Santeri Saari

ÄLYPEILIN KÄYTTÖ ÄLYASUMISESSA

Tieto- ja viestintätekniikan koulutusohjelma

2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ÄLYPEILIN KÄYTTÖ ÄLYASUMISESSA

Saari, Santeri
Satakunnan ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutusohjelma
maaliskuu 2018
Ohjaaja: Ylikoski, Mauri
Sivumäärä: 43
Liitteitä: 0

Asiasanat: sulautettu tietotekniikka, teknologia, älytalot, asuminen

---

Tässä opinnäytetyössä tutkittiin älypeilin (smart mirror) käyttöönottoa sekä mahdollisuuksia älyasumisessa. Älypeilillä tarkoitetaan peiliä, jossa voidaan näyttää tietoja, kuten kellonaika, kalenteri ja sääennuste. Tässä työssä älypeili toteutettiin käyttämällä Raspberry Pi-tietokonetta, johon asennettiin avoimen lähdekoodin MagicMirror²-alusta. Peili itsessään koostui Philipsin 49-tuumaisesta televisiosta ja puoliläpäisevästä peilistä sen päällä.

Älypeiliin kehitettiin älyasumista tukevia toiminnallisuuksia, kuten kotiautomaatiojärjestelmästä tietoa hakeva moduuli sekä käyttäjien kirjautumisten seurantajärjestelmä. Näiden kehittämistä varten asennettiin erilaisia ohjelmistoja, kuten web-palvelin ja tietokantajärjestelmä, jotta toiminnallisuudet saatiin luotua. Peilissä hyödynnettiin myös MagicMirror^2-alustan mukana tulevia ja muiden kehittämiä moduuleja, jotta peiliin saatiin enemmän ominaisuuksia, kuten kasvojentunnistus ja etähallinta.

Älypeili käyttöönotettiin osana BaltSe@nior-hankkeen ICT-integroituja ratkaisuja. Hankkeessa oli tarkoituksena kehittää ikäihmisten asumista tukevia prototyyppejä, joita voitaisiin tulevaisuudessa tuotteistaa alan yritysten toimesta Itämeren alueen maissa. Peilin kenttätestaus on tarkoitus suorittaa syksyllä 2018 Ulvilan MeWet-kodissa muiden hankkeen prototyyppien kanssa.

USING A SMART MIRROR IN SMART LIVING

Saari, Santeri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information and Communications Technology
March 2018
Supervisor: Ylikoski, Mauri
Number of pages: 43
Appendices: 0

Keywords: ICT, technology, smart homes, smart living

The purpose of this thesis was to install a smart mirror and research its possibilities as a part of smart living. Smart mirror is a mirror that can be used to display information for the user, for example a clock, weather information and calendar notifications. In this thesis the smart mirror was implemented using a Raspberry Pi with an open source platform called MagicMirror^2. The mirror itself consisted of a 49-inch Philips TV and a one-way mirror on top of it.

Functionality development for the mirror was also carried out during the thesis, which included an information display module that uses a home automation system and a user login tracking system. Both of these, like the mirror prototype itself, were developed from a smart living standpoint. For developing the functionalities different programs had to be installed, like a web-server and a database system. Modules that came with the platform itself as well as downloadable modules were used to have more features on the mirror, like facial recognition and remote control.

The smart mirror was implemented as one of the ICT-integrated solutions in a project called BaltSe@nior. The purpose of the project was to create prototypes for seniors' living environments that different companies from the industry in the Baltic Sea region could later benefit from. Field tests for the mirror are supposed to be carried out in autumn of 2018 in the MeWet-home in Ulvila along with other prototypes from the project.

SISÄLLYS

LYHENTEET

API	Ohjelmointirajapinta (Application Programming Interface)
APT	Paketinhallintatyökalu (Advanced Package Tool)
CSI	Kameraliitin (Camera Serial Interface)
DSI	Näyttöliitin (Display Serial Interface)
eMMC	Integroitu muistipiiri (embedded MultiMediaCard)
FullHD	Täysteräväpiirto (Full High-Definition)
GPIO	Ohjelmoitava yleiskäyttöinen portti (General Purpose Input/Output)
HDMI	Liitin ja liitäntästandardi digitaaliselle kuvalle ja äänelle (High-Definition Multimedia Interface)
HTML	Web-sivujen merkintäkieli (HyperText Markup Language)
ICT	Tieto- ja viestintätekniikka (Information and Communications Technology)
IP-osoite	Verkkoon kytketyn laitteen yksilöivä osoite (Internet Protocol address)
JSON	Yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen (JavaScript Object Notation)
LAMP	Avoimen lähdekoodin ohjelmakokoelma (Linux, Apache, MySQL/MariaDB, PHP/Perl/Python)
LED	Valoa säteilevä diodi (Light-Emitting Diode)
MeWet	Hyvinvointiteknologian tutkimus- kehitys ja oppimisympäristö (Multifunctional environment for Well-being enhancing technology)
MIPI	Maailmanlaajuinen liitäntöjen standardisointiliitto (Mobile Industry Processor Interface)
MySQL	Relaatiotietokantaohjelmisto (My Structured Query Language)
OLED	LED-tyyppi, jota käytetään mm. näyttöpaneeleissa (Organic Light-Emitting Diode)
PHP	Palvelinpuolen komentosarjakieli (PHP: Hypertext Preprocessor)

RCA	Liitin analogisen kuvan ja äänen siirtämiseen (Radio Corporation of America)
REST	HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen (REpresentational State Transfer)
RSS	Verkkosyötemuoto (Really Simple Syndication)
SAMK	Satakunnan ammattikorkeakoulu
SD, SDHC	Muistikorttityyppi (Secure Digital, Secure Digital High Capacity)
Sudo	Ohjelma, jonka avulla voidaan suorittaa ohjelmia toisen käyttäjän oikeuksilla (Substitute User Do)
TRRS	Liitin analogisen äänen ja kuvan siirtämiseen (Tip-Ring-Ring-Sleeve)
USB	Sarjaväyläarkkitehtuuri ja liitin datan ja virran siirtämiseen (Universal Serial Bus)
WLAN	Langaton lähiverkkoteknologia (Wireless Local Area Network)
WWW	Web-sivuista koostuva Internet-palvelu (World Wide Web)

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli toteuttaa älypeilin prototyypin käyttöönotto ja tarkastella sen tuomia mahdollisuuksia älyasumisen tukena. Älyasumisella tarkoitetaan teknologian hyödyntämistä osana päivittäistä elämää kotona. Siinä voidaan hyödyntää esimerkiksi tietotekniikkaa ja automaatiota.

Älypeili toteutettiin käyttämällä Raspberry Pi-tietokonetta, jonka käyttöjärjestelmänä oli Raspbian Jessie. Tämän päälle asennettiin MagicMirror^2-alusta, jonka avulla itse älypeiliosuus toteutettiin. Peilin näyttönä käytettiin 49-tuumaista televisiota, joka oli sijoitettu puoliläpäisevän peilin taakse.

Opinnäytetyö toteutettiin osana BaltSe@nior-hanketta, jonka yhtenä osa-alueena on kehittää ICT-integroituja ratkaisuja ikäihmisten asumiseen. Hankkeessa kehitettyjen prototyyppien tarkoituksena on antaa ideoita Itämeren alueen maiden yrityksille, jotka voisivat myöhemmin tuotteistaa yksittäisiä prototyyppejä tai kokonaisuuksia.

Opinnäytetyössä toteutettu älypeili on toimiva laitekokonaisuus, mutta ei kuitenkaan valmis myytäväksi. Se on lähinnä proof-of-concept-tyyppinen testialusta, jonka tuotteistaminen vaatisi lisäkehitystä.

2 LAITTEISTO

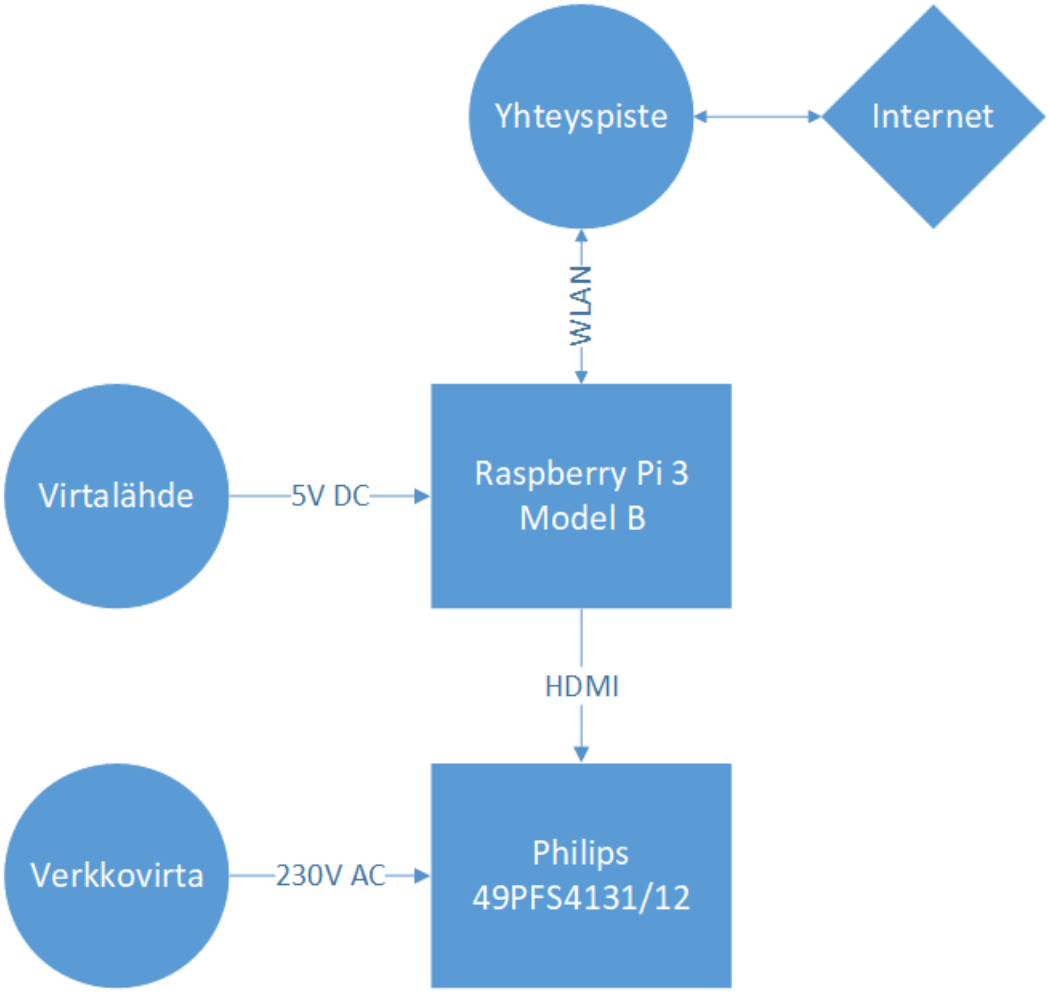
2.1 Raspberry Pi

Älypeilin tietokoneena käytettiin Raspberry Pi 3 Model B -tietokonetta. Raspberry Pi on yhden piirilevyn tietokone, joka julkaistiin vuonna 2012. Se kehitettiin opetuskäyttöön ja kehitysmaihin, mutta halvan hintansa vuoksi on hyvin suosittu myös erilaisten harrastajien parissa. Pienen kokonsa ja virrankulutuksensa vuoksi se soveltuu myös hyvin erityyppisiin sulautettuihin ratkaisuihin. (Upton & Halfacree 2013, 24.)

