Eero Lehtinen

S7-1500 LOGIIKAN WEB-SERVERIN OMINAISUUDET

Automaatiotekniikan koulutusohjelma
2016

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

S7-1500 LOGIIKAN WEB-SERVERIN OMINAISUUDET

Lehtinen, Eero
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Marraskuu 2016
Ohjaaja: Suvela, Timo
Sivumäärä: 46
Liitteitä: 19

Asiasanat: Web-serveri, tehokkuus, logiikkaohjain, kaavio, OEE

Tämän opinnäytetyön pääasiallisena aiheena oli tutkia Siemensin S7-1500 -logiikan web-serverin ominaisuuksia. Työn tilaajana oli Raumaster Paper Oy.

Tavoitteena oli tutkia minkälaista ja miten diagnostiikka- ja tuotantodataa saataisiin esitettyä selaimessa. Ideana oli luoda muutama esimerkkisivu, joita Raumaster Paper Oy voisi sitten myöhemmissä projekteissaan hyödyntää.

Alun teoriaosuudessa käydään läpi KNL-laskentaa standardeja tutkimalla sekä web-servereiden toimintaa.

Käytännön osuudessa esitellään S7-1500 -logiikan web-serverin ominaisuuksia ja käydään niiden tarjoamia mahdollisuuksia läpi.

S7-1500 LOGIC CONTROLLER'S WEB SERVER FEATURES

Lehtinen, Eero
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Automation technology
November 2016
Supervisor: Suvela, Timo
Number of pages: 46
Appendices: 19

Keywords: Web server, effectiveness, logic controller, OEE, chart

---

The main purpose of this thesis was to study Siemens' S7-1500 logic controller's web server features. The client was Raumaster Paper Ltd.

The aim was to study how and what sort of diagnostics and production data could be displayed in a web browser. The idea was to create a few sample pages which Raumaster Paper could use in their later projects.

The theory section starts with OEE calculation and continues with web server functions.

The practical section presents the logic controller's web server features and what possibilities they offer.

SISÄLLYS

SYMBOLI- JA TERMILUETTELO

OEE – Overall Equipment Effectiveness. Laitteistojen tehokkuuden laskentaa.

TEEP- Total Effective Equipment Performance. Sama kuin OEE, mutta verrataan kalenteriaikaan.

HTML - Hypertext Markup Language. Ohjelmointikieli, jolla voi tehdä nettisivuja.

HTTP - Hypertext Transfer Protocol. Protokolla tiedonsiirtoon netissä.

TCP - Transmission Control Protocol. Protokolla yhteyksien luomiseen tietokoneiden välille netissä.

CSS - Cascading Style Sheets. Antaa tyyliohjeita HTML-dokumentille.

AJAX - Asynchronous JavaScript And XML. Tapa luoda tehokkaampi päivitys nettisivuille.

XML - Extensible Markup Language. Kuvauskieli, jota käytetään tiedonvälitykseen.

AWP - Automation Web Programming. Komentoja, jotka auttavat logiikan ja HTML-sivun välistä tiedonvaihtoa.

ENUM – Enumeration types. Muuttaa logiikan arvoja tekstiksi HTML-sivulla.

MIT-lisenssi on ohjelmistolisenssi, joka sallii ohjelmiston kaikenlaisen käytön, kunhan alkuperäiset tekijänoikeustiedot säilytetään.

Utf-8 -koodaustapa, joka määrittää mitä merkkejä voidaan käyttää kirjoitettaessa koodia.

CPU - Central Processing Unit. Tässä tilanteessa siis logiikkaohjain.

Watch table - Taulukko, jonne on listattu muuttujia monitoroitaviksi.

TIA Portal - Siemensin ohjelmointityökalu logiikoiden ja käyttöliittymien ohjelmointiin.

OB - Organization block. Organisaatiolohko, joka lohkolla oma tarkoitus. Esim. OB1:seen lisätään syklisiä ohjelmakutsuja.

PLC - Programmable Logic Controller. Ohjelmoitava logiikka.

DB – Data Block. Tietokanta, johon voidaan tallettaa tietoa.

1 JOHDANTO

Nykyaikainen automaatioteknologia hyödyntää nousevissa määrin internet-teknologiaa, jonka avulla on mahdollista saada esimerkiksi lähiverkon kautta suora yhteys järjestelmään. Moni logiikkaohjainten valmistaja tarjoaakin mahdollisen web-server -ominaisuuden, joko suoraan integroituna logiikkaan tai erillisenä moduulina.

Opinnäytetyön pääasiallisena aiheena oli tutkia Raumaster Paper Oy:lle Siemens S7-1500 logiikan web-serverin ominaisuuksia ja sen tuomia mahdollisuuksia diagnostiikkatietojen näyttämiseen selaimessa. Työn tarkoituksena on tehdä tilaajan määrittelemiä esimerkkisivuja, joista olisi hyötyä tulevaisuuden projekteissa.

Työssä käydään läpi web-serverissä olevia valmiita sivuja ja mitä mahdollisuuksia käyttäjän itse luomat sivut antavat. Selvitetään, millä tavoin logiikan muuttujien talletettuja tietoa pystyy näyttämään selaimella sekä miten ja minkälaista diagnostiikkatietoja saadaan luettua logiikasta.

Työssä tutkitaan myös laitteiston käytettävyyttä ja sen esittämistä standardien avulla. Tutustutaan myös hieman KNL-laskentaan ja tutkitaan, miten sitä pystyisi hyödyntämään.

Raumaster Paper Oy on vuonna 2003 perustettu automaattisia laitteistoja paperiteollisuuteen suunnitteleva ja valmistava yritys. Palvelut ja tuotteet sisältävät mm. rullankäsittelyä, pituusleikkureita, hylsynkäsittelyä ja automaattisia varastointijärjestelmiä. Vuonna 2015 Raumaster Paperin henkilökuntamäärä oli yli 50. Raumaster Paper on osa Raumaster konsernia. (Fonectan www-sivut 2016)

2 LAITTEISTON TEHOKKUUS

Tuottavuutta halutaan tavallisesti parantaa isoin askelin: ostetaan uusi kone, jolla tehtyjen tuotteiden valmistusaikoja verrataan aiempiin. Tekemiseen voi silti jäädä vanhan toimintatavan jäänteenä paljon tuhlausta: tehdään ylimääräisi kappaleita, käytetään huonoja asetustapoja, tulee korjaamista tai uudelleen tekemistä, tuotteiden siirtämistä, kääntelemistä tai pinoamista, työkalujen tai materiaalien etsimistä, turhia välivarastoja, odottelua ja muita häiriöitä. Muutama vuosi kuluu ja huonontuvia tuotantolukuja selitetään käyntihäiriöiden ja kulumisen yhteisvaikutuksena. (Villanen 2013)

Tuotantojärjestelmän tehokkuuden laskemiseksi on luotu OEE (Overall Equipment Effectiveness) Industry Standard (suom. KNL-laskenta). OEE:n idea on verrata teoreettista maksimitulosta saatuun tulokseen. Kyseisellä mallilla voidaan laskea yhden koneen tai vaikkapa koko tuotantolinjan kokonaistehokkuus. Kokonaistehokkuus saadaan laskettua kolmen eri muuttujan avulla.

- Käytettävyys (K)
- Toiminta-aste (N)
- Laatukerroin (L)

OEE:n tavoitteena on siis löytää kaikki tuotannossa tapahtuvat hävikit. Löytääkseen kaikki piilotetutkin hävikit, kaikki häviöt pitää kategorisoida. Saaduissa arvoissa tuloksen mittayksikkönä käytetään prosentteja. Laskiessa on tärkeää huomata, että mikään arvo ei ole yli 100% (OEE Industry Standardin www-sivut 2016)

2.1 Käytettävyys (K)

Käytettävyyttä voidaan lähteä hahmottamaan miettimällä, tekeekö kone sille annettua tehtävää.

$$\frac{Käyntiaika}{Käyntiaika + Seisokkiaika}$$

Missä

Käyntiaika on aika, jolloin kone tekee sille annettua tehtävää.

Seisokkiaika, on ajanjakso jolloin kohde ei ole tuotannossa käytön tai kunnossapidon vaatimien toimenpiteiden vuoksi. Seisokkiaikaan vaikuttavia käytön toimenpiteitä ovat mm. asetukset, säädöt, puhdistukset, kuluvien osien vaihdot. (PSK 6201 2016, 5)

Esimerkiksi 8 tunnin työvuoron aikana kone voi olla 480 minuuttia päällä, josta se on toiminnassa 360 minuuttia. Saadaan

$$\frac{360\ min}{480\ min} = 0{,}75 \rightarrow 75\%$$

(OEE Industry Standardin www-sivut 2016)

Käytettävyyden hävikit kategorisoidaan yleensä kahteen eri pääluokkaan:

- Suunnittelematon pysähdys. Kattaa ajat, jolloin kone tai laitteisto on suunniteltu olevan käynnissä, mutta ei ole jonkinlaisesta viasta johtuen. Yleisimpiä syitä voivat olla muun muassa konerikko, työkalun rikkoutuminen, suunnittelematon kunnossapito. Hieman harvinaisempina syinä voi olla muun muassa koneenkäyttäjän tai materiaalin puuttuminen ja linjaston tukkiutuminen. Yleissääntönä suunnittelemattoman pysähdyksen ja lyhyen pysähdyksen voi erottaa vaikkapa sopimalla, että kaikki kestoltaan yli 2 minuuttia kestävät pysähdykset kuuluvat suunnittelemattomiin pysähdyksiin.
- Suunniteltu pysähdys. Kattaa ajat, jolloin kone tai laitteisto on suunniteltu olevan käynnissä, mutta ei ole johtuen asetusten säätämisestä. Yleisimpiä syitä voivat olla muun muassa koneen siirtyminen toiseen tilaan, asetuksien muuttaminen ja työkalujen säätö. Harvinaisempina syinä voi olla muun muassa laitteiston puhdistus, käynnistysaika, suunniteltu kunnossapito ja laadun tarkistus. (Vorne Industries 2016.)

2.2 Toiminta-aste (N)

Toiminta-astetta/suorituskykyä voidaan lähteä hahmottamaan miettimällä millä nopeudella kone työskentelee.

$$\frac{Tuotanto}{Nimellistuotantokyky\ x\ Käyntiaika}$$

Missä

Tuotanto on tuotannon määrä, sekä hyvät että huonot osat.

Nimellistuotantokyky on valmistajan ilmoittama nimellistuotantokyky tai erikseen selvitetty arvo.

Käyntiaika on aika, jolloin kone tekee sille annettua tehtävää. (PSK 6201 2016, 7)

Esimerkiksi kone tuottaa 10 osaa minuutissa, joten sen nimellistuotantokyky on 10 kpl/min, käyntiaika on 360 minuuttia ja todellinen tuotanto on 2880 kappaletta. Näin saadaan

$$\frac{2880}{10kpl/min\ x\ 360min} = 0{,}8 \rightarrow 80\%$$

(OEE Industry Standardin www-sivut 2016)

Ilmoitettu nimellistuotantokyky kannattaa aina ottaa vastana hieman varauksella. Koneen valmistaja voi ilmoittaessaan nimellistuotantokykyä jättää siihen pienen marginaalin. Se voi olla esimerkiksi luotettavuuden tai vaihtelevuuden takia. Aina kuitenkin kannattaa asettaa nimellistuotantokyky mahdollisimman suureksi, vaikka kyseinen nopeus ei olisikaan sillä hetkellä saavutettavissa. Tämä estää toiminta-asteen menemisen yli 100%:n ja kertoo myös, että vielä on potentiaalia parantaa. Koneen käsitellessä useampaa eri tuotetta, jokaiselle lasketaan oma nimellistuotantokyky, jossa huomioidaan kone-tuotesuhde. Tämä tekee laskemisesta hieman haastavampaa ja laskiessa yhteistä toiminta-astetta tarvitsee jokaisen tuotteen toiminta-asteesta ottaa keskiarvo painotettuna aikaan. Jotta tämän pystyisi laskemaan, tarvitsee tietää, kauanko mitäkin tuotetta tehtiin ja mikä oli lopputulos. (Koch 2016)

Esimerkkinä koneen käyntiaika on 360 minuuttia. Koneella käsitellään 120 minuutin ajan tuotetta A, jonka nimellistuotantokyky on 12 kpl/min. Tuotetta B käsitellään 240 minuuttia, nimellistuotantokyky on 4 kpl/min. Tuotteen A tuotanto oli 1100 kpl ja tuotteen B 700 kpl.

$$\left(\frac{120}{360} x \frac{1100}{(12x120)}\right) + \left(\frac{240}{360} x \frac{700}{(4x240)}\right) = 0{,}74 \rightarrow 74\%$$

Ideaalituote koneelle tietenkin olisi semmoinen, joka sallisi koneen käytön maksiminopeudella. On kuitenkin hyvin mahdollista, että kone käsittelee useampaa erilaista tuotetta, mistä johtuen sitä ei voida laittaa maksiminopeudella toimimaan. Esimerkiksi leikkuriin voi tulla eri paksuisia tai eri määrä paperia leikattavaksi kerralla. Tästä johtuen on luotu OEE Top arvo. OEE Top arvo lasketaan täysin samoin kuin OEE arvo, mutta toiminta-asteessa käytetään aina nimellistuotantokyvyssä koneen suurinta mahdollista nopeutta, eli jätetään tuote huomioimatta. Esimerkkinä koneen nopeus ohuelle tuotteelle on 60 kpl/min ja paksummalle 30 kpl/min. Paksumman tuotteen kohdalla on siis huomattavasti pienempi suorituskyky. Tätä tietoa voi hyödyntää, vaikka miettimällä voidaanko paksumpi tuote siirtää käsiteltäväksi tehokkaampaan koneeseen. (Koch 2016)

Suorituskyvyn hävikit kategorisoidaan yleensä kahteen eri pääluokkaan:

- Lyhyt pysähdys. Kattaa ajat, jolloin kone tai laitteisto pysähtyy lyhyeksi ajaksi (useimmiten pari minuuttia) ja koneenkäyttäjä saa ratkaistua ongelman. Yleisimpiä syitä voivat olla muun muassa materiaalin virheellinen syöttö, materiaalijumit, väärät asetukset, tukkeutuneet sensorit ja säännölliset nopeat puhdistukset. Tähän kategoriaan kuuluvat yleensä pysähdykset jotka ovat kestoltaan alle 5 minuuttia eivätkä vaadi kunnossapidon henkilökuntaa.
- Hidastettu nopeus. Kattaa ajan, jolloin kone tai laitteisto toimii hitaammalla nopeudella kuin nimellistuotantokyky. Yleisimpiä syitä voivat olla muun muassa likainen tai kulunut välineistö, huono voitelu, ala-arvoinen materiaali, huonot ympäristöolosuhteet, koneenkäyttäjän kokemattomuus, käynnistys ja sammutus. (Vorne Industries 2016.)

2.3 Laatukerroin (L)

Laatukerrointa voidaan hahmottaa miettimällä, kuinka moni osa täyttää laatuvaatimukset.

$$\frac{Tuotanto - Hylätty\ tuotanto}{Tuotanto}$$

Missä

Tuotanto on tuotannon kokonaismäärä, sekä hyvät että huonot osat.

Hylätty tuotanto on kaikki se tuotanto, joka ei täytä laatuvaatimuksia. Voidaan ajatella ”ensimmäisellä kerralla oikein” –mentaliteetilla, vaikka tuotetta pystyisi käyttämään jalostamalla sen uudestaan, se on hylätty. (PSK 6201 2016, 7)

Esimerkiksi tuotanto oli 2880, joista 144 ei täyttänyt laatuvaatimuksia. Täten laatukerroin on

$$\frac{2880 - 144}{2880} = 0{,}95 \rightarrow 95\%$$

(OEE Industry Standardin www-sivut 2016)

Laadun hävikit kategorisoidaan yleensä kahteen eri pääkategoriaan:

- Hylätty tuotanto, Kattaa normaaliajossa tulleet hylätyt tuotteet. Sisältää myös tuotteet, jotka voidaan työstää uudelleen. Yleisimpiä syitä voivat olla muun muassa väärät asetukset, virheellinen käsittely joko koneelta tai koneenkäyttäjältä.
- Vähennetty tuotantonopeus. Kattaa hylätyt tuotteet, kun prosessi ei ole normaalinopeudella. Sisältää myös tuotteet, jotka voidaan käsitellä uudelleen. Yleisimpiä syitä voivat olla muun muassa koneen epäonnistunut siirtyminen toiseen tilaan, uuden tuotteen tullessa väärät asetukset, laitteisto joka tarvitsee aikaa toimiakseen normaalisti eli ’warmupin’ ja laitteet jotka luovat luonnostaan jätettä käynnistyksen jälkeen. (Vorne Industries 2016.)

2.4 Kokonaistehokkuus

Edellä laskettujen kolmen arvon perusteella voidaan laskea koneen kokonaistehokkuus kertomalla saadut arvot keskenään.

Käyttäen aikaisempia kolmea arvoa saamme kokonaistehokkuudeksi
$75\% * 80\% * 95\% = 57\%$ (OEE Industry Standardin www-sivut 2016)

Kokonaistehokkuuden voi myös laskea hieman yksinkertaisemmallakin kaavalla:

$$\frac{Hyväksytty\ tuotanto\ x\ Nimellistuotantokyky}{Käyttöaika}$$

Missä
Hyväksytty tuotanto on laatuvaatimukset täyttävä tuotannonmäärä.
Nimellistuotantokyky on valmistajan ilmoittama nimellistuotantokyky tai erikseen selvitetty arvo.
Käyttöaika on suunniteltu tuotantoaika, eli käyntiaika + seisokkiaika.
Laskemalla OEE tällä tavoin saadaan sama tulos kuin aikaisemmin, mutta tämä tapa ei juurikaan anna mahdollisuutta yksilöidä, missä puutteet ovat. (Vorne Industries 2016.)

