman tuloksen kuin Jetson Nano. Suurin osa testeistä ei kuitenkaan ole tuottanut muilla alustoilla ollenkaan tulosta.

Taulukko 1. Kehitysalustojen suorituskykyjen vertailu eri tekoälysovelluksilla (Franklin 2019)

Model	Application	Framework	NVIDIA Jetson Nano	Raspberry Pi 3	Raspberry Pi 3 with Intel Neural Compute Stick 2	Google Edge TPU Dev Board
ResNet-50 (224×224)	Classification	TensorFlow	36 FPS	1.4 FPS	16 FPS	x
MobileNet-v2 (300×300)	Classification	TensorFlow	64 FPS	2.5 FPS	30 FPS	130 FPS
SSD ResNet-18 (960×544)	Object Detection	TensorFlow	5 FPS	x	x	x
SSD ResNet-18 (480×272)	Object Detection	TensorFlow	16 FPS	x	x	x
SSD ResNet-18 (300×300)	Object Detection	TensorFlow	18 FPS	x	x	x
SSD Mobilenet-V2 (960×544)	Object Detection	TensorFlow	8 FPS	x	1.8 FPS	x
SSD Mobilenet-V2 (480×272)	Object Detection	TensorFlow	27 FPS	x	7 FPS	x
SSD Mobilenet-V2 (300×300)	Object Detection	TensorFlow	39 FPS	1 FPS	11 FPS	48 FPS
Inception V4 (299×299)	Classification	PyTorch	11 FPS	x	x	9 FPS
Tiny YOLO V3 (416×416)	Object Detection	Darknet	25 FPS	0.5 FPS	x	x
OpenPose (256×256)	Pose Estimation	Caffe	14 FPS	x	5 FPS	x
VGG-19 (224×224)	Classification	MXNet	10 FPS	0.5 FPS	5 FPS	x
Super Resolution (481×321)	Image Processing	PyTorch	15 FPS	x	0.6 FPS	x
Unet (1x512x512)	Segmentation	Caffe	18 FPS	x	5 FPS	x

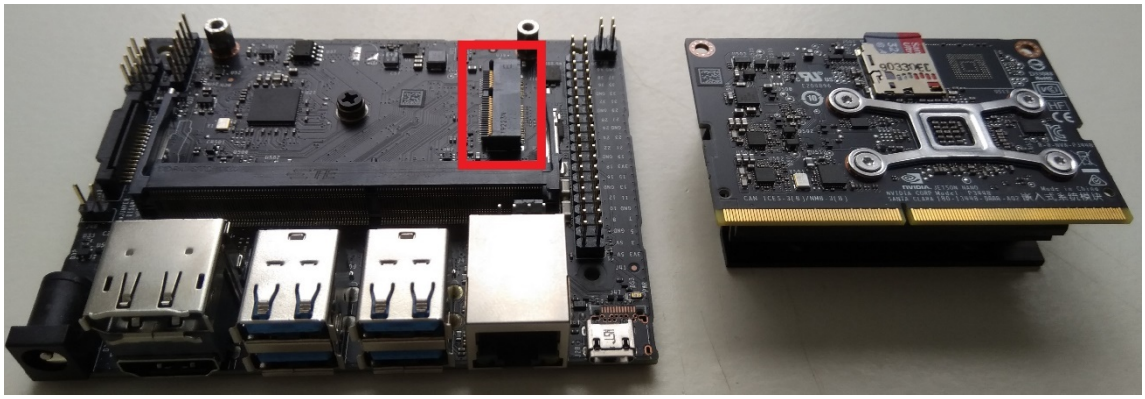
Taulukossa on vasemmalta lähtien kuvattuna testattu ohjelmisto, käyttötarkoitus ja kehys, jossa ohjelmisto toimii. Oikealla puolella näkyy, kuinka ohjelmointialustat ovat suoriutuneet eri käyttötapauksissa. Jetson Nano on suoriutunut kaikista testeistä tuloksen arvoisesti ja kyennyt toistamaan videokuvaa jokaisella sillä testatulla tekoälyn sovelluksella. Muilla alustoilla hylätyt testitulokset on merkitty rastilla.

Raspberry Pi 3 oli edullisin vaihtoehto mutta ei suorituskykynsä takia tullut kysymykseen. Koska muut vaihtoehdot olivat suunnilleen saman hintaisia, ohjelmointialustaksi valittiin testimenestyjä Jetson Nano (Taulukko 2).

Taulukko 2. Ohjelmointialustojen hintavertailu syksyllä 2019 (Mouser Electronics 2019a; 2019b; 2019c; 2019d)

	NVIDIA Jetson Nano	Raspberry Pi 3	Raspberry Pi 3 ja Intel Neural Compute Stick 2	Google Edge TPU Dev Board
Hinta	115,83 €	43,26 €	43,26 € + 71,10 € = 114,36 €	149,16 €

Jetson Nanosta puuttui mahdollisuus käyttää langatonta verkkoyhteyttä. Sen olisi voinut asentaa erillisen verkkokortin avulla (Kuvio 4). Puute ei ollut ongelma, koska kuvaamiseen käytetyssä kamerassa oli mahdollisuus ainoastaan langalliseen yhteyteen. Oli perusteltua käyttää langallista tietoverkkoyhteyttä Jetson Nanoon, koska myös kamera tarvitsi sitä.



Kuvio 4. Jetson Nano purettuna

Kuviossa vasemmalla on Jetson Nanon alapuolinen piirilevy ja oikealla päälle kiinnitettävä piirilevy. Isommassa piirilevyssä näkyy vasemmassa laidassa CSI-2 -kameraliitin. Alalaidassa on vasemmalla ensimmäisenä viiden voltin neljän ampeerin tasajännitteen sisäänmeno. Seuraavaksi päällekkäin ovat Display- ja HDMI-kaapeleiden ulostulot. Sen jälkeen Jetson Nanosta löytyy neljä kappaletta USB 3.0 -ulostuloja. Näiden jälkeen tulevat Ethernet-portti ja viiden voltin kahden ampeerin USB 2.0 -sisäänmeno.

Piirilevyn oikeassa laidassa näkyvät 40 GPIO-pinniä. GPIO on lyhenne sanoista General Purpose Input Output. Osalla pinneistä on kiinteä tehtävä, kuten jännitteen ulosanto. Vapaita pinnejä voi ohjelmoida omaan tarkoitukseen sopivaksi. (Barnes 2015, 32–33.)

Punaisella suorakulmiolla on kuvioon merkitty langattoman verkkokortin paikka. Verkkokorttia ei pääse asentamaan ilman, että piirilevyt irrotetaan toisistaan.

