KOTIAUTOMAATIOJÄRJESTELMIEN INTEGRAATIO

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikan opinnäytetyö

Tieto- ja viestintätekniikka, Riihimäki

kevät 2021

Laura Lähteenmäki

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikka, insinööri Tiivistelmä

Riihimäki

Tekijä	Laura Lähteenmäki	Vuosi 2021
Työn nimi	Kotiautomaatiojärjestelmien integraatio	
Ohjaaja	Toni Laitinen	

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutustua integraatioon ja luoda käyttöliittymä, jolla voidaan tarkastella ja muuttaa kahden eri kotiautomaatiojärjestelmän tietoja. Työn oli tilannut yksityishenkilö ja se tehtiin hänen vaatimusmäärittelyidensä pohjalta. Kaikki fyysiset asennukset kotiautomaation toimintaa varten oli toteutettu ja käyttöliittymällä on tarkoitus vain hallita siihen liitettyjä järjestelmiä.

Teoriaosuudessa perehdyttiin taloautomaation perusteisiin ja hyötyihin, ja tiedonvälitystekniikoihin ja integraatioiden toteutustapoihin. Käytettävät teknologiat integraation toteutuksessa olivat SOAP-pohjainen www-sovelluspalvelurajapinta ja Modbus TCP/IP-tiedonsiirtoprotokolla. Yhdistettävät järjestelmät ovat Elko Living Systemin sähkönohjausjärjestelmä ja EH-NET-palvelin, johon on liitetty lämmityksen ja ilmavaihdon Ouman-säätimet.

Lopputulokseksi saatiin rakennettua Windows-sovellus, jonka avulla kotiautomaatiota on jatkossa helpompi hallita yhdestä käyttöliittymästä. Järjestelmien välille luotiin toimiva tiedonsiirtoyhteys ja näin vähennettiin tarvetta käyttää erikseen molempien järjestelmien omia sovelluksia.

Avainsanat integraatio, rakennusautomaatio, rajapinnat (tietokoneohjelmat)
Sivut 27 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Information and Communication Technologies
Riihimäki

Abstract

Author Laura Lähteenmäki
Year 2021
Subject Integration for two home automation systems
Supervisor Toni Laitinen

ABSTRACT

The goal of this project was to get familiar with the integration of applications and to create a user interface that could be used to monitor and change the data of two different home automation systems in a specific household. The project was commissioned by an individual person and it was conducted based on their requirement specifications. All the physical installations for the home automation had been completed long before this project and the user interface was made to control the systems that are used for the home automation.

The theoretical part of the thesis included getting to know the basics and benefits of home automation, the technologies and methods for exchanging data as well as the implementation methods of integration. The technologies used in the actual integration were a SOAP-based Web-service and the Modbus TCP/IP protocol. The systems connected together were Elko Living System (home automation for electrical devices) and an EH-NET server, to which the heating and ventilation controls by Ouman are connected.

The result was a Windows application that makes it easier to manage the home automation from a single interface. A working data connection was established between the systems which reduced the need to use two separate applications for the systems.

Keywords Home automation, integration, interface
Pages 27 pages

Sisällys

Kuvat, taulukot ja kaavat

1 Johdanto

Tämän opinnäytetyön tavoitteena on tutustua integraatioiden toteuttamiseen ja luoda sovellus, joka vaihtaa tietoa kahden kotiautomaatiojärjestelmän kanssa, ja jonka avulla voidaan tarkastella ja muuttaa siihen liitettyjen järjestelmien tietoja. Keskeinen tutkimuskysymys opinnäytetyössä oli, kuinka tiedonsiirto järjestelmien ja käyttöliittymän välillä tapahtuu.

Työn aihe valikoitui pienen sattuman kautta ja kiinnostuksesta kotiautomaatioon. Perhepiirissäni oli tarve toteutettavalle käyttöliittymälle ja tulevaisuudessa on hyvin todennäköistä rakentaa kotiautomaatiota myös omaan kotiin, joten kehitystyön ottaminen oli jopa itsestäänselvyys. Aihevalinnan puolesta puhui myös sen nykyaikaisuus ja konkreettinen tarve lopputuotteelle.

Kehittämistyönä toteutettiin käyttöliittymä Windows-sovelluksena, jolla voidaan hallita kahta eri valmistajan taloautomaatiojärjestelmää. Lopputuloksen esittäminen on rajattu koskemaan tiedonsiirtoa ja lopullista käyttöä, ja graafinen toteutus on jätetty pois. Käyttöliittymä tehtiin omakotitalokiinteistöön, johon kaikki asennukset oli tehty ja automaation taso hyvä, mutta järjestelmät eivät vaihtaneet keskenään tietoa ja niiden hallintaa varten tarvittiin erilliset.

Työn teoriaosuudessa selvitetään taloautomaation perusteet ja hyödyt, ja keskitytään järjestelmien väliseen tiedonsiirtoon ja sen mahdollistaviin tekniikoihin. Läpikäytävät aiheet ovat integraatio, rajapinta, www-sovelluspalvelu ja Modbus-protokolla. Itse käyttöliittymä toteutettiin Visual Studio 2019-ohjelmistolla, .NET Framework -sovellus, ja tiedonsiirrossa hyödynnetään www-sovelluspalvelua ja Modbus TCP:tä.

2 Taloautomaatio

2.1 Älykoti ja taloautomaatio

Älytalolle on esitetty monia erilaisia määritelmiä, mutta yleisesti sen voidaan nykyisin sanoa tarkoittavan rakennusta, joka toimii jonkinlaisessa vuorovaikutuksessa käyttäjän kanssa (Tolppi, 2018, ss. 7–8). Tieteen termipankin (Tieteen termipankki, 2018) mukaan älykodilla tarkoitetaan taloa tai asuntoa, jossa on etäkäytön mahdollistava tietoverkko, joka yhdistää antureita,

kodinkoneita ja laitteita asukkaiden tarpeita varten. Älytalossa voidaan siis sanoa olevan älykästä ja automatisoitua teknologiaa, jota käyttäjä voi mahdollisesti ohjata, ja jolla pyritään saavuttamaan optimaalinen määrä mukavuutta ja energiatehokkuutta (Tolppi, 2018, ss. 7–8).

Taloautomaatioon voidaan kytkeä useita järjestelmiä ja laitteita, jotka kommunikoivat keskenään, ja joiden ohjaus suoritetaan keskitetysti (Motiva Oy, 2019a). Automaation piiriin voi kuulua kodin valaistuksen, ilmanvaihdon ja lämmityksen tarkkailu ja ohjaus, kosteusantureita ja kodin turvallisuustekniikka, kuten kamera- ja kulunvalvonta ja palovaroittimet (Palo, 2016, s. 16). Myös veden- ja sähkönkulutuksen seuranta ovat yleisiä taloautomaatioon lukeutuvia ominaisuuksia (Motiva Oy, 2019b).

Taloautomaatiota voidaan hyödyntää niin suurissa kiinteistöissä, kuin yksittäisissä huoneistoissakin (Motiva Oy, 2019a). Automaation toteutustapa voi olla langallinen tai langaton, ja sitä voidaan hyödyntää vain osassa kiinteistön järjestelmiä, tai kaikessa mahdollisessa (Motiva Oy, 2019b). Monet automaatioratkaisut on mahdollista toteuttaa vanhoihinkin kiinteistöihin, kunhan tarvittava tiedonsiirtoratkaisu on käytössä (Motiva Oy, 2019a). Automaatio on mahdollista toteuttaa myös osissa, esimerkiksi remonttien yhteydessä (Motiva Oy, 2019b).

2.2 Taloautomaation hyödyt

Hyvin toteutettu, kattava ja älykäs taloautomaatio mahdollistaa talotekniikan osien ja kokonaisuuden tehokkaan ja helpon seurannan, valvonnan, ohjauksen ja optimoinnin (Motiva Oy, 2019b). Sen hyötyinä ovat asumismukavuuden ja kiinteistön energiatehokkuuden maksimoiminen, turvallisuuden parantaminen ja arjen yleinen helpottaminen (Motiva Oy, 2019a). Etenkin säästö asumiskustannuksissa on yksi automaatiojärjestelmän suorista hyödyistä (Motiva Oy, 2019b).

Automaatiojärjestelmällä voidaan siis säästää energiaa, rahaa ja konkreettisesti myös aikaa, ja lisätä asumisviihtyvyyttä (Motiva Oy, 2019a). Taloautomaatiolla saavutettavia hyötyjä ovat muun muassa sisäilmaston ylläpito, valaistuksen säätö ja ohjaus, energiansäästö ja etäohjaus ja -valvonta (Motiva Oy, 2019b).

Automaation avulla esimerkiksi lämmitysmuoto ja -teho on mahdollista toteuttaa kotonaoloon pohjautuen, ja lisäksi jopa sääolosuhteiden ja pörssisähkön hinnanvaihtelun perusteella.

Vedenkulutusta voidaan seurata ja käyttöveden lämmitystä säätää paikalla olevien henkilöiden tarpeisiin sopivaksi. Automaatiolla on myös mahdollista hidastaa ilmanvaihtoa ja sulkea pistorasiat ja valaistus kotoa poissaolon ajaksi, ja toimintojen palauttaminen normaaleiksi ennen kotiinpaluuta onnistuu etäohjatulla taloautomaatiolla. (Motiva Oy, 2019b)

Kodin turvallisuus parantuu valvontajärjestelmän avulla, mikäli järjestelmä mahdollistaa kiinteistön seurannan myös etänä. Tällöin sen kautta voi saada erilaisia hälytyksiä laitteiden mittauksista, ja esimerkiksi vesivahinko on mahdollista havaita heti sen tapahduttua. Taloautomaatio helpottaa kodin ylläpitoa ja mahdollisesti vaikuttaa positiivisesti kodin jälleenmyyntiarvoon. (Motiva Oy, 2019b)

3 Tiedonsiirto järjestelmien välillä

3.1 Tiedonsiirto

Tietojen siirto järjestelmien välillä tulee kyseeseen, kun toisesta järjestelmästä halutaan siirtää tietoa toisen järjestelmän käytettäväksi. Siirrettävänä voi olla yksi tietue tai suurikin joukko tietoa. Tapoja suoritukseen on useita.