Käyttäen aikaisempia arvoja saadaan siis

$$(2736kpl * 6s)/(480min * 60s) = 0,57 -> 57\%$$

2.5 Käyttöaste

Kokonaistehokkuuden rinnalla voidaan myös mitata käyttöastetta, jotta saadaan kokonaisajastakin hieman käsitystä.

Standardi PSK 6201 antaa käyttöasteen laskukaavaksi:

$$\frac{Käyttöaika}{Kalenteriaika}$$

Josta saadaan prosenttimäärä käyttöajan suhteesta kalenteriaikaan.

OEE:n kanssa kuitenkin yleisin tapa laskea ajankäytön tehokkuus lienee TEEP (engl. Total Effective Equipment Performance). TEEP on tehokkuuden muuttuja, joka ottaa

kantaa tuotantolinjaston todelliseen kapasiteettiin. Se ottaa huomioon sekä laitteiston, että ajan hävikit. TEEP siis ottaa huomioon myös ajan, jolloin laitteistoa ei ole suunniteltu käytettävän, eli muun muassa, jos tehdas on suljettuna. TEEP saadaan kertomalla OEE käyttöasteella (engl. utilization). (Vorne Indistries 2016.)

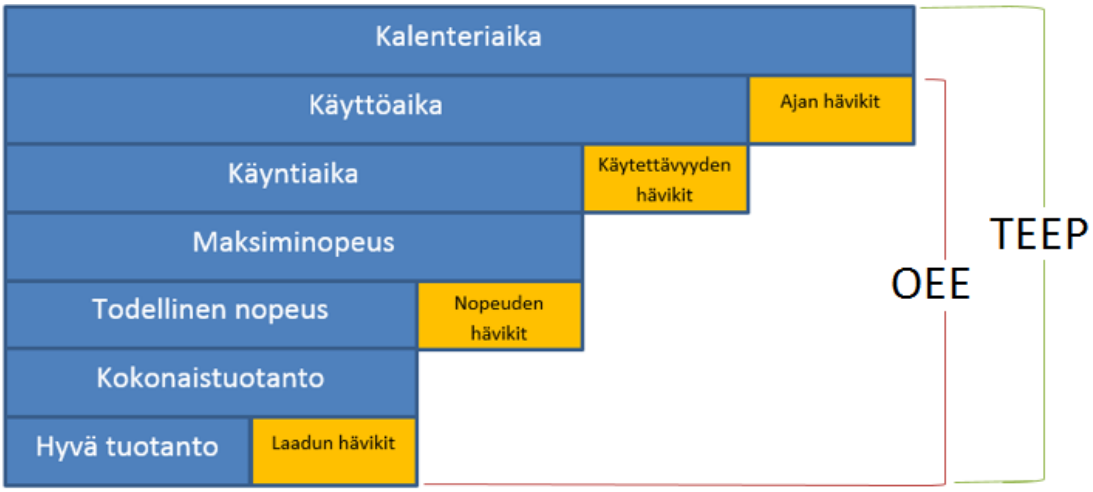
Esimerkiksi OEE on 57%, viikon aikana käyttöaika on 120 tuntia ja koska viikossa on 168 tuntia, saadaan käyttöasteeksi

$$\frac{120h}{168h} = 0{,}71 \rightarrow 71\%$$

Näin ollen TEEP on

$$0{,}57\ x\ 0{,}71 = 0{,}40 \rightarrow 40\%$$

Kuvasta 1 siis voidaan tulkita, että OEE lasketaan suunnitellusta tuotantoajasta (käyttöaika), kun taas TEEP kalenteriajasta. OEE ja TEEP antavat siis hyvän mahdollisuuden tutkia ja kehittää laitteistoa ja prosesseja. Se miten hävikit kategorisoidaan ei ole tärkeintä, kunhan ne tulevat selville.



Kuva 1. OEE:n ja TEEP:n suhde.

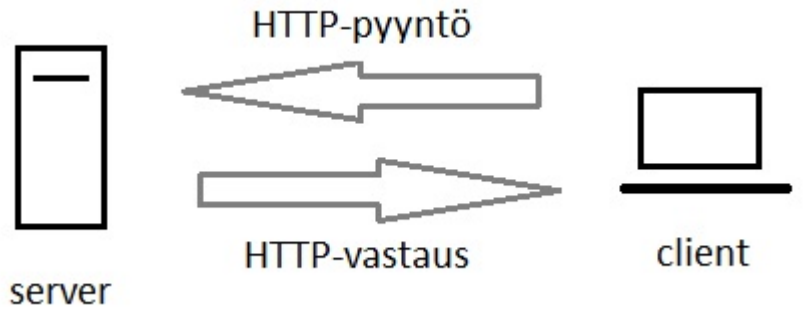
3 WEB-SERVERI

WWW-palvelimella tai toisin sanoen web-serverillä voidaan tarkoittaa tietokonetta/laitteistoa, ohjelmistoa tai molempia yhdessä. (Mozilla Developer Network, 2016)

Laitteistona ajateltuna web-serveri on tietokone/laitteisto, joka säilyttää nettisivun tiedostot (HTML-dokumentit, kuvat jne.) ja toimittaa ne käyttäjälle. Web-serveri on yhteydessä Internettiin ja siihen voi päästä käsiksi verkko-osoitteen kautta. (Mozilla Developer Network, 2016)

Ohjelmistona ajateltuna web-serveri pitää sisällään monia komponentteja, jotka ohjaavat web-käyttäjien pääsynhallintaa serverin tiedostoihin, minimissään HTTP-ominaisuuden. HTTP-palvelin on osa ohjelmistoa, joka ymmärtää verkko-osoitteita ja HTTP-kommunikointia. (Mozilla Developer Network, 2016)

Yksinkertaisuudessaan, kun selain (client) tarvitsee web-serverillä säilytettävän tiedoston, pyytää se sitä HTTP:n avulla. Pyynnön saavuttua oikealle web-serverille (laitteistolle/tietokoneelle), HTTP-palvelin (ohjelmisto) lähettää pyydetyn dokumentin takaisin HTTP:n kautta, kuten kuvassa 1. (Mozilla Developer Network, 2016)



Kuva 1. Asiakas-palvelinarkkitehtuuri.

3.1 Tiedonsiirto

HTTP on protokolla, joka suunniteltiin selaimen ja web-serverin väliseen kommunikointiin ja tiedonsiirtoon. Se perustuu asiakas-palvelin -malliin, jossa asiakas (client) avaa yhteyden, suorittaa pyynnön ja odottaa kunnes saa vastauksen. (Mozilla Developer Network, 2016)

Kun asiakas haluaa kommunikoida serverin kanssa, se suorittaa seuraavat askeleet:

1. Avataan TCP-yhteys. TCP-yhteyttä käytetään lähetettäessä yksi tai useampi pyyntö ja vastauksen saamisessa. Asiakas voi avata uuden yhteyden, uudelleen käyttää olemassa olevaa yhteyttä tai avata useamman yhteyden kerralla.
2. Lähetetään pyyntö. Pyyntö lähetetään HTTP-viestinä, jossa selviää
    - pyynnön metodi,
    - haettavan tiedon polku,
    - HTTP-protokollan versio,
    - ylätunniste, jossa mahdollisia lisätietoja haettavasta tiedostosta ja asiakkaasta itsestään.
3. Luetaan serveriltä saatu vastaus. Saatu vastaus pitää sisällään
    - HTTP:n version,
    - status-koodi, joka kertoo, oliko pyyntö onnistunut ja jos ei, niin miksi,
    - status-viesti, lyhyt kuvaus status-koodista,
    - ylätunniste, jossa mahdollisia lisätietoja tiedostosta tai serveristä,
    - mahdollisesti palautettava tiedosto.
4. Suljetaan tai käytetään yhteyttä seuraaviin pyyntöihin.
   (Mozilla Developer Network, 2016)

HTTP:n avulla voidaan myös siirtää muun muassa välimuistiin tietoa, evästeitä ja kontrolloida kirjautumisia vaikkapa keskustelupalstalle.
(Mozilla Developer Network, 2016)

Reaaliaikaisen datan siirtämiseen HTTP ei ole optimaalinen johtuen sen siirtämästä ylätunniste- ja evästetiedoista, jotka voivat useamman pyynnön jälkeen kasvaa isoiksi, sillä jokaisen tiedon siirtämiseen tarvitaan uusi pyyntö. Ratkaisun tähän tarjoaa

WebSocket-yhteys. WebSocket tarjoaa pysyvän yhteyden asiakkaan (client) ja serverin välillä, jossa molemmat voivat lähettää dataa, milloin vain. (West, 2016)

WebSocket yhteys muodostetaan HTTP-pyynnön avulla, joka sisältää tiedon, että yhteys haluttaisiin luoda WebSocket:n avulla. Jos serveri tukee WebSocket-protokollaa, se hyväksyy ehdotuksen ja korvaa HTTP-yhteyden WebSocket-yhteydellä, joka hyödyntää TCP-yhteyttä kuten HTTP:kin.
(West, 2016)

3.2 Työssä käytettyjä menetelmiä

HTML (Hyper Text Markup Language) on merkintäkieli, jonka avulla on mahdollista esittää tietoa internetissä. HTML koostuu eri tunnisteista (tageista), joista jokainen kuvaa sivun eri osaa. Nykyinen HTML versio on HTML5.
Yksinkertainen HTML-sivu voisi näyttää seuraavanlaiselta:

```
<!DOCTYPE html>
<html>
<head>
<title>Sivu 1</title>
</head>
<body>
<h1>Otsikko</h1>
<p>Kappale</p>
</body>
</html>
```

Liitteessä 1 selitetty hieman tarkemmin HTML-sivun tunnisteita.
(w3schools:n www-sivut, 2016)

CSS (Cascading Style Sheets) määrittelee miltä nettisivu näyttää selaimessa. Muotoilu tehdään yleensä valitsimeen (eng. selector) ja aaltosulkujen sisään. Tyylit voi määritellä erilliseen .css päätteiseen tiedostoon tai laittaa HTML-sivulla <style></style> -tunnisteiden sisään.
Seuraavassa esimerkkinä <h1> </h1> -tunnisteiden sisällä olevan tekstin muotoilua:

```
.h1 {
   color: green;
   text-align: center;
}
```

(Mozilla Developer Network, 2016)

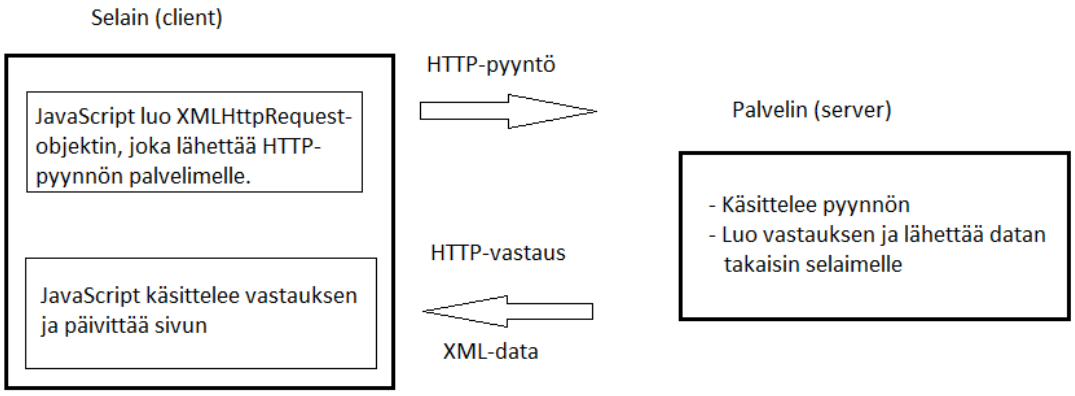
JavaScript ohjelmointikielellä saadaan nettisivulle lisää toiminallisuutta vaikkapa interaktio käyttäjän kanssa. JavaScriptiä käytetään yleensä skriptien tekemiseen, jonka selain suorittaa, jos sillä on lupa. JavaScript määritellään joko HTML-sivulle <script></script> -tunnisteiden väliin tai erilliseen .js päätteiseen tiedostoon joka luetaan sivun latautuessa.

Esimerkkinä ponnahdusikkunan teko tapahtuisi seuraavanlaisesti:

```
<script>
window.alert(Hei!);
</script>
```

(Korpela, 2016)

AJAX (Asynchronous JavaScript and XML) on lyhykäisyydessään kommunikointia serverin kanssa XMLHttpRequest-objektin avulla, kuten kuvassa 2. Sen avulla voi lähettää ja vastaanottaa dataa eri muodoissa, mm. JSON, XML ja HTML. Tärkein AJAX:n ominaisuus on asynkroninen päivitys, eli se mahdollistaa sivun tietojen päivityksen lataamatta koko sivua uudelleen.



Kuva 2. AJAX-päivitys.

AJAX esimerkki, jossa pyydetään HTTP:n avulla test.html nimistä tiedostoa serveriltä:

```
<script type="text/javascript">
```

```
(function() {
  var httpRequest;
  document.getElementById("ajaxButton").onclick = function() { makeRequest('test.html'); };

  function makeRequest(url) {
    httpRequest = new XMLHttpRequest();

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    httpRequest.onreadystatechange = alertContents;
    httpRequest.open('GET', url);
    httpRequest.send();
  }

  function alertContents() {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
})();
</script>
```

Eli tässä esimerkissä klikataan linkkiä, jolloin MakeRequest()-funktio pyytää test.html-tiedostoa ja alertContents()-funktio tarkistaa vastauksen tilan ja mahdollisesti näyttää test.html-tiedoston.

Liitteessä 6 on käyty AJAX:ia tarkemmin läpi.

(Mozilla Developer Network, 2016)

XML (Extensible Markup Language) on eräänlainen kieli tiedostoille, joissa on jäsenneltyä tietoa. XML-tiedoston etu verrattuna HTML-tiedostoihin on käyttäjän mahdollisuus itse luoda tunnisteet, eli tagit.
XML-dokumentti voi näyttää vaikkapa seuraavanlaiselta:

<data>
<mittaukset>Mittaus 1</mittaukset>
<minimi>200<minimi>
</data>

(O'Reilly, Media Inc, 2016)

jQuery on pieni, nopea ja monipuolinen JavaScript-kirjasto, jonka avulla on mahdollista muokata HTML-sivun toimintaa ja ominaisuuksia. jQuery perustuu avoimeen lähdekoodiin. (The jQuery Foundation, 2016)

Flot on Javascript-kirjasto jQuerylle, jolla voi piirtää erilaisia kaavioita. Flot pyrkii olemaan yksinkertainen, näyttävä ja vuorovaikutteinen. Flot on MIT-lisenssin alla. (Flotcharts:n www-sivut, 2016)

JustGage on JavaScript-kirjasto, jolla saadaan tehtyä mittareita. JustGage tarvitsee toimiakseen Rapahel-kirjaston, jonka saa ladattua JustGagen sivuilta. Toimii MIT-lisenssin alla. (JustGagen ww-sivut, 2016)

AWP (Automation Web Programming) -komennot ovat luotu erityisesti tiedonvaihtoon CPU:n ja HTML-tiedoston välille. AWP-komentoja hyödynnetään Siemensin logiikoiden web-servereissä käyttäjän itse tehdyillä sivuilla. AWP-komentojen avulla saadaan muun muassa luettua ja asetettua logiikassa olevien muuttujien arvoja HTML-sivun välityksellä. Selkeyden vuoksi AWP-komennot yleensä listataan HTML-sivun alkuun ja voivat näyttää seuraavanlaiselta:

<!-- AWP_In_Variable Name='"WebData".Muuttuja2' Name='"WebData".Muuttuja3' -->

Tässä esitelty WebData DB:ssä sijaitsevat muuttujat Muuttuja2 ja Muuttuja3.
Sivut, joilla käytetään AWP-komentoja pitää olla UTF-8 koodausta. (S7-1500 Web server 2016, 62)

Enumeja (enumeration types) käytetään Siemensin logiikoiden web-serverin itse luotujen sivujen ohjelmoinnissa muuttamaan logiikassa olevien muuttujien arvoja teksiksi tai toisinpäin. Enumeissa hyödynnetään AWP-komentoja

<!-- AWP_Enum_Def Name='Asento' Values='0:"Ylös", 1:"Alas", 2:"Vasen", 3:"Oikea"' -->

(S7-1500 Web server 2016, 67)

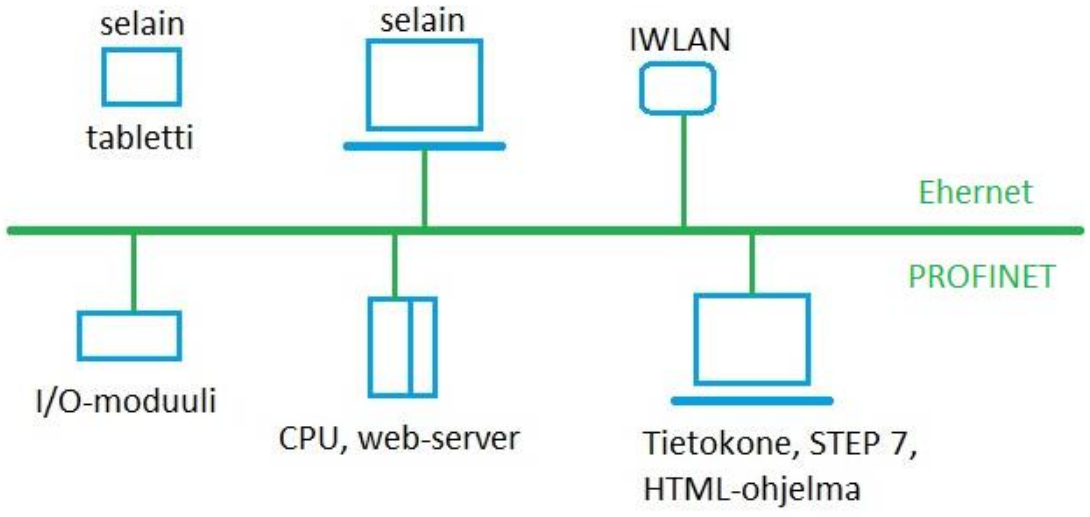
4 LOGIIKKA

4.1 Logiikan web-server

Nykyaikainen automaatioteknologia hyödyntää nousevissa määrin internet-teknologiaa, jonka avulla on mahdollista saada esimerkiksi lähiverkon kautta suora yhteys järjestelmään (Kuva 3). Moni logiikkaohjainten valmistaja tarjoaakin mahdollisen web-server -ominaisuuden, joko suoraan integroituna logiikkaan tai erillisenä moduulina.

