


KARELIA-AMMATTIKORKEAKOULU
Tieto- ja viestintäteknikan koulutusohjelma

Samu Tiainen

**AVOIMEN LÄHDEKOODIN KOTIAUTOMAATIOKESKUS
ÄLYKOTIYMPÄRISTÖÖN**

Opinnäytetyö
Toukokuu 2019

	<p>OPINNÄYTETYÖ Toukokuu 2019 Tieto- ja viestintäteknikan:n koulutus Ammattikorkeakoulututkinto Tikkarianne 9 80200 JOENSUU +358 13 260 600 (vaihde)</p>						
<p>Tekijä(t) Samu Tiainen</p>							
<p>Nimeke Avoimen lähdekoodin kotiautomaatiokeskus älykotiympäristöön</p>							
<p>Tiivistelmä</p> <p>Esineiden internet (IoT) -laitemäärien on arvioitu kasvavan yli 10 miljardia ensivuoteen mennessä. IoT:n ihmisille tuoma hyöty on suuri ja heidän tietoisuutta asiasta tulisi lisätä. Kotiautomaation tarve kumpuaa ihmisen perustarpeista ja niitä huomioimalla sekä IoT-laitteita hyödyntämällä saadaan heidän elämästä mukavampaa ja vaivattomampaa. Nykypäivänä on kuluttajille suunnattuja kotiautomaatiolaitteita paljon mutta suurin ongelma niissä on se, että niiden ohjaus tapahtuu vain laitevalmistajan omalla sovelluksella.</p> <p>Tässä tutkimuksessa selvitettiin, voidaanko avoimen lähdekoodin ohjelmistolla rakentaa toimiva älykotiympäristö yksityisasuntoon. Tutkimuksessa käytettiin konstruktivistista tutkimusnäkökulmaa sekä User story -menetelmää. Rakensin testiympäristön omaan asuntoon käyttäen edullisia komponentteja ja ilmaista kotiautomaatio-ohjelmistoa.</p> <p>Osana tutkimusta pystyttiin rakentamaan toimiva järjestelmä käyttäen edullisia ja avointa lähdekoodia tukevia komponentteja. Lisäksi tutkimuksessa luotu järjestelmä on erittäin joustava ja helposti laajennettavissa. Tämän tyyppiseen järjestelmään pystyttäisiin helposti yhdistämään useita kodin IoT -laitteita niiden käytön tehostamiseksi.</p>							
<p>Kieli Suomi</p>	<table border="1"> <tr> <td>Sivuja</td> <td>47</td> </tr> <tr> <td>Liitteet</td> <td>3</td> </tr> <tr> <td>Liitesivumäärä</td> <td>4</td> </tr> </table>	Sivuja	47	Liitteet	3	Liitesivumäärä	4
Sivuja	47						
Liitteet	3						
Liitesivumäärä	4						
<p>Asiasanat esineiden internet, älykoti, Home Assistant</p>							

 Karelia UNIVERSITY OF APPLIED SCIENCES	THESIS May 2019 Information and communication technology Tikkarinne 9 80200 JOENSUU FINLAND + 358 13 260 600 (switchboard)						
Author (s) Samu Tiainen							
Title Open source home automation for smart home environment							
Abstract <p>The Internet of Things (IoT) devices are estimated to increase by over 10 billion by next year. The benefit that IoT bring to people is great but people's awareness should be increased. The need for home automation comes from peoples basic needs. When you take that into consideration and utilize the IoT equipment it will make peoples life's more comfortable and effortless. Nowadays, there are a lot of home automation devices for consumers, but the biggest problem is that they are controlled only by the manufacturer's own application.</p> <p>This study examined whether open source software can build a functional home environment for a private home. This study used constructive research perspective and the User story method. I built a test environment for my home using low cost components and free home automation software.</p> <p>As part of the research, a workable system could be built using low cost and open source support components. In addition, the system created in the study is very flexible and easily expandable. This type of system could easily be combined with multiple home IoT devices to enhance their use.</p>							
Language Finnish	<table border="1"> <tr> <td>Pages</td> <td>47</td> </tr> <tr> <td>Appendices</td> <td>3</td> </tr> <tr> <td>Pages of Appendices</td> <td>4</td> </tr> </table>	Pages	47	Appendices	3	Pages of Appendices	4
Pages	47						
Appendices	3						
Pages of Appendices	4						
Keywords Internet of things, Smart home, Home Assistant							

Sisältö

1	Johdanto	2
2	Tarkoitus Ja Tehtävä	3
3	Tietoperusta	4
3.1	ESINEIDEN INTERNET (IOT)	4
3.2	VAIVATON JA MUKAVA KOTI	6
3.3	ÄLYKOTI	8
3.4	MENETELMÄLLISET VALINNAT	9
4	Empiria; Älykotiympäristön Rakentaminen	11
4.1	HOME ASSISTANT	11
4.2	RASPBERRY PI (RASPI)	14
4.3	NODEMCU (NODE)	15
4.4	MQTT	16
4.5	MOSQUITTO	17
4.6	SAMBA	17
4.7	KÄYTTÖYMPÄRISTÖN ASENNUS	18
4.7.1	<i>Raspi Ja Hass</i>	18
4.7.2	<i>Samba</i>	20
4.7.3	<i>Mqtt Ja Mosquitto</i>	21
4.7.4	<i>Nodemcu Mqtt Ja Wifi</i>	24
4.7.5	<i>Himmenninpiirin Asennus</i>	27
4.7.6	<i>Järjestelmäarkkitehtuuri</i>	29
4.8	KÄYTTÄJÄTARINOIDEN TOTEUTUS	31
4.8.1	<i>User Story 1: Valokatkaisija</i>	31
4.8.2	<i>User Story 2: Manuaalinen Himmennys</i>	34
4.8.3	<i>User Story 3: Automaattinen Himmennys</i>	36
4.9	YHTEENVETO	40
5	Pohdinta	43
6	Lähteet	46

Liitteet:

Liite 1. Samba -konfiguraatio

Liite 2. Himmennin noden -koodi

Liite 3. Sensori noden -koodi

1 Johdanto

Kansainvälisen ICT-alan tutkimus- ja konsultointiyritys Gartnerin [1] mukaan vuonna 2016 noin 6,4 miljardia laitetta on kytkettynä internetiin. Mutta tämä on vasta alkua, sillä vuonna 2020 internetiin arvioidaan olevan kytkettynä jopa 20 miljardia laitetta. Tämä tarkoittaisi yli kolmea laitetta maapallon jokaista ihmistä kohden. Tätä erilaisten laitteiden kytkemistä verkkoon kutsutaan esineiden internetiksi, IoT (Internet of Things). Käyttökokemuksen kannalta on oleellista, että eri toimittajien ja eri toiminnallisuuksiin tarkoitettut (IoT) laitteet saadaan toimimaan yhdessä.

Olen itse huomannut älykkään kodin hyödyt ja sen tuoman helpotuksen arkeen. Arjen askareet nopeutuvat, jolloin itselle jää aikaa omiin mielenkiinnon kohteisiin. Esineiden internetin tuomat hyödyt ovat kaikkien saatavilla, mutta nykypäivänä ihmisillä ei ole vielä riittävästi tietoa asiasta. Tiedon puute vähentää luottamusta esineiden internetiin ja sen myötä ihmiset eivät ole valmiita hyväksymään teknologiaa osana arkea. Haluan työlläni osoittaa ihmisille, miten yksinkertaisesta ilmiöstä on kyse. Haluan myös näyttää, että toimivan älykodin luomiseen ei tarvita suurta taloudellista investointia.

Monet ihmiset ovat huolissaan älylaitteiden tietojen keräämisestä sekä niiden lähettämisestä pilvipalveluihin ja laitteistojen valmistajille. Avoimen lähdekoodin omaavan ohjelmiston voi rakentaa täysin irti internetistä, jolloin kaikki järjestelmän tieto on vain järjestelmän omistajalla. Tämän tutkimuksen tarkoituksena on löytää keino yhdistää käyttäjän määrittelemät IoT-laitteet yhteen järjestelmään.

2 Tarkoitus ja tehtävä

Tämän tutkimuksen aiheena on selvittää, miten avoimen lähdekoodin omaavalla ohjelmistolla pystytään rakentamaan yksinkertainen älykotiympäristö yksityisasuntoon. Kokonaisuutta pyritään selvittämään käyttäjäkokemusten sekä henkilökohtaisten tutkimusten kautta. Empiirisessä osassa rakennetaan yksinkertainen älykotiympäristö hyödyntäen ilmaisohjelmistoa pohjana.

Tutkimuskysymyksenä on, voidaanko avoimen lähdekoodin omaavalla ohjelmistolla rakentaa kelvollinen älykotiympäristö yksityisasuntoon.

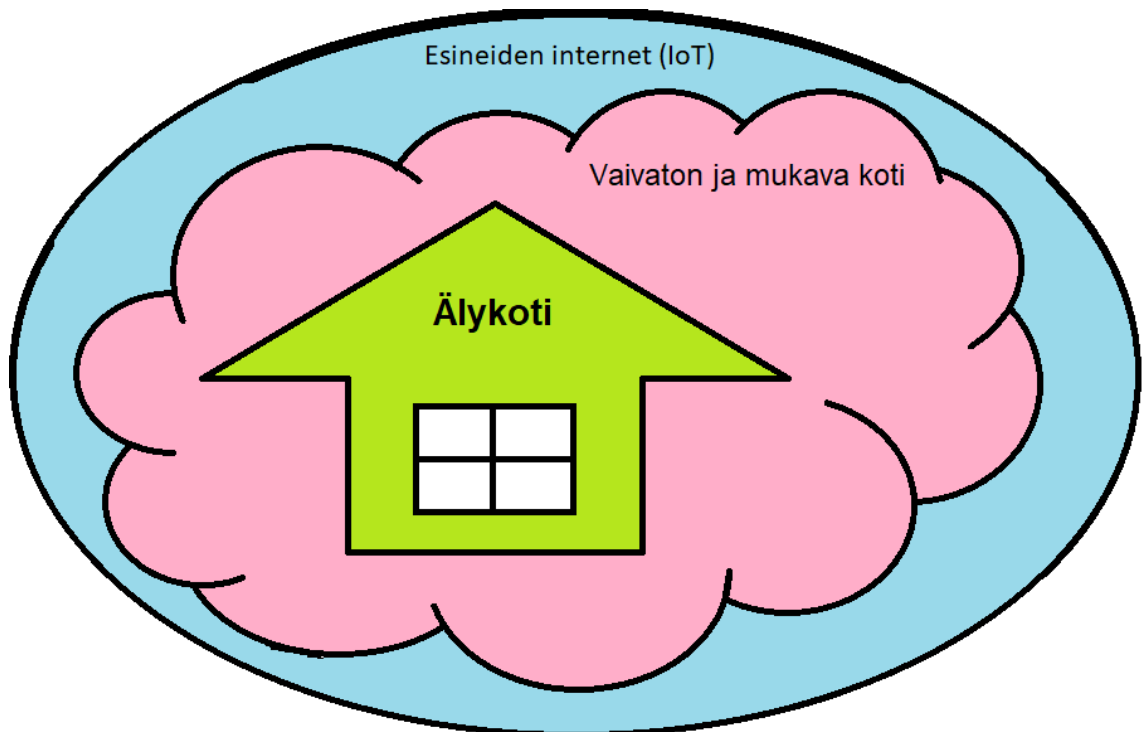
Tutkimuksen käytännön tilanteena voisi olla yksityisasunto, johon halutaan tuoda arkea helpottavia laitteita, kuten älylamppuja, ja hallita niitä yksinkertaisen ohjelmiston tai käyttöliittymän avulla.

Tutkimuksen tavoitteena on selvittää, voidaanko älykotiympäristö rakentaa ilman kaupallisia ohjelmistoja tai ratkaisuja.

Tämän työn empiirisessä osassa tutkitaan tietyn kotiautomaati-ohjelmiston soveltumista haluttuun älykotiratkaisuun, joten tulokset ja johtopäätökset eivät välttämättä sellaisenaan ole soveltuvia muihin käyttötarkoituksiin. Tutkimus pitää sisällään kotiautomaatio-ohjelman ja siihen kytkettyjä IoT-laitteita. Laitteiden määrällä on vaikutusta kokonaisratkaisun käytettävyyteen ja hyötyihin, mutta tässä tutkimuksessa rajoitettiin kytkettyjen IoT-laitteiden määrä kolmeen.

3 Tietoperusta

Tässä luvussa selvitetään esineiden internet -käsitettä sekä sen mahdollisuuksia täyttää ihmisten perustarpeita. Käydään läpi esineiden internetin mahdollisuuksia luoda vaivaton ja mukava koti. Luvussa tarkastellaan myös älykodin vaikutusta ja tarpeita (Kuva 1).



Kuva 1. IoT, vaivaton ja mukava koti sekä älykoti.

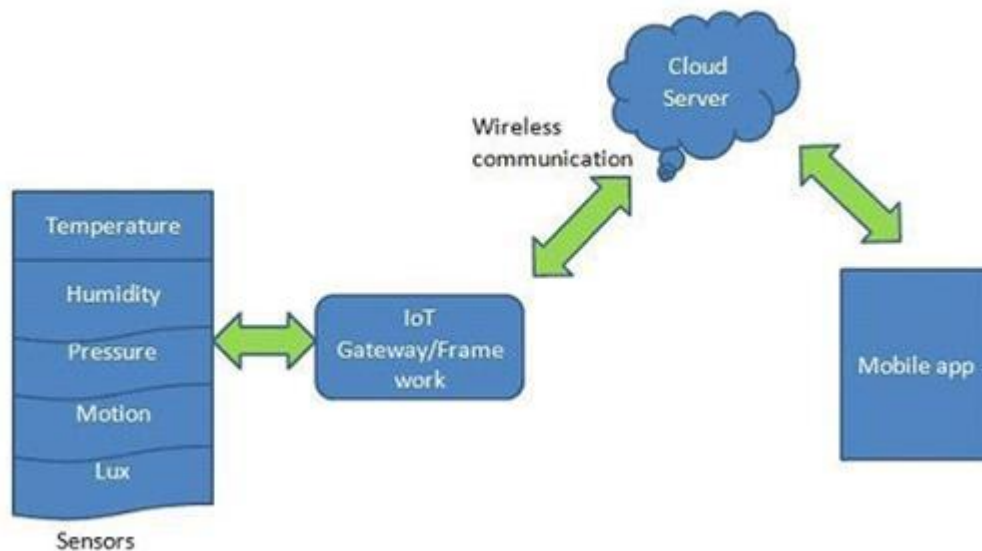
3.1 Esineiden internet (IoT)

Esineiden internetillä (englanniksi: internet of things, IoT) tarkoitetaan Atzorin, Ieran ja Morabiton [2] mukaan toistensa kanssa kommunikoivia asioita ja esineitä, jotka esiintyvät kaikkialla ympärillämme. Minolin [3] mukaan tavallista internetiä pyritään sisäistämään esineiden internetiin, jossa perinteisen internetin ympärille tuodaan erilaisten fyysisten esineiden yhteen liitoksia.

Esineiden internetillä on suomen kielessä muutamia rinnakkaisia käsitteitä, kuten asioiden internet ja teollinen internet. Esineiden internet-käsitteellä on lähes

sama tarkoitus kuin M2M-käsitteellä (machine to machine). M2M-käsitettä käytetään enemmän liiketoiminnassa yritysten prosessien automatisoinnissa ja tehostamisessa. [4]

IoT on maailmanlaajuinen megatrendi, jossa fyysiset laitteet liitetään antureilla ja sensoreilla verkkoon. Verkkoon liitetyt laitteet pystyvät aistimaan ympäristöään, viestimään ja toimimaan käsittelemänsä tiedon perusteella älykkäästi (Kuva 2). [5]



Kuva 2. IoT järjestelmän perustoiminta.