Yksinkertaisimmillaan tietoja voidaan kopioida järjestelmästä toiseen tietue kerrallaan käsin, tai siirtää niitä lataamalla tiedot manuaalisesti järjestelmästä esimerkiksi Excel-tiedostoon, ja ajamalla ne sitten uuteen järjestelmään. Järjestelmät eivät välttämättä pysty suoraan käsittelemään toistensa tietoja, jolloin dataa voidaan joutua etsimisen ja syöttämisen lisäksi muuntamaan käsityönä. (LogiNets, n.d.)

Automatisoitu tiedonsiirto on usein manuaalista työtä tehokkaampi ja nopeampi tapa siirtää tietoja järjestelmien välillä. Manuaalinen tiedonsiirto syö tehokkuutta ja lisää virheiden mahdollisuutta; esimerkiksi näppäilyvirheitä, tai tiedon ajautumista väärään paikkaan. Automatisoituja tiedonsiirtotapoja on esimerkiksi integraatio ja robottien opettaminen kopioimaan tietoja järjestelmästä toiseen. (LogiNets, n.d.)

3.2 Järjestelmäintegraatio

Kun eri tekniikoilla tai alustoilla toteutetut järjestelmät halutaan saada vaihtamaan tietoja, eli niin sanotusti keskustelemaan keskenään, puhutaan järjestelmäintegraatiosta (Haglund, 2018a). Alfamen (Alfame Oy, n.d.) mukaan integraatio voi tulla aiheelliseksi, kun:

- tiedon määrä kasvaa, mutta se sijaitsee useammassa paikassa
- tarvitaan ominaisuuksia, joita nykyinen järjestelmä ei tarjoa
- otetaan käyttöön uusia järjestelmiä
- halutaan siirtää tietoa yhdestä järjestelmästä toisen hyödynnettäväksi.

Integraatio voi olla vain esimerkiksi kahden eri komponentin välinen, yksinkertainen tiedonsiirto, tai hyvin laaja ja monimutkainen yhdistelmä useita eri järjestelmiä, joista tiedonmuruset kerätään helposti saataville (Alfame Oy, n.d.). Yhteen liitettäviä järjestelmiä voisi olla esimerkiksi verkkokauppa, varastonhallintajärjestelmä, laskutusjärjestelmä ja kirjanpitojärjestelmä.

Haglundin (Haglund, 2018a) mukaan järjestelmäintegraatiota ei tulisi nähdä pelkkänä tiedonsiirtona, vaan kaikkine hyötyineen. Hyvin toteutettu integraatio järjestelmien välillä tehostaa toimintaa, koska:

- manuaalinen työ jää välistä
- voidaan välttyä saman tiedon syöttämiseltä ja sen päivittämiseltä useampiin eri järjestelmiin
- tietoja ei tarvitse etsiä eri järjestelmistä
- tietojen muuttumista ja eri järjestelmien toimintaa ei tarvitse jatkuvasti tarkkailla

Integraatio siis vähentää virheiden määrää ja säästää aikaa ja kustannuksia, ja sen avulla voidaan automatisoida prosesseja ja hyödyntää helpommin kaikkea saatavilla olevaa tietoa. (Haglund, 2018a)

Integraatio voidaan toteuttaa monella eri tavalla, mihin vaikuttavat yhdistettävien järjestelmien rajapinnat, tietovarastot ja alustat. Integraatio voi olla yksisuuntainen, jolloin toinen järjestelmä lähettää ja toinen vastaanottaa tietoja, tai kaksisuuntainen, jolloin molemmat osapuolet voivat

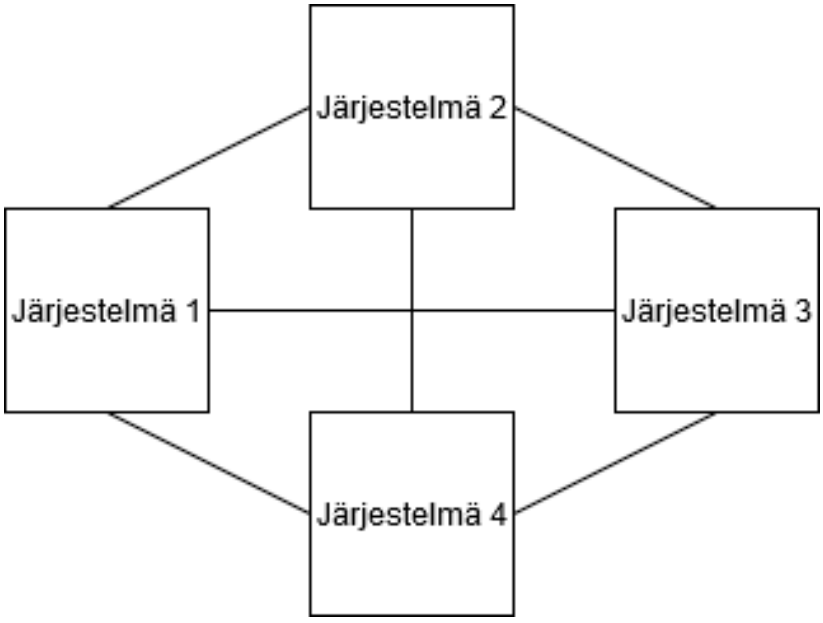
sekä lähettää, että vastaanottaa tietoja. (LogiNets, n.d.) Integraation suunnitteluvaiheessa ja toteutustapoja miettiessä tulee miettiä esimerkiksi mitä tietoja, missä muodossa ja kuinka usein niitä halutaan siirtää (Toivanen, n.d.).

3.3 Integraation toteutustavat

Integraation toteutustavoilla tarkoitetaan tässä työssä arkkitehtuurimalleja, joilla integraatioita voidaan toteuttaa. Tässä luvussa esitellään kolme yleistä mallia.

Yksinkertaisin toteutustapa integraatiolle on kuvassa 1 esitetty Point-to-Point-integraatio, eli niin sanottu suora integraatio. Kuvan viivat esittävät luotua integraatiota. Malli toimii yhdestä yhteen periaatteella, eli data kulkee nimensä mukaisesti pisteestä pisteeseen, eikä integraatioalustaa tarvita. Integraatio tulee luoda erikseen jokaisen eri järjestelmän välille, jos ne halutaan vaihtamaan tietoa keskenään. (Toivanen, n.d.)

Kuva 1. Point-to-point-integraatio



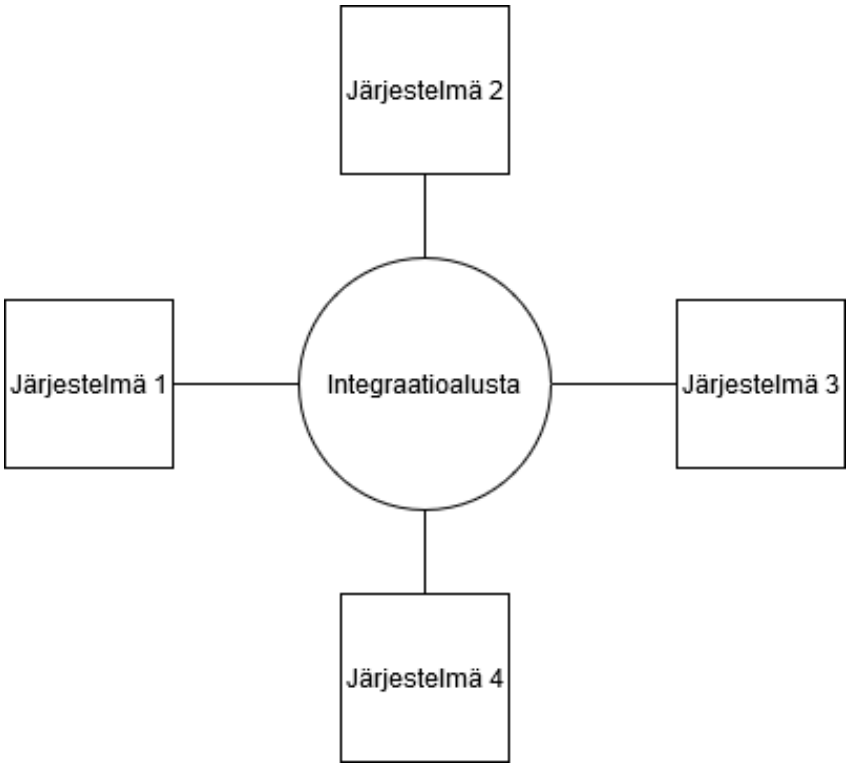
Väliaikaisissa integraatioissa ja pienissä projekteissa pisteestä pisteeseen integraatio voi olla juuri oikea valinta. Jos kyseessä on monen eri järjestelmän yhteen liittäminen, on Point-to-Point-integraatio yleensä huono tapa sen toteuttamiselle, sillä lopputulos on usein sekava, monimutkainen ja vaikeasti hallittavissa ja valvottavissa. Vältettävissä olevaa työtä tulee

esimerkiksi, mikäli jossain vaiheessa jokin integraatioon liitetyistä järjestelmistä tullaan vaihtamaan. Tällöin kaikki järjestelmään tehdyt integraatiot on luotava uudelleen jokaiseen mukana olevaan järjestelmään. (Toivanen, n.d.)

Integraatioalusta on nykyaikainen tapa toteuttaa integraatioita. Integraatioalustoja on monia ja niiden toiminnallisuudet saattavat erota toisistaan. Alusta ei ole lopullinen integraatioratkaisu, vaan sen tarkoituksena on toimia tietovirtojen kuljettajana ja valvojana. Alustat voivat toimia mahdollistajina ja toteutusalustoina sekä sisäisten, että julkisten rajapintojen kanssa. (Toivanen, n.d.)

Alustaa käytettäessä voidaan puhua keskitetystä integraatiosta, joka on havainnollistettu kuvassa 2. Integraatioalusta siirtää tietoa järjestelmien välillä, ja sen avulla voidaan toteuttaa eri rakenteisten sanomien muuttaminen toiseen muotoon ja protokollamuunnokset, milloin eri tiedonsiirtoprotokollia käyttävien järjestelmien kommunikointi saadaan toimimaan. Keskitetty integraatio toimii yhdestä moneen periaatteella ja sen hyötyjä ovat datan muuntamisen lisäksi valvonnan ja ylläpidon helppous. (Toivanen, n.d.)

