**IoT Järjestelmän toteutus**

**IoT Face Recognition**

Jaber Askari (M2947)  
Samson Azizyan (M3156)  
Tuukka Bordi (M2296)  
Arttu Häyrynen (M3350)

Kurssityö

Joulukuu 2019

Tekniikan ala

Insinööri (AMK), Tieto- ja viestintätekniikka

**Sisältö**

[1 Johdanto 3](#_Toc27411131)

[2 Raspberry Pi 4](#_Toc27411132)

[2.1 Main.py 4](#_Toc27411133)

[2.2 Auth.py 5](#_Toc27411134)

[2.3 cl\_facerec.py 6](#_Toc27411135)

[3 ESP32 7](#_Toc27411136)

[3.1 Wifi Server (wifi\_server.ino) 7](#_Toc27411137)

[3.2 ESP Client 7](#_Toc27411138)

[4 Grove.py 7](#_Toc27411139)

[5 Pohdinta 8](#_Toc27411140)

# Johdanto

Kurssin alussa saimme tehtävän suunnitella ja rakentaa IoT-järjestelmä. Ryhdyimme lähestymään tätä sillä ajatuksella, että haluamme tehdä järjestelmän, jonka koemme itsellemme mielekkääksi ja jonka rakentaminen opettaisi meille samalla jotain uutta. Ryhdyimme tutkimaan internetin tee-se-palstoja ja siinä toiveessa, että löytäisimme projektin, joka soveltuisi kurssin vaatimuksiin, olisi mielekäs toteuttaa ja johon olisi olemassa jo valmiiksi hyviä ohjeita ja vinkkejä. Totesimme heti aluksi, että emme halua tehdä sitä perinteistä sääasema -projektia, vaan haluamme tähdätä hieman korkeammalle. Tästä syystä päädyimme tutkimaan projekteja, jotka eivät itsessään ole ns. IoT -projekteja, mutta olisivat mukautettavissa sellaisiksi.

Päädyimme suunnittelemaan kasvojentunnistusjärjestelmän, joka rakentuu Raspberry Pi:n ja sen kameramoduulin varaan. Laajensimme järjestelmän langattomia ominaisuuksia ottamalla projektiimme mukaan ESP32 –laitteen, jonka kanssa on tarkoitus kommunikoida langattomasti. Raspberry Pi suorittaa varsinaisen kasvojentunnistuksen ja ESP32 mahdollistaa järjestelmän ulkopuolisten laitteiden ohjauksen. Käytimme lähtökohtana Seeed studion älylukko projektia, jossa kasvojentunnistusta käytetään sähkölukon ohjaukseen.

Järjestelmämme tarkoituksena on toimia käyttäjän autentikoinnin välineenä ja tuoda kaksivaiheinen autentikointi eli 2FA reaalimaailmaan. 2FA:n ollessa yleinen menetelmä kirjautua monille internet sivustoille, halusimme luoda järjestelmän, joka mahdollistaa sen vaivattoman käytön myös muissa ohjaus- ja valvontakäytöissä.

Ensimmäiset kunnianhimoiset ajatukset olivat luoda järjestelmä kahvinkeittimen ohjaamiseksi, jossa kasvojentunnistusta käytetään menetelmänä, jolla järjestelmävalvoja voi myöntää tai kieltää käyttäjältä kahvinkeittimen käyttöluvan. Tästä luovuttiin kuitenkin hyvin pian koska totesimme sellaisen järjestelmän olevan tarpeettoman monimutkainen tehtävänantoon nähden. Päädyimme suoraviivaistamaan järjestelmää ja ohjaamaan yksinkertaista LEDiä, kahvinkeittimen sijaan, koska tehtävänantona oli osoittaa järjestelmän toimivuus eikä niinkään luoda lopullinen tuote. Tämä ratkaisi monia ongelmia mitä tulee heikko- ja vahvavirtajärjestelmien sulauttamisen yhteydessä.

