Teemu Vartiainen

Paloilmoitinjärjestelmän informaationäyttö

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

25.10.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Teemu Vartiainen Paloilmoitinjärjestelmän informaationäyttö 42 sivua + 2 liitettä 25.10.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Automaatiotekniikka
Suuntautumisvaihtoehto	Automaation tietotekniikka
Ohjaaja(t)	Lehtori Antti Liljaniemi Järjestelmäasiantuntija Aapo Vuoristo

Tässä insinöörityössä toteutettiin pelastusviranomaisten työtä helpottava Web-pohjainen käyttöliittymä, josta jo ensisilmäyksellä pystyy näkemään lauenneiden palohälyttimien sijainnit kiinteistössä. Käyttöliittymälle tuleva informaatio saadaan kiinteistön automaatiojärjestelmästä.

Työssä oli kaksi erillistä osa-aluetta: backend ja frontend, eli palvelinpuolen sovellus ja sen käyttöliittymä. Backend-sovellus kommunikoi kiinteistön automaatiojärjestelmän kanssa OPC UA -protokollalla. Käyttöliittymälle - ja edelleen käyttäjälle - informaatio siirtyy WebSocket-rajapinnan välityksellä.

Tuloksena saatiin rakennettua toimiva käyttöliittymä, josta pelastusviranomaiset saavat tehokkaasti reaaliaikaista tietoa kiinteistöstä. Tämä tieto sisältää muun muassa kiinteistön kerrosten pohjapiirustukset, asemakaavan sekä useita muita pelastustyötä helpottavia tietoja.

Avainsanat	web, käyttöliittymä, opc ua, beckhoff

Author(s) Title Number of Pages Date	Teemu Vartiainen Fire alarm system information display 42 pages + 2 appendices 25 October 2015
Degree	Bachelor of Engineering
Degree Programme	Automation Engineering
Specialisation option	Information Technology in Automation
Instructor(s)	Antti Liljaniemi, Senior Lecturer Aapo Vuoristo, Control System Specialist

The aim of this Bachelor's thesis was to create a web-based user interface for a fire alarm system. The main purpose of this user interface is to help rescue personnel during their missions, in case of fire. All the information will be received from building automation systems.

There are two main sections in this thesis: the backend, for server side application and the frontend, for user interface. The backend section contains an OPC UA client for receiving information from the building automation system. Also the backend application is a WebSocket server for publishing the information to the user of this information display.

As a result, we managed to create a good and very useful user interface where, in case of emergency, rescue personnel are able to see easy all the required information concerning the building. For example, that information contains layouts of the levels of the building, fire alarm and motion sensor information, as well as other details to make their work easier.

Keywords	web, user interface, opc ua, beckhoff

Sisällys

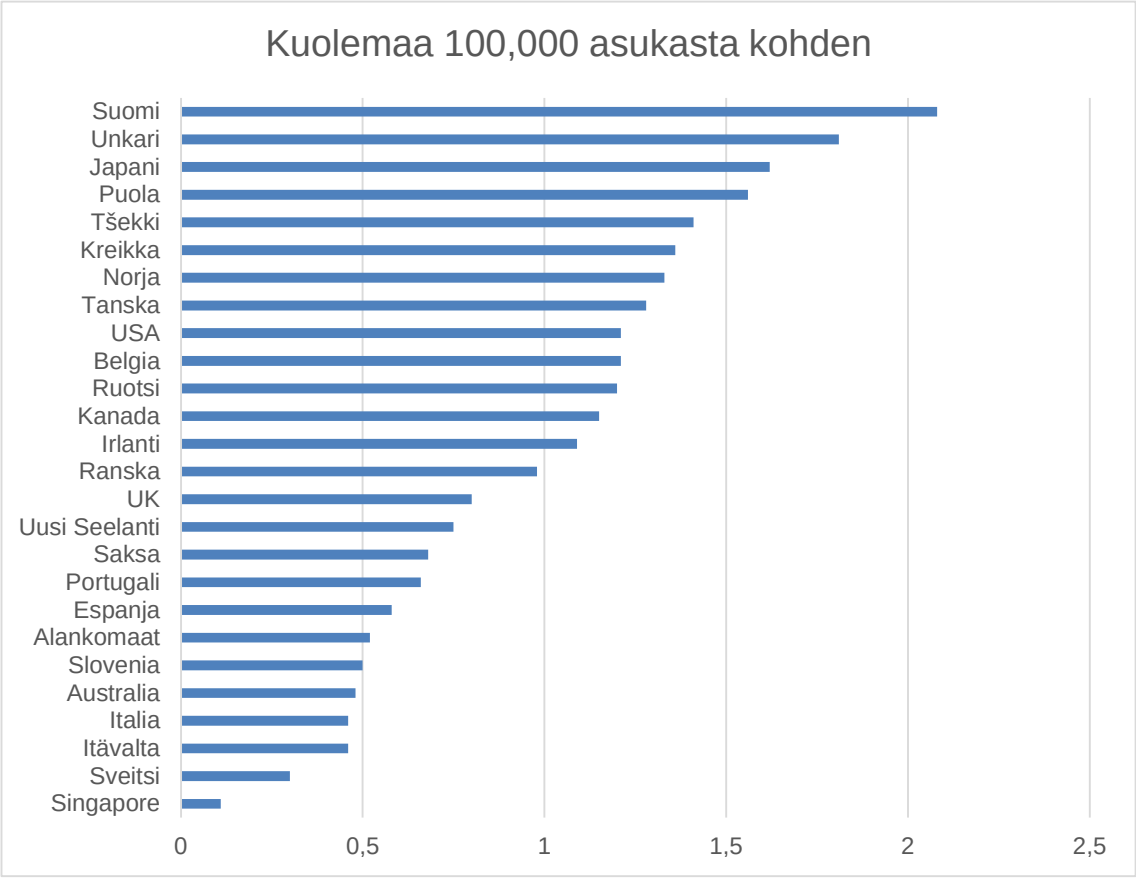
Käsitteet ja lyhenteet

PLC	Programmable Logic Controller, Ohjelmoitava logiikka.
OPC UA	Avoin tiedonsiirtoprotokolla, josta UA – Unified Architecture - on uusin versio.
Client	Asiakas, käyttäjä.
Server	Palvelin, johon ”client” voi ottaa yhteyden. Palvelin voi olla esimerkiksi verkkopalvelin tai jonkin tiedonsiirtoprotokollan palvelin.
Backend	Ohjelmoinnissa käytettävä termi, ohjelman osa, joka ei ulospäin näyttäisi tekevän mitään
Frontend	Ohjelmoinnissa käytettävä termi, jonkinlainen käyttöliittymä backend-sovellukseen
WebSocket	Palvelimen ja selaimen välinen tiedonsiirtoyhteys. Websocketia käyttämällä voidaan toteuttaa dynaamisesti päivittyvää tietoa verkkosivulla. Websocket verkkosivuilla toteutetaan tyypillisesti JavaScriptin kautta.
HTML	HyperText Markup Language, ”hypertekstin merkintäkieli”. Käytetään verkkosivujen toteutuksessa. Kuuluu HTML5-kieleen.
HTML5	Pääsääntöisesti sama, kuin HTML, mutta pitää sisällään myös CSS ja JavaScriptin. W3C:n suositus vuodesta 2014 lähtien.
CSS	Cascading Style Sheets, ”porrastetut tyyliarkit”. Verkkosivun tyyliohjeet, eli esimerkiksi sivun väritykset ja fonttien koot. Kuuluu HTML5-kieleen.
JavaScript	Komentosarjakieli. Tyypillisesti verkkosivuilla käytettävä kieli, jolla saadaan toteutettua dynaamisista sisältöä. Kuuluu HTML5-kieleen.

W3C	World Wide Web Consurtiom, kansainvälinen yhteisö. Ylläpitää WWW-standardia tai suosituksia. Verkkosivut: http://www.w3.org/
IP	Internet Protocol, tiedonsiirtoprotokolla. Tieto siirretään IP-osoitteen perusteella laitteesta toiseen.

1 Johdanto

Suomi on useissa kansainvälisissä vertailuissa listan kärjessä, valitettavasti myös palokuolematilastossa. Vuosina 2006 – 2008 Suomessa kuoli yli kaksi henkeä 100,000 asukasta kohden, joka on vertailumaiden heikoin tulos. Mikään muu vertailumaa ei ylittänyt kahden hengen rajaa, ja muun muassa myös Ruotsissa (1,20 kuolemaa 100,000 asukasta kohden) ja Norjassa (1,33) palokuolemat näyttävät tämän tilaston valossa huomattavasti paremmalta. Vertailun tulokset ovat nähtävissä kuvassa 1. [1]



Kuva 1. Vuonna 2006 – 2008 välillä tapahtuneet palokuolemat maittain. [1]

1.1 Taustat

Isojen kiinteistöjen, kuten kauppa- tai yrityskeskusten, palohälytykset voivat viedä palokunnalta ylimääräistä aikaa, kun he tutkivat pohjapiirustuksista mistä päin rakennusta hälytys on tullut. Toistaiseksi palokunta joutuu hälytyksen tullessa menemään ensin paloilmaisinkeskukseen ja tarkistamaan mikä ilmaisin hälyttää ja miten kyseiseen tilaan parhaiten pääsee. Ilmaisimet ovat merkattu pohjapiirrokseen positio numeroilla ja kiinteistön ollessa suuri, voi positionumeron etsimiseen mennä aikaa useita minuutteja. [2]

Toisaalta ajateltaessa pientaloja, kuten omakotitaloja ja rivitaloja, tulipalon uhreja lähdetään kiinteistöstä etsimään ”oikean- tai vasemmankäden taktiikalla”. Tällä tarkoitetaan sitä, että palomiehet kylmästi vain arvaavat kumpaan suuntaan kiinteistöä olisi parempi lähteä etsimään mahdollisia palon uhreja, tai savua ollessa paljon, jopa palon lähdettä. Eteneminen toteutetaan niin, että pidetään joko oikeaa tai vasenta kättä kokoajan seinässä kiinni. Näin kuljetaan koko rakennus ympäri. Heillä ei siis ole rakennukseen astuessaan yleensä tietoa missä päin palonlähde, esimerkiksi sauna, sijaitsee. [2]

Myös poliisi on kiinnostunut kiinteistöjen pohjapiirustuksista. Koska kiinteistöjen pohjapiirustusten saanti on hyvin hankalaa ja aikaa vievää, erilaiset suunnitelmat on haastavia tehdä. Heidän tulee pyytää aluehallintovirastosta nämä haluamansa dokumentit ja ne ovat auki vain arkipäivisin, joten myös kriisitapauksissa on pohjapiirrosten saanti nopeasti usein hyvin haastavaa. Näin ollen, jos poliisilla olisi mahdollisuus saada kiinteistönpohjapiirros muutamalla hiiren näpäytyksellä, toisi se ison edun tehokkuuteen. Myös pääsy reaaliaikaiseen tietoon, kuten liiketunnistininformaatioon, tehostaisi poliisin toimintaa kriisitilanteissa. [2]

Tekniikan kehittyessä ja erilaisten energiatehokkuusvaatimusten myötä, tulee kiinteistöautomaatio yleistymään huomattavasti lähivuosina. Tämä koskee niin isoja kiinteistöjä, kuten kauppakeskuksia ja muita julkisia rakennuksia, kuten myös yksityisiä asuinkiinteistöjä. Automaatiolla voidaan ohjata kiinteistön perustoimintoja, kuten valaistusta, ilmanvaihtoa ja lämmitystä. Joissakin kohteissa voidaan mennä myös

esimerkiksi ovien, sälekaihtimien tai saunan ohjaamiseen sekä energian- ja vedenkulutuksen seurantaan. [3]

Vaikka rakennusautomaatio jatkaa kasvuaan ja isoissa kiinteistöissä onkin lain vaatima automaattinen paloilmoitinjärjestelmä, ei siellä usein ole muuta tietoa, kuin ”talo palaa” tai ”hälytys ilmaisimessa ABC.123” -tiedot. Tämän päivän teknisillä ratkaisuilla voitaisiin tarjota viranomaisille tarkempaa tietoa syttyneestä palosta tehokkaasti. Pienemmissä kiinteistöissä, kuten omakotitalot ja vapaa-ajan asunnot, nämä eivät kuulu tämän lain piiriin, eikä niitä vapaaehtoisesti haluta niiden korkean hinnan vuoksi. Mutta sopivalla kiinteistöautomaatiojärjestelmällä tällaisen toiminnon integrointi olisi järkevä ja edullinen vaihtoehto.

1.2 Työn tavoite

Tämän insinöörityön tavoitteena on esittää yksi mahdollinen ratkaisu yllä mainittuihin ongelmiin. Työssä tullaan suunnittelemaan ja toteuttamaan käyttöliittymä, josta viranomaiset voivat saada kohteesta selville oleellisia tietoja:

- kiinteistön pohjapiirustukset, jossa palovaroitin- ja liiketunnistintiedot olisivat näkyvillä
- asemakaavakuvan, sisältäen muun muassa hyökkäystiet ja kokoontumispaikat sekä
- kohdekortin, josta käy ilmi muun muassa kiinteistön omistajan yhteystiedot, kiinteistön kerrosten lukumäärän ja onko kiinteistössä esimerkiksi kaasupulloja.

Kaikki tämä pitäisi pystyä saamaan näkyviin avaamalla käyttöliittymä ja klikkaamalla hiirtä muutaman kerran ruudulla.

Käyttöliittymästä ja siinä käytetyistä ratkaisuista pyritään toteuttamaan sellaiset, että ne toimisivat mahdollisimman hyvänä lähtökohtana niin teknisesti, kuin informatiivisesti myös myöhempään ja laajempaan käyttöön. Se myös pyritään toteuttamaan sellaiseksi, että se olisi mahdollisimman laitteistoriippumaton. Tekniikat, joita työssä tullaan käyttämään, pyritään valitsemaan siten, että ne olisivat vielä useamman vuoden päästä täysin toimintakykyisiä. Käyttöliittymää käytetään pääsääntöisesti

kosketusnäytölliseltä päätteeltä, joten tämä asia pyritään ottaman mahdollisimman hyvin huomioon käytettävyyden kannalta.

