SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Antti Mäenpää

Tiedonkeruujärjestelmä taajuusmuuttajille

Opinnäytetyö
Kevät 2022
Automaatiotekniikan tutkinto-ohjelma

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Automaatiotekniikka

Suuntautumisvaihtoehto: Sähköautomaatio

Tekijä: Antti Mäenpää

Työn nimi: Tiedonkeruujärjestelmä taajuusmuuttajille

Ohjaaja: Marko Hietamäki

Vuosi: 2022 Sivumäärä: 47

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Pesmel Oy:lle tiedonkeruujärjestelmä, jonka avulla tietoa voidaan kerätä ja tallentaa ABB:n valmistamilta taajuusmuuttajilta. Tehtävänä oli suunnitella järjestelmälle ohjelmistoarkkitehtuuri ja toteuttaa sen pohjalta tiedonkeruujärjestelmä. Työn tavoitteena oli myös selvittää toteutetun tiedonkeruujärjestelmän tietoverkon kuormittavuus.

Toteutetun tiedonkeruujärjestelmän tarkoituksena on tallentaa dataa myöhempää tarkastelua ja analysointia varten. Tallennettua dataa hyödynnetään vikatilanteiden selvittämisessä ja osana ennakoivaa kunnossapitoa. Tallennettua dataa on tarkoituksena hyödyntää myös automaatiosuunnittelun apuna, kun automaatiokomponentteja mitoitetaan.

Työn teoriaosuudessa esitellään taajuusmuuttajien rakenne ja toimintaperiaate yleisesti. Teoriaosuudessa kerrotaan myös OPC-arkkitehtuureista, Java-ohjelmointikielestä, Relaatiotietokannoista ja SQL-ohjelmointikielestä. Teoriaosuuden lopussa esitellään myös työssä käytetyt työkalut ja ohjelmat.

Työn tuloksena saatiin tiedonkeruujärjestelmä, jonka avulla haluttujen parametrien arvoja voidaan lukea ABB:n taajuusmuuttajilta ja tallentaa niitä tietokantaan myöhempää analysointia varten. Työhön kuului myös tutkimusosa, jossa onnistuttiin selvittämään toteutetun tiedonkeruujärjestelmän tietoverkon kuormitus. Tutkimustulosten avulla järjestelmän osien toteutustapoja optimointiin ja verkon kuormitusta saatiin pudotettua selvästi.

Työn toimeksiantaja oli työhön tyytyväinen. Toteutettu tiedonkeruujärjestelmä täytti sille asetut tavoitteet. Toteutettua tiedonkeruujärjestelmää tullaan testaamaan lisää toimeksiantajan tulevassa projektissa. Työn lopussa pohditaan kehityskohtia toteutetulle järjestelmälle.

[1] Asiasanat: tiedonhankinta, ABB, Java, tietokannat,

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Automation Engineering

Specialisation: Electrical Automation

Author: Antti Mäenpää

Title of thesis: Data collection system for frequency converters

Supervisor: Marko Hietamäki

Year: 2022 Number of pages: 47

The thesis was ordered by Pesmel Oy. The objective of the thesis was to design and develop a data collection system that can store data from ABB frequency converters. The objective was to design a software architecture for system and implement the data collection program based on it. The objective was also to figure out network load of data collection system.

The purpose of the implemented data collection system is to store data for later review and analysis. The stored data can be used to diagnose faults and as a part of preventive maintenance. The stored data can also be utilized in automation designing when sizing automation components.

The beginning of the theoretical part presents the structure and operating principle of frequency converters in general. The theory part also presents OPC architectures, Java programming, Relational databases, and SQL programming. The tools and software used in the work are presented at the end of the theory part.

As the result of the thesis there was a data collection system that can read and store data from ABB frequency converters. Another goal of the research was to figure out the network load of the data collection system and it was achieved successfully. The data collection system was optimized based on the research results and consequently, the network load reduced significantly. At the end of the thesis, some development ideas for the data collection system were considered.

The client was satisfied with the work. The implemented data collection system met the objectives set by the client. The system will be further tested in the client’s future project.

[1] data acquisition: ABB, Java, databases

SISÄLTÖ

Opinnäytetyön tiivistelmä 2
Thesis abstract 3
SISÄLTÖ 4
Kuva-, kuvio- ja taulukkoluettelo 6
Käytetyt termit ja lyhenteet 8
1 JOHDANTO 9
1.1 Pesmel Oy 9
1.2 Työn tausta 9
1.3 Työn tavoite 9
2 TAAJUUSMUUTTAJAT 10
3 OPC 12
3.1 OPC Data Access (OPC DA) 13
3.2 OPC Unified Architecture (OPC UA) 13
4 JAVA 15
4.1 Johdanto Javaan 15
4.2 Java-ohjelmointikielen taustaa 15
4.3 Java-ohjelmointikielen ominaisuudet tiivistettynä 15
4.4 Ohjelman kirjoittaminen ja kääntäminen 17
4.5 Java JDBC 18
5 TIETOKANTA 20
5.1 Relaatiotietokanta 21
6 SQL-KIELI 23
6.1 Johdanto SQL-kieleen 23
6.2 SQL-kielen historiaa 23
6.3 SQL-ohjelmointi 23
7 TYÖSSÄ KÄYTETYT TYÖKALUT 25
7.1 ABB Drive Composer -ohjelmisto 25
7.2 KEPServerEX-ohjelmisto 26
7.3 IntelliJ IDEA -ohjelmointiympäristö 26

7.4 SQL Server Management Studio 27
7.5 Wireshark 28
8 LÄHKÖKOHTA 30
8.1 Ohjelman suunnittelu 30
8.2 Järjestelmän hyödyt 31
9 JÄRJESTELMÄN TOTEUTUS 32
9.1 ABB Drive Composer Pro -ohjelmiston konfigurointi 33
9.2 KEPServerEX-ohjelman käyttö 35
9.3 Java-ohjelma 41
9.4 Microsoft SQL Server -tietokannan toteutus 42
10 YHTEENVETO JA POHDINTA 44
LÄHTEET 46

Kuva-, kuvio- ja taulukkoluettelo

Kuva 1. ABB:n valmistama taajuusmuuttaja. 10

Kuva 2. Taajuusmuuttajan lohkokaavio. 11

Kuva 3. Drive Composer Pro -ohjelmiston näkymä. 25

Kuva 4. KEPServerEX-kommunikointialusta. 26

Kuva 5. IntelliJ IDEA -ohjelmointiympäristö. 27

Kuva 6. Microsoft SQL Server Management Studio. 28

Kuva 7. Wireshark-verkkoanalyysityökalu. 29

Kuva 8. Yhteys muodostettu taajuusmuuttajaan Drive Composer Pro – konfigurointityökalulla. 34

Kuva 9. Yhteyden muodostaminen Drive Composer PRO:n OPC DA -palvelimelle. 35

Kuva 10. Taajuusmuuttajalta kerättävät parametrit. 36

Kuva 11. KEPServerEX-alustan luoma kansiorakenne OPC-tageille. 37

Kuva 12. Wireshark-työkalulla mitattu verkon kuormitus käyttäen useita OPC-ryhmiä. 38

Kuva 13. Verkon kuormituksen mittaustulos useilla OPC-ryhmillä. 38

Kuva 14. Selkeytetyt OPC-tagiryhmät KEPServerEX-kommunikointialustalla. 39

Kuva 15. Wireshark-työkalulla mitattu verkonkuormitus käyttäen laitekohtaisia OPC-ryhmiä. 39

Kuva 16. Verkon kuormituksen mittaustulos laitekohtaisilla OPC-ryhmillä. 40

Kuva 17. Verkon kuormituksen mittaustulos Wireshark-työkalulla. 40

Kuva 18. Verkon kuormituksen mittaustulokset Wireshark-työkalulla. 41

Kuva 19. Taajuusmuuttajista luotu lista Java-ohjelmaan. 41

Kuva 20. Yhteyden muodostaminen tietokantaan Java-ohjelman JDBC-rajapintaa käyttäen. 42

Kuva 21. Tietokantaan tallennettua dataa taajuusmuuttajilta. 43

Kuvio 1. OPC-kommunikointimalli. 12

Kuvio 2. OPC DA -kommunikointimalli. 13

Kuvio 3. OPC UA -kommunikointimalli. 14

Kuvio 4. Java-ohjelman muodostaminen. 18

Kuvio 5. Java JDBC -kommunikointimalli tietokantaan. 19

Kuvio 6. Tietokannan yleinen rakenne. 20

Kuvio 7. Relaatiomalli myyntitapahtumista. 22

Kuvio 8. Yleisimmät SQL-komennot. 24

Kuvio 9. Tiedonkeruujärjestelmän ohjelmistoarkkitehtuuri. 33

Kuvio 10. ABB:n taajuusmuuttajan ja Drive Composer Pron välinen kommunikointimalli. 34

Taulukko 1. Osa Java SE-kehitysympäristön komponenteista 17

Taulukko 2. Tietokannan taulun rakenne. 43

Käytetyt termit ja lyhenteet

Lähdekoodi	Ohjelmointikielen syntaksin eli sääntöjoukon mukaista ohjelman toimintaa kuvaavaa tekstiä.
Ohjelmointirajapinta	Määritelmä, jonka mukaan ohjelmat tekevät pyyntöjä ja vaihtavat tietoja keskenään.
PLC	Ohjelmoitava logiikka (programmable logic controller) jonka avulla voidaan ohjata automaatiojärjestelmää ja siihen liitettyjä laitteita.
Suodatuskuristin	Suodatuskuristimen tarkoituksena on pitää välipiirin virta tasaisena.
Syntaksi	Sääntöjoukko siitä miten ohjelmakielen eri elementeistä muodostetaan lauseita ja ohjelmia.
Tyristorisuuntaaja	Suuntaaja, joka koostuu tyristoreista, joita ohjataan ohjauspulssin avulla.

1 JOHDANTO

1.1 Pesmel Oy

Asiakastiedon (i.a.) mukaan Pesmel Oy on vuonna 1978 perustettu automaatioalan yritys. Yrityksen päätoimipaikka sijaitsee Kauhajoella. Pesmel Oy:n liikevaihto vuonna 2020 oli 40,5 miljoonaa euroa. Yritys työllistää noin 140 henkilöä.

Pesmelin (i.a.) mukaan yritys toimittaa automaatiojärjestelmiä ympäri maailmaa pääasiassa sellu-, paperi-, metalli-, ja rengasteollisuuteen. Yrityksellä on yli 40 vuoden kokemus automaatioratkaisujen toimittamisesta erilaisiin tuotantolaitoksiin. Yrityksellä on myös tytäryhtiöt Pohjois- Amerikassa, Euroopassa ja Aasiassa.

1.2 Työn tausta

Yritys ei tällä hetkellä tallenna yksittäistäisten automaatiokomponenttien dataa myöhempää analysointia varten. Tämän opinnäytetyön tarkoituksena on selvittää, miten ABB:n taajuusmuuttajilta voidaan kerätä ja tallentaa dataa myöhempää tarkastelua varten. Automaatiojärjestelmän vikatilanteiden selvittäminen on helpompaa, mikäli yksittäisiltä laitteilta tallennettua dataa voidaan tarkastella jälkikäteen. Taajuusmuuttajilta tallennettua dataa voidaan hyödyntää myös suunnitteluvaiheessa laitteiden mitoituksessa. Tallennettua dataa voidaan hyödyntää myös osana ennakoivaa kunnossapitoa.

1.3 Työn tavoite

Työn tavoitteena on suunnitella ja toteuttaa tiedonkeruujärjestelmä, jonka avulla ABB:n taajuusmuuttajilta voidaan kerätä ja tallentaa dataa myöhempää tarkastelua varten. Tiedonkeruujärjestelmän tulee muodostaa yhteys suoraan taajuusmuuttajaan ilman, että data kiertää PLC:n kautta. Ohjelmiston tulee olla itsenäinen tiedonkeruujärjestelmä, joka voidaan ottaa käyttöön automaatioverkossa olevalle Windows PC:lle. Työn tavoitteena on tutkia myös toteutetun tiedonkeruujärjestelmän verkon kuormittavuutta. Työn suunnittelussa tulee ottaa huomioon tiedonkeruujärjestelmän myöhempi laajentaminen.

2 TAAJUUSMUUTTAJAT

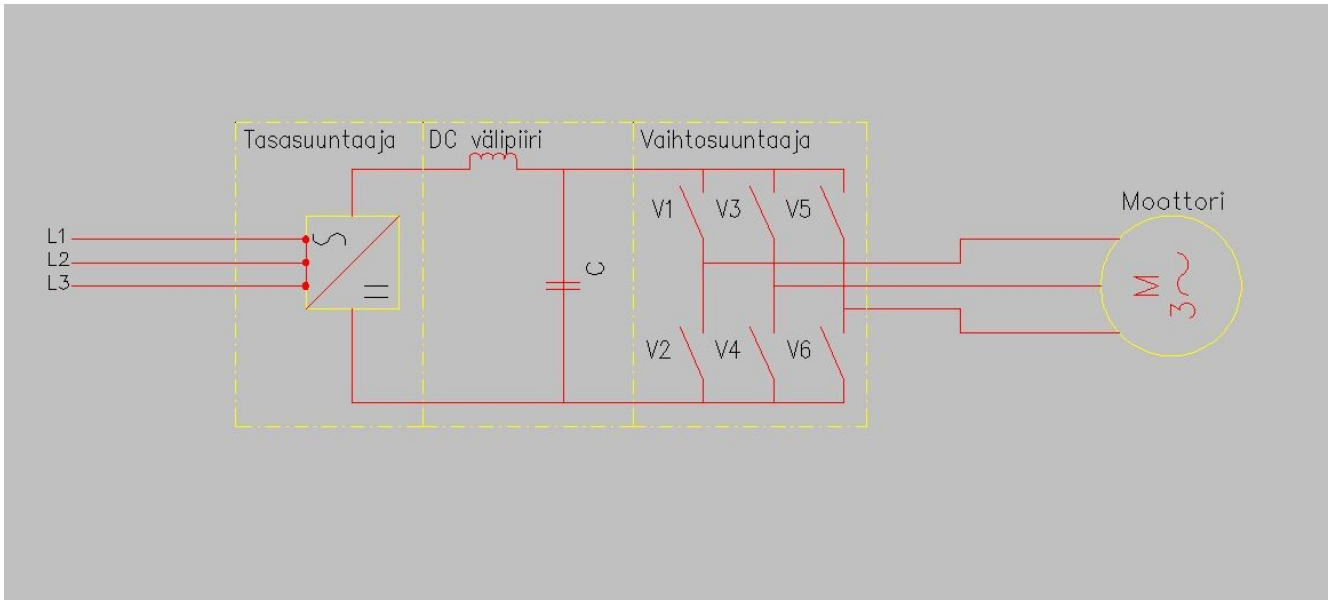
Orrberg ym. (2017, s.31–33) esittävät, että vaihtovirran ja jännitteen taajuutta muutetaan taajuusmuuttajalla. Taajuusmuuttajia käytetään vaihtovirtamoottorin vääntömomentin ja pyörimisnopeuden säätöön. Oikosulkumoottorin ja taajuusmuuttajan yhdistelmä on tehokas ja taloudellinen ratkaisu teollisuuden moottorisovelluksiin, joissa moottorin pyörimisnopeutta pitää pystyä säätämään (mts. 33). Taajuusmuuttaja kostuu kolmesta pääosasta, joita ovat tasasuuntaaja, välipiiri ja vaihtosuuntaaja (mts. 31). Kuvassa 1 esitellään ABB:n valmistamaa taajuusmuuttajaa.

Kuva 1. ABB:n valmistama taajuusmuuttaja.

Aura ja Tonteri (1996, 341) kertovat, että tasasuuntaaja on virtapiirin osa, joka muuttaa vaihtosähköenergian tasasähköksi. Taajuusmuuttajan tasavirtavälipiiri on sähköenergian varastointipaikka, josta tasasähkö vaihtosuunnataan vaihtosähkömoottorille. Kuvassa 2

havainnollistetaan taajuusmuuttajan rakennetta. Taajuutta voidaan säätää eri tavoin, joten välipiirin ja vaihtosuuntaajan rakenne jakavat taajuusmuuttajat eri tyyppeihin:

- Taajuuden muuttaminen suuntaajalla, jonka tasavirtavälipiirin virtaa säädetään (Aura & Tonteri, 1996, s. 458). Tällöin tyristorisuuntaaja ja suodatuskuristin muodostavat tasavirran, joka syötetään halutun suuruisina virtapulsseina vaihtosuuntaajan kautta vaihtosähkökoneen käämien läpi.
- Taajuuden muuttaminen suuntaajalla, jonka välipiirin jännitettä ohjataan (Aura & Tonteri, 1996, s. 461). Tällöin vaihtosuuntaaja muodostaa ohjaustavan määräämiä vaihtojännitepulsseja. Tätä tapaa käytettäessä syötetään moottorille vaihtosuuntaajalla suorakaidepulsseja, joiden amplitudia säädetään tasajännitepiirissä.
- Taajuuden muuttaminen suuntaajalla, jonka jännite on vakio (Aura & Tonteri, 1996, s. 462). Tällöin säätö tapahtuu säätämällä jännitepulssin suhteellista pituutta jakson aikana.