Jotta projektimme hallinta ja dokumentointi olisi johdonmukaista ja työnjako olisi paremmin organisoitua, päätimme käyttää Gitlabia. Se helpotti työskentelyä myös kurssin tuntien ja luokan ulkopuolella, tiedostojen jaon helpottuessa. Tämä myös helpotti ja suoraviivaisti myös viikoittaista raportointia opettajalle. Gitlabin käyttö saikin kiitosta heti projektin alusta asti.

# Raspberry Pi

## Main.py

Jaoimme projektin kokonaisuuden osiin helpottaaksemme siihen liittyvän työn ja ajan hallintaa. Ohjelman tekeminen moduuleiksi helpottaa myös ohjelman logiikan ymmärrystä ja muokkausta, joten pyrimme tekemään kaikista projektimme ominaisuuksista moduuleita.

Alla on pääohjelmamme koodi. Pääohjelma sisältää koko projektin toimintalogiikan. Logiikka toimii näissä osissa:

* Kasvon tunnistus (cl\_facerec –pythonmoduuli)
* Autentikointi (auth –pythonmoduuli)
* Ledin etäohjaus (esp32client –pythonmoduuli)

Lisäksi käytämme pythonin json-moduulia tunnettujen käyttäjien tietojen parsimiseksi json-tiedostosta.

Ohjelma toimii seuraavasti näissä käyttötilanteissa:

1. Käyttäjä ottaa kuvan, mutta käyttäjää ei tunnisteta: ohjelma alkaa hälyttää summerilla 2 sekuntia
2. Ohjelma ottaa kuvan, käyttäjä tunnistetaan mutta käyttäjä ei lähetä salasanaa tunnistusohjelman hallinnoimaan sähköpostiin: ajan loputtua (n. 2 min) laite laskee epäonnistuneen yrityksen, ei sytytä lediä (joka olisi onnistuneen autentikoinnin merkki), tallentaa kuvan tuntemattomasta käyttäjästä aikaleimoineen ja palaa kasvojentunnistukseen
   * Sama käyttäjä yrittää tunnistautua 3 kertaa mutta epäonnistuu joka kerralla: laite alkaa hälyttää summerilla 2 sekuntia
   * Laite ottaa kuvan, käyttäjä tunnistetaan mutta käyttäjä lähettää väärän salasanan tunnistusohjelman hallinnoimaan sähköpostiin: ohjelma ei tee mitään ennen kuin autentikoinnin aika kuluu umpeen.
3. Laite ottaa kuvan, käyttäjä tunnistetaan, käyttäjä lähettää oikean salasanan autentikointiajan sisällä: laite lähettää verkon kautta komennon ESP32 -laitteelle joka sytyttää ledin.

Autentikointiaika on määritetty auth -moduulissa, kuvan ottaminen ja sen vertaaminen nykyisiin käyttäjiin tapahtuu cl\_facerec –moduulissa niin kuin myös kuvan tallentaminen epäonnistuneiden yritysten jälkeen. Jos käyttäjän todentaminen onnistuu, aktivoidaan LED-valo ESP32-laitteessa tekemämme esp32client-moduulin kautta.

## Auth.py

Osana käyttäjän tunnistamista ohjelmoimme ohjelmallemme moduulin, joka lähettää ja lukee tiedon tunnistuksesta sähköpostilla. Tämä luo erillisen turvallisuuskerroksen järjestelmäämme. Moduuli käyttää tähän sisään koodattua googlen gmail-sähköpostiosoitetta, jonka loimme tätä projektia varten. Moduulissa on kolme funktiota: gen\_key, joka luo avaimen, jota käytämme tunnistuksessa, sendEmail, joka lähettää avaimen sisältävän sähköpostin erikseen määritellylle käyttäjälle ja readEmail, joka lukee ja etsii sähköpostilaatikosta viestiä, jossa on avaimen sisältävä otsikko.

Moduuliin toteutettiin rajoitinominaisuus, joka estää ohjelman jumiutumista sähköpostin lukuun readEmail –funktiossa. Ongelmamme oli, että projektin alussa sähköpostin luku oli toimenpide, joka vaati käyttäjän toimia (napin painallusta), jonka jälkeen ohjelma tarkisti, löytyikö sähköpostista oikea avain.

