Dariush Moghadampour

NODE.JS:N TOIMINTA WINCC OA -JÄRJESTELMÄSSÄ

Tietotekniikka

2025

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Dariush Moghadampour
Opinnäytetyön nimi	Node.js:n toiminta WinCC OA -järjestelmässä
Vuosi	2025
Kieli	suomi
Sivumäärä	33
Ohjaaja	Harri Lehtinen

Tässä opinnäytetyössä tavoitteena oli kehittää automatisoitu ohjelmisto, joka hakee ja käsittelee suojareleiden COMTRADE-muodossa olevia vikaraportteja WinCC OA -järjestelmässä käyttäen Node.js:ää ja JavaScriptiä.

Työssä hyödynnettiin WinCC OA:n uusimman version 3.20 tarjoamia uusia ominaisuuksia, joissa Node.js ja JavaScript toimivat natiiveina työkaluina. Ohjelmiston tarkoituksena oli automatisoida vikaraporttien haku ja siirto releeltä FTP-protokollaa hyödyntäen sekä muuntaa COMTRADE muodossa olevat vikaraportit JSON-muotoon datan luettavuuden parantamiseksi.

Työn tuloksena syntyi kehityskelpoinen ratkaisu, jossa toteutuu suurin osa työlle asetetuista tavoitteista.

Avainsanat FTP, JSON, COMTRADE, Node.js, JavaScript

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Dariush Moghadampour
Title	Functionality of Node.js in WinCC OA system
Year	2025
Language	Finnish
Pages	33
Name of Supervisor	Harri Lehtinen

The goal of this thesis was to develop an automated disturbance recorder that retrieves and processes fault reports from protection relays in COMTRADE format within the WinCC OA system using Node.js and JavaScript.

The latest version 3.20 of WinCC OA was leveraged, utilizing the new capabilities where Node.js and JavaScript function as native tools.

The purpose of the software was to automate the retrieval and transfer of fault reports from the relay via the FTP protocol and to convert the COMTRADE format fault reports into JSON format to improve the readability of the data.

As a result of this work, a viable solution was developed that fulfills most of the set requirements and objectives for the project

Keywords FTP, JSON, COMTRADE, Node.js, JavaScript

SISÄLLYS

KUVAT

AVAINSANALUETTELO

AVOIN LÄHDEKOODI

Avoin lähdekoodi tarkoittaa ohjelmistoa, jonka lähdekoodi on vapaasti saatavilla, muokattavissa ja mahdollistaa kehittäjille sen muokkaamisen omien tarpeidensa mukaan.

JSON

JSON eli (JavaScript Object notation) on avoimen standardin tiedostomuoto tiedonvälitykseen ja tallennukseen.

COMTRADE

COMTRADE eli (Common Format for Transient Data Exchange for power systems) on tiedostomuoto oskillografian ja tilatietojen tallentamiseen.

NPM

NPM eli (Node Package Manager) on pakettien hallintaan sekä erilaisten JavaScript-kirjastojen ja moduulien asentamiseen tarkoitettu työkalu (NPM, 2023).

IEC-61850

IEC-61850 on älykkäiden sähköjärjestelmien, sekä sähkölaitteiden käyttämä protokolla. Se mahdollistaa eri valmistajien laitteiden integroinnin ja hallinnan.

FTP

FTP eli (File Transfer Protocol) on protokolla, joka on suunniteltu tiedostojen siirtoon tietokoneiden ja laitteiden välillä. FTP:n avulla voi ladata sekä lähettää tiedostoja niin internetin tai sisäisen verkon kautta.

JavaScript-kirjasto

JavaScript kirjasto on kokoelma esikirjoitettuja koodinpätkiä, joita voidaan liittää osaksi omaa projektia oman ohjelman tarpeiden mukaan (General assembly, 2021).

JavaScript-moduuli

JavaScript moduuli on esikirjoitettu koodinpätkä, joka voidaan liittää osaksi omaa projektia sen tarpeiden mukaan (Turing, n.d.).

Skripti

Skripti on sarja komentoja ja se voidaan suorittaa tietokoneella tai ohjelmointikielen avulla (Brave, 2023).

Node.js

Node.js on avoimen lähdekoodin ajoympäristö JavaScript ohjelmakoodin suorittamiseen (Node.js, n.d.).

1 JOHDANTO

Tämän opinnäytetyön aiheena on Node.js:n toiminta WinCC OA -valvomojärjestelmässä. Työ toteutetaan yhteistyössä VEO Oy:n kanssa. VEO on merkittävä toimija energia-alalla. Se on erityisesti erikoistunut automaatio-, energia- ja sähköistysratkaisuihin (VEO, n.d.). VEO on hyödyntänyt jo vuosien ajan WinCC OA-valvomoa lukuisissa eri projekteissaan, mutta tiettyjen prosessien hallintaan käytetään vielä ulkoisia ohjelmistoja. Nyt uusimmassa WinCC OA:n versiossa on tullut mahdolliseksi hyödyntää Node.js-pohjaisia ratkaisuja, joita myös VEOn on mahdollisesti tarkoitus hyödyntää, mikäli ne sopivat osaksi VEOn käyttötarkoitusta.

Tämän työn tavoitteena on selvittää, kuinka Node.js voidaan integroida WinCC OA-järjestelmään niin, että ulkoisten ohjelmistojen tarve vähenee, sekä selvittää miksi Node.js:n pohjalle rakennetut ohjelmistot ovat parempi vaihtoehto VEOn tarpeisiin. Mikäli tämä työ onnistuu, on sitä tarkoitus jatkokehittää ja ottaa se käyttöön osana VEOn PowerSCADA-toimitusta

Työssä keskitytään erityisesti COMTRADE-tiedostojen hallintaan, jonka useimmat suojareleet generoivat aina vika- tai laukaisutilanteessa. Tavoitteena on kehittää tätä varten JavaScript-pohjainen ohjelmisto, joka automatisoi näiden COMTRADE-tiedostojen hakemisen releiltä ja muuntaa ne JSON-muotoon luettavuuden helpottamiseksi, sekä tallentaa saadut tiedostot paikallisesti tai pilveen.

