KARELIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutusohjelma

Samu Kainulainen

ÄLYKKÄÄN VALAISTUKSEN TOTEUTUS AVOIMILLA TYÖKALUILLA

Opinnäytetyö
Toukokuu 2019

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2019
Tieto- ja viestintätekniikan koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Samu Kainulainen

Nimeke
Älykkään valaistuksen toteutus avoimilla työkaluilla

Toimeksiantaja
Tapio Kainulainen

Tiivistelmä

Projektin tarkoituksena oli suunnitella ja toteuttaa älykäs valaistusjärjestelmä omakotitalon alakertaan hyödyntäen avoimen lähdekoodin työkaluja. Järjestelmä tuli suunnitella tulevaisuutta ja laajennettavuutta silmällä pitäen.

Projekti eteni seuraavasti: Ensin suunniteltiin arkkitehtuuri, jolla järjestelmä toteutettaisiin. Tähän kuuluvat Arduinot, Raspberry Pi, releet sekä optoerottimet. Tämän jälkeen luotiin tarvittava koodi sekä opeteltiin sopivan IoT-alustan käyttö. Viimeisessä vaiheessa järjestelmä asennettiin tarvittavien johdotusten kera. Tässä viimeisessä vaiheessa korjattiin virheitä koodissa sekä vaihdettiin aiemmin valittu IoT-alusta sopivampaan.

Projektin lopputuloksena saavutettiin toimiva älykodin valaistusjärjestelmä, jota on helppo ryhtyä laajentamaan myös uusille alustoille. Kuten oletettua, IoT-alustan vaihdos tuotti ongelmia, sillä kaikki edellisen alustan ratkaisut eivät sopineet tälle uudelle alustalle. Kuitenkin nämä puuttuvat ratkaisut pystyttiin tuottamaan ilman suurempia ongelmia, ja huolimatta pienistä viivästyksistä, järjestelmä saatiin toimintakuntoon aikataulussaan. Tuotettu järjestelmä toimii pohjana tuleville päivityksille.

Kieli	Sivuja	43
suomi	Liitteet	9
	Liitesivumäärä	32

Asiasanat
opinnäytetyö, opinnäytetyön prosessi, IoT, kotiautomaatio, home assistant, valaistus, älykoti

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2019
Degree Programme in Information and Communication technology

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Author (s)
Samu Kainulainen

Title
Creating a Smart Lighting System with Open-Source Resources

Commissioned by
Tapio Kainulainen

Abstract

This project was designed to create a smart lighting solution using open source software. This installation covered a living room located in a basement of a detached house. System needed to be future-proofed in terms of storage, expandability and processing power.

The project was carried out as follows: first, the architecture was planned, consisting major parts like Arduinos, Raspberry Pi, relays and optocouplers. Then, required code was created and suitable IoT platform was studied. In the last phase, system was installed along with all the needed wiring. In this last step, multiple bugs were fixed and IoT platform was changed to more suitable one.

The outcome of the project was a working smart home system with good expandability across multiple platforms. As expected, the IoT platform change did cause some issues that required correcting, since although many of the features from previous platform could be transferred, some custom-made features needed to be made from the beginning. Despite some minor setbacks to the schedule, system achieved functional state, which works as a basis for the upcoming improvements.

Language	Pages	43
Finnish	Appendices	9
	Pages of Appendices	32

Keywords
thesis, thesis process, IoT, home automation, home assistant, lighting, smart home

Sisältö

Liitteet

Käsitteet

Avoin lähdekoodi	Vapaasti saatavissa ja muokattavissa oleva ohjelmisto, usein käyttäjäryhmän yhteisvoimin kehittämä.
Arduino	Avoimen laitteiston mikro-ohjain ja ohjelmointiympäristö.
Avr	Atmel-valmistajan mikrokontrolleriperhe.
Big data	Tarkoittaa suuria, järjestelemättömiä tietomassoja, jotka muuttuvat nopeasti ja jotka vanhenevat nopeasti.
Broker	MQTT-viestinvälityspalvelin.
Home Assistant	Avoimen lähdekoodin kotiautomaatiojärjestelmä.
IoT	Esineiden internet, tarkoittaa internetiin liitettyjä fyysisiä laitteita, jotka lähettävät tai vastaanottavat automaattisesti dataa ja mahdollistavat esimerkiksi automaattisen laitteiden hallinnan.
IoT-alusta	IoT-laitteita ja niiden hallintaa varten kehitetty järjestelmä, mahdollistaa esimerkiksi datan visualisoinnin.
Konfigurointi	Asetusten säätäminen.
Kotiautomaatio	Kodin laitteiston automaattinen ohjaus.
Mikrokontrolleri	Mikropiiri, joka sisältää mikroprosessorin sekä muisti- ja liityntälohkoja. Ohjelmoitavissa.
MQTT	Lähetys- ja vastaanottopohjainen viestiprotokolla.
NAS	Network Access Storage, eli verkkoon kytketty levypalvelin.
Pilvipalvelu	Internetissä sijaitseva tiedon varastointi- tai laskentapalvelu.
Raspberry Pi	Yhden piirilevyn tietokone, joka perustuu ARM-mikroprosessoriarkkitehtuuriin.
Repository	Ohjelmavarasto, usein internetissä.
Shield	Lisälaite, joka liitetään Arduinon päälle sen pinneihin.

Skripti	Tehtävien automatisointiin tarkoitettu komentokieli, kevyempi kuin ohjelmointikieli.
SSH	Secure Shell, protokolla salattuun tietoliikenteeseen.
Thingsboard	Avoimen lähdekoodin IoT -alusta.
Topic	Viestien organisointihierarkia.
Älykoti	Kotiautomaatiolla toteutettu ympäristö, joka helpottaa arkea ja parantaa talon käytettävyyttä.

1 Johdanto

Esineiden internet (IoT) on käsitteenä ollut olemassa jo 1990-luvulta lähtien, mutta suuremman väestön tietoisuuteen se on löytänyt tiensä vasta viimeisen vuosikymmenen aikana. Internetiin yhdistettyjen laitteiden määrä kasvaa vuosi vuodelta lähes eksponentiaalisesti, ja tällä hetkellä IoT-laitteita onkin noin 3.5 -kertaisesti ihmisiin verrattuna. [1.]

Suuri osa laitteista on mobiililaitteita kuten tabletteja ja puhelimia, joissa esineiden internetiä hyödynnetään kuluttajia seuraamalla, eli niin sanotun Big Datan keräyksessä analysointia varten. Kuitenkin suuri osa IoT:n potentiaalista on vielä hyödyntämättä. Yritykset ja valtiot heräilevät mahdollisiin etuihin, joita voidaan saavuttaa tämän avulla, mutta vielä on yksi osa-alue, jonka ensiaskeleita tämän uuden teknologian suhteen olemme päässeet seuraamaan vasta viimeisen vuosikymmenen aikana: ihmisten arki ja eläminen eli älykodit.

Jo aikaa ennen IoT:n soveltamista on ollut erilaisia tapoja lisätä "älykkyyttä" taloon, muun muassa hallintapaneeleilla, mutta tämä viimeisin aalto mahdollisuuksineen on kuitenkin aivan toista luokkaa. Googlatessa "älykoti" saa tulokseksi kirjavan valikoiman ehdotuksia valaistuksesta, kodinkoneista, musiikkilaitteista sekä kodin turvalaitteista. Näillä on kuitenkin edessään sama ongelma kuin aikaisemman aallon tuotteissa: monimutkaisuus. Monella laitteella on nettisivua, mobiilisovellusta ja kirjautumistunnusta laitteen hallitsemiseen ja seurantaan, sillä yritykset kilpailevat näillä keskenään. Tästä kärsii eniten loppukäyttäjä, joka turhautuu sovellusten ja hallintapaneelien viidakkoon. Varoittavana esimerkkinä voimme lukea artikkeleita huonoista kokemuksista, kun valon vaihtaminen viekin yhden painalluksen sijasta viisi, ja koko ominaisuus jää tämän takia käyttämättä. Tähän on kuitenkin ratkaisu ympäri maailmaa työskentelevien, avoimen lähdekoodin järjestelmiä kehittävien ohjelmoijien ansiosta.

Älykotia toteuttaessa IoT ja kotiautomaatio kulkevat käsi kädessä, ja tämän tiimoilta onkin käynnistynyt muun muassa seuraavia projekteja:

- Thingsboard
- OpenHAB
- Home Assistant
- Calaos
- Domoticz
- MisterHouse
- Pytomation
- EventGhost
- IoBroker
- Pimatic
- OpenMotics.

Ensimmäiset näistä ovat alkaneet vuonna 2013 ja uusia projekteja käynnistyy koko ajan lisää.

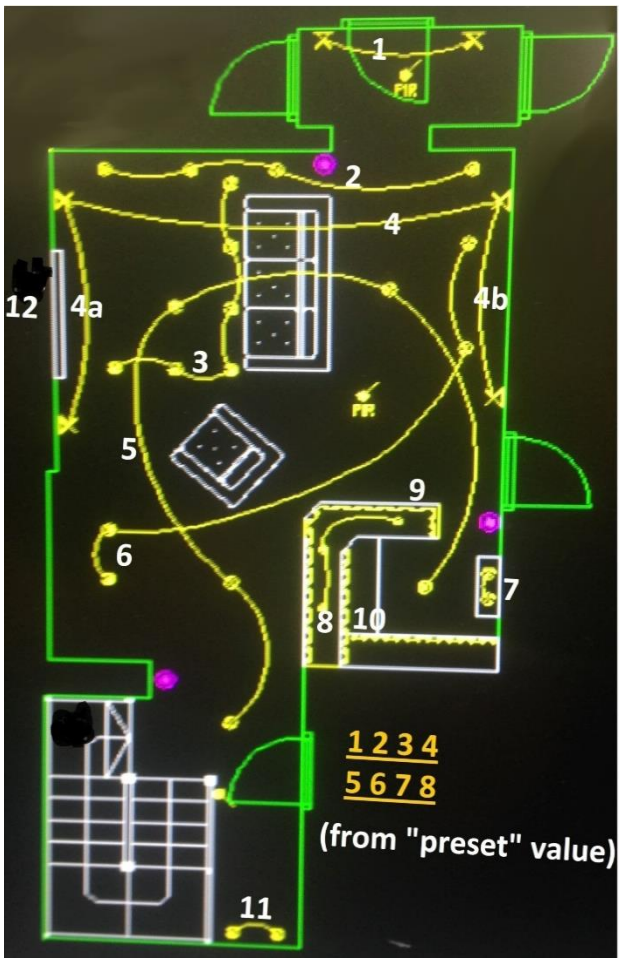
Älykkään kodin toteuttaminen tällaisella järjestelmällä eroaa merkittävästi aikaisempana mainittuun monen ratkaisun sekasotkuun. Kotiautomaatiojärjestelmät tukevat lukuisia laitteita ja protokollia, ja juuri nämä liitäntämahdollisuudet tekevät niistä verrattoman arvokkaan työkalun älykkään kodin toteuttamisessa.

Opinnäytetyöni alkoi saada muotonsa kahden viimeisen opintovuoteni aikana. Vanhempieni taloon oli kaavailtu ohjauspaneelilla säädettäviä valoja, mutta remontointi muissa osissa taloa vei kiireellisempänä huomion tältä ominaisuudelta. Kun keskustelussa tuli ilmi oma kiinnostukseni IoT-ratkaisuihin ja -järjestelmiin, syntyi idea mahdollisesta opinnäytetyöstä. Tarkoituksena olisi toteuttaa järjestelmä, jolla pystyttäisiin ohjaamaan valoja seinässä olevilla painikkeilla sekä sovelluksen tai nettisivun avulla puhelimesta käsin.

2 Älykäs valaistus omakotitalossa

2.1 Kohde ja toteutusvaihtoehdot

Opinnäytetyön kohde on vanhempieni omakotitalo, jonka hiljattain uusittu alakerta kaipasi kiinnostavaa ratkaisua valojen ohjaukseen. Tilassa oleva valaistus oli jo asennettu, joten yksittäisiä valoja ei ohjattaisi, vaan tarkoituksena olisi muodostaa valoskenaarioita valoryhmien (kiviseinä, liekkivalot, baarin valaistus) avulla (Kuva 1). Myös valojen ohjaukseen tarkoitetut ohjauspaneelit, jotka sisältävät kahdeksan eri painiketta, olivat jo asennettuina. Tehtäviksi muodostuivat siis sähkökaapin sisällön päättäminen (releistys, optoerottimet sekä virranjakelu) ja järjestelmän arkkitehtuurin suunnittelu. Myös ominaisuuksista, joita tultaisiin lisäämään, keskusteltiin useaan otteeseen.



Kuva 1. Kehitystyön apuna käytetty pohjakartta valaistavasta tilasta.

Keskeisiksi aiheiksi järjestelmässä nousivat valopainikkeiden painalluksien lukeminen, kyseisen tiedon pohjalta toimiminen sekä valaistuksen muokkaus pyydetyn datan pohjalta. Olin työharjoitteluni aikana tutustunut ilmaiseen Thingsboard nimiseen IoT-alustaan, joten sen käyttö projektissa tuntui luonnolliselta. Tärkeimmäksi syyksi IoT-alustan käyttöön muodostui halu siihen, että valopainikkeisiin sidotut esiasetukset tai "scenet" olisivat helposti muokattavissa (Liite 1), joten tätä ajatusta pyrin soveltamaan myös läpi toteutuksen. Muokattavuus toi kuitenkin mukanaan monimutkaisempia lähestymistapoja, sillä niin sanotut "kovakoodatut" asetukset haluttiin pitää minimissään (Kuva 2).

	0	1	2	3	4	5	6	7	8	9
preset	pois	tv	elokuva	baari	musiikki	peli	hämy	kivet	siivous	kulkuvalo
group1				1		1		1	1	1
group2				1	1	1		1	1	1
group3						1			1	
group4				1	1					
group5									1	
group6				1					1	
group7				1					1	
group8		1		1		1	1		1	1
group9				1	1	1		1	1	
group10				1						
group11						1			1	
group12		1	1							

Kuva 2. Ensimmäiset suunnitellut esiasetukset.

Mahdollisena toteutuksena harkitsin vaihtoehtoa siitä, että niin valojen ohjaus kuin painikkeiden lukeminen voitaisiin hoitaa Raspberry Pi -tietokoneen GPIO-pinnien avulla. Tämä lähestymistapa kuitenkin hylättiin: tulevaisuudessa sensorointia tulee olemaan pitkin taloa, eli GPIO-pinnien määrä ei riittäisi, sekä tarvittavan kaapeloinnin määrä kasvaisi huomattavasti. Seuraavana vaihtoehtona nousi esille idea tarvittavien komponenttien rajapinnoittamisesta mikro-kontrollerien avulla: Arduino sopisi tähän tehtävään loistavasti suuren saatavuutensa, laajan tuen sekä kustannustehokkaan hintansa vuoksi. Tarkemmin Arduino Uno, kaikkein tunnetuin ja käytetyin kyseisestä tuoteperheestä, tarvitsisi kuitenkin tavan, jolla liikuttaa dataa IoT -alustan ja itsensä välillä. Yhteystapoja löytyy monia niin Ethernet -yhteyteen (MQTT, AMQP) kuin väyläliikenteenkin (MODBUS, I2C)

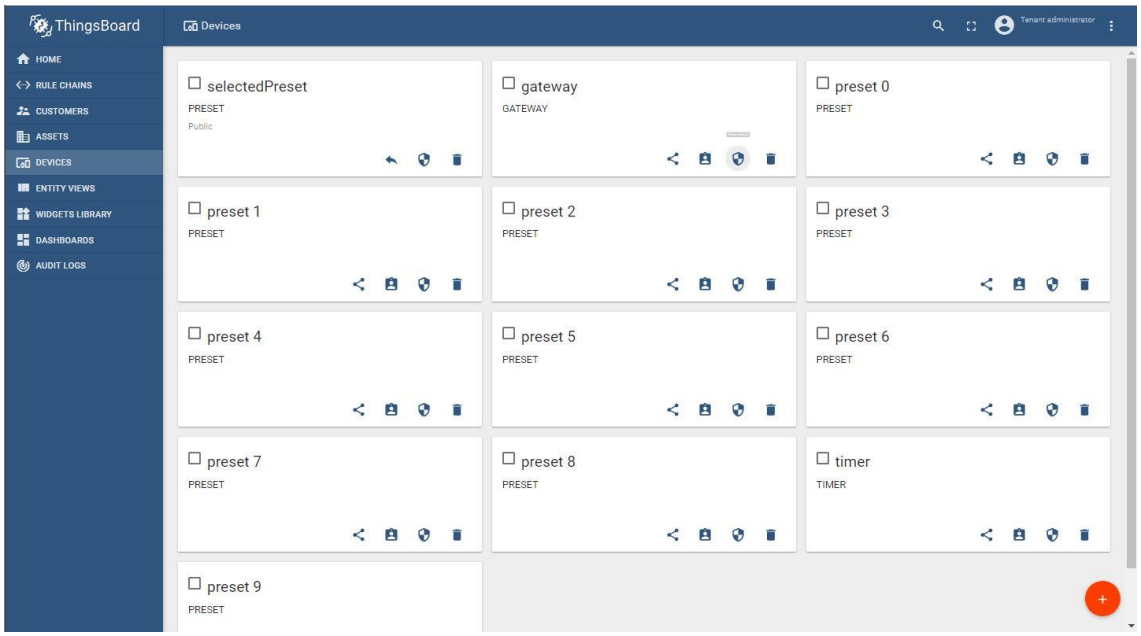
perustuvina. Kuitenkin tarkoituksen ollessa se, että järjestelmä tulee sijoittumaan eri puolille taloa, sekä tiedostettaessa helpon laajennettavuuden ja muokattavuuden merkityksen toteutuksessa, päädyin käyttämään yhteyksissä MQTT-protokollaa. Tämän myötä Arduinomme saisi päälleen Ethernet "shieldin", eli sen pinneihin kiinnittyvän lisäkortin, joka vaikkakin varaten osan pinneistä omaan käyttöönsä, antaa meille tavan yhdistää Arduinon lähiverkkoon Ethernet-kaapelilla.

Näin oli järjestelmän suunnitelma valmis: keskuksena (”hubina”) toimisi tietokone, jonka kanssa samassa lähiverkossa olevat Arduinot kommunikoisivat. Tieto välittyisi MQTT:llä, viestimuotona käytettäisiin universaalia JSON-formaattia. Valopainikkeista saatava tieto välittyisi Arduinolta Raspberry Pi -tietokoneen IoT-alustaan (Thingsboard), josta käsittelyn jälkeen ohjeistus toivotuista valoasetuksista välitettäisiin jälleen valoista vastaavalle Arduinolle. Mahdollinen tulevien anturei-den data noudattaisi samaa reittiä kuten nappien painallukset, eikä se vaatisi isoja muutoksia kaiken tarpeellisen jo ollessa valmiina (Liite 2).

2.2 IoT-alustat

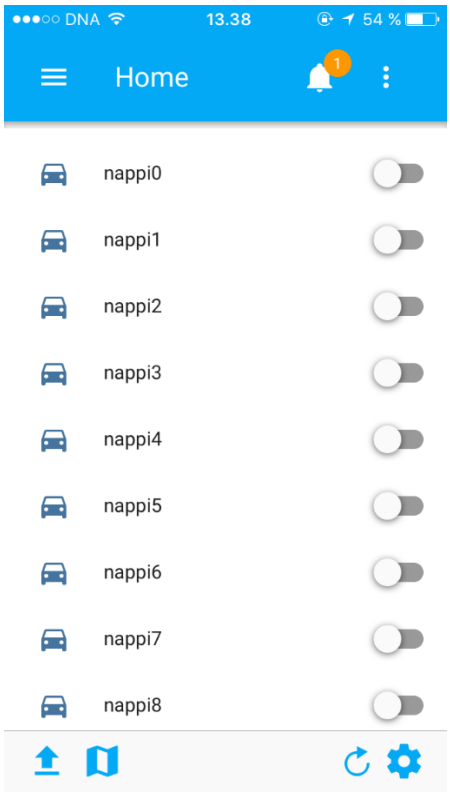
Käyttämäni järjestelmät (Thingsboard, Home Assistant) jakoivat monia yhtymäkohtia, mutta niiden erot valaisevat hyvin myöhemmin tapahtunutta päätöstä alustan vaihdosta.

Thingsboard on vuonna 2016 alkunsa saanut IoT-alusta, joka on suunniteltu yrityksiä ja teollisuutta ajatellen. Tämän järjestelmän etuina ovat helppo skaalattavuus ja vankka rakenne (Kuva 3). Yrityksen sivuilla on esimerkkejä muun muassa älykkäästä viljelystä ja linja-autojen statistiikan seurannasta ja visualisoinnista [2].



Kuva 3. Thingsboardin käyttöliittymä ja sen Devices-valikko.

Home Assistant on ensimmäisten joukossa ilmaantuneita kotiautomaatiojärjestelmiä. Tämä vuonna 2013 aloitettu järjestelmä on mahdollista asentaa useammanlaisena variaationa, mutta yleisimmät ovat Hassbian ja Hass.io. Nämä eroavat suurimmalta osin ajoympäristöltään: Hassbian on Raspbian Jessien muokattu versio, josta on riisuttu turhia osia pois, ja esiasennettu Home Assistantiin tarvittavat osat. Hass.io (Kuva 4) on Docker -virtualisointiohjelman sisällä pyörivä versio Home Assistantista, jonka ulkopuoliseen käyttöjärjestelmään ei käyttäjän tarvitse koskea [3]. Etuna tässä jälkimmäisessä saavutetaan suurempi turvallisuus (ajo virtualisoituna) ja toimintavarmuus. Hass.io:n tapauksessa pystytään myös lisäosia asentamaan helpommin, sekä päivitykset voidaan laittaa tapahtumaan automaattisesti. Automaattiset päivitykset toki tuovat mukanaan riskin siitä, että tulevaisuudessa päivitys saattaa rikkoa jotain järjestelmästä, mutta tästä ei ole liian suurta huolta, sillä Hass.io:n sisäänrakennettu snapshot-ominaisuus antaa mahdollisuuden varmuuskopioiden luomiseen ja hallinnointiin.