Siemensin S7-1500, Schneiderin Modicon M251 ja Panasocin FP7 ovat logiikoita, joissa on integroituna web-serveri. Pohjimmiltaan ne ovat hyvin samanlaisia. Kaikissa on valmiit sivut, joista pystyy monitoroimaan muuttujia ja näkemään diagnostiikkatietoja. Suurin ero lienee omien sivujen luominen. Schneiderilla ja Panasonicilla on oma ohjelmistoympäristö, jolla pystyy tekemään sivuja aivan kuin tekisi niitä HMI-päätteelle. Siemensillä tämmöistä ei ole ja sivut ovatkin tehtävä jollain muulla ohjelmistolla, esim. NetBeans hyödyntäen muun muassa HTML:llää, CSS:ssää ja JavaScriptiä.

(S7-1500 Web server 2016) (Modicon M251 Logic Controller 2016, 92-105) (FP7 PLC 2016)



Kuva 3. Web-serverin hyödyntämismahdollisuuksia.

Sallimalla selaimella pääsyn logiikan tietoihin, muuttujien monitorointi ja tietyssä määrin muuttujien arvojen asettaminen on mahdollista ilman lisälaitteita ja -ohjelmia verkon vaikutusalueella. Logiikan muuttujien ja sen asettaminen on siis mahdollista hieman pidemmältäkin välimatkalta vaikkapa tabletilla. Web-server ominaisuutta hyödynnettäessä tosin kannattaa muistaa turvallisuus (esim. palomuuri, käyttäjätunnukset jne). (S7-1500 Web server 2016, 10)

Kuten aiemmin todettu, reaaliaikaisen datan siirtämiseen nopeudeltaan hyvä tapa olisi WebSocket. Se ei kuitenkaan tunnu olevan hirveän yleinen ominaisuus tällä hetkellä logiikoiden web-servereissä. Niinpä muuttujien päivitys AJAX:llakin voi isommissa projekteissa tai pienempi tehoisella CPU:lla ja useampaa muuttujaa kerralla monitoroidessa kestää yllättävän kauan.

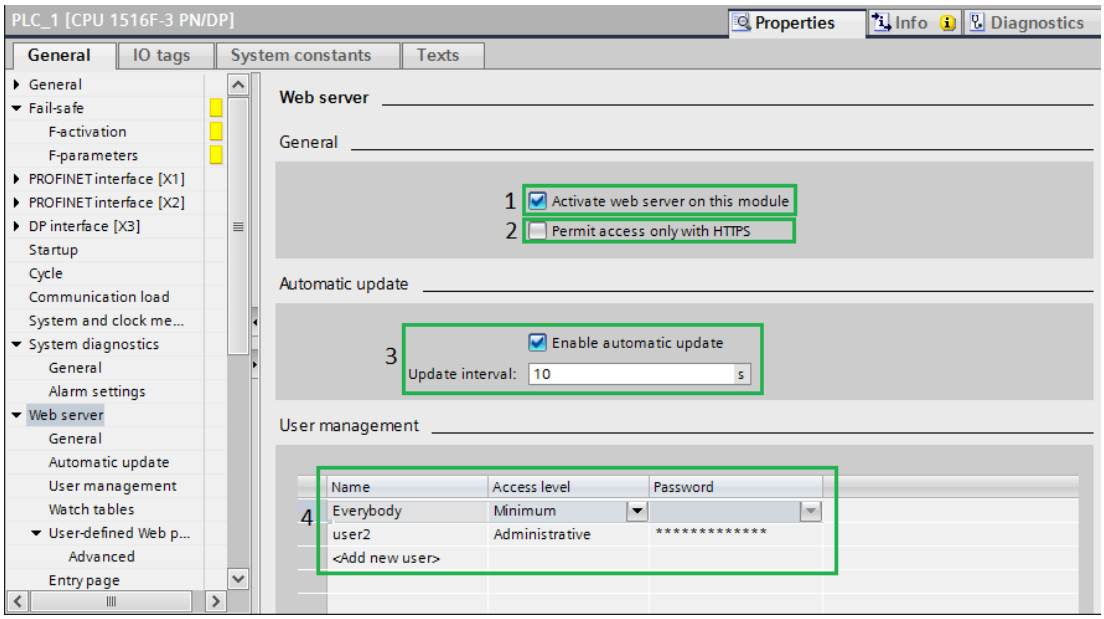
Siemensin Tag status -sivulla 10:nen muuttujan päivitys kesti keskimäärin 350ms, 20:nen 550ms ja 30:nen 750ms. Itse luoduilla sivuilla F5-päivityksellä 10 muuttujan päivittämiseen meni keskimäärin 100ms ja 20:nen 200ms. AJAX:lla tehtynä 10 muuttujan päivitys kesti keskimäärin 110ms ja 20:nen 250ms. Näihin aikoihin vaikuttaa tietenkin sivujen sisältö muilta osin. Jos AJAX:ssa hyödynnettävässä XML-dokumentissa on paljon ylimääräistä (esim. turhia tunnisteita), aikaa menee enemmän. Muuttujien sulkeminen pois kommentteina ei auta tässä tilanteessa. Myöskään muuttujan koko ei juurikaan vaikuta päivitysaikaa, eli täysimittaisen string-tyyppisen muuttujan siirto vie yhtä kauan kuin yhden bit-tyyppisen muuttujan siirto. F5-päivityksellä

koko sivu latautuu aina uudelleen, joten kuvat yms. vaikuttavat. Myös CPU:n muu ohjelma vaikuttaa sykliaikoihin. Testi tehtiin S7-1516F-3PN/DP -logiikalla.

Siemensin sivuilla painotetaan kuitenkin, että web-serverit eivät pysty korvaamaan perinteistä käyttöliittymää järjestelmän ohjauksessa ja valvonnassa, sillä web-sovellusten toiminta ei ole riittävän reaaliaikaista. HMI-päätteiden korvaaminen ei siis ole web-serverin tavoitteena.
(Creating user-defined we-pages on S7-1200/S7-1500 2016, 7)

4.2 Web-serverin aktivointi

Web-serveri saadaan käyttöön avaamalla CPU:n Properties-kohta ja kuvassa 4 kohdasta 1 voidaan valita web-serveri aktivoitavaksi.



Kuva 4. Web-serverin asetuksia 1.

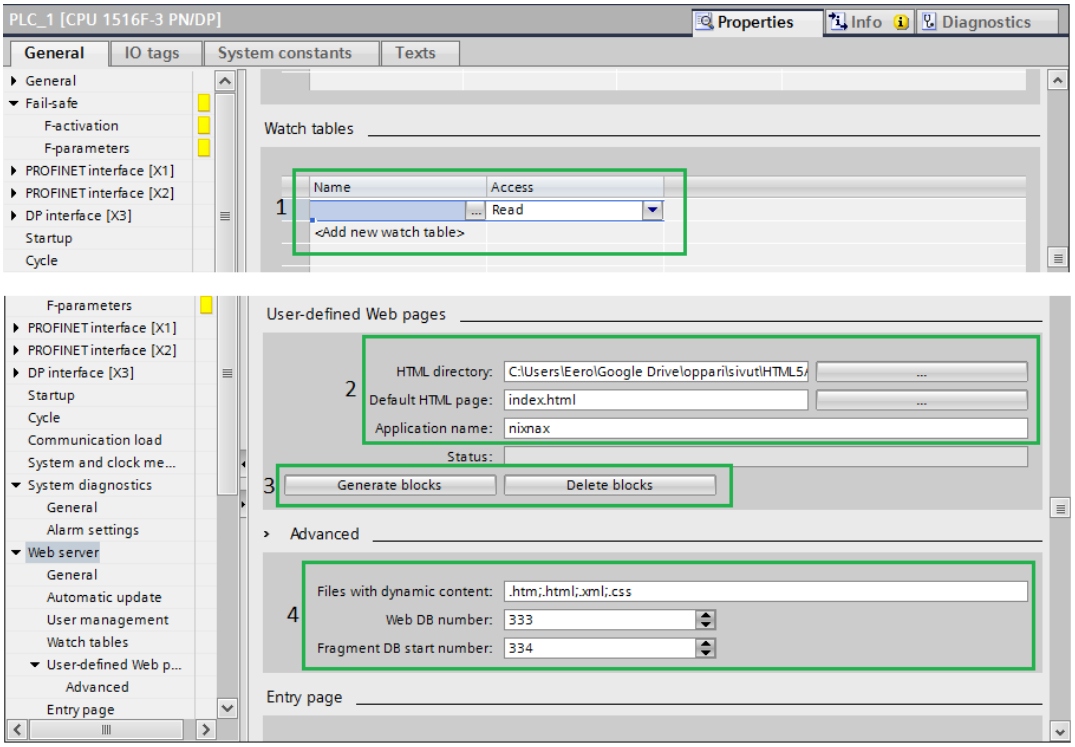
Kohdassa 2 voi määrittää haluaako yhteyden muodostuvan suojatulla protokollalla.

Kohdassa 3 voi valita haluaako Siemensin valmiina oleville sivuilla automaattisen päivityksen ja millä aikavälillä.

Kohdassa 4 luodaan käyttäjät sivuille, joille voi antaa eri käyttöoikeuksia. Annettavia oikeuksia ovat muun muassa CPU:n tilan vaihtaminen (start/stop), itse luotujen sivujen

käyttäminen ja muuttujien monitorointi. Käyttäjille voidaan ja kannattaa erikseen määritellä mitä he pystyvät sivuilla tekemään ja salasanat. Salasanan pituus on vähintään 8 merkkiä. Käyttäjien maksimilukumäärä on 20. (S7-1500 Web server 2016, 16)

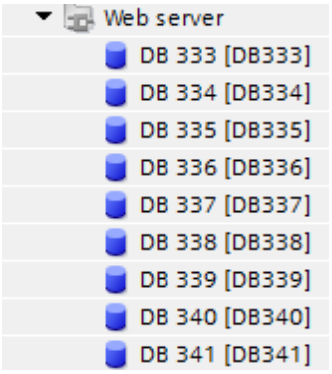
Kuvan 5 kohdassa 1 voi laittaa valmiina olevalle Watch tables- sivulle muuttujalistoja näytettäväksi. Kerrallaan voi monitoroida vain yhtä listaa, mutta listojen kokonaismäärä riippuu käytössä olevasta muistikortista. (S7-1500 Web server 2016, 57)



Kuva 5. Web-serverin asetuksia 2.

Käyttäjän halutessa luoda omat sivut, kohdassa 2 määritellään missä kansiossa sivut ovat, eli annetaan polku sivujen kansioon tietokoneessa. Kerrotaan myös, mikä sivu aukeaa, kun itse luoduille sivuille mennään, eli annetaan niille etusivu. Lisäksi vapaaehtoisena sivulle/projektille voi antaa nimen.

Kohdassa 3 luodaan tarvittavat tiedostot, jotta saadaan ladattua luodut HTML-sivut logiikkaan. Riippuen sivujen koosta, tiedostoja voi tulla useampikin, kuten kuvassa 6.



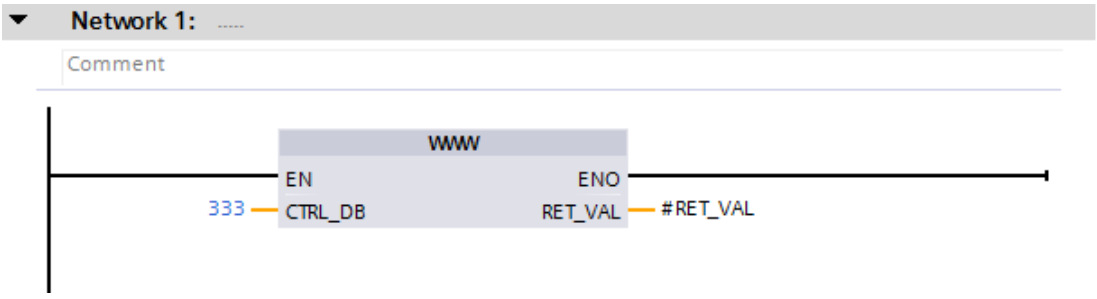
Kuva 6. Generoinnista tulleet tiedostot.

Kuvan 5 kohdassa 4 määritellään, minkä tyyppistä dynaamista sisältöä sivuilla käytetään. Logiikka tunnistaa automaattisesti .js, .htm ja .html. Tarvittaessa voi lisätä muita. JavaScriptiä ei kannata laittaa, sillä sivujen generointi ei välttämättä tykkää kaikista JavaScript-tiedostoista.

Sivut siis pitää generoida ennekuin ne ladataan logiikkaan ja generointi luo niistä tiedostot, joiden aloitusnumerot voi kohdassa 4 määritellä. Generointi oli kohdassa 3.

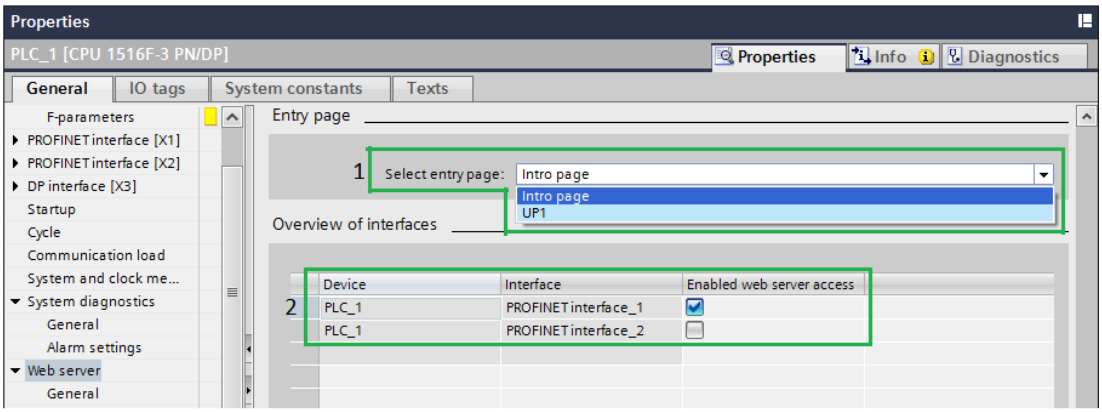
Siemensin manuaalin mukaan 1500-sarjan logiikkaan saa lisää turvallisuutta tiedonsiirtoon asentamalla erillisen kommunikointiprosessorin (CP 1543-1), jonka asetuksista saa laitettua mm. HTTPS-yhteyden päälle ja aktivoitua palomuurin. (Creating user-defined we-pages on S7-1200/S7-1500, 28)

Käyttäjän omat sivut tarvitsevat toimiakseen WWW-kutsun PLC-ohjelmassa, kuten kuvassa 7. Jotta päivittäminen toimisi, tarvitsee WWW-komentoa kutsua syklisesti. CTRL_DB:hen tuleva luku on sama kuin kuvassa 5 kohdassa 4 oleva Web DB number. RET_VAL muuttuja on INT-tyyppinen ja siihen funktio palauttaa funktion tilatiedon.



Kuva 7. WWW-funktion kutsu.

Kuvassa 8 kohdassa 1 voi määrittää mille sivulle logiikka vie, kun selaimella kirjaudutaan CPU:hun. Intro page vie siemensin tekemälle etusivulle, josta pääsee katsomaan valmiina olevia sivuja ja UP1 vie käyttäjän omille sivuille.



Kuva 8. Web-serverin asetuksia 3

Kohdassa 2 voi määrittää mistä Profinet-liitynnöistä on pääsy web-serveriin.

Lopuksi kaikki pitää tietenkin vielä ladata logiikkaan, jotta muutokset tulisivat voimaan.

4.3 Valmiit ominaisuudet

Web-serverin sivuja pääsee tutkimaan laittamalla CPU:lle määritellyn IP-osoitteen selaimen osoiteriville, esim. http://192.168.0.1. Tietokoneen tai muun laitteen, jolla halutaan selata web-serverin sivuja pitää olla samassa lähiverkossa kuin CPU:n. Sivut on testattu toimiviksi seuraavilla selaimilla:

- Internet Explorer (Versiot 8-11)
- Mozilla Firefox (Versiot 22-32)
- Google Chrome (Versiot 33-38)
- Mobile Safari ja Chrome iOS
- Android selain ja Android Chrome (S7-1500 Web server 2016, 10)

Aukeava sivu on Siemensin tekemä Intro-sivu. Painamalla ENTER-kohtaa pääsee eteenpäin. Seuraava sivu on niin sanottu Etusivu (Kuva 9), josta näkee CPU:sta yleistietoa ja tarjoaa mahdollisuuden kirjautua sisään. Sisäänkirjautumisen jälkeen päästään näkemään ja käyttämään sivuilla valmiina olevia toimintoja. Esimerkiksi CPU:n saa ohjattua RUN- tai STOP-tilaan. Sivun vasemmassa laidassa on valmiina valikko, josta näkee lisätietoa CPU:n toiminnasta ja siihen yhteydessä olevista laitteista. Valmiilta sivuilta saatavissa oleva tieto on hyvin pitkälti samaa, mikä on saatavilla TIA Portal -ohjelmasta.