Aikaisemmin suojareleen luomat vikaraportit on saatu haltuun siten, että vikatilanteen sattuessa jonkun on täytynyt mennä fyysisesti paikan päälle imuroimaan saadut raportit suojareleeltä manuaalisesti. Tämän työn avulla olisi tarkoitus saada vikaraporttien haku automatisoitua käyttämällä FTP-tiedostonsiirtoprotokollaa siten, että vikaraportit saataisiin helposti ja nopeasti haltuun ja analysoitua, mikä nopeuttaa merkittävästi mahdollisten vikatilanteiden selvittämistä. Tällaisia ratkaisuja

on jo olemassa kuten Helios power solutions:in Axon Comtrade-järjestelmä. Sen tarkoituksena on kerätä, tallentaa ja käsitellä automaattisesti suojareleiden ja muiden sähköverkon laitteiden vikaraporttien hakeminen ja tapahtumatietojen lukeminen (Helios power solutions, n.d.). Tällaiset järjestelmät ovat yleistymässä modernin teollisuusautomaation keskuudessa, joten nyt myös VEOlla halutaan tutkia, olisiko tällaisen ratkaisun toteuttaminen mahdollista myös heidän valvomojärjestelmäänsä. WinCC OA:n versio 3.20 toi uutena mahdollisuutena JavaScript ja Node.js teknologioiden integroimismahdollisuuden, jonka avulla pystytään rakentamaan skaalautuvia ohjelmistoja ja liittämään ne suoraan yhteyteen SCADA-järjestelmän kanssa. Nämä teknologiat tarjoavat loistavaa suorituskykyä ja tarjoavat myös laajan kattauksen erilaisia kirjastoja ja moduuleja, joita voi käyttää apuna ohjelmistokehityksessä.

2 KÄYTETY TEKNOLOGIAT JA TIETORAKENTEET

Tässä opinnäytetyössä käytetään monia eri teknologioita sekä tietorakenteita. Näitä ovat muun muassa WinCC OA, Node.js, Javascript, JSON, sekä COMTRADE. Työssä sovelletaan jo olemassa olevaa tutkimustietoa ja dokumentaatiota, jota löytyy WinCC OA:n sekä Node.js:n sivuilta. Dokumentaatio pitää sisällään tietoa näiden teknologioiden yhteensopivuudesta.

2.1 WinCC OA -järjestelmä

WinCC OA on Siemensin kehittämä SCADA-järjestelmä, joka mahdollistaa erilaisten prosessien, tuotantovirtojen, koneiden ja laitosten visualisoinnin ja hallinnan. WinCC OA on suunniteltu erityisesti suurille ja monimutkaisille sovelluksille sekä projekteille, joissa tarvitaan mukautettuja toiminnallisuuksia (WinCC OA, n.d.).

WinCC OA tarjoaa skaalautuvia ratkaisuja, jotka mahdollistavat globaalin käytön verkon kautta. Tähän kuuluvat myös IOS- ja Android-pohjaiset käyttöliittymät sekä selainpohjaiset ohjelmistot (WinCC OA, n.d.).

2.2 Node.js

Node.js on avoimen lähdekoodin JavaScript pohjainen palvelinpuolen ympäristö. Node.js on todella suorituskykyinen sen V8 JavaScript moottorin ansiosta (Node.js, n.d.).

Node.js on suosittu front-end ohjelmistokehittäjien keskuudessa, mutta nyt sen käyttö on laajentunut myös teollisuusautomaatiossa ja valvomo-ohjelmistoissa. Tämä johtuu siitä, että Node.js mahdollistaa nopean tiedonsiirron, API-integraatiot ja reaaliaikaisen käsittelyn (Node.js, n.d.).

2.3 JavaScript

JavaScript on ohjelmointikieli, joka toimii niin selainympäristöissä, kuin myös palvelinpuolella Node.js:n avulla. Tässä projektissa JavaScriptiä käytetään COMTRADE- ja JSON-tiedostojen käsittelyyn, koska JavaScript tukee JSON-tiedostoformaattia, joka taas mahdollistaa tiedostojen muuntamisen helposti käsiteltävään ja luettavaan muotoon (MDN Web Docs, 2025).

Ohjelmointikielenä JavaScript on kevyt, tulkattava ja nopea sekä monipuolinen. JavaScriptiä käytetään pääsääntöisesti Web-ohjelmointiin ja se on suosittu erityisesti sen monipuolisuuden takia, sillä JavaScript toimii niin selaimessa, kuin myös palvelimella. Sen suuri ekosysteemi tarjoaa myös valtavan määrän kirjastoja, joiden avulla saadaan rakennettua skaalautuvia sekä interaktiivisia ohjelmistoja.

2.4 JSON

JSON on avoimen standardin tiedostomuoto tiedonvälitykseen ja tallennukseen (JSON, 2024). JSON on tiedostomuotona hyvä, koska sen sisältämä data on helposti luettavassa muodossa, jonka takia tässä työssä suoritetaan tiedostojen muuntaminen COMTRADESTA JSON:iin sen luettavuuden vuoksi. JSON perustuu myös JavaScriptin objektien syntaksiin, mutta se on keliriippumaton, joten sitä käytetään todella laajasti myös eri ohjelmointikielten kanssa.

2.5 COMTRADE

COMTRADE on tiedostomuoto, jota käytetään tallentamaan erilaisiin sähköjärjestelmiin liittyviä tilatietoja. Näitä tiedostoja luovat yleensä erilaiset älykkäät elektroniset laitteet, kuten suojareleet. Suojarele luo COMTRADE-tiedoston aina ennen ja jälkeen virhetilanteen. Tämän

avulla tiedetään, miten ja mistä releessä tapahtunut virhe tai vikatilanne johtuu (National Instruments, 2023).

2.6 NPM

NPM eli (Node Package Manager) on pakettien hallintaan sekä erilaisten JavaScript-kirjastojen ja moduulien asentamiseen tarkoitettu työkalu. Se on suosittu erityisesti JavaScript- ja TypeScript-kehittäjien keskuudessa. Tässä työssä sitä käytetään muun muassa yhteyden luomiseen WinCC OA:n sekä Node.js:n välille käyttämällä winccoa-manager moduulia. Mukaan kuuluu myös moduulit ESLint, Prettier sekä Basic-ftp, jota tarvitaan suojareleen FTP-serverin kanssa kommunikointiin (NPM, 2023).

