SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TESTILAITTEEN SUUNNITTELU JA TOTEUTUS HIOMAKONEEN OHJAUSJÄRJESTELMÄN KEHITYSTÄ VARTEN

TEKIJÄ/T: Daniel Harjunheimo

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Sähkötekniikan tutkinto-ohjelma	
Työn tekijä(t) Daniel Harjunheimo	
Työn nimi Testilaitteen suunnittelu ja toteutus hiomakoneen ohjausjärjestelmän kehitystä varten	
Päiväys 25.02.2020	Sivumäärä/Liitteet 84/12
Ohjaaja(t) TkL Pekka Väänänen ja Arto Toppinen	
Toimeksiantaja/Yhteistyökumppani(t) RollTest Measurements & Controls / Mika Purhonen	

Tiivistelmä

Paperi ja metalliteollisuudessa liikkuu suuri määrä investointeja. Isot tuotantolinjat vaativat tehokasta ja hyvin suunniteltua huoltojärjestelmän, jotta tuotantolinjan käyttöastetta voitaisiin pitää mahdollisimman korkealla. Näillä teollisuudenaloilla liikkuu suuri määrä teloja ja valsseja, jotka ovat merkittävässä roolissa eri tuotantolinjoja ja niiden koneita. Niille on asetettu kovat vaatimukset sekä käytönaikana että huoltotoimenpiteissä. Luonnollisen kulumisen ja monen muun tekijän summana, telat ja valssit voivat kulua hyvinkin nopeasti huolettavaan kuntoon. Tämä on tuonut tarpeen luoda tehokkaan ja luotettavan tavan huoltaa teloja ja valsseja, jotta haluttu laatu pystyttäisiin ylläpitämään mahdollisimman tehokkaasti. Hyvin suunnitellulla ja valmistetulla telahiomakoneella voidaan hioa nopeasti ja tehokkaasti toleranssiarvojen mukainen työkappale, jolloin telojen ja valssien osalta käyttöaikaa voidaan maksimoida.

Työssä käsitellään yleisesti telahiomakoneen rakennetta, erilaisia teloja ja siihen liittyvää termistöä, mittalaitetta ja telageometriaa. Työn suunnitteluvaiheessa tutkitaan erilaisia ohjelmistotyökaluja, menetelmiä ja tietokanta hallintajärjestelmiä sovelluskehityksessä, joita voidaan hyödyntää tämän tyyppisissä järjestelmissä. Olennainen osa ohjausjärjestelmää on myös automaatio-osio, johon kuuluu ohjelmoitavat logiikat ja moottorinohjausyksiköt. Työssä tutustutaan myös OPC UA-standardin, ja tutkitaan kuinka sitä voisi soveltaa tämänkaltaisissa järjestelmissä, missä pitää saada PLC-laitteen ja käyttöliittymäsovelluksen välille tietoliikenneyhteys rakennettua.

Työn tuloksena syntyi ohjausjärjestelmän prototyyppi, jolla voidaan ottaa yhteyttä PLC-laitteeseen käyttöliittymäsovelluksen avulla. Käyttöliittymäsovelluksella voidaan ottaa mm. yhteyttä tietokantaan ja visualisoimaan PLC-laitteelta tulevia mittaustietoja graafeiksi. PLC-laitteen ja tietokoneen välinen yhteys saatiin muodostettua OPC UA-standardin pohjalta, hyödyntämällä Siemensin UAClientHelperAPI-apuluokkaa.

Lopputuloksena saatiin rakennettua yksinkertainen, mutta selkeä kokonaisuus ohjausjärjestelmästä. Lopputyö tuo hyvän pohjan tuotekehitysprojektin jatkamiselle, missä voidaan soveltaa esimerkiksi lopputyössä tehdyn käyttöliittymäsovelluksen koodirakennetta ja yhteydenluontimenetelmiä.

Avainsanat
Telahiomakone, PLC, OPC UA, Windows Forms, Käyttöliittymäsovellus, PostgreSQL, ODBC, TIA Portal

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Electrical Engineering			
Author(s) Daniel Harjunheimo			
Title of Thesis Design and implementation of a test device for the development of a grinding machine control system			
Date	25 February 2020	Pages/Appendices	84/12
Client Organization /Partners RollTest Measurements & Controls Oy. Ltd			

Abstract

The purpose of this thesis was to create a simple control system prototype with a graphical user interface. The thesis was done by studying the following points: basic structure of grinding machine, what kind of rolls are used in production lines, what basic terms are used with rolls, the basic measuring device for rolls and basics of roll geometry. This thesis followed the plan-do-check-adjust procedure.

First, in the planning section, different databases, programming tools and ways to establish a communication between the graphical user interface and the programmable logic controller were studied. After the planning procedure, the implementation started which included the creation of the graphical user interface and establishing a communication connection between different devices and machines. The checking procedure contained different types of evaluation methods, with which the results were presented to the client and counterproposals were received. Finally, an adjustment procedure was carried out where counterproposals were evaluated and then implemented in the beginning of a new plan-do-check-adjust procedure.

The work resulted in a prototype control system. This system can establish a connection between the PLC and the graphical user interface, where data and control commands can be exchanged between each other. The connection supports the OPC UA standard and was created by using Siemens's UAClientHelperAPI class helper. The graphical user interface also supports communication between the database and converts measurement data into a graph that can be used in the grinding procedure.

As a result of this thesis, a proper foundation for continuing an official product development project was established the controlling system and the graphical user interface enabled all the necessary tools for further development. For example, all the tools for creating the connection between different components can be found in the code of the graphical user interface.

Keywords
roll grinding machine, PLC, OPC UA, Windows Forms, UI application, PostgreSQL, ODBC, TIA Portal

ALKUSANAT

Tämä insinöörityö on toteutettu RollTest Measurements & Controls Oy:lle Varkaudessa talvella vuosina 2019–2020. Ensimmäinen mainin lopputyöstä minulle tuli ensimmäisessä työhaastattelussa vuoden 2019 alussa. Alkuvaiheessa en ollut aivan varma mistä lopputyössä oli kyse ja riittääkö minulla rahkeet sen toteuttamiseen. Syksyllä minulle tarjottiin mahdollisuus päästä toteuttamaan lopputyötä koulun ohella, ja en kadu päivääkään, että pääsin tekemään kyseistä lopputyötä.

Lopputyön aloittaminen oli aavistuksen verran haastavaa, mutta aiheen auetessa työ lähti etenemään omalla painolla. Työn aikana tuli monia ylä- ja alamäkiä, joita pääsin kokemaan sekä kollegoiden kanssa että yksin. Tästä haluankin kiittää tekniikan lisensiaatti Pekka Väänästä ja insinööri Miikka Ojaa, suuresta avusta ja monista neuvoista mitä annoitte. Pekka on antanut kaiken tuen opinnäytetyön aikana sekä teoreettisella että käytännön puolella. Miikkaa haluaisin kiittää lopputyön aikaisesta kannustamisesta ja teknisestä avustuksesta. Haluan myös kiittää kaikkia RollTest Measurements & Controlsin henkilökuntaa mukavasta ja tarmokkaasta ympäristöstä. Lopuksi haluaisin kiittää omaa Äitiäni ja kihlattuani lopputyön aikaisesta kannustuksesta, jotka jaksoivat piristää minua myös huonoina hetkinä ja uhrasivat omaa aikaa minua varten.

Varkaudessa 24.2.2020
Daniel Harjunheimo

SISÄLTÖ

LYHENTEET JA NIIDEN MÄÄRITELMÄT

3D	Three-dimensional space. Kolmiulotteisuus.
API	Application programming interface. Ohjelmointirajapinta.
C	C-akseli. Telan pyöritys / C-ohjelmointikieli.
CIL	Common Intermediate Language. Binäärimuotoinen välikieli, jota käytetään .NET ympäristössä yhdessä CLR:n kanssa.
CLR	Common Language Runtime. Virtuaalinen ajoympäristö, jota käytetään .NET ympäristössä.
CNC/NC	Computer Numerical Control. Numeerinen ohjaus, jota hyödynnetään työstökoneiden ohjauksessa.
CU	Control Unit. Ohjausyksikkö.
DB	Database. Tietokanta.
FBD	Function Block Diagram. Toimintolohkokaavio, yksi PLC-laitteen ohjelmointikielistä.
GTK	GIMP ToolKit. Graafinen käyttöliittymäkirjasto.
GUI	Graphical User Interface. Graafinen käyttöliittymä.
HMI	Human-Machine-Interface. Ihmisen ja laitteen välinen käyttöliittymä.
IEC	International Electrotechnical Commission. Kansainvälinen sähköalan standardointiorganisaatio.
IL / STL	Instruction List / Statement List. Jäsennelty teksti, yksi PLC-laitteen ohjelmointikielistä.
I/O	Input / Output. Sisään- ja ulostulo.
IoT	Internet of Things. Esineiden Internet.
ISO	International Organization for Standardization. Kansainvälinen standardointiorganisaatio.
ITU	International Telegraph Union. Kansainvälinen televiestintäorganisaatio.
JSON	JavaScript Object Notation. Tiedonvälitykseen tarkoitettu tiedostomuoto.
LA / LAD	Ladder Diagram. Tikapuumenetelmä, yksi PLC-laitteen ohjelmointikielistä.
LED	Light-Emitting Diode. Hohtodiodi.
ODBC	Open Database Connectivity. Standardoitu rajapinta tietokannoille.
OLE	Object Linking & Embedding. Microsoftin kehittämä tiedonjakomenetelmä.
OPC (UA)	Object Linking and Embedding for Process Control (Unified Architecture). Laitteiden ja koneiden välinen kommunikaatio standardi, jota käytetään erityisesti automaatiossa.
PC	Personal Computer. Tietokone.
PID	Proportional–Integral–Derivative. Suhde-Integroiva-Derivoiva, sähkötekniikassa käytetty kaavaa, joka soveltuu erilaisiin säädinlaitteisiin.
PLC	Programmable Logic Controller. Ohjelmoitava logiikka, jota hyödynnetään teollisuusautomaatiossa.
Qt	Ohjelmisto ja graafisen käyttöliittymiin tarkoitettu kirjastokokonaisuus.
RCC	RollCal Classic. RollTest Measurements & Controlsin kehittämä ja tuottama mittalaite.
ROD	Rotary Encoder. Kulma-anturi.
RTMC	RollTest Measurements & Controls. Akronyymi toimeksiantajan yritysnimestä.

RTU	Remote Terminal Unit. Etäpäälaite. Käytetään SCADA-valvontajärjestelmissä.
SCADA	Supervisory Control And Data Acquisition. Tehtaiden ja suurien laitoksien valvomojärjestelmä.
SFC / GRAPH	Sequential Function Chart. Sekvenssinen toimontokaavio, yksi PLC-laitteen ohjelmointikielistä.
SQL	Structured Query Language. Standardoitu kyselykieli, jota käytetään paljon relaatiomallisissa tietokannoissa.
ST / SCL	Structured Text / Structured Control Language. Jäsennelty teksti, yksi PLC-laitteen ohjelmointikielistä.
TIA Portal	Totally Integrated Automation Portal. Siemensin tarjoama suunnitteluohjelmisto automaatioon.
U	U-akseli. Hiomasupportti, eli hiomakiven asettelu.
UI	User Interface. Käyttöliittymä.
UWP	Universal Windows Platform. Microsoftin luoma uusi rajapinta Windows 10 alustoilla. Korvaa vanhan WinRT rajapinnan. Tukee myös sovelluksien kehittämistä.
UX	User Experience. Käyttäjäkokemus.
WF	Windows Forms, myös WinForms. Microsoftin tarjoama graafinenluokkakirjasto käyttöliittymien kehittämistä varten.
WinCC	SIMATIC WinCC. Siemensin tarjoama SCADA ja HMI järjestelmä.
WinRT	Windows Runtime. Microsoftin vanha rajapinta Windows-käyttöjärjestelmää varten.
WPF	Windows Presentation Foundation. Microsoftin tarjoama graafinenluokkakirjasto, jonka tarkoituksena on korvata Windows Forms-luokkakirjasto.
X	X-akseli. Bombeerauslaite, eli hiomakiven liike säteissuuntaisesti.
XAML	Extensible Application Markup Language. XML-pohjainen tiedonjakoon tarkoitettu tiedostotyyppi.
XML	Extensible Markup Language. Tiedonvälitykseen tarkoitettu tiedostotyyppi.
Z	Z-akseli. Pituuskelkan liike.