Kun reaaliaikaista tietoa analysoidaan automatisoidusti, saadaan enemmän informaatiota päätöksenteon tueksi. IoT:n idea lyhykäisyydessään on, että laitteisiin ja koneisiin lisätään älyä. Laitteet pystyvät aistimaan ympäristöään ja aistiensa perusteella ne alkavat hoitamaan tehtäviä tai viemään havainnoistaan viestiä eteenpäin. Laitteiden verkostointi internetin avulla helpottaa laiteviestien keräämistä ja hyödyntämistä. [6]

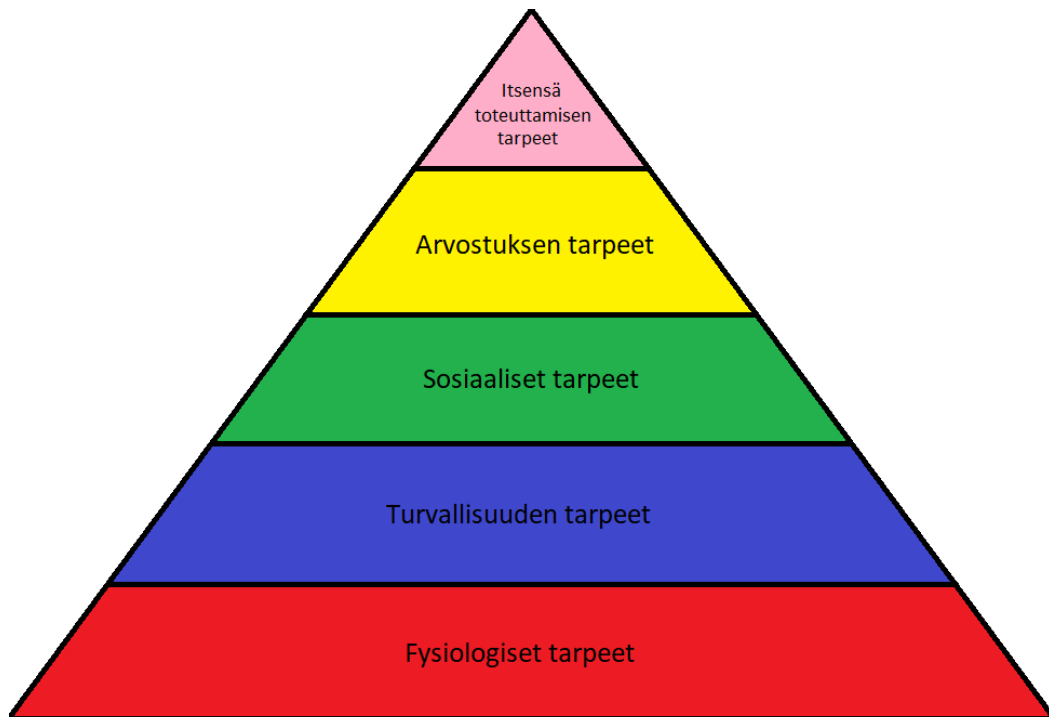
Esineiden internet on laaja käsite ja jotta sitä pystyttäisiin paremmin ymmärtämään, pitää määritellä muutamia siihen liittyviä elementtejä. Yksi keskeinen elementti on infrastruktuuri, joka mahdollistaa esineiden internetiin liittyvän tiedon keräämisen, tallentamisen ja jakamisen. Käytännössä tämä tarkoittaa mobiili- tai wifi-verkkoa, joka mahdollistaa IoT:n tuoman kapasiteettitarpeen kasvattamisen.

Jatkuva internetyhteys sekä esineiden internetiin liittyvät pilvipalvelut ovat oleellinen osa kokonaisuutta, koska kerättävä data on tallennettava paikkaan, josta se voidaan edelleen jakaa muille laitteille. Käytännössä mikä tahansa laite, joka täyttää vaatimukset laskentatehon, verkkoyhteyden, datan käsittelyn ja ohjelmiston osalta, voi kuulua esineiden internetiin. [7]

Esineiden internetillä on kuitenkin muutamia ongelmia, jotka hidastavat teknologian kehitystä. Esineiden internetin tulee taata luottamus, yksityisyys ja turvallisuus [2], jotta kuluttajat hyväksyvät teknologian. On tärkeää estää hakkerien hyökkäykset ja estää asiattomien pääsy henkilökohtaisiin tietoihin verkon yli [8], mitä on valitettavasti viime aikoina ilmennyt useaan otteeseen perinteisessä internetissä. Myös ihmisten yksityisyys on taattava [8], koska muutoin kaupalliset yritykset pyrkivät rikkomaan ihmisen yksityisyyden saadakseen kaiken mahdollisen tiedon parantaakseen liiketoimintaansa. Jos esineiden internet jättää turvallisuuden ja yksityisyyden huomiotta, luottamuksen rakentaminen on todella vaikeaa kuluttajien keskuudessa.

3.2 Vaivaton ja mukava koti

Teknologian määrä ihmisten kodeissa on kasvanut nopeaa vauhtia viime vuosina, ja kasvu tulee jatkumaan tulevaisuudessakin. Tulevaisuudessa esineiden internet tulee olemaan mullistava asia ihmisten arjessa. Maslown tarvehierarkiateoria kuvaa ihmisten tarpeita ja tarpeiden tasoa hiarkisessa järjestyksessä alkaen ihmisen perustarpeista [9]. Maslown tarvehierarkiaa kuvataan useimmiten pyramidina (Kuva 3). Tarvehierarkiaa voidaan käyttää markkinoinnissa arvioinnin apuna määrittelemään, mitä kuluttajan tarvetta tuote tai palvelu tyydyttää [10].



Kuva 3. Maslown tarvehierarkia.

Roblesin ja Kimin [11] mukaan älykäs koti lisää asukkaan mielenrauhaa ja tekee asukkaan elämästä helpompaa ja kätevämpää. Matternin ja Floerkemeierin [12] mukaan esineiden internet tekee ihmisen elämästä miellyttävämpää ja viihtyisämpää. Älykkään kodin tuomat asiat tekevät kodin tehtävistä vaivattomampia ja parantaa käyttäjän ajankäyttöä.

Maslown tarvehierarkian alimmalla tasolla ovat ihmisen fysiologiset tarpeet, joihin kuuluvat muun muassa vesi ja ruoka. Nämä tarpeet ovat ensisijaisia ihmiselle ja ne ovat olemassa ilman esineiden internetiä. [9]

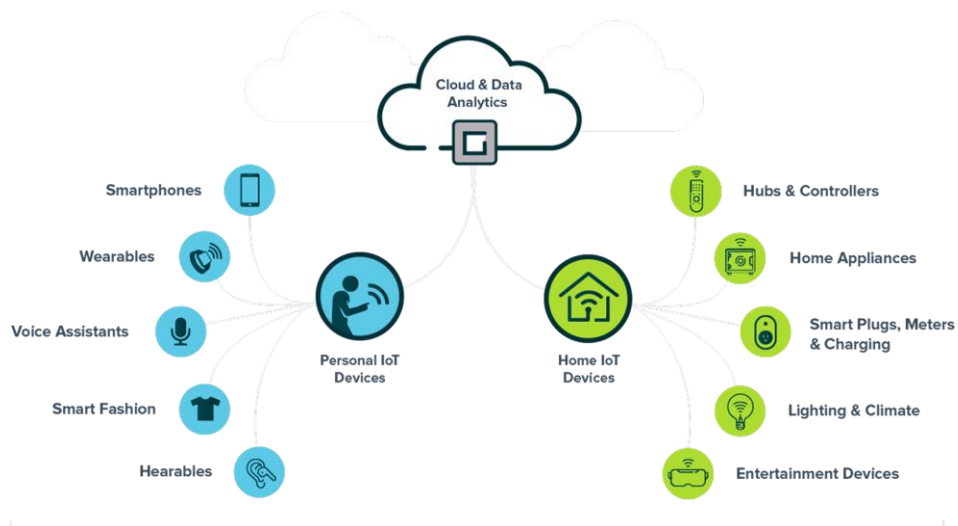
Turvallisuuden tunne on Maslown tarvehierarkian toinen taso, johon sisältyy ihmisen terveys [9]. Esineiden internet–teknologia mahdollistaa huomattavia terveyshyötyjä asukkailleen erilaisten sensorien, verkkoyhteyksien ja sovellusten avulla [13].

Usein hienoilla ja uusilla teknisillä laitteilla ihmiset haluavat parantaa sosiaalista asemaansa ja hakea arvostusta muilta ihmisiltä [13]. Nämä tarpeet kuuluvat arvostuksen ja sosiaalisuuden tarpeisiin, jotka ovat Maslown tarvehierarkian kolmas ja neljäs taso [9].

Älykäs koti säästää käyttäjän aikaa, jolloin hänelle jää enemmän aikaa toteuttaa omia mieltymyksiään. Ihmisen mieltymystensä toteuttaminen kuuluu Maslown tarvehierarkiassa ylimpään tarvetasoon eli itsensä toteuttamisen tarpeisiin [9]. Ihmisten on tärkeää huolehtia omasta ja muiden terveydestä.

3.3 Älykoti

Kotona olemisen mukavuutta ja vaivattomuutta on yritetty parantaa jo aikojen alusta asti. Nuotio helpotti ihmisten ruuan valmistusta ja tarjosi lämpöä, sähkölamppu mahdollisti valon saannin pimeään aikaan ja nykypäivänä kotiautomaatio hienosäätää useita arkeen liittyviä asioita. Ensimmäisen sukupolven kotiautomaatiota mainostettiin vapaa-ajan tehokkuuden ja perheitten yhdessäolon kasvattamisena. [14] Esineiden internet on yhä enemmän osana arkipäivää. Yhä useammasta taloudesta löytyy esimerkiksi etäluettava sähkömittari, älytelevisio, valvontakamera tai aktiivisuusranneke. Kaikki edellä mainitut ovat niin sanottuja IoT-laitteita mutta Esineiden internetin täysi hyöty on vielä käyttämättä. (Kuva 4)



Kuva 4. Esimerkkejä kodin IoT-laitteista.

Kotiautomaatio (tai älykoti) on kokonaisuus, jossa kodin laitteiden ja sovellusten toiminnot ovat vahvasti automatisoituja. Langattomalla kotiautomaatioverkolla tarkoitetaan kodin laitteiden ja sovellusten yhteen toimivuutta esineiden internet-

teknologian avulla. Langaton kotiautomaatioverkko mahdollistaa kodin käyttömu- kavuuteen ja tehokkuuteen liittyvien sovelluksien ohjaamista ja seuraamista lan- gattomasti [15]. Kotiautomaatiojärjestelmiä ohjataan yleensä älypuhelimella, eril- lisellä monitorilla, web-käyttöliittymällä tai ääniohjauksella.

Kotiautomaatiotekniikalla on vielä selvitettävä useita ongelmia ennen kuin se al- kaa yleistymään ihmisten keskuudessa. Kotiautomaatiotekniikan ongelmia ovat muun muassa monimutkaisuus, yhteensopivuus eri kodin tekniikoiden kanssa ja liitännöiden joustamattomuus [16]. Starsnic [17] nostaa esiin myös kotiverkkojen käyttäjäystävällisyyden kuluttajien laajan hyväksynnän saamiseksi. Esineiden in- ternet tulee leviämään muun muassa autoihin, kaupunkeihin, teollisuuteen ja toi- mistoihin. On kuitenkin muistettava että, yksi tärkein esineiden internetin sovelta- misala on kotiautomaatio [15], minkä vuoksi ongelmia pyritään tosissaan ratkaisemaan.

3.4 Menetelmälliset valinnat

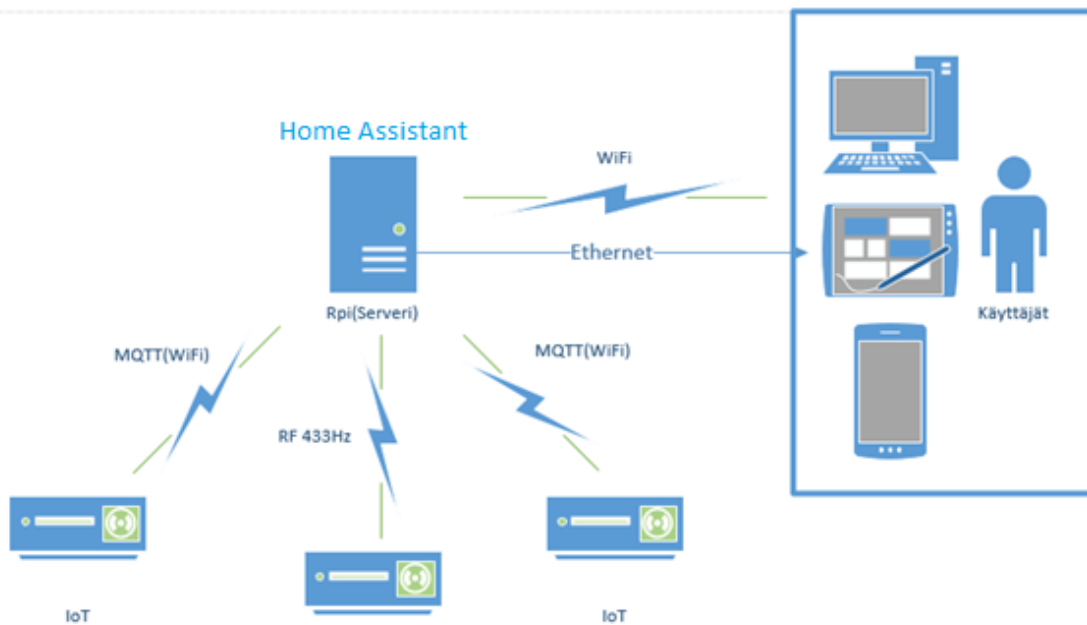
Tutkimuksen empiirisen osan vaatimusmäärittelyjen työvälineenä käytetään "User story" -kuvaus tapaa (käyttäjätarina). "User story" on ohjelmistokehityk- sessä käytetty työkalu vaatimuksen korkean tason määrittelyyn loppukäyttäjän näkökulmasta. Käyttäjätarina kuvaa käyttäjän roolia, mitä hän haluaa ja miksi. Se kuvaa myös loppukäyttäjän kannalta arvoa tuottavia toiminnollisuuksia. Käyttäjä- tarina auttaa luomaan yksinkertaistetun kuvauksen vaatimuksesta. Työkalu kes- kittyy lopputulokseen ja antaa hyvän kuvan tutkimuksen suunnasta ja tarpeista. Työkalun ei ole tarkoitus antaa kiveen hakattuja ohjeita vaan pintapuolisen ku- vauksen tarvittavista asioista. Tämä antaa tekijälle vapaammat kädet toteutuksen suhteen. Koska käyttäjätarinat kuvaavat todella hyvin ongelmatilanteen loppurat- kaisun saavuttamista ja tukevat uusien ideoiden luomista, oli minulle selvää käyt- tää tutkimuksessani konstruktivistista tutkimusnäkökulmaa.