## cl\_facerec.py

cl\_facerec on ohjelmassame se ohjelma, joka huolehtii kasvojen tunnistuksesta sekä kuvien tallentamisesta aikaleimoineen. Teimme tästä kirjastosta luokkaolion, jotta kasvojen enkoodauksien käsittely olisi nopeampaa. Se tapa, millä face\_recognition –kirjasto toimii, on vertailla kuvien enkoodauksia. Tämän vertailun perusteella kirjasto kertoo, oliko kasvo sama, kuin antamillamme kuvien enkoodauksilla.

Kun cl\_facerec –luokan olio luodaan, lataa se enkoodaukset kaikista haluamistamme kasvoista \_\_init\_\_() funktiossa. Näin toteutettuna tätä lataamista ei tarvitse suorittaa enää uudestaan.

Tämä luokka sisältää kaksi metodia, compare() ja save\_img(). Compare() ottaa kuvan ja kutsuu face\_recognition -kirjastosta kuvien vertailumetodin compare\_faces() -funktion. Tämän mukaan compare() palauttaa joko tunnistetun kasvon nimen, tai stringin “error” jos kasvojen tunnistuksessa tapahtui virhe, tai None jos mikään kasvoista ei vastannut kuvassa olevan henkilön kasvoja.

# ESP32

## Wifi Server (wifi\_server.ino)

Tämä osa-alue oli koodistamme ainoa, joka tehtiin eri kielellä kuin Pythonilla. Ohjelma ajetaan ESP32:lla, joka antaa käskyn luoda socketti, joka kuuntelee porttia 80. ESP32 käynnistää tai sammuttaa LEDin sen mukaan vastaanottaako serveri numeron 1 tai 0.

## ESP Client

Tämä moduuli luotiin keskustelemaan ESP32:n kanssa, joka pyörittää edellisessä kappaleessa mainittua ohjelmaa. Käytännössä ohjelma yhdistää koodissa määriteltyyn osoitteeseen ja lähettää sille parametrissa annetun viestin, meidän ohjelmamme tapauksessa numeron 1 tai 0 tavuina.

# Grove.py

Grove.py -moduulissa on kaksi funktiota: buzzer() ja button(). Buzzer on yksinkertainen funktio, joka soittaa hälytyssummeria sekunnin välein. button() taas odottaa, kunnes nappia on painettu, jonka jälkeen funktio pyöritys loppuu.

Funktioitten käyttötarkoituksena on antaa lisätoimintoja ja helpottaa kasvojentunnistuksen toteutusta. Esimerkiksi button() funktion avulla voit käyttää GrovePi-setin nappia kuvan ottamiseen näppäimistön sijasta. Tämä tekee sovelluksemme käytöstä paljon helpompaa esimerkiksi ulkotiloissa, kun näppäimistöä ei tarvita. Buzzer taas tehtiin yksinkertaisesti hälytystä varten.

Molemmat näistä funktioista käyttävät erikseen ostettavan GrovePi-setin laitteita. GrovePi:n laitteiden käyttämiseksi tarvitaan GrovePi Board.

# Pohdinta

Projektina kasvojen tunnistus ja sen käyttö laitteen ohjaukseen internetin yli on mielenkiintoinen ja ajankohtainen. Saimme myös hyvää harjoitusta todellisen projektityöskentelyyn ja sen haasteisiin. Hyvänä esimerkkinä haasteista on toiminut työn jaon järkeistäminen, kun varsinainen projektityö tehdään hajautetusti eri paikoissa, eri aikoina. Tämä loikin meille haasteita, joita emme osanneet ennakoida työhön ryhtyessämme. Kuitenkin veimme projektin loppuun asti, ja lopputuloksestakin tuli hyvä ja monipuolinen. Varsinkin kaksivaiheinen tunnistus oli todella onnistunut ja voimme sanoa, että siitä tuli hyvä. Kaiken kaikkiaan jokainen oppi tästä työstään jotain.