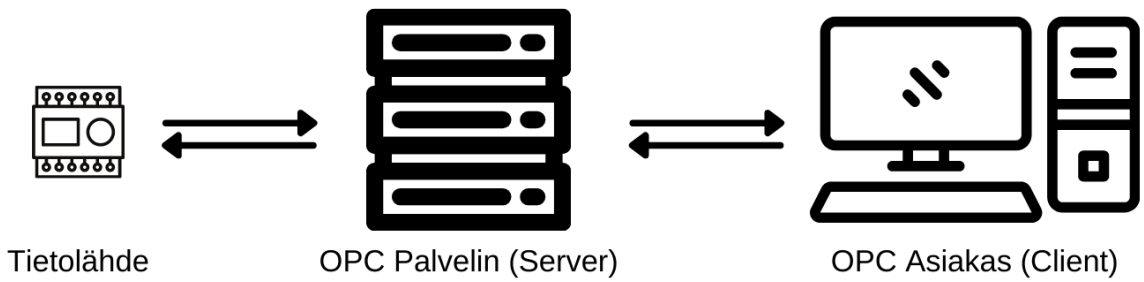
Kuva 2. Taajuusmuuttajan lohkokaavio.

3 OPC

OPC Foundationin (i.a.) mukaan OPC on yhteensopivuusstandardi luotettavaan ja turvalliseen tiedonvaihtoon teollisen automaation alalla. OPC-lyhenne tulee sanoista Open Platform Communications. Uusin OPC on alustariippumaton, sen avulla voidaan toteuttaa tiedonkulkua useiden eri valmistajien laitteiden välillä. OPC-säätiö vastaa standardin kehittämisestä ja ylläpidosta. OPC on alan laitetoimittajien, ohjelmistokehittäjien ja loppukäyttäjien kehittämä spesifikaatio. OPC:n spesifikaatiot määrittelevät käyttöliittymien, ohjelmistojen ja palvelimien rajapintojen välistä yhteyttä ja kommunikointia sekä mahdollistavat muun muassa reaaliaikaisen datan lukemisen, hälytysten ja tapahtumien seurannan ja pääsyn datahistoriaan.

OPC Foundationin (i.a.) mukaan OPC-standardi julkaistiin ensimmäisen kerran vuonna 1996. Sen tarkoituksena oli standardisoida laitteiden väliset yhteydet. Sen tuloksena syntyi kommunikointitapa, jonka avulla loppukäyttäjä voi toteuttaa eri laitteiden välistä kommunikointia saumattomasti OPC-yhteyden kautta. OPC-standardi oli aluksi rajoitettu vain Windows-käyttöjärjestelmään.

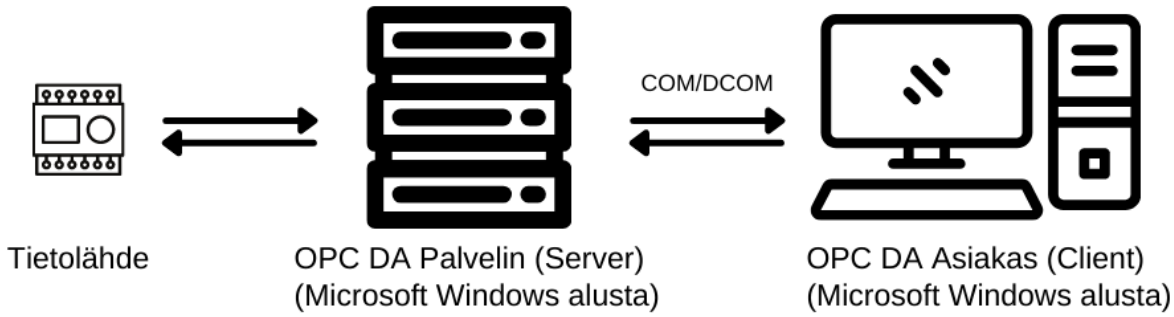
Kepwaren (i.a.-a) mukaan OPC-yhteys toimii asiakas-palvelin-kommunikaatiomallin mukaan. Tällöin vähintään yksi sovellus tai laite toimii dataa tarjoavana palvelimena ja toinen ohjelma tai laite dataa käyttävänä asiakkaana. OPC Routersin (i.a.) mukaan OPC-palvelin voidaan toteuttaa erillisenä ohjelmistona tai sulautettuna OPC-palvelimena laitteeseen tai koneen ohjaimeen. Kuviossa 1 havainnollistetaan OPC-kommunikointimallia.



Kuvio 1. OPC-kommunikointimalli.

3.1 OPC Data Access (OPC DA)

Matrikon (i.a.) kertoo, että OPC DA on ensimmäinen OPC-säätiön määrittelemä arkkitehtuuri. Ennen OPC DA -arkkitehtuuria eri valmistajien tuotteet vaativat ohjaimen, jonka avulla voitiin luoda yhteys kolmannen osapuolen laitteeseen. OPC Data Access (OPC DA) -arkkitehtuuri mahdollistaa reaaliaikaisen lukemisen ja kirjoittamisen eri laitteiden välillä. Schneider Electricin (i.a.) mukaan OPC DA -kommunikointi palvelimen ja asiakasohjelman välillä perustuu Microsoftin COM/DCOM2 -teknologiaan. OPC DA -palvelin kommunikoi tietolähteen kanssa. OPC DA -palvelimella olevaa dataa voidaan kirjoittaa ja lukea OPC DA -asiakasohjelman avulla. Kuviossa 2 havainnollistetaan OPC DA -kommunikointimallia.



Kuvio 2. OPC DA -kommunikointimalli.

3.2 OPC Unified Architecture (OPC UA)

OPC Foundationin (i.a.) mukaan OPC UA on vuonna 2008 julkaistu alustariippumaton arkkitehtuuri, joka yhdistää aiempien OPC-versioiden spesifikaatiot yhteen runkoon. OPC UA -yhteyttä voidaan käyttää jokaisessa käyttöjärjestelmässä eikä se ole Windows-riippuvainen kuten OPC DA. OPC Routersin (i.a.) mukaan aiempi OPC-standardin käyttämä DCOM-pohjainen kommunikaatio on korvattu OPC UA:ssa TCP/IP-, HTTP/HTTPS- ja SOAP-protokollilla.

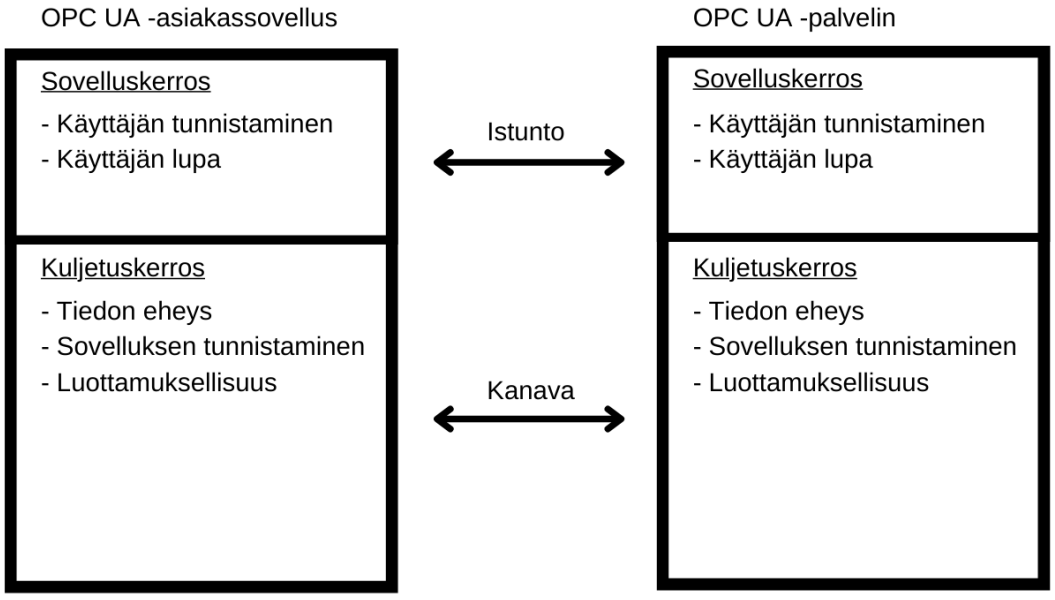
Real Time Automationin (i.a.) OPC UA -spesifikaatio määrittelee seuraavia tiedonkuljetusprotokollia, joita OPC UA -asiakasohjelmien tulee tukea:

- SOAP/http-siirtoprotokolla. HTTP on yhteydetön, tilaton, pyyntö- ja vastausmallilla toimiva protokolla, jota käytetään esimerkiksi aina kun avataan verkkosivu. SOAP on

XML-viestintäprotokolla, XML mahdollistaa mekanismin ohjelmoida viestejä muille sovelluksille.

- HTTPS-siirtoprotokolla. HTTPS on suojattu versio HTTP:stä. Se tarkoittaa, että asiakasohjelman ja palvelimen välinen viestintä on salattua.
- UA TCP -siirtoprotokolla. Tämä on yksinkertainen TCP-pohjainen protokolla. UA TCP soveltuu palvelinlaitteille, joilla ei ole XML- tai HTTP/SOAP-kuljetusmahdollisuutta. UA TCP perustuu binääriprotokollaan, joka mahdollistaa tiedonsiirron toteuttamisen myös matalan tason palvelimilla.

Real Time Automationin (i.a.) mukaan OPC UA -kommunikaatio on rakennettu kerroksittain. Varsinainen tiedonkuljetuskerros muodostuu jo aiemmin kerrotuista TCP/IP-, SOAP/http- ja HTTPS-siirtoprotokollista. Kuljetuskerroksen lisäksi kommunikaatio muodostuu asiakasohjelman luomasta kanavasta OPC UA -palvelimen ja asiakasohjelman välille. Kanava on todennettu yhteys asiakasohjelman ja palvelimen välillä. Kun kanava on muodostettu, asiakasohjelma luo istunnon OPC UA -palvelimen ja asiakasohjelman välille. OPC UA -kommunikointimallia asiakkaan ja palvelimen välillä havainnollistetaan kuviossa 3.



Kuvio 3. OPC UA -kommunikointimalli.

4 JAVA

4.1 Johdanto Javaan

Vesterholm ja Kyppö (2010, s. 21) esittävät, että Java on Sun Microsystemsin kehittämä olio-ohjelmointikieli, joka on suunnattu tietoverkkoihin. Se on rakennettu C++-ohjelmointikielen pohjalta. Javan tyypillisiä ominaisuuksia ovat sen olioperusteisuus, vahva tyypitys, C++-ohjelmointikieltä muistuttava syntaksi, viitesemantiikka ja web-ohjelmointituki. Javan alkuperäisenä ideana oli kehittää ohjelmointikieli, joka soveltuisi sulautettuihin järjestelmiin ja elektroniikkaan, kuten televisioihin, kelloihin, kahvinkeittimiin ja vastaaviin ympäristöihin.

4.2 Java-ohjelmointikielen taustaa

Vesterholm ja Kyppö (2010, s. 21) toteavat, että James Gosling loi vuonna 1991 Sun Microsystemsin suojissa ensin ohjelmointikielen nimeltä Oak. Oak-kielen kehittämiseen osallistuivat myös Jonathan Payne, Ed Frank, Chris Warth ja Patrick Naughton. Oak kehitettiin lähinnä sulautettuja järjestelmiä silmällä pitäen. Oak-ohjelmointikielen ajatuksena oli poistaa käännettävien ohjelmointikielien perusongelma, jolloin eri prosessorimalleille oli käännettävä ohjelma uudelleen.

Vesterholmin ja Kyppön (2010, s. 21) mukaan vastaavaa ideaa internetissä käytettäväksi ryhdyttiin ajamaan World Wide Webin yleistyessä. HotJava-selain julkaistiin vuonna 1994, jonka myötä myös ohjelmointikielen nimeksi tuli Java. Javan pohjalla olivat C- ja C++-ohjelmointikielet. Javasta päätettiin kehittää käyttöjärjestelmästä riippumaton, kevyt ja turvallinen ohjelmointikieli.

4.3 Java-ohjelmointikielen ominaisuudet tiivistettynä

Vesterholm ja Kyppö (2010, s. 23–24) esittävät, että Java on olio-ohjelmointikieli, jonka ohjelmoinnista käytetään luokkia ja olioita. Ohjelmaa kirjoittaessa käytetään ohjelmointikielen valmiita luokkakirjastoja. Javan lähdekoodi käännetään tavukoodiksi, joka ei ole riippuvainen laitteesta tai käyttöjärjestelmästä. Java-ohjelman suoritusvaiheessa tavukoodi ajetaan virtuaalikoneella (Java Virtual Machine, JVM). Tämä suoritustapa tekee siitä hitaamman kuin suoraan konekielelle käännetyn tavallisen ohjelman. Java Virtual Machineen on ohjelmoitu

laitelähteiset piirteet, minkä vuoksi JVM ohjelmoidaan erikseen jokaiselle käyttöjärjestelmälle tai prosessoriarkkitehtuurille. Tämän ansiosta samaa ohjelmaa voidaan ajaa esimerkiksi Windowsilla, Linuxilla ja Macintoshilla.

Vesterholmin ja Kyppön (2010, s. 24) mukaan Javassa on automaattinen roskien keruu, tämä tarkoittaa sitä, että olioiden varaama muisti vapautetaan automaattisesti olion elinkaaren päättyessä. Java on lisäksi turvallinen ohjelmointikieli, koska ohjelmoija ei pääse käsittelemään suoraan koneen muistia muistiosoitteilla operoiden. Javan vahvana puolena on myös ohjelman poikkeusten käsittely. Javassa ohjelman virhetilanteet ilmoitetaan aiheuttamalla poikkeus. Ohjelman kääntövaiheessa kääntäjä tarkistaa, että mahdollisille koodin poikkeuksille on kirjoitettu käsittely. Tällöin ohjelmoijan on pakko huomioida myös se, miten ohjelman virhetilanteissa toimitaan.

Vesterholmin ja Kyppön (2010, s. 24) mukaan Javan kotisivuilta ladattava kehitysympäristö on nimeltään JDK (Java SE Development kit). Tästä on myös välillä käytetty nimeä SDK (Software Development Kit). JDK sisältää työkalut Java-ohjelmien tekemiseen ja ajamiseen. JDK-ympäristön lisäksi Javan web-sivulta on saatavissa JRE (Java SE Runtime Environment). JRE sisältää vain Java-ohjelmien ajamisessa tarvittavat työkalut, joita ovat virtuaalikone ja web-selaimeen asentuva Java Plug-In.

Taulukko 1. Osa Java SE -kehitysympäristön komponenteista (perustuu Vesterholm & Kyppö, 2010, s. 25).