Kuva 4. Home Assistantin mobiilikäyttöliittymä.

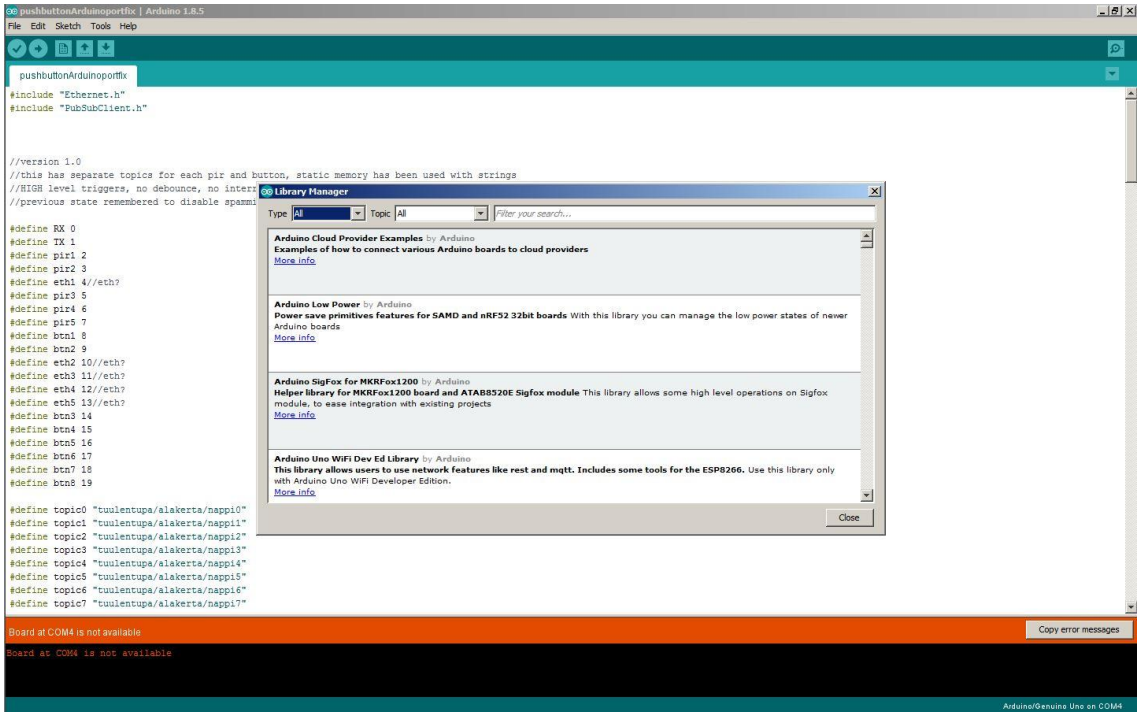
3 Kehittäminen ja asennus

3.1 Kehitysympäristö

Pystytin kehitysympäristön pöytätietokoneelle kotonani. Siltä käsin tein opinnäytetyön kehitysvaihetta useamman kuukauden Raspberry Pi:n, Arduinojen sekä demopiirilevyjen avulla. Kehitys siirtyi toteutuspaikkakunnalle siinä vaiheessa, kun järjestelmä pystytettiin ja kaikki laitteisto saatiin saman keskuksen sisään. Tämän jälkeen pääpaino siirtyi raportointiin, kehityksen painottuessa pitkille viikonlopuille. Pääosin näiden aikana implementoitiin uusia ominaisuuksia ja korjattiin käytössä havaittuja epäkohtia. Tämä tilanne palveli myös hyvin niin ohjelmointivirheiden löytämisen kuin ratkaisun toimivuuden testauksessa käyttäjän näkökulmasta.

3.1.1 Käytetyt ohjelmistot

Arduinoiden ohjelmointia varten latasin Arduino IDE:n, joka on ilmainen, avoimen lähdekoodin ohjelmointiympäristö mikrokontrollereille. Yleisimmin tätä käytetään nimenomaan Arduinojen ohjelmointiin, mutta myös muiden valmistajien laitteille sekä jopa muiden mikropiirivalmistajien piireille löytyy ratkaisuja tällä ohjelmistolla. IDE:n kautta on mahdollista ladata kirjastoja (Kuva 5), joiden avulla lisäominaisuuksien tai laitteiden hyödyntäminen on mahdollista (sensorit, ”shieldit” ja kommunikaatioprotokollat). [4.]

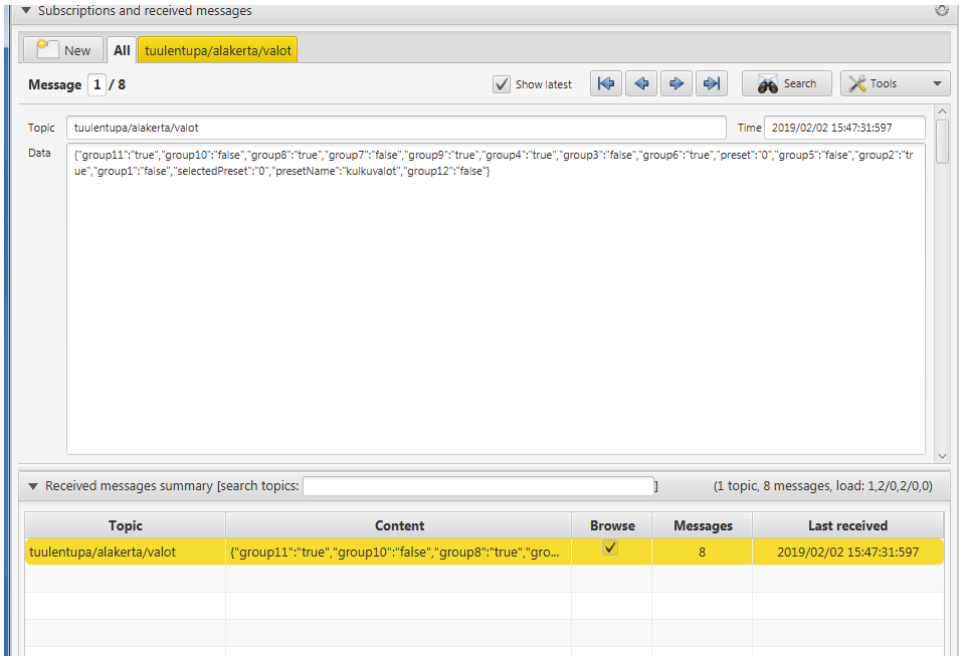


Kuva 5. Arduino IDE.

Thingsboardia pystyttäessäni tarvitsin etäyhteyden Raspberry Pi:lle, ja ensimmäiset asetukset teinkin PuTTY-ohjelmalla SSH-yhteyden avulla. Kuitenkin luontevamman käytettävyyden vuoksi siirryin käyttämään VNC Viewer-ohjelmaa, jolla sain otettua etätyöpöytäyhteyden Raspberrylle. Tämän ansiosta minulla oli käytössä kaksi työpöytää, ja esimerkiksi useamman tiedoston konfigurointi ja varmentaminen olivat vaivatonta. Lisäksi asensin Raspberrylle vielä Thingsboardia

varten Mosquitto MQTT-brokerin sekä PostgreSQL-tietokannan. Viimeisenä lisäsin Thingsboard Gateway-sovelluksen ja konfiguroin ohjelmat toimintavalmiiksi.

Tietokoneelleni asensin MQTT-Spy-ohjelman (Kuva 6) simuloidakseni ja tarkastellakseni Raspberry Pi:n MQTT-brokerin kautta tapahtuvaa dataliikennettä. Pystyin siis kehityksen eri vaiheissa luomaan skenaarioita eri laitteille ja näin päättämään MQTT:n sisäisten topicien hierarkian sekä lähetettävän JSON-viestin koostumuksen.



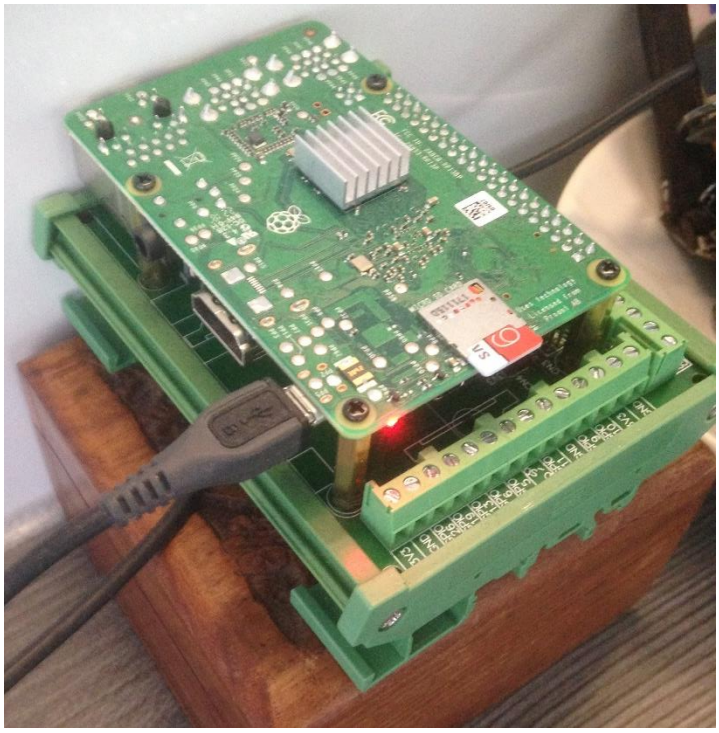
Kuva 6. Mqtt-Spy.

3.1.2 Käytetty laitteisto

Kehitysympäristön alustana toimi pöytäkoneeni, vaikkakin suoritustehovaatimuksien suhteen koko kehityksen olisi pystynyt suorittamaan kannettavalta tietokoneelta. Etuna tässä kuitenkin oli parempi työnkulku sekä työergonomia.

Raspberry Pi:tä varten ei vaadittu ympäristöltä muuta kuin verkkoyhteys ja virtalähde, sillä koko kehityksen ajan se pysyi niin sanotussa ”headless”-tilassa, eli

ilman siihen kytkettyä näyttöpäätettä ja hallintalaitteita (Kuva 7). Yhteys tapahtui nimenomaan etätyöpöytäyhteyden kautta.



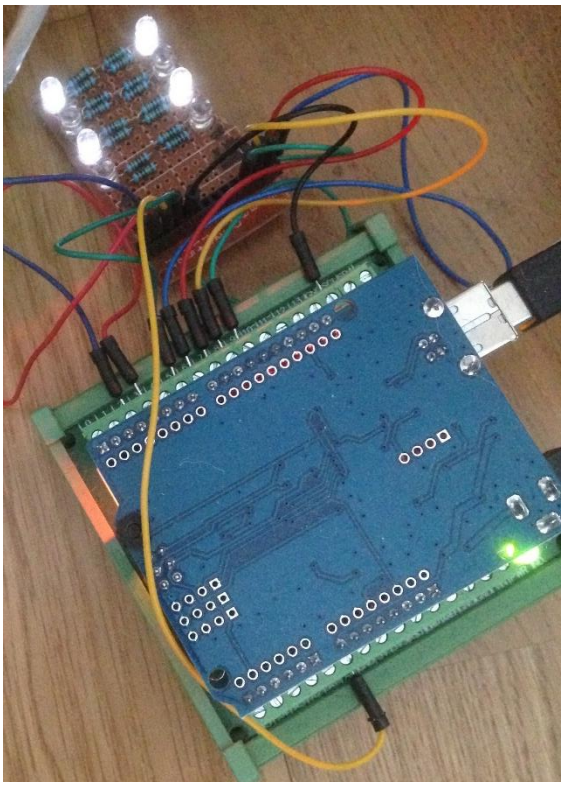
Kuva 7. Raspberry Pi kehitysympäristössä. Langaton hadless-tila.

Arduinot tarvitsivat ohjelmointia varten usb b-kaapelin, jolta ne myös saivat toimintaansa tarvittavan virran. Tämän lisäksi niiden Ethernet-piiriä varten tarvittiin myös asianmukainen piuha. Tietokoneelta vedettiin usb-jakaja Arduinojen lähelle, josta lisävirran liittämisen jälkeen kytkettiin jokainen Arduinoista hubiin.

3.2 Kehitystyö

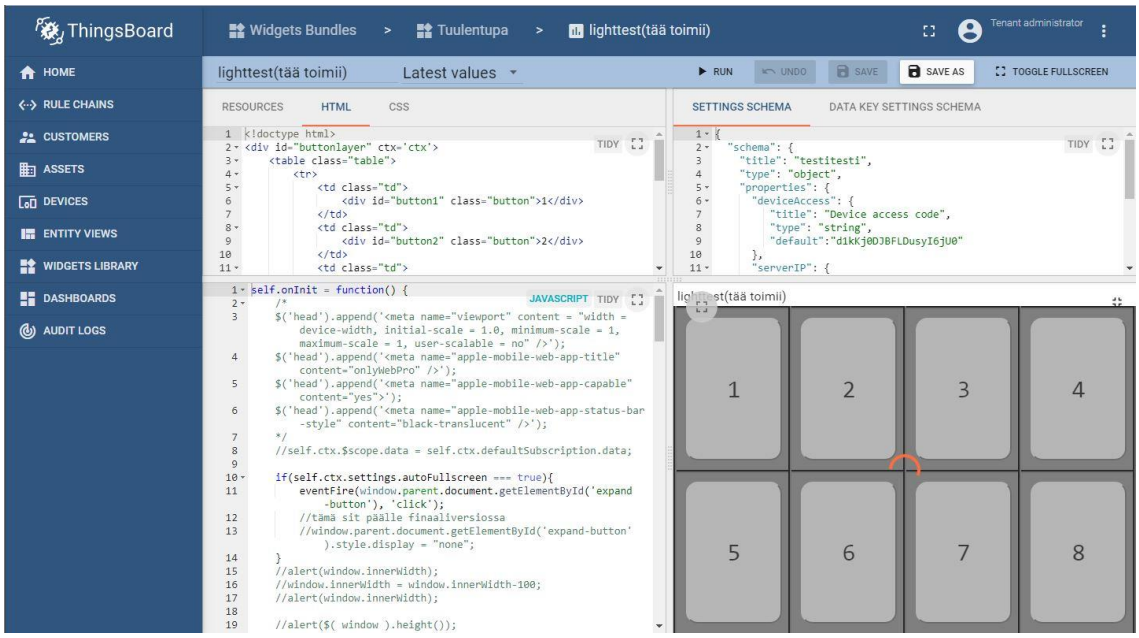
Ensimmäisenä ryhdyin kehittämään koodia Arduinoille (Liitteet 7, 8 ja 9): niiden tarvitsi kommunikoida Ethernet-kaapelia pitkin MQTT:llä, sekä viestien tulisi noudattaa JSON-formaattia. Lisäksi tarvitsin simuloidun ympäristön valopainikkeille sekä valoille, asennuskohteen ja kehityspaikan ollessa eri kaupungeissa ei olisi mahdollista kehittää ohjelmistoja paikan päällä, oikeassa ympäristössä. Tämä si-

muloitu ympäristö koostui ledeistä, vastuksista sekä napeista juotettuna piirilevylle (Kuva 8), ja ajoi asiansa ongelmitta pois lukien ongelmaa, jonka huomasin myöhemmin. Ledit syttyvät ilman tätäkin, mutta esimerkiksi releitä käytettäessä on tarpeen asettaa pinMode -funktion parametriksi "OUTPUT". Tällä parametrilla nimeämme kyseisen pinnin lähtevän datan pinniksi, eli se ei voi vastaanottaa dataa asetuksen ollessa edellämainitussa tilassa.



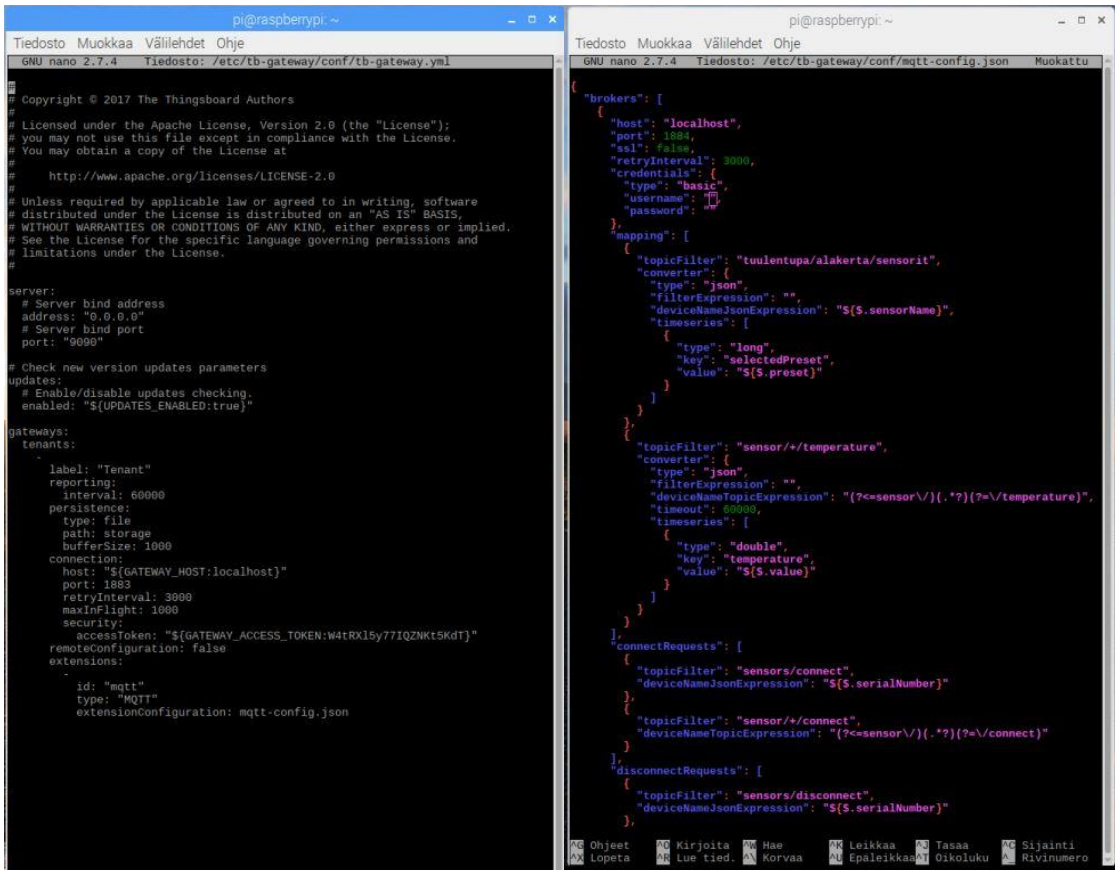
Kuva 8. Arduino ja kehitysympäristö.

Seuraavaksi kehitettävänä oli IoT-alustamme Thingsboard. Tarkoituksena oli luoda painikkeet valojen ohjaukselle, ikään kuin digitaalisena vastikkeena seinästä löytyville kytkimille. Thingsboard on Java-pohjainen avoimen lähdekoodin IoT-alusta, jonka "dashboardit" eli näkymät on mahdollista täyttää erilaisilla pienoisohjelmilla eli "widgeteillä". Nämä ohjelmoidaan käyttäen HTML-, CSS- ja JavaScript-kieliä, eli ne ovat ikään kuin pieniä nettisivuja (Kuva 9). Oli siis luonnollista aloittaa tästä graafisesta puolesta.



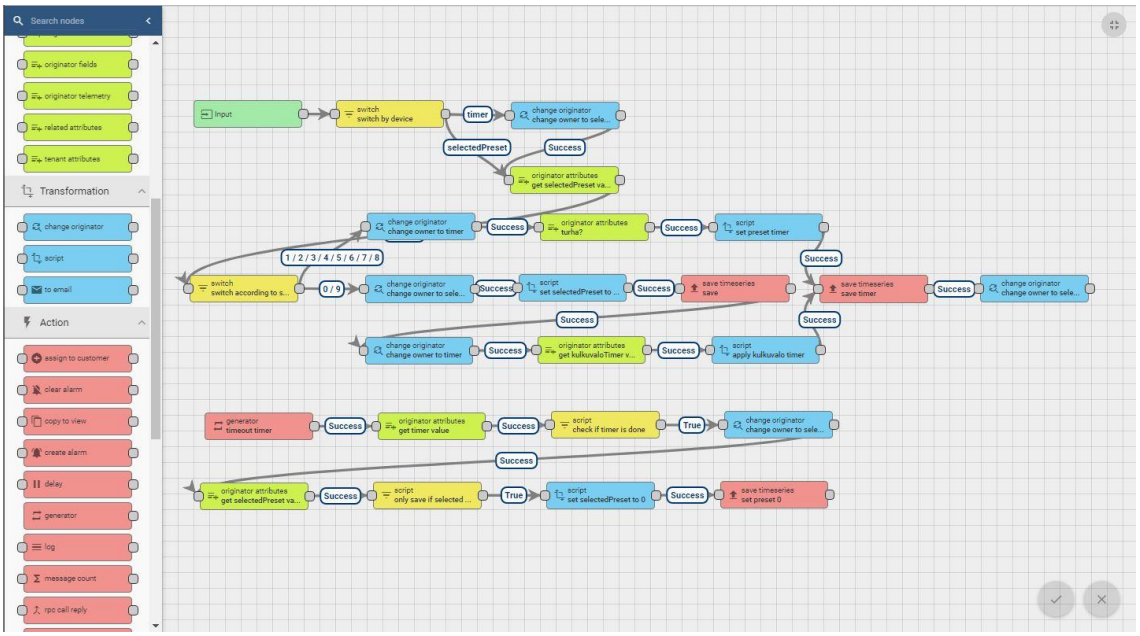
Kuva 9. Thingsboardin widget-kehitystyökalu.