Taustaohjelman puolen rakenne pyritään myös rakentamaan siten, että pienillä muutoksilla olisi mahdollisuus luoda yhteys lähes mihin tahansa kiinteistöautomaatiossa käytettävään järjestelmään. Näin ollen tämän hälytysjärjestelmän käyttöliittymä ei olisi laitteistosta tai valmistajasta riippuvainen ohjelmisto vaan pystyisi toimimaan lähes missä laitteessa tahansa.

Tärkein tutkimusväline sovelluksen - käyttöliittymän ja taustaohjelman - luomiseen lienee Internet ja sen erilaiset verkkosivut ja keskustelupalstat. Järjestelmä tulee vaatimaan jonkin laajassa käytössä olevan tiedonsiirtoprotokollan käyttöä. Alan kirjallisuus sekä laitevalmistajien tällä hetkellä tukemat protokollat olisi hyvä tutkia ja selvittää. Myös valitun tiedonsiirtoprotokollan olisi hyvä olla myös pitkälle tulevaisuudessa käytettävä. IoT (engl. Internet of Things), esineiden internet, on nyt paljon esillä ja tämä on hyvä pitää mielessä myös tätä insinöörityötä tehdessä.

Työn esimerkkinä ja hälytysjärjestelmän pilottikohteena tulevat toimimaan Beckhoff Automation Oy:n uudet toimitilat, jonne myös palokunta tulee tekemään harjoitushälytyksen tulevan syksyn aikana. Tällöin palokunnalle tarjottaisiin tarkkaa tietoa muun muassa kiinteistön pohjapiirustuksista ja missä päin rakennusta hälytys on tullut ja on sillä hetkellä aktiivisena. Harjoituksessa tullaan mittaamaan muun muassa kuinka paljon tällainen järjestelmä auttaisi ja nopeuttaisi pelastusviranomaisia työtehtävissään ja voisiko näin ollen turvata omaisuutta sekä säästää jopa ihmishenkiä nykyistä paremmin.

Työn tärkein tavoite on saada tämän tyyppiselle ratkaisulle hyvä alku, jota olisi mahdollista monistaa helposti laajempaan käyttöön, ehkä jopa viranomaissuositukseksi kiinteistön automaatiota suunniteltaessa. Tätä kautta saadaan mahdolliset hälytystilanteet mahdollisesti paremmin ja nopeammin haltuun, ja sitä pienennettyä niin henkilö- kuin omaisuusvahinkoja.

1.3 Dokumentaation rakenne

Luku 2 sisältää vaatimusmäärittelyn tehtävälle työlle. Vaatimusmäärittelyssä määritellään mitä järjestelmässä tulee olla, miten sen tulee toimia ja mitä laite- ja ohjelmistovaatimuksia järjestelmällä saa tai pitää olla.

Kolmas luku keskittyy järjestelmän kuvaukseen, eli kuinka se on rakennettu. Luvussa kerrotaan muun muassa laitteistojen fyysiset sijainnit ja tiedonsiirtoreitit.

Luvussa 4 käsitellään järjestelmän ”backend”-osuus, eli ohjelmistokerros, jossa tietoa siirretään ja käsitellään eteenpäin käyttöliittymälle.

Frontend-osio, luku 5, sisältää kaiken visuaalisen puolen ominaisuudet, käytettävyyden ja niin edelleen. Tämä on työn pääosuus, joka on tämän hetken ”puuttuva rengas” pelastus- ja sammutustehtävissä.

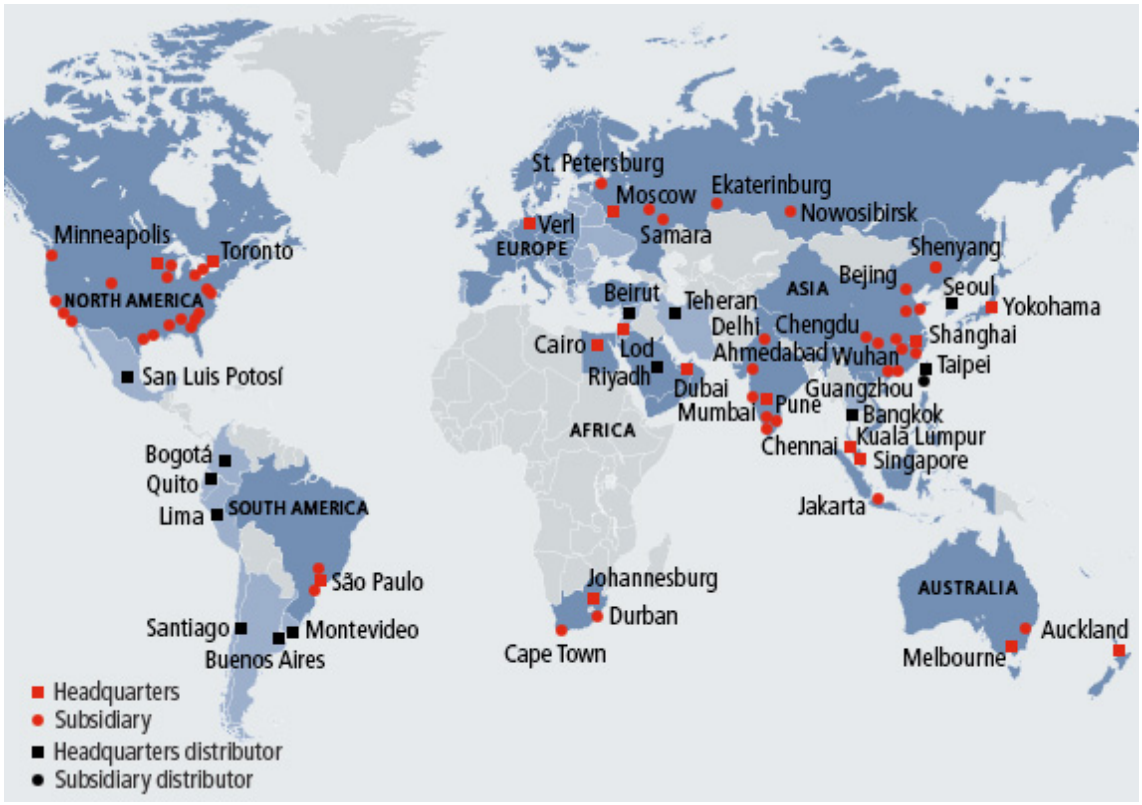
Luku 6 keskittyy loputa lopputulokseen; mitä tuli tehtyä, miten siinä onnistuttiin sekä kuinka järjestelmää voisi kehittää eteenpäin. Luvussa pohditaan myös järjestelmän tulevaisuutta yleisellä tasolla.

1.4 Beckhoff Automation Oy

Beckhoff Automation Oy on saksalaisen automaatiojärjestelmiä valmistavan Beckhoff Automation GmbH & Co. KG:n tytäryhtiö. Suomessa Beckhoff on toiminut vuodesta 1986 edustajan välityksellä, kunnes se perusti Suomeen oman yhtiön vuonna 2000. Toimipisteitä Beckhoffilla on Suomessa kolme: Hyvinkäällä, Seinäjoella ja Tampereella. Jatkossa toimipiste tulee olemaan myös Tallinnassa, Virossa, jota tällä hetkellä hoidetaan Hyvinkään toimipisteen kautta. Kaikissa Suomen toimipisteissä on edustettuna myynti, tekninen tuki, koulutus, tuotekehitys, sovellukset ja huolto. Hyvinkäällä on myös Beckhoff komponenttien varasto. [4]

Pääkonttori Beckhoff Automation GmbH & Co KG:llä on Saksan Verlissä, ja sillä on yhteensä 11 toimipaikkaa eri puolilla Saksaa. Beckhoff yhtiönä työllistää maailmanlaajuisesti (kuva 2) 2,800 henkeä ja sen liikevaihto vuonna 2014 oli 510

miljoonaa euroa. Kasvua edellisvuoteen verrattuna oli 17 prosenttia. Tytäryhtiötä sillä on yhteensä 34 ja jälleenmyyjiä yli 70. [4]



Kuva 2. Beckhoff Automation GmbH & Co. KG:n tytäryhtiöt ja jälleenmyyjät maailman laajuisesti. [4]

Beckhoffin automaatiojärjestelmään voidaan integroida kaikki kiinteistön toiminnot, kuten:

- Lämmitykset ohjaus
- Valaistuksen ohjaus
- Ilmanvaihto
- Hälytykset
- Kulunvalvonta
- Kulutuksen (sähkö, vesi) seuranta.

Kaikkien komponenttien sisältyessä samaan järjestelmään, saadaan automaatiojärjestelmästä tehtyä hyvin kustannustehokas. Toisaalta Beckhoffin

järjestelmä on helposti skaalautuva, eli siihen voidaan valita vain tarvittavat komponentit eikä näin ollen tarvitse järjestelmässä olla mitään ylimääräisiä liityntöjä.

2 Vaatimusmäärittely

Vaatimusmäärittely on liitteen 1 pohjalta laajennettu ja tarkennettu, niin käyttäjän, kuin myös järjestelmän näkökulmasta katsottuna. Liite kumminkin määrittelee vain vaadittavia toimintoja, joten tämä vaatimusmäärittely sisältää huomattavasti enemmän ja tarkemmin järjestelmän toimintaa ja toimintoja sääteleviä asioita. Vaikka tämän kaltaiselle järjestelmälle ei ole olemassa omaa standardia, voidaan SFS EN 54-standardia käyttää joissakin toiminnollisuuksissa soveltuvin osin.

Vaatimusmäärittely on kirjoitettu laajempaa käyttöä silmällä pitäen, joten välttämättä kaikki vaatimusmäärittelyn ominaisuudet ja määrittelyt eivät tule olemaan esimerkkikohteessa toteutettuna.

Järjestelmän ensisijaisena käyttäjänä ovat pelastusviranomaiset.

2.1 Käyttöympäristö

Järjestelmää tullaan pääsääntöisesti käyttämään liikkuvasta paloautosta, tai ambulanssista, joissa on omat tietokoneet ja näytöt. Jos kiinteistössä on erillinen näyttöpääte, voidaan hälytyksen ollessa aktiivisena, näyttää tieto myös siinä. Kosketusnäytöllisellä paikallispäätteellä voidaan myös tietoja selata samalla tavalla, kuin paloautosta tai ambulanssista, tai jostain muusta ulkopuolisesta yhteydestä.

Mobiililaitteiden yhteensopivuutta, toiminnollisuutta tai käytettävyyttä ei tarvitse tässä vaiheessa huomioida.

2.2 Vaatimukset

Raaliaikaisen tiedonsiirto käyttäjälle voidaan toteuttaa, jos kiinteistössä on toimiva Internet-yhteys. Tällöin myös modeemin, reitittimen tai palomuurin asetuksista pitää huomioida ainakin se, että tiedonsiirto kulkee ulospäin valittua protokollaa käyttäen. Myös sisäänpäin tuleva liikenne pitää olla avoinna määritetyllä portilla, jos itse käyttöliittymä sijaitsee kiinteistössä itsessään. 3G-tasoinen Internet-yhteys pitää olla riittävä, jotta tieto saadaan kohtuullisessa ajassa käyttäjälle näkyviin.

Kiinteistön IP-osoite tulee olla ennakkoon tiedossa, jotta päätteeltä voidaan ottaa yhteys kiinteistön palo- ja liiketunnistininformaatioon.. Tämä tarkoittaa sitä, että kiinteistöllä pitää olla joko kiinteä IP-osoite tai jokin muu ratkaisu, jotta yhteyden luominen ilman erillistä selvittelyä on mahdollista.

Toisaalta, jos jokin sovelluksen osa – kuten taloautomaatiojärjestelmä – on toimimaton, tulee käyttöliittymän kuitenkin toimia mahdollisimman informatiivisesti. Kyseisessä tapauksessa käyttöliittymässä tulee näkyä kohdan 2.5 Toiminnallisuudet tiedot, ilman reaaliaikaista tiedonsiirtoa. Näin ollen suositus olisi, että vaikka käyttöliittymän voisi integroida myös automaatiojärjestelmään, kannattaa se asentaa toiselle laitteelle. Laajassa käytössä olisi järkevää pitää käyttöliittymäpuoli fyysisesti erillään kohdekiinteistöstä.

2.2.1 Laitteistovaatimukset

Itse hälytysjärjestelmän pitää toimia mahdollisimman pienin resurssein ja mahdollisimman kevyellä laitteistolla. Järjestelmästä pitää olla ainakin yksi liitäntä, jota kautta saadaan luotua Internet-yhteys. Tämä voi olla LAN- tai WLAN-liitäntä.

Järjestelmä ei saa vaatia lisälaitteiston hankintaa alun perin suunniteltuun kiinteistönautomaatioon, jos muut, laitteisto- tai ohjelmistovaatimukset nykyisessä järjestelmässä jo toteutuvat. Palo- ja liiketunnistimet tulee tietenkin olla jo kiinteistöautomaation puolelta olemassa.

2.2.2 Ohjelmistovaatimukset

Hälytysjärjestelmän ohjelmisto pitää tehdä niin, ettei se ole käyttöjärjestelmästä riippuvainen. Tai, niin että se kattaa mahdollisimman suuren osan tällä hetkellä käytössä olevista käyttöjärjestelmistä.

Lähtökohtaisesti järjestelmä tulee suunnitella Windows-käyttöjärjestelmään, mutta olisi hyvä, jos se toimisi ilman suuria muutoksia myös muissa käyttöjärjestelmissä.

2.2.3 Esimerkkikohteen muutosvaatimukset

Beckhoff Automation Oy:n aulassa on yksi kosketusnäytöllinen pääte, jossa pääsääntöisesti on kiinteistön normaali käyttöliittymä. Taustalle pitää kumminkin lisätä erillinen toiminto, joka avaa hälytysjärjestelmän sivun, jos tai kun palohälytys on aktiivinen. Näin pelastushenkilökunnalla on mahdollisuus saada kiinteistöstä kuva heti sinne saapuessa. Nykyiseen käyttöliittymään tulee lisätä myös erillinen linkki tähän paloilmoitinjärjestelmän käyttöliittymään.