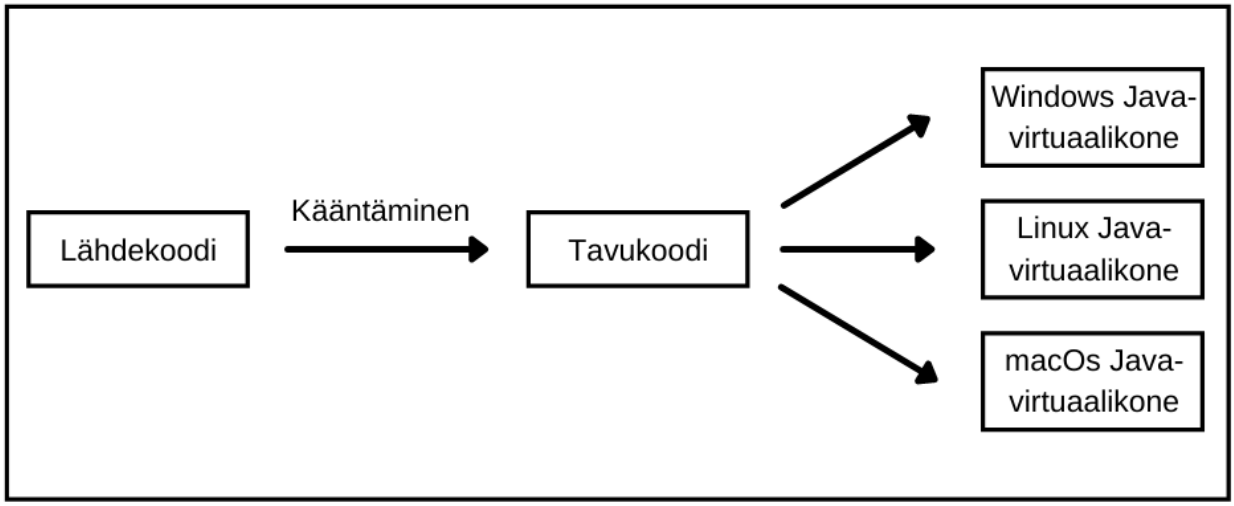
<table>
<tr><td rowspan="2">Kehitystyökalut ja rajapinnat</td><td>Virtuaali-kone, java</td><td>Kääntäjä, javac</td><td>Virheenjäl-jitin, JPDA</td><td colspan="2">javadoc</td></tr>
<tr><td colspan="4">Seuranta</td></tr>
<tr><td>Jakeluteknolo-giat</td><td colspan="2">Java Web Start</td><td colspan="3">Java Plug-In</td></tr>
<tr><td rowspan="2">Käyttöliittymä-komponentit</td><td colspan="2">Swing</td><td colspan="3">AWT</td></tr>
<tr><td>Ääni</td><td>Syöttötavat</td><td>Java 2D</td><td colspan="2">Saatavuus</td></tr>
<tr><td>Intregrointi-raja-pinnat</td><td>RMI</td><td>JDBC</td><td>JNDI</td><td>Skriptit</td><td>COBRA</td></tr>
<tr><td rowspan="3">Peruskirjastot</td><td colspan="2">Lokiin kirjaaminen</td><td colspan="3">Paikallistamistuki</td></tr>
<tr><td>Kokoelmat</td><td>Beand</td><td>Uusi I/O</td><td>JNI</td><td>XML</td></tr>
<tr><td>lang & util</td><td>Työkalut</td><td>Asetukset</td><td>Verkkotuki</td><td>Tietoturva</td></tr>
<tr><td colspan="6">Java-virtuaalikone</td></tr>
<tr><td>Ympäristöt</td><td>Windows</td><td>Linux</td><td>Mac OS X</td><td colspan="2">Muut</td></tr>
</table>

4.4 Ohjelman kirjoittaminen ja kääntäminen

Vesterholm ja Kyppö (2010, s. 34–35) toteavat, että Java-ohjelman lähdekoodi voidaan yleensä kirjoittaa millä tahansa tekstieditorilla. Java-lähdekooditiedosto on käännettävä ohjelmaksi ennen kuin se voidaan ajaa. Kääntäjä tarkistaa, että ohjelmakoodi on syntaksin mukaista. Mikäli ohjelma on syntaksiltaan oikein, kääntäjä tuottaa siitä tavukoodia sisältävän .class-päätteisen tiedoston (mts. 35). Kääntäjä luo ohjelman jokaiselle luokalle ja liittymälle oman tavukooditiedoston. Tavukoodi ei ole riippuvainen tietystä käyttöjärjestelmästä. Tavukoodin ajamista varten jokaiselle käyttöjärjestelmälle tarvitaan tavukoodia tulkitseva Java-virtuaalikone.

Vesterholm ja Kyppö (2010, s. 35–36) kertovat, että Java-virtuaalikone lukee ja tulkitsee tavukoodin sen arkkitehtuurin mukaiselle konekielelle. Ohjelmaa suorittaessa Javassa tapahtuu myös linkittämistä (mts. 36). Tavukooditiedosto sisältää tiedon siitä, mitä muita luokkia

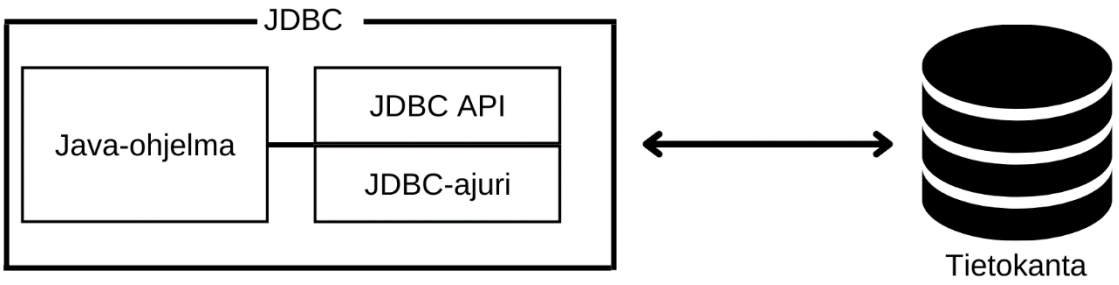
kyseinen luokka käyttää. Ohjelman suoritusvaiheessa virtuaalikone lataa tiedon perusteella tarvittavat luokat ja linkittää ne yhteen. Kuviossa 4 havainnollistetaan Java-ohjelman suorittamista.



Kuvio 4. Java-ohjelman muodostaminen.

4.5 Java JDBC

JavaTpointin (i.a.-a) JDBC on ohjelmointirajapinta, joka mahdollistaa Java-ohjelmille yhteyden ja erilaiset kyselyt tietokannasta. JDBC-nimitys tulee sanoista Java Database Connectivity. JDBC-ohjelmointirajapintaa voi käyttää mihin tahansa relaatiotietokantaan tallennettuun taulukkomuotoiseen tietoon. JDBC:n avulla tieto voidaan tallentaa, päivittää, poistaa ja hakea tietokannasta. Kuviossa 5 havainnollistetaan Java-ohjelman kommunikointia tietokantaan JDBC-ohjelmointirajapinnan avulla.



Kuvio 5. Java JDBC -kommunikointimalli tietokantaan.

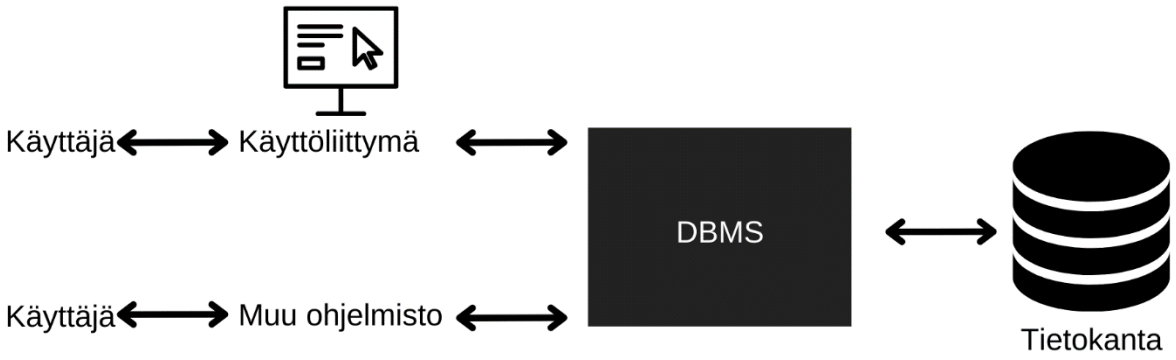
IBM:n (i.a.-a) mukaan JDBC-ohjelmointirajapinta kostuu joukosta Java-ohjelmointikielellä kirjoitettuja rajapintoja ja luokkia. JDBC-vakiospesifikaatio mahdollistaa Java-ohjelman yhteyden muodostuksen mihin tahansa tietokantajärjestelmään, kunhan tietokannalla on tietokannanhallintajärjestelmä (DBMS). JDBC-ajurit voidaan jakaa neljään eri tyyppiin:

- 1-tyyppi on JDBC-ODBC Bridge -ajuri, joka kääntää JDBC-kyselyt Microsoft ODBC-kyselyiksi ja välittää ne ODBC-ohjaimelle.
- 2-tyyppi on NATIVE-ajuri, joka muuntaa JDBC-kyselyt DBMS-kohtaisiksi asiakassovelluskutsuiksi.
- 3-tyyppi on JDBC-NET-ajuri, joka lähettää JDBC-kyselyt verkkopalvelimelle, joka kääntää kutsut DBMS-kohtaiseksi verkkoprotokollaksi.
- 4-tyyppi on Native-protocol-ajuri, joka kääntää JDBC-kyselyt suoraan DBMS-kohtaiseksi verkkoprotokollaksi. Tämän ajurin avulla Java-ohjelma voi muodostaa yhteyden suoraan tietokantapalvelimeen.

5 TIETOKANTA

Microsoftin (i.a.-a) mukaan tietokanta on tiedon keräämiseen ja järjestämiseen käytettävä työkalu. Tietokannat voivat sisältää esimerkiksi henkilöihin, tuotteisiin tai tilauksiin liittyviä tietoja. Monet tietokannat saattavat olla aluksi tekstinkäsittelyohjelmalla luotuja luetteloita ja laskentataulukoita. Oraclen (i.a.-a) mukaan tietokanta on kokoelma tietoa tai dataa, joka on tyypillisesti tallennettuna tietokonejärjestelmään.

Taylorin (2010, s. 8) mukaan tietokantaa ohjaa yleensä tietokannan hallintajärjestelmä (DBMS). Tietokannan hallintajärjestelmä sisältää joukon ohjelmia, joita käytetään tietokannan ja sen sovellusten käsittelyyn ja hallintaan. Tietokanta on pohjimmiltaan rakenne tiedon säilyttämiseen ja DBMS on työkalu, jota käytetään tietokannan rakentamiseen ja tietokannan sisältämien tietojen käyttämiseen, muokkaamiseen ja hallitsemiseen. Kuviossa 6 esitellään tietokannan yleistä rakennetta.



Kuvio 6. Tietokannan yleinen rakenne.

Oracle (i.a.-a) esittää, että nykyisin käytössä olevista tietokantatyypeissä tieto on tallennettu tyypillisesti riveinä ja sarakkeina taulukkosarjoissa tietojen käsittelyn tehostamiseksi. Tieto on tällöin helposti käytettävissä, hallittavissa, muokattavissa ja päivitettävissä. Useimmat tietokannat käyttävät SQL-kieltä tietojen kirjoittamiseen ja kyselyihin. Taylorin (2010, s. 12) mukaan tietokannat voidaan jakaa kolmeen eri tyyppiin, joita ovat hierarkkinen tietokanta, verkkotietokanta ja relaatiotietokanta. Tässä luvussa esitellään tarkemmin relaatiotietokantaa.

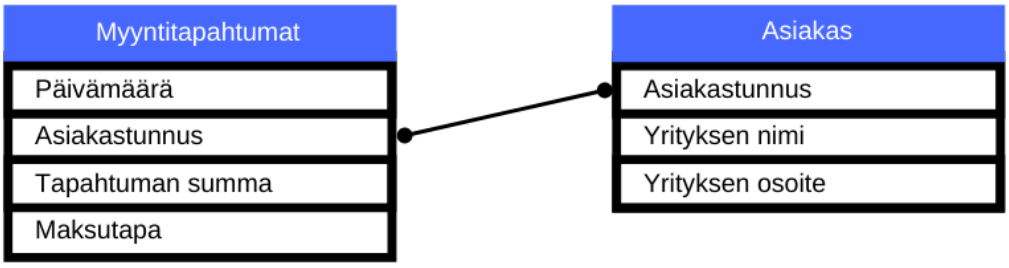
5.1 Relaatiotietokanta

Taylorin (2010, s. 13) mukaan IBM:n tutkija E.F. Codd julkaisi ensimmäisen relaatiotietokantamamallin vuonna 1970. Tämä alkoi näkyä tuotteissa noin vuosikymmen myöhemmin. Relaatiotietokannat ovat lähes korvanneet aikaisemmat tietokantamallit. Suurin syy tälle on relaatiotietokannan rakenteen muokkaaminen ilman, että varsinaista sovellusta tarvitsee muokata.

Hovin (2008, s. 6) mukaan kaikki tiedot relaatiotietokannoissa tallentuvat tauluihin. Taylorin (2010, s.14) mukaan jokainen taulu on relaatio. Relaatiotietokannan joustavuuden mahdollistaa se, että tiedot sijaitsevat tauluissa, jotka ovat usein toisistaan riippumattomia. Tällöin taulukkoon voidaan lisätä tietoa, poistaa ja muokata tietoa ilman, että se vaikuttaa muiden taulukoiden tietoihin. Edellytyksenä tälle on kuitenkin se, että kyseinen taulu ei ole muun taulun ylätaso.

Hovin (2008, s. 6) mukaan relaatiotietokannan taulu sisältää rivejä (row) ja sarakkeita (column). Oraclen (i.a.-b) mukaan jokainen taulun rivi on tietue (record), jolla on yksilöllinen tunnus, jota kutsutaan avaimeksi (key). JavaTpoint (i.a.-b) kertoo, että taulun sarake on pystysuora kokonaisuus taulussa, joka sisältää kaikki taulukon tiettyyn kenttään syötetyt tiedot. Esimerkiksi taulun nimi -sarake sisältää kaikki tiedot tallennetuista nimistä. TechTargetin (i.a) mukaan jokaisella taululla on yksilöllinen perusavain (primary key), joka tunnistaa taulun tiedot. Tauluille voidaan asettaa relaatioita käyttämällä viiteavaimia (foreign key), jotka linkittävät taulut toisen taulukon perusavaimeen.

IBM:n (i.a.-b) mukaan relaatiotietokannan esimerkkinä voidaan ajatella yrityksen myynninhallintaa, joka voidaan toteuttaa helposti relaatiotietokannan avulla. Tällöin esimerkiksi asiakastaulun sarakkeet voivat olla asiakastunnus, yrityksen nimi ja osoite. Vastaavasti myynnintapahtumataulu voisi sisältää päivämäärän, asiakastunnuksen, tapahtuman summan ja maksutavan. Nämä kaksi taulua voidaan yhdistää yhteisen asiakastunnus-kentän avulla. Relaatioiden avulla tietokannasta voidaan hakea monipuolisia raportteja ja vastaavasti rajata haettavaa tietoa tehokkaasti. Kuviossa 7 havainnollistetaan relaatiotietokannan taulujen välistä relaatiota.



Kuvio 7. Relaatiomalli myyntitapahtumista.