Saadessani tämän valmiiksi siirryin konfiguroimaan datan liikettä: Thingsboard Gateway on ohjelma, jonka avulla saadaan ohjattua sekä suodatettua sisään tulevia viestejä esimerkiksi OPCUA-, Sigfox- ja MQTT-lähteistä (Kuva 10). Tämän avulla pystyin ohjailemaan sensoridatan oikeaan virtuaaliseen kohdelaitteeseen, josta sitä hyödynnettiin napin "tilatietona", ja pystyttiin lukemaan valoasetuksia lähettäessä. [5.]



Kuva 10. Thingsboardin konfigurointitiedostoja.

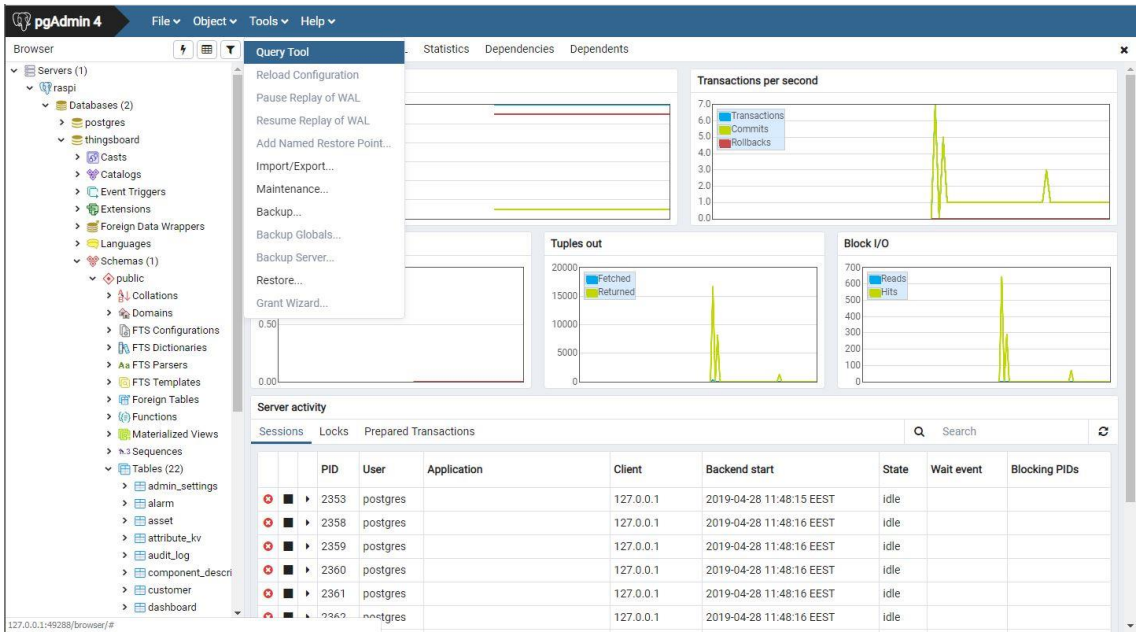
Datan käsittely Thingsboardilla tapahtuu hyödyntäen sen uutta ominaisuutta, RuleChainia (Kuva 11). Tämä flow-ohjelmointityökalu pitää sisällään "nodeja", joiden avulla pystytään toteuttamaan erilaisia modifiointeja datalle, tekemään laskutoimituksia sekä lähettämään dataa eteenpäin. Näitä ominaisuuksia hyödyntäen toteutin logiikan, jonka pohjalta valoja painettaessa haettiin kyseisen napin "preset" ja lähetettiin sen pohjalta valoja ohjaavan Arduinon MQTT-topiciin boolean-pohjainen lista, joka sisälsi ohjeet jokaista valoryhmää koskien. Myös liiketunnistimiin pohjautuva kulkuvalojen ohjaus sekä valojen automaattinen sammutus kuuluivat tähän logiikkaan.



Kuva 11. RuleChain-automatisointityökalu.

3.3 Asennus

Ensimmäisiin varsinaisiin ongelmiin projektissa törmättiin järjestelmää asentaessa. Viestintä tapahtui hyvin, mutta löytyi yksi merkittävä ongelma, jota aiemmin ei ilmennyt: kun valokytkintä painettiin joko fyysisestä tai digitaalisesta sijainnista, ensimmäiset vaihdot sujuivat ongelmitta. Kun niitä painettiin lisää, esimerkiksi kytkimiä 5 ja 6, jäi järjestelmä vaihtelemaan niiden välillä, eikä asettunut kumpaankaan. Poissulkevasti (PostgreSQL (Kuva 12), MQTT broker, Arduinot, Thingsboard) saimme rajattua ongelman Thingsboardin RuleChainiin ja kävi ilmeisen selväksi, että kyseinen järjestelmä ei ainakaan tässä kehityksen vaiheessa soveltuisi käyttötarkoitukseemme, sillä saapuvan datan tulee ilmestyä vain ja ainoastaan kerran. Sensoridatan kohdalla tämä ei olisi ongelma, sillä arvojen vaihtelu ei haittaisi. Kuitenkin kyseisen ongelman ilmetessä näin näkyvästi ei tällainen ominaisuus olisi hyväksyttävää.



Kuva 12. PGAdmin, PostgreSQL-hallintatyökalu.

Tästä johtuen jouduimme tekemään hankalan valinnan: jatkaako yrittämistä ja ratkaisun etsimistä tällä nyt ilmeisen tarkoitukseen sopimattomalla alustalla vai vaihtaako alustaa tyystin toiseen ja mahdollisesti tehdä uudelleen suurikin osa opinnäytetyön toteutuksesta. Aikaa "deadlineen" eli viimeiseen ajankohtaan raportin palauttamiselle oli kuitenkin tässä vaiheessa vain pari kuukautta, eikä valinta ollut helppo. Koska järjestelmä oli tämänkertaisen asennuksen päätteeksi kuitenkin saatava toimimaan vähintäänkin perusominaisuuksiltaan, päädyimme kovakoodaamaan valoasetukset lähettävän pään Arduinolle, jonka asetimme lähettämään viestit oman topicinsa sijasta vastaanottavan Arduinon topiciin, jossa odotetaan booleanpohjaista JSON-listaa. Kyseinen lista oli siis täsmälleen saman muotoinen kuin IoT-alustalta saapuva viesti, erona vain kovakoodaus. Tämä ratkaisu kuljetti datan MQTT:tä pitkin, ja oli pohjimmiltaan puolikas järjestelmä. Kuitenkin jo saimme kokemuksia siitä, miltä kyseinen valojenohjaus tuntuu käyttää arjessa.

Palatessani paikkakunnalle seuraavalla viikolla oli aika suorittaa suuri muutos: Vaihtaisimme Thingsboardin Home Assistant -kotiautomaatiojärjestelmään. Olin saanut kyseisestä ohjelmistosta vihiä ystävältäni, joka myös teki projekteja IoT:n

sekä älykotien parissa. Monia kysymyksiä myöhemmin sain itseni vakuuttuneeksi onnistumisen mahdollisuudesta: niin vanha kuin uusikin järjestelmä viestisi käyttäen MQTT -protokollaa, sekä ymmärtäisi JSON-formaattia, eli suuri osa Arduinoihin tekemästäni työstä olisi hyödynnettävissä. Jatkoin siis eteenpäin järjestelmän vaihdoksessa: asensin Hass.io:n Raspberry Pi:n käyttöjärjestelmäksi, asensin lisäosat koskien MQTT brokeria sekä flow-ohjelmointia (Node-RED), sekä aloitin purkamalla edellisen käynnin aikana koodatut tilapäiset ratkaisut jälleen toimivaksi, yhteensopivaksi järjestelmäksi.

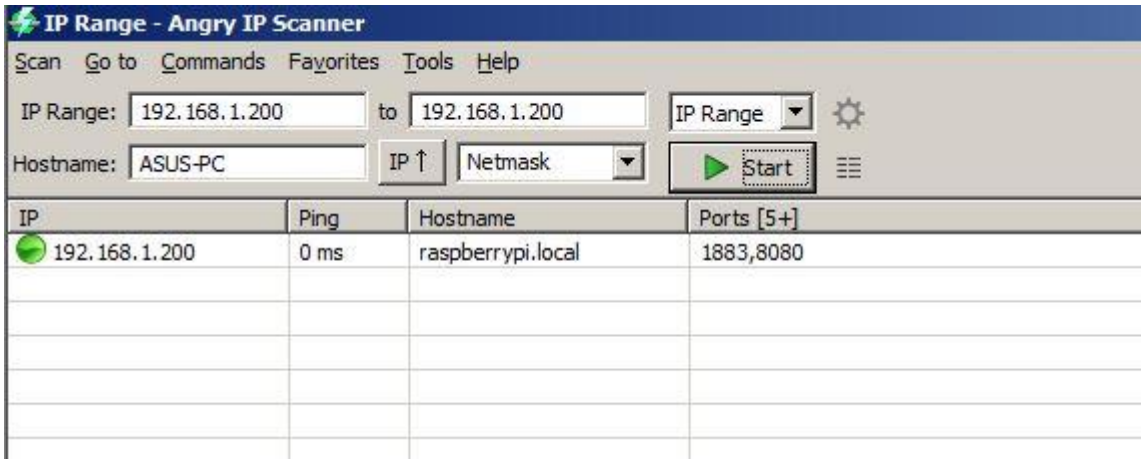
Tämän jälkeen loin aikaisemmin Thingsboardin RuleChainissa olleen logiikan uudelleen vastaavaan flow-ohjelmoinnin lisäosaan Node-REDin Home Assistantin sisällä. Node-RED jakoi monia ominaisuuksia aikaisemman RuleChainin kanssa, joten logiikan mieltämiseen ja ylimääräiseen päänvaivaan ei kulunut tämän osalta ajallisia resursseja. Kun sain opeteltua, kuinka tämän järjestelmän ”nodet” toimivat, ja miten niiden liittäminen ja viestinvälitys toimivat, pystyin toteuttamaan seuraavan: Arduino lähettää painetun painikeen tiedon eteenpäin, Node-RED havaitsee sen ja valinnan pohjalta luo JSON-viestin joka lähetetään valoja ohjaavalle Arduinolle. Jos valoja ohjataan puhelimella tai tietokoneella, on prosessi suurilta osin samankaltainen: automaatio havaitsee kyseisen painikkeen aktiivisuuden, ja ohjaa viestin samaa polkua kuin MQTT-viestillä saapuvan datan osalta. Seuraavissa osioissa selvitän tarkemmin Hass.io:n asennusta, konfigurointia sekä komponentteja.

Kun järjestelmä saatiin toimimaan (Liite 3), innostuimme implementoimaan vielä alkuperäiseen suunnitelmaan kuulumattoman osan, sähköohjattavan verhon. Tämän lisääminen oli helposti saavutettava ominaisuus: sen tilaa voidaan ohjata pulssitiedolla, joko ”auki” tai ”kiinni” -johtoon lähetettävällä, muutaman kymmenen millisekunnin pulssilla. Verhon ohjaus tapahtuu valojen tavoin, eli Boolean -totuusarvolla. Näin saamme lähetettyä verhon halutun tilan samanlaisena viestinä kuin valojenkin.

3.3.1 Hass.io

Hass.io on Home Assistantin suositeltu ja käytetyin asennustapa. Asennus tapahtuu suurimmalta osin automatisoituna: ensin levykuva poltetaan muistikortille esimerkiksi Etcher-ohjelmalla, tämän jälkeen muistikortti kytketään Raspberry Pi:hin. Käynnistämisen jälkeen Hass.io hakee tarvitsemansa tiedostot palvelimelta. Tähän kuluu noin 20 minuuttia, jonka aikana kuitenkin jo näkyy kuva ja ilmoitus päivitysten lataamisesta Raspberryn IP-osoitteessa, portissa 8123. Kun päivitysten lataus on valmis, kyseiseen osoitteeseen nettiselaimella yhdistämällä pääsee luomaan käyttäjätunnukset ja kirjautumaan sisään. Tämä osoite on Home Assistantin osoite, johon esimerkiksi puhelimen sovellus yhdistetään. [6.]

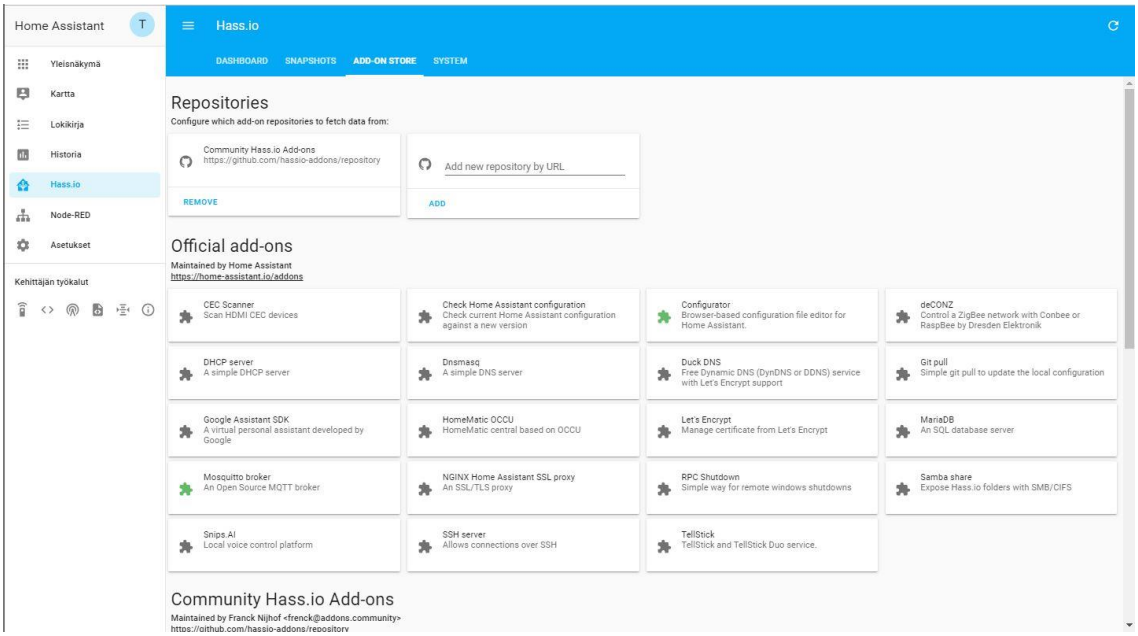
Järjestelmän käynnistymistä (portit 1883, MQTT ja 8123, Home Assistant) seurasin Angry IP Scannerilla (Kuva 13). Tällä työkalulla voin asettaa haluamani IP-alueen sekä portit, joita haluan skannata. Ohjelmasta ilmenee, onko IP-osoitteen takana laitetta, onko se päällä, sekä onko asetettuja portteja auki.



Kuva 13. AngryIP Scanner -työkalu.

Tunnusten luomisen jälkeen käyttäjä päätyy states-sivulle. Tällä sivulla näkyvät löydetyt laitteet sivun yläosassa ja sivun keskelle voidaan luoda uusia hallintapaneeleita ja ilmaisimia. Vasemmalla sivulla aukeaa valikko, josta päästään valitsemaan kohta Hass.io, ja tämän sisältä ”add-on store” (Kuva 14). Tästä valikosta löytyy suurin osa tarvittavista komponenteista, toki myös uusienkin repositoryjen

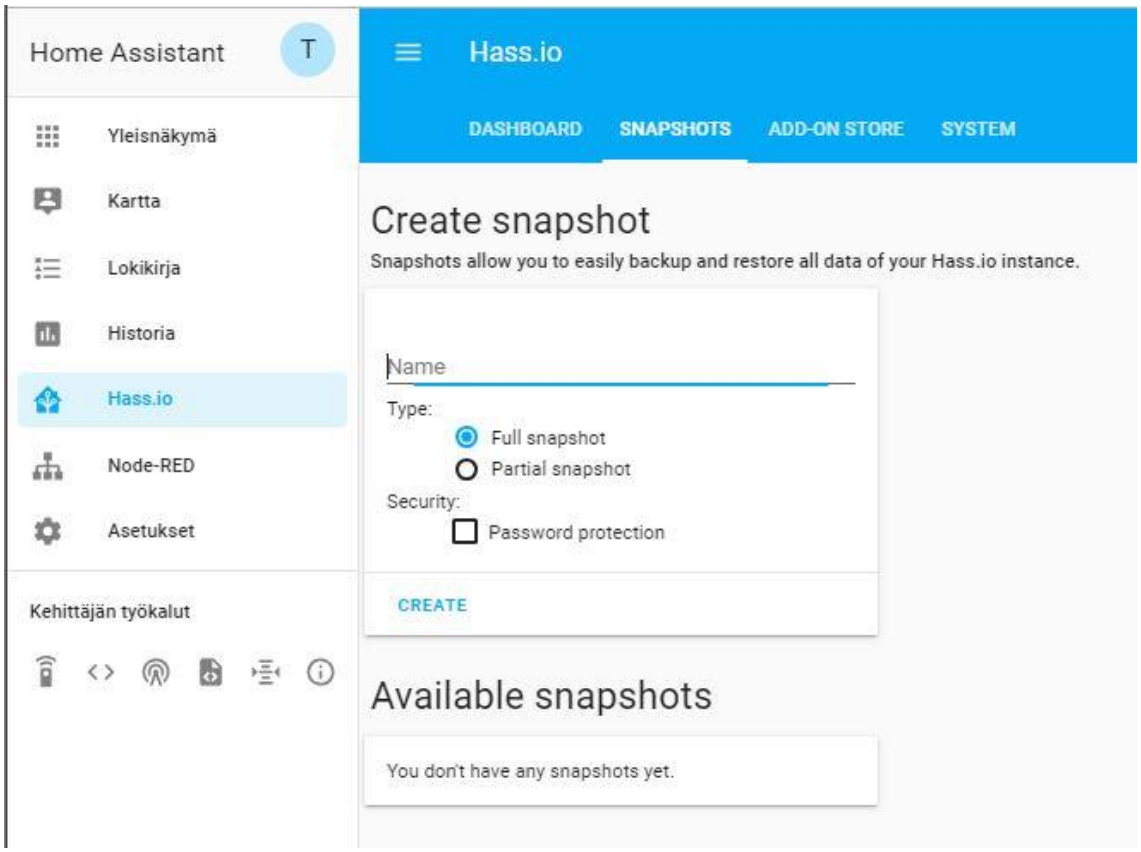
lisääminen onnistuu tästä valikosta. Samaisesta ikkunasta näemme myös välilehdet ”snapshotteihin” eli varmuuskopioihin sekä tarkempiin järjestelmän tietoihin.



Kuva 14. Add-on Store.

3.3.2 Snapshot ja varmuuskopiointi Hass.io:lla

Tämän ominaisuuden avulla on mahdollistettu helppo, ajon aikana tapahtuva varmuuskopiointi (Kuva 15). Sen hallinnointi tapahtuu recorder-elementillä, jonka konfigurointi on mahdollista Home Assistantin configuration.yaml -tiedostossa. Näiden avulla voidaan palauttaa täysin uusi asennus aikaisempaan tilaan, jos snapshot on tallessa koneen ulkopuolella.



Kuva 15. Home Assistantin Snapshot-valikko.

Snapshot-ominaisuus on mahdollista myös automatisoida esimerkiksi Node-RE-Din tai Home Assistantiin sisäänrakennetun automation -skriptaustyökalun avulla kutsumalla ajastettuna snapshotin service-kutsua. Nämä ovat järjestelmän sisäisiä kutsuja, joilla voidaan käynnistää tapahtumia, tässä tapauksessa joko osittaisen tai täyden varmuuskopion luonti.

On myös mahdollista lähettää varmuuskopio automaattisesti esimerkiksi NAS-levylle tai pilvipalveluun (Dropbox, Google Drive). Näihin työkalut löytyvät joko sisäänrakennettuina tai add-oneina.

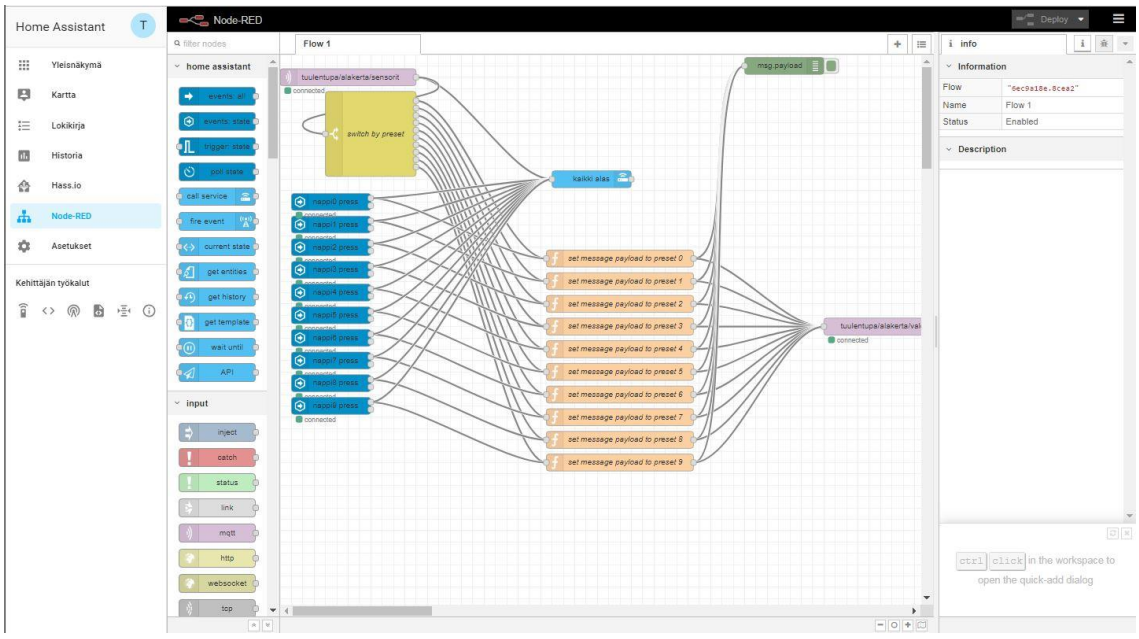
Varmuuskopiota tehdessä on myös hyvä muistaa, mitä kaikkea on kopioimassa. Esimerkiksi tietokanta pitää sisällään paljon turhaa dataa, jonka tallentamisen optimointi on suuresti suositeltavaa. Myös osa Home Assistantin omista tiedoista on sellaisia, joita ei ole välttämättä tarvittavaa tallentaa joka varmuuskopioon.