Konstruktivistisen tutkimuksen lähtökohdat ovat käytännön ongelmallisessa tilan- teessa ja lopputulosta pyritään käyttämään ongelman ratkaisemiseen. Tutkimuk- seen kuuluu ongelman sitominen aiempaan tietämykseen sekä ratkaisun toimi- vuuden osoittaminen empiirisesti. Konstruktivistisen tutkimusotteen päätekijä on

teoreettinen analyysi, minkä tuloksena muodostuu ratkaisu tiettyyn ongelmaan. Konstruktiiivisessa tutkimuksessa uuden konstruktion kehittäminen on koko prosessin päätavoite. Konstruktiiivisessa tutkimuksessa teoreettinen analyysi on tärkeässä roolissa, jolloin aiempaa teoriaa analysoidaan ja sen perusteella päätellään vastauksia tutkimuskysymyksiin. Pohdiskelun avulla muodostetaan jokin uusi idea ratkaisuna käsiteltävään tutkimusongelmaan. Konstruktiiivisessa tutkimusotteessa korostuvat luovuus ja innovatiivisuus. [18]

4 Älykotiympäristön rakentaminen

Työn tarkoituksena oli testata älykotijärjestelmän rakentamista käyttäen vain avoimen lähdekoodin ohjelmistoja sekä edullisia komponentteja. Edullisuutensa takia järjestelmän keskipisteeksi valittiin Rpi-serveri, johon oli helppo asentaa kotiautomaation keskuksena toimiva avoimen lähdekoodin omaava Home Assistant-sovellus. Home Assistant-ohjelmisto sisältää käyttöliittymän, jonka kautta sitä voidaan ohjata millä tahansa internetiin kytketyllä laitteella. Halutut IoT-laitteet voidaan kytkeä järjestelmään Rpi-serverin tarjoamien bluetooth-, wifi- tai rf-yhteyksien kautta (Kuva 5).



Kuva 5. Tutkimuksen arkkitehtuuri.

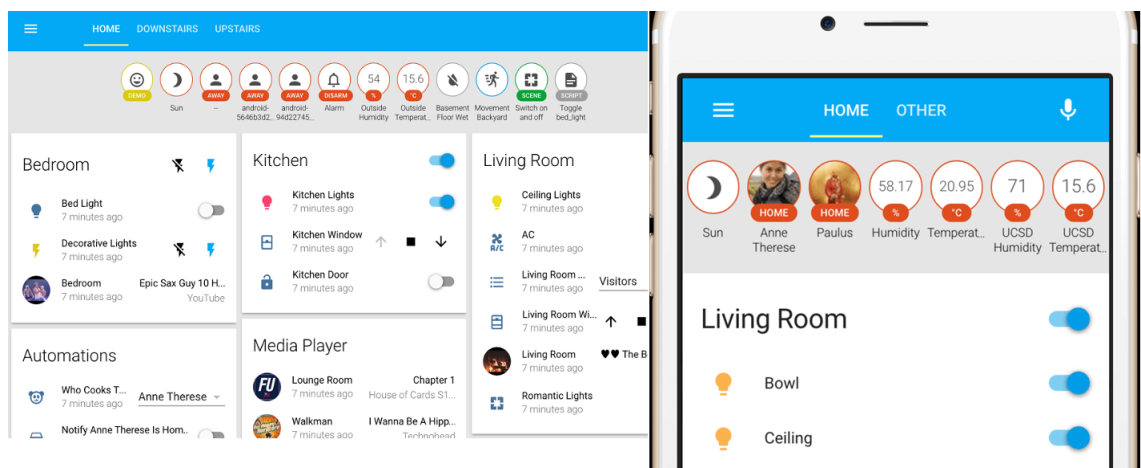
4.1 Home Assistant

Home Assistant on Python 3 -pohjainen avoimen lähdekoodin omaava ohjelmisto, joka on tehty toimimaan kotiautomaation keskuksena. Paulus Schoutsen julkaisi vuonna 2013 kotiautomaatiojärjestelmän (HASS), jonka tarkoitus oli tuoda kotiautomaation ohjaus lähelle käyttäjää, sen sijaan että kaikki ohjaus tapahtuisi pilvessä. Järjestelmä haluttiin myös toimimaan ilman internetyhteyttä, jotta tietoturva olisi mahdollisimman tehokas.

HASS tukee tällä hetkellä noin 1200 eri laitetta, joiden käyttöön löytyy valmiit ohjeet ja kommunikointirajapinnat. Schoutsen julkaisee Home Assistantille uuden versiopäivityksen noin kaksi kertaa kuukaudessa ja tekee tarvittavat dokumentoinnit ja ohjeet erittäin hyvin. Lisäksi yhteisö on erittäin suuri ja elävä, foorumin aktiivisuudesta päätellen. Yhteisöä löytyy myös Reddit-verkkosivustolta sekä Discord-keskustelusovelluksesta, joten ongelmien tai omien projektien jakaminen on hyvin helppoa ja tukea löytyy nopeasti.

HASS toimii monella käyttöjärjestelmällä sen useiden eri versioiden ansiosta. Tutkimuksessa käytettiin RasPi-tietokonetta, johon valittiin Hassbian-niminen versio. Se on RasPi-tietokoneelle tarkoitettu Rasbian-käyttöjärjestelmän muokattu versio. Muokatussa versiossa on siis sisäänrakennettuna HASS-ohjelma. Muita versioita on muun muassa HASS.io, joka on pelkistetty käyttöjärjestelmä, jossa HASS toimii minimaalisin vaatimuksin. HASS pystytään siis asentamaan yleisimpiin käyttöjärjestelmiin, mikä tekee sen käytöstä paljon monipuolisemman. Tutkimuksessa käytetty Home Assistantin versionumero on 0.82.1.

Asennettuna HASS toimii selainsovelluksena, johon pääsee käsiksi sen omalla IP-osoitteella. Valmiissa käyttöliittymässä on paljon ominaisuuksia ja asetuksia, joilla hallitaan kotiautomaatiolaitteita (Kuva 6). HASS konfiguroidaan halutunlaiseksi muokkaamalla ohjelmistoon kuuluvaa YAML-tiedostoa. Esimerkiksi jos halutaan lisätä älyvalo-ohjauksen, niin HASS-kotisivuilta löytyy valmis dokumentointi ja käyttöohjeet haluttuun älyvaloon. Sivulta löytyvä konfiguraatio koodi lisätään YAML-tiedostoon ja älyvalo on toiminnassa.



Kuva 6. Esimerkki käyttöliittymästä tietokoneella sekä puhelimella.

4.2 Raspberry Pi (RasPi)

Raspberry Pi on sarja pieniä mikrotietokoneita, jotka on suunniteltu Englannissa Raspberry Pi -yhdistyksen avulla. RasPi-sarjan tarkoituksena oli helpottaa tietotekniikan perusteiden opettamista kouluissa ja kehitysmaissa. Sarjan kolmas versio (RasPi 3 model B), jota tutkimuksessa käytetään, on tuonut markkinoille Raspberry Pi Trading. Se on vastuussa kehitystyöstä, kun taas aikaisemmin mainittu Raspberry Pi -yhdistys edustaa hyväntekeväisyys- ja opetuspuolta.

RasPi 3 model B on siis sarjan kolmas malli, johon on lisätty paljon uusia ominaisuuksia sekä paranneltu komponentteja. Uusina ominaisuuksina on muun muassa sisäänrakennettu WiFi, Bluetooth ja USB boot. Myös aikaisemmissa malleissa ollut yksiytiminen ja 32-bittinen suoritin on vaihdettu 64-bittiseen moniydinsuorittimeen (Kuva 7).



Kuva 7. Raspberry Pi Model B.

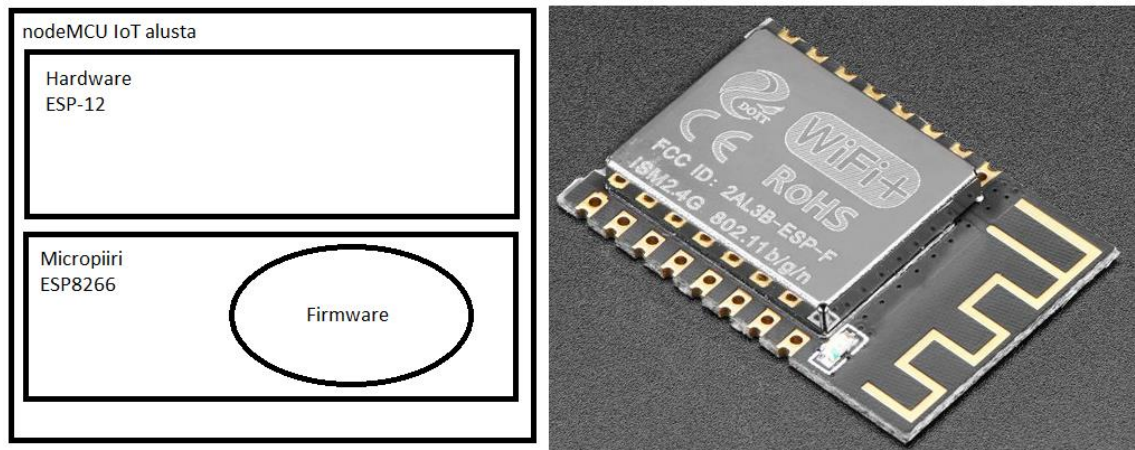
Raspberry Pi 3 malli B info:

- Neliydin 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 langaton LAN ja Bluetooth Low Energy (BLE)
- 100 Base Ethernet
- 40-pin jatkettu GPIO
- 4 USB 2 porttia
- 4 napainen stereosignaali ja komposiittivideoportti

- Normaali HDMI
- CSI kamera portti Raspberryn omalle kameralle
- DSI portti Raspberryn omaa kosketusnäyttöä varten
- Micro SD portti
- Micro USB 2.5A virtaliitin.

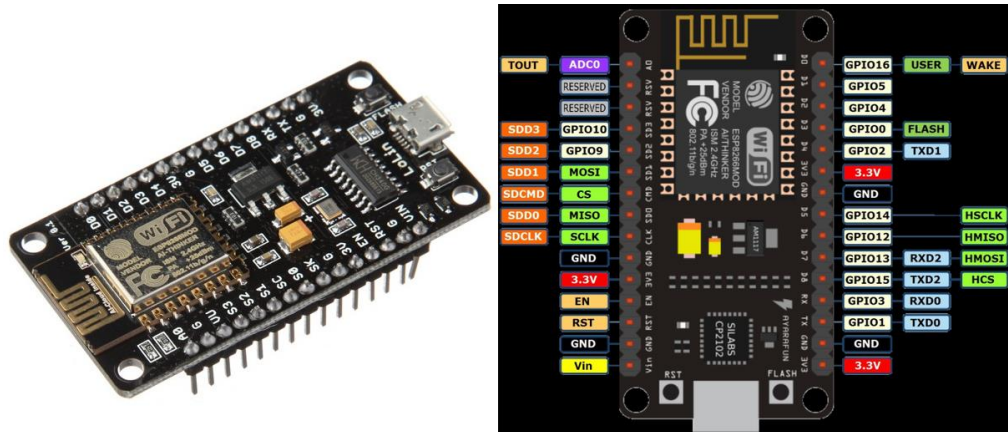
4.3 NodeMCU (node)

NodeMCU on avoimen lähdekoodin omaava IoT-alusta. Se sisältää ESP8266 WiFi-piirin ja oman firmwaren sekä ESP-12-moduulin (Kuva 8). NodeMCU-nimi viittaa siis ohjelmarakenteeseen, joka on mikrotietokoneen sisällä, mutta asioita helpottaen viittaa nimellä kokonaisuuteen.



Kuva 8. Kaavio laitteen rakenteesta sekä kuva ESP-12-moduulista.

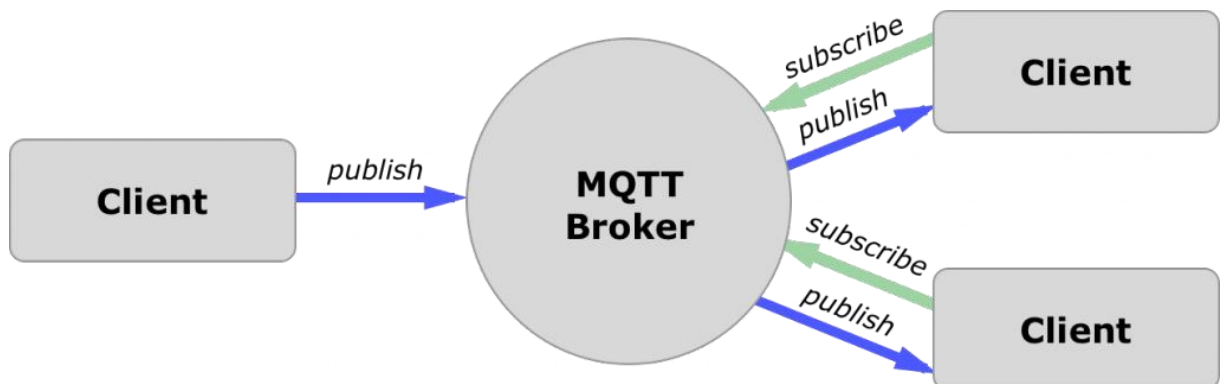
Kokonaisuuteen (Kuva 8) kuuluu siis ESP8266-mikrokontrolleri, josta löytyy täysi WiFi-tuki. Lisäksi nodesta löytyy 30 ohjelmoitavaa pinniä (Kuva 9), joihin voi kytkeä erilaisia sensoreita, kytkimiä tai pieniä näyttöjä. Nodessa on myös mikro-USB-paikka ja siihen tarvittavat ohjauskomponentit. Laitteen sisältämä mikrokontrolleri ohjelmoidaan mikro USB-portista, käyttäen Arduino IDE-ohjelmaa ja ohjelmointikielenä toimii C.



Kuva 9. NodeMCU-kehitysalusta ja pinnien kuvaukset.

4.4 MQTT

MQTT (Message Queuing Telemetry Transport) on ”julkaise ja tilaa” -menetelmää käyttävä kommunikaatioprotokolla (Kuva 10). MQTT toimii TCP/IP-protokollan päällä. Se on tarkoitettu laitteille, jotka lähettävät pieniä määriä dataa tai laitteille, joilla ei ole hyvää internetyhteyttä. ”Julkaise ja tilaa” -menetelmä vaatii erillisen ohjelman, joka hallinnoi ja välittää viestejä. MQTT-protokollan ensimmäinen versio julkaistiin vuonna 1999 ja vuonna 2013 MQTT sai virallisen standardin Oasikselta. Oasis on voittoa tavoittelematon kansainvälinen järjestö, jonka tarkoituksena on IT-alan standardien kehitys ja leviämisen edistäminen.



Kuva 10. MQTT toimintamalli.

MQTT-järjestelmä sisältää siis Käyttäjät sekä hallinnointiohjelman (BROKER). Käyttäjä voi joko julkaista viestejä tai tilata tietyn aiheen, jonka avulla vastaanote-

taan viestejä. Jokaisella käyttäjällä on oltava yhteys BROKER:iin. Viestien lähetys ja vastaanottaminen toimii käytännössä siten, että käyttäjä lähettää haluamansa viestin haluamaansa aiheeseen. Tämän jälkeen kaikki käyttäjät, jotka ovat tilanneet kyseisen aiheen vastaanottavat lähetetyn viestin. Kaikki viestit menevät BROKER:iin joka hoitaa viestien edelleen lähetyksen aiheen tilanneiden kesken. Viestin lähettäjän ei tarvitse tietää mitään vastaanottajasta kommunikoinnin toimimiseksi. Viestin lähettäjä voi laittaa viestiin "Retain"-lipun jonka avulla aina viimeiseksi lähetetty viesti jää BROKER:ille tietoon. Tämän avulla uusi laite, joka tilaa aiheen, saa viimeiseksi aiheeseen lähetetyn viestin tietoonsa. Jos viestissä ei ole "Retain"-lippua eikä kukaan ole tilannut aihetta, viesti poistuu automaattisesti. Protokollassa on myös kätevä ominaisuus nimeltä LWT (Last Will and Testament). Kun käyttäjä yhdistää BROKER:iin LWT-ominaisuutta käyttäen se voi määrätä syntaksissa oman viestin ja aiheen, jonka BROKER suorittaa, jos laitteeseen tulee jokin ongelma, kuten yhteyden katkaisu tai virran menetys.