Raspberry Pin ensimmäinen malli oli Raspberry Pi 1 Model B. Se julkaistiin helmikuussa 2012, ja oli Raspberry Pi Foundationin ensimmäinen tuote. Sen jälkeen julkaistuissa versioissa on ollut muun muassa tehokkaampia prosessoreja ja enemmän muistia. Raspberry Pistä on myös Model A -mallisarja, joka on fyysiseltä kooltaan vähän kapeampi sekä Zero-malli, joka on kooltaan vieläkin pienempi. Sulautettuja ratkaisuja varten on valikoimassa myös Compute Module-mallisarja, joka on kannettavan tietokoneen muistikamman kokoinen.

Raspberry Pissä ei ole käyttöjärjestelmää valmiiksi asennettuna, vaan se asennetaan SD- tai micro-SD-muistikortille, tai eMMC-muistiin, riippuen laitteen mallista. Käyttöjärjestelmä asennetaan kirjoittamalla käyttöjärjestelmätiedosto suoraan tallennusmedialle levykuvankirjoitustyökalulla.

Raspberry Pi tukee useita eri käyttöjärjestelmiä, joissa on tuki ARM-arkkitehtuurille. Raspberry Pi Foundationin kehittämä Raspbian on Debian-nimisestä Linux-käyttöjärjestelmästä tehty versio Raspberry Pitä varten. Se on valmistajan suosittelema käyttöjärjestelmä, ja siinä on otettu huomioon laitteen ominaisuudet ja rajoitukset, joten se on hyvin optimoitu. Muina käyttöjärjestelmävaihtoehtoina ovat useat Linux-pohjaiset käyttöjärjestelmät, kuten esimerkiksi openSUSE, CentOS, Ubuntu MATE sekä Fedora. Microsoftilta on olemassa Raspberry Pille soveltuva Windows 10 IoT Core, joka on Windows 10:stä tehty versio sulautettuihin järjestelmiin. Myös erilaiset mediakeskuskäyttöjärjestelmät, kuten OpenELEC ja OSMC, ovat suositttuja Pin hyvän mediatoistokyvyn ja pienen koon ansiosta. (Cawley 2017.)



Kuva 1. Lohkokaavio älypeilin laitteistosta.

2.1.1 Raspberry Pi 3 Model B

Tässä opinnäytetyössä käytettävä Raspberry Pi-malli on Raspberry Pi 3 Model B. Se on julkaistu helmikuussa vuonna 2016. Se on tähän mennessä julkaistuista malleista tehokkain ja sisältää eniten ominaisuuksia.

Laitteen järjestelmäpiirinä on Broadcom BCM2837, jonka arkkitehtuuri on 64-bittinen. Aikaisemmissa Raspberry Pi versioissa on ollut 32-bittinen arkkitehtuuri. 3 Model B:n suorittimena on ARM Cortex A53, joka on neliytiminen, ja sen kellotaajuus on 1,2 GHz. Näytönohjain on Broadcom VideoCore IV, ja se kuuluu samaan järjestelmäpiiriin suorittimen kanssa. Keskusmuistia on yksi gigatavu, joka on jaettu suorittimen ja näytönohjaimen käyttöön. Tallennustilana toimii microSDHC-kortti, jolle asennetaan käyttöjärjestelmä ja tallennetaan käyttäjän tiedostot. Tiedostojen tal-

lentamiseen voidaan myös käyttää massamuisteja neljän USB-portin kautta. (Benchoff 2016, Chacos 2016.)

Videoulostulona on HDMI-liitin, jonka kautta saadaan FullHD-resoluutioista kuvaa. Vaikka RCA-liitintää ei enää 3 Model B:stä löydykään, saadaan komposiittivideo 3,5 mm TRRS-liittimen kautta ulos. Laitteeseen voidaan myös liittää suoraan LCD-paneeli MIPI (Mobile Industry Processor Interface) DSI (Display Serial Interface)-liitännällä. Sitä käytetään, kun halutaan liittää esimerkiksi kosketusnäyttö suoraan Raspberry Pihin. HDMI- ja TRRS-liittimistä saadaan kuvan lisäksi myös ääni, vaihtoehtoisesti digitaalinen tai analoginen. (Gibbs 2016.)

Verkkoyhteyden muodostamista varten 3 Model B:stä löytyy Ethernet-sovitin ja uutena ominaisuutena vanhoihin malleihin verrattuna 802.11n WLAN-yhteys. Uutta on myös Bluetooth 4.1. (Heath 2016.)

Raspberry Pi tarvitsee toimiakseen viiden voltin virtalähteen, kuten esimerkiksi älypuhelimen laturin. Ilmoitettu virrankulutus on kaksi ampeeria. Pienempivirtaisempaa virtalähdettä käytettäessä laite ilmoittaa salamasymbolilla liian vähäisestä virransaannista. Raspberry Pissä on micro-USB-liitin, jonka kautta virta syötetään. Virtapainiketta laitteessa ei ole, joten tietokone käynnistyy, kun se liitetään virtalähteeseen.

Yksi Raspberry Pin ominaispiirteistä ovat GPIO-pinnit, jotka löytyvät piirilevyltä. Näihin pinneihin voidaan liittää esimerkiksi antureita, LEDejä, kytkimiä ja muita komponentteja. Kyseisiä pinnejä voidaan hyödyntää useissa eri ohjelmointiympäristöissä. (Benchoff, 2016.)

Raspberry Pihin voidaan liittää kamera 15-pinnisen MIPI CSI (Camera Serial Interface)-liittimen kautta. Kameran ensimmäisessä versiossa oli viisi megapikseliä, toisessa kahdeksan. Kameralla voidaan ottaa yksittäisiä kuvia tai videota, ja sitä voidaan hyödyntää esimerkiksi kasvojen tunnistuksessa.

2.2 Philips 49PFS4131/12

Näyttönä älypeilissä käytettiin Philips 49PFS4131/12-mallista 49-tuumaista televisiota. Television soveltuminen älypeilikäyttöön riippuu muutamasta tekijästä, jotka eivät välttämättä ole kovin ilmeisiä käytettäessä televisiota normaalikäytössä. Esimerkiksi maksimikirkkaus on hyvin tärkeä, sillä puoliläpäisevä peili tummentaa kuvaa merkittävästi.

Näytön natiiviresoluutio on 1920x1080 pikseliä, joka on sama, kuin Raspberry Pin maksimiresoluutio normaalikäytössä. Pistä saadaan siis hyvälaatuinen kuva ulos, kun voidaan käyttää optimaalista resoluutiota. Ruudun päivitystaajuutena molemmat laitteet tukevat 60:ntä hertsiä, joka on hyvinkin riittävä määrä älypeilin kaltaiselle alustalle.

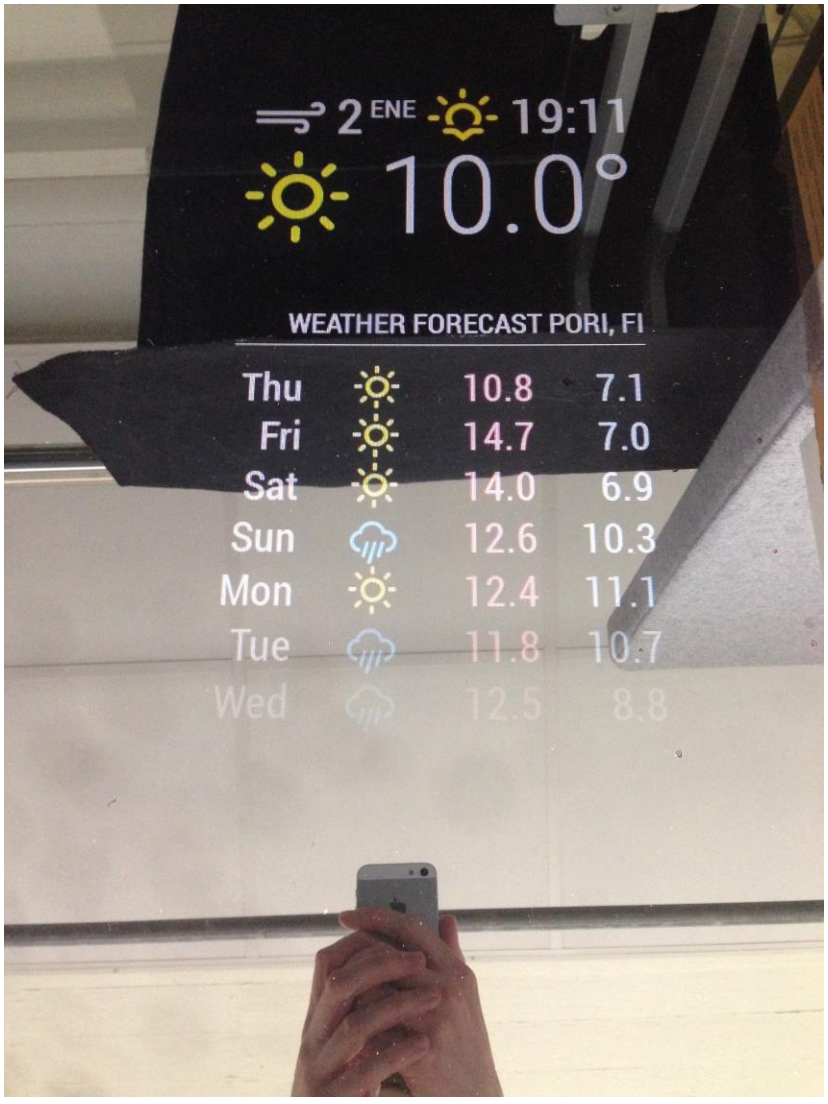
Näyttöpaneelin kirkkausarvo on 280 kandelaa per neliömetri (cd/m^2). Kandela (cd) on valovoiman yksikkö; yksi kandela vastaa suurin piirtein yhden kynttilän lähettämää valon määrää yhden steradiaanin avaruuskulmaan. 280 cd/m^2 on sisätilakäyttöön peilin läpi riittävä arvo, mutta jos peiliä haluttaisiin käyttää kirkkaasti valaistussa huoneessa tai ulkona, niin kirkkaus olisi hyvä olla suurempi peilin tummentavan vaikutuksen takia.

2.3 Puoliläpäisevä peili

Puoliläpäisevä peili on peilityyppi, jonka tarkoituksena on toimia toiselta puolelta peilinä, toiselta puolelta ikkunana. Peilipuolen ympäristön on oltava valoisampi kuin ikkunapuolen jos peilin halutaan toimivan oikein. Tämän tyyppisiä peilejä käytetään paljon esimerkiksi valvontahuoneissa, joista voidaan katsoa mitä lasin toisella puolen tapahtuu, ilman että valvojia voidaan nähdä huoneesta.

Tässä opinnäytetyössä puoliläpäisevän peilin tehtävänä on peittää televisio katsojan näkyvistä. Huone, johon peili sijoitetaan, toimii valoisana puolena ja televisio tummana. Kuten aiemmin mainittiin, suuri osa MagicMirror^2-alustan näkymästä on mus-

taa, joten nämä alueet toimivat peilinä, koska ne ovat lähes täysin tummia; näyttöpaneelin taustavalo valaisee kaikki pikselit, mutta mustista pikseleistä se ei erotu kovin hyvin. Näytettävään informaatioon käytetään valkoista tai muita kirkkaita värejä, jotka erottuvat hyvin peilin takaa. Kirkkaat pikselit ovat kirkkaampia kuin peilin ympäristö, joten ne näkyvät peilin läpi. Kuvassa 1 näkyy, miten kirkas teksti näkyy peilin läpi. Hyvän näkyvyyden takaamiseksi peilin asennuspaikka pitää kuitenkin valita oikein: esimerkiksi auringonvalo estää tekstin näkyvyyden lähes kokonaan. Jos peili haluttaisiin asentaa ulos, pitäisi television maksimikirkkauden olla huomattavasti suurempi.