Kuva 9. Valmiina oleva etusivu.

- Diagnostics-kohdassa on lisätietoa CPU:sta ja sen muistin käytöstä. TIA Portalissa Online & Diagnostics -kohdan Diagnostics.

- Diagnostic Buffer -kohdassa on viestejä logiikan toiminnasta, muun muassa CPU:n käynnistys- ja virhetietietoja. Myös Online & Diagnostics -kohdassa.

- Module information näyttää kytketyt laitteistot ja niiden tilan. Niitä klikkaamalla saa lisätieto laitteistosta. Verrattavissa TIA Portalin Device Overview -kohtaan.

- Alarms näyttää vikailmoitukset. Samat kuin TIA Portal -ohjelmassa.

- Communication näyttää informaatiota logiikan Profinet- ja Ethernet-liitynnöistä. Verrattavissa TIA Portalin Online & Diagnostics -kohdan alla oleviin tietoihin.

- Topology näyttää Profinet verkon topologian ja Profinet-laitteiden tilan. Klikkaamalla laitetta pääsee Module information -sivulle. TIA Portalissa on myös Topology-kohta.

- Tag status -sivulla pystyy monitoroimaan ja muokkaamaan muuttujien arvoja. Absoluuttista osoitetta ei voi monitoroida.

- Watch tables -sivulla pystyy monitoroimaan ennalta määriteltyjä watch tableja. Luvussa 4.2 kerrotaan miten niitä voi lisätä.

- Customer pages -valikon kautta pääsee käyttäjän itse luotuihin sivuihin. (WWW-funktio pitää olla. Funktio esitelty ylempänä, itse sivuista alempana lisää.)

- File browser näyttää muistikortilla olevat tiedostot.

- DataLogs näyttää luodut DataLogit. DataLogeja voi luoda TIA Portal:ssa DataLogCreate-funktiolla. DataLogiin voi tallentaa esimerkiksi muuttujien arvoja.

(S7-1500 Web server 2016, 22-59, 88-89)

Siemens suositteleekin käyttämään heidän oma tekemiään, valmiiksi integroituja diagnostiikka mahdollisuuksia, kuten web-serverin tarjoamat Diagnostics ja Topology -sivut, jos tiedon haussa ei tarvita erillisiä diagnostiikkafunktioita (esim. DeviceStates- ja GET_DIAG-funktioita). Integroidut diagnostiikkatoiminnot myös pysyvät päällä logiikan mentyä stop-tilaan, toisin kuin käyttäjän omat. (Diagnostics in the User Program with S7-1500, 4) Asiasta lisää luvussa 5.1.

Sivuista on olemassa myös ns. ”basic” -versio. Basic-versioon pääsee laittamalla CPU:n IP:n perään /basic. Esim. 192.168.0.1/basic. Nämä ovat tarkoitettu laitteille joissa on pienempi ruutu, esimerkiksi kännykkä. (S7-1500 Web server, 91)

5 ESIMERKKISIVUT

Opinnäytetyöni yhtenä osana oli tutkia S7-1500-logiikan web-serverin ominaisuuksia ja sen tuomia mahdollisuuksia tuotantodatan ja diagnostiikkatietojen näyttämiseen selaimessa. Sain Raumaster Paper Oy:ltä lainaksi S7-1516F-3PN/DP-logiikan sivujen testaamista varten.

5.1 Diagnostiikka

Yhtenä mahdollisuutena diagnostiikkatietojen näyttämiseen on käyttää TIA Portaalin tarjoamia diagnostiikkaan tarkoitettua funktioita ja virhe OB:eita.
Esimerkiksi logiikan käyttöjärjestelmä kutsuu OB 82 lohkoa aina kun moduulin tila vaihtuu ja OB 83:a kutsutaan, kun konfiguroitu ja aktivoimaton moduuli kytketään tai otetaan pois käytöstä.
DeviceSates-funktiolla saa I/O-moduuleiden tilan. DeviceName-funktiolla saa vaikkapa vikaantuneen I/O-moduulin nimen.
Lisänä vielä voi käyttää Program_Alarm- ja Get_AlarmStates-funktioita, joiden avulla pystyy tekemään omia hälytysviestejä ja toimintoja. Nämä viestit tulevat myös näkyviin web-serverin Alarms-sivulle ja TIA Portalin Alarms-kohtaan.
Näistä ja mainitsematta jääneistä funktioista ja OB:sta voidaan siis rakennella käyttötarpeeseen sopiva ohjelma. Saadut tiedot voidaan näyttää selaimessa ja täten hyödyntää vaikkapa tuotantodataluvussa esitellyissä ominaisuuksissa.

Toinen vaihtoehto on käyttää valmiiksi CPU:hun integroituja diagnostiikkamahdollisuuksia, kuten tässä tapauksessa web-serverissä olevia valmiita sivuja. (Luku 4.3)
Koska itse luoduille sivuille tehtiin oma pohja (luku 5.2.1), olisi ollut hyvä saada diagnostiikka sulautettua sinne, esim. iframe:n avulla. Johtuen valmiiden sivujen skripteistä ja muista toiminnollisuuksista, en saanut upotettua niitä toimivina luomalleni pohjalle. Tärkeimmät sivut diagnostiikan kannalta eli Topology ja Module information jäivät upotuksen jälkeen puutteellisiksi toiminnoiltaan. Topology sivua en saanut näyttämään tarvittavia tietoja. Module information toimi melko hyvin, mistä näkee

samoja asioita kuin Topolgysta, joten sivun upottaminen on mahdollista, mutta esimerkiksi kirjautuminen sivuille ohjaa aina etusivulle.

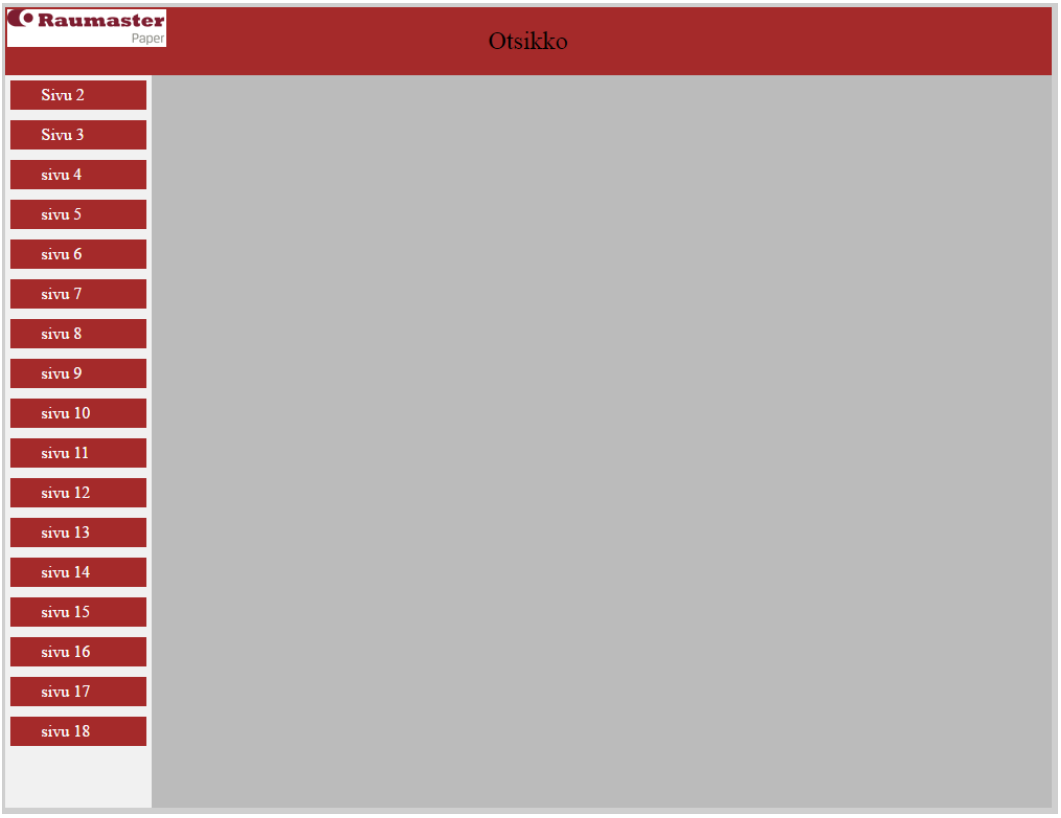
CPU:n ja Profinetin diagnostiikan esittämiseen paras tapa lienee käyttää logiikassa valmiina olevia sivuja (Topology, Module information jne.) Sieltä näkee mielestäni laitteiden tilan ja virheet varsin hyvin, kunhan vain on konfiguroinut laitteet oikein. Avaamalla ne siis vaikkapa uuteen välilehteen on minun suosittelemani vaihtoehto.

5.2 Tuotantotiedot

Tuotantotiedot perustuvat lähinnä logiikasta tulevien arvojen näyttämiseen selaimessa. Muuttujien avulla ja erilaisten HTML-sivuille saatavien ominaisuuksien avulla saadaan muun muassa näytettyä logiikassa talletetun muuttujan arvo, taustavärin vaihtuminen muuttujan suuruuden perusteella ja piirrettyä erilaisia kaavioita. Suurimpana rajoitteen itsetehdyissä HTML-sivuissa lienee niiden tiedostokoko, sillä TIA Portal varoittaa sivujen tiedostokoon olevan liian iso noin 1,1 Mb:n jälkeen.

5.2.1 Template

Ensimmäisenä oli tarkoitus tehdä esimerkkisivuille yhtenäinen tausta kaikille esimerkkisivuille. Tein siis template sivun (Kuva 10, Liite 1), joka on tehty HTML-kielellä CSS-muotoilua hyödyntäen. Sivu on yhden ison kehyksen ympärillä, jotta se saadaan pysymään keskellä sivua eikä osat vaihda paikkaa selaimen kokoa muutettaessa.

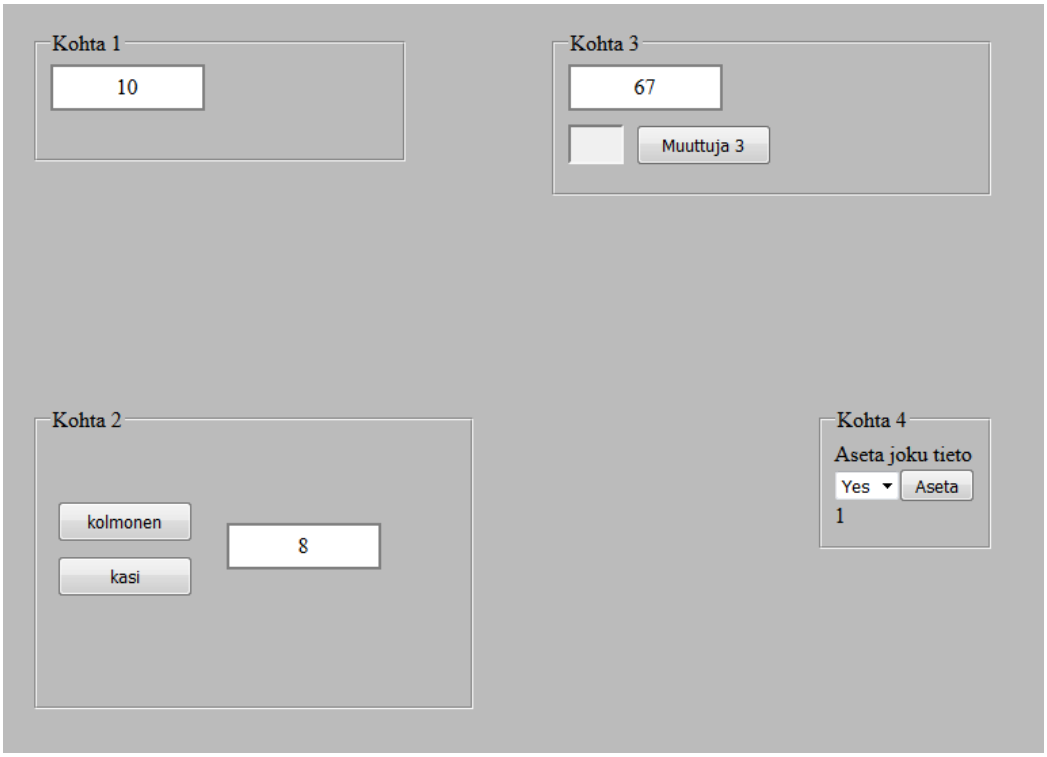


Kuva 10. Template

Kehyksen sisään ylhäälle on luotu ylätunnus divHeader, jonka muotoilu saadaan pagestyle.css-tiedostosta (Liite 2). Template sivun HTML-koodissa siihen on lisätty otsikkoteksti ja logo. Vasemmalla on menu, jolle luotiin aluksi valkoinen pohja ja siihen päälle valikot ja joka muuttaa valikon muotoilua kursorin ollessa sen päällä. Lopputilaan luotiin laatikkomainen tyhjä tila (divBody), joka luo harmaan alueen sivun tyhjään osaan, jonne on tarkoitus laittaa sivun sisältö. Halutessaan sivulle voi lisätä alatunnisteen.

5.2.2 Sivu 2

Sivulla 2 (Kuva 11, Liite 3) esitellään PLC:stä luettu tieto hyödyntäen AWP-komentoja. Miten tietoa saadaan esille HTML-sivulla ja miten sitä pystytään muokkaamaan sivun kautta.



Kuva 11. Sivu 2, AWP-komentoja ja painikkeita.

Kohdassa 1 näkyy Muuttuja1:sen arvo, jolle on muotoiltu pieni laatikko ympärille. Muuttujaa saadaan näkyviin sivulle hyvin yksinkertaisella komennolla:

$:= "WebData".Muuttuja1:$

Muuttuja on PLC:ssä WebData-nimisessä tiedostossa Muuttuja1 nimellä. Muuttujaa ei tarvitse erikseen esitellä, koska sitä vain luetaan sivulla. Sivulle on myös laitettu automaattinen päivitys viiden sekunnin välein, jotta Muuttuja1 päivittyisi ilman käyttäjän suorittamaa päivitystä, eli esim. F5-painallusta.

Kohdassa 2 on Muuttuja2:sen arvo, jota pystyy muuttamaan sille luoduilla painikkeilla. Muuttuja2 on esitelty sivun koodissa ylhäällä seuraavasti:

<!-- AWP_In_Variable Name='"WebData".Muuttuja2' Name='"WebData".Muuttuja3' -->

Painikkeita tehdessä, niille on annettu arvoksi kahdeksan ja kolme, ja Muuttujan2 arvo vaihtelee sen mukaan, kumpaa on painettu. Painikkeet tehtiin hyödyntäen form-elementtiä ja post-metodia, jonka avulla saadaan lähetettyä muuttunut tieto HTTP-viestinä serverille.

Kohdassa 3 näkyy Muuttuja3:n arvo, jota voidaan muuttaa laittamalla haluttu arvo tekstikenttään ja painamalla painiketta, joka lähettää uuden arvon eteenpäin. Muuttuja3 on esitelty Muuttujan2:sen yhteydessä. Tekstikenttä ja painike on luotu käyttäen form-elementtiä ja post-metodia.
Näin saatiin siis tekstikenttä, johon kirjoittaessa ja painettaessa painiketta saadaan muutettua Muuttuja3:n arvoa.

Kohdassa 4 on dropdown-valikko, josta saa valita valmiiksi määritellyistä vaihtoehdoista haluamansa ja saa sen käyttöön painamalla Aseta-painiketta. Sen hetkinen arvo näkyy numerona valikon alla. Muuttuja4 on esitelty HTML-sivun yläosassa ja sille määritellään oma nimi (YesNo) HTML-sivulla AWP-komentoa hyädyntäen.

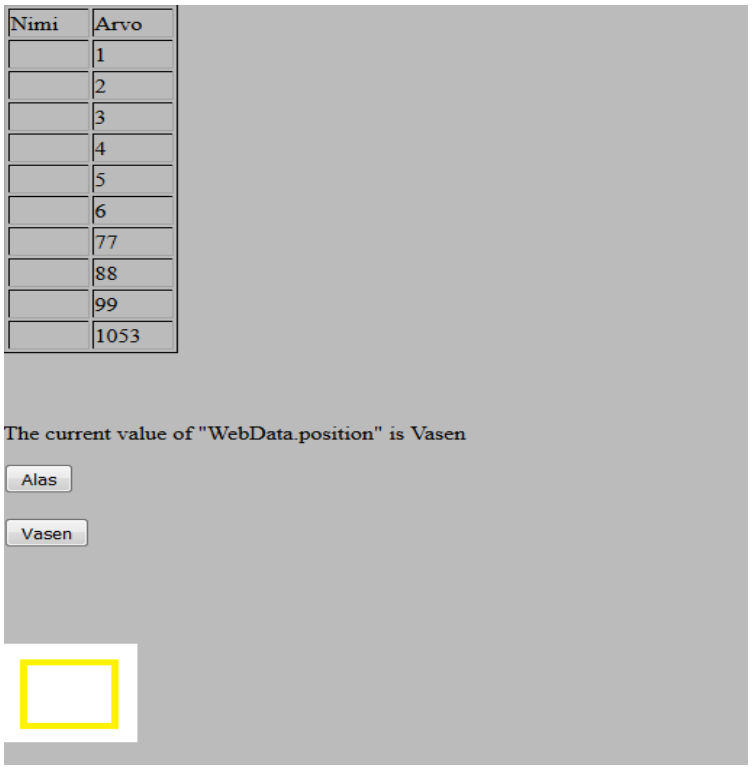
<!-- AWP_In_Variable Name='YesNo' Use='"WebData".Muuttuja4' -->

Dropdown-valikko ja painike on tehty form-elementtiä ja post-metodia hyödyntäen kuten aiemmissakin kohdissa. Arvo saatiin näkyviin viittaamalla Muuttuja4:lle annettuun nimeen:

:=YesNo:

5.2.3 Sivu 3

Sivulla 3 (Kuva 12, Liite 4) on ensimmäisenä taulukko, johon puretaan tieto Arvo-nimisestä tiedostosta, jossa tieto on taulukko-muodossa (array). 1500-logiikka osaa itse purkaa taulukon, kunhan vain sen esittelee oikein. Sivulla ei erikseen ole päivitystä.