MQTT-viestin koko voi olla minimissään jopa kahden tavun mittainen ja maksimissaan melkein 256 tavua. Viestin sisältö lähetetään pelkkänä tekstinä, joten se ei sisällä minkään näköistä tietoturvaominaisuutta. Tietoturvaa voidaan parantaa TCP protokollan sisältämien ominaisuuksien avulla tai lähettämällä kryptattuja viestejä.

4.5 Mosquitto

MQTT protokolla vaatii siis BROKER:in toimiakseen, eli erillisen ohjelman, joka hallitsee viestien kulkua. Tätä varten tutkimukseen valittiin ohjelman nimeltä Mosquitto. Mosquitto on erittäin kevyt ja vähän virtaa vievä MQTT BROKER-ohjelma, joka toimii monella eri laitteella. Ohjelma toimii C-kielellä ja sen käyttö on helppoa ja vaivatonta. Kyseinen ohjelma valittiin suosion ja hyvän dokumentoinnin takia.

4.6 Samba

Samba on ohjelmista se, joka tarjoaa Windows-verkkojen palvelut, kuten tiedostonjaon, Unix-tyylisille käyttöjärjestelmille kuten Linuxille. Samban avulla pääsee millä tahansa tietokoneella (Windows) käsiksi RasPin sisällä oleviin tiedostoihin. Tämä auttaa huomattavasti siinä, kun tarvitsee muokata joitakin tiedostoja RasPi-käyttöjärjestelmässä tai HASS-konfiguraatioissa.

4.7 Käyttöympäristön asennus

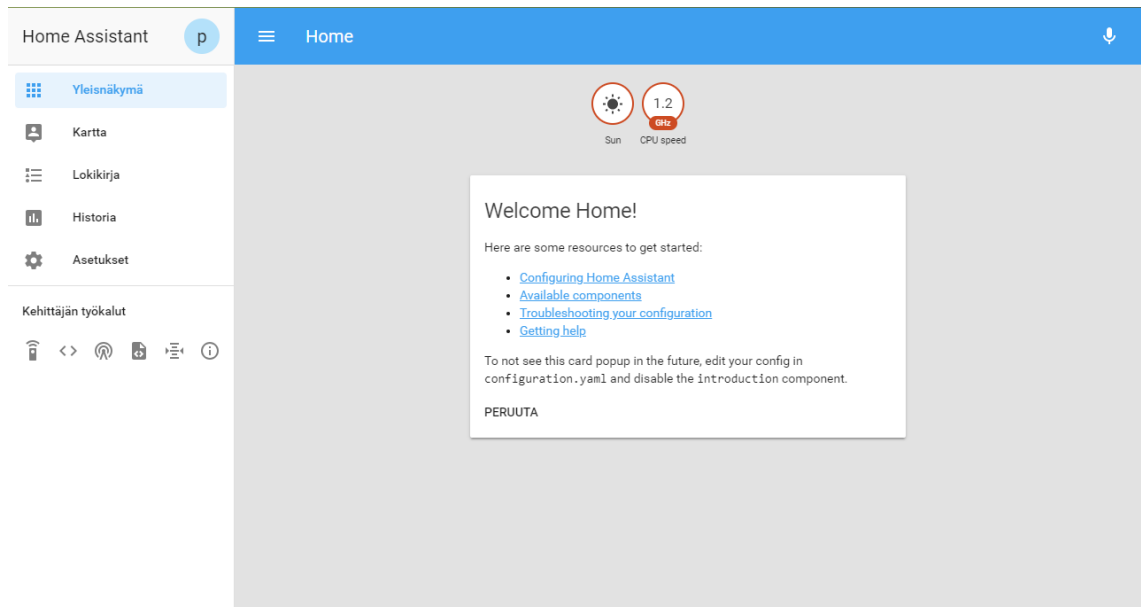
Tässä luvussa on kuvattu yksityiskohtaisesti tutkimuksessa käytetyn käyttöympäristön asennus. Työkaluina on käytetty itselle joko ennalta tuttuja tai yleisesti kehuttuja ohjelmistoja, jotka on valittu kustannusten ja saatavilla olevien ohjeiden mukaan. Asennusten vaiheissa on pyritty löytämään selkeät ohjeet eri osioille, jotta ne saataisiin toimimaan mahdollisimman varmasti.

4.7.1 RasPi ja HASS

Serverin asennusta varten tarvitaan RasPi-tietokone, microSD-muistikortti ja käyttöjärjestelmä. Käyttöjärjestelmä ladataan HASS verkkosivulta ja poltetaan muistikortille. Muistikortti laitetaan RasPi:ssa olevaan paikkaan ja kytketään se virtoihin. RasPi:t tunnistaa muistikortin ja käyttöjärjestelmän automaattisesti, sekä asentaa käyttöjärjestelmän itsestään.

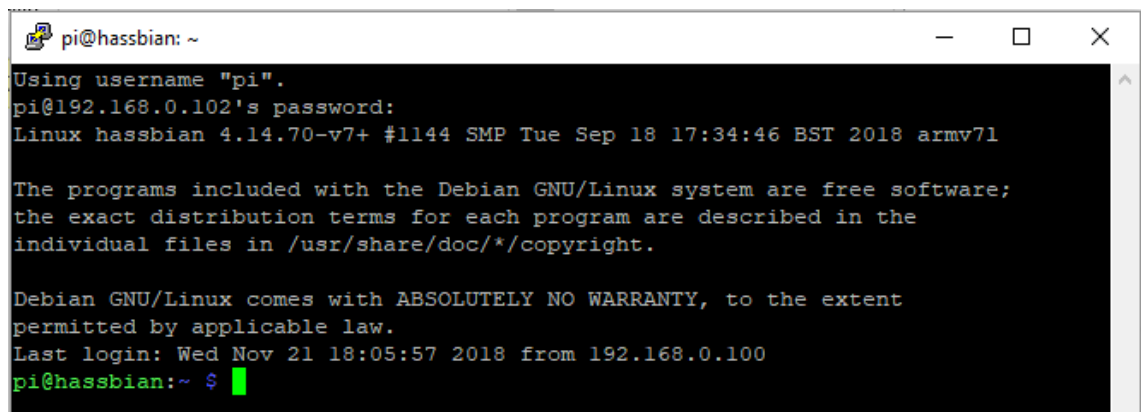
Seuraavaksi tarvitaan RasPi:n IP-osoite. Tätä varten on tehty fiksu ohjelma nimeltä FindMyPi, joka etsii automaattisesti lähiverkossa olevat RasPi:t ja ilmoittaa niiden IP-osoitteet. Laitteen IP:n saa tietoon usealla eri tavalla mutta tämä oli tutkimukselle helpoin.

Kun käyttöjärjestelmän asennus on suoritettu ja RasPi:n IP-osoite hankittu, pitää kotitietokoneen selaimella mennä annettuun IP-osoitteeseen ja porttiin (:8123), johon HASS on automaattisesti asennettu. Selaimessa avautuu HASS käyttöliittymä (Kuva 11), jolloin asennuksen voi todeta onnistuneeksi.



Kuva 11. Selaimen avautuva HASS käyttöliittymä.

Seuraavaksi järjestelmä pitää päivittää ja oletussalasana vaihtaa. Tätä varten tarvitaan "ssh" -yhteys käyttöjärjestelmän komentoriiviin. Yhteys komentoriiviin muodostetaan käyttämällä "Putty" -nimistä ohjelmaa. Ohjelma yhdistetään laitteen IP-osoitteeseen ja kirjaudutaan oletustunnuksilla sisään (Kuva 12).



Kuva 12. Ssh yhteyden luominen komentoriiviin.

Käyttöjärjestelmä päivitetään kahdella komennolla.

Tämä lataa viimeisimmät päivitykset

```
sudo apt-get update
```

Tämä asentaa ladatut päivitykset

```
sudo apt-get upgrade
```

Tämän jälkeen päivitetään HASS komennolla:

```
sudo hassbian-config upgrade homeassistant
```

Viimeisenä vaihdetaan oletussalasana kirjoittamalla "passwd"-komennon ja vaihtamalla sen avulla salasanan. Tämän jälkeen käyttöjärjestelmän ja serverin asennus on valmis.

4.7.2 Samba

Samba asennetaan RasPi:lle käyttäen komentoriviä. Ssh yhteyden jälkeen ko-

Asennetaan samba ja avataan konfiguraatiotiedosto:

```
sudo apt-get update
sudo apt-get install samba
sudo nano /etc/samba/smb.conf
```

mentoriville syötetään seuraavat komennot.

Konfiguraatio tiedosto tyhjennetään ja tilalle kopioidaan valmis koodi, joka löytyy liitteistä (Liite 1).

viimeisenä luodaan sambaan käyttäjä ja käynnistetään ohjelma uudelleen

```
sudo smbpasswd -a pi
(kirjoita salasana ja paina enter)
sudo service smbd restart
```

Jos asennus sujui onnistuneesti, pitäisi Windowsin tiedosto-ohjelmasta löytyä lähiverkko kohdasta lisäämäsi laite.

4.7.3 MQTT ja Mosquitto

Mosquitto asennetaan RasPi:lle käyttäen komentoriviä. Ssh yhteyden jälkeen komentoriville syötetään seuraavat komennot.

Tarvittava avain ja tiedosto saadaan tällä tavalla:

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
```

Tämän jälkeen tuodaan tiedosto saataville:

```
cd /etc/apt/sources.list.d/
```

Ladataan ohjelma:

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

Viimeisenä päivitetään tiedot ja asennetaan mosquitto:

```
apt-get update
apt-get install mosquitto
```

Kun ohjelmisto on asennettu pitää vielä muuttaa konfigurointi tiedostoa. Tiedosto avataan seuraavalla komennolla:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Tiedostoon lisätään seuraavat rivit:

```
allow_anonymous false
```

- rajoittaa tuntemattomien pääsyn

```
password_file /etc/mosquitto/pwfile
```

- vaatii salasanan käyttäjältä

- avaa portin kuuntelua varten

Mosquitto:n toiminnan testaamiseksi tarvitaan kaksi eri ssh-yhteyttä RasPi:in

```
listener 1883
```

(Kuva 13)

Kuva 13. Mosquitton testaaminen käyttäen kahta ssh-yhteyttä.

Toiseen komentoriviin syötetään seuraava komento:

```
mosquitto_sub -d -u username -P password -t "dev/test"
```

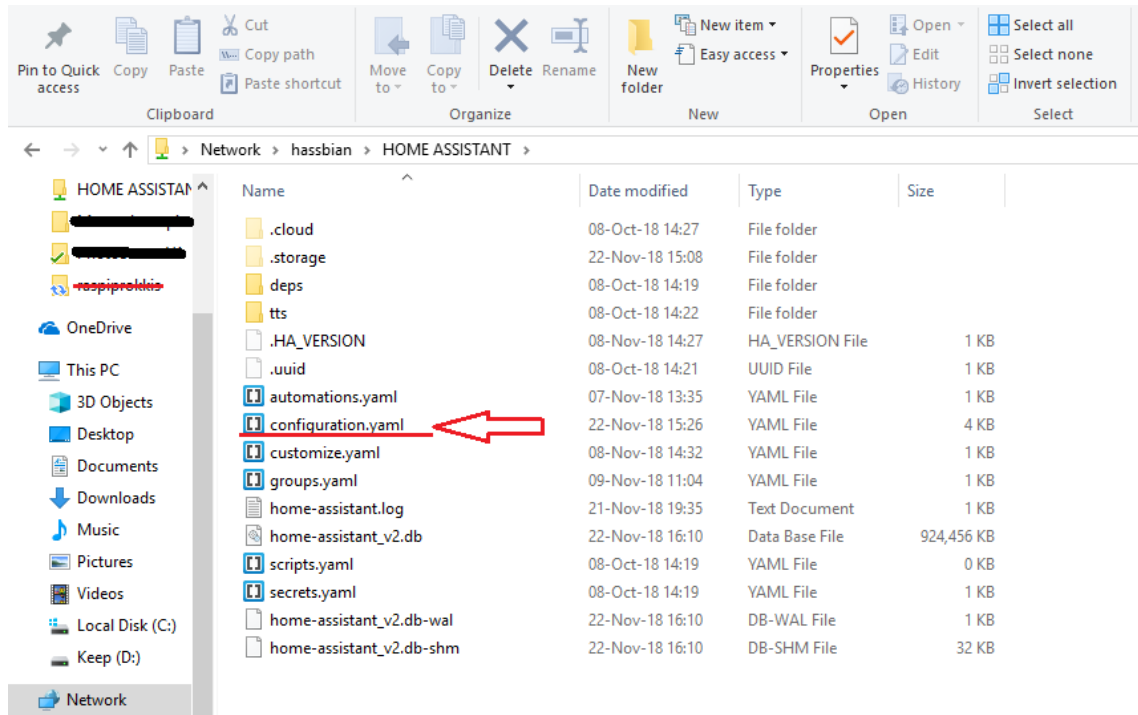
Tällä luodaan käyttäjä joka tilaa aiheen "dev/test"

Ja toiseen seuraava:

```
b. mosquitto_pub -d -u username -P password -t "dev/test" -m "Hello world"
```

Tällä luodaan käyttäjä joka lähettää viestin "Hello world" aiheeseen "dev/test"

HASS vaatii muutoksia konfiguraatioon, jotta MQTT saadaan toimimaan. Avaa HASS tiedostojen sijainti ja kansioista HOME ASSISTANT löytyy tiedosto configuration.yaml (Kuva 14).



Kuva 14. HASS tiedostorakenne sekä konfiguraatiotiedosto.

Konfiguraatio tiedosto avataan millä tahansa muokkausohjelmalla ja sinne lisätään seuraavat rivit mihin tahansa kohtaan.