Kuviossa oikealla näkyy Jetson Nanon pienempi piirilevy. Sen alalaidassa näkyvät pinnit, joilla se kiinnittyy isompaan piirilevyyn. Ylälaidassa näkyvät muistikortin paikka ja muistikortti, joka sisältää Jetson Nanon käyttöjärjestelmän ja henkilökohtaiset tiedostot. Piirilevyn alla on nähtävissä jäähdytyssiili, joka jakaa lämpöä pois piirilevyltä.

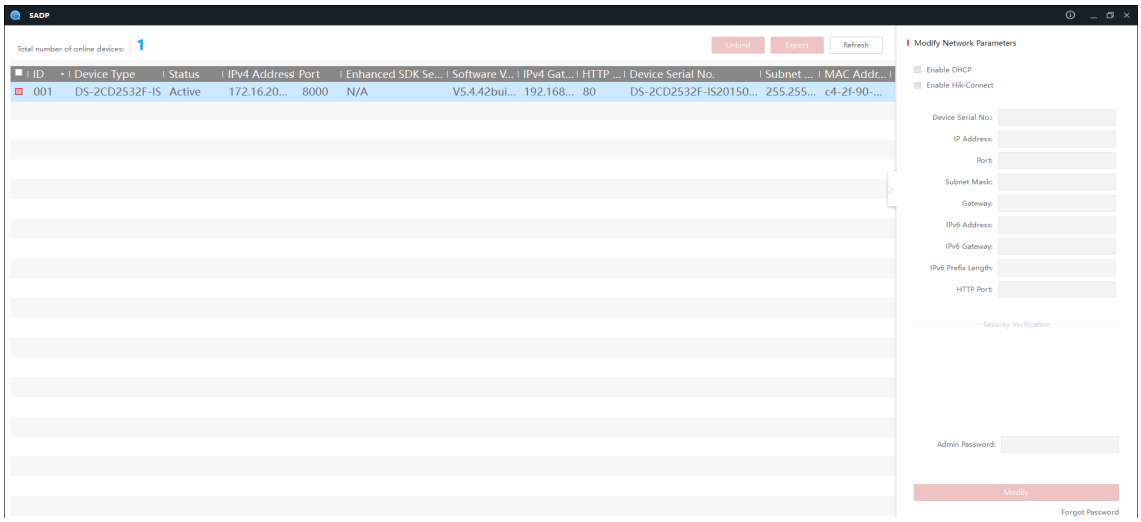
3.2 Kameran käyttöönotto

Järjestelmän kamerana käytettiin Hikvisionin valvontakameraa. Siihen kuuluivat itse kamera sekä PoE-adapteri, joka yhdisti virransyötön ja verkkoyhteyden samaan kaapeliin (Kuvio 5). Kamerassa oli sisäänrakennettu mikrofoni, jota pystyi käyttämään äänikomentojen antoon, mikä helpotti ylimääräisten laitteiden hankkimista.

Kuvio 5. Hikvision-kamera

Kuviossa nähdään vasemmalla puolella PoE-adapteri, joka yhdistää siihen mustassa johdossa tulevan käyttöjännitteen ja harmaassa johdossa tulevan verkkoyhteyden yhteen keltaiseen johtoon. Keltaisessa johdossa kameralle kulkevat sekä virta että verkkoyhteys. Oikealla puolella on kamera kiinnitettynä metalliseen jalustaansa.

Koska kameran käyttäjätunnusta ja salasanaa ei tiedetty, sille täytyi tehdä tehdasasetusten palautus. Tämä suoritettiin irrottamalla sähköjohto, pitämällä kameran reset-nappia painettuna, kytkemällä sähköjohto uudelleen ja pitämällä reset-painiketta pohjassa kymmenen sekunnin ajan (Phil 2018). Sen jälkeen Hikvisionin sivuilta ladattiin SADP-työkalu, jolla otettiin alustettu kamera takaisin käyttöön ja asetettiin sille salasana ja IP-osoite (Kuvio 6).



Kuvio 6. SADP-työkalu Hikvision-kameran asetusten muuttamiseen

Kuviossa näkyy keskellä ylhäällä sinisellä pohjalla verkosta löytyvä IP-kamera. Oikealla puolella näkyvät kentät, joiden avulla voidaan asettaa uusi kiinteä IP-osoite ja salasana.

Videokuvan testaukseen ja toistoon käytettiin Gstreamer-videosoitinta ja gst-launch-1.0-työkalua. Videokuva saatiin näkyviin kirjoittamalla Linuxin terminaaliin komento *gst-launch-1.0 playbin uri=rtsp://<username>:<password>@<ip-address>* (Pratik 2017; Gstreamer 2019). Komennossa *<username>* tarkoittaa kameran käyttäjätunnusta, *<password>* kameran salasanaa ja *<ip-address>* kameran IP-osoitetta.

3.3 Kasvojentunnistus

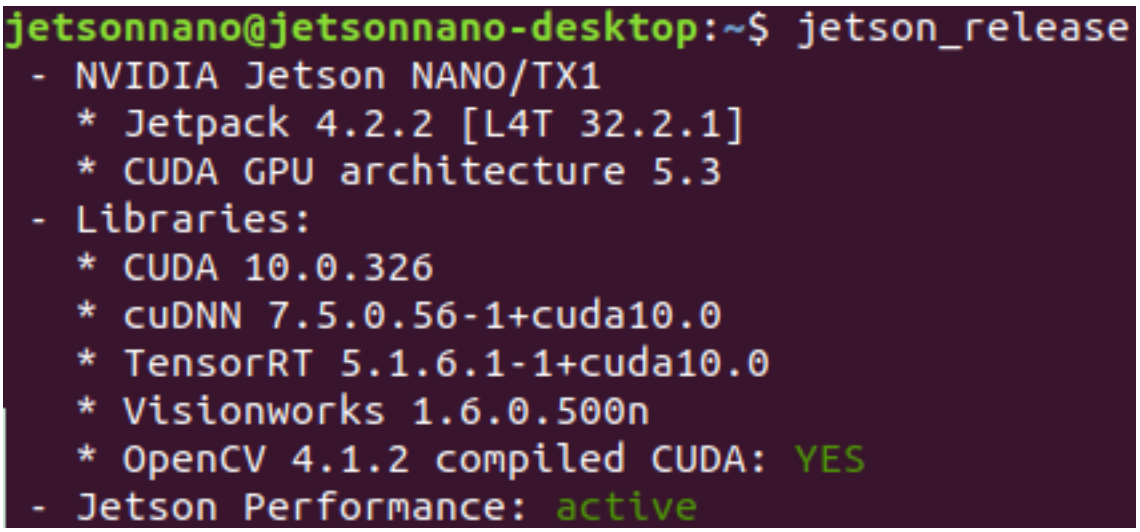
3.3.1 Ohjelmointikirjastojen käyttöönotto

Järjestelmän ohjelmointikielenä toimi Python3 ja kasvojentunnistuksen pääkirjastoina OpenCV- ja dlib-tekoälykirjastot. OpenCV-kirjastolla luettiin yksittäisiä kuvia videokamerasta. Kasvojentunnistus tehtiin Face Recognition -moduulilla, joka toimi käyttäen dlib-kirjastoa. Moduuli asennettiin komennolla *pip3 install face_recognition*.