2.3 Liitynnät ja rajapinnat

Liityntä järjestelmään tulee olla avointa, standardisoitua ja laajalti tuettua protokollaa. Näin järjestelmään liittyminen olisi mahdollista useimmilla ohjelmistoilla ja protokollalla olisi myös vankka tulevaisuuden näkymä. Protokolalla pitää olla laaja käyttäjäkunta laitevalmistajien piirissä, sekä valitulla protokolalla tulee olla vankat tulevaisuuden näkymät. Yhteys hälytysjärjestelmään otetaan etänä, joten vähintään yksi verkkoliitäntä pitää olla jokaisella osapuolella käytettävissä.

Tiedonsiirto järjestelmästä käyttäjälle tulee olla kevyt ja nopea, mutta kumminkin riittävän monipuolinen mahdollisten laajenemisien myötä. Sanomana on hyvä käyttää jotain standardia, jolloin sen purkaminen käyttöliittymän päässä olisi mahdollisimman helppoa ja selkeää.

2.4 Toiminnallisuudet

Pasi Kääriäiseltä saadun muistilistan (liite 1) sekä käytyjen keskustelujen [1] mukaan, käyttöliittymään pitää saada seuraavia näkymiä:

- Pohjapiirros
- Asemakaavapiirros
- Kohdekortti.

Näitä on liitteessä ja keskustelujen pohjalta myös tarkennettu oheisella tavalla:

Pohjapiirros:

- Pohjapiirroksesta on käytävä ilmi, missä tilassa hälyttänyt paloilmaisin on ja sen numero. Kuvasta on tultava myös visuaalisesti ilmi, kuinka kauan hälytys on ollut päällä.
- Pohjapiirroksessa pitää olla huonerajat, sekä niissä palo-osastot rajattu kolmipistekatoviivoin.
- Savunpoistoluukut pitää olla pohjakuvassa selvästi näkyvissä
- Se paikka, josta hälytys on aktiivisena, tulee näkyä punaisena pallona, tai muuten selkeästi ilmi tulevana symbolina. Mitä pidempään hälytys on ollut päällä, sitä isompi pallon tulee olla.
- Jos kiinteistössä on kerroksia viisi tai alle, voidaan ne laittaa päävalikkoon kaikki näkyviin. Muussa tapauksessa kerrosten pitää sivulla olla erillinen kerroksen valinta-näppäin.
- Oletussivuna näkyy aina hälyttävän kerroksen pohjakuva.

Asemakaavapiirroksessa on käytävä ilmi:

- Ilmansuunta ja lähikadut
- Paloilmoitinkeskus, sprinklerikeskus ja palokunnan sprinkleri-syöttö, savunpoistokeskus sekä putkilukko
- Hyökkäystiet, eli ovet joista pääsee rakennukseen sisälle
- Ilmanvaihdon, kaasu, veden ja sähkön pääsulkujen sijainnit
- Palavien nesteiden ja muiden vaarallisten aineiden sijainnit
- Väestönsuoja, ilmanvaihtokonehuone, kokoontumispaikka, portti ja ajoreitit.

Kohdekortin tiedot, jotka ovat liitteestä 2 mainittu.

3 Tekniset ratkaisut

Testikohteen kiinteistön automaatiojärjestelmänä toimii Beckhoff CX2040 sulautettu PC., joka on kuvassa 3 keskellä Kyseinen laite hallinnoi useita kiinteistön toiminnollisuuksia, kuten valaistusta ja ilmanvaihtoa, sekä lämmitystä/jäähdystystä ja kulunvalvontaa.

WWW-palvelimena on virtuaalinen tietokone, johon on asennettuna Windows 2008 Server –käyttöjärjestelmä, Internet Information Services WWW-palvelin, TwinCAT ADS sekä nykyinen kiinteistön Web-pohjainen käyttöliittymä. WWW-palvelimella on myös PHP-tuki, jota käytetään kiinteistössä tietokannasta muun muassa lämpötilatietojen kyselyissä ja näyttämisessä graafisessa kaaviossa.

Kuva 3. CX2000-sarjan sulautettu PC. [4]

Ensisijaisena näyttöpäätteenä toimii Beckhoffin toimitilan aulassa oleva 24 tuumainen CP3924-monikosketusnäyttö. Yhteyden luominen kiinteistön käyttöliittymään onnistuu myös yrityksen tietokoneilta tai älypuhelimilla.

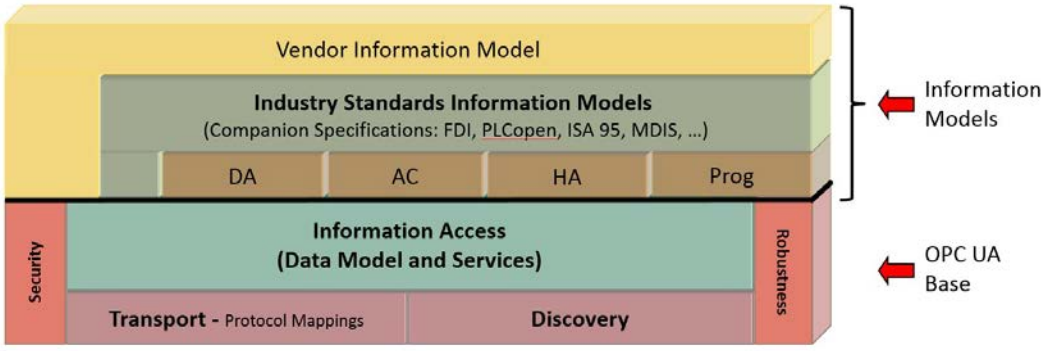
3.1 Tiedonsiirto

Tiedonsiirtoprotokolloja on olemassa hyvin suuri kirjo. Kiinteistöautomaatiossa tutuimmat lienevät KNX tai BACnet. Teollisuuden puolella yleisiä protokollia on vielä laajemmin käytössä, joita ovat muun muassa OPC UA, Modbus, TCP/IP. Lisäksi on

vielä laitevalmistajien omat tiedonsiirtoprotokollat kuten Beckhoffin ADS. Jotkin protokollat ovat Ethernet-pohjaisia, jotkin esimerkiksi sarjaliikennepohjaisia. Tietenkin kaikki jotka eivät ole Ethernet-pohjaisia, karsiutuu saman tien pois, koska muunnin tulee käyttämään vain verkkoliityntöjä.

Myös uusia vaihtoehtoja, kuten MQTT, pohdimme vaihtoehdoiksi, koska se vaikuttaa todella lupaavalta tiedonsiirtoprotokollolta. MQTT:llä on myös muutamia nimekkäitä käyttäjiä, kuten esimerkiksi Facebook, joka käyttää sitä Messenger-sovelluksessaan[5]. Mutta koska MQTT on kumminkin aika vähän käytetty päätettiin se jättää toistaiseksi syrjään.

Pohdituista tiedonsiirtoprotokoloista sopivimmaksi tuntui kumminkin OPC UA. OPC on laajalti tunnettu, turvallinen ja vanhahko protokolla; vanhempi versio, tunnetaan myös nimellä ”OPC Classic” ja se on perujaan 90-luvulta alusta.[7] Uudempi versio, eli nimeltään OPC Unified Architecture (OPC UA, kuva 4), on myös laiteriippumaton, joten se ajaa perusidean tarkoitusta. Useat laitevalmistajat tukevat OPC UA:ta omissa järjestelmissään ja näiden yhdistäminen toisiinsa ei pitäisi olla ongelma. OPC UA -palvelin tarvitsee vain asentaa ja konfiguroida järjestelmän ohjaimelle sopivaksi jonka jälkeen yhteys palvelimeen on hyvin suoraviivaista.

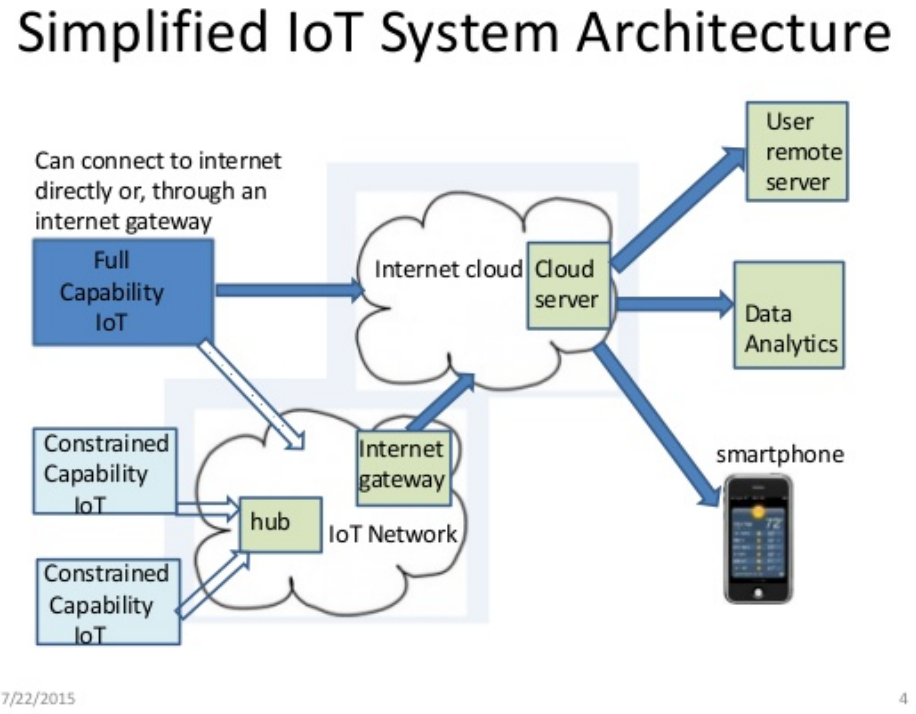


Kuva 4. OPC UA arkkitehtuuri. [6]

Useat käyttöliittymäsovellukset tukevat OPC UA:ta, ja protokololla voidaan saada myös aika helposti kaksi tai useampia erilaisia logiikkoja kommunikoimaan keskenään. Laitteet voivat olla palvelimia, asiakkaita tai molempia.

OPC UA on nyt myös tulossa kiinteistöautomaatioonkin mukaan. Näin ollen kiinteistön automaatiosta on mahdollista ja helppoa yhdistää lähes mihin tahansa toiseen

järjestelmään, laitteisiin tai sovellukseen. Myös IoT–ajatusmaailma on OPC UA:lla helposti toteutettavissa, onhan se ”teolliseen internettiin” juuri sopiva ratkaisu.[8] IoT-tyyppinen ratkaisu (kuva 5) tämän tyyppisen hälytysjärjestelmänkin tulisi oikeastaan olla. Hätäkeskus esittää tässä tapauksessa pilveä, kiinteistöt tietolähdettä ja näyttöpäätteen edessä oleva henkilö on luonnollisesti käyttäjä, joka pystyy ottamaan yhteyden hälytyskeskuksen järjestelmään mistä päin vain, eli tässä tapauksessa esimerkiksi paloautosta.



Kuva 5. Yksinkertaistettu tapa esittää IoT kuvana. [9]

OPC UA:ta ylläpitää ja kehittää OPC Foundation, jonka kotisivuilta löytyy paljon tietoa protokollasta, mukaan lukien ohjelmaesimerkkejä ja markkinointimateriaalia. Jotkin sivut ja materiaalit kumminkin ovat vain jäsenille tai yritysjäsenille. [6]

On luontevaa, että käyttöliittymä on Web-pohjainen, koska silloin se on:

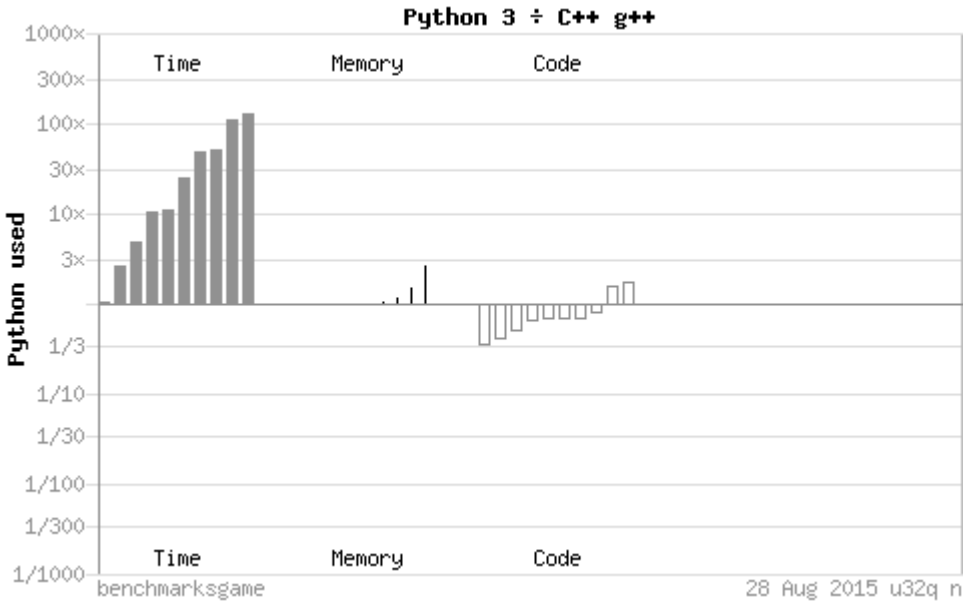
- Laitteistoriippumaton käyttäjän näkökulmasta
- IoT-ajatusmaailmalle sopiva
- Helposti muokattava
- Mahdollisuus integroida myös muihin mahdollisiin tarpeisiin.

Verkkosivut ovat myös yleensä erittäin hyvin taaksepäin yhteensopivat, nyt tehty sivu toimii hyvin todennäköisesti samalla tavalla myös usean vuoden kuluttua. Esimerkiksi 90-luvulla tehdyt verkkosivut, jotka vielä ovat olemassa sellaisinaan, toimivat samalla tavalla tämän päivän verkkoselaimilla, kuin niiden luonti hetkelläkin. [10]

3.2 WebSocket-palvelin

Python-ohjelmointikieli tuntui tähän toteutukseen sopivalta ratkaisulta, koska sillä pystyy nopeasti ja tehokkaasti toteuttamaan monenlaisia sovelluksia, niin komentoriviltä ajettavia kuin myös graafisia ohjelmia. Kirjastotarjonta on Pythonissa myös hyvin laaja jo oletuskirjastojen kautta, ja kun sille löytyi helposti valitulle protokollalle (OPC UA) hyvä ja toimiva kirjasto oli Pythonin valinta helppo päätös. Python-tulkki on saataville lähes kaikille alustoille: käyttöjärjestelmistä esimerkiksi Windows, Linux ja OS X, prosessoriarkkitehtuureissa x86, x64 ja ARM.