Yksi kopio ei vie merkittävästi tilaa, mutta tilansäästö konkretisoituu, kun tarkastellaan esimerkiksi useamman kuukauden aikaisia kopioita. Loppukädessä on kuitenkin käyttäjästä itsestään kiinni, kuinka tämän toteuttaa, mutta jokaisessa varmuuskopiointiratkaisussa pitäisi toteutua nämä ehdot: [7.]

- Varmuuskopiointi tapahtuu automaattisesti.
- Varmuuskopioita on olemassa vähintään kolme kappaletta.
- Varmuuskopiosta on olemassa kopio, joka ei sijaitse kyseisellä laitteella.
- Varmuuskopioita luodaan niin tiheään, että millä tahansa hetkellä tapahtuva kriittinen skenaario ei aiheuta merkittävää tietohäviötä.

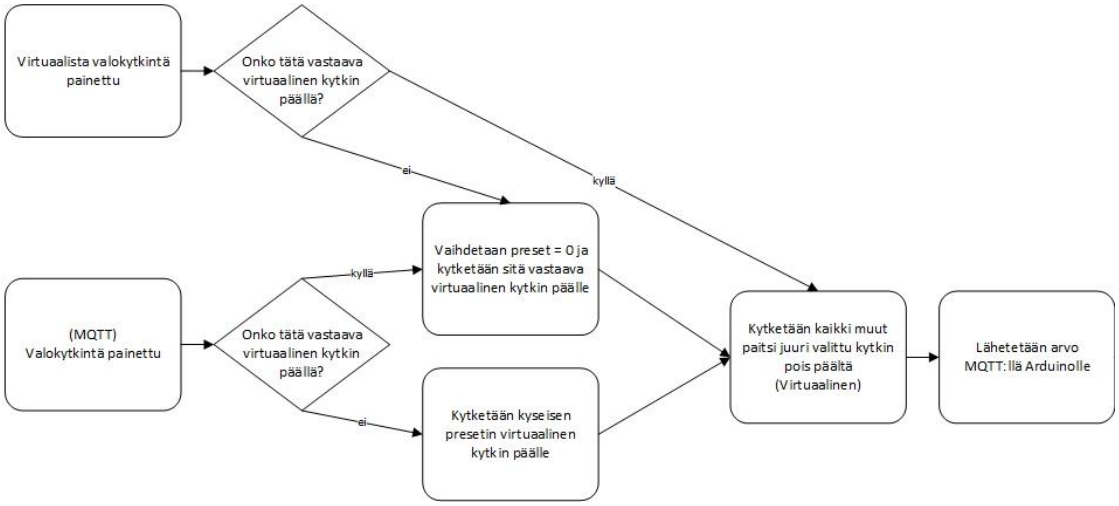
3.3.3 Node-RED

Node-RED on komponenttiorientoitunutta flow-ohjelmointiparadigmaa käyttävä avoimen lähdekoodin työkalu, jolla voidaan luoda automatisointeja Home Assistantin sisälle (Kuva 16). Siinä ”nodet” ovat logiikassa mustia laatikoita, joille voidaan asettaa sisään- ja ulostuleva data, sekä valita parametreja toimintaa varten (Liite 4).



Kuva 16. Node-RED.

Node-REDissä toteutetaan järjestelmän automatisointi ja logiikka. Ratkaisussamme toteutin rinnakkaisten (fyysinen, virtuaalinen) valokytkinten toiminnan seuraavanlaisesti:



Kuva 17. Toteutettu logiikka node-REDin sisällä.

Tällä ratkaisulla käyttäjä näkee valitun esiasetuksen myös Home Assistantin käyttöliittymästä, ja kumman tahansa kytkimen painaminen näkyy samalla tavalla käyttäjälle (Kuva 17). Tämä selkeyttää käyttäjäkokemusta ja vähentää turhia varmistuspainalluksia.

3.3.4 Mosquitto MQTT Broker

Home Assistantin Mosquitto-lisäosa ei eroa toiminnaltaan normaalista Mosquitto-asennuksesta. Tämä Broker mahdollistaa viestinnän MQTT-protokollan kautta, jota käytämmekin koko järjestelmän osien yhdistämisessä ympäri taloa.

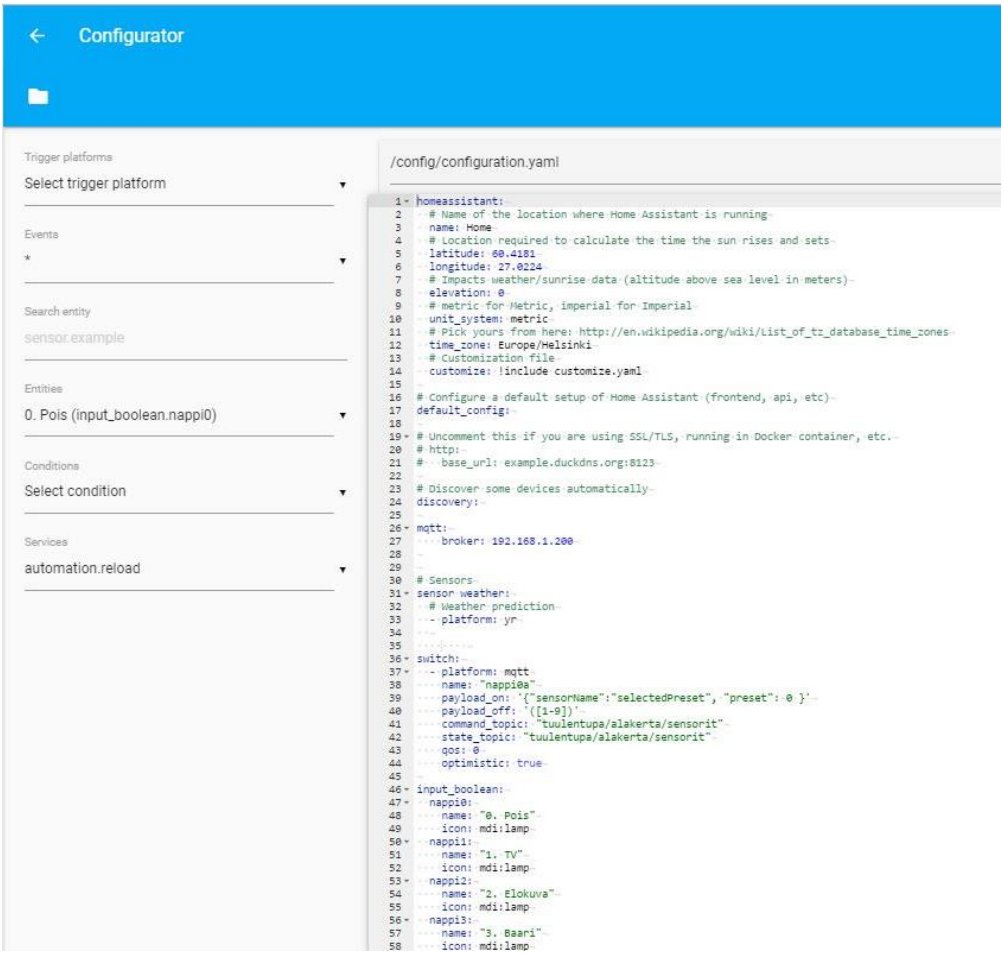
MQTT-viestit lähetetään topiceihin, joiden hierarkia on useimmiten seuraavanlainen: talo/kerros/järjestelmä/laite. Eli suuremmat käsitteet pitävät sisällään spesifimpiä käsitteitä, ja lopuksi alimpana hierarkiassa voi olla yksittäinen laite tai sen

ID, jos tälle on tarvetta. Ylempään topictasoon lähetetty viesti kulkeutuu hierarkiassa alempiin tasoihin (topic1, topic1/subtopic1), mutta ei viereisiin topiceihin (topic1/subtopic1, topic1/subtopic2). Tällä tavoin voidaan esimerkiksi tietylle laitteistoryhmälle lähettää komentoja uudelleenkäynnistyksestä tai asetusten muutoksesta. Home Assistant tukee MQTT:n kohdalla myös laitestatusta, eli laite päivittää tilansa säännöllisin väliajoin ja lähettää siitä viestin tiettyyn topiciin.

Lisäosan konfigurointi tapahtuu sen välilehdellä tekstiriviin ennen käynnistämistä, sekä configuration.yaml -tiedostoon. MQTT-yhteydelle voidaan halutessa asettaa salasana ja käyttäjätunnus, mutta anonymous-vaihtoehto on myös mahdollinen.

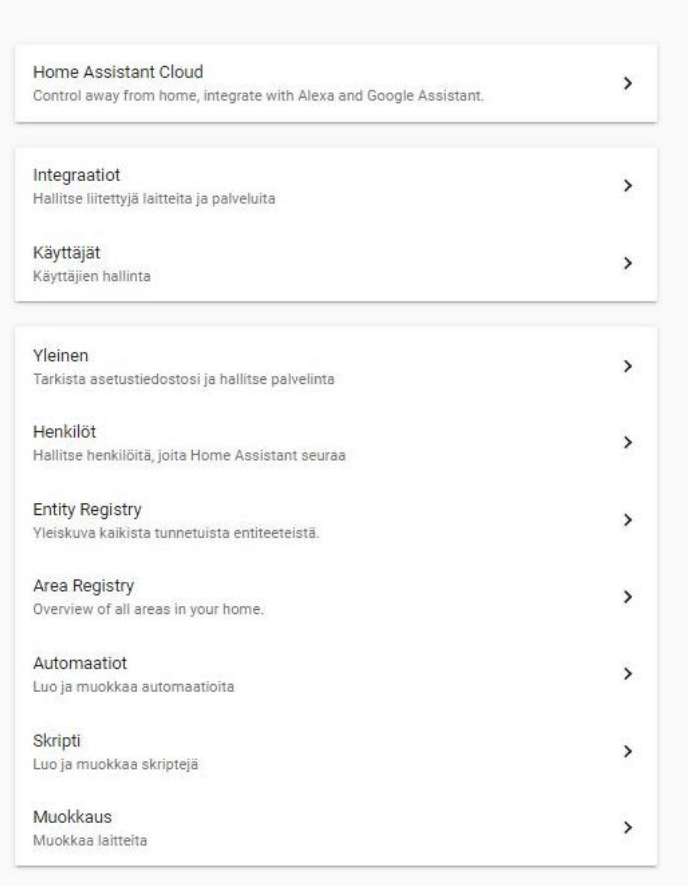
3.3.5 Configurator

Configurator-lisäosan avulla päästään muokkaamaan Home Assistantin tiedostoja, eli konfiguroimaan asetuksia, ryhmittelemään laitteita sekä lisäämään nappeja, sensoreita ja laitteita (Kuva 18). Myös näiden liitäntätapoja, kuvakkeita sekä nimiä voidaan muokata halutusti.



Kuva 18. Configurator add-on.

Kun Configurator-työkalulla muokkaus on lopetettu ja painettu tallennuskuvaketta, on suositeltavaa käydä jokaisen muokkauskerran jälkeen tarkistamassa asetusten oikeellisuus (Kuva 19). Tämä saavutetaan menemällä ensin asetuksiin, sieltä ”yleinen”, ja sivun ylälaidasta kohtaan ”tarkista asetukset”.



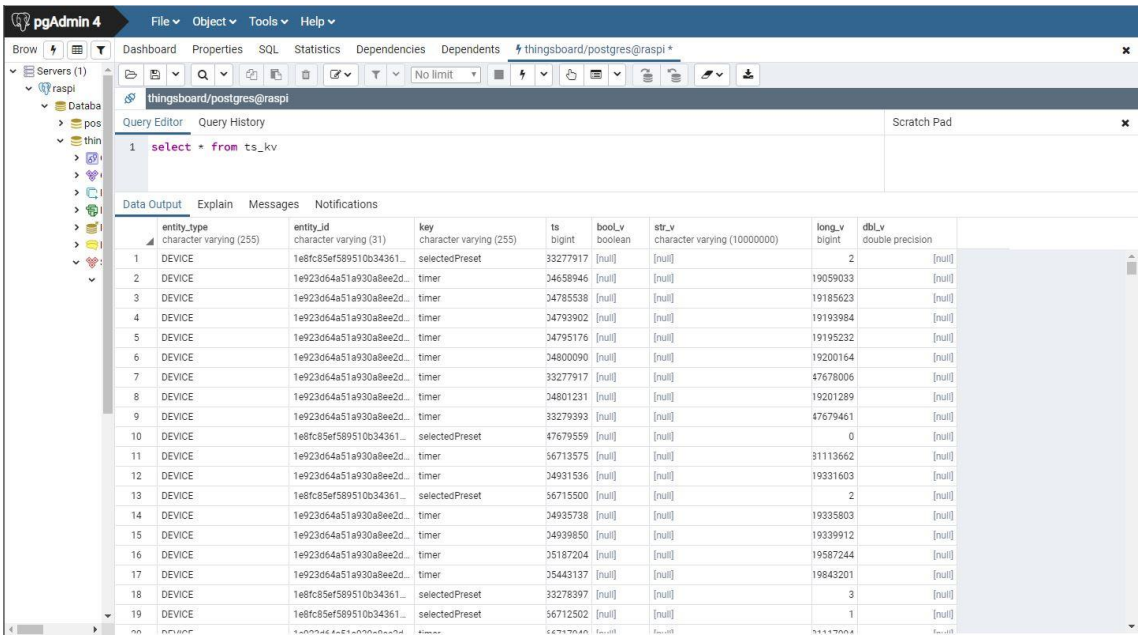
Kuva 19. Home Assistantin ”Asetukset”-valikkonäkymä.

Tämä ilmoittaa, jos asetustiedostossa on syntaksivirheitä, eli sellaisia vikoja, jotka rampauttaisivat järjestelmän toiminnan. Kun asetukset on tallennettu ja tarkastettu, on suositeltavaa käynnistää Home Assistant uudelleen. Tämä onnistuu painamalla nappia sivun alalaidassa.

3.3.6 Tietokanta ja vaihtoehdot

Thingsboard käyttää oletusarvoisesti HSQLDB-tietokantaa ja Home Assistant SQLite-tietokantaa. Molemmat näistä ovat usealle alustalle sopivia, kevyitä tietokantoja. Thingsboard-toteutukseni aikana vaihdoin senhetkiseksi tietokannaksi PostgreSQL-tietokannan, koska sen avulla pystyin lähiverkon sisällä ottamaan yhteyden siihen, poistamaan turhia tietueita (Kuva 20) ja varmuuskopioimaan niitä helpommin, sillä esimerkiksi thingsboardista tietokannan varmuuskopioin-

tiominaisuus olisi vaatinut maksullisen Professional Editionin. Kuitenkin käyttäessä PostgreSQL -tietokantaa, pystytään tietokantaa käsittelemään järjestelmän ulkopuolelta.



Kuva 20. Query tool käytössä PGAdmin-ohjelmalla. Näkyvillä Thingsboardin laitetietoja.

Home Assistantin tapauksessa on hyvin todennäköistä, että teen samankaltaisen vaihdoksen tulevaisuudessa. Saavutettavat edut toiminnallisuudessa toimivat riittävänä perusteluna vaihtoon. Hass.io:n tapauksessa ainakin tällä hetkellä on kuitenkin se tilanne, että tietokannan tulee sijaita verkkosijainnissa, esimerkiksi verkkokovalevyllä. Syynä tähän on se, että erillisten ohjelmien asennus tulee tehdä lisäosina. Hassbianin tapauksessa tämä ei kuitenkaan ole ongelma, sillä silloin on mahdollista asentaa ohjelmia pääkäyttäjänä. Hass.io:n käyttämä SQLAlchemy tukee MySQL-, MariaDB-, PostgreSQL- ja MS SQL Server -tietokantoja oletusarvoisen SQLiten lisäksi. Hass.io:n add-onina löytyy kuitenkin jo MariaDB, joten muiden tietokantojen implementointi lienee väistämätöntä tulevaisuudessa.

4 Järjestelmän ominaisuudet ja toiminta

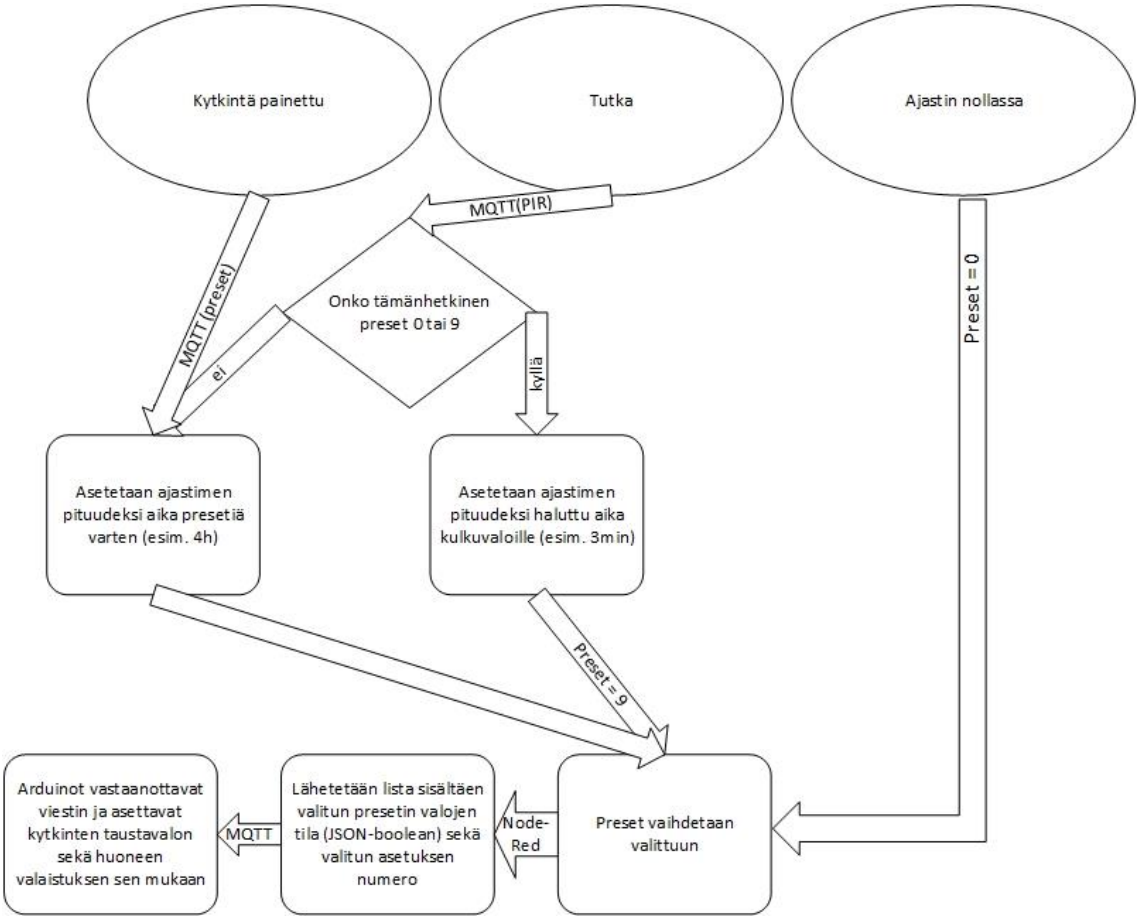
Järjestelmän toimintalogiikka pyritään toteuttamaan tulevaisuutta silmällä pitäen. Vaikkakin tämänhetkisessä toteutuksessa kohteena ovat pelkästään alakerran valot, on tulevien päivitysten osalta todennäköistä, että siihen lisätään kerros kerrokselta uusia valoja ohjattavaksi. Järjestelmän osat antavat jo mahdollisuuden siihen: lähiverkon kattaessa koko talon ja piha-alueen, jo käytössä olevaa protokollaa (MQTT), on helppo hyödyntää myös tulevien hallintalaitteiden lisäämisessä. Kytkennältään nämä eivät siis lainkaan poikkea alakertaan asennetuista laitteista: kytketään vain verkkokaapeli, virta sekä tarvittavat tulo- ja lähtösignaalit kytkimiltä tai releiltä (Liitteet 5 ja 6).

4.1 Toimintalogiikka

Skenaarioita on valaistusjärjestelmän käytön suhteen kolme:

- Valokytkintä painetaan.
- Tutka havaitsee liikettä.
- Automaattiajastin menee nollaan.

Nämä tuottavat yhdessä ratkaisun jonka avulla estetään valojen unohtuminen päälle ja näin turha virrankulutus, toteutetaan automaattiset kulkuvalot (Kuva 21) sekä mahdollistetaan tulevien lisäasetusten tekeminen.



Kuva 21. Järjestelmän input-skenaariot.

4.2 Käyttäjäkokemus

Käyttäjälle rajapinta pyritään pitämään mahdollisimman samankaltaisena verrattuna normaaliin valokytkimeen. Kun valokytkintä painetaan, käynnistyy tietty ryhmä valoja, ja kun sitä painetaan uudelleen, valot sammuvat. Tämä toimii niin virtuaalisessa kuin fyysisessäkin kytkimessä. Tässä intuitiivisessa käyttöliittymässä käyttäjää ei tarvitse neuvoa, vaan he oivaltavat oitis perusominaisuuksien toiminnan. Myös esimerkiksi lasten ja ikääntyneemmän väestön on näin helppo siirtyä käyttämään ”älykästä valaistusta”. [9.]

Jos esimerkiksi pääkäyttäjä kuitenkin haluaa jonkinlaista muokattavuutta tai datan seurantaa, on hänelle mahdollista tehdä oma näkymä, josta voidaan hallita

valojen esiasetuksia tai seurata talon sensorien lähettämää dataa, kuitenkaan puuttumatta varsinaiseen koodiin tai tekstimuotoiseen konfigurointiin.