Kuva 2. Säätiedot ja sääennuste älypeilissä.

3 OHJELMISTO

3.1 Raspbian Jessie

Älypeilissä käytetty käyttöjärjestelmä on Raspbian Jessie. Raspbian on Debian-nimisen Linux-jakelun Raspberry Pitä varten tehty versio. Sen kehittäjä on Raspberry Pi Foundation, joka on myös Raspberry Pin kehittäjä. Jessie on julkaistu vuonna 2015. Opinnäytetyön aikana julkaistiin uudempi versio, Raspbian Stretch, mutta sen tarjoamat ominaisuudet eivät eronneet merkittävästi Jessiestä, joten päivitystä ei nähty tarpeellisena. Päivityksestä olisi saattanut myös ilmetä ongelmia, joten päivitys jätettiin tekemättä. (Long 2015.)

Käyttöjärjestelmänä Raspbian on kuin moni muukin moderni moniajokäyttöjärjestelmä. Järjestelmä käynnistyy automaattisesti graafiseen käyttöympäristöön, mutta käynnistymisasetuksia voidaan muuttaa käyttämällä raspi-config-asetustyökalua. Raspbianista on ladattavissa myös Lite-versio, jossa ei ole graafista käyttöympäristöä, mutta sitä ei käytetty, sillä suuri osa älypeilin ohjelmoinnista tehtiin suoraan Raspberry Pillä käyttäen graafista ympäristöä hyödyksi.

3.2 MagicMirror^2

MagicMirror^2 on modulaarinen avoimen lähdekoodin alusta, joka on suunniteltu älypeilikäyttöön. Alustan kehitti Xonay Labs:in Michael Teeuw. Julkaisun jälkeen kuka tahansa on voinut osallistua kehitykseen GitHub-palvelun kautta. Yhteisön käyttäjät voivat myös jakaa kehittämiään moduuleja toisille.

Modulaarisuudella tarkoitetaan, että alustalle voidaan kehittää erilaisia moduuleita, jotka toimivat omatoimisesti muista riippumatta. Moduuleja voidaan lisätä ja poistaa, aktivoida ja deaktivoida käyttäjän niin halutessa ilman, että alustassa tapahtuu mitään muutoksia. Moduulien kehittäjät voivat myös päivittää moduulejaan muusta järjestelmästä riippumatta.

MagicMirror^2 on ulkonäöltään hyvin yksinkertainen ja pelkistetty, koostuen suurimmalta osaltaan mustasta taustasta. Syynä tähän on puoliläpäisevän peilin toimintaperiaate; kaikki tummat alueet näytöllä toimivat normaalina peilinä kirkkaan tekstin näkyessä läpi.

Jos yhtään moduulia ei ole aktiivisena, näkymä on täysin musta. Moduuleja lisätään antamalla niille jokin tietty alue käytettäväksi. Alueet ja moduulien muut asetukset määritetään config.js-tiedostossa.

MagicMirror^2 toimii Node.js-alustalla. Node.js kehitettiin JavaScript-koodin suorittamiseksi palvelinpuolella.

3.3 MySQL

Käyttäjätietokantaa varten tarvittiin tietokantajärjestelmä, johon voitaisiin kirjata käyttäjien kirjautumisia peilin käytön seuraamiseksi. Tähän tarkoitukseen valittiin MySQL-tietokantajärjestelmä. Se on avoimen lähdekoodin relaatiotietokantajärjestelmä, joka on osa LAMP-ohjelmakokoelmaa. LAMP-lyhenne muodostuu sanoista Linux, Apache, MySQL (tai MariaDB) ja PHP (tai Perl tai Python). LAMP-kokoelman vahvuutena on sen monipuolisuus ja ilmaisuus, jonka vuoksi sitä käytetään esimerkiksi edullisissa webhotelleissa. (Laaksonen, 2009.)

3.4 Apache

Apache on alun perin vuonna 1995 julkaistu avoimen lähdekoodin web-palvelin (Ridruejo & Kallen 2002, 16). Älypeiliin tarvittiin web-palvelinta web-sivua varten, jolta voidaan tarkastaa käyttäjien kirjautumiset. Web-palvelimen avulla tarkastelu voidaan suorittaa toiselta tietokoneelta verkon yli.

3.5 PHP

PHP (PHP: Hypertext Preprocessor) on avoimen lähdekoodin komentosarjakieli. Puhekielessä käytetään usein sanaa skriptauskieli. PHP:n skriptejä suoritetaan palvelinpuolella, jota hyödynnetäänkin usein perinteisten HTML-sivujen kanssa erilaisten toiminnallisuuksien tekemisessä. Älypeilissä sitä tarvittiin web-sivulla, josta tarkastetaan käyttäjien kirjautumiset. Sen avulla tehtiin painike, joka tekee MySQL-kyselyn, jonka avulla nähdään, kuka on kirjautunut minäkin päivänä. (Laaksonen, 2011.)

4 ASENTAMINEN

Älypeilin käyttöä varten pitää ensin asentaa tarvittavat ohjelmistot. Ensin asennetaan käyttöjärjestelmä, ja tehdään siihen tarvittavat asetukset, määritykset ja muutokset. Sen jälkeen asennetaan ohjelmat, kuten MagicMirror[2], Apache2, MySQL ja muita.

4.1 Raspbianin asennus

Ensimmäisenä vaiheena Raspberry Pin käyttämiseen on Raspbian Jessien asennus. Käytin Windows 10:llä varustettua tietokonetta, jossa oli asennettuna Rufus-niminen levykuvakirjoitustyökalu. Työkalussa on paljon asetuksia, mutta ainoat joita pitää muuttaa ovat laite, johon levykuva kirjoitetaan ja mikä levykuva halutaan kirjoittaa. Device-pudotusvalikosta valitaan haluttu microSD-kortti. Tämän jälkeen painetaan kuvaketta, jossa on CD-levy ja levyasema, joka avaa tiedoston valitsimen. Siitä valitaan Raspian Jessien levykuva. Sen jälkeen painetaan Start-painiketta ja kuitataan vielä ohjelman antama varoitus levyn tietojen menettämisestä. Kun levykuva on kirjoitettu onnistuneesti kortille, voidaan painaa Close-painiketta, jolloin ohjelma sulkeutuu. Sen jälkeen poistetaan microSD-kortti tietokoneesta turvallisesti, ja kortti on valmis asennettavaksi Raspberry Pihin.

Kun Raspberry Pi käynnistetään ensimmäisen kerran uuteen käyttöjärjestelmään, näkyviin tulee työpöytä. Raspbianin tehtäväpalkki sijaitsee ruudun yläreunassa, josta löytyy muun muassa ohjelmavalikko sekä tilarivi. Painamalla tilarivillä olevaa verkkoyhteyden kuvaketta aukeaa valikko avoimista langattomista verkoista. Klikkaamalla jotain verkkoa järjestelmä pyytää syöttämään salausavaimen, jos verkko on suojattu. Kun oikea salausavain on syötetty, verkkoyhteyden kuvake muuttuu näyttämään ensin yhdistämisanimaatiota, ja sen jälkeen signaalin vahvuutta.

Kun verkkoyhteys on saatu, kannattaa suorittaa seuraavat komennot:

```
sudo apt-get update
sudo apt-get upgrade
```

Sudo tarkoittaa, että komento suoritetaan pääkäyttäjänä (Super User). Sudon käyttö edellyttää salasanan syöttämistä. Apt-get-komennolla käytetään pakettienhallintaoh-

jelma APT:ta (Advanced Package Tool), joka on vastuussa pakettien, esimerkiksi ohjelmien, asennuksista. Update-parametri päivittää pakettilistan, eli tiedon pakettilähteiden sisällöstä. Upgrade-parametri päivittää vastaavasti järjestelmään asennetut paketit. Näiden komentojen avulla siis varmistetaan, että asennetut ohjelmat ovat uusimpia versioita. Komennot kirjoitetaan komentokehotteeseen (Terminal), joka avataan näppäinyhdistelmällä Ctrl+Alt+T. (Hayward, 2012.)

Soveltuakseen älypeilikäyttöön Raspbianin asetuksiin joudutaan tekemään joitain muutoksia. Yksi tärkeimmistä on ruudun tyhjentämisen (screen blanking) poistaminen käytöstä. Sen ideana on tyhjentää kuva, kun laite on ollut käyttämättä tietyn ajan, jolloin ruudussa näkyy pelkkää mustaa näytön taustavalon ollessa yhä aktiivisena. Se on käytössä näytön palamisen (screen burn-in) ja kuvan pysymisen (image persistence) estämiseksi. Näillä termeillä tarkoitetaan tilannetta, jossa näyttöteknologiasta riippuen näyttöruutuun jää joko pysyviä tai väliaikaisia jälkikuvia, jotka johtuvat liian pitkästä yhtäjaksoisesta liikkumattomasta kuvasta. Uudemmissa näytöissä tämä on kuitenkin hyvin vähäistä ja väliaikaista. Ruudun tyhjentäminen tapahtuu oletuksena kymmenen minuutin käyttämättömyyden jälkeen, ja kuva palaa takaisin siirtämällä hiirtä ja painamalla mitä tahansa näppäintä näppäimistöstä. Älypeilissä ei kuitenkaan käytetä kumpaakaan, joten ruudun tyhjentäminen poistetaan käytöstä seuraavasti: Ensin kirjoitetaan komentokehotteeseen:

```
sudo nano /etc/lightdm/lightdm.conf
```

Kyseinen tekstitiedosto on ikkunanhallintaohjelman asetustiedosto. Sieltä haetaan kohta, jossa lukee:

```
[SeatDefault]
```

Sen alle lisätään rivi:

```
xserver-command=X -s 0 dpms
```

Poistumiseen painetaan Ctrl+X, sitten Y, Enter ja tiedosto on tallennettu. Kun Raspberry Pi käynnistetään seuraavan kerran, ruutua ei enää tyhjennetä. (Kyrnin, 2017.)

4.2 MagicMirror²:n asennus

Tämän jälkeen voidaan asentaa MagicMirror²-alusta. Se asennetaan GitHub-palvelun kautta joko lataamalla .zip-tiedosto, kloonaamalla repositorio tai suorittamalla komento:

```
bash -c "$(curl -sL
https://raw.githubusercontent.com/MichMich/MagicMirror/master/installers/raspberry.sh)"
```

Tämä komento ajaa bash-skriptin, josta löytyvät tarvittavat tiedot MagicMirror² asentamiseen. Bash-skripti on joukko komentorivillä suoritettavia komentoja, jotka suoritetaan peräkkäin. Komennon suorittamisen aikana asennuksen vaiheita voidaan seurata komentokehotteesta rivi riviltä. Jos asennuksen aikana ilmenee ongelmia, asennus keskeytyy. Onnistuneen asennuksen jälkeen komentokehote palaa perustilaansa odottamaan käyttäjän syötettä.