2.7 FTP ja SFTP

FTP on tiedostojensiirtoon tarkoitettu internetprotokolla, jota käytetään tietokoneiden ja erilaisten laitteiden välillä. FTP on hyödyllinen kaikkeen mahdolliseen tiedostonsiirtoon erilaisten älykkäiden laitteiden välillä. Tässä työssä FTP:tä tarvitaan tiedostojen siirtoon releeltä tietokoneelle tai pilveen. Suojareleellä on oma sisäänrakennettu FTP palvelin, jonne se tallentaa vikaraportit (Gillis, 2024). Vaikka tässä työssä tullaan käyttämään FTP:tä sen yksinkertaisuuden takia, tullaan jatkossa siirtymään turvallisempaan tiedostonsiirtoprotokollaan. Tällainen on esimerkiksi SFTP eli Secure File Transfer Protocol. SFTP on tyypillisin vaihtoehto, jota käytetään FTP:n sijasta sen turvallisuuden takia. FTP lähettää datan aina ”selväkielisenä, joka tekee siitä helposti siepattavan, kun taas SFTP salaa lähetetyn datan SSH:n avulla (RunCloud, 2021).

SSH on protokolla, jota käytetään muun muassa etänä kirjautumiseen, tunnistautumiseen ja yhteyden ottamiseen eri laitteille. SSH:n tehtävä

on salata verkon yli liikkuva data ja sitä on syytä käyttää aina, jos lähetetään arkaluontoista dataa esimerkiksi yrityksen tiedostoja, tietoja tai salasanoja (Ylönen & Longvick, 2006). Tämän työn aikana ei ole vielä nähty tarpeelliseksi käyttää SFTP:tä tiedoston siirtoon, sillä kaikki tiedostonsiirrot tapahtuvat tällä hetkellä eristetyssä aliverkossa, josta ei ole pääsyä internetiin tai ulkoisiin verkkoihin. Tässä ympäristössä ei siksi ole kriittistä suojata salauksella, koska verkko on jo täysin hallinnassa ja suojattu ulkopuoliselta liikenteeltä.

2.8 IEC-61850

IEC-61850 on älykkäiden sähkölaitteiden sekä sähköjärjestelmien käyttämä protokolla, joka mahdollistaa laitteiden ja järjestelmien kommunikoinnin valmistajasta riippumatta (ABB, n.d.). Tässä työssä suojarele käyttää tätä protokollaa WinCC OA valvomon kanssa kommunikoimiseen. IEC-61850 käyttää myös erilaisia dataobjekteja ja tageja, jotka liittyvät erilaisiin mittaustietoihin. WinCC OA voi lukea näitä tageja ja reagoida niihin.

2.9 Teknologioiden integrointi ja käytännön toteutus

Edellä mainittujen teknologioiden avulla on tarkoitus tutkia ja lähteä toteuttamaan tehokasta ja automatisoitua sekä VEOn tarpeisiin räätälöityä ratkaisua, joka voisi tulevaisuudessa korvata ulkoiset ratkaisut ja parantaa ennen kaikkea sähköverkon vikatapahtumien hallintaa. Toki ajatuksena on testata ensin järjestelmää miniatyyrimuodossa, jolloin sitä voidaan tulevaisuudessa jatkokehittää, mikäli näin halutaan tehdä.

3 TYÖN TOTEUTUS

3.1 Työn toteutus ja tarkoitus

Työn tavoitteena on tutkia VEOn käytössä olevaa Siemensin WinCC OA -valvomojärjestelmää ja sen uusimman version 3.20 mukana tulleita ominaisuuksia, joita ovat Node.js ja JavaScript. Tarkoituksena on rakentaa JavaScript- ja Node.js-pohjainen järjestelmä, joka hakee suojareleeltä sen luomat vikaraportit ja tallentaa ne joko paikallisesti tai pilveen ja kääntää COMTRADE-tiedostomuodon JSON-muotoon datan luettavuuden parantamiseksi.

Työssä tutkitaan myös tiedostonsiirtoprotokollaa (FTP), jota käytetään vikaraporttien siirtoon laitteiden välillä. Mikäli työ onnistuu, siitä voi tulla jatkokehityksen avulla osa VEOn PowerSCADA-toimitusta ja siten päästäisiin irti myös ulkoisten ohjelmistojen käytöstä. Tällainen ohjelmisto nopeuttaa myös vikatilanteiden ratkaisua, kun vikaraportit saadaan automaattisesti tuotua mahdollisesti, jopa SCADA-järjestelmään asti.

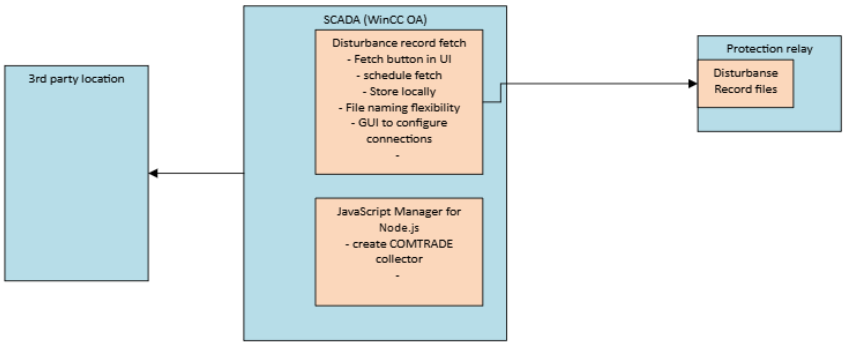
Työ on myös hyvin ajankohtainen Node.js:n osalta, koska siitä on tullut osa modernia automaatiota, joten olisi myös syytä tutkia, onko se hyvä ratkaisu myös VEOlle. Tämän työn avulla pyritään myös näyttämään, miten Node.js:n ja JavaScriptin skaalautuvalla ja mukautuvalla ohjelmistoarkkitehtuurilla voidaan kehittää sekä integroida omia ohjelmistoja erilaisiin automaatiojärjestelmiin ja nähdä, mitä hyötyjä näistä moderneista teknologioista on erilaisille automaatioratkaisuille.

3.2 Työn vaatimukset

Työn toteutus vaatii lisensoidun WinCC OA -järjestelmän, minkä tahansa suojareleen ja ohjelmointiympäristön. Tässä tapauksessa käytettiin Visual Studio Code -ohjelmointiympäristöä. Näiden lisäksi tarvittiin asennettu Node.js sekä erilaisia kirjastoja, jotka mahdollistavat erilaisia toiminnallisuuksia esimerkiksi FTP:n käytön ohjelmassa (Basic-ftp). Kirjastot asennetaan käyttämällä Node Package Manageria (NPM), joka on pakettien ja JavaScript-kirjastojen ja moduulien asentamiseen sekä hallintaan tarkoitettu työkalu.