5 Toteutuksesta liikeideaksi

Opinnäytetyöprojektin edetessä voimistui ajatus, moniko asiakas olisi esimerkiksi sähkösaneerauksen yhteydessä kiinnostunut liittämään älykästä ohjausta taloonsa. Tietoa hankkiessani löysin useita artikkeleita aiheesta, jossa nimenomaan kuluttajille suunnattuja valaistusjärjestelmiä ja älykotiratkaisuja on punnittu ja vaikuttaakin siltä, että yleinen kiinnostus on levinnyt harrastelijoiden ja pioneerien lisäksi myös tavallisten kansalaisten joukkoon [8]. Oli siis aika miettiä tällaisen ratkaisun hyödyntämistä tulevaisuuden työpaikkana, sekä realistisesti ajatellen idean toteutuksen kannattavuutta.

5.1 Kustannukset

Tämän ratkaisun kustannukset jakautuivat seuraavasti:

- kotelo, läpiviennit, kourut, kiskot, riviliittimet, N/PE-kisko 298 €
- releet (ssr ja kaksi perinteistä relettä) 106 €
- optoerottimet 200 €
- sulakeriviliittimet 14 €
- päätypuristimet 10 €
- virtalähteet 68 €
- kytkin 30 €
- Arduinot, ethernet shieldit ja kannakkeet 120 €
- Raspberry Pi ja muistikortti 98 €
- pientarvikkeet, holkit ja johtimet 20 €
- verkkokaapelit 20 €
- Haaroitusliittimet 16€
- Kokonaiskustannus 1000€

Opinnäytetyössäni asennustyö ei ollut kustannuserä muutoin kuin ajan osalta, sillä sähkösuunnittelu ja verkkovirtakytkennät hoituivat tilaajan toimesta, sekä pienvirtakytkennät minun toimestani.

Huomioitavaa kuitenkin on, että kyseinen ratkaisu toteutettiin ominaisuudet ja osien laatu edellä, sekä tulevaisuutta varten. Esimerkiksi releitä on hankittu tuplat tulevaa laajennusta ajatellen. Vastaavanlainen toimiva, mutta pienempi järjestelmä on mahdollista toteuttaa huomattavasti edullisemmin:

- Raspberry Pi ja muistikortti 58 €
- Arduino, ethernet shield ja kannakkeet 13 €
- 2 kappaletta 8 releen kortteja sekä dupont-johdot 44 €
- virtalähde ja riviliittimet 25 €
- kotelointi 20 €
- kasaus ~30min. 30 €
- kokonaiskustannus 190 €

Tässä kokoonpanossa on etuna se, että asiakkaalle voitaisiin toteuttaa tarvittavan kokoinen paketti: jos tarve on hallita vain valoja tai kodin medialaitteita, ei ole tarvetta kuin muutamaan releeseen sekä yhteen Arduinoon ja Raspberry Pi:hin.

Jälkimmäisessä ratkaisussa on esimerkkinä laskettu ratkaisu, jossa puhelimella Home Assistantin käyttöliittymästä voitaisiin joko manuaalisesti tai automatisoituna hallita esimerkiksi auton lämmitystä, talon valaistusta tai salaojapumppaamoja. Tässä data liikkuisi vain serveriltä ulospäin, mutta jos dataa haluttaisiin sisään, voitaisiin toinen releistä vaihtaa optoerottimeksi, tai lisätä arduino niitä varten.

Kun kyseinen paketti kasattaisiin, osille laskettaisiin esimerkiksi 20% myyntikate, sekä tekijälle kohtuullinen palkka, olisi paketin hinta 200 - 300 euron välillä. Jos asennuskohde ei ole rakenteilla oleva talo, eli sähkömies tekisi muutaman tunnin kytkentätöitä, olisi koko paketin hinta noin 500 - 600 euron verran.

Lisäksi jälkimmäisen paketin hinnoitteluun valittiin verkkosähkökomponentit suomesta, niiden hinta on kiinasta tilattaviin noin nelinkertainen. Näin ollen kuitenkin vastuu releiden hajotessa olisi maahantuojalla. Suurempien myyntivolyymien kohdalla olisi kuitenkin perusteltua ryhtyä tuomaan maahan kaikki komponentit.

5.2 Kilpailevat järjestelmät

Älykkään kodin toteutukseen myytäviä kaupallisia järjestelmiä ovat esimerkiksi

- KNX
- Philips HUE
- IKEA Trådfri
- Loxone
- Apple HomeKit

Näissä hinta ja ominaisuudet ovat suurimpia vertailukohteita Home Assistant-pohjaiseen järjestelmään. IKEA:n, HomeKit:n ja Philipsin kohdalla hinta on verrattain edullinen: aloituspakkaukset ovat 80 - 120 € välillä, ja sisältävät kolme lamppua ja hallintalaitteen. Lisälamput ovat 40 - 60 € välillä. Kuitenkin nämä ratkaisut koskevat vain valaistusta. [10, 11.]

Kattavammissa ratkaisuissa (KNX, Loxone) hinnat taas nousevat suuresti: esimerkiksi KNX- komponenttien hinnat alkavat 400 € seutuja ja jatkuvat useampaan tuhanteen euroon. Näillä saadaan ominaisuuksia sekä luotettavuutta, mutta talon modernisoinnissa hinta voi nousta hyvinkin suureksi. [12.]

5.3 Ratkaisu

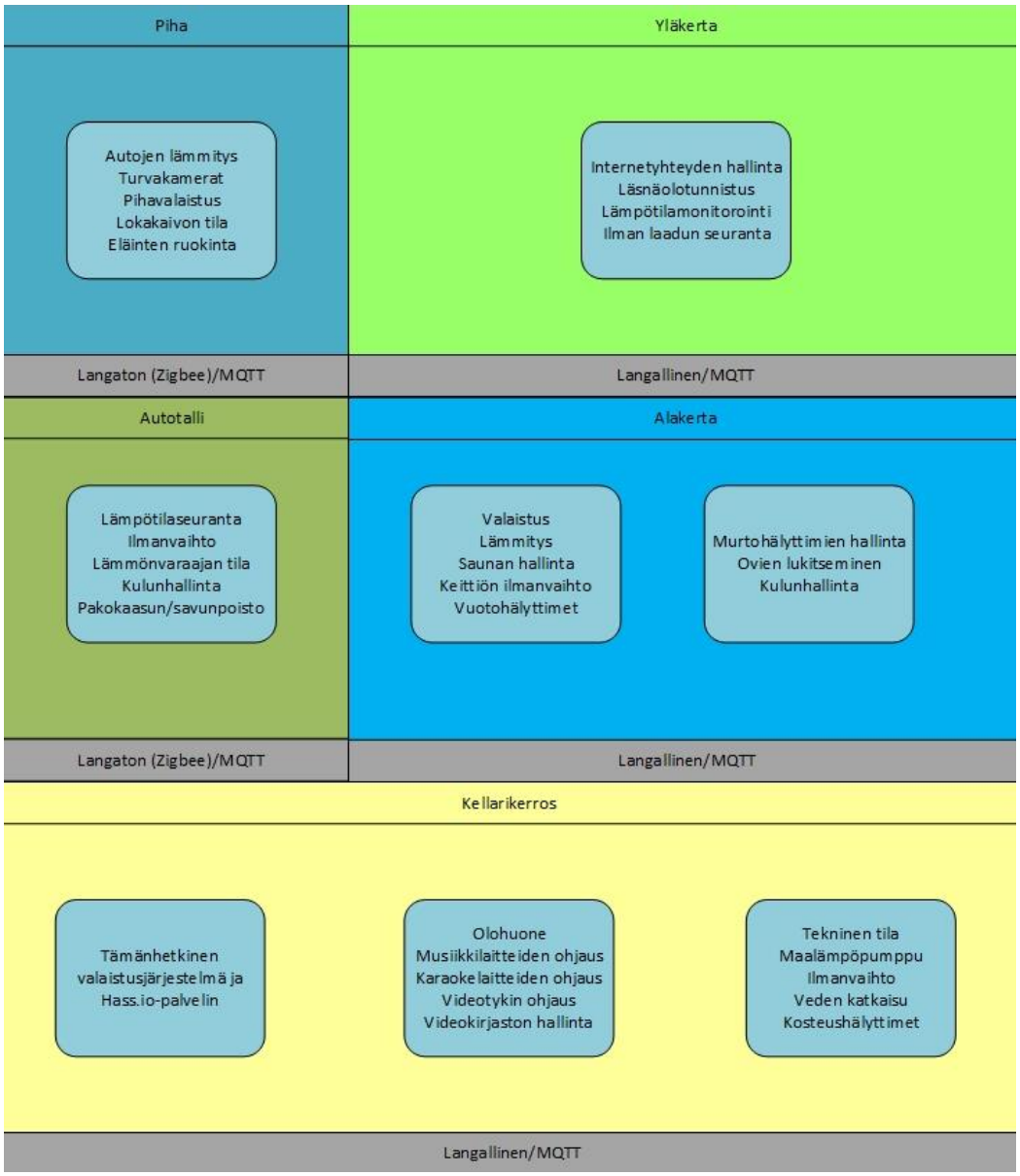
Näillä järjestelmillä on kuitenkin yksi yhteinen piirre: Home Assistant tukee kaikkia edellä mainittuja komponentteja [13]. Näin ollen asiakkaan ei tarvitse heittää pois olemassa olevaa järjestelmää, vaan se voidaan liittää uuden järjestelmän alle ja näin modernisoida talon järjestelmä avoimeksi ratkaisuksi.

Tämän lisäksi löydettäessä miellyttäviä komponentteja (esimerkiksi HomeKit-musiikkiohjain) on mahdollista ottaa se käyttöön ilman tarvetta pakollisille osahankinnoille kyseiseltä valmistajalta.

6 Järjestelmä tulevaisuudessa

Opinnäytetyöni käsitellessä pääosin valaistusta, on järjestelmä kuitenkin rakennettu tulevaisuutta varten, ja se tuleekin tulevaisuudessa pitämään sisällään paljon enemmän ominaisuuksia. Näihin lukeutuu sensoritiedon hyödyntäminen, jolla pystytään mahdollistamaan erilaiset hälytykset. Tämän sensoridatan sisältö voi vaihdella paljonkin, ja vaikka suurta osaa siitä ei välttämättä tarvitsekaan tallentaa arkistointia varten (lokakaivon tila, kosteushälyttimet), on osalla merkitystä talon juoksevien kulujen seurannassa ja ennustuksessa, esimerkiksi maalämpöpumpun käyntiasteella ja aurinkopaneelien tuotolla.

Seuraavassa kuvassa (Kuva 22) olen kartoittanut mahdollisia laajennuskohteita niin talon sisä- kuin ulkopuolellakin. Valtaosan liittämiseen voidaan käyttää talon kiinteää sisäverkkoa, mutta pihalle sijoitettavissa sensoreissa voi esimerkiksi langattoman Zigbee-yhteyden käyttö olla perusteltua. [14.]



Kuva 22. Kehitysmahdollisuudet.

Suunniteltuihin ominaisuuksiin kuuluu myös kotona/poissa -asetuksen luonti. Tällä saataisiin automatisoitua se, että kotoa poistuessa talon murtohälyttimet kytkeytyvät päälle ja veden tulo katkeaa. Näillä pystytään myös ennaltaehkäisemään kiinteistöön kohdistuvia potentiaalisia vahinkoja. [15.]

7 Yhteenveto

Avoimen lähdekoodin älykotijärjestelmät ovat muiden avoimen lähdekoodin projektien tavoin hyvin usein harrasteprojekteja, joita ajavat eteenpäin motivoituneet osaajat. Vaikka avoin kehitys onkin joidenkin yritysten kohdalla jo arkipäivää, eivät Home Assistantin kaltaiset projektit saa välttämättä osakseen kokopäiväistä kehityspanosta. Tästä johtuen uusien ominaisuuksien lisääminen sekä ohjelmointivirheiden paikkaaminen voivat viedä aikaa. Myös ongelmakohtiin ratkaisun löytäminen ei ole taattua, sillä aktiivisesta yhteisöstä huolimatta voi tulla eteen tilanteita, johihin kukaan ei osaa tai kerkeä vastata.

Hyödyt tämänkaltaisen järjestelmän hyödyntämisestä ovat kuitenkin selkeät. Koska projekteja ei ajeta rahallisella motiivilla (kehittäjiä voi tukea esimerkiksi Patreon-palvelun kautta), ei ominaisuuksia kehittäessä mennä helpointa reittiä, vaan se tehdään oikein ja toimivaksi. Lisäksi vältetään se, että suurempien yritysten agendat vaikuttaisivat kehitykseen: tuesta ei suljeta ulkopuolelle kenenkään valmistamia laitteita, vaan jokainen saa kehittää ja tätä kautta laajentaa projektin kattavuutta.

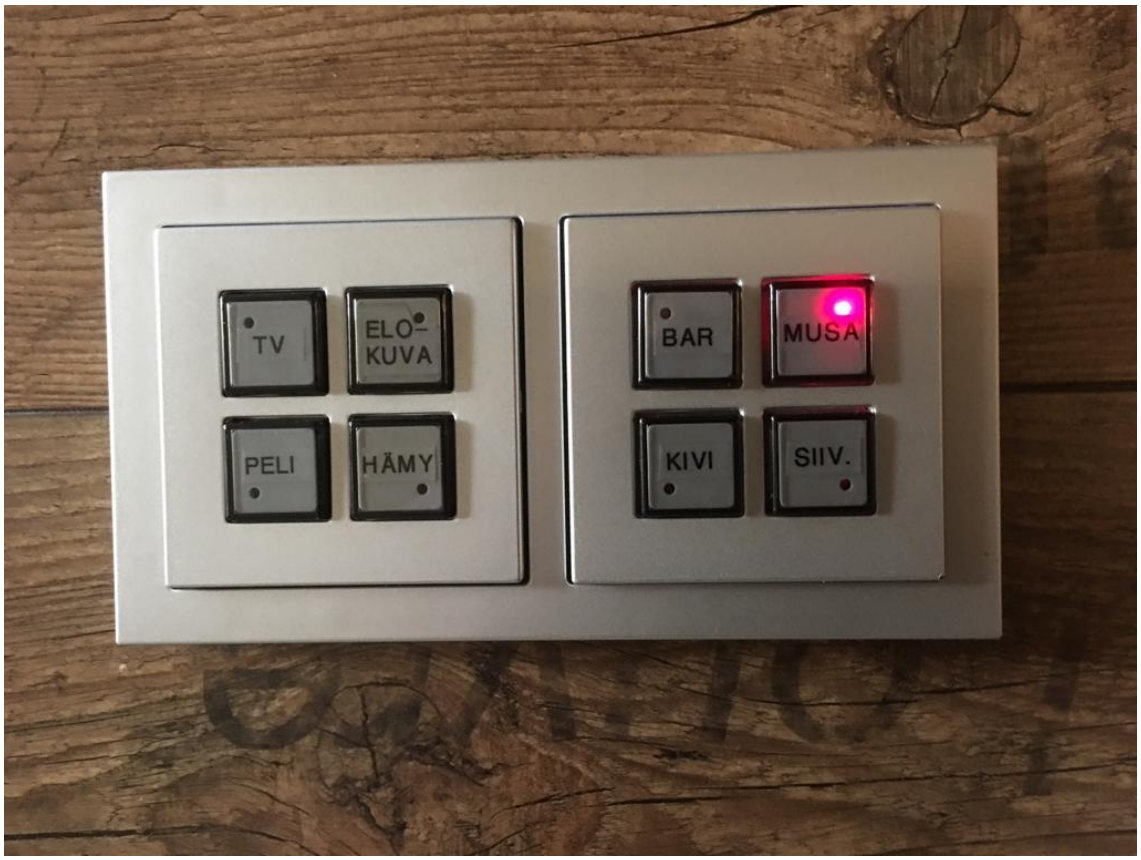
Vaikka avoimen lähdekoodin projektit muuttuvat koko ajan käyttäjäystävällisemmiksi asennuspaketteineen ja ohjeistuksineen, ei tällaista lähestymistapaa voi suositella kaikille. Ongelmia tulee eteen lähes poikkeuksetta, mitään ei käytännössä löydy ”plug-and-play” -valmiudella, vaan usein vaaditaan vähintäänkin konfigurointia tiedostoon. Tästä johtuen ilman kokemusta ohjelmoinnista voi tällainen lähestymistapa muuttua ikuisuusprojektiksi. Kuitenkin jo hyvin pienelläkin ymmärryksellä asiasta pystyy luomaan yksinkertaisen konfiguraation sekä käyttöliittymän muokkaukset ja automaatiot, ja näin ollen saavuttamaan toimivan järjestelmän. Järjestelmän dokumentointi on kattava, ja yhteisö aktiivinen. Törmätessä ongelmaan on todennäköistä, että joku on jo ratkaissut sen. Suurin panostus tämän tyyppisessä ratkaisussa on aika, jota työssä käyvällä ei usein ole tarpeeksi muutenkaan. Tämä lähestymistapa kuitenkin mahdollistaa suuremman

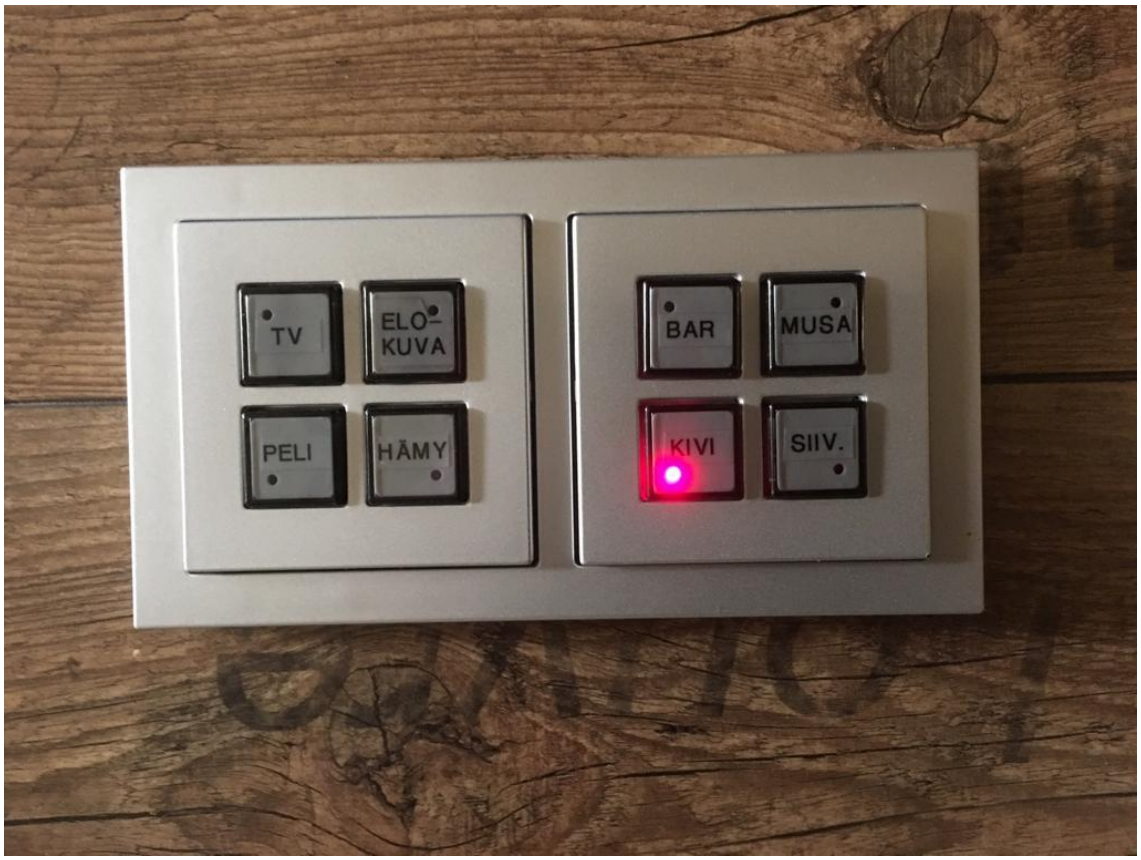
muokattavuuden, alemmat kustannukset sekä toimii jatkuvana oppimisympäristönä aiheesta kiinnostuneelle.

