



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

**MAURI MUSTONEN**  
**SÄHKÖASEMAN ÄLYKKÄÄN ELEKTRONIIKKALAITTEEN**  
**VIESTIEN TILAUS JA PROSESSOINTI**  
Diplomityö

Tarkastaja: Professori Kari Systä

Jätetty tarkastettavaksi 17. touko-  
kuuta 2018

## TIIVISTELMÄ

**MAURI MUSTONEN:** sähköaseman älykkään elektroniikkalaitteen viestien tilaus ja prosessointi  
Tampereen teknillinen yliopisto  
Diplomityö, 15 sivua  
Toukokuu 2018  
Tietotekniikan koulutusohjelma  
Pääaine: Ohjelmistotuotanto  
Tarkastaja: Professori Kari Systä

Avainsanat: Ohjelmistotuotanto, IEC 61850, MMS, AMQP

*Tiivistelmä on suppea, 1 sivun mittainen itsenäinen esitys työstä: mikä oli ongelma, mitä tehtiin ja mitä saatiin tulokseksi. Kuvia, kaavioita ja taulukoita ei käytetä tiivistelmässä.*

*Laita työn pääkielellä kirjoitettu tiivistelmä ensin ja käänös sen jälkeen. Suomenkielille kandidaatintyölle pitää olla myös englanninkielinen nimi arkistointia varten.*

Sähkönjakeluverkko on tärkeä osa nykyistä yhteiskuntaa ja sen päivittäistä toimintaa. Siksi verkon toiminta on elintärkeää yhteiskunnan sujuvan toiminnan kannalta. Sähköverkko koostuu sähköntuotantolaitoksista, sähkölinjoista ja sähköasemista. Sähköverkon eri komponenttien avulla sähkö toimitetaan tuotantolaitoksesta kuluttajan seinäpistokeeseen asti. Sähköasemat ja niiden automatisointi ovat tärkeässä roolissa verkon yleisen toiminnan ja turvallisuuden takaamiseksi.

Nykypäivänä sähköverkon sähköasemien automatisointiin käytetään tietokoneita, kommunikointi- ja verkkolaitteistoja [5, s. 659]. Asemien automatisointi toteutetaan *älykkäillä elektroniisilla laitteilla* (eng. IED), jotka toteuttavat aseman monitoroinnin ja ohjaamisen. IED:t konfiguroidaan toteuttamaan aseman erilaisia funktioita ja toiminnallisuuksia. Monen eri toimijan toimiessa allalla, voisivat kaikki vapaasti toteuttaa oman version laitteistaan ja määrittää kuinka näitä käytettäisiin. Tästä seurauksena olisi, että laitteet eivät olisi yhteensopivia muiden toimittajien laitteiden kanssa. Tilannetta helpottamaan ja mahdollistamaan eri valmistajien yhteensopivuuden laitteiden välillä. On *International Electrotechnical Commission* määrittänyt standardin nimeltä IEC 61850. Standardi määrittää kommunikointi protokollat sähköasemien IED laitteille. Standardi määrittää abstrakteja malleja sähköaseman kommunikoinnin mallintamiseen oliopohjaisesti. Tekniikasta tai toteutuksesta riippumaton standardimalli voidaan määrittää toimivaksi erilaisilla tekniikoilla. Standardi määrittää malleja myös erilaisten datapisteiden tilaukseen viesteinä IED:ltä. Eri asiakastoteutukset voivat tilata laitteeseen määritettyjä datapisteitä standardin määrittämillä säännöillä. Standardin ansiosta viestien muoto on ennaltamäärätty ja on tekniikasta riippumaton. Asiakasohjelmisto voi käyttää viestejä esimerkiksi jännitemittatietojen tilaukseen.

Tässä diplomityössä keskitytään yllämainitun viestejä tilaavan asiakasohjelmiston suunnitteluun ja toteutukseen. Työ sisältää tutkimusta ja vertailua arkkitehtuuriin ja toteutukseen liittyen. Lopullinen toteutus asiakasohjelmistosta tilaa viestejä IED:ltä, prosessoi ja uudelleenjulkaisee ne myöhempää käyttöä varten. Ennen työn aloitusta, ohjelmistosta oli jo saatavilla protoversio, jota hyödynnettiin uuden toteutuksen suunnittelussa. Protoversiossa olevat ongelmat ja puutteet korjattiin uuteen toteutukseen.

## ALKUSANAT

*Mistä tämän diplomityönaiheen sain ja kiittää eri ihmisiä ketä työssä oli sidoshenkilöinä.*

Tampereella, 19.4.2018

Mauri Mustonen

## SISÄLLYSLUETTELO

1.	JOHDANTO .....	1
1.1	Tausta .....	1
1.2	Laajuus .....	1
1.3	Tavoitteet.....	1
2.	TEORIA.....	2
2.1	IEC 61850 -standardi .....	2
2.1.1	Standardin määrittämä abstraktimalli .....	2
2.1.2	Viestiblokin konfigurointi ja tilaus .....	2
2.1.3	Viestin rakenne .....	2
2.2	Abstraktimallin sovitukset MMS-protokollaan .....	2
2.2.1	MMS-protokolla .....	2
2.3	Advanced Message Queuing Protocol .....	3
2.3.1	Viestien välitysmekanismit.....	3
2.3.2	Tilaus ja julkaisu -mallin osat.....	3
3.	ALKUTILANNE .....	4
3.1	Kokonaiskuva.....	5
3.2	Ratkaistavat ongelmat .....	5
3.3	Tutkimuskysymykset .....	6
4.	SUUNNITTELU.....	7
4.1	Suorituskyvyn parantaminen .....	7
4.2	Järjestelmän hajautus .....	7
4.3	Ohjelmiston parametrisointi.....	7
4.4	Arkkitehtuurin suunnittelu .....	7
4.5	Prosessoidun viestin muoto.....	7
5.	TOTEUTUS .....	8
5.1	Ohjelmiston toteutuksen valinta.....	8
5.2	Kielen valinta .....	8
5.3	RabbitMQ .....	8
5.4	Käytettävät kirjastot .....	8
5.4.1	libiec61850 .....	8
5.4.2	rabbitmq-c.....	9
5.4.3	JSON-formatointi .....	9
5.5	Jatkokehitys.....	9
6.	ARVIOINTI .....	10
7.	TULOKSET.....	11
8.	YHTEENVETO.....	12
	LÄHTEET.....	15

## LYHENTEET JA MERKINNÄT

AMQP	engl. <i>Advanced Message Queuing Protocol</i>
FFI	engl. <i>Foreign Function Interface</i> , mekanismi, jolla ajettava ohjelma voi kutsua toisella kielellä implementoitua funktiota
GIL	engl. <i>Global Interpreter Lock</i> , tulkattavassa kielissä oleva globaali lukitus, joka rajoittaa yhden säikeen suoritukseen kerrallaan
HAL	engl. <i>Hardware Abstraction Layer</i> , laitteistoabstraktiotaso abstraktoimaan laitteen toiminnallisuus lähdekoodista
IED	engl. <i>Intelligent Electronic Device</i> , sähköaseman älykäs elektroninen laite, joka tarjoaa toimintoja monitorointiin ja kontrollointiin
MMS	engl. <i>Manufacturing Message Specification</i>
RCB	engl. <i>Report Control Block</i> , raporttien konfigurointiin ja tilaukseen tarkoitettu lohko asiakasohjelmalle

# 1. JOHDANTO

*Kirjoita tähän johdantoa työstä ja aiheesta. Kuinka työ valittiin ja miksi tekijä valitsi tämän työn. Kirjoita myös mitä tehtiin. Kokonaiskuva työstä pitäisi saada johdannosta. Alusta lukijaa todella hyvin yleismaallisella kuvalla ja taustalla. Asiaa pitäisi olla hyvin hallussa ennen teoriaosuuteen siirtymistä.*