3.3 Työn vaiheet

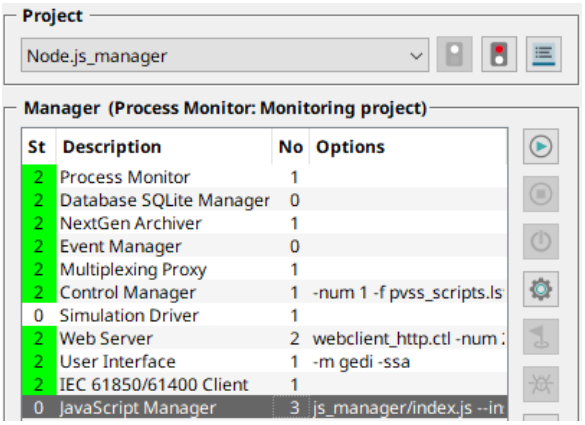
Työ alkoi sillä, että tehtiin kaavio, jonka avulla pystyttiin visualisoimaan varsinaisen tuotoksen toiminnallisuutta (Kuva 1.). Tämän jälkeen asennettiin viimeisin WinCC OA:n versio 3.20. Kun valvomojärjestelmä oli asennettu, luotiin uusi projekti, jonne määriteltiin kaksi uutta manageria. IEC-61850 Client sekä JavaScript manager. Managerit ovat moduuleja WinCC OA -järjestelmässä ja ne vastaavat erilaisista tehtävistä (WinCC OA, n.d.).



Kuva 1. Järjestelmän kaavio.

Jokaisella managerilla on oma id ja oma tehtävä. Tässä tapauksessa IEC-61850 Client manager kerää tietoa suojareleestä ja tuo sen valvomoon ja puolestaan JavaScript manager mahdollistaa JavaScript koodin

suorittamisen. Kuvasta 2 voi nähdä projektissa käytetyt managerit. Kun itse valvomoympäristö on valmis, sinne voidaan lisätä erilaisia älykkäitä sähkölaiteita kuten esimerkiksi suojareleitä.



Kuva 2. Managerit WinCC OA -järjestelmässä.

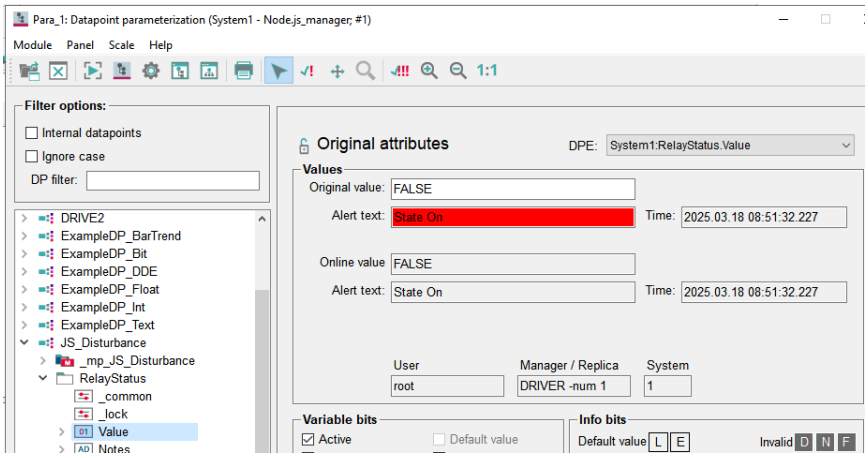
Tässä työssä käytettiin Schneiderin P3F30-suojarelettä (Kuva 3.). Suojarele on laite, jonka tehtävänä on havaita erilaiset viat tai vaaralliset olosuhteet sähköjärjestelmässä. Toisin sanoen suojarele on sähköverkon ”turvalaite”, jonka tehtävänä on eristää vikapaikka muusta sähköverkosta nopeasti ja automaattisesti heti vian sattuessa (Eaton, n.d.).

Kuten muutkin laitteet, suojarele täytyy myös konfiguroida oikein, jotta se kykenee kommunikoimaan esimerkiksi SCADA-järjestelmien kanssa. Myös FTP-protokolla täytyy ottaa erikseen käyttöön releen asetuksista, tämä käynnistää releen sisäisen FTP-palvelimen, johon rele tallentaa vikatilanteessa luodut vikaraportit. Kun konfigurointi on valmis, releestä saadaan ladattua ulos SCL-tiedosto. SCL-tiedosto pitää sisällään releen konfigurointitiedot, kuten viestintäasetukset ja datapisteet ja se toimii releen ”ajurina” WinCC OA -järjestelmälle. Kun valvomon ja releen välinen kommunikointi oli luotu, sille luotiin oma datapiste valvomon kautta. Datapisteet edustavat valvomon konkreettisia komponentteja, joita WinCC OA -prosessinohjausjärjestelmä ohjaa (WinCC OA, n.d.).

Kuva 3. Schneider P3F30 suojarele.

Datapisteen tarkoituksena on simuloida releessä tapahtunutta vikatilannetta. Datapiste luodaan valvomosta käsin aikaisemmin annettujen konfigurointitietojen avulla. Datapiste voi liittyä mihin tahansa mittaukseen ja tässä työssä sitä simuloidaan jännitemittaukseen liitetyllä booleantyyppisellä datapisteellä. Datapisteen tarkoituksena on tässä tapauksessa reagoida ylijännitteeseen, jolloin suojarele laukeaa ja valvomo puolestaan ilmoittaa siitä (WinCC OA, n.d.).

Kuvasta 4 voidaan nähdä, miten datapiste reagoi, kun suojarele on laukaistu. Kuvasta 4 voi huomata, että ”state on” -bitti on punaisella, mikä tarkoittaa sitä, että releellä on tapahtunut jokin virhe tai releeseen ei saada yhteyttä. Kun laukaisu kuitataan valvomosta state bitin tila palautuu normaaliksi. Tämä ei kuitenkaan tarkoita sitä, että vika olisi poistunut itse releeltä, mutta ajatuksena onkin, että valvomon operaattori saa välittömän tiedon mahdollisesta vikatilanteesta. Tällöin myös tiedetään, että suojarele on luonut vikaraportit, jotka voidaan nyt noutaa releeltä JavaScriptin sekä Node.js:n avulla. Se mahdollistaa nopean pääsyn vikaraportteihin, jolloin päästään välittömästi tutkimaan syytä releen laukeamiselle.



Kuva 4. Datapiste valvomossa.