Kuva 2. Keskitetty integraatio integraatioalustaa hyödyntäen



Integraatioalustan etuna on, että siihen liitettyjen järjestelmien ei tarvitse olla toisistaan tietoisia, sillä integraatio tehdään vain alustaan. Tällä tavalla mikä tahansa järjestelmistä voidaan vaihtaa ja uusi integraatio täytyy toteuttaa vain uuden järjestelmän ja alustan välille.

Jotkin nykyisistä palveluna toimitettavista integraatioalustoista (iPaaS = Integration Platfrom as a Service) osaavat tietovirtojen valvonnan ja muuntamisen lisäksi prosessien automatisoinnin; prosessien tekninen kulku liitetään integraatioalustalle ja järjestelmät kytkeytyvät toisiinsa yhden suuremman prosessin osina. Automaation liittäminen integraatioalustaan tehostaa toimintaa, sillä alusta kykenee hoitamaan prosessit nopeasti toimimalla suoraan rajapintatasolla ja välittämättä käyttäjän toimista. (Toivanen, n.d.)

Kolmas malli toteuttaa integraatio on hyvin samantapainen keskitetyn integraation kanssa. ESB-integraatio on alustapohjainen integraatio, joka hyödyntää palveluväylää. Palveluväyläpohjaista integraatiosta kutsutaan ESB-integraatioksi, joka tulee lyhenteenä sanoista Enterprise Service Bus. (Toivanen, n.d.)

ESB-integraatio pystyy esimerkiksi publish/subscribe-viestinvälitykseen, jossa tietoa haluavat järjestelmät ilmoitetaan kuuntelijoiksi eri aiheille ja palveluväylä reitittää niille oikeat sanomat. Yleensä väylässä on myös käytössä niin kutsuttu jonojärjestelmä, johon sanoma tallennetaan ennen kuin se saavuttaa kohdemääränpäänsä. Nykyään kuitenkin suositaan enemmän tapahtumapohjaista tiedonvälitystä, jossa sanomat lähetetään jonon sijasta suoraan kohdejärjestelmän rajapintaan. (Toivanen, n.d.)

3.4 Ohjelmointirajapinta

Ohjelmointirajapinta (Application Programming Interface, API) on tekniikka, joka mahdollistaa kahden eri ohjelman välisen datan vaihdon, eli sen avulla voidaan tietyllä tavalla liittää sovelluksia yhteen. Rajapinnat ovat siis mahdollisuus jakaa tietoa, ja ne ovat laajasti käytössä monilla elämän ja liiketoiminnan osa-alueilla. Esimerkiksi monet matkapuhelinsovellukset käyttävät rajapintoja hyödykseen. (Haglund, 2018b)

Käytännönläheinen esimerkki rajapinnan hyödyntämisestä on lomamatkojen vertailusivusto, jolla näytetään matkoja eri tarjoajien valikoimasta. Vertailusivusto hakee siis matkat näytille niitä tarjoavien yritysten rajapintojen kautta.

API määrittää miten tietoja tai palveluita tarjotaan ohjelmistosta muille, ja sitä voisikin kuvailla ovena itse järjestelmään. Yksi järjestelmä voi sisältää erilaisia rajapintoja; Datarajapinnan kautta dataa voidaan siirtää järjestelmästä toiseen ja toiminnallinen rajapinta voi tarjota mahdollisuuksia muuttaa järjestelmän tietoja. Samassa järjestelmässä voi myös olla sisäisiä, julkisia ja rajoitettuja rajapintoja. (Poikola ym., 2014)

Kun puhutaan sisäisestä rajapinnasta, tai yksityisestä rajapinnasta, tarkoitetaan vain organisaation käytössä olevaa APIa, kun taas ulkoiset, tai julkiset, rajapinnat on julkaistu kaikkien käytettäväksi. Rajapinta voidaan myös julkaista vain yhteistyökumppaneille tai muuten rajoitetulle joukolle. (Toivanen, n.d.) Avoin rajapinta eli OpenAPI on julkisesti tarjolla oleva ja hyvin dokumentoitu rajapinta (Poikola ym., 2014). Esimerkiksi Google tarjoaa karttapalveluunsa julkisen rajapinnan, jota yritykset voivat hyödyntää omilla verkkosivuillaan.

Selkeä hyöty rajapintojen käyttämisestä on tilanteessa, kun tulee aiheelliseksi vaihtaa yksi integraatioon liitetyistä järjestelmistä. Esimerkiksi toiminnanohjausjärjestelmä voidaan vaihtaa, eikä sitä kutsuva järjestelmä huomaa vaihtoa, sillä kutsut lähetetään rajapintaan, eikä itse järjestelmään. (Toivanen, n.d.)

3.4.1 Avoin rajapinta

Kun puhutaan avoimesta rajapinnasta, tarkoitetaan rajapintaa, johon käytännössä kuka tahansa voi yhdistää minkä tahansa sovelluksen ilmaiseksi, ja sen kaikki ominaisuudet ovat julkisia ja avoimesti saatavilla. Avoimien rajapintojen olemassaolon takia on mahdollista kehittää lukuisia uusia järjestelmiä, jotka ovat yhteensopivia kaikkien rajapintaa käyttävien ohjelmistojen kanssa. (Poikola ym., 2014)

Tulee huomioida, että vaikka järjestelmän rajapinta on avoin, ei itse järjestelmän käyttäminen silti välttämättä ole maksutonta. Eli esimerkiksi jokin laskutusjärjestelmä voi tarjota avoimen

rajapinnan, jonka kautta järjestelmä voidaan kytkeä olemassa olevaan tilausjärjestelmään, mutta itse laskutusjärjestelmän käyttämisestä voidaan periä maksua.

Avoimen rajapinnan määrittelyyn liittyy avoin dokumentaatio, käyttöönotettavuus ja testattavuus. Rajapinnan dokumentoinnin tulee olla kattava ja vapaasti saatavilla verkon kautta. Dokumentaation tulee olla tehty niin riittävällä tasolla, että avoin rajapinta voidaan ottaa käyttöön itsenäisesti, ilman toimittajalta kysyttäviä lisätietoja. Käyttöönoton tulee voida tapahtua myös ilman järjestelmäntoimittajan erikoistoimia, ja rajapinnan testausta varten tulee olla tarjolla testiaineisto. (Poikola ym., 2014)

3.4.2 REST ja SOAP

Yksi yleinen arkkitehtuurimalli rajapintojen toteuttamiseen on REST, Representational State Transfer. REST on tutkija Roy Fieldingin vuonna 2000 luoma malli, joka määrittelee joukon ehtoja, jotka rajapinnan on toteutettava. Datan välittämiseen rajapintamallin kanssa voidaan käyttää useita tietoformaatteja, sillä REST ei määritä niiden standardia. (Mikkonen, 2017)

Kun puhutaan RESTistä, tulee huomioida sen voivan tarkoittaa arkkitehtuurimallin lisäksi RESTful Web-serviceä, eli REST-arkkitehtuuria toteuttavaa www-sovelluspalvelua. Tämä käsitellään työssä myöhemmin.

RESTiä pidetään yksinkertaisena ja luotettavana mallina rajapintojen toteuttamiseen, mistä syystä siitä tuli suosittu jo julkaisunsa alkuvuosina. Rajapintamalli on hyvin tulkinnan vapaa ja tarjoaa rajapinnan suunnittelijalle paljon vapauksia. Rajapintaratkaisun heikkouksina taas pidetään sen kykyä selviytyä äärimmäisen suuresta ja monimutkaisesta dataskaalasta. (Mikkonen, 2017)

SOAP, eli Simple Object Acces Protocol, on avoimen standardin XML-perustainen viestintäprotokolla. Protokollalla voidaan välittää tietoa internetin yli ja sitä hyödynnetään www-sovelluspalveluissa. SOAPin etuna on, että sitä voidaan käyttää viestin välitykseen eri käyttöjärjestelmillä toimivien ja eri teknologioilla toteutettujen sovellusten välillä, koska SOAP-viestit kirjoitetaan täysin XML:llä. (Tutorialspoint, n.d.-a)

REST-pohjainen rajapinta eroaa edeltäjästään SOAP-rajapintaa toteuttavasta tiedonvälitystavasta pohjaamalla vahvasti HTTP-protokollan ominaisuuksiin. Tässä tavassa pyynnön luonnetta kuvataan käyttämällä HTTP:n eri metodeja; get, post ja delete, ja pyyntöjä lähetetään eri uri-osoitteisiin. Näin toimimalla dataan ei tarvitse sisällyttää toivotusta pyynnöstä metatietoja ja tieto pysyy helppolukuisempana. SOAP-protokollaa toteuttava rajapinta lähettää kaikki pyynnöt samaan osoitteeseen post-metodilla ja metatieto sisällytetään lähetettävään dataan. (Mikkonen, 2017)

Jos vertaillaan SOAPia ja RESTiä tulee huomioida, että REST on rajapinnan toteutusmalli ja SOAP viestintäprotkolla. SOAP ei voi hyödyntää RESTiä, mutta REST voi hyödyntää SOAPia.

SOAP toimii vain XML-formaattien kanssa, mutta REST-rajanpinnan kanssa voidaan käyttää useaa tietoformaattia. Yksi yleisimmistä on JSON, JavaScript Object Notation, joka on avoimen standardin tiedostomuoto, joka esittää datan tekstinä. JSON on yksinkertainen, kevyt ja nimestään huolimatta riippumaton JavaScriptistä. JSON mahdollistaa kuitenkin JS-objektien tallentamisen tekstimuotoon, ja siirtämisen internetin välityksellä. (W3Schools, n.d.)

3.5 Www-sovelluspalvelu

Web service, eli www-sovelluspalvelu, on ohjelma, jolla tarkoitetaan internetin yli tarjottavaa palvelua sovelluksille. On huomioitava, että englannin kielistä web service-termiä käytetään kuvaamaan myös verkkopalvelua (internet-palvelu), joka tarkoittaa palvelua, joka tarjotaan selaimen kautta loppukäyttäjille, esimerkiksi sähköpostipalvelu. (Sanastokeskus TSK ry, n.d.) Tässä työssä web servicellä viitataan verkon yli tarjottavaan sovelluspalveluun ja siitä käytetään termipankin suosittelemaa termiä: www-sovelluspalvelu.