Asennuksen jälkeen ohjelma voidaan käynnistää ensimmäisen kerran. MagicMirror²:n mukana tulee config.js-asetustiedoston sample- eli esiasetusversio, jossa on määritettynä muutaman vakiomoduulin konfiguraatiot. Se löytyy config-kansiosta nimellä config.js.sample. Nimeämällä se config.js:ksi saadaan hyväksyttävä (valid) konfiguraatiotiedosto, jolloin MagicMirror² voidaan käynnistää. Jos MagicMirror² on asennettu käyttämällä bash-skriptiä, uudelleennimeäminen tapahtuu automaattisesti. Käynnistäminen tapahtuu suorittamalla seuraavat komennot komentokehotteessa.

```
cd MagicMirror
npm start
```

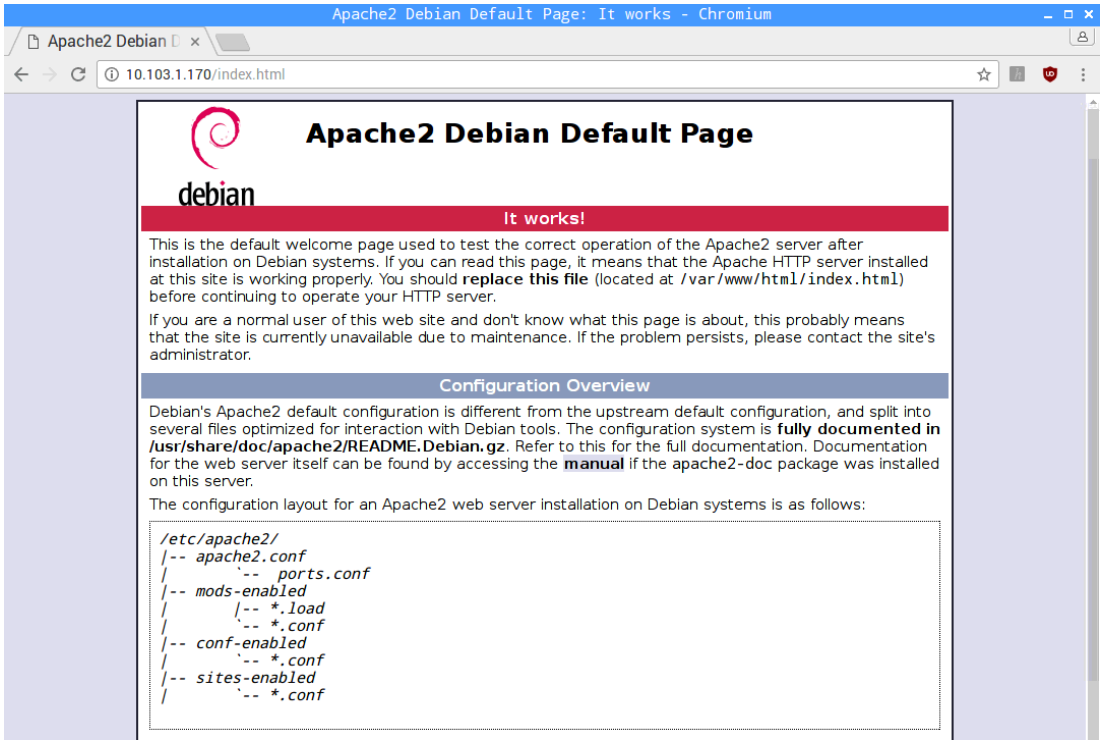
Ensimmäisellä komennolla (cd = change directory) navigoidaan MagicMirror-kansioon ja toisella käynnistetään ohjelma. Käynnistyskomennon jälkeen komentokehotteeseen alkaa ilmestyä ohjelmiston kertomia lokitietoja käynnistymisen eri vaiheista, kuten alustan ja moduulien lataamisen onnistumisesta. Hetken kuluttua MagicMirror²:n kuva avautuu koko ruudun kokoiseksi, ja voimme nähdä ruudulla muutamia moduuleita.

4.3 Web-palvelimen asennus

Web-palvelinta asennettaessa asennettiin itse palvelin sekä siihen lisättävät lisäosat. Web-palvelimeksi valittiin Apache sen helppouden, monipuolisuuden sekä ilmaisuuden vuoksi. Apachen asentaminen tapahtuu suorittamalla seuraava komento komentokehotteessa:

```
sudo apt-get install apache2 -y
```

Kun asennus on suoritettu, voidaan Apachen toimivuus testata navigoimalla Internet-selaimella joko osoitteeseen http://localhost/ tai laitteen omaan IP-osoitteeseen. Jos kaikki toimii kuten pitääkin, näkyviin tulee web-sivu (Kuva 2).



Kuva 3. Apachen mukana tuleva index.html kansiosta /var/www/html.

Apachen asentamisen jälkeen voidaan asentaa PHP. Sen asentaminen tapahtuu komennolla:

```
sudo apt-get install php libapache2-mod-php -y
```

Kun asennus on valmis, aikaisemmin mainittu index.html pitää vaihtaa index.php:ksi. Tämä on helppo tehdä komentoriviltä komennoin:

```
sudo rm index.html
sudo geany index.php
```

Ensimmäinen komento poistaa index.html-tiedoston ja toinen avaa Geany-tekstieditorin tiedostolle index.php. Tallentamalla se kansioon /var/www/html se on valmiina PHP-ohjelmointia varten.

Osaksi web-palvelinta tarvitaan myös MySQL-tietokantajärjestelmä. Sen asentaminen tapahtuu syöttämällä seuraava komento komentokehotteeseen:

```
sudo apt-get install mysql-server --fix-missing
```

Hetken kuluttua aukeaa ”Configuring mysql-server-5.5”-ikkuna, jossa kehote pyytää syöttämään salasanan pääkäyttäjälle. Salasana pyydetään vielä varmistamaan, jonka jälkeen asennus on valmis. Toinen tietokantaan liittyvä asennus on PHP-lisäosa mysqli. Sitä käytetään MySQL-tietokannan käsittelemiseksi PHP-sivulta. Sen asentaminen tapahtuu komennolla:

```
sudo apt-get install mysqli
```

Tämän jälkeen tarvittavat web-palvelimen osat on asennettu. Seuraavassa luvussa perehdytään niiden konfigurointiin.

5 OHJELMA

Tässä luvussa raportoidaan käyttöönotettu älypeili ja siihen kehitetyt osat. Luvussa kerrotaan myös mihin tarkoitukseen mikäkin osa kehitettiin ja miten ne toimivat älyasumisen tukena. Joidenkin kehitettyjen ominaisuuksien testaamista varten käytettiin testiympäristöjä varsinaisen käyttöympäristön keskeneräisyyden vuoksi. Myös näiden ympäristöjen kehitystä käydään läpi.

5.1 MagicMirror^2-moduulien asennus ja konfigurointi

Ensimmäisenä vaiheena älypeilin käyttöönottoa varten vakiomoduulit konfiguroitiin toimimaan halutulla tavalla. Sen jälkeen asennettiin moduuleja, jotka antaisivat hyödyllistä tietoa käyttäjille, sekä moduuleja, jotka ovat olennaisia peilin toiminnan kannalta, kuten kasvontunnistuksen moduuli.

5.1.1 Vakiomoduulien konfigurointi

MagicMirror^2:en mukana tulee kahdeksan moduulia: kello (Clock), kalenteri (Calendar), säätiedote (Current Weather), sääennuste (Weather Forecast), uutisotsikot (News Feed), kohteliaisuudet (Compliments), ”Hei, maailma” (Hello World) ja hälytys (Alert). Näistä moduuleista pois jätettiin kohteliaisuudet ja ”Hei, maailma” niiden tuntuessa tarpeettomilta kyseisessä sovelluskohteessa. Kalenterimoduuli vaihdettiin toiminnallisuudeltaan samanlaiseen, mutta visuaaliselta esitykseltä erilaiseen MMM-MyCalendar-moduuliin. Jäljelle jääneistä vakiomoduuleista ainoa joka ei vaatinut mitään muutoksia oli kellomoduuli. Se näytti kellonajan halutulla tavalla heti alusta asti.

Moduulien konfiguraation muuttaminen tapahtuu config.js-tiedoston kautta, jossa määritellään moduulien nimet, niiden sijainnit ruudussa sekä itse asetukset. Config.js-tiedostossa määritellään myös muita asetuksia älypeilille. Moduulien asetukset sijaitsevat kohdassa:

```
modules: [
   {
      module: "testimoduuli",    //moduulin nimi
      position: "top_right",    //moduulin sijainti ruudulla
      config: {
         //moduulin asetukset
         }
   },
```

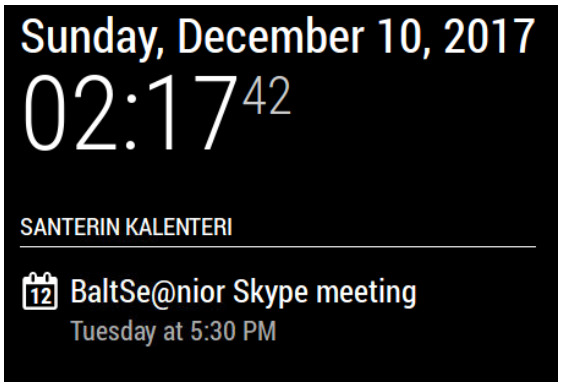
Moduulin nimi ja sijainti annetaan aina samalla tavalla moduulista riippumatta, ellei moduuli ole piilotettu moduuli, joka toimii näyttämättä mitään ruudussa. Moduulikohtaiset konfiguraatiot riippuvat moduulin ominaisuuksista. Jokaisen moduulin vakioasetukset löytyvät moduulin omasta asetustiedostosta, joten jos config.js-tiedoston moduuliasetuksen joku kohta jätetään kirjoittamatta, sen arvoksi tulee automaattisesti vakioasetus.

Kalenterimoduuli on yksi olennaisimmista osista älypeilin toimintaa, sillä se toimii tehokkaasti käyttäjän henkilökohtaisena apuvälineenä. Alkutilanteessa moduuli toimi käyttämällä iCal-kalenteritiedostoa, josta se luki kalenterimerkinnät moduuliin ja esitti ne sitten älypeilillä. Käytettävyyden kannalta haluttiin kuitenkin kalenterimerkinnät jonkun webkalenteripalvelun kautta, jolloin samat kalenterimerkinnät voidaan näyttää usealla laitteella peilin ollessa yksi niistä. Tähän tarkoitukseen valittiin Google Calendar.

Google Calendarin käyttämiseksi pitää tehdä muutama toimenpide, jotta Google Calendarin antamat kalenterimerkinnät saadaan kalenterimoduuliin. Ensin tehtiin uusi kalenteri, johon tullaan lisäämään peilillä näkyviin haluttavat merkinnät. Kun kalenteri on luotu, mennään sen ”Asetukset ja jakaminen”-sivulle. Sieltä löytyvä ”Salainen osoite iCal-muodossa” on se, mitä käytetään älypeilin config.js-tiedostossa. Sen avulla voidaan käyttää kalenteria erilaisissa palveluissa, kuten älypeilin moduulissa, ilman, että kalenterista itsessään tehdään julkista. Kalenterin salainen iCal-osoite kirjoitetaan seuraavasti config.js-tiedostoon:

```
{
                module: "MMM-MyCalendar",       //Moduulin nimi
                header: "Santerin kalenteri",  //Kalenterin otsikko
                position: "top_left",          //Sijainti
                classes: 'default everyone',  //Kenelle kalenteri
näkyy
                config: {
                        timeFormat: "fi",
                        calendars: [
                                {
                                        symbol: "calendar-check-o ",
                                        url:
"https://calendar.google.com/calendar/ical/kalenterinSalainen/Osoite
/basic.ics"
                                },
                        ],
                        maximumEntries: 6,
                        maxTitleLength: 35,
                        fetchInterval: 30000
                }
        },
```

Tällaisella konfiguraatiolla ainoa näkyvä kalenteri on äsken lisätty kalenteri. Kohta ”calendars” on tyypiltään taulukko, johon voitaisiin lisätä useampia kalentereja näkymään saman otsikon alla. Tällä tavoin voidaan yhdistää useasta palvelusta käytettäviä kalentereja yhteen näkymään.