## 1.1 Tausta

## 1.2 Laajuus

## 1.3 Tavoitteet

## 2. TEORIA

*Tähän kohtaan kirjoittaa teoriaa mitä tarvitaan työn toteutuksen ymmärtämisen kannalta. Kaikki työssä tarvittava teoria kuvataan tämän otsikon alla.*

### 2.1 IEC 61850 -standardi

*Kirjoita yleisesti mikä on IEC 61850 -standardi ja mitä varten se on olemassa. Kerro myös kuinka standardi on pilkottu pienempiin dokumentteihin ja mitä kukin käsittelee.*

#### 2.1.1 Standardin määrittämä abstraktimalli

*Kirjoita tähän mitä standardin IEC 61850-7-2 osuudessa määritellään abstraktoimalla fyysisiä laitteita ja palveluita rajapinnoiksi ja olioiksi. Käsittelee standardin Abstract communication service interface (ACSI).*

#### 2.1.2 Viestiblokin konfigurointi ja tilaus

*Kirjoita tähän IEC 61850 -standardin määrittämästä abstraktista raportointimallista. Tätä raportointi mekanismeja tullaan käyttämään raporttien tilauksessa ja sen konfigurointi täytyy ymmärtää toteutettavan ohjelmiston kannalta.*

#### 2.1.3 Viestin rakenne

*Kirjoita tähän standardin määrittämästä viestin rakenteesta ja mitä tietoa se sisältää. Kerro myös sen vaihtoehtoisista kentistä.*

### 2.2 Abstraktimallin sovitukset MMS-protokollaan

*Kirjoita kuinka ylempi ACSI sovitetaan MMS-protokollan palveluiksi ja tietotyypeiksi standardin IEC 61850-8-1 osuuden mukaan. Tähän myös miten raportointi toimii MMS-protokollan päällä.*

#### 2.2.1 MMS-protokolla

*Selitä lyhyesti mikä on MMS-protokolla ja vähän sen tietotyypeistä. Tämän tarkoitus on pohjustaa tulevaa IEC 61850 abstraktien olioiden (ACSI) sovitusta tämän protokollan päälle.*

## 2.3 Advanced Message Queuing Protocol

*Kirjoita tähän AMQP määrittävästä standardista, mikä sen tarkoitus on ja mihin sitä voidaan käyttää.*

### 2.3.1 Viestien välitysmekanismit

*Mitä mekanismeja AMQP tarjoaa viestien välittämiseen osapuolille. Näitä on jono, reititys suoraan osapuolien välillä ja viestin julkaisu ja tilaaminen.*

### 2.3.2 Tilaus ja julkaisu -mallin osat

*Kirjoita tähän AMQP tarjoamista viestien julkaisu ja tilaus -mallin osista osapuolten kesken. Kerro mitä eri osat tekevät ja mikä niiden tehtävä viestien välittämisessä on. Englanniksi osia ovat esim. exchange, queue, publisher ja consumer.*



### 3. ALKUTILANNE

*Pohjista miksi suunniteltava ohjelmisto tarvitaan toteuttaa yritykseen johon työn teen. Alustava suunniteltu ohjelmiston toteutus olisi tilata IEC 61850 -standardin määrittämiä raportteja ja muokata ne uuteen muotoon ja julkaista ne eteenpäin tilaavalle ohjelmalle käyttäen AMQP-standardin määrittämää viestintää. Jonon tilaava asiakasohjelmisto voi olla mikä tahansa muu ohjelmisto. Viestien lopullinen muoto voisi olla JSON.*

Nykyisin sähköasemilla älykkäillä elektronisilla laitteilla (engl. *Intelligent Electronic Devices*, **IED**) asemilla voidaan toteuttaa tuhansia eri datapisteitä, jotka kuvaavat aseman toiminnallisuutta ja konfiguroitavuutta. Tämän konfiguroitavuuden ansiosta IED:tä voidaan asemalla käyttää erilaisina sähkölaitteina, kuten sulakkeina tai muuntajina. IEC 61850 -standardin abstraktit datamallit määrittävät IED:n datapisteiden rakenteet, muodot ja tyytit. Standardin mukaan erillisistä datapisteistä voidaan muodostaa haluttuja datajoukkoja (engl. *data set*). Datajoukkot ovat helppo tapa kuvata halutut tai tärkeät datapisteet yhdeksi yhteinäiseksi joukoksi. [2].

Asiakas-palvelin-arkkitehtuurissa asiakkaan on mahdollista tilata datajoukkojen sen hetkisiä arvoja IEC 61850 -standardin määrittäminä raportteina konfiguroitavilla parametreilla, jotka konfiguroidaan ennen tilausta. Arkkitehtuurissa asiakas tekee tilauksen palvelimelle (tässä tapauksessa IED), jonka jälkeen palvelin lähettää raportteja asiakkaalle automaattisesti, jonkin asiakkaan konfiguroiman ehdon täytyessä. Standardi määrittää kuinka raportteja voidaan esim. välittää TCP/IP-protokollan avulla. Yksi raportti sisältää mm. tietojoukon sen hetkisiä arvoja ja syyn raportin lähetykseen (esim. arvon muuttuminen). Palvelin ylläpitää tilausta kunnes asiakas lopettaa tilauksen tai yhteys osapuolten välillä katkeaa. Asiakas tilaa raportit konfiguroimalla palvelimella olevan erillisen raportointilohkon (engl. *Report Control Block*, **RCB**). Lohkolla voi konfiguroida mm. raporttien sisältämiä vaihtoehtoisia kenttiä ja erilaisia liipaisimia raporttien generointiin. Standardi määrittää että yhdellä RCB:llä voi olla vai yksi tietojoukko ja yksi tietojoukko voi olla viitattuna monessa eri RCB:ssä. Yhdessä IED:ssä voi olla määritetynä monta RCB:tä. Yhtä tilaavaa asiakasta kohden on yksi RCB instanssi. [1, s. 91–130].

Tulevissa kappaleissa pohjustetaan työn alussa olemassa olevan ohjelmiston arkkitehtuuria, mitkä olivat sen komponentit ja niiden toiminnallisuus. Tämän jälkeen pohditaan toteutuksen ongelmia, ja mitä työssä pyritään ratkaisemaan uudella toteutuksella. Asetettujen tutkimuskysymysten ja ongelmien kautta pyritään löytämään uudelle ohjelmiston arkkitehtuurille pohjaa ja ratkaisua siihen liittyviin päätöksiin.

### 3.1 Kokonaiskuva

*Kirjoita tähän osioon kokonaiskuva alkutilanteesta missä oltiin ennen työn aloittamista. Selvennä kuvilla alkutilanteen arkkitehtuuria.*

Työn aloitusvaiheessa oli jo toteutettuna ohjelmisto raporttien tilaukseen ja käsittelyyn. Tämä toteutus oli puutteellinen, ei helposti skaalautuva, ja huono suorituskvyylytään. Alkuperäinen ohjelmisto oli lähellä enemmän ensimmäistä prototyyppiä ennen todellista toteutusta. Työn tarkoituksena oli suunnitella ja toteuttaa uusi toteutus, joka ratkaisisi entisen ongelmakohdat.

Alkuperäisessä toteutuksessa asiakasohjelmisto oli toteutettu Ruby-ohjelmointikielellä. IEC 61850 -standardin määrittämien palveluiden ja tietorakenteiden toteutukseen käytettiin avoimen lähdekoodin libIEC61850-kirjastoa<sup>1</sup>. Kirjasto on ohjelmoitu C-kielellä ja sen avulla voidaan toteuttaa kumpikin palvelin- ja asiakasohjelmisto. Tässä toteutuksessa tarvittiin vain asiakasohjelman osuutta Ruby-osuuden toteutukseen. Kirjasto abstraktoi standardin määrittämiä palveluita ja tietorakenteita ohjelmoijalle helpoiksi funktioiksi ja C-kielen rakenteiksi. Normaalisti C-koodin funktioiden kutsuminen Rubysta suoraan ei ole mahdollista ilman erillistä liitosta. Seurauksena Rubyyn oli tehty laajennos libIEC61850-kirjastoon käyttäen Rubylle saatavaa ruby-ffi -kirjastoa<sup>2</sup> (engl. *Foreign Function Interface*, **FFI**). Liitoksen avulla libIEC61850-kirjasto voi hoitaa standardin vaatiman matalan tason toiminnan ja Ruby-ohjelmisto voi keskittyä vaadittuun toiminnallisuuteen.

Kirjasto toteuttaa raporttien vastaanoton palvelimelta erillisellä säikeellä. Säie käynnistetään kun asiakasohjelma asettaa funktion takaisinkutsuntaa varten raportin saapuessa ja aloittaa tilauksen. Asetettua funktiota kutsutaan asynkronisesti erillisestä säikeestä raportin saapuessa asiakkaalle. Takaisinkutsun suorituksen jälkeen, suoritus palaa takaisin säikeeseen.

### 3.2 Ratkaistavat ongelmat

*Kirjoita tähän mitä ongelmia edellisen toteutuksen kanssa on ja mitä yritään ratkoa. Mainitse suorituskvyykyongelmista Rubylla ja libiec61850-kirjastoa käyttäen.*

Työn alussa olevan ohjelmiston ongelmia oli mm. ei helposti skaalautuvuus, huono suorituskvyyky raporttien määrän ollessa suuri, eikä ohjelmisto tukenut kaikkia standardin määrittämiä toiminnallisuuksia. Ohjelmistoa voisi enemmän pitää ensimmäisen toteutuksen prototyyppinä. Ohjelman suoritusalueena käytettiin Linuxia.

Ohjelmisto pystyi tilaamaan ja vastaanottamaan raportteja yhdeltä IED:ltä ja siinä monelta määritellyltä RCB:ltä. Ohjelmisto prosessoi ja tallensi raportteja tietokantaan muuta

<sup>1</sup><http://libiec61850.com>

<sup>2</sup><https://github.com/ffi/ffi>

käyttöä varten. Tilanteessa, jossa raportteja tilaavassa järjestelmässä on monta osaa, jotka kaikki tarvitsevat raporttien tietoja reaaliajassa. Joutuvat eri osat tässä tilanteessa kyselemään tietoja tietokannasta, ilman erillistä tietoa niiden saapumisesta. Tämä aiheuttaa turhaa kuormaa tietokannalle ja tietojen saaminen reaaliajassa ei ole mahdollista. Myöskin jos komponentti tarvitsee tietyn tyyppin raportteja, ei kaikkea tietoa, ongelma on sama.

Ohjelmiston suorituskyky paikoin raporttien määrän ollessa suuri aiheutti ongelmia. Syynä Rubyn toteutuksessa oli oletustulkissa (*CRuby*) oleva globaali lukitus (engl. *Global Interpreter Lock*, **GIL**). Vaikka Rubyn säie on oma käyttöjärjestelmän tarjoama säie, GIL estää säikeiden yhtäaikaisen suorituksen ja vain yksi säie on suorituksessa kerrallaan [3, s. 131–133]. Linux-pohjaisella käyttöjärjestelmällä *libIEC61850*-kirjaston laitteistoabstraktiokerros (engl. *Hardware Abstraction Layer*, **HAL**) käyttää POSIX-säikeitä [4]. Linux-käyttöjärjestelmän säikeet ovat suorituksessa yhtä aikaa ja moniytimisellä prosessoreilla asioita tapahtuu samalla ajan hetkellä. Nyt raportin saapuessa, C-prosessin säikeen suoritus kutsuu takaisinkutsuntaan asetettua funktiota, joka on implementoitu Rubyn puolella. On funktion suoritus GILin alaista suoritusta. Ruby-prosessin myös suorittaessa muuta toimintaa takaisinkutsujen välissä, on Rubyn suorituskyky ohjelmiston pullonkaulana raporttien määrän ollessa tiheää.