```
mqtt: (Herättää ominaisuuden HASS:in sisällä)
  BROKER: 127.0.0.1 (Määrittää osoitteen jossa Mosquitto toimii)
  keepalive: 60 (Minuutin välein tarkistaa Mosquitton toiminnan)
  client_id: pi (Raspberryn käyttäjä)
  username: pi (Mosquittoon annetut tunnukset)
  password: salasana
```

Kun HASS konfiguraatiota muuttaa pitää muutokset tarkastaa ja käynnistää HASS uudelleen. Nämä asiat voi tehdä suoraan käyttöliittymästä menemällä Asetukset > Yleinen ja painamalla "Tarkista asetukset" nappia ja tämän jälkeen sivun alaosasta "Käynnistä uudelleen". Uudelleen käynnistys vie 1-5 minuuttia jolloin sivun voi päivittää ja uudet konfiguroinnit astuvat voimaan.

Tähän mennessä on toiminnassa MQTT yhteys HASS:in ja Mosquitton välillä. Yhteyden voi testata avaamalla HASS käyttöliittymästä kehittäjän työkalut kohdasta "MQTT", ja syöttämällä "dev/test" kohtaan "topic" ja halutun viestin kohtaan "payload". Pitää myös avata samanlainen ssh yhteys ja syöttää komentoriville alla oleva komento.

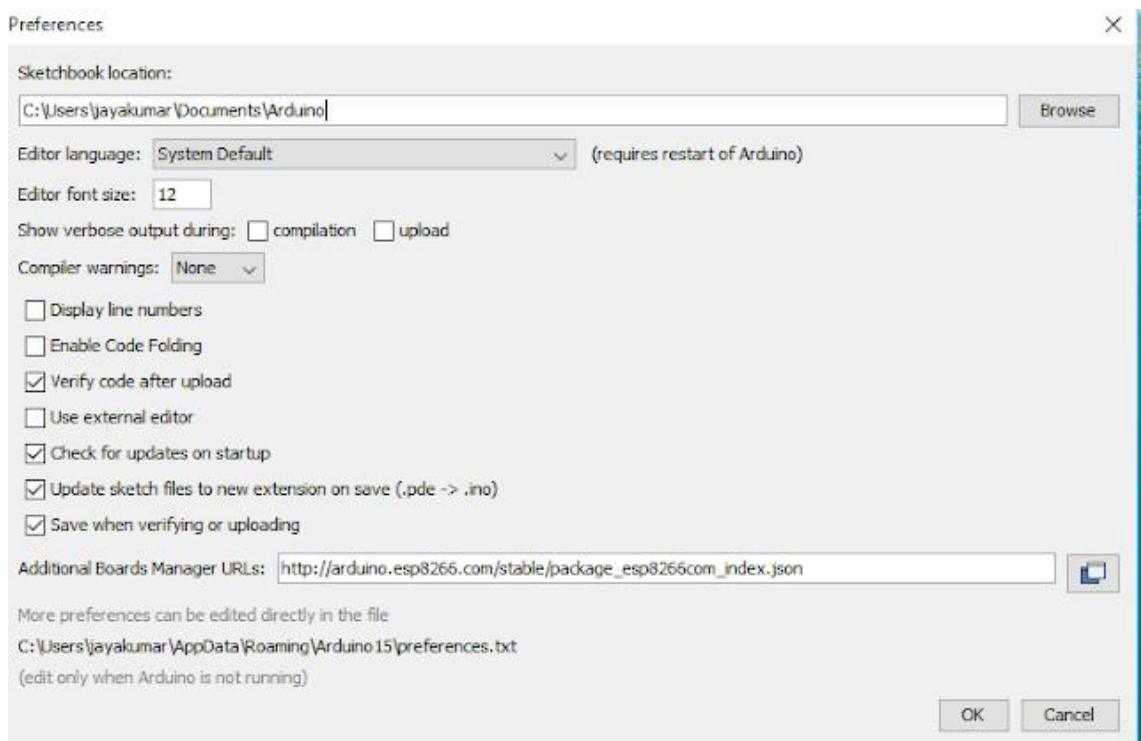
```
mosquitto_sub -d -u username -P password -t "dev/test"
```

Jos viesti näkyy ssh yhteys ikkunassa niin ohjelman voi todeta toimivaksi.

4.7.4 NodeMCU MQTT ja WiFi

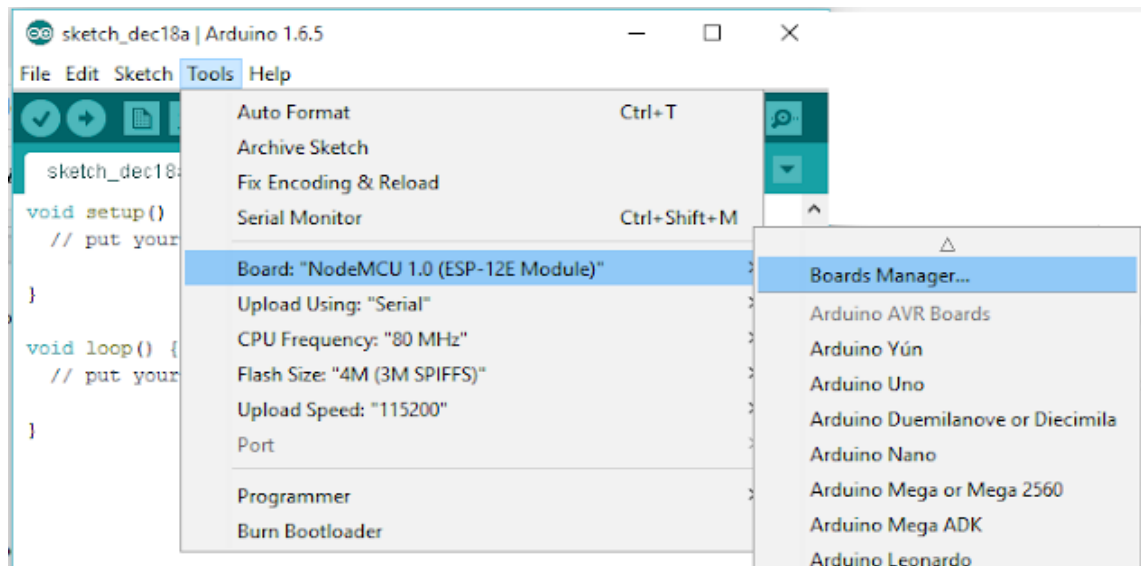
Tutkimuksessa ohjelmoitiin node käyttäen Arduino IDE-sovellusta jonka saa ilmaiseksi Arduinon omilta sivuilta. Arduino IDE-sovellukseen lisätään tuki esp8266 piirille menemällä välilehteen File > preferences ja lisäämällä "Additional Boards Manager" kohtaan tämä URL (Kuva 15).

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Kuva 15. Preferences ja Additional Boards Manager

Tämän jälkeen kohdasta Tools > Board valitaan "Boards Manager" (Kuva 16).



Kuva 16. Boards Manager

Etsitään ja asennetaan ohjelma nimeltä "esp8266 by esp8266 community". Viimeisenä kohdasta Tools > Manage Libraries, etsitään ja asennetaan ohjelma nimeltä "PubSubClient".

Kun tarvittavat kirjastot on asennettu, kirjoitetaan ohjelma. Ensimmäisenä lisätään alla olevat kirjastot.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

Seuraavaksi määritellään Käyttäjätietojen muuttujat, jotta yhdistäminen olisi helpoa.

```
const char* ssid = "WiFi-nimi"; //WiFi verkon nimi
const char* password = "SALASANA"; //Verkon salasana
const char* mqtt_server = "RasPi-IP"; //Mosquitto serverin osoite
const char* clientId = "DimmerClient"; //Käyttäjän nimi
const char* userName = "pi"; //MQTT käyttäjänimi
const char* passWord = "SALASANA"; //ja salasana
```

Tutkimuksessa käytetään molemmissa nodeissa samoja tunnuksia paitsi "clientId" joka pitää olla kullakin käyttäjällä uniikki.

Sitten luodaan käyttäjät "wifi" ja "pubsub" ohjelmistoa varten:

```
//luodaan WiFi käyttäjä sekä PubSub käyttäjä
WiFiClient DimClient;
PubSubClient client(DimClient);
```

Lisätään funktio joka yhdistää annettuun lähiverkkoon.

```
void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  // yhdistetään wifiverkkoon annetulla käyttäjätunnuksella sekä salasanalla
  WiFi.begin(ssid, password);
  //odotetaan kunnes yhteys on muodostettu
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  // jos yhteys muodostetaan ilmoitetaan laitteen oma ip osoite
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
}
```

Alla oleva funktio muodostaa yhteyden ja uudelleenyhdistää noden MQTT BROKER:iin.

```

void reconnect() {
  // Muodostetaan MQTT yhteyttä uudelleen
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Jos yhteyttä ei saada luodaan uusi sattumanvarainen clientID
    String clientId = "Dimmer-";
    clientId += String(random(0xffff), HEX);
    // Yhdistetään MQTT:hen ja lähetetään brokerille LWT
    if (client.connect(clientId.c_str(), willTopic, 1, "OFFLINE")) {
      //connect(Käyttäjänimi ja salasana, LWT aihe,"retain" , Viesti)
      Serial.println("connected");
      // tässä kohtaa voidaan suorittaa halutut viestit tiettyihin aiheisiin

      // ... ja ruvetaan kuuntelemaan haluttuja aiheita
      client.subscribe("dev/test");

    } else { //Valitaan jos yhteyttä ei saada
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Jos yhteyttä ei saatu yritetään uudelleen 5 sekunnin kuluttua
      delay(5000);
    }
  }
}

```

Viimeisenä tarvitaan "callback"-niminen funktio, jota kutsutaan aina kun tilattuun aiheeseen tulee viesti.

```

//Funktio kutsutaan automaattisesti kun tilattuun topikkiin tulee viesti
//viestissä on mukana topik, payload sekä viestin pituus
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  //For loopissa muutetaan byte muodossa tullut viesti char merkkiseksi
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

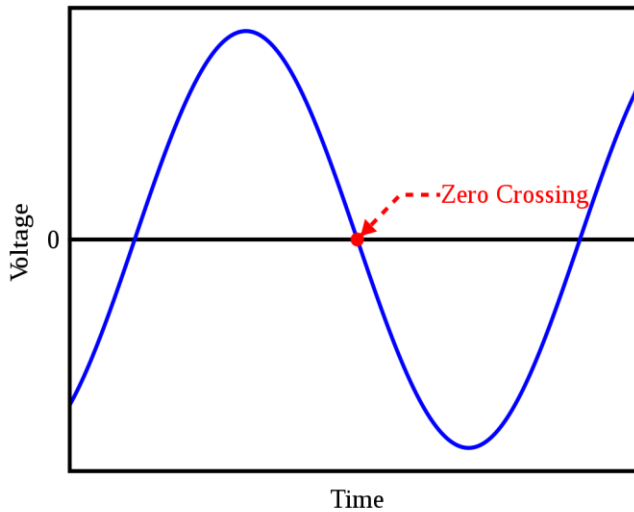
```

Tähän mennessä node kykenee ottamaan yhteyden lähiverkkoon sekä MQTT BROKER:iin. Lisäksi pystytään lähettämään ja vastaanottamaan viestejä.

4.7.5 Himmenninpiirin asennus

Tutkimuksessa käytettävä himmenninpiiri on himmennettävälle LED-valoille tarkoitettu moduuli. Moduulissa on zero-cross tunnistus, jolla himmennys saadaan

toimimaan kunnolla. Zero-cross ominaisuus lukee jännitteen siniaaltoja ja suorittaa keskeytysfunktion, kun siniaalto on 0 tilassa. Himmenninpiiriä ohjataan digitaalisesti (Kuva 17).



Kuva 17. Zero crossing.

Tarvittavat muuttujat sekä ohjelmat himmennin nodeen.

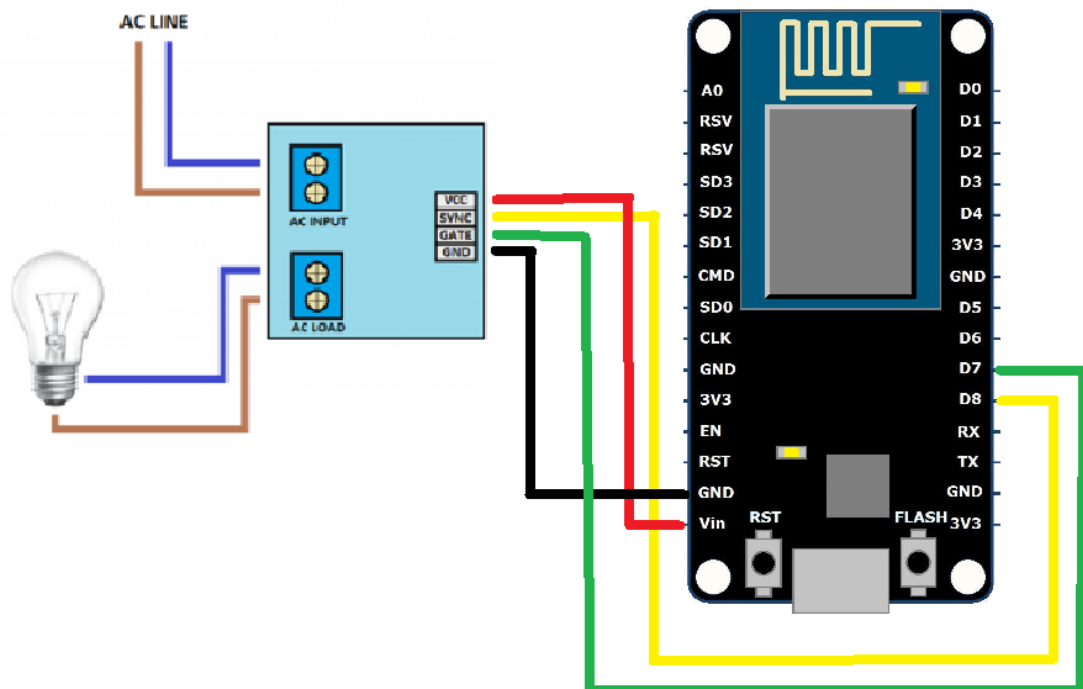
```
//Tarvittavat apumuuttujat
unsigned char AC_LOAD = D7; // Pinni jolla käynnistetään himmennys
unsigned char dimming = D8; // Pinni jolla säädetään kirkkautta
volatile boolean zero_cross = 0; // muuttuja joka kertoo onko zero cross ylitetty

void zero_crosss_int() // Funktio suoritetaan kun siniaalto on nolla tilassa
{
  int dimtime = (75 * dimming); // Määrätään himmennykseen käytettävä aika
  delayMicroseconds(dimtime); // Off cycle
  digitalWrite(AC_LOAD, HIGH); // kytetään kirkkauden kasvatus päälle
  delayMicroseconds(10); // Odotetaan tarvittava aika
  digitalWrite(AC_LOAD, LOW); // Kytetään himmennys pois päältä
}

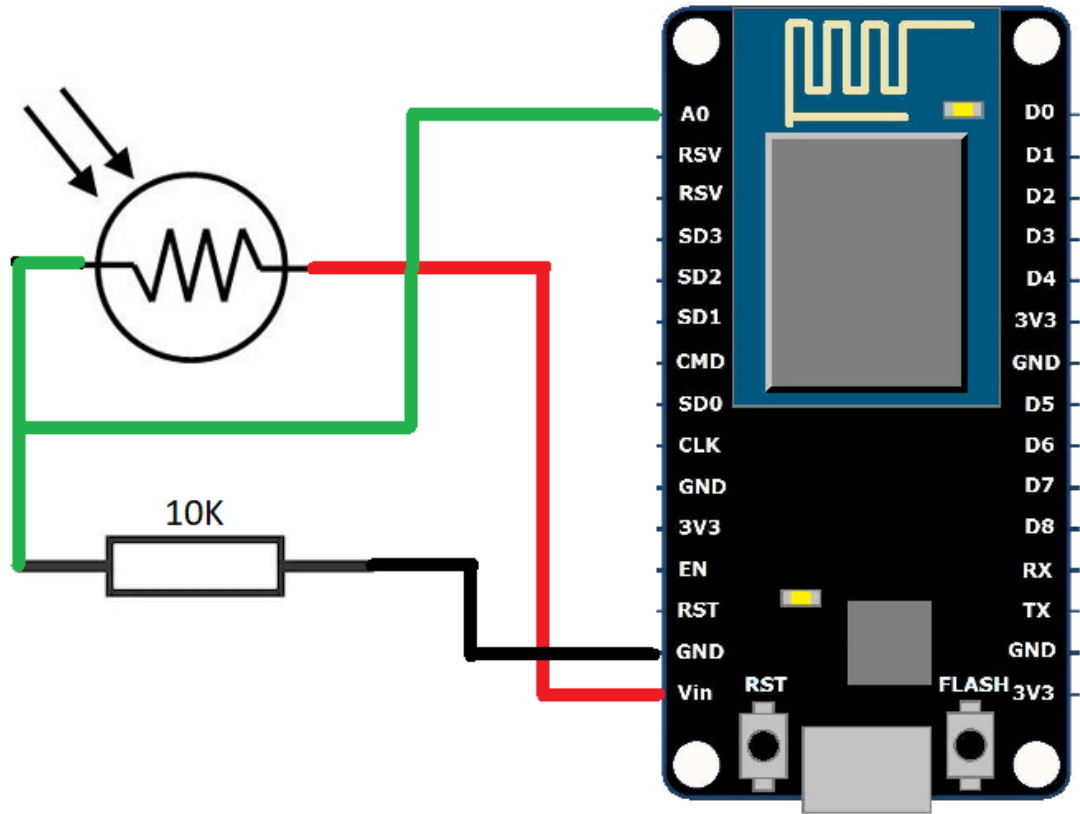
//Lisätään setup funktioon
pinMode(AC_LOAD, OUTPUT); // asetetaan AC Load pinni ulostuloksi
// Valitaan pinni joka tarkkailee zero crossia
attachInterrupt(15, zero_crosss_int, RISING);
```

4.7.6 Järjestelmäarkkitehtuuri

Järjestelmän kytkennät tehtiin käyttäen koekytkentälevyä sekä hyppylankoja. Himmennin node kytkiin himmennin moduuliin neljällä johdolla. Maa saadaan noden "GND" -pinnistä ja virta tulee USB:n kautta "Vin" -pinniin. Nämä pinnit kytetään moduuliin vastaaville paikoille. "D8" -pinnillä ohjataan himmennuksen tasoa ja zero-cross tieto tulee "D7" -pinniin (Kuva 18). Sensori noden kytkennässä käytetään 10 kilo ohmin resistoria ylösvetovastuksena jonka avulla varmistetaan pinniin A0 tulevan signaalin tila. Vastus kytketään maan ja photoresistorin välille. Photoresistorin arvo saadaan analogipinniin "A0", resistorin ja photoresistorin välistä. Virrat saadaan "Vin" -pinnistä (Kuva 19). Kummallakin nodella on langaton WiFi yhteys RasPi serveriin, jotta MQTT viestitys onnistuu. Serveri on yhteydessä lähiverkkoon ethernetkaapelilla ja serveriin sekä käyttöliittymään pääsee käsiksi millä tahansa lähiverkkoon kytketyllä laitteella.



Kuva 18. Himmennin noden kytkentäkaavio.



Kuva 19. Sensori noden kytkentäkaavio.

4.8 Käyttäjätarinoiden toteutus

Kun aloitin käyttäjätarinoiden rakentamisen, päätin aluksi muutaman vaatimuksen. Ensimmäisenä vaatimuksena oli, että mahdollinen käyttäjä voi olla kuka tahansa eli käyttäjätarinan käyttäjäkohdalla ei ole merkitystä. Päätin myös, että kaikki listattavat ominaisuudet tulee löytyä HASS-käyttöliittymästä. Käyttäjätarinoihin piti myös laatia "Definition of done" (DoD) eli tarkastuslista jonka avulla käyttäjätarinan valmiutta arvioidaan.

4.8.1 User Story 1: Valokatkaisija

"Käyttäjänä haluan, että HASS käyttöliittymässä on kytkin, jotta voin kytkeä valon päälle tai pois."

Definition of Done:

- Käyttöliittymässä on kytkin valon ohjaukseen
- Kytkimellä saa valon päälle
- Kytkimellä saa valon pois päältä

Kytkimen lisäämiseksi muokataan HASS konfiguraatio tiedostoa. Normaalin kytkimen ohjeet löytyvät HASS kotisivuilta, "components" välilehdeltä. Etsitään hakuksella "switch" ja koska viestintäprotokollana toimii MQTT, valitaan komponentti "MQTT switch". Sivulta löytyy ohjeet komponentin käyttöön ja lisäämiseen.

Alla olevalla koodinpätkällä lisätään kytkin HASS käyttöliittymään. Koodi lisätään "configuration.yaml" tiedostoon.

```

switch:
- platform: mqtt
  name: "valoKytkin"    #Nimetään kytkin
  #Valitaan aihe johon kytkin julkaisee tilansa
  command_topic: "koti/keittio/valo/kytkin"
  #Valitaan aihe josta kytkin lukee tilansa
  state_topic: "koti/keittio/valo/status"
  # Aihe johon LWT viesti julkaistaan
  availability_topic: "koti/keittio/valo/saatavuus"
  #Viesti jonka LWT julkaisee jotta toiminnallisuus toimii
  payload_available: "ONLINE"
  payload_not_available: "OFFLINE"
  #Viesti minkä kytkin julkaisee haluttuun aiheeseen
  #kytkin = päällä jolloin MQTT viesti = ON
  payload_on: "ON"
  #kytkin = pois päältä jolloin MQTT viesti = OFF
  payload_off: "OFF"
  # viesti jolla määrätään kytkimen tila
  state_on: "ON"
  state_off: "OFF"
  # Määrätään kytkimen lähettämä viesti tallennettavaksi
  retain: true

```

Jotta kytkin ilmestyy käyttöliittymään, tarvitaan vain kohdat "name" ja "command_topic". Muut kohdat tarvitaan myöhemmin ohjelmassa.

Nyt käyttöliittymästä löytyy kytkin, joka lähettää viestin "ON" tai "OFF" (Kuva 20) aiheeseen "koti/keittio/valo/kytkin". Kytkin lukee myös aiheesta "koti/keittio/valo/status" tulevia viestejä ja muuttaa tilaa niiden mukaan.



Kuva 20. Kytkimen tilat (on/off).

Seuraavaksi ohjelmoidaan himmennin node. Ensimmäisenä lisätään "callback" -funktion ominaisuus joka vastaanottaa tietystä aiheesta tulevia viestejä ja käsittelee niitä.

```

//Tarvittavat muuttujat
//Pinni D8 on kytketty himmentimen ohjaus pinniin
int dimming = D8; //Dimming muuttujalla säädetään valon kirkkaus 10 = 100% ja 127 = 0%

//callback funktio suoritetaan kun laitteen kuuntelemaan kanavaan tulee viesti
void callback(char* topic, byte* payload, unsigned int length) {
  String strPayload; //Muuttuja johon tallennetaan aihe josta viesti tulee
  for (int i = 0; i < length; i++) {
    strPayload.concat((char)payload[i]);
  }
  //IF looppia suoritetaan kun HASS kytkimeen asetewtusta aiheesta saapuu viesti
  if (String(command_topic).equals(topic)) {
    //Jos viestissä lukee "ON"
    if (strPayload.equals("ON")) {
      //Säädetään himmentimen tila maksimiin
      dimming = 10;
      //ja kutsutaan funktio
      publishLightState();
    }
    //Jos viestissä lukee "OFF"
    else if (strPayload.equals("OFF")) {
      //Säädetään himmentimen tila minimiin
      dimming = 127;
      //ja kutsutaan funktio
      publishLightState();
    }
  }
}

```

Funktio "publishLightState()" päivittää kytkimen tilan HASS-käyttöliittymään.

```

//Muuttuja johon on tallennettu haluttu aihe
const char* command_topic = "koti/keittio/valo/kytkin";

void publishLightState() {
  //Jos valon kirkkaus on jotain muuta kuin 127 eli valo on päällä
  if (dimming != 127) {
    //Luodaan viesti nimel "msg" jonka koko on 4 tavua ja viestissä lukee "ON"
    snprintf(msg, 4, "ON");
    //Viesti lähetetään aiheeseen jota käyttöliittymässä oleva kytkin kuuntelee
    //(Aihe johon viesti lähetetään, viesti, Retain -lippu)
    client.publish(state_topic, msg, true);
    Serial.print("[state_topic]");
    Serial.println(msg);
  }
  else {
    //Jos valon kirkkaus on 127 eli valo on pois päältä
    snprintf(msg, 4, "OFF");
    client.publish(state_topic, msg, true);
    Serial.print("[state_topic]");
    Serial.println(msg);
  }
}

```

Nyt kytkimeltä tulevat viestit luetaan ja säädetään lampun tila sen mukaan. Lisäksi kerrotaan vielä käyttöliittymän kytkimelle missä tilassa lamppu on.

4.8.2 User Story 2: Manuaalinen himmennys

”Käyttäjänä haluan, että HASS-käyttöliittymässä on liukukytkin, jotta voin säätää valon kirkkautta.”

Definition of Done:

- Käyttöliittymässä on liukukytkin valon säätöön
- Liukukytkin säätää valon kirkkautta

Liukukytkintä varten tarvitaan komponentti nimeltä ”input_number”. Lisäksi toiminnallisuuden parantamiseksi tarvitaan kaksi automaatiota. Komponentti lisätään kytkimen tavoin ”configuration.yaml” tiedostoon ja automaatio lisätään ”automations.yaml” tiedostoon.

Configuration.yaml lisäys:

```
input_number: #komponentti
  dimmslider: #valitaan tyyliksi liukukytkin
    name: Himmennys #nimetään se
    initial: 100 #Arvo mikä on käynnistyksessä
    min: 0 #Minimi arvo
    max: 100 #Maksimi arvo
    step: 1 #askeleen suuruus
    unit_of_measurement: step #Arvon määrä
```

Automations.yaml lisäys:

```
- alias: set dimm #Automaation nimi
  trigger: #Suoritetaan kun aiheeseen saapuu viesti
    platform: mqtt
    topic: 'koti/keittio/valo/kirkkaus/aseta' #aihe
  action:
    service: input_number.set_value #muutetaan input_numberin arvoa
    data_template:
      entity_id: input_number.dimmslider
      value: "{{ trigger.payload }}" #arvoksi muutetaan viestissä ollut arvo
      #Lähetetään liukukytkimen arvo haluttuun aiheeseen.
- alias: Temp slider moved #Automaation nimi
  trigger:
    platform: state
    entity_id: input_number.dimmslider
  action:
    service: mqtt.publish
    data_template:
      topic: "koti/keittio/valo/kirkkaus/aseta"
      retain: true
      payload: '{{ trigger.to_state.state | int }}'
```

Ensimmäisellä automaatiolla muutetaan liukukytkimen arvo vastaamaan lampun kirkkautta. Toisella automaatiolla otetaan liukukytkimen arvo ja lähetetään se viestinä aiheeseen "koti/keittio/valo/kirkkaus/aseta".

Nyt käyttöliittymässä on liukukytkin, joka lähettää arvon 0-100.(Kuva 21)



Kuva 21. Käyttöliittymän liukukytkin.

Seuraavaksi lisätään himmennin nodeen ominaisuus, jolla otetaan liukukytkimeltä tuleva arvo ja säädetään valon kirkkaus sen mukaan.

Lisäys tehdään "callback" -funktioon:

```
//kun vastaanotetaan viesti aiheesta
const char* brightness_command_topic = "koti/keittio/valo/kirkkaus/aseta";

else if (String(brightness_command_topic).equals(topic)) {
  //Vesti muutetaan integeriksi ja tallennetaan muuttujaan
  int brightness = strPayload.toInt();
  //muuttujan arvo 0-100 skaalataan 127-10
  //Eli jos arvo on 0 niin lopputulos on 127
  brightness = map(brightness, 0, 100, 127, 10);
  //skaalattu arvo annetaan dimming muuttujalle joka säättää kirkkautta
  dimming = brightness;
  //suoritetaan funktio joka ilmoittaa kyseisen kirkkausarvon käyttöliittymälle.
  publishDimm();
  publishLightState();
}
```

publishDimm() funktio lähettää kutsuttaessa "dimming" muuttujan arvon.

```
void publishDimm() {
  snprintf(msg, MSG_SIZE, "%d", dimming);
  client.publish(brightness_state_topic, msg, true);
  Serial.print("[brightness_state_topic]");
  Serial.println(msg);
}
```

Nyt saadaan käyttöliittymän liukukytkimen arvolla muutettua valon kirkkautta.

4.8.3 User Story 3: Automaattinen himmennys

”Käyttäjänä haluan, että HASS-käyttöliittymässä on kytkin, jotta voin kytkeä valon automaattisen himmennyksen päälle tai pois.”

Definition of Done:

- Käyttöliittymässä on Kytkin automaattisen himmennyksen ohjaukseen
- Kytkin kytkee automaattisen himmennyksen päälle
- Kytkin kytkee automaattisen himmennyksen pois päältä

Automaattisen himmentimen kytkin luodaan samalla tavalla kuin aikaisempi kytkin. Ensin tehdään sensori node. Nodessa on kiinni photoresistori joka mittaa valon arvoa.

Configuration.yaml

lisäys:

```
- platform: mqtt
  name: "autodimmKytkin"
  command_topic: "koti/keittio/valo/autodimm"
  state_topic: "koti/keittio/valo/autodimm/status"
  availability_topic: "koti/keittio/valo/saatavuus"
  payload_available: "ONLINE"
  payload_not_available: "OFFLINE"
  payload_on: "ON"
  payload_off: "OFF"
  state_on: "ON"
  state_off: "OFF"
  optimistic: false
  retain: true
```

Sensori noden koodi:

```
//Tarvittavat apumuuttujat
const int pResistor = A0; // Ilmoitetaan missä pinnissä photoresistori on
int photovalue = 0; //muuttuja johon photoresistorin arvo talletetaan
int setvalue = 0; //
unsigned long startMillis;
unsigned long currentMillis;

//Setup funktioon
startMillis = millis(); //Määritellään ohjelman alkamisaika

//Loop funktioon
currentMillis = millis(); //Tallennetaan tämänhetkinen aika
if(currentMillis - startMillis >= 60000) { //jos on kulunut minuutti
  photovalue = analogRead(pResistor); //Tallennetaan photoresistorin arvo
  publishLightValue(); //Lähetetään arvo
  startMillis = currentMillis; //muutetaan aloitusaika tämänhetkiseksi ajaksi
}

//publishLightValue() funktio
//Lähetetään valosensorin arvo himmentimelle
void publishLightValue() {
  snprintf(msg, MSG_SIZE, "%d", photovalue);
  client.publish("koti/keittio/sensor/valo/arvo", msg, true);
  Serial.print("[ValoArvo]");
  Serial.println(msg);
}
```

Nyt saadaan valosensorin arvo lähetettyä suoraan himmennin piirille.

Seuraavaksi himmennin noden ohjelmointi.

```
//Tarvittavat muuttujat
boolean autodimm = false; //muuttuja kertoo onko automaattinen himmennys päällä
unsigned long startMillis;
unsigned long currentMillis;
int minlight = 900; //halutun kirkkauden minimiarvo
int maxlight = 950; //halutun kirkkauden maksimiarvo
int lightValue = 0; //muuttujaan tallennetaan valosensorilta saatu arvo

//Jos automaattinen himmennys on päällä
else if (String(dimvalue_topic).equals(topic)) {
  if (autodimm) {
    //tallennetaan valosensorilta saatu arvo muuttujaan
    Serial.println("AUTODIMM:TRUE");
    lightValue = strPayload.toInt();
  }
  else{
    Serial.println("AUTODIMM:FALSE");
  }
}
//jos autohimmennys kytkintä painetaan
//muutetaan tilaa joko päälle tai pois
else if (String(autodimm_topic).equals(topic)) {
  if (strPayload.equals("ON")) {
    autodimm = true;
    //ilmoitetaan käyttöliittymälle himmennyksen tila
    publishDimmState();
  }
  else if (strPayload.equals("OFF")) {
    autodimm = false;
    publishDimmState();
  }
}
}
```

```

//publishDimmState() funktio
void publishDimmState() {
  if (autodimm) {
    snprintf(msg, MSG_SIZE, "ON");
    client.publish(dimmm_state_topic, msg, true);
    Serial.print("[dimmm_state_topic]");
    Serial.println(msg);
  }
  else {
    snprintf(msg, MSG_SIZE, "OFF");
    client.publish(dimmm_state_topic, msg, true);
    Serial.print("[dimmm_state_topic]");
    Serial.println(msg);
  }
}

//loop funktioon
currentMillis = millis(); //get the current time
if(autodimm){ //Jos automaattinen himmennys on päällä
  if (currentMillis - startMillis >= 200) { //suoritetaan 2 millisekunnin välein
//Kutsutaan "checkLight" funktiota jonka parametriksi asetetaan falosensorin arvo
    checkLight(lightValue);
    startMillis = currentMillis;
  }
}

//checklight funktio
void checkLight(int light){
//jos valoarvo on isompi kuin haluttu arvo eikä valo ole vielä sammunut
  if(light > maxlight && dimming != 127){
    Serial.println("YLI");
    //pienennetään kirkkautta
    dimming++;
  }
  //Jos valoarvo on pienempi kuin haluttu arvo eikä valo ole maksimi kirkkaudessa
  if(light < minlight && dimming != 10){
    Serial.println("ALI");
    //kasvatetaan kirkkautta
    dimming--;
  }
  if(light <= maxlight && light >= minlight){
    Serial.println("SOPIVA!");
  }
  Serial.print(dimming);
}
}

```

Acti
Go to

Nyt saadaan valosensorin arvon mukaan muutettua valon kirkkautta. Kirkkauden säätämiseen määrättiin haluttu huoneen kirkkausarvo, jonka jälkeen himmennystä kasvatetaan tai pienennetään, riippuen onko nykyinen kirkkaus yli vai alle halutun. Haluttu kirkkausarvo jaettiin kahteen osaan, maksimi- ja minimiarvot. Tämä helpottaa himmennuksen sulavuudessa, ettei valo muuta jatkuvasti kirkkautaan.

4.9 Yhteenveto

Taulukko 1. Käyttäjätarinat ja niiden definition of done suoritukset.

	Definition of Done:	
User Story 1: Valokatkaisija	Käyttöliittymässä on kytkin valon ohjaukseen	Tehty
	Kytkimellä saa valon päälle	Tehty
	Kytkimellä saa valon pois päältä	Tehty
User Story 2: Manuaalinen himmennys	Käyttöliittymässä on liukukytin valon säätöön	Tehty
	Liukukytin säätää valon kirkkautta	Tehty
User Story 3: Automaattinen himmennys	Käyttöliittymässä on Kytkin automaattisen himmennyksen ohjaukseen	Tehty
	Kytin kytkee automaattisen himmennyksen päälle	Tehty
	Kytin kytkee automaattisen himmennyksen pois päältä	Tehty

Käyttäjätarinat auttoivat tutkimuksen läpiviennissä valtavasti. Käyttäjätarinoiden antama tarkka kuvaus tarvittavista ominaisuuksista helpotti työn tekemistä ja kokonaisuuden näkemistä. Kaiken kaikkiaan käyttäjätarinoiden ominaisuudet saatiin tehtyä ja DoD kunkin tarinan kohdalla suoritettua (Taulukko 1).

Tutkimus sujui kaikin puolin hyvin vaikka ongelmia tuli lähes joka kohdassa. Käyttöympäristön asennus oli helppoa ja nopeaa hyvien dokumentointien ansiosta. Jokaiseen vaiheeseen ja asennukseen oli selkeät ohjeet ja pikkuongelmiin kuten salasanan resetoimiseen löytyi nopeasti apua. Suurin ongelma asennusvaiheessa oli se, kun Mosquitto ja HASS toimi molemmat RasPi:lla samassa IP-osoitteessa. En saanut Mosquitto -ohjelmistolla yhteyttä Home Assistantiin samalla IP-osoitteella, koska se oli jo käytössä. Yhteys piti ottaa localhost osoitteen

seen eli 127.0.0.1. HASS konfigurointi oli yksiselitteistä ja jälleen huolella dokumentoitua, eikä epäselvyyksiä sen suhteen ollut. Python pohjainen YAML-kirjoituskieli oli minulle uutta, joten sen kanssa tuli hieman virheitä. Nimesin kohtia väärin ja sen takia koko järjestelmä meni sekaisin. Löysin kuitenkin virheet nopeasti ja sain asian korjattua. Käyttöliittymän sai helposti juuri sellaiseksi kuin sen halusi ja ohjelmiston muu käyttö tuntui sujuvalta.

Nodejen kanssa ongelmia ilmeni hieman enemmän. Käytetyt kirjastot olivat minulle uusia eikä niiden toiminta ollut aivan suoraviivaista. Koska käytin useaa kirjastoa niin yhtä kaikenkattavaa ohjetta ei ollut, vaan minun piti käyttää hyväksi erilaisia foorumeita ja muiden käyttäjien esimerkkikoodeja. Lisäksi paljon aikaa kului uusien asioiden opiskeluun. Ensimmäinen ongelma ohjelmistossa oli "map"-funktio, joka siis muuttaa tietyn arvovälin haluttun kokoiseksi. Ongelmana oli se, että suoritin "map"-funktion ohjelman loopissa ilman viiveitä. Tästä seurasi se että "map"-funktio ei ehtinyt tekemään laskutoimituksia ennen kuin se kutsuttiin uudelleen, jolloin ohjelma kaatui. Muutin ohjelmaa siten että "map"-funktiota ei tarvinnut enää käyttää. Halutut arvot annettiin jo valmiiksi halutussa muodossa eikä niitä tarvinnut enää muuttaa. Toinen vaihtoehto olisi ollut sijoittaa "map"-funktio jonkun toisen funktion sisään ja lisätä sinne viive, jotta funktiolla olisi aikaa suoriutua. En valinnut tätä vaihtoehtoa sillä halusin välttää viiveiden käyttöä ohjelmassani.

Automaattisen himmennuksen kanssa ilmeni isoin pulma. Ensimmäisessä versiossa tein himmentimen logiikan siten että se sääti himmennuksen arvoa suoraan vertaamalla photoresistorin arvoa. Koska säädettävä valo on samassa huoneessa photoresistorin kanssa niin aina kun valon arvo muuttui, muuttui myös photoresistorin arvo. Tämän seurauksena ohjelma yritti jatkuvasti korjata itseään, jolloin valo kirkastui ja himmeni jatkuvasti eikä tasaista tilaa löytynyt. Ratkaisin tämän ongelman asettamalla halutun valoisuuden huoneeseen (photoresistorin arvo välillä 900-950) ja tekemällä funktion, joka tarkistaa tämänhetkisen photoresistorin arvon sekä säättää himmennystä sen mukaan, jotta arvo saataisiin oikeaksi. Nyt himmennys tuntuu luonnolliselta eikä valo enää vilku.

Uskon että ohjelmiston osista on mahdollista tehdä tehokkaampia ja ohjelmistokoodista olisi saanut pienemmän, jos kokemusta olisi ollut enemmän. Olen silti tyytyväinen omaan suoritukseen ja ohjelmiston toimintalogiikkaan. Mietin myös MQTT viestien lähetystä ja aiheita. Yhtenä vaihtoehtona oli, että lähetän viestejä vai yhteisen aiheeseen käyttäen "json" -muotoilua, jolla olisin saanut eri dataa yhteen viestiin. Tutkin asiaa ja tulin siihen lopputulokseen, että MQTT on niin kevyt protokolla, ettei nykyisessä suoritustavassa pitäisi olla mitään ongelmaa. Ohjelmistologiikan olisi myös voinut tehdä eri tavalla. Logiikan ja datan käsittelyn olisi voinut tehdä RasPi:n sisällä, koska siinä on enemmän laskentatehoa. Tällä olisi pienennetty nodejen kuormaa ja mahdollisesti saatu järjestelmästä tehokkaampi. Päädyin kuitenkin suorittamaan logiikan noden sisällä koska suuria laskutoimituksia tai funktioita ei suoriteta niin nodessa pitäisi riittää tehot tämänkokoiseen ohjelmaan. Ohjelmisto sisältää paljon keskeytysfunktioita ja niitä ei käsitellä hyvin. Jos kaksi keskeytystä sattuu samaan aikaan, niin ohjelmisto kaatuu. Pidemmällä käytöllä ilmenee ongelma, joka sammuttaa valon hetkeksi ja käynnistää sen uudelleen. Tällöin node suorittaa kevyen resetoinnin ja käynnistää itsensä uudelleen. Harvinaisempaan vikana ohjelmisto menee niin jumiin, että se kaataa koko noden. Tällöin vaaditaan manuaalinen resetointi.

Varsinaisia kehitysideoita ei tule mieleen. Ohjelmisto tekee halutun asia todella hyvin. Ohjelmiston keskeytyksiä voisi parantaa, jolloin ohjelma toimisi sujuvammin. Lisäksi automaattisen himmennuksen säätämistä voisi parantaa lisäämällä jonkinlaisen hystereesi funktion, jolloin himmennuksen raja-aroissa ei ilmene turhaa välkyntää tai ohjelmallista laskentaa.

5 Pohdinta

Ryynänen [7] kirjoittaa blogissaan siitä, miten jatkuva internetyhteys ja IoT-pilvipalvelut ovat oleellinen osa IoT-kokonaisuutta. Tutkimukseni kuitenkin todistaa, että toimivan järjestelmän ei tarvitse olla yhteydessä internetiin. Tietenkin suuremmissa järjestelmissä jatkuva internetyhteys tuo isoja hyötyjä käyttäjälle, mutta myös tietoturvariskit kasvavat. Jatkuva internetyhteys tai pilvipalvelut eivät siis mielestäni ole oleellista, kun puhutaan toimivasta IoT-järjestelmästä.

Atzori [2] puhuu kirjoituksessaan IoT-laitteiden ongelmista ja tarkemmin laitteiden luottamuksesta, yksityisyydestä ja turvallisuudesta. Jos IoT-järjestelmä pystytään rakentamaan irti verkosta, jäävät nämä ongelmakohdat täysin käyttäjän vastuulle. Tällöin kukaan ulkopuolinen ei pääse järjestelmään käsiksi ilman fyysistä yhteyttä. Home Assistant mahdollistaa siis IoT:n suurimpien ongelmien sivuuttamisen yksinkertaisella ratkaisulla. Mielestäni internetistä irti oleva järjestelmä on turvallisempi ja luotettavampi käyttää, koska tiedän tarkalleen, missä kaikki tieto sijaitsee. Uskon silti, että internet yhteyden tuomat edut kasvattavat järjestelmän käyttömukavuutta ja mahdollistavat parempien ominaisuuksien sisällyttämistä järjestelmään.

Mietin tutkimukseni aikana usein Maslown tarvehierarkiaa [9] ja sen yhteyttä älykkääseen kotiin. Kuten hierarkia kertoo, ovat ihmiselle tärkeimpiä fysiologiset tarpeet mutta se ei tarkoita, etteivätkö itsensä toteuttamisen tarpeet ole myös tärkeitä. Itseasiassa kaikki tarvehierarkian osiot ovat tärkeitä ihmiselle. Niinpä mukavaa kotia miettiessä mikä tahansa laite, joka auttaa ihmisiä missään tarvehierarkian kohdassa on suoraan positiivinen lisäys ihmisen elämään. Jos jokainen älykäs laite tuo jotain positiivista ihmisten elämään, tällöin älykkäällä kodilla voi olla suurikin vaikutus elämän jokaiseen osa-alueeseen.

Gill ym. [16] kirjoittaa kotiautomaation ongelmista ja siitä miten huono yhteensopivuus ja joustavuus kyseisellä tekniikalla on. Uskon, että Home Assistant ohjaa kotiautomaatiota parempaan suuntaan ratkaisemalla edellä mainittuja ongelmia ja rakentamalla mahdollisimman monipuolisen ja joustavan käyttöliittymän kaikkien saataville. Mielestäni älykodin hyötyjä olisi syytä tuoda enemmän, jotta sen

yleistyminen ja kehitys voisi kasvaa entistä nopeammin. Myös niin sanotun älykodin saatavuus on syytä saada mahdollisimman helpoksi. Kun laitteiden hinnat ja käyttömukavuus saadaan sopivaksi, on ihmisillä pienempi kynnyks tutustua aiheeseen ja ottaa se käyttöön omassa elämässään.

Home Assistantissa ei tällä hetkellä ole varmaa yhdistettävien laitteiden maksimirajaa. Joillain käyttäjillä on jopa 1000 laitetta yhdistettynä ja toiminnassa. Laitemäärän pullonkaulana toimii käytössä oleva laitteisto, kuten wifi-reitittimet tai serverin suoritusteho. Wifi-verkkoon voidaan kytkeä teoriassa noin 250 laitetta, mutta käytössä voi olla useampia wifi-reitittimiä [19]. RF-laitteiden määrällä ei ole varsinaista rajaa, mutta tiedonsiirron viive laitteiden välillä kasvaa mitä useampi laite toimii samalla taajuudella. Jos järjestelmä toimii yhdessä taloudessa niin uskon, että normaali järjestelmä pystyy sisältämään kaikki talouden älylaitteet. Jos järjestelmän on tarkoitus toimia esimerkiksi jonkin yrityksen tiloissa, jossa yhdistettäviä laitteita saattaa olla huomattavasti enemmän, on syytä miettiä tarkemmin järjestelmän kommunikointimahdollisuuksia ja -rajoitteita. Myös käytössä ollut MQTT-protokolla on rajallinen yhteyksien suhteen, mutta niin sanottuja "BROKEREITA" voi olla useampi, jolloin kyseinen ongelma häviää. Joissakin laitteissa on mahdollista ketjuttaa tiedonsiirtoa samanlaiselta laitteelta toiselle, jolloin useampi laite voi viedä vain yhden laitteen tiedonsiirtoresurssin.

Ohjelmiston käyttöliittymässä ei mielestäni ollut mitään suuria muutoksentarpeita. Montaa eri asiaa voisi kuitenkin helpottaa. Esimerkiksi konfiguraatitiedoston muokkaaminen on hieman hankalaa, sillä tiedostoon ei pääse käyttöliittymästä käsiksi. Tähän on olemassa apuohjelma, joka tuo konfiguraatitiedoston selaimen. Home Assistantin uusin päivitys (0.90) mahdollistaa konfiguraatitiedoston muokkaamisen suoraan käyttöliittymästä ja vieläpä käyttäjäystävällisellä tavalla. Home Assistantiin on saatavilla paljon sen käyttöön liittyviä kolmannen osapuolen apuohjelmia ja usein nämä integroidaan käyttöliittymään uudemmissa päivityksissä. Viimeinen asia joka mielestäni parantaisi käyttöliittymää, sekä itse ohjelmistoa on laitteiden lisääminen. Mitä enemmän laitteita Home Assistant tukee, sitä monipuolisemmat käyttömahdollisuudet sillä on.

Älykoti ei ole vain kasa IoT -laitteita, jotka antavat käyttäjälleen tietoa. Älykkyys perustuu mahdollisuuteen kerätä ja analysoida tietoa sekä suorittaa tehtäviä sen perusteella automaattisesti. Jatkotutkimuksissa olisikin hyvä keskittyä IoT-järjestelmän automatisointiin sekä sen tuomiin hyötyihin. Parhaassa tapauksessa käyttäjän ei tarvitse tehdä mitään vaan järjestelmä hoitaa tarvittavat asiat automaattisesti. Uskon, että tekoäly sekä koneoppiminen tuovat tälle alueelle uusia mahdollisuuksia. Myös IoT-järjestelmän tietoturva on erittäin tärkeää ja sitä on syytä tutkia tarkemmin. Se miten tietoturvaa voidaan käyttää vastaavassa järjestelmässä vaikuttaa suoraan sen luotettavuuteen ja sitä kautta käyttäjämäärään. Uskon että lohkoketjuteknologiasta voisi olla suuri hyöty tietoturvan parantamiseen. Tutkimukseni kohdistui yksityisasunnon IoT-järjestelmään, mutta olisi mielenkiintoista nähdä, miten vastaava järjestelmä sopeutuisi esimerkiksi koko taloyhtiön käyttöön tai vastaavasti ison yrityksen toimitiloihin.

6 Lähteet

- [1] R. v. d. Meulen, "Gartner," 2017. [Online]. Saatavissa: <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>. [Viitattu 15.04.2019].
- [2] L. I. A. & M. G. Atzori, The internet of things: A survey. Computer networks, 2010.
- [3] D. Minoli, Building the Internet of Things with IPv6 and MIPv6 : The Evolving World of M2M Communications, Somerset, NJ, USA: John Wiley & Sons, 2013.
- [4] M. N. R. H. & P. N. R. Alam, The evolution of M2M into IoT. Communications and Networking (BlackSeaCom), 2013.
- [5] itewiki. [Online]. Saatavissa: <https://www.itewiki.fi/opas/iot-ja-teollinen-internet/>. [Viitattu 15.04.2019].
- [6] A. Sormunen, "Mikä ihmeen IoT?," 2016. [Online]. Saatavissa: <https://blogit.mpy.fi/mik%C3%A4-ihmeen-iot/>. [Viitattu 15.04.2019].
- [7] T. Ryyänen, "Internet of things selkokielellä," 2016. [Online]. Saatavissa: <https://blogit.haaga-helia.fi/ryynanen/2016/02/29/mita-internet-of-things-voitarkoittaa-selkokielella/>. [Viitattu 15.04.2019].
- [8] R. H. Weber, Internet of Things – New security and privacy challenges, COMPUTER LAW & SECURITY REPORT, 2010.
- [9] A. H. Maslow ja R. Frager, Motivation and personality, New York: Harper and Row, 1987.
- [10] J. Lahtinen ja A. Isoviita, Asiakaspalvelun Ja Markkinoinnin Perusteet, Avaintulos Oy, 2001.
- [11] R. J. & K. T. Robles, "Applications, systems and methods in smart home technology: a review," 2010.
- [12] F. & F. C. Mattern, "From active data management to event-based systems and more.," tekijä: *From the Internet of Computers to the Internet of Things.*, 2010, pp. 242-259.
- [13] J. Helin, "Esineiden Internet Luo Kodistasi Älykkään," Jyväskylän yliopisto, Jyväskylä, 2015.
- [14] K. MUNDLE, "Toptal," [Online]. Saatavissa: <https://www.toptal.com/designers/interactive/smart-home-domestic-internet-of-things>. [Viitattu 15.04.2019].
- [15] C. & P. J. Gomez, "Wireless home automation networks: A survey of architectures and technologies," *IEEE Communications Magazine*, osa/vuosik. 6/2010, nro 48, pp. 92 - 101, 2010.

- [16] K. Y. S. Y. F. & L. X. Gill, "A zigbee-based home automation system.," *Consumer Electronics, IEEE*, osa/vuosik. 2009, nro 55, pp. 422-430, 2009.
- [17] M. Starsinic, "System architecture challenges in the home M2M network.," tekijä: *Applications and Technology Conference*, Long Island, 2010.
- [18] E. L. K. & S. A. Kasanen, "Konstruktiivinen tutkimusote liiketaloustieteessä," *Liiketaloudellinen Aikakauskirja*, nro 3, pp. 305-308, 1991.
- [19] B. Mitchell, "Lifewire," 2018. [Online]. Saatavissa: <https://www.lifewire.com/how-many-devices-can-share-a-wifi-network-818298>. [Viitattu 15.04.2019].

Liite 1. Samba-configuraatio

[global]

netbios name = Raspi

server string = The Pi File Center

workgroup = WORKGROUP

hosts allow =

socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=65536

SO_SNDBUF=65536

remote announce =

remote browse sync =

[HOMEPI]

path = /home/pi

comment = No comment

browsable = yes

read only = no

valid users =

writable = yes

guest ok = yes

public = yes

create mask = 0777

directory mask = 0777

force user = root

force create mode = 0777

force directory mode = 0777

hosts allow =

[HOME ASSISTANT]

path = /var/opt/homeassistant

comment = No comment

browsable = yes

read only = no

valid users =

writable = yes

guest ok = yes

public = yes

create mask = 0777

directory mask = 0777

force user = root

force create mode = 0777

force directory mode = 0777

hosts allow =

Liite 2. Himmennin noden-koodi