Www-sovelluspalvelulle ei ole tarkkaa määritelmää. W3C (W3C) kuvailee ohjelmaa niin, että sen voidaan sanoa olevan mikä tahansa verkon välityksellä koneiden, laitteiden ja järjestelmien välisen kommunikaation mahdollistava ohjelmisto. Oraclen mukaan (Oracle, 2013a) www-sovelluspalvelu on client/server-malliin perustuva sovellus, joka toimii internetin yli HTTP-protokollaa hyödyntäen. Tutorials Point taas kuvailee (Tutorialspoint, n.d.-b) www-sovelluspalveluiden olevan mitä tahansa avoimiin standardeihin perustuvia internet-palveluita, jotka kommunikoivat toisten palveluiden kanssa saadakseen aikaiseksi tiedonvaihdon näiden järjestelmien välille.

Käytännössä www-sovelluspalvelu on siis www-pohjainen ohjelmointirajapinta. Ero API-rajapintaan on lähinnä siinä, että www-sovelluspalveluun täytyy aina liittyä internet-yhteyden avulla.

Ominaista www-sovelluspalvelulle on alustariippumattomuus ja laajennettavuus. Ohjelmalla voidaan tarjota eri ohjelmistoalustoilla ja -kehyksissä toimivien sovellusten välille tavanomainen kommunikointitapa. Palvelu voi olla hyvin yksinkertainen, vaikka vain yhden resurssin tarjoaminen asiakkaalle, tai suuri ja monimutkainen kokonaisuus. Palveluita voidaan myös yhdistää toisistaan riippumattomina (=löyhä kytkentä) ja saada aikaiseksi lisäarvoa tuottavia toimintoja. (Oracle, 2013a)

Sovelluspalvelun arkkitehtuurissa on käytännössä kolme olennaista roolia: palvelun tarjoaja (provider), palvelun käyttäjä (requester) ja palvelun välittäjä (broker). Palvelun tarjoaja määrittää palvelun ja julkaisee sen välittäjälle, jonka avulla palvelun käyttäjä löytää sen ja luo yhteyden palvelun tarjoajaan. (IBM, 2000)

Muun muassa Tutorials Point (Tutorialspoint, n.d.-b) määrittelee www-sovelluspalvelun perusstandardeiksi XML:n, SOAPin ja WSDL:n. Nämä teknologiat eivät kuitenkaan ole ainoita, joilla web-palveluita voidaan toteuttaa. Myös REST-pohjaiset www-sovelluspalvelut ovat yleisiä.

XML, Extensible Markup Language, on World Wide Web Consortiumin kehittämä rakenteellinen kuvauskieli, joka määrittää tiedon merkintämuodon. XML on avoimen standardin kieli ja sitä pidetään merkintäkielien standardina. XML:ää voidaan käyttää sekä tiedostomuotona että tiedonvälitysformaattina järjestelmien välillä. Kieli on kehitetty niin, että sekä kone että ihminen ymmärtävät sitä. (Tutorialspoint, n.d.-d)

WSDL, Web service description language, on Ariban, Microsoftin ja IBM:n yhteistyössä kehittämä XML-pohjainen kuvauskieli, jota käytetään standardina kuvaamaan www-sovelluspalveluita. WSDL määrittelee mitä toimintoja palvelu antaa ja miten siihen päästään käsiksi. Asiakasohjelma voi lukea WSDL-tiedoston. (Tutorialspoint, n.d.-c)

Lyhykäisyydessään Tutorials Point (Tutorialspoint, n.d.-b) kuvaa www-sovelluspalvelun toimintaa niin, että WSDL kertoo mitä palveluja on saatavilla ja miten, XML:ää käytetään määrittämään ja

merkitsemään data ja SOAPia sen siirtämiseen. Asiakasohjelma ottaa yhteyttä palveluun, lukee WSDL-tiedoston ja kutsuu haluamaansa tiedostossa listattua funktiota SOAPin avulla.

Toinen yleinen www-sovelluspalvelun toteutusmalli on nimeltään RESTful Web Service, eli REST-pohjainen www-sovelluspalvelu. Nimensä mukaisesti kyseessä on siis www-sovelluspalvelu, joka toteuttaa REST-arkkitehtuuria. REST-tyyppiset sovelluspalvelut eivät vaadi erityisesti XML-viestejä tai WSDL-dokumenttia, vaan niiden kanssa voidaan hyödyntää eri teknologioita. REST-palvelut toimivat myös yleensä SOAP-pohjaisia www-sovelluspalveluita paremmin juuri HTTP:n kanssa. REST-pohjaisen www-sovelluspalvelun kehittäminen on yleensä edullista ja sen omaksuminen ja käyttöönotto on helppoa, koska malli on kevytrakenteinen ja hyödyntää yleisiä ja tunnettuja teknologioita, kuten HTTP ja URI. (Oracle, 2013b)

3.6 Modbus

Modbus-protokolla on sovelluskerroksen viestintäprotokolla, jonka avulla erilaisiin väyliin ja verkkoihin yhdistetyt laitteet voivat kommunikoida keskenään. Modbus mahdollistaa esimerkiksi master-laitteena toimivan tietokoneen ja monen slave-laitteen yhteen liittämisen joko sarjaliitäntänä (Modbus Serial) tai Ethernet-liitäntänä (Modbus TCP). (WAGO Finland Oy, n.d.)

Protokolla on tekniikasta ja valmistajasta riippumaton, yksinkertainen client/server- (tai slave/master-) malliin perustuva tiedonsiirtoprotokolla, jota käytetään helppoon, nopeaan ja luotettavaan tiedonsiirtoon älykkäiden elektroniikkalaitteiden välillä, ja monitoroimaan ja hallitsemaan erilaisia laitteita. (Modbus Organization, n.d.)

Modbus-protokolla julkaistiin vuonna 1979 ja se on muodostunut de facto-, eli yleisesti käytössä olevaksi, standardiksi muun muassa teollisessa ympäristössä. Protokollan alkuperäinen julkaisija oli Modicon, mutta nykyään protokollan kehittämisestä vastaa Modbus Organisaatio, joka on avoin ja voittoa tavoittelematon yhteisö. Yksi merkkipaalu protokollalle oli vuosi 1999, jona se niin sanotusti tuotiin 2000-luvulle, kun avoin Modbus TCP/IP spesifikaatio kehitettiin. (Modbus Organization, n.d.) Tässä työssä keskitytään Modbus TCP:hen.

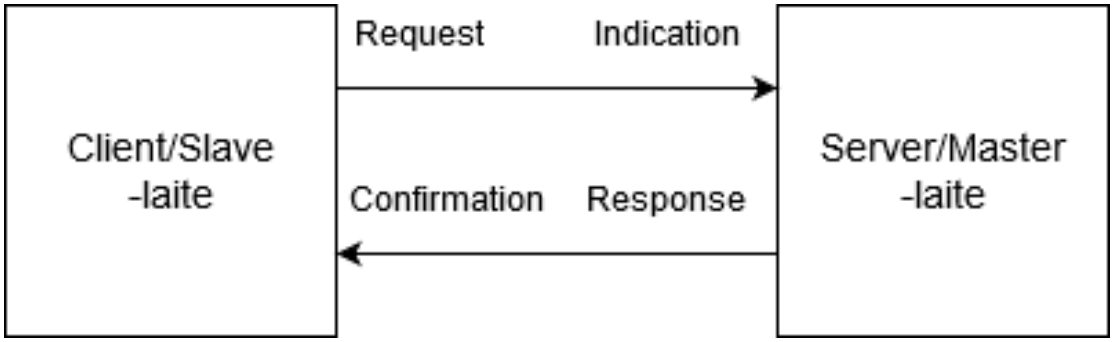
Modbus TCP/IP on tiedonsiirtoprotokolla, josta on tullut suosittu sen avoimuuden, yksinkertaisuuden ja edullisuuden takia. Protokolla toimii client/server-mallilla ja sen ainoa

vaatimus on, että kytketyt laitteet ovat samassa IP-osoiteavaruudessa keskenään (WAGO Finland Oy, n.d.). Protokolla mahdollistaa laitteiden etäkäytön myös internetin yli ja se toimii TCP/IP-pinossa portissa 502 (Modbus Organization, n.d.).

Kuvassa 3 on esitetty Modbus TCP/IP:n viestintämalli, joka perustuu neljään viestityyppiin, jotka Modbus organisaation tuottama opas esittelee (Modbus Organization, 2006, s. 2):

- Request: viesti, jonka client lähettää verkkoon käynnistääkseen vuorovaikutuksen
- Indication: server-puolen saama request viesti
- Response: server-laitteen lähettämä viesti
- Confirmation: client puolen saama response-viesti

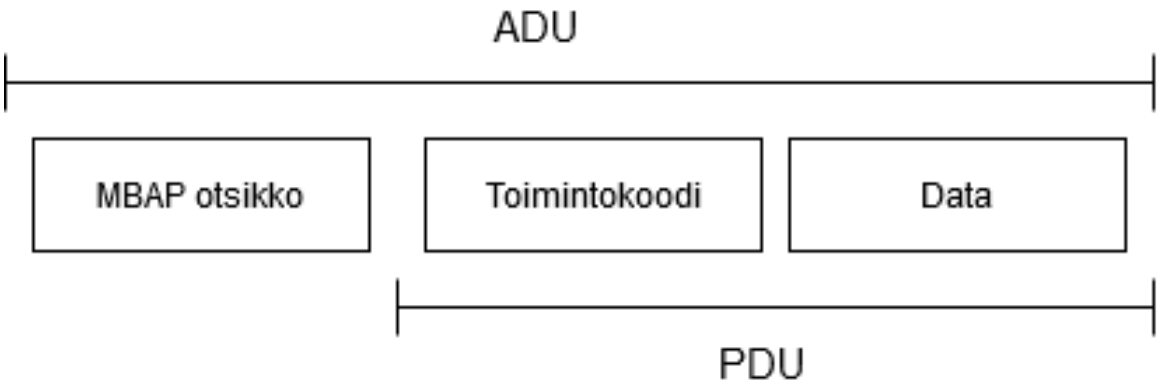
Kuva 3. Modbus TCP/IP-protokollan viestityypit



Keskeisiä lyhenteitä ja käsitteitä Modbus-protokollan ymmärtämisessä ovat PDU ja ADU. PDU, eli Protocol Data Unit; vapaasti suomennettuna protokollan tietoyksikkö, on lähetettävä viesti, joka sisältää tietokenttiä yhteyskäytännöstä ja dataa. ADU on lyhenne sanoista Application Data Unit, eli sovelluksen tietoyksikkö. ADU (myös ADPU, eli Application Protocol Data Unit) tarkoittaa sovelluskerroksen datayksikköä/-pakettia.