1 JOHDANTO

Tässä luvussa tutustutaan tarkemmin lopputyön taustoihin, sen toteutetukseen, tavoitteisiin ja työn rajaukseen. Taustoissa käsitellään tarkemmin lopputyön aiheen syntymistä ja selvitetään mitkä tekijät ovat johtaneet aiheen synnylle. Toteutuksessa kerrotaan tarkemmin työn tutkimusvaiheesta, suunnittelusta ja itse fyysisestä toteutuksesta. Lopussa käsitellään työn tavoitteita ja rajauksista.

1.1 Tausta

Suomessa paperi-, metsä ja terästeollisuudessa liikkuu suuria määriä investointeja eri tuotantolinjojen ja koneiden ympärillä. Varsinkin paperi- ja metsäteollisuudella on ollut suuri merkitys Suomen taloudelle historian aikana. Suomesta löytyy monia suuri metsä- ja paperialan tekijöitä, kuten UPM-Kymmene, Stora Enso ja Metsä Group. Nämä suuret yritykset luovat ison osan Suomen viennistä metsäteollisuudessa, ja luovat tätä kautta Suomen kansalaisille lisää vaurautta ja työpaikkoja. Vuonna 2018, Suomen viennistä 16,1 % oli tuotettu sellun, paperin ja paperituotteiden avulla. (Elinkeinoelämän keskusliitto, 2018)

Näiden koneiden ja tuotantolinjojen varaan on rakennettu suuri määrä investointeja ja työpaikkoja, ja monet alihankkijat ja yritykset ovat riippuvaisia näistä koneista. Nämä laitteet ja linjastot voivat hyvinkin olla monimutkaisia ja moniosaisia kokonaisuuksia, jotka vaativat suuren määrän työvoimaa ja tietotaitoa huoltotoimenpiteissä. Koneiden ja tuotantolinjojen käyttöastetta pyritään pitämään mahdollisimman korkealla, jotta tuotannosta saataisiin mahdollisimman kannattava. Tämä on johtanut siihen, että koneiden ja linjojen huolloista on muuttunut entistä intensiivisemmäksi ja suunnitelluksi. Seisokit pitää suunnitella mahdollisimman tiukkaan aikatauluun, jotta koneen käyttöaste pysyisi korkeana. Tämä vaatii suurta määrää työvoimaa ja huoltotarvikkeita, joita pitää olla seisokin aikana aina saatavilla. Jos jokin toimenpide hidastuu ja aiheuttaa merkittävää hidastumista toiselle toimenpiteelle, pitää ratkaista ongelma mahdollisimman nopeasti, jotta laitteen aiheuttamat tappiot saadaan minimoitua. Pahimmassa tapauksessa pienemmät huollot joudutaan tekemään koneiden ollessa käynnissä, koska koneiden uudelleen käynnistäminen vaatii merkittävän määrän aikaa, joka taas laskee koneen käyttöastetta ja tätä kautta kannattavuutta.

Yksi olennainen osa paperikonetta on sen sylinterinmuotoiset telat, joita voi löytyä monia kymmeniä yhdestä koneesta. Teloja voidaan valmistaa monista eri materiaaleista, riippuen käyttötarkoituksesta. Nykyaikaisien telojen vaippamateriaaleiksi päätyvä joko valurauta tai putken muotoiset teräslevyt. Yleisimmät pinnoitteet ovat kovametalli, kumi ja erilaiset polymeerit

Telojen pinnat voivat kulua hyvinkin nopeasti epätasaiseksi, suuren käyttöasteen ja kitkan seurauksena. Epätasaiset pinnat teloissa vaikuttavat suoraan lopputuotteen laatuun, mutta onneksi tämä epätasaisuus on mahdollista korjata hiomalla. Tämänkaltaiset prosessit vaativat hiojalta ja telahiomakoneen valmistajilta suurta tarkkuutta ja tietotaitoa. Hionnan aikana voidaan teloja tai

valsseja hioa jopa muutaman mikrometrin tarkkuudella, jotta pinnasta saataisiin mahdollisimman tasainen. Hionnan aikana pitää myös huomioida telan ympyrämäisyyttä, sylinterimäisyyttä, halkaisijaa ja monia muita muotoihin vaikuttavia tekijöitä. Myös fysikaaliset ilmiöt, kuten gravitaatio ja lämpötila vaikuttavat telan rakenteeseen. Telan hiomisella on myös rajansa, eikä kaikkia teloja voida hioa tietyn minimihalkaisijan jälkeen. Telat voidaan uudelleen pinnoittaa, jonka jälkeen niiden epätasaisuuksia voidaan jälleen hioa, kun pinnoitetta on riittävä määrä. Tästä kaikesta on syntynyt tarve luoda toimiva ja tarkka hiontajärjestelmä, jotta teloista saataisiin mahdollisimman suoria pienellä hiontamäärällä.

Lopputyön toimeksiantaja valmistaa tämänkaltaisia hiontajärjestelmiä erilaisille paperitehtaille. Nykyteknologia on mahdollistanut entistäkin tehokkaamman ja tarkemman automaatiojärjestelmän, ja tätä on yksi monista syistä lopputyön aiheen synnylle. Aikaisemmat mittaus- ja ohjausjärjestelmät on toteutettu eri tavalla ja monien eri valmistajien avulla, mutta tämän lopputyön tarkoituksena on luoda ohjausjärjestelmä Siemensin valmistamista automaatiojärjestelmistä ja ratkaisuista. Tämä mahdollistaa yksinkertaisemman ja nopeamman huoltotoimenpiteen, koska vian sattuessa, riittää että huoltohenkilö irrottaa viallisen komponentin, ja tilaa uuden samalla tuotenumerolla varustetun komponentin valmistajan maahantuojalta. Tämän kaltainen modulaarisuus helpottaa kaikkien työtä, sekä tehtaalla toimivan hiojan, että laitevalmistajaa. Ohjausjärjestelmästä pitää saada myös mahdollisimman turvallinen, jotta käyttäjälle ja ulkopolisille työntekijöille ei tapahtuisi mitään vahingon sattuessa. Koneen turvallisella pysäytyksellä tai sammutuksella on myös tärkeä rooli vahingon sattuessa.

1.2 Toteutus ja työn rakenne

Lopputyö aloitetaan telahiomakoneen taustatutkimuksella. Taustatutkimuksen tarkoituksena on vahvistaa opinnäytetyöntekijän ymmärrystä telahiomakoneen rakenteesta ja sen toiminnasta. Tutkimuksessa tutustutaan tarkemmin telahiomakoneen rakenteisiin, teloihin ja sen termistöön, bombeeraukseen, nippipaineeseen, mitta- ja ohjausjärjestelmään sekä telageometriaan. Telahiomakoneen rakenteissa tutustutaan tarkemmin sen perusrakenteisiin, mitä eri osia koneesta löytyy ja kuinka se toimii. Teloissa ja sen termistöissä tutustutaan tarkemmin eri telatyyppeihin ja bombeeraukseen ja nippipaineeseen. Bombeeraus on oleellinen osa ohjausjärjestelmän rakentamista, ja sen syistä kerrotaan tarkemmin tässä raportissa. Mitta- ja ohjausjärjestelmässä tutustutaan toimeksiantajan vanhempaan mittalaitteeseen ”RollCal Classic”, josta haetaan pohjaa uuden ohjausjärjestelmän rakentamisessa. Lopuksi käsitellään tarkemmin telageometriaa, jonka tarkoituksena on rakentaa opinnäytetyöntekijälle ymmärrys siitä, mitkä asiat vaikuttavat telan rakenteisiin ja milloin telasta saadaan riittävän hyvän muotoinen. Toimeksiantaja oli keväällä 2019 ELY-keskuksen rahoittamana teettänyt ulkopuolisella konsultilla markkinatutkimuksen ja valmisteluhankkeen liittyen alkavaan tuotekehitysprojektiin, jota hyödynnettiin myös kokonaiskuvan rakentamisessa.

Telahiomakoneen taustatutkimuksen jälkeen, aloitetaan tutkimaan ja suunnittelemaan tarkemmin käyttöliittymäsovellusta. Käyttöliittymäsovelluksen tutkimuksessa etsitään erilaisia työkaluja ja teknologioita käyttöliittymän rakentamiseen, erilaisten tietokanta hallintajärjestelmien soveltuvuutta käyttöliittymään ja PLC:n eri rajapintoja sekä siihen löytyviä valmiita käyttöliittymiä. Käyttöliittymäsovelluksen rakentamistutkimuksessa, painotetaan Microsoftin .NET-ohjelmistokehyksien tutkimista ja niihin soveltuvia ohjelmointikieliä. Vertailun vuoksi, tutkimuksessa verrataan myös alustariippumattomiin ohjelmistokehyksiin, kuten GTK:n ja Qt:n. Tietokanta hallintajärjestelmiä tutkimuksessa etsitään suosittuja tietokantoja, ja selvitetään niitten välisiä rakenteellisia eroja ja tutkitaan erilaisia lisenssipaketteja. PLC luvussa käsitellään ja tutkitaan itse PLC-laitetta, ja siihen rakennettuja HMI- ja SCADA-järjestelmiä. Luvussa myös tutkitaan ja selvitetään erikseen WinCC käyttöliittymän soveltuvuutta lopputyöhön. Luvun lopussa kootaan kaikki tutkittu materiaali yhteen, ja suunnitellaan sen pohjalta lopullinen ratkaisu lopputyölle.

Kummassakin taustatutkimuksessa käytetään erilaisia lähteitä, kuten ammattikorkeakoulujen virallista opinnäytetyö julkaisu portaalia Theseusta, yliopistojen virallisia ja julkisia nettikirjastoja ja monia muita verkosta löytyviä aiheeseen sopivia artikkeleita. Taustatyöllä on suuri merkitys itse lopputyölle ja opinnäytetyöntekijälle. Hyvin tehty taustatutkimus luo pohjan onnistuneelle lopputyölle. Tämä vahvistaa samalla opinnäytetyöntekijän ajatusmaailmaa, kokonaiskuvaa ja luo punaisen langan lopputyölle.

Taustatyön jälkeen, opinnäytetyöntekijä kutsuu kaikki tarvittavat osapuolet ensimmäiseen palaveriin, jossa käsitellään itse työtä, osapuolien vaatimukset ja muita tarvittavia aiheita. Palaverin jälkeen opinnäytetyöntekijä aloittaa itse fyysisen työn, jos kaikki tarvittava tieto on olemassa, ja kaikki osapuolet ovat antaneet hyväksynnän työn aloittamista varten.

Taustatutkimuksien jälkeen raportissa tutustutaan tarkemmin PLC-ohjelmointiin ja siihen perustuvaan verkkokurssiin, jonka opinnäytetyöntekijä kävi lopputyön ohella. Tässä luvussa tutustutaan tarkemmin Siemensin TIA Portal -kehitysalustaan ja IEC 6113-3-standardiin, jossa määritellään tarkemmin PLC-laitteisiin soveltuvista ohjelmointikielistä. Luvun lopussa käsitellään verkkokurssin rakennetta ja siitä saadun tiedon hyödyntämistä lopputyössä. Lopuksi kootaan kaikki opittu ja tutkittu tieto yhteen, ja toteutetaan suunnitelman mukainen testilaite. Testilaitteen toteutuksessa käsitellään PLC-laitteen ohjelmakoodia ja käyttöliittymän rakentamiseen sovellettuja työkaluja, kuten ohjelmointikehystä, tietokanta hallintajärjestelmää ja siihen käytettyjä tekniikoita.

Työn loppupuolella pidetään vielä loppupalaveri, jossa kutsutaan kaikki tarvittavat osapuolet keskustelemaan lopputyön tilanteesta. Tässä vaiheessa kaiken pitäisi olla loppusuoralla työn ja raportin osalta.

1.3 Tavoite ja rajaus

Lopputyön tavoitteena on luoda mahdollisimman käytännöllinen, turvallinen ja käyttäjäystävällinen ohjausjärjestelmä ja käyttöliittymäsovellus. Toimeksiantaja on määrännyt seuraavat tavoitteet lopputyölle:

- Ohjausjärjestelmällä pitää pystyä mittaamaan koetelan akselinsuuntaista halkaisijaprofiilia.
- Ohjausjärjestelmällä pitää pystyä mittaamaan koetelan heittoa ja säteensuuntaista halkaisijaeroprofiilia.
- Käyttöliittymään pitää pystyä luomaan mittaustuloksista graafi, esimerkiksi koetelan ympyrämäisyydestä.
- Käyttöliittymään kautta pitää pystyä lisäämään mittaustuloksia tai koetelan arvoja tietokantaan sekä
- testaamaan hätäseis-painikkeen toimintoja ja asettamaan raja-arvoja ohjausjärjestelmälle.

Työn alkuperäiset tavoitteet olivat hyvinkin kunnianhimoisia, ja niitä jouduttiin rajaamaan useaan kertaan lopputyön aikana. Koska kyseessä on isompi tuotekehitysprojekti, lopputyö on vain yksi osa tätä suurempaa projektia. Salassapitosyistä lähdekoodia ja PLC-ohjelmaa ei tulla julkaisemaan tässä raportissa, ja itse raportti tulee olemaan hyvin pelkistetty lopputyöstä.

Opinnäytetyön kohderyhmä on hyvin laaja, koska työssä käsitellään monia eri tekniikkaan liittyviä osa-alueita. Esimerkiksi raportissa tullaan käsittelemään paljon tietotekniikkaa, automaatioon, konetekniikkaan ja fysiikkaan liittyviä termejä, ja jokainen näistä osa-alueista on hyvin erilainen. Työtä pyritään pitämään hyvin yleisellä tasolla, että moni lukija pystyy ymmärtämään lopputyössä mistä opinnäytetyössä on kyse.

2 TAUSTATUTKIMUS TELAHIOMAKONEESTA JA SEN TOIMINNASTA

Taustatutkimuksen tehtävänä on rakentaa opinnäytetyöntekijälle kokonaiskuvarakenne siitä, minkälaisessa ympäristössä lopputyön tuottama prototyyppilaitteisto tulisi toimimaan. Taustatutkimuksessa tutkitaan esimerkiksi telahiomakoneen, mittalaitteen ja ohjausjärjestelmän perusrakennetta, tyypilliset telatyypit ja niihin kuuluvia termistöjä ja ilmiöitä, ja telageometrian mittaamisesta.

2.1 Telahiomakone ja sen rakenne

Telahiomakoneella on merkittävä rooli paperi- ja metalliteollisuudessa. Sen avulla hiotaan paperiteollisuudessa teloja ja metalliteollisuudessa valsseja haluttuun muotoihin, jonka kautta pystytään vaikuttamaan suoraan tuotteiden laatuun ja tuotannon nopeuteen. Telahiomakoneen rakentaminen vaatii äärimmäistä tarkkuutta, koska pienetkin rakenteelliset virheet voivat siirtyä mekaanisesti hionnan kautta telan pintaan, ja tätä kautta tuotteen laatuun. (Kähkönen, 2003, s. 10).

KUVA 1. Yksilaikkainen telahiomakone ja mittalaitteena RollCal 3 -mittausjärjestelmä. (RTMC, 2012)

Telahiomakone koostuu pääsääntöisesti seuraavista osista:

- Pituusliikkeen kelkan johteet eli hiomakoneen runko
- Työkappalejohteet (joko erilliset johteet tai integroitu koneen runkoon)
- Karalaatikko eli työkappaleen käyttökoneisto
- Telapukit
- Kärkipylkkä

- Pituusliikkeen kelkka
- Poikittaiskelkka eli supportti
- Hiomakara
- Apujärjestelmät: hiomanestejärjestelmä, pölynpoistojärjestelmä

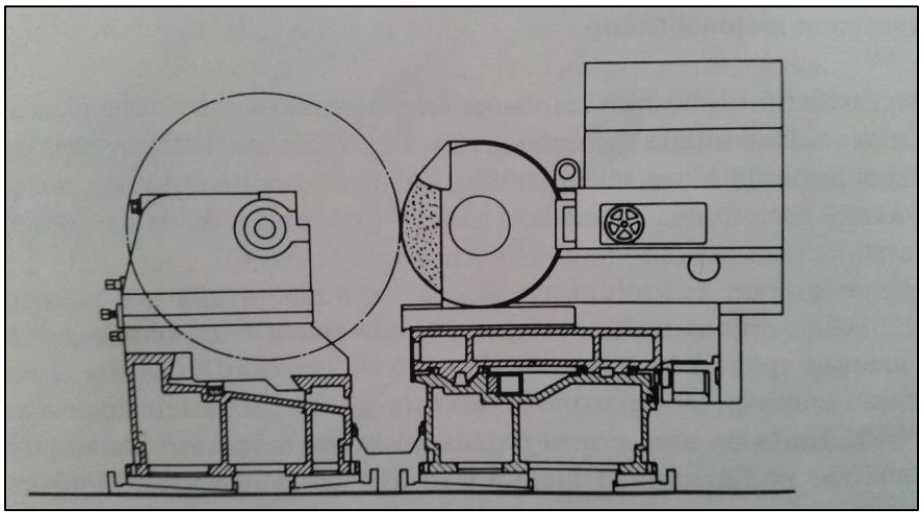
Kaikki näitä osia ei käsitellä lopputyössä, mutta merkittävimmät ohjausjärjestelmän kannalta avataan seuraavissa luvuissa. (Timperi, 2015; Väänänen, 2020)

2.1.1 Hiomakonetyypit

Kaksilaikkaisen hiomakoneen etuina ovat sen ”kelluva” mahdollisuus, hiontanopeus ja tarkkuus. Kelluvassa tilassa laikat päästetään vapaaksi, jolloin telahiomakoneen kelkka ei ohjaa laikkoja, vaan itse työkappale ohjaa niitä. Tätä hyödynnetään varsinkin telan viimeistelyvaiheessa. Tyypillinen hiontaproseduuri kaksilaikkaisessa hiomakoneessa aloitetaan oikaisu- ja rouhintahionnalla, jossa telahiomakoneen kelkat lukitaan johteiden suuntaisesti. Tämän avulla telasta saadaan mahdollisimman suoraviivainen. Lopuksi tela voidaan viimeistellä kelluvalla hionnalla. Hankalahkon käytön takia, kaksilaikkaisista telahiomakoneista on tehty yksilakkaisia. (Kähkönen, 2003)

Kuten aikaisemmin tuli ilmi, yksilaikkainen hiomakone on näistä kahdesta konetyypistä se yleisin konemalli. Yksilaikkaisessa koneessa ei ole kelluvan hionnan mahdollisuutta, joten hionta toteutetaan aina kelkka lukittuna johteisiin. Tämä johtaa siihen, että johteiden merkitys korostuu entistä enemmän yksilaikkaisessa koneessa. Niiden suoruudella ja yhdensuuntaisuudella on erittäin korkea vaatimus, koska pienetkin virheet siirtyvät hionnan kautta työkappaleeseen. (Kähkönen, 2003)

2.1.2 Runko ja johteet



KUVA 2. Yksilaikkaisen telahiomakoneen runkorakenne (Arjas, 1983)

Telahiomakoneen ja sen perustusten muodostama rakenne on oltava jäykkä ja johteiden mahdollisimman suorat ja tasomaiset, jotta hionnasta saataisiin mahdollisimman laadukasta. Johteiden rakentaminen on aikaa vievää ja vaatii suurta tarkkuutta, ja huoltojen laiminlyöntiä ei suositella niiden

osalta. Tämä johtaa siihen, että johteiden kulumista pitää seurata tarkasti ja noudattaa valmistajan huolto-ohjeita, jotta isolta huoltotöitä voitaisiin välttyä. Hiomalaikan kontaktikohdan pituussuuntaisen liikeradan poikittaissuuntainen suoruusvirhe aiheuttaa kaksinkertaisen virheen telan halkaisijassa. Tämän kaltaiset virheet voivat syntyä esimerkiksi johteiden epäsuoruudesta tai epätasaisuudesta. Onneksi nykyteknologia mahdollistaa johteiden tuottaman virheen kompensoinnin hionnan aikana, mutta vanhoissa laitteissa tämänkaltaista kompensointia voidaan ainoastaan toteuttaa manuaalisesti. Johteiden materiaalin mukaan, on tärkeätä seurata johteiden kulutusta, ja huoltaa/oikoa niitä tarvittaessa. Oikaisulla saadaan pidettyä myös johteiden kulumisjälkiä kurissa. (Kähkönen, 2003)

Vanhimmissa telahiomakoneissa liukujohteet ovat voideltu hydrodynaamisesti. Uusimmissa telahiomakoneissa on käytetty liukulaakerijohteita, jotka toimivat paineöljykalvon varassa. Uusia menetelmiä on koitettu myös soveltaa johteiden rakentamisessa, mutta niissä on epäonnistuttu. Esimerkiksi johteita on yritetty rakentaa vierintälaakereiden tasolaakeriyksiköiden avulla. Parhaimmat ja luotettavimmat telahiomakoneet toimivat edelleen hydrostaattisella johdevoitelulla. (Väänänen, 2020)

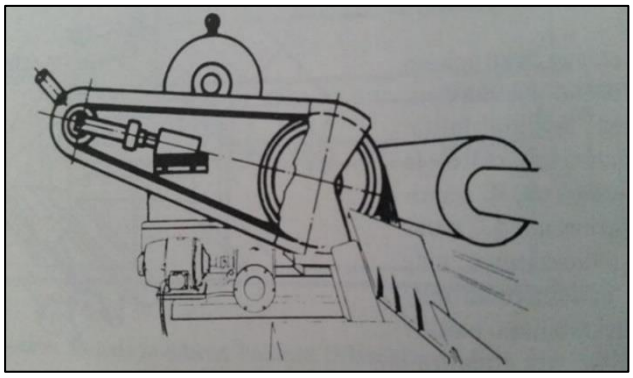
KUVA 3. Yksilaikkaisen telahiomakoneen kelkka. (RTMC, 2019)

Kuvassa 5, on yksilaikkaisen telahiomakoneen kelkka, josta löytyy mm. käyttäjäpaneeli, mittalaite, hiomalaikka, sähkökaappi ja ohjausjärjestelmä. Kelkan alapuolella näkyy johderunko, jonka päällä kelkka liikkuu edestakaisin. Kelkka on olennainen osa hiojan työympäristöä ja sen ympärille on myös rakennettu suurin osa sen älystä.

2.1.3 Hiomalaikka ja nauhahionta

Laikkahiontaa kutsutaan pyöröhionnaksi, jolloin telan ja hiomalaikat pyörivät vastakkaisiin suuntiin toisiinsa nähden. Hiomalaikka koostuu kolmesta eri osasta: Hiontajyvä, sideaine ja niiden välisestä ilmatilasta. Hiontajyvä toimii hiontamateriaalina ja sideaine sitoo hiontajyvät toisiinsa. Yleisimpiä laikkamateriaaleja ovat: Alumiinioksidi, piikarbidi, boorinitridi ja timantti. Hiomalaikkoja on tarjolla

markkinoilla monenlaisia, ja jokainen laikka tarjoaa erilaisia ominaisuuksia erilaisiin käyttötarkoituksiin. Hiomalaikan valintaan vaikuttaa telan pinnoite, poistettavan materiaalin määrä, haluttu pinnanlaatu, telaprofiiliin terävyys, telan ja laikan välinen kontaktipinnan koko ja monia muita erilaisia ominaisuuksia. (Holma, 2016)



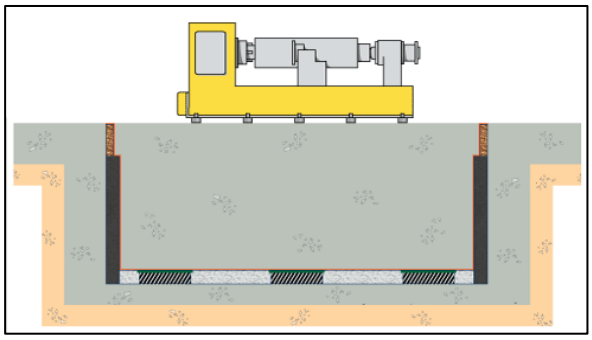
KUVA 4. Nauhahiontalaite asennettuna kelkkaan ja oikealla puolella hiottava tela. (Arjas, 1983)

Laikkahionnan lisäksi on olemassa nauhahiontamenetelmä, jossa telahiomakoneen kelkkaan asennetaan nauhahiontalaite. Nauhahiontalaite koostuu: Kontaktilaikasta, kiristyslaikasta ja niiden välillä pyörivästä hiomanauhasta. Kuvassa 6 näkyy nauhahiontalaitteen sivuttaiskuva, jossa näkyy kaikki edellä mainitut osat. Karamoottorilla pyöritetään kontaktilaikan kautta hiontanauhaa, jonka avulla hiojat pystyvät vaikuttamaan suoraan hiontaan. Kiristyslaikalla voidaan säätää hiomanauhan kireyttä. Nauhan laikkoja valmistetaan eri materiaaleista, mutta se usein pinnoitetaan kumilla. Hiomanauha valmistetaan paperista tai kangasmateriaalista ja pinta hiontajyvistä. Pinta valmistetaan samoista materiaaleista kuin hiomalaikalla. Nauhahionnan etuihin kuuluu sen nopea vaihtaminen ja parempi värähtelynsietokyky. (Holma, 2016)

Eri hiojat tykkäävät eri hiontamenetelmistä. Monille hiojille nauhan käyttö on helpompaa, koska sen vaihto voidaan suorittaa hyvinkin nopeasti. Se on myös yleisesti ottaen tehokkaampi tapa hiota, mutta sen haittapuolena on sen kovempi melutaso laikkahiontaan verrattuna. Toisissa tehtaissa suositaan laikkahiontaa, koska siihen on totuttu. (Väänänen, 2020)

2.1.4 Hiomakoneen perustus

Perustuksen rakennetaan yleensä erillinen tärinänvaimennusjärjestelmä jousista, jonka avulla suodatetaan ulkopuolelta, esimerkiksi paperikoneelta tulevaa tärinää. Telahiomakoneen betonijärkäleen massa voi vaihdella suuresti. Teoriassa suositellaan 10:1 suhdetta, mutta joissakin tapauksissa jopa 4:1 suhde on riittävä. (Väänänen, 2020)



KUVA 5. Farrat:in vaimennusjärjestelmä betoniperustukselle. (Farrat, 2019)

Hiomakoneen asennus betoniperustukseen suoritetaan säädettävien kiilakenkien avulla. Näiden kiilakenkien avulla hiomakoneen runkojohteiden geometriaa voidaan säätää tarkasti.

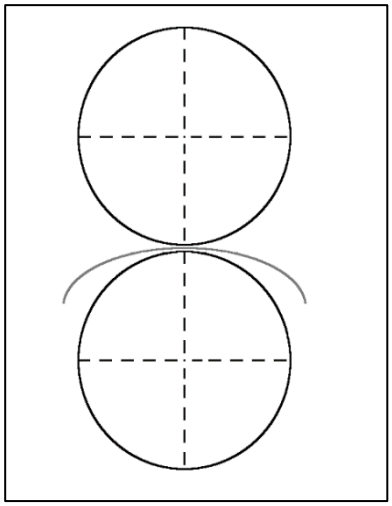
2.2 Telatyypit ja teloihin liittyvää termistöä

Tässä luvussa tutustutaan tarkemmin erilaisiin telatyyppeihin ja teloihin liittyvistä termeistä, kuten nippipaine ja bombeeraus. Telat muodostavat yhden suuren kokonaisuuden paperiteollisuudessa. Telojen avulla vaikutetaan mm. tuotteen valmistusnopeuteen ja sen laatuun. Lopputuotteen laadun kannalta on tärkeätä käsitellä teloja oikealla tavalla ja huoltaa niitä määrätyillä tavoilla ja ajanjaksoina. Teollisuudessa käytetään monia erilaisia teloja ja jokaisella telatyypillä on omat erityiset ominaispiirteet ja käyttötarkoitukset. Jokainen tela on suunniteltu mahdollisimman optimaaliseksi omaan työhönsä nähden. Esimerkiksi pehmeät telat luovat leveämmän nippiviivan ja niiden lämmönkehitys on korkeampi. (Ilves, 2009)

Kuten aikaisemmin tuli ilmi, paperiteollisuudesta löytyy monenlaisia telatyyppejä, jotka soveltuvat omiin tarkoituksiinsa. Kalantereissa voidaan jakaa teloja moneen eri osa-alueeseen. Telojen ominaisuuksilla voidaan vaikuttaa suoraan tuotteen laatuun ja tuotantonopeuteen. Kovilla teloilla pystytään saavuttamaan korkeampia käyttöasteita, mutta värähtelyiden aiheuttamat häiriöt vaikuttavat suuresti telojen kautta lopputuotteeseen. Pehmeillä teloilla pystytään ehkäisemään näitä värähtelyitä, mutta niiden elinkaari on paljon lyhempi kuin kovilla teloilla. (Ilves, 2009)

2.2.1 Puristuskohta tai nippipaine - Nip

Paperi- ja metalliteollisuudessa on yleistä nähdä kahden tai useamman telan välistä kosketusta. Tätä telojen välistä kontaktia kutsutaan nippipaineeksi, ja englanniksi ”nip”. Kuvassa 10 näkyy vasemmalla, kuinka kaksi nippitelaa aiheuttaa Fuji-kalvoon puristuksen. Nippipaineen merkitys eroaa eri teollisuusaloilla, mutta esimerkiksi paperiteollisuudessa tätä painetta hyödynnetään veden poistamisessa paperimassasta ja pinnoitteen tekemisessä. (Valmet, 2013; Menges Roller Co.: Matthew Menges, 2009)



KUVA 6. Sivuprofiilikuva kahdesta telasta ja niiden väliin jäävästä kalvosta.

Painetta voidaan laskea muutamalla eri menetelmällä, mutta näistä yleisimpiä ovat seuraavat:

- Fuji-menetelmä, jossa painetta mitataan dynaamisesti. Kuvassa 9 on esimerkki Fuji-kalvosta ja sen jälkien lukemisesta. Mittaus toteutetaan pujottamalla kalvo kuormitettuihin ja ryömintä vaiheessa oleviin teloihin. Kalvon tarkoituksena on mitata ainoastaan nippipaineen profiilia. Fuji-menetelmä tarjoaa myös erillisen kalvolukija, jolla voidaan myös mitata painetta numeerisesti. Fuji-menetelmän etuna aikaisempiin menetelmiin on sen laaja herkkyysmittaus. (Valmet, 2013)

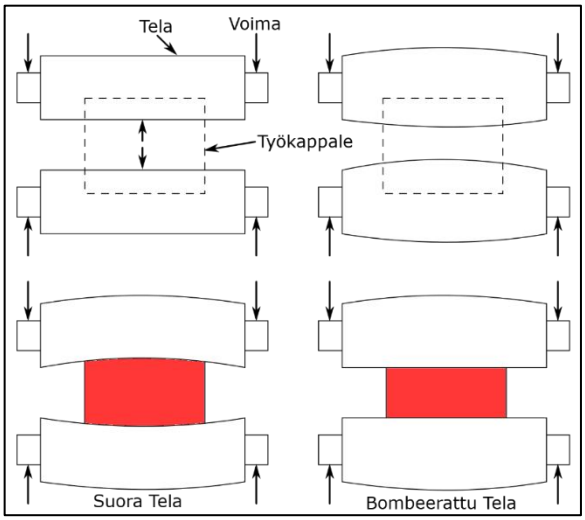
- Hiilipaperimenetelmä, jossa painetta mitataan staattisesti. Hiilipaperi pujotetaan yhden telan ympärille, jonka jälkeen teloja pyöräytetään yhden kerran ympäri ja nippipaineesta jää kuva 10 mukainen jälki. Paperille jää sininen merkintä, ja mitä tummempi väri, sitä enemmän puristusta on kohdistunut hiilipaperiin. (Valmet, 2013, s. 2; Menges Roller Company, 2014)
- Sähköinen menetelmä, jossa painetta mitataan dynaamisesti. Muistuttaa hiilipaperi menetelmää, mutta painetta mitataan elektronisella ”paperilla”. Paperi muuntaa pinnasta tulevan paineen sähkövirraksi (mA, milliampeeri), joka rekisteröidään tietokoneella. Edelliseen menetelmään verrattuna, elektronisella paperilla voidaan mitata painetta pidemmältä aikajaksolta, mutta se on myös merkittävästi kalliimpi kuin hiilipaperi. (Valmet, 2013, s. 3)

KUVA 7. Hiilipaperiin kohdistunut paine ja sen tuloksien lukemista. (Menges Roller Company, 2014)

Mittauksen aikana on myös tärkeätä huomioida kaksi muuta muuttuja: Telan kovuus ja sen halkaisijaprofiili kahden telan pinnan poikki. Näiden kahden arvon perusteella pystytään kertomaan, vaikuttaako mittausarvoihin kuluminen tai kemiallinen pehmeneminen. Mittauksen aikana saadaan myös olennainen tieto, onko tela bombeerattu. Jos tela on bombeerattu, saadaan tietoon myös, millä tavalla se on bombeerattu. (Valmet, 2013, s. 4)

2.2.2 Bombeeraus - Crown

Bombeerauksella tarkoitetaan jonkin esineen hiottua muotoa. Telan ja valssin tapauksessa joko kuperaksi tai koveraksi. Telan bombeerauksella pyritään minimoimaan telan painoa ja saamaan nippipaine mahdollisimman vakioksi koko telan leveydellä. Bombeerauksen suuruus riippuu telan rakenteesta, massasta, rakennusmateriaalista, pintamateriaalista ja käyttötarkoituksesta. Nykyisin bombeerattuja teloja ja valsseja käytetään toissijaisissa käyttötarkoituksissa, ja kriittisissä tapauksissa käytetään taipumakompensoituja teloja (Timperi, 2015; RTMC, Väänänen, 2019)



KUVA 8. Bombeertaun ja tavallisen telan ero. Kuvassa näkyy kuinka työkappale vaikuttaa telan muotoon. (Materials and Processes in Manufacturing, 2003, s. 388)

Telojen ja valssien bombeerauksella varmistetaan nippipaineen tasaisuus, estetään taivutuksen syntymiä ilmiöitä telan ollessa hiottavana ja ehkäistään turhia kulumisia. Taipumisella on suuri vaikutus

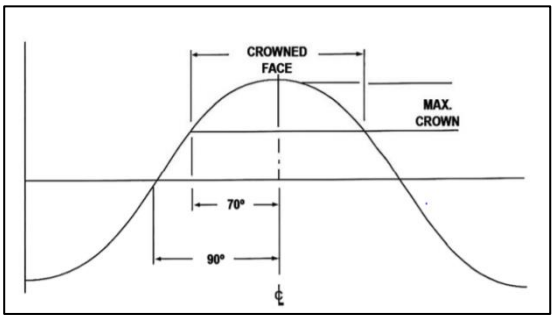
nippipaineeseen, koska taipuminen aiheuttaa epätasaisuutta ja tämän kautta nippipaine ei ole tasainen kaikissa osissa telaa. Taipumiselta ei voida myöskään kokonaisuudessa välttyä, koska esimerkiksi lämpötilaerot ja loppuun ajetuissa teloissa epätasaisuutta tulee vakisinkin. (Menges Roller Co.: Matthew Menges, 2009)

2.2.3 Bombeerauksen toteuttaminen

Bombeerauksen toteuttaminen telalla vaatii suurta tarkkuutta ja varovaisuutta. Tämä pätee varsinkin silloin kun telan bombeeraus päätetään toteuttaa joko hiomalla tai leikkaamalla. Prosessi on itsessään sen verran haastava, että bombeerauksen tekemättä jättäminen voi olla halvempaa, kuin toteuttaa epäonnistunut bombeeraus. Eli suurempia taloudellisia vahinkoja voidaan toteuttaa jo alusta lähtien, jos bombeeraus epäonnistuu. (Menges Roller Co.: Matthew Menges, 2009)

Bombeerauksen tekeminen ei vaadi suurtakaan materiaalin poistamista telasta. Useimmissa tapauksissa puhutaan millin sadasosista, tai jopa kymmenyksistä. Bombeeraus voidaan laskea kosinikäyrän avulla tarkasti, mutta suuntaan antavia arvoja voidaan laskea yksinkertaisen taipumiskaavan avulla. Bombeeraus muotoon vaikuttavat kolme muuttujaa: bombeerauksen maksimi arvo telan keskellä, bombeerauksen profiili telassa ja bombeerauksessa käytetyn kosinin arvoa (Valmet, 2013, s. 6; Menges Roller Co.: Matthew Menges, 2009)

Nykyaikaiset CNC-hiomakoneet osaavat automaattisesti laskea käyttäjän halutun bombeeraus muodon, kunhan kaikki tarvittavat tiedot telan fyysisistä ominaisuuksista ja käyttökohteesta on tiedossa. Hiomakone laskee hiojalle oikean bombeeraus muodon, ja hioja pyrkii hiomaan muodon mukaisen telan. (Menges Roller Co.: Matthew Menges, 2009)



KUVA 9. Bombeerauksen kosinikäyrän muotokaavio. (Valmet, 2013, s. 6)

Bombeerauksen tekeminen telalle tai valssille vaatii suurta tarkkuutta, ammattitaitoa ja pelisilmää hiojalta. Kuten aikaisemmin tuli ilmi, bombeerauksen jättämättä tekeminen voi olla tietyissä tapauksissa halvempaa, kuin siinä epäonnistuminen. Pitää muistaa, että bombeerauksen päätavoite on saada tasainen nippipaine kaikissa telan tai valssin kohdissa. (Valmet, 2013; RTMC, Väänänen, 2019)

2.2.4 Termotela

Termotelat, tai lämpötela, ovat suunnattu erityisesti paperiteollisuuden käyttöön, ja niitä käytetään paljon kalantereissa. Termotelan vaippaan on porattu pituussuuntaisesti kanavia, joiden sisällä kulkee väliaine, joka voi olla vesi, öljy tai kuuma höyry. On myös olemassa myös sähkömagneettisesti kuumennettavia teloja, jotka lämmittävät väliaineen. Telan pinnan lämpötila voi nousta jopa 200 ºC, jolloin paperirata muokkautuu helpommin ja paremmin kuin matalammissa lämpötiloissa. (Rauhala, 2010, s. 39)

Termotelojen yleisimpinä ongelmina voidaan pitää sen kalleutta ja suurta massaa. Niiden suuri hinta johtuu lämmittävän väliaineen kierrättämiseen tarvittavan laitteiston korkeasta hinnasta. Suuri massa johtuu paksusta vaipasta, jonka sisällä on kanavat. Suuri massa aiheuttaa myös enemmän värähtelyongelmia, joita vältetään teloissa. Myös väliaineen lämmitysmenetelmät ovat kalliita. (Suomi Patenttinro FI 113556 B, 2004)

2.2.5 Taipumakompensoitu tela

Taipumakompensoinnilla tarkoitetaan telamassojen ja työkuormituksen aiheuttamaa taipumisen kompensointia säätämällä telan sisäisiä paine-eroja. Tämän telatyypin vahvuuksiin kuuluu dynaamisuus, hyvä värähtelyvaimennuskyky, matala energiankulutus ja vähäinen huoltotarve. Onnistunut rakenne pystytään luomaan kapeiden kuormituselementtien ja ohuen telavaipan ansiosta. Tämän kaltainen tela soveltuu parhaiten kriittiseen käyttöön, eli puristin- ja kalanteriteloiksi. (Lahtinen, 2009; RTMC, Väänänen, 2019)

Haluttu nippipaine pystytään saavuttamaan säätämällä taipumakompensoidun telan kuormituselementtejä. Näitä elementtejä voidaan ohjata servoventtiilien avulla, jotka säätävät kuormituselementtien öljynpainetta. Kuormituselementtien ohjaus on jaettu moneen eri lohkoon, koko nipin matkalla, eli elementtien määrä riippuu telan leveydestä. Tämän avulla painetta voidaan kohdistaa tarkasti tietyssä kohtaa telaa. (Lahtinen, 2009)

2.2.6 Polymeeri ja kuitutela

Polymeeri- ja kuitutelat kuuluvat pehmeisiin teloihin. Vaikka nimityksestä voisi luulla, että telat ovat pehmeitä, todellisuudessa ne ovat melko kovia. Telatyypin nimi on syntynyt siitä, että sitä on verrattu muihin teloihin. Esimerkiksi pehmeät telat ovat aavistuksen verran pehmeämpiä kuin esim. termotelat. Polymeeri- ja kuitutelat ovat ulkoisesti hyvin saman näköiset, kun kaksi edellistä telatyyppiä. (Ilves, 2009)

Kuituteloissa, päiden välille on puristettu kuitumateriaali, joka on rakennettu n. 15–30 % villasta, ja loput kuidusta muodostuu puuvillasta. Telan elastisuutta voidaan säätää villamassan määrällä, jolloin telan markkeerauksen kestävyys paranee ja telan toipuminen kuormaan aiheuttamista jäljistä paranee. Myös merkittävä etu polymeeritelaan verrattuna on se, että kuitutelojen pintavaurioita pystytään hiomaan helposti. Kuitutelan heikkona puolena voidaan pitää sen elastisuutta, varsinkin jos se sisältää paljon puuvillaa. Tämän johtaa siihen, että kovissa kuituteloissa on huono markkeerauksen sietokyky, ja tätä kautta telan elinkaari on lyhyt. (Ilves, 2009)

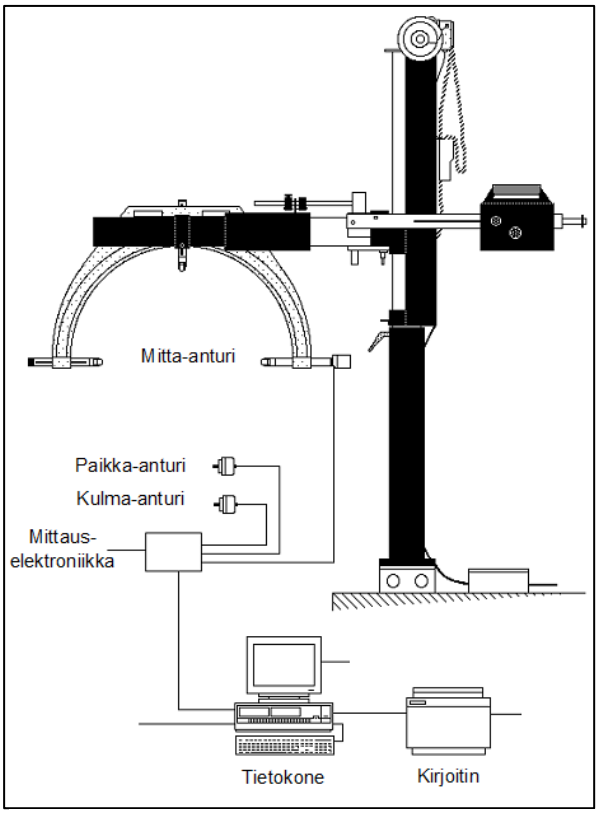
Polymeeriteloissa on pyritty korjaamaan kuitutelojen huonoa elastisuutta. Polymeeriteloissa on parempi markkeerauksen kestävyys, kuin kuituteloissa. Polymeeritelat kestävät myös paremmin viivakuormaa ja korkeita lämpötiloja. Hyvän kestävyyden ansiosta, polymeeritelat voivat jatkaa jopa

kymmenen kertaa pitempään toimintakuntoisena kuin kuitutela. Polymeeritelat ovat myös rakenteellisesti hieman erilaisia kuin kuitutelat. Polymeeritelan pinta koostuu kahdesta eri kerroksesta, pohja- ja pintakerroksesta. Polymeeritelan huonoina puolina voidaan pitää sen korkea hintaa ja vaikeita hionta mahdollisuuksia. Telan pinnan vaurioituessa, uudelleen pinnoitus voi tulla hyvinkin kalliiksi. (Ilves, 2009)

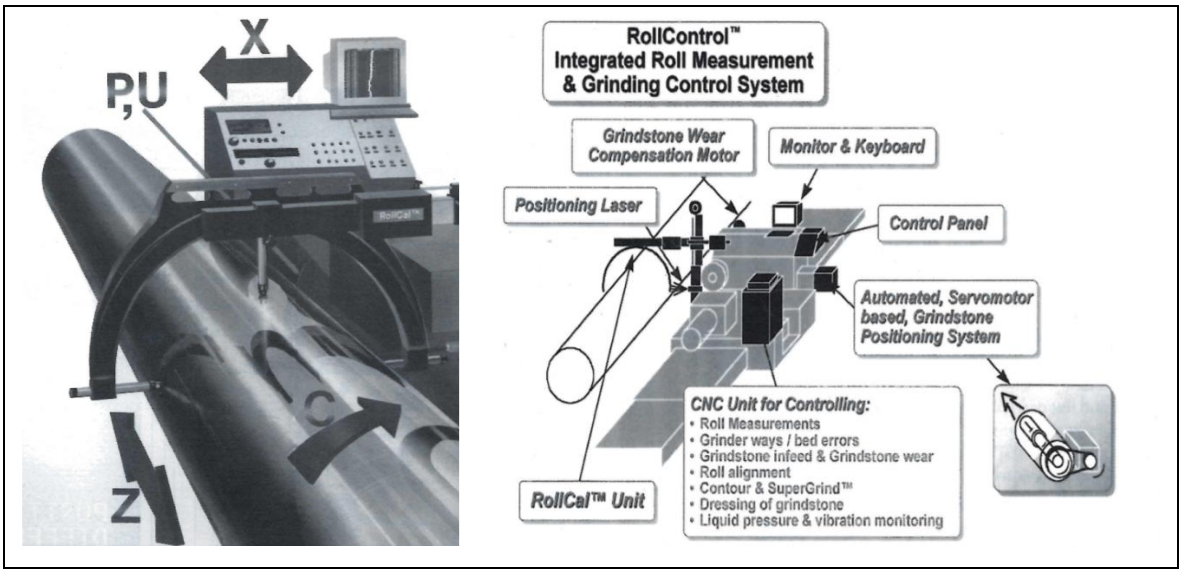
2.3 Mittalaite

Tässä luvussa tutustutaan tarkemmin aikaisempiin mittalaitteisiin ja ohjausjärjestelmiin. Esimerkki mittalaitteeksi valikoitui RTMC:n RollCal Classic (RCC). RCC telamittalaite on kehitelty auttamaan paperikoneiden telojen valmistajia sekä hiojia telojen huoltotehtävissä. RCC antaa erittäin tarkan tiedon telan muodosta, ja sen toistotarkkuus on 0,001 mm. Siihen voidaan myös tarvittaessa liittää hionnanohjausjärjestelmä, josta kerrotaan myöhemmin. (RTMC, 2014)

RCC mittaa telan muodon mittakärkeen kiinnitettävällä elektronisilla pulssiantureilla. Anturin sisältä löytyy herkästi liikkuva mittakara, joka siirtää telan pinnalta saadut arvot sähköisesti mittauselektroniikalle. Paikka- ja kulma-anturit kertovat mittauselektroniikalle, missä kohti telaa mitta-anturi on menossa ja missä kohti johteita kelkka sijaitsee. (RTMC, 2014)



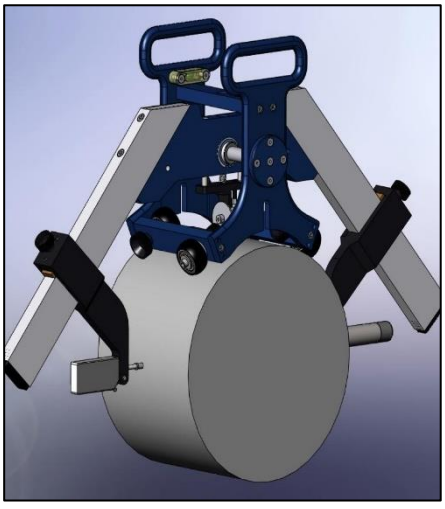
KUVA 10. RollCal Classic telamittalaitteisto. (RTMC, 2014)



KUVA 11. Vasemmassa kuvassa nähdään mittalaitteen akseleiden suunnat ja niidet nimitykset. Oikealla kuvataan mittaus- ja hiontajärjestelmän rakennetta (Pulp & Paper Canada, 1997)

Kuvassa 11 kuvataan hiomakoneen akseleiden suuntia. Oikealla puolella kuvassa näkyy mittalaitteen ja ohjausjärjestelmän rakennekuva, ja kuinka ne sijoittuvat kelkassa ei paikkoihin. Hiomakoneen liikeakseleista käytetään yleensä seuraavia nimityksiä:

- Z = Kelkan pituusliike.
- X = Poikittaiskelkan liike.
- U = Bombeerausliike (sama suunta kuin X:llä).
- C = Telan pyöritys
- G = Hiomakaran/laikan pyöritys.



KUVA 12. RTMC:n RollSaddle. (RTMC, 2018)

RCC mittalaitteella voidaan toteuttaa kolme erilaista mittausta:

- Pituussuuntainen halkaisijaeromittaus
- Kehänsuuntainen halkaisijaeromittaus
- Heittomittaus

Pituussuuntaisessa halkaisijaeromittauksessa vapaasti kelluva mittakaari ajetaan telan yli sen pituussuunnassa. Kehäsuuntaisessa mittauksessa ohjausjärjestelmälle syötetään mittauskohta (Z-akseli) ja mitattavien kierroksien lukumäärä. Heittomittaus voidaan toteuttaa yhdellä anturilla, joka lukitsemalla mittalaitteen vaakaliike tai ratsumitan tapauksessa laittamalla mittausanturi magneettijalkaan. (Väänänen, 2020)

Ratsumalliset mittalaitteet ovat modulaarisia ja joustavia ja tarjoavat helpon tavan mitata työkappaletta, ilman kiinteätä työpistettä. RTMC:n tarjoama ratsu on myös yhteensopiva valmistajan sovelluksien ja graafisen käyttöliittymän kanssa, jolloin käyttäjien ei tarvitse opetella uusia järjestelmiä saadakseen onnistunut lopputulos. (RTMC, 2018)

2.4 Telageometria

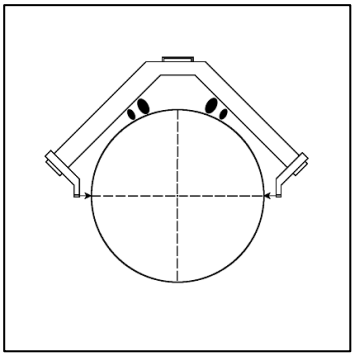
Tässä luvussa tutustutaan tarkemmin telageometriaan, sen virheisiin ja toleransseihin sekä telojen ja valssien epäkeskeisyyteen. Nämä tekijät ovat olennainen osa telojen ja valssien huoltotoimenpiteitä, ja niitä hyödyntäen hiojat ja laitevalmistajat pyrkivät luomaan mahdollisimman suoria ja ideaalisia työkappaleita, jotta lopputuotteesta saataisiin mahdollisimman hyvä.

2.4.1 Sylinterimäisyys ja sen toleranssit

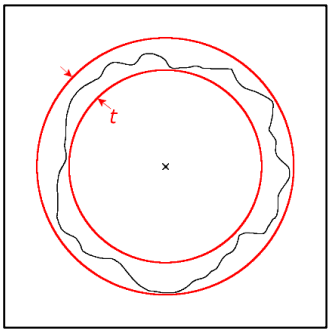
Telojen ja valssien muotovirheitä voidaan havaita mm. niiden kulumisen, ympyrämäisyyden ja suoruuden perusteella. Telageometriassa noudatetaan ISO 110:01:2017-standardia, jossa määritellään erilaisia muotoja ja niiden toleransseja. Eri muotojen toleranssiarvoja voidaan määritellä yhdeksi alueeksi, useammaksi alueeksi tai tilaksi kahden viivan sisällä tai välillä. Niiden kuvastaminen on helpompaa visuaalisesti, ja luvun aikana löytyy muutama esimerkkikuva missä näihin viitataan. Näissä kuvissa toleranssiarvoihin viitataan kirjaimella *"t"*. (Widmaier, 2012; RollTest, 2001)

Lieriömäisen kappaleen geometria voidaan jakaa kolmeen eri osaan: halkaisijaero, suoruus ja ympyrämäisyys. Kappaleen sylinterimäisyys saadaan yhdistämällä halkaisija, suoruus ja poikkileikkausten ympyrämäisyys. Seuraavaksi käsitellään jokaista osa-aluetta:

- Akselin halkaisijaero tai halkaisijaprofiili
    - Halkaisijalla tarkoitetaan kunkin poikkileikkauksen keskimääräistä halkaisijaa kappaleen pyörähtäessä yhden kerran pituusakselinsa ympäri. Pituussuuntaisella halkaisijaerolla tarkoitetaan halkaisijan vaihtelua siirryttäessä kappaleen pituusakselin suunnassa toiseen kohtaan. Se sisältää esimerkiksi kappaleen kartiomaisuuden. Jos tela on bombeerattu, ilmoitetaan halkaisijaero poikkeamana halutusta bombeeratusta halkaisijasta. Halkaisijaeroa ei ole määritelty ISO 1101-standardissa, vaikkakin se on yleisin mittaus telojen ja valssien suhteen. (RollTest, 2001)
- Ympyrämäisyys
    - Poikkileikkauksen ympyrämäisyydellä tarkoitetaan mitatun poikkileikkausprofiilin ulko- ja sisäpuolelle piirrettyjen kahden samankeskeisen ympyröiden välistä pienintä säteiseroa *t*. Referenssiympyröiden keskipisteen määrittämiseen voidaan käyttää useita eri menetelmiä, yleensä neliösumma-menetelmää. Kuvassa 17 nähdään mustalla ympyrämäisyysmittauksesta saatu



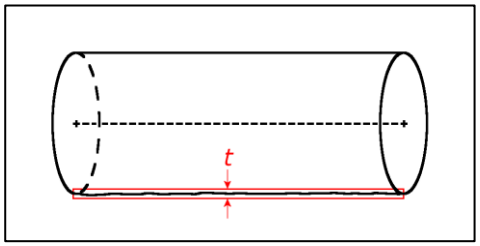
KUVA 13. Hahmotelma ratsusta telan päällä. Ratsun sivut sijoittuvat telan keskelle, josta voidaan mitata telan halkaisija.



KUVA 14. Telan tai valssin ympyrämäisyys toleranssialueen sisäpuolella.

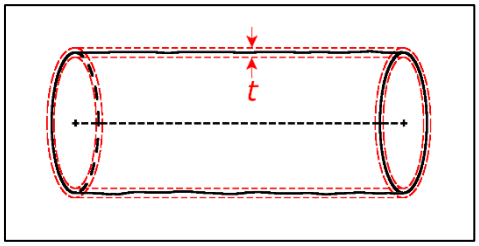
mittaustulos, joka jää kahden punaisen toleranssialueen sisäpuolelle. (RollTest, 2001; Väänänen, 2020)

- Suoruus
  - Lieriömäisen kappaleen suoruudella tarkoitetaan sen poikkileikkausten keskipisteiden muodostamaa käyrää. Usein ollaan kiinnostuneita suoruuden projektiosta vaakatasossa tai pystytasossa. Telahionnassa suoruuden merkitys on yleensä vähäinen. Jos koko tela hiotaan samalla kiinnityksellä, kuten yleensä tehdään, tulee siitä automaattisesti suora. Jos telaa pidetään päistään tuettuna samassa asennossa pitkän aikaa, syntyy siihen epätasaisen lämpötilajakauman vuoksi taipuman muutos, mikä näkyy säteisheittona, jos telaa pyöritetään. Tela kuitenkin jälleen oikenee, kun sitä jonkin aikaa pyöritetään pituusakselinsa ympäri ja lämpötilajakauma tasaantuu. Telan omasta massasta aiheutuva jatkuva taipuma alaspäin voi aiheuttaa hionnassa halkaisijavirhettä, mutta ei kuitenkaan koskaan suoruusvirhettä. (RollTest, 2001; Widmaier, 2012; Väänänen, 2020)



KUVA 15. Telan suoruus toleranssialueen sisäpuolella.

- Sylinterimäisyys
  - Kuten aikaisemmin tuli ilmi: sylinterimäisyyttä ei voida suoraan mitata, vaan mittaamalla telan tai valssin halkaisijaero, ympyrämäisyys useasta poikkileikkauksesta pituusakselin matkalla sekä suoruus, saadaan käytännössä laskettua työkappaleen sylinterimäisyys.

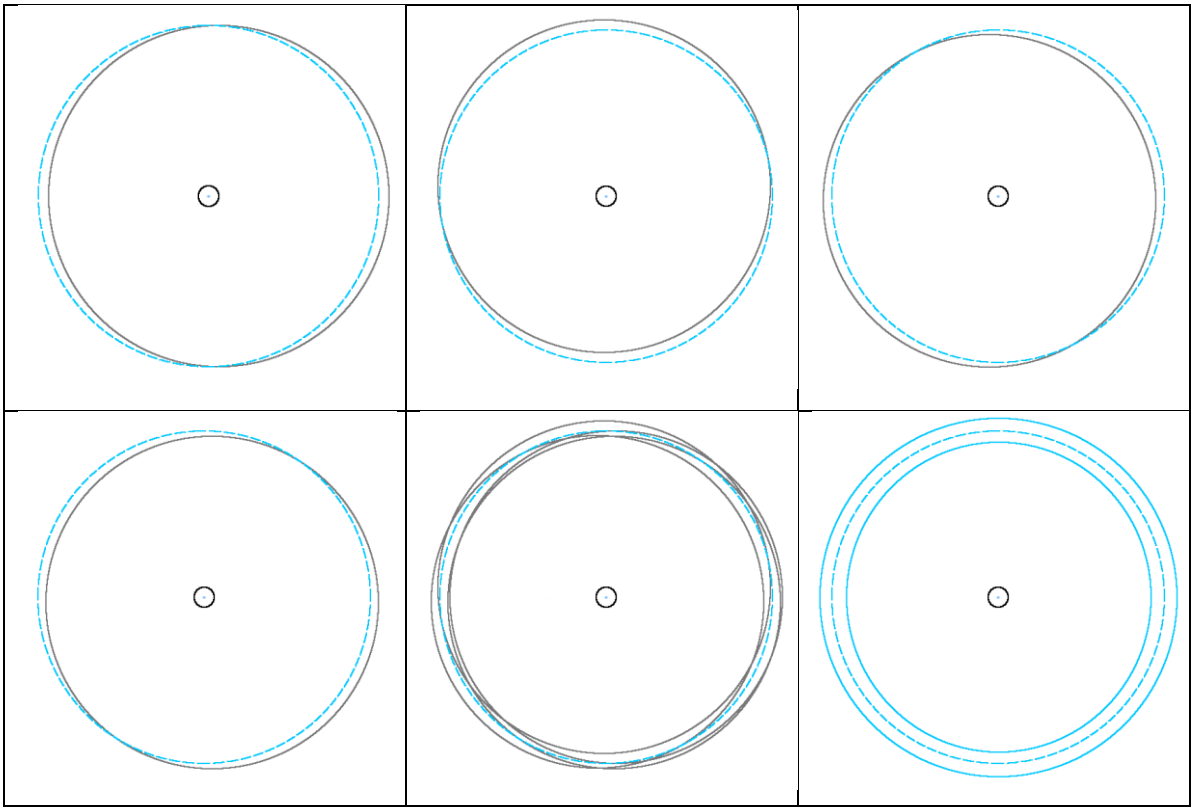


KUVA 16. Sylinterimäisen kappaleen toleranssialue.

Halkaisijaprofiilia yleisesti sekoitetaan suoruuden kanssa. Helpoin tapa erottaa näiden kahden mittauksen välinen ero on: kuvitellaan, että tela on taipunut hieman keskeltä, painovoiman takia, ja täten muistuttaa hieman banaania. Telan halkaisija on myös saman mittainen kaikissa mitatuissa kohdissa. Suoruusmittauksessa mittatuloksena tulee lähes suora pinta tai hieman kaareva, riippuen telan mittauskulmasta. Halkaisijaeromittauksessa mittatulokseksi tulee aina suora pinta, missä tahansa mittakulmassa. (Widmaier, 2012)

2.4.2 Telan epäkeskeisyys

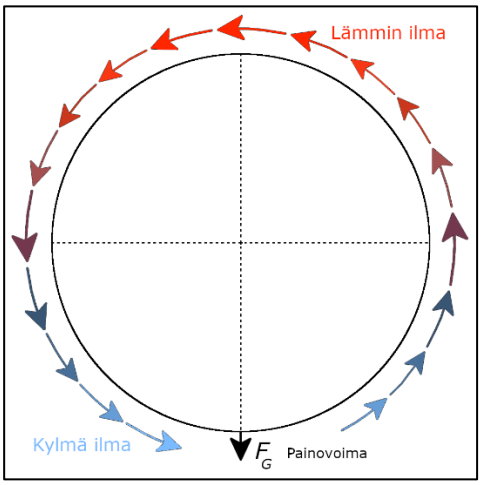
Olennainen osa telageometriaa on myös telan epäkeskeisyys, jota voidaan kutsua myös kiertoakselin virheliikkeeksi. Se ei ole toivottu ominaisuus telan huoltotoimenpiteissä ja käytössä, koska se aiheuttaa suurta muutosta telan liikeradassa. Alla olevassa kuviossa on esimerkki epäkeskeisen telan liikeradasta. (Widmaier, 2012)



KUVA 17. Epäkeskeisen telan liikerata muutokset. Kuvasarjaa luetaan vasemmalta oikealle ja ylhäältä alas. Ensimmäisessä kuvassa nähdään sinisellä katkopohjaisella ympyrällä ideaalisen ympyrän, joka on keskitetty. Harmaalla ympyrällä viitataan epäkeskeiseen telaan, jonka keskipiste liikkuu mustan ympyrän mukaan, ensimmäisestä neljänteen kuvaan. Viidennessä kuvassa nämä kaikki epäkeskeiset telat ovat sijoitettu yhteen, josta voidaan havaita epäkeskeisen telan liikeradan muutokset. Viimeisessä kuvassa nähdään ideaalisen ympyrän ja epäkeskeisen telan liikeratojen erot.

Epäkeskeisyys ja liikeradan muuttuminen voi johtua joko geometrisestä virheestä tai se itse voi olla geometrisen virheen lähde. Yksi mahdollinen syy tämän ilmiön syntymiselle on työstöprosessin aikainen virhe. Esimerkiksi jos työstettävä työkappale on koneistettu erilaisilla työstökoneilla tai telatuilla, sen kiertoakseli voi muuttua. Toinen yleinen virheen syntymislähde on epätäydelliset laakerit joko työstökoneella tai itse työkappaleella. (Widmaier, 2012)

Fysikaaliset ilmiöt vaikuttavat myös suuresti telan liikerataan ja sen rakenteeseen. Esimerkiksi muodonmuutosta syntyy gravitaation ja telan pyörimisen yhteisvaikutuksesta, keskihakuvoimasta, lämpötilaeroista ja monesta muusta tekijästä. Nämä muodonmuutokset voivat muokata kiertoakselin liikerataa ja muuttaa oikein hiotun (toleranssien mukainen) telan epäkeskeiseksi. Muodonmuutoksien vaikutus myös suurenee, mitä isommilla pyörimisnopeuksilla telaa käytetään. Tämän kaltaisista muutoksista voidaan osittain päästä eroon, jos telaa aletaan pyörittämään. Pyörittäminen tasoittaa esimerkiksi painovoiman ja lämpötilaerojen aiheuttamia muutoksia telaan. (Widmaier, 2012)



KUVA 18. Lämpötilaero ja painovoima vaikuttavat telan muotoon. Painovoima aiheuttaa telan roikkumista ja lämpötilaero lämpölaajenemista telan yläosassa, minne lämmin ilma kohoaa.

2.5 Tutkimuksen lopputulos työn kannalta

Tässä pääluvussa tutustuttiin telahiomakoneen rakenteeseen ja toimintaan, erilaisiin telatyyppeihin ja niiden termistöön, mittalaitteeseen sekä telageometriaan ja siihen liittyviin osa-alueisiin. Kaikki nämä aiheet olivat tärkeässä roolissa opinnäytetyöntekijän kokonaiskuvan rakentamisen kannalta. Tutkimuksessa tutustuttiin tarkemmin rakenneympäristöön, missä ohjausjärjestelmän on tarkoitus toimia. Opinnäytetyöntekijän mielestä on tärkeätä tietää, mitkä ulkopuoliset muuttujat voivat vaikuttaa ohjausjärjestelmän toimintoihin, ja mihinkä ohjausjärjestelmän pitäisi pystyä missäkin ympäristöolosuhteissa.

Telahiomakoneen rakenne luvussa tutustuttiin sen perusrakenteeseen, mihin sitä käytetään, millä tavalla teloja hiotaan ja mihin tarkoitukseen mitta- ja ohjausjärjestelmää käytetään. Opinnäytetyöntekijän mielestä oli mielenkiintoista tutkia esimerkiksi telahiomakoneen perustusta, koska työn alussa hänellä ei ollut aavistustakaan, kuinka iso merkitys perustuksen vaimennuksella on mittalaitteeseen.

Tutkimuksessa käsiteltiin muutamaa eri telatyyppiä, bombeerausta ja nippipainetta. Tässä luvussa tutustuttiin tarkemmin, miksi teloja pitää hiota, mihin niitä käytetään ja mitkä tekijät ovat olennaisia niiden käytössä. Bombeeraus on lopputyön kannalta olennainen aihealue, koska laitteen pitäisi pystyä laskemaan tulevaisuudessa hiojalle oikea bombeeraus hiontaa varten.

Pääluvun lopussa tutustuttiin tarkemmin RollTest Measurements & Controlsin tarjoamaan ”RollCal Classic”-mittalaitteeseen ja telageometriaan. Nämä aihealueet olivat suuressa roolissa lopputyön kannalta, koska niiden pohjalta prototyyppiä tultiin rakentamaan. RollCal Classic tarjosi yksinkertaisen ja hyvä esimerkin tavallisesta mittalaitteesta, jolla voitaisiin suorittaa tavallisia mittauksia huoltotoimenpiteissä. Telageometrialla vahvistettiin opinnäytetyöntekijän ymmärrystä. Siinä tutkittiin:

kuinka mitataan tela, milloin hiottava työkappale on tarpeeksi hyvä uudelleenkäyttöä varten ja mitkä fysikaaliset ilmiöt vaikuttavat telan rakenteeseen ja sen liikerataan pyöriessä.

3 KÄYTTÖLIITTYMÄN SUUNNITELU JA VAIHTOEHTOJEN TUTKIMINEN

Tässä pääluvussa tutustutaan erilaisiin ohjelmointiin liittyviin työkaluihin ja kuinka tämän kaltaiseen ohjausjärjestelmän rakentaminen onnistuu erilaisilla menetelmillä. Luvussa tutustutaan mm. ohjelmointikieliin, -kehyksiin, -ympäristöihin, tietokantoihin ja niiden hallintajärjestelmiin, PLC ja automaatiossa käytettyihin työkaluihin, kuten HMI, SCADA ja WinCC sekä OPC UA-standardiin. Luvun lopussa suunnitellaan tarkemmin, millä työkaluilla käyttöliittymäsovellus tullaan rakentamaan ja kuinka saadaan PLC:ltä tietoa sovellukselle.

3.1 Käyttöliittymä

Käyttöliittymäsovelluksella on iso merkitys tässä lopputyössä. Sen tarkoituksena on luoda ohjausjärjestelmä ihmisen ja laitteen välille. Käyttöliittymän avulla, laitekäyttäjän pitää pystyä ohjaamaan PLC:n avulla esimerkiksi servomoottoreita tai muita automaatiolaitteita. Yhteyden pitää myös toimia myös toiseen suuntaan, esim. mittatietoa tai vikailmoitukset pitää pystyä näyttämään käyttäjälle. Lopputyön tarkoituksena on luoda yksinkertainen käyttöliittymän, minkä avulla voidaan ohjata PLC:tä tai saamaan anturitietoa näytölle. Käyttöliittymän kautta pitää pystyä myös tallentamaan tietoa tietokantaan ja muuntamaan mittatietoa graafeiksi.

3.1.1 Ohjelmointikielet, -kehykset ja -ympäristöt

Tässä luvussa tutustutaan erilaisiin ohjelmointikieliin, -kehyksiin ja -ympäristöihin. Luvussa käydään läpi esim. C#- ja C++-ohjelmointikieliä, mitkä ovat tuttuja opinnäytetyöntekijälle. Muitakin ohjelmointikieliä on myös tarjolla, mutta Microsoftin tarjoamat ohjelmointikielet soveltuvat parhaiten Windows-alustalle, koska ne ovat integroitu kyseiselle alustalle ja niihin löytyy laaja työkalu valikoima. Tarjolla on myös monia muita ohjelmistokehyksiä, eikä pelkästään Microsoftin tarjoamia, mutta tässä työssä keskitytään yleisimpiin ja tutuimpiin opinnäytetyöntekijälle.

3.1.1.1 C#-ohjelmointikieli

C# on Microsoftin kehittämä ohjelmointikieli .NET-alustalle, joka julkaistiin vuonna 2001. C#-ohjelmointikieli on tarkoitettu sovelluskehittäjille, jotka luovat Windows-työpöytäsovelluksia, XML-verkkopalveluita, tietokantoja ja monia muita sovelluksia eri tarkoituksiin. Se on hyvin yleispätevä ohjelmointikieli, joka toimii erinomaisesti monessa ei tarkoituksessa. Tämän kielen vahvuuksiin kuuluu sen avoin lähdekoodi, yksinkertaisuus, modernisuus, muokattavuus ja hyvät tulevaisuuden näkymät. C# soveltuu erinomaisesti Microsoftin ”Visual Studio”-kehitysympäristön kanssa, koska siihen on integroitu monia hyödyllisiä ominaisuuksia C#-ohjelmointikieltä varten. Esimerkiksi Visual Studio tarjoaa C#:lle edistyneen koodieditorin, tehokkaan grafiikkasuunniteluun tarkoitetun työkalun, virheenseurantajärjestelmän, automaattinen muistinhallintajärjestelmä ja monia muita hyviä ominaisuuksia. (Chand, 2019; Microsoft, 2015; TutorialsTeacher, 2017; Minal, 2018)

C# kuuluu olio-ohjelmointikieliin. Olio-ohjelmointi on ohjelmointiparadigma, jossa ohjelmointisuunnittelu on rakennettu datan tai objektien ympärille, sen sijaan että ne olisivat rakennettuja funktioihin tai eri logiikoihin. Lyhyesti sanottuna, olio-ohjelmoinnissa pyritään muokkaamaan itse objektia, ilman että siihen tarvittaisiin erillisiä funktioita. Oliokeskeisenä kielenä, C# tukee kapseloinnin, perimisen ja polymorfismin käsitteitä. (Margaret & Sarah, 2020; Microsoft, 2015)

C# määritellään korkeatason ohjelmointikieleksi, eli se on hyvin abstraktinen ohjelmointikieli. Esimerkiksi C++ määritellään keskitason ohjelmointikieleksi, jossa päästään aavistuksen verran lähemmäksi rautatasoa, verrattuna C#:iin. Abstraktisuus mahdollistaa monia erilaisia ominaisuuksia ohjelmointikielelle, esimerkiksi C# voidaan luoda nollautuvan ja enum tyyppisiä arvoja sekä luomaan lambadalausekkeita. (Microsoft, 2015; Minal, 2018)

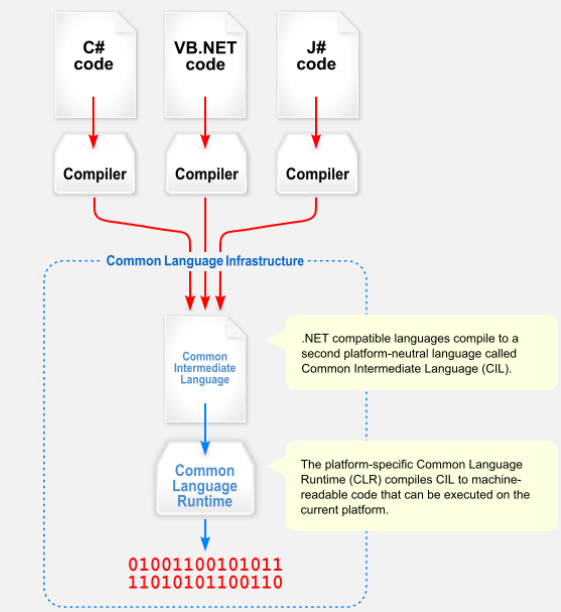
3.1.1.2 C++-ohjelmointikieli

C++, on Bjarne Stroustrupin kehittämä yleiskäyttöinen ohjelmointikieli, joka syntyi C-ohjelmointikielen jatkajaksi. C++ omaksui olio-ohjelmointiin kuuluvia piirteitä, joita C-ohjelmointikielessä ei ole tarjolla. C++ on hyvin suosittu ja se on maailmanlaajuisesti levinnyt monille eri teollisuusaloille. Sitä voidaan hyödyntää esimerkiksi mikroprosessorien, työpöytäsovelluksien, käyttöjärjestelmien ohjelmoimisessa. Kuten aikaisemmassa luvussa tuli ilmi: C++ on keskitason ohjelmointikieli, eli se on aavistuksen verran lähempänä rautatasoa. Se soveltuu erinomaisesti tehoa vaativiin sovelluksiin. Sen tehokkuus johtuu sen valmistaa käännöksestä, eli sitä ei tarvitse erikseen kääntää konekielelle. Usein nämä kääntäjät syövät paljon aikaa, ja tätä kautta ohjelmointikielen tehokkuutta. (W3Schools, 2017)

Ohjelmointikielen hyvinä puolina voidaan pitää sen yhteensopivuutta, tehokkuutta, nopeutta, pientä kokoa rakenteellisesti ja syntaksin samankaltaisuutta monien muiden ohjelmointikielien kanssa. Suuren yhteisön ansiosta, C++ on yhteensopiva monien eri kirjastojen ja kääntäjien. Kääntäjiä löytyy lähes kaikista merkittävistä käyttöjärjestelmistä. Suuri yhteisö mahdollistaa myös ohjelmointikielelle pitkää elinkaarta, eli C++-ohjelmointikieltä tullaan näkemään vielä tulevaisuudessakin monissa eri paikoissa. Huonojakin puolia löytyy, kuten turvattomuus, monimutkaisuus, huonoa muistinkäsittelyä, yksinkertainen oliollisuus, muokattujen toimintojen puuttuminen, tiettyjen tietotyyppien puuttuminen ja syntaksin joustamattomuus. Ohjelmointikielen turvattomuus syntyy monista pienistä ominaisuus puutteista. Esimerkiksi kielestä puuttuu automaattiset rajatarkastukset, virheellisien osoittimien tarkistus ja monia muita pieniä hyödyllisiä ominaisuuksia sovelluskehittäjän kannalta. Huonolla muistinkäsittelyllä tarkoitetaan sitä, että sovelluskehittäjä joutuu itse käsittelemään muistia. Tämä voi toisaalta olla hyvä asia, jos kehittäjä haluaa itse ohjata muistinkäyttöä tarkasti. (DataFlair, 2020)

3.1.1.3 .NET Framework, .NET Core ja .NET 5

.NET Framework on Microsoftin kehittämä ohjelmistokehys Windows-käyttöjärjestelmää varten, joka pohjautuu .NET Standard -määrittelyyn. Ensimmäinen julkaisuversio tuotiin markkinoille vuonna 2002, ja sen tarkoituksena oli tuoda markkinoille uusi työkalu työpöytäsovelluksien kehittämistä varten. Ohjelmistokehyksen tarkoituksena on olla sovelluskehittäjän apuvälineenä, jonka avulla pystytään nopeuttamaan sovelluksen rakentamista. .NET Frameworkista sisältää ison luokkakirjaston, laajan työkalu valikoiman ja tukee monia ohjelmointikieliä. Sen avulla pystytään luomaan mm. nettisivuja, työpöytäsovelluksia ja pyörittämään serverillä ohjelmia, mutta sen toiminta rajoittuu ainoastaan Windows-alustoille. .NET Framework sisältää myös toisen merkittävän ja hyödyllisen työkalun: Common Language Runtime (CLR), joka toimii tässä ohjelmistokehyksessä suoritusmoottorina. Sen tehtäviin kuuluu mm. sovelluksen ajaminen, säikeiden hallinta, muistinhallinta ja monia muita tärkeitä toimintoja sovelluksen suorittamisen ja ylläpidon kannalta. (Edelphi, 2018; Microsoft, 2019)



Ominaisuudet				.NET Framework
Uusia ominaisuuksia WPF, WCF ja WF	Parannuksia ohjelmistokehykseen			4.8 2019
Uusia ominaisuuksia WPF, WCF, WF ja ASP.NET	Parannuksia Win10			4.7 2017
	.NET Standard 2.0 tuki			
NGen	Parannuksia ohjelmistokehykseen			4.6 2015
.NET Native	Muutoksia kirjastoihin			
Uusia ominaisuuksia WPF, WCF, WF ja ASP.NET	Parannuksia ohjelmistokehykseen			4.5 2012
CLR 4.0	MEF	TPL		4 2010
LINQ	WCF	AJAX tuki	Uusia kehitys-ominaisuuksia	3 / 3.5 2006
WPF	WF			
CLR 2.0	ASP.NET 2.0	Uusia kehitysominaisuuksia ja 64-bittinen tuki		2 2005
CLR 1.1	IPv6 tuki	ODBC tuki	Uusia ASP.NET ominaisuuksia	1 / 1.1 2002

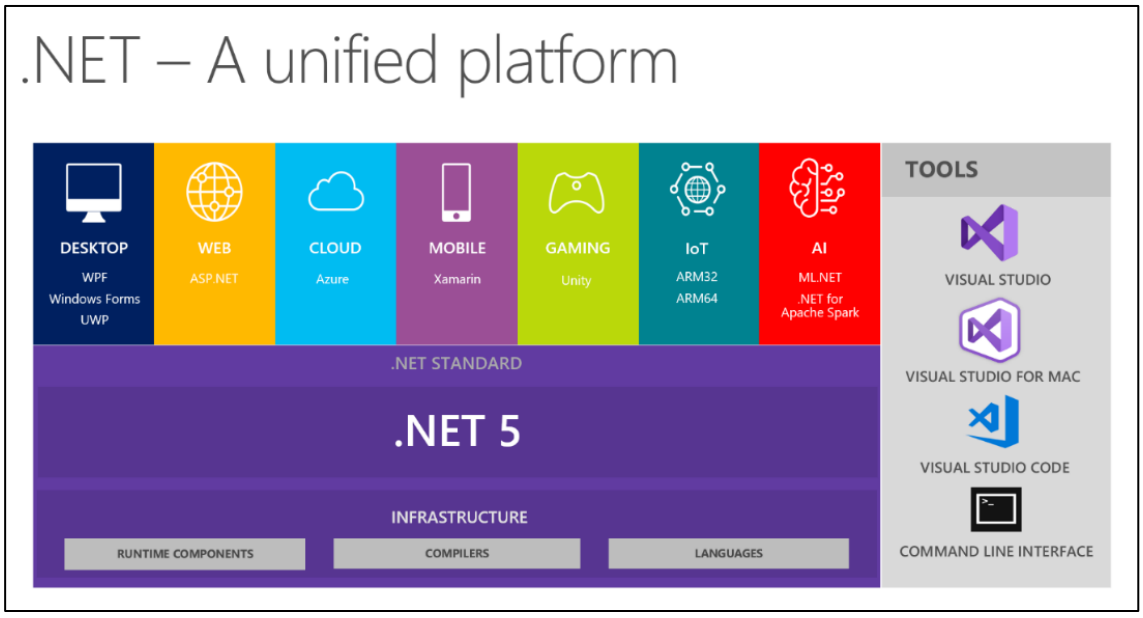
KUVA 19. .NET Framework arkkitehtuuri ja sen ominaisuudet. Vasemmalla kuvassa nähdään eri ohjelmointikieliä, niiden kääntäjiä sekä CIL:n ja CLR:n rakennetta. Oikeanpuoleisessa taulukossa näkyy .NET Frameworkin kehityskaari. (Microsoft, 2020; Piiroinen, 2008)

.NET Core on myös Microsoftin kehittämä ohjelmistokehys, joka noudattaa .NET Standard määritelmää. Se julkaistiin vuonna 2016 .NET Frameworkin seuraajaksi. Edeltäjäänsä eroten, .NET Core perustuu avoimenlähdekoodiin ja järjestelmäriippumattomuuteen. .NET Corella voidaan rakentaa sovelluksia monelle eri alustalle, käyttöjärjestelmälle, laitteelle, pilveen ja IoT-järjestelmiin. Vuonna 2019 .NET Coresta julkaistiin uusi versio, jonka mukana .NET Coreen implementoitiin WinForms ja WPF. Valitettavasti niistä ei toteutettu alustariippumatonta versiota, vaan tarkoituksena oli tuoda vanhat työkalut uudelle, turvallisemmalle ja nopeammalle kehykselle. Implementaatio ei ollut myöskään täydellinen, ja monia työkaluja ei saatu tuotua Framework alustalta Core alustalle. Esimerkiksi ”Designer”-käyttöliittymien suunnitteluun tarkoitettu työkalu jäi Framework alustalle. Myöhemmissä julkaisuissa ilmoitettiin, että työkalut tuodaan myös Core kehykselle, mutta pienellä viiveellä. (Microsoft, 2019; Stack Overflow, 2019)

Vuonna 2019 Microsoft ilmoitti, että .NET Framework 4.8 tulee olemaan viimeinen iso päivitys kyseisellä kehyksellä. Samaisessa ilmoituksessa myös ