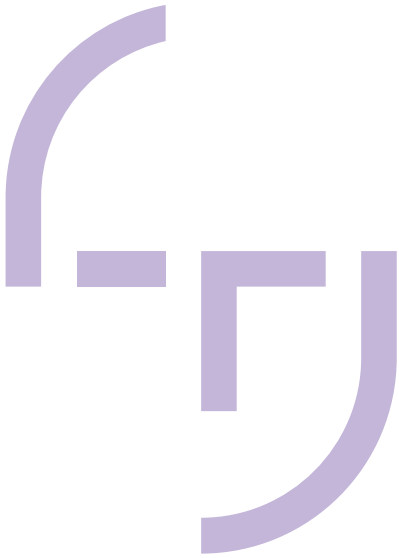
Tampereen ammattikorkeakoulu



Kiinteistöautomaatiojärjestelmän hälytyksen luominen

Tuomas Tuominen

OPINNÄYTETYÖ
Toukokuu 2020

Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikka
Automaatiotekniikka

TUOMINEN, TUOMAS:
Kiinteistöautomaatiojärjestelmän hälytyksen luominen

Opinnäytetyö 33 sivua, joista liitteitä 0 sivua
Toukokuu 2020

Assemblin Oy:lle toteutettavan opinnäytetyön tavoitteena on kehittää ohjelma rakennusautomaation hälytysviestien välittämiseksi mobiilisovellukseen, joka edistäisi yrityksen käyttämän rakennusautomaation hallintajärjestelmän toiminnallisuutta. Tarve ohjelman kehittämiselle syntyi uuden rakennusautomaation hallintajärjestelmän version myötä, jossa uutena ominaisuutena oli mobiilisovellus hälytyksille. Mobiilisovelluksen avulla on mahdollista tavoittaa kiinteistöjen automaatiosta vastaavat henkilöt nopeammin, mikä nopeuttaa vikatilanteisiin reagoimista. Kiinteistöjen vikatilanteisiin nopeamman puuttumisen ansiosta ongelmien vaikutuksia on mahdollista vähentää.

Kehitettävän ohjelman tulee muuttaa etäkohteista vastaanotettavat tekstimuotoiset hälytysviestit ohjelmallisesti vertailtavan datan muotoon, jotta ohjelman avulla olisi mahdollista analysoida hälytykset automaattisesti. Ohjelman on tarkoitus välittää hälytysviestin tiedot mobiilisovellukseen, jos ennalta määritettävät kriteerit kriittisen hälytyksen tunnusmerkeistä täyttyvät. Tämän lisäksi viestien vastaanoton jälkeen kuitataan lähettäneelle kohteelle hälytykset vastaanotetuiksi, jotta vältettäisiin viestien toistuva lähettäminen.

Ohjelman avulla voidaan hälytysviesteihin liittyvien toimintojen lisäksi valvoa keskusvalvomon ja etäkohteiden välisen tiedonsiirtoyhteyden vakautta. Yhteyden vakautta valvotaan viestien tapahtuman ja vastaanoton aikaleimoja vertailemalla. Tiedonsiirtoyhteyden valvonnalla ohjelma edistää kohteiden ylläpidon laatua, kun tiedonsiirtoyhteyden laadussa tapahtuvat muutokset voidaan havaita nopeammin.

Asiasanat: hälytys, kiinteistöautomaatio, Python, tiedonsiirto

ABSTRACT

Tampere University of Applied Sciences
Degree programme in Electrical and Automation Engineering
Automation Engineering

TUOMINEN, TUOMAS:
Creating alarm for building automation system

Bachelor's thesis 33 pages, appendices 0 pages
May 2020

Thesis implemented for Assemblin Oy intents to create program for sending alarm messages of building automation into a mobile application, which would enhance performance of the building management system software used by the company. Need for developing the program formed from the new version of building management systems which introduced as a new feature the mobile application for alarms. With the mobile application it is possible to reach personel in charge of properties automation even faster, which improves the time elapsed in reaction to occurrence of fault. By reacting faster in to faults occurred in properties, impact of the flaws can be reduced.

Program developed will modify plain text formatted alarm messages received from remote location properties into data format that is comparable by software to enable software automatically performing analysis on received messages Programs intent is to send information of alarm messages to the mobile application if predefined criteria for critical alarm are fulfilled. In addition, after receiving message program will sends a response to the remote location as a confirmation of received message which informs sender to not send that specific message repeatedly.

Besides operations involving alarm messages, stability of data transfer connection between remote location property and central control room can be monitored with the program. Supervising of connection stability is executed by comparing timestamps of event and when the message was received. By monitoring network connection stability program enhances quality of properties maintenance, when changes in network stability can be detected faster.

Key words: alarm, building automation, Python, data transfer

SISÄLLYS

LYHENTEET JA TERMIT

API	Ohjelmointirajapinta (Application Programming Interface)
CSV	Pilkulla erotettujen arvojen tiedostomuoto (Comma Separated Value)
CPU	Prosessori (Central Processing Unit)
GSM	Matkapuhelinjärjestelmä (Global System for Mobile)
HMI	Käyttöliittymä (Human Machine Interface)
HTTP	Hypertekstin siirtoprotokolla (Hypertext Transfer Protocol)
HTTPS	Turvallinen hypertekstin siirtoprotokolla (Hypertext Transfer Protocol Secure
HVAC	Lämmitys, Vesi ja ilmanvaihto (Heating, Ventilation and Air Conditioning)
IETF	Internet tekniikan työryhmä (Internet Engineering Task Force)
IP	Verkkokerroksen protokolla (Internet Protocol)
I/O	Sisääntulo/ulostulo (Input/Output)
IT	Tietotekniikka (Information Technology)
LVI	Lämmitys-, vesijohto- ja ilmanvaihtotekniikka
pip	Python-pakettien manageröinti (Pip Installs Packages)
Python	Ohjelmointikieli
PC	Tietokoneen yleisnimitys (Personal Computer)
PNS	Push ilmoitus palvelu (Push Notification Service)
RAU	Rakennusautomaatio
RFC	Kommenttien pyyntö (Request For Comments)
TCP	Tietoliikenne protokolla (Transmission Control Protocol)
UTF	Unicoden muunnos formaatti (Unicode Transformation Format)

1 JOHDANTO

Opinnäytetyö on toteutettu Assemblin Oy:n toimeksiannosta. Assemblin tiedosti käyttämässään rakennusten hallintajärjestelmässään mahdollisuuden välittää työntekijöiden mobiililaitteisiin hälytyksiä järjestelmän mobiilisovelluksen välityksellä. Hälytyksiä käsittelevän mobiilisovelluksen käytön mahdollisti uusi hallintajärjestelmän versio, johon mobiilisovellus tuotiin uutena ominaisuutena. Opinnäytetyön tarkoituksena on toteuttaa ohjelma hälytysten välittämiseksi mobiilisovellukseen. Kehitettävän ohjelman tulee vastaanottaa hälytyksiä hallintajärjestelmässä olevista kiinteistöistä. Vastaanotettavat hälytysviestit käsitellään, jotta viestejä voidaan analysoida ohjelmallisesti. Mobiilisovellukseen lähetettävät viestit määritetään viestien analysoinnin perusteella, kun kriteerit välitettävälle hälytykselle toteutuvat. Hälytysten vastaanottamisen jälkeen hälytyksen lähettäneelle etäkohteelle kuitataan hälytykset vastaanotetuiksi.

Tämän lisäksi toteutettavan ohjelman avulla voidaan tarkkailla lähettäjän ja hallintajärjestelmän välistä verkkoyhteyttä, mikä toteutuu hälytysviestin tapahtuman ja vastaanoton aikaleimaa vertailemalla. Verkkoyhteyden laadun valvonta edistää järjestelmän toiminnallisuuden monipuolisuutta tunnistamalla heikompia etä- ja keskusvalvomoiden välisiä yhteyksiä. Verkkoyhteyden valvonnan avulla edistetään kohteiden ylläpidon laatua, kun heikomman verkkoyhteyden etäkohteet ovat paremmin tunnistettavissa niiden verkkoyhteyden parantamiseksi voidaan ryhtyä toimenpiteisiin. Koska luotavan ohjelman tulee kommunikoida tiedonsiirrossa internetin välityksellä, opinnäytetyön ohjelmointikieleksi valitaan Python.

Jos järjestelmän havaitsemat viat välitetään ainoastaan operaattorin tietokoneelle, virhetilanteiden vasteaika saattaa kasvaa kohtuuttomaksi työajan ulkopuolella. Tästä syystä järjestelmän kriittisten viestien tehokkuutta parantaa nopea kiinteistöjen automaatiosta vastaavien henkilöiden tavoittaminen, joka tänä päivänä toteutuu mobiililaitteiden avulla.

2 ASSEMBLIN OY

Assemblin toimittaa kiinteistöautomaation palveluita ja ratkaisuja, joiden avulla edistetään kiinteistöjen kulutuksen optimointia rakennusten ilmastointiin, veteen ja energiaan liittyvissä järjestelmissä. Yhtiön toiminta sijoittuu enimmäkseen Ruotsiin, mutta vaikuttaa myös Norjassa ja Suomessa. Suomessa yrityksellä on toimipisteitä Helsingissä, Turussa, Oulussa, Hyvinkäällä ja Tampereella. Yrityksellä on osaamista ja kokemusta eri tyyppisistä ja kokoisista kiinteistöistä, jonka lisäksi henkilökuntaa ja asentajia koulutetaan jatkuvasti ammattitaidon ja osaamisen ylläpitämiseksi sekä kehittämiseksi. Assemblinilla on SETI Oy:n tunnustama RAU-pätevyys, joka myönnetään rakennusautomaatiojärjestelmiä asentaville tai ylläpitäville yrityksille niiden täyttäessä vaaditut RAU-urakoitsijakriteerit. Rakennusautomaatiourakoiden toteutuksessa huomioidaan asiakkaiden tarpeet ja toiveet, joissa tärkeitä kriteerejä ovat järjestelmien käytettävyys sekä tarpeellisuus. Asiakkaiden kohteisiin voidaan sopimusten mukaan lisätä automaatiota tai kohteiden automaatiojärjestelmä voidaan myös uudistaa täysin. (Assemblin rakennusautomaatiourakointi)

Vaihtoehtoisesti jo olemassa olevan automaatiojärjestelmän säätö voidaan analysoida ja konfiguroida uudelleen, jotta kulutuksen vähentäminen toteutuisi. Asiakkaille toteutettujen projektien energian kulutusta valvotaan tasaisin väliajoin, jotta projekteissa tavoitteiksi asetetut kulutusten vähennykset toteutuisivat. Asennettuja automaatiojärjestelmiä tulee testata ja valvoa toiminnan varmistamiseksi, sillä automaattisia toimintoja sisältävät järjestelmätkin edellyttävät valvontaa tavoitellun toiminnan saavuttamiseksi. Yrityksen tarjoamien etähallintapalveluiden avulla on mahdollista saavuttaa kustannussäästöjä ilman lisäinvestointejakin. Etähallintapalvelu toteutuu etäyhteyden avulla kiinteistöjen rakennusautomaatiojärjestelmiin, joista saatavissa olevan mittausten historiadatan avulla on määritettävissä rakennusten kulutuksen profiilia. Mittausten historiadatan analysoimisella on mahdollista tunnistaa esimerkiksi sähkön kulutuksen ylimääräisiä kuormia tai vuotavia vesikalusteita sekä toimimattomia ohjauksia. Analysoinnin perusteella asetusarvoja tai säätökäyriä voidaan säätää energiatehokkaamman ohjauksen saavuttamiseksi. (Assemblin energiatehokkuus)

Rakennusten etähallintapalvelu edellyttää tietoliikenneyhteyden kiinteistön rakennusautomaatiojärjestelmän sekä energiatehokkuus- ja elinkaaripalveluiden etähallintapalvelun välillä. Tämä tietoliikenneyhteys mahdollistaa energia-asiantuntijoiden pääsyn rakennusautomaatiojärjestelmään ja rakennuksen taloteknisten järjestelmien hallintaan. Etähallintapalvelun etuna on sen riippumattomuus automaatiojärjestelmästä, koska etäyhteys voidaan rakentaa mihin tahansa automaatiojärjestelmään asiakkaan toiveesta. Etähallintapalvelun avulla varmistetaan energiatehokkuuden lisäksi laadukkaat sisäilman-olosuhteet. Assemblinin arvioiden mukaan etähallintapalveluiden avulla on saavutettavissa 5 – 20 %:n säästöt kulutukseen ja pienillä investoinneilla yli 30% säästötkin ovat saavutettavissa. (Assemblin etähallinta)

3 KIINTEISTÖAUTOMAATIO

Kiinteistöautomaatiolla ohjataan kiinteistön teknisiä toimintoja ilman ihmisen jatkuvaa valvontaa. Tavoitteena on edistää kiinteistön toimintaa jatkuvan valvonnan ja ohjauksen avulla. Kiinteistöautomaation avulla saavutetaan parempi kiinteistön säätö ja säästöjä kulutuksessa. Monista kiinteistöautomaation eduista huolimatta se on ainoastaan työkalu, joka vaatii taitavan käyttäjän ja järjestelmän ylläpitämisen sekä huollon toimiakseen. (Suomäki, Vepsäläinen 2013, 11)

3.1 Historia

Ensimmäiset kiinteistöautomaation yritykset takautuvat 1950-luvulle asti, jolloin ensimmäisiä lämmön-, vesijohto ja ilmanvaihtotekniikan eli LVI-automaation yrityksiä syntyi. Tuolloin säätimet olivat analogisia ja hälytykset välitettiin pienjännitekaapeloinneilla. 1960-luvun alussa transistoritekniikka alkoi yleistyä kaupallisiin sovelluksiin, mikä mahdollisti useampi portaisten säätöjen toteuttamisen sähköisesti. Ensimmäiset talovalvontajärjestelmät kehitettiin 1970-luvulla tapahtuneen öljykriisin johdosta, koska öljyn hinta kaksinkertaistui luoden tarpeen ohjata rakennusten toimintoja kattavammin energian säästön kannalta. Vuosikymmenen lopussa toteutettiin ensimmäiset keskitetyt talovalvomot, joissa useampia taloja yhdistettiin valvomoon. (ST Käsikirja, 2018, 13-14)

Digitaalisia säätimiä alettiin integroimaan valvontajärjestelmiin 1980-luvulla, mikä mahdollisti säätimien parametrien asettelun valvomosta. Tiedonsiirrossa käytettiin kiinteitä puhelinlinjoja yhdistämään taloja yhteen talovalvomoon. Tämä mahdollisti tärkeimpien hälytysten välittämisen, koska valvomoiden ja alakeskusten oli mahdollista avata puhelinlinja tarvittaessa. Alakeskuksia alettiin valmistamaan itsenäisiksi 1990-luvun alussa prosessorin eli CPU:n avulla. Itsenäinen alakeskus kykenee vastaamaan kohteensa ohjauksesta, kun valvomo vastasi seurannasta yleisesti. Laitevalmistajien kehitellessä laitteita ja järjestelmiään syntyi useita suljettuja järjestelmiä, jotka eivät kyenneet kommunikoimaan toisten laitevalmistajien laitteiden kanssa. Laitevalmistajien kehittelemien järjestelmien välille yritettiin kehittää rajapintoja sekä laiteajureita, mutta niiden toi-

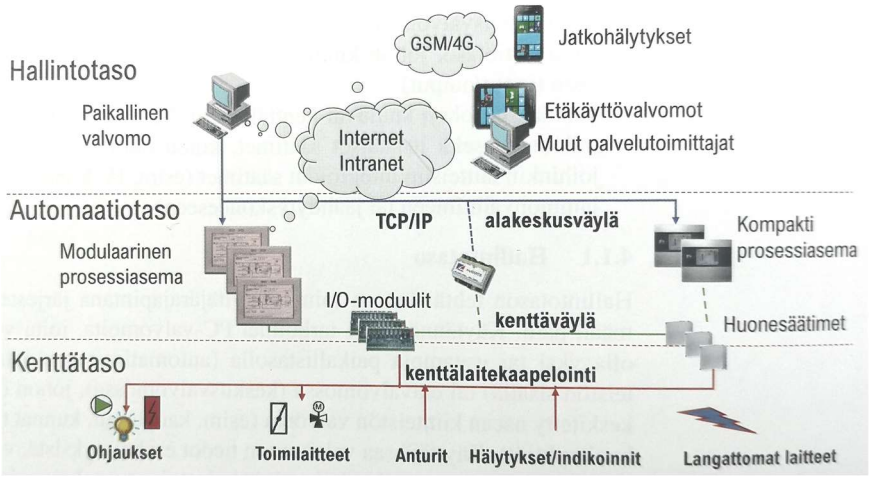
minnallisuus oli kaikesta huolimatta puutteellista. Rajapinta ja laiteajuri ratkaisujen kehittäminen oli myös kallista ja aikaa vaativaa, mikä johti ideaan avoimista ja hajautetuista järjestelmistä. (ST Käsikirja, 2018, 15-16)

1990-luvulla ohjelmisto- ja puolijohdetekniikka kehittyivät vakaannuttaen rakentamisessa perusmallina toteutetun kolmitasoisen hierarkian, joka koostuu valvomo-, alakeskus- ja huonelaitetasosta. Kolmitasoista hierarkiaa käytetään vielä tänäkin päivänä. Saman vuosikymmenen aikana hälytysten siirto GSM-verkon välityksellä yleistyi. Järjestelmien päivystystietoja välitettiin aluksi ryhmähälytyksinä, mutta tästä siirryttiin vuosikymmenen puolivälin jälkeen GSM-tekstiviesti formaattiin. Kun internet yleistyi 2000-luvun alussa, etävalvonta alkoi yleistymään. Internetin avulla mahdollisuus eri järjestelmien käyttämiseen kasvoi, joka laajensi käytettävän järjestelmän valikoimaa. (ST Käsikirja, 2018, 16-17)

3.2 Rakenne

Kiinteistöautomaation kehitystä on edistänyt internetin, tietotekniikan sekä elektroniikan komponenttien kehitys. Myös jatkuva palveluiden ja tuotteiden kehitys on edistänyt kiinteistöautomaatiolla saavutettua viihtyvyyttä ja energian kulutuksen säästöjä. Koska kiinteistöautomaation järjestelmät pohjautuvat yhä enemmän IT-teknologiaan, niiden hinta on säilynyt edullisena suorituskyvyn ja ominaisuuksien kasvusta huolimatta. (ST Käsikirja, 2018, 59)

Keskitettyjen kiinteistöautomaatiojärjestelmien aikaan muodostui käsitys kolmen päätason hierarkiasta (kuva 1), joka koostuu hallintatasosta, automaatiotasosta sekä kenttätasosta. Hallintatasoon lasketaan paikallisvalvomot ja etävalvomot. Automaatiotasolla on alakeskukset ja Input/Output eli I/O-moduulit. Kenttätasossa on kenttälaitteet eli anturit, toimilaitteet ja itsenäiset säätimet. (ST Käsikirja, 2018, 59)