Vaikka Pythonia arvostellaankin usein hitaaksi, esimerkiksi vertailemalla C++ ja Pythonia (kuva 6), nähdä selvästi Pythonin rauhallisuuden, ei se haittaa tässä sovelluksessa lainkaan. Toisaalta Pythonin kirjoittaminen taas on nopeampaa ja koska sovellus ei ole aikakriittinen, on se tähän sovellukseen todella hyvä vaihtoehto.



Kuva 6. ”Time”-palkit näyttävät erään Python vs. C++ testin tuloksia. Isompi palkki huonompi. [11]

Jos myöhemmin olisi tarvetta tehdä käyttöliittymä muuntimelle, saisi sen aika helposti rakennettua käyttäen esimerkiksi Kivy-frameworkia.

3.3 Käyttöliittymä

Käyttöliittymäsuunnittelu oli sinänsä helppo ja nopea, eihän varsinaisessa käyttöliittymässä ole kuin kolme sivua: Pohjapiirros, Asemakaava ja Kohdekortti. Vaikkakin Pohjapiirros-sivuja voikin olla useampia, ei se itse suunnitteluun varsinaisesti vaikuttanut. Perusulkoasun, eli elementtien kuten linkkien ja kuvien sijoittelun, hahmotelma syntyi hyvin jo projektin aloituksessa pidettyjen keskustelujen aikana.

Toiminnaltaan päätettiin, että käytetään WebSocketia tiedonsiirtoon sen helppokäyttöisyyden ja nopeuden vuoksi. Toinen vaihtoehto, MQTT, olisi vaatinut hieman huomattavasti enemmän työtä. MQTT olisi myös vaatinut erillisen ”brokerin”, palvelimen. Tämä olisi pitänyt luoda erikseen, ja vaikka WebSocketikin vaatii myös oman palvelimensa, on MQTT:n broker toistaiseksi raskaampi vaihtoehto eikä sille löytynyt kirjastoa Pythonille. Sillä voisi mahdollisesti saada hyvin toteutettua useamman kiinteistön pääkeskuksen. Tätä asiaa pohditaan vielä jatkokehittelyssä ja tulevaisuuden pohdinnoissa.

WebSocketin kilpaileva vaihtoehto olisi ollut AJAX, mutta tästä on pidempi vertailu Backend-kehityksessä.

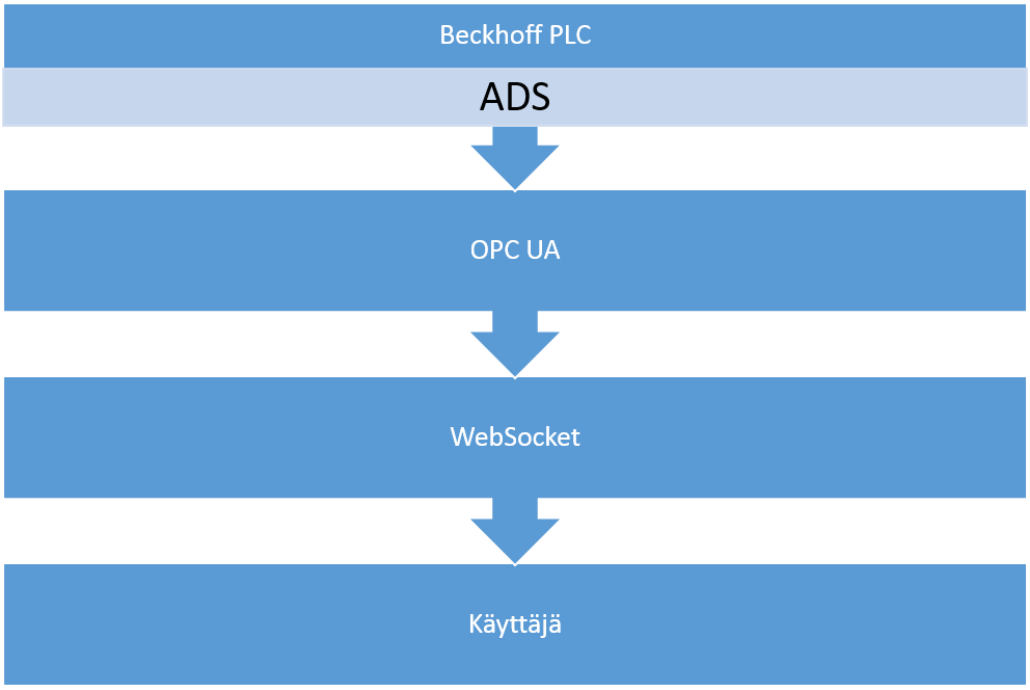
Käyttöliittymän toimivuus on helpohko testata jopa itsekseen: tarvitsee vain ladata ja asentaa muutamia verkkoselaimia omalla koneelle ja katsoa miten käyttöliittymä käyttäytyy eri tilanteissa. Mutta koska käyttöliittymä on verkkoselaimella toimiva, voisi siitä tehdä erillisen testiversion ja pyytää eri ihmisiä testaamaan käyttöliittymää käyttämällään selaimella. Tällä tavoin saisi myös suuremman otannan erilaisista kokoonpanoista ja varmistettua paremmin sen toimivuudesta.

3.4 Yleisrakenne

Avattuna rakenne on sisältää seuraavat komponentit:

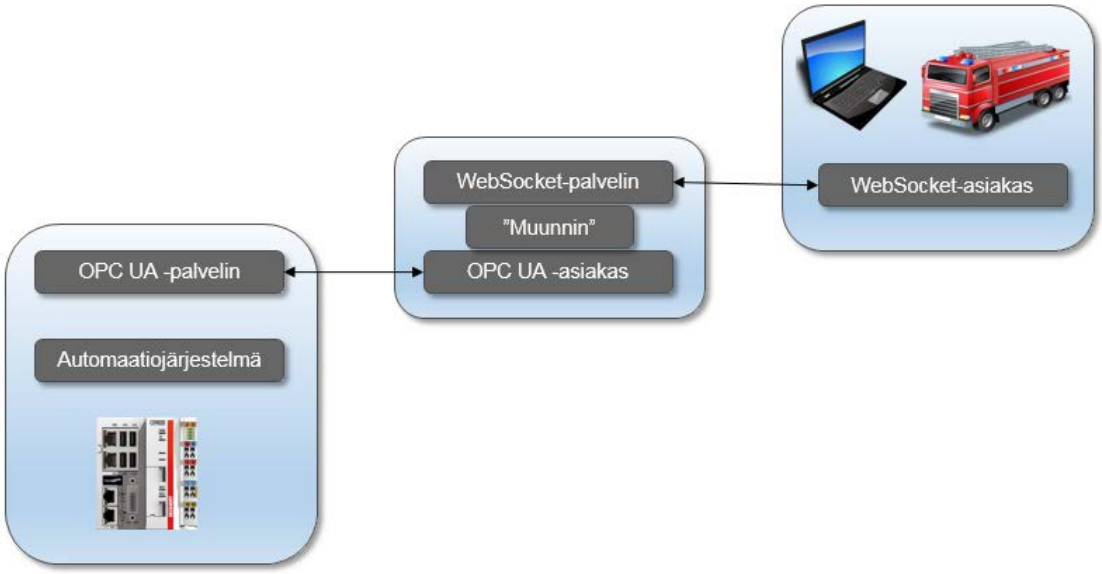
- Automaatiojärjestelmä
- OPC UA -palvelin
- OPC UA -asiakas
- WebSocket-palvelin
- WebSocket-asiakas.

Näistä jokainen komponentti voidaan pitää toisistaan fyysistesti erillään tai integroida yhdeksi isoksi kokonaisuudeksi. Jos järjestelmä olisi yleisessä käytössä, olisi silloin paras tai ainakin hyvin selkeä, että OPC UA -palvelin olisi automaatiojärjestelmässä itsessään, OPC UA -asiakas ja WebSocket-palvelin (työssä nimettynä muuntimeksi) samalla palvelimella ja WebSocket-asiakas on käyttäjän verkkoselain. Yksinkertainen tiedonsiirtorakenne on havainnollistettu kuvassa 7.



Kuva 7. Tiedonsiirtokerrokset automaatiojärjestelmästä käyttäjälle.

Laajassa käytössä kiinteistön automaatiojärjestelmässä olisi OPC UA -palvelin, joka on luonnollisesti itse kiinteistössä. ”Muunnin” sijaitsee viranomaistahon omalla suojatulla palvelimella, johon pelastushenkilökunta ottaa yhteyden omalla tietokoneellaan ja ovat WebSocket-asiakkaita.Ylesrakenne järjestelmästä ja millaista protokollaa käyttäen tieto saadaan automaatiojärjestelmästä käyttäjälle on nähtävissä kuvasta 8.



Kuva 8. Yleisrakenne järjestelmästä.

Yllä oleva ratkaisu vaatii kiinteistön puolelta palomuurilta tietyn portin avaamisen (esimerkiksi Beckhoffin OPC UA -palvelimella oletuksena portin numero on 4840). Myös kiinteistöllä pitää olla tiedossa oleva IP-osoite. Toisaalta viranomaispuolella taas uusien kohteiden liittyessä järjestelmään ei tarvitse palvelimelle tehdä varsinaisesti mitään muutoksia. Järjestelmän laajemmassa käytössä pitää viranomaisilla olla kumminkin erikseen määritelty osoitetietokanta, jonne voidaan lisätä määritetyillä tiedoilla – ehkä katuosoitteen perusteella – verkko-osoite, jota kautta kiinteistöön voidaan luoda yhteys. Tätä osiota ei kumminkaan tässä työssä käsitellä lainkaan.

Esimerkkikohteessa kiinteistön automaatiojärjestelmä on erillään muista. Kaikki muut, eli OPC UA -palvelin, OPC UA -asiakas, WebSocket-palvelin ja WWW-palvelin, ovat samalla virtuaalisella palvelimella, jonne saa yhteyden sisäverkon kautta.

4 Backend-kehitys

Backend, eli tyypillisesti palvelinpuolen sovellus, on sisältää toiminnoiltaan kommunikoinnin automaatiojärjestelmästä käyttäjälle. Kommunikointi automaatiojärjestelmään on toteutettu OPC UA -protokollaa käyttäen, josta sanoma välitetään käyttäjälle käyttäen WebSocket-tekniikkaa.

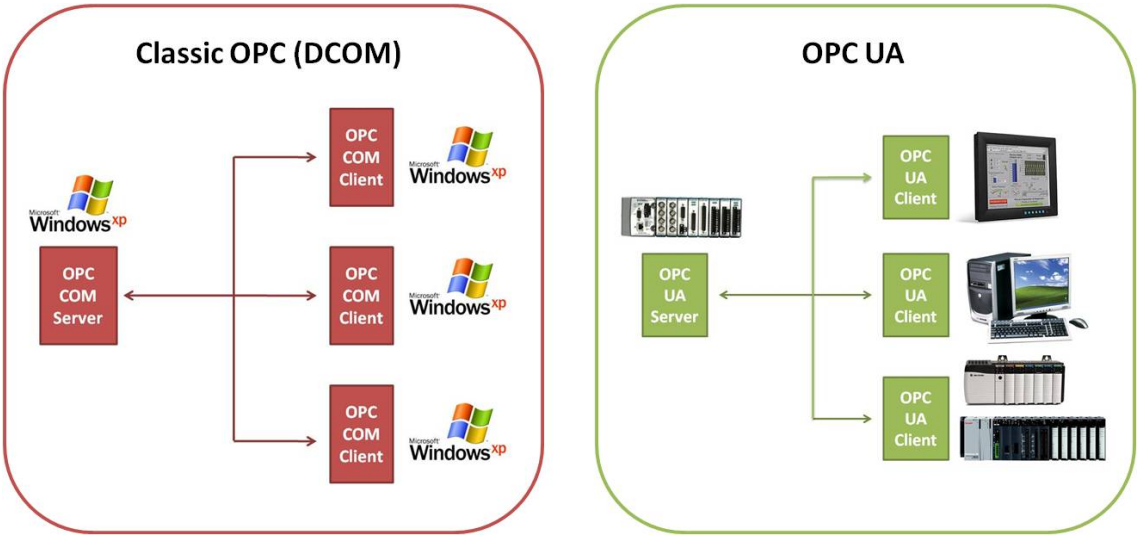
4.1 OPC UA

OPC Unified Archtitecture (OPC UA) on uudempi versio vanhasta, OPC-protokollasta. OPC UA on teollisuuden M2M-protokola (laitteelta laitteelle), ja sillä on laaja laitevalmistajien tarjoama tuki. OPC UA:ta ylläpitää OPC Foundation.

OPC UA on oikeastaan edeltäjänsä toimintojen yhteenveto. Vanhemmassa OPC:ssä oli - historiallisista syistä - kaikki toiminnot erilaisissa paketeissa. Näitä toimintoja ovat [12]:

- OPC DA (data access, eli suora reaaliaikainen tiedonhaku järjestelmästä)
- OPC HDA (historical data access, historia tietojen haku)
- OPC AE (Alarms & Events, eli hälytys- ja tapahtumatiedot, ei reaaliaikainen tieto)
- OPC Batch (batch-ohjausjärjestelmien suora yhdistys OPC-järjestelmään)
- OPC Data eXchange (suora palvelin-palvelin kommunikointi)
- OPC Security (palvelimen lähettämien tietojen salaus)
- OPC XML-DA (tietojen esittäminen XML-muodossa).

Kyseiset toiminnot sisältyvät OPC UA:han, joka on myös - edeltäjästään poiketen – järjestelmäriippumaton protokolla. Vanhempi OPC vaati Windows-käyttöjärjestelmän, sen DCOM-liityntöjen takia, mutta uudempi versio on käyttöjärjestelmä riippumaton (kuva 9).



Kuva 9. Vanhemman OPC:n (”Classic OPC”) ja uudemman OPC UA:n käyttöjärjestelmä riippuvuus kuvana. [13]

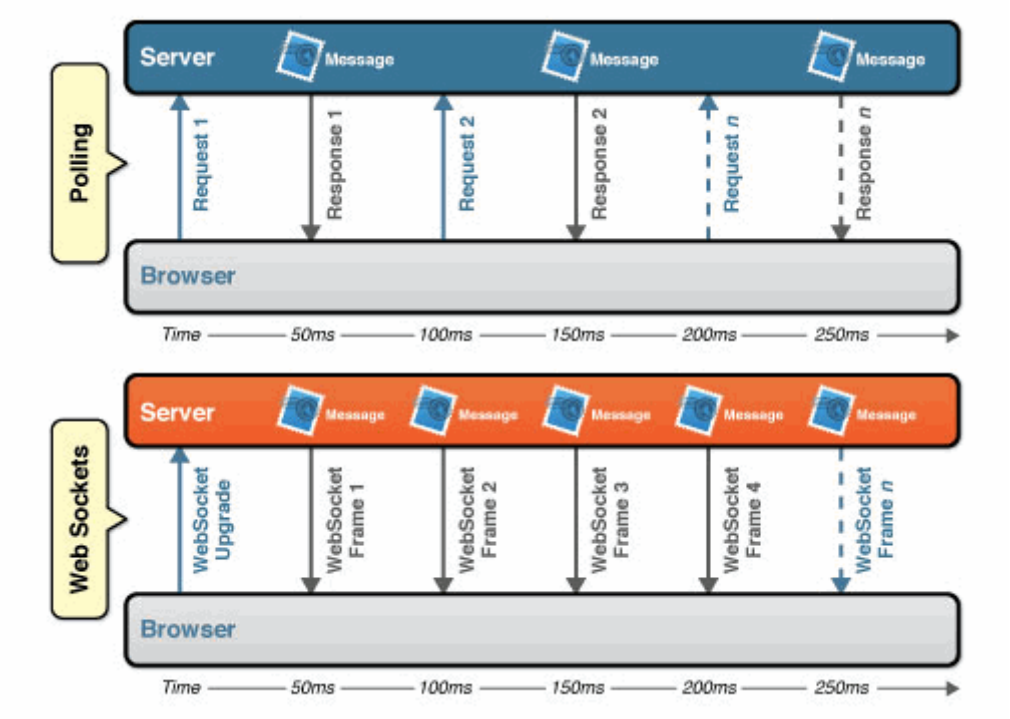
Internetistä löytyy paljon materiaalia OPC UA –protokolasta, kuten lähdekoodeja ja esimerkkejä asiakassovelluksen ohjelmointiin. Myös valmiita paketteja löytyy, joista jotkut ovat myös ilmaisia. Vaikka OPC UA:lle löytyy myös verkkoselaimelle sopivia versiota, esimerkiksi Javalla kirjoitettu, päätimme luoda kyseisen sovelluksen itse. Näin ollen saamme kaikki haluamamme toiminnot integroitua samaan ohjelmaan. Myös laiteriippumattomuuden pystymme omalla ohjelmallamme paremmin varmistamaan.

4.2 Websocket

Vielä pari vuotta sitten AJAX-tekniikka oli hyvin suosittu Web-pohjaisissa käyttöliittymissä. Sillä saa kätevästi haettua tietoa valitusta tietolähteestä aina valitulla päivitysajalla. Myös Beckhoffin tarjoama WebService toimii AJAX-kutsujen kautta, joten se on ollut aina helppo valinta. Beckhoffilla oli myös vielä pari vuotta sitten tuettuna TcScript-niminen skriptikirjasto, jota kautta logiikan ja verkkosivujen välinen yhteys oli erittäin helppo tehdä. Tässäkin informaation haku toimi parhaiten AJAX-pyynnöillä.

Suurin ongelma AJAX-tyyppisessä tekniikassa on se, että vaikka uutta tietoa ei olisikaan saatavilla, joudutaan kaikki tiedot hakemaan silti aina erikseen. Tämä johtuu HTTP-protokolan ”kysymys-vastaus”-tavasta toimia. Ratkaisuna nopeampaan ja luotettavampaan tiedonsiirtoon verkkoselaimelle tarjoaa Websocket-tekniikka.

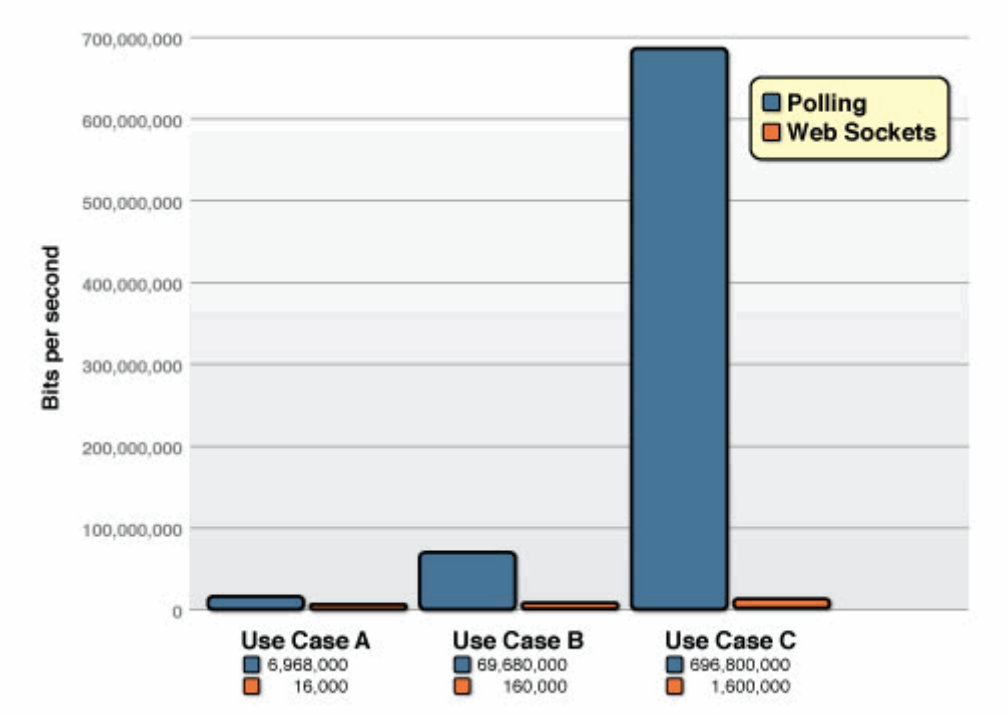
HTTP-protokolla toimii siis siiten, että verkkoselain pyytää sivun tiedot ja palvelin vastaa siihen. WebSocket toiminta on hieman erilainen: lähtökohtaisesti verkkoselain ilmoittaa palvelimelle, että ”olen paikalla”. Kun palvelimella on uutta informaatiota, lähettää se kaikille paikalla oleville asiakkaille automaattisesti. Muussa tapauksessa ei varsinaisesti tapahdu mitään näiden välillä. Kuvassa 10 tapahtumaa on hieman havainnollistettu.



Kuva 10. AJAX ja WebSocket toimivuuden eroavaisuudet. [14]

Kuvan 10 ”Polling” osassa lähetetään pyyntö 1 (”Request 1”), ja saadaan vastaus 1 (”Response 1”). Aikaa yhteen lähetys-pyyntö-kierrokseen menee esimerkki tapauksessa 100 millisekuntia. Kun vastaus on saatu, lähetään välittömästi uusi pyyntö (”Request 2”) ja niin edelleen. Tämä vastaa AJAX-tekniikkaa.

WebSocket-puolella käyttäjä ottaa yhteyden palvelimelle, muttei varsinaisesti pyydä mitään. Palvelin kuitenkin lähettää ja selain vastaanottaa ensimmäisen sanoman 50ms kohdalla, toisen 100ms ja niin edelleen. Kuvan mukaan siis WebSocket on kaksi kertaa nopeampi. Todellinen nopeusero on paremmin havaittavissa kuvasta 11, jossa on havainnollistettu WebSocketin vaatimaa liikennöintikuormaa verrattuna AJAX-tekniikkaan.



Kuva 11. Vertailussa AJAX (”polling”) ja WebSocket. Isompi palkki, enemmän liikennettä. [14]

Tutkitaan kuvan 11 tapausta tarkemmin: HTTP-otsake on 871 tavun kokoinen, eikä siinä ole vielä edes mitään ”oikeaa” tietoa. Ajatellaan, että Case C –kohdassa on 100,000 käyttäjää yhteydessä kahteen eri palveluun, toinen palvelin käyttää ”polling” ja toinen WebSocket-tekniikkaa. Molemmat lähettävät kerran sekunnissa uutta tietoa. ”Polling”-palvelimen siirtämä tieto on 871 x 100,00 = 87,100,000 tavua, eli yli 650 Mbps. Vastaavasti WebSocket-palvelimen otsake on vain kahden tavun kokoinen, jolloin vastaava luku (2 x 100,000) on noin 1,5 Mbps. [14]

AJAX-ratkaisuissa on se hyvä puoli, että se ei vaadi mitään lisäohjelmistoja väliin, riittää jos logiikalta saa tiedot vaikka tekstitiedostona tallennettua. WebSocket vaatii oman palvelimen, jonne verkkosivu ottaa yhteyden. Toisaalta WebSocket palvelimen rakentaminen on hyvin helppoa, joten ongelma sellaisen rakentaminen ei ollut.

4.3 Python

Pythonin on kehittänyt Guido van Rossum vuonna 1990. Kieltä pidetään helppona oppia, koska sen syntaksi on yksinkertaista ja kielellä on korkean tason tietorakenteet. [15]

Python on tulkattava ohjelmointikieli, mikä tarkoittaa sitä, että se on saman tien suoritettavissa ilman erillistä kääntämistä. Pieni esimerkki Python-ohjelmasta on esitetty kuvassa 12. Haittapuolena tässä on se, ettei se nopeudessaan lähtökohtaisesti pärjää esimerkiksi C++:lle tai Javalle. Toisaalta, koska Pythonissa on korkean tason tietorakenteen ja sitä on nopea ohjelmoida, saadaan sillä hyvin nopeasti tehtyä monipuolisia ohjelmia. Pythonia voidaan käyttää myös C:n tai C++:n jatkeena tai liitännäisenä, jolloin kieltä voidaan käyttää todella tehokkaasti. Nimi ohjelmointikielelle tulee suositusta Brittiläisestä TV-ohjelmasta Monty Python. [15]

```
1   class Person(object):
2       def __init__(self, name, age):
3           self.name = name
4           self.age = age
5
6       def get_name(self):
7           return self.name
8
9       def __str__(self):
10          return "My name is {name} and I'm {age} years old"\
11              .format(name=self.name, age=self.age)
12
13  man = Person("Sheldon", 27)
14  woman = Person("Penny", 22)
15  print(man.get_name) # Sheldon
16  print(str(woman))   # My name is Penny and I'm 22 years old
```

Kuva 12. Pythonissa käytettyä syntaksia, useista kielistä poiketen välilyönneillä luokissa ja funktiossa on suuri merkitys toimivuuden kannalta.

Valitsimme OPC UA – WebSocket –muuntimen ohjelmointikieleksi Pythonin, sen laajan laitteisto- ja ohjelmistotuen takia. Myös sen laaja kirjastotarjonta (tässä käytettyinä OPC UA) sekä nopea kirjoittaminen vaikuttivat suuresti valintaamme. Kirjoitettu protokollamuunnin voidaan sijoittaa niin logiikan sulautetulle käyttöjärjestelmälle, kuin myös esimeriksi verkkopalvelimelle, kunhan siellä on Python tulkki asennettuna. Samaa muunninta voidaan käyttää myös myöhemmissä projekteissa ja se on laajennettavissa myös muille tiedonsiirtoprotokolloille.

4.4 Sovellus

Tämä ohjelma tulee esimerkkikohteessa suoritetavaksi virtuaalipalvelimelle, jossa on myös asennettuna OPC UA- ja WWW-palvelimet. Myöhemmissä versioissa tämä

sovellus on se osa järjestelmää, joka olisi tarkoitus olla viranomaisten omalla palvelimella ja johon he itse ottavat sitten yhteyden.

Ohjelman toimii tapahtumakohtaisesti: kun jokin arvo muuttuu automaatiojärjestelmässä, eli jokin hälytys aktivoituu, tulee siitä tieto ohjelmalle OPC UA –protokollalla. Ohjelma käsittelee sitä hieman ja lähettää sitten tiedon eteenpäin JSON-muotoisena WebSocket-protokollaa käyttäen kaikille paikalla oleville käyttäjille.

