PLC – PC kommunikoinnin turvallisuuskäytäntöjen päivittäminen Makron Automation Oy:llä

LAB-ammattikorkeakoulu
Insinööri (AMK) Konetekniikka
2023 Aki Pajarinen

Tiivistelmä

Tekijä	Julkaisun laji	Valmistumisaika
Aki Pajarinen	Opinnäytetyö, AMK	Kevät 2023
	Sivumäärä 51	

Työn nimi

PLC – PC kommunikoinnin turvallisuuskäytäntöjen päivittäminen Makron Automation Oy:llä

Tutkinto ja koulutusala

Insinööri (AMK), konetekniikan koulutus

Toimeksiantajaorganisaatio

Makron Automation Oy

Tiivistelmä

Opinnäytetyön tavoitteena on toteuttaa verkkoselainpohjainen käyttöliittymä Siemensin logiikalle, perustaa verkkopalvelin käyttöliittymälle sekä salata, suojata ja sertifioida laitteiden välinen viestintä.

Makron Automation esitti toiveen vastata jatkuvasti kasvavaan kysyntään kyberturvallisuuden päivittämiseksi käyttöön ottamalla käyttöön Siemensin uuden Web-API rajapinta teknologian.

Työ toteutettiin kirjoittamalla käyttöliittymän verkkosivu web-kehitys työkaluilla ja sulauttaa se keskustelemaan logiikan kanssa verkkopohjaisilla tiedonsiirtomenetelmillä.

Haluttuihin tavoitteisiin päästiin ja yhteydet saatiin turvattua urkinnalta, sekä yhteyden muodostus sertifioitua.

Asiasanat

Web-API, verkkopalvelin, verkkoselain, Siemens, Sertifikaatti, kyberturvallisuus, käyttöliittymä

Abstract

Author	Type of Publication	Published
Aki Pajarinen	Bachelor's thesis	Spring 2023
	Number of Pages 51	

Title of Publication

Upgrade of the communication information security policies between PC & PLC

Degree, Field of Study

Bachelor of Engineering, Mechanical Engineering

Organisation of the client

Makron Automation Oy

Abstract

The purpose of this thesis is to establish a web-browser based User interface for a Siemens PLC, establish a web server and secure and certify the communication between the devices.

Makron Automation presented a request to answer a growing need for cybersecurity, by harnessing the new Siemens Web-API web technology into use.

The thesis was executed by writing the website for the interface using web-development technologies and fusing the interface to communicate with the logic by using web-based communication methods.

The desired objectives were reached and the connections between all devices were made secure and establishing the connection was certified.

Keywords

Web-API, webserver, web-browser, Siemens, Certificate, Cybersecurity, Interface

Sisällys

1 Johdanto ... 1
2 Makron Oy ... 2
2.1 Yrityksen historia ... 2
2.2 Yritys nykyään ... 3
3 Tausta ja tavoite ... 4
4 PLC ... 5
4.1 Ohjelmoitava logiikka ... 5
4.2 Siemens S7-1500 ... 5
5 Siemens TIA-Portal ... 7
5.1 Kehitystyökalu ... 7
5.2 Ohjelmalohkot ... 7
5.3 Muuttujatyypit ... 8
5.4 Logiikan ohjelmointi kielet ... 9
5.5 Web-API ... 12
5.6 Konfiguraatio ... 13
5.7 Sertifikaatti ... 13
5.8 PUT/GET ... 16
6 Ohjelmointikielet ... 17
6.1 Ohjelmointikielet yleisesti ... 17
6.2 JavaScript ... 17
6.3 HTML ... 18
6.4 CSS ... 19
6.5 JavaScript JSON ja AJAX ... 20
6.5.1 JSON ... 20
6.5.2 AJAX ... 21
6.6 PHP ... 23
6.7 C# ... 24
7 Verkkosivujen luomiseen käytettävät työkalut ... 26
7.1 Verkkosivut ... 26
7.2 VS Code ohjelmointiympäristö ... 26
7.3 Microsoft IIS ... 27
7.4 Kyberturvallisuus ... 28
7.5 Wireshark ... 29
8 Logiikkaohjelma ja käyttöliittymä ... 30

8.1 Logiikkaohjelma TIA-Portalissa 30
8.2 Käyttöliittymän koodi 32
8.3 Yhteyden suojaus 39
9 Makron Service 42
10 Tulokset ja vertailu 43
10.1 Tarkasteltavien asioiden määrittely 43
10.2 Testien tulokset 43
10.2.1 Web-API 44
10.2.2 Service 46
11 Yhteenveto ja pohdinta 48
Lähteet 49

1 Johdanto

Siemens on viimevuosina päivittänyt käytäntöjään luoda verkkosivupohjainen käyttöliittymä ohjelmoitavalle logiikalle. Käyttöliittymän luominen on mahdollistettu lähes kokonaan Siemensin oman kehitysympäristön ulkopuolella, missä automaatiojärjestelmän ohjauspaneelin käyttöliittymä tavallisesti rakennetaan. Uudella verkkopalvelinpohjaisella menetelmällä käyttöliittymän luojan ei tarvitse ymmärtää varsinaisen ohjelmoitavan logiikan toiminnasta, vaan käyttöliittymä voidaan toteuttaa tavanomaisten verkkosivujen tavoin.

Vastaavanlainen käyttöliittymän luonti on ollut mahdollista aikaisemmin, mutta se on vaatinut vahvaa ohjelmallista integrointia ohjelmoitavan logiikan puolella. Uudella menetelmällä käyttöliittymän kehittäjälle riittää logiikan IP-osoite ja lista muuttujista, joita logiikkaohjelma käyttää.

Samalla uudistus nykyaikaistaa ja turvallistaa verkkoselainpohjaisen käyttöliittymän ja ohjelmoitavan logiikan välistä yhteyttä vaatimalla yhteydeltä nykypäiväisiä suojausmenetelmiä. Kyberhyökkäykset ovat varteenotettava uhka nykypäivänä, Sillä lähes kaikki älylaitteet ovat kiinni jonkinlaisessa verkossa.

Tämän opinnäytetyön tarkoituksena on perustaa ohjelmoitavan logiikan kanssa keskusteleva palvelin-PC, jonka avulla voi verkkoyhteyden välityksellä lukea ja kirjoittaa tietoa ohjelmoitavalle logiikalle, sekä luoda verkkoselainpohjainen käyttöliittymä ohjelmalle. Yhteysketju asiakaslaitteelta logiikalle tulee suojata ja sertifioida. Tavoitteena on käyttää onnistuneesti tiedonsiirtoon Siemensin uutta Web-rajapintaa entisen sokettiyhteyden sijasta ja varmistaa, ettei vanha suojaamattomampi yhteys pääse enää käsiksi logiikan tietoihin.

2 Makron Oy

2.1 Yrityksen historia

Makron Oy on Jalo Paanasen vuonna 1974 perustama tuotantokoneita toimittava yritys. Alkuaikoina yritys toimitti koneita kaikille halukkaille ja palveli kaikkia teollisuudenaloja. (Makron Oy 2023a.)

Historiansa alussa vuosina 1974–1990. Makron kehitti tuotantoteknologioita, polyuretaanisten jäähdytyspaneelien, elementtitalojen, hirsitalojen ja sellueristeen tuotantoon ja valmistukseen. Teknologiaa kehitettiin yhteistyössä suomalaisten asiakkaiden kanssa. Tuotantokoneita ja teknologiaa vietiin Venäjälle, Pohjois-Amerikkaan, Pohjoismaihin ja Eurooppaan. (Makron Oy. 2023a.)

Suomen ja Neuvostoliiton välinen kaupan aikakausi vuosina 1980–1990 oli Makron Oy:lle tuottoisa. Neuvostoliitossa oli tuolloin kysyntää kehittyneille teknologioille. Samana aikakautena Makron Oy aloitti toimintansa sopimusvalmistajana suomalaisille koneteknologiayrityksille. Neuvostoliiton vientikaupan supistuminen 90-luvun laman koetellessa oli vaikeata aikaa Makron Oy:lle. (Makron Oy 2023a.)

Vuonna 1999 Pekka Leppänen osti Makron Oy:n nimen ja konepajatoiminnan. Toiminta keskitettiin 4000m^2 tehtaaseen Jokimaalla. Muutama vuosi myöhemmin, vuonna 2004 Makron siirtyi Hollolaan, josta yritys osti itävaltalaisen Andritzin konepajan. (ESS. 2019.) Makronin liikevaihto kasvoi tuolloin 15 miljoonalla (Makron 2023a).

Kun Konepajatoiminta vahvistui, oli selvää, että Makronin oli kehitettävä älykkäitä palveluita. Vuosina 2006–2013 Makron osti Elmont Oy:n, Mekateam Oy:n ja osan Solmex Oy:stä vahvistaakseen myös konesuunnittelun, teollisuusautomaation sekä asennus- ja elinkaaripalvelujen osaamistaan. (Makron Oy 2023a.)

Makron osti Makron Engineeringin teknologiat ja liiketoiminnan vuonna 2014. Tällöin hirsitalojen, elementtitalojen, selluvillaeristeen ja sandwich-paneelien tuotantoteknologiat palasivat Makronille. (Makron Oy 2023a.)

Makron myi hirsitalotuotannon ja lopetti Hollolan konepajan vuonna 2020. Jatkossa Makron keskittyy teollisuusautomaatioon, mekaniikkasuunnitteluun ja keskittää valmistuksen Viron konepajalle. (Makron Oy 2023a.)

2.2 Yritys nykyään

Makron on teollisuuden kokonaistoimittaja. Yrityksen ydinosaaminen koostuu vaativien koneiden ja tuotantolinjojen suunnittelusta ja valmistuksesta sekä teollisuuden automaatioratkaisujen toimittamisesta. Makronilla ei ole varsinaisesti tiettyä päätuotetta, vaan yritys valmistaa koneita ja laitteita sopimusvalmistajana projektiluontoisesti. (Makron Oy 2023b.)

Makronin pääkonttori ja automaatiokeskusvalmistus sijaitsee Lahdessa ja konepajavalmistus Virossa. Lisäksi yrityksellä on toimistot Savonlinnassa ja Espoossa. Työntekijöitä Makronilla on yli 100. Liikevaihtoa yrityksellä on yli 13 miljoonaa. (Makron Oy 2023b.)

Makron suunnittelee ja valmistaa koneita metsä-, paperi-, meri-, sellu- ja kaivosteollisuuteen. Metsäteollisuuteen toimitetaan mm. sahalaitoksien tuotantolinjoja, levyjen valmistus- ja käsittelylinjoja, raaka-aineen käsittelylinjoja ja pellettien tuotantolinjoja. Paperiteollisuuteen toimitetaan esimerkiksi rullaus- ja rullankäsittelykoneita. Meriteollisuuteen Makron toimittaa jätteen- ja jätevedenkäsittelyjärjestelmiä, alipainejärjestelmiä ja järjestelmiä puhtaan veden tuotantoon laivoille. (Makron Oy 2023b.)

Suunnittelupalveluihin kuuluu sekä mekaniikka-, laitos- ja prosessisuunnittelu että automaatio-, ja sähkösuunnittelu. Yritys tarjoaa myös tuotekehityspalveluja ja teknistä laskentaa parantaakseen koneen tai laitteen ominaisuuksia ja varmistaakseen sen kestävyyden ja turvallisuuden. Teollisuusautomaatioon kuuluu vahvasti sähkö- ja automaatioasennukset, PLC ja PC ohjelmointi, keskusvalmistus ja käyttöönotto. (Makron Oy 2023b.)

Myös asennus- ja elinkaaripalvelut kuuluvat yrityksen toimituslaajuuteen. Palvelu kattaa niin turvallisuus- ja modernisointikartoitukset, kuin ennakko- ja korjaushuollot. Yritys huolehtii asennuksista ja käyttöönotoista projekteilleen, sekä toimittaa varaosia koneisiin ja järjestelmiin. (Makron Oy 2023b.)

3 Tausta ja tavoite

Teknologian kehittyminen ja uusien laitteiden kytkeminen internettiin lisää jatkuvasti tarvetta luoda kyberturvallisempia laitteita ja palveluja. Älylaitteet ovat integroituneet ihmisen arkipäivään niin, ettei maailmaa ilman niitä voi enää kuvitella. Älylaitteet ja palvelut sisältävät arkaluontoista tietoa ja niiden kautta voi saada pääsyn lähes kaikkeen yksittäisen ihmisen tai yrityksen hallinnoimaan tietoon, viesteihin, tileihin ja jopa maksuvälineisiin. Jatkuva tietojen kalastelu ja urkinta on jokapäiväistä ja varomaton internetin käyttäjä saattaa langeta uhriksi kyberhyökkäykselle.

Viimeisen vuosikymmenen aikana tuotantoteollisuudessa on alettu panostamaan palomuureihin ja virustorjuntaan ja mahdolliset kyberuhkat otetaan tosissaan. Vallitseva maailman tilanne venäjän ja ukrainan sodan myötä on saanut myös yritysmaailman tuomaan kyberturvallisuutta uudelle tasolle mahdollisten kyberhyökkäysten pelossa. Yrityksissä rajataan tehokkaasti, mitä työkoneelle saa asentaa. IT-osasto käyttöönottaa uudet työkoneet ja asentaa niihin ohjelmat, ettei käyttäjä vahingossa asenna haitallisia ohjelmia epäluotettavista lähteistä ja sen myötä altista yrityksen verkkoa hyökkäykselle.

Automaatioteollisuudessa ollaan siirtymässä koneiden paikallisista ohjauspulpeteista ja paneeleista verkon yli vaikuttaviin käyttöliittymiin, joita pystyy tarvittaessa käyttämään tehtaan ulkopuolelta. Laitetoimittajien sekä huolto- ja tukipalvelujen on mahdollista hallita etäyhteydellä automaatiolaitetta ohjaavaa tietokone ulkomailta. Tämä vähentää tarvetta lähettää huoltohenkilökuntaa paikan päälle, mikäli ongelman voi korjata verkon välityksellä. Käytäntö säästää työaikaa ja resursseja, mutta aiheuttaa myös tietoturvariskejä. Asiaankuulumattomat tahot voivat pyrkiä murtautumaan järjestelmään ja urkkia tietoja.

Myös Siemens on uudistanut mahdollisuuksia kommunikoida ohjelmoitavan logiikan kanssa verkon yli. Opinnäytetyön tavoitteena on käyttöönottaa logiikalle sisäänrakennettu Web-API, perustaa verkkopalvelin ja käyttöliittymä kommunikoimaan logiikkaohjelman kanssa ja ennen kaikkea varmistaa laitteiden välisten yhteyksien tietoturvallisuus.

4 PLC

4.1 Ohjelmoitava logiikka

Ohjelmoitava logiikkaohjain, lyhennettynä PLC (Programmable Logic Controller), on pieni peruselementteihinsä karsittu tietokone, jota käytetään käsittelemään automaatioprosessin ohjauksia. Näitä laitteita on saatavilla monenlaisessa muodossa eri valmistajilta. (Wayand 2020.)

PLC:lle ladataan ohjelma, jota se suorittaa vaiheittain. PLC voi olla yksittäinen laite, joka suorittaa toimintoja tai erilaisten moduulien kehikko, jota käytetään täyttämään automaatiojärjestelmän tarpeet. Kehikon muut osat voivat sisältää esimerkiksi virtalähteen, erillisiä I/O moduuleja ja ulkoisia prosessoreja. Esimerkiksi CNC-koneessa PLC voisi ohjata työkalun kohdistusta, toimielinten liikettä ja moottorien ajoja. (Wayand 2020.)

Ohjelmoitavien logiikoiden kehitys aloitettiin 1960-luvun lopulla. Ohjauslogiikkaan perustuvan laitteen asennuksen olennainen tarkoitus oli poistaa korkeat kustannukset monimutkaisten relepohjaisten ohjausjärjestelmien jatkuvasta uusimisesta ja vaihtamisesta suurille yhdysvaltalaisille autovalmistajille. Mekaaniset releet kuluvat käytössä, vievät valtavasti tilaa ja niiden johdottaminen ja asentaminen on monimutkaista. Näiden järjestelmien tilalle kehitettiin PLC. (Wayland 2020.)

PLC on käytännössä tietokone vailla hiirtä, näppäimistöä tai näyttöä. Ne ovat suunniteltu toimimaan vuosikymmeniä ympärivuorokautisesti teollisuuden ympäristöissä, joissa ne altistuvat tärinälle, pölylle, korkeille lämpötiloille ja sähkömagneettiselle häiriölle. PLC koostuu muistista, virtalähteestä ja prosessorista (CPU). Sen yhteydessä on yleensä oltava myös sisään- ja ulostulomoduulit, joihin yhdistetään anturit ja toimilaitteet. Tyypillisesti PLC toimii 24V tasavirralla. (Instblog.)

Ohjelmakierron alussa logiikka lukee sisääntulojen tilan, esimerkiksi anturilta tulevan tiedon. Ohjelmakierron lopuksi ulostulot asetetaan haluttuun tilaan ohjelman mukaisesti. Ulostulon tilaa voidaan käyttää käynnistämään esimerkiksi moottori.

4.2 Siemens S7-1500

Kuvassa 1 on Siemens S7-1500 PLC CPU 1513-PN yhdessä virtalähteen ja useamman I/O moduulin kanssa. Varsinainen CPU on korostettuna kuvassa. CPU:ssa itsessään on helppokäyttöinen hallintapaneeli, jolla pääsee tutkimaan ja muokkaamaan ohjelmoitavan logiikan ominaisuuksia. 1513–1 PN CPU:ssa on kaksi ethernet-porttia, joiden avulla logiikka kytketään ethernet-verkkoon.