KUVA 1. Kiinteistöautomaation yleinen rakenne (ST Käsikirja, 60)

Hierarkian hallintataso toimii käyttäjärajapintana järjestelmään tietokone- eli PC-valvomoiden kautta. PC-valvomo voi olla paikallinen tai etäältä toimiva etävalvomo, josta käytetään termiä keskusvalvomo. Keskusvalvomon etuna on mahdollisuus tuoda useiden kiinteistöjen valvonta yhteen keskusvalvomoon, josta on pääsy kaikkiin siihen kytkettyihin valvomoihin. Käyttäjä voi valvomosta lukea tietoa järjestelmään tuoduista hälytyksistä, tarkastella graafisia prosessikuvia sekä tehdä muutoksia ohjauksen asetusarvoihin ja aikaohjelmiin. Tiedonsiirto hallintatasolla toteutetaan paikallisesti Ethernet-väylällä ja etävalvontaan laajakaistatekniikkaan pohjautuvilla internet yhteyksillä. TCP-IP-protokollaan pohjautuvat paikallisverkot eli LAN -verkot sekä internetyhteydet tarjoavat luotettavan ja nopean tiedonsiirtoyhteyden. Avoin tiedonsiirto tarjoaa mahdollisuuksia etähallinnan kannalta, mutta kasvattaa samalla tietoturvan merkitystä. (ST Käsikirja, 2018, 59-60)

Keskitetyissä järjestelmissä automaatiotasolla toimivat itsenäiset alakeskukset ja niihin liitetyt I/O-moduulit. Alakeskus vastaa prosessien ohjauksesta ohjaamalla I/O-pisteiden välityksellä kohteen kenttälaitteita. Kuten hallinta tason kommunikointi, automaatiotason kommunikointi toteutuu LAN-verkon ja TCP-IP -protokollan välityksellä. (ST Käsikirja, 2018, 60-61)

Kenttätasolla anturit välittävät järjestelmään reaaliaikaista dataa prosessien tilasta. Anturien mittaustietojen avulla alakeskukseen ohjelmoidut ohjelmat vertaavat mittaustietoja asetusarvoihin, jonka jälkeen toimilaitteita ohjataan asetusarvojen toteuttamiseksi. Alakeskukselta voidaan väylän avulla kommunikoida

myös hajautettuihin I/O-moduuleihin. Puhaltimia ja pumppuja ohjataan usein erillisellä logiikalla taajuusmuuttajista, jotka kommunikoivat alakeskusten kanssa. Hajautetun I/O:n ja itsenäisten säätimien kommunikointi alakeskuksille toteutuu yleensä kenttäväylän kautta. Kenttäväylätekniikassa on useampia standardeja mm. ModBus, KNX ja BACnet. Eri projekteihin valittava väylä riippuu sovelluksesta, valituista laitteista ja urakoitsijan tarjonnasta. (ST Käsikirja, 2018, 61)

Nykyään rakennusautomaatiojärjestelmiin voidaan integroida useampia taloteknisiä järjestelmiä, kuten turvallisuus-, kulunvalvonta- ja varoitinjärjestelmiä. Useampien järjestelmien yhteen integroiminen ja ylläpito vaatii enemmän osaamista kokonaisuuden hallinnasta. Nämä integroitavat järjestelmät voidaan liittää valvomoon, jolloin käyttäjä voi järjestelmän käyttöliittymästä hallita niitä. Integroimalla järjestelmiä rakennusautomaatioon investointikustannukset vähenevät, koska integroitavat järjestelmät voidaan liittää rakennusautomaatioon. Lisäksi kulunvalvontaa mittaavien järjestelmien avulla on mahdollista analysoida tarkemmin kiinteistöjen kulutusta ja tuoda esille turhaa kulutusta aiheuttavia kiinteistön osa-alueita. (ST Käsikirja, 2018, 60-31)

3.3 Hälytykset

Rakennusautomaatiossa hälytykset toteutetaan ohjelmallisesti ja hälytyksen ehtojen toteutuessa järjestelmä hälyttää ilmenneestä viasta. Hälytyksestä ilmoittavassa viestissä osoitetaan kohteen osoite tai pistepositio. Hälytykset jaetaan kiireellisyysluokkiin ja kiireelliset hälytykset priorisoidaan välitettäviksi huoltohenkilöstölle, jotta niihin pystytään reagoimaan niiden edellyttämällä nopeudella. Kiireelliset hälytykset ovat henkeä tai terveyttä uhkaavat vaaratilanteet ja huomattavaa taloudellista vahinkoa aiheuttavat vikatilanteet. Vähemmän kiireellisiin hälytyksiin voidaan sen sijaan reagoida tilannekohtaisesti pidemmillä viiveillä. Hälytyksiin reagoitavan toimintatavan valinta helpottuu, kun hälytykset priorisoidaan huolellisesti perusteltavin tavoin. Useiden kohteiden etävalvonnasta vastaavassa keskusvalvomossa esiintyy suuria määriä hälytyksiä päivittäin. Kriittisiin hälytyksiin reagointia vaikeuttaa keskusvalvomoon saapuva suuri määrä

hälytyksiä, koska henkilöstön reagointi ilmeneviin hälytyksiin puutuu suuresta hälytysten määrästä johtuen. (ST Käsikirja, 2018, 222-223)

Internet tekniikan työryhmä IETF on ehdottanut RFC standardia vuoden 2019 huhtikuussa. RFC standardi eli kommentteja pyytävä standardi kuvaa push ilmoitusten palvelua PNS, jota käytetään herättämään keskeytettyjen istuntojen käynnistysprotokolla eli SIP. PNS palvelun push ilmoituksella voidaan herättää käyttäjän laite, vaikka käyttäjän laite olisi keskeytetyn istunnon tilassa. Laitteiden ja erityisesti mobiililaitteiden resurssien säästämiseksi käyttöjärjestelmät keskeyttävät sovellusten käyttämisen, jotta akun varauksen kestoa säästettäisiin. (RFC standardi)

4 OHJELMOINTIKIELI

4.1 Python

Python on korkean tason ohjelmointikieli, jonka avulla voidaan toteuttaa tehtäviä useilla eri ohjelmoinnin osa-alueilla. (Python yleisesti)

Pythonin historia

Python ohjelmointikielen loi 1990-luvun alussa hollantilainen Guido van Rossum, joka työskenteli tuolloin Amsterdamissa CWI:llä. Van Rossum työskenteli projektin parissa, joka tarvitsi syntaksia käsittelevän skriptikielen. Pelkästään projektia varten kehitettävä ohjelmointikieli ei vaikuttanut järkevältä, joten van Rossum kehitti yleisen ohjelmointikielen Pythonin. Python on avointa lähdekoodia eli kaikille käytettävissä ja siihen on ajan saatossa lisätty monia kirjastoja laajemman käytettävyyden parantamiseksi. (Pythonin historia)

Pythonin ominaisuudet

Python on joustava, helppokäyttöinen, avoimeen lähdekoodin pohjautuva olio-ohjelmointiin perustuva ohjelmointikieli, joka on suunniteltu optimoimaan kehityksen nopeutta. Python ohjelmointikieltä kuvataan usein skriptikieleksi helppokäyttöisyytensä takia ja osittain sen mahdollisuudesta yhdistää eri ohjelmistojen komponentteja sovellukseksi. Python tunnetaan yleisesti myös korkean tason ohjelmointikielenä, joka viittaa ohjelmointikielessä valmiiksi sisäänrakennettujen ominaisuuksien käyttöönoton mahdollisuuteen. Näiden sisäänrakennettujen ominaisuuksien avulla on mahdollista automatisoida yksinkertaisia tehtäviä, jotka muuten vaatisivat ohjelmoijilta aikaa manuaalisesti toteutettuna. (Lutz, 2006)

Python omaa useita standardin tyyppisiä ja valmiiksi sisällytettyjä toimintoja, kuten numeeriset, sekvenssit, luokat ja poikkeukset. Osa toiminnoista tukee funktioiden avulla usean tyyppisiä objekteja käytännöllisemmän vertailun mahdollistamiseksi. Python sisältää ainoastaan yhden tyypin moduuli objekteja riippumatta

moduulien lähteen ohjelmointikielestä. Moduuli voi sisältää funktiomääritelmiä sekä suoritettavia lauseita. Lauseiden tarkoituksena on alustaa moduuli, jonka takia ne suoritetaan vain ensimmäisellä kerralla tai toistuvasti niiden kuuluessa komentosarjaan. Python on luonut konseptin paketeista tarjotakseen nimeämishierarkian ja helpottaakseen moduulien järjestelyä. Paketit itsessään ovat moduuleita, jotka sisältävät useampia moduuleja. Jotta yhden Python koodin moduuli voi käyttää toisen moduulin koodia, käytetään `import`-komentoa. Ohjelmaan tuodaan moduuleja `import`-komennon avulla. Moduuleja käyttämällä ohjelmointikielellä toteutettavien ohjelmien mahdollisuudet laajenevat. Moduuleiden avulla mahdollistetaan mm. graafisen käyttöliittymän, tiedostojen käsittelyn, datan keräämisen verkosta ja monia muita toteutuksia. Jokaisella moduulilla on oma nimensä ja pakettien sisällä olevien moduulien nimet erotetaan nimen alussa olevasta paketin nimestä pisteellä. Python tarkastaa ensimäistä kertaa moduuleja tuotaessa moduulien nimen vastaavuuden, jos nimeä vastaavaa moduulia ei löydy järjestelmä ilmaisee `ModuleNotFoundError` -viestin virheen merkiksi. (Python moduulien käyttö)

Pythonin moduulit

Ennen moduulien tuomista ohjelmaan ne on asennettava tietokoneelle `pip`:n avulla. Pip on Pythonin pakettien manageroinnin työkalu, jonka avulla ulkoisia Pythonin paketteja ladataan tietokoneelle komentokehotteesta yksinkertaisten komentojen avulla. (Python moduulien asennus)

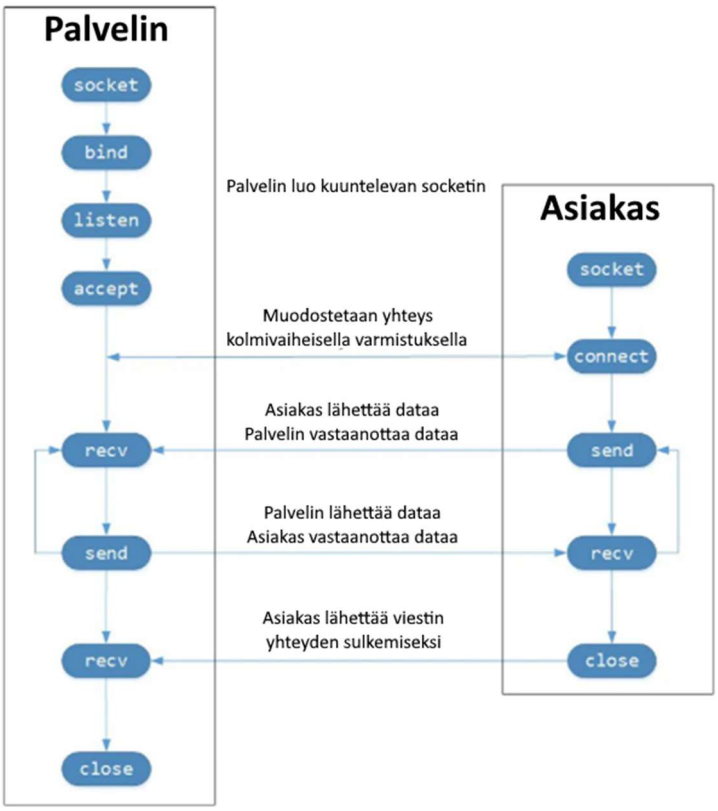
Aikaa käsittelevät järjestelmät tarvitsevat toimiakseen `time` -moduulin tuomisen järjestelmään. Moduulin avulla on mahdollista toteuttaa erilaisia toimintoja aikaan liittyen. Aikaa voidaan esittää monessa muodossa ja järjestyksessä, mutta matemaattisten operaatioiden ja vertailujen suorittamiseksi yksinkertainen lukema on suoraviivainen lähestymistapa, kuten vuosien ja kuukausien suodattaminen aikaleimasta. Yksittäinen lukema ajan ilmaisemiseksi on epoch aika, joka alkaa vuoden 1970 ensimmäisestä sekuntista. (Pythonin aika moduuli)

Jotta Python ohjelman on mahdollista kommunikoida tietoverkon kanssa, käytetään `socket` -moduulia. Moduuli tarjoaa pääsyn BSD socket -rajapintaan, jossa

on mahdollista käyttää TCP/IP -protokollaa datan siirtoon ja kommunikointiin. (Pythonin socket moduuli)

Berkeley Software Distributionissa kehitettiin vuonna 1983 ensimmäinen versio BSD socket API:sta, joka on ohjelmointirajapinta prosessien väliseen kommunikointiin. Viimeinen versio 4.4 julkaistiin vuonna 1995 ja TCP/IP -protokolla perustuu tähän vielä tänä päiväkin. (Richard Stevens, Fenner, Rudoff, 2003)

Kuvassa 2 esitetään client-server eli asiakkaan ja palvelimen välisen tiedonsiirron rakenne. Rakenne alkaa kuvan 2 vasemmalla puolella palvelimen puolen socketin määrittämisestä kuuntelua varten. Socketille määritetään IP-osoite ja portti, joiden kautta tiedonsiirron tulee tapahtua. IP-osoitteelle määritetään tämän jälkeen formaatti, jossa määritetty osoite on annettu. IP-osoitteen formaatin lisäksi socketille ilmaistaan tiedonsiirtotapa. TCP-tiedonsiirtoprotokolla on luotettava menetelmä, jossa tiedonsiirto toteutuu luotettavasti. Tiedonsiirrossa varmistetaan myös datan luettavuuden toteutuminen samassa järjestyksessä, kuin lähettäjän datan lähetyksessä, kun käytetään TCP-protokollaa. (Asiakas-palvelin viestintä)



KUVA 2. Asiakas-palvelin viestinnän rakenne (Asiakas-palvelin viestintä, muokattu)

Socketin määritysten jälkeen portti sidotaan `bind`-komennolla, jotta tiedonsiirtoyhteys varmistetaan. Tämän jälkeen palvelin asetetaan kuuntelemaan tulevaa tietoliikennettä `listen`-komennolla, joka asettaa palvelimen hyväksymään tulevia yhteyksiä. Vaikka palvelin on hyväksyvässä tilassa `accept`-komennon ansiosta, palvelin odottaa saapuvia yhteyksiä. Asiakkaan yhdistäessä palvelimelle tuodaan uusi socket objekti esittäen uutta yhteyttä, joka on IP-osoitteen ja portin numeron sisältävä pari. Tiedonsiirtoyhteyden muodostamisen jälkeen asiakas voi välittää viestinsä palvelimelle. Palvelin vastaanottaa viestin, jonka jälkeen asiakkaalle palautetaan vastaus. Tiedonsiirron sulkemiseksi asiakas lähettää palvelimelle viestin yhteyden sulkemisesta, jonka jälkeen asiakas sulkee oman yhteytensä. Palvelimen vastaanottaessa yhteyden sulkemisesta ilmoittavan viestin sulkee palvelin oman yhteytensä. Jos yhteyttä ei suljettaisi, yhteys jäisi avoimeksi ja uuden yhteyden muodostaminen monimutkaistuu. (Asiakas-palvelin viestintä)

Tietokoneet antavat kirjaimille ja muille symboleille niitä vastaavat numerokoodit, koska tietokoneet ja datansiirto prosessoivat fundamentaalisesti pelkästään numeroita. Tiedostojen siirrossa välitettävä data pakataan biteiksi lähettäjän päässä ja viesti puretaan vastaanoton yhteydessä. Datan pakkaus mahdollistaa suurien tekstimäärien pakkaamisen pieneksi määräksi dataa. Datan pakkauksessa käytetään Unicodea, joka antaa jokaiselle kirjaimelle, numerolle tai symbolille oman numerokoodin. (Unicode teoria)

Tiedostojen käsittelyssä Python tarjoaa useita sisäänrakennettuja toimintoja, joiden avulla tiedostoja voidaan avata tai käsitellä. Tiedostojen käsittelyyn on tarjolla myös moduuleita käyttöjärjestelmiä varten, mutta niiden käyttö riippuu tarkoituksesta. Tiedostojen käsittelyssä tiedostojen sulkeminen on tärkeää, jotta avatut tiedostot eivät kuluta käyttöjärjestelmän rajallisia resursseja turhaan. Tiedostojen ja niiden sisältöä käsittelevät toiminnot mahdollistavat useita tapoja datan käsittelyyn ja tiedostojen hallintaan. `Open`-komento antaa ohjelmalle pääsyn tiedostoon, komento tuo tiedoston ohjelmaan luettavaksi sekä muokattavaksi. `Open`-komento perässä oleviin sulkeisiin määritellään avattava tiedosto sekä tiedoston käsittelyn tapa, joka erotetaan kohdetiedostosta pilkulla. Tiedostoa voidaan käsitellä taulukossa 1 esitettyjen menettelyjen mukaisesti. (Python sisäänrakennetut toiminnot)

TAULUKKO 1. Tiedoston käsittelytavan määritys (Python sisäänrakennetut toiminnot, muokattu)

Tunnus tiedoston käsittelytavalle	Tarkoitus
’r’	Lukua varten (oletus)
’w’	Kirjoitusta varten
’x’	Tiedoston luomista varten
’a’	Tiedon lisäämistä varten
’b’	Binäärinen tila
’t’	Teksti tila
’+’	Päivittämistä varten (luku ja kirjoitus)

Lähdekoodin kommentointi on olennainen osa ohjelmointia ja se helpottaa ohjelman lukemista, kun lähdekoodin rivejä on selitetty sanallisesti. Kommentoinnissa Python käyttää #-merkkiä, joka sijoitetaan kommenttirivin alkuun. Kommenttirivi loppuu automaattisesti rivin vaihtuessa. Lähtökohtaisesti lähdekoodin tulee olla rakenteeltaan selkeää ja pelkän rakenteen avulla havainnollistaa koodin vaiheiden tarkoitusta. Kuitenkin lähdekoodiin voidaan joutua palaamaan muutosten ja päivitysten osalta myöhemmin, jolloin ohjelman kommentoinnista on hyötyä ohjelman kokonaisuuden ymmärtämisessä. Laadukkaan kommentoinnin tulee olla informatiivista ja antaa yleiskuvaa ohjelman vaiheista. (Python sisäänrakennetut toiminnot)