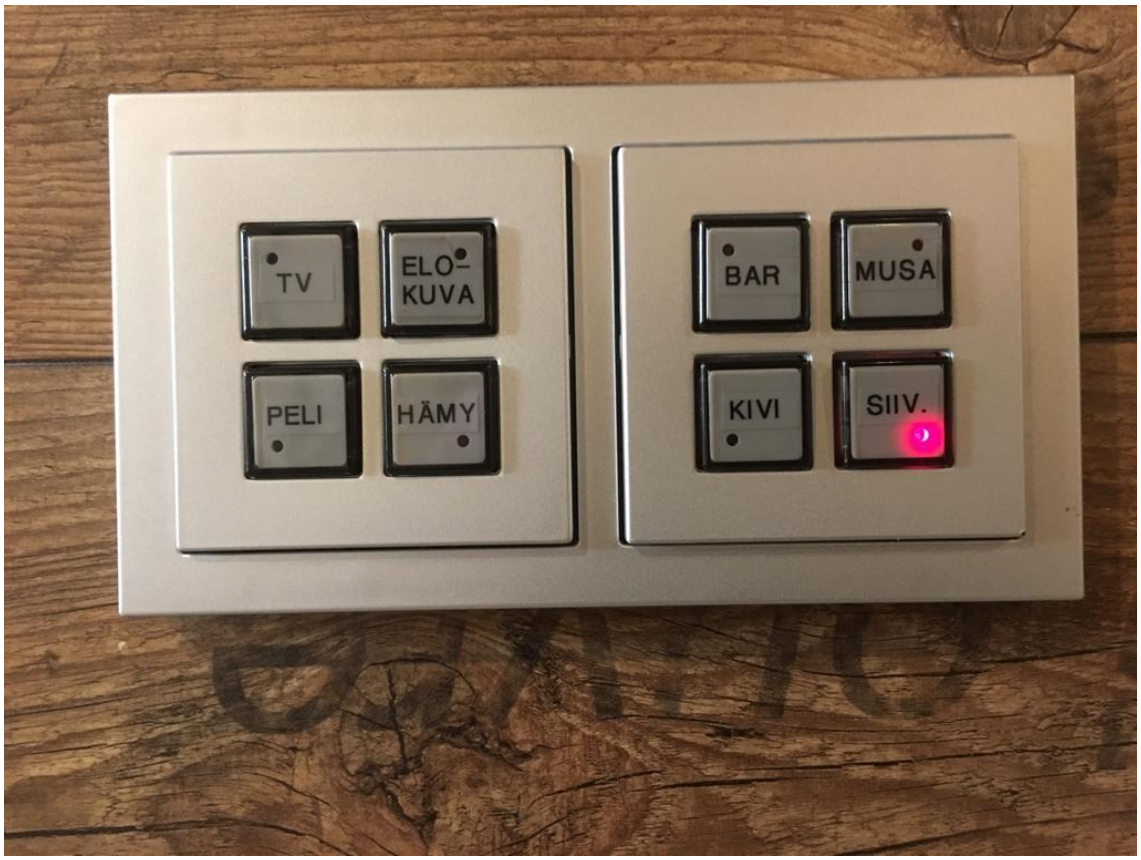
Lähteet

1. Lueth, K. 2018. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. IoT Analytics. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.
2. The ThingsBoard Authors. 2019. What is ThingsBoard?. ThingsBoard Documentation. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/.
3. Home Assistant. 2018. Home Assistant vs. Hass.io. Home Assistant. [Viitattu 6.5.2019] Saatavissa: https://www.home-assistant.io/faq/ha-vs-hassio/.
4. Arduino. 2019. Getting Started with Arduino and Genuino products. ARDUINO. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.arduino.cc/en/Guide/HomePage.
5. The ThingsBoard Authors. 2019. IoT Gateway. ThingsBoard IoT Gateway. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://thingsboard.io/docs/iot-gateway/.
6. Home Assistant. 2019. Install Home Assistant. Home Assistant. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.home-assistant.io/getting-started/.
7. Alex Mayer. 2017. The 3-2-1 Backup Rule – An Efficient Data Protection Strategy. NAKIVO Blog. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.nakivo.com/blog/3-2-1-backup-rule-efficient-data-protection-strategy/.
8. Østergaard, M. 2018. Tee kodistasi älykäs. Älykoti. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://kotimikro.fi/yhteiskunta/alykoti/tee-kodistasi-alykas.
9. Bondarik, E. 2019. How IoT for Senior Care Promotes Autonomy and Social Inclusion. The Doctor Weighs In. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://thedoctorweighsin.com/iot-senior-care-autonomy-social-inclusion/.
10. Cruz, B. 2017. IS IKEA'S SMART HOME PLATFORM REALLY THAT CHEAP?. HomeAlarmReport. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://homealarmreport.com/ikea-smart-home-price-comparison/.
11. Charlton, A. 2018. How much does it cost to turn the average house into a smart home?. GearBrain. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.gearbrain.com/average-smart-home-build-cost-2589554626.html.
12. SMARTech.pl. 2019. How much does a smart home system cost?. SMARThome.eu. [Online]. [Viitattu 6.5.2019]. Saatavissa: http://www.smarthome.eu/how-much-does-a-smart-home-system-cost.
13. Home Assistant. 2019. KNX. Home Assistant. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.home-assistant.io/components/knx/.
14. Zigbee Alliance. 2019. What is Zigbee?. Zigbee Alliance. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.zigbee.org/what-is-zigbee/.

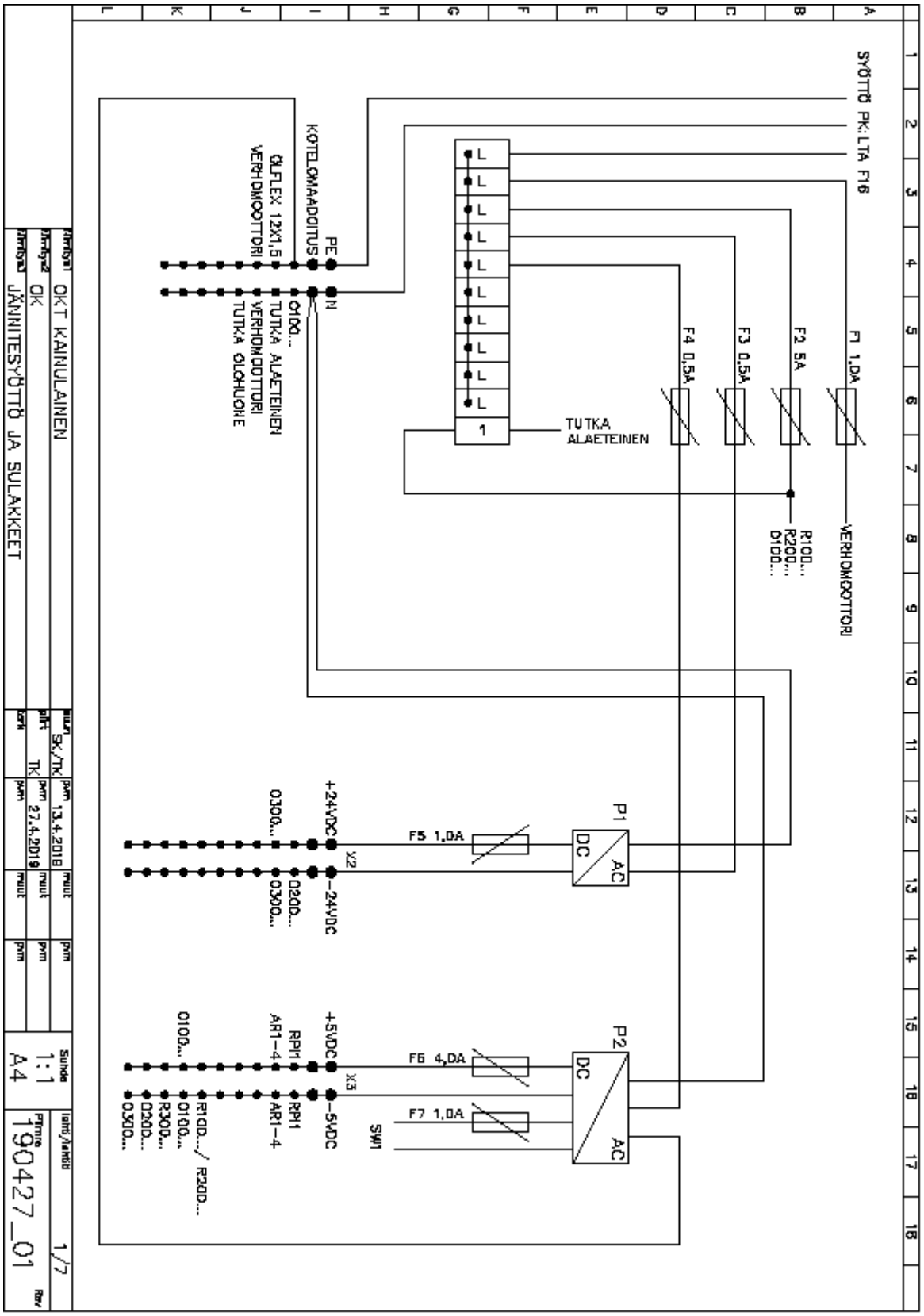
15. Tyrell smart home. 2018. Advantages of a Smart Home. Tyrell smart home. [Online]. [Viitattu 6.5.2019]. Saatavissa: https://www.tyrrellsmart-home.com/2018/12/23/advantages-of-a-smart-home/.


TV
ELO-
KUVA
PELI
HÄMY
BAR
MUSA
KIVI
SIIV.

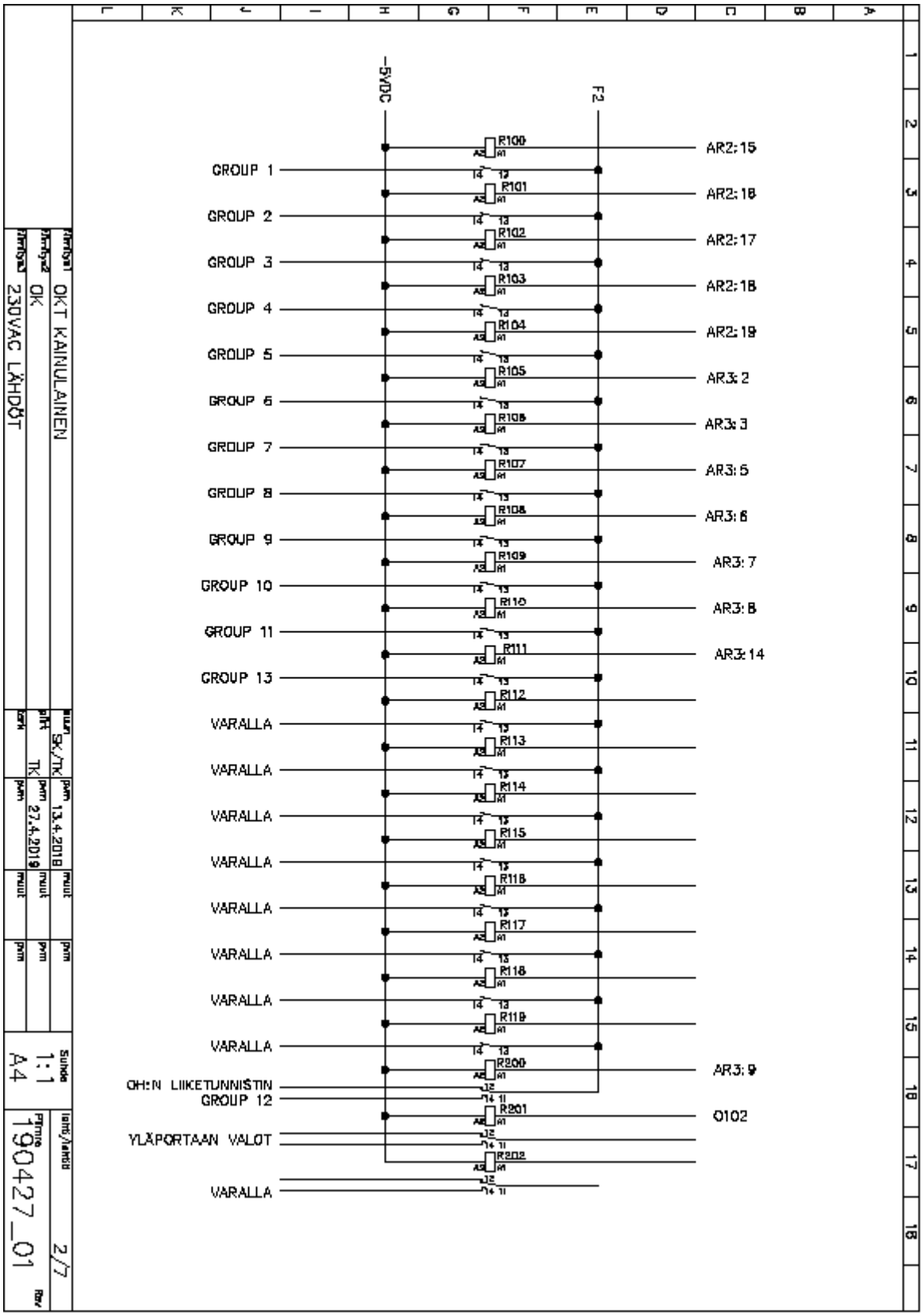

TV
ELO-
KUVA
PELI
HÄMY
BAR
MUSA
KIVI
SIIV.

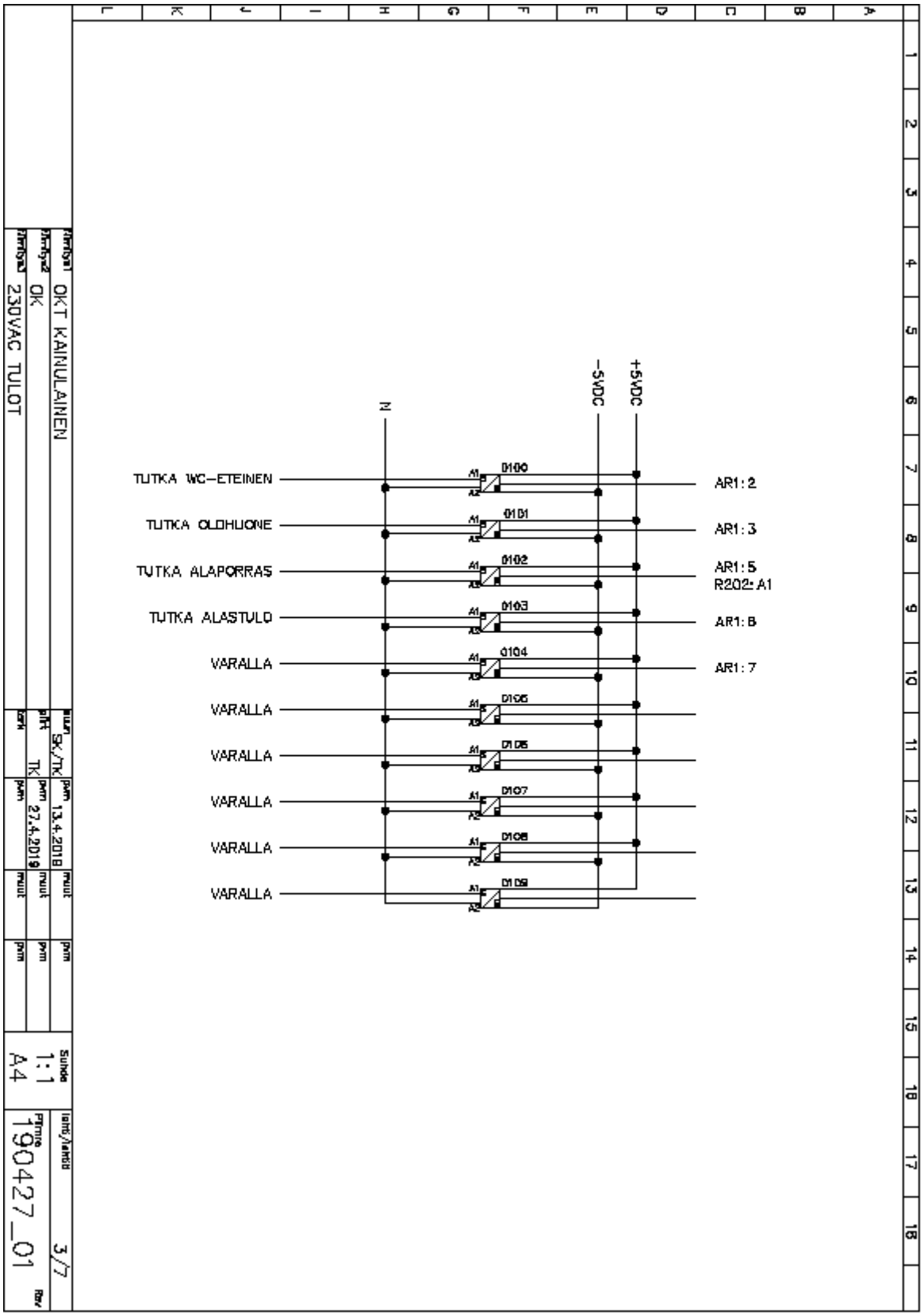

TV
ELO-
KUVA
PELI
HÄMY
BAR
MUSA
KIVI
SIIV.

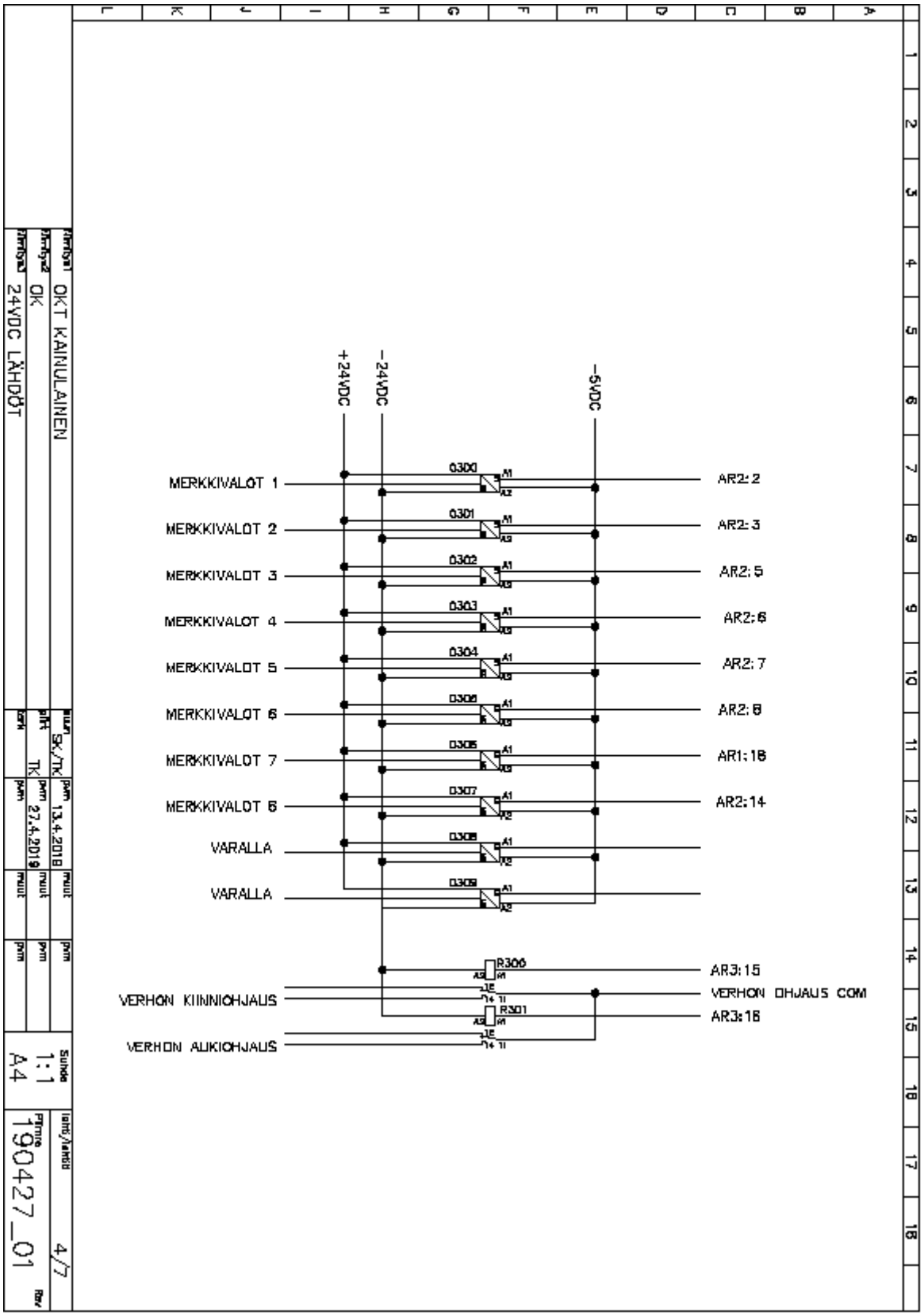

TV
ELO-
KUVA
PELI
HÄMY
BAR
MUSA
KIVI
SIIV.