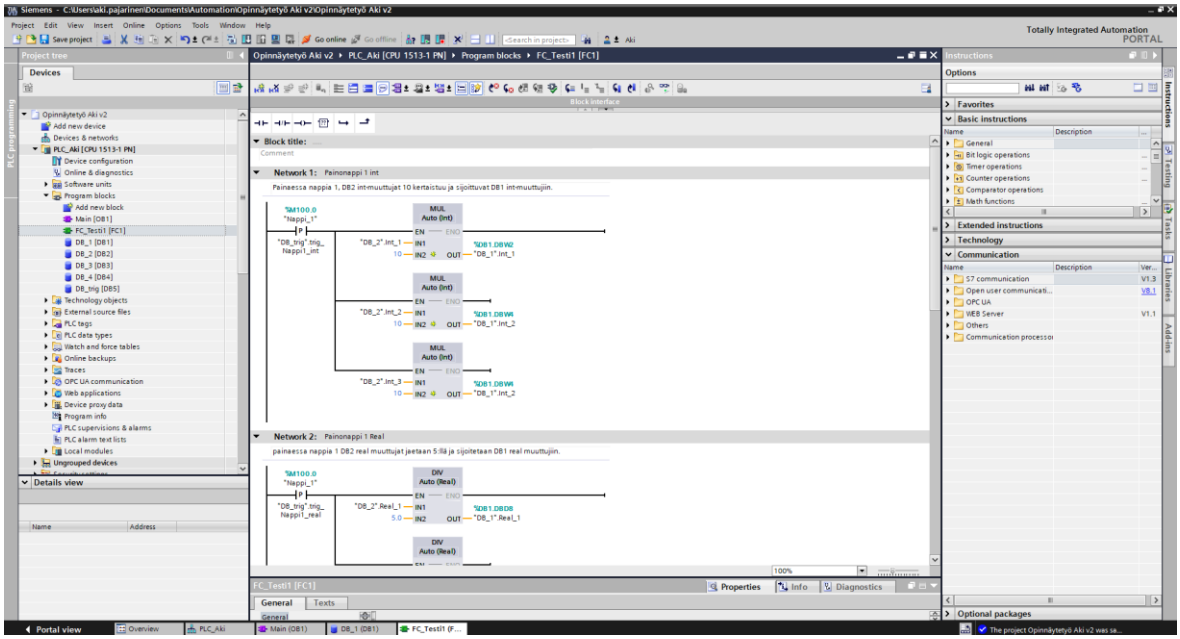
Kuva 1. Ohjelmoitava logiikka (Siemens 2022a)

Kaikki Siemensin S7-1500 sarjan logiikat tukevat uudempaa Siemens Web-API (Application Programming Interface) tekniikkaa, joka on tarpeellinen PLC:n hallintaan verkkoyhteyden yli. Siemens Web-API on Siemensin logiikalle kehittämä rajapinta, joka mahdollistaa viestien lähettämisen ja vastaanottamisen verkon välityksellä logiikalle. Siemensillä on useita erilaisia ohjelmoitavia logiikoita, mutta kaikki niistä eivät tue Web-API ominaisuuksia.

5 Siemens TIA-Portal

5.1 Kehitystyökalu

Siemens TIA-Portal (Totally Integrated Automation Portal) on kehitysympäristö. Aiemmissa sukupolvissa eri toiminnot toteutettiin kokonaan eri ohjelmistoilla. Nyt TIA-Portal yhdistää kaikki toiminnot samaan ohjelmaan (kuva 2). Kehitysympäristön avulla pystyy tekemään kaiken automaatiojärjestelmään tarvittavan ohjelmoinnin, konfiguroinnin ja käyttöönoton, sekä käyttöliittymien luonnin ja toteuttamisen.



Kuva 2. Siemens TIA-Portal V17 ohjelmointiympäristö

5.2 Ohjelmalohkot

Ohjelmalohkot ovat tapa koostaa ohjelmakoodia, joilla on tietty tapa vuorovaikuttaa toisiinsa. TIA-Portalissa ohjelmalohkoja on neljää eri tyyppiä: organisointilohko (OB), funktio (FC), funktiolohko (FB) ja datalohko (DB). Organisointilohkot toimivat rajapintana varsinaisen ohjelman ja käyttöjärjestelmän välillä ja muita ohjelmalohkoja kutsutaan organisointilohkon sisällä. Ohjelmalohkojen lukemisjärjestyksessä on tietty hierarkia ja ohjelmakierrollisesti lohkot toteutuvat kutsumisjärjestyksessä. Organisointilohkot toteutuvat laitteiston määräämässä järjestyksessä. Niitä on erilaisia eri tarkoituksiin, mutta ainoastaan Program cycle-organisointilohko on pakko olla jokaisessa logiikkaohjelmassa, sillä se sisältää varsinaisen ohjelman rakenteen. Muiden ohjelmalohkojen sisällä ohjelma toteutuu etenemisjärjestyksessä. Kehitysympäristössä virtapiirit luetaan ylhäältä alaspäin.

Funktio on lohko, jolla ei ole omaa muistia. Sitä voi kutsua saman ohjelman sisällä useissa eri instansseissa. Se voi ottaa sisääntuloarvoja ja palauttaa ulostuloarvoja suorituksen jälkeen. Funktiolohko suoritetaan samalla tavalla kuin funktio, mutta sille määritellään oma instanssikohtainen datalohko, joka toimii muistina säilyttääkseen arvoja suorituksen ulkopuolella. Datalohkot sisältävät ennalta määrätyn tietorakenteen ja niitä on instanssikohtaisia ja globaaleja. Globaaliin datalohkoon pääsee käsiksi muista ohjelmalohkoista ja instanssikohtaiseen datalohkoon pääsee käsiksi vain sitä käyttävä instanssi.

5.3 Muuttujatyypit

Logiikkaohjelmoinnissa käytetään paljon erilaisia muuttujatyyppejä. TIA-Portal tarjoaa logiikkaohjelmointia varten erilaisia muuttujatyyppejä datan käsittelyyn ja säilöntään (kuva 3). Muuttujatyypit seuraavat koko lailla yleistä käytäntöä maailmanlaajuisesti, kuitenkin pienillä eroavaisuuksilla kielestä tai ohjelmasta riippuen.

Name	Data type
Static	
Bool_1	Bool
Bool_2	Bool
Bool_3	Bool
Int_1	Int
Int_2	Int
Int_3	Int
Real_1	Real
Real_2	Real
Real_3	Real
Str_1	String
Str_2	String
Str_3	String

Kuva 3. Datalohkon sisällä olevia muuttujia

Tässä opinnäytetyössä käsiteltävät datatyypit rajataan vain muutamaan: String, Int, Word, Bool, Real ja Array. Datatyyppien toiminnasta selitetään vain perusperiaate.

String voi sisältää minkä tahansa merkkijonon – numeerisen tai aakkosnumeerisen. TIA-Portalissa String-datatyyppi voi sisältää 254 merkkiä. String-datatyyppiä käytetään yleensä jonkin tekstin varastoimiseen ja muokkaamiseen. Vaikka se voi sisältää numeroita, logiikkaohjelma ei ymmärrä String-tyyppiin varastoitua numeroa varsinaisesti numerona, vaan käsittelee sen tekstimerkkinä ja se täytyy kääntää erikseen numeromuotoon. Lukija ymmärtää numeron numerona, mutta String-tyypin sisältämä numero tulkitaan tietokoneella samalla tavalla, kuin kirjoitettuna, esimerkiksi ”nolla” tai ”yksi”. Tietokone ei tässä tapauksessa ymmärrä kyseessä olevan numero.

Int on lyhenne sanasta integer, eli kokonaisluku. Se sisältää 16 bittiä. TIA-Portalissa Int-muuttuja hyväksyy itseensä kokonaisluvut välillä -32768 ... 32767. Int on lähtökohtaisesti etumerkillinen muuttuja, tarkoittaen sitä, että se voi pitää sisällään positiivisia ja negatiivisia kokonaislukuja ja sitä käytetään matemaattisissa toiminnoissa.

Word-datatyyppi sisältää 16 bittiä. Se on etumerkitön, joten se hyväksyy vain positiivisia kokonaislukuja. TIA-Portalissa on mahdollista tarkastella Word-muuttujan yksittäisen bitin tilaa ja toisin kuin Int-muuttujaa, Word-muuttujaa käytetään loogisissa operaatioissa. TIA-Portalissa Word-muuttujan sisältämä tieto esitetään yleensä heksadesimaalina.

Bool-muuttuja sisältää vain yhden bitin, jonka arvo on joko 0 tai 1. Tätä käytetään True/False operaattorina. Esimerkiksi painonappi voi toimia tällä tavalla.

Real-datatyyppi pystyy vastaanottamaan huomattavasti Int-datatyyppiä suurempia lukuja ja se hyväksyy desimaalilukuja.

Array (kuva 4) on tärkeä datatyyppi PLC-ohjelmoinnissa. Se on menetelmä ryhmitellä saman tietotyypin muuttujia pitkäksi luetteloksi ja käyttää jokaista sisäistä elementtiä indeksin kautta. (Bee 2022.)

Array_Block

		Name	Data type	Monitor value	Comment
1		Static			
2		Array_1	Array[0..29] of Bool		Array 1
3		Array_1[0]	Bool	FALSE	Element 0
4		Array_1[1]	Bool	FALSE	Element 1
5		Array_1[2]	Bool	FALSE	Element 2
6		Array_1[3]	Bool	FALSE	Element 3
7		Array_1[4]	Bool	FALSE	Element 4
8		Array_1[5]	Bool	FALSE	Element 5
9		Array_1[6]	Bool	FALSE	Element 6
10		Array_1[7]	Bool	FALSE	Element 7
11		Array_1[8]	Bool	FALSE	Element 8
12		Array_1[9]	Bool	FALSE	Element 9

Kuva 4. Array of Bool TIA-Portalissa (Bee 2022)

5.4 Logiikan ohjelmointi kielet

Ohjelmointikielet ovat kieliä, joilla koneita ohjelmoidaan. Ohjelmointikieliä on useita, kukin eri tarpeisiin. Toiset kielet ovat tarkoitettu verkkoympäristön kehittämiseen, toiset toiminnallisen ohjelman kirjoittamiseen. Osa kielistä on dynaamisia, helposti reaaliajassa muokattavia ja osa taas omaksuu helposti valmiita ohjelmakokonaisuuksia itseensä. Jotkut ohjelmointikielet on tarkoitettu puhtaasti logiikkaohjelmointiin. Ohjelmointikielten perusta on

lähtökohtaisesti aina englanniksi, ja ne eroavat toisistaan pikemminkin rakenteeltaan ja syntaksilta, kuin varsinaiselta sanastolta, niin kuin perinteisemmät ihmisten puhekielet.

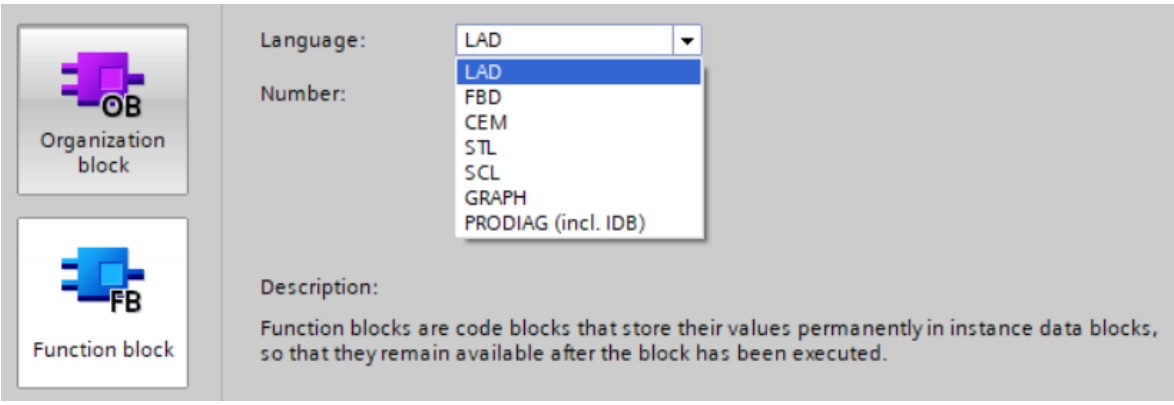
Logiikkaohjelmointikielille on olemassa Standardi IEC 61131-3, jota logiikkavalmistajat pyrkivät pääpiirteittäin seuraamaan. Jokaisella valmistajalla kuitenkin ovat omat tapansa toimia, eivätkä he täysin noudata standardia esimerkiksi ohjelmointikielten nimeämisessä. (Hanssen 2015, 200).

Standardi IEC 61131-3 esittää viittä pääkieltä käytettävän logiikoiden ohjelmoimiseen:

- Structured text – ST
- Function Block Diagram – FBD
- Ladder Diagram – LD
- Instruction List – IL
- Sequential Function Chart – SFC

Kielistä kaksi, ST ja IL, ovat tekstipohjaisia ja kolme muuta graafisia. IL näistä muistuttaa eniten alkukantaista konekieltä ja ST taas nykypäiväisempää ohjelmointikieltä, kuten Pascal tai jopa C. (Hanssen 2015, 200).

Valmistajat eivät ole millään tavalla velvoitettuja noudattamaan standardia ja tämän myötä useimmat valmistajat ovat nimenneet ohjelmointikielet omalla tavallaan. Kuvassa 5 on esitetty, miten TIA-Portal tarjoaa useaa vaihtoehtoa ohjelmointikieleksi luodessa uutta funktiolohkoa (FB).



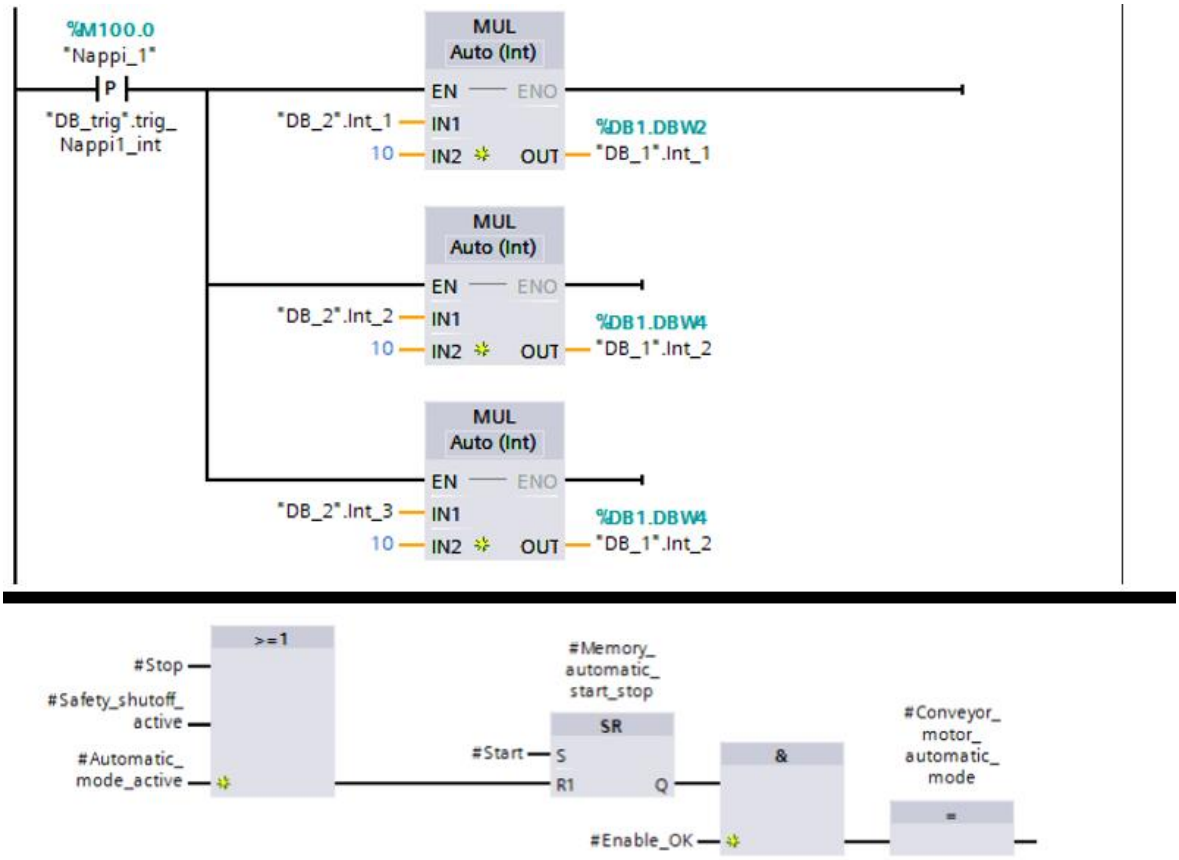
Kuva 5. Funktiolohkon ohjelmointikielet

Siemensin nimeäminä viisi pääkieltä samassa järjestyksessä kuuluvat seuraavasti:

- Structured Control Language – SCL

- Function Block Diagram – FBD
- Ladder Logic – LAD
- Statement List – STL
- GRAPH

Yleisimmin Siemens TIA-Portalissa käytetty kieli on LAD. Se muistuttaa luettavuudeltaan relekaaviota ja on helpoiten omaksuttava. Myös vianmääritys ja monitorointi on helpoin toteuttaa LAD-kielellä. FBD muistuttaa ulkonäöltään eniten LAD-kieltä (kuva 6), mutta siinä ohjelmointikomponentteja yhdistetään toisiinsa, luoden niistä toiminnallinen kokonaisuus.