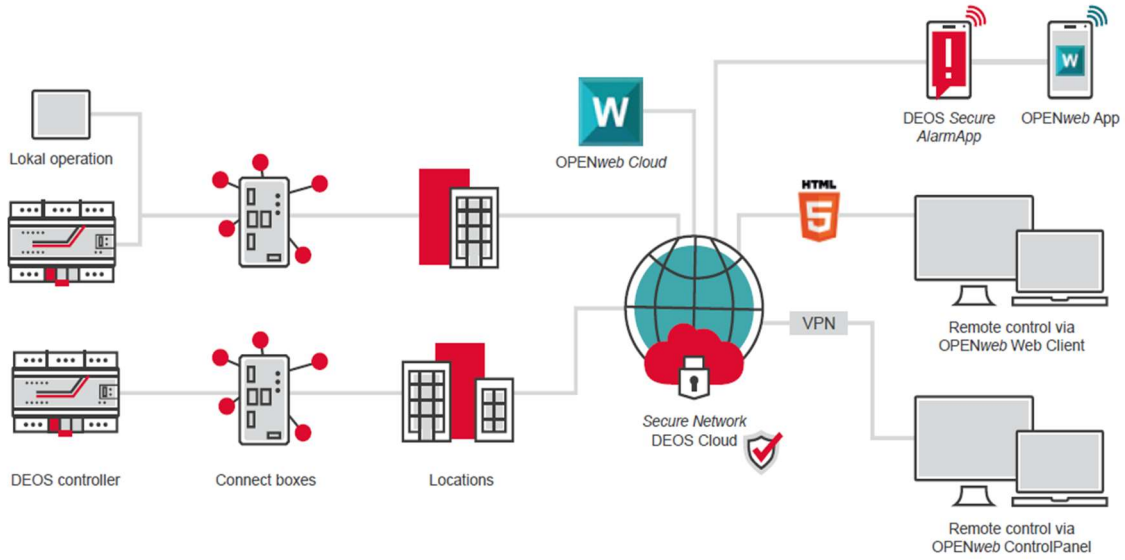
Jotta lähdekoodia suoritetaan jatkuva toimisena itsenäisenä ohjelmana, komentokehotteen kautta PyInstaller. PyInstaller lukee kirjoitetun Python ohjelman läpi ja pakkaa kaikki ohjelman edellyttämät moduulit yhteen kansioon tai vaihtoehtoisesti yksittäiseen itsenäisesti suoritettavaan exe-tiedostoon. (PyInstaller teoria)

Ohjelmien käyttöjärjestelmiltä vaatimia resursseja voidaan vähentää ohjelmaketkujen avulla, joita käytetään `treading`-komennon avulla. Ohjelmaketju on erillinen toteuttamisen kanava, joka mahdollistaa kahden tai useamman asian toteutuksen samanaikaisesti. Ohjelma rakentaminen ohjelmaketjupohjaiseksi voi myös selkeyttää ohjelman rakennetta. (Ohjelmaketjujen teoriaa)

5 OHJELMISTOT

5.1 OPENweb -hallintajärjestelmä

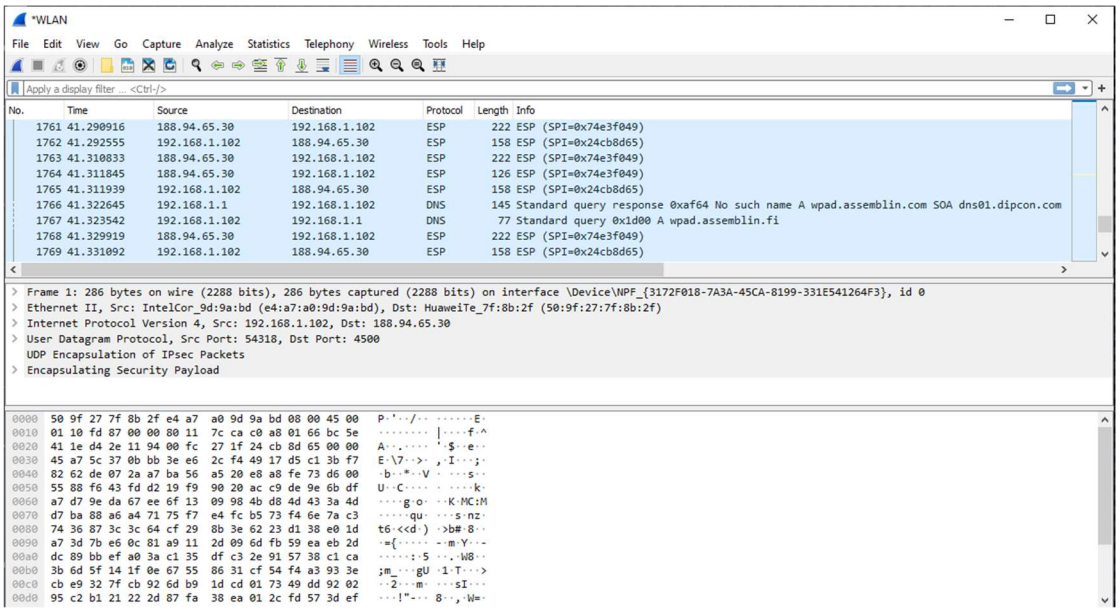
DEOS AG:n valmistama BACnet building management system (BMS) OPENweb -järjestelmä on verkkopohjainen rakennusten hallintajärjestelmä, jonka avulla valvotaan ja ohjataan HVAC-järjestelmiä sekä huonesäätöjä. Järjestelmää voidaan käyttää paikalliselta koneelta, etäohjatusta keskusvalvomosta tai mobiililaitteelta. Kiinteistöjen automaation mittaus- sekä tilatiedot voidaan välittää ja tallentaa DEOS-pilvipalveluun, josta ne ovat tämän jälkeen useiden eri päätelaitteiden käytettävissä. Tallennetun datan avulla on mahdollista luoda mittauksista graafeja kuvastamaan järjestelmän käyttäytymistä tarkasteltavalta ajanjaksolta. Kuvassa 3 esitetään rakennusten hallintajärjestelmän rakennetta, joka havainnollistaa tiedon siirtymistä rakennuksesta keskusvalvomoon asti. (BACnet BMS OPENweb)



KUVA 3. Rakennuksen hallintajärjestelmän rakennekuva (BACnet BMS OPENweb)

5.2 Wireshark -pakettianalysaattori

Wireshark on avoimeen lähdekoodiin perustuva ohjelmisto verkossa välitettyjen datapakettien analysoimiseksi, jonka avulla kaapataan valvottavan verkon välittämiä datapaketteja ja esitetään ne mahdollisimman yksityiskohtaisesti analysointia varten. Projektin käynnisti vuonna 1998 Gerald Combs ja projektin kehitystyöhön on osallistunut vapaaehtoisia tietoverkon ammattilaisia ympäri maailman. Wiresharkin avulla voidaan toteuttaa tietoverkon vian etsintää ja turvallisuuden valvontaa. Ohjelmistokehittäjät voivat Wiresharkin avulla testata sovellusprotokollien käyttäytymistä. Wiresharkin tiedostojen kaappaus soveltuu tietoverkon eri median välityksen tyyppeihin, kuten Ethernet- , WLAN- , Bluetooth- ja USB-tiedonsiirtoihin. Ohjelman avulla kaapatuista paketeista voidaan suodattaa paketteja vastaamaan hakukriteereitä, joiden avulla saadaan tuotua esille haluttuja tiedonsiirrossa välitettyjä paketteja. Kaapattuja datapaketteja voidaan myös tallentaa tiedostoiksi myöhempää käyttöä varten. Kuvassa 4 havainnollistetaan ohjelmiston graafisen käyttöliittymän yleisnäkymän rakenne. (Wireshark yleisesti)



KUVA 4. Wireshark -ohjelmiston yleisnäkymä

Kuvassa 4 ylhäällä sijaitsee ohjelman työkalupalkki, jonka alapuolella on suodatin kaapattujen pakettien suodattamiseen. Ohjelman käynnistyksen yhteydessä määritetään analysoitava verkko, josta siirtyviä verkon datapaketteja kaapataan

analysointia varten. Verkosta kaapatut paketit esitetään datapakettien listassa suodatinpalkin alla ja suodattimien avulla kaapatuista datapaketeista voidaan rajata näytettäviksi ainoastaan tutkimuksen kannalta olennaiset paketit. Datapakettien listasta voidaan valita yksittäinen rivi eli datapaketti tarkempaa tarkastelua varten, kun rivi valinnan jälkeen paketin tiedot avautuvat kuvan 4 keskimmäiseen ikkunaan ja alimmassa ikkunassa esitetään paketin tavut.
(Wireshark käyttäjän opas)

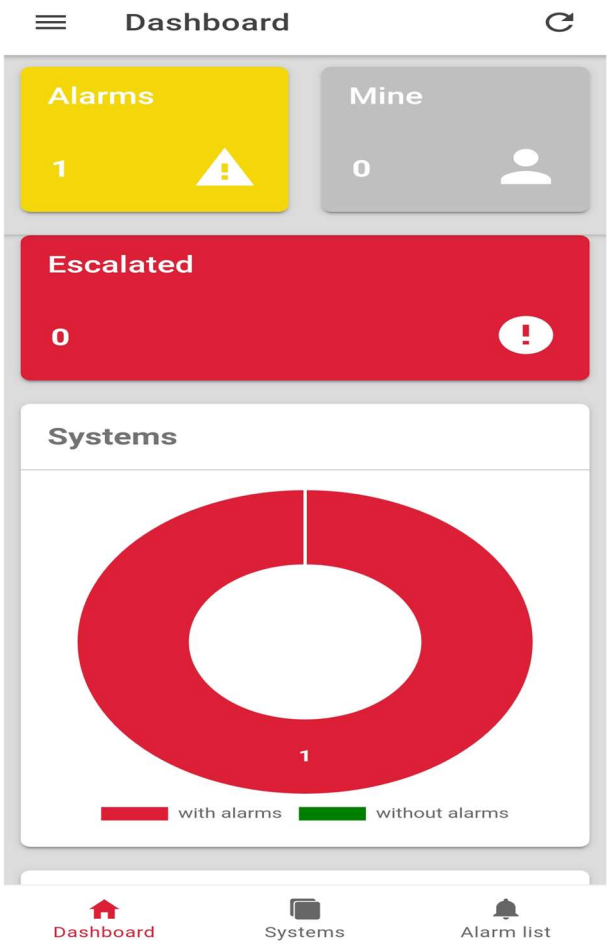
6 TOTEUTETTAVA OHJELMA

Ohjelman avulla välitetään DEOS AG:n OPENweb rakennusten hallintajärjestelmässä olevien etäkohteiden hälytykset uuteen DEOSalarm mobiilisovellukseen, joka tuotiin käyttöön uuden hallintajärjestelmän version myötä. Mobiililaitteisiin välitettävien hälytysten avulla voidaan tavoittaa nopeammin kiinteistöjen automaatiosta vastaavat henkilöt. Mobiilisovelluksen avulla järjestelmän hälytyksien vastaanottaminen helpottuu, koska kiinteistöjen automaatiosta vastaavien henkilöiden on mahdollista saada tieto ilmenneestä hälytyksestä missä tahansa.

6.1 Rakenteen määritys

Ohjelmistosuunnittelussa ohjelman luonti alkaa vaatimusten määrittämisellä eli määritetään, mitä toimintoja ohjelman tulee suorittaa. Vaatimukset tuovat esille ohjelmassa tarvittavat rajapinnat, joiden kanssa ohjelman tulee työskennellä. Ohjelman vaatimuksiksi muodostuivat kommunikointi internetin välityksellä, tiedostojen käsittely sekä aikaa käsittelevät toiminnot. Ohjelmalta vaadittavien rajapintojen kanssa toimimisen perusteella ohjelmointikieleksi valitaan Python, jonka korkean tason ohjelmointikielellä vaadittavat toiminnot ovat käytännöllisesti toteutettavissa. Kun ohjelman vaatimukset ovat selkeästi määritetty, voidaan ohjelmalle hahmotella ohjelman rakenne eli järjestys toimintojen suorittamiselle kaikissa sen vaiheissa.

Ohjelman rakenteessa tulee ensin muodostaa yhteys asiakkaaseen, jotta hälytysten vastaanottaminen on mahdollista. Yhteyden muodostamisen jälkeen vastaanotettavat hälytysviestit tallennetaan tekstitiedostoon. Tekstitiedostosta viestit muotoillaan vaihe kerrallaan csv-tiedostoon, josta viestilokia voidaan lukea vaivattomasti Excelin avulla. Kun viestien loki muutetaan Excelissä luettavaksi csv-tiedostoksi, on mahdollista ohjelman avulla analysoida viestien dataa. Datasta voidaan havaita hälytysten tyyppi, jonka perusteella ohjelma lähettäisi vaadittavien kriteerien täyttyessä DEOSalarm mobiilisovellukseen ilmoituksen hälytyksestä. Kuvassa 5 esitetään mobiilisovelluksen yleisnäkymä.



KUVA 5. DEOSalarm mobiilisovelluksen yleisnäkymä

Kuvassa 5 esitetyllä mobiilisovelluksella vastaanotetaan OPENweb hallintajärjestelmän hälytyksiä. Systems ikkunan avulla käyttäjälle ilmaistaan visuaalisesti järjestelmien yleistilanne, jolloin saadaan kattava havainnointi kokonaisvaltaisesta järjestelmien tilanteesta. Tiedoston tallennuksen ja käsittelyn jälkeen luotava palvelin lähettää asiakkaalle kuittauksen vastaanotetusta hälytyksestä. Kuittaus viestin vastaanotosta ei kuittaa itse hälytystä, vaan ilmoittaa asiakkaalle ainoastaan viestin vastaanotetuksi. Ilman viestin kuittaamista asiakas lähettäisi viestin toistuvasti tasaisin väliajoin, mikä aiheuttaisi ylimääräistä liikennettä kohteiden välisessä yhteydessä.

Rakenteen määrittelyn jälkeen ohjelman lähdekoodia voidaan alkaa kirjoittamaan. Lähdekoodia voidaan suorittaa ohjelman kehittämisen aikana toiminnallisuuksien testaamiseksi ja virheiden havaitsemiseksi ulostulosta. Ohjelman testaus säännöllisin väliajoin helpottaa toiminnallisuuden kartoittamista sekä virhei-

den havaitsemista lähdekoodista. Kun ohjelman osakokonaisuus todetaan toimivaksi, voidaan ohjelmaa kehittää eteenpäin toiminnallisuuksia lisäämällä. Toiminnallisuuksien lisäyksen jälkeen ohjelman kokonaisvaltainen toiminta tulee varmistaa. Alkuperäisten toimintojen tulee toteutua vastaavalla tavalla, kuin ennen lisättyjä toimintoja.

Yksittäisen viestin rakenne esitetään kuvassa 6. Viestin rakennetta voidaan luonnehtia kaksiosaisena, jossa tyhjän rivin yläpuolella ensimmäisessä osiossa ilmaistaan viestin yleiset tiedot ja jälkimmäisessä osiossa on viestin varsinaisen sanoman sisältö. Viestin varsinainen sisältö on todellisuudessa yksittäinen rivi, jossa arvot ovat pilkulla erotettuja eli csv-formaatissa.

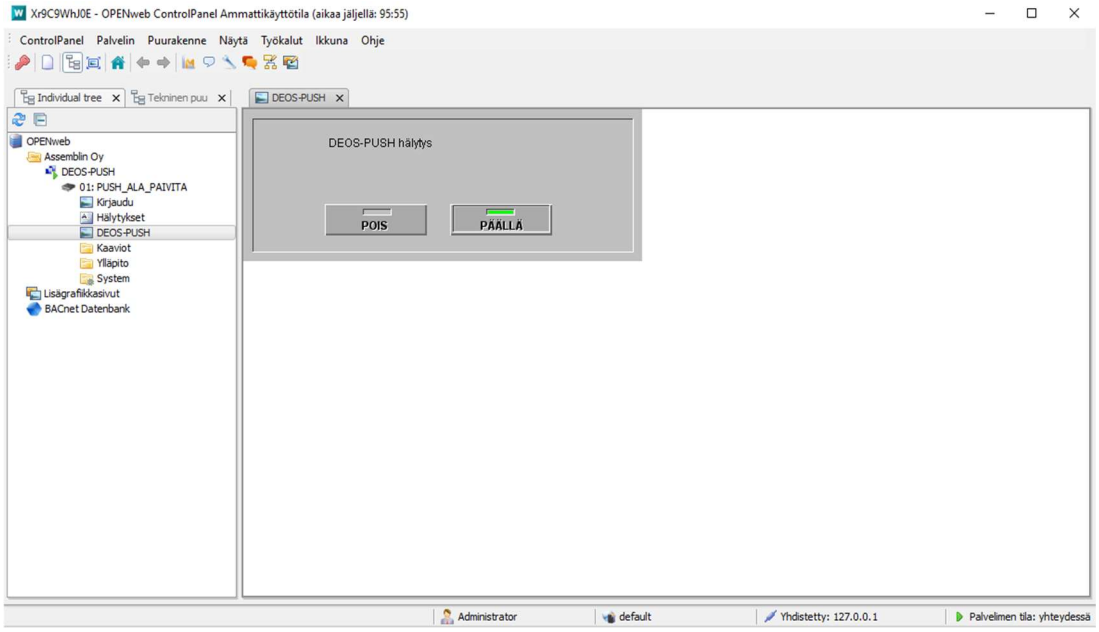
```
POST /pushEvents HTTP/1.1
Content-Length: 495
Content-Type: application/json; charset=UTF-8
Host: 127.0.0.1:5053
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_172)
Accept-Encoding: gzip,deflate

{"reporterId":"69","token":"410954793","secToken":null,"events":[{"eventObject":"12","dateTime":1554367895000,"state":false,
"stateText":"POIS","message":"VAK2: TULEEKO HÃ„LYTYS LATAUKSEN JÃ„LKEEN? DMX","serverId":"0001","controllerId":"01","controllerGroupId":"0002",
"serverName":"JT OPENweb 9 Server ;-)","controllerGroupName":"Assemblin_C710","controllerName":"01: JT","priority":"0","eventType":"0",
"instruction":"Ohjetietoja ei lÃ¶ytynyt","sourceType":0,"sourceProtocol":0,"eventId":12619}]}
```


KUVA 6. Yksittäisen viestin rakenne

Kuvassa 6 esitettävä yksittäisen viestin rakenne on hankalasti luettava ja useiden hälytysviestien seasta manuaalinen tiedon etsintä hidastuu vaikeasta luettavuudesta johtuen. Hälytysten nopea havaitseminen ennaltaehkäisee huoltojen laajuutta ja palvelee kiinteistöjä ja kiinteistöissä olevien henkilöiden hyvinvointia. Kuvan 6 mukaisten viestien ohjelmallinen tulkinta on myös haastavaa, koska viestin rakenne ei ole yhtenäinen. Viestien rakennetta tulee käsitellä, jotta kriittiset hälytykset voidaan tunnistaa ohjelmallisesti. Kun kriittisten hälytysten tunnistaminen on mahdollista toteuttaa automaattisesti ohjelman avulla, kriittiset hälytykset voidaan välittää nopeasti mobiilisovellukseen. Nopea hälytysten välittäminen mobiilisovellukseen on tärkeää, jotta ohjelman avulla tavoitettaisiin nopeammin kiinteistöjen automaatiosta vastuussa olevat toimihenkilöt.

6.2 Ohjelman toteutus

Luotavan ohjelman ensimmäisenä osana luodaan rakenteen määrittelyn mukaisesti verkkoyhteys hälytysten vastaanottamiseksi OPENweb -hallintajärjestelmän ohjelmasta, johon yrityksen toimesta oli luotu kuvan 7 mukainen testausympäristö kiinteistöä eli etäkohdetta simuloivaksi asiakkaaksi.



KUVA 7. OPENweb -ohjelman testausympäristö

Asiakkaalta lähetetään hälytyksen tilan muuttuessa hälytysviesti, joka palvelimen tulee vastaanottaa. Ensimmäisessä ohjelman ketjussa määritetään ainoastaan kuunneltava IP-osoite ja portti, joissa tiedonsiirron tulee tapahtua. Toisessa ohjelmaketjussa määritetään yhteyden muodostamista varten `socket` ominaisuudet, joita varten lähdekoodiin on tuotava socket moduuli.

```
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

KUVA 8. Socket määrittäminen

Kuvassa 8 on esitetty socketin alustamisessa käytettävä ohjelmarivi, jossa `socket.socket` tarkoittaa moduulista `socket` tuotavaa toimintoa `socket`. Toiminnon sisälle sulkeisiin määritetään `AF_INET` ilmaisemaan käytettävän internet protokollan muodoksi IPv4 -formaattia. Tiedonsiirron tyypiksi määritetään

SOCK_STREAM avulla TCP-protokolla. Kun luotavalle palvelimelle on määritetty kuunneltava IP-osoite ja portti sekä edellä mainitut socketin ominaisuudet, voidaan aloittaa määritetyn asiakkaan kuunteleminen vastaanotettavan datan varalta listen-komennolla. Kuuntelun aloituksen jälkeen siirrytään seuraavaan ohjelmaketjuun, jossa yhteyspyynnöt hyväksytään accept-komennolla. Vaikka yhteyden hyväksyvä accept-komento toteutuu ajallisesti ennen etäkohteelta toteutuvaa yhteyden muodostamista, yhteys on estetty ennen ensimmäistä viestin välitystä etäkohteelta ohjelman käynnistämisestä. Kun määritetystä osoitteesta lähtee ensimmäisen kerran palvelimen käynnistämisen jälkeen dataa, ohjelman hyväksyy muodostettavan yhteyden. Vasta yhteyden hyväksynnän jälkeen asiakkaalta lähetetyn viestin data vastaanotetaan.

Vastaanotettavat viestit sisällytetään vastaanoton yhteydessä muuttujaan nimeltä data, koska muuttujan avulla vastaanotettavalle viestille toteutettavien toimintojen käyttö lähdekoodissa on yksinkertaisempaa. Vastaanotettavat viestit saapuvat palvelimeen tavuina, mikä määritetään data muuttujasta type-komennon avulla. Tavu muotoisten viestien sisältö tulee purkaa takaisin tekstiksi, jotta vastaanotettavien viestien tallentaminen toteutuisi onnistuneesti. Ilman sisällön purkamista viestien sisältämän tekstin kirjoittaminen tiedostoon ilman ylimääräisiä symboleja tai tietojen korruptoitumista ei olisi mahdollista tavu muodosta johtuen. Viestien sisältö puretaan käyttämällä UTF-8 koodausta komennon decode avulla. Komennon decode sulkeiden sisälle ei tarvitse määrittää viestiin purkuun liittyviä määrityksiä, koska Pythonin decode-komennossa UTF-8 koodaus on oletusasetuksena. Viestin sisällön purkaminen UTF-8 koodauksella esiintyy kuvassa 9 keskimmäisellä rivillä.

```
#Määritetään viesteille tallennustiedosto joka toimii myös varmuuskopiona
source='BackUp.txt'
#Kohde tiedosto csv-formaatin tiedostolle johon muokattu viestiloki siirretään
destination='message_history.csv'

#Lisätään vastaanotettu viesti txt tiedostoon
with open(source, 'a+') as file:

    file.write(str(data.decode())+',"time received":'+str(time_stamp)+'}]}]}\n')

#Avataan lähdetiedosto
with open(source,'r') as f:

    #Kirjoitetataan lähdetiedosto rivi kerrallaan kohdetiedostoon
    with open(destination, "w") as file:
        for line in f:
            file.write(line)
```

KUVA 9. Viestin sisällön purku, tallennus ja siirto

Kuvassa 9 esitetään viestin sisällön purkamisen lisäksi viestin kirjoittaminen tekstitiedostoon ja sisällön siirto csv-tiedostoon. Viestin lisääminen tekstitiedostoon toteutetaan Pythonin valmiiksi sisäänrakennettujen `with`- ja `open`-komentojen avulla, joten ylimääräisiä moduuleja ei tekstitiedostoon kirjoittamisen yhteydessä tarvita. Kuvan 9 yläosassa määritetään ohjelmalle lähdetiedosto, johon vastaanotettavat viestit kirjoitetaan vastaanoton yhteydessä sekä kohdetiedosto mihin viestien muokattu sisältö tallennetaan. Muuttujien käyttö viestien käsittelyssä selkeyttää lähdekoodin rakennetta. Lisäksi ohjelman muokkaaminen uusille asiakkaille on helpompaa, kun tiedostojen nimeäminen tarvitsee ilmaista vain yhdessä lähdekoodin kohdassa.

Tekstin käsittelyssä `open`-komentoa suoritetaan `with`-komennon sisällä, koska `with`-komennon avulla tiedoston sulku toteutuu automaattisesti tiedoston käsittelyn jälkeen. Viestien tekstin käsittely toteutetaan ohjelmassa useassa vaiheessa toimintavarmuuden parantamiseksi, vaikka tämä lisääkin lähdekoodin pituutta. Useassa vaiheessa toteutettavasta tekstin käsittelystä huolimatta `with`-komentoa käyttämällä lähdekoodissa vaadittavien rivien määrä vähenee, koska käsiteltävää tiedostoa ei tarvitse jokaisen tiedoston avaamisen jälkeen sulkea erillisellä komentorivillä. Ylimääräisten rivien välttäminen lähdekoodissa helpottaa sen lukemista. Viestien muotoilussa csv-formaattiin on tavoitteena muotoilla yksittäisen viestin sisältö yhdelle riville, jolloin niiden lukeminen yksinkertaistuu.

Uuden hälytysviestin tiedostoon kirjoittamisessa on olennaista huomioida aikaisemman datan säilyminen. Aikaisempien hälytysten säilyttämiseksi tiedostossa käytetään uuden hälytyksen kirjoittamisen yhteydessä ilmaisua ’a’, joka ilmaisee ohjelmalle uuden viestin lisäämisestä tiedoston loppuun. Viestin muotoilussa kohdetiedoston sisältö luetaan ilmaisun ’r’ avulla, jonka jälkeen `replace`-komennolla ilmaistaan tekstille toteutettava muutos. `Replace`-komennolle määritetään merkkijonoja korvattavaksi toisilla. Määritetyn merkkijonon muutoksen jälkeen tiedosto avataan uudelleen ’w’ ilmaisulla, joka tarkoittaa tiedostoon kirjoittamista. Viestin muotoilu edellyttää huolellista määrittelyä korvattavien tekstin osien määrittelyssä, jotta viestin sisältö säilyy muotoilun aikana ja sen jälkeen.

Viesteistä on mahdollista poistaa toistuvat elementit, jotka voidaan ilmaista ylätunnisteena otsikon tapaan tiedoston ensimmäisellä rivillä.

Ohjelman avulla tekstitiedostoon kirjoitettujen viestien rakennetta voidaan suodattaa ja muotoilla erinäisten komentojen avulla. Viestien muokkaamisessa avataan ensin tekstitiedosto, jonka jälkeen kaikki muokkaukset tehdään vaiheittain lähdekoodissa lähdetiedoston avaamisen alla. Lähdetiedoston rivit luetaan yksi kerrallaan csv- eli kohdetiedostoon. Kuvan 6 viestistä suodatetaan ensiksi kaikki tyhjät rivit. Rivien lopussa on rivin vaihtoon viittaava tunniste ”\n”, jonka avulla viestin jokaisen rivin päätteeksi lisätään pilkku csv-tiedostotyypin pohjustamiseksi. Seuraavaksi kaikki rivin vaihdon tunnisteet poistetaan tiedostosta. Viestin viimeisenä esiintyvän arvon perässä olevan ainutlaatuisen merkkijonon avulla viestit erotetaan toisistaan. Erottelussa poikkeava merkkijono korvataan rivin vaihdolla.

Jotta viestien sisällön pituutta voidaan optimoida, toistuvat otsikko määritteet itse arvojen edestä poistetaan tekstistä. Otsikkojen poistaminen viesteistä selkeyttää huomattavasti viestien sisällön lukua, kun suuri määrä ylimääräisestä tekstistä saadaan suodatettua pois. Tässä vaiheessa viestien muokkaus on valmis, joten kirjoitetaan viesteistä poistetut otsikot ylimmälle tiedoston riville, jonka jälkeen viestit kirjoitetaan csv-tiedostoon. Kuvassa 10 on esitettynä muotoilun jälkeen saavutettu lopputulos.

Post	Content lenght	Content Type	Host:Port	Connection	User-Agent	Accept En	Accept En	reporterID	Token	secToken	eve	Datetime	State
POST /pushEvents HTTP/1.1	479	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587629075000	EPÄTOSI
POST /pushEvents HTTP/1.1	483	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587629086000	TOSI
POST /pushEvents HTTP/1.1	479	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587631183000	EPÄTOSI
POST /pushEvents HTTP/1.1	483	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587631213000	TOSI
POST /pushEvents HTTP/1.1	479	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587714524000	EPÄTOSI
POST /pushEvents HTTP/1.1	483	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587714554000	TOSI
POST /pushEvents HTTP/1.1	479	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587714564000	EPÄTOSI
POST /pushEvents HTTP/1.1	483	application/json; charset=UTF-8	[redacted]	Keep-Alive	Apache-HttpClient/4.5.5 Java/1.8.0_202	gzip	deflate	:1234	641830723	null	1	1587714595000	TOSI

KUVA 10. Viestien rakenne muotoilun jälkeen

Kuvassa 10 ylimmällä rivillä on otsikkoina viestien sisällöstä poistetut arvojen nimitykset. Otsikkorivin jälkeen alkaa vastaanotetuista hälytyksistä koostuva viestiloki, jossa yksittäinen rivi vastaa yhtä viestiä. Viestien käsittelyn lopputulok-

sesta voidaan todeta viestien rakenteen soveltuvan paremmin myös manuaaliseen tarkasteluun, vaikka lähtökohtana oli mahdollistaa viestien sisällön automatisoitu analysointi luotavan ohjelman avulla.

Viestin loppuun lisätään aikaleima viestin tiedostoon kirjoittamisen ajankohdasta. Tämä aikaleima mahdollistaa etäkohteen ja palvelimen välisen verkkoyhteyden analysointia, kun viestin vastaanoton ja tapahtuman ajankohtaa voidaan vertailla. Aikaleiman arvo lisätään viestin sisällön perään viestin rakennetta mallintaen, jotta ulkoasu olisi yhtenevä.

Viestin vastaanoton jälkeen palvelin lähettää asiakkaalle viestin kuittaukseksi viestin vastaanotosta, kun viestin vastaanotosta ilmoitetaan lähettäjälle asiakas lopettaa viestin uudelleen lähettämisen. Palvelimen on siis tärkeää välittää asiakkaalle viesti hälytyksen vastaanotosta, jotta viestien toistuvaa lähettämistä ei ilmenisi. Ilman toistuvaa viestien lähetystä tiedonsiirtoyhteys asiakkaan ja palvelimen välillä pysyy paremmin avoinna uusien viestien välittämiselle.

Ohjelman kehityksessä ja testaamisessa tuli kiinnittää huomiota jatkuvan toiminnan varmuuteen. Ohjelman on toimittava jatkuvasti, jotta kaikki kiinteistöjen hälytykset voidaan vastaanottaa ja kriittisten tilanteiden tapauksessa hälytykset välitettäisiin mobiilisovellukseen. Kun ohjelman rakenne ja toiminta vaikuttavat jatkuva toimiselta, kirjoitetusta lähdekoodista voidaan tehdä itsenäisesti suoritettava exe-tiedosto PyInstallerin avulla. Komentokehotteessa siirrytään kansioon, jossa ohjelman lähdekoodin tiedosto sijaitsee. Itsenäinen exe-tiedosto paketti saadaan kuvan 11 mukaisella komennolla.

```
>pyinstaller --onefile  server.py
```

KUVA 11. Lähdekoodin muuttaminen exe-tiedostoksi

Kuvassa 11 esitettävällä komentorivillä ilmaistaan aluksi Pyinstallerin käytöstä. Seuraava osa alkaa kahdella väliviivalla ja sanalla onefile, jolla ilmaistaan toteutettavalle muunnokselle yhden tiedosto paketin luomisesta. Viimeisenä komentoon kirjoitetaan lähdekoodin tiedoston nimi, josta exe-tiedosto halutaan tehdä. Kun ohjelma on muutettu itsenäisesti suoritettavaksi ohjelmaksi, ohjelman toiminnan testaus varmistetaan vielä simuloitavan ympäristön avulla.

7 POHDINTA

Työn taustana oli mobiilisovelluksen käytön integroiminen kiinteistöautomaatiojärjestelmän yhteyteen. Mobiilisovelluksen avulla on mahdollista nopeuttaa järjestelmässä ilmeneviin hälytyksiin reagoimista, kun kiinteistöjen automaatiosta vastuussa olevat toimihenkilöt voidaan tavoittaa nopeammin. Hälytyksiin nopeamman reagoinnin avulla tapahtuneisiin vikoihin kuluvaa vasteaikaa voidaan vähentää huomattavasti, jolloin vikojen vaikutuksia kiinteistöihin tai kulutukseen voidaan vähentää. Vikatilanteiden vaikutusten vähentäminen parantaa yleistä yrityksen tarjoaman palvelun laatua, jolloin asiakkaiden tyytyväisyys tarjottavaan palveluun kasvaa.

Tavoitteena oli kehittää ohjelma välittämään järjestelmän kriittiset hälytyksen rakennusten hallintajärjestelmään kehitettyyn DEOSalarm mobiilisovellukseen. Työn avulla ei saavutettu mobiilisovelluksen käyttöönottoa, mutta työn avulla toteutettiin pohjatyö hälytysten ohjelmalliseen analysointiin. Opinnäytetyössä toteutettu ohjelma muokkaa tekstinä vastaanotettavat viestit ohjelmallisesti analysoitavan datan muotoon, josta datan analysointia on mahdollista toteuttaa. Ennen viestien käsittelyä viestit olivat pelkästään yksinkertaista tekstiä, mutta viestien muotoilu csv-tiedostoformaattiin mahdollistaa arvojen vertailua viestien datasta. Datan avulla viestien tiedoille on mahdollista toteuttaa ylimääräisiä operaatioita, jotka pelkässä tekstimuodossa eivät olisi mahdollisia. Näiden ylimääräisten operaatioiden avulla viestien sisällöstä on mahdollista saada lisää tietoa, joka ei ilmenisi suoraan viestin sisällöstä.

Mobiilisovellukseen on tarkoitus välittää ainoastaan kriittisimmät kohteiden ja taloudellisuuden kannalta vaikuttavat hälytykset, jotta DEOSalarm mobiilisovelluksen kautta vastaanotettavien hälytysten määrä säilyisi kohtuullisena vastuuhenkilöstölle heikentämättä hälytyksien merkittävyyttä. Ohjelma voidaan tulevaisuudessa ottaa käyttöön kaikissa yrityksen OPENweb hallintajärjestelmään yhteydessä olevissa etäkohteissa. Opinnäytetyössä haasteellista oli verkon välityksellä toteutettavan tiedonsiirron ehdonalaisuuksien sisäistäminen, joka hidasti ohjelman lopullisen toteutuksen valmistumista.

Viestien lokiin voitaisiin tulevaisuudessa lisätä epoch aikaleimojen lisäksi niitä vastaavat yleisen aikaformaatin aikaleimat, jotta niiden lukeminen olisi käytännöllisempää. Graafisen käyttöliittymän lisääminen ohjelman käyttämiseksi edistäisi ohjelman käytettävyyttä, jolloin uudet hallintajärjestelmän kohteet voitaisiin lisätä ohjelmaan itse ohjelmasta käsin. Tulevaisuudessa opinnäytetyö auttaa mobiilisovelluksen käyttöönotossa etäkohteiden kriittisten hälytysten valvonnassa. Sovellus tulee käyttöön rakennusautomaatio urakoinnin vastuutoimihenkilöillä.

LÄHTEET

Asiakas-palvelin viestintä. Luettu 29.4.2020
https://realpython.com/python-sockets/

Assemblin Oy rakennusautomaatiourakointi. Luettu 6.5.2020
https://fi.assemblin.com/palvelut/rakennusautomaatio/rakennusautomaatioura-kointi

Assemblin Oy energiatehokkuus. Luettu 6.5.2020
https://fi.assemblin.com/palvelut/energiatehokkuus-ja-elinkaaripalvelut/ener-giatehokkuus-ja-assemblin

Assemblin Oy etähallinta. Luettu 6.5.2020
https://fi.assemblin.com/palvelut/energiatehokkuus-ja-elinkaaripalvelut/ener-giatehokkuus-ja-assemblin

BACnet BMS OPENweb. Luettu 3.3.2020
https://www.deos-ag.com/wp-content/uploads/2020-01-23_BR_OPEN-web_EN_low.pdf

Ohjelmaketjujen teoriaa. Luettu 6.5.2020
https://realpython.com/intro-to-python-threading/#what-is-a-thread

Lutz, M. 2006. Programming Python.

PyInstaller teoria. Luettu 28.4.2020
https://www.pyinstaller.org/

Pythonin historia. Luettu 3.3.2020
https://docs.python.org/3/license.html

Pythonin moduulien asennus. Luettu 7.5.2020
https://docs.python.org/3/installing/index.html

Python moduulien tuonti. Luettu 2.5.2020
https://docs.python.org/3/reference/import.html

Python moduulien käyttö. Luettu 6.5.2020
https://docs.python.org/3/reference/import.html

Python sisäänrakennetut toiminnot. Luettu 6.5.2020
https://docs.python.org/3/library/functions.html

Python yleisesti. Luettu 9.5.2020
https://docs.python.org/3/faq/general.html

RFC standardi. Luettu 9.5.2020
https://tools.ietf.org/html/rfc8599

Richard Stevens, Fenner, Rudoff, 2003. UNIX Network Programming: Vol 1: The Sockets Networking API

Suomäki, J & Vepsäläinen, S. 2013. Talotekniikan automaatio – Käyttäjän opas.

ST Käsikirja. 2018. Rakennusautomaatiojärjestelmät

Unicode teoria. Luettu 1.5.2020
https://docs.python.org/3/howto/unicode

Wiresharkista yleisesti. Luettu 5.3.2020
https://www.wireshark.org/

Wireshark käyttäjän opas. Luettu 5.3.2020
https://www.wireshark.org/docs/wsug_html_chunked/