Kun suurin osa kommunikoinneista oli saatu toimimaan, aloitettiin itse skriptin kehitys. Skriptin ajatuksena on automaattisesti kirjautua suojareleen FTP-palvelimelle, hakea sieltä vikaraportit sekä tallentaa saadut tiedostot. Luomassani skriptissä tallennus tapahtuu paikallisesti, sillä niin on helpoin testata sen toimivuutta. Skripti itsessään on suhteellisen yksinkertainen. Aluksi määritellään sille tarvittavat Node.js-kirjastot (Kuva 5.) Näiden kirjastojen avulla mahdollistetaan FTP-yhteys, tiedostojen luonti ja hakemistojen tarkistus sekä tiedostopolkujen käsittely.

```
const ftp = require('basic-ftp');
const fs = require('fs');
const path = require('path');
```

Kuva 5. Käytetyt kirjastot.

Tämän jälkeen luotiin tiedostojen lataamista varten funktio, jolle annettiin FTP-yhteydelle välttämättömät parametrit, joidenka avulla päästään sisään releen FTP-palvelimelle ja lataamaan itse tiedostot (Kuva 6.)

```
async function downloadComtradeFiles(ftpHost, ftpUser, ftpPassword, remoteDir, localDir) {
    const client = new ftp.Client();
    client.ftp.verbose = true;
    try {
        console.log("Connecting to FTP server");
        await client.access({
            host: ftpHost,
            user: ftpUser,
            password: ftpPassword,
            secure: false
        });

        console.log("Connected to FTP server");
```

Kuva 6. Yhteyden luonti FTP-palvelimelle.

Kun yhteys on saatu, haetaan kaikki .dat- ja .cfg-päätteiset tiedostot (Kuva 7) ja ladataan ne releeltä. Skriptiin on tehty myös virheen käsittelyä try-catch-lohkoa käyttämällä. Kaikki virheet tallennetaan error-muuttujaan ja mahdolliset virheet kirjoitetaan konsoliin (Kuva 8). Kuvasta 8 voi huomata myös, kuinka yhteys suljetaan aina, kun tiedostojen lataus on suoritettu. Muuttujat ja funktiokutsu löytyvät koodin lopusta. Muuttujiksi on määritelty suojareleen IP-osoite, releen FTP-käyttäjä, FTP-salasana, etähakemisto sekä paikallinen hakemisto, jonne haetut tiedostot tallennetaan (Kuva 9).

```
for (const file of files) {
    if (file.name.endsWith(".cfg") || file.name.endsWith(".dat")) {
        const localFilePath = path.join(localDir, file.name);
        console.log(`Downloading: ${file.name}...`);
        await client.downloadTo(localFilePath, file.name);
        console.log(`Saved: ${localFilePath}`);
    }
}

console.log("Download complete");
```

Kuva 7. Tiedostojen lataus.

```
try {
    console.log("Connecting to FTP server");
    await client.access({
        host: ftpHost,
        user: ftpUser,
        password: ftpPassword,
        secure: false
    });

    console.log("Connected to FTP server");

    await client.cd(remoteDir);

    const files = await client.list();
    console.log("Files in remote directory: ", files.map(f => f.name));

    if (!fs.existsSync(localDir)) {
        fs.mkdirSync(localDir);
    }

    for (const file of files) {
        if (file.name.endsWith(".cfg") || file.name.endsWith(".dat")) {
            const localFilePath = path.join(localDir, file.name);
            console.log(`Downloading: ${file.name}...`);
            await client.downloadTo(localFilePath, file.name);
            console.log(`Saved: ${localFilePath}`);
        }
    }

    console.log("Download complete");

} catch (error) {
    console.error(error);
} finally {
    client.close();
}
```

Kuva 8. Virheenkäsittely ohjelmassa.

```
const FTP_HOST = "192.168.50.9";
const FTP_USER = "conf";
const FTP_PASSWORD = "2";
const REMOTE_DIR = "/";
const LOCAL_DIR = "C:/comtrade";

downloadComtradeFiles(FTP_HOST, FTP_USER, FTP_PASSWORD, REMOTE_DIR, LOCAL_DIR);
```

Kuva 9. Funktiokutsu ja määritellyt muuttujat.

3.4 Aineiston keruu

Aineistoa on saatu kerättyä työn aikana monella eri tavalla, esimerkiksi WinCC OA:n järjestelmälokia lukemalla ja koodieditorin konsolista. Valvomon järjestelmälokista eli log vieweristä näkee aina reaaliaikaisesti järjestelmän käynnistyksen, virheet ja hälytykset. Sen avulla näkee myös, onko järjestelmä käynnistynyt onnistuneesti (Kuva 10). Koodieditorista näkee taas tiedostojen siirtoajat ja koodissa tapahtuvat virheet, sekä ohjelman suorituksen tapahtumat (Kuva 11).

```
WCCOAiec61850(1), 2025.03.28 11:09:18.082, SYS,  SEVERE,     54, Unexpected state, IEC61850IEDConnection::initiateResponse, Connection attempt failed for server BA0901_P3F30 Error: Timed Out
WCCOAiec61850(1), 2025.03.28 11:09:18.111, SYS,  INFO,        0, Disconnect request for server [primary]BA0901_P3F30, clearing all queues
WCCOAiec61850(1), 2025.03.28 11:09:18.113, SYS,  INFO,        0, IEC61850IEDConnection::finalize, BA0901_P3F30
WCCOAiec61850(1), 2025.03.28 11:09:49.025, SYS,  INFO,        0, IEC61850IEDConnection::createConnection, Start creating connection for BA0901_P3F30
WCCOAiec61850(1), 2025.03.28 11:09:49.026, SYS,  INFO,        0, IEC61850IEDConnection::createConnection, Done creating connection for BA0901_P3F30
```

Kuva 10. WinCC OA log viewer.

```
Connecting to FTP server
Connected to 192.168.50.9:21 (No encryption)
< 220 VampG5 FTP server ready

> OPTS UTF8 ON
< 501 not supported

Login security: No encryption
> USER conf
< 331 Password required for conf

> PASS ###
< 230 User conf logged in

> FEAT
< 500 command not implemented

> TYPE I
< 200 Type set to I

> STRU F
< 500 command not implemented

> OPTS UTF8 ON
< 501 not supported

Connected to FTP server
> CWD /
< 250 CWD command successful

Trying to find optimal transfer strategy...
```

Kuva 11. FTP-konsolin loki.