*Kirjoita tähän vielä ongelmasta kun tilataan monta RCB:tä. Raporttien tullessa Rubyn puolelle, ei Rubyn muu koodi saa tilattua loppuja RCB:tä kirjaston lukitusten takia. Ja yhteys aikakatkeaa tämän takia. Selitä lukituksista tarkemmin ja myös liitä pätkiä libIEC61850-kirjaston koodista. Syyn selityksen voi siirtää muualle. Kirjoittaa vain että on ongelma, ja selvitys miksi, muualla.*

### 3.3 Tutkimuskysymykset

*Esitä tässä työlle asetettuja tutkimuskysymyksiä. Näitä voisi olla esim. seuraavat:*

- *Mikä on syynä huonoon suorituskykyyn alkutilanteen toteutuksella?*
- *Kuinka suorituskyky paremmaksi verrattuna nykyiseen toteutukseen?*
- *Mitkä ohjelmiston arkkitehtuurin suunnittelumallit (design patterns) olisivat sopivia tämän kaltaisen ongelman ratkaisemiseen? Mitä niistä pitäisi käyttää ja mitä ei?*
- *Mikä olisi sopiva lopullisen prosessoidun tiedon muoto?*
- *Kuinka järjestelmä hajautetaan niin että tiedon siirto eri osapuolten välillä on mahdollista ja joustavaa (push vs pull, message queue jne.)?*

## 4. SUUNNITTELU

*Kirjoittaa tähän kuinka toteutettava arkkitehtuuri suunniteltiin ja kuinka päätöksiin päädyttiin. Kirjoitusta myös miten tilattuja raportteja käsitellään ja kuinka niitä julkaistaan eteenpäin. Tarkoituksena olisi saada raportit nykyaikaiseen JSON muotoon.*

### 4.1 Suorituskyvyn parantaminen

*Miksi entisen toteutuksen suorituskyky on huono ja mitä voitaisiin tehdä sen parantamiseksi. Kirjoita vaikutuksista tähän ja mihin tuloksiin päädyttiin.*

### 4.2 Järjestelmän hajautus

*Lähde erilaisista hajautuksista (pull vs push, message queue) ja päättää mikä sopii tähän toteutukseen parhaiten ja miksi.*

### 4.3 Ohjelmiston parametrisointi

*Kirjoita mitä asiakasohjelman pitää tehdä jotta raportit saadaan tilattua ja mitä parametrejä ohjelmisto tarvitsee toimiakseen. Kuinka käyttäjä kontrolloi ohjelman eri asetuksia?*

### 4.4 Arkkitehtuurin suunnittelu

*Määritä ohjelman tarkempaa arkkitehtuuria mitä voidaan käyttää asetettujen ja yllämainittujen asioiden saavuttamiseen ja tarkentamiseen. Mitä jos käyttäjä tilaa monta viestiklokkia, niin missä järjestyksessä asiat tehdään jne.*

### 4.5 Prosessoidun viestin muoto

*Kirjoita tähän mihin muotoon viestit lopussa tallennetaan esim. JSON. Miksi tähän valintaan päädyttiin. Kerro myös kuinka raportin alkuperäistä rakennetta muokattiin uuteen muotoon sopivaksi.*

## 5. TOTEUTUS

*Kirjoita tähän osioon siitä kuinka suunniteltu arkkitehtuuri toteutettiin ja millä tekniikoilla. Tämä osio käyttää lyhyitä koodiesimerkkejä hyväkseen selittämään lukijalle kuinka toteutus tehtiin, jotta lukija voisi itse toteuttaa samanlaisen ohjelmiston.*

### 5.1 Ohjelmiston toteutuksen valinta

*Kirjoita tähän miksi päädyttiin tietynlaiseen ohjelmiston toteuttamiseen. Työssä on mietitty komentorivipohjaista toteutusta. Lisäksi mille alustalle ohjelmisto suunnitellaan Windows vai Linux.*

### 5.2 Kielen valinta

*Kirjoita tähän mikä kieli valittiin toteutuksen tekemiseen ja miksi tämä. Alustava suunnitelma on toteuttaa C-kielillä.*