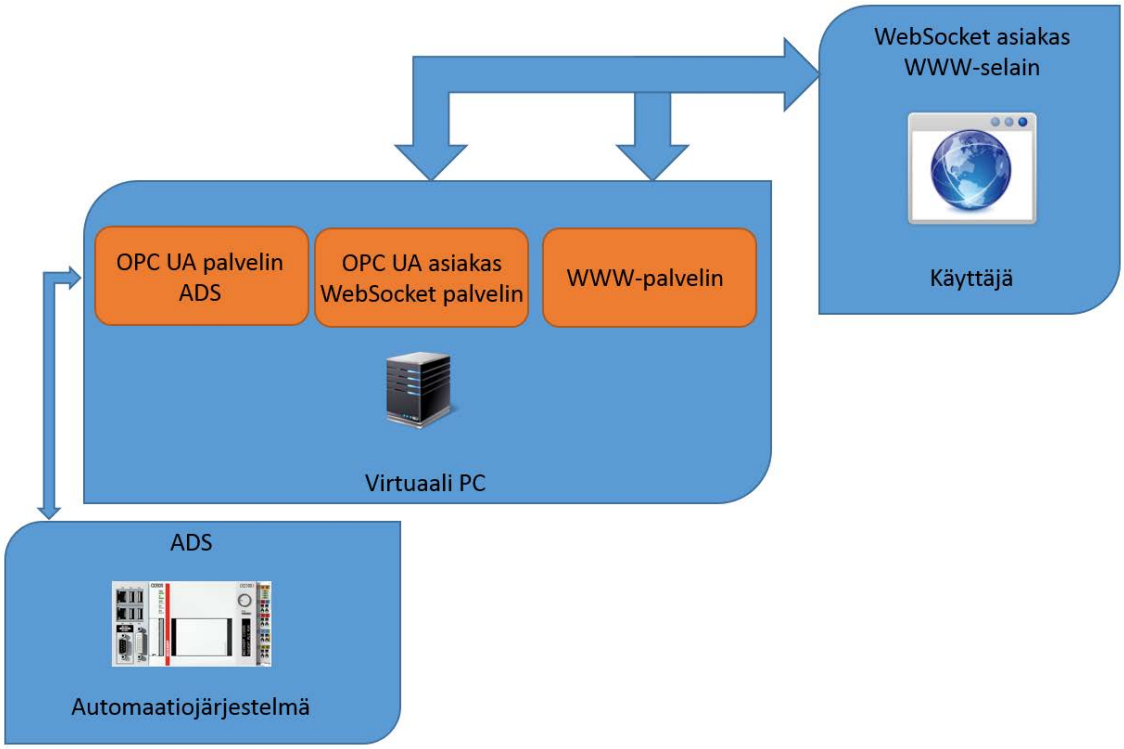
Viimeisimmät tiedot tallennetaan ohjelman sisäiseen muuttujaan, tavallaan pysyväksi tiedoksi. Kun joku ottaa yhteyden palvelimeen lähetetään nämä tiedot sille välittömästi yhteyden ollessa kunnossa. Näin ollen ohjelmalla on aina viimeisin saatu tieto kokoajan käytössä ja jaettuna sen käyttäjille.

Protokollamuunnin-ohjelmassa käytetään Free OPC UA –kirjastoa, sekä autobahn frameworkia. Free OPC UA –kirjasto on testattu kehittäjän toimesta muutaman OPC UA -palvelimen kanssa, mukaan lukien Beckhoffin OPC UA. Kirjastoa kehitetään edelleen, mutta toistaiseksi mikään toiminto ei ole jäänyt kyseisen kirjaston takia pois.

Autobahn-framework sisältää WebSocket ja WAMP (Web Application Messaging Protocol) –protokollat ja se toimii sekä Python 2, että Python 3 –versioissa. Autobahn ei sisällä verkkopalvelinominaisuuksia, mutta jotain toista frameworkia käyttäessä olisi mahdollista kumminkin sisältää kaikki käyttöliittymä-komponentit (OPC UA –yhteys, WebSocket-lähetykset ja WWW-palvelin) samaan ohjelmaan ja näin saavuttaa ehkä hieman parempi hallittavuus sovellukselle.

Sovelluksessa on mahdollista käyttää myös useampaa OPC UA palvelinta ilman erillistä lisäohjelmointia. Ajatuksena tässä on se, että jos myöhemmin on tarvetta tällaisella, on se ominaisuus jo valmiina.

Käytännössä muunnin edustaa 3.4 Yleisrakenne –kohdan mukaisesti OPC UA -asiakas ja WebSocket-palvelin –osioita. Sovellus on havainnollistettu kuvassa 13.



Kuva 13. Tiedonsiirron välineet ja reitti PLC:ltä käyttäjälle.

Kuten mainittu jo aiemminkin, automaatiojärjestelmä on erillään muista. Automaatiojärjestelmä kommunikoi TwinCAT ADS:n kautta OPC UA palvelimelle. Muuntimen, eli OPC UA -asiakkaan ja WebSocket-palvelimen, kautta saadaan yhteys OPC UA -palvelimeen ja lähetettyä sanoma kaikille WebSocket-asiakkaille. Käyttöliittymä ladataan WWW-palvelimelta, jota kautta saadaan esitettyä varsinainen käyttöliittymä. Tämän kautta saadaan myös luotua WebSocket-palvelimeen yhteys.

4.5 Tiedonsiirto käyttäjälle

Tieto siirretään muuntimelta käyttäjälle JSON-muodossa. JSON-standardi on hyvin yksinkertainen ja se perustuu avain-arvo-tyyppiseen viestirakenteeseen. Avain on aina jokin numeraalinen tai merkkijono-arvo. Arvokentässä voi taas olla lähes ihan mitä vain: objekteja, taulukkoja tai perustietotyyppejä kuten liukuarvoja, boolean-arvoja tai merkkijonoja. [16]

Kun käyttäjä ottaa yhteyden järjestelmään päätelaitteellaan, lähetetään ”alustustiedot”.

Kuvassa 14 näemme erään alustussanoman. Rivit 1 ja 16 ovat JSON-viestin aloitus- ja lopetusmerkit, joiden väliin tulee varsinaiset tiedot. Rivillä 2 lähetään alustustieto, ”init”, jolloin käyttäjän puolella oleva sovellus ymmärtää, että tiedot ovat alustukseen tarkoitettuja.

Rivit 3 – 15 ovat OPC UA noden tietoja. Nämä rivit toistuvat sanomassa niin monta kertaa, kuin eri nodeja on määritelty. Tieto sisältää seuraavaa:

- Rivi 3: tunnistetieto, käytetään verkkosivulla moneen tarkoitukseen
- Rivi 4: aikaleima, jolloin arvo on viimeksi muuttunut. Käytetään palohälytystiedoissa graafisena, sekä erona nykyiseen aikaan.
- Rivi 5: noden arvo, 1 = päällä, 0 = pois
- Rivi 6: käyttöliittymän asetukset alkavat tästä
- Rivi 7: Mistä rakennuksesta on kyse, rakennuksen id-numero.
- Rivi 8: Noden käyttöliittymässä käytettävä tyyppi, ”firealarm” tarkoittaa palovaroitintietoa ja ”motion” tarkoittaa liikkeentunnistusta.
- Rivi 9: Missä rakennuksen kerroksessa anturi sijaitsee
- Rivit 10-13: Koordinaatiotietoja käyttöliittymälle, kertoo missä kohti pohjakarttaa kyseinen anturi sijaitsee.
- Rivi 14 sulkee anturintieto-osion, ja rivi 15 JSON-sanoman. Näiden väliin tulee seuraavan anturin tiedot ja niin edelleen.

```
{
        "init": "true",
        "living_rm": {
                "timestamp": "2015-08-02 12:38:48.130000",
                "value": "1",
                "hmi": {
                        "building": "1",
                        "type": "firealarm",
                        "level": "1",
                        "position": {
                                "y": "390",
                                "x": "575"
                        }
                }
        }
}
```

Kuva 14. Alustustiedot JSON-muodossa.

Kun jokin arvo muuttuu, ei kumminkaan kaikkea tieto ole tarvetta tai järkevää lähettää uudestaan. Näin ollen sovellus lähettääkin vain oleelliset tiedot kuten kuvassa 15 esitetty.

```
{
        "entrance": {
                "timestamp": "2015-08-02 12:53:24.731000",
                "type": "motion",
                "value": "0"
        }
}
```

Kuva 15. Päivitetty tieto tulee myös JSON-muodossa.

Rivillä 2 on sama tunnistetieto, kuin alustussanomassa

Rivi 3: aikaleima, jolloin arvo on muuttunut

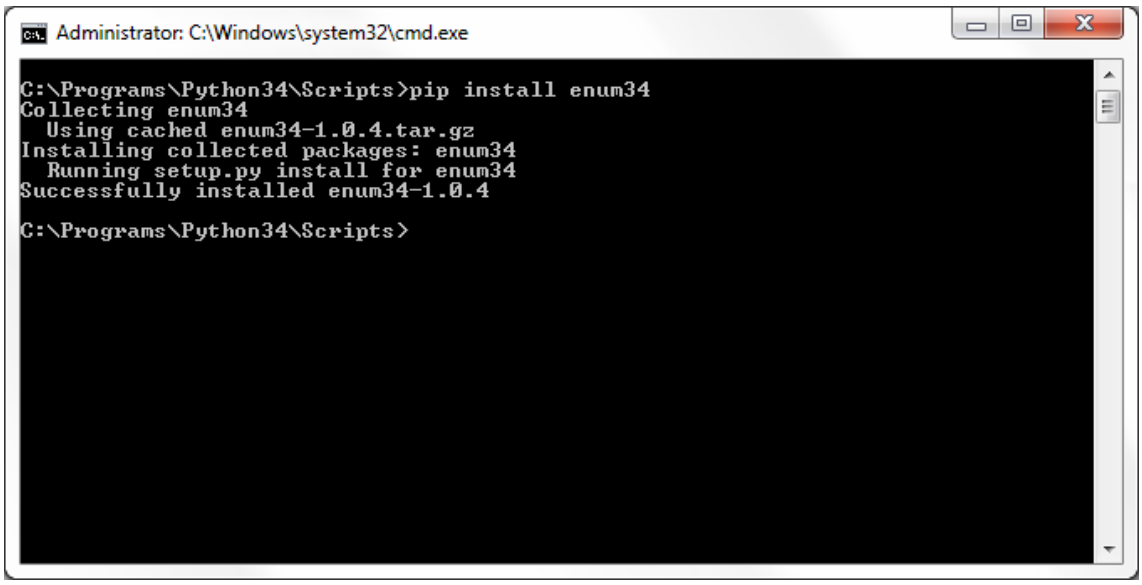
Rivi 4: Käyttöliittymässä käytettävä tyyppitunnus

Rivi 5: Uusi anturitieto.

4.6 Asennus

Jos järjestelmä asennetaan laitteeseen, jossa ei ole ennestään Python-tulkkia, tulee se ladata osoitteesta http://www.python.org/. Ohjelma on testattu niin Pythonin 2.7, kuin myös 3.4-versioissa, jotka ovat molemmat vapaasti ladattavissa yllä olevasta osoitteesta. OPC UA –kirjasto vaatii Python 2.7-versiossa enum34 ja futures-kirjastojen asennuksen, joita taas 3.4-versiossa ei tarvitse erikseen asentaa.

Kun Python-tulkki on onnistuneesti asennettu, tulee asentaa Trollius-kirjasto. Sen saa asennettua käyttämällä pip-komentoa: *pip install trollius*. Kuvassa 16 on asennettuna enum34-kirjasto käyttäen pip-komentoa. Trollius-kirjaston vaatii niin Autobahn-framework, kuten myös OPC UA -kirjasto. Autobahn toimisi myös asyncio-kirjastolla trolliuksen korvaajana, jonka takia muun muassa ohjelmakoodissa onkin nimetty kaikki näitä toimintoja vaativat muuttujat asyncio-nimellä. Sovelluksessa käytetään kumminkin trolliusta, koska OPC UA kirjasto taas vaatii nimenomaan sen.



Kuva 16. Enum34-kirjasto asennettu pip-työkalulla.

Autobahn-frameworkin asennus sujuu myös yhtä yksinkertaisesti, kuin Trollius-kirjastonkin: käyttämällä pip-komentoa. *pip install autobahn* –komento asentaa automaattisesti Autobahn-frameworkin. Muut pip-komennolla asennettavat ovat enum34 ja futuers-kirjastot. Käytä näiden asentamiseen komentoja *pip install enum34* ja *pip install futures*, niin kirjastot asentuvat automaattisesti.

OPC UA –kirjaston saa ladattua osoitteesta https://github.com/FreeOpcUa/python-opcua. Joillakin käyttöjärjestelmillä pystyy lataamaan suoraan github-osoitteista tietyllä komennolla koko paketin, mutta myös erikseen lataaminen yllä mainitusta osoitteesta on mahdollista. OPC UA –kirjaston latauksen jälkeen pakattu tiedosto tulee luonnollisesti purkaa ennen asennusta. Tiedoston purkamisen jälkeen pystytään ajamaan setup.py-niminen tiedosto komennolla *python setup.py install* ja kirjasto asentuu automaattisesti.

Jos Windows käyttöjärjestelmässä tulee virheilmoitus, ettei komento python ole tunnettu, voi *python setup.py install* –komennon ajaa myös esimerkiksi käyttäen polkua, jonne Python-tulkki on asennettu. Esimerkiksi komento *"C:\Python27\python.exe" setup.py install* hakee python.exe-tiedoston C:\Python27-kansiosta.

Yhteenvetona, tarvitaan siis seuraavat esivalmistelut ennen, kuin itse ohjelmaa pystytään ajamaan:

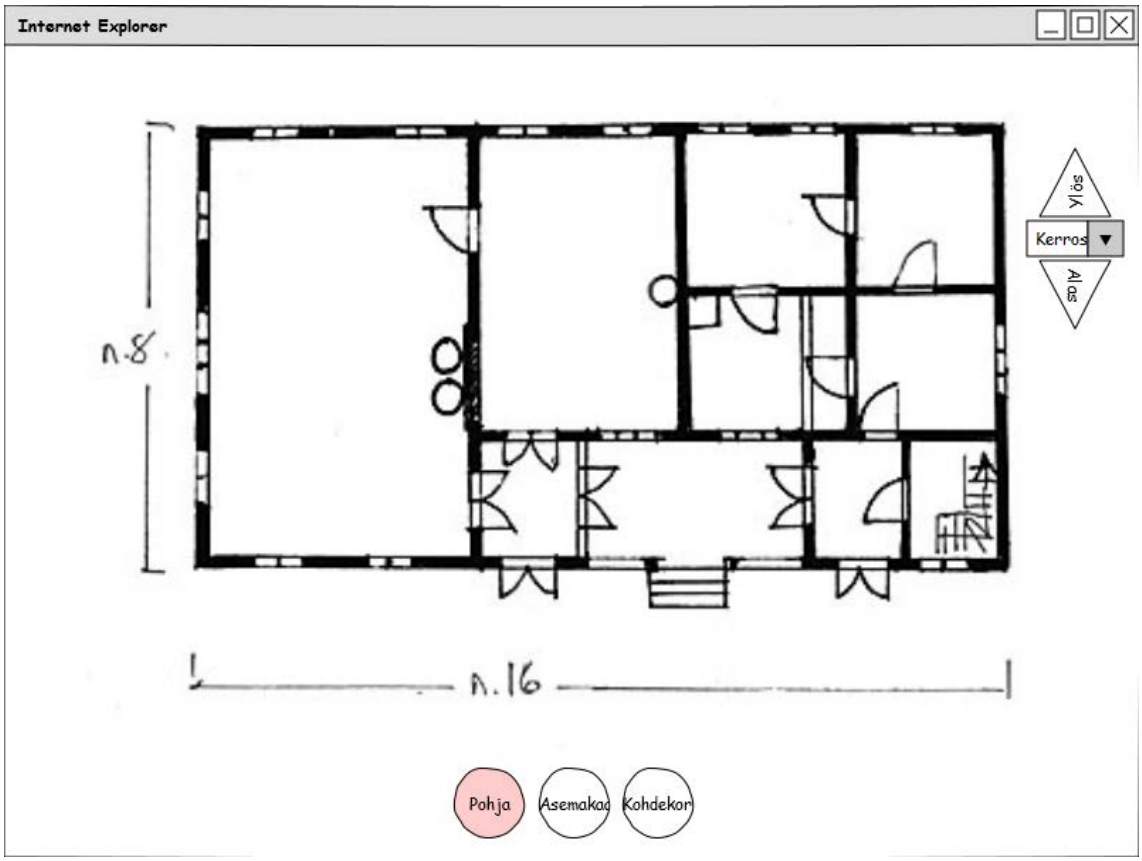
- Python-tulkki
- Autobahn-framework
- Trollius-kirjasto
- OPC UA –kirjasto
- enum34-kirjasto
- futures-kirjasto.