OpenCV oli valmiiksi asennettuna Jetson Nanossa mutta sen asennus oli puutteellinen. Asia korjattiin asentamalla kirjasto uudelleen lähdekoodista tarvittavilla ohjelmallisilla lisäosilla.

Ensimmäiseksi Jetson Nanoon asennettiin muistinvaihto-ominaisuus, koska sen RAM-muisti ei ollut riittävän suuri. Tämä tapahtui lataamalla GitHubista asennuskansio ja navigoimalla terminaalilla tähän kansioon. Kansiossa asennus suoritettiin komennolla *. installSwapfile.sh*. Aloittava piste on osa komentoa. Muistinvaihdolla Jetson Nano pystyi käyttämään kestomuistia työmuistin jatkeena, jolloin muistia oli tarpeeksi. (JetsonHacksNano 2019a.)

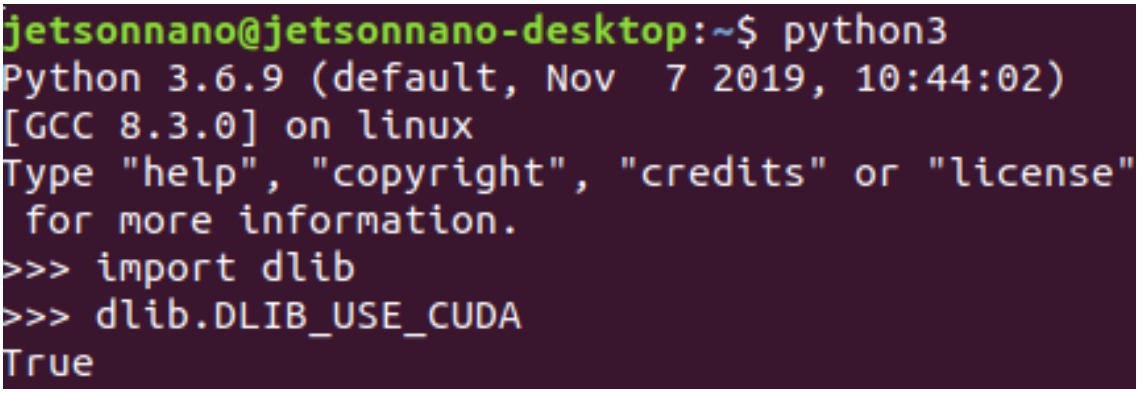
Seuraavaksi GitHubista ladattiin asennuskansio, joka sisälsi ohjelman OpenCV:n asentamista varten (JetsonHacksNano 2019b). Ohjelmasta tarkistettiin, että siihen oli määritelty uusin versio ja tärkeimpinä parametreina CUDA-, Gstreamer- ja Python3-tuet. Sen jälkeen terminaalissa navigoitiin asennuskansioon ja asennus suoritettiin komennolla *. buildOpenCV.sh.* Aloittava piste on osa komentoa. Asennus vei aikaa kolme tuntia ja komennolla *jetson_release* tarkistettiin, oliko asennus onnistunut (Kuvio 7).



Kuvio 7. OpenCV:n uusi asennus ohjelmallisilla lisäosilla

Kuviosta nähdään, kuinka komento *jetson_release* tulostaa Jetson Nanon tiedot. Toiseksi alimmalta riviltä käy ilmi, että OpenCV-kirjasto käyttää CUDA-rajapintaa. Laskentatehoa ohjataan keskussuorittimelta grafiikkasuorittimelle, jolloin saavutetaan suurempi kuvataajuus (NVIDIA 2020).

Dlib-kirjastoa ei ollut Jetson Nanossa valmiina. Asennusta varten se ladattiin dlibin nettisivuilta. Ennen asennusta tiedostosta *dlib/cuda/cudnn_dlibapi.cpp* poistettiin rivi 854, koska kirjastossa oli virhe. Korjauksen jälkeen kirjasto asennettiin siirtymällä asennuskansioon ja suorittamalla komento *sudo python3 setup.py install*. (Geitgey 2019.) Lopuksi tarkistettiin, että dlib-kirjastossa oli CUDA käytössä. Tämä tapahtui avaamalla Python3-konsoli, ottamalla dlib-kirjasto käyttöön ja katsomalla DLIB_USE_CUDA-parametrin arvo (Kuvio 8).



Kuvio 8. Dlib asennettuna CUDAlla

Kuviosta käy ilmi, että Pythonin dlib-kirjastossa on CUDA käytössä. Ensin on avattu Python3-konsoli komennolla *python3*. Sen jälkeen on otettu dlib-kirjasto käyttöön koodilla *import dlib*. Viimeisenä on katsottu DLIB_USE_CUDA-parametrin arvo, joksi Python palauttaa tosi.

3.3.2 Kuvataajuuden optimointi

Kirjastojen asennuksen jälkeen ryhdyttiin ohjelmoimaan järjestelmää. Ohjelmointi suoritetiin Face Recognition -moduulia käyttävän esimerkkikoodin pohjalle. Python-ohjelma toisti videokuvaa, kunnes kuvaan ilmestyi kasvot. Sen jälkeen videokuvaan alkoi kertyä viivettä ja kuvataajuuden laskua. Tämä johtui siitä, että kamera kuvasi enemmän kuvia kuin Jetson Nano ehti käsitellä niitä. Koska kaikki kuvat täytyi käsitellä, kuvataajuus alkoi laskea.

Ongelma ratkaistiin suorittamalla ohjelman eri osia rinnakkain ja asettamalla kuvia jonoon. Kuvaaminen kameralla, raakojen kuvien käsittely ja valmiiden kuvien näyttäminen suoritettiin kaikki samaan aikaan rinnakkain. Kuvat siirrettiin vaiheelta toisella jonossa, jota edellinen vaihe täytti ja seuraava vaihe purki. Jos käsittelemättömiä kuvia kertyi jonoon nopeammin kuin niitä ehdittiin käsitellä eteenpäin, ne unohdettiin ja mentiin suoraan uusimpaan kuvaan. Näin videokuva pysyi reaaliajassa.