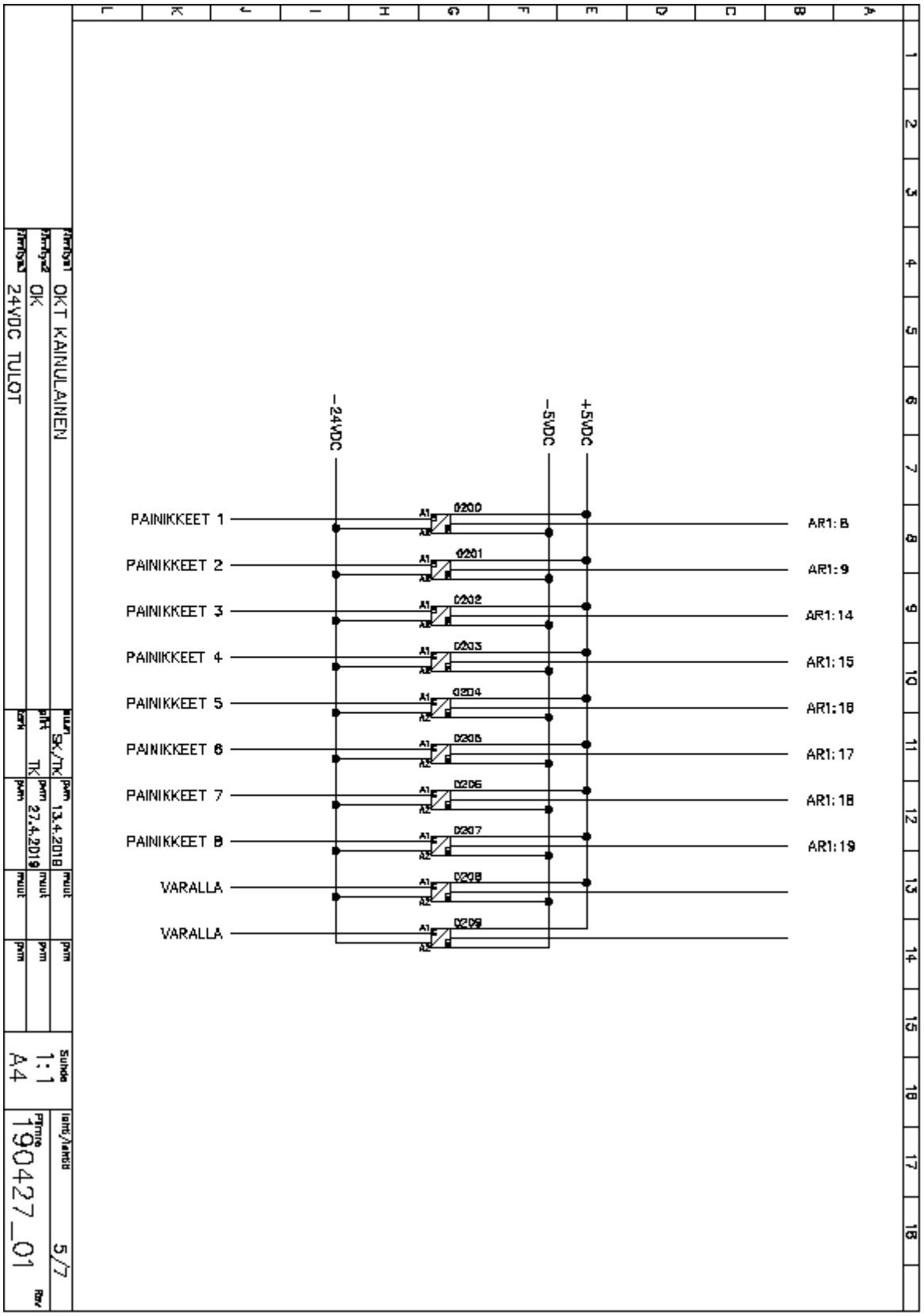


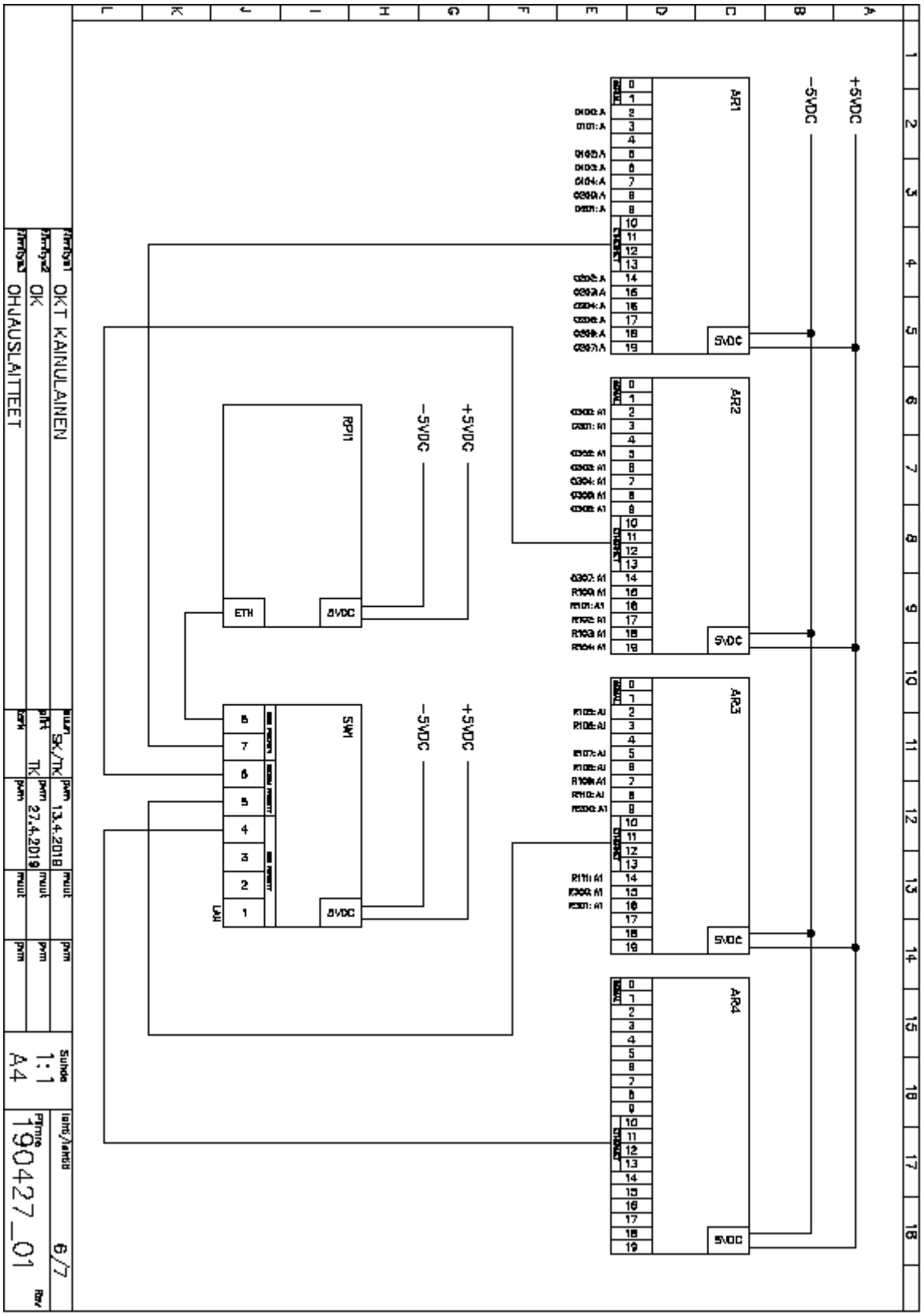
Piirtäjä	OKT KAINULAINEN										
Tarkast.	OK										
Piirros	JÄNNITESYÖTTÖ JA SULAKKEET										
		Muut.	SK./TK	pvm						Suhde	Lehti/Lehtiä
		SPR	TK	13.4.2018						1:1	1/7
		pvm	pvm	27.4.2019						A4	
		muut	muut	muut						Piirros	Rev
		pvm	pvm	pvm						190427_01	

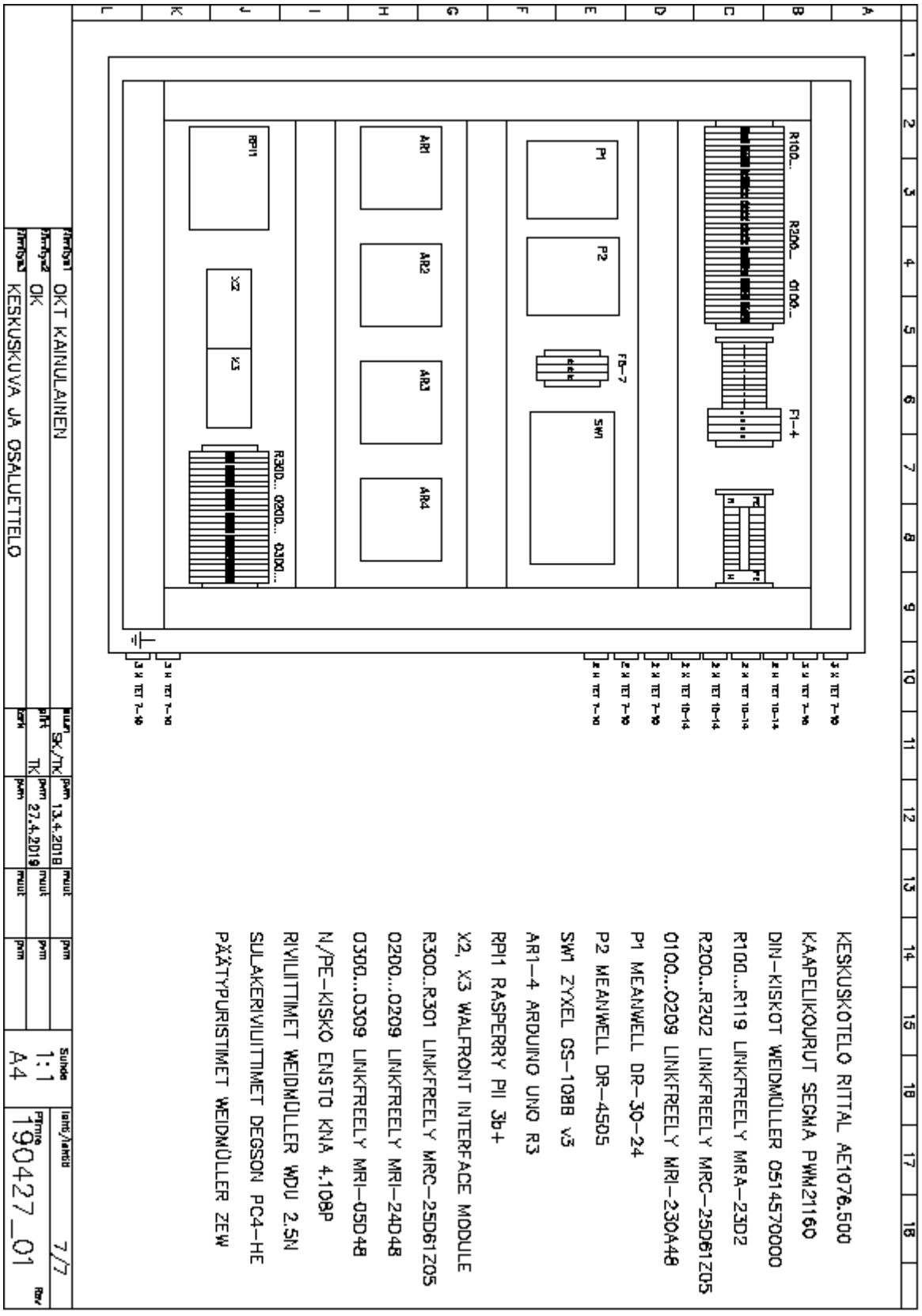

-5VDC
F2
R100
R101
R102
R103
R104
R105
R106
R107
R108
R109
R110
R111
R112
R113
R114
R115
R116
R117
R118
R119
R200
R201
R202
GROUP 1
GROUP 2
GROUP 3
GROUP 4
GROUP 5
GROUP 6
GROUP 7
GROUP 8
GROUP 9
GROUP 10
GROUP 11
GROUP 13
VARALLA
VARALLA
VARALLA
VARALLA
VARALLA
VARALLA
VARALLA
VARALLA
OH:N LIIKETUNNISTIN
GROUP 12
YLÄPORTAAN VALOT
VARALLA
AR2:15
AR2:16
AR2:17
AR2:18
AR2:19
AR3:2
AR3:3
AR3:5
AR3:6
AR3:7
AR3:8
AR3:14
AR3:9
Q102
OKT KAINULAINEN
OK
230VAC LÄHDÖT
SK/TK
TK
13.4.2018
27.4.2019
Suhde
1:1
A4
Piirros
190427_01
2/7

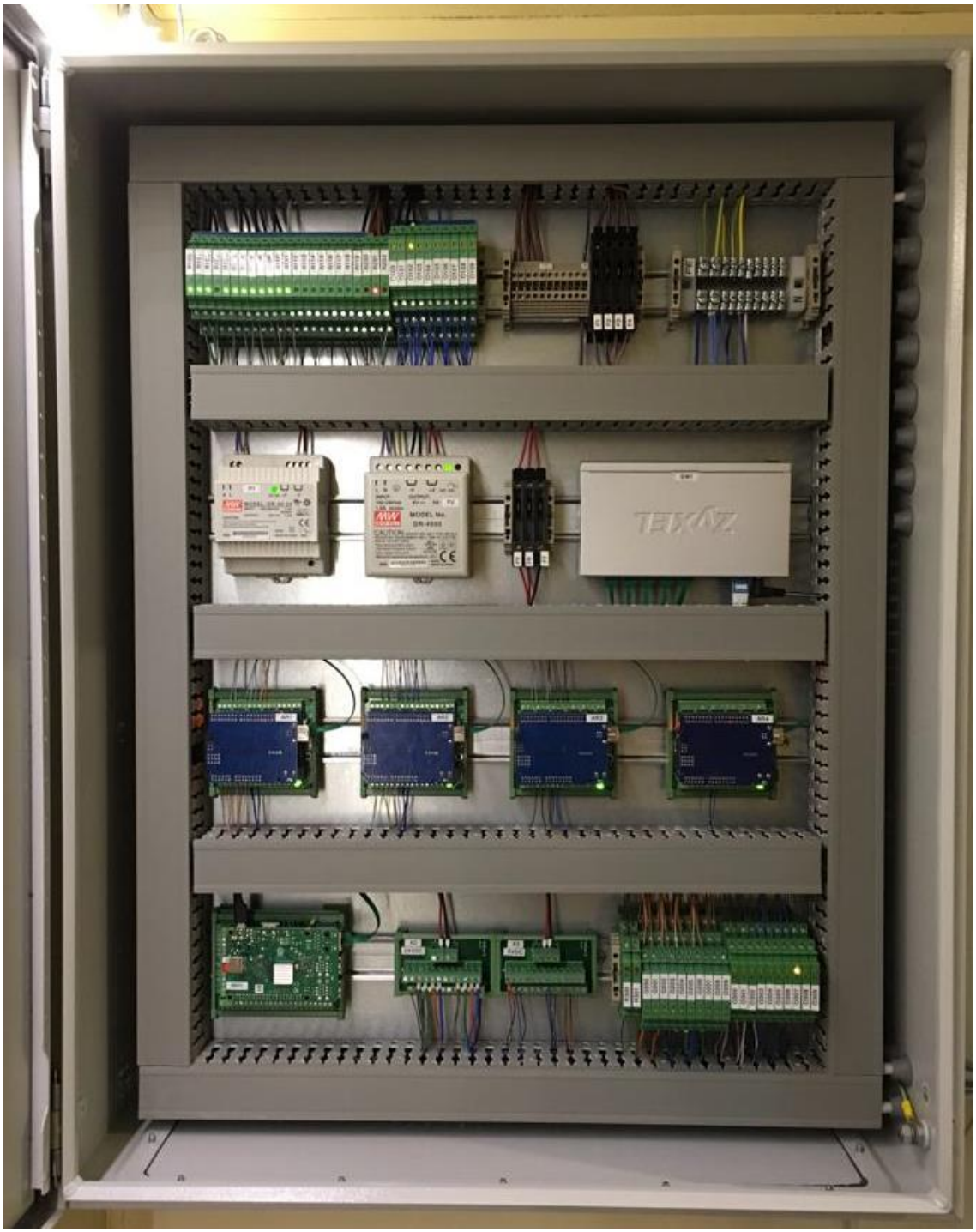

-5VDC
+5VDC
N
TUTKA WC-ETEINEN
TUTKA OLOHUONE
TUTKA ALAPORRAS
TUTKA ALASTULO
VARALLA
VARALLA
VARALLA
VARALLA
VARALLA
D100
D101
D102
D103
D104
D105
D106
D107
D108
D109
AR1:2
AR1:3
AR1:5
R202:A1
AR1:6
AR1:7
OKT KAINULAINEN
OK
230VAC TULOT
SK/TK
TK
13.4.2018
27.4.2019
3/7
1:1
A4
190427_01

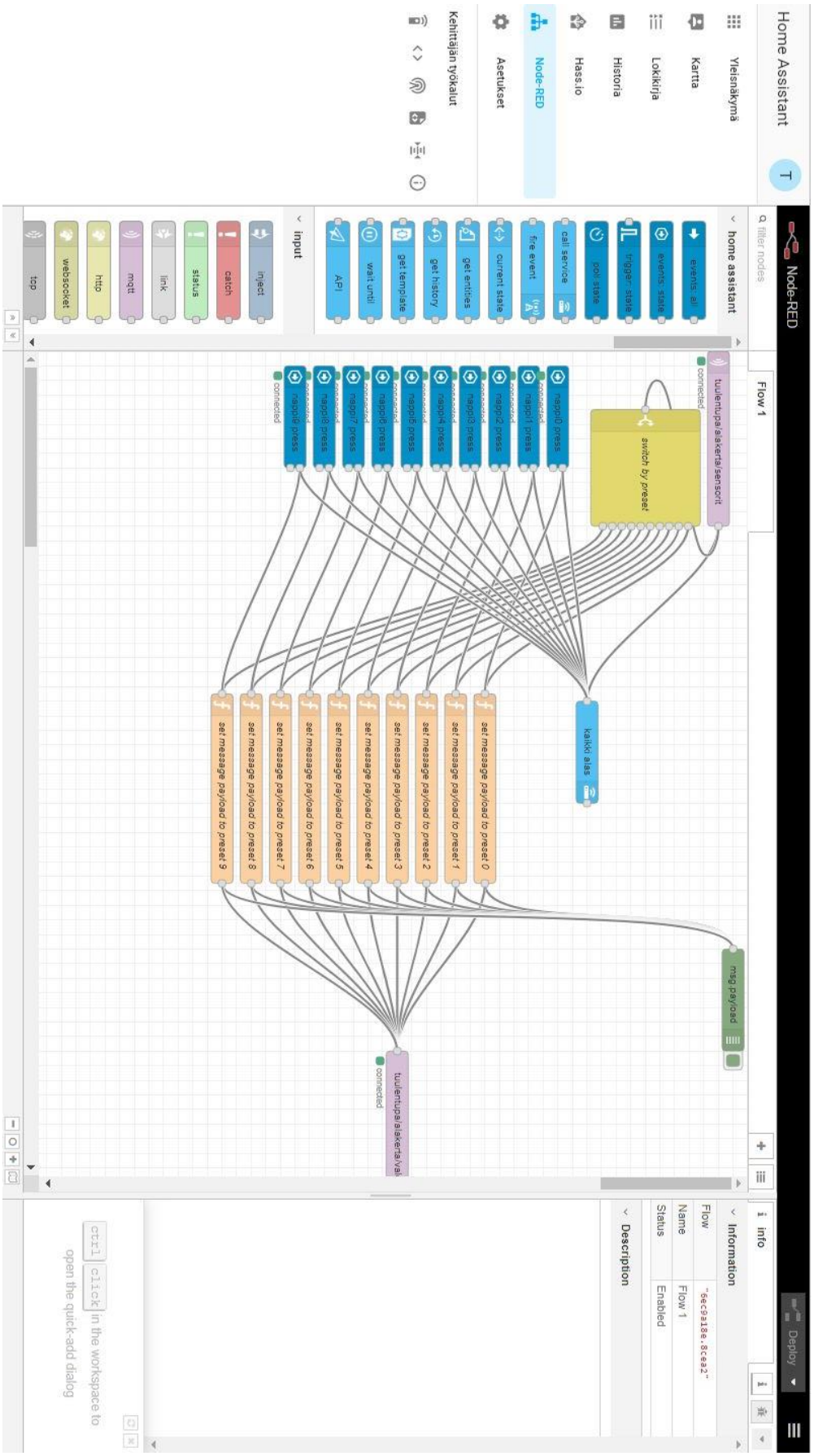

+24VDC
-24VDC
-5VDC
MERKKIVALOT 1
MERKKIVALOT 2
MERKKIVALOT 3
MERKKIVALOT 4
MERKKIVALOT 5
MERKKIVALOT 6
MERKKIVALOT 7
MERKKIVALOT 8
VARALLA
VARALLA
VERHON KIINNIOHJAUS
VERHON ALIKIOHJAUS
VERHON OHJAUS COM
AR2:2
AR2:3
AR2:5
AR2:6
AR2:7
AR2:8
AR1:16
AR2:14
AR3:15
AR3:16
K300
K301
K302
K303
K304
K306
K305
K307
K308
K309
R306
R301
Nimitys1 OKT KAINULAINEN
Nimitys2 OK
Nimitys3 24VDC LÄHDÖT
Suunn. SK/TK
pvm 13.4.2018
pvm 27.4.2019
Suhde 1:1
A4
Piirros 190427_01
Rev
Sivu/sivuja 4/7

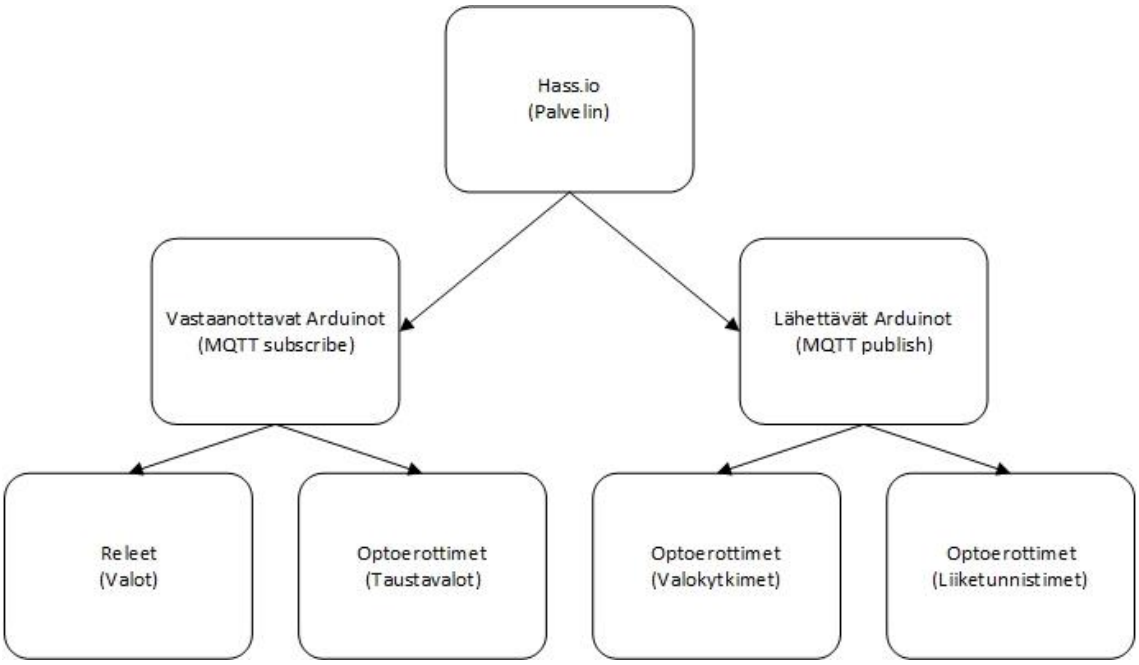

-24VDC
-5VDC
+5VDC
PAINIKKEET 1
PAINIKKEET 2
PAINIKKEET 3
PAINIKKEET 4
PAINIKKEET 5
PAINIKKEET 6
PAINIKKEET 7
PAINIKKEET 8
VARALLA
VARALLA
D200
D201
D202
D203
D204
D205
D206
D207
D208
D209
A1
A2
AR1: 8
AR1: 9
AR1: 14
AR1: 15
AR1: 16
AR1: 17
AR1: 18
AR1: 19
OKT KAINULAINEN
OK
24VDC TULOT
SK/TK
TK
13.4.2018
27.4.2019
Suhde
1:1
A4
Piirros
190427_01
5/7