Kuva 4. Kellonaika- ja kalenterimoduulit.

Kalenteri näkyy kellon alapuolella johtuen niiden ”position”-arvosta (Kuva 3). Molemmat ovat määritelty ”top_left”-alueelle, jolloin niiden paikka määräytyy config.js-tiedoston järjestyksen mukaan. Kellomoduuli on määritelty ennen kalenteria listassa, jolloin se sillä on suurempi prioriteetti saada oma paikkansa.

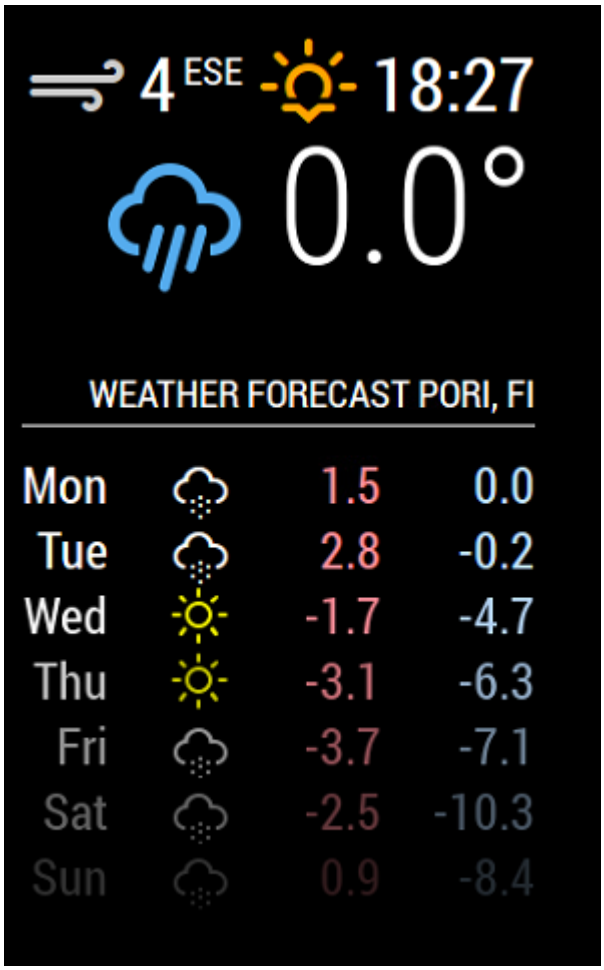
Säätiedote- ja sääennustemoduulit vaativat toimiakseen API- eli ohjelmointirajapinta-avaimen, jonka avulla säätiedot saadaan OpenWeatherMapin palvelimelta. Avaimen saaminen edellyttää rekisteröitymistä palveluun ja sen avulla käyttäjä tunnistetaan. API-avain on pitkä merkkijono, jonka palvelin antaa pyyntöä vastaan. API-

avaimia käytetään usein tiedon hakemiseen erilaisten ohjelmien kautta, kuten tässä tapauksessa MagicMirror[2]:n moduulin läpi. Tällaisella avaimella ei kuitenkaan pysty kirjautumaan itse palveluun, jolloin käyttäjän tunnus pysyy turvassa, vaikka joku saisikin kopioitua API-avaimen ja käytettyä sitä omaan tarkoitukseensa. API-avaimien hyötynä on myös mahdollisten bottihakujen aiheuttama palvelimen ylikuormitus, sillä API-avaimilla on yleensä jokin käyttörajoitus, jolla voidaan estää liian monen kyselyn tekeminen tiettynä aikavälinä. (De 2017, 113.)

Säätiedotteen ja sääennusteen moduulit käyttävät sijainnin valitsemisessa kaupungin ja maan nimeä tai vaihtoehtoisesti paikkatunnistetta (locationID). Paikkatunnisteen syöttäminen config.js-tiedostoon yliajaa kaupungin ja maan avulla valitun säätiedotteen. Lista tunnisteista löytyy OpenWeatherMapin sivulta http://openweathermap.org/help/city_list.txt. Sivulta löytyvä lista kaupungeista on hyvin pitkä, joten oikea kaupunki on helpointa löytää käyttämällä näppäinyhdistelmää Ctrl+F ja kirjoittamalla sitten halutun paikan nimen. Älypeiliä kehitettiin Porissa, joten hain listalta Pori, ja löysin sitä vastaavan paikkatunnisteen 640999. Kyseinen tunniste kirjoitetaan konfiguraatioon seuraavasti:

```
{
    module: "currentweather",
    position: "top_right",
    classes: 'default everyone',
    config: {
        location: "Pori",
        locationID: "640999",
        appid: "c2a57cb31ee2c443d660d14155302185"
    }
},
```

Yllä olevassa koodissa konfiguroidaan säätiedotemoduuli näkymään oikeassa yläreunassa. Sääennustemoduulia varten module-kohtaan kirjoitettaisiin ”weatherforecast” muun koodin pysyessä samanlaisena. Tällöin säätiedote ja sääennuste näkyisivät allekkain. Koodinpätkässä näkyvä API-avain ei ole työssä käytetty avain sen jakamisen aiheuttamien ongelmien takia. Kuka tahansa voisi käyttää avainta, jolloin avaimen käyttörajoitus tulisi nopeasti vastaan ja älypeili saattaisi jäädä ilman säätietoja. Kuvassa 4 näkyy säätietojen ja sääennusteen graafinen esitys.



Kuva 5. Säätietojen ja sääennusteen moduulit.

Uutisotsikot-moduuli tarvitsee toimiakseen yhden tai useamman uutissivuston RSS-syötteen. Jokaista RSS-syötettä varten annetaan oma otsikko, joten käyttäjän on helppo tunnistaa, mikä uutisotsikko on miltäkin sivustolta. Uutisotsikot määritellään seuraavasti:

```
{
        module: "newsfeed",
        position: "bottom_bar",
        config: {
                feeds: [
                        {
                                title: "Satakunnan Kansa
                                url:
"https://www.satakunnankansa.fi/?feed=uutiset&o=Satakunnan%20uutiset
&k=0&ma=0&c=3"
                        },
                        {
                                title: "Helsingin Sanomat",
                                url: "http://www.hs.fi/uutiset/rss/"
                        }
                ],
                showSourceTitle: true,
```

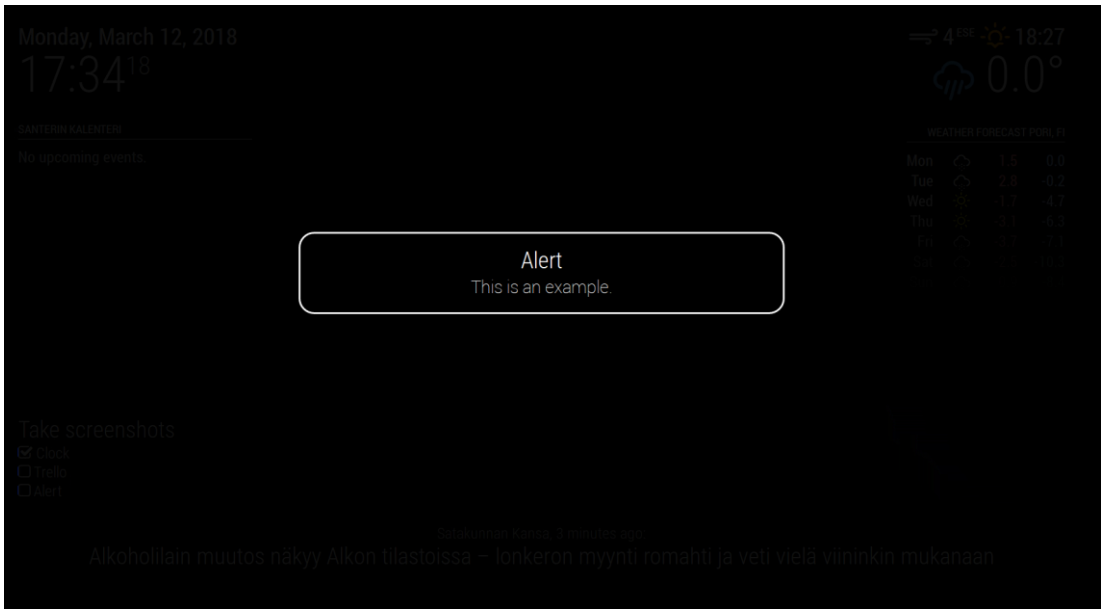
```
                showPublishDate: true
        }
},
```

Oheisessa esimerkissä uutisotsikot konfiguroitiin näyttämään sekä Satakunnan Kansan että Helsingin Sanomien tuoreimmat uutiset. ”Feeds” on tyypiltään taulukko, jonka sisälle kirjoitetaan kaikki halutut RSS-syötteet ja niiden otsikot. Feeds-taulukosta löytyvien RSS-syötteiden uutisotsikot näkyvät ruudulla satunnaisessa järjestyksessä. Kuvassa 5 näkyy uutisotsikoiden graafinen esitys.

Kuva 6. Uutisotsikon moduuli.

Hälytysmoduulin (Alert) tarkoituksena on kiinnittää käyttäjän huomio, kun käyttäjää halutaan tiedottaa jostakin. Se sisältää kaksi tilaa: ilmoitus (Notification) ja hälytys (Alert). Ilmoitus on ruudun yläreunaan ilmestyvä pieni valkoinen tekstilaatikko (Kuva 6). Se toimii saamalla käskyn joltain toiselta moduulilta, ja näyttämällä sitten halutun tiedon. Hälytys toimii samalla tavalla, mutta sen visuaalinen esitys on erilainen; hälytyksessä kuva tummenee, jonka jälkeen keskelle ilmestyy tekstilaatikko (Kuva 7). Kuvan tummentaminen auttaa hyvin huomion kiinnittämisessä hälytykseen.

Kuva 7. Alert-moduulin ilmoitus. Lähetetty kauko-ohjauksella,



Kuva 8. Alert-moduulin hälytys. Lähetetty kauko-ohjauksella.

5.1.2 Ladattavat moduulit

MagicMirror[2]:ta varten kehitettyjä moduuleita on ladattavissa sivustolla modules.magicmirror.builders. Moduulit ovat eri kehittäjien tekemiä ja ne ovat lajiteltu kategorioittain. Tässä työssä ladattavista moduuleista käyttöön otettiin MMM-Facial-Recognition, MMM-Facial-Recognition-Tools, MMM-Remote-Control ja MMM-Trello. Yleisin tapa asentaa moduuleja on kloonata repositorio ja sitten asentaa moduuli siitä. Se tapahtuu seuraavilla komennoilla:

```
git clone repositorionOsoite.git # repositorion osoite, joka halutaan kloonata
cd repositorionNimi # äsken kloonatun repositorion kansio
npm install # asennuskomento, joka asentaa riippuvuudet
```

Asentaessa uusia moduuleita on tärkeää, että repositoriota kloonatessa ollaan oikeassa kansiossa, joka on MagicMirror/modules. Jos näin ei ole, kloonatut tiedostot sijoittuvat väärään paikkaan eivätkä toimi osana MagicMirror[2]:ta. Kun uuden moduulin asennus on suoritettu, se otetaan käyttöön samalla tavalla kuin muutkin moduulit, eli lisäämällä se config/config.js-tiedostoon.