```
//Lisätään tarvittavat kirjastot
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Määritellään tarvittavat käyttäjänimet ja salasanat
const char* ssid = "Samun_valtakunta"; //WiFi verkon nimi
const char* password = "asdasdasd"; //Verkon salasana
const char* mqtt_server = "192.168.0.102"; //Mosquitto serverin osoite
const char* clientId = "DimmerClient"; //Käyttäjän nimi
const char* userName = "pi"; //MQTT käyttäjänimi
const char* passWord = "lasdf123"; //ja salasana

//Himmentimen muuttujat

unsigned char AC_LOAD = D7; // Output to Opto Triac pin
unsigned char dimming = D8; // Dimming level (0-100)
volatile int i = 0; // Variable to use as a counter volatile as it is in an interrupt
volatile boolean zero_cross = 0; // Boolean to store a "switch" to tell us if we have crossed
zero
boolean autodimm = false;
unsigned long startMillis;
unsigned long currentMillis;

int minlight = 900;
int maxlight = 950;
int lightValue = 0;
bool sended;
int dc = 0;

//luodaan wificlientti sekä PubSub clientti
WiFiClient DimClient;
PubSubClient client(DimClient);
//tarvittavat apumuuttujat
const uint8_t MSG_SIZE = 20;
char msg[MSG_SIZE];

const char* command_topic = "koti/keittio/valo/kytkin";
const char* brightness_command_topic = "koti/keittio/valo/kirkkaus/aseta";
const char* state_topic = "koti/keittio/valo/status";
const char* brightness_state_topic = "koti/keittio/valo/kirkkaus";
const char* willTopic = "koti/keittio/valo/saatavuus";
const char* autodimm_topic = "koti/keittio/valo/autodimm";
const char* dimmvalue_topic = "koti/keittio/sensor/valo/arvo";
const char* dimm_state_topic = "koti/keittio/valo/autodimm/status";
// Wifi yhteyden muodostus

void setup_wifi() {
```

```

// yhdistetään wifiverkkoon annetulla käyttäjätunnuksella sekä salasanalla
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

//odotetaan kunnes yhteys on muodostettu
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

randomSeed(micros());
// jos yhteys muodostetaan ilmoitetaan laitteen oma ip osoite
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
Serial.println();
Serial.print("MAC: ");
Serial.println(WiFi.macAddress());
}

//callback funktio suoritetaan kun laitteen kuuntelemaan kanavaan tulee viesti
void callback(char* topic, byte* payload, unsigned int length) {
  String strPayload; //Muuttuja johon tallennetaan aihe josta viesti tulee
  for (int i = 0; i < length; i++) {
    strPayload.concat((char)payload[i]);
  }
  //IF looppia suoritetaan kun HASS kytkimeen asetewtusta aiheesta saapuu viesti
  if (String(command_topic).equals(topic)) {
    //Jos viestissä lukee "ON"
    if (strPayload.equals("ON")) {
      //Säädetään himmentimen tila maksimiin
      dimming = 10;
      //ja kutsutaan funktio
      publishLightState();
    }
    //Jos viestissä lukee "OFF"
    else if (strPayload.equals("OFF")) {
      //Säädetään himmentimen tila minimiin
      dimming = 127;
      //ja kutsutaan funktio
      publishLightState();
    }
  }
}

//kun valoarvoa säädetään
else if (String(brightness_command_topic).equals(topic)) {
  int brightness = strPayload.toInt();
  brightness = map(brightness, 0, 100, 127, 10);
  dimming = brightness;
  publishDimm();
}

```

```

    publishLightState();
}
//Valosensorin arvo
else if (String(dimmmvalue_topic).equals(topic)) {
    if (autodimm) {
        Serial.println("AUTODIMM:TRUE");
        lightValue = strPayload.toInt();
    }
    else {
        Serial.println("AUTODIMM:FALSE");
    }
}
//jos autohimmennys kytkintä painetaan
else if (String(autodimm_topic).equals(topic)) {
    if (strPayload.equals("ON")) {
        autodimm = true;
        publishDimmState();
    }
    else if (strPayload.equals("OFF")) {
        autodimm = false;
        publishDimmState();
    }
}
}

void zero_crosss_int() // function to be fired at the zero crossing to dim the light
{
    int dimtime = (75 * dimming); // For 60Hz =>65
    delayMicroseconds(dimtime); // Off cycle
    digitalWrite(AC_LOAD, HIGH); // triac firing
    delayMicroseconds(10); // triac On propagation delay (for 60Hz use 8.33)
    digitalWrite(AC_LOAD, LOW); // triac Off
}

void publishDimm() {
    snprintf(msg, MSG_SIZE, "%d", dimming);
    client.publish(brightness_state_topic, msg, true);
    Serial.print("[brightness_state_topic]");
    Serial.println(msg);
}

void publishLightState() {
    if (dimming != 127) {
        snprintf(msg, 4, "ON");
        client.publish(state_topic, msg, true);
        Serial.print("[state_topic]");
        Serial.println(msg);
    }
    else {
        snprintf(msg, 4, "OFF");
        client.publish(state_topic, msg, true);
        Serial.print("[state_topic]");

```

```

    Serial.println(msg);
  }
}

void publishAvailability() {
  snprintf(msg, MSG_SIZE, "ONLINE");
  client.publish(willTopic, msg, true);
  Serial.print("[willTopic]");
  Serial.println(msg);
}

void publishDimmState() {
  if (autodimm) {
    snprintf(msg, MSG_SIZE, "ON");
    client.publish(dimmm_state_topic, msg, true);
    Serial.print("[dimmm_state_topic]");
    Serial.println(msg);
  }
  else {
    snprintf(msg, MSG_SIZE, "OFF");
    client.publish(dimmm_state_topic, msg, true);
    Serial.print("[dimmm_state_topic]");
    Serial.println(msg);
  }
}

void checkLight(int light) {
  if (light > maxlight && dimming != 127) {
    Serial.println("YLI");
    dimming++;
  }
  if (light < minlight) {
    Serial.println("ALI");
    if (dimming != 10) {
      dimming--;
    }
  }
  if (light <= maxlight && light >= minlight) {
    Serial.println("SOPIVA!");
  }
  Serial.print(dimming);
  Serial.print("---DC:");
  Serial.println(dc);
}

void setup() {
  // put your setup code here, to run once:
  pinMode(AC_LOAD, OUTPUT); // asetetaan AC Load pinni ulostuloksi
  attachInterrupt(15, zero_crosss_int, RISING); // Valitaan pinni joka tarkkailee zero
  crossia
  Serial.begin(115200); //Avataan sarjaportti testaamista varten
}

```

```

setup_wifi(); //Yhdistetään wifi
client.setServer(mqtt_server, 1883); //Yhdistetään MQTT
client.setCallback(callback); //Asetetaan MQTT kuuntelija
}

void loop() {
  if (!client.connected()) { //Tarkistaa jatkuvasti mqtt yhteyden
    reconnect();
  }
  client.loop(); //Pitää MQTT viestijä silmällä
  currentMillis = millis(); //get the current time
  if (autodimm) {
    if (currentMillis - startMillis >= 200) { //test whether the pertime of the current LED
      brightness
      checkLight(lightValue);
      startMillis = currentMillis; //IMPORTANT to save the start time of the current LED
      brightness
    }
  }
}

void reconnect() {
  // Muodostetaan wifi yhteyttä uudelleen
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    dc++;
    // Jos yhteyttä ei saada luodaan uusi sattumanvarainen clientID
    String clientId = "Dimmer-";
    clientId += String(random(0xffff), HEX);
    // Yritetään yhdistää uudelleen
    if (client.connect(clientId.c_str(), willTopic, 1, 1, "OFFLINE")) {
      Serial.println("connected");
      // Kun yhteys on muodostettu lähetetään jokaiseen kanavaan viesti
      publishAvailability();
      publishLightState();
      publishDimmState();
      publishDimm();

      // ... ja ruvetaan kuuntelemaan haluttuja kanavia
      client.subscribe(command_topic);
      client.subscribe(autodimm_topic);
      client.subscribe(dimmvalue_topic);
      client.subscribe(brightness_command_topic);
    } else { //Valitaan jos yhteyttä ei saada
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Joes yhteyttä ei saatu yritetään uudelleen 5 sekunnin kuluttua
      delay(5000);
    }
  }
}
}

```


Liite 3. Sensori noden-koodi

```
//Lisätään tarvittavat kirjastot
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Määritellään tarvittavat käyttäjänimet ja salasanat

const char* ssid = "Samun_valtakunta";
const char* password = "asdasdasd";
const char* mqtt_server = "192.168.0.102";
const char* clientId = "LightSensorClient";
const char* userName = "pi";
const char* passWord = "lasdf123";

//luodaan wificlientti sekä PubSub clientti
WiFiClient LightSensorClient;
PubSubClient client(LightSensorClient);

const int pResistor = A0; // Photoresistor at Arduino analog pin A0
int photovalue = 0;
int setvalue = 0;
unsigned long startMillis;
unsigned long currentMillis;

//tarvittavat apumuuttujat
const uint8_t MSG_SIZE = 20;
char msg[MSG_SIZE];

// Wifi yhteyden muodostus
void setup_wifi() {
  Serial.print("TESTI!");
  delay(10);
  // yhdistetään wifiverkkoon annetulla käyttäjätunnuksella sekä salasanalla
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  //odotetaan kunnes yhteys on muodostettu
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  // jos yhteys muodostetaan ilmoitetaan laitteen oma ip osoite
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
```

```

Serial.println(WiFi.localIP());
Serial.print("MAC: ");
Serial.println(WiFi.macAddress());
}

// function called when a MQTT message arrived
void callback(char* p_topic, byte* p_payload, unsigned int p_length) {
}

void reconnect() {
  // Muodostetaan wifi yhteyttä uudelleen
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Jos yhteyttä ei saada luodaan uusi sattumanvarainen clientID
    String clientId = "LightSensor-";
    clientId += String(random(0xffff), HEX);
    // Yritetään yhdistää uudelleen
    if (client.connect(clientId.c_str(), "koti/keittio/sensor/saatavuus", 1, 1, "OFFLINE")) {
      Serial.println("connected");
      // Kun yhteys on muodostettu lähetetään jokaiseen kanavaan viesti
      publishLightValue();
      publishAvailability();
    } else { //Valitaan jos yhteyttä ei saada
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Jos yhteyttä ei saatu yritetään uudelleen 5 sekunnin kuluttua
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup_wifi(); //Yhdistetään wifi
  client.setServer(mqtt_server, 1883); //Yhdistetään MQTT
  client.setCallback(callback);
  startMillis = millis(); //initial start time
}

void loop() {

  if (!client.connected()) { //Tarkistaa jatkuvasti mqtt yhteyden
    reconnect();
  }
  client.loop(); //Pitää MQTT viestijä silmällä

  currentMillis = millis(); //get the current time
  if(currentMillis - startMillis >= 2000) { //60000test whether the period has elapsed

```

```
    photovalue = analogRead(pResistor);
    publishLightValue();
    startMillis = currentMillis; //IMPORTANT to save the start time of the current LED
    brightness
  }
}
```

```
void publishLightValue() {
  snprintf(msg, MSG_SIZE, "%d", photovalue);
  client.publish("koti/keittio/sensor/valo/arvo", msg, true);
  Serial.print("[ValoArvo]");
  Serial.println(msg);
}
```

```
void publishAvailability() {
  snprintf(msg, MSG_SIZE, "ONLINE");
  client.publish("koti/keittio/sensor/saatavuus", msg, true);
  Serial.print("[Availability]");
  Serial.println(msg);
}
```