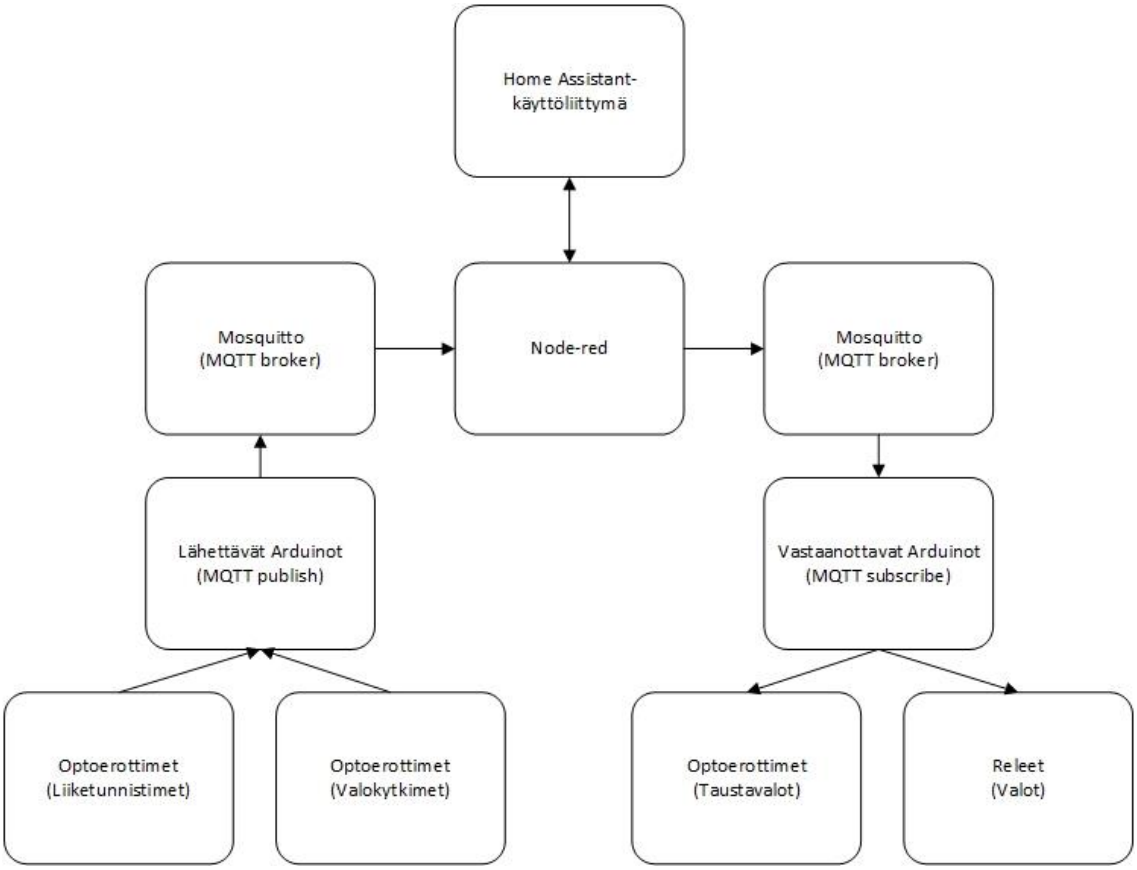

+5VDC
-5VDC
AR1
AR2
AR3
AR4
5VDC
RPI1
ETH
5VDC
SW1
LAN
OKT KAINULAINEN
OK
OHJAUSLAITTEET
SK/TK
13.4.2018
27.4.2019
1:1
A4
190427_01
6/7


KESKUSKOTELO RITTAL AE1076.500
KAAPELIKOURUT SEGMA PWM21160
DIN-KISKOT WEIDMÜLLER 0514570000
R100...R119 LINKFREELY MRA-23D2
R200..R202 LINKFREELY MRC-25D61Z05
0100...0209 LINKFREELY MRI-230A48
P1 MEANWELL DR-30-24
P2 MEANWELL DR-4505
SW1 ZYXEL GS-108B v3
AR1-4 ARDUINO UNO R3
RPI1 RASPERRY PII 3b+
X2, X3 WALFRONT INTERFACE MODULE
R300..R301 LINKFREELY MRC-25D61Z05
0200...0209 LINKFREELY MRI-24D48
0300...0309 LINKFREELY MRI-05D48
N/PE-KISKO ENSTO KNA 4.10BP
RIVILIITTIMET WEIDMÜLLER WDU 2.5N
SULAKERIVILIITTIMET DEGSON PC4-HE
PÄÄTYPURISTIMET WEIDMÜLLER ZEW
KESKUSKUVA JA OSALUETTELO
OKT KAINULAINEN
OK
OK
1:1
A4
190427_01
7/7


TEKNZ


Home Assistant
Yleisnäkymä
Kartta
Lokikirja
Historia
Hass.io
Node-RED
Asetukset
Kehittäjän työkalut
Node-RED
Flow 1
home assistant
events: all
events: state
trigger: state
poll state
call service
fire event
current state
get entities
get history
get template
wait until
API
input
inject
catch
status
link
mqtt
http
websocket
tcp
tuulentupa/askeltaja/sensorit
switch by preset
kaikki alas
msg.payload
tuulentupa/askeltaja/val
set message payload to preset 0
set message payload to preset 1
set message payload to preset 2
set message payload to preset 3
set message payload to preset 4
set message payload to preset 5
set message payload to preset 6
set message payload to preset 7
set message payload to preset 8
set message payload to preset 9
info
Information
Description
Deploy
ctrl click in the workspace to open the quick-add dialog


Hass.io
(Palvelin)
Vastaanottavat Arduinot
(MQTT subscribe)
Lähettävät Arduinot
(MQTT publish)
Releet
(Valot)
Optoerottimet
(Taustavalot)
Optoerottimet
(Valokytkimet)
Optoerottimet
(Liiketunnistimet)


Home Assistant-
käyttöliittymä
Mosquitto
(MQTT broker)
Node-red
Mosquitto
(MQTT broker)
Lähettävät Arduinot
(MQTT publish)
Vastaanottavat Arduinot
(MQTT subscribe)
Optoerottimet
(Liiketunnistimet)
Optoerottimet
(Valokytkimet)
Optoerottimet
(Taustavalot)
Releet
(Valot)

```
#include "Ethernet.h"
#include "PubSubClient.h"



//version 1.0
//this has separate topics for each pir and button, static memory has been used with strings
//HIGH level triggers, no debounce, no interrupt. doublepress not on this end.
//previous state remembered to disable spamming to MQTT. only send message once per press.

#define RX 0
#define TX 1
#define pir1 2
#define pir2 3
#define eth1 4//eth?
#define pir3 5
#define pir4 6
#define pir5 7
#define btn1 8
#define btn2 9
#define eth2 10//eth?
#define eth3 11//eth?
#define eth4 12//eth?
#define eth5 13//eth?
#define btn3 14
#define btn4 15
#define btn5 16
#define btn6 17
#define btn7 18
#define btn8 19

#define topic0 "########/alakerta/nappi0"
#define topic1 "########/alakerta/nappi1"
#define topic2 "########/alakerta/nappi2"
#define topic3 "########/alakerta/nappi3"
#define topic4 "########/alakerta/nappi4"
#define topic5 "########/alakerta/nappi5"
#define topic6 "########/alakerta/nappi6"
#define topic7 "########/alakerta/nappi7"
```

```
#define topic8 "########/alakerta/nappi8"
#define topic9 "########/alakerta/nappi9"
#define topicPir1 "########/alakerta/pir1"
#define topicPir2 "########/alakerta/pir2"
#define topicPir3 "########/alakerta/pir3"
#define topicPir4 "########/alakerta/pir4"
#define topicPir5 "########/alakerta/pir5"



//mosquitto
uint8_t server[6] = {192, 168, ###, ###};
int port = ####;

//all pins listed for cycling them through
int pins[] = {RX, TX, pir1, pir2, eth1, pir3, pir4, pir5, btn1, btn2, eth2, eth3, eth4, eth5, btn3, btn4, btn5, btn6, btn7, btn8};

//device settings
byte mac[] = { #x##, #x##, #x##, #x##, #x##, #x## };
byte ip[] = { 192, 168, ###, ### };

//mosquitto
//const char username[] = "########";
//const char password[] = "########";
const char clientID[] = "########";
//const char topic[] = "########/alakerta/sensorit";

//state variables
bool btn1b = false;
bool btn2b = false;
bool btn3b = false;
bool btn4b = false;
bool btn5b = false;
bool btn6b = false;
bool btn7b = false;
bool btn8b = false;
bool pir1b = false;
bool pir2b = false;
```

```
bool pir3b = false;
bool pir4b = false;
bool pir5b = false;

EthernetClient ethClient;
PubSubClient client(server, port, ethClient);



//check if connections are allright and maintain the MQTT connection.
void maintainEth() {
  if (!ethClient.connected()) {
    Ethernet.begin(mac, ip);
    Serial.println(F("connecting ethernet"));
  }
  if (!client.connected()) {
    client.connect(clientID);
    //client.connect(clientID, username, password);
    Serial.println(F("connecting mqtt"));
  }
  else {
    //Serial.println(client.state());
  }
  client.loop();
}

void checkPIR() {
  if (digitalRead(pir1) != pir1b) {
    pir1b = digitalRead(pir1);
    if (btn1b) {
      publishData(String(F("pir")).c_str(), String(topicPir1).c_str(), 2);
    }
  }
  if (digitalRead(pir2) != pir2b) {
    pir2b = digitalRead(pir2);
    if (btn2b) {
      publishData(String(F("pir")).c_str(), String(topicPir2).c_str(), 2);
    }
  }
```

```
  if (digitalRead(pir3) != pir3b) {
    pir3b = digitalRead(pir3);
    if (btn3b) {
      publishData(String(F("pir")).c_str(), String(topicPir3).c_str(), 2);
    }
  }
  if (digitalRead(pir4) != pir4b) {
    pir4b = digitalRead(pir4);
    if (btn4b) {
      publishData(String(F("pir")).c_str(), String(topicPir4).c_str(), 2);
    }
  }
  if (digitalRead(pir5) != pir5b) {
    pir5b = digitalRead(pir5);
    if (btn5b) {
      publishData(String(F("pir")).c_str(), String(topicPir5).c_str(), 2);
    }
  }
}

//go through buttons to check if any are pressed
void checkButtons() {

  if (digitalRead(btn1) != btn1b) {
    btn1b = digitalRead(btn1);
    if (btn1b) {
      publishData(String(F("button")).c_str(), String(topic1).c_str(), 1);
    }
  }
  if (digitalRead(btn2) != btn2b) {
    btn2b = digitalRead(btn2);
    if (btn2b) {
      publishData(String(F("button")).c_str(), String(topic2).c_str(), 1);
    }
  }
  if (digitalRead(btn3) != btn3b) {
    btn3b = digitalRead(btn3);
    if (btn3b) {
```

```
      publishData(String(F("button")).c_str(), String(topic3).c_str(), 1);
    }
  }
  if (digitalRead(btn4) != btn4b) {
    btn4b = digitalRead(btn4);
    if (btn4b) {
      publishData(String(F("button")).c_str(), String(topic4).c_str(), 1);
    }
  }
  if (digitalRead(btn5) != btn5b) {
    btn5b = digitalRead(btn5);
    if (btn5b) {
      publishData(String(F("button")).c_str(), String(topic5).c_str(), 1);
    }
  }
  if (digitalRead(btn6) != btn6b) {
    btn6b = digitalRead(btn6);
    if (btn6b) {
      publishData(String(F("button")).c_str(), String(topic6).c_str(), 1);
    }
  }
  if (digitalRead(btn7) != btn7b) {
    btn7b = digitalRead(btn7);
    if (btn7b) {
      publishData(String(F("button")).c_str(), String(topic7).c_str(), 1);
    }
  }
  if (digitalRead(btn8) != btn8b) {
    btn8b = digitalRead(btn8);
    if (btn8b) {
      publishData(String(F("button")).c_str(), String(topic8).c_str(), 1);
    }
  }
}


void publishData(const char timeseries[], const char topic[], int i) {
  //publish data to MQTT topic. tmpstr is used to buld the string
```

```
  String tmpstr = F("");
  Serial.println(F("publishdata"));
  if (i == 1) {
    tmpstr = F("{\"sensorName\":\"########\", \"value\": ");
    tmpstr += timeseries;
    tmpstr += F(" }");
  }
  if (i == 2) {
    tmpstr = F("{\"sensorName\":\"########\", \"value\": ");
    tmpstr += timeseries;
    tmpstr += F(" }");
  }

  //client.publish(topic, tmp);
  client.publish(topic, (char*) tmpstr.c_str());
}

void setup() {
  for (int i = 0; i <= 19; i++) {
    if (i != 10 && i != 11 && i != 12 && i != 13 && i != 4) {
      pinMode(pins[i], INPUT);
    }
  }

  //Ethernet.begin(mac, ip);
  Serial.begin(9600);
  //Serial.println("setup done");

  //client.connect(clientID, username, password);
  if (client.connect(clientID)) {
    //Serial.println("mqtt connection up");
  }
  else {
    //Serial.println(client.state());
  }
}

void loop() {
```

```
  checkButtons();
  checkPIR();
  maintainEth();
}
```

```
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Ethernet.h>

//version 1.0
//active HIGH

#define RX 0
#define TX 1
#define back1 2
#define back2 3
#define eth1 4//eth?
#define back3 5
#define back4 6
#define back5 7
#define back6 8
#define back7 9
#define eth2 10//eth?
#define eth3 11//eth?
#define eth4 12//eth?
#define eth5 13//eth?
#define back8 14
#define group1 15
#define group2 16
#define group3 17
#define group4 18
#define group5 19



uint8_t server[6] = {192, 168, ###, ###};
int port = ####;



int pins[] = {RX, TX, back1, back2, eth1, back3, back4, back5, back6, back7, eth2, eth3, eth4, eth5, back8, group1,
group2, group3, group4, group5};


bool sub = false;
```

```
byte mac[] = { #x##, #x##, #x##, #x##, #x##, #x## };
byte ip[] = { 192, 168, ###, ### };

const char clientID[] = "########";
const char topic[] = "########/alakerta/valot";

EthernetClient ethClient;
PubSubClient client(server, port, ethClient);


//this is triggered, when message arrives to subscribed topic
void callback(char* intopic, byte* payload, unsigned int length) {
  Serial.println("callback");
  //we go through the received payload
  StaticJsonBuffer<500> jsonBuffer;
  JsonObject& payloadObj = jsonBuffer.parseObject(payload);
  changeBacklight(payloadObj["preset"]);
  changeLights(group1, payloadObj["group1"]);
  changeLights(group2, payloadObj["group2"]);
  changeLights(group3, payloadObj["group3"]);
  changeLights(group4, payloadObj["group4"]);
  changeLights(group5, payloadObj["group5"]);
  Serial.println("----------------end of print-------------------------");
}


void formatBacklight() {
  digitalWrite(back1, LOW);
  digitalWrite(back2, LOW);
  digitalWrite(back3, LOW);
  digitalWrite(back4, LOW);
  digitalWrite(back5, LOW);
  digitalWrite(back6, LOW);
  digitalWrite(back7, LOW);
  digitalWrite(back8, LOW);
}
```

```
//change lights according to input
void changeLights(int pin, bool a) {
  //if pin's state isn't same as requested, change it.
  digitalWrite(pin, a);
  Serial.print("pin: ");
  Serial.println(pin);
  Serial.print("value: ");
  Serial.println(a);
  Serial.println("");
  Serial.print("testing: ");
  Serial.println(digitalRead(pin));
}


//change backlights according to input
void changeBacklight(int i) {
  //clear current backight setting

  formatBacklight();

  //change backlight according to selected preset
  switch (i) {
    case 1:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back1, HIGH);
      break;

    case 2:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back2, HIGH);
      break;

    case 3:
      Serial.print("backlight ");
      Serial.println(i);
```

```
      digitalWrite(back3, HIGH);
      break;

    case 4:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back4, HIGH);
      break;

    case 5:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back5, HIGH);
      break;

    case 6:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back6, HIGH);
      break;

    case 7:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back7, HIGH);
      break;

    case 8:
      Serial.print("backlight ");
      Serial.println(i);
      digitalWrite(back8, HIGH);
      break;

    default:
      break;

  }
}
```

```
//check if connections are allright and maintain the MQTT with loop
void maintainEth() {
  if (!ethClient.connected()) {
    Ethernet.begin(mac, ip);
    Serial.println("reconnected ethernet");
  }
  if (!client.connected()) {
    client.connect(clientID);
    //client.connect(clientID, username, password);
    Serial.println("reconnected mqtt");
  }
  if (client.connected()) {
    if (sub) {}
    else {
      sub = client.subscribe("########/alakerta/valot");
      if (sub) {
        Serial.println("sub success");
      }
      else {
        Serial.println("sub fail");
      }
    }
  }
  if (client.state() != 0) {
    Serial.println(client.state());
  }
  client.loop();
}


void setup() {
  client.setCallback(callback);
  for (int i = 0; i <= 19; i++) {
    if (i != 10 && i != 11 && i != 12 && i != 13 && i != 4 && i != 0 && i != 1) {
      pinMode(pins[i], OUTPUT);
      digitalWrite(i, LOW);
```

```
    }
  }
  Serial.begin(9600);
  while (true) {
    if (!ethClient.connected()) {
      Ethernet.begin(mac, ip);
      Serial.println("connected ethernet");
      break;
    }
  }
  while (true) {
    if (!client.connected()) {
      client.connect(clientID);
      //client.connect(clientID, username, password);
      Serial.println("connected mqtt inside setup");
      break;
    }
  }
}

void loop() {
  maintainEth();
  //Serial.println("test");
}
```

```
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Ethernet.h>

//Version 1.0
//active HIGH, pulse signal to curtains, pinouts in order

#define RX 0
#define TX 1
#define group6 2
#define group7 3
#define eth1 4//eth?
#define group8 5
#define group9 6
#define group10 7
#define group11 8
#define group12 9
#define eth2 10//eth?
#define eth3 11//eth?
#define eth4 12//eth?
#define eth5 13//eth?
#define group13 14
#define onVerho 15
#define offVerho 16
#define nan1 17
#define nan2 18
#define nan3 19



uint8_t server[6] = {192, 168, ###, ###};
int port = ####;

//curtain control pulse length
int verhoDelay = 50;

int pins[] = {RX, TX, group6, group7, eth1, group8, group9, group10, group11, group12, eth2, eth3, eth4, eth5, group13, onVerho, offVerho, nan1, nan2, nan3};

bool sub = false;

byte mac[] = { #x##, #x##, #x##, #x##, #x##, #x## };
```

```
byte ip[] = { 192, 168, ###, ### };

const char clientID[] = "########";
const char topic[] = "########/alakerta/valot";

EthernetClient ethClient;
PubSubClient client(server, port, ethClient);


//this is triggered, when message arrives to subscribed topic
void callback(char* intopic, byte* payload, unsigned int length){
  //we go through the received payload
  StaticJsonBuffer<500> jsonBuffer;
  //change lights according to request
  JsonObject& payloadObj = jsonBuffer.parseObject(payload);
  changeLights(group6, payloadObj["group6"]);
  changeLights(group7, payloadObj["group7"]);
  changeLights(group8, payloadObj["group8"]);
  changeLights(group9, payloadObj["group9"]);
  changeLights(group10, payloadObj["group10"]);
  changeLights(group11, payloadObj["group11"]);
  changeLights(group12, payloadObj["group12"]);
  changeLights(group13, payloadObj["group13"]);
  changeVerho(payloadObj["verho"]);
  Serial.println("----------------end of print-------------------------");
}

//sen pulse data to curtains
void changeVerho(bool verho){
  if(verho){
    digitalWrite(onVerho, HIGH);
    delay(verhoDelay);
    digitalWrite(onVerho, LOW);
  }
  else
  {
    digitalWrite(offVerho, HIGH);
    delay(verhoDelay);
    digitalWrite(offVerho, LOW);
  }
}

//change lights according to input
```

```
void changeLights(int pin, bool a) {
  //if pin's state isn't same as requested, change it.
  digitalWrite(pin, a);
  Serial.print("pin: ");
  Serial.println(pin);
  Serial.print("value: ");
  Serial.println(a);
  Serial.println("");
  Serial.print("testing: ");
  Serial.println(digitalRead(pin));
}


//check if connections are allright and maintain the MQTT with loop
void maintainEth(){
  if (!ethClient.connected()) {
    Ethernet.begin(mac, ip);
    Serial.println("reconnected ethernet");
  }
  if (!client.connected()) {
    client.connect(clientID);
    //client.connect(clientID, username, password);
    Serial.println("reconnected mqtt");
  }
  if(client.connected()){
    if(sub){}
    else{
      sub = client.subscribe("########/alakerta/valot");
      if(sub){
        Serial.println("sub success");
      }
      else{
        Serial.println("sub fail");
      }
    }
  }
  if(client.state() != 0){
    Serial.println(client.state());
  }
  client.loop();
}
```

```
void setup() {
  client.setCallback(callback);
  for(int i = 0; i <= 19; i++){
    if (i != 10 && i != 11 && i != 12 && i != 13 && i != 4 && i != 0 && i != 1) {
      pinMode(pins[i], OUTPUT);
      digitalWrite(i, LOW);
    }
  }
  Serial.begin(9600);
  while(true){
    if (!ethClient.connected()) {
      Ethernet.begin(mac, ip);
      Serial.println("connected ethernet");
      break;
    }
  }
  while(true){
    if (!client.connected()) {
      client.connect(clientID);
      //client.connect(clientID, username, password);
      Serial.println("connected mqtt inside setup");
      break;
    }
  }
}

void loop(){
  maintainEth();
}
```