MMM-Facial-Recognition on kasvontunnistukseen tehty moduuli, jonka on kehittänyt Paul-Vincent Roll. Sen päätoimintona on kasvojentunnistuksella toimiva profiilijärjestelmä, jota hyödynnetään älypeilin muokkaamisessa eri käyttäjien mukaiseksi. Moduulia varten tarvitaan referenssikuvia, joihin käyttäjän kasvoja verrataan. Näiden

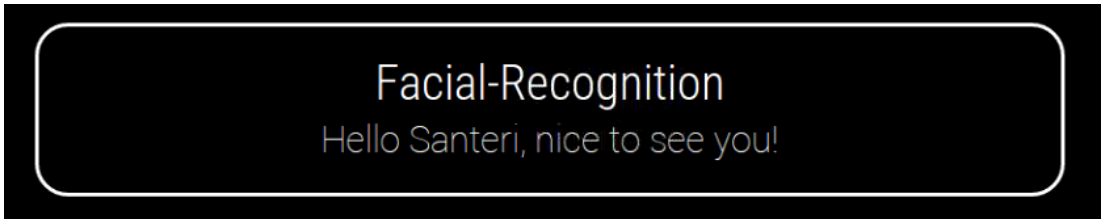
ottaminen tapahtuu MMM-Facial-Recognition-Tools-työkalun avulla. Ensin navigoidaan työkalun kansioon. Sitten suoritetaan komento:

```
python capture.py
```

Komento suorittaa Pythonin kautta kasvontunnistuksen referenssikuvienottotyökalun. Työkalu pyytää valitsemaan käytettävän algoritmin sekä kuvattavan henkilön nimen. Seuraamalla ruudussa näkyviä ohjeita saadaan otettua kuvat, ja painamalla Ctrl+C-näppäinyhdistelmää voidaan lopettaa kuvaus. Kuvauksen loputtua työkalu generoi training.xml-tiedoston, joka sisältää kuvattujen henkilöiden referenssitiedot, johon kirjautuvia käyttäjiä verrataan. Kyseinen xml-tiedosto pitää vielä kopioida kansioon modules/MMM-Facial-Recognition/, jotta järjestelmä löytäisi referenssikuvien tiedot. Lopuksi lisätään vielä config/config.js-tiedostoon seuraavat tiedot:

```
{
    module: "MMM-Facial-Recognition",
    config: {
        // Array with usernames (copy and paste from training
script)
        users: ['Santeri', 'Testihenkilö'],
        }
},
```

Kasvojentunnistusmoduuli ei siis tarvitse paikkaa peilissä, koska se ei näytä informaatiota itse. Henkilöiden nimet, joista on otettu referenssikuvat, kirjoitetaan users-taulukkoon. Muihin käytössä oleviin moduuleihin tulee lisätä tämän jälkeen lisätä ”classes”-kohta, jossa määritellään kenelle näkyy mikäkin moduuli. Default tarkoittaa, että moduuli näkyy, vaikka ei olla kirjauduttu. Everyone taas tarkoittaa, että moduuli näkyy jokaiselle käyttäjälle. Nyt kun peili käynnistetään ja asetutaan kameran eteen, saadaan peililtä tervehdys alert-moduulin kautta (Kuva 8).



Kuva 9. Tervehdysteksti onnistuneesta kirjautumisesta.

MMM-Remote-Control, jonka on kehittänyt GitHub-käyttäjä Jopyth, on tarkoitettu peilin etähallintaan Internet-selaimen kautta. Sitä voidaan käyttää moduulien näyttämiseen ja piilottamiseen, ilmoitusten ja hälytysten lähettämiseen, config.js-tiedoston muokkaamiseen sekä näytön ja Raspberry Pin sammuttamiseen. Moduuli lisätään config.js tiedostoon seuraavasti:

```
{
    module: 'MMM-Remote-Control'
    // uncomment the following line to show the URL of the remote
control on the mirror
    // , position: 'bottom_left'
    // you can hide this module afterwards from the remote control
itself
},
```

Peilin osoite saadaan siis näkyviin poistamalla kommenttimerkintä position-kohdasta. Kauko-ohjauksen osoite on muotoa iposoite:portti/remote.html.

Laitteiden IP-osoitteet, joilta kauko-ohjausta halutaan käyttää, pitää määritellä config.js-tiedoston alkuosassa ipWhiteList-taulukkoon. Jos määrittelemättömältä laitteelta yritetään ottaa yhteyttä, saadaan ilmoitus web-sivulla, jossa pyydetään tarkastamaan config.js-tiedosto. Jos laite on määritelty, saadaan auki sivu, josta voidaan valita haluttu toiminto. Kuvassa 9 on näkyvissä ympäristö Google Chrome-selaimessa.

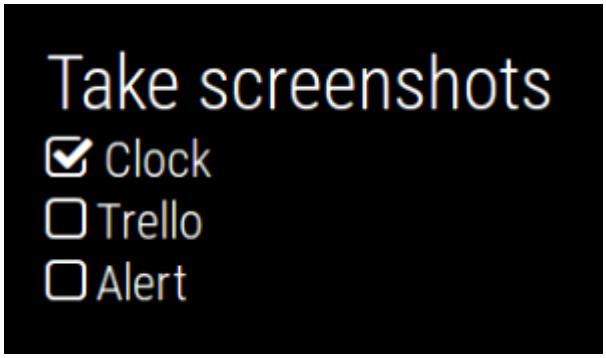
Kuva 10. Kauko-ohjausympäristö Internet-selaimessa.

MMM-Trello, jonka on kehittänyt GitHub-käyttäjä Jopyth, on tarkoitettu näyttämään Trello-kortteja. Trello on palvelu, joka on kehitetty toimimaan yhden tai useamman henkilön muistilistana, johon voidaan asettaa tavoitteita ja tehtäviä. MMM-Trellon määrittelemiseksi tarvitaan Trellosta kolme erilaista tunnistetta: API-avain, poletti (token) ja listan tunniste. API-avaimen ja poletin saa sivulta https://trello.com/app-

key. API-avain toimii samalla tavalla kuin säätietojen hakemisessa käytetty API-avain. Poletin avulla pyydetään käyttäjää hyväksymään hänen datansa käyttö. Sellaisen voi generoida itselleen samalta sivulta. Listan tunnisteen saaminen on tehty Trellossa vaikeimmaksi näistä kolmesta. Helpoin tapa saada se on avata haluttu taulu (board), vaihtaa osoiteriville taulun nimen ja viimeisen kauttaviivan tilalle .json, ja tarkastella siitä avautuvaa tiedostoa. Listan tunniste löytyy listan nimen jälkeen kohdasta ”id”. Nämä kolme tunnistetta syötetään siis config/config.js-tiedostoon seuraavasti:

```
{
    module: 'MMM-Trello',
    position: 'bottom_left',
    classes: 'default everyone',
    config: {
        api_key: "apiAvain",     //oma API-avain
        token: "poletti",        //generoitu poletti
        list: "listanTunniste"   //halutun listan tunniste
}
```

Näiden määrittelyjen avulla seuraavalla käynnistyskerralla peiliin pitäisi ilmestyä haluttu Trello-lista, joka näkyy kuvassa 10.



Kuva 11. MMM-Trello-moduulin näkymä.

5.1.3 Oma moduuli – Tiedon haku kotiautomaatiojärjestelmästä

Älypeiliä varten kehitettiin myös valmiiden moduulien lisäksi oma moduuli, jonka tarkoituksena oli hakea tietoa älykodin kotiautomaatiojärjestelmästä. Kyseisen moduulin kehitys täysin käyttövalmiiksi oli kuitenkin vaikeaa, koska kohdeasunnon järjestelmistä ei ollut vielä tarkkaa tietoa. Tämän vuoksi käytettiin itsetehtyä testiympäristöä, jonka avulla simulointiin oikeaa järjestelmää.

Moduulin kehitys aloitettiin kysymällä yhdeltä kohdeasunnon kotiautomaatioratkaisujen kehittäjältä, minkälaisessa muodossa järjestelmässä saataisiin tietoa ulos. Päädyimme ratkaisuun, joka perustuisi HTTP GET-metodiin.

GET-metodiin perustuvaa tiedonhakua varten asennettiin ensin json-server-niminen ohjelma, jonka avulla saadaan käyttöön REST (REpresentational State Transfer)-ympäristö. REST on joukko arkkitehtuurillisia periaatteita, jotka ovat suunnattu web-palveluille, jotka käyttävät järjestelmäresursseja (Rodriguez, 2008). Json-serverin asennus tapahtuu kirjoittamalla komentokehotteeseen komento:

```
npm install -g json-server
```

Kun asennus on valmis, tehdään ohjelman käyttöön json-tiedosto, joka sisältää käsiteltävää tietoa. Tässä tapauksessa json-tiedostoon laitettiin testiksi huoneen sisäilman lämpötila, koska se voisi olla yksi kotiautomaatiojärjestelmän mittaamista arvoista. Json-tiedosto näyttää siis tältä:

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "info": [
    { "id": 1, "title": "Temperature", "body": "21&#176;C" }
    ],
  "profile": { "name": "typicode" }
}
```

Lähde: https://github.com/typicode/json-server

Tämä json-tiedosto tallennetaan Pi-käyttäjän kotikansioon nimellä db.json. Sen jälkeen itse ohjelma voidaan käynnistää komennolla:

```
json-server –watch db.json
```

Nyt tiedot ovat valmiit käytettäväksi, joten seuraavaksi vuorossa oli moduulin kehitys, joka käyttäisi kyseisiä tietoja. Uuden moduulin luomista varten on kehitetty skripti, joka automatisoi moduulin kehitysprosessin, ja tekee tarvittavat tiedosto- ja kansiopohjat uutta moduulia varten. Sen on kehittänyt Rodrigo Ramírez Norambuena, ja skripti voidaan suorittaa komennolla:

```
bash -c "$(curl -sL https://raw.githubusercontent.com/roramirez/MagicMirror-Module-Template/master/create_module.sh)"
```

Skriptin ajamisen aikana käyttäjältä kysytään moduulin nimeä, joksi kirjoitin Home-Automation. Kun skripti on suoritettu onnistuneesti, voidaan tiedostoja muokata käsin. Ne löytyvät hakemistosta MagicMirror/modules/Home-Automation.

Home-Automation-moduulin toiminta perustuu sen JavaScript-tiedostoon, josta löytyvät tarvittavat funktiot tiedon hakemiseen ja esittämiseen. Tieto haetaan getData-funktiolla, joka on seuraavanlainen:

```
    getData: function() {
        var self = this;

        var urlApi = "http://localhost:3000/info/1";
        var retry = true;

        var dataRequest = new XMLHttpRequest();
        dataRequest.open("GET", urlApi, true);
        dataRequest.onreadystatechange = function() {
            console.log(this.readyState);
            if (this.readyState === 4) {
                console.log(this.status);
                if (this.status === 200) {
                    self.processData(JSON.parse(this.response));
                } else if (this.status === 401) {
                    self.updateDom(self.config.animationSpeed);
                    Log.error(self.name, this.status);
                    retry = false;
                } else {
                    Log.error(self.name, "Dataa ei voitu ladata.");
                }
                if (retry) {
                    self.scheduleUpdate((self.loaded) ? -1 :
self.config.retryDelay);
                }
            }
        };
        dataRequest.send();
    },
```

Muuttuja urlApi sisältää osoitteen, josta tieto haetaan ja johon lähetetään GET-pyyntö (ks. dataRequest). Myöhemmin funktiossa getDom käsitellään samaa tietoa, ja saadaan se muotoon, joka voidaan näyttää älypeilissä:

```
    getDom: function() {
        var self = this;

        // create element wrapper for show into the module
        var wrapper = document.createElement("div");
        // If this.dataRequest is not empty
        if (this.dataRequest) {
            var wrapperDataRequest = document.createElement("div");
            wrapperDataRequest.innerHTML = this.dataRequest.body;

            var labelDataRequest = document.createElement("label");
            // Use translate function
            //             this id defined in translations files
            labelDataRequest.innerHTML =
this.translate("TEMPERATURE");
            wrapper.appendChild(labelDataRequest);
            wrapper.appendChild(wrapperDataRequest);
        }
```