Kun esivalmistelut on saatu tehtyä, tulee muuntimen configuration.json-tiedostoa muokata oikeanlaiseksi ennen käyttöä. Kohdassa 4.5 Tiedonsiirto käyttäjälle, on esitetty millaisia tietoja kyseisessä tiedostossa tulee olla.

Ohjelma käynnistyy komennolla python main.py. Tämän jälkeen ohjelma ottaa yhteyden configuration.json-tiedostossa määritetylle OPC UA –palvelimelle ja luo WebSocket-palvelimen, joka toimii portissa 9000. On mahdollista, että käyttöjärjestelmän palomuuria joutuu konfiguroimaan jonkin verran ennen, kuin WebSocket-yhteyttä pystytään käyttämään.

5 Frontend-kehitys

Alkuperäinen luonnos, joka sisälsi aloituspalaverista tulleista mietteistä käyttöliittymän ulkoasusta on nähtävissä kuvasta 17. Peruspohja on pysynyt samanlaisena koko kehityksen ajan.



Kuva 17. Ensimmäinen luonnos käyttöliittymästä.

Alkuperäisestä luonnoksesta poiketen, päävalikko (kuvassa 17 alla olevat pallot) on siirretty kuvan oikeaan reunaan. Valtaosa ihmisistä on oikeakätisiä, joten valikon ollessa myös oikealla, ei käyttöliittymää käyttäessä mikään tieto ei koskaan jää käden alle. Myöskin valikon valinnat eivät ole palloja, vaan pyöristetyin kulmin olevia nelikulmioita.

Kuvassa 17 oikeassa reunassa oleva valikko on oletuksena piilotettu. Se kumminkin tulee näkyviin päävalikossa silloin, jos kerroksia on enemmän, kuin viisi. Tällöin kaikkia kerroksia ei luetella päävalikossa, vaan pudotusvalikon kautta. Lopullinen valikon sijainti on kuvassa 18.

Muuten luonnoksessa näkyvät ”Asemakaava”- ja ”Kohdekortti”-valinnat ovat edelleen käyttöliittymässä yhden näppäimen päässä.

Kuva 18. Käyttöliittymästä eräs versio, testauksen vuoksi lisätty ”Reset”-näppäin valikkoon.

5.1 HTML5

HTML5 sisältää käsitteenä kolmea erilaista ohjelmointikieltä: HTML:n, CSS:n ja JavaScriptin. HTML:llä luodaan sivun perusrunko, missä päin mikäkin elementti sijaitsee (kuva 19). HTML-kielellä ei tyypillisesti tehdä minkäänlaisia muotoiluja, vaikka se itsessään mahdollistaakin jonkinlaisia muotoiluja teksteihin. Kumminkin yksittäiset sanat tai lauseet on mahdollista, ja järkevääkin joissain tapauksissa, muun muassa lihavoida, alleviivata tai kursivoida HTML:n kautta.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Otsikko</title>
</head>