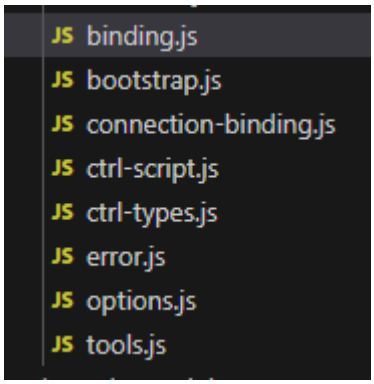
4 TYÖN TULOSTEN ANALYSOINTI

Tämän työn aikana saatiin paljon arvokasta tietoa Node.js- ja JavaScript- teknologioista ja niiden integroinnista automaatiojärjestelmiin. Vaikka työ itsessään on vielä kesken, tämänhetkiset havainnot ovat antaneet mahdollisuuden arvioida näiden teknologioiden käyttöä osana VEOn automaatiojärjestelmiä.

4.1 Työn tulokset

Työn lopputuloksena onnituttiin kehittämään Node.js- ja JavaScript-pohjainen järjestelmä, joka kykenee kommunikoimaan suojareleen kanssa ja hakemaan FTP:n avulla sieltä vikaraportteja ja tallentamaan ne. Ohjelmiston, ja suojareleen kommunikointi valvomon kanssa myös onnistuu, joten suurin osa kaikesta kommunikoinnista saatiin myös toteutettua. Työhön mahtuu myös ominaisuuksia, joita ei onnistuttu toteuttamaan, kuten COMTRADE-tiedostojen muuntaminen JSON- muotoon. Se ei onnistunut, sillä Node.js ei tarjoa vielä sellaisia moduuleja, joiden avulla se onnistuisi.

Node.js:n ja WinCC OA -valvomon kommunikointi onnistui myös suurimmaksi osaksi, mutta jostain syystä valvomoon liitetty JavaScript manager ei käynnisty. JavaScript manager käynnistyy managerille annettujen manager-parametrien avulla (WinCC OA n.d.) ja winccoa-connection moduulin avulla, jotka voi nähdä kuvista 12 ja 13.

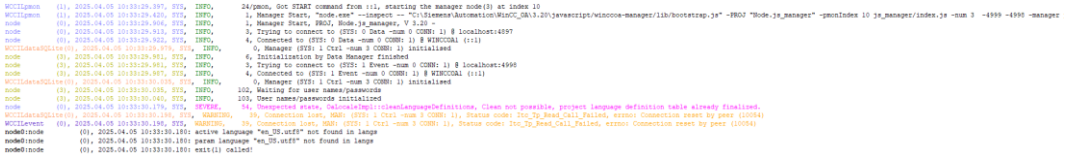


Kuva 12. WinCC OA:n käyttämät JavaScript-kirjastot.

```
//@ts-expect-error Importing an AddOn without type declaration
const winccoaconnection_1 = require("../../../bin/winccoaconnection.node");
class ConnectionBinding {
```

Kuva 13. winccoa-connection-moduuli.

winccoa-connection-moduuli käyttää node-tiedostoa, joka mahdollistaa Node.js:n ja WinCC OA:n välillä tapahtuvan tiedonsiirron ja toiminnallisuuden, kuten datapisteiden lukemisen. Valvomossa syntynyt ongelma on mitä luultavimmin WinCC OA -valvomossa, mutta siihen ei ole vielä keksitty ratkaisua. JavaScript managerin lokista voikin huomata, että valvomo selkeästi saa kutsun käynnistykselle, joten itse yhteys valvomon ja Node.js:n välillä toimii, mutta jostain syystä kuitenkin lopulta kaatuu käynnistysprosessin aikana (Kuva 14).



Kuva 14. JavaScript-managerin käynnistys.

4.2 Tulosten tarkastelu ja johtopäätökset

Työn ja tutkimuksen lähtökohtana oli tutkia, miten WinCC OA ja Node.js toimivat yhdessä sekä kehittää Node.js- ja JavaScript-pohjainen ohjelmisto erilaisia kirjastoja ja moduuleja hyödyntämällä sekä rakentaa oma ohjelmisto, joka voidaan integroida WinCC OA -valvomoon. Työssä suurimmassa osassa oli ehdottomasti COMTRADE-tiedostojen käsittely ja muuntaminen sekä tallentaminen.

Saatujen tulosten perusteella voidaan todeta, että suurin osa työn tavoitteista saatiin toteutettua. Yhteys suojareleen, tuotetun ohjelman ja WinCC OA:n välillä toimii ja tiedostojen nouto sekä tallennus pystytään toteuttamaan luotettavasti. Toteuttamatta jäi työn alussa suunniteltu COMTRADE-tiedostojen muuntaminen JSON-muotoon, sillä tähän ei ollut vielä kehitetty minkäänlaista JavaScript-moduulia tai kirjastoa, jolla tämä olisi saatu onnistumaan. Myös WinCC OA:n ja Node.js:n välinen yhteys toimii, mutta jostain tuntemattomasta syystä valvomon JavaScript manager ei käynnisty.

Ongelmista huolimatta saatiin kuitenkin kerättyä paljon tietoa WinCC OA:sta ja moderneista web-teknologioista ja huomattiin, että niiden hyödyntäminen teollisuusautomaatiossa on täysin toimiva ja käyttökelpoinen ratkaisu, kunhan mahdolliset järjestelmän ja teknologioiden rajoitteet ja vaatimukset otettiin huomioon. Tämä ohjelmisto on kehitettävissä ja jatkokehityksen avulla, sitä voitaisiin viedä pidemmälle ja mahdollisesti käyttää osana VEOn PowerSCADA-toimitusta. Tämä ohjelmisto parantaisi valvomojärjestelmän itsenäisyyttä ja reagointikykyä erilaisiin vikatilanteisiin sen automaattisen tiedonsiirron avulla, mikä vuorostaan tehostaa vian hallintaa, koska tieto virheistä saadaan nopeasti.

Työn myötä myös VEO on saanut kuvan siitä, miten modernia web-teknologiaa voidaan hyödyntää osana teollisuusautomaatiota. Lisäksi vastataan kysymykseen, onko tämä ratkaisu pitkällä aikavälillä VEOn käyttötarkoitukseen käyttökelpoinen vaihtoehto. Tulevaisuudessa voidaan

mahdollisesti tutkia, onko olemassa joitain muita teknologioita, joita voitaisiin hyödyntää valvomojärjestelmien rinnalla tai sisällä. lisäksi voitaisiin tutkia esimerkiksi muita tiedoston siirtoon olevia erilaisia tapoja kuin FTP ja kirjastot, esimerkiksi jotain rajapintaa. Tässä työssä haluttiin nähdä nimenomaan JavaScriptin ja Node.js:n tarjoamien moduulien ja kirjastojen ominaisuuksia.