```
        return wrapper;
    },
```

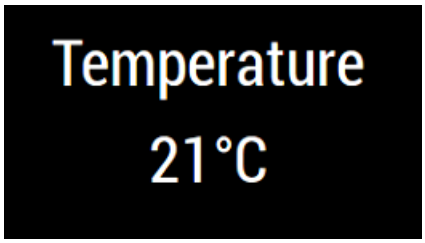
Seuraava vaihe on lisätä käännös tunnisteelle TEMPERATURE. Se lisätään tiedostoon translations/en.json:

```
{
  "TITLE": "Title",
  "UPDATE": "Update",
  "TEMPERATURE": "Temperature"
}
```

Käännöstiedostoja voi olla useampia, jos tuettuna on eri kieliä. Niistä MagicMirror[2] selvittää, mitä tekstiä ruudulla tulisi näkyä. Kotiautomaatiomoduulin toimintaa varten pitää vielä lisätä se config.js-tiedostoon seuraavasti:

```
        {
            module: "Home-Automation",
            position: "bottom_right",
            classes: 'default everyone',
            config: {
            }
        },
```

Tämän jälkeen peili voidaan käynnistää. Peilin pitäisi näyttää nyt lämpötila ruudulla, kuten kuvassa 11.



Kuva 12. Kotiautomaatiojärjestelmästä haetun lämpötilan esitys.

5.2 Käyttäjien kirjautumisten seurantajärjestelmä

Yhtenä tärkeänä asiana älypeiliä suunnitellessa pidettiin älykodissa asuvien henkilöiden seurantaa, jotta heidän tilastaan tiedettäisiin edes suuntaa antavia asioita. Tämän konseptin pohjalta saatiin idea, että peili tallentaisi muistiin, kun sitä on käyttänyt tietty käyttäjä. Idea todettiin toteuttamiskelpoiseksi, sillä tämän tyyppinen toiminta olisi suhteellisen helppo integroida älypeilin toimintaan.

Ensimmäinen toimenpide tällaisen toiminnon kehittämisessä oli määritellä, miten toiminto voidaan toteuttaa, mitä järjestelmiä tarvitaan, miten kirjautumista tarkkaillaan ja mistä älypeilin osasta kirjautumistieto voitaisiin tallentaa. Ensimmäiseksi ide-

aksi muodostui tallentaa jokaisesta kirjautumisesta uusi rivi tekstitiedostoon, jossa olisi käyttäjän nimi sekä kirjautumishetken päivämäärä. Se kuitenkin havaittiin huonoksi edes prototyyppikäyttöön sen ollessa huonosti skaalautuva ja toteutukseltaan ei juurikaan tietokantaa helpompi. Ainoana etuna oli asennettavien komponenttien pienempi määrä. Seuraavana ideana oli tietokanta. Tietokantaa ja sen käsittelyä varten tarvittiin muutama asennettava komponentti, joita ilmeni työn edetessä. Tietokantajärjestelmäksi valittiin MySQL, jo aiemmin, kappaleessa kolme mainituin perusteluin.

Edellisessä kappaleessa asennettiin MySQL, joten se on valmiina tietokannan luomista varten. Tietokanta käyttäjien kirjautumisia varten luodaan MySQL-hallintajärjestelmässä komentoriviä käyttäen. Ensin avataan MySQL-järjestelmä komennolla:

```
sudo mysql -u root -p
```

Komennon syöttämisen jälkeen ohjelma kysyy salasanaa. Kun se on syötetty onnistuneesti, päästään MySQL-ympäristöön. Uuden tietokannan luomiseksi käytetään komentoa:

```
mysql> CREATE DATABASE USERLOGINS;
```

MySQL-kehotteessa on tärkeää lopettaa kaikki komennot puolipisteeseen. Jos sitä ei käytetä, kehote odottaa, että komento jatkuu vielä enter-painikkeen painamisen jälkeenkin. Tietokannan luomisen jälkeen se pitää valita aktiiviseksi ja luoda siihen taulu. Taulua luodessa määritellään sen nimi ja siihen kuuluvien kenttien nimet sekä tyypit. Taulu luodaan seuraavasti:

```
mysql> CREATE TABLE users (ID int NOT NULL AUTO_INCREMENT, UserName
VARCHAR(255), Date DATE, PRIMARY KEY(ID));
```

Users-taulussa on siis kentät käyttäjän tunnusnumerolle (ID), nimelle, joka on maksimissaan 255 merkkiä pitkä merkkijono (VARCHAR), sekä päivämäärälle, joka on päivämäärämuodossa (DATE). Taulun avaimena (PRIMARY KEY) on tunnusnumero; joka rivillä on eri tunnusnumero automaattisen lisäyksen (AUTO_INCREMENT) ansiosta, jolloin rivit voidaan yksilöidä tämän kentän avulla.

Kun taulu on luotu, voidaan vielä tarkastaa, että sen kentät ovat oikein. Se tapahtuu kirjoittamalla komento:

```
mysql> DESCRIBE users;
```

Komento antaa kuvauksen kyseisen taulun kentistä ja niiden tyypeistä. Kuvassa 12 on esimerkki kuvauksesta.

```
mysql> describe users;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| ID       | int(11)      | NO   | PRI | NULL    | auto_increment |
| UserName | varchar(255) | YES  |     | NULL    |                |
| Date     | date         | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

Kuva 13. describe users-komennon tuloste.

Taulun kuvaus näyttää oikealta, joten MySQL-kehote voidaan sulkea, sillä taulun jatkokäsittely tapahtuu myöhemmin JavaScriptin yli. Kehote suljetaan komennolla:

```
mysql> exit
```

Kun käyttäjä astuu kameran eteen, kasvontunnistusmoduuli havaitsee käyttäjän kasvot, ja joko tunnistaa käyttäjän tai käsittelee häntä vieraana. Jos käyttäjä tunnistettiin, kasvontunnistusmoduuli lähettää tiedotemoduulille tiedon, minkä niminen käyttäjä astui peilin eteen. Tämä tapahtuu modules/MMM-Facial-Recognition/node_helper.js-tiedostosta löytyvässä koodin osassa:

```
if (message.hasOwnProperty('login')){
        console.log("[" + self.name + "] " + "User " +
self.config.users[message.login.user - 1] + " with confidence " +
message.login.confidence + " logged in.");
        self.sendSocketNotification('user', {action: "login", user:
message.login.user - 1, confidence: message.login.confidence});
}
```

Kyseisessä koodissa raportoidaan kasvontunnistuksen havaitsema henkilön nimi (self.config.users[message.login.user - 1]) ja tapahtuneen tunnistamisen varmuus (message.login.confidence) konsoli-ikkunaan. Tätä kohtaa on helppo käyttää kirjautumisen kirjaamisessa tietokantaan, koska koodia kutsutaan aina, kun käyttäjä kamera havaitsee käyttäjän.

MySQL-tietokantaa varten node_helper.js-tiedostoon lisätään tiedot tietokannasta, johon tallennetaan kirjautumiset. Lisätään siis seuraavat rivit:

```
var con = mysql.createConnection({
  host: "localhost",
  user: "root",
```

```
  password: "salasana",
  database: "USERLOGINS"
});
```

Oheisessa koodissa annetaan tiedot, joiden avulla tietokantaa päästään käsittelemään ja tekemään siihen muutoksia ja kyselyitä. Localhost tarkoittaa, että SQL-palvelin löytyy paikallisesta (local) osoitteesta. User- ja password-kohdat määrittävät käyttäjänimen ja salasanan joilla on käyttöoikeus kyseiseen tietokantaan. Database on tietokannan nimi, joka on annettu kyseiselle tietokannalle. Näiden tietojen lisäämisen jälkeen tarvitaan tarvittava tieto, joka voidaan syöttää tietokantaan, eli kirjautuvan käyttäjän nimi sekä päivämäärä. Se tapahtuu seuraavanlaisen koodin avulla node_helper.js-tiedostossa:

```
            Date.prototype.today = function () {
                return this.getFullYear() + "-" +
(((this.getMonth()+1) < 10)?"0":"") + (this.getMonth()+1) + "-" +
((this.getDate() < 10)?"0":"") + this.getDate();
            }

            var newDate = new Date();
            var datetime = newDate.today();
            var usersname = self.config.users[message.login.user -
1];
            var values = [
                        [usersname, datetime]
                        ];
            con.connect(function(err) {
                        if (err) throw err;
                        console.log("Connected!");
                        var sql = "INSERT INTO users (UserName, Date)
VALUES ?";
                        con.query(sql, [values], function (err, re-
sult) {
                                    if (err) throw err;
                                    console.log("Kirjattu");
                        });
```

Oheisessa koodissa tehdään siis ensin uusi konstruktori (Date.prototype.today), jonka avulla luodaan MySQL-yhteensopivasti muotoiltu päivämäärä. Sen jälkeen luodaan uusi päivämääräolio (newDate), jonka jälkeen sen avulla tehdään muuttuja (datetime). Kun päivämäärä on saatu muuttujaan, voidaan sen jälkeen ottaa kirjautuneen käyttäjän nimi. Se saadaan MMM-Facial-Recognitionin käyttämästä taulukosta (self.config.users[message.login.user]). Lukemalla siitä edellinen arvo saadaan äsken kirjautuneen käyttäjän nimi, joka luetaan muuttujaan usersname. Muuttuja sql on tietokantaan lähetettävä data. Se lähetetään käyttämällä query-komentoa, joka ottaa parametrikseen lähetettävän muuttujan. Lopuksi vielä annetaan komentoriville tuloste, jotta saadaan tarkasteltua, onnistuiko komennon suoritus.

Seuraavaksi pitää tehdä php-sivu, jonka kautta kirjautumisia tarkastellaan. Se tehdään index.php-sivusta, joka luotiin Apachen asentamisen jälkeen. Aluksi määritellään web-sivun graafinen osuus seuraavasti:

```
<h1>Magic Mirror usage</h1>
<form action="index.php" method="post">
    <label>Please enter a name:</label>
    <input name="UName" type="UserName" placeholder="Name:">
    <br>
    <br>
    <input type="submit" value="Enter">
</form>
```

Tällä tehdään siis web-sivulle tekstikenttä ja hakupainike, jolla kysely lähetetään. Sen jälkeen vuorossa on PHP-osa. Se aloitetaan seuraavasti:

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

$servername = "localhost:3306";
$username = "root";
$password = "salasana";
$dbname = 'USERLOGINS';

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

PHP-osan alku määritellään avaamalla PHP-osio <?php-merkinnällä. Sitten määritellään tietokanta samaan tapaan kuin aikaisemmassa node_helper.js-tiedostossa. Määrittelyn jälkeen avataan yhteys. Lopuksi voidaan lisätä itse tietokantaa käsittelevä osuus:

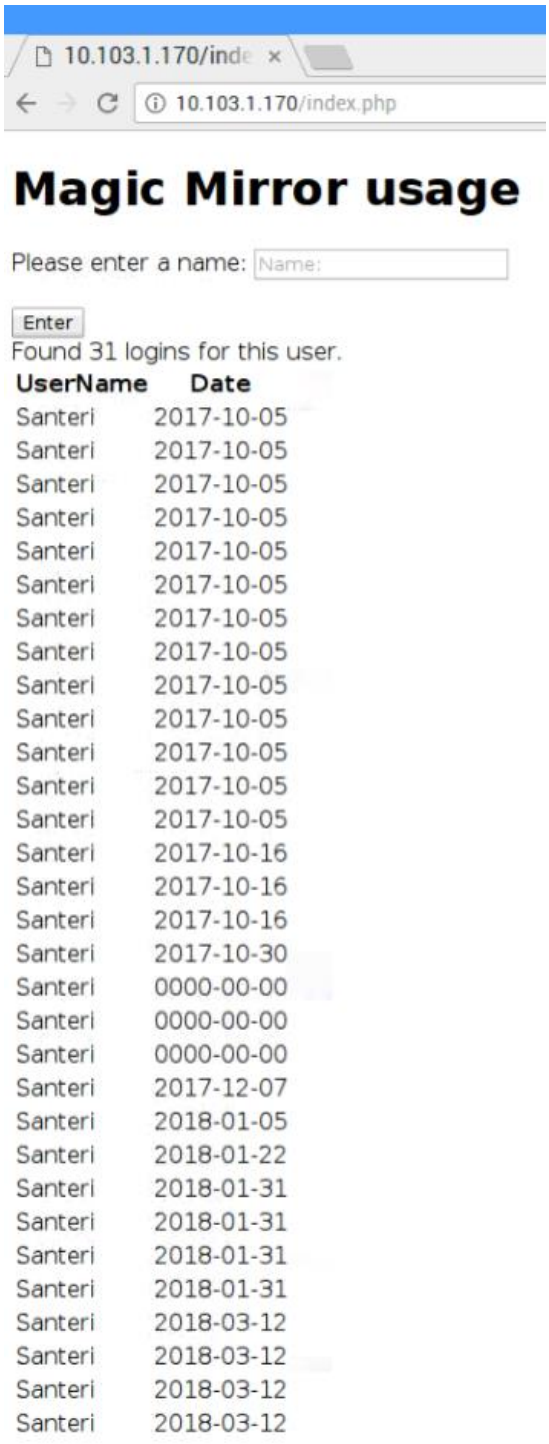
```
$UName = $_POST['UName'];

$sql = "SELECT * FROM users WHERE UserName LIKE '%$UName%'";
if(isset($_POST['UName']) != ""){
if($result = $conn->query($sql)){
    printf("Found %d logins for this user.", $result->num_rows);
    echo '<table class="table table-striped table-bordered table-
hover">';
        echo "<tr>
        <th>UserName</th>
        <th>Date</th>
        </tr>";
    while($row = $result->fetch_array(MYSQLI_ASSOC))
    {
        echo "<tr><td>";
        echo $row['UserName'];
        echo "</td><td>";
        //echo $row['ID'];
```

```
        //echo "</td><td>";
        echo $row['Date'];
        echo "</td><td>";
    }
    echo "</table>";
    $result->close();
}
}
```

Tässä skriptissä otetaan tekstikenttään kirjoitettu nimi POST-metodilla ja lähetetään sen perusteella mysqli-kysely tietokantaan. Saatujen kirjautumisten määrä tulostetaan sivulle ja näytetään taulukossa nimet ja päivämäärät.

Kun kaikki tarvittava on nyt konfiguroitu, voidaan taas käynnistää MagicMirror[2]. Kun alusta on täysin käynnistynyt, testataan kasvojentunnistuksen toiminta nostamalla kameramoduuli kasvojen korkeudelle, katsomalla sitä kohti ja odottamalla hetki. Kun kamera on saanut kaapattua kuvan, ruutuun tulee tervehdysteksti. Jos henkilö on järjestelmän tuntema, lukee tekstissä käyttäjän nimi. Jos ei, siinä lukee stranger eli muukalainen. Onnistuneen kirjautumisen jälkeen ruudulla näkyvät moduulit ovat ne, jotka on määritelty config.js-tiedostossa kyseiselle käyttäjälle sekä moduulit, jotka ovat määritelty kaikille käyttäjille (everyone). Nyt kun kirjautuminen on suoritettu, voidaan avata WWW-selain. Navigoidaan sillä Raspberry Pin IP-osoitteeseen, kirjoitetaan näkyvissä olevaan tekstikenttään kirjautuneen henkilön nimi ja painetaan Enter. Tämän jälkeen näemme kirjautumisen ajankohdan kyseiselle henkilölle.



Kuva 14. Kirjautumisten seurantajärjestelmän web-sivulla tehty haku nimllä 'Santeri'.

Jos haku onnistui ja näkyviin tuli päivämäärä kuten kuvassa 13, niin järjestelmä on silloin onnistuneesti määritetty. Nyt tietokantaan tulee merkintä aina, kun joku tunnetuista käyttäjistä astuu peilin eteen.

6 KEHITYSMAHDOLLISUUDET

Vaikka peilistä tulikin valmis prototyyppi sellaisenaan, siihen voitaisiin kuitenkin kehittää vielä monenlaisia toimintoja ja ominaisuuksia. Kehitysmahdollisuuksia tukee myös peilin alustan modulaarinen rakenne; asetustiedoston, käyttäjäprofiilien ja kauko-ohjauksen avulla peilillä voi olla käytössä ja näkyvissä juuri ne moduulit, joita tietty henkilö haluaa käyttää.

Yksi kehitysmahdollisuuksista on peilin pelillistäminen. Pelillistämisratkaisuna voisi olla esimerkiksi fyysisesti aktivoiva peli, joka pyytäisi tekemään lyhyen liikuntasuorituksen, kun peili tunnistaisi käyttäjän. Peili voisi ensin tarkistaa milloin viime suoritus on tehty muokkaamalla luotua tietokantaa pitämään niistä kirjaa. Fyysisen aktiivisuuden tarkkailu voitaisiin tehdä esimerkiksi pienellä Bluetooth-anturilla, joka voitaisiin yhdistää Raspberry Pi 3 Model B:ssä olevaan Bluetooth-piiriin.

Myös peilin laitteistoa itsessään voitaisiin parantaa ja kehittää. Vaikka Raspberry Pi 3 Model B on tarpeeksi tehokas MagicMirror^2-alustan ajamiseen, saattaisivat raskaat moduulit aiheuttaa ongelmia peilin toiminnalle. Tässä tapauksessa tehokkaampi prosessori ja suurempi keskusmuisti olisivat hyödyksi. Tarkoitukseen valittua televisiotakin voitaisiin parantaa. Kuten luvussa kaksi mainittiin, työssä käytetyn television maksimikirkkaus ei ole kovin suuri, joten kirkkaasti valaistu huone ei välttämättä sovellu peilin asennuspaikaksi. Televisiona kannattaisi käyttää suuremmalla kirkkausarvolla varustettua OLED-teknologiaan (Organic Light-Emitting Diode) perustuvaa mallia, sillä se toimisi paremmin yhdessä puoliläpäisevän peilin kanssa. OLED-näyttöpaneelit eivät valaise pikseleitä, jotka ovat mustia, joten peilin mustat kohdat näyttäisivät siis entistä peilaavimmilta ja teksti näkyisi selkeämmin läpi.

7 YHTEENVETO

Opinnäytetyönä älypeili oli mielenkiintoinen projekti, jossa oli mahdollista hyödyntää monenlaisia opintojen aikana opittuja taitoja ja syventää niitä entisestään. Työn aikana opin myös paljon uutta muun muassa tietokannoista ja PHP-ohjelmoinnista, sekä monen järjestelmän yhteistoiminnasta.

Työn toteutuksessa oli sekä helppoja että vaikeita osuuksia. MagicMirror^2:sta löytyi hyvät dokumentaatiot, jotka helpottivat huomattavasti peilin käyttöönottoa. Toisaalta oman moduulin ja kirjautumisten seurannan ja tietokannan tekemiseen kului paljon aikaa, sillä minulla ei ollut aiempaa kokemusta sellaisista. Lopputuloksena älypeili toteutti kuitenkin asetetut vaatimukset.

Älypeilistä on tulossa yksi SAMKissa esiteltävistä hyvinvointia edistävän teknologian tutkimusryhmän prototyypeistä. Kävimme esittelemässä sitä Tukholman huonekalu- ja valomessuilla yhdessä muiden BaltSe@niorissa kehitettyjen ICT-integroitujen ratkaisujen kanssa. Messuilla rakensimme malliksi makuuhuoneen, jossa prototyypit olivat osana kokonaisuutta. Ideoimme messujen aikana, kuinka kaikkien prototyypit voisivat toimia yhtenä kokonaisuutena loppusyksystä Ulvilan MeWet-kodissa järjestettävässä kenttätestauksessa.

Älypeilin jatkokehitystä varten tarkoituksena on tarjota moduulien kehittämistä opiskelijoiden projektityöksi esimerkiksi aiemmin mainitun pelillistämisen osalta. Myös mahdollista yritysyhteistyötä on kaavailtu.

LÄHTEET

Benchoff, B. 2016. Introducing the Raspberry Pi 3. Hackaday 28.2.2016. Viitattu 3.1.2018. https://hackaday.com/2016/02/28/introducing-the-raspberry-pi-3/

Cawley, C. 2017. 10 Operating Systems You Can Run With Raspberry Pi. MakeUseOf 14.8.2017. Viitattu 4.1.2018. https://www.makeuseof.com/tag/7-operating-systems-you-can-run-with-raspberry-pi/

Chacos, B. 2016. Raspberry Pi 3 review: The revolutionary $35 mini-PC cures its biggest headaches. PCWorld 19.4.2016. Viitattu 31.12.2017. https://www.pcworld.com/article/3057888/computers/raspberry-pi-3-review-the-revolutionary-35-mini-pc-cures-its-biggest-headaches.html

De, B. 2017. API Management: An Architect's Guide to Developing and Managing APIs for Your Organization. Apress 17.3.2017.

Gibbs, M. 2016. New Raspberry Pi Model B Launched. Network World 29.2.2016. Viitattu 3.1.2018. https://www.networkworld.com/article/3038786/internet-of-things/new-raspberry-pi-3-model-b-launched.html

Hayward, D. 2012. How to get started with the Raspberry Pi. CNet 29.11.2012. Viitattu 5.3.2018. https://www.cnet.com/how-to/how-to-get-started-with-the-raspberry-pi/

Heath, N. 2016. Raspberry Pi 3: The inside story from the new $35 computer's creator. TechRepublic 28.2.2016. Viitattu 31.12.2017. https://www.techrepublic.com/article/raspberry-pi-3-the-inside-story-from-the-new-35-computers-creator/

Kyrnin, M. 2017. LCD Image Persistence. LifeWire 7.11.2017. Viitattu 5.3.2018. https://www.lifewire.com/lcd-image-persistence-833037

Laaksonen, A. 2009. MySQL ja PHP: Osa 1 – Johdanto. Ohjelmointiputka 2009. Viitattu 15.3.2018. https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=mysqlphp01

Laaksonen, A. 2011. PHP-ohjelmointi: Osa 1 - Johdanto. Ohjelmointiputka 2011. Viitattu 2.1.2018. https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_01

Long, S. 2015. Jessie is here. Raspberry Pi Foundation 29.9.2015. Viitattu 2.1.2018. https://www.raspberrypi.org/blog/raspbian-jessie-is-here/

Ridruejo, D. & Kallen, I. 2002. Sams Teach Yourself Apache 2 in 24 Hours. Sams Publishing. Viitattu 2.1.2018. https://books.google.fi/books?id=cnDuw7GV4uYC&lpg=PP1&hl=fi&pg=PP1#v=onepage&q&f=false

Rodriguez, A. 2008. RESTful Web services: The basics. IBM developerWorks 6.11.2008, 9.2.2015. Viitattu 24.2.2018. https://www.ibm.com/developerworks/library/ws-restful/index.html

Upton, E. & Halfacree, G. 2013. Raspberry Pi User Guide. Hoboken: Wiley. Viitattu 8.10.2017. http://ebookcentral.proquest.com/lib/samk/detail.action?docID=1574363