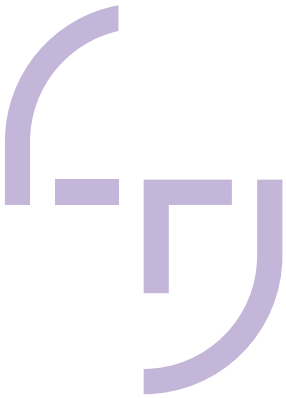
Tampereen ammattikorkeakoulu



Integraatiotestijärjestelmä automaatio-ohjaimelle

Sami Huopalainen

OPINNÄYTETYÖ
Marraskuu 2023

Tietotekniikka
Sulautetut järjestelmät ja elektroniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Sulautetut järjestelmät ja elektroniikka

HUOPALAINEN, SAMI,
Integraatiotestijärjestelmä automaatio-ohjaimelle

Opinnäytetyö 31 sivua, joista liitteitä 0 sivua
Marraskuu 2023

Automaatiojärjestelmissä esiintyy nykypäivänä entistä enemmän erilaisia ohjelmistototeutuksia. Tämä aiheuttaa järjestelmälle valtavan määrän yhteensopivuuksia rikkovia uhkia. Automaatio-ohjaimilla tämä on yleinen ongelma käyttöjärjestelmän ja automaatio-ohjelmistojen välillä. Vikojen ennaltaehkäisyyn tarvitaan testijärjestelmiä. Opinnäytetyön aiheena oli automaatio-ohjaimen integraatiotestijärjestelmä Valmet Automation Oy:lle.

Testijärjestelmäkokonaisuus on laaja kokoelma sekä ohjelmisto-, että fyysisiä osia. Automaatio-ohjaimen tapauksessa ympärille tarvitaan automaatiojärjestelmäympäristö, jossa ohjainta voidaan testata. Tämän automaatiojärjestelmän päälle rakennetaan eristetty testijärjestelmäkokonaisuus. Testijärjestelmään kuuluu ohjelmisto, työkaluja sekä fyysisiä laitteita. Tärkein ja suurin kokonaisuus näistä on ohjelmisto, jonka toteutukseen työ on painottunut. Ohjelmisto on jaettu selkeisiin kokonaisuuksiin, jotka sisältävät yksittäisiä testejä.

Opinnäytetyön tuloksena toteutui järjestelmä, joka toimii vakaana pohjana jatkokehitykselle. Järjestelmän toteutuksen aikana todettiin selkeitä epäkohtia, joiden korjaaminen on tulevaisuudessa tarpeellista. Järjestelmästä löytyi myös puutteita. Alustavaa pohdintaa näiden puutteiden korjaamiseksi on jo tehty. Järjestelmä on myös paikantanut kriittisiä vikoja ohjaimilla. Tämä antaa hyvän kuvan järjestelmän hyödyllisyydestä kehityksen apuna.

Asiasanat: testiautomaatio, integraatiotestaus, automaatiojärjestelmä, automaatio-ohjain

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Embedded Systems and Electronics

HUOPALAINEN, SAMI:
Integration Test System for Automation Controller

Bachelor's thesis 31 pages, appendices 0 pages
November 2023

Automation systems contain an increasing amount of software components nowadays. This causes a great amount of possible compatibility break points. On automation controllers this is a common problem between the operating system and the automation software. To prevent these problems, test systems are needed. The subject of this thesis was the integration test system for automation controller, and it was commissioned by Valmet Automation Oy.

A test system is a collection of software and physical components. In the case of automation controller, an automation environment is needed as a base. The test system is implemented in isolation on top of this environment. Test components include software, software tools and physical equipment. The most extensive and important one from these is the software which is also the main focus of this work. The software is divided into clear sets of tests.

The outcome of this thesis is a test system that will be a stable base for further development. Clear flaws were discovered while working with the software that need fixes in the future. Also, some lack of functionality was found. Initial planning was made for fulfilling these shortages. The test system has also located some critical bugs in the tested components which gives great picture of how useful it is for controller developing.

Key words: test automation, integration testing, automation, controller

SISÄLLYS

LYHENTEET

I/O	Kuvastaa laitetta, prosessia tai ohjelmaa, joka siirtää dataa laitteelta tai laitteelle
PRP	Parallel redundancy protocol. Verkkokahdennusprotokolla.
DAN	Dual access node, kahden väylän laite. PRP-protokollan termi.
SAN	Single access node, yhden väylän laite. PRP-protokollan termi.
RedBox	Redundancy box. PRP-laitetyyppi, joka kahdentaa verkkoliikennettä.
CI/CD	Ohjelmistokehityksen malli, jossa ohjelmistomuutokset ovat tiheitä. (engl. Continuous integration / continuous delivery.)
MTU	Tietoliikennepakettien maksimilähetyskoko. (engl. Maximum transmission unit)
MAC	Laitteen verkkoliitännän tunniste. (engl. Media access control)

1 JOHDANTO

Automaatiojärjestelmät ovat vuosien saatossa kehittyneet digitaalisempaan suuntaan ja nykyään ohjelmistot ovatkin jo pääroolissa ympäristön toiminnan kannalta. Etenkin teollisuuden automaatiojärjestelmästä voidaan löytää useita erillisiä ohjelmisto-osia ja laitteita, joilla niitä ajetaan. Ohjelmistojen ja laitteiden paljous aiheuttaa järjestelmän kehityksessä valtavan määrän mahdollisia vianaiheuttajia ja näiden mahdollisten vikojen ehkäisyyn tarvitaan testijärjestelmiä.

Tämä työ käsittelee teollisuuden automaatio-ohjaimelle tehdyn integraatiotestijärjestelmän toteutusta. Testit toteutetaan tukemaan automaatio-ohjaimen käyttöjärjestelmän kehitystä, sekä varmistamaan ohjaimen ja automaatio-ohjelmiston integraation säilyvyyttä. Testiympäristö toteutetaan fyysisillä laitteilla mahdollisimman luonnollisen testituloksen saavuttamiseksi. Työ painottuu testiohjelmistoon ja sen kehitykseen.

Työn tuloksena on tarkoitus tuottaa järjestelmä, jota voidaan kehityksessä käyttää ohjelmistojen hyväksymiseen ennen sen virallista jakelua. Työn ei ole tarkoitus luoda valmista tuotetta, vaan pikemminkin tuottaa ensimmäinen versio ja sen toimintaa kuvaava dokumentaatio.

2 TAVOITTEET JA VAATIMUKSET

Integraatiotestijärjestelmän tavoite on toteuttaa fyysisillä laitteilla kokonaisuus, jolla voidaan ajaa automaattisesti testejä automaatio-ohjaimille niiden käyttöjärjestelmän kehityksen tueksi. Nimensä mukaisesti järjestelmän päätehtävä on varmistaa ohjelmistojen välistä integraatiota, eli yhteentoimivuutta. Ohjelmistotestaukseen liittyy yleisiä tavoitteita ja esimerkiksi IEEE-standardin mukaan ohjelmistotestaus on prosessi, jossa ohjelmistotuotteen ominaisuuksia evaluoidaan vaatimusten mukaisesti mahdollisten puutteiden, vikojen, turvallisuuden, luotettavuuden ja suorituskyvyn osalta (IEEE 1059-1993 1996, 4). Testijärjestelmä vähentää manuaalisesta testaamisesta aiheutuvaa työtaakkaa, sekä nopeuttaa ohjelmistokehityksen aikana aiheutuvien virheiden paikantamista ja korjaamista. Järjestelmän pitkäaikaisena tavoitteena on toimia ohjaimen käyttöjärjestelmäversioiden validoimiseen ennen virallista julkaisua.

Testijärjestelmän täytyy fyysisesti sisältää vain laitteita, joita yhtiö toimittaa myös asiakkaille. Tämä takaa järjestelmän tuottamat tulokset mahdollisimman luotettaviksi. Nämä laitevaatimukset ovat määritetty yhtiökohtaisesti. Järjestelmän täytyy sisältää vähintäänkin kaikki ne ohjainmallit, jotka ovat aktiivisessa kehityksessä. Järjestelmän verkkototeutuksessa käytetään vain yhtiön yleisesti käyttämiä laitteita ja protokollia. Järjestelmän täytyy myös muilta fyysisiltä ominaisuuksiltaan mukailla toimitettavaa järjestelmää. Ohjaimissa käytetyt muistikortit ovat esimerkiksi vain niitä, joita yhtiö myös toimittaa. (Tervo 2022.)

Fyysisten ominaisuuksien lisäksi myös ohjelmistoon liittyy tavoitteita ja vaatimuksia, joiden täyttäminen on kokonaisuuden kannalta tärkeää. Testiohjelmiston tulee olla tehokasta, luotettavaa, sekä sen tulee kattaa ohjelmistotestaamiselle asetetut vaatimukset. Testiohjelmisto on syytä toteuttaa helposti skaalattavaksi mahdollisten vaatimusmuutosten tai -lisäyksien varalta. Testiohjelmiston tuottama tulosdata on oltava helposti luettavissa, ja siitä on saatava mahdollisimman helposti selville ilmentyvät virheet. Ohjelmisto on pyrittävä myös toteuttamaan mahdollisimman autonomiseksi. Tällä parannetaan järjestelmän tuottamaa hyötyä suhteessa siitä aiheutuvaan työhön.

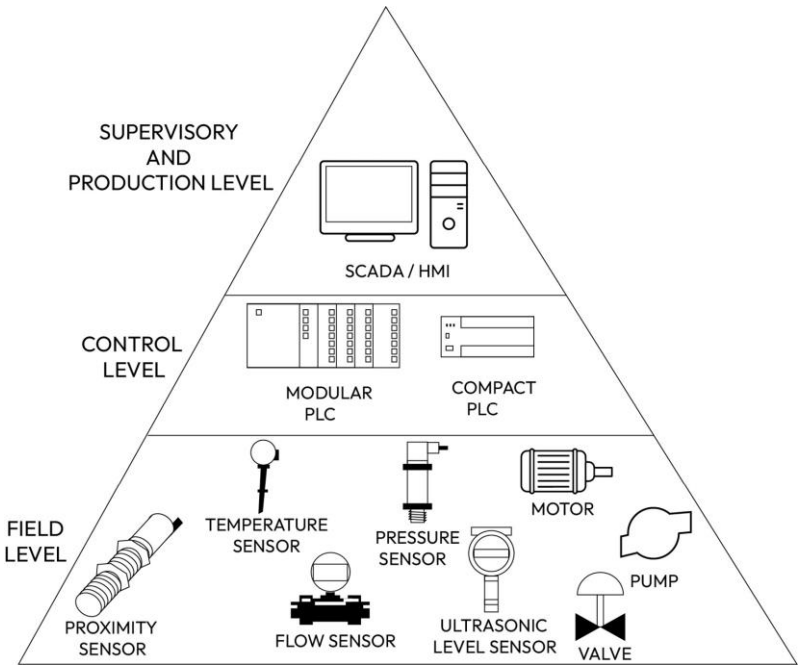
3 JÄRJESTELMÄ

Integraatiotestijärjestelmä vaatii toimiakseen sekä fyysisiä että ohjelmisto-osia. Järjestelmä voidaan jakaa yksinkertaisesti kahteen kokonaisuuteen, joista toinen osuus on automaatiojärjestelmä ja toinen on testijärjestelmä. Integraatiotestijärjestelmä ei varsinaisesti vaadi itselleen personoitua automaatiojärjestelmää, jonka päälle se voidaan toteuttaa. Testijärjestelmän optimoinnin kannalta tämä on kuitenkin järkevää. Automaatiojärjestelmä sisältää vaatimusten mukaiset ohjaimet, automaatiopalvelimen ja verkkolaitteet. Testijärjestelmäosiin taas kuuluu testiohjelmisto, jatkuvan integroinnin ratkaisut ja testijärjestelmän fyysiset osat, kuten verkkotestauslaite ja ohjattava verkkovirtakatkaisin.

3.1 Automaatiojärjestelmä

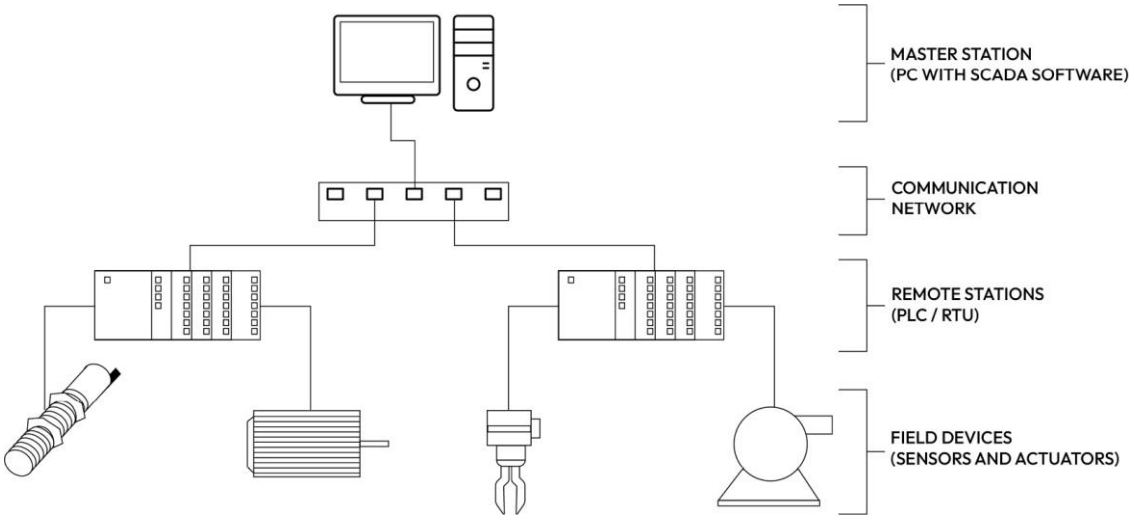
Teollisuuden automaatiojärjestelmään kuuluu parhaillaan valtava määrä laitteita, joilla on monia käyttötarkoituksia. Automaatio-ohjaimen tehtävä automaatiojärjestelmässä on tarjota laskentaa, tietojenkäsittelyä ja I/O:n eli datan sisään- ja ulostulon hallintaa. Automaatio-ohjaimet pystyvät toimimaan ytiminä tai ne voivat olla verkottuneita yhteen ja hajallaan järjestelmässä. (Lamb 2013, 3.1.)

Kuvio 1 esittää teollisuuden automaatiojärjestelmän perustasot pyramidimallina. Ylimpänä pyramidissa on hallintataso, jota seuraa ohjaustaso ja pohjalta löytyy kenttätaso. Kenttätason laitteita ei testijärjestelmään tarvita, koska integraatiotesteissä varmistetaan vain ohjaimen ja automaatio-ohjelmiston välistä toimintaa. Hallintatason laite järjestelmässä on nimeltään automaatiopalvelin. Tämä automaatiopalvelin toimii graafisena käyttöliittymänä järjestelmämäärityksiin sekä yleiseen hallinnointiin.



Kuvio 1. Automaatiojärjestelmän perustasot (Olushola, Akande 2023, 1).

Automaatiopalvelin on järjestelmän vastuulaite ja se toimii hyvin verkkopalvelimen tapaisesti tarjoten yhdistetyille laitteille automaatiopalvelun. Laite on myös hallinnallisesti voimakkain ja sen kautta järjestelmän ohjaaja tekeekin kaikki järjestelmäkonfiguraatiot. Automaatio-ohjain saa esimerkiksi toiminnallisuutensa automaatiopalvelimelta. Kuviosta 2 voidaan havaita, että järjestelmä tarvitsee myös verkkolaitteita. Nämä automaation verkkolaitteet poikkeavat perinteisen tietoliikenneverkon vaatimuksista.



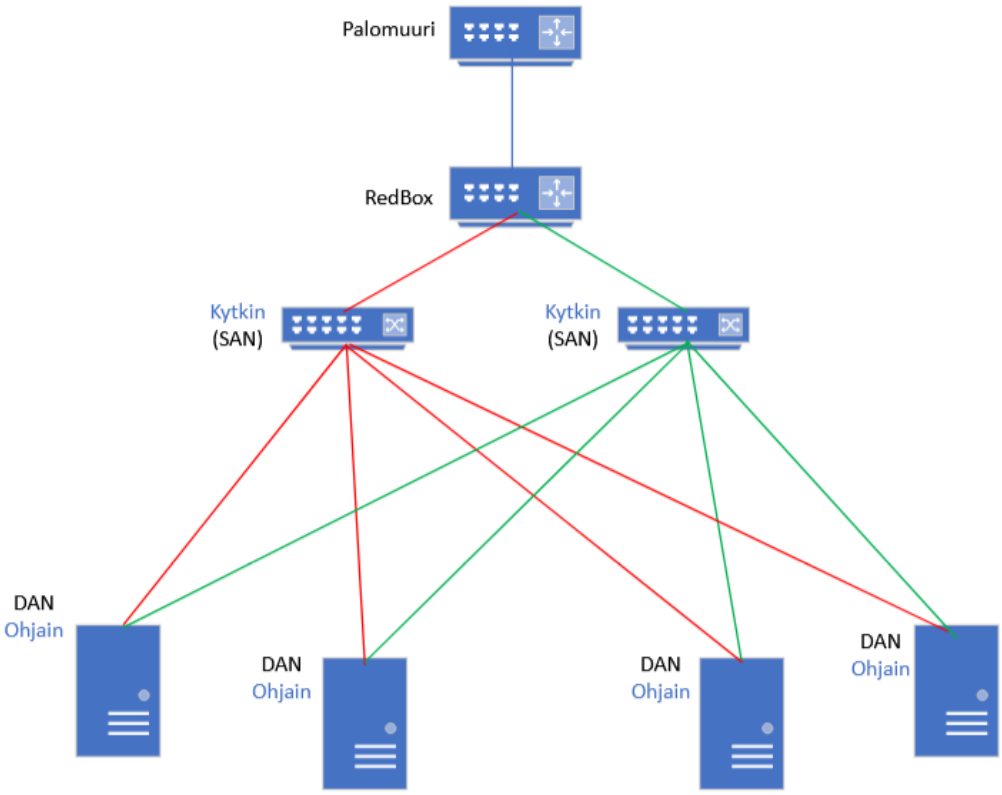
Kuvio 2. Yksinkertaistettu automaatioverkko (Olushola, Akande 2023, 11).

Automaatiojärjestelmän tietoliikenneverkko on huomattavasti vaativampi kuin perinteinen verkko. Kuten taulukosta 1 voidaan havaita, on vaatimukset hyvinkin tiukat. Testijärjestelmää varten vaatimukset ovat hieman taulukon vaatimuksia kevyemmät, mutta verkossa on käytettävä silti samoja protokollia ja fyysisiä laitteita vaatimusten mukaisesti. Automaatioverkoissa yleisesti käytettyyn verkkokahdennukseen käytetään testijärjestelmässä protokollaa nimeltä PRP (engl. Parallel redundancy protocol).

Taulukko 1. Perinteisen ja automaatioverkon vaatimukset (Galloway, Hancke 2013, 861).

	Automaatioverkko	**Perinteinen verkko**
Päätarkoitus	Fyysisten laitteiden hallinta	Tiedon prosessointi ja siirto
Sovellusalue	Prosessi- ja valmistusteollisuus	Koti- ja yritysympäristöt
Hierarkia	Syvä, toiminnallisesti erotettu hierarkia. Monta protokollaa ja fyysistä standardia	Olematon, integroitunut hierarkia yhtenäisellä protokollalla ja fyysisillä standardeilla
Vikaantumisen vakavuus	Korkea	Matala
Vaadittava luotettavuus	Korkea	Keskitaso
Vasteaika	250us-10ms	50+ ms
Ennakoitavuus	Korkea	Matala
Datan koostumus	Pieniä paketteja jaksollista ja jaksotonta liikennettä	Suuria ja jaksottomia paketteja
Ajallinen eheys	Vaaditaan	Ei vaadittu
Toimintaympäristö	Hankala ympäristö, pölyä, kuumuutta ja tärinää	Usein herkille laitteille tarkoitettu puhdas ympäristö

Verkkoprotokollaan liittyy testattavia asioita, joita testijärjestelmäkin käsittelee. Kuviossa 3 on esitelty protokollan topologiaa ja siihen on myös kirjattu tärkeitä termejä. Protokollassa testijärjestelmää varten tärkeintä on havaita, että ohjaimet kytketään fyysisesti kahteen väylään. Kahteen väylään kytketystä laitteesta käytetään termiä DAN (engl. Dual Access Node). DAN-laite lähettää verkkoliikennettä kahdennetusti molempia väyliään pitkin. Laite osaa myös vastaanottaa kahdennetun verkkoliikenteen. Termiä SAN (engl. Single Access Node) käytetään yhden väylän laitteesta, joka on kytketty kahdennetussa verkossa vain toiseen väylään. SAN-laitteen lähettämä liikenne voi verkossa kahdentua, jos se reitillään kulkee RedBox-laitteen (engl. redundancy box) kautta. RedBox on tärkeä protokollan osa ja sen tarkoitus on hallita liikennettä kahdennetun ja ei-kahdennetun verkon välillä. (Jussinmäki 2023, 10-11.)

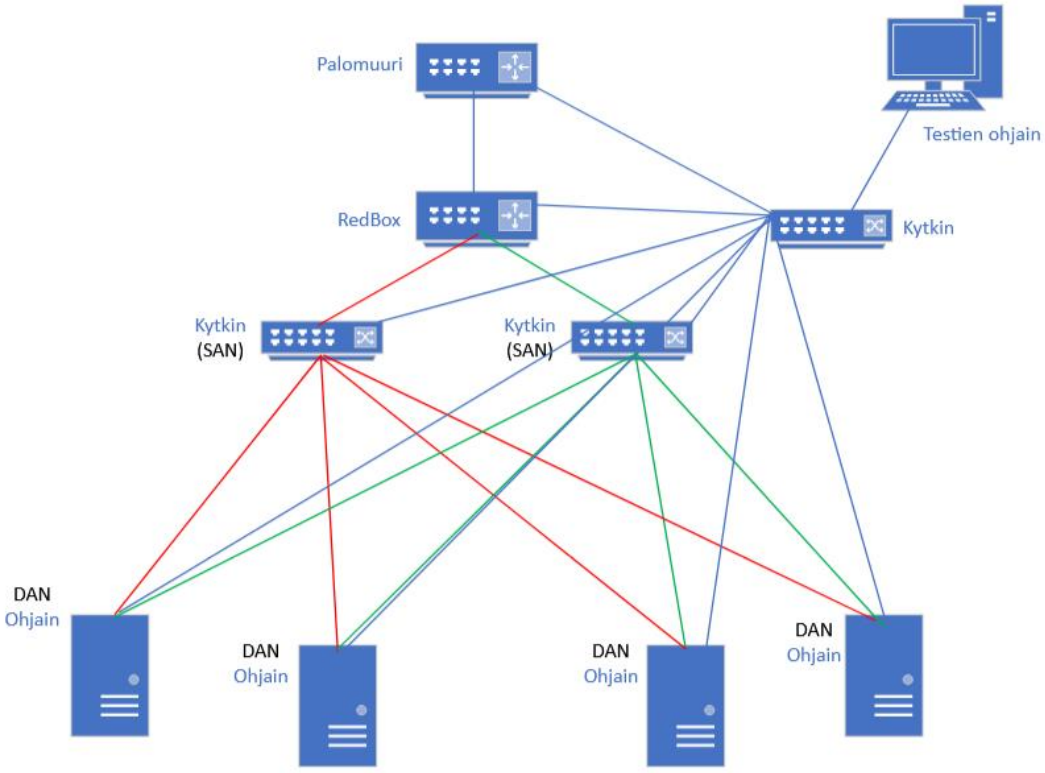


Kuvio 3. PRP topologia.

3.2 Testijärjestelmä

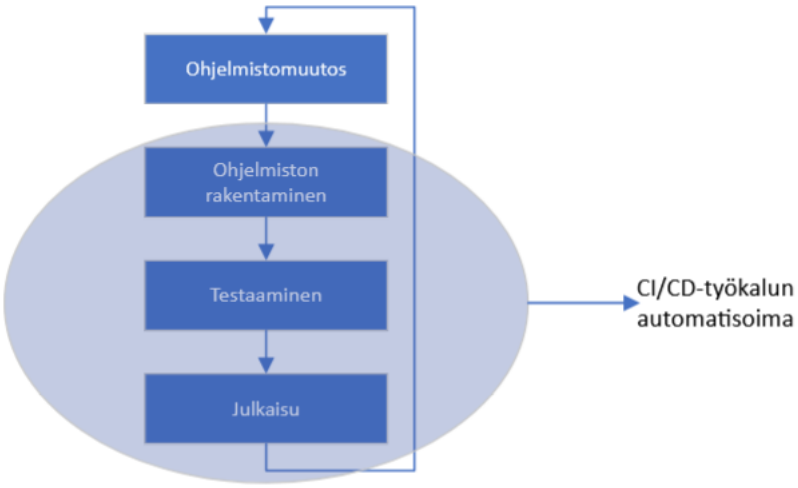
Testijärjestelmä on erillinen kokonaisuus, joka on luotu edellä mainitun automaatiojärjestelmän päälle. Testijärjestelmään, automaatiojärjestelmän tavoin, kuuluu fyysisiä laitteita ja ohjelmisto-osia. Testijärjestelmästä vastaa hallintalaite, joka on perinteinen tietokone. Tällä laitteella on monia tehtäviä järjestelmän toteutuksessa. Näistä tehtävistä tärkeimpänä laite suorittaa testiautomaatio-ohjelmiston ajamisen. Laite on fyysisesti yhteydessä kaikkiin järjestelmän ohjaimiin ja verkkolaitteisiin. Yhteys muihin laitteisiin on luotu omana verkkototeutuksenaan, joka on eristetty automaatioverkoista. Kuviossa 4 on visualisoitu testiverkko PRP verkon päällä. Testien aikainen yhteys hallintalaitteelta automaatio-ohjaimille ja muille laitteille muodostetaan tämän verkon kautta.

Testijärjestelmässä jatkuvan integroinnin ratkaisuna on niin sanottu CI/CD-työkalu. Lyhenne CI tulee englannin kielen sanoista ”continuous integration” ja CD sanoista ”continuous deployment” tai ”continuous delivery”. Suomeksi tämä tarkoittaa jatkuvaa integraatiota ja jatkuvaa toimitusta tai julkaisua. Työkalu auttaa ohjelmistokehityksessä automatisoimaan uusien ohjelmisto-osien versioiden rakentaminen, testaaminen sekä niiden julkaiseminen. (Synopsys 2017.)



Kuvio 4. Testijärjestelmän verkko PRP-verkon päällä.

Työkalua pyritään hyödyntämään jokaisen automaatiojärjestelmän ohjelmisto-osan kehityksessä. Toteutukset vaihtelevat, mutta pääsääntöisesti työkalun avulla ohjelmistoille ajetaan kuvion 5 mukaisia vaiheita. Työkalua käytetään myös jossain määrin yksittäisiin testiautomaatiotapauksiin. Integraatiotestijärjestelmälle tärkeänä ominaisuutena työkalu antaa myös mahdollisuuden säilöä testien tuloksia.



Kuvio 5. CI/CD-työkalun automatisoimat vaiheet.

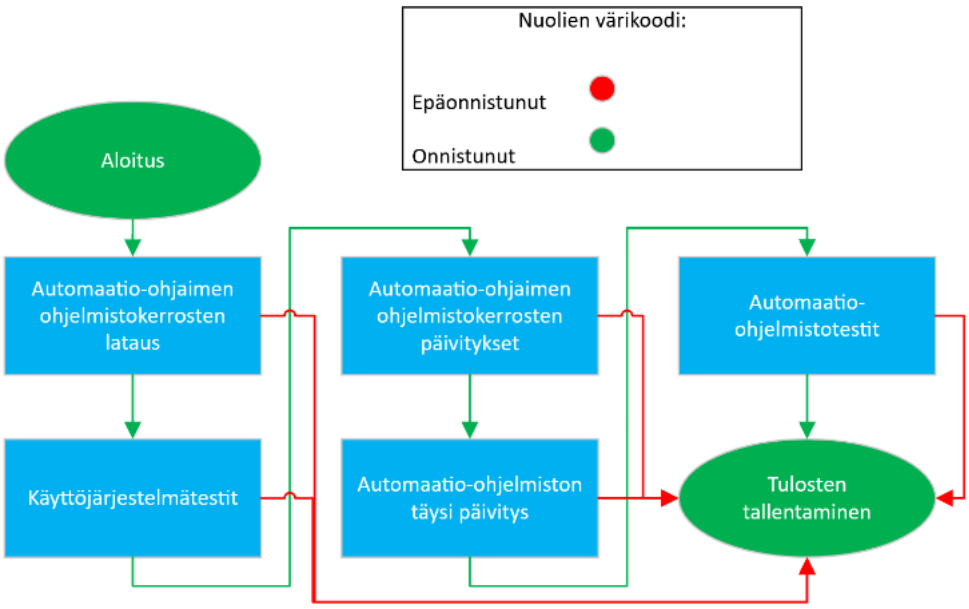
Testijärjestelmää tukee myös kaksi fyysistä laitetta. Laitteet ovat verkkoyhteyden kautta ohjattava verkkovirran katkaisija, sekä verkkotestauslaite. Katkaisijalla luodaan testien aikana simuloitu sähkökatkos. Verkkotestauslaite taas on tavallinen DAN-laite. Laitetta käytetään verkon toiminnan varmistamiseksi sekä suorituskykytestaukseen.

4 OHJELMISTO

Testijärjestelmän ohjelmisto toteuttaa järjestelmässä sen olennaisen tehtävän. Ohjelmisto on kehitetty testaamaan halutut toiminnallisuudet automaatio-ohjaimen käyttöjärjestelmästä, sekä integraatiosta automaatio-ohjelmiston kanssa. Ohjelmisto hoitaa myös testattavien ohjelmisto-osien lataamisen sekä testien tulosten kokoamisen.

Testiohjelmisto on helppo jakaa osioihin ja paikantamisen helpottamiseksi se on hyvin kannattavaa. Tähän jakoon on valittu selkeät pääalueet ja nämä alueet ovat jaettu testeihin, jotka käsittelevät yksittäisiä toiminnallisuuksia. Mitä tarkemmaksi jaottelun saa, sitä helpommin ja nopeammin löydetään virheenaiheuttaja testijärjestelmän tuottamasta tulosdatasta. Jos testit eivät ole tarpeeksi kohdennettuja, voi rikkovan ohjelmiston etsimiseen joutua käyttämään tarpeettoman paljon aikaa.

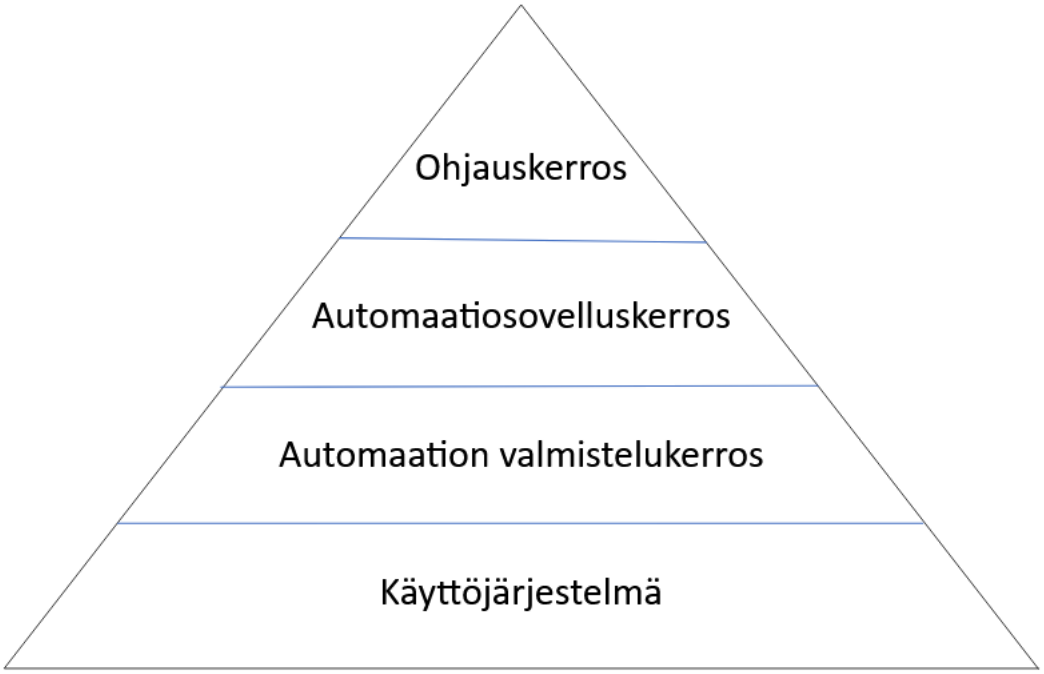
Itse ohjelmistototeutus on pyritty saamaan muotoon, jossa sen eteneminen on loogista ja selkeää. Tällä helpotetaan testien tulosraportin luettavuutta. Kokonaisuudet on jaettu kuvion 6 mukaiseen järjestykseen.



Kuvio 6. Testien kokonaissekvenssi.

4.1 Automaatio-ohjaimen ohjelmistokerrosten lataus

Automaatio-ohjaimen ohjelmistot voidaan jakaa neljään eri kerrokseen. Ohjelmistokerroksissa on kolme automaatio-ohjelmistoon kuuluvaa ohjelmistoa sekä käyttöjärjestelmä. Automaatio-ohjelmiston kerrokset toimivat käyttöjärjestelmän päällä kuvion 7 tapaan. Jokaisen ohjelmistokerroksen toiminta on riippuvainen sen alla olevasta ohjelmistosta.

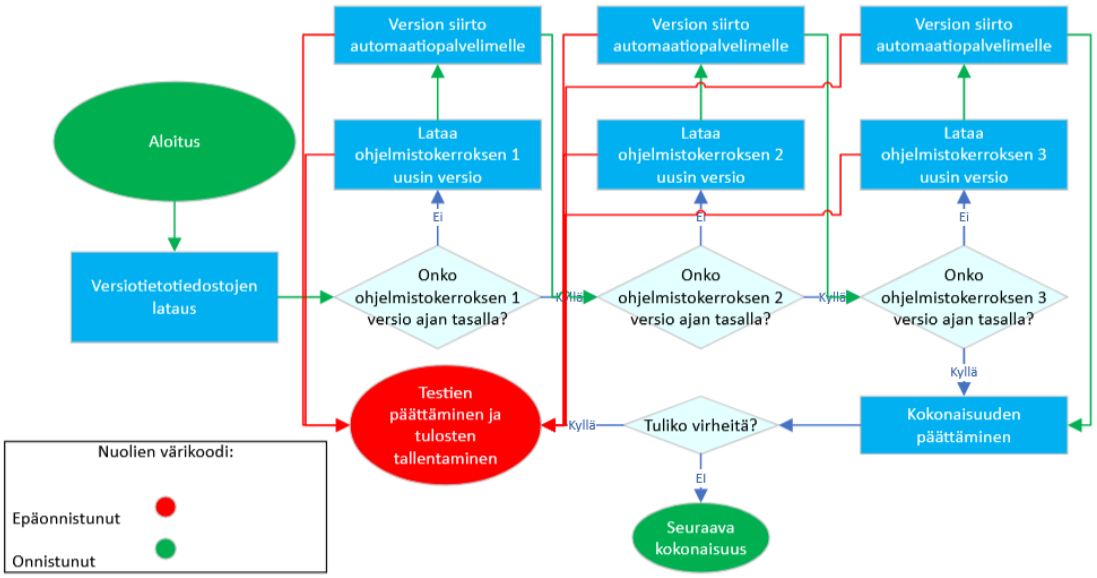


Kuvio 7. Automaatio-ohjaimen ohjelmistokerrokset.

Testijärjestelmän alussa ladataan verkosta automaatio-ohjelmiston kerrosten uusimmat versiot, jos niitä ei vielä löydy automaatiopalvelimelta. Uusimmat versiot saadaan ladattua CI/CD-työkalulla. Nämä kerrokset toteuttavat ohjaimella eri tehtäviä automaatioon liittyen.

CI/CD sivustolta löytyy myös ohjelmistokerroksiin liittyvä versiotietotiedosto. Tämä tiedosto sisältää itse ohjelmistokerroksen versiotiedon, sekä sen sisältävien ohjelmisto-osien versiotiedot. Näitä versiotietoja hyödynnetään testijärjestelmässä päivittämisen yhteydessä.

Testikokonaisuuksien etenemistä kuvataan vuokaavioilla samaan tapaan kuin kokonaisjärjestelmää on esitetty kuviossa 6. Ohjelmistojen latausosuuden toiminta on kuvattu kuviossa 8. Etenemistä voidaan kuvata tällä tavalla laajasti, mutta luettavuuden helpottamiseksi esitetään yleensä sekvenssistä kuitenkin vain yksittäiset testitapaukset.



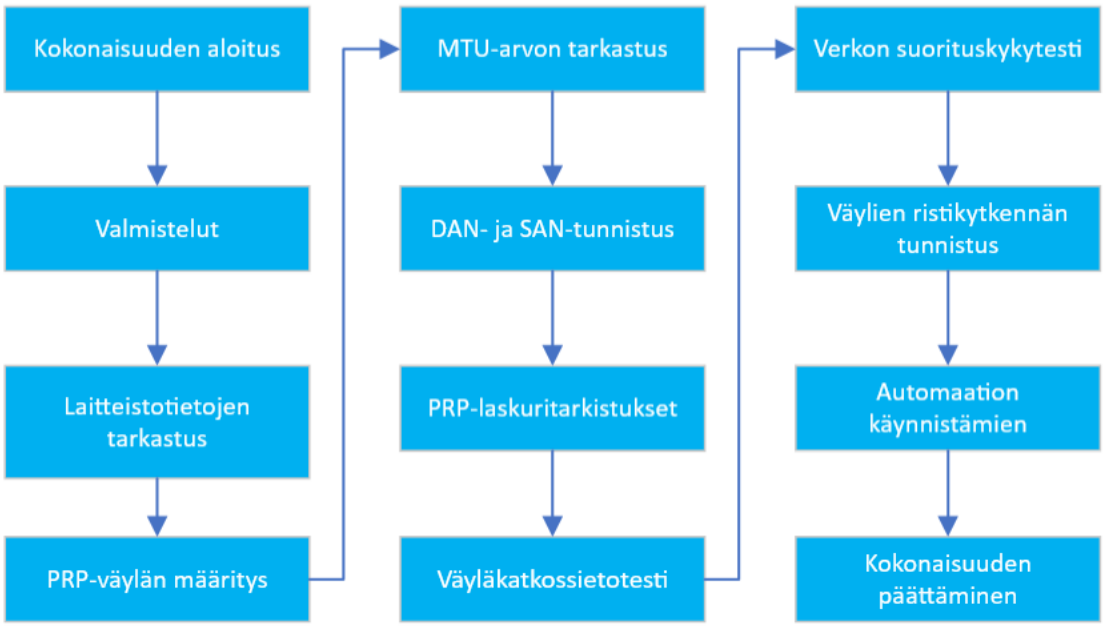
Kuvio 8. Ohjelmistokerrosten latauksen laaja esitysmalli.

Testien lisäksi kokonaisuuksissa tapahtuu myös asioita järjestelmän hallinnan helpottamiseksi. Ohjelmistojen lataus lopetetaan poistamalla ylimääräisiä ohjelmistoversioita automaatiopalvelimelta. Jos tätä ei tehdä, voi automaatiopalvelimen muisti täyttyä vanhoista ohjelmistoversioista ja aiheuttaa odottamattomia toimintahäiriöitä.

4.2 Käyttöjärjestelmätestit

Käyttöjärjestelmätesteissä on tarkoituksena varmistaa käyttöjärjestelmän perustoiminnallisuuksia. Perustoiminnallisuuksien testaaminen on järjestelmässä ensimmäinen varsinainen testikokonaisuus ja sen läpäiseminen onkin hyvin tärkeää ennen muita testejä. Perustoiminnallisuus testataan ohjaimelle ilman automaatio-ohjelmistoa. Jos ohjaimella on jo automaatio-ohjelmisto, voidaan se käynnistää uudelleen tilassa, jossa automaatio ei käynnisty. Käyttöjärjestelmätestit halutaan ajaa ohjaimella ilman automaatio-ohjelmiston käynnistymistä, jotta se ei vaikuttaisi negatiivisesti ohjaimen käyttöjärjestelmän testaukseen. Automaatio-ohjelmisto lisää muun muassa ohjaimen verkkoliikennettä huomattavan paljon, joka vaikeuttaa tässä kokonaisuudessa olevien verkkotestien ajamista.

Testit käyttöjärjestelmän osuudessa keskittyvät laajasti verkon toiminnallisuuksien varmistamiseen. Verkko on automaatiojärjestelmässä hyvin tärkeä osa, kuten taulukosta 1 voidaan päätellä. Verkkotesteissä testataan PRP-protokollan häiriönsietoa, virheentunnistuksia sekä verkon suorituskykyä. Nämä testit on määritetty PRP-testisuunnitelmassa (Stenvik 2021).



Kuvio 9. Käyttöjärjestelmätestien kuvaus.

Valmistelut alkavat lähes jokaisessa tulevassa testikokonaisuudessa yhteyden muodostamisella ohjaimelle. Tämän testikokonaisuuden alussa muodostetaan yhteydet myös verkossa oleviin kytkimiin, joiden avulla verkkotestejä suoritetaan. Toinen lähes jokaisessa testikokonaisuudessa oleva valmistelu on testeihin vaikuttavien muuttujien määrittely. Näiden muuttujien perusteella voidaan ohittaa tai suorittaa testejä tietyille ohjaintyypeille. Tiedot saadaan ohjaimelta tai automaatiopalvelimelta.

Laitteistotiedot, jotka tarkistetaan ohjaimelta, on tallennettu tiedostoon testijärjestelmän hallintalaitteelle. Yksinkertaisuudessaan katsotaan vain täsmäävätkö hallintalaitteen ja ohjaimen tiedot keskenään. Hallintalaitteen tiedot on pidettävä ajan tasalla käsin ohjaimien laitteistomuutosten yhteydessä. Tarkistus on syytä olla testikokonaisuudessa, mutta toteutus on herkkä virheille.

Verkkotestejä varten ohjaimelle määritetään PRP-väylä. Väylän määrityksen onnistuminen tarkistetaan katsomalla ohjaimen MTU-arvo (engl. Maximum Transmission Unit). MTU tarkoittaa verkkopakettien maksimilähetyskokoa ja sen arvo on bittejä (Cloudflare, N.D). PRP-protokollaa käyttäessä arvo on 1494 ja se paljastaakin helposti, onko määritys onnistunut.

SAN- ja DAN-tunnistus on vahvasti PRP-protokollan toimintaan liittyvä testi. Ohjaimen on tarkoitus tallentaa SAN- ja DAN-laitteet itselleen muistiin, jotta se osaa käyttää verkkoa tehokkaasti. Ohjain ei lähetä SAN-laitteelle tarkoitettua lähetystä kahdennetusti. Ohjaimen kuuluu löytää ja tunnistaa näitä yksiköitä verkossaan vastaanotettujen pakettien avulla. DAN-laitteet tunnistetaan verkkolähetyksestä niin sanotun PRP-trailerin avulla. SAN-laitteen lähetyksissä ei tätä traileria ole. Verkkopaketeista säilytetään lähettäjän osoitetietoja. Molemmille DAN- ja SAN-paketeille on olemassa oma taulukko, johon ne säilötään. Testi on toteutettu lähettämällä kysely tunnetulle SAN- ja DAN-laitteelle. Tämä testi varmistaa taulukkojen toiminnan lisäksi myös verkon toimintaa yleisesti.

PRP-laskureita on ohjaimilla paljon. Nämä laskurit ovat laajasti vastuussa kahdennuksen diagnostiikan tuottamisesta. Näistä laskureista kaksi tärkeintä on lähetettyjen ja vastaanotettujen pakettien laskurit. Lähetettyjä ja vastaanotettuja

paketteja lasketaan ohjaimilla etenkin vikojen toteamista varten. Laskureiden toiminta varmistetaan testeissä lähettämällä tunnettu määrä lähetyksiä verkkotestauslaitteelle. Verkon yleisen liikenteen takia laskureissa voi esiintyä pientä heiluntaa, joka on otettu huomioon laskureiden tarkastuksessa. Suuret heilunnat kertovat yleensä ohjaimen tai verkon viasta.

Väyläkatkoshäiriönsietotesti toteutetaan luomalla verkkokatkos ohjaimen ja RedBox-laitteen välille. Ohjaimen kuuluu toisen väylän katketessa säilyttää verkkoyhteys ilman katkoksia. Ohjaimen tulee myös tunnistaa vikatila. Väyläkatkos toteutetaan sulkemalla ohjaimen ja RedBox-laitteen väliseltä kytkimeltä yhteys. Verkkoyhteyden säilyvyys todetaan tämän jälkeen lähettämällä paketteja verkkotestauslaitteelle. Väyläkatkossa varmistetaan myös laskureiden toiminnan säilyvyys.

Verkon suorituskykytesti ei tässä kokonaisuudessa varsinaisesti mittaa ohjaimen suorituskykyä. Testin tarkoitus on rasittaa ohjaimen verkkoväylää maksimiteholla. Rasitustestin sivutuotteena saadaan tieto ohjaimen verkon suorituskyvystä. Suorituskykytesti on toteutettu ulkoisella ohjelmistolla ja se ajetaan verkkotestauslaitetta vasten. Verkkotestauslaite on suorituskyvyltään kaikkia ohjaimia tehokkaampi, joten se ei rajoita rasitustestiä. Suorituskyvyn tulokset ovat luettavissa testien päätyttyä lokista.

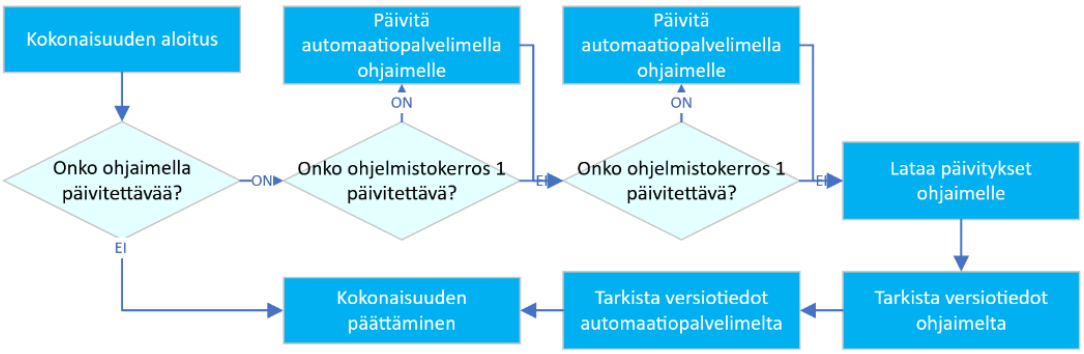
Väylien ristiinkytkentä on yleinen ja helppo virhe, joten sen tunnistaminen ohjaimella on tärkeää. Väyliä ei kuitenkaan voi kesken testien kytkeä fyysisesti ristiin. Ristiinkytkentää täytyy simuloida ohjaimella määrittämällä PRP-väylä uudestaan asettamalla väylät väärinpäin. Ohjaimelta vikatila voidaan todeta laskureista. Tätä varten ohjaimella on omat laskurit, jotka kasvavat aina kun ohjaimen väylä saa paketin väärällä väylätunnisteella.

Automaation käynnistyksessä on tässä kokonaisuudessa iso vaikutus sillä, onko ohjaimella automaatio-ohjelmisto ennestään. Jos ohjaimella ei ole automaatio-ohjelmistoa kokonaisuutta ajaessa, ladataan se sille automaatiopalvelimelta. Ohjelmisto täytyy ladata ensin testien hallintalaitteelle, josta ne siirretään ohjaimelle. Ohjelmistot on pakattu yhteen tiedostoon siirron helpottamiseksi. Siirron jälkeen

ohjain on käynnistettävä uudestaan, jotta automaatio-ohjelmisto käynnistyy. Tämä on ensimmäinen kohta, jossa integraatio voidaan havaita rikkinäiseksi.

4.3 Automaatio-ohjaimen ohjelmistokerrosten päivitykset

Automaatio-ohjaimen ohjelmistokerrosten päivitysosuudessa keskitytään niihin ohjelmistokerroksiin, jotka ovat päivitettävissä automaatiopalvelimelta. Näihin lukeutuu kuvion 7 pyramidin kaksi ylintä kerrosta. Kerrokset on pystyttävä päivittämään joko yhdessä tai erikseen. Testikokonaisuus voidaan myös tarvittaessa ohittaa kokonaan, jos päivitettäviä kerroksia ei sillä hetkellä ole.



Kuvio 10. Automaatio-ohjaimen ohjelmistokerrosten päivityssekvenssi.

Ohjelmistopäivitys alkaa tarkastamalla versiotiedot, jotta kokonaisuus ohitetaan tarvittaessa mahdollisimman nopeasti. Uusimmat versiotiedot saadaan ohjelmistoversioiden latausosuudessa hankituista versiotietotiedostoista. Näitä versiotietoja vertaillaan automaatiopalvelimella näkyviin ohjaimen versiotietoihin. Ajossa oleva versiotieto voitaisiin tarvittaessa tarkistaa myös ohjaimelta itseltään, mutta sen pitäisi olla sama palvelimella näkyvän kanssa.

Päivitykset tapahtuvat automaatiopalvelimelta siten, että siellä oleviin ohjaimen automaatiokonfiguraatioihin vaihdetaan uudet versiotiedot. Tiedot vaihdetaan vain kerroksilta, jotka päivityksen tarvitsevat. Versioiden päivittämisen jälkeen uusi konfiguraatio ladataan automaatiopalvelimelta ohjaimelle. Kerrosten asen-

nus on erilainen riippuen päivitettävistä kerroksista. Jos automaatiosovelluskerros päivitetään, aiheuttaa se ohjaimelle uudelleenkäynnistymisen. Pelkkä ohjauskerroksen päivittäminen ei saa ohjainta käynnistymään uudelleen.

Päivittämisen jälkeen on tärkeää tarkistaa, että versiot ovat päivittyneet oikein. Versiotiedot tulee tarkistaa ohjaimelta sekä automaatiopalvelimelta. Ohjaimen versiotiedot saadaan näkyviin tulostamalla käynnissä olevat prosessit. Tämä varmistaa version päivittymisen lisäksi myös sen, että automaatio-ohjelmisto on lähtenyt oikein käyntiin. Version tarkastus automaatiopalvelimelta lisää varmuutta onnistuneesta päivityksestä ja antaa myös tiedon siitä, että ohjaimen kommunikointi automaatiojärjestelmässä toimii oikein.

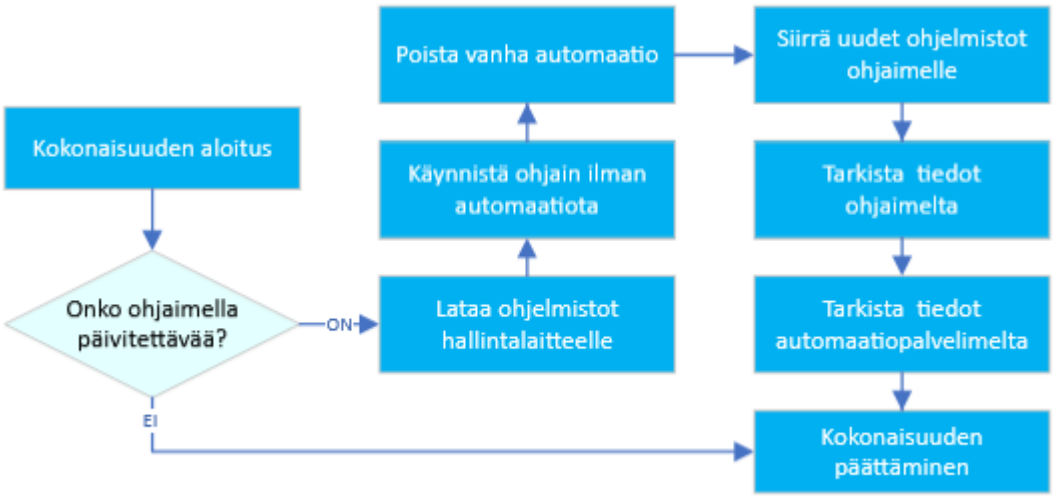
Virheet, jotka ilmenevät päivitysten yhteydessä, ovat yleensä hyvinkin kriittisiä. Pienemmät viat tulevat esille vasta testien viimeisessä kokonaisuudessa. Kaikkein yleisin vika on ohjelmistojen integraation menettäminen. Tämä havaitaan yleensä siitä, että ohjain menee uudelleenkäynnistyskierteeseen. Vika on kuitenkin myös suhteellisen helppo löytää, koska syyllinen on poikkeuksetta aina jompikumpi päivitetyistä kerroksista.

Päivitysosuuden viimeistelyyn kuuluu vielä vanhojen kerrosten versioiden poistaminen ohjaimelta. Poistamalla vanhat versiot ennaltaehkäistään ohjaimen hyvinkin rajallisen tallennustilan täyttymistä.

4.4 Automaatio-ohjelmiston täysi päivitys

Automaatio-ohjelmiston täysi päivitys on testijärjestelmän toinen päivityskokonaisuus. Tässä päivityksessä päivitettäviin osiin kuuluu nimensä mukaisesti koko automaatio-ohjelmisto. Järjestelmän testikokonaisuuksien järjestelyn takia tämä osuus ei kuitenkaan todellisuudessa päivitä kuin automaatio-ohjelmiston viimeisen kerroksen, jota edellisessä päivitysosuudessa ei voitu päivittää. Testijärjestelmä on siis mahdollista ajaa pelkästään tällä päivityksellä, jolloin koko automaatio-ohjelmisto päivittyisi yhdellä kertaa. Tässä kahden päivityksen kokonaisuu-

dessa on kuitenkin isoja hyötyjä. Automaatiopalvelimen kautta ohjaimen päivittäminen on kriittinen toiminnallisuus. Mahdollinen integraation rikkova kerros on myös helpompi paikantaa tämän jaon avulla.



Kuvio 11. Automaatio-ohjaimen ohjelmistojen täyden päivityksen sekvenssi.

Täyden päivityksen osuus on ajankäytöllisesti erittäin optimoitu. Siinä verrataan automaatio-ohjaimen automaation valmistelukerroksen versiota ohjaimella luvussa 4.1 ladattuun versiotietotiedostoon. Päivityksiä aletaan tehdä vain tarvittaessa. Tämä ohitus on ajankäytöllisesti hyvin tarpeellinen päivityksen ollessa pitkäkestoinen.

Päivitystarpeen toteamisen jälkeen voidaan alkaa suoraan päivittämään ohjainta. Automaation valmistelukerros nimittäin päivittää automaatiopalvelimelle siirtäessä versiotietonsa itsenäisesti. Ohjaimen päivityksen tarpeesta voidaan siis päätellä, että uudet versiotiedot löytyvät jo palvelimelta päivitystä aloitettaessa. Täydessä ohjelmistopäivityksessä ohjelmistokerrokset ladataan ohjaimelle yhtenä tiedostona automaatiopalvelimelta testien hallintalaitteen kautta. Samaa latausmenetelmää käytetään luvussa 4.2 automaation käynnistystestissä.

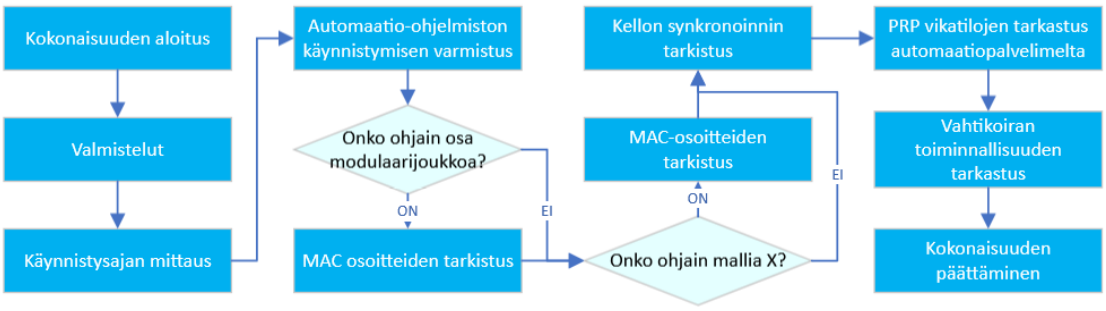
Ohjaimella on tämän päivityksen aikana automaatio-ohjelmisto jo käynnissä. Ohjelmistoa ei voi kuitenkaan sammuttaa käynnissä olevalta ohjaimelta päivittämistä varten. Ohjain on siis ensiksi käynnistettävä käyttäen automaatio-ohjelmistotonta tilaa. Tätä käynnistysmuotoa käytettiin jo myös luvussa 4.2. Ohjaimelta

voidaan huolettomasti poistaa koko automaatio-ohjelmisto, kun se on uudelleenkäynnistynyt. Päivityksen päättää ohjelmiston siirto hallintalaitteelta ohjaimelle.

Päivitetyn ohjaimen versiotiedot tarkistetaan vielä ohjaimelta itseltään ja automaatiopalvelimelta. Tässä vaiheessa tarkistetaan jokainen ohjelmistokerros. Tämä ei käytännössä ole välttämätöntä, kun päivityksen kohteena oli vain automaation valmistelukerros. Tarkastus ei kuitenkaan vie paljon aikaa, eikä versioiden sekoittuminen ole ollenkaan mahdotonta.

4.5 Automaatio-ohjelmistotestit

Automaatio-ohjelmiston testauksessa varmistetaan ohjaimen perustoiminnallisuutta käyttöjärjestelmän ja automaatio-ohjelmiston integraatiossa. Käynnistyksen yhteydessä automaatio-ohjelmisto mukauttaa ohjaimen asetuksia automaation ajoa varten. Nämä konfiguraatiot on luotu automaatiopalvelimella ja ne säätävät esimerkiksi ohjaimen verkkoasetuksia. Tämän lisäksi automaatio-ohjelmisto hyödyntää ohjaimen tietoja automaatiopalvelimella esimerkiksi vikatilojen havaitsemiseen. Näiden vikatilojen havaitseminen on järjestelmänvalvonnan kannalta kriittinen ennaltaehkäisevä toiminto.



Kuvio 12. Automaatio-ohjelmistotestien sekvenssi.

Käynnistysajan mittausta varten käytetään kappaleessa 3.2 mainittua verkkovirran katkaisijaa. Katkaisijalla luotu verkkovirran sulkeminen testaa samalla myös ohjainten sähkökatkonsietokykyä. Käynnistysaika mitataan verkkovirran päälle laittamisesta ohjaimen kanssa yhteyden muodostamiseen. Aikaa verrataan toi-

vottuun käynnistymisaikaan ja jos käynnistyminen on kestänyt toivottua pidempään, annetaan testijärjestelmän lokiin varoitus käynnistymisajan pituudesta. Käynnistysajan mittaamisella voidaan analysoida ohjelmiston suorituskyvyn kehittymistä. Uudelleenkäynnistämisen jälkeen varmistetaan automaatio-ohjelmiston normaali käynnistyminen. Käynnistys varmistetaan hakemalla ohjaimen prosesseista automaatio-ohjelmistoja.

MAC-osoitteiden tarkistukset liittyvät testijärjestelmässä ohjaimilla käytettyyn apuohjelmaan. MAC-osoite on laitteen verkkoliitännälle annettu tunniste (Geeksforgeeks 2023). Apuohjelma on vastuussa kolmella ohjaintyypillä sisäisen kytkinkortin ohjaamisesta ja ilman sitä verkon kahdennus ei toimisi oikein. Molemmissa testeissä tarkistetaan ohjaimen MAC-osoitteita hieman eri tavoin. Apuohjelma käsittelee ohjaimilla tätä osoitetta ja jättää näin toiminnastaan jäljen. Nämä molemmat testit ohitetaan ohjaimilla, joissa apuohjelmaa ei hyödynnetä.

Automaatiojärjestelmälle on tärkeää, että järjestelmän laitteiden kellot on synkronoitu keskenään. Kellot synkronoidaan aina erikseen päätetyn laitteen kellon mukaan ja testijärjestelmässä tämä rooli on määritelty automaatiopalvelimelle. Tietoverkkojen, kuten monien muidenkin tiedonkeräyksessä hyödynnetään aikaleimoja tapahtumaraporteissa, joita taas käytetään vikatilanteiden purkamisessa. On tärkeää, että näiden lokitietojen välillä aikaleimat ovat keskenään verrattavissa. Tämä helpottaa lokien analysointia.

PRP-vikatilojen tarkastuksessa simuloidaan muutama verkkovirhetila ja tarkistetaan automaatiopalvelimelta, että se saa tiedon näiden virhetilojen syntymisestä. Testattavat vikatilat ovat väyläkatkos sekä ristikytkentä. Vikatilat simuloidaan tässä testissä samalla tavalla kuin käyttöjärjestelmätesteissä. Väyläkatkoksessa siis suljetaan kytkimeltä väylät, joihin ohjaimet ovat liitetty ja ristikytkennässä määritellään verkkoväylät ristiin. Ohjain lähettää automaatiopalvelimelle tiedon näistä vikatiloista kokonaislukutunnisteella.

Vahtikoiratesti viittaa ajastimeen, jonka tarkoitus on uudelleenkäynnistää laite vikatiloissa (Stajano, Anderson N.D). Vikatilat on määritetty ohjelmistoon ja ne ovat yleensä hyvin kriittisiä vikoja, kuten automaatio-ohjelmiston odottamaton sulkeutuminen. Automaatio-ohjelmiston odottamaton sulkeutuminen onkin juuri se, jolla

tämä testi toteutetaan. Ohjaimelle annetaan automaatio-ohjelmisto prosessin ”tappokomento”, jolloin automaatio-ohjelmisto sulkeutuu ja ohjaimen pitäisi uudelleen käynnistyä vahtikoiratoiminnon avulla. Uudelleenkäynnistymisen jälkeen varmistetaan vielä, että ohjain käynnistyy automaatio-ohjelmiston kanssa normaalisti.

5 JATKOKEHITYS

Testijärjestelmä on kuin mikä tahansa muukin järjestelmä eli se vaatii jatkuvaa kehitystä. Kehitystä voidaan toteuttaa testijärjestelmässä moneen osa-alueeseen ja niistä suurin kehityksen kohde on testiohjelmisto. Testijärjestelmää voi kuitenkin kehittää myös muuttamalla sen fyysisiä ominaisuuksia, kuten lisäämällä testattavaa laitteistoa. Testijärjestelmän kehityksessä pitää ottaa tarkasti huomioon ohjaimien ja automaatio-ohjelmiston kerrosten kehitys, koska niissä tapahtuvat uudistukset voivat rikkoa testijärjestelmän toimintoja. Nämä saattavat ilmetä testijärjestelmää ajaessa, mutta on myös täysin mahdollista, että niitä ei havaita järjestelmällä ollenkaan. Esimerkkinä järjestelmän mahdollisesta rikkojasta on käyttöjärjestelmätesteissä käytetty laitteistotietolistaa, joka on päivitettävä aina käsin muutosten yhteydessä. Virhe pystytään havaitsemaan järjestelmällä itsellään tehden siitä vähemmän kriittisen, mutta kehityksessä tulisi silti pyrkiä tilaan, jossa ei ole manuaalista päivittämistä.

Testijärjestelmän ohjelmiston kehitys vaatii laajaa yhteistyötä kaikilta niiltä kehitysryhmiltä, joiden ohjelmistoa järjestelmä testaa. Itse testijärjestelmän kehittäjiltä on vaikea vaatia ymmärrystä jokaisesta ohjelmistosta, jonka toimintaa testit varmistavat. Tämän takia jokaisen kehitysryhmän tulisi määrittää mahdollisia tarpeitaan testijärjestelmälle ja kertoa myös kriittisistä toiminnallisista muutoksista testijärjestelmän kehittäjille. Tämä antaa aikaa reagoida suuriin muutoksiin, jotka mahdollisesti vaikuttavat negatiivisesti testijärjestelmään. Parhaimmillaan testijärjestelmän kehitys vaatii huomattavan vähän aikaa verrattuna itse kehitettäviin tuotteisiin.

5.1 Testiohjelmistokehitykset

Järjestelmän tavoitetila on se, että mitään ohjelmisto-osia ei tarvitsisi päivittää manuaalisesti. Tämän tavoitteen estää nykyisessä testijärjestelmän tilassa ohjaimen käyttöjärjestelmä, jolle ei ole luotu automaattista päivitystä. Oh-

jaimen käyttöjärjestelmä on ohjelmisto, jota ei ole tarkoitettu nykyisessä muodossaan täysin päivitettäväksi. Käyttöjärjestelmän päivittäminen siis vaatisi testijärjestelmään vähintäänkin uuden käyttöjärjestelmäversion latauksen, testikokonaisuuden sekä mahdollisesti muutoksia itse ohjaimen käyttöjärjestelmään. Käyttöjärjestelmämuutoksien kanssa päivittämisen helpottamiseksi täytyy kuitenkin olla erityisen varovainen. Toteutus on nimittäin helposti altis tietoturvariskeille.

Testijärjestelmässä ei ole lopulta kovinkaan suurta määrää integraatioon liittyviä testejä. Integraation säilyvyys on testeistä silti helposti todettavissa siitä, käynnistyykö automaatio-ohjelmisto ohjaimella. Integraation laajempi testaaminen voisi silti olla ajankäytöllisesti kannattavaa ja integraatiotestijärjestelmään helposti toteutettavaa. Laajempaa testikokonaisuutta varten täytyisi tehdä alustava suunnitelma siitä mitä halutaan testattavaksi. Tätä suunnittelua varten tarvittaisiin automaatio-ohjelmistokehittäjien ja testijärjestelmäkehittäjien yhteistyötä.

5.2 Versioyhteensopivuustestaus

Jatkokehityksessä laaja osa-alue on versioyhteensopivuustestaus. Automaatiojärjestelmässä laitekokonaisuudet ja laitteiden versiot voivat vaihdella laajastikin. Yleensä yhtiöt lupaavat laitteiden versioille yhteensopivuutta julkaisuversioiden välille etenkin taaksepäin. Tämä yhteensopivuuslupaus voi olla mitä vain yhdestä kaikkiin julkaisuihin, mutta yleisesti ottaen numero on mahdollisimman pieni. Näin saadaan asiakkaat päivittämään laitteistot uudempiin versioihin, sen ollessa yleisesti tehokkuuden ja turvallisuuden kannalta tärkeää. Monet yhtiöt tekevät tätä myös rahan takia.

Versioiden välinen testaus ei itse testijärjestelmän ohjelmistolta pitäisi vaatia suuria muutoksia, riippuen siitä millaista tapaa käytetään versioiden väliseen testaukseen. Käytännössä tähän on ainakin kaksi vaihtoehtoa. Joko versioita vaihdellaan aktiivisesti testijärjestelmässä ohjelmistollisesti, tai testijärjestelmässä kasvatetaan fyysisten laitteiden määrää. Suurin vaikeus järjestelmässä on hyvinkin todennäköisesti automaatiopalvelimen julkaisuversion vaihtaminen, jos sellainen

vaaditaan testattavaksi. Laitteen päivittäminen on työlästä ja se ottaa huomattavan paljon aikaa jopa fyysisesti. Ohjelmallisesti aika luonnollisesti kasvaa tiedonsiirron vaikeutuessa. Fyysisen järjestelmän kasvattaminen tuo taas muita ongelmia esiin. Suurin estävä tekijä voi hyvinkin olla fyysisten laitteiden heikko saatavuus.

Versioiden välisen testauksen toteuttamiseksi täytyy ensimmäisenä määritellä, mitä kaikkia ohjelmisto-osia ja laitteita haluaan testata. Kyseisen testijärjestelmän tapauksessa minimi tälle olisi todennäköisesti automaatio-ohjelmiston julkaisuversioiden ristitestaus käyttöjärjestelmän julkaisuversioiden kanssa. Toinen määritettävä tekijä testausta varten on se, kuinka monta versiota taaksepäin jokaisesta ohjelmistosta halutaan testattavaksi.

Versiotestaus on siis kokonaisuutena iso ja sen suunnitteluun on syytä käyttää aikaa. Versiotestauksella voi huonosti suunniteltuna aiheuttaa joko suuria kustannuksia tai ajanhukkaa järjestelmässä. Versiotestausta voisi jopa ajatella toteutettavaksi täysin erillisenä testijärjestelmänä. Integraation testaus ristiin voitaisiin esimerkiksi suorittaa integraatiotestin jälkeen niille versioille, jotka ovat testin läpäisseet.

6 POHDINTA

Työn tuloksena toteutunut järjestelmä on vakaa pohja jatkokehitettäväksi. Järjestelmän toteutus on helposti skaalattavissa ja sen tämänhetkinen testikattaus on jo laajuudeltaan riittävä. Järjestelmä vaatii kuitenkin selkeästi jatkokehittämistä ja nykyisessä tilassaan sen käyttö vaatii vielä tarpeettoman paljon fyysistä työtä. Kun järjestelmästä automatisoidaan vielä käyttöjärjestelmän päivitys, lähentelee se täysin autonomista järjestelmää.

Järjestelmä ei vielä täytä tarvetta käyttöjärjestelmäohjelmiston julkaisujen hyväksyttämisestä. Järjestelmässä ei esimerkiksi käydä vielä läpi kaikkia PRP:n testaussuunnitelman testejä. Automaatio-ohjelmiston ja käyttöjärjestelmän integraatiota varten taas pitäisi määrittää selkeät vaatimukset. Näiden pohjalta testien tulokset on helpompi todeta riittäviksi tai riittämättömiksi.

Järjestelmän kehityksen aikana on onnistuttu tunnistamaan kriittisiä virheitä automaatio-ohjaimelta. Nämä kriittiset virheet on havaittu käyttöjärjestelmästä sekä automaatio-ohjelmiston integraatiosta. Osa vioista pystyttiin toteamaan ohjelmistosta, joka on ollut jo pitkään olemassa. Muutama vika tuli ilmi myös aktiivisesta kehityksestä. Virheiden löytäminen kertoo ohjelmistokehityksessä selkeästi puuttuvasta testauksesta, jota tällä järjestelmällä täytetään.

Jatkokehitystä varten testijärjestelmälle on tehty hyvä alustava suunnittelu ja pohdinta tärkeimpien kehityskohteiden osalta. Tämän alustavan suunnitelman pohjalta voidaan järjestelmää kehittää eteenpäin vaivattomasti nykytilastaan. Jatkon kannalta on myös otettu kantaa mahdolliseen kehitysmallin kehittämiseen. Kehitysmallia voidaan hyödyntää tulevaisuudessa esimerkiksi järjestelmän resurssisuunnittelussa.

LÄHTEET

[1] Tervo. ”AEP integration/compatibility testing environment,” Sisäinen dokumentaatio. Valmet Automation Oy, 2022.

[2] IEEE std 1059-1993 "IEEE Guide for Software Verification and Validation Plans," https://ieeexplore-ieee-org.liproxy.tuni.fi/stamp/stamp.jsp?tp=&arnumber=838043

[3] Lamb, Frank. 2013. “Components and Hardware.” Chap. 3 in Industrial Automation: Hands-On. 1st ed. New York: McGraw-Hill Education. https://www-accessengineeringlibrary-com.libproxy.tuni.fi/content/book/9780071816458/chapter/chapter3

[4] Olushola, Akande. 2023. “Industrial Automation from Scratch”

[5] B. Galloway and G. P. Hancke, "Introduction to Industrial Control Networks," in *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 860-880, Second Quarter 2013, Doi: 10.1109/SURV.2012.071812.00124.

[6] Rossel, Sander. “Continuous Integration, Delivery, and Deployment *:* Getting Started with the Processes and the Tools to Continuously Deliver High-Quality Software*”*, Packt Publishing, Limited, 2017. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=5117840.

[7] Stenvik. ”PRP network driver – test plan,” Sisäinen dokumentaatio. Valmet Automation Oy, 2021.

[8] Synopsys. “CI/CD” https://www.synopsys.com/glossary/what-is-cicd.html#:~:text=CI%20and%20CD%20stand%20for,are%20made%20frequently%20and%20reliably.

[9] Cloudflare. ”What is MTU?,” https://www.cloudflare.com/en-gb/learning/network-layer/what-is-mtu/

[10] Stajano. Frank. Anderson. Ross. “The Grenade Timer:Fortifying the Watchdog Timer Against Malicious Mobile Code”, https://www.cl.cam.ac.uk/~rja14/Papers/grenade.pdf

[11] Jussinmäki, M. 2023. Kahdennetun ja ei-kahdennetun tietoverkon välisen siirtymän toteutus Linuxilla. Diplomityö. https://trepo.tuni.fi/handle/10024/150663

[12] Geeksforgeeks. “What is MAC address?,” https://www.geeksforgeeks.org/mac-address-in-computer-network/