Modbus-protokollan määrittelemä PDU on kaksiosainen: Toimintokoodi + data. Toimintokoodi määrittää minkälainen toimenpide suoritetaan ja data sisältää siihen liittyvän tiedon, eli viestin. Client, joka käynnistää tiedonsiirron, rakentaa ADU:n. (Modbus Organization, 2006, s. 4) TCP/IP-verkossa Modbus TCP:ssä request- ja response-viesteissä ADUn rakenne on kolmiosainen, kuten kuvassa 4 on esitetty. Kaksiosaisen PDU:n lisäksi se sisältää otsikon (MBAP = modbus application protocol header), jota käytetään identifioimaan ADU. (tcip dok s5)

Kuva 4. Modbus TCP/IP-protokollan viestirakenne



Modbus TCP:n ADU:n otsikossa määritellään viestinvaihdon tunniste, protokollan tunniste, joka Modbus-viestinnässä on 0, viestin pituus ja mahdollisesti client-laitteen laiteosoite. Client, joka aloittaa viestinvaihdon, siis määrittää otsikossa olevan tunnisteen ja server-laite kopioi sen. Näin osapuolet pysyvät perillä toisiinsa liittyvistä viesteistä. (Modbus Organization, 2006, s. 5)

Yksinkertaistaen Modbus TCP toimii siis niin, että client lähettää serverille viestin, eli ADU-paketin, jossa se määrittää toimenpiteen ja tarjoaa siihen datan. Server vastaanottaa viestiin ja pyydetty toiminto voidaan toteuttaa.

4 Kehittämistyö

4.1 Alkukartoitus

Käytännön työ tehtiin vuonna 2005 rakennettuun, kaksikerroksiseen omakotitaloon, joka sijaitsee Riihimäellä. Talon yläkerrassa on kaksi makuuhuonetta, olohuone, keittiö, ruokailutila ja kylpyhuone. Olohuoneessa on takka, jota käytetään talvisin lämmitykseen. Alakerrassa on sauna ja kylpyhuone, kodinhoitohuone, vessa, takkahuone ja yksi makuuhuone. Lisäksi alakerrassa on varasto- ja teknisiä tiloja.

Rakennuksen lämmitysmuotona on maalämpö, talon katolla on aurinkopaneelit omavaraista sähköntuotantoa varten, ja kiinteistön ilmanvaihto on koneellinen tulo-/poistoilmanvaihto lämmöntalteenotolla. Kiinteistössä on koko talon kattava lattialämmitys, joka ulottuu myös teknisiin tiloihin ja autotalliin.

Taloon on rakennusaikana mahdollistettu kattavan kotiautomaation hyödyntäminen ja kaikki tarvittavat asennukset on toteutettu jo silloin. Käytössä olevat automaatiojärjestelmät ja niiden hallinta toimivat, mutta eivät kommunikoi keskenään. Ohjauslaitteet ja järjestelmät kiinteistössä ovat:

- Elko Living System: Ohjelmoitava sähkönohjausjärjestelmä
- EH-NET-palvelin: Ethernet-liitännäinen palvelin, joka toimii Modbus-väylän master-laitteena
- Ouman-säätimet lämmitykselle ja ilmanvaihdolle, jotka on kytketty EH-NET-palvelimeen

Kiinteistön sähköt; sisä- ja ulkovalot, osa pistorasioista, kiuas ja muita kodinkoneita on liitetty Elko Living System -järjestelmään. Elkon kotiautomaatiojärjestelmä mahdollistaa erilaisten toimintojen ohjelmoimisen toteutettavaksi oman käyttöliittymänsä kautta. Tähän käyttöliittymään on valmiiksi kirjattu esimerkiksi kaikki ajastukset, joita keskusyksikkö toteuttaa. Myös valojen ja pistokkeiden hallinta on ohjelmoitu logiikkaan valmiiksi.

Lämmitysjärjestelmä, joka kattaa lattialämmityksen ja käyttöveden lämmityksen hallitsemisen, on liitetty EH-203GL-säätimeen, ja ilmanvaihtokone EH-105-säätimeen. Säätimet on yhdistetty kotiverkkoon kytkettyyn EH-NET-palvelimeen Modbus-väylällä. Molempien järjestelmien antamia tilatietoja tarkkaillaan niiden omissa käyttöliittymissä, joiden kautta myös laitteistojen säätäminen onnistuu jotakuinkin riittävästi.

Työn tavoitteena on toteuttaa käyttöliittymä, joka toimii viestin välittäjänä järjestelmien välillä ja käyttäjältä molemmille järjestelmille. Rakennettava sovellus voisi käyttäjän pyynnöstä lähettää Elkon keskusyksikölle käskyn jonkin sinne luodun toimenpiteen toteuttamiseksi. Pidemmälle kehitettynä sovellus voisi hakea ulkolämpötilan Oumanin järjestelmästä, analysoida sen ja lähettää tarvittavat, ennalta määritellyt toimenpiteet Elkon järjestelmään, joka tekee tarvittavat toimet, kuten kytkee autojen lämmityksen päälle, oman järjestelmänsä kautta.

4.2 Määrittely

Työn tilaaja halusi yksinkertaisen ja helppokäyttöisen hallintajärjestelmän, jonka kautta voidaan hallita sekä sähkö- että LVI-järjestelmiä yhdellä sovelluksella. Tarkoituksena oli, että jo olemassa

olevaa tietoa ei tarvitse mitata uudelleen, eikä molempien järjestelmien käyttöliittymiä ole avattava tietojen tarkastelua ja muuttamista varten erikseen.

Itse käyttöliittymään haluttiin näkyviin kellonaika ja päivämäärä, sisä- ja ulkolämpötila ja kotona/poissa-kytkin, joka säätää automaattisesti kiinteistön ja käyttöveden lämmityksen, ja poissa-tilassa sammuttaa mahdollisesti päällä olevat valot.

Muita vaatimuksia, jotka käyttöliittymälle asetettiin olivat:

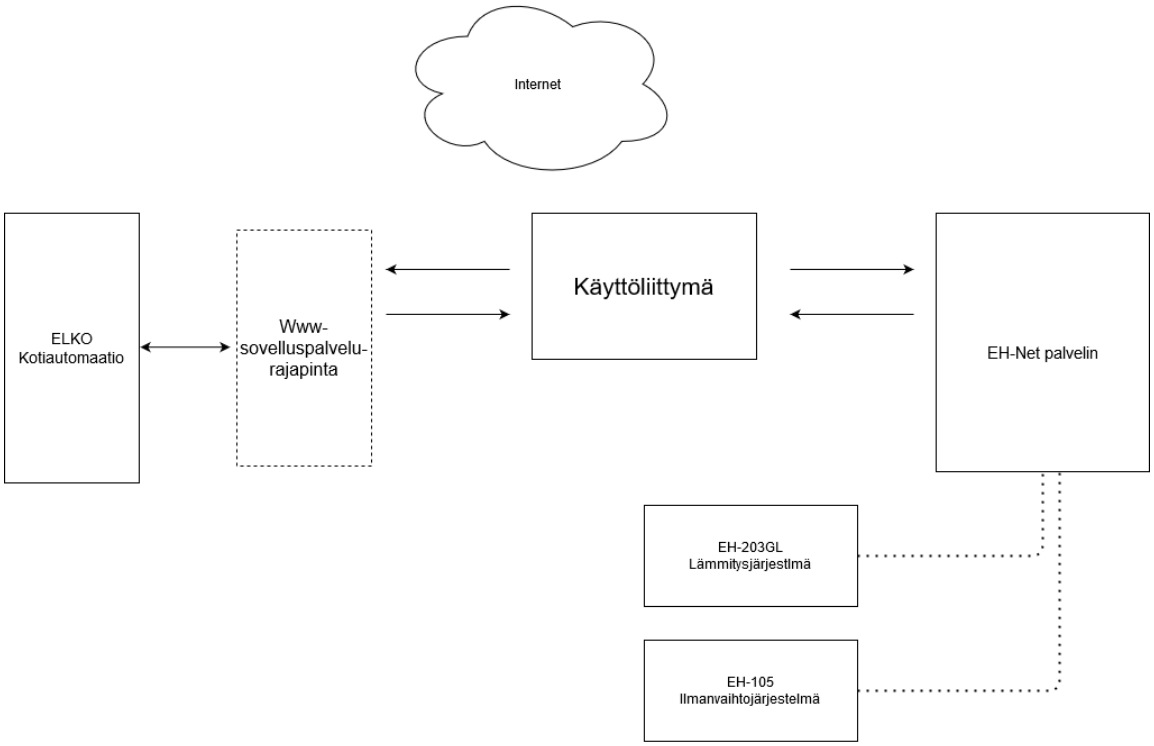
- Autojen lämmityksen päälle ja pois kytkeminen
- Autojen lämmityksen ajastaminen
- Lämmitysjärjestelmän tarkkailu ja säätäminen (huonelämpötila)
- Kiukaan päälle ja pois kytkeminen
- Kompressorin tila (päällä/pois)
- Ulko- ja sisävalojen säätäminen ja ajastaminen
- Käyttöveden lämmityksensäätö
- Historiatietojen tarkastelu

Valojen säätämisessä haluttiin mahdollisuus kytkeä kaikki talon valot päälle ja pois, sekä sama mahdollisuus erikseen ylä- ja alakerralle. Toiveena oli myös saada eräänlainen siivoustila-painike, joka kytkee kaikki talon valot päälle ja palautettaessa normaalitilaan muistaa valojen edelliset asetukset ja palauttaa ne.

4.3 Suunnittelu

Tuotteen vaatimusmäärittelyn jälkeen perehdyttiin laitteistojen tiedonvaihtomahdollisuuksiin. Oumanin järjestelmä toteuttaa työssä aiemmin esiteltyä Modbus-tiedonsiirtoa Ethernet-verkossa ja Elko Living Systemin dataan päästään käsiksi käyttämällä laitteiston tarjoamaa SOAP-protokollaa hyödyntävää www-sovelluspalvelua. Kuvassa 5 on esitetty piirros, joka kuvaa luotavan järjestelmäkokonaisuuden arkkitehtuuria.

Kuva 5. Toteutettavan järjestelmäkokonaisuuden ylätason arkkitehtuuri



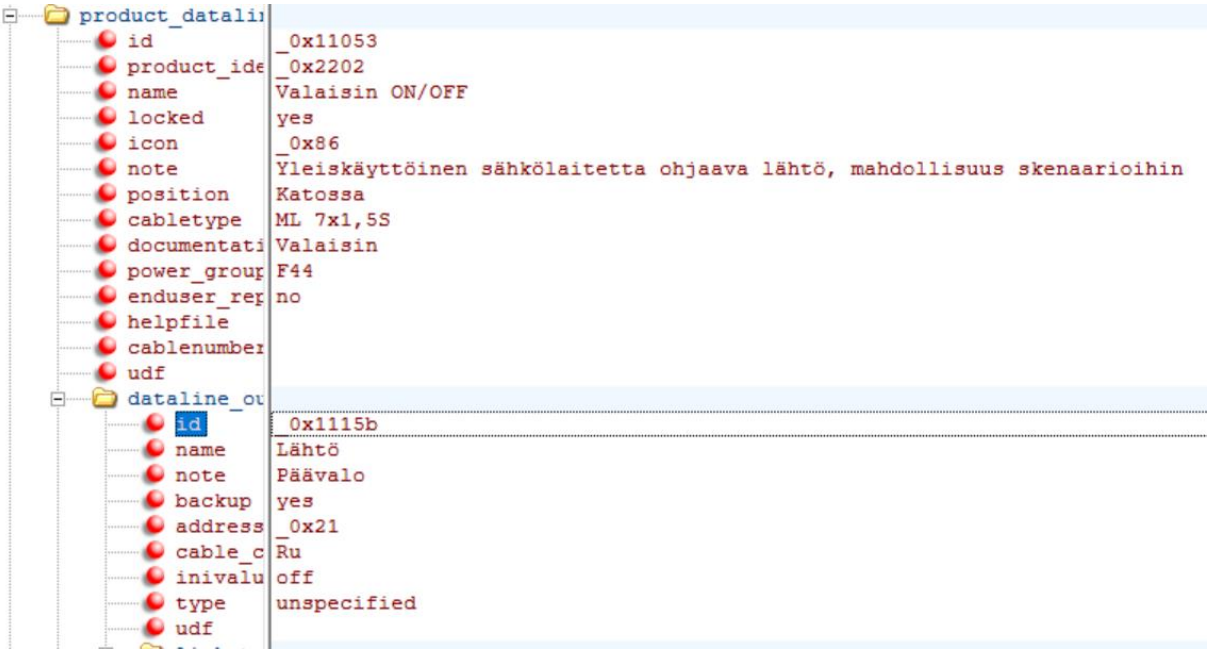
Kuvassa esitetään koko laitteiston toimivan saman internet-yhteyden alla. Käyttöliittymä lähettää ja vastaanottaa tietoa www-sovelluspalvelun kautta, joka taas kommunikoi Elkon kotiautomaatiojärjestelmän keskusyksikön kanssa. Viestit lähetetään keskusyksikön rajapintaan, ei suoraan järjestelmään. Käyttöliittymän kautta voidaan myös lähettää Modbus-protokollan mukaisesti viestinvaihdon aloittava viesti väylän master-laitteelle, eli EH-NET-palvelimelle. Palvelimen laiteosoitteissa toimivat lämmitysjärjestelmän ja ilmanvaihtojärjestelmän omat säätimet, joiden rekistereistä voidaan lukea tietoja ja joihin voidaan lähettää tietoa.

Sovelluksen turvallisuuden varmistaminen jätettiin melko vähälle huomiolle, koska ulkopuolisten pääsyä, ja ylipäätään kiinnostusta, järjestelmään pidettiin hyvin pienenä. Modbus-väylä vaatii samassa IP-osoiteavaruudessa toimimisen, joten ulkopuolisilla ei ole ilman verkon tunnuksia pääsyä laitteiston tietoihin. Molemmat järjestelmät vaativat tietojen lukemista ja lähettämistä varten autentikointitiedot. Sovellus tullaan asentamaan paikallisesti tilaajan valitsemalle laitteelle, joten sen haltuun saaminen vaatisi jo ponnisteluja mahdolliselta uhkaajalta. Käyttöliittymän avaaminen ei kuitenkaan erikseen vaadi kirjautumista esimerkiksi salasanan avulla.

Koska toiminnot olivat luotuina jo valmiiksi Elkon järjestelmään, jäi käyttöliittymän tehtäväksi yksinkertaisesti laukaista halutun toiminnon toteutus, joten suunnittelussa ei tarvinnut erikseen kiinnittää huomiota, miten jokin toivottu toimenpide saadaan aikaiseksi. Käyttöliittymään oli käytännössä tehtävä lukutoiminto, joka noutaa tietoja Elkon järjestelmästä, ja datan lähetystoiminto, jolla voidaan muun muassa syöttää ajastuksia ja lähettää toimintojen toteutuskäskyjä keskusyksikölle. Sovellukseen suunniteltiin painikkeita, jotka painettaessa lähettävät niihin kiinnitetyn toimenpiteen funktiokoodin, eli sysäyksen, www-sovelluspalvelurajapinnan kautta Elkon keskusyksikölle, joka suorittaa toimenpiteen.

Elkon ohjelmoitavaan logiikkaan luotuja toimintoja päästiin tarkastelemaan XML-tiedostosta, joka oli mahdollista ladata ulos järjestelmästä. Tiedoston lukemisesta teki helppoa XML-Notepad-sovellus, jonka avulla toimintoja oli helppo listata puumalliin. Kuvassa 6 on kuvakaappaus tiedoston lukemisesta XML-Notepadilla.

Kuva 6. Kuvakaappaus Elkon järjestelmään ohjelmoidusta logiikasta



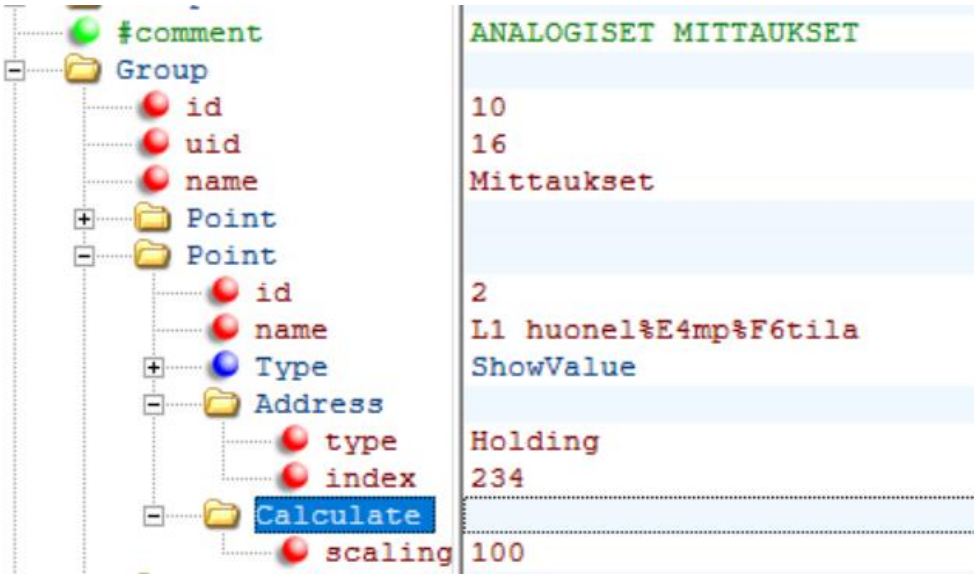
Yllä olevassa kuvassa esitellään toiminto, jolla voidaan laittaa ruokailutilan kattovalaisin päälle. Tiedostosta etsitään toimintokoodi, joka on kirjattu kohtaan dataline_output. Toimintokoodi voidaan kirjata painikkeen taakse ja näin lähettää se käyttöliittymästä keskusjärjestelmälle, joka toteuttaa toimenpiteen.

Myös kaikki luettavat tiedot, kuten onko jokin tietty valo päällä, on mahdollista toteuttaa id-koodien perusteella. Lähes kaikki rakennuksen valot ja tärkeimmät pistorasiat on liitetty järjestelmään ja nimetty ohjelmoitavaan logiikkaan selkeästi, mistä syystä niiden hakeminen ja hyödyntäminen oli kätevää.

Modbus-tiedonsiirron käyttäminen oli toimintokuvauksensa lupausten mukaisesti helppoa ja yksinkertaista: Käyttöliittymästä, joka toimii slave-laitteena, voidaan lähettää yksinkertainen viesti master-laitteelle, joka esimerkiksi palauttaa jonkin tietyn arvon. Suurin osa Oumanin järjestelmästä tarvittavista tiedoista oli lukutietoja, kuten mitatut lämpötilat.

Rekisterit, joista tietoa luetaan ja joihin tietoa syötetään, on valmiiksi asetettu ja ne löytyvät säätimistä ulos ladattavista tiedoista, jotka voidaan esittää XML-muodossa. XML-Notepad oli jälleen kätevä työkalu, jolla tietoja oli helppo etsiä ja tarkastella. Alla olevassa kuvassa 7 on esitetty, miten rekisterit ja niiden tarkoitus esitetään tiedostossa.

Kuva 7. Kuvakaappaus Modbus-väylässä olevan slave-laitteen rekisteristä



Address-, eli otsikkokansio kertoo rekisterin tyypin ja indeksin, eli osoitteen. Holding-tyyppi tarkoittaa, että rekisterissä pidetään tietoa. Kun sieltä halutaan lukea arvo, lähetetään kutsu lukea tietue rekisteristä: rekisterinumero - 1. Scaling tarkoittaa mittakaavaa, jossa tieto on. Skaalauksen ollessa 100 luetaan rekisteristä lämpötilaksi esimerkiksi 2200, joka tarkoittaa 22 astetta.

Alla olevassa kuvakaappauksessa, kuva 8, on näytetty kuinka sisälämpötilan arvo luetaan Modbus-väylää ja -tiedonsiirtoa hyödyntäen koodillisesti. Ensin luodaan Client, jolle annetaan parametreiksi master-laitteen IP-osoite ja portti, jota käytetään. Unit Identifier on tieto, joka lisätään Modbus ADUn otsikkoon, ja se kertoo slave-laitteen osoitteen. Yhteyden muodostamisen jälkeen tähän osoitteeseen lähetetään ”lue rekisteri”-viesti, jonka parametreiksi annetaan rekisteri, josta alkaen ja kuinka monta rekisteriä tietoa luetaan.