Kuvataajuutta pystyi muuttamaan vaihtamalla kuvien kokoa. Suuremmilla kuvilla kasvot löytyivät kauempaa, mutta kuvataajuus kärsi suuremman prosessoinnin takia. Pienemmillä kuvilla kuvataajuus oli parempi, mutta kohteen täytyi olla todella lähellä kameraa, jotta kasvot löytyivät. Sopiva suhde oli käyttää 960 x 540 kokoisia kuvia, jotka tunnistusta varten kutistettiin kolmasosaan. Näin kuvataajuus saatiin tarkoitukseen riittäväksi ja kasvot löytyivät noin metrin etäisyydeltä.

3.3.3 Kasvojen opetus

Esimerkkikoodissa oli valmiina uusien kasvojen tunnistaminen. Ei ollut järkevää opettaa järjestelmälle uusia kasvoja kuvaamalla jokaista haluttua kasvoa erikseen. Toimivampi vaihtoehto oli käyttää Face Recognition -moduulin *load_image_file()*-metodia, jolla luettiin jo olemassa olevia kuvia. Opetus tapahtui

käymällä silmukassa läpi jokainen opetuskuva ja etsimällä kasvot niistä. Henkilöiden nimet otettiin kuvatiedostojen nimistä ja tallennettiin muun tarpeellisen tiedon kanssa tietokantaan. Varsinainen robottikoodi luki sen jälkeen tietokannasta tunnetut kasvot ja etsi niitä videokuvasta.

Tuntemattoman henkilön havaitessaan järjestelmä kysyi, haluaako henkilö tulla tallennetuksi tietokantaan mahdollista myöhempää tervehtimistä varten. Jos henkilö antoi luvan, kysyttiin lisäksi, millä nimellä hän haluaa tallentua. Jos lupaa ei annettu, tiedot säilyivät ainoastaan Python-koodin muuttujassa, josta ne hävitettiin viiden minuutin kuluttua.

3.4 Puheentunnistus

3.4.1 Tekstistä puheeksi

Jotta järjestelmä pystyi toistamaan haluttuja ääniä, siinä käytettiin tekstistä puheeksi -kirjastoa. Ensisijaisena vaihtoehtona käytettiin internetyhteyden vaativaa Googlen gtts-kirjastoa. Sen tuottama ääni oli selkeää ja se toimi suomen kielellä. Kirjasto asennettiin komennolla *pip3 install gtts*.

Gtts-kirjaston ongelmaksi muodostui äänten yhteensopivuus Pythonin äänentoiston kanssa. Aluksi käytetty playsound-moduuli pystyi toistamaan luotuja äänitiedostoja, mutta ei toiminut asynkronisesti Linux-käyttöjärjestelmällä. Ohjelman suoritus keskeytyi äänitiedoston toistamisen ajaksi eikä enää jatkunut toiston loputtua.

Äänentoisto-ongelma korjattiin vaihtamalla Pygame-kirjastoon, joka pystyi soittamaan ääniä taustalla. Pygame asennettiin komennolla *pip3 install pygame*. Se ei toistanut tekstistä luotuja äänitiedostoja, koska ne eivät sisältäneet metadataa eli tietoa itsestään. Ongelma korjattiin vaihtamalla manuaalisesti äänien tiedostopäätteet waveista mp3:ksi. Sen jälkeen Pythonin FFmpeg-kirjastolla muutettiin mp3-tiedostot takaisin waveiksi. Näin äänitiedostoihin ilmestyi metadata ja tiedostoja pystyttiin toistamaan asynkronisesti Pygame-kirjastolla. FFmpeg-kirjasto Pythoniin asennettiin komennolla *pip3 install ffmpeg-python*.

Jos internetyhteyttä ei ollut saatavilla, vaihtoehtona oli käyttää pyttsx3-kirjastoa, joka asennettiin komennolla *pip3 install pyttsx3*. Se pystyi muuttamaan tekstiä puheeksi paikallisesti käyttämällä eSpeak-moottoria, joka asennettiin komennolla *sudo apt-get install libespeak1*. Python-koodissa moottori otettiin käyttöön antamalla se parametriksi rakentajametodille *pyttsx3.init('espeak')*.

Pyttsx3-kirjaston tuottaman äänen laatu oli robottimaisempaa kuin Googlen tuottama. Tästä syystä sitä käytettiin ainoastaan varavaihtoehtona, jos internetyhteyttä ei ollut saatavilla.

3.4.2 Puheesta tekstiksi

Jotta järjestelmä pystyi vastaanottamaan äänikomentoja, siinä käytettiin SpeechRecognition puheesta tekstiksi -kirjastoa. Kirjasto asennettiin komennolla *pip3 install SpeechRecognition*. Siinä oli mahdollista käyttää useiden eri valmistajien, kuten Googlen ja IBM:n, rajapintoja. Puheentunnistukseen käytettiin Googlen rajapintaa, koska sen käyttöön oli Python-kirjastossa API-avain valmiiksi määriteltynä (Amos 2020). Avaimessa oli rajoituksena 50 ilmaista käyttökertaa päivässä, eikä ollut takuita, ettei Google muuttaisi avainta ilman ennakkovaroitusta (Amos 2020). Tämä ei ollut ongelma, koska oli epätodennäköistä tehdä niin montaa rajapintakyselyä päivän aikana ja järjestelmä oli vielä alkutestausvaiheessa.

Kirjaston käyttöä varten Jetson Nanoon asennettiin FLAC ja SWIG. FLAC on häviötön äänitiedostojen pakkausmuoto (FLAC 2020) ja SWIGillä voidaan käyttää C- ja C++-kielisiä ohjelmia mm. Pythonissa (SWIG 2019). FLAC ja SWIG asennettiin komennolla *sudo apt-get install flac swig*.

Puheentunnistusta pystyi tekemään sekä äänitiedostoista että mikrofonista. Mikrofonin olisi pitänyt olla suoraan kiinni Jetson Nanossa, jotta puheentunnistuskirjaston mikrofonimetodia olisi voinut käyttää. Verkon yli käytettävään kameramikrofoniin se ei toiminut.

Asia korjattiin nauhoittamalla FFmpegillä ääntä kamerasta ja tallentamalla se tiedostoon. Sen jälkeen käytettiin normaalia tiedostosta tunnistamista muuttamaan ääni tekstiksi ja käyttämään tekstiä koodissa.

Myös puheentunnistus tarvitsi varasuunnitelman, jos internetyhteyttä ei ollut. Tähän käytettiin puheentunnistuskirjaston *recognize_sphinx()*-metodia, joka käytti paikallista PocketSphinx-moottoria. Ennen kuin moottorin pystyi asentamaan, oli Jetson Nanoon asennettava kaksi tarvittavaa äänipakettia. Ne asennettiin komennolla *sudo apt-get install libpulse-dev libasound2-dev*. Tämän jälkeen PocketSphinx-tuki Pythoniin asennettiin komennolla *pip3 install pocketsphinx*.