Kuva 12. Enumien esittelyä.

Toisena on enumien esittelyä. Enumilla siis voi muuttaa PLC:n muuttujan arvon tekstiksi ja toisinpäin. Koodissa sivun ylälaidassa on aluksi luotu enumi, jolle on määritelty tiettyihin arvoihin tietyt tekstit, jonka jälkeen tulee määrittely, mikä muuttuja on kyseessä.

<!-- AWP_Enum_Def Name='Asento' Values='0:"Ylös",
1:"Alas", 2:"Vasen", 3:"Oikea"' -->
<!-- AWP_Enum_Ref Name='"WebData".position' Enum="Asento" -->

Sivulle on luotu alas- ja vasen-painikkeet, joka ohjaa position-muuttujaa ja napit, jolla tätä on testattu. Riippuen siis viimeksi painetusta painikkeesta, sivulla lukee joko Vasen tai Alas.

Alimpana sivulla on AWP:n avulla tehty kuvan muuttuminen riippuen logiikalta tulevan tiedon arvosta.

<img src='kuvat\kuutio:="WebData".kuutio:.png' alt="kuutio" >

Kuva vaihtuu kuutio-muuttujan arvon perusteella. Kuvat ovat tallennut kuutio0 ja kuutio1 nimillä. Riippuen onko muuttujan arvo 0 vai 1, tulee kuvaksi kuutio0 tai kuutio1.

5.2.4 Sivu 4

Sivulla 4 (Kuva 13, Liite 5) esitellään AJAX:n (Asynchronous JavaScript And XML) avulla tehty PLC-muuttujien automaattinen päivitys. Päivittäminen tapahtuu JavaScriptin ja XMLHttpRequest-komennon avulla ja haettavat tiedot on yksilöity XML-tiedostoon. Sivulle on luotu kaksi erilaista kehikkoa, joiden sisälle on sijoitettu muuttujia. Muuttujille on luotu vielä erikseen pienet laatikot ympärille ja muuttujat on yksilöity niille annetun id:n perusteella.

Kuva 13. AJAX esimerkki

Sivulla <script> </script> -tagien välissä oleva readData_XML -komento kutsuu xml_esim_1.js -nimistä JavaScript-tiedostoa (Liite 6), jonka avulla tiedonpäivitys on tehty. JavaScriptin loppuun on tehty if-lauseen avulla muuttujan laatikon värin muutos.

JavaScript-koodi käsittelee päivitettävien muuttujien tiedot IO_xml_1.xml -tiedostosta (Liite 7), jonne on määritelty jokainen muuttuja erikseen ja muuttujien tunnisteet (id:t) sekä minimi- ja maksimirajat.

Sivulla on myös tekstilaatikko ja painike, joiden avulla saa muutettua Lämpötila-muuttujan arvoa. Päivittämisessä on hyödynnetty iframe-elementtiä, jonka avulla saa dokumentti sijoitettua dokumentin sisälle. Tekstikenttä ja painike ovat siis tehty eri HTML-sivuille (Painike_sivu.html, Liite 8) ja iframen avulla kyseinen sivu saadaan näkymään tällä sivulla. Näin tehtynä painiketta painaessa sivu 3 ei päivity vaan ainoastaan sivu, jolla painike sijaitsee.

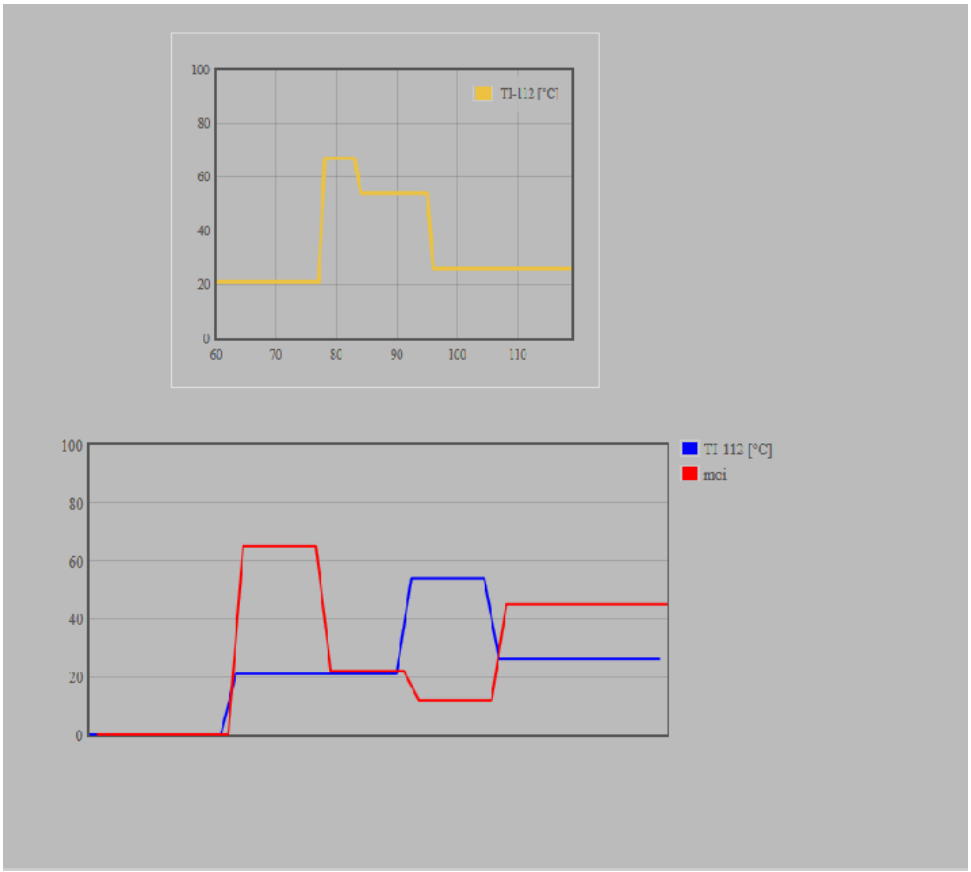
```
<iframe class='temperature' style="border:0" scrolling="no" frameborder="0" src="Painike_sivu.html" >
</iframe>
```

Oikeanpuoleisen kehyksen alimmaisessa laatikossa on enumilla muutettu numero tekstiksi, kuten aikaisemminkin Saadakseen enumit päivittymään AJAX:lla tarvitsee Enumin esittely yksinkertaisesti vain siirtää XML-tiedostoon, jossa on listattu päivitettävät muuttujat.

Sivun vasemmassa alareunassa on kuva, jossa on vihreä laatikko, laatikko muuttuu keltaiseksi eli toisin sanoen kuva vaihtuu, riippuen val_2-muuttujan arvosta. Kuvan paikka on esitelty sivulla ja vaihto ja sen ehdot tapahtuvat xml_esim_1.js-tiedostossa.

5.2.5 Sivu 5

Sivulle 5 (Kuva 14, Liite 9) on luotu 2 trendikaaviota. Kaavioiden tekoon on hyödynnetty valmiita internetistä saatavia jQuery- ja Flot-kirjastoja, jotka on lisätty sivulle samoin kuin muutkin skriptit.



Kuva 14. Trendikaavioita

Kaavioiden muodostus tapahtuu erillisellä JavaScript-tiedostolla, joka sisältää kaavion automaattisen päivityksen ja asetukset. Ylempi kaavio on tehty trend1.js (Liite 10) nimisellä ja alempi trend2.js (Liite 11) nimisellä tiedostolla. Tiedostojen alkuun on luotu päivitys AJAX:n avulla ja lopussa kaavion tarkemmat asetukset, kuten y-akselin minimi- ja maksimiarvot, viivan väri sekä tunniste kaaviolle, jotta se saadaan näkyviin HTML-sivulle. Lisää asetuksista on selitetty kaavioiden JavaScript tiedostoissa.

Kaavioille on luotu oma paikka sivuilla laittamalla kaavion tunniste div-elementtiin sekä luotu sille CSS:llä id ja muotoiluja. Trendi1 on tässä tapauksessa kaavion tunniste.

#trendi1 { width: 100%; height: 100%; font-size: 14px; line-height: 1.2em; }

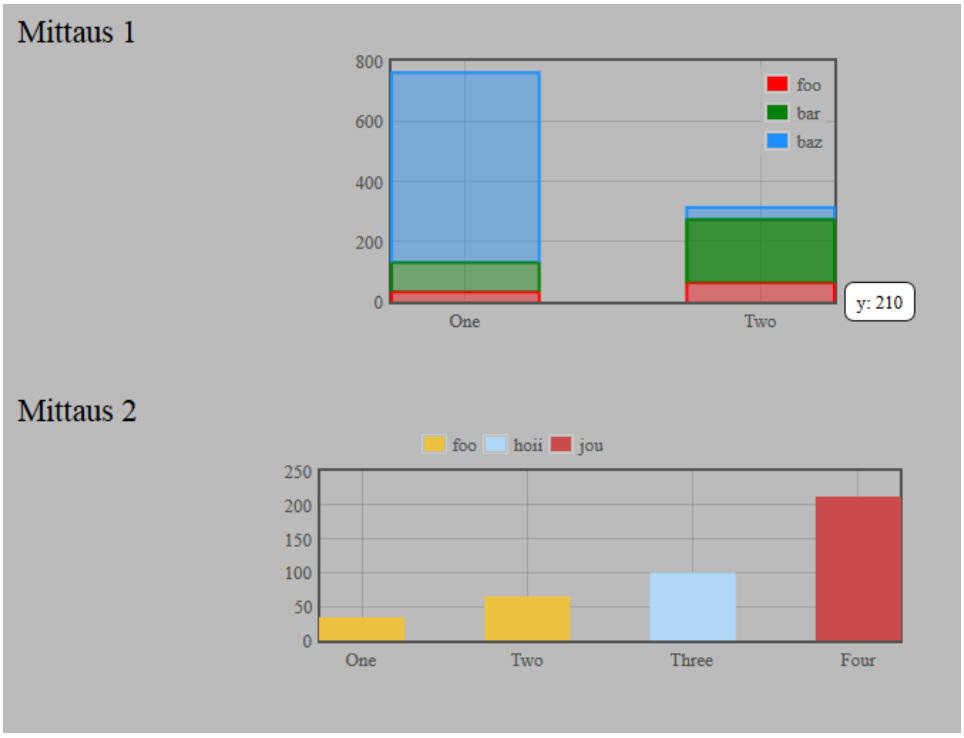
* *

Kaavion selitteiden paikkaa saa myös vaihdettua, kuten alemmassa kaaviossa on tehty.

Kaavion tieto tulee siis suoraan logiikassa olevasta muuttujasta, joten se ei erityisesti varastoidu mihinkään ja alkaa aina alusta, kun sivun päivittää. Kaavioiden muuttujat on laitettu IO_xml_2.xml (Liite 12) nimiseen tiedostoon

5.2.6 Sivu 6

Sivulle 6 (Kuva 15, Liite 13) on tehty kaksi pylväskaaviota. Kaavion teossa on hyödynnetty valmiita jQuery- ja Flot-kirjastoja. Molempiin on lisätty vihjelaatikko, joka tulee esiin, kun kursorin vie kaavion päälle. Vihjelaatikon tekemiseen on hyödynnetty flot-kirjaston liitännäistä (flot.tooltip.min.js). Mittaus 1 on kasautuva pylväskaavio, johon piti ottaa käyttöön flot.stack.min.js -tiedosto.



Kuva 15. Pylväskaavioita.

Kaavioiden muodostus tapahtuu erillisellä JavaScript-tiedostolla, joka sisältää kaavion automaattisen päivityksen ja asetukset. Mittaus 1 on tehty barchart1.js (Liite 14) nimisellä ja Mittaus 2 barchart2.js (Liite 15) nimisellä tiedostolla. Tiedostojen alkuun on luotu päivitys AJAX:n avulla ja lopussa kaavion tarkemmat asetukset, kuten selitteet,

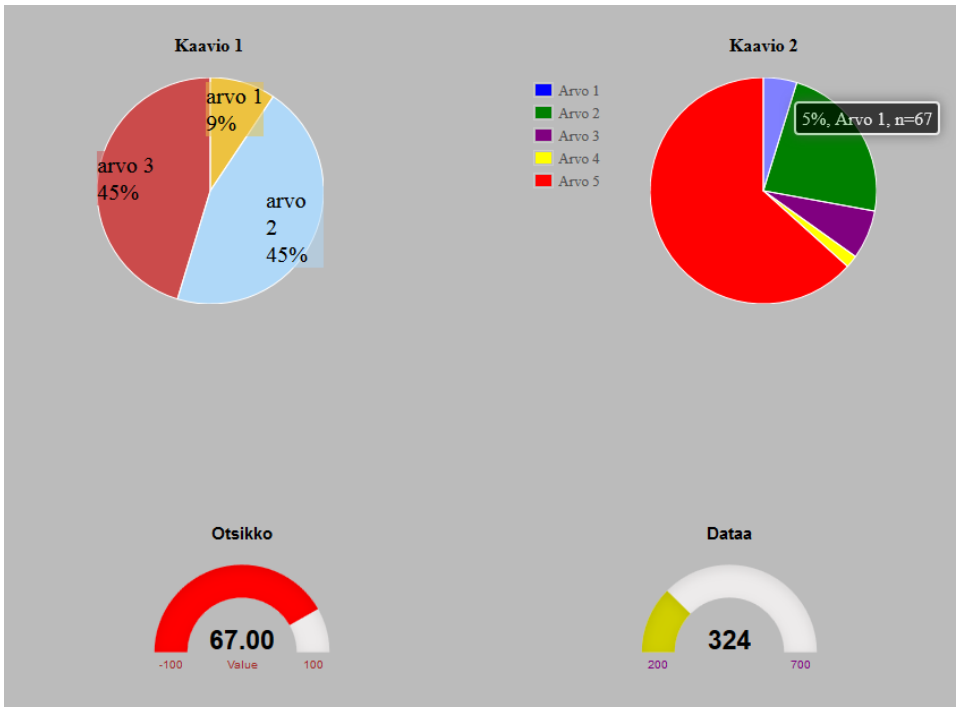
palkkien väri sekä tunniste kaaviolle, jotta se saadaan näkyviin HTML-sivulle. Lisää asetuksista on selitetty kaavioiden JavaScript tiedostoissa.

Kaavioille on luotu oma paikka sivuilla laittamalla kaavion tunniste div-elementtiin sekä luotu id CSS:llä, kuten trendikaavioissakin ylempänä

Kaavion selitteiden paikkaa saa myös vaihdettua, kuten alemmassa kaaviossa on tehty. Kaavioiden x-akselin arvot skaalautuvat automaattisesti logiikalta tulevan arvon perusteella. Kaavioiden y-akselin arvon saa näkyviin viemällä kursorin haluamansa palkin päälle.

5.2.7 Sivu 7

Sivulle 7 (Kuva 16, Liite 16) on luotu ylhäälle kaksi ympyräkaaviota. Ympyräkaavioiden tekoon on hyödynnetty jQuery- ja Flot-kirjastoja ja tooltip-liitännäistä.



Kuva 16. Ympyrädiagrammit ja mittarit.

Kaavioiden muodostus tapahtuu erillisellä JavaScript-tiedostolla, joka sisältää kaavion automaattisen päivityksen ja asetukset. Vasemmanpuoleinen kaavio on tehty piechart.js (Liite 17) nimisellä ja oikeanpuoleinen piechart2.js (Liite 18) nimisellä tiedostolla. Oikeanpuoleisessa kaaviossa saa vihjelaatikon näkyville viemällä kursorin haluttuun kohtaan, jonka tekoon on hyödynnetty samaa liitännäistä kuin pylväskaavioissa.

Alempana on kaksi mittaria, jotka on luotu raphael- ja JustGage-kirjastoilla.
Kaavioiden muodostus tapahtuu erillisellä JavaScript-tiedostolla, joka sisältää kaavion automaattisen päivityksen ja asetukset. Tällä kertaa kaaviot tehtiin samassa JavaScript tiedostossa, xml_gauge.js (Liite 19). Mittarit voivat selaimesta riippuen ymmärtää CSS:lla annetut tyylin muokkaukset hieman eri tavoin.

Tiedostojen alkuun on luotu päivitys AJAX:n avulla ja lopussa kaavion/mittarin tarkemmat asetukset, kuten minimi-ja maksimiarvot, mittarin väri sekä tunniste kaaviolle/mittarille, jotta se saadaan näkyviin HTML-sivulle. Lisää asetuksista on selitetty kaavioiden JavaScript tiedostoissa.

6 YHTEENVETO

Työn tekeminen sujui rauhallisen tasaista tahtia, ilman mitään ratkaisemattomiksi jääneitä ongelmia. Aiheen tutkimisessa ja työnteossa auttoi suuresti Timo Suvelan ja Hannu Asmalan pitämä Internetohjelmoinninkurssi, joka antoi hyvät pohjat työni aiheelle. Kyseinen kurssi myös vaikutti aiheen valintaan.

Selaimella suoritettu monitorointi on mielestäni hyvä lisä logiikasta saatavien tietojen valvontaan ja seurantaan. Aluksi muunkin kuin pelkän muuttujan arvon näyttäminen tuntui hiukan monimutkaiselta, mutta pienen tekemisen jälkeen asioita alkoi ymmärtää paremmin.