Kuva 6. LAD- ja FBD-kieli TIA-Portalissa (Siemens 2018, 15)

STL ja SCL sen sijaan ovat tekstipohjaisia kieliä, jotka esitellään kuvassa 7. Ylempi STL on alkukantainen konekieli, jossa annetaan suoria komentoja ohjelmoitavalle logiikalle. SCL muistuttaa enemmän nykyaikaisia ohjelmointikieliä, joita on huomattavasti helpompi tulkita.

Network 4: STL

Comment

```
1        L     #i
2        L     #j
3        +I
4        T     #testaus_muuttuja
5
6
```

Network 5: SCL

Comment

```
1  #testaus_muuttuja := 0;
2
3  IF (#testaus_muuttuja = 1) THEN
4      FOR #i := 1 TO 100 DO
5          FOR #j := 1 TO 100 DO
6              #cosine := COS(#j);
7          END_FOR;
8      END_FOR;
9  END_IF;
```

Kuva 7. Tekstipohjaiset kielet STL ja SCL TIA-Portalissa

5.5 Web-API

V2.8 laiteohjelmistopäivityksen myötä Siemens S7-1500 logiikoiden verkkopalvelin on tarjonnut mahdollisuuden myös Web-API verkkorajapintaan. Sen avulla käyttäjä pystyy kommunikoimaan PLC:n kanssa jonkin verkkosovelluksen välityksellä. Web-API mahdollistaa verkon välityksellä tiedon kirjoittamisen ja lukemisen PLC:lle. (KB controls 2022.)

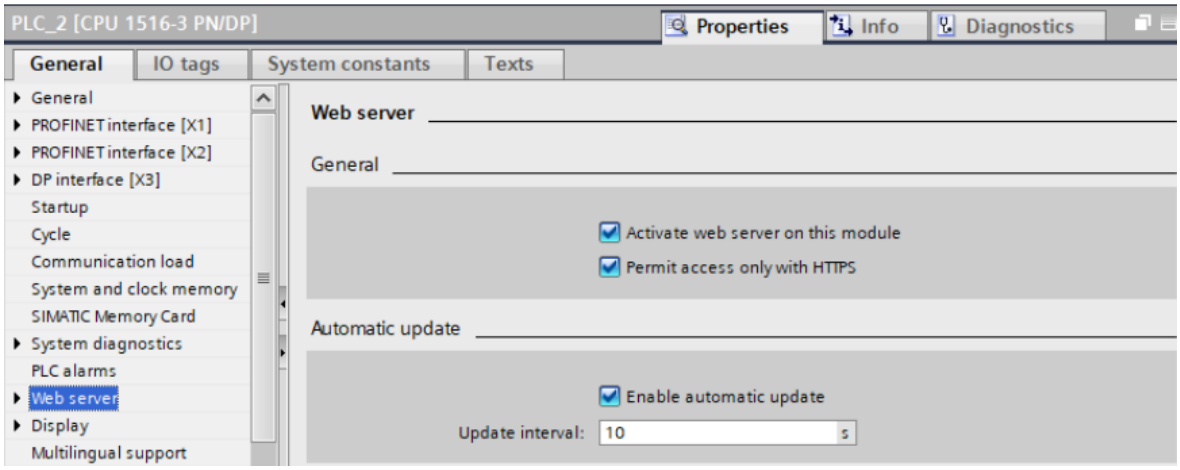
API on rajapinta, joka määrittää säännöt ja toimintatavat yhteyden muodostamiselle laitteiden välillä. Web-API ei ole teknologia vaan konsepti. Samalla tavalla kuin jokin käyttöliittymä yhdistää ihmisen ja tietokoneen, Web-API sallii kahden laitteen yhdistää toisiinsa käyttäen HTTP protokollaa. (Tutorialsteacher 2023.)

Suurin etu uudessa Web-API uudistuksessa on se, että se käyttää JSON RPC 2.0 protokollaa tiedon jäsentelyyn. JSON RPC 2.0 on standardoitu tapa pyytää etälaitteita suorittamaan menettelytapoja ja noutamaan tuloksia. Tämän ansiosta tavallisen verkkokehittäjän on mahdollista luoda verkkosivu, joka kykenee kommunikoimaan PLC:n kanssa ilman syvempää tietoa varsinaisesta logiikkaohjelmoimisesta tai tietoa PLC:n toiminnasta. (KB controls 2022.)

Siemens on aikaisemminkin tarjonnut mahdollisuuden käyttäjän luomille verkkosivuille, mutta tiedonsiirto on tapahtunut raskaammalla HTML-tiedostoformaatilla ja vaatinut logiikkaohjelmalta ohjelmakierrollisen WWW (SFC 99) komennon synkronoidakseen logiikan datan verkkosivun kanssa. Käyttäjän luomat verkkosivut piti myös määrittää aikaisemmin PLC:n asetuksiin, jotta logiikka osasi kohdistaa tiedon oikeaan osoitteeseen.

5.6 Konfiguraatio

Jotta Web-API ominaisuuksia voi käyttää, on logiikasta kytkettävä ensin verkkopalvelin päälle (kuva 8). Se on oletuksena poistettu käytöstä logiikan perusasetuksissa.



Kuva 8. Verkkopalvelimen aktivoiminen TIA-Portalissa (Siemens 2022b)

Verkkopalvelin mahdollistaa suorittimen hallinnan ja valvonnan verkon yli. Arvioinnit, diagnostiikka ja modifikaatiot ovat siten mahdollisia pitkienkin matkojen päästä. Tiedon seuranta ja tarkastelu on mahdollista ilman TIA-Portal ohjelmointiympäristöä, käyttäen verkkoselainta internetin välityksellä. Turvallisuudesta tulee kuitenkin huolehtia, käyttäen esimerkiksi palomuuria yhteyden turvaamiseksi. (Siemens 2022b.)

Asetuksista täytyy määrittää myös yhteyden salliminen vain HTTPS-protokollaa käyttäen. Verkkopalvelimeen ei pysty yhdistämään suojaamattomalla HTTP-yhteydellä. Tämän lisäksi PLC:lle on oltava asetettuna IP-osoite ja aika- ja päivämääräasetukset on oltava ajan tasalla. Lisäksi käytettävään verkkoselaimeen on oltava asennettuna PLC:n tarjoama voimassa oleva sertifikaatti. (Siemens 2022b.)

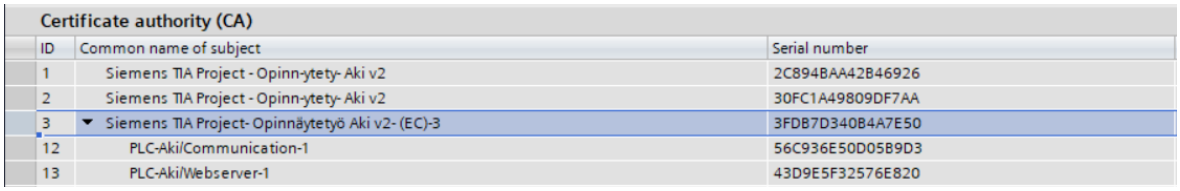
5.7 Sertifikaatti

Verkkopalvelimeen yhdistäminen vaatii HTTPS-yhteyden ja se puolestaan edellyttää voimassa olevan sertifikaatin. Siemensin verkkopalvelin vaatii sertifikaatin, jonka on myöntänyt luotettava varmenteen myöntäjä (CA-signed certificate). Esimerkiksi Siemens itse voi toimia luotettavana varmenteen myöntäjänä ja tällaisen sertifikaatin voi luoda suoraan TIA-Portaalin sisällä. Sekä palvelimelle, että yhteyttä ottavalle verkkoselaimelle on oltava asennettuna sama sertifikaatti. (Siemens 2022b.)

SSL (Secure Sockets Layer) on salaukseen perustuva internet suojausprotokolla. Se on kehitetty yhteyden yksityisyyden, autentikoinnin ja tietojen eheyden varmistamiseen internet viestinnässä. SSL on nykyisen TLS-salauksen (Transport Layer Security) edeltäjä. SSL/TLS:ää käyttävän verkkosivuston URL-osoitteessa on HTTPS eikä HTTP. (Cloudflare 2023.)

Yksityisyyden takaamiseksi SSL, tai tänä päivänä TLS, salaa verkossa siirrettävät tiedot. Tämä tarkoittaa sitä, että ulkopuolisen tahon siepatessa viestin, joka on salattu TLS-protokollalla, sieppaaja näkee vain sekavan merkkiyhdistelmän, jonka salauksen purkaminen on lähes mahdotonta.

Jotta salattua yhteyttä varten CA-sertifikaatin voi luoda, on TIA-Portalin sisällä ensin suojattava käsiteltävä TIA-projekti asettamalla sille käyttäjätunnus ja salasana. Lisäksi projektille tulee asettaa voimaan yleiset turvallisuuskäytännöt. Tämän jälkeen tarvittavat sertifikaatit voi luoda TIA-Portalin Certificate manager ominaisuudella (kuva 9). (Siemens 2022b.)

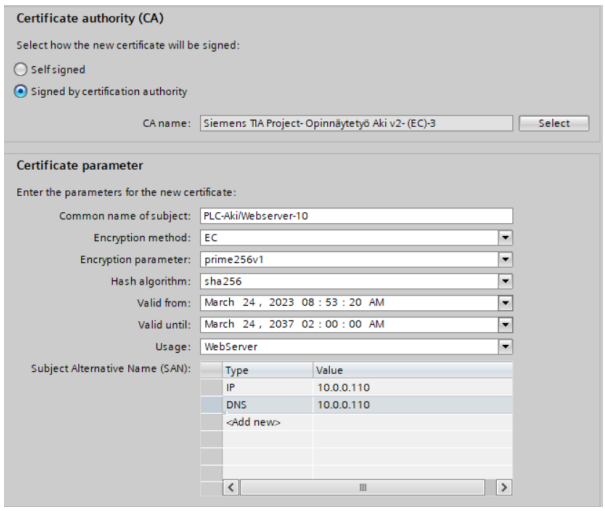


Certificate authority (CA)

ID	Common name of subject	Serial number
1	Siemens TIA Project - Opinn-ytety- Aki v2	2C894BAA42B46926
2	Siemens TIA Project - Opinn-ytety- Aki v2	30FC1A49809DF7AA
3	Siemens TIA Project- Opinnäytetyö Aki v2- (EC)-3	3FDB7D340B4A7E50
12	PLC-Aki/Communication-1	56C936E50D05B9D3
13	PLC-Aki/Webserver-1	43D9E5F32576E820

Kuva 9. Certificate manager (Siemens 2022b)

Lisäksi PLC vaatii kaksi muuta sertifikaattia sisäisesti. Logiikkaohjelmalla on oltava voimassa Webserver-certificate (Verkkopalvelin sertifikaatti), jonka voi luoda TIA-Portal kehitysympäristössä (kuva 10) tai se luodaan automaattisesti ottaessa yleiset turvallisuuskäytännöt voimaan. Ilman voimassa olevaa verkkopalvelinsertifikaattia logiikkaohjelmaa ei pysty lataamaan logiikalle ollenkaan, mikäli Web-server ominaisuus on otettuna käyttöön logiikassa.

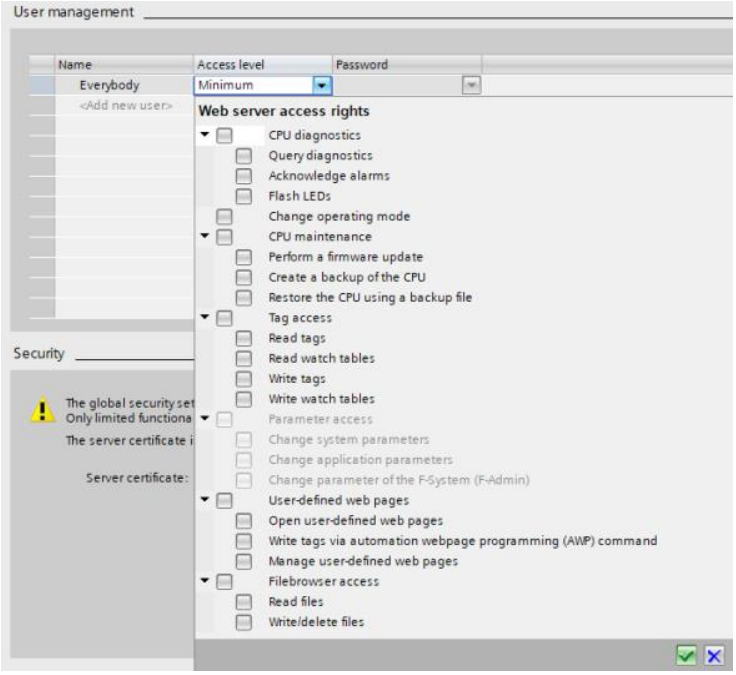


Kuva 10. Verkkopalvelinsertifikaatin luominen TIA-Portalissa

Toinen vaadituista on Communication-sertifikaatti, joka vaaditaan TIA-Portalin ja HMI:n väliseen turvalliseen kommunikointiin. Se luodaan samaan tapaan kuin verkkopalvelin sertifikaatti. Nämä kaksi sertifikaattia tarvitsee olla vain PLC:n sisällä, eikä niitä erikseen vaadita palvelin- tai asiakaskoneelta käyttäessä Web-API rajapintaa. Mikäli logiikkaohjelmassa ei käytettäisi yleisiä turvallisuuskäytäntöä, CA-sertifikaatin sijasta voisi käyttää verkkopalvelin sertifikaattia tunnistautumiseen.

Pääsyä PLC:n tietoihin on suojattava myös muilla keinoin, kuten käyttäjätunnusten luomisella ja käyttäjän oikeuksien rajaamisella (kuva 11). TIA-projektiin on mahdollista luoda eri käyttäjiä eri oikeuksilla, kuten luku- tai kirjoitusoikeudet. (Siemens 2022b.)

Oletuksena verkkopalvelimeen kirjaudutaan käyttäjällä ”Everybody”. Tällä käyttäjällä on minimaaliset oikeudet verkkopalvelimen käyttöön, sekä datan käsittelyyn. Kun TIA-projektin asetukset ovat kunnossa, tulee projekti ladata PLC:lle. Tämän jälkeen PLC:n puolesta Web-API on valmis toimimaan. (Siemens 2022b.)



Kuva 11. Käyttäjien oikeuksien määrittäminen (Siemens 2022b)

5.8 PUT/GET

Siemensin logiikat tarjoavat suoran tavan vaihtaa tietoa kahden samassa verkossa sijaitsevan logiikan kesken. PUT/GET-kommunikointi käyttää sisäänrakennettua Profinet-rajapintaa tai teollisuuteen tarkoitettua Ethernet-rajapintaa tiedonsiirtoon logiikassa. Kommunikointimetodia käyttääkseen on logiikan asetuksista sallittava PUT/GET-kommunikointi.

PUT/GET-kommunikointi vaatii datalohkolta optimoimattoman muistialueen käytön, jolloin muuttujiin pystyy viittaamaan absoluuttisella osoituksella. Karkeasti optimoimattomassa datalohkossa muistialueilla olevat tiedot sijaitsevat niille osoitetussa kentässä, siinä missä optimoitu datalohko pakkaa sen sisältämän tiedon mahdollisimman tiiviisti, käyttäen muistialueen tilaa vähemmän.

PUT/GET-kommunikoinnin käyttö kahden logiikan välillä vaatii toisen logiikan ohjelmalta Put- ja Get- Operaattorien käyttöä. Tiedonsiirron voi hoitaa yksipuolisesti toisen logiikan puolesta. PUT/GET-kommunikointia on mahdollista hyödyntää myös muilla tavoin, kuten Ethernet-rajapintaa käyttäen suoralla sokettiyhteydellä. PC:lle kirjoitettu ohjelma voi simuloida toista logiikkaa ja hyödyntää sokettiyhteyttä. Tällöin logiikkaohjelmalta ei välttämättä vaadita operaattorien käyttöä ollenkaan.

6 Ohjelmointikielet

6.1 Ohjelmointikielet yleisesti

Ohjelmointikielen avulla voidaan tietokoneelle antaa ohjeita kielellä, jota tietokone ymmärtää. On olemassa monia eri ohjelmointikieliä, samaan tapaan kuin on olemassa puhekieliä eri kansojen välillä. Pääpiirteittäin ohjelmointikielten tarkoitus on kuitenkin pystyä kommunikoimaan tietokoneen kanssa. (Computer science 2023.)

Tietokone ymmärtää kieltä, jota kutsutaan binääriksi. Kieli koostuu täysin numeroista 1 ja 0. Ihmisen on vaikea tulkita binääriä ja siksi avuksi tarvitaan ohjelmointikieliä. Ohjelmointikieliä kirjoitetaan ihmiselle helpommin tulkittavassa muodossa ja käännetään binääriksi tietokonetta varten. Kullakin ohjelmointikielellä on omat erityispiirteensä, vaikka yleisesti kielet ovat samankaltaisia. (Computer science 2023.)

Ohjelmointikielten avulla tietokoneet voivat käsitellä nopeasti ja tehokkaasti suuria ja monimutkaisia tietoja. Esimerkiksi jos ihmiselle annetaan luettelo satunnaisista numeroista, jotka vaihtelevat yhdestä kymmeneen tuhanteen ja häntä pyydetään järjestämään numerot nousevaan järjestykseen, voidaan olettaa tämän vievän huomattavan paljon aikaa ja sisältävän jonkin määrän virheitä. Tietokone pystyy tähän tehtävään hetkessä. (Computer science 2023.)