6 SQL-KIELI

6.1 Johdanto SQL-kieleen

TechTargetin (i.a.) mukaan SQL on standardoitu ohjelmointikieli, jota käytetään relaatiotietokantojen hallintaan ja eri toimintojen suorittamiseen relaatiotietokannoissa oleville tiedoille. Hovin (2004, s. 14) mukaan SQL-nimi tulee sanoista Structured Query Language eli struktuuroitu kyselykieli. SQL ei kuitenkaan ole vain kyselykieli, vaan se mahdollistaa seuraavat toiminnot:

- tietokannan rakenteen määrittelyn ja muuttamisen
- kyselyt tietokannasta
- tietojen lisäykset, muutokset ja poistot
- tapahtumakäsittelyn ohjaamisen
- tietokannan valtuuksien ja turvallisuuden määrittelyn
- API-rajapinnat muille ohjelmointikielille

6.2 SQL-kielen historiaa

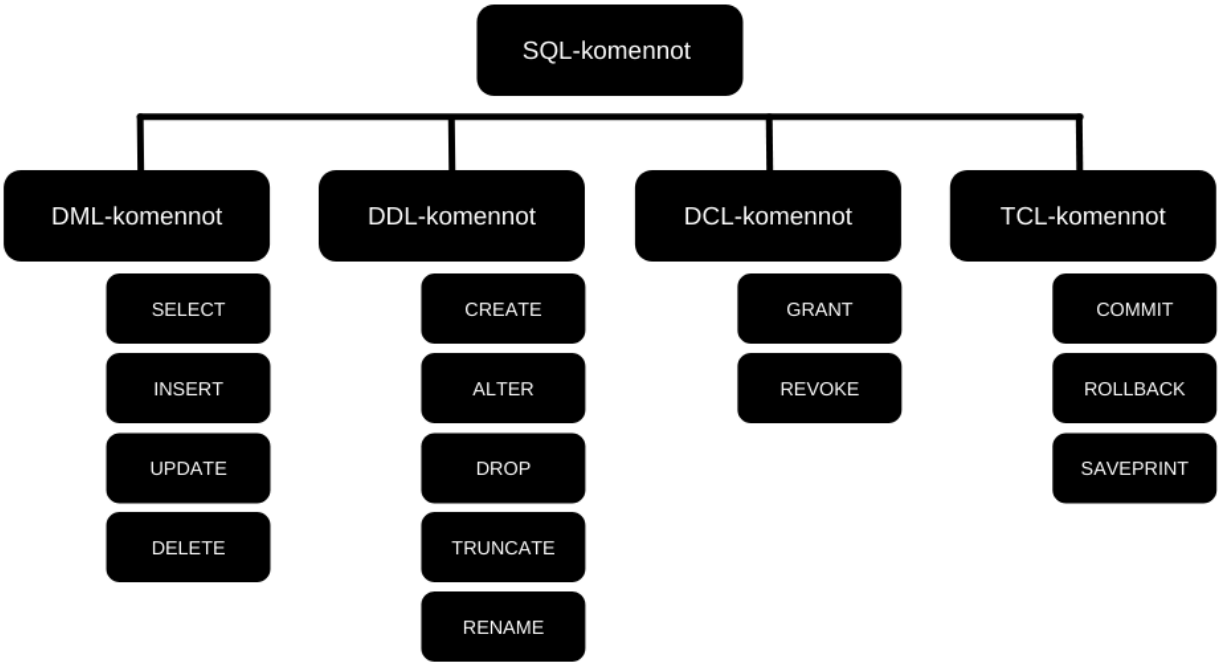
Hovin (2004, s. 17) mukaan IBM kehitti relaatiomallin mukaista prototyyppitietokantaa R-nimisessä projektissa 1970-luvun puolivälissä. Tietokantakieleksi valittiin SEQUEL-niminen kieli, jonka oli kehittänyt D.D Chamberlainin tutkimusryhmä. Myöhemmin kyseistä kieltä alettiin nimittää nimellä SQL (Structured Query Language). IBM julkaisi ensimmäisen SQL-tuotteensa SQL/DS vuonna 1981. Oracle toi markkinoille kuitenkin ensimmäisen SQL-tuotteen vuonna 1979. 1980-luvulla syntyi useita SQL- pohjaisia tuotteita, kuten Sybase, Tandem Nonstop-SQL, Informix, Ingres ja RDB. Microsoft toi markkinoille oman SQL-relaatiotietokantansa SQL Serverin 1990-luvun lopulla.

6.3 SQL-ohjelmointi

TechTargetin (i.a.) mukaan SQL on ohjelmointikieli, joka on suunniteltu relaatiotietokantojen tiedon lisäämiseen, muokkaamiseen ja tiedon poimimiseen. Ohjelmointikielenä SQL:llä on komennot ja syntaksi toimintojen toteuttamiseksi. GeeksforGeeksin (i.a.) mukaan SQL-komennot voidaan jakaa neljään eri luokkaa:

- Data Manipulation Language (DML): Komennot mahdollistavat tietojen lisäämisen, tietojen päivittämisen, rivien poistamisen ja tietojen valitsemisen olemassa olevista tauluista.
- Data Definition Language (DDL): Komennot mahdollistavat taulujen luomisen, muokkaamisen, poistamisen ja riippuvuussuhteiden luomisen.
- Data Control Language (DCL): Komennot mahdollistavat tietokannan asetusten ja käyttäjäoikeuksien määrittelyn. DCL-komentojen avulla voidaan määrittää eri käyttäjille sallittuja operaatioita.
- Transaction Control Language (TCL): Komennot mahdollistavat tietokannan tapahtumien hallitsemisen.

Esiteltyjen luokkien SQL-komentoja esitellään kuviossa 8.

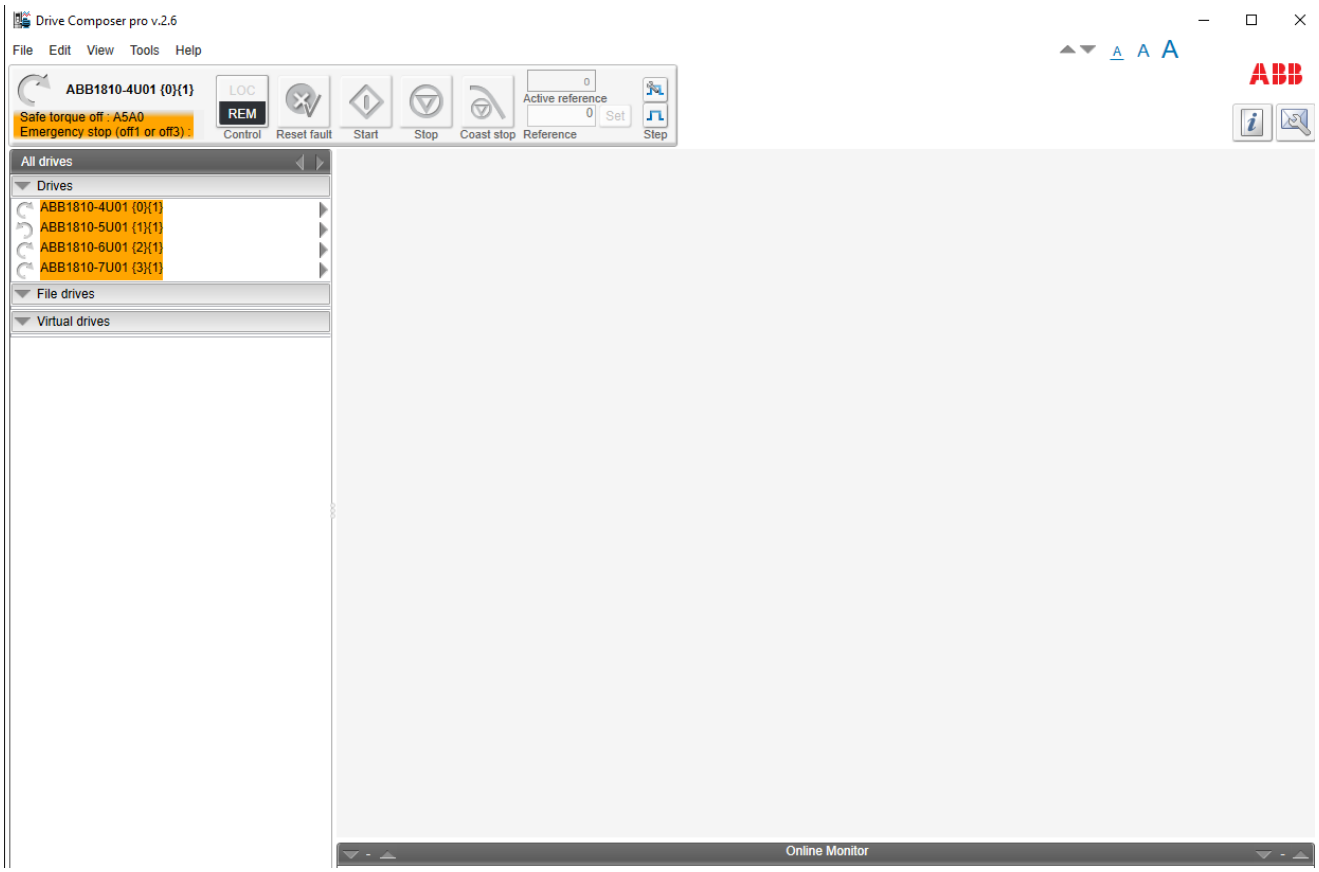


Kuvio 8. Yleisimmät SQL-komennot.

7 TYÖSSÄ KÄYTETYT TYÖKALUT

7.1 ABB Drive Composer -ohjelmisto

ABB:n (i.a.) mukaan Drive Composer on käyttöönotto- ja monitorointityökalu ABB:n yhtenäisen taajuusmuuttajatuoteperheen tuotteille. Drive Composer -ohjelmiston avulla voidaan konfiguroida taajuusmuuttajan parametreja, sekä tarkastella ja kerätä haluttuja arvoja taajuusmuuttajalta. Ohjelmistosta on saatavilla Entry- ja Pro-versiot. Entry-versio mahdollistaa perustason valvonnan ja parametrien konfiguroinnit sekä taajuusmuuttajan paikallisen PC-ohjauksen. Entry-versio on ladattavissa ilmaiseksi ABB:n verkkosivuilta. Kuvassa 3 esitellään Drive Composer Pro -ohjelmiston näkymää, kun yhteys taajuusmuuttajiin on muodostettu.



Kuva 3. Drive Composer Pro -ohjelmiston näkymä.

7.2 KEPServerEX-ohjelmisto

Kepwaren (i.a.-b) mukaan KEPServerEX-ohjelmisto on teollisuuden johtava kommunikointialusta. KEPServerEX on suunniteltu käyttäjäystävälliseksi, se mahdollistaa käyttäjille erilaisten automaatiolaitteiden ja ohjelmistosovellusten yhdistämisen, monitoroinnin, hallinnan ja ohjaamisen yhden alustan kautta. KEPServerEX-alusta hyödyntää OPC-standardia (muun muassa OPC DA ja OPC UA) ja keskeisiä IT-viestintäprotokollia kuten MQTT, REST, SNMP ja ODBC. Kuvassa 4 on havainnekuva KEPServerEX-kommunikointialustan näkymästä.

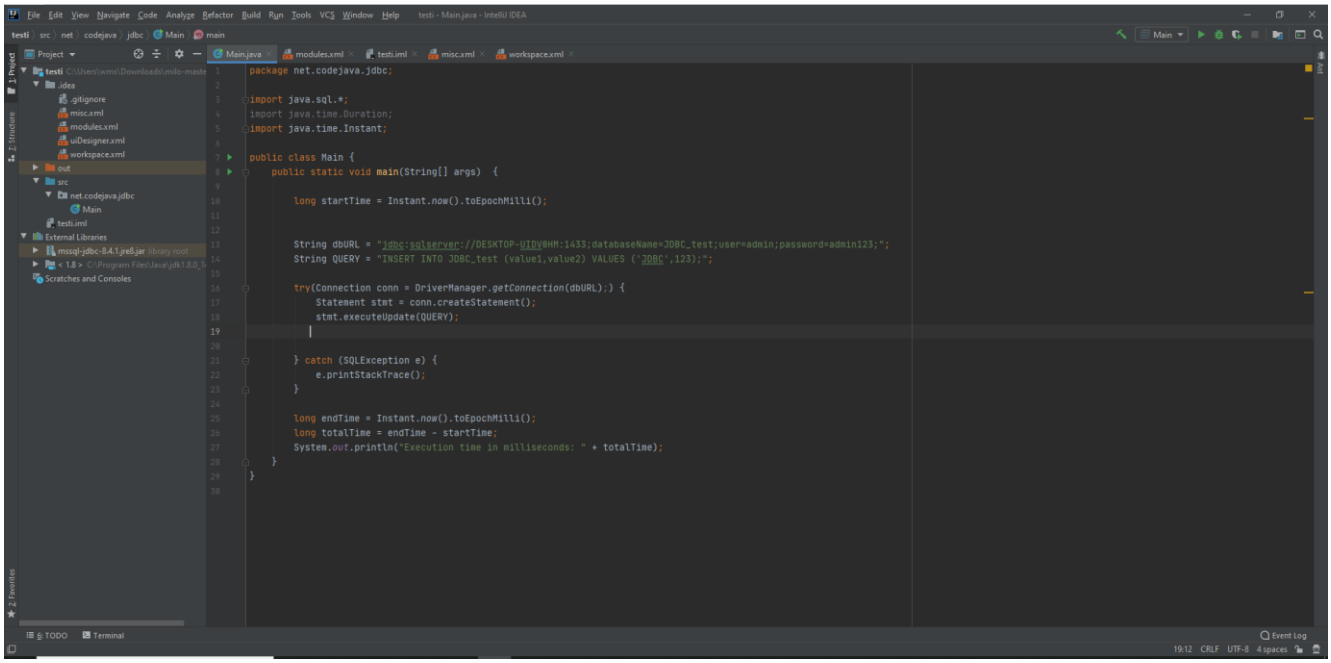
Kuva 4. KEPServerEX-kommunikointialusta.

7.3 IntelliJ IDEA -ohjelmointiympäristö

Jet-Brainsin (i.a.) mukaan IntelliJ IDEA on Javalle, Kotlinille, Scalalle ja Groovylle soveltuva ohjelmointiympäristö. IntelliJ IDEA -ohjelmistoympäristöä voidaan lisäksi laajentaa ilmaisilla laajennuksilla, joiden avulla ohjelmistoympäristö soveltuu käytettäväksi myös Gon, Pythonin, SQL:n, Rubyn ja PHP:n kanssa. Ohjelmistoympäristö on suunniteltu helpottamaan ja nopeuttamaan työskentelyä laajennuksilla, pikakuvakkeilla sekä mahdollisuudella muuttaa käyttöliittymää käyttäjälle mieleiseksi.

Jet-Brainsin (i.a.) mukaan ohjelmistoympäristö sisältää lisäksi useita kehittäjätyökaluja kuten tuen useimmille käännössovelluksille, versionhallintajärjestelmille, testikehyksille, sovelluspalvelimille ja komentorivipäätteille. IntelliJ IDEA -kehitysympäristössä on myös koodin

tarkistusominaisuus, jonka avulla kehittäjä kykenee koodaamaan nopeammin ja noudattamaan laatustandardeja. Kuvassa 5 on havainnekuva IntelliJ IDEA -ohjelmointiympäristöstä.



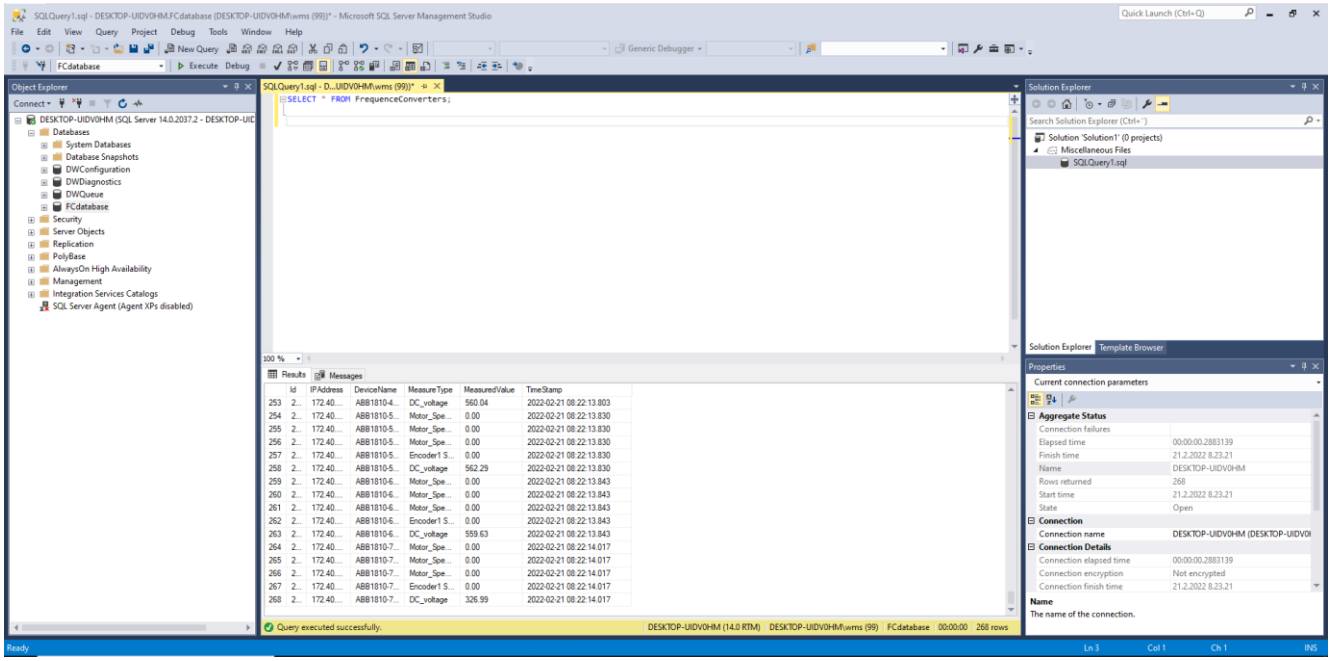
Kuva 5. IntelliJ IDEA -ohjelmointiympäristö.

7.4 SQL Server Management Studio

Microsoftin (i.a.-b) mukaan SQL Server Management Studio on kehitysympäristö SQL-infrastruktuurien hallintaan. SQL Server Management Studio sisältää seuraavat komponentit:

- Object Explorer, jonka avulla voidaan tarkastella ja hallita SQL Serverin sisältämiä objekteja. Object Explorerin avulla voidaan myös hallita useampia SQL Servereitä.
- Qyery Editor, jonka avulla voidaan kirjoittaa SQL-kyselyjä ja -komentoja.
- Template Explorer, jonka avulla voidaan selata valmiita käytettävissä olevia malleja ja tuoda malleja suoraan koodinmuokkausikkunaan.
- Solution Explorer, jonka avulla voidaan selata ja avata tallennettuja Query- ja Text-editoreja.
- Visual Database Tools, joiden tietokannan kyselyjä, taulukoita ja kaaviotietoja voidaan visualisoida.

Kuvassa 6 esitellään Microsoft SQL Server Management Studion näkymää.



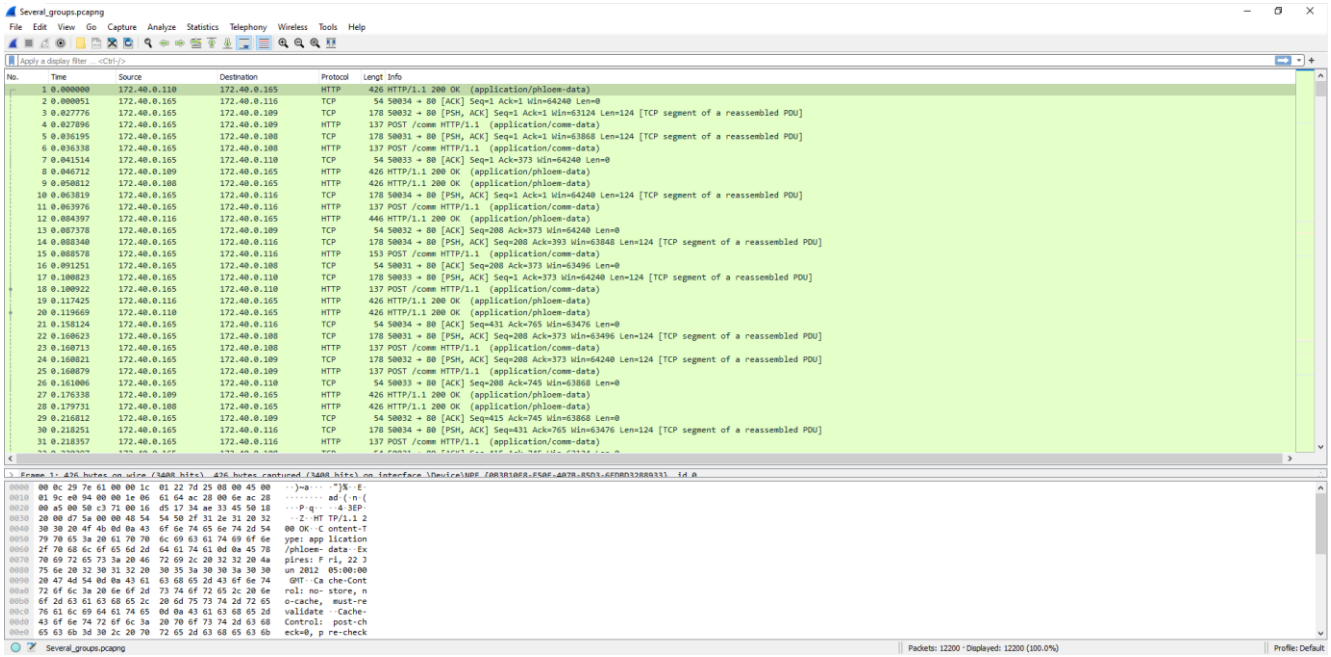
Kuva 6. Microsoft SQL Server Management Studio.

7.5 Wireshark

Wiresharkin (i.a.) mukaan Wireshark on maailman johtava ja eniten käytetty verkkoanalyysityökalu. Sen avulla voidaan analysoida verkon tapahtumia pääpiirteittäin aina mikroskooppiselle tasolle asti. Wireshark-verkkoanalyysityökalun kehitys aloitettiin vuonna 1998. Nykyisin Wireshark-verkkoanalyysityökalu sisältää seuraavat ominaisuudet:

- Satojen verkkoprotokollien syvällinen analysointi
- Livekaappaus verkkoliikenteestä ja myöhempi offline-analyysi
- Yhteensopiva Windows-, Linux- ja macOS-käyttöjärjestelmissä
- Monipuoliset suodattimet analysointiin
- VoIP-analyysi
- Salauksen purkaminen monille eri protokollille
- Wireshark-tallenne voidaan tallentaa XML-, PostScript-, CSV- tai tekstimuotoon

Kuvassa 7 esitellään Wireshark-verkkoanalyysityökalun ohjelmistonäkymää.



Kuva 7. Wireshark-verkkoanalyysityökalu.

8 LÄHKÖKOHTA

8.1 Ohjelman suunnittelu

Ohjelman suunnitteluvaiheessa työn toimeksiantajan kanssa käytiin läpi ohjelman vaatimuksia. Ohjelman tulisi soveltua käytettäväksi Windows PC:llä. Ohjelmalla tuli saada yhteys suoraan ABB:n taajuusmuuttajaan ilman, että tiedonkeruuohjelman ja taajuusmuuttajan välinen dataliikenne kulkisi ohjelmoitavan logiikan kautta. Tällöin ohjelma ei olisi riippuvainen ohjelmoitavasta logiikasta. Toteutettavan ohjelman päätarkoitus olisi tallentaa ABB:n valmistamilta taajuusmuuttajilta haluttujen parametrien arvoja myöhempää tarkastelua varten. Valmiin tiedonkeruuohjelman käyttöönoton tulisi olla myös mahdollisimman yksinkertainen, että ohjelman käyttöönotto ei vie liikaa resursseja projektilta.

Työn toimeksiantajan vaatimusten läpikäynnin jälkeen suoritettiin tiedonhakua toteutetuista tiedonkeruujärjestelmistä. Yksittäisten automaatiokomponenttien tai laitteiden tiedonkeruujärjestelmiä oli toteutettu melko vähän. Useimmiten automaatiokomponenttien ja järjestelmien tiedonkeruujärjestelmän tietolähteenä käytettiin ohjelmoitavaa logiikkaa. Työn vaatimuksena oli tiedonkeruujärjestelmän toteuttaminen ilman ohjelmoitavaa logiikkaa, joten näistä ei varsinaisesti ollut apua.

Ohjelman suunnitteluvaiheessa toimeksiantajan kanssa käytiin läpi, mitä arvoja taajuusmuuttajalta halutaan tallentaa. Oleelliseksi asiaksi suunnitteluvaiheessa nousi myös taajuusmuuttajilta kerättävien parametrien arvojen päivitystaajuus. Toimeksiantajan kanssa todettiin, että joidenkin kerättyjen parametrien arvon päivitystaajuus tulisi olla vähintään 1000 ms, että tallennushistoriasta on hyötyä ja että siitä voidaan tehdä päätelmiä automaatiojärjestelmän toiminnasta vikatilanteissa. Suunnitteluvaiheessa todettiin myös, että osalle parametreista riittää useiden minuuttien tai tunnin päivitystaajuus.

Suunnitteluvaiheessa otetiin yhteyttä taajuusmuuttajien valmistajaan ABB:hen. ABB:n henkilöstön kanssa järjestettiin palaveri. He esittelivät yrityksen omaa ratkaisuvaihtoehtoa tiedonkeruuta varten. Kyseistä ratkaisua ei kuitenkaan ollut saatavilla vallitsevan komponenttipulan vuoksi. Lisäksi itse toteutettu ratkaisu soveltui paremmin kohdeyrityksen tarpeisiin. ABB:n henkilöstö kertoi kuitenkin hyödyllisiä vinkkejä ohjelman suunnitteluun ja mihin asioihin

ohjelman suunnittelussa kannattaa kiinnittää huomiota, että tiedonkeruuohjelmisto ei kuormita automaatioverkkoa liikaa.

ABB:n yhteyshenkilö kertoi, että taajuusmuuttajaan olisi mahdollista muodostaa yhteys ilman ohjelmoitavaa logiikkaa ABB:n oman Drive Composer Pro -konfigurointityökalun avulla. Drive Composer Prosta on saatavilla erillinen versio, joka mahdollistaa haluttujen arvojen lukemisen ABB:n taajuusmuuttajalta OPC DA -yhteyden avulla. Suunnitteluvaiheessa tutkittiin myös vaihtoehtoa Drive Composer Pro -ohjelmiston pois jättämiseksi tiedonkeruuohjelmistossa. ABB:n mukaan Drive Composer Pro -ohjelmisto kommunikoi taajuusmuuttajan suuntaan http/https-protokollalla, jonka sisällä on ABB:n patentoitu Phloem-protokolla. Kyseisen protokollan spesifikaatio ei ole julkinen eikä siitä ole saatavilla julkista materiaalia. Näin ollen Drive Composer Pron mukana asentuvaa OPC DA -palvelinta ei voida korvata kolmannen osapuolen OPC DA -palvelimella.

8.2 Järjestelmän hyödyt

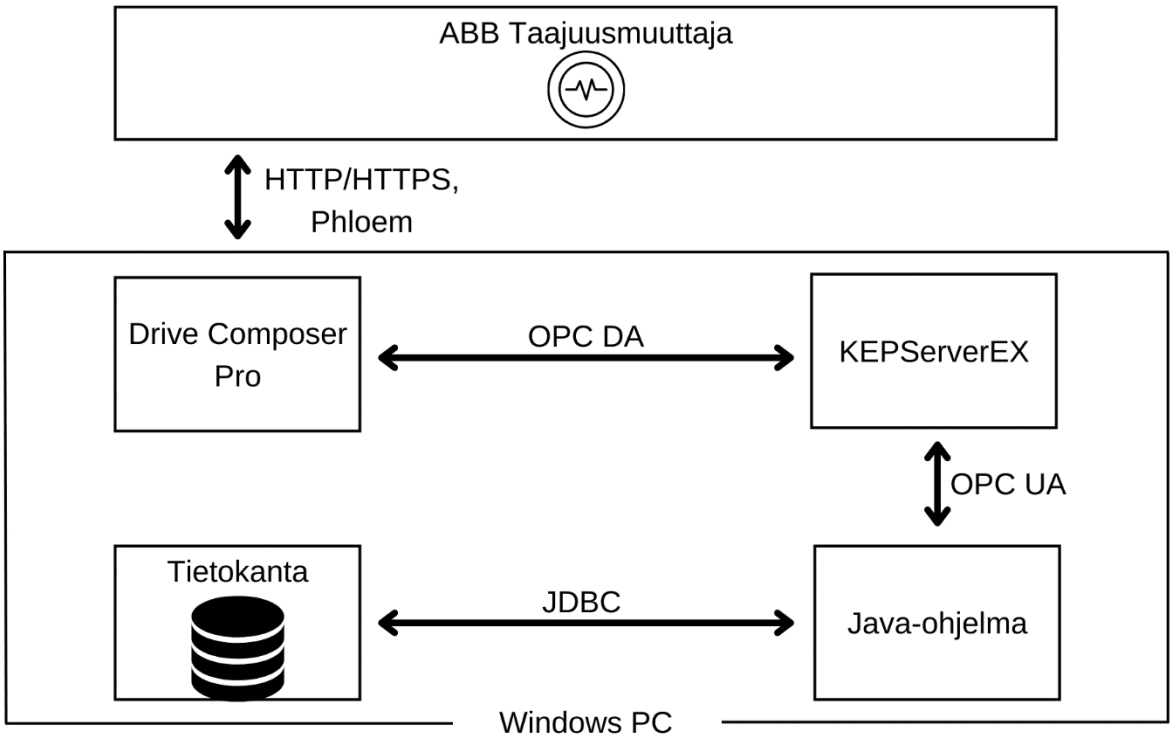
Tiedonkeruujärjestelmän on tarkoituksena mitata ja tallentaa tietoa käytössä olevan automaatiojärjestelmän taajuusmuuttajilta. Automaatiojärjestelmän vikatilanteissa taajuusmuuttajilta tallennettua dataa voidaan käyttää apuna vikatilanteiden selvittämisessä. Tallennetun datan avulla voidaan vikatilanteiden lisäksi ennakoida mahdollisia tulevia vikoja. Tallennettua dataa voidaan hyödyntää myös suunnitteluvaiheessa. Taajuusmuuttajien kuormitettavuutta voidaan tutkia tallennetun datan perusteella. Tällöin voidaan välttyä hankintavaiheessa taajuusmuuttajan yli- tai alimitoitukselta.

9 JÄRJESTELMÄN TOTEUTUS

Järjestelmän toteutusvaihe aloitettiin määrittelemällä tiedonkeruujärjestelmälle ohjelmistoarkkitehtuuri. Arkkitehtuurin määrittämisessä pyrittiin huomiomaan ohjelmiston käyttöönoton ja käytön helppous, jotta sen käyttöönotto kuluttaisi mahdollisimman vähän resursseja automaatioprojektin aikana. Tiedonkeruuohjelman ohjelmistoarkkitehtuuri päädyttiin toteuttamaan seuraavalla tavalla:

- ABB:n Drive Composer Pro -konfigurointityökalu, sitä käytetään rajapintana datan lukemiseen suoraan taajuusmuuttajalta. Drive Composer Pron käyttö katsottiin välttämättömäksi, koska ABB:n patentoimasta Phloem-protokollasta ei ole saatavilla julkista tietoa.
- KEPServerEX-kommunikointialusta, sitä käytetään työssä myös OPC-palvelimena. Kommunikointialusta tarjoaa valmiin OPC DA Clientin, joka soveltuu suoraan käytettäväksi DriveComposer Pron OPC DA -rajapinnan kanssa.
- Java-ohjelma, sillä luetaan dataa KEPServerEX-kommunikointialustan OPC UA -palvelimelta ja siirretään dataa tietokantaan.
- Microsoft SQL Server -relaatiotietokantajärjestelmä, johon taajuusmuuttajilta kerätty data tallennetaan.

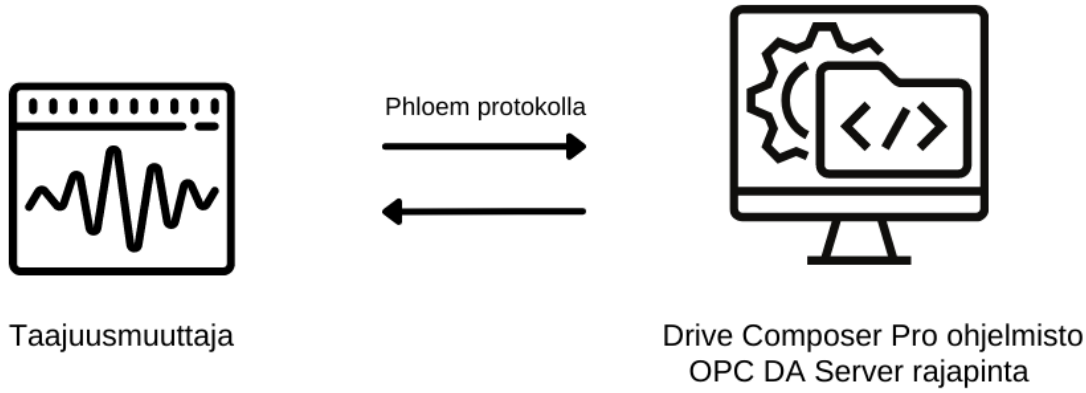
Kuviossa 9 havainnollistetaan tiedonkeruuohjelmiston arkkitehtuuria.



Kuvio 9. Tiedonkeruujärjestelmän ohjelmistoarkkitehtuuri.