PocketSphinxissä ei ollut valmiina englannin lisäksi muita kieliä. Ranskan, italian ja kiinan kielet olisi voinut asentaa erillisistä paketeista, mutta muut kielet olisi pitänyt ohjelmoida alusta asti itse (Zhang 2018). Tähän ei opinnäytetyössä ollut aikaa, joten varajärjestelmä toimi ainoastaan englanniksi. Lisäksi PocketSphinxin toimintavarmuus oli heikompi kuin Googlen. Välillä se ei ymmärtänyt ollenkaan puhetta tai tulkitsi sanoja väärin.

3.5 Tietokanta

3.5.1 Asennus

Järjestelmän tietokantapalvelimena toimi MariaDB, joka asennettiin komennolla *sudo apt-get install mariadb-server*. Tietokantapalvelimen asennus viimeisteltiin komennolla *mysql_secure_installation*, jolla asetettiin palvelimen pääkäyttäjälle salasana, estettiin etäkirjautuminen ja poistettiin anonyymit käyttäjät ja testitietokannat. Tämän jälkeen tietokantaa voitiin käyttää komennolla *mariadb -u root -p*.

Palvelimeen oli ensimmäiseksi luotava tietokanta. Tämä tapahtui komennolla *CREATE DATABASE <tietokannannimi>;*. Ennen kuin SQL-kyselyitä pystyi tekemään, oli valittava, mitä tietokantaa käytettiin. Tämä tapahtui komennolla *USE <tietokannannimi>;*. Tämän jälkeen tietokantaan voitiin luoda tauluja ja tallentaa tauluihin tietoa.

3.5.2 Yhteys Pythoniin

Tietokantakirjastona Pythonissa käytettiin mysql-connector-python-kirjastoa, joka asennettiin komennolla *pip3 install mysql-connector-python*. Kirjaston käytössä oli tärkeää huomata, ettei SQL-kyselyissä saanut käyttää valmiita Python-

muuttujia, sillä ne mahdollistivat SQL-injektioiden tekemisen. Turvallinen tapa kyselyiden tekemiseen oli käyttää järjestettyä ja muuttumatonta listaa, joka sisälsi halutut arvot. Tämä lista annettiin parametriksi kyselyiden tekemiseen tarkoitetulle *execute()*-metodille. (Python 2020.)

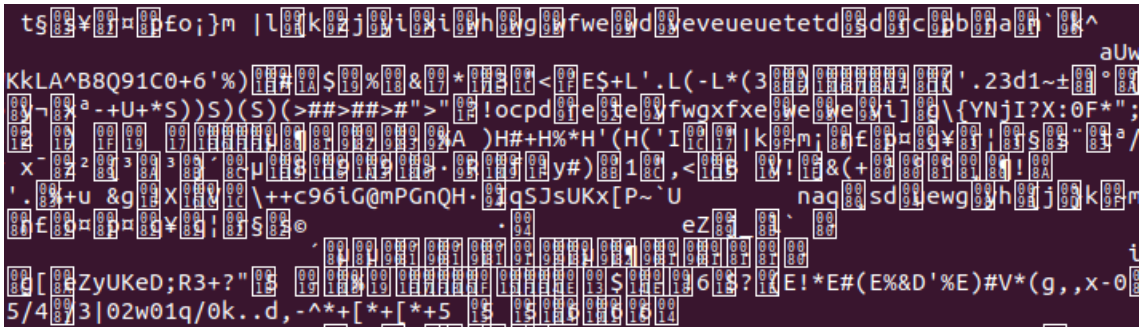
Tietokantaan tallennettiin henkilöistä nimi, kasvokuva, kasvokoodaus ja tieto, kuuluiko henkilö henkilökuntaan. Lisäksi tallennettiin apukentät, jotka kertoivat, oliko henkilöä jo tervehditty ja oliko vierailu jo laskettu. Apukentillä estettiin jatkuva tervehtiminen ja laskeminen, jos henkilö jäi oleskelemaan kameran eteen.

Ongelmaksi muodostui, missä muodossa kasvokuvat ja -koodaukset tallennettiin. Pythonissa ne olivat moniulotteisina taulukoina. Niitä ei voinut sellaisenaan tallentaa tietokantaan, joten ne muutettiin tavujonoiksi taulukon *dumps()*-metodilla. Muunnoksen jälkeen ne tallennettiin tietokannan tavumuotoiseen kenttään.

Hakiessa tietoa takaisin tietokannasta Pythoniin käytettiin numpy-kirjaston *loads()*-metodia. Numpy asennettiin komennolla *pip3 install numpy*. Metodilla tietokannassa tavumuotoisena olleet kasvokuvat ja -koodaukset muutettiin takaisin taulukoiksi. *dumps()*-metodia kutsuttiin suoraan taulukolle, mutta *loads()*-metodia kutsuttiin staattisesti numpy-kirjastosta *numpy.loads().* Metodille annettiin parametriksi tavumuotoinen data.

3.5.3 PhpMyAdmin

Tarkoituksena oli käyttää tietokantaa ainoastaan terminaalissa. Tämä oli haastavaa, koska kasvokuvat ja -koodaukset olivat tietokannassa tavumuodossa. Jos tietokannasta haki kasvoja, terminaali näytti tietokannan kentän kaiken tiedon ihmiselle ymmärtämättömässä muodossa (Kuvio 9).



Kuvio 9. Tietokantanäkymä terminaalissa

Kuviosta ilmenee, ettei tietokantaa kannattanut ylläpitää pelkällä terminaalilla. Nimet olisivat tulostuneet selkokielisesti, mutta kasvotiedot tulostuivat ihmiselle ymmärtämättömässä muodossa.

Tietokannan hallintaan käytettiin phpMyAdmin-työkalua, joka on selaimessa toimiva työkalu tietokantojen käsittelyyn. Se tarvitsee toimiakseen PHP-ohjelmointikielen, joka on tarkoitettu web-sivujen toiminnallisuuden ohjelmointiin. PhpMyAdminissa yksittäisistä kentistä ei näytetä koko sisältöä, vaan tieto on tiivistetty ymmärrettävään muotoon (Kuvio 10).

PHP asennettiin komennolla *sudo apt-get install libapache2-mod-php*. Komennon mukana asentui myös apache-palvelin, joka näytti phpMyAdmin-työkalua selaimessa. PhpMyAdminin terminaaliasennus ei onnistunut, joten se ladattiin phpMyAdminin omilta verkkosivuilta ja purettiin web-palvelimen juurihakemistoon, joka oli */var/www/html* (Petäjäjärvi 2019, 58). Purkuasennuksen jälkeen selaimella avattiin *localhost/index.php*, jossa aukesi tietokannan kirjautumisikkuna.