5 YHTEENVETO

Työn aiheena oli tutkia Node.js- ja JavaScript-teknologioita ja niiden toimintaa WinCC OA -valvomossa ja saada lopuksi aikaan sisäänrakennettu ohjelmisto, jonka avulla pystytään hakemaan suojareleeltä COMTRADE-muodossa olevia vikaraportteja automaattisesti virhetilanteen sattuessa ja tallentaa ne. Vaikka projekti on vielä hieman kesken, tähän mennessä saavutetut tulokset tarjoavat runsaasti tärkeää tietoa ja antavat myös suunnan mahdolliselle jatkokehitykselle. Tämänhetkiset havainnot ja analyysit ovat antaneet mahdollisuuden arvioida modernien web-ohjelmointikielien ja WinCC OA -valvomojärjestelmän yhteensopivuutta sekä niiden käytöstä teollisuusautomaatiossa. Seuraavissa luvuissa tullaan käsittelemään muun muassa tulosten suhdetta tavoitteisiin, eettisiä näkökohtia sekä mahdollisia jatkokehityksen kohteita, jolla järjestelmää saataisiin jatkojalostettua tulevaisuudessa.

Työ eteni vaivattomasti ja suurimmaksi osaksi suunnitelmien mukaan. Toteutus vaati iteratiivisia kokeiluja muun muassa tiedostonsiirron osalta. Työn aikana jouduttiin myös hieman muuttamaan suunnitelmaa, kun kävi ilmi, että COMTRADE-tiedostoja ei pysty muuntamaan JSON-muotoon. Tämä ei kuitenkaan vaikuttanut kriittisesti työn etenemiseen. Työtä tehtäessä myös oma osaamien Node.js:n ja WinCC OA:n osalta kehittyi huomattavasti ja opin myös paljon uutta teollisuusautomaatiosta. Kehitys tuki myös huomattavasti työn etenemistä.

Suurin osa tämän opinnäytetyön tavoitteista saavutettiin. Työn aikana saatiin rakennettua onnistuneesti WinCC OA valvomoympäristö, joka kommunikoi releen ja tuotetun ohjelmiston kanssa, sekä itse suojareleen kanssa sinne määritellyn datapisteen kautta. Työn tulosten perusteella voidaan onnistuneesti todistaa, että Node.js voidaan integroida WinCC OA valvomoon ja sen, että suojareleen vikaraporttien haku voidaan hakea JavaScriptin sekä Node.js:n tarjoamien kirjastojen ja moduulien avulla. Ainoat saavuttamattomat olivat JavaScript managerin käynnistymisongelmat ja COMTRADE-tiedostojen muuntaminen.

Työn aikana noudatettiin hyvää tutkimusetiikkaa. Kaikki työn aikana tehdyt testaukset toteutettiin erillisessä virtuaalisessa kehitysympäristössä. Kaikki työssä lisenssin vaatineet ohjelmistot olivat täysin laillisesti käytössä ja erilaisia tietoturvanäkökohtia otettiin huomioon järjestelmäyhteyksissä. Työn aikana käytettiin vain sisäistä verkkoa. Myös kaikessa dokumentoinnissa otettiin huomioon, ettei mitään luottamuksellista tai organisaatiokohtaista tietoa paljasteta.

Tällaisessa valvomojärjestelmään rakennetussa ohjelmistossa ja tiedoston siirrossa on monenlaisia asioita, joita voidaan kehittää. Päällimmäisenä olisi jatkokehityksessä hyvä ottaa huomioon tiedostonsiirtoprotokollat ja panostaa tiedostonsiirron turvallisuuteen sekä tutkia, olisiko mahdollista siirtyä johonkin turvallisempaan protokollaan tai käyttää jonkinlaista rajapintaa tiedoston siirtoon. Työn aikana oli myös pohdintaa graafisesta käyttöliittymästä, jonka avulla haetut vikaraportit saataisiin tuotua suoraan SCADA-järjestelmään ilman, että niitä tarvitsee itse käydä hakemistosta avaamassa. Tämä lisäisi merkittävästi valvomon käyttäjäystävällisyyttä, koska tiedostoihin käsiksi pääseminen on näin vaivatonta ja raporttien visualisointi onnistuisi myös helpommin.

Työssä yksi osa-alue oli myös aiemmin mainittu COMTRADE-tiedostojen muuntaminen. Tulevaisuutta ajatellen olisi hyvä, jos vikaraporttien data saataisiin helposti luettavaksi. Tätä ominaisuutta varten voitaisiin mahdollisesti, koittaa kehittää oma kirjasto, joka pystyisi parseroida COMTRADE-tiedoston JSON-muotoon tai vaihtoehtoisesti tutkia, jos tulevaisuudessa löytyisi jonkinlainen parser kirjasto. JSON-dataa voisi myös käyttää esimerkiksi järjestelmän sisäisessä analytiikassa muun muassa automatisoitujen hälytysten ja historiatietojen analysoinnin muodossa. Myös kehitettyä ohjelmaa täytyy jatkokehittää lisäämällä virheenkäsittelyä, sillä nykyisessä versiossa se on hyvin perustavanlaatuista. Tulevaisuudessa olisi tärkeää tehdä mahdollisimman vankkarakenteinen virheenkäsittely ja myös jonkinlainen loki, joka mahdollistaa vikadiagnostiikan itse ohjelmistossa.

Aiemmin otettiin esiin myös tiedostonsiirtoprotokolla FTP, jota käytettiin tässä projektissa, vaikka se on täysin toimiva ratkaisu se ei kuitenkaan ole kovin turvallinen. Sen sijaan voisi siirtyä käyttämään paremmin suojattua SFTP:tä, joka salaa lähetetyn datan. Näin varmistetaan se, että tiedostojen luottamuksellisuus ja eheys säilyy, sekä luottamuksellista dataa ei vuoda ulkopuolelle.

Tiedostonsiirron suojaaminen on todella tärkeää, sillä nykyisessä maailmantilanteessa kyberuhkat ovat yleistyneet ja hyökkääjät ovat usein myös hyvin ammattimaisia ja käyttävät kehittyneitä menetelmiä, kuten tietojenkalastelua ja haittaohjelmia datan varastamiseen (Asianet broadband, n.d.). Tällaisen järjestelmän on oltava myös turvallinen, sillä se käsittelee oikeassa tilanteessa jonkin oikean projektin tiedostoja, joita ei haluta vuotaa ulkopuolelle.