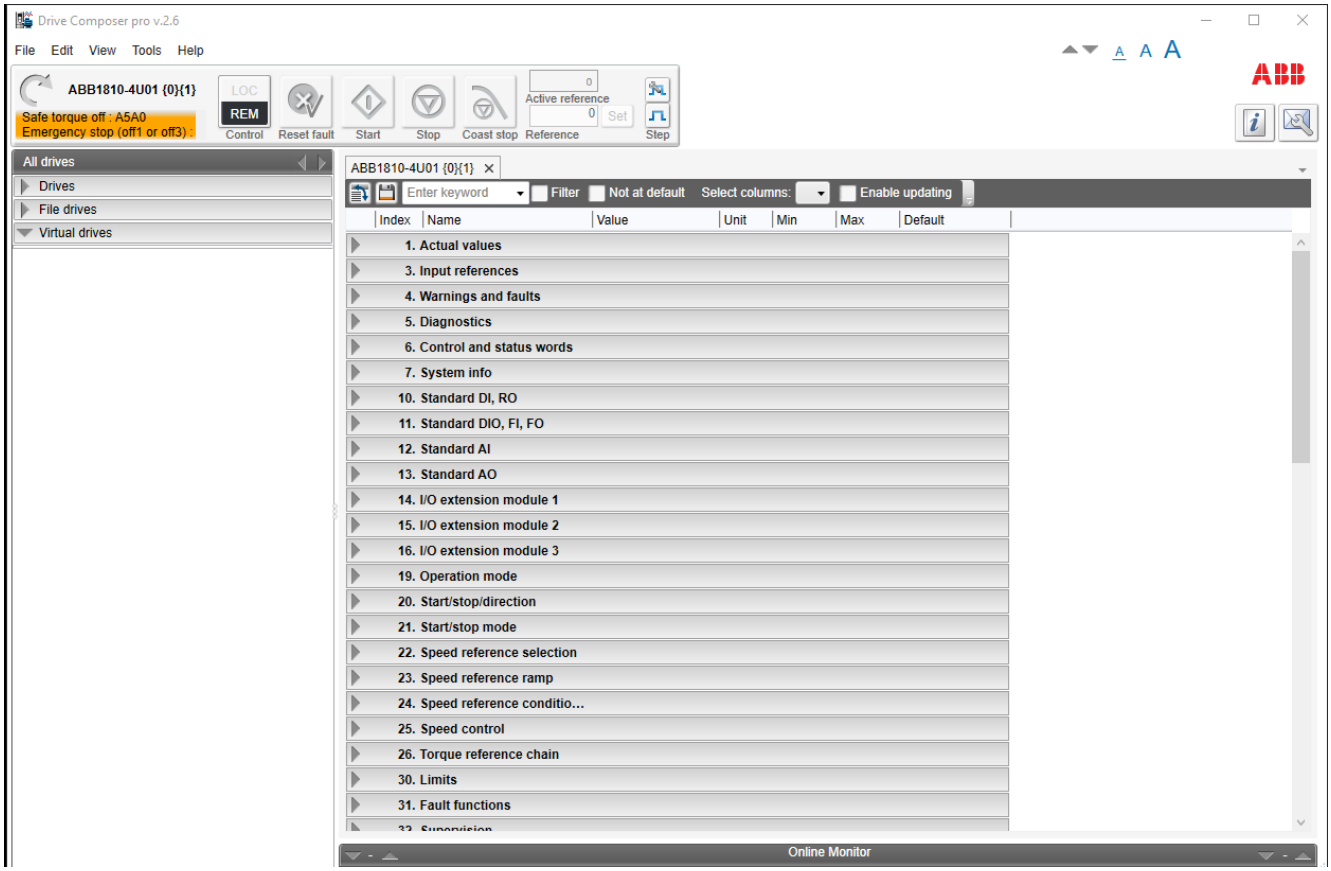
9.1 ABB Drive Composer Pro -ohjelmiston konfigurointi

Windows-kehitysympäristön virtuaalikoneelle asennettiin ABB Drive Composer Pro –konfigurointityökalu. Drive Composer Pron asennuksen yhteydessä ohjelmistolle asentuu myös OPC DA Server -rajapinta. Drive Composer Pro -ohjelmisto kommunikoi samassa verkossa olevien ABB:n taajuusmuuttajien kanssa ABB:n patentoimalla Phloem-protokollalla. Kyseisen protokollan spesifikaatio ei ole julkinen, eikä siitä ollut saatavilla valmistajan virallista materiaalia. Kuviossa 10 havainnollistetaan ABB:n taajuusmuuttajan ja Drive Composer Pron välistä kommunikointia.



Kuvio 10. ABB:n taajuusmuuttajan ja Drive Composer Pron välinen kommunikointimalli.

Drive Composer Pro -konfigurointityökalun käyttöönotto sujui helposti. Ohjelmiston asennuksen jälkeen Drive Composer Pron asetuksiin määritettiin taajuusmuuttajien IP-osoitteet, joihin yhteys haluttiin muodostaa. Konfigurointityökalu tarjosi myös automaattisen haun kaikille samassa verkossa oleville ABB:n taajuusmuuttajille. Kuvassa 8 on näkymä Drive Composer Pro -konfigurointityökalulla muodostetusta yhteydestä taajuusmuuttajaan.

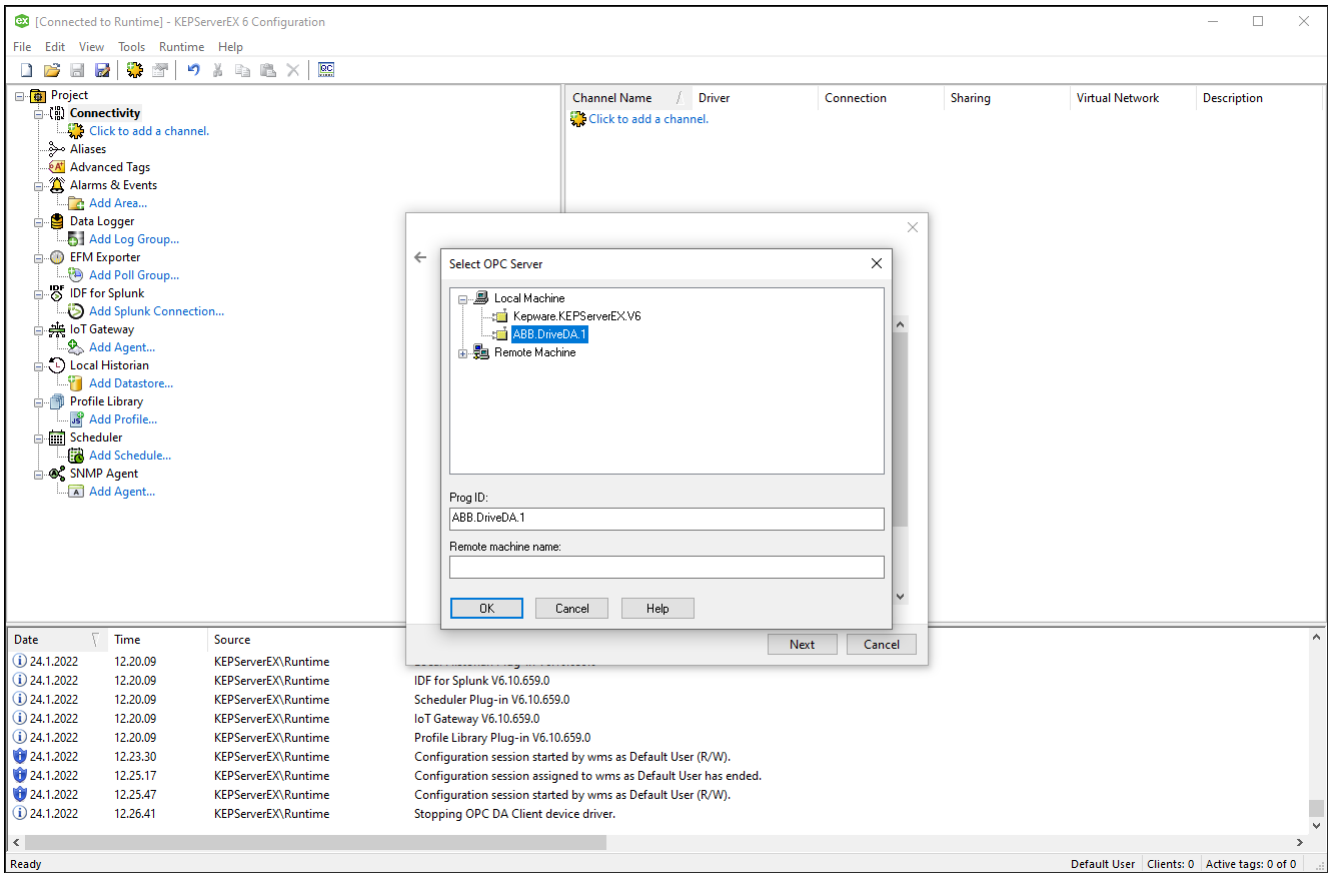


Kuva 8. Yhteys muodostettu taajuusmuuttajaan Drive Composer Pro –konfigurointityökalulla.

Drive Composer Pro toimii työssä toteutetussa tiedonkeruuohjelmistossa ikään kuin rajapintana välittämään tietoa ABB:n valmistamilta taajuusmuuttajilta varsinaiselle OPC-palvelimelle. Tietoa kerätessä ABB:n patentoima Phloem-protokolla toimittaa valittujen parametrien tiedot Drive Composer Pro -ohjelmiston OPC DA server -rajapintaan. Rajapintaan tuotuja tietoja voidaan lukea OPC DA -asiakassovelluksen (clientin) avulla.

9.2 KEPServerEX-ohjelman käyttö

Työssä käytettiin varsinaisena OPC-palvelimena KEPServerEX-kommunikointialustaa. KEPServerEX-kommunikointialusta tarjosi valmiin OPC DA –asiakasrajapinnan. Tätä käyttäen alustalle luotiin OPC DA -kanava, jonka kautta saatiin yhteys taajuusmuuttajiin Drive Composer Pron OPC DA -rajapintaa käyttäen. Kuvassa 9 on nähtävillä Drive Composer PROn tarjoama OPC DA Server -rajapinta.



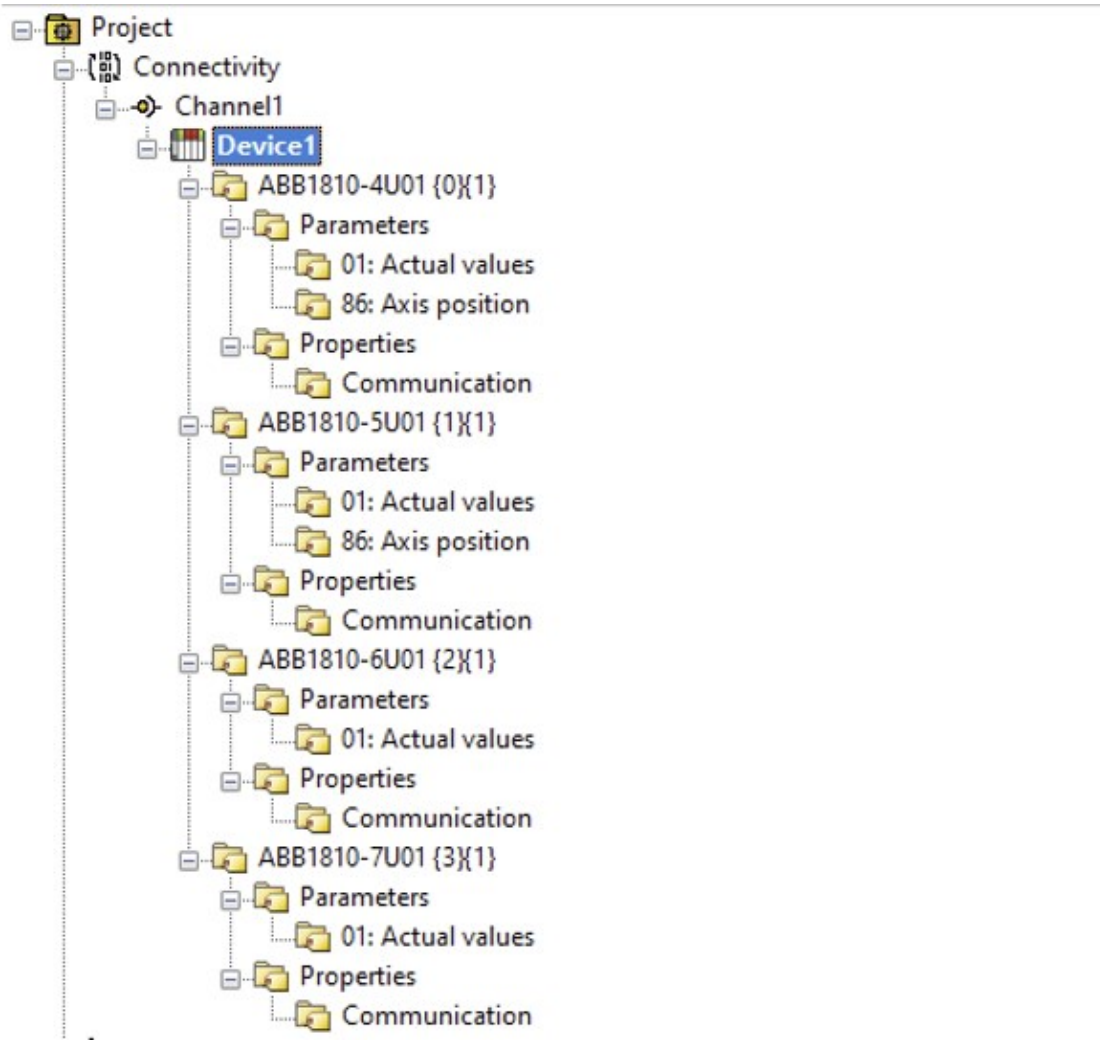
Kuva 9. Yhteyden muodostaminen Drive Composer PRO:n OPC DA -palvelimelle.

Yhteyden muodostamisen yhteydessä valittiin myös taajuusmuuttajilta kerättävät parametrit. Testiympäristössä oli käytössä neljä ABB:n taajuusmuuttajaa. Jokaiselta taajuusmuuttajalta kerättiin kuvan 10 mukaiset parametrit.

Tag Name	Address	Data Type	Scan Rate	Scaling
URL	{0}Communication.URL	String	10000	None
Name	{0}{1}Properties.Name	String	10000	None
86_01: Axis status [NoUnit]	{0}{1}Par.86.1	DWord	1000	None
86_03: Actual velocity [Units/s]	{0}{1}Par.86.3	Double	1000	None
86_02: Actual position [Units]	{0}{1}Par.86.2	Double	1000	None
01_11: DC voltage [V]	{0}{1}Par.1.11	Double	1000	None
01_10: Motor torque [%]	{0}{1}Par.1.10	Double	1000	None
01_07: Motor current [A]	{0}{1}Par.1.7	Double	1000	None
01_05: Encoder 2 speed filtered [rpm]	{0}{1}Par.1.5	Double	1000	None
01_04: Encoder 1 speed filtered [rpm]	{0}{1}Par.1.4	Double	1000	None
01_03: Motor speed % [%]	{0}{1}Par.1.3	Double	1000	None
01_02: Motor speed estimated [rpm]	{0}{1}Par.1.2	Double	1000	None
01_01: Motor speed used [rpm]	{0}{1}Par.1.1	Double	1000	None

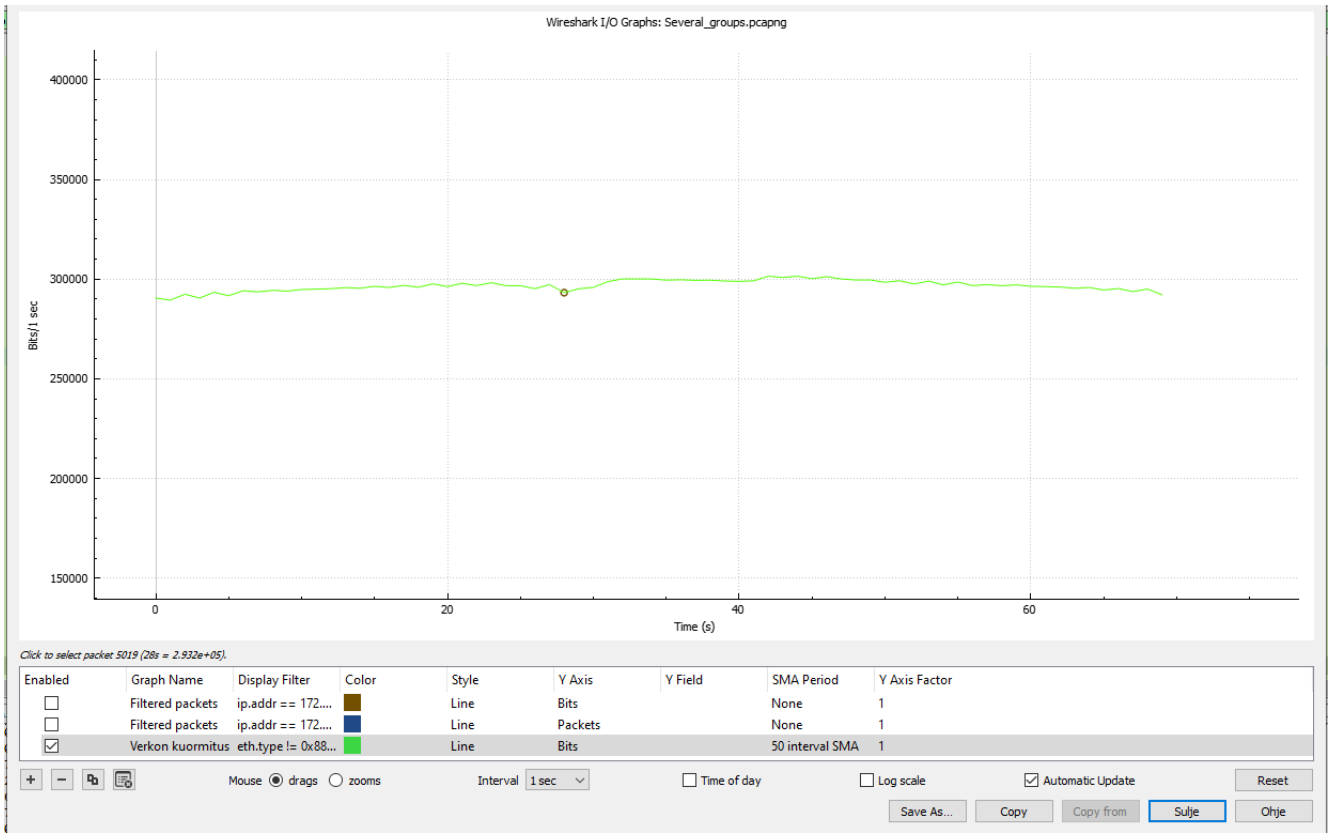
Kuva 10. Taajuusmuuttajalta kerättävät parametrit.