Ennen kuin tietokantaa pääsi käyttämään selaimessa, oli terminaalissa annettava oikeudet käyttää tietokantaa phpMyAdmin-työkalulla. Tämä tapahtui komennolla *GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY <salasana> WITH GRANT OPTION;*. Komennolla root-käyttäjä sai kaikki oikeudet kaikkiin tietokantoihin kirjautuessaan paikalliselta koneelta ja sai oikeuden myöntää oikeuksia myös muille. Salasanan pystyi halutessaan vaihtamaan tai pitämään samana kuin aiemmin määritelty. (Petäjäjärvi 2019, 59–60.)

id ▾ 1	firstname	lastname	staff	already_greeted	already_counted	face_encoding	face_image
4	Ville	*NULL*	1	0	0	[BLOB - 1.6 KiB]	[BLOB - 90.9 KiB]

Kuvio 10. Tietokantanäkymä phpMyAdminilla

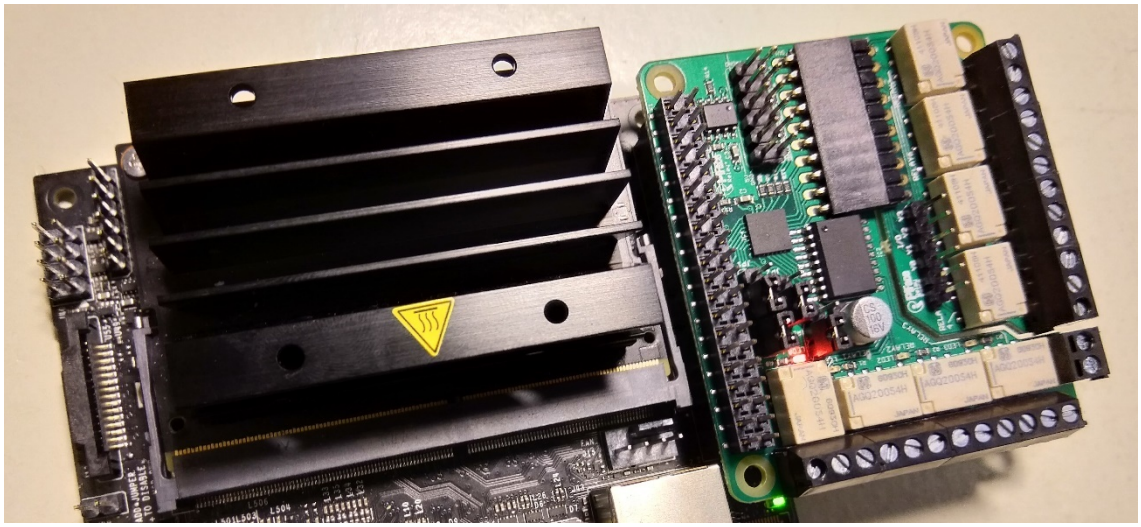
Kuviossa on esitettynä, miltä tietokanta näyttää phpMyAdmin-työkalun avulla. Nimet ja apukentät ovat selkokielisenä ja kasvotiedot on tiivistetty ainoastaan kentän tietotyypiksi ja kooksi.

3.6 Sähkölukko

3.6.1 Relekortin kytkeminen

Kasvojentunnistusta oli tarkoitus käyttää TEQUn toimiston oven sähkölukon hallintaan. Jotta Jetson Nanolla olisi voinut ohjata sähkölukkoa, oli käytettävä apuna relettä. Rele on sähkömagneetilla toimiva kytkin, jolla voidaan pienellä virtapiirillä ohjata isompaa virtapiiriä.

Releenä käytettiin PiFace Relay Plus -relekorttia, joka oli alun perin Raspberry Piille suunniteltu. Koska Jetson Nanon 40 GPIO-pinniä olivat samanlaiset kuin Raspberry Piissä, relekortti oli mahdollista kytkeä suoraan Jetson Nanoon (Kuvio 11). Kortin ohjelmointiin tarvittiin pifacerelayplus-moduulin lisäksi pifacecommon-moduuli, jossa oli kaikille PiFace-korteille yhteisiä määrityksiä. Moduulit asennettiin komennoilla *pip3 install pifacerelayplus* ja *pip3 install pifacecommon*.



Kuvio 11. PiFace Relay Plus yhdistettynä Jetson Nanoon

Kuviossa on esitetty, kuinka PiFace-relekortti on kytketty Jetson Nanon GPIO-pinnien päälle. Vihreällä piirilevyllä olevassa releessä palaa punainen valo, joka kertoo releen olevan ON-asennossa.

3.6.2 SPI-väylän käyttöönotto

Relekortti kommunikoi Jetson Nanon kanssa SPI-väylän kautta. Se ei ollut vakiona käytössä, eikä sen käyttöönotto onnistunut kuten Raspberry Piissä. Jetson Nanossa ei ollut valmista asetusvalikkoa eri väylien käyttöönottoon.

SPI-väylä otettiin käyttöön lataamalla GitHubista pakattu paketti *flash-dtb-update-2019-12-09.tar.gz* ja suorittamalla sen sisältämä *flashme.sh* asennusohjelma (Joseph 2019b). Ohjelma suoritettiin komennolla *sudo ./flashme.sh /dev/mmcblk0* (Joseph 2019a). Käyttöönotto tarkistettiin komennolla *ls /dev/spi**, joka listasi käytettävissä olevat SPI-väylät (Kuvio 12).

Kuvio 12. SPI-väylä käytössä

Kuviossa näkyy, kuinka komento *ls /dev/spi** tulostaa kaikki /dev-kansiossa olevat spi-alkuiset tiedostot. Koska spidev0.0 tulostuu, relekortin tarvitsema SPI-väylä on käytössä.

Käyttöönotosta huolimatta graafisella editorilla suoritettu relekoodi ei toiminut. Ongelmana oli, että ainoastaan root-käyttäjällä oli SPI-väylän käyttöoikeudet. Tämä korjattiin komennolla *sudo chmod 666 /dev/spidev0.0*, jolloin kaikki käyttäjät saivat oikeuden lukea ja kirjoittaa SPI-väylään. Lisäksi kaikille käyttäjille annettiin kirjoitusoikeus */sys/class/gpio/export*-kansioon komennolla *sudo chmod 222 /sys/class/gpio/export*.

Oikeuksien muutoksen jälkeen relekortti alkoi toimimaan graafisesti suoritetulla Python-koodilla. Kuitenkin laitteen uudelleenkäynnistämisen yhteydessä juuri asetetut oikeudet palautuivat takaisin entiselleen. Ongelma ratkaistiin käyttämällä Linuxin ajastuspalvelu crontabia. Palveluun pääsi muokkaamaan ajastuksia komennolla *crontab -e*. Crontabiin lisättiin uudelleenkäynnistyksen yhteyteen suoritettavaksi SPI-väylän oikeuksien muutos koodilla *@reboot sleep 5 && sudo chmod 666 /dev/spidev0.0 && sudo chmod 222 /sys/class/gpio/export*. Komento

odottaa viisi sekuntia, jonka jälkeen se suorittaa oikeuksien vaihdon. Ilman odotusta uudelleenkäynnistymisen yhteydessä käyttöjärjestelmällä on liikaa yhtäaikaisia suoritettavia asioita, eivätkä oikeudet vaihdu.

Viimeisenä ongelmana oli poistaa käytöstä sudon käyttämisen yhteydessä kysyttävä salasana. Sudo on Linux-käyttöjärjestelmässä oleva ohjelma, jota käyttämällä peruskäyttäjä saa pääkäyttäjän oikeudet. Sitä käyttämällä peruskäyttäjä tekee tietoisen valinnan suorittaa mahdollisesti vaarallisia komentoja. Sudoa käytettäessä käyttäjältä kysytään käyttäjän salasana. (Petäjäjärvi 2019, 36.)

Käynnistämisen yhteydessä crontab suoritti sudo-komentoja mutta ei tarjonnut käyttäjälle mahdollisuutta syöttää salasanaa. Tämän takia komennot jäivät suorittamatta, eivätkä oikeudet SPI-väylään muuttuneet. Salasanan kysely poistettiin siirtymällä sudoers-tiedostoon komennolla *sudo visudo*. Tiedoston alimmalle riville lisättiin *<user> ALL=(ALL) NOPASSWD: ALL*, jossa *<user>* tarkoittaa omaa käyttäjänimeä. Lopuksi salasanavälimuisti tyhjennettiin komennolla *sudo -k*. (Moore 2015.)

4 POHDINTA

Tekoälyn käyttö on monipuolinen ja nykyaikainen tapa tehdä asioita. Tavoitteena oli tehdä TEQUa tunnetummaksi asiakkaille ja opiskelijoille, mihin järjestelmä varmasti pystyy. Se tuo TEQUn palveluille lisää mielenkiintoa ja näkyvyyttä. Kulunvalvonnan osalta se kuitenkin kaipaa vielä lisää tutkimuksia.

Nykyisessä muodossaan järjestelmää ei voi käyttää kriittisten tilojen valvontaan ja ovien avaamiseen. Siitä pitäisi saada turvallisempi ja luotettavampi. Ei ole hyvä, että järjestelmää on mahdollista huijata valokuvalla. Lisäksi on varmistettava, että järjestelmä pysyy internetyhteydessä, jotta sen puheentunnistusominaisuus säilyy luotettavana.

Opinnäytetyön tekemisen aikana maailmassa levisi koronavirus, jonka vuoksi oli poistuttava laboratoriosta. Sähkölukkoa ei voitu testata etänä, joten sen liittämistä järjestelmään ei ole tutkittu. Lisäksi ei tiedetä, oliko relekortti sopiva vaihtoehto lukon hallintaan.

Opinnäytetyön tekemisen kannalta olisi ollut parempi, jos ensin olisi tutkittu kunnolla aiheen teoria ja vasta sen jälkeen toteutettu järjestelmä. Työ tehtiin käytäntö edellä, joten tiettyjä teoriaosuuksia jäi huomioimatta. Kasvojen opetuksessa käytettiin ainoastaan yhtä kuvaa usean sijaan. Lisäksi jäi epäselväksi, toimiko Pythonin kasvojentunnistuskirjasto oikeasti HOG-algoritmin mukaan.

Järjestelmän kehitysalustana testattiin ainoastaan Jetson Nanoa, joten ei ole varmuutta, etteikö jokin muu alusta olisi ollut parempi. Jetson Nanon kuvataajuus saatiin nostettua tarkoitukseen soveltuvalle tasolle, mutta ei ole tietoa, olisiko sitä saanut vielä nostettua. Kaikki merkit kuitenkin viittasivat laitteiston maksimikäyttöön.

Lopuksi todetaan, että tekoäly on hieno keksintö ja sitä voidaan käyttää moneen eri tarkoitukseen. Sen käyttö tulee varmasti kasvamaan tulevaisuudessa, kunhan varmistetaan, että se todella toimii. Tekoälyn mahdollisuudet ovat lähes rajattomat, kunhan ei kehitetä ihmistä älykkäämpää supertekoälyä, joka syrjäyttää ihmiset täysin.

LÄHTEET

Amos, B., Ludwiczuk, L. & Satyanarayanan, M. 2016. OpenFace: A general-purpose face recognition library with mobile applications. Carnegie Mellon University. School of Computer Science. Viitattu 13.5.2020 http://reports-archive.adm.cs.cmu.edu/anon/anon/2016/CMU-CS-16-118.pdf.

Amos, D. 2020. The Ultimate Guide To Speech Recognition With Python. Viitattu 21.4.2020 https://realpython.com/python-speech-recognition/.

Apple 2019. About Face ID advanced technology. Viitattu 9.12.2019 https://support.apple.com/en-us/HT208108.

Barnes, R. 2015. Raspberry Pi. The Complete Manual. 5. painos. Bournemouth: Imagine Publishing Ltd. E-kirja. Viitattu 13.5.2020 https://www.academia.edu/19259375/Raspberry_Pi_The_Complete_Manual.

Dalal, N. & Triggs, B. 2005. Histograms of Oriented Gradients for Human Detection. INRIA Rhône-Alpes. France. 14.4.2020 http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf.

FFmpeg 2020. About FFmpeg. Viitattu 30.3.2020 https://www.ffmpeg.org/about.html.

FLAC 2020. Faq. Viitattu 27.3.2020 https://xiph.org/flac/faq.html#general__what_is.

Franklin, D. 2019. Jetson Nano Brings AI Computing to Everyone. Viitattu 4.11.2019 https://devblogs.nvidia.com/jetson-nano-ai-computing/.

Geitgey, A. 2016. Machine Learning is Fun! Part 4: Modert Face Recognition with Deep Learning. Viitattu 8.4.2020 https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

Geitgey, A. 2019. Build a Hardware-based Face Recognition System for $150 with the Nvidia Jetson Nano and Python. Viitattu 23.1.2020 https://medium.com/@ageitgey/build-a-hardware-based-face-recognition-system-for-150-with-the-nvidia-jetson-nano-and-python-a25cb8c891fd.

Google 2018. Global Web Index, Voice Search Insight Report, Global Data n= 400,0001, 2018. Viitattu 16.4.2020 https://www.thinkwithgoogle.com/data/voice-search-mobile-use-statistics/.

Google 2020. Cloud Tensor Processing Units (TPUs). Viitattu 18.5.2020 https://cloud.google.com/tpu/docs/tpus.

Grabianowski, E. 2020a. How Speech Recognition Works. Viitattu 15.4.2020 https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition1.htm.

– 2020b. How Speech Recognition Works. Viitattu 15.4.2020 https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition2.htm.

Gstreamer 2019. gst-launch-1.0. Viitattu 19.11.2019 https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html?gi-language=c.

Hikvision 2020. SADP for windows. Viitattu 17.4.2020 https://www.hikvision.com/en/support/tools/destop-tools/sadp-for-windows/.

JetsonHacksNano 2019a. Install a swap file on the NVIDIA Jetson Nano Developer Kit. This should help with memory pressure issues. Viitattu 20.1.2020 https://github.com/JetsonHacksNano/installSwapfile.

– 2019b. Scripts for build OpenCV on the NVIDIA Jetson Nano Developer Kit. Viitattu 20.1.2020 https://github.com/JetsonHacksNano/buildOpenCV.

Joseph, G. 2019a. jetson-nano-support. Viitattu 24.3.2020 https://github.com/gtjoseph/jetson-nano-support/tree/l4t_32.2.1.

– 2019b. Updated chip select to use the GPIO control method. Viitattu 24.3.2020 https://github.com/gtjoseph/jetson-nano-support/releases/tag/v1.0.2.

Keränen, M. 2017. Mestarien liigan finaalissa ”orwellilaiset” turvatoimet – valvontakamerat skannaavat kaikkien kasvot ja perästä voi kuulua. Tekniikka & talous 28.4.2017. Viitattu 10.10.2019 https://www.tekniikkatalous.fi/uutiset/mestarien-liigan-finaalissa-orwellilaiset-turvatoimet-valvontakamerat-skannaavat-kaikkien-kasvot-ja-perasta-voi-kuulua/875a348a-a233-3c55-a778-0459bfaba4fb.

Kulche, P. 2019. Gezichtsherkenning op smartphone niet altijd veilig. Viitattu 20.12.2019 https://www.consumentenbond.nl/veilig-internetten/gezichtsherkenning-te-hacken.

La, L. 2018. 10 best phones with facial recognition: iPhone X, Note 9, LG G7 and more. Viitattu 20.11.2019 https://www.cnet.com/news/10-best-phones-with-facial-recognition-iphone-x-note-9-galaxy-s9-lg-g7/.

Lapin AMK 2020. Avoin TKI-toiminta. Viitattu 13.5.2020 https://www.lapinamk.fi/fi/Yrityksille-ja-yhteisoille/Avoin-TKI-toiminta.

Moore, J. 2015. Remove sudo Password Prompt. Viitattu 24.3.2020 http://jonmoore.duckdns.org/index.php/linux-articles/58-remove-sudo-password-prompt.

Mouser Electronics 2019a. Viitattu 4.11.2019 https://www.mouser.fi/ProductDetail/Seeed-Studio/102110268?qs=sGAEpiMZZMtw0nEwywcFgJjuZv55GFNmtnsvbGdwckQu55RGCrVu2w%3D%3D.

– 2019b. Viitattu 4.11.2019 https://www.mouser.fi/ProductDetail/Raspberry-Pi/6010602?qs=sGAEpiMZZMve4%2FbfQkoj%252BKjrCjQ40HJu1Igf3%2Fr9r8Y%3D.

– 2019c. Viitattu 4.11.2019 https://www.mouser.fi/ProductDetail/Intel/NCSM2485DK?qs=sGAEpiMZZMsZOJ0x%252BXAYs7%2F3KGuSzFtUexLpSiISQy4RjhJQIiIcxQ%3D%3D.

– 2019d. Viitattu 4.11.2019 https://www.mouser.fi/ProductDetail/Coral/G950-01455-01?qs=sGAEpiMZZMve4%2FbfQkoj%252BMdjKcWOMi0pA3qE2GPZkCk%3D.

MuleSoft 2020. What is an API? (Application Programming Interface). Viitattu 17.4.2020 https://www.mulesoft.com/resources/api/what-is-an-api.

NVIDIA 2020. CUDA Zone. Viitattu 20.4.2020 https://developer.nvidia.com/cuda-zone.

Petäjäjärvi, J. 2019. Palvelinjärjestelmät. Linux-palvelimet. Luento. Lapin ammattikorkeakoulu. Syksyllä 2019. Viitattu 4.3.2020 https://moodle.eoppimispalvelut.fi/pluginfile.php/528794/mod_resource/content/0/linux-materiaali-2019-12-02.pdf.

Phil 2018. How to Reset a Hikvision IP camera using the pushbutton - updated 30th October 2018. Viitattu 19.11.2019 https://www.use-ip.co.uk/forum/threads/how-to-reset-a-hikvision-ip-camera-using-the-pushbutton-updated-30th-october-2018.1767/.

Pietikäinen, M. & Silvén, O. 2019. Tekoälyn haasteet – Koneoppimisesta ja konenäöstä tunnetekoälyyn. Oulu: Oulun yliopisto. Konenäön ja signaalianalyysin keskus. E-kirja. Viitattu 8.4.2020 http://jultika.oulu.fi/files/isbn9789526224824.pdf.

Pratik, K. 2017. Hikvision Raspberry Pi as Viewer. Viitattu 19.11.2019 https://medium.com/@kumarpratik/hikvision-raspberry-pi-as-viewer-9500369a6e8b.

Python 2020. sqlite3 – DB-API 2.0 inteface for SQLite databases. Viitattu 15.5.2020 https://docs.python.org/3/library/sqlite3.html.

Skycode Oy 2020. Mitä tekoäly on? Viitattu 15.4.2020 https://tekoäly.info/mita_tekoaly_on/.

SWIG 2019. Executive Summary. Viitattu 27.3.2020 http://www.swig.org/exec.html.

Tepsa, T. 2019. Lapin ammattikorkeakoulu. Opinnäytetyön ohjaaja. Aloituspalaveri 8.10.2019.

Uusiteknologia 2019. Kasvojentunnistus sopii opiskelijan läsnäolon seurantaan. Uusiteknologia 5.2.2019. Viitattu 20.11.2019 https://www.uusiteknologia.fi/2019/02/05/kasvojentunnistus-sopii-opiskelijan-lasnaolon-seurantaan/.

Van der Velde, N. 2019. How Does Speech Recognition Technology Work. Viitattu 15.4.2020 https://www.globalme.net/blog/how-does-speech-recognition-technology-work/.

Zhang, A. 2018. Notes on using PocketSphinx. Viitattu 21.4.2020 https://github.com/Uberi/speech_recognition/blob/master/reference/pocketsphinx.rst.