Kuva 8. Koodiesimerkki arvon lukemisesta Modbus-protokollan avulla

```
class Lampo
{
    ModbusClient modbusClient = new ModbusClient("192.168.0.101", 502);

    2 references
    public Lampo()
    {
        this.modbusClient.UnitIdentifier = 5;
    }

    3 references
    public double getSisaLampotila()
    {
        try
        {
            modbusClient.Connect();
            int[] readInputRegisters = modbusClient.ReadHoldingRegisters(233, 1);
            modbusClient.Disconnect();
            double lampotila = (1.00 * readInputRegisters[0]) / 100;
            return lampotila;
        }
        catch (Exception e)
        {
            return 0;
        }
    }
```

Yllä olevassa esimerkissä viesti lähetetään Modbus-väylän laiteosoitteeseen 5, jossa lämmitysjärjestelmän säädin on. Säätimestä luetaan yksi rekisteri numerosta 233 alkaen. Rekisterin sisältö luetaan taulukkoon, yhteys katkaistaan ja tulos muutetaan luettavaan muotoon, joka sitten palautetaan käyttöliittymään, eli Modbus-väylän slave-laitteelle Samalla periaatteella tietoja voitiin myös lähettää master-laitteelle.

Näiden toimintojen ja tietojen avulla käyttöliittymän rakentaminen oli mahdollista aloittaa.

4.4 Toteutus

Käyttöliittymä toteutettiin Visual Studio 2019 -ohjelmalla ja alustaksi valittiin .NET Framework WPF-sovellus (Windows Presentation Foundation), sillä alusta oli tekijälle tuttu ja tilaaja toivoi kannettavalla tietokoneellaan toimivaa käyttöliittymää. Ohjelmointikieleksi valittiin C#, koska sekin oli tekijälle entuudestaan tuttu, ja lisäksi varma kieli toimivan kokonaisuuden aikaansaamiseksi, koska muutama löydetty, samantyylinen toteutus oli luotu samalla kielellä.

Ensimmäisenä testattiin rajapintayhteyden ja Modbus-väylän yhteyden muodostaminen ja tiedonsiirron toimivuus ja luotiin integraatiot. Rajapintayhteys Elko Living Systemiin saatiin hyödyntämällä valmista www-sovelluspalvelu-rajapintaa, jonka järjestelmä tarjoaa. Modbus TCP-yhteyden muodostamista varten löydettiin avuksi Stefan Rossmanin luoma kirjasto: EasyModbusTCP v.5.5 (http://easymodbustcp.net/en/).

Www-sovelluspalvelun kautta yhteyden muodostaminen ja tiedonsiirto onnistui valmistajalta saadulla, luottamuksella annetulla ohjeella, joten kirjastoja ja käytettyjä funktioita ei esitellä tässä työssä tarkasti. Lyhykäisyydessään prosessi eteni niin, että ensin VS-projektiin lisättiin palveluviittaus; www-sovelluspalvelun URL-osoite, eli IP-osoite, johon laite on kytketty ja josta WSDL-tiedosto löytyy ohjelman luettavaksi ja hyödynnettäväksi. Tämän jälkeen mahdollistettiin laitteeseen kirjautuminen syöttämällä autentikointitiedot, minkä jälkeen resursseja voitiin hyödyntää lukemalla tietoja ja lähettämällä toimintoja laukaisevia funktiokoodeja.

Yhteys Oumanin järjestelmään, Modbus-väylän kautta, ei vaatinut kuin samassa IP-osoiteavaruudessa toimimisen, ja valmiin kirjaston hyödyntämisen. Järjestelmään avattiin yhteys käyttäjätietojen kera ja pyydettiin vastausta master-laitteelta. Master-laite palautti vastauksen pyydetystä rekisteristä ja se muutettiin tarpeen mukaan käyttöliittymässä luettavaan muotoon. Esimerkki tästä on esitetty luvussa 4.3.

Kun yhteys molempiin järjestelmiin oli saatu luotua, ja tiedonhaku ja testitoimintojen toteutus onnistui, luotiin valmiiksi luku- ja kirjoitusfunktiot, joita voitiin hyödyntää käyttöliittymässä. Sen jälkeen käyttöliittymään lisättiin painikkeet ja tekstikentä, ja tarvittavat funktiokoodit ja rekisterit kirjattiin niiden taakse. Käyttöliittymä ohjelmoitiin hakemaan tietoa automaattisesti järjestelmistä kymmenen sekunnin välein. Tällä syklillä päivitettiin kaikkia haluttuja lämpötilatietoja. Kellonaika

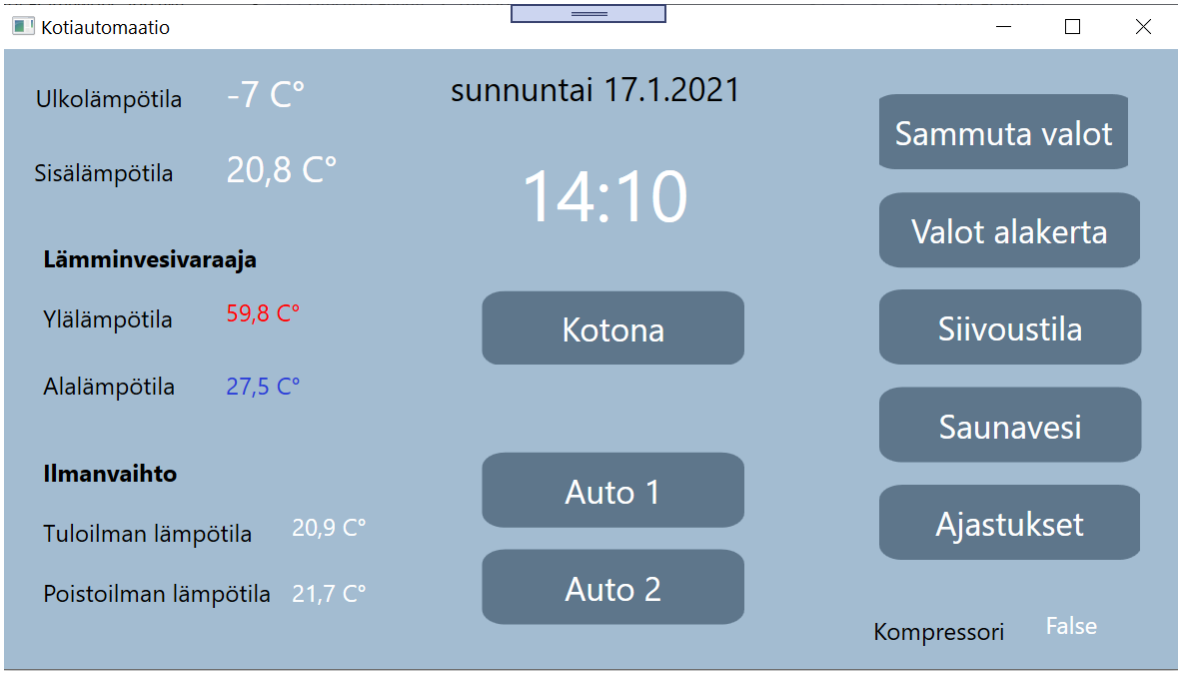
ja päivämäärä noudetaan samalla syklillä Elkon järjestelmästä, mutta päivitetään sovellukseen minuutin tarkkuudella.

Tärkeää toteutuksessa oli huomioida etenkin virheensieto, sillä järjestelmän tietopyynnöt eivät aina onnistuneet. Tätä varten luotiin logiikka, jonka mukaan tiedonhakukutsun palauttaessa virheen, näytetään käyttöliittymässä edellisen haun mukainen arvo.

5 Lopputulos

Lopputuloksena saatiin toimiva kokonaisuus, jonka avulla voitiin tarkastella ja muuttaa molempien siihen liitettyjen järjestelmien tietoja. Alla, kuvassa 9, on esitetty käyttöliittymän pääikkuna. Käyttöliittymän ulkoasun syvempi suunnittelu jäi kokonaisuudessaan tekemättä, mistä syystä sovellus onkin varsin ruma.

Kuva 9. Kuva luodun käyttöliittymän pääikkunasta



Lämpötilatiedot: sisä- ja ulkolämpötila, lämminvesivaraajan ylä- ja alalämpötila ja ilmanvaihtokoneen poisto- ja tuloilman lämpötilat, haetaan Modbus-väylän avulla ja päivitetään käyttöliittymään näkyville. Oikealla alakulmassa oleva kompressorin tilatieto kertoo onko kompressori päällä, eli lämmittääkö se tällä hetkellä. Tieto haetaan lämmitysjärjestelmän

säätimestä. Ikkunassa olevat päivämäärä ja kellonaika noudetaan Elkon keskusyksiköltä. Luoduista painikkeista voidaan lähettää nimensä mukainen sysäys keskusyksikölle: Sammuta valot-painike sammuttaa kaikki valot, ja Auto 1 -painike käynnistää toisen auton lämmityksen. Ajastukset-painikkeesta voidaan Elkon tietokantaan lähettää uudet ajastukset ulkovaloille ja autojen lämmityksille.

Pääikkunan lisäksi sovelluksessa on muutama apuikkuna, jotka aukeavat tietojen syöttämistä varten. Tällainen ikkuna löytyy sisälämpötilan kohdalta, jossa voidaan muuttaa haluttua sisälämpötilaa, minkä seurauksena Modbus-väylän master-laite välittää tiedon joko lämpötilan nostamiseksi tai vähentämiseksi. Samanlainen yksinkertainen ikkuna uuden arvon lähettämiseen on luotu lämminvesivaraajan lämpötilojen kohdille, mistä tieto menee varaajan säätimeen ja vettä aletaan lämmittää tai lämmitys lopetetaan. Lisäksi autojen ja pihavalojen ajastaminen aukeavat omaan ikkunaan.

Joitain vaatimusmäärittelyssä asetettuja ominaisuuksia jäi tästä sovellusversiosta toteuttamatta. Esimerkiksi historiatietoja ei voi käyttöliittymän kautta tarkastella. Tämä ominaisuus koettiin tässä vaiheessa toisasteiseksi ja se siirrettiin kehitysjonossa myöhemmäksi. Kiukaan säätäminen jätettiin käyttöliittymästä pois, sillä toiminto oli käytössä epävarma.