<body>
Sisältö
</body>

</html>
```

Kuva 19. Standardin mukainen HTML-sivun pohja.

CSS edustaa verkkosivuilla sivun tyyliä, eli miten HTML-elementit sivulla näkyvät. Sillä siis ei ole oikeastaan sivun toiminnollisuuden kannalta suoranaista vaikutusta, mutta ulkoasullisesti sitäkin enemmän. CSS:llä saadaan kumminkin elementtejä esimerkiksi piilotettua tai näytettyä. Sillä voidaan myös määritellä elementille muun muassa taustaväri, tekstin väri, sijainti ja reunuksen koko. Myös kaikki tekstin muotoilut, kuten lihavointi, alleviivaus ja kursivointi, voidaan toteuttaa CSS:llä.

JavaScript on skriptikieli, jota voidaan käyttää monenlaisin eri tavoin verkkosivuilla. JavaScriptille on saatavilla monenlaisia erilaisia kirjastoja, kuten grafiikkaa, animointia tai muita toimintoja helpottavia kirjastoja. Yksi suuri ja laajasti tunnettu JavaScript-kirjasto on jQuery, jota on myös tässä työssä käytetty.

jQuery-kirjasto sisältää monenlaisia toiminnollisuuksia, jotka helpottavat verkkosivujen toteutusta. Kirjastossa on toiminnollisuuksia alkaen yksinkertaisista elementin valinnoista, niiden muotoiluun ja esimerkiksi animointiin. Muun muassa animointi tapahtuu käytännössä niin, että CSS-muotoilua muutetaan jatkuvasti.

JavaScript on tämän tyyppisissä ratkaisuissa välttämättömyys, ilman sitä ei voida toteuttaa interaktiivista sisältöä. JavaScript ei vaadi käyttäjältä mitään liitännäisiä, kuten esimerkiksi Flash tai Java, joita eivät kaikki mobiiliselaimet edes tue. Käyttöliittymää ei lähtökohtaisesti käytetä mobiililaitteilla, mutta mahdollisimman laaja käyttöympäristötuki on tärkeä asia, eikä sitä halua missään tapauksessa ensimmäisenä jättää sovelluksesta pois, varsinkin jos se ei rajoita tekemistä.

5.2 WebSocket

Käyttöliittymäpuolen WebSocket-toteutus on hyvin lyhyt, vain noin 30 riviä ilman kommentteja. Toteutus sisältää tiivistettynä seuraavat toiminnot:

- Yhteyden luonti
- Yhteys on luotu
- Viesti saapunut
- Yhteys katkennut.

Myös toiminto, jos yhteyttä ei voida luoda, on toteutettu samaan tiedostoon. Tämän toiminnon toiminta tulee näkyviin vain, jos käyttäjän verkkoselain ei tue WebSocket-toimintoja. Käyttöliittymä ei lähetä viestejä WebSocket-palvelimelle päin, joten viestin lähetys –toimintoa ei siihen ole luotu, vaikka se olisi mahdollista.

Jos WebSocket-yhteyttä ei pystytä luomaan, ilmoitetaan siitä käyttäjälle viestillä: ”Ei yhteyttä: Reaaliaikainen tieto ei saatavilla! Yritetään hetken kuluttua uudelleen.”. Näin ollen kaikki oleellinen tieto, kuten rakennuksen kaikki pohjapiirrokset, asemakaavakuva sekä kohdekorttitiedot ovat kumminkin mahdollista nähdä. Reaaliaikaista tietoa, eli palohälytin- ja liiketunnistustietoja ei luonnollisestikaan tällöin voida näyttää. Käyttöliittymä ajaa kumminkin ensisijaisen asiansa pitkälti ilman näitä tietojakin.

Kun yhteys on luotu, ilmoitetaan siitä käyttäjälle viestillä ”Yhteys OK”. Käyttäjän ei tarvitse tehdä tämän jälkeen mitään, vain odottaa että muuntimelta tulee ”alustustieto”. Tämän tiedon pitäisi tulla välittömästi, kun yhteys on onnistuneesti luotu. Tällöin siirrytään myös ”Viesti saapunut” –toimintoon.

Vaikka viestit periaatteessa tulevat JSON-muodossa, ymmärtää selain sen käytännössä vain pitkänä merkkijonona. Tämä merkkijono käsitellään siten, että siitä saadaan tehtyä JavaScript-objekti. Käsittely tehdään jQuery-kirjastoa käyttäen. Näin ollen saadaan tiedot taas helposti käsiteltyä JavaScriptissä normaaleina objekteina, ja tietojen haku on helppoa toteuttaa.

Jos tämä objekti sisältää ”init”-arvon, lähetetään se ensin alustusfunktiolle. Näin saadaan kaikki perustiedot alustettua heti yhteyden luonnin jälkeen. Muussa

tapauksessa objekti lähetetään erilliselle tilojen päivitykseen tarkoitetulle funktiolle, jossa saatu tieto ensin puretaan siten, että kaikki uudet tiedot saadaan päivitettyä. Myös pohjapiirustukseen voidaan näin ollen piirtää tarpeelliset merkinnät.

5.3 Yleiskuvaus toiminnoista

JavaScript-ohjelma sisältää monenlaisia toimintoja, jotka ovat oleellisia käyttöliittymän toiminnan kannalta. Mainitsemisen arvoisia toimintoja ovat muun muassa alustus-, hälytys- ja liiketunnistusfunktiot, sekä järjestysfunktio, joka järjestää hälytykset aikajärjestykseen.

5.3.1 Alustusfunktio

Alustusfunktiota kutsutaan heti sivun latauksen yhteydessä, mutta myös silloin, kun yhteys on muuntimelle onnistuneesti luotu. Näin ollen kaikki tärkeät toiminnot – pohjakuvat, asemakaava, kohdekortti – saadaan näkyville saman tien, vaikka vastaus muuntimelta kestäisikin pidemmän aikaa, tai vaikka yhteys olisi poikki.

Toiminto luo määritellyn valikkorakenteen automaattisesti. Yhteyden ollessa muuntimelle kunnossa, automaattisesti luodaan myös kaikki HTML-elementit, sekä tallennetaan saadut arvot muistiin. Näin ollen arvoja voidaan käyttää muissa toiminnoissa helposti ja nopeasti ilman jatkuvaa verkkoyhteyttä.

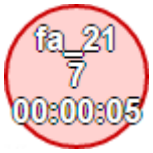
Kun muuntimelta on saatu alustustieto, kutsutaan alustusfunktiota uudestaan saadulla JSON-objektilla. Tätä tietoa käytetään muun muassa hälytys- ja liiketunnistintietojen muistiin kirjoittamiseen sekä järjestysfunktion kutsuun. Alustustiedolla saadaan myös tieto mahdollisesta ensimmäisestä lauenneesta palohälyttimestä, jolloin pystytään avaamaan kyseisen hälyttimen kerros heti näkyville.

5.3.2 Hälytysfunktio

Hälytysfunktiolla käsitellään kaikki ”firealarm”-tyyppiset arvot, eli kaikki paloilmoittimien antamat hälytykset. Tämä funktio piirtää pohjapiirustukseen määritetylle kohdalle pyöreän, punaisen pallon (kuva 19), jos hälytys on lauennut. Pallo suurenee ajan myötä ja on näin ollen visuaalisesti hyvin helposti havainnoitavissa.

Pallon sisällä näytetään hälyttimestä ylimmäisenä positiotieto, joka löytyy myös virallista kiinteistön papereista, jos se on vaadittu kyseissä kohteessa. Vaihtoehtoisesti se on itse keksitty kuvaava tieto. Joka tapauksessa tämä tieto pitää olla yksilöllinen, jotta se voidaan näyttää käyttöliittymässä.

Keskimmäisenä tietona on juokseva numerointi. Tämä numero näyttää hälyttimen laukeamisjärjestysnumeron. Näin ollen ensimmäinen huone tai tila, jossa hälytys on lauennut, on hyvin nopea löytää pohjapiirustuksesta. Kuvan 20 palohälytin on lauennut seitsemäntenä.



Kuva 20. Käyttöliittymässä näkyvä lauenneen paloilmoittimen kuvake.

Alimmaisena tietona kerrotaan kuinka kauan hälytyksen laukeamisesta on kulunut aikaa. Arvo on reaaliaikainen ja se päivittyy, vaikkei yhteyttä muuntimelle olisikaan. Toisin sanoen, jos yhteys automaatiojärjestelmään katkeaa, ei tietoa menetetä. Tätä tietoa käytetään luonnollisesti myös pallon koon määrittelyssä.

5.3.3 Liikkeentunnistusfunktio

Liiketunnistinfunktio piirtää määriteltyyn paikkaan keltaisen, pienen ympyrän, kun liikettä on havaittu. Pallon sisällä on ajastin, joka ilmoitta kuinka kauan aikaa viimeisestä havaitusta liikkeestä on. Maksimiaika on määritelty tuntiin, jonka jälkeen pallo häviää pohjapiirustuksesta. Myös pallon taustaväri muuttuu ajan pidennettyä: se muuttuu enemmän läpikuultavammaksi. Tällä on pyritty helpottamaan pohjapiirustuksesta tuoreimpien tietojen löytämistä.

Liiketunnistimen esittäminen vaatii jatkuvan verkkoyhteyden muuntimelle. Jos yhteys katkeaa kesken kaiken, jää viimeiseksi saatu arvo voimaan.

5.3.4 Järjestysfunktio

Alustuksessa saatu tieto siirtyy tämän funktion kautta. Funktio järjestää palohälytintiedot laukeamisaikaleiman mukaiseen järjestykseen.

Molemmat aikaleimat, nykyinen aika ja hälyttimen aikaleima, käsitellään UTC-aikoina. Tällä ei ole käyttäjän näkökulmasta merkitystä, mutta OPC UA käyttää UTC-aikaleimaa, jonka takia myös käyttäjän aikaleima pitää tarkistaa UTC-muodossa.

OPC UA antaa aikaleiman ”luettavammassa muodossa” ja ilman aikavyöhyke tietoa, koska aikaleima on aina UTC. Koska virallinen W3C-organisaation suosittelema muoto on YYYY-MM-DDThh:mm:ss.sssTZD [17], tulee muuntimelta saatu aikaleima käsitellä ensin oikeaan muotoon ennen varsinaista laskentaa. Aikaleimassa olevien kirjainten merkitykset ovat lueteltu alla:

- YYYY Vuosiluku, esimerkiksi 2015
- MM Kuukausi, 08
- DD Päivämäärä, 15
- hh Tunnit, 12
- mm Minuutit, 54
- ss Sekunnit, 00
- sss Mikrosekunnit, 000.

Aikaleimassa oleva T on syntaksi, jolla erotetaan päivämääräleima ja kellonaikaleima toisistaan. Yleensä tästä syntaksista kumminkin poiketaan, ja T-kirjain korvataan tyhjällä merkillä, eli välilyönnillä. Näin myös OPC UA tekee.

TZD on aikavyöhyke, joka merkitään esimerkiksi +02:00 tai -02:00. Jos leimassa on Z, tarkoittaa se UTC-aikavyöhykettä, eli +00:00. Kumminkaan -00:00 ei ole sallittu merkintätapa aikavyöhykkeessä. Tätä leimasinta käytetään niin muuntimella, kuin käyttöliittymän puolellakin, koska UTC-aika ei tunne kesä- tai talviaikaa, vaan pysyy aina samana. Esimerkiksi aikaleima voisi olla 2015-12-06T18:00+02:00, eli 6. päivä joulukuuta 2015, kello 18 aikavyöhykkeellä +2 tuntia. Sama esitettynä UTC-aikana olisi 2015-12-06T16:00+00:00 tai 2015-12-06T16:00Z. Tämä aikaleima muoto on standardisoitu: ISO 8601.

5.4 Muokattavuus

Käyttöliittymä on pyritty toteuttamaan niin, ettei sitä tarvitse juuri muokata, jos sovellus kopioidaan toiseen kiinteistöön. Kumminkin joitakin asetuksia pitää muokata, jotta käyttöliittymä toimisi uuden järjestelmän kanssa. Käyttöliittymässä on erillinen konfigurointi-tiedosto, jonne tarvittavia parametreja tulee antaa JSON-muodossa. Muokattavia parametreja ovat:

- WebSocket-osoite
- Palohälytinelementin suurin koko
- Rakennukset ja niiden kerrokset sekä pohjapiirrosten kohdepolku, kuvan leveys- ja pituus-tiedot pikseleinä.

Kaikki muut tiedot, paitsi WebSocket-osoite, olisi mahdollista saada myös muuntimen alustustiedoissa. Tätä ei kumminkaan ole toteutettu, koska jos yhteyttä ei olekaan, ei joitakin oleellisia tietoja tiedettäisi. Tällaisessa tilanteessa ei verkkosivulla näkyisi pahimmassa tapauksessa mitään.

Asemakaavakuva sekä kohdekortti on tallennettava määrättyyn kansioon, määrätyllä nimellä, jolloin niiden määrittelyihin ei tarvitse ollenkaan koskea. Molemmat ovat vain isoja kuvia, joten niiden tallennus ja ylläpito on hyvin helppoa eikä vaadi minkäänlaista verkkosivutekniikoiden tarkempaa tietämystä tai ohjelmointitaitojen opiskelua. Tietojen ylläpitäjän tulee vain tietää missä kansiossa nämä oleelliset kuvat sijaitsevat ja kopioida uudet vanhojen tilalle, kun päivitys tarvetta ilmenee.

6 Yhteenveto

Sovellus saatiin toteutettua selkeäksi ja nopeaksi käyttää, niin isossa palvelimessa, kuin myös kevyessä varustetussa laitteessa. Käyttöliittymä avautuu miellyttävän nopeasti, vaikka reaaliaikaisentiedon yhteyden luonnissa saattaa välillä mennä hieman pidempään.

Työssä käytettyjen tekniikoiden valintaperusteena on käytetty pitkän elinkaaren ohjelmistoratkaisuja: OPC:lla on jo vankka jalansija useissa paikoissa, Web-pohjaiset käyttöliittymät tulevat jatkamaan yleistymistään IoT-ratkaisujen lisääntymisen myötä ja Python käyttäjiä näyttäisi keskustelupalstojen perusteella olevan kokoajan enemmän. Toistaiseksi ei ole näkyvissä syytä miksi käyttöliittymä ei toimisi esimerkiksi 20 vuoden päästä.

Toisaalta, käyttöliittymä itsessään ei sisällä toistaiseksi minkäänlaista salausta tai autentikointia, joten sen kytkeminen suoraan julkiseen Internettiin voi olla vakava riski. Varsinkin, jos liitettynä ovat myös liikkeentunnistintiedot, jolloin osaava rikollinen pääsisi myös käsiksi kaikkiin tietoihin. Tämän vuoksi laajassa käytössä OPC UA – tiedonsiirto pitää olla vähintään salattuna ja käyttöliittymään pääseminen vain paikallisnäytöstä ja esimerkiksi pelastusviranomaisten kautta.

Kyseessä oli pilottikohde, eikä yllämainittuja tietoturva-asioita tarvinnut erikseen ottaa huomioon. Nämä asiat on hyvä tiedostaa, jos järjestelmää otetaan laajaan käyttöön. Kaikki asetetut tavoitteet pilottijärjestelmälle onnistuttiin hyvin täyttämään.

6.1 Jatkokehitys ja tulevaisuus

Muunninta tullaan kehittämään niin muokattavuudeltaan, kuin yleiseltä toiminnallisuudeltaan monipuolisemmaksi. Lisäominaisuuksiksi lisätään laajempi protokollatuki . Tällöin muunninta voidaan käyttää myös muihin tarkoituksiin, kuin tämän insinöörityön sovellukseen.

MQTT vaikuttaa mielenkiintoiselta protokollalta jatkokehityksen kannalta ja se perustuu julkaisu/kuuntelu-periaatteeseen. Kun MQTT broker vastaanottaa uuden viestin, välitetään se kaikille kyseistä otsikkoa kuunteleville käyttäjille. Tämä voisi tarkoittaa

tämän järjestelmän kannalta sitä, että otsikko voisi olla esimerkiksi yksinkertaistettuna muotoa maakunta/kaupunki/katu/talo, jolloin yhteyden luonti kohdekiinteistöön olisi erittäin suoraviivaista. Kaikki järjestelmässä olevat kiinteistöt voisivat näin ollen olla yhteydessä MQTT brokeriin ja uusien lisääminen olisi vain uusi otsikko. Minkäänlaista konfigurointia ei palvelin puolelle tarvitsisi tehdä.

Käyttöliittymän Asemakaava-kuvassa olisi mielenkiintoista käyttää myös muun muassa Ilmatieteenlaitoksen Avoin data –palvelua. Kuvassa voisi näkyä esimerkiksi tuulen suunta ja voimakkuus, sateen määrää viime päivinä ja tällä hetkellä sekä muuta vastaavaa, mikä mahdollisesti vaikuttaa pelastustehtävään. Tietenkin niin, ettei pelastusviranomaisille kaikkein oleellisin tieto jäisi näiden lisätoimintojen varjoon.

Lähteet

1. World Fire Statistics, https://www.genevaassociation.org/media/186303/ga2011-fire27.pdf, luettu 26.9.2015

2. Kääriäinen Pasi, ja Pelkonen Sami, keskustelu 23.6.2015

3. Jussi Piispanen, Toimialapäällikkö, Beckhoff Automation Oy, keskustelu 28.9.2015

4. Beckhoff Automation Oy kotisivut: http://www.beckhoff.fi, luettu 27.6.2015

5. MQTT, http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger, luettu 15.8.2015

6. OPC Foundation, https://opcfoundation.org/, luettu 5.9.2015

7. OPC History, http://www.opcconnect.com/history.php, luettu 5.9.2015

8. Stefan Hoppe esitys, https://opcfoundation.org/wp-content/uploads/2014/08/2_Industrial-Revolution-2014-StefanHoppe.pdf, luettu 26.9.2015

9. SliderShare, http://www.slideshare.net/dev_bhattacharya/internet-of-things-applications-covering-industrial-domain, luettu 26.9.2015

10. Gizmodo, http://gizmodo.com/5960831/23-ancient-web-sites-that-are-still-alive, luettu 28.9.2015

11. Computer Language Benchmark Game, http://benchmarksgame.alioth.debian.org/u32q/benchmark.php?test=all&lang=python3&lang2=gpp&data=u32q, luettu 29.8.2015

12. OPC – Open Process Control, Jari Savolainen, https://wiki.metropolia.fi/display/koneautomaatio/OPC+-+Open+Process+Control, luettu 19.7.2015

13. National Instruments -kotisivut, http://www.ni.com/white-paper/13843/en/, luettu 5.9.2015

14. WebSocket.org-kotisivut, https://www.websocket.org/quantum.html, luettu 5.9.2015

15. Python Software Foundation kotisivut: https://docs.python.org/2/faq/general.html#general-python-faq, luettu 19.7.2015

16. JSON standardi, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, luettu 15.8.2015

17. Date and Time Formats, W3C: http://www.w3.org/TR/NOTE-datetime, luettu 27.7.2015

Liite 1. Muistilista paloilmoitin näkymään

Alla on kopio Pasi Kääriäiseltä saadusta muistilistasta koskien käyttöliittymää.

Pohjapiirros jossa näkyy:

- hälyttänyt / hälyttäneet ilmaisimet kyseisessä tilassa ja ilmaisimen numero
- Huonejako josta selviää myös palo-osaston rajat (kolmipistekatkoviiva)
- Savunpoistoluukut

Asemakaavapiirros josta käy ilmi:

- ilmansuunta ja lähikadut
- Paloilmoitinkeskus, sprinklerikeskus ja palokunnan sprikler syöttö, savunpoistokeskus, putkilukko
- Palokunnan hyökkäystiet eli ovet, joista mahdollisuus tulla sisälle
- Katolle johtavat tikkaat ja mahdollinen kuivanousu putki (palokunnan veden syöttöä varten katolle)
- Ilmanvaihdon, kaasun, veden ja sähkön pääsulkujen sijainnit
- Palavien nesteiden ja muiden vaarallisten aineiden varastojen sijainnit
- Väestönsuoja
- Ilmanvaihtokonehuone
- Kokoontumispaikka
- Portti ja ajoreitit

Muuta muistettavaa

- Ilmaisimien ja ryhmien irtikytkentä miten voidaan hoitaa näytöllä
- Palokellojen irtikytkentä
- Hälytyksen kuittaus eli laitteiston saattaminen normaaliin tilaan
- Voidaanko järjestelmää testata esim. kuukausittain antamalla koehälytys ja nähdään, että laitteisto toimii

Liite 2. Kohdekortti

Kiinteistön kohdekortti, joka halutaan myös paloilmoitinjärjestelmän käyttöliittymään sähköisessä muodossa.

Vantaan kaupunki
Keski-Uudenmaan pelastuslaitos
Mellersta Nylands räddningverk

KOHDEKORTTI Numero 00000

Täyttäjä	Puhelin	Päivämäärä
Pekka Palomestari	112112	11.2.2006

Perusyksiköt		Erikoisyksiköt	
Osoite: Paloasemankatu 12		Kaupunginosa: Palokylä	Karttaruutu: 0889
Nimi: Kiinteistö Oy Pelastuslaitos		Keskuskojeen sijainti: A portaan tuulikaapissa	
Sprinklerikeskus ja syötöt: Paloasemankadun ja Palotarkastajankadun risteyksessä			
Sammutussuunnitelman sijainti: Paloilmoittimen yhteydessä A-portaassa		Avaimet: Putkilukko A-portaan oikealla puolella seinässä	
Paloposti: Paloasemankatu 6 kohdalla, Palotarkastajankatu 3 kohdalla			
PÄÄSULUT	Sähkö: Kellarissa, käynti A-portaasta	Vesi: Palotark.kadun puolella	Ilmastointi: A-portaan tuulikaapissa
Savunpoisto: Painovoimainen, laukaisu A portaan tuulikaapissa,			

Puhelin (keskus): 09-11211	Paloilmoituslaitoksen hoitaja: Paavo Palomies	Puhelin työhön: 09-112113	Puhelin kotiin: 09-1234567
Yhdyshenkilö I: Seppo Savusukeltaja		Puhelin työhön: 09-112114	Puhelin kotiin: 09-2345678
Yhdyshenkilö II: Esko Ensiauttaja		Puhelin työhön: 09-112191	Puhelin kotiin: 09-3456789
Yhdyshenkilö III: Varma Varushoitaja		Puhelin työhön: 09-112231	Puhelin kotiin: 09-4567891
Vartiointiliike: Vartiointiliike Palokärki		Klo: 21-07	Puhelin: 09-3214568

Kiinteistön käyttötarkoitus: Paloasema	KERROKSIA: 3	Maan päällä: 2	Maan alla: 1

Lämmitys: ☑ kaukolämpö ☐ öljykeskus ☐ lämminilmakehitin ☐ sähkö ☐ muu, mikä	
Ilmastointikonehuone: ☑ kellarissa ☐ ullakolla ☐ katolla	Katkaisimen sijainti: A portaan tuulikaapissa
Hälytysjärjestelmä: ☑ lämpö ☑ savu ☑ painike ☑ muu	Sammutusjärjestelmä: ☑ sprinkleri ☐ CO2/Haloni ☐ vaahto ☐ muu

Kiinteistön vakuutusyhtiö: Vahinkoyhtiö täystuho	Irtaimiston vakuutusyhtiö: sama	
Kiinteistön omistaja: Palokäen kaupunki	Osoite: Kaupungintie 1	Puhelin: 09-12321
Palotark.piiri	Nuohouspiiri	Postiosoite: 12345 Palokäki

Kerros: Ulkorakennus	Vaaralliset aineet tai muut riskitekijät: nestekaasua (max 150 kg)	Katsastuspvm.: 01.01.2001

Lisätiedot: Rakennus ympärivuorokautisessa käytössä.