Tuotantodatan esittämiseen olen tyytyväinen. Tietenkin hiukan vaikeutti, kun ei tarkalleen määritelty mitä pitäisi tehdä, muuta kuin ominaisuuksia esitellä. Varmasti jäi paljon erilaisia tapoja käymättä läpi, mutta yritin valita semmoisia, jotka olisivat HMI-päätteen kanssa samantyylisiä. Myös luvussa kaksi olevia laskuja laitteiston tehokkuuteen liittyen voisi hyödyntää itse tehdyillä sivuilla.

Harmittamaan jäi hiukan diagnostiikan saatavuus itse tehdyille sivuille. Sen olisin halunnut saada näkymään järkevästi tekemäni templaten sisään. Tällä hetkellä ainoa mahdollisuus, jonka keksin olisi ollut käyttää TIA Portal -ohjelman diagnostiikkalohkoja. Se olisi yksi kehittämisenkohde. Toki Siemensin omat diagnostiikkasivut vaikuttavat hyviltä.

Oppimisen kannalta työn tekeminen oli mielenkiintoinen kokemus. Jotkut asiat olivat muistissa Internetohjelmoinninkurssilta, joitain ominaisuuksia joutui etsimään ja käymään alusta alkaen läpi. Sivujen tekemisessä haastavaa oli HTML-kielen mahdollisuus tehdä asiat monella eri komennolla. JavaScriptillä tehtävät asiat vaativat aika paljon ymmärtämistä, miten ko. kieli toimii ja varsinkin kaavioiden saaminen toimiviksi oli aluksi haastavaa.

LÄHTEET

Villanen, H. 2013. Tuotantokoneiden kokonaistehokkuus, OEE (Overall Equipment Effiency). Viitattu 15.6.2016. http://www.prosessitaito.fi/Tuotantokoneiden_kokonaistehokkuus_OEE.pdf

OEE Industry Standardin www-sivut 2016. 2016. Viitattu 2.6.2016. http://oeeindustrystandard.oeefoundation.org/

PSK 6201 2016, 5. Kunnossapito. Käsitteet ja määritelmät. Maintenance. Terms and definitions. 2011. 3.p. PSK Standardisointiyhdistys ry. Helsinki: PSK. Viitattu 2.7.2016. http://www.psk-standardisointi.fi

Vorne Industries Inc. Six Big Loesses. Viitattu 20.6.2016 http://www.oee.com/oee-six-big-losses.html http://www.oee.com/calculating-oee.html

Vorne Industries Inc. Calculate TEEP. Viitattu 27.6.2016 http://www.oee.com/teep.html

PSK 6201 2016, 7. Kunnossapito. Käsitteet ja määritelmät. Maintenance. Terms and definitions. 2011. 3.p. PSK Standardisointiyhdistys ry. Helsinki: PSK. Viitattu 2.7.2016. http://www.psk-standardisointi.fi

Arno Koch. 2016. OEE Coach. Viitattu 8.6.2016. http://oeecoach.com/different-materials/ http://oeecoach.com/mixed-output/

Mozilla Developer Network, 2016. What is a web server?. Viitattu 24.10.2016. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

Mozilla Developer Network, 2016. An overview of HTTP. Viitattu 23.10.2016. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

West. ’An Introduction to WebSockets’. 18.10.2013. Viitattu 22.10.2016. http://blog.teamtreehouse.com/an-introduction-to-websockets

w3schools:n www-sivut. 2016 Viitattu 1.9.2016 http://www.w3schools.com/

Mozilla Developer Network, 2016. CSS. Viitattu 19.2016 https://developer.mozilla.org/en-US/docs/Web/CSS

Jukka Korpela. JavaScript ja vastaavat. Viitattu 8.10.2016 https://www.cs.tut.fi/~jkorpela/webjulk/3.2.html

Mozilla Developer Network, 2016. AJAX. Getting Started. Viitattu 13.10.2016. https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

O’Reilly, Media Inc. 2016. What is XML? Viitattu 8.10.2016 http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58

The jQuery Foundation. 2016. Viitattu 8.10.2016. https://jquery.com/

Flotcharts:n www-sivut. 2016. Viitattu 8.10.2016. http://www.flotcharts.org/

Justgagen ww-sivut. 2016. Viitattu 8.10.2016. http://justgage.com/

S7-1500 Web server 2016, 62. Viitattu 18.10.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

S7-1500 Web server 2016, 67. Viitattu 18.10.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

S7-1500 Web server 2016. Viitattu 20.10.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

Modicon M251 Logic Controller 2016, 92-105. Viitattu 20.10.2016. http://www.rakurs.su/wp-content/uploads/2015/01/Rukovodstvo-po-programmirovaniya-Modicon-M251-eng.pdf

FP7 PLC 2016. Viitattu 20.10.2016. https://www.panasonic-electric-works.com/eu/fp7-plc.htm

S7-1500 Web server 2016, 10. Viitattu 21.10.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

Creating user-defined we-pages on S7-1200/S7-1500 2016, 7. Viitattu 7.8.2016 https://cache.industry.siemens.com/dl/files/496/68011496/att_858372/v2/68011496_S7-1200_1500_Webserver_DOCU_v21_en.pdf

S7-1500 Web server 2016, 16. Viitattu 8.8.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

S7-1500 Web server 2016, 57. Viitattu 24.10. 2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

Creating user-defined we-pages on S7-1200/S7-1500, 28. Viitattu 23.10.2016. https://support.industry.siemens.com/cs/attachments/68011496/68011496_S7-1200_1500_Webserver_DOCU_v21_en.pdf

S7-1500 Web server 2016, 10. Viitattu 18.10.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

S7-1500 Web server 2016, 22-59, 88-89. Viitattu 20.9.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

Diagnostics in the User Program with S7-1500, 4. Viitattu 19.201.2016. https://cache.industry.siemens.com/dl/files/758/98210758/att_887071/v2/98210758_User_defined_diagnostics_DOKU_v22_en.pdf

S7-1500 Web server, 91. Viitattu 17.9.2016. https://support.industry.siemens.com/cs/attachments/59193560/s71500_webserver_function_manual_en-US_en-US.pdf?download=true

Fonectan www-sivut. 2016 Viitattu 8.11.2016 https://www.fonecta.fi/

LIITE 1 (Template)

```
<!doctype html> <!--Dokumentin tyyppi-->
<html> <!--html-tagien väliin tulee kaikki sisältö -->
<head> <!--head-tagien sisään tulee head-elementit, mm. skriptit -->
<meta charset="utf-8"> <!--Merkistö. Metan avulla voi myös ilmoittaa mm. sivujen kielen. -->
<title>title</title> <!--Sivun otsikko. Näkyy selaimen ylälaidassa -->
<link rel="stylesheet" type="text/css" href="pagestyle.css"> <!--Kutsuu pagestyle-tiedostosta moutoiluja-->

<style>
</style>
</head> <!--Head-elementin lopetus-->

<body> <!--Body-tagien väliin tulee sivujen sisältö -->
 <div class="mainwindow"> <!--Luo "mainwindow":n. Määritelty pagestyle.css -->
  <div class="divHeader"> <!--Luo ns. yläpalkin -->
  <span>Otsikko</span> <!--Yläpalkin otsikkoteksti. -->
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" > <!--Ylä-
palkin kuva muotoilun kanssa. -->
</div> <!--Sulkee divHeader classin -->

        <div class="menu"> <!--Menu/navigaatio palkin sivut -->
      <a href="sivu 2.html">Sivu 2</a>
      <a href="sivu 3.html">Sivu 3</a>
      <a href="sivu 4.html">sivu 4</a>
      <a href="sivu 5.html">sivu 5</a>
      <a href="sivu 6.html">sivu 6</a>
      <a href="sivu 7.html">sivu 7</a>


</div> <!--Sulkee menu classin -->

    <div class="divBody"> <!--Luo alueen johon sivujen sisältö tulee-->
    </div>

 </div>
</body>
</html> <!--Sulkee html-elementin -->
```

LIITE 2 ½ (pagestyle)

```
body { /* Sivun taustaväri */
	background: #CFCFCF;
}

.mainwindow { /* Mainwindow-laatikon luonti ja muotoilu */
	position:relative;
	margin-left:auto;
	margin-right:auto;
	width:1000px;

}

.divHeader { /* divHeaderin luonti ja muotilu */
	position: relative;
	height: 65px;
	background-color: #A52A2A;
	line-height: 65px;
	text-align: center;
	vertical-align: middle;
	font-size:24px;
}
.logo { /* Logon paikka */
	position: absolute;
	top: 2px;
	left:2px
}

.divBody { /* divBodyn muotoilu  */
	position: relative;
	height: 700px;
	background-color: #bbb;
	width: 860px;
	left: 140px;
	top:-700px;
}

.menu { /* Menupalkin pohja */
   list-style-type: none;
   width: 140px;
   background-color: #f1f1f1;
   height: 700px;
  }

.menu a { /* Menun tekstin muokkaus */
   display: inline-block;
	width:70px;
   padding: 0 30px;
   background: #A52A2A;
```

```
 font-size: 16px;
    text-decoration: none;
    line-height: 28px;
    margin: 5px;
    color: white;
}

.menu a:hover { /* Osoitin linkin päällä */
    background-color: #FF7F24;
    color: white;
    font-style:oblique;
}
```

```
<!-- AWP_In_Variable Name='"WebData".Muuttuja2' Name='"WebData".Muuttuja3' -->      <!-- Useamman muuttujan esittely -->

<!-- AWP_In_Variable Name='YesNo' Use='"WebData".Muuttuja4' -->          <!-- Muuttujalla eri nimi sivulla ja PLC:ssä -->

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="refresh" content="5"> <!-- Sivun päivitys 5s välein -->
<title>title</title>

<link rel="stylesheet" type="text/css" href="pagestyle.css">

<style>

.box {  /* Arvojen ympärillä olevien laatikkojen muotoilu */
    background-color: #FFF;
    width: 100px;
    border: 2px solid gray;
    position: relative;
    padding: 5px;
    margin-bottom: 5px;
    text-align: center;
}
.kohta1 {
    position: absolute;
    top: 50px;
    left: 50px;
}

.kohta2 { /* Painikkeiden sijoittelu*/
   position: absolute;
   top: 325px;
   left: 50px;
}

.kohta3 {
   position: absolute;
   top: 50px;
   right: 100px;
}

.dropdown { /* dropdown-valikon sijoittelu */
    position: absolute;
    top: 325px;
```

```
 right: 100px;
}

</style>
</head>

<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >
</div>

     <div class="menu">
        <a href="index.html">Main</a>
       <a href="sivu 2.html">Sivu 2</a>
       <a href="sivu 3.html">Sivu 3</a>
       <a href="sivu 4.html">Sivu 4</a>
       <a href="sivu 5.html">Sivu 5</a>
       <a href="sivu 6.html">Sivu 6</a>
        <a href="sivu 7.html">Sivu 7</a>


</div>
    <div class="divBody">

        <!-- Luo laatikon, jonka sisällä on Muuttuja1:n arvo -->
        <fieldset class="kohta1" style=" width: 250px; height: 75px;">
            <legend>Kohta 1</legend>
              <div class="box" > :="WebData".Muuttuja1: </div>
       </fieldset>


       <fieldset class="kohta2"  style="width: 300px; height: 200px;">
            <legend>Kohta 2</legend>
        <!-- Määritellään nappien paikka sivulla -->
          <div class="box" style="left: 130px; top: 60px; " > :="WebData".Muuttuja2: </div>

       <!-- Luo elementin, jolla voi muokata tietoa post-tyylillä  -->
       <form method="post" action="">
       <!-- Luo 'submit'-tyyppisen painikkeen, value on painikkeen teksti ja style on muotoilu -->
       <input type="submit" value="kolmonen" style="height: 30px; width: 100px; margin: 5px">
       <!-- Käyttäjälle näkymätön, määritellään muuttuja ja sen arvo -->
       <input type="hidden" name='"WebData".Muuttuja2' value="3">
       </form> <!-- Sulkee form-elementin -->

       <form method="post" action="">
```

```
 <input type="submit" value="kasi" style="height: 30px; width: 100px; margin: 5px">
      <input type="hidden" name="'WebData".Muuttuja2' value="8">
      </form>
      </fieldset>



      <fieldset class="kohta3" style=" width: 300px; height: 100px;">
         <legend>Kohta 3</legend>
           <div class="box"> :="WebData".Muuttuja3: </div>
               <form method="post" action="">
           <input type="text" name="'WebData".Muuttuja3' style="height: 23px; width: 35px; background-color:
buttonface" >
           <input type="submit" value="Muuttuja 3" style="height: 30px; width: 100px; margin: 5px">
           </form>
       </fieldset>

    <div class="dropdown">
      <fieldset >
         <legend>Kohta 4</legend>

        Aseta joku tieto
   <form method="post">
   <select name='YesNo'>
   <option value=1>Yes</option>
   <option value=0>No</option>
   </select><input type="submit" value="Aseta"> </form>
  :=YesNo:     <!-- Näyttää Muuttuja4:n arvon -->

      </fieldset>
          </div>
    </div>
 </div>
</body>
</html>
```

```
<!-- AWP_Enum_Def Name='Asento' Values='0:"Ylös",
1:"Alas", 2:"Vasen", 3:"Oikea'" -->
<!-- AWP_Enum_Ref Name='"WebData".position' Enum="Asento" -->

<!-- AWP_In_Variable Name='"WebData".position' -->
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>

<link rel="stylesheet" type="text/css" href="pagestyle.css">

<style></style>

</head>

<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >

</div>

    <div class="menu">
    <a href="index.html">Main</a>
      <a href="sivu 2.html">Sivu 2</a>
      <a href="sivu 3.html">Sivu 3</a>
      <a href="sivu 4.html">Sivu 4</a>
      <a href="sivu 5.html">Sivu 5</a>
      <a href="sivu 6.html">pSivu 6</a>
       <a href="sivu 7.html">Sivu 7</a>
</div>
    <div class="divBody">

  <table border="1" width="120px">
          <tr><td>Nimi</td><td>Arvo</td></tr>
          <!-- AWP_Start_Array Name='"Arvot".values' -->
          <tr><td>:=values[0]:</td><td>:=value:</td></tr>
          <!-- AWP_End_Array -->
       </table>
       <br>
      <br>
       <p>  "WebData.position" is :="WebData".position: </p>

      <form method="post" action="">
      <input type="submit" value="Alas" >
```

```
<input type="hidden" name='"WebData".position' value="Alas">
      </form>
      <br>
      <form method="post" action="">
      <input type="submit" value="Vasen" >
      <input type="hidden" name='"WebData".position' value="Vasen">
      </form>
      <br>
      <br>
      <br>
      <br>
<!-- src on polku kuvien kansioon, jos eri kansiossa kuin sivut.
Kuva vaihtuu kuutio-muuttujan arvon perusteella.
Kuvat ovat tallennut kuutio0 ja kuutio1 nimillä.
Riippuen onko muuttujan arvo 0 vai 1, tulee kuutio0 tai kuutio1 -->
<img src='kuvat\kuutio:="WebData".kuutio:.png'  alt="kuutio" >

    </div>
<footer>
</footer>
</div>

<script>
</script>
</body>
</html>
```

```
<!-- AWP_In_Variable Name="'WebData".Temperature' -->

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>
<link rel="stylesheet" type="text/css" href="pagestyle.css">
<script type="text/javascript" src="xml_esim_1.js"></script> <!--Lisää skriptin -->

<style> /* sivun omia muotoiluja */
.kehikko2 {
    width: 35%;
    height: 60%;
    border:1px solid grey;
    position: absolute;
    top: 15%;
    right: 10%;
    background-color: #D3D3D3;
    }

.laatikko {
    position: relative;
    background-color: #FFF;
    width: 100px;
    border: 2px solid gray;
    top: 50px;
    left: 50px;
    padding: 5px;
    margin-bottom: 5px;
    text-align: center;
    }

.temperature {
    position: relative;
    top: 75px;
    left: 50px;
    }

fieldset {
    width: 200px;
    position: absolute;
    top:15%;
    left: 5%;
    }

    #myImage {
       position: absolute;
```

```
 top: 350px;
      left: 30px;
    }
</style>



</head>

<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >

</div>
     <div class="menu">
    <a href="index.html">Main</a>
       <a href="sivu 2.html">Sivu 2</a>
       <a href="sivu 3.html">Sivu 3</a>
       <a href="sivu 4.html">Sivu 4</a>
       <a href="sivu 5.html">Sivu 5</a>
       <a href="sivu 6.html">Sivu 6</a>
        <a href="sivu 7.html">Sivu 7</a>

</div>
    <div class="divBody">

        <fieldset> <!--Yksi tapa luoda laatikko -->
            <legend> max 200 min 100 </legend> <!--Laatikon otsikko -->
        <div class="laatikko" id="val_1"> :="WebData".Temperature: </div> <!--Luo laatikon aiemmin määritel-
lyillä asetuksilla ja näyttää sen sillä Webdata datablockissa olevan Temperature-muuttujan arvon. -->
        <div class="laatikko" id="val_2"> :="WebData".val_2: </div>


        <img id="myImage" src="" alt="kuva hukassa"> <!--Tähän muodostuu kuva. scr on JavaScriptissä -->

        <iframe class='temperature'
                        style="border:0" scrolling="no"
                        frameborder="0" src="Painike_sivu.html" > <!--Iframen asetuksia-->

                        Your browser does not support inline frames
                        or is currently configured not to display inline frames.
                    </iframe>

      </fieldset>
```

```
<div class="kehikko2">
          max 150 min 50
         <div class="laatikko" id="val_3"> :="WebData".val_3: </div>
          <div class="laatikko" id="val_4"> :="WebData".val_4: </div>
          <div class="laatikko" id="val_5"> :="WebData".val_5: </div>
          <div class="laatikko" id="val_6"> :="WebData".val_6: </div>
          <div class="laatikko" id="val_7"> :="WebData".val_7: </div>
          <div class="laatikko" id="val_8"> :="WebData".position:: </div>


       </div>

    </div>
 <footer>
 </footer>
 </div>

 <script>
readData_XML(); /*Kutsuu xml_esim_1.js tiedostossa olevaa ReadData-funktiota */
</script>


</body>
</html>
```

```
var xmlHttp;
//Luo yhteyden
function createXMLHttpRequest (){
	if (window.XMLHttpRequest)
	{// code for IE7+, Firefox, Chrome, Opera, Safari
	xmlHttp=new XMLHttpRequest();
	}
else
	{// code for IE6, IE5
	xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
	}
}

/* Lähettää HTTP-pyynnön serverille.
 * IO_xml_1.xml tiedostossa on muuttujat, josta tiedot haetaan.
 * GET on metodi, jolla tieto haetaan. True määrittää pyynnön asynkroniseksi.
 * readData_XML() sama kuin HTML-sivulla*/
function readData_XML () {
	createXMLHttpRequest ();
	xmlHttp.onreadystatechange = handleStateChange;
	xmlHttp.open("GET","IO_xml_1.xml",true);
	xmlHttp.send(null);
	setTimeout("readData_XML()",2000);
}

/* Suorittaa pyynnön kommunikoinnin.
 * Pyynnön tila.
 * Serveriltä saadaan vastaus.
 * Serveriltä saatua vastausta voidaan prosessoida.
 * Palauttaa tiedon xml:nä*/
function handleStateChange () {
   if (xmlHttp.readyState == 4) {
	if (xmlHttp.status == 200) {

		var xmlDoc = xmlHttp.responseXML;

			//Tallentaa x muuttujaan tagit <measures> </measures> tagien sisällä. Viittaa XML-tiedostoon.
			var x=xmlDoc.getElementsByTagName("measures");

			// Käy kaikki tiedot läpi XML-tiedostossa.
			for (i=0;i<x.length;i++)
			{

				//Lukee ja tallentaa div_id-tagien välisen tiedon
				div_id=x[i].getElementsByTagName("div_id")[0].childNodes[0].nodeValue;

				//Lukee ja tallentaa value-tagien välisen tiedon
				value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);
```

```
                    // Näyttää konsolissa valuen
                    console.log ("value= " + value);

                    /*Päivittää XML:ssä olevan div_id-tagin avulla tiedon HTML-sivulle*/
                    docDiv = document.getElementById(div_id);
                    docDiv.innerHTML = value;

                    //Siirretään testi2 sanaan XML-tiedoston toinen arvo. Hyädynnetään kuvan vaihdossa
                    var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                    //Minimi- j maksimiarvojen luku
                    limit_max=(x[i].getElementsByTagName("max")[0].childNodes[0].nodeValue);
                    limit_min=(x[i].getElementsByTagName("min")[0].childNodes[0].nodeValue);

                    console.log ("max limit= " + limit_max); // show value on console

                    //Päivittää div-elementin if-lauseen mukaan.
                    if ((parseInt(value) > parseInt(limit_max)) || (parseInt(value) < parseInt(limit_min))) {
                        docDiv.style.backgroundColor='red';

                    }
                    else {
                        docDiv.style.backgroundColor='green';

                    } //Kuvanvaihtoa. myImage kuvan ID sivulla. src on kuvan polku kuvat kansiossa
                        if (testi2 == 2) {
                    document.getElementById('myImage').src='kuvat/kuutio1.png';

                    }
                    else {document.getElementById('myImage').src='kuvat/kuutio0.png';

                    }
                }
            }
        }
    }
}
```

```
<!-- AWP_Enum_Def Name='Asento' Values='0:"Ylös",
1:"Alas", 2:"Vasen", 3:"Oikea'" -->

<!-- AWP_Enum_Ref Name='"WebData".position' Enum="Asento" -->

<siemens> <!--Alkaa ja loppuu siemens tageilla -->
<measures><!--Tänne voi lisätä tageja tarpeen mukaan -->
<div_id>val_1</div_id> <!--Tämän avulla sijoittaa arvon HTML sivulla oikeaan paikkaan -->
<value>:="WebData".Temperature:</value> <!--Sijoitettava muuttuja. WebData datablockissa muuttuja Tempera-
ture -->
<max>200</max> <!--Maksimiarvo -->
<min>100</min> <!--Minimiarvo -->
</measures>

<measures>
<div_id>val_2</div_id>
<value>:="WebData".val_2:</value>
<max>200</max>
<min>100</min>
</measures>

<measures>
<div_id>val_3</div_id>
<value>:="WebData".val_3:</value>
<max>150</max>
<min>50</min>
</measures>

<measures>
<div_id>val_4</div_id>
<value>:="WebData".val_4:</value>
<max>150</max>
<min>50</min>
</measures>

<measures>
<div_id>val_5</div_id>
<value>:="WebData".val_5:</value>
<max>150</max>
<min>50</min>
</measures>

<measures>
<div_id>val_6</div_id>
<value>:="WebData".val_6:</value>
<max>150</max>
<min>50</min>
```

```
</measures>

<measures>
<div_id>val_7</div_id>
<value>:="WebData".val_7:</value>
<max>150</max>
<min>50</min>
</measures>

<measures>
<div_id>val_8</div_id>
<value>:="WebData".position:</value>
<max>150</max>
<min>50</min>
</measures>
</siemens>
```