6 Yhteenveto ja pohdinta

Kokonaisuudessaan työ oli onnistunut ja opettavainen. Työn suorittamisessa korostui erityisesti projektinhallinta, joka piti sisällään alkukartoituksen, toteutuksen suunnittelun ja aikataulutuksen, tiedonhankinnan, sovelluksen suunnittelun ja lopullisen toteutuksen. Työtä tehdessä heräsi kiinnostus kokonaisvaltaiseen projektinhallintaan ja arkkitehtuurisuunnitteluun.

Alussa työmäärä vaikutti melko pieneltä, mutta tehdessä tuli selväksi, että yksinkertainenkin ja valmiita toimintoja hyödyntävä käyttöliittymä ei ole itsestään selvä toteutus. Testaamista ja koodin refaktorointia oli tehtävänä odotettua enemmän. Yllättäen haastavinta oli luoda käyttöliittymälle edes etäisesti miellyttävä ulkoasu. WPF:n tarjoamat ominaisuudet ovat varsin alkeelliset, ja esimerkiksi painikkeen ulkoasun muokkaaminen vaatii paljon ohjeiden kaivelua ja toteutuksen tuskaa.

Testausvaiheessa kävi myös selväksi, että suunnitelmassa ja toteutuksessa jäi huomiotta, että laitteistoja saatetaan käyttää edelleen myös omien järjestelmiensä kautta, ja tietysti myös fyysisesti esimerkiksi valokatkaisijoista. Tällöin käyttöliittymään koodatut oletustiedot eivät välttämättä pidä paikkaansa. Esimerkiksi kotona/poissa-kytkimen aloitustilaksi on oletettu kotona. Kuitenkin, jos eteisessä ei ole kotiin tullessa painettu fyysistä kytkintä, joka kytkee kotona-tilan päälle, on käyttöliittymässä väärä tieto, kun se käynnistetään.

Tämän ongelman poistamiseksi tulisi käyttöliittymän käynnistämisen yhteydessä juosta läpi funktio, joka tarkistaa kytkettyjen komponenttien tilan ja merkitsee sen muistiin. Toiminto suunniteltiin ja sitä testattiin hieman, mutta toteutusta ei aikataulusyistä viety tähän versioon.

Yhdeksi ongelmaksi meinasi muodostua teknologia, joka on tulossa jo tiensä päähän. Rakennusaikana käyttöönotettu Elko Living System-järjestelmä oli uusi, mutta nyt, yli 15 vuoden jälkeen, järjestelmää voi sanoa jo vanhaksi. Tietoja järjestelmän rajapinnasta ja järjestelmästä ylipäätään oli vaikea saada. Lähitulevaisuudessa fyysinen laitteisto tulee päivittää uudempaan, jolloin myös rajapinta tulee muuttumaan.

Modbus-väylässä olevat laitteet sen sijaan toimivat moitteetta, eikä suurempia ongelmia ollut. Välillä tiedonhaussa tapahtui virhe, mutta niiden vaikutus sovelluksen toimintaan saatiin eliminoitua hyvin. Tietojen lähettämisessä ei havaittu virheitä.

Seuraava askel kehitystyössä on lisätä tilojen tarkastaminen ja muuttaa sovelluksen ulkoasua. Lisäksi kymmenen sekunnin välein tapahtuvaa tiedon hakua voisi olla tarpeen keventää. Esimerkiksi päivämäärän ja kellonajan hakeminen näin usein antaa turhaa kuormaa ja voi näin vaikuttaa sovelluksen toimivuuteen ja verkon kuormitukseen. Kotiverkossa ongelma ei kuitenkaan ole suuri. Jatkokehityksenä tulisi myös kiinnittää huomiota sovelluksen turvallisuuteen ja mahdollisesti lisätä kirjautumisikkuna ennen sovelluksen tietoihin pääsemistä.

Koko opinnäytetyöprojektia katsoessa olisi kehitysvaiheessa tapahtuvaan järjestelmän testaamiseen pitänyt käyttää enemmän aikaa. Työtä tehtiin pääosin ilman yhteyttä integroituihin järjestelmiin, jolloin virheet tiedonsiirrossa jäivät havaitsematta ja testauksen jälkeen muutettavaa oli runsaasti. Ohjelmointityön suorittaminen paikan päällä, tai toimivan etäyhteyden avulla olisi ollut toimivin ratkaisu suorittaa käyttöliittymän toteutus.

Jatkoa ajatellen opinnäytetyöstä jäi kuitenkin paljon käteen ja kynnys toteuttaa kotiautomaatio ja sen hallintajärjestelmä itsenäisesti omaan kotiin on matalampi. Projektinhallinta ja ohjelmointitaidot vahvistuivat työn pohjalta ja kokonaan uutta osaamista tuli integraatioista.

Lähteet

Alfame Oy. (n.d.). *Integraatio-opas, Opas sujuvaan tietojärjestelmäintegraatioon.* Haettu 28.11.2020 osoitteesta https://www.alfame.com/integraatio-opas-opas-sujuvaan-tietojarjestelmaintegraatioon?hsCtaTracking=5663f0c6-e2a6-4453-9eab-5b39a9e6b286%7C07805598-c897-4789-8058-245cc5b22d41

Haglund, J. (9.1.2018a). *Järjestelmä integraatio, mitä se on selkokielellä?.* [Integraatio-opas] (Alfame) Haettu 30.11.2020 osoitteesta https://www.alfame.com/blog/jarjestelmaintegraatio-mita-se-on-selkokielella

Haglund, J. (19.6.2018b). *API:t käytännössä - selkokielinen katsaus*. Haettu 11.30.2020 osoitteesta Alfame: https://www.alfame.com/blog/apit-kaytannossa-selkokielinen-katsaus

IBM. (6.9.2000). *Web Services architecture overview*. Haettu 12.12.2020 osoitteesta https://www.ibm.com/developerworks/web/library/w-ovr/

LogiNets. (n.d.). *Miten saada järjestelmät keskustelemaan keskenään? - järjestelmäintegraatio*. Haettu 25.11.2020 osoitteesta https://loginets.com/fi/miten-saada-jarjestelmat-keskustelemaan-keskenaan-jarjestelmaintegraatio/

Mikkonen, J. (27.5.2017). *Rest on nettipalveluiden yhteinen kieli*. Haettu 17.8.2020 osoitteesta Tivi: https://www.tivi.fi/uutiset/rest-on-nettipalveluiden-yhteinen-kieli/23703ab5-dd19-383e-a422-ebfc3d910583

Modbus Organization. (2006). *Modbus messaging on TCP/IP implementation guide V1.0b.* [Sovellusopas] Haettu 12.12.2020 osoitteesta https://modbus.org/specs.php

Modbus Organization. (n.d.). *Modbus FAQ: About The Protocol.* Haettu 10.12.2020 osoitteesta https://modbus.org/faq.php

Motiva Oy. (25.10.2019a). *Taloautomaatio on järkisijoitus - Motiva*. Haettu 30.11.2020 osoitteesta https://www.motiva.fi/koti_ja_asuminen/hyva_arki_kotona/taloautomaatio

Motiva Oy. (25.10.2019b). *Taloautomaatio pientalossa - Motiva*. Haettu 30.11.2020 osoitteesta https://www.motiva.fi/koti_ja_asuminen/hyva_arki_kotona/taloautomaatio

Oracle. (2013a). *The Java EE 6 Tutorial: What Are Web Services?* Haettu 8.12.2020 osoitteesta https://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html

Oracle. (2013b). *The Java EE 6 Tutorial: Types of Web Services*. Haettu 8.12.2020 osoitteesta https://docs.oracle.com/javaee/6/tutorial/doc/gigsx.html

Palo, H. (2016). *Kotiautomaatiojärjestelmän valinta omakotitaloon.* Opinnäytetyö, Vaasan ammattikorkeakoulu. http://urn.fi/URN:NBN:fi:amk-2016060111390

Poikola, A., Kivekäs, O., Kettunen, J., Polo, T., Laine, S., Aaltonen, J., ... von Willebrand, M. (11.10.2014). *Avoimen rajapinnan määritelmä*, 1.0. Haettu 20.11.2020 osoitteesta http://avoinrajapinta.fi/

Sanastokeskus TSK ry. (n.d.). *TEPA-Termipankki: Web-palvelu*. Haettu 9.12.2020 osoitteesta https://termipankki.fi/tepa/fi/haku/web-palvelu

Tieteen termipankki. (2018). Nimitys:älytalo. https://www.tieteentermipankki.fi/wiki/Nimitys:älytalo

Toivanen, A. (n.d.). *Integraatiot ja integraatioalustat - lyhyt oppimäärä*. (H. F. Oy, Tuottaja) Haettu 25.11.2020 osoitteesta https://hiq.fi/ajankohtaista/integraatio/

Tolppi, P. (2018). *Älytalosta arvoa asiakkaalle.* Kanditaatintutkielma, Tampereen yliopisto. http://urn.fi/URN:NBN:fi:tty-201902151245

Tutorialspoint. (n.d.-a). *SOAP - Quick Guide*. Haettu 8.12.2020 osoitteesta https://www.tutorialspoint.com/soap/soap_quick_guide.htm

Tutorialspoint. (n.d.-b). *Web Services - Quick guide*. Haettu 9.12.2020 osoitteesta https://www.tutorialspoint.com/webservices/web_services_quick_guide.htm

Tutorialspoint. (n.d.-c). *WSDL - Quick Guide*. Haettu 9.12.2020 osoitteesta https://www.tutorialspoint.com/wsdl/wsdl_quick_guide.htm

Tutorialspoint. (n.d.-d). *XML - Overview*. Haettu 9.12.2020 osoitteesta https://www.tutorialspoint.com/xml/xml_overview.htm

W3Schools. (n.d.). *JSON - Introduction*. Haettu 8.12.2020 osoitteesta https://www.w3schools.com/js/js_json_intro.asp

WAGO Finland Oy. (n.d.). *MODBUS - Teollisen tiedonsiirron protokolla.* Haettu 10.12.2020 osoitteesta Nopea tiedonsiirto automaatio- ja kenttälaitteiden välillä: MODBUS: https://www.wago.com/fi/modbus