Työ kehitti omaa ymmärrystäni automaatiosta, ohjelmistokehityksestä ja niiden yhdistämisestä. Kun työ alkoi, WinCC OA oli minulle täysin tuntematon järjestelmä. Myös Node.js-teknologiana oli lähes täysin tuntematon minulle. Työn ansiosta opin erityisesti avoimen lähdekoodin työkaluista ja niiden käytöstä sekä Node.js:n ja JavaScriptin kirjastoekosysteemistä ja moduuleista. Koen erityisen arvokkaaksi sen, että pystyin käyttämään omia ohjelmointitaitojani tällaisen käytännön ongelman ratkaisemiseksi.

Vaikka työ ei täysin valmistunut koen, että sillä on potentiaalia kehittyä ja olla osana VEOn PowerSCADA-toimitusta. Uskon myös, että tulevaisuudessa vastaavanlaiset ohjelmistot tulevat lisääntymään, koska Node.js:n ja JavaScriptin käyttö lisääntyy jatkuvasti myös automaatiototeutuksissa. Työ vahvisti myös minulle sen, että ohjelmistokehitys on tulossa osaksi automaatioteollisuutta, mikä avaa varmasti runsaasti uusia urapolkuja niin automaatio kuin ohjelmistoalan osaajille.

Tämän työn tulokset tarjoavat VEOlle hyvän pohjan kehittää omia sisäisiä ohjelmistoja automaatiojärjestelmiin. Mikäli tätä ohjelmistoa kehitellään ja laajennetaan tulevaisuudessa, sen tuomat edut voivat parantaa

automaatiojärjestelmien tehokkuutta ja nopeutta. Esimerkiksi automatisoitu vikaraporttien haku ja käsittely vapauttaa resursseja manuaalisista toiminnoista. Tulevaisuudessa ohjelmiston kehittäminen mahdollistaa myös monipuolisempia toimintoja reaaliaikaisten hälytyksien ja raportointityökalujen muodossa, jotka ovat suoraan WinCC OA -järjestelmässä. Tällaiset ohjelmistot voivat avata myös uusia ovia teollisuusautomaatiossa, jos ohjelmointi otettaisiin osaksi automaatioprojekteja. Näin saataisiin räätälöityä VEOn asiakkaille entistä yksilöllisemmät järjestelmät juuri heidän käyttöään varten.

LÄHTEET

ABB. (n.d.). *IEC 61850*. Noudettu 17.2.2025 osoitteesta https://new.abb.com/control-systems/fi/system-800xa/hajautettu-800xa-ohjausjarjestelma/kenttavaylaprotokollat/iec-61850

Asianet broadband (n.d.). *Why File Encryption is Essential in Today's Digital World.* Noudettu 24.4.2025 osoitteesta https://asianet-broadband.in/why-file-encryption-is-essential-in-todays-digital-world/

Brave. (2023). *What is a script?* Noudettu 6.4.2025 osoitteesta https://brave.com/glossary/script/

Eaton. (n.d.) *What is a protective relay?* Noudettu 17.4.2025 osoitteesta https://www.eaton.com/sg/en-us/products/electrical-circuit-protection/fundamentals-of-protective-relays.html

Gillis, A. S. (2024). *What is FTP?* TechTarget. Noudettu 17.2.2025 osoitteesta https://www.techtarget.com/searchnetworking/definition/File-Transfer-Protocol-FTP

General Assembly. (2021). *What is a JavaScript library?* Noudettu 6.4.2025 osoitteesta https://generalassemb.ly/blog/what-is-a-javascript-library/

Helios power solutions. (n.d.). *Axon Comtrade – Automatic electrical protection collection system.* Noudettu 17.4.2025 osoitteesta https://heliosps.com/product/axon-comtrade-automatic-electrical-protection-collection-system/

MDN Web Docs. (2025). *JavaScript.* Noudettu 17.4.2025 osoitteesta https://developer.mozilla.org/en-US/docs/Web/JavaScript

National Instruments. (2023). *COMTRADE File Format (Electrical Power Toolkit).* Noudettu 17.2.2025 osoitteesta https://www.ni.com/docs/en-US/bundle/labview-electrical-power-toolkit-api-ref/page/lveptconcepts/comtrade_format.html?srsltid=AfmBOop-3fPehRlH_A0vUHRdy3EwQvrQ0hRX5S104MXWDRVruGxwFNAa

Node.js. (n.d.). *About Node.js*. Noudettu 6.4.2025 osoitteesta https://nodejs.org/en/about

Npm. (2023). *About npm*. Noudettu 31.3.2025 osoitteesta https://docs.npmjs.com/about-npm

RunCloud. (2021). *FTP vs. SFTP – What's The Difference & Why It Matters.* Noudettu 17.4.2025 osoitteesta https://runcloud.io/blog/ftp-vs-sftp

Turing. (n.d.) *Understanding JavaScript modules: Explanations & Examples.* Noudettu 6.4.2025 osoitteesta https://www.turing.com/kb/javascript-modules

VEO. (n.d.). *Yritys.* Noudettu 15.2.2025 osoitteesta https://veo.fi/fi/yritys/

JSON.org. (2024). *Introducing JSON.* Noudettu 17.2.2025 osoitteesta https://www.json.org/json-en.html

WinCC OA. (n.d -a). *Administration of managers.* Noudettu 5.4.2025 osoitteesta https://www.winccoa.com/documentation/WinCCOA/3.18/en_US/Pmon_Consolepanel/Pmon_Consolepanel-23.html

WinCC OA. (n.d. -b). *Create a data point*. Noudettu 25.3.2025 osoitteesta https://www.winccoa.com/documentation/WinCCOA/3.18/en_US/Referenz_PARA/Referenz_PARA-27.html

WinCC OA. (n.d. -c). *Data points as bearer of information.* Noudettu 31.3.2025 osoitteesta https://www.winccoa.com/documentation/WinCCOA/3.18/en_US/GettingStarted/GettingStarted-27.htmln

WinCC OA. (n.d. -d). *What is WinCC OA?* Noudettu 17.2.2025 osoitteesta https://www.winccoa.com/documentation/WinCCOA/3.18/en_US/GettingStarted/GettingStarted-02.html

WinCC OA. (n.d. -e). *WinCC OA JavaScript Manager for Node.js.* Noudettu 17.2.2025 osoitteesta https://www.winccoa.com/documentation/WinCCOA/latest/en_US/Pmon_Console-panel/Pmon_Consolepanel-23.html

Ylönen, T. & Longvick, C. (2006). *The Secure Shell (SSH) Protocol Architecture, RFC 4251 (Standards Track)*. IETF. https://www.ietf.org/rfc/rfc4251.txt