LIITE 8 (Painike_sivu)

```
<!-- AWP_In_Variable Name="'WebData".Temperature' -->

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>


	</head>

	<body>

				<form method="post" action="" >
 <input type="text" name="'WebData".Temperature' style="height: 23px; width: 35px; background-color: whitesmoke" >
 <input type="submit" value="Lämpötila" style="height: 30px; width: 100px; margin: 5px; background-color: yellow">
</form>


	</body>

</html>
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>

<!--Kutsuu pagestyle-tiedostosta moutoiluja-->
<link rel="stylesheet" type="text/css" href="pagestyle.css">
<!--Lisää skriptejä kaavioiden muodostukseen-->
<script type="text/javascript" language="javascript" src="jquery-3.0.0.min.js"></script>
<script src="jquery.flot.min.js"></script>
<script src="trend1.js"></script>
<script src="trend2.js"></script>
<style>
        .container { /*Ylemmän trendin ympärille luotu laatikko/tausta */
          position: absolute;
          top: 25px;
          left: 150px;
           width: 350px;
           height: 250px;
           padding: 20px 15px 15px 15px;
           border: 1px solid #ddd;

        }
        #trendi1 { /*Ylemmän trendin id. Koko erittäin tärkeä näkyvyyde kannalta! */
           width: 100%;
           height: 100%;
           font-size: 14px;
           line-height: 1.2em;
        }


        #trendi2 {
           position: absolute;
           left: 50px;
           top: 350px;
           width: 550px;
           height: 250px;
        }
        #legendholder {
           position: absolute;
           top: 350px;
           left: 600px;
        }
</style>

</head>
```

```
<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >

</div>

    <div class="menu">
   <a href="index.html">Main</a>
       <a href="sivu 2.html">Sivu 2</a>
       <a href="sivu 3.html">Sivu 3</a>
       <a href="sivu 4.html">Sivu 4</a>
       <a href="sivu 5.html">Sivu 5</a>
       <a href="sivu 6.html">Sivu 6</a>
        <a href="sivu 7.html">Sivu 7</a>
</div>
    <div class="divBody">

       <div class="container"><!--Ylemmän trendin ympärille luotu laatikko-->
          <div id="trendi1"></div> <!--Ylempi trendi-->
             </div>
        <br><br><br>
        <div id="trendi2"></div>
        <div id="legendholder"></div> <!--Alemman trendin selitteet-->

    </div>
 <footer>
 </footer>
 </div>

</body>
</html>
```

```
// Aloittaa funktion, jonka sisällä kaavion päivitys ja luonti tapahtuu
$(function() {

//muuttujia
var xmlHttp;
var updateInterval = 5000; //pävitysväli
var trendData = [];
var totalPoints = 60;
var xval = 0;

// Asettaa alkuarvot, eri kaavioissa voi olla erilainen tapa antaa arvot.
function initData() {
	for(var i = 0; i < totalPoints; i++) {
		trendData.push([xval,0.0]);
		xval += 1;
	}
}

// Luo yhteyden tiedostoon
function createXMLHttpRequest() {
    if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
		xmlHttp=new XMLHttpRequest();
   }
   else {// code for IE6, IE5
		xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
}

function readData() {
   createXMLHttpRequest();
   xmlHttp.onreadystatechange = handleStateChange;
   xmlHttp.open("GET","IO_xml_2.xml",true);
   xmlHttp.send(null);
   setTimeout(readData,updateInterval);
}

function handleStateChange() {
	if (xmlHttp.readyState == 4) {

		if (xmlHttp.status == 200) {

			var xmlDoc = xmlHttp.responseXML;

			var x=xmlDoc.getElementsByTagName("measures");

			for (i=0;i<x.length;i++)
			{
```

```
 value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                console.log ("value= " + value); // show value on console

                var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);

          /*Datan arvojen muokkaus
          Slice(1) jättää ensimmäisen merkin huomioimatta
          Push() lisää uuden tiedon ja parseFloat purkaa muttujan arvon*/
        trendData = trendData.slice(1);
        trendData.push([xval, parseFloat(testi)]);
                xval += 1;

            //Päivittää datan
        plot.setData([{ data: trendData, label: "TI-112 [°C]" }]);
        plot.setupGrid();  //Päivittää asteikon, selitteen jne.
        plot.draw();   //Päivittää piirtoalueen
      }
    }
}
}
initData();     // Kutsuu initData-fukntiota

/* Kaavion asetukset.
#trendi1 on kaavion tunniste. Pitää laittaa samannimisenä sivulle.
Kaavio luodaan ensimmäisen kerran. Dataan tulee aikaisemmin määritelty muuttuja.
Voi antaa myös nimikkeen ja värin, tosin ne pitää antaa uudelleen ylempänä päivityksen yhteydessä myös.
data-rivi on siis samakuin ylempänä setData-rivi     */
var plot = $.plot("#trendi1", [
      { data: trendData, label: "TI-112 [°C]" } ], {
      //Kaavion tarkempia asetuksia
      series: { shadowSize: 0 },
      //y-akselin arvot
      yaxis: { min: 0, max: 100 },
      xaxis: { show: true},
      //viivan paksuus
      lines: {lineWidth: 3 }
});

readData();     // Päivitys
//Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.


});
```

```
$(function() {
var xmlHttp;
var updateInterval = 10000;
var trendData = [];
var totalPoints = 40;
var xval = 0;
var trendData2 = [];

// Muodostaa 2 eri trendiviivaa. Arvot sijoitetaan myöhemmin
//toltalPoints vaikuttaa kuinka monta mittauspistettä on. Tietenkin päivitysväli vaikuttaa
//Itse koin sopivaksi noin 50. Testaamalla selviää omaan tarpeeseen hyvä
function initData() {
	for(var i = 0; i < totalPoints; i++) {
		trendData.push([xval,0.0]);
		xval += 1;
		trendData2.push([xval,0.0]);
		xval += 1;
	}
}

function createXMLHttpRequest() {

   if (window.XMLHttpRequest) {
		xmlHttp=new XMLHttpRequest();
   }
   else {
		xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
}

function readData() {
   createXMLHttpRequest();
   xmlHttp.onreadystatechange = handleStateChange;
   xmlHttp.open("GET","IO_xml_2.xml",true);
   xmlHttp.send(null);
   setTimeout(readData,updateInterval);
}

function handleStateChange() {
	if (xmlHttp.readyState == 4) {

		if (xmlHttp.status == 200) {

			var xmlDoc = xmlHttp.responseXML;


			var x=xmlDoc.getElementsByTagName("measures");
```

```
console.log (x);

                for (i=0;i<x.length;i++)
                {


                    value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                    console.log ("value= " + value);

                    var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                    //Sijoitetaan muuttujien arvot
                    trendData = trendData.slice(1);
                    trendData.push([xval, parseFloat(testi)]);
                    xval += 1;

                    trendData2 = trendData2.slice(1);
                    trendData2.push([xval, parseFloat(testi2)]);
                    xval += 1;


                // Pöivitetään data. Tähän molempien viivojen tiedot.
        plot.setData([{ data: trendData, label: "TI-112 [°C]", color: "blue" }, { data: trendData2, label: "moi",
color: "red" }]);
        plot.setupGrid();
        plot.draw();
    }
  }}}
initData();
//tunniste on #trendi2
var plot = $.plot("#trendi2", [
    { data: trendData, label: "TI-112 [°C]" }, { data: trendData2, label: "moi" } ], {
    series: { shadowSize: 0 },
    //y-akslein minimi- ja maksimirvot. Kannattaa määritellä etukäteen.
    yaxis: { min: 0, max: 100 },
    //x-akselin tiedot piilotettu
    xaxis: { show: false},
    //Määritellään selitteen asetuksia.
    legend: {show: true, container: "#legendholder"}

     //Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.
});

readData();     // Update
});
```

```
<siemens>
<measures>
<title>Lampatila</title>
<div_id>val_1</div_id>
<value>:="WebData".val_1:</value>
<max>200</max>
<min>100</min>
</measures>
<measures>

<div_id>val_2</div_id>
<value>:="WebData".val_2:</value>

</measures>
<measures>

<div_id>val_3</div_id>
<value>:="WebData".val_3:</value>

</measures>
<measures>

<div_id>val_4</div_id>
<value>:="WebData".val_4:</value>

</measures>
<measures>

<div_id>val_5</div_id>
<value>:="WebData".val_5:</value>

</measures>
<measures>

<div_id>val_6</div_id>
<value>:="WebData".val_6:</value>

</measures>
</siemens>
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>

<!--Kutsuu pagestyle-tiedostosta moutoiluja-->
<link rel="stylesheet" type="text/css" href="pagestyle.css">
<script type="text/javascript" language="javascript" src="jquery-3.0.0.min.js"></script>
<script type="text/javascript" language="javascript" src="jquery.flot.min.js"></script>
<script type="text/javascript" src="jquery.flot.stack.min.js"></script>
<script type="text/javascript" src="barchart1.js"></script>
<script type="text/javascript" src="barchart2.js"></script>

<script type="text/javascript" src="jquery.flot.tooltip.min.js"></script>
<style>
#barchart1 { margin: auto; width: 350px; height: 200px;  } /*Koko erittäin tärkeä näkyvyyde kannalta! */
#barchart2 {margin: auto; width: 450px; height: 150px;  }

.teksti {
    font-size: 22px;
    left: 20px;
    position: relative;
}
</style>

</head>

<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >

</div>
     <div class="menu">
   <a href="index.html">Main</a>
       <a href="sivu 2.html">Sivu 2</a>
       <a href="sivu 3.html">Sivu 3</a>
       <a href="sivu 4.html">Sivu 4</a>
       <a href="sivu 5.html">Sivu 5</a>
       <a href="sivu 6.html">Sivu 6</a>
        <a href="sivu 7.html">Sivu 7</a>
</div>
    <div class="divBody">
       <br> <br><div class="teksti">  Mittaus 1  </div>
     <div id="barchart1"></div> <!--Tunniste ylemmälle kaaviolle-->
     <br>
```

```
<br>
    <div class="teksti">  Mittaus 2  </div>
   <div id="legendholder" style="position: relative; left: 300px;"></div>
     <div id="barchart2" ></div>


   </div>
 <footer>

 </footer>
 </div>

</body>
</html>
```

```
// Aloittaa funktion, jonka sisällä kaavion päivitys ja luonti tapahtuu
$(function () {

// muuttujia
var xmlHttp;
var arvo1 = [];
var arvo3 = [];
var arvo2 = [];

// Asettaa alkuarvot, eri kaavioissa voi olla erilainen tapa antaa arvot.
//1 ja 2 meinaa x-akselin kohtia, eli luo kahteen eri kohtaan 3 pylvästä päällekkäin.
function initData() {

        arvo1.push([1,0.0],[2,0.0] );
        arvo2.push([1,0.0],[2,0.0] );
        arvo3.push([1,0.0],[2,0.0] );}

// Luo yhteyden tiedostoon
function createXMLHttpRequest() {
   if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlHttp=new XMLHttpRequest();
   }
   else {// code for IE6, IE5
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }}

/*  Lähettää HTTP-pyynnön serverille
   IO_xml_2.xml tiedostossa on muuttujat, josta tiedot haetaan.
   GET on metodi, jolla tieto haetaan. True määrittää pyynnön asynkroniseksi.
   2000 on päivitysväli millisekunteina*/
function readData() {
   createXMLHttpRequest();
   xmlHttp.onreadystatechange = handleStateChange;
   xmlHttp.open("GET","IO_xml_2.xml",true);
   xmlHttp.send(null);
   setTimeout(readData,2000);
}

/* Suorittaa pyynnön kommunikoinnin.
  Pyynnön tila. Serveriltä saadaan vastaus.
  Serveriltä saatua vastausta voidaan prosessoida.
  Palauttaa tiedon xml:nä*/
function handleStateChange() {
   if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var xmlDoc = xmlHttp.responseXML;

            //Tallentaa x muuttujaan tagit <measures> </measures> tagien sisällä
```

```
var x=xmlDoc.getElementsByTagName("measures");
                    // Käy kaikki tiedot lävitse XML-tiedostossa.
                      for (i=0;i<x.length;i++) {

                      /*      Lukee datan (PLC:n arvon/valuen) XML-tiedostosta
                            Lukee ja tallentaa datan arvon, joka on <value> .. </value> - tagien välissä */
                      value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

  //Selkeyden vuoksi arvot tallennettu eirllisiin muuttujiin
  var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
  var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);
  var testi3 = parseInt(x[2].getElementsByTagName("value")[0].childNodes[0].nodeValue);
  var testi4 = parseInt(x[3].getElementsByTagName("value")[0].childNodes[0].nodeValue);
  var testi5 = parseInt(x[4].getElementsByTagName("value")[0].childNodes[0].nodeValue);
  var testi6 = parseInt(x[5].getElementsByTagName("value")[0].childNodes[0].nodeValue);


                     /*Datan arvojen muokkaus
                      Slice(1) jättää ensimmäisen merkin huomioimatta
                      Push() lisää uuden tiedon ja parseFloat purkaa muttujan arvon*/
                      arvo1 = arvo1.slice(1);
                      arvo1.push([1, parseFloat(testi)]);
                      arvo1 = arvo1.slice(1);
                      arvo1.push([2, parseFloat(testi2)]);

                      arvo2 = arvo2.slice(1);
                      arvo2.push([1, parseFloat(testi3)]);
                      arvo2 = arvo2.slice(1);
                      arvo2.push([2, parseFloat(testi4)]);

                      arvo3 = arvo3.slice(1);
                      arvo3.push([1, parseFloat(testi5)]);
                      arvo3 = arvo3.slice(1);
                      arvo3.push([2, parseFloat(testi6)]);

        //Päivittää datan
       plot.setData([{data: arvo1, label:"foo", color: "red" },
                {data: arvo2, label: "bar", color: "green" },
                {data: arvo3, label: "baz", color: "#1E90FF" }]);
       plot.setupGrid(); //Päivittää asteikon, selitteen jne.
       plot.draw();  //Päivittää piirtoalueen
     }}}}

initData(); // Kutsuu initData-fukntiota


/* Kaavion asetukset.
```

```
#barchart1 on kaavion tunniste. Pitää laittaa samannimisenä sivulle.
Kaavio luodaan ensimmäisen kerran. Dataan tulee aikaisemmin määritelty muuttuja.
Voi antaa myös nimikkeen ja värin, tosin ne pitää antaa uudelleen ylempänä päivityksen yhteydessä myös.
var plot kohdan data arvot olisi hyvä olla plot.setDatan kanssa yhtäläisiä    */

var plot = $.plot("#barchart1",
          [{data: arvo1, label:"foo" },
              {data: arvo2, label: "bar" },
              {data: arvo3, label: "baz" }], {

//Kaavion tarkempia asetuksia
  series: {stack: true,  //stäkkäys päälle tai pois
       lines: {show: false, steps: false },
       //palkkien leveys ja keskitys
       bars: {show: true, barWidth: 0.5, align: 'center'}},
    //x-akseleiden nimet. Pystyy laittamaan pois näkyvistäkin
       xaxis: {ticks: [[1,'One'], [2,'Two']]},
       grid: {hoverable: true}, //tooltipin kannalta tärkeä
       tooltip: {    //tooltip
                 show: true, //tooltip
                 content: "y: %y" //tooltipin sisältö.
              },

  });
    readData();  //päivitys
});

//Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.
//Käytössä tooltip-plugin, jonka löytää flotchartin plugineista sekä suoraan https://github.com/krzysu/flot.tooltip
//Yllä olevassa linkissä tarkemmin asetuksia liittyen ko. pluginiin
```

```
$(function () {

var xmlHttp;
var updateInterval = 1000;
var trendData = [];
var trend2Data = [];
var trend3Data = [];

// Voi antaa kahdella eri tavalla datan, molemmat esitelty.
function initData() {

		trendData.push([1,0.0],[2,0.0] );
		trend2Data.push([3, 0.0] );
		 trend3Data.push([4, 0.0] );
		}


function createXMLHttpRequest() {

   if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
		xmlHttp=new XMLHttpRequest();
   }
   else {// code for IE6, IE5
		xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
   }
}

function readData() {
   createXMLHttpRequest();
   xmlHttp.onreadystatechange = handleStateChange;
   xmlHttp.open("GET","IO_xml_2.xml",true);
   xmlHttp.send(null);
   setTimeout(readData,updateInterval);
}

function handleStateChange() {
      if (xmlHttp.readyState == 4) {

		if (xmlHttp.status == 200) {

			var xmlDoc = xmlHttp.responseXML;
			var x=xmlDoc.getElementsByTagName("measures");

			console.log (x);

			for (i=0;i<x.length;i++)
			{
```

```
value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                    console.log ("value= " + value);

                    var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi3 = parseInt(x[2].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi4 = parseInt(x[3].getElementsByTagName("value")[0].childNodes[0].nodeValue);


                trendData = trendData.slice(1);
                trendData.push([1, parseFloat(testi)]);

                trendData = trendData.slice(1);
                trendData.push([2, parseFloat(testi2)]);

                trend2Data = trend2Data.slice(1);
                trend2Data.push([3, parseFloat(testi3)]);

                trend3Data = trend3Data.slice(1);
                trend3Data.push([4, parseFloat(testi4)]);


        plot.setData([{ data: trendData, label: "foo" }, { data: trend2Data, label: "hoii" },{ data: trend3Data, label:
"jou" }  ]);
        plot.setupGrid();
        plot.draw();
      }
   }
}
}


initData();

 var plot = $.plot($("#barchart2"),[
 { data: trendData, label: "foo"},{ data: trend2Data, label: "hoii" },{ data: trend3Data, label: "jou" } ], {
 series: {stack: 0, //Ei ole kasautuva, eli yksi arvo yhdessä kohdassa
            lines: {show: false, steps: false },
            //pylväiden paksuus jne.
            bars: {show: true, barWidth: 0.5, align: 'center', fill:1}},
        //x-akselin selityksiä. Numerot iittaavat datassa oleviin kohtiin.
     xaxis: {ticks: [[1,'One'], [2,'Two'],[3, 'Three'], [4, 'Four']]},
   grid: {hoverable: true},
   tooltip: {
            show: true,
            //Näyttää y-akselin arvot
```

```
    content: "y: %y"
            },
      legend: {show: true, container: "#legendholder", noColumns : 4}

  });

    readData();
});
//Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.
//Käytössö tooltip-plugin, jonka löytää flotchartin plugineista sekä suoraan https://github.com/krzysu/flot.tooltip
//Yllä olevassa linkissä myös tarkemmin asetuksia liittyen ko. pluginiin
```

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>title</title>

<link rel="stylesheet" type="text/css" href="pagestyle.css">
<script src="jquery-3.0.0.min.js"></script>

<script type="text/javascript" src="jquery.flot.min.js"></script>
<script type="text/javascript" src="jquery.flot.pie.min.js"></script>
<script type="text/javascript" src="piechart.js"></script>
<script type="text/javascript" src="piechart2.js"></script>
<script type="text/javascript" src="jquery.flot.tooltip.min.js"></script>

<script type="text/javascript" src="raphael-2.1.4.min.js"></script>
<script type="text/javascript" src="justgage.js"></script>
<script type="text/javascript" src="xml_gauge.js"></script>

<style>
#piechart1 {  /*Koko erittäin tärkeä näkyvyyde kannalta! */
    position:  absolute;
    top: 10%;
    left: 10%;
    width: 200px;
    height: 200px;
}
 #piechart2 { /*Koko erittäin tärkeä näkyvyyde kannalta! */
 position:  absolute;
    top: 10%;
    right: 10%;
    width: 200px;
    height: 200px;
    }

    #legendholder1 {
        position: absolute;
        top: 10%;
        right: 38%
    }

    #g1 { /*Mittarin paikka ja muotoilua.  */
        position: absolute;
        top: 65%;
        left: -25%;
    }
    #g2 {
```

```
 position: absolute;
       top: 65%;
       right: -25%;

    }
</style>


</head>

<body>
 <div class="mainwindow">
  <div class="divHeader">
  <span>Otsikko</span>
  <img class = "logo" src="raumasterpaper_logo.png" width="152" height="36" alt="Raumaster paper" >

</div>
     <div class="menu">
    <a href="index.html">Main</a>
       <a href="sivu 2.html">Sivu 2</a>
       <a href="sivu 3.html">Sivu 3</a>
       <a href="sivu 4.html">Sivu 4</a>
       <a href="sivu 5.html">Sivu 5</a>
       <a href="sivu 6.html">Sivu 6</a>
        <a href="sivu 7.html">Sivu 7</a>
</div>
     <div class="divBody">
        <br>

       <div id="piechart1"></div>  <!--Ensimmäisen kaavion paikka-->
       <div id="piechart2"></div>
        <div id="legendholder1"></div>

         <h4 style="position: absolute; top: 10px; left: 18%">Kaavio 1</h4>
        <div id="g1"></div> <!--Mittarin paikka-->

        <h4 style="position: absolute; top: 10px; right: 18%">Kaavio 2</h4>
       <div id="g2"></div>

     </div>
 <footer></footer>
 </div>



</body>
</html>
```

```
$(function () {

 var xmlHttp;
  var updateInterval = 1000;
  var arvo1 = [];
   var arvo2 = [];
   var arvo3 = [];

 //Kaavioon siirrettävän datan tyyli. Kolme eri arvoa
 function initData() {

       arvo1.push(0.0);
       arvo2.push(0.0);
       arvo3.push(0.0);
}

 function createXMLHttpRequest() {

    if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
            xmlHttp=new XMLHttpRequest();
    }
    else {// code for IE6, IE5
            xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}

 function readData() {
    createXMLHttpRequest();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET","IO_xml_2.xml",true);
    xmlHttp.send(null);
    setTimeout(readData,updateInterval);
}


function handleStateChange() {
      if (xmlHttp.readyState == 4) {

            if (xmlHttp.status == 200) {

                  var xmlDoc = xmlHttp.responseXML;


                  var x=xmlDoc.getElementsByTagName("measures");


                  for (i=0;i<x.length;i++)
                  {
```

```
                value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                console.log ("value= " + value);

                var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                var testi3 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);



                arvo1 = arvo1.slice(1);
                arvo1.push(parseFloat(testi));


                arvo2 = arvo2.slice(1);
                arvo2.push(parseFloat(testi2));

                arvo3 = arvo3.slice(1);
                arvo3.push(parseFloat(testi3));



        plot.setData([{label:"arvo 1", data: arvo1 }, { label: "arvo 2",  data: arvo2}, { label: "arvo 3",  data:
arvo3}]);
        plot.setupGrid();
        plot.draw();
      }
   }
}
}

initData();


var plot = $.plot("#piechart1", [{label:"arvo 1", data: arvo1 }, { label: "arvo 2",  data: arvo2}, { label: "arvo 3",
data: arvo3}], {
  series: {
            pie: { //Kaavion tyyppi
        show: true,
        //Kaavion koko
        radius: 1,
        label: {
          //näyttää etiketin/labelin
          show: true,
          //muokkaa paikkaa jossa etiketti/label sijaitsee
          radius: 3/4,
          //labelin näyttämä tieto. Nyt määritelty näyttämään prosentteina
```

```
 formatter: function (label, series) {
                return "<div style='font-size:20px;'>" + label + "<br/>" +Math.round(series.percent) + "%</div>";
                },
                //etiketin/labelin taustan asetuksia
            background: {
                opacity: 0.5,
                //color: '#000',

            }
        }
    }
  },
  legend: {
     show: false
  }
 });

  readData();
});

//Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.
 //Käytössö tooltip-plugin, jonka löytää flotchartin plugineista sekä suoraan https://github.com/krzysu/flot.tooltip
 //Yllä olevassa linkissä asetuksia liittyen ko. pluginiin
```

```
$(function () {

 var xmlHttp;
  var updateInterval = 1000;
  var arvo1 = [];
   var arvo2 = [];
   var arvo3 =[];
   var arvo4 =[];
   var arvo5 =[];

 function initData() {  //5 eri arvoa


        arvo1.push(0.0);
        arvo2.push(0.0);
        arvo3.push(0.0);
        arvo4.push(0.0);
        arvo5.push(0.0);

}


 function createXMLHttpRequest() {
    // Create a connection to the file.
    if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
                xmlHttp=new XMLHttpRequest();
    }
    else {// code for IE6, IE5
                xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}

 function readData() {  // Send XMLHttpRequest to PLC
    createXMLHttpRequest();
    xmlHttp.onreadystatechange = handleStateChange;
    xmlHttp.open("GET","IO_xml_2.xml",true);
    xmlHttp.send(null);
    setTimeout(readData,updateInterval);
}


function handleStateChange() {
       if (xmlHttp.readyState == 4) {

              if (xmlHttp.status == 200) {

                     var xmlDoc = xmlHttp.responseXML;
```

```
var x=xmlDoc.getElementsByTagName("measures");


                for (i=0;i<x.length;i++)
                {

                    value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

                    console.log ("value= " + value);

                    var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi3 = parseInt(x[2].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi4 = parseInt(x[3].getElementsByTagName("value")[0].childNodes[0].nodeValue);
                    var testi5 = parseInt(x[4].getElementsByTagName("value")[0].childNodes[0].nodeValue);


                    arvo1 = arvo1.slice(1);
                    arvo1.push(parseFloat(testi));


                    arvo2 = arvo2.slice(1);
                    arvo2.push(parseFloat(testi2));


                    arvo3 = arvo3.slice(1);
                    arvo3.push(parseFloat(testi3));

                    arvo4 = arvo4.slice(1);
                    arvo4.push(parseFloat(testi4));

                    arvo5 = arvo5.slice(1);
                    arvo5.push(parseFloat(testi5));


        //Päivitystä
        plot.setData([{data: arvo1, label:"Arvo 1", color:"blue" }, { label: "Arvo 2",  data: arvo2, color:"green" }, {
label: "Arvo 3",  data: arvo3, color: "purple"} , { label: "Arvo 4",  data: arvo4, color: "yellow"}, { label: "Arvo 5",
data: arvo5, color: "red"}]);
        plot.setupGrid();
        plot.draw();
      }
   }
}
}

initData();
```

```
var plot = $.plot("#piechart2", [{data: arvo1, label:"Arvo 1", color:"blue" }, { label: "Arvo 2",  data: arvo2,
color:"green" }, { label: "Arvo 3",  data: arvo3, color: "purple"} , { label: "Arvo 4",  data: arvo4, color: "yellow"},
{ label: "Arvo 5",  data: arvo5, color: "red"}], {
      series: {
        pie: {
          show: true,
          raduius: 3/4,

        }
      },
      //Sallii ominaisuuden joka näyttää tieotoja, kun kaavion päällä
      grid: {
        hoverable: true
      },
      //Näyettävien tieotjen määritys
      tooltip: {
        show: true,
        content: "%p.0%, %s, n=%n", // Näyttää prosentit, pyöristää 2 desimaaliin
        shifts: {
          x: 20,
          y: 0
        },
        defaultTheme: false
      },
      legend: {
        show: true, container: "#legendholder1",
    }
    });

readData ();

  });
  //Lisää asetuksia flotchart:n sivuilta (http://www.flotcharts.org/) document-osiossa.
//Käytössö tooltip-plugin, jonka löytää flotchartin plugineista sekä suoraan https://github.com/krzysu/flot.tooltip
  //Yllä olevassa linkissä asetuksia liittyen ko. pluginiin
```

```
//Alku hieman erilainen verrattuna muihin
$(document).ready(function(){

var xmlHttp;
function createXMLHttpRequest (){

if (window.XMLHttpRequest)
	{// code for IE7+, Firefox, Chrome, Opera, Safari
	xmlHttp=new XMLHttpRequest();
	}
else
	{// code for IE6, IE5
	xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
	}}

function readData () {
	createXMLHttpRequest ();
	xmlHttp.onreadystatechange = handleStateChange;
	xmlHttp.open("GET","IO_xml_2.xml",true);
	xmlHttp.send(null);
	setTimeout(readData,2000);
}

function handleStateChange () {

	if (xmlHttp.readyState == 4) {

		if (xmlHttp.status == 200) {

			var xmlDoc = xmlHttp.responseXML;
			var x=xmlDoc.getElementsByTagName("measures");

			console.log (x);


			for (i=0;i<x.length;i++)
			{
				value=(x[i].getElementsByTagName("value")[0].childNodes[0].nodeValue);

				console.log ("value= " + value);

			var testi = parseInt(x[0].getElementsByTagName("value")[0].childNodes[0].nodeValue);
			var testi2 = parseInt(x[1].getElementsByTagName("value")[0].childNodes[0].nodeValue);

				//Suorittaa arvon päivityksen
```

```
g1.refresh(parseFloat(testi));
                    g2.refresh(parseFloat(testi2));

                }
            }   }}

      //Luo mittarin ja määrittelee sen astukset
     //Kaaviossa voi erikseen määritellä värit riippuen muuttujien arvojen suuruudesta
    // Kaavio myös automaattisesti vaihtaa mittarin väriä riippuen arvosta
      var g1 = new JustGage({
        id: "g1", //Vertaa ylempänä refresh-kohtaan
        value: -9, //alkup.arvo
        min: -100, // minimiarvo
        max: 100, //maksimiarvo
        //asetuksia liittyen otiskkoon ja tekstin väreihin
        title: "Otsikko",
        titleFontColor: "black",
        label: "Value",
        labelFontColor: "brown",
        decimals: 2,
    gaugeWidthScale: 1.0,
    //Erikseen määritellyt arvot
    customSectors: [{
      color : "#00ff00",
      lo : -100,
      hi : 0
    },{
      color : "#ff0000",
      lo : 0,
      hi : 100
    }],
    counter: true
  });

   var g2 = new JustGage({
        id: "g2",
        value: 20,
        min: 200,
        max: 700,
        title: "Dataa",
        titleFontColor: "black",

        labelFontColor: "purple"
      });

//Lisää asetuksia justgagen sivuilta (http://justgage.com/) Demos-kohdasta
 readData();  });
```