6.2 JavaScript

JavaScript on asiakaspuolen ohjelmointikieli, joka toimii asiakasselaimen sisällä ja käsittelee komentoja selaimessa palvelimen sijaan. Se sijoitetaan yleensä HTML- tai ASP- tiedostoon. Nimestään huolimatta se ei liity Javaan. (Computer science 2023.)

JavaScriptiä (kuva 12) käytetään ensisijaisesti Web-kehityksessä erilaisten verkkosivujen elementtien luomiseen ja manipuloimiseen. Elementtejä voivat olla esimerkiksi vieritysominaisuudet, kellonajan ja päivämäärän tulostaminen, kalenterin luominen ja muut tehtävät, jotka eivät ole mahdollisia pelkästään HTML-kieltä käyttäen. Sitä voidaan käyttää myös pelien ja API:en luomiseen. (Computer science 2023.)

```
144 function readdbvariable() {
145
146     let dbval = document.getElementById("readdbvalue")
147     fetch(TARGET_IP, {
148         method: "POST",
149         headers: {
150             "Content-type": "application/json",
151             "X-Auth-Token": AUTH_TOKEN
152         },
153         body: JSON.stringify({
154             "id": messageId++,
155             "jsonrpc": "2.0",
156             "method": "PlcProgram.Read",
157             "params": {
158                 "var": "\"DB_" + dbnum_global + "\"." + dbvartype + "_" + dbvarnum_global,
159             }
160         })
161     })
162         .then(response => response.json())
163         .then((data) => {
164             dbval.innerHTML = data.result
165         })
166         .catch(e => console.error(e))
167 }
```

Kuva 12. Siemensin Read-metodi JavaScript-kielellä

6.3 HTML

HyperText Markup Language (HTML) on yksinkertainen merkintäkieli, jota käytetään luomaan hypertekstiasiakirjoja, jotka ovat siirrettävissä alustasta toiseen. HTML- dokumentit ovat SGML-dokumentteja (Stantard Generalized Markup Language), joissa on yleinen semantiikka ja jotka sopivat esittämään tietoa useista eri sovelluksista. (Whatwg 2023.)

HTML-dokumentit ovat tekstitiedostoja, jotka yleensä jaetaan verkkoselaimeen verkkopalvelimen tai paikallisen tallennustilan kautta ja tulkitaan sen jälkeen multimediaverkkosivuksi. Web-sivut rakennetaan käyttämällä HTML-elementtejä. Elementtejä on eri muotoisia ja ne mahdollistavat verkkosivuille sisällön luomisen, kaikkea perustyylisistä tekstikappaleista lomakkeisiin, painikkeisiin ja kuviin. (Futurelearn 2023.)

HTML-dokumentit koostuvat elementtien ja tekstin puusta. Jokainen elementti on merkitty lähdekoodissa aloitustunnisteella, kuten <body> ja osion sulkevalla lopputunnisteella </body>. Elementtien tulee aina olla yhtenäisiä tunnisteiden sisällä, menemättä lähdekoodissa päällekkäin missään vaiheessa. (Whatwg 2023.)

Useimmiten HTML-dokumentti koostuu head- ja body-osioista. Head tyypillisesti sisältää verkkosivun otsikon ja ulkoa kutsuttavat tiedostot ja dokumentit. Body sisältää verkkosivulla esiintyvän asian. Varsinainen asia on se, mitä verkkosivu sisältää avattaessa sivu selaimella. Kuvassa 13 esitellään HTML-dokumentin rakennetta. Jokainen HTML-dokumentti tulee alkaa <!DOCTYPE > julistuksella, jonka sisällä kerrotaan selaimelle, minkä tyyppinen dokumentti on.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Kuva 13. Tyypillisen HTML-tiedoston rakenne. (W3Schools 2023a)

6.4 CSS

CSS (Cascading Style Sheets) on tyylitaulukkokieli, jota käytetään kuvaamaan HTML- tai XML- kielellä kirjoitettujen asiakirjojen esitystapaa. CSS kuvaa, kuinka elementit tulee tulkita näytöllä, paperilla, puheessa tai muussa mediassa. CSS on yksi avoimen webin ydinkielistä ja se on standardoitu W3C-spesifikaatioiden mukaisesti kaikissa Web-selaimissa. (MDN Web Docs 2023a.)

Käytännössä CSS-dokumentti sisältää muotoiluohjeet sisällön esitystavalle verkkosivuilla. CSS-tyylisivuja käytetään verkkosivujen tyyliin ja asettelemiseen esimerkiksi muuttamaan sisällön fonttia, väriä, kokoa ja väliä tai jakamaan se useisiin sarakkeisiin, lisäämään animaatioita tai muita koristeellisia ominaisuuksia. (MDN Web Docs 2023a.)

CSS-tyylisivuja (kuva 14) voidaan käyttää lähdekoodissa kolmella eri tavalla: elementin sisällä, dokumentin sisäisesti tai kokonaan ulkoisesti omana tiedostonaan. Elementin sisällä muotoilu liitetään suoraan muotoiltavaan kohteeseen tekstissä. Dokumentin sisällä voidaan määrittää koko yksittäiselle verkkosivulle tietty tyyli tietynlaisille elementeille, kuten otsikoille tai tekstikappaleille. Ulkoisesti käyttäessä luodaan erillinen CSS-tiedosto, joka sisältää tarvittavat tyyli- ja muotoiluohjeet ja tähän tiedostoon viitataan lähdekoodissa. Tämä mahdollistaa helposti saman tyylin tuotavaksi useammille eri verkkosivuille. (W3schools 2023b.)

```
149  /*Bottom right area----------------------------------------*/
150  .bottom-right {
151      width: 35%;
152      height: 300px;
153
154      color: rgb(240, 234, 214);
155      font-weight: 800;
156      font-size: large;
157      font-family: Helvetica;
158
159      padding: 5px;
160
161      display: flex;
162      flex-direction: row;
163      justify-content: center;
164      align-items: center;
165
166      background-color: rgb(40, 40, 40);
167      border: solid 2px rgb(60, 60, 60);
168      border-radius: 25px;
169  }
170
171  /* Bottom Right section, Left box */
172  .bright-L {
173      height: 100%;
174
175      display: flex;
176      flex-direction: column;
177      justify-content: center;
178
179      text-align: center;
180  }
181
182  .bright-L-1 {
183      width: 300px;
184
185      display: flex;
186      flex-direction: row;
187      justify-content: center;
188      align-items: center;
189  }
```

Kuva 14. CSS tyylisivu erillisessä tiedostossa

Ulkoasun määrittäminen on tärkeää verkkosivujen luettavuuden ja toiminnan kannalta. CSS-dokumentti sisältää usein määrityksen esimerkiksi sille, miten verkkosivu muuntautuu tietokoneen selaimelta mobiililaitteen selaimelle. Ilman näitä määrityksiä verkkosivut ovat käytännössä käyttökelvottomat.

6.5 JavaScript JSON ja AJAX

6.5.1 JSON

JSON (JavaScript Object Notation) ei ole oma ohjelmointikielensä, vaan kevyt tiedostonsiirtomuoto, joka on tarkoitettu datan varastoimiseen ja helppoon siirtämiseen. Se on helposti luettavaa ja kirjoitettavaa ja tietokoneelle helppo jäsentää ja luoda. (JSON.)

JSON on tekstimuoto, joka on täysin kieliriippumaton, mutta käyttää käytäntöjä, jotka ovat tuttuja C-kieliperheen ohjelmoijille, kuten C, C++, C#, Java, JavaScript, Perl ja Python. Nämä ominaisuudet tekevät JSONista ihanteellisen tiedonsiirtokielen. (JSON.)

JSON on formaatti, jota käytetään eri ohjelmointikielien sisällä tiedonsiirtoa varten. JSONia käytetään usein datan lähettämiseen palvelimelta verkkosivulle. JSON-syntaksimuoto perustuu kahteen päärakenteeseen: objektit ja taulukot (kuva 15). Objekti on kokoelma nimi-

ja arvopareja. Taulukko on järjestetty arvoluettelo. (Elgabry 2016.) Riippuen ohjelmointiympäristöstä ja käytetystä kielestä, taulukko voi myös sisältää eri tyypin muuttujia.

```
{
        "id": 4911,
        "firstName":"John",
        "lastName":"Walter",
        "isStudent": true
};
```

```
[
     "Belgium", "Brazil", "Cameroon", "Canada", "Chile", "Denmark"
]
```

Kuva 15. Objekti ja taulukko (Elgabry 2016.)

6.5.2 AJAX

AJAX (Asynchronous JavaScript and XML) ei ole tekniikka sinänsä, vaan lähestymistapa useiden olemassa olevien teknologioiden yhteiskäyttöön, kuten HTML tai XHTML, CSS, JavaScript, DOM, XML ja tärkeimpänä XMLHttpRequest-objekti. Nämä tekniikat yhdistämällä AJAX-mallissa verkkosovellukset pystyvät tekemään nopeita, inkrementaalisia päivityksiä käyttöliittymään lataamatta koko verkkosivua uudelleen. Tämä tekee sovelluksesta nopeamman ja paremmin käyttäjän pyyntöihin vastaavan. (MDN Web Docs 2023b.)

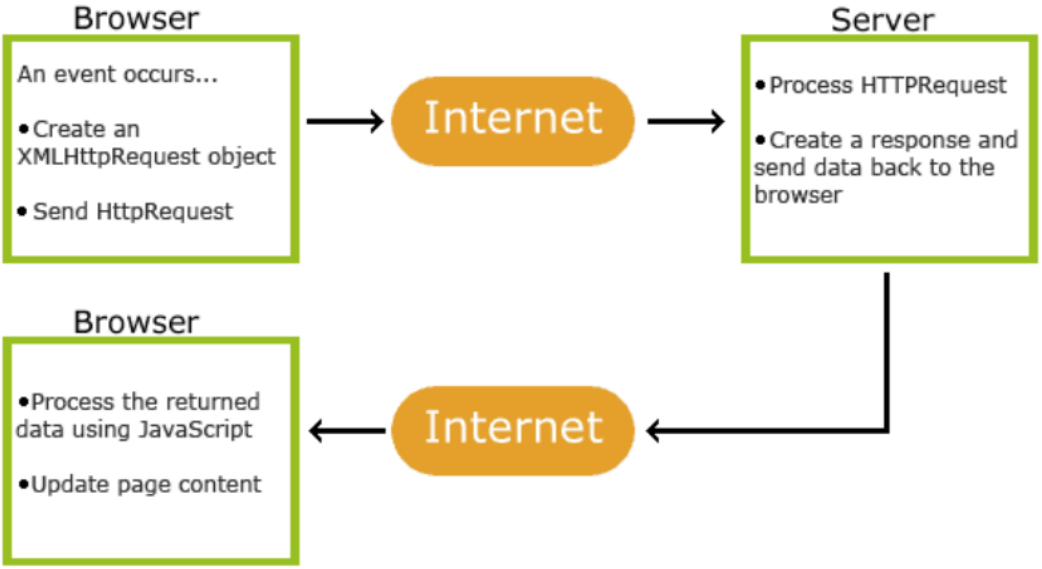
XMLHttpRequest on objekti JavaScriptissa, mitä käytetään kanssakäymiseen verkkopalvelimen kanssa. Sen avulla voit vastaanottaa dataa verkkosivulta joutumatta päivittämään kokonaan käsiteltävää verkkosivua. Tämä mahdollistaa käsiteltävän verkkosivun osittaisen päivittämisen, häiritsemättä ja katkaisematta käyttäjän toimia sivuilla. (MDN Web Docs 2023c.)

Kuvassa 16 luodaan yksinkertainen painonappi, joka suorittaa XMLHttpRequest-metodin, pyytäen vastausta paikallisesta test.HTML tiedostosta. Mikäli yhteys muodostuu, ohjelma palauttaa tiedostosta haetun vastauksen ponnahdus ilmoituksena käyttäjälle.

```
<button id="ajaxButton" type="button">Make a request</button>

<script>
  (() => {
    let httpRequest;
    document
      .getElementById("ajaxButton")
      .addEventListener("click", makeRequest);

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

      if (!httpRequest) {
        alert("Giving up :( Cannot create an XMLHTTP instance");
        return false;
      }
      httpRequest.onreadystatechange = alertContents;
      httpRequest.open("GET", "test.html");
      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>
```

Kuva 16. Yksinkertaisen XMLHttprequest objektin luominen ja käyttö (MDN Web Docs 2023b.)

Kuvassa 17 on esitettynä Pipeline-kaavio XMLHttpRequest-objektin toiminnasta. Käyttäjä luo toiminnallaan selaimessa tapahtuman, jonka takia luodaan ja lähetetään pyyntö internetin välityksellä palvelimelle. Palvelin vastaanottaa, käsittelee ja vastaa pyyntöön halutulla tavalla. Tämä voi tarkoittaa esimerkiksi jonkin kuvan avaamista verkkoselaimella. Kun pyyntö on käsitelty onnistuneesti, palvelin palauttaa selaimelle halutun tiedon.



Kuva 17. XMLHttpRequest toiminto kaaviona (W3schools 2023c)

6.6 PHP

PHP (PHP: Hypertext Preprocessor) on laajalti käytetty avoimen lähdekoodin yleiskäyttöinen komentosarjakieli, joka soveltuu erityisesti verkkokehitykseen ja voidaan upottaa HTML:ään. Useiden komentojen sijaan HTML-tulostukseen PHP-Sivut sisältävät HTML:ään upotettua PHP-koodia, jolla on jokin toiminnallinen tarkoitus. Kuvassa 18 HTML:n sisällä tulostetaan teksti ”Hi, I’m a PHP script!” käyttäen upotettua PHP-koodia. (PHP 2023.)

```
<!DOCTYPE html>
<html>
<head>
<title>Example</title>
</head>
<body>

<?php
            echo "Hi, I'm a PHP script!";
        ?>

</body>
</html>
```

Kuva 18. PHP-koodia HTML:n sisällä (PHP 2023)

PHP-koodilla on aloitustunniste ”<?php” ja lopputunniste ”?>”. Näillä tunnisteilla voidaan erotella HTML:n sisällä PHP-koodi, ikään kuin siirtyen koodin sisällä hetkeksi PHP-tilaan ja pois. Kuvassa 19 koodi on kirjoitettu HTML-kielellä ja sen sisällä osittain PHP-kielellä.

```
<h1 class="akiHeading">This is is plain HTML with CSS styling applied</h1>
<p><?php  echo "This is in PHP, but printed out to the web page." ?></p>
<p><?php  "This PHP will not show on the page"; ?></p>

<p> This text is in HTML. The following contains data in PHP</p>
<p class="akidata"> <?php
echo "DATA INT1:" . $data_int1 . "<br>";
echo "DATA INT2:" . $data_int2 . "<br>";
echo "DATA INT3:" . $data_int3 . "<br>"; ?>
</p>
```

Kuva 19. PHP-koodia käytetty HTML tiedoston sisällä

PHP:n erottaa esimerkiksi asiakaspuolen JavaScriptistä se, että koodi suoritetaan palvelimella, joka tuottaa asiakkaalle lähetettävän HTML:n. Asiakas saa tulokset kyseisen PHP-koodin suorituksesta, mutta ei tiedä taustalla olevaa koodia. HTML tiedosto voidaan määrittää kokonaan PHP:llä, jolloin käyttäjät eivät tiedä, minkälainen koodi ohjelman taustalla pyörii. (PHP 2023.) Lopputuotteessa HTML on nähtävissä, mutta PHP-osio ei, ellei PHP:tä tulosteta ”echo” komennolla (kuva 20).

Kuva 20. Koodin käyttäytyminen verkkosivulla

Oleellista PHP-koodin käytössä on sen näkymättömyys verkkosivun lähdekoodissa. Koska PHP suoritetaan palvelinpuolella, se ei palaa asiakkaan puolelle nähtäväksi. Kuvassa 21 on ote asiakaspuolella selaimessa suoritettavasta lähdekoodista, jossa palvelinpuolen PHP ei näy. Asiakaspuolen lähdekoodiin pääsee käsiksi kuka tahansa käyttämällä selaimen kehitystyökalua, eikä koodin luojan puolesta ole usein haluttua näyttää ohjelman koodia ulkopuolelle.

```
<h1 class="akiHeading">This is is plain HTML with CSS styling applied</h1>
<p>This is in PHP, but printed out to the web page.</p>
<p></p>
<p> This text is in HTML. The following contains data in PHP</p>
<p class="akidata">…</p>
```

Kuva 21. Ote lähdekoodista asiakkaan puolella

6.7 C#

C# (C Sharp) on moderni tyyppiturvallinen olio-ohjelmointikieli. C#:n avulla kehittäjät voivat rakentaa turvallisia ja vankkarakenteisia sovelluksia, jotka toimivat .NET ympäristössä.

C#:n juuret ovat C-kieliperheessä ja kieli on tuttua C-, C++- ja JavaScript-ohjelmoijille. (Microsoft 2023.)

C# on yksi suosituimpia ohjelmointikieliä. Se on helppokäyttöinen, yksinkertainen ja sillä on valtava tukiyhteisö. Kieli on oliosuuntautunut, mikä tekee siitä selkeän, muokattavan ja helposti uudelleenkäytettävän. C# koodiin on helppo tuoda valmiita koodikomponentteja valmiista kirjastoista. (W3schools 2023d.)

7 Verkkosivujen luomiseen käytettävät työkalut

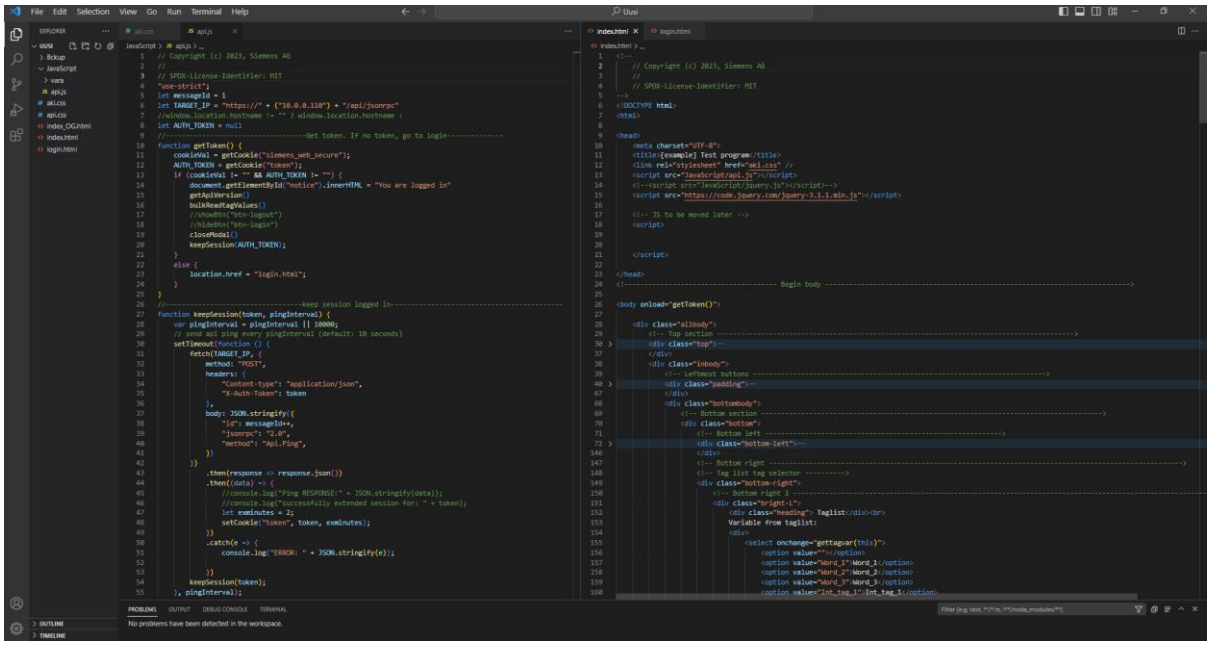
7.1 Verkkosivut

Verkkosivusto on kokoelma julkisesti saatavilla olevia, toisiinsa linkitettyjä Web-sivuja, joilla on yksi verkkotunnus. Yksilö, ryhmä, yritys tai organisaatio voivat luoda ja ylläpitää verkkosivustoja useisiin eri tarkoituksiin. Yhdessä kaikki saatavilla olevat verkkosivustot muodostavan World Wide Webin (www). Yleensä verkkosivu on yksittäinen HTML dokumentti, joka on liitetty muihin samanlaisiin hyperlinkkien avulla. (Rouse 2020.)

Verkkosivut tyypillisesti tarvitsevat palvelimen, joka niitä isännöi, sekä kehitysympäristön, jolla niitä luodaan ja ylläpidetään.

7.2 VS Code ohjelmointiympäristö

VS Code (Visual Studio Code) on ilmainen ja kevyt, mutta tehokas lähdekoodieditori, joka on työpöydällä toimiva ohjelma (kuva 22). Se on saatavilla Windowsille, Linuxille ja macOS:lle. Siinä on sisäänrakennettu tuki JavaScriptille, TypeScriptille ja Node.js:lle. VS Codessa on rikas laajennuksien ekosysteemi muille ohjelmointikielille, kuten C++, C#, Python, PHP ja .NET. (VS Code 2023.)



Kuva 22. Visual Studio Code lähdekoodieditori

VS Code tekee koodista helposti luettavaa. Sisäänrakennetuilla työkaluilla voi formatoida koodin eri osiot selkeään automaattisesti sisennettyyn muotoon yhdellä pikanäppäimellä.

Lisäksi editori värjää koodin automaattisesti eri toimintojen mukaan parantaen sen luettavuutta. VS Code on yksi maailman käytetyimmistä koodieditoreista, joten siihen on saatavilla paljon ladattavia lisäosia eri toimintojen helpottamiseen.

Lähdekoodieditorin avulla voi luoda verkkosivun kokonaisuudessaan. Luomalla yhden tai useamman dokumentin, esimerkiksi HTML-dokumentin toteuttamaan verkkosivun ulkoasua, JavaScript-dokumentti luomaan sisältöä verkkosivulle ja CSS-dokumentti muotoilemaan ja tyylittelemään verkkosivun ulkoasua.

VS Code:lle on ladattavissa valmiita kirjastoja, joiden toimintoja kutsumalla koodissa voit helposti toteuttaa tahdotut toiminnot, kuten POST-request palvelimen ja asiakaan välillä.

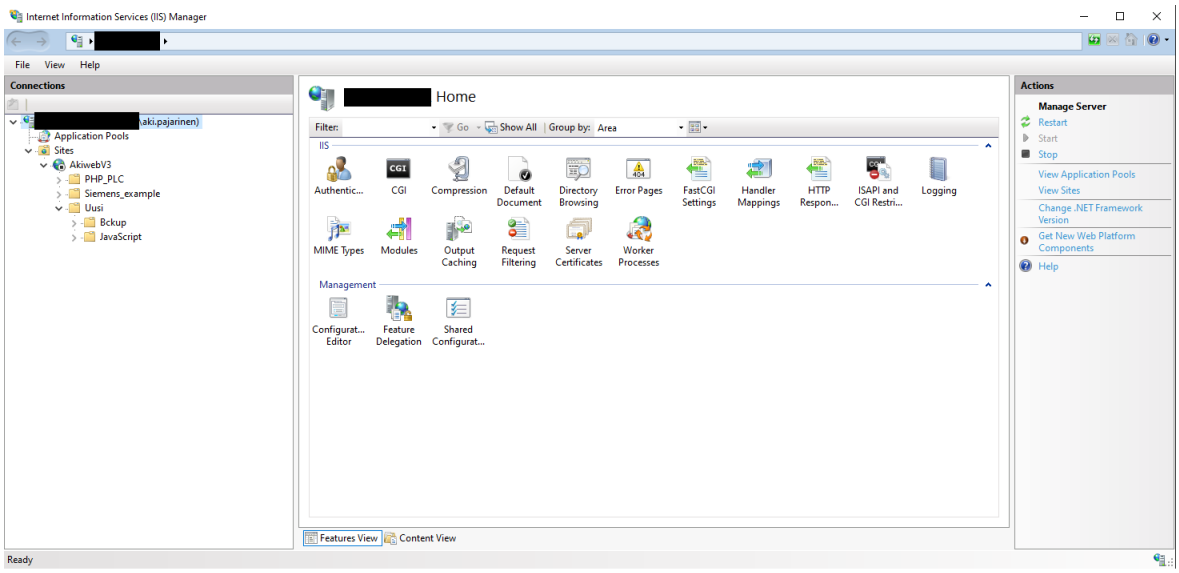
7.3 Microsoft IIS

Microsoft IIS (Internet Information Services) on Microsoftin verkkopalvelin, joka toimii Windows-järjestelmissä. Se tehtävä on isännöidä ja palvella HTML-sivuja ja tiedostoja. (Rosencrance 2019.)

IIS-verkkopalvelin hyväksyy pyynnöt etäasiakastietokoneilta ja palauttaa asianmukaisen vastauksen. Tämän perustoiminnon avulla web-palvelimet voivat jakaa ja toimittaa tietoja lähiverkossa (LAN), kuten yrityksen intranetissä tai suuralueverkoissa (WAN), kuten internetissä. (Rosencrance 2019.)

Web-palvelin toimittaa tietoja käyttäjille useissa muodoissa, kuten staattisten HTML:llä koodattujen verkkosivujen kautta tiedostoja lähettämällä ja vastaanottamalla. Tiedostot voivat olla esimerkiksi tekstitiedostoja, kuvatiedostoja tai paljon muuta. (Rosencrance 2019.)

IIS (kuva 23) voi toimia palvelimena isännöimään esimerkiksi verkkoselainpohjaista käyttöliittymää ohjelmoitavaa logiikkaa varten. Ohjelman kautta määritellään verkkosivun vaatimat protokollat, asetukset ja mahdolliset vaadittavat sertifikaatit.



Kuva 23. Microsoft IIS

7.4 Kyberturvallisuus

Kyberturvallisuus on käytäntö suojella järjestelmiä, verkkoja ja ohjelmia digitaalisilta hyökkäyksiltä. Kyberhyökkäysten tarkoituksena on päästä käsiksi, muuttaa tai tuhota arkaluonteisia tietoja, kiristää rahaa käyttäjiltä lunnasohjelmien avulla tai keskeyttää normaalit liiketoimintaprosessit. (F-secure 2023.)

Maailmanlaajuisesti palvelut ja toiminnot kytkeytyvät jatkuvasti vahvemmin verkkoon. Lähes kaikki kriittiset toiminnot ovat jollain tavalla verkossa, kuten rahaliikenne, henkilötiedot ja tunnistautuminen, yritysten tiedot ja toiminnot sekä ihmisten välinen kommunikointi. Tämä kutsuu puoleensa kyberrikollisia, jotka yrittävät hyötyä varastamalla yksityistä dataa tai huijaamalla itselleen joko omaisuutta tai etua horjuttamalla kilpailijan järjestelmää tai mainetta. (F-secure 2023.)

Kyberhyökkäyksiä on luokiteltu erityyppisiksi, kuten kiristysohjelmat, vakoiluohjelmat, troijalaiset, tietojenkalastelu, Man-in-the-middle- tai jopa palvelunestohyökkäykset. Kuluttaja pystyy suojautumaan näiltä hyökkäyksiltä arjessa pitämällä palomuurin ja virusturvan ajantasaiseksi päivitettynä ja olemalla varovainen ladatessaan internetistä ohjelmia tai tiedostoja. Tekstiviestillä ja sähköpostilla tulleet linkit kannattaa tarkastaa useaan otteeseen ennen avaamista, sillä ne voivat johtaa jonkin tunnetun palvelun näköissivustolle, jonne syöttäessä henkilötiedot ja salasanan antaakin ne todellisuudessa kyberhyökkääjälle oikean palvelun sijasta. (F-secure 2023.)

Internetissä asiakas- ja palvelinyhteyden välistä viestintää on myös mahdollista yrittää urkkia siihen tarkoitetuilla ohjelmistoilla. Huonosti rakennettu yhteys ja vanhentuneiden, suojaamattomien protokollien käyttö altistaa yhteyden hyökkäykselle.

On tärkeää käyttää suojattua ja turvallista protokollaa verkkoyhteyden muodostamiseen palvelimen ja asiakkaan välille. Esimerkiksi HTTPS (Hypertext Transfer Protocol Secure) protokolla, joka on tarkoitettu hyperteksti-dokumenttien, kuten HTML-dokumentttien siirtoon. Se on tarkoitettu kommunikoimaan verkkopalvelimen ja verkkoselaimen välillä. (MDN Web Docs 2023d.)

HTTP ja HTTPS ovat koko lailla sama protokolla sillä erotuksella, että HTTPS on suojattu. Käytännössä tämä tarkoittaa sitä, että HTTPS-protokollalla lähetetyt viestipaketit salataan sekoittamalla sisältö täysin lukukelvottomaksi, jos se kaapataan matkalla. Sisällön avaamiseen tarvitaan avain, jonka kommunikoivat osapuolet vaihtavat toisilleen yhteyttä luodessa.

HTTP-protokollalla viestiä lähettäessä kuka tahansa, joka monitoroi yhteyttä, voi lukea viestin sellaisenaan. Esimerkiksi pääsemällä samaan verkkoon käsiksi, missä joko palvelinlaite tai asiakaslaite on, voi tarkastella laitteiden välistä viestiliikennettä käyttämällä siihen tarkoitettuja ohjelmia, kuten Wireshark.

7.5 Wireshark

Wireshark on verkkoprotokolla analysaattori, jolla voi tarkastella verkkoyhteydessä kulkevaa datapakettiliikennettä. Se kuuntelee verkkoyhteyttä ja poimii verkossa kulkevien datapakettien tiedot. Ohjelmassa voi suodattaa liikennettä, keskittyäkseen tarvittaessa tiettyyn yhteyteen. Yritykset käyttävät ohjelmaa tietoturva-aukkojen haisteluun. Salaamattomat viestit Wireshark pystyy näyttämään selkokielellä, mutta salauksia sillä ei pysty purkamaan.

8 Logiikkaohjelma ja käyttöliittymä

8.1 Logiikkaohjelma TIA-Portalissa

Verkkorajapinnan määrittämistä ja testaamista varten varsinaisen logiikkaohjelman ei tarvitse olla monimutkainen. Toimivuuden määrittämiseen riittää muutama yksinkertainen funktiolohko, joiden toimintaa pystyy verkkopalvelimella pyörivän käyttöliittymän kautta ohjaamaan. Tärkeimmät elementit verkkorajapinnan testaamista varten, ovat itse muuttujat.

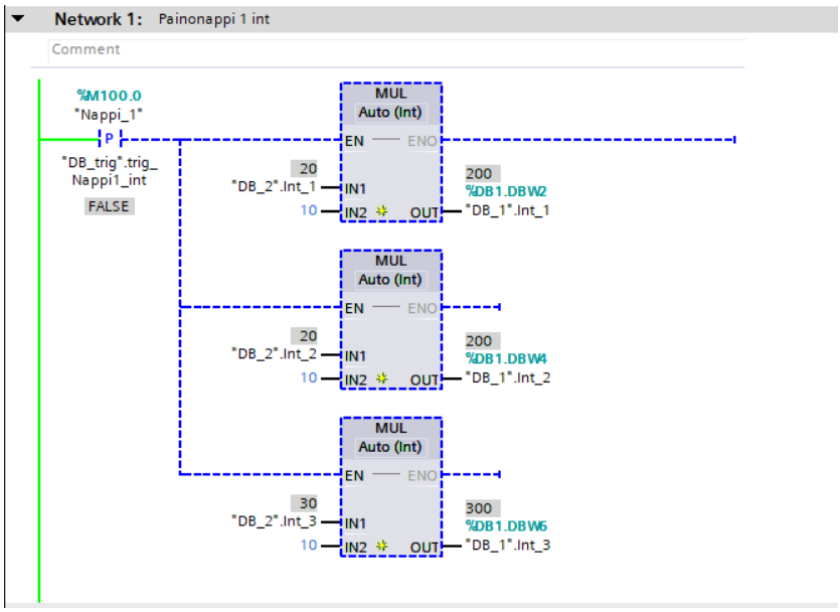
Pääasiallisesti TIA-Portalilla logiikkaa ohjelmoidessa muuttujia käytetään varastoimaan erilaisia tietoja ja tiloja. Numeeriset muuttujat voivat sisältää esimerkiksi moottorin paikoitustietoja ja bittimuuttujat On/Off tiloja, kuten moottorin päällä olotilan.

Testiohjelmaan on tehty 4 datalohkoa (DB), jotka kukin sisältävät identtisen listan muuttujia. Tämän lisäksi on tehty tagilista (kuva 24), joka sisältää useita eri datatyypin muuttujia muutamalla iteraatiolla.

		Name	Data type	Address	Retain	Writable from HMI/OPC UA/Web API	Accessible from HMI/OPC UA/Web API	Visible in HMI engineering
1		Nappi_1	Bool	%M100.0	☐	☑	☑	☑
2		Nappi_2	Bool	%M100.1	☐	☑	☑	☑
3		Nappi_3	Bool	%M100.2	☐	☑	☑	☑
4		Word_1	Word	%MW102	☐	☑	☑	☑
5		Word_2	Word	%MW104	☐	☑	☑	☑
6		Word_3	Word	%MW106	☐	☑	☑	☑
7		Int_tag_1	Int	%MW110	☐	☑	☑	☑
8		Int_tag_2	Int	%MW112	☐	☑	☑	☑
9		Int_tag_3	Int	%MW114	☐	☑	☑	☑
10		Real_tag_1	Real	%MD120	☐	☑	☑	☑
11		Real_tag_2	Real	%MD124	☐	☑	☑	☑
12		Real_tag_3	Real	%MD128	☐	☑	☑	☑
13		Bool_1	Bool	%M100.3	☐	☑	☑	☑
14		Bool_2	Bool	%M100.4	☐	☑	☑	☑
15		Bool_3	Bool	%M100.5	☐	☑	☑	☑
16		Nappi2-time	Time	%MD200	☐	☑	☑	☑
17		Nappi3-time	DInt	%MD208	☐	☑	☑	☑
18		Nappi2-reset	Bool	%M116.0	☐	☑	☑	☑

Kuva 24. Tagilista TIA-Portalissa

Varsinainen logiikkaohjelma on toteutettu yhdellä funktiolla (FC). Funktio sisältää neljä virtapiiriä. Ensimmäinen virtapiiri Network 1 (kuva 25) sisältää yksinkertaisen toiminnon, jossa Nappi_1-muuttujan mennessä false-tilasta true-tilaan ohjelma ottaa datalohko DB_2:sta muuttujien Int_1, Int_2 ja Int_3 arvot, kymmenkertaistaa ne ja kirjoittaa datalohkoon ”DB_1” vastaavien muuttujien paikalle saadut tulokset.



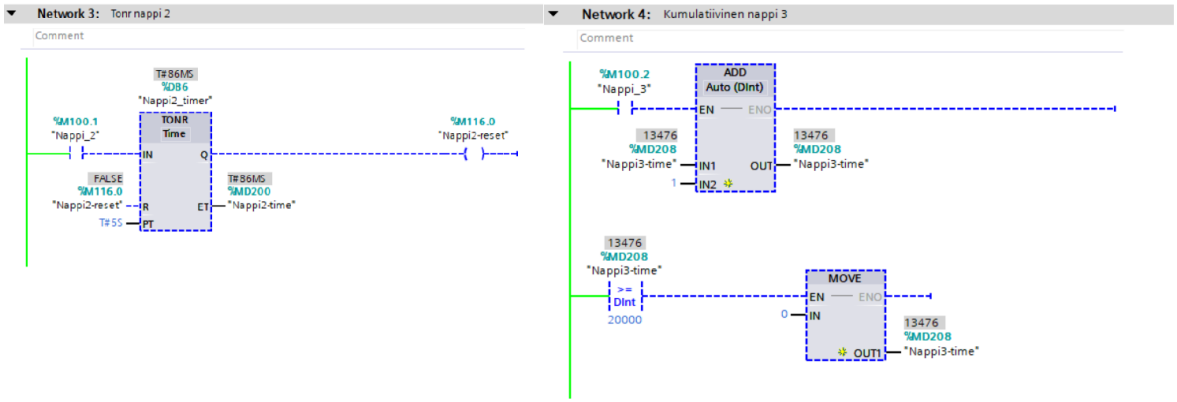
Kuva 25. Network 1 virtapiiri

Seuraava virtapiiri Network 2 suorittaa samankaltaisen toiminnon käyttäen datalohkon DB_2 Real-muuttujia, jakaen ne viidellä ja sijoittaen taas vastaavasti saadut tulokset datalohkon DB_1 samannimisten Real-muuttujien arvoiksi. Nappi_1 muuttujaa käskytetään verkkoselainpohjaisen käyttöliittymän kautta. Sen datatyyppi on Bool, tehden siitä on/off tiloina toimivan muuttujan (I/O).

Virtapiirit 3 ja 4 (kuva 26) ovat kumpikin omanlaisiaan kasautuvia laskureita. Virtapiiri 3 sisältää kumulatiivisen ajastimen, joka kerryttää aikaa Nappi_2 muuttujan ollessa tilassa 1 ja pysähtyy napin mennessä tilaan 0. Ajastimelle on asetettu 5 sekunnin aika, jonka kertyessä täyteen ajastimen ulostulo kytkeytyy päälle. Ulostulo aktivoi kärjen, jonka avulla ohjelma nollaa ajastimen itsensä. Ajastimen aika kertyy reaaliaikaisesti millisekunteina.

Askelma 4 sisältää matemaattisen toiminnon, joka lisää Nappi3-time nimiseen DINT-datatyypin muuttujaan luvun 1 jokaisella ohjelmakierrolla, sillä aikaa kun Nappi_3 muuttuja on aktivoituneena. Alemmalla rivillä Nappi3-time:lle on annettu rajaksi luku 20000, jonka jälkeen Nappi3-time muuttuja palaa takaisin nollaan.

Laskurit lisäävät kumpikin itseensä luvun yksi aina millisekunnin välein, mikäli askelman ehdot toteutuvat. Tätä lukemaa voidaan seurata verkkoselainpohjaisessa käyttöliittymässä reaaliaikaisesti 10 millisekunnin päivitysintervallilla.



Kuva 26. Kasautuvat laskurit

8.2 Käyttöliittymän koodi

Logiikkaohjelman rakenne riippuu täysin siitä, minkälaiselle automaatiojärjestelmälle se on tarkoitettu. Varsinaista Web-API rajapintaa käyttääkseen suurin työ tehdään TIA-Portal kehitysympäristön ja logiikkaohjelmoinnin ulkopuolella. Tapoja sen toteuttamiseen on monia. Tässä opinnäytetyössä käyttöliittymän verkkosivut ovat kirjoitettu käyttäen JavaScript-, HTML-, ja CSS-kieliä. Verkkosivuja isännöidään Windows-pohjaisella Microsoft IIS-palvelinohjelmalla.

Siemensin ohjelmoitavan logiikan Web-API rajapintaa hyödyntääksesi, tulee se logiikassa olla asetettuna käyttöön, tarvittavat edellytykset täytetty ja logiikalla täytyy olla IP-osoite. Ensimmäisenä lähdekoodissa täytyy kutsua tarvittavat dokumentit ja kirjastot käyttöön HTML-dokumentissa (kuva 27).

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>[example] Test program</title>
    <link rel="stylesheet" href="aki.css" />
    <script src="JavaScript/api.js"></script>
    <!--<script src="JavaScript/jquery.js"></script>-->
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

Kuva 27. Kutsutaan ulkoisia dokumentteja ja kirjastoja

Koodin varsinainen toiminta tapahtuu api.js JavaScript-tiedostosta tuoduilla funktioilla, joilla suoritetaan toiminnot halutusti ja muokataan auki olevaa HTML-dokumenttia dynaamisesti.

Heti verkkosivun käynnistyessä lähdekoodissa määritetään kohdelaitteen IP-osoite, protokollaksi määritetään HTTPS ja ajetaan getToken funktio (kuva 28). Funktio tarkistaa, onko asiakaslaitteella olemassa voimassa oleva valtuutus (Token).

```
4   "use-strict";
5   let messageId = 1
6   let TARGET_IP = "https://" + ("10.0.0.110") + "/api/jsonrpc"
7   let AUTH_TOKEN = null
8   //----------------------------------Get token. If token is not valid, go to login--------------
9   function getToken() {
10      cookieVal = getCookie("siemens_web_secure");
11      AUTH_TOKEN = getCookie("token");
12      if (cookieVal != "" && AUTH_TOKEN != "") {
13          document.getElementById("notice").innerHTML = "You are logged in"
14          getApiVersion()
15          bulkReadtagValues()
16          closeModal()
17          keepSession(AUTH_TOKEN);
18      }
19      else {
20          location.href = "login.html";
21      }
22  }
```

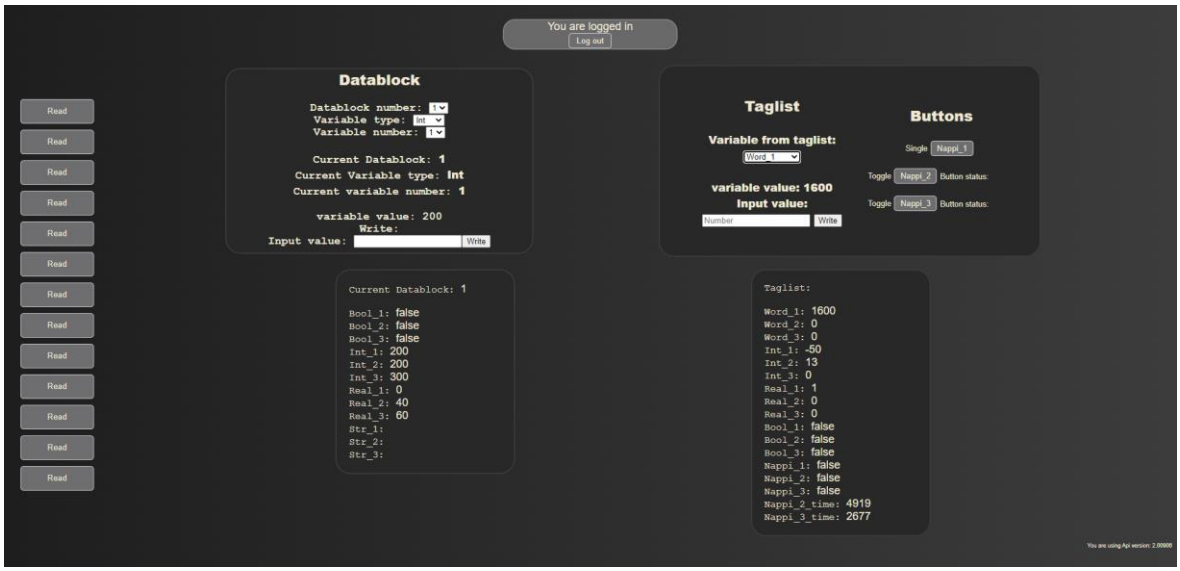
Kuva 28. IP-osoitteen määrittäminen ja valtuutuksen tarkistus

Jos voimassa olevaa valtuutusta ei ole, verkkosivu pyytää kirjautumaan sisään PLC:lle asetetun käyttäjätunnuksen ja salasanan avulla (kuva 29). Mikäli käyttäjätunnus ja salasana on oikein, ohjelma luo valtuutuksen ja päästää varsinaiselle sisältösivulle. Valtuutusta pidetään voimassa 2 minuuttia ja ajastin nollataan jokaisesta toimenpiteestä verkkosivulla. Mikäli verkkosivu on käytöttä ajastimen loppuun asti, valtuutus vanhenee ja se tulee uusia kirjautumalla uudelleen sisään.

Kuva 29. Sisäänkirjautuminen

Riippuen logiikkaohjelmasta ja sen sisältämistä muuttujista, verkkosivun lähdekoodi koostuu luku- ja kirjoitus pyynnöistä PLC:n ja palvelimen välillä, sekä ehdoista, jotka määrittävät, koska pyynnöt toteutuvat. Jonkin muuttujan tilaa voidaan päivittää jatkuvasti verkkosivulle ja toiset voidaan päivittää esimerkiksi jonkin tapahtuman yhteydessä. Tapahtuma voi olla napin painallus verkkosivulla tai jonkun toisen muuttujan tilan muutos PLC:n muistissa.

Käyttöliittymän ulkoasu ja toiminnot riippuvat täysin siitä, minkälaiseksi se on lähdekoodissa määritetty. Tämän opinnäytetyön esimerkkiverkkosivu esitetään kuvassa 30.



Kuva 30. Esimerkkiverkkosivu toimintoineen

Verkkosivun vasempaan laitaan on asetettu nappeja hahmottamaan, miten näytettävän sivun vaihto voitaisiin toteuttaa käyttöliittymässä.

Web-API mahdollistaa lukemisen ja kirjoituksen logiikan muuttujille, jotka sijaitsevat joko logiikan datalohkoissa tai tagilistassa. Esimerkkiverkkosivulla ”Datablock” otsikon alla voidaan pudotusvalikoista määrittää luettava datalohko ja sen sisältä luettava muuttuja valitsemalla esimerkiksi Datablock number: 1, Variable type: Int ja Variable number: 1. Muuttujat datalohkossa ovat ennaltamääritettyjä ja kyseinen kombinaatio määrittää kirjoitettavaksi muuttujaksi DB_1.INT_1 muuttujan. Saman valinta-alueen sisällä Variable value-kenttä näyttää kyseisen muuttujan senhetkisen arvon.

Valinta-alueen alapuolella on listattuna kaikki datalohkon sisältämät muuttujat ja näytetään kunkin tämänhetkinen arvo. Kuvassa 31 esitetään datalohko 1:n sisältämien muuttujien tilat.

```
Current Datablock: 1

Bool_1: false
Bool_2: false
Bool_3: false
Int_1: 200
Int_2: 200
Int_3: 300
Real_1: 0
Real_2: 40
Real_3: 60
Str_1:
Str_2:
Str_3:
```

Kuva 31. Datalohko 1 muuttujien tilat

Muuttujien luku ja muuttujiin kirjoittaminen hoidetaan lähdekoodissa JSON-RPC 2.0 protokollalla käyttäen POST-pyyntöä. Kuvassa 32 luetaan yhden muuttujan tila määritetystä datalohkosta. Funktio readdbvariable tarvitsee käyttäjältä kolme arvoa, jotka aiemmin määriteltiin. Näiden kolmen muuttujan arvon avulla muodostetaan osoite muuttujalle, johon POST-pyyntö ohjataan. Funktio lähettää JSON-pakettina PLC:n Web-API:lle viestin, jossa pyydetään suorittamaan PlcProgram.Read-metodi, eli lukutoiminto. Web-API suorittaa lukufunktion määritetylle muuttujalle logiikan puolella ja palauttaa muuttujan senhetkisen arvon vastaavanlaisena JSON-pakettina palvelimelle. Palvelin vastaanottaa paluuviestin ja kirjoittaa sen HTML elementiksi, minkä arvo pystytään verkkosivulla näyttämään.

```
137 //DB--------------------Read variable value from datablock-----------------------------------------
138 //-----------------------------------Expecting 3 inputs. DB number, var type and var number----------
139
140
141 function readdbvariable() {
142
143     let dbval = document.getElementById("readdbvalue")
144     fetch(TARGET_IP, {
145         method: "POST",
146         headers: {
147             "Content-type": "application/json",
148             "X-Auth-Token": AUTH_TOKEN
149         },
150         body: JSON.stringify({
151             "id": messageId++,
152             "jsonrpc": "2.0",
153             "method": "PlcProgram.Read",
154             "params": {
155                 "var": "\"DB_" + dbnum_global + "\"." + dbvartype + "_" + dbvarnum_global,
156             }
157         })
158     })
159         .then(response => response.json())
160         .then((data) => {
161             dbval.innerHTML = data.result
162         })
163         .catch(e => console.error(e))
164 }
```

Kuva 32. Yksittäisen muuttujan tilan lukufunktio

Kirjoittaminen muuttujaan tapahtuu vastaavalla tavalla käyttäen PlcProgram.Write-metodia. Kirjoitusfunktio kuitenkin vaatii lukufunktiota enemmän ehtoja ja parametrejä. Lukufunktio pystyy lukemaan suoraan kaikkia datatyyppejä, mutta kirjoitusfunktiolle täytyy jollain tavalla määrittää, mitä datatyyppiä ollaan kirjoittamassa. Esimerkiksi INT-muuttuja hyväksyy vain kokonaislukuja ja STRING-muuttuja merkkijonoja.

Kuvassa 33 on esitetty kirjoitusfunktio, joka kirjoittaa numeraalisia muuttujia logiikalle. Tämä on toteutettu lähettämällä verkkosivulta numero tekstimuodossa kirjoitusfunktiolle. Kirjoitusfunktio lähettää viestinä saadun arvon ja käskyn suorittaa arvolle parseFloat-funktio viestin saapuessa vastaanottajalle. Tässä yhteydessä se koostaa tekstimuotoisesta arvosta numeraalisen muodon, joka voidaan kirjoittaa sopivaan muuttujaan logiikassa. Paluuviestinä logiikalta palaa vain kuittaus siitä, onko lähetys onnistunut. Kirjoitusfunktio varmistaa ensimmäisenä valtuutuksen voimassaolon. Mikäli valtuutus on mennyt vanhaksi, ei kirjoittaminen toteudu.

```
165 //DB--------------------------------Write for a num variable-------------------------------------
166 //-------------------------------------------This write uses parsefloat to numberize received string--
167 function writenumvariable() {
168     if (AUTH_TOKEN !== null) {
169         let variable = document.getElementById("inputvar")
170         fetch(TARGET_IP, {
171             method: "POST",
172             headers: {
173                 "Content-type": "application/json",
174                 "X-Auth-Token": AUTH_TOKEN
175             },
176             body: JSON.stringify({
177                 "id": messageId++,
178                 "jsonrpc": "2.0",
179                 "method": "PlcProgram.Write",
180                 "params": {
181                     "var": "\"DB_" + dbnum_global + "\"." + dbvartype + "_" + dbvarnum_global,
182                     "value": parseFloat(variable.value)
183                 }
184             })
185         })
186             .then(response => response.json())
187             .then((data) => {
188                 if (data.result) {
189                     console.log("Api responds with " + data.result)
190                     bulkReaddbValues();
191                     readdbvariable();
192
193                 }
194             })
195             .catch(e => console.error(e))
196     }
197 }
```

Kuva 33. Kirjoitusfunktio numeroille

STRING-muuttujaan kirjoitus on toteutettu omalla kirjoitusfunktiolla, jossa value-muuttujan arvoksi annetaan verkkosivulta vastaanotettu teksti suoraan merkkijonona (kuva 34). Numeron kirjoituksesta eroten parseFloat-funktio on jätetty pois.

```
    },
    body: JSON.stringify({
        "id": messageId++,
        "jsonrpc": "2.0",
        "method": "PlcProgram.Write",
        "params": {
            "var": "\"DB_" + dbnum_global + "\"." + dbvartype + "_" + dbvarnum_global,
            "value": variable.value
        }
    })
})
```

Kuva 34. JSON viesti STRING-muuttujan kirjoittamiselle

BOOL-muuttujan kirjoittamiselle on myös hyvä tehdä oma kirjoitusfunktionsa. BOOL-muuttujalla voi olla kaksi tilaa: 1 ja 0. Nämä vastaavat yleisemmin tiloja true ja false. Tyypillisesti 0 vastaa false-tilaa ja 1 true-tilaa. Kuvassa 35 esitetään JSON-paketti BOOL-muuttujan kirjoittamiselle, jossa HTML-tiedostosta lähetetään funktiolle arvo.

```
231 //DB-----------------------------------write for bool a variable-----------------------------------
232 //-----------------------------------------True false: Expect true, if not, false----------------
233 function writeboolvariable(element) {
234     if (AUTH_TOKEN !== null) {
235         fetch(TARGET_IP, {
236             method: "POST",
237             headers: {
238                 "Content-type": "application/json",
239                 "X-Auth-Token": AUTH_TOKEN
240             },
241             body: JSON.stringify({
242                 "id": messageId++,
243                 "jsonrpc": "2.0",
244                 "method": "PlcProgram.Write",
245                 "params": {
246                     "var": "\"DB_" + dbnum_global + "\"." + dbvartype + "_" + dbvarnum_global,
247                     "value": (element.value === "true")
248                 }
249             })
250         })
251             .then(response => response.json())
252             .then((data) => {
253                 if (data.result) {
254                     console.log("Api responds with " + data.result)
255                     bulkReaddbValues();
256                     readdbvariable();
257                 }
258             })
259             .catch(e => console.error(e))
260     }
261 }
```

Kuva 35. BOOL-muuttujan kirjoitus

Funktion vastaanottama arvo on toteutettu merkkijonona. Riippuen käyttöliittymässä asetetusta tilasta, arvo on kirjoitettuna joko ”true” tai ”false” tekstinä. Funktion JSON-paketin

sisällä on annettu ehto, jossa odotetaan arvon olevan ”true”. Jos arvo ei ole ”true”, se vertausoperaattoria ”===” käyttäen on automaattisesti ”false”. Funktion arvoksi voisi siis syöttää ”false” tekstin sijasta tekstin ”Mikki Hiiri” ja vertausoperaattori muuttaisi siitä automaattisesti ”false” arvon.

Testikäyttöliittymän (kuva 30) oikeassa laidassa taas on vastaavanlainen luku- ja kirjoitusjärjestelmä tagilistan muuttujille. Tagilistan muuttujien kirjoittamiselle ja lukemiselle lähdekoodissa on vastaavanlaiset funktionsa muuttujaan viittaavan osoitteen vuoksi. Toisin kuin datalohkoihin osoittaessa tarvitsee aina nimetä käsiteltävä datalohko, tagilistaa ei tarvitse nimetä erikseen. Osoitteessa tarvitsee viitata vain muuttujaan nimeltä (kuva 36).

```
body: JSON.stringify({
    "id": messageId++,
    "jsonrpc": "2.0",
    "method": "PlcProgram.Write",
    "params": {
        "var": "\"" + tagvar + "\"",
        "value": parseFloat(variable.value)
    }
})
```

Kuva 36. Tagilistan muuttujan parametri osoitteessa

Tagilistaan kirjoittaminen ja lukeminen on toteutettu kirjoittamalla funktioon yksi parametroitu arvo, jonka voi valita pudotusvalikosta. Pudotusvalikossa on kunkin tagilistassa sijaitsevan muuttujan nimet (kuva 37). Ohjelmoijan täytyy ennalta tietää kaikkien logiikassa sijaitsevien muuttujien nimet, jotta niihin voidaan koodissa viitata. Uusia muuttujia ei voi luoda logiikalle verkkosivun kautta, vaan se on tehtävä TIA-Portal kehitysympäristössä.

Kuva 37. Tagilistassa sijaitsevat muuttujat

Automaatiojärjestelmän ohjauksessa painikkeet ovat tärkeässä roolissa. Testiverkkosivulle on toteutettu kahdentyyppisiä painikkeita: pito- ja kertapainallus (kuva 38).

```
513  /* --------------Toggle button function---------------------------- */
514  function tglnappi(element) {
515      var nappistat = document.getElementById("btn-nappi" + element);
516      var nappimes = document.getElementById("nappi" + element);
517      if (nappistat.value == "false") {
518          nappistat.value = "true";
519
520      }
521      else if (nappistat.value == "true") {
522          nappistat.value = "false";
523      }
524      nappimes.innerHTML = nappistat.value
525      console.log(nappistat.value)
526      writebutton(element);
527  }
528  /* --------------Single action button function---------------------------- */
529  function nappi(element) {
530      var nappistat = document.getElementById("btn-nappi" + element);
531      var nappimes = document.getElementById("nappi" + element);
532
533      nappistat.value = "true";
534      nappimes.innerHTML = nappistat.value;
535      writebutton(element);
536      nappistat.value = "false";
537      nappimes.innerHTML = nappistat.value;
538      writebutton(element);
539      bulkReaddbValues();
540  }
```

Kuva 38. JavaScript-koodi pito- ja kertapainallusnappien toiminnalle

Kukin nappi on toteutettu omalla numerolla ja kummallekin nappityypille on oma funktio. Kertapainallusnapille rakennettu nappi-funktio kirjoittaa yhdellä kerralla peräkkäin logiikalle nappia vastaavan BOOL-muuttujan arvoksi ensin true ja heti perään false. Painonappien toiminta tulee huomioida ensisijaisesti logiikkaohjelmassa.

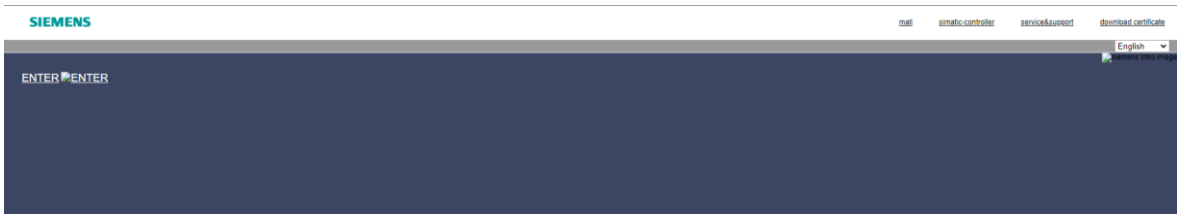
Pitopainikkeen tglnappi-funktio vaihtaa logiikalle nappia vastaavan BOOL-muuttujan arvon true tilaan, mikäli se oli aikaisemmin false ja sama toisinpäin. HTML-dokumentissa käyttöliittymän painonapin toimintatavaksi on määritetty ”onclick” mikä tarkoittaa, että napin toiminto tapahtuu vain kerran. Nappia pohjassa pitämällä pitotila ei siis muutu joka ohjelmakierrolla. Myös kertapainallusnappi on määritetty samalla toimintatavalla. Funktioille annetaan arvoksi napin numero, jonka avulla funktio suorittaa numeroa vastaavalle muuttujalle tilan muutoksen logiikalla.

8.3 Yhteyden suojaus

Logiikan ja palvelimen yhteys on suojattu monella tavalla. Siemens Web-API:n käyttö vaatii palvelimen ja logiikan välisellä yhteydellä käytettävän HTTPS-protokollaa, joka salaa edestakaisin kulkevat viestipaketit. Logiikalle on asennettu varmennettu sertifikaatti, jota myös vaaditaan yhdistävältä asiakaslaitteelta.

Mikäli asiakaslaitteelle ei ole asennettu asianmukaista sertifikaattia, voi esimerkiksi lähdekoodissa luoda yhdistävälle asiakaslaitteelle ponnahtamaan ilmoituksen sertifikaatin puutteesta. Tämän opinnäytetyön tapauksessa sertifikaatin puute johtaa sisäänkirjautumisen epäonnistumiseen.

Siemensin logiikoilla on sisäänrakennettu verkkosivu, johon voi yhdistää kirjoittamalla selaimessa osoiteriville suoraan logiikan IP-osoitteen. Tässä tapauksessa yhdistäminen tapahtuu kirjoittamalla osoitekenttään https://10.0.0.110. Yhdistettäessä aukeaa kuvan 39 osoittama näkymä.



Kuva 39. Siemensin logiikalle sisäänrakennettu verkkosivu

Sivun oikeassa yläkulmassa tarjotaan optiota ”download certificate”, eli lataa sertifikaatti. Tätä käyttämällä asiakaskone voi hakea itselleen PLC:n vaatiman sertifikaatin yhteyden luomiseen. PLC sijaitsee usein ainoastaan yrityksen lähiverkossa, eikä sillä ole pääsyä internettiin, joten sertifikaatin lataustoiminto helpottaa asianmukaisten asiakaslaitteiden käyttöönottoa.

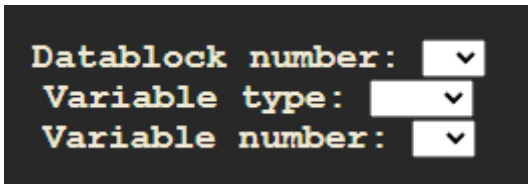
Pääsyn sisäänrakennetulle sivulle voi estää logiikan asetuksista. TIA-Portal kehitysympäristössä on aloitussivuksi oletuksena määritetty Siemensin sisäänrakennettu sivu, jos yhdistää suoraan IP-osoitteen avulla logiikkaan. Sivulle pääsyn voi estää ohjaamalla IP-osoitteeseen yhdistävä asiakaslaite mukautetuille verkkosivuille, jossa voidaan esimerkiksi ilmoittaa pääsyn olevan estetty. Jos logiikasta asetetaan ohjaus mukautetulle verkkosivulle, mutta varsinaista mukautettua verkkosivua ei ole olemassa, asiakaslaitteen yhdistysyritys päättyy tyhjään verkkosivuun, jossa ilmoitetaan, ettei verkkosivu ole saatavilla. Tässä tapauksessa sertifikaattia ei pääse suoraan lataamaan, vaan se täytyy toimittaa asiakaslaitteelle jollain muulla tavalla. TIA-Portal kehitysympäristön sisällä sertifikaatista voi luoda erillisen asennettavan tiedoston, jonka voi toimittaa asianmukaisille asiakaslaitteille asennettavaksi manuaalisesti.

TIA-Portal projektille tulee myös luoda käyttäjätunnus ja salasana ja näitä tulee vaatia käyttöliittymän lähdekoodissa. On mahdollista luoda useita kirjautumisvaihtoehtoja, kukin eri

oikeuksilla. Käyttäjille voi määrittää muuttujien kirjoitus- ja lukuoikeudet tai esimerkiksi vain lukuoikeuden.

Web-API tarjoaa mahdollisuuden käyttää ”token”-menetelmää, minkä avulla voit pyytää logiikalta tunnisteen. Tätä tunnistetta käytetään keksinä (Cookie) valtuuttamaan yhteys. Se koostuu 28-tavuisesta merkkijonosta, jota vaaditaan jokaista toimintoa käyttäessä. Jokaisen onnistuneen toiminnon yhteydessä token-tunnisteen voimassaolon ajastin nollataan. Voimassaolo vanhenee kahden minuutin kuluessa viimeisen toiminnon suorittamisesta.

Lähdekoodissa on pystyttävä viittaamaan kuhunkin muuttujaan, jotta niitä voidaan lukea tai niihin voidaan kirjoittaa. Logiikkaohjelman datalohkot tai tagilistat voivat siis sisältää muuttujia, joihin ei viitata lähdekoodissa. Käyttöliittymässä voi olla kuitenkin tapoja viitata muuttujaan parametroimalla sen nimi (kuva 40). Käyttöliittymä voi sallia käyttäjän määrittävän viitattava muuttuja, esimerkiksi kirjoittamalla sen nimi tekstikenttään manuaalisesti.



Kuva 40. Parametroitava muuttujaviittaus

Yksittäisille muuttujille voi antaa ja evätä luku- ja kirjoitusoikeuksia. Tällä keinolla varmistetaan, että vaikka parametroitava, tai suoraan kirjoitettava muuttujaan viittaus menetelmä sallisi käyttäjän viitata johonkin muuttujaan, jota ei tule voida lukea tai johon ei tule voida kirjoittaa, voidaan estää toiminta rajaamalla suoraan yksittäisen muuttujan muokkausoikeudet. Samaan tapaan kuin yksittäisen muuttujan, myös kokonaiselle datalohkolle voi määrittää luku- ja kirjoitusoikeudet Web-API:n rajapintaa hyödyntäen.

9 Makron Service

Makron Service on tämänhetkinen tapa monitoroida automaatiojärjestelmää selainpohjaisella käyttöliittymällä. Koko Service-järjestelmä koostuu kolmesta pääkomponentista: Verkkopalvelin, tietokanta ja Makron Service.

Makron Service on itse kirjoitettu rajapintaohjelma, joka toimii palvelimen ja PLC:n välissä. Se on ohjelma, joka pyörii Windows-järjestelmässä taustalla Windows-Service sovelluksena. Ohjelma keskustelee PLC:n kanssa viestipakettien avulla, käyttäen suoraa sokettiyhteyttä. Luku ja kirjoitus PLC:lle tapahtuu absoluuttisella osoittamisella logiikan muistialueeseen.

Käytännössä Service lukee halutun tiedon PLC:ltä absoluuttisesta osoitteesta ja kirjoittaa tiedon tietokantaan. Verkkopalvelimen puolella pyörivä ohjelma tarkkailee muutoksia tietokannassa ja muutosten ilmentyessä päivittää verkkoselaimelle tuoreen tiedon.

Turvallisuussyistä Makron Service-ohjelman tarvitsemat tiedot ovat vahvasti kovakoodattuja ohjelman sisään. Ulkopuolisella taholla ei ole pääsyä tarkastelemaan ohjelman koodia, sillä koodi pakataan yhdeksi suoritettavaksi tiedostoksi asiakkaan laitteelle.

10 Tulokset ja vertailu

10.1 Tarkasteltavien asioiden määrittely

Opinnäytetyössä tehtävänä oli perustaa verkkopalvelin ja verkkosivupohjainen käyttöliittymä logiikkaohjelmalle. Logiikan kanssa keskusteluun tuli käyttää uutta logiikalle sisäänrakennettua Web-API rajapintaa. Yhteyksien kyberturvallisuus mahdollisuuksia tuli tarkastella ja käytettävät yhteydet turvata. Tarkoituksena oli myös tutkia nykyisen sokettiyhteyttä hyödyntävän käyttöliittymän turvallisuutta ja ominaisuuksia.

Uuden Web-API rajapinnan turvallisuusominaisuuksia oli tarkoitus testata eri tilanteissa. Logiikalle yhdistävältä yhteydeltä tuli vaatia voimassa oleva sertifikaatti ja yhteyden muodostus tuli evätä, mikäli sertifikaattia ei ole. Sertifikaatin jakelu tuli olla rajattua ja hallittavissa. Sekä logiikan ja palvelimen välinen yhteys, että palvelimen ja asiakaskoneen välinen yhteys tuli turvata ja kulkevat viestipaketit salata. Logiikkaohjelmassa tuli pystyä luku- ja kirjoitussuojaamaan yksittäisiä muuttujia, sekä kokonaisia datalohkoja. Logiikkaohjelmaan piti myös luoda käyttäjätunnuksia eri oikeuksilla, kuten luku- ja kirjoitusoikeudet.

Vertailussa otetaan huomioon myös kummankin käyttöliittymän käyttöönottomahdollisuudet ja toteutuksen hankaluus. Lisäksi käyttöliittymien luku- ja kirjoitusnopeuksia tulee verrata, sekä arvioida kunkin käyttöliittymän muokattavuus toimittajan puolesta.

10.2 Testien tulokset

Tietoturvan näkökulmasta Web-API tarjoaa paljon enemmän mahdollisuuksia logiikan ja yhteyksien turvallistamiseksi. Erona on yhteyksien käyttämät teknologiat ja protokollat. Web-API käyttää verkkoteknologioita ja Service puolestaan ei. Kummallakin vaihtoehdolla on puolensa ja haittansa. Vaikka Web-API rajapinnan käyttöä voidaan oletusarvoisesti helpommin turvata, on se Siemensin logiikalle sisäänrakentama ominaisuus. Tästä syystä se ei ole ollenkaan yhtä helposti laitteentoimittajan muokattavissa, kuin esimerkiksi Service-ohjelma on. Service-ohjelma on käytännössä vastaavanlainen rajapintasovellus, mutta se pyörii palvelinkoneella ja sen koodi on laitteentoimittajan puolesta täysin muokattavissa.

Käyttöönotto PLC:n puolella onnistuu Servicen osalta helposti. Käytännössä logiikalle täytyy sallia PUT/GET-kommunikointi valintaruutua napsauttamalla. Web-API puolestaan vaatii enemmän tekemistä, sillä sille täytyy luoda ja asettaa sekä sertifikaatit, että käyttäjätunnukset verkkopalvelimen aktivoimisen lisäksi.

Käyttöönotto PC:n puolella käy Web-API:n tapauksessa helpommin, sillä iso osa tiedonkäsittelyyn tarvittavasta koodista ja ohjelmasta on valmiiksi sisäänrakennettuna PLC:n verkkopalvelimella. Web-API rajapintaa tarvitsee siis vain kutsua ulkoisella käyttöliittymällä, jolloin vastauskäytäntöjen luominen jää ohjelmoijalta pois. Käyttöliittymä hyödyntää yleistä verkkoteknologiaa, joten sen rakentamiseen on tarjolla paljon ennestään hyväksi todettuja, valmiita koodipaketteja ja käytäntöjä.

Viestin lähetys- ja vastaanottosyklin nopeuksia tarkasteltiin myös. Joidenkin automaatiojärjestelmien käytettävyyden kannalta tiedonsiirron nopeus voi olla kriittistä. Datapakettien pyynnistä vastaanottohetkeen kummasakin tapauksessa meni vain millisekunteja. Servicen käyttämä sokettiyhteys osoittautui hieman nopeammaksi, keskiarvolla 7.4 ms läpimenoajalla, kun taas Web-API:n välityksellä viesti kulki 8.3 ms läpimenoajalla. Tulos saatiin vertaamalla Wiresharkilla lähetetyn ja vastaanotetun datapaketin aikaa. Mitattu aika kattaa yhteyden muodostamisen ja läpimenoajan kevyellä viestillä, ilman suurta kuormaa.

10.2.1 Web-API

Web-API rajapintaa käyttävän käyttöliittymän viestipakettien turvallisuutta testattiin käyttämällä Wireshark-pakettianalysaattoria. Kuvassa 41 esitetään Wiresharkilla kaapattu HTTP protokollaa käyttävä viesti asiakaslaitteen ja verkkopalvelimen väliltä. Viestiä ei ole suojattu TLS-salauksella.

Ohjelman koodi on kirjoitettu JavaScript-kieltä käyttäen, eikä sitä ole piilotettu PHP-kielellä. Kaapattu viestipaketti sisältää lähdekoodin täydellisyydessään ja se on luettavissa selkokielellä.

```
    Content-encoded entity body (gzip): 3299 bytes -> 24245 bytes
    File Data: 24245 bytes
Line-based text data: text/html (465 lines)
    \uFEFF<!--\n
        // Copyright (c) 2023, Siemens AG\n
        //\n
        // SPDX-License-Identifier: MIT\n
    -->\n
    <!DOCTYPE html>\n
    <html>\n
    \n
    <head>\n
        <meta charset="UTF-8">\n
        <title>[example] Test program</title>\n
        <link rel="stylesheet" href="aki.css" />\n
        <script src="JavaScript/api.js"></script>\n
        <!--<script src="JavaScript/jquery.js"></script>-->\n
        <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>\n
    \n
    \n
    </head>\n
    <!------------------------------------- Begin body ------------------------------------------------------------------------------------->\n
    \n
    <body onload="getToken()">\n
    \n
        <div class="allbody">\n
            <!-- Top section ------------------------------------------------------------------------------------------------------------->\n
            <div class="top">\n
                <div class="top2">\n
                    <span id="notice">Please log in first</span><br>\n
                    <!-- <button id="btn-login" onclick="openModal()"> Login </button> OBSOLETE -->\n
                    <button id="btn-logout" class="button3" onclick="apiLogout()"> Log out </button>\n
```

Kuva 41. Kaapattu HTTP-protokollaa käyttävä viesti sisältää lähdekoodin kokonaisuudessaan

Sama viesti kaapattiin uudelleen, käyttäen viestin lähettämiseen HTTPS-protokollaa. TLS-suojaus sekoittaa viestin tulkintakelvottomiksi merkeiksi, eikä kaappaaja saa siitä käytännössä mitään irti. Kuva 42 esittää TLS-salauksella suojatun viestin sisällön.

```
····{L·· ····U···
4··W···· ·JM·>···
·····>+ ··Y··*·m
···w··+· ·Y··9·lJ
·····@0 ·^···Ps·
G····:·x 5·····y·
\·V2··G· !^·····
l:····· P{···~··
·A&· !y· ·4·Z····
·H··$T·] ····>··P
·"c····· ···^·%j·
·····>·n ·····O
·=····· ···w····
··k·g··· ··7_··v[
··q····· ···_··W
#····· U:·vI·o·
·····R·X o··Y··>2
I···F··· ··o·····
?·R··AQ· 7·····
·····4·. ·····
·····q·+ ·4->·~|·
·3h·0·<· $···aD·n
····W·a: yG·····u
··b·vkpT h··,··Lk
·····o· `·····
T·lj···· zd\· ···
}·s1··^· FY·····8
?·s····· ·:A%1···
··Z·~·.· ·····r·Z
P···\d·· ·T%(····
```

Kuva 42. TLS-Salauksella suojatun viestin sisältö

Web-API rajapintaa hyödyntävän käyttöliittymän viestipaketit voidaan siis suojata monella tavalla. HTTPS-protokolla onnistuneesti salaa viestit ja murtaminen on käytännössä mahdotonta. Sertifikaatin vaatiminen PLC:n puolesta on tehty helpoksi ja asianmukaisen sertifikaatin puuttuessa yhteyden muodostus pysähtyy virheilmoitukseen. Sertifikaatin jakelu yrityksen sisällä voidaan rajata hoidettavaksi tietyn tahon toimesta, mikäli PLC:ltä estetään sertifikaatin noutaminen sisäänrakennettua verkkosivua käyttäen.

Yksittäisten muuttujien, sekä kokonaisten datalohkojen kirjoitus- ja lukusuojaus onnistui helposti logiikkaohjelman sisällä. Datalohkojen asetukset tarjoavat valintaruuduilla parametroitavat suojaukset kullekin muuttujalle, sekä datalohkolle itselleen on erikseen valintaruuduittain valittavana luku- ja kirjoitusoikeudet. Web-API rajapintaa käyttävällä käyttöliittymällä lukusuojatun muuttujan lukeminen tuottaa tulokseksi UNDEFINED arvon.

Web-API ei vaadi PUT/GET-kommunikointimenetelmää. Poistamalla logiikan asetuksista valintaruudusta menetelmän käytöstä, estää aiempaa sokettiyhteyttä käyttävää järjestelmää yhdistämästä logiikalle ollenkaan, eikä tietojen urkkiminen ollut täten mahdollista.

10.2.2 Service

Sokettiyhteyttä hyödyntävä service-käyttöliittymä on huomattavasti vaikeampi suojata. Sellaisenaan PUT/GET-kommunikointimenetelmällä sokettiyhteyden muodostamista ei voi suojata sertifikaatilla. Sertifikaatit ovat verkkoteknologiaa, eikä suora sokettiyhteys käytä sitä. Samasta syystä PUT/GET-kommunikointimenetelmällä lähettettyjä viestejä ei voi salata palvelimen ja PLC:n välillä, sillä datapaketti muodostetaan eri tavalla, eikä yhteyttä muodostaessa vaihdeta salaukseen vaadittavia kättelyjä ja avainpareja. Kuvassa 43 esitetään kaapatun datapaketin tiedot Wiresharkin sisällä. PLC on lähettänyt palvelimelle datapaketin käyttäen S7COMM protokollaa, eli PUT/GET-kommunikointimenetelmää. Datapaketissa pyydetään käyttämään Read Var-funktiota.

146	11.792889	10.0.0.110	10.0.0.23	S7COMM	265	ROSCTR:[Ack_Data] Function:[Read Var]

Kuva 43. Kaapattu datapakettirivi Wiresharkin sisällä

Paketin sisältö on tarkasteltavissa yksityiskohtaisesti. Tässä tapauksessa pyritään lukemaan muuttujat datalohko 1:stä (kuva 44).

```
Parameter: (Read Var)
    Function: Read Var (0x04)
    Item count: 1
    Item [1]: (DB 1.DBX 0.0 WORD 93)
        Variable specification: 0x12
        Length of following address specification: 10
        Syntax Id: S7ANY (0x10)
        Transport size: WORD (4)
        Length: 93
        DB number: 1
        Area: Data blocks (DB) (0x84)
        Address: 0x000000
```

Kuva 44. Pyydetty datalohko viestipaketin sisällä

Varsinaisen viestin runko on pitkä ja monimutkainen. Kuvassa 45 on esitettynä kaapattu paluuviesti, joka sisältää pyydetyt tiedot ja muuttujien arvot. Viestin sisältämä tieto esitetään binäärinä, joiden arvot vastaavat viestin muodostamiseen tarvittavia merkkejä ja varsinaisten muuttujien arvoja.

```
0000  01110100 11100101 11111001 01100100 01011001 00101001 00101000 01100011   t··dY)(c
0008  00110110 11110111 10001111 00111000 00001000 00000000 01000101 00000000   6··8··E·
0010  00000000 11111011 00010000 10001100 01000000 00000000 01000000 00000110   ····@·@·
0018  00010100 11101101 00001010 00000000 00000000 01101110 00001010 00000000   ·····n··
0020  00000000 00010111 00000000 01100110 11100001 10100011 00011000 10001010   ···f····
0028  00000110 01111110 01110100 00100000 01101001 10011011 01010000 00011000   ·~t i·P·
0030  00100000 00000000 01110111 11111100 00000000 00000000 00000011 00000000    w·····
0038  00000000 11010011 00000010 11110000 10000000 00110010 00000011 00000000   ·····2··
0040  00000000 00000000 00000001 00000000 00000010 00000000 10111110 00000000   ·····
0048  00000000 00000100 00000001 11111111 00000100 00000101 11010000 00000000   ·····
0050  00000000 00000100 01010111 00101111 01011001 01111010 01010001 00000000   ··W/YzQ·
0058  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0060  00000000 00000000 00000000 11111110 00000000 00000000 00000000 00000000   ·····
0068  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0070  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0078  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0080  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0088  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0090  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0098  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00a0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00a8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00b0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00b8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00c0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00c8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00d0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00d8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00e0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00e8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00f0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
00f8  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0100  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ·····
0108  00000000                                                                  ·
```

Kuva 45. Kaapattu paluuviesti sisältäen datalohkolta luetut arvot

Kuvassa 45 rivi 50 sisältää kolmen muuttujan tiedot, joita testikäyttöliittymä pyytää noudettavaksi. Halutut muuttujat ovat INT-tyyppisiä, eli kokonaislukuja. INT-muuttuja käsittää 2 tavua, eli 16 bittiä. Yksi tavu siis sisältää 8 bittiä. Ensimmäinen tavu rivillä 50 on tyhjä. Seuraavat 6 tavua sisältävät muuttujien Int_1, Int_2 ja Int_3 arvot. Arvot ovat ilmoitettu binäärinä ja vastaavat kokonaislukua seuraavasti:

- 00000100 01010111 = 1111
- 00101111 01011001 = 12121
- 01111010 01010001 = 31313

Kuvassa 46 esitetään datalohkon senhetkiset arvot logiikan datalohkossa.

5	Int_1	Int	2.0	0	1111	
6	Int_2	Int	4.0	0	12121	
7	Int_3	Int	6.0	0	31313	

Kuva 46. Datalohkon sisältämät arvot muuttujille

Datalohkojen sisältämiä muuttujia ei myöskään voitu suojata luku- ja kirjoitusoikeuksia rajoittamalla, sillä ne ovat määritetty vain Web-API rajapintaa varten, eikä PLC:n puolesta voida vaatia sisäänkirjautumista yhteyden muodostamista varten.

11 Yhteenveto ja pohdinta

Työn tavoite oli perehtyä uuteen tapaan luoda käyttöliittymiä Siemensin logiikkaohjelmille, sekä turvallistaa sen käyttämät yhteydet. Tämän lisäksi piti testata tämänhetkisen menetelmän tietoturvallisuusominaisuuksia ja varmistaa, ettei vanha menetelmä pääse käsiksi logiikan tietoihin uuden menetelmän ollessa käytössä.

Kaikkiin alussa asetettuihin tavoitteisiin päästiin ja opinnäytetyön voidaan katsoa onnistuneen hienosti. Verkkopalvelin perustettiin ja käyttöliittymä saatiin luotua. Yhteyksien suojaukset onnistuivat halutulla tavalla. Toimeksiantajayritys voi varmasti jatkokehittää järjestelmää opinnäytetyön pohjalta ja todennäköistä on, että tulevaisuudessa uudesta verkkoteknologioita hyödyntävästä Web-API rajapinnasta tulee jopa standardi ominaisuus Siemensin logiikoille ohjelmistoa luodessa. Verkkoteknologioiden hyödyntäminen mahdollistaa entistä helpommin modernien käyttöliittymien suunnittelun erilaisille laitteille.

Haasteellisen opinnäytetyöprosessista teki se, että Verkkoteknologioihin pohjautuvan käyttöliittymän rakentamiseen vaaditaan Web-kehitys työkaluja ja taitoja. Työn tekijällä ei prosessia aloittaessa ollut käytännössä aavistustakaan näistä, joten kaiken aiheeseen liittyvän on saanut opiskella tekemisen yhteydessä.

Käyttöliittymiä ja menetelmiä vertailemalla opinnäytetyössä toisiinsa saa helposti sellaisen kuvan, että Web-API olisi huomattavasti tietoturvallisempi ja Service ei juuri lainkaan. Tulee kuitenkin pitää mielessä se seikka, että automaatiojärjestelmän palvelinkone, jolla kummassakin tapauksessa verkkopalvelimet pyörivät, sijaitsevat lähes poikkeuksetta kyseisen automaatiojärjestelmän yhteydessä. Automaatiojärjestelmille on yleensä yrityksessä sisäinen verkko, jonka sisällä kaikki yhteys tapahtuu. Näin ollen ulkopuolisella taholla ei ole pääsyä järjestelmään ollenkaan. Vaikka sokettiyhteydellä siirreltäviä tietopaketteja on mahdollista sopivilla välineillä purkaa auki, ei sille todellisuudessa kovin suurta riskiä ole. Jos taho, joka tahtoisi vahingoittaa järjestelmää olisi jo päässyt käsiksi verkkoon, jossa palvelin sijaitsee, olisi se kuitenkin käytännössä mitätön uhka verrattuna siihen, mitä muuta hyökkääjä siinä tilanteessa voisi tehdä. Ensisijainen tietoturva tulee aina kyseiseen verkkoon pääsyn hallitsemisesta ja rajaamisesta.

Lähteet

Bee, L. 2022. Tia Portal basics – Working with arrays. Blogi. Viitattu 22.3.2023. Saatavissa: https://liambee.me/siemens/tia-portal-basics-working-with-arrays/

Computer Science. 2023. Computer programming languages. Viitattu 6.3.2023. Saatavissa https://www.computerscience.org/resources/computer-programming-languages/#javascript

Cloudflare. 2023. What is SSL? Viitattu 10.3.2023. Saatavissa: https://www.cloudflare.com/learning/ssl/what-is-ssl/

Elgabry, O. 2016. JSON in a nutshell. Blogi. Viitattu 6.3.2023. Saatavissa https://medium.com/omarelgabrys-blog/json-in-a-nutshell-7d638dfea7cc

Futurelearn. 2023. HTML in a nutshell. Viitattu 6.3.2023. Saatavissa https://www.futurelearn.com/info/courses/software-development-basics/0/steps/259147

F-seruce. 2023. Mikä on kyberhyökkäys?. Viitattu 23.3.2023. Saatavissa: https://www.f-secure.com/fi/articles/what-is-a-cyber-attack

Hansen, D. 2015. Programmable logic controllers: A Practical Approach to IEC 61131-3 Using CoDeSys. John Wiley & Sons, incorporated.

Instblog. What is PLC? Blogi. Viitattu 8.3.2023. Saatavissa: https://instrumentationblog.com/what-is-plc-complete-guide/

JSON. Introducing JSON. Viitattu 3.6.2023. Saatavissa: https://www.json.org/json-en.html

KB controls. 2022. Working with S7-1500's new WEB API. Viitattu 9.3.2023. Saatavissa: https://www.kb-controls.io/post/working-with-the-s7-1500s-new-web-api

Makron Oy. 2023a. Sisukasta yrittäjyyttä ja suuria unelmia. Viitattu 20.2.2023. Saatavissa https://makron.com/fi/yritys/historia/

Makron Oy. 2023b. Tietoa Makronista. Viitattu 20.2.2023. Saatavissa https://makron.com/fi/yritys/tietoa-makronista/

Puttonen, K. 2019. Sukupolvi vaihtuu Makronilla Hollolassa. Etelä-Suomen Sanomat 21.12.2019.

MDN Web Docs. 2023a. CSS: Cascading style sheets. Viitattu 6.3.2023. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/CSS

MDN Web Docs. 2023b. AJAX. Viitattu 6.3.2023. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

MDN Web Docs. 2023c. XMLHttpRequest. Viitattu 6.3.2023. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

MDN Web Docs. 2023d. HTTP. Viitattu 23.3.2023. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/HTTP?retiredLocale=fi

Microsoft. 2023. A tour of the C# language. Viitattu 8.3.2023. Saatavissa: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

PHP. 2023. What is PHP? Viitattu 8.3.2023. Saatavissa: https://www.php.net/manual/en/intro-whatis.php

Siemens. 2018. Learn-/Training document. Viitattu 8.3.2023. Saatavissa: https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/basics-programming-s7-1200/sce-031-200-fb-programming-s7-1200-r1709-en.pdf

Siemens. 2022a. Simatic s7-1500 CPU 1513-1 PN. Viitattu 8.3.2023. Saatavissa: https://cache.industry.siemens.com/dl/files/842/109752842/att_936139/v2/s71500_cpu1513_1_pn_dtc_manual_en-US_en-US.pdf

Rouse, M. 2020. Website. Blogi. Viitattu 23.3.2023. Saatavissa: https://www.techopedia.com/definition/5411/website

Siemens. 2022b. S71500 Webserver Function manual. Viitattu 9.3.2023. Saatavissa: https://cache.industry.siemens.com/dl/files/560/59193560/att_898124/v1/s71500_webserver_function_manual_en-US_en-US.pdf

Rosencrance, L. 2019. Internet Information Services (IIS). Blogi. Viitattu 23.3.2023. Saatavissa: https://www.techtarget.com/searchwindowsserver/definition/IIS

Tutorialsteacher. 2023. What is WEB API? Viitattu 9.3.2023. Saatavissa: https://www.tutorialsteacher.com/webapi/what-is-web-api

VS Code. 2023. Why Visual Studio Code? Viitattu 23.3.2023. Saatavissa: https://code.visualstudio.com/docs/editor/whyvscode

W3schools. 2023a. HTML Tutorial. Viitattu 6.3.2023. Saatavissa https://www.w3schools.com/html/default.asp

W3schools. 2023b. HTML CSS. Viitattu 6.3.2023. Saatavissa https://www.w3schools.com/html/html_css.asp

W3schools. 2023c. JavaScript AJAX. Viitattu 6.3.2023. Saatavissa: https://www.w3schools.com/xml/ajax_intro.asp

W3schools. 2023d. C# Introduction. Viitattu 8.3.2023. Saatavissa: https://www.w3schools.com/cs/cs_intro.php

Wayand, B. 2020. What is a PLC. Blogi. Viitattu 8.3.2023. Saatavissa: https://www.mroelectric.com/blog/what-is-a-plc/

Whatwg. 2023. HTML living stantard. Viitattu 6.3.2023. Saatavissa https://html.spec.whatwg.org/multipage/introduction.html#abstract