Työn yhtenä tavoitteena oli toteuttaa taajuusmuuttajien tiedonkeruujärjestelmä, joka ei kuormita automaatioverkkoa liikaa. KEPServerEX-ohjelmiston konfigurointivaiheessa automaatioverkon kuormitusta mitattiin Wireshark-työkalua käyttäen. KEPServerEX-kommunikointialusta loi automaattisesti OPC-ryhmät jokaiselle taajuusmuuttajalle ja sen parametreille. KEPServerEX jakoi kunkin taajuusmuuttajan parametrit alikansioihin, mikä monimutkaisti OPC-ryhmien kansiorakenteita. Kuvassa 11 esitellään KEPServerEX-kommunikointialustan luomaa kansiorakennetta kerättäville OPC-tageille.



Kuva 11. KEPServerEX-alustan luoma kansiorakenne OPC-tageille.

Tiedonkeruujärjestelmän verkonkuormitusta aloitettiin mittaamaan KEPServerEX-kommunikointialustan automaattisesti luomilla OPC-tagiryhmillä. Päivitystaajuudeksi asetettiin jokaiselle OPC-tagille 1000 ms. Näillä arvoilla verkon kuormitukseksi mitattiin keskiarvollisesti noin 300 kbit/s. Kuvassa 12 ja 13 esitellään Wireshark-työkalulla saatuja tuloksia verkon kuormituksesta.



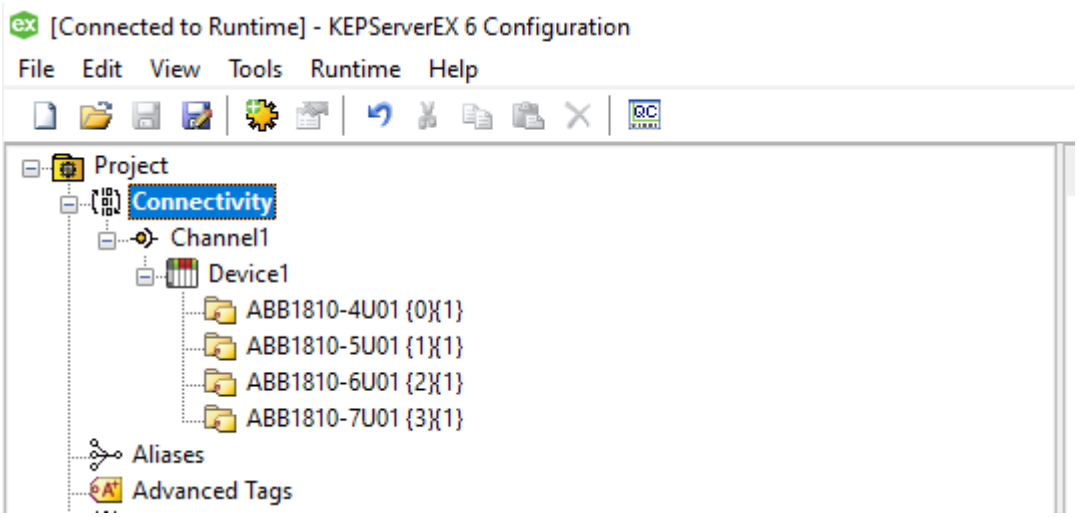
Kuva 12. Wireshark-työkalulla mitattu verkon kuormitus käyttäen useita OPC-ryhmiä.

Statistics

Measurement	Captured	Displayed	Marked
Packets	12200	12163 (99.7%)	—
Time span, s	69.864	69.864	—
Average pps	174.6	174.1	—
Average packet size, B	214	215	—
Bytes	2611659	2609817 (99.9%)	0
Average bytes/s	37 k	37 k	—
Average bits/s	299 k	298 k	—

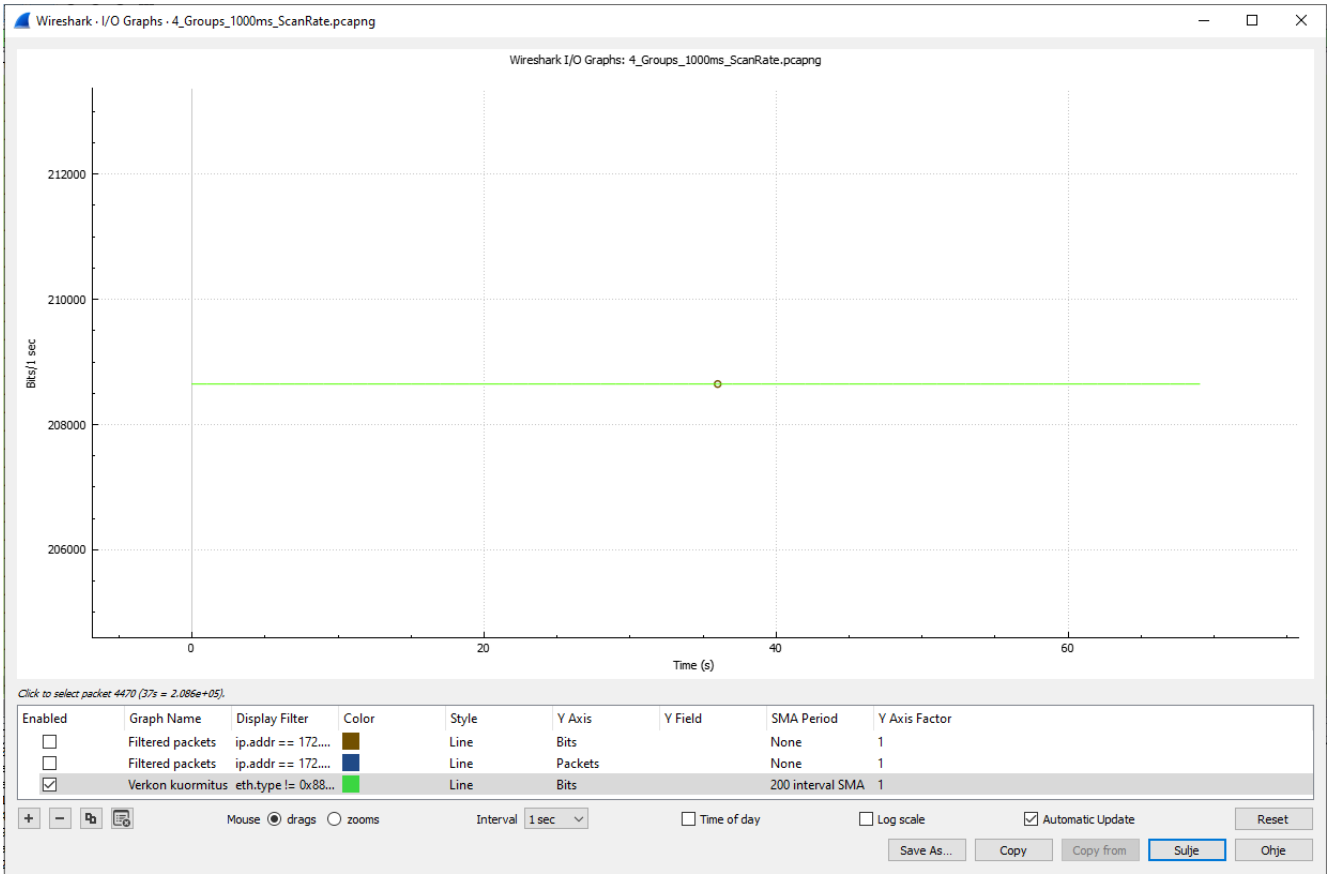
Kuva 13. Verkon kuormituksen mittaustulos useilla OPC-ryhmillä.

Wireshark-työkalulla mitattu verkon kuormitus 300 kbit/s oli hyväksyttävä eikä sen katsottu kuormittavan automaatioverkkoa liikaa. Verkon kuormitus pyrittiin kuitenkin minimoimaan ja sitä varten tutkittiin lisää KEPServerEX-kommunikointialustan luomaa kansiorakennetta. Seuraavaan verkon kuormitustestiin kansiorakenne selkeytettiin poistamalla ylimääräiset alikansiot kunkin taajuusmuuttajan alta. Selkeytetyssä kansiorakenteessa jokaisen taajuusmuuttajan parametrit olivat kukin omissa taajuusmuuttajakohtaisissa kansiossa ja kaikki ylimääräiset välikansiot poistettiin. Kuvassa 14 esitellään KEPServerEX-alustalle selkeytettyä OPC-tagien ryhmittelyä.



Kuva 14. Selkeytetyt OPC-tagiryhmät KEPServerEX-kommunikointialustalla.

Verkon kuormitusta mitattiin uudelleen selkeytetyllä OPC-tagiryhmittelyllä Wireshark-työkalua käyttäen. OPC-tagien päivitystaajuus oli edelleen sama 1000 ms. Keskimääräisen verkon kuormituksen mittaustulokseksi saatiin 212 kbit/s, joka oli selvästi vähemmän kuin aiempi 300 kbit/s. Näin voitiin todeta, että OPC-ryhmien minimoiminen pudottaa verkon kuormitusta selvästi. Kuvassa 15 ja 16 esitellään Wireshark-työkalulla saatuja mittaustuloksia.



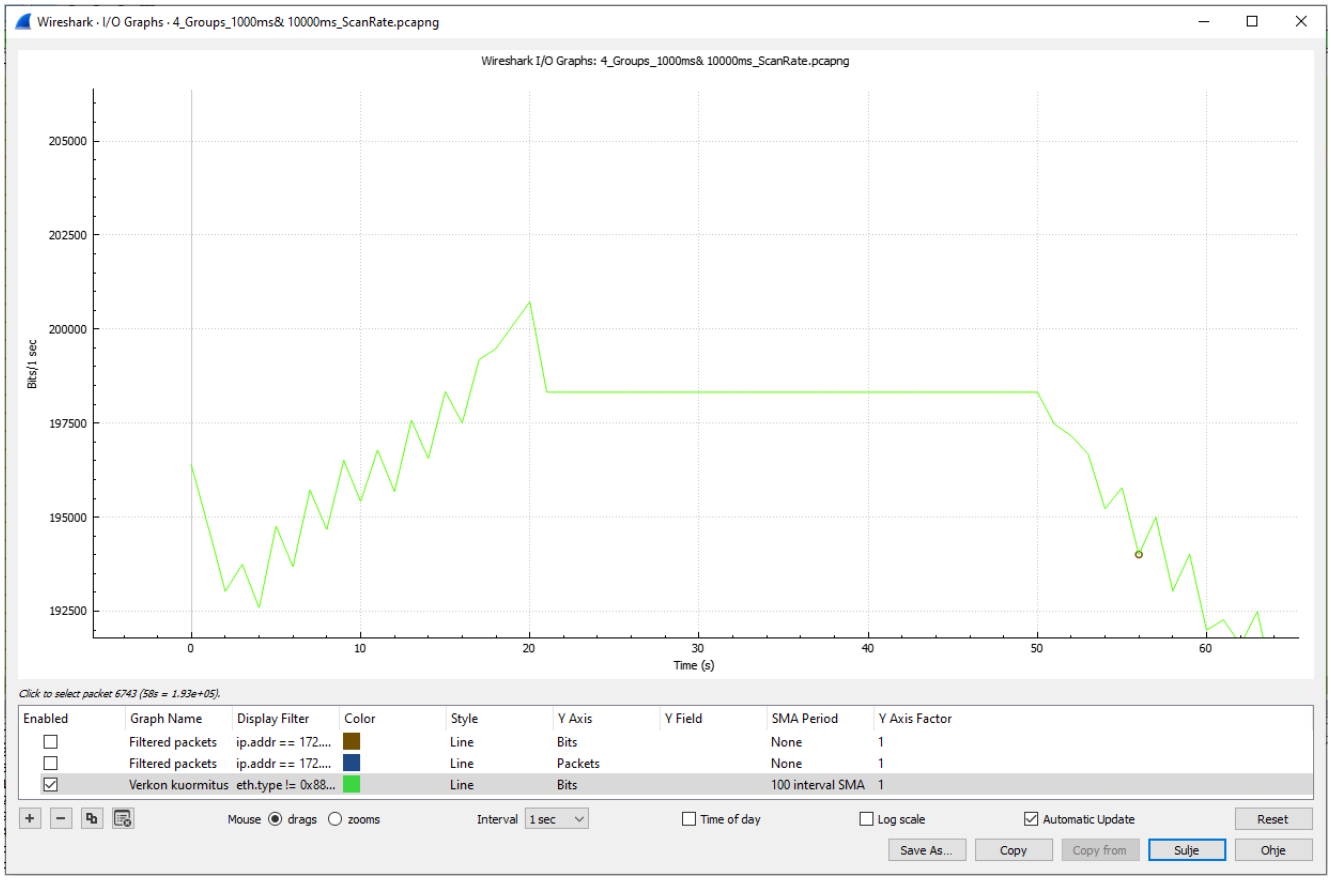
Kuva 15. Wireshark-työkalulla mitattu verkonkuormitus käyttäen laitekohtaisia OPC-ryhmiä.

Statistics

Measurement	Captured	Displayed	Marked
Packets	8430	8393 (99.6%)	—
Time span, s	69.595	69.595	—
Average pps	121.1	120.6	—
Average packet size, B	220	220	—
Bytes	1851199	1848860 (99.9%)	0
Average bytes/s	26 k	26 k	—
Average bits/s	212 k	212 k	—

Kuva 16. Verkon kuormituksen mittaustulos laitekohtaisilla OPC-ryhmillä.

Tehtyihin muutoksiin oltiin tyytyväisiä, koska verkon kuormitus putosi selvästi. Kaikkia taajuusmuuttajalta luettavia parametreja ei kuitenkaan ollut tarvetta lukea päivitystaajuudella 1000 ms. Näin ollen taajuusmuuttajalta luettavaan IP-osoitteeseen ja laitenimeen asetetiin päivitystaajuudeksi 10000 ms ja testattiin, voidaanko tällä muutoksella vielä minimoida verkon kuormitusta. Näillä muutoksilla keskimääräinen verkon kuormitus oli 203 kbit/s. Kuvassa 17 ja 18 esitellään päivitystaajuuden muuttamisen vaikutusta mittaustulokseen.



Kuva 17. Verkon kuormituksen mittaustulos Wireshark-työkalulla.

Statistics

Measurement	Captured	Displayed	Marked
Packets	8162	8133 (99.6%)	—
Time span, s	70.069	70.069	—
Average pps	116.5	116.1	—
Average packet size, B	219	219	—
Bytes	1783731	1782351 (99.9%)	0
Average bytes/s	25 k	25 k	—
Average bits/s	203 k	203 k	—

Kuva 18. Verkon kuormituksen mittaustulokset Wireshark-työkalulla.