### 5.3 RabbitMQ

*Kirjoita tähän RabbitMQ toteutuksesta. Kirjasto toteuttaa AMQP-standardin määrittämiä eri viestintämalleja. Kerro kuinka sitä hyödynnetään tässä työssä ja vähän sen että mitä vaatii.*

### 5.4 Käytettävät kirjastot

*Kirjoita tähän erilaisista kirjastoista mitä toteutukseen valittiin ja miksi. Alaotsikoita voi lisätä jos toteutukseen tarvitaan muita kirjastoja.*

#### 5.4.1 libiec61850

*IEC 61850 -standardin toteuttava C-kirjasto joka tekee raskaan työn standardin määrittämien palveluiden toteuttamiseen ja muodostamiseen. Kirjasto tarjoaa rajapinnat serveri- ja asiakasohjelmiston toteuttamiseen, mutta vain asiakasohjelmiston rajapintoja käytetään. Kirjasto tarjoaa myös rajapinnat haluttujen raporttien tilaamista varten. Kirjaston nettisivu täältä: <http://libiec61850.com/libiec61850/>.*

### 5.4.2 rabbitmq-c

*RabbitMQ:n rajapinnan toteuttava kirjasto C-kielen ohjelmille. Kirjastolla voidaan toteuttaa julkaisevia ja tilaavia ohjelmistoja. Kirjastosta käytetään julkaisevan puolen toteutusta. Kirjasto löytyy täältä: <https://github.com/alanxz/rabbitmq-c>.*

### 5.4.3 JSON-formatointi

*Joku kirjasto JSON formatointiin C-kielelle. Näkyy olevan parikin vaihtoehtoa. Perustele tähän valinta ja miksi.*

## 5.5 Jatkokehitys

*Kirjoita tähän ideoita mitä jää jatkokehitykseen ja mitä ohjelmistossa on puutteita tai mitä jäi tekemättä.*

## 6. ARVIOINTI

*Kirjoitta tähän arviota työn tuloksista.*

## 7. TULOKSET

*Kirjoita tähän lopputuloksen analysoinnista ja peilaa saatuja tuloksia työlle alussa asetettuihin kysymyksiin. Mitä jäi saavuttamatta, mitä saavutettiin ja miten hyvin? Mitä olisi voinut parantaa? Voi jakaa aliotsikoihin jos tarvetta.*



## 8. YHTEENVETO

*Kirjoita tähän ensin arviointi ja yhteenveto työstä ja sen lopputuloksista. Mitä hyötyjä työnantaja työstä saa ja jatkokehitysideoita. Mitä työssä meni hyvin ja mitä olisi voinut tehdä toisin?*

*Kommentteja työtä aloittaessa:*

- *Olisiko hyvä, että lähdet työssäsi erilaisista hajautus paradigmoista (push vs pull; message queue), perustelet valintasi ja sitten menet suunnitteluun ja toteutukseen?*
- *Ja olisi hyvä, että työ perustelee miksi tuota MQ arkkitehtuuria yleensä (ja rabbitMQ:ta) käytetään.*

*Things to do now:*

- *Laittaa aihe hyväksyntään.*
- *Lähde kirjoittamaan teoriaa ja ennen sitä yleistä tasoa missä ollaan. Yleinen korkea taso sen takia, että lukija ymmärtää mistä edes on kyse. Pidä koko ajan kirjoittaessa mielessä top-down lähestymistapa! Erittäin tärkeä!!!*
- *Loppu otsikoida niin että ensin on tulokset, niiden arviointi ja yhteenvedo mainituissa järjestyksessä.*
- *Kirjoittaessa miettiä asioita mistä kirjoitetaan ja pitää kontekstista kiinni.*
- *Pidä lauseet simppelineinä ja helppolukuisina! Älä turhaan vaikeuta hommaa lukijalle ja se ei tuo työhön yhtään mitään lisäarvoa! Todella tärkeä asia ajatella! Jos lause käsittää monta asiaa, pilko se pienempiin erillisiin lauseisiin.*
- *Muihinkin lähteisiin voi viitata kuin tieteellisiin. Toki yritä löytää tieteellisiä julkaisuja mahdollisuuksien mukaan. Osoittaa että olet perehtynyt asiaan paremmin.*
- *Kun kirjoitat asiaa esim. että entisessä ohjelmassa oli ongelma että ei skaalaudu helposti tai on huono suorituskyky. Kerro mistä johtopäätös tulee. Tämä ei ole lukijalle selvää tietoa.*
- *Teorien ja yleisen osuuden kirjoittamisen jälkeen, sovi palaveri Karin kanssa.*
- *Työn otsikko on hyvä, ei tarvitse olla erikseen "ohjelmallisesti"sanaa.*
- *Työn päätason otsikoita laittaa enemmän kuvaavimmiksi kuin "Alkutilanne"ja "Teoria".*
- *Käytä työssä viesti sanaa raportin sijaan. Tuo lukijalle esille että se tarkoittaa standardin mukaisia raportteja.*

*Huomioituja asioita toisten dipoissa:*

- *Tärkeät sanat esitellään tekstissä ensimmäisen kerran kursiivilla painottamisen takia. Tämän jälkeen ei enää samaa sanaa kursivoida.*
- *Todella paljon erilaisia lähteitä käytetty! Blogiposteja, kirjoja, ja tapahtumien kirjoituksia (IEEE). On myös nettisivuja käytetty lähteenä kun mainitaan esim. Git ja jotain muita sivuja. Nämä tietysti voi olla myös alaliitteenä sivulla.*
- *Tosi hyvin kirjoitettu! Todella selkeää tekstiä ja etenee hyvin ja on lukijalle ystävällinen.*

- *Johdanto on pilkottu otsikoihin työn alkutilanteen selvittämiseksi hyvin ennen teoriaa. Ja teoriaosuus alkaa joustavasti johdannon jälkeen järkevästi.*
- *Kun listataan tekstiä, sana on ensin kurstiivilla ja on selitetty asiaa. Kohta loppuu puolipisteeseen (;). Tämän jälkeen jatkuu pienellä seuraava aihe ja päättyy myös puolipisteeseen. Viimeinen kohta alkaa myös pienellä, mutta päättyy pisteeseen normaalisti. Seuraava kappale alkaa normaalisti. Listassa lauseet muokkautuvat yhteen esim. käyttäen ja sanaa.*
- *Kysymys teoriassa mikä työssä on jäljellä oli kirjoitettu kurstiivilla.*

## LÄHTEET

- [1] IEC 61850-7-2 Communication networks and systems for power utility automation - Part 7-2: Basic information and communication structure - Abstract communication service interface (ACSI), International Electrotechnical Commission, International Standard, Aug. 2010, 213 p. Saatavissa (viitattu 16.5.2018): <https://webstore.iec.ch/publication/6015>
- [2] R.E. Mackiewicz, Overview of IEC 61850 and Benefits, 2006. Saatavissa (viitattu 14.5.2018): <https://ieeexplore.ieee.org/document/4075831>
- [3] R. Odaira, J.G. Castanos, H. Tomari, Eliminating Global Interpreter Locks in Ruby Through Hardware Transactional Memory, SIGPLAN Not., Vol. 49, Iss. 8, Feb. 2014, pp. 131–142. Saatavissa (viitattu 16.5.2018): <http://doi.acm.org/10.1145/2692916.2555247>
- [4] Official repository for libIEC61850, the open-source library for the IEC 61850 protocols <http://libiec61850.com/libiec61850>, GitHub verkkosivu. Saatavissa (viitattu 17.5.2018): <https://github.com/mz-automation/libiec61850>
- [5] B.D. Stockton, Design Guide for Rural Substations, United States Department of Agriculture, June 2001, 763 p. Saatavissa (viitattu 25.5.2018): [https://www.rd.usda.gov/files/UEP\\_Bulletin\\_1724E-300.pdf](https://www.rd.usda.gov/files/UEP_Bulletin_1724E-300.pdf)