9.3 Java-ohjelma

Varsinainen tiedonkeruuohjelma päädyttiin toteuttamaan Java-ohjelmointikieltä käyttäen. Java-ohjelmointikieleen päädyttiin, koska Java ohjelmointikielenä oli tuttu kohdeyrityksessä. Java-ohjelmointiin käytetiin IntelliJ IDEA -kehitysympäristöä. Java-ohjelmalla muodostettiin yhteys KEPServerEX-kommunikointialustan OPC UA –palvelimelle. Yhteyden muodostamiseen käytettiin avoimen lähdekoodin Eclipse Milo OPC UA -toteutusta. Tämän avulla yhteys Java-ohjelman ja OPC UA -palvelimen välille saatiin helposti toteutettua. Yhteyden avulla OPC UA -palvelimelle kerättyjä parametreja taajuusmuuttajilta saatiin luettua Java-ohjelmaan.

Työn yhtenä tavoitteena oli toteuttaa ohjelma, joka on mahdollisimman helposti käyttöönotettava myös tulevissa projekteissa. Siksi Java-ohjelmaan luotiin lista taajuusmuuttajista, niiden nimien perusteella. Tällöin uuden taajuusmuuttajan lisääminen ohjelmistoon vaatii ainoastaan taajuusmuuttajan nimen, joka on nähtävillä KEPServerEX-kommunikointialustan OPC UA -palvelimelta. Kun taajuusmuuttajan laitekohtainen nimi lisätään Java-ohjelmaan, lukee ohjelma määritetyt parametrit taajuusmuuttajalle OPC UA -palvelimelta. Osa taajuusmuuttajista on itsepaikoittavia, jolloin paikoitus tapahtuu suoraan taajuusmuuttajan omasta ohjauksesta. Java-ohjelmaan luotiin boolean-muuttuja, joka ilmaisee, onko taajuusmuuttaja paikoittava. Mikäli taajuusmuuttaja on paikoittava, luetaan kyseiselle taajuusmuuttajalle OPC UA -palvelimelta myös akselitiedot. Kuvassa 19 esitellään Java-ohjelman listausta taajuusmuuttajista.

```
// List of ABB Drives
List<ABBDrive> ABBDrives = Arrays.asList(new ABBDrive( id: "ABB1810-4U01 {0}{1}",  positioned: true,  TwoEncoders: true),new ABBDrive( id: "ABB1810-5U01 {1}{1}",
        positioned: true, TwoEncoders: true), new ABBDrive( id: "ABB1810-6U01 {2}{1}",  positioned: true, TwoEncoders: true),
        new ABBDrive( id: "ABB1810-7U01 {3}{1}",  positioned: false, TwoEncoders: false));
```

Kuva 19. Taajuusmuuttajista luotu lista Java-ohjelmaan.

OPC-tagien lukemisessa hyödynnettiin Javan HashMap-objektia, jolloin OPC-tagit tallennettiin avain-arvo-pareihin. OPC-tagit lisättiin HashMappiin for-toistorakenteen sisällä, sen

ehtona oli käydä läpi jokainen Array-listaan lisätty taajuusmuuttaja. Parametrien lukemista OPC UA -palvelimelta varten luotiin metodi, joka palauttaa HashMapin arvot. Metodin ja avoimen lähdekoodin Eclipse Milon -toteutuksen avulla OPC-tagit saatiin luettua OPC UA -palvelimelta kerralla ja tallennettua ne HashMapin arvoihin.

Java-ohjelman projektitiedostoon asennettiin JDBC-ajuri, jonka avulla taajuusmuuttajilta luettavat parametrit saatiin tallennettua tietokantaan. Työn yhtenä tavoitteena oli toteuttaa tiedonkeruujärjestelmä, joka on helposti käyttöönotettava myös tulevissa projekteissa. Tästä syystä neljännen tyypin JDBC-ajuri osoittautui parhaaksi vaihtoehdoksi, koska se ei vaadi erillisten ohjelmistojen asentamista Java-ohjelmaan tai tietokantaan. Ohjelmassa päädyttiin käyttämään neljännen tyypin JDBC-ajuria, joka muodostaa yhteyden suoraan tietokantaan. Kuvassa 20 on esimerkkikuva yhteyden muodostamisesta tietokannan ja Java-ohjelman välille.

```
23          //Connection to Microsoft SQL Server database
24          String url = "jdbc:sqlserver://DESKTOP-UIDV0HM:1433;databaseName=DataCollectionDB;user=admin;password=admin123;";
25          Connection con = DriverManager.getConnection(url);
26          System.out.print("Connected to database");
```

Kuva 20. Yhteyden muodostaminen tietokantaan Java-ohjelman JDBC-rajapintaa käyttäen.

9.4 Microsoft SQL Server -tietokannan toteutus

Java-ohjelmalla luetut taajuusmuuttajien parametrit OPC UA -palvelimelta tallennettiin tietokantaan datan myöhempää tarkastelua varten. Työssä käytettiin Microsoft SQL Server -tietokantaa. Tietokantaan luotiin taulu, johon taajuusmuuttajalta kerättävien parametrien arvot tallennettiin Java-ohjelman JDBC-ohjelmointirajapinnan avulla. Taulukossa 2 esitellään tiedonkeruutaulun rakennetta, mihin tieto tallennettiin.

Taulukko 2. Tietokannan taulun rakenne.

	Id	IPAddress	DeviceName	Measure-Type	Measured-Value	TimeStamp
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Tietokannan tauluun luotiin taulukon 2 mukaiset sarakkeet. Id-sarake on taulun identiteettisarake. Id kasvaa aina, kun tauluun tuodaan tietoa. Taulukon IPAddress-sarakkeeseen tallennetaan taajuusmuuttajan IP-osoite, jolta tiedonkeruuta tehdään. Taulukon DeviceName-sarakkeeseen tallennetaan taajuusmuuttajan yksilöllinen laitenimi. MeasureType-sarakkeeseen tallennetaan luettavan parametrin nimi ja MeasuredValue-sarakkeeseen mitatun parametrin arvo. TimeStamp-sarakkeeseen tallennetaan tapahtuman aikaleima. Kuvassa 21 esitellään tietokannan tauluun tallennettua dataa taajuusmuuttajilta.

Results Messages

	Id	IPAddress	DeviceName	MeasureType	MeasuredValue	TimeStamp
3871	6773	172.40.0.108:80	ABB1810-4U01	Motor_Speed_Used	0.00	2022-03-09 14:35:47.930
3872	6774	172.40.0.108:80	ABB1810-4U01	Motor_Speed_Estimated	0.00	2022-03-09 14:35:47.930
3873	6775	172.40.0.108:80	ABB1810-4U01	Motor_Speed_%	0.00	2022-03-09 14:35:47.930
3874	6776	172.40.0.108:80	ABB1810-4U01	Encoder1 Speed	0.00	2022-03-09 14:35:47.930
3875	6777	172.40.0.108:80	ABB1810-4U01	Motor_Current	0.00	2022-03-09 14:35:47.930
3876	6778	172.40.0.108:80	ABB1810-4U01	Motor_Torque	0.00	2022-03-09 14:35:47.930
3877	6779	172.40.0.108:80	ABB1810-4U01	DC_voltage	566.14	2022-03-09 14:35:47.930
3878	6780	172.40.0.108:80	ABB1810-4U01	Encoder2 Speed	0.00	2022-03-09 14:35:47.930
3879	6781	172.40.0.108:80	ABB1810-4U01	Axis_Status	67.00	2022-03-09 14:35:47.930
3880	6782	172.40.0.108:80	ABB1810-4U01	Axis_Position	10.01	2022-03-09 14:35:47.930
3881	6783	172.40.0.108:80	ABB1810-4U01	Axis_Velocity	0.00	2022-03-09 14:35:47.930

Kuva 21. Tietokantaan tallennettua dataa taajuusmuuttajilta.

10 YHTEENVETO JA POHDINTA

Tämän työn tavoitteena oli toteuttaa tiedonkeruujärjestelmä, jonka avulla voidaan tallentaa ABB:n valmistamien taajuusmuuttajien parametrien arvoja myöhempää tarkastelua varten. Työn tavoitteena oli toteuttaa järjestelmä, joka on yhteensopiva Windows PC:n kanssa. Vaatimuksena tiedonkeruujärjestelmälle oli myös yhteyden muodostaminen suoraan taajuusmuuttajiin ilman, että data kiertää ohjelmoitavan logiikan kautta. Työn tavoitteena oli myös selvittää toteutetun tiedonkeruujärjestelmän verkon kuormitus.

Toteutusvaihe aloitettiin tiedonkeruujärjestelmän suunnittelulla ja tiedonhaulla olemassa olevista tiedonkeruujärjestelmistä. Suunnitteluvaiheessa oltiin yhteydessä taajuusmuuttajien valmistajaan ABB:hen. ABB:n henkilöstöltä saatiin hyödyllistä tietoa tiedonkeruujärjestelmän suunnitteluun ja toteuttamiseen. Suunnitteluvaiheen edetessä tiedonkeruujärjestelmälle määritettiin ohjelmistoarkkitehtuuri, jonka mukaan tiedonkeruujärjestelmää toteutettiin. Ohjelmistoarkkitehtuuria määrittäessä otettiin huomioon järjestelmän helppokäyttöisyys ja myöhempi laajentaminen.

Varsinaisen tiedonkeruujärjestelmän toteutus aloitettiin asentamalla Windows PC:lle ABB:n DriveComposer Pro -konfigurointityökalu, joka tarjosi OPC DA -rajapinnan taajuusmuuttajille. Tämän jälkeen Windows PC:lle asennettiin KEPServerEX-kommunikointialusta. Kommunikointialustalle luotiin taajuusmuuttajakohtaiset OPC-ryhmät, joihin lisättiin taajuusmuuttajilta luettavat parametrit ja määritettiin niille halutut päivitystaajuudet. Kommunikointialustan käyttöönoton jälkeen Java-ohjelman ohjelmointi aloitettiin. Java-ohjelman yhteyden muodostamiseen OPC UA -palvelimelle käytettiin avoimen lähdekoodin Eclipse Milo -toteutusta. Java-ohjelmaan asennettiin JDBC-ajuri, jonka avulla luetut parametrit saatiin tallennettua tietokantaan myöhempää tarkastelua varten. Java-ohjelman toteuttamisen jälkeen tiedonkeruujärjestelmälle perustettiin Microsoft SQL Server -tietokanta. Tietokantaan tallennettiin taajuusmuuttajilta kerättyjen parametrien arvot.

Tiedonkeruujärjestelmän testaaminen aloitettiin keräämällä dataa yhdeltä testiympäristössä olevalta ABB:n taajuusmuuttajalta. Testauksen aikana, kun taajuusmuuttajia lisättiin tiedonkeruujärjestelmään, tuli ongelmaksi uusien taajuusmuuttajien parametrien käsittely ja lisääminen tiedonkeruujärjestelmään. Tästä johtuen taajuusmuuttajia päädyttiin käsittelemään listana Java-ohjelmassa. Tällöin uuden taajuusmuuttajan lisääminen tiedonkeruujärjestelmään vaatii ainoastaan yksilöllisen laitenimen lisäämisen Java-ohjelmaan, jolloin ohjelma hakee

lisätyn taajuusmuuttajan parametrit automaattisesti OPC UA -palvelimelta, johon ne on valittu taajuusmuuttajalta Drive Composer Pron tarjoaman OPC DA -rajapinnan kautta.

Työn lopputuloksesi saatiin toimiva tiedonkeruujärjestelmä. Toteutetun tiedonkeruujärjestelmän avulla ABB:n taajuusmuuttajilta voidaan lukea ja tallentaa haluttuja parametreja ja tarkastella niitä myöhemmin tietokannan kautta. Tiedonkeruujärjestelmään voidaan lisätä helposti lisää taajuusmuuttajia, joista tietoa kerätään. Toteutettu tiedonkeruujärjestelmä muodostaa yhteyden suoraan taajuusmuuttajiin ilman, että data kiertää PLC:n kautta. Toteutetun tiedonkeruujärjestelmän kautta voidaan analysoida laitteiden kuormitusta sekä selvittää taajuusmuuttajan parametrien arvoja vikatilanteissa ja tehdä niistä päätelmiä. Toteutettu tiedonkeruujärjestelmä on tarkoituksena ottaa käyttöön toimeksiantajayrityksen tulevassa projektissa.

Työn tavoitteissa onnistuttiin hyvin ja toteutettu tiedonkeruujärjestelmä täyttää sille asetetut vaatimukset. Tiedonkeruujärjestelmän verkon kuormitus saatiin myös mitattua tarvittavalla tarkkuudella. Verkon kuormituksesta saatujen tutkimustulosten avulla tiedonkeruujärjestelmän osia optimointiin, minkä ansiosta verkon kuormitusta saatiin pudotettua selvästi.

Tulevaisuudessa tiedonkeruujärjestelmälle tullaan toteuttamaan käyttöliittymä, jonka avulla tietokantaan tallennettua dataa voidaan esittää monipuolisemmin. Lisäksi tiedonkeruujärjestelmälle voisi toteuttaa kerättävien parametrien raja-arvojen määrittämisen. Tällöin käyttöliittymällä voitaisiin esittää käyttäjälle suoraan näkymä epänormaaleista tilanteista kerätyn datan perusteella.

LÄHTEET

ABB. (i.a.). *Drive Composer.* https://new.abb.com/drives/fi/ohjelmistotyokalut/drive-composer

Asiakastieto. (i.a.). *Pesmel Oy.* https://www.asiakastieto.fi/yritykset/fi/pesmel-oy/21203130/yleiskuva

Aura, L. & Tonteri, A. J. (1996). *Sähkökoneet ja tehoelektroniikan perusteet.* WSOY.

GeeksforGeeks. (i.a.). *SQL | DDL, DML, TCL and DML.* https://www.geeksforgeeks.org/sql-ddl-dml-tcl-dcl/

Hovi, A. (2004). *SQL-opas.* Docendo.

IBM. (i.a.-a). *Types of JDBC drivers.* https://www.ibm.com/docs/en/i/7.2?topic=jdbc-types-drivers

IBM. (i.a.-b). *Relational Databases.* https://www.ibm.com/cloud/learn/relational-databases

JavaTpoint. (i.a.-a). *Java JDBC Tutorial.* https://www.javatpoint.com/java-jdbc

JavaTpoint. (i.a.-b). *What is RDBMS.* https://www.javatpoint.com/what-is-rdbms

JetBrains. (i.a.). *IntelliJ IDEA.* https://www.jetbrains.com/idea/features/

Kepware. (i.a.-a). *OPC Interoperability: Open Connectivity Through Open Standards.* https://www.kepware.com/en-us/products/kepserverex/opc-interoperability/

Kepware. (i.a.-b). *KEPServerEX.* https://www.kepware.com/en-us/products/kepserverex/

Matrikon. (i.a.). *OPC Data Access (OPC DA) Version & Compatibility.* https://www.matrikonopc.com/opc-server/opc-data-access-versions.aspx

Microsoft. (i.a.-a). *Perustiedot tietokannasta.* https://support.microsoft.com/fi-fi/office/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204

Microsoft. (i.a.-b). *What is SQL Server Management Studio (SSMS)?.* https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

OPC Foundation. (i.a.). *What is OPC?.* https://opcfoundation.org/about/what-is-opc/

OPC Router. (i.a.). *What is OPC UA? A practical introduction.* https://www.opc-router.com/what-is-opc-ua/#OPC-Specifications

Oracle. (i.a.-a). *What is Database?* https://www.oracle.com/database/what-is-database/

Oracle. (i.a.-b). *What is a Relational Database (RDBMS) ?.* https://www.oracle.com/database/what-is-a-relational-database/

Orrberg, M., Ylinen, T., & Kauppila, J. (2017). *Sähköasennukset: 3* (4., uudistettu painos.). Sähköinfo Oy.

Pesmel. (i.a.). *About us.* https://pesmel.com/about-us/

Real Time Automation. (i.a.). *AN INTRODUCTION TO OPC UA.* https://www.rtautomation.com/technologies/opcua/

Schneider Electric. (i.a.). *General information on OPC DA.* https://product-help.schneider-electric.com/Machine%20Expert/V1.1/en/OPCDA/OPCDA/General_Info_on_OPC/General_Info_on_OPC.htm

Taylor, A. G. (2010). *SQL for dummies* (7th ed.). Wiley Publishing.

TechTarget. (i.a.). *Structured Query Language (SQL).* https://www.techtarget.com/searchdatamanagement/definition/SQL

Vesterholm, M. & Kyppö, J. (2010). *Java-ohjelmointi* (8. uud. p.). Talentum.

Wireshark. (i.a.). *About.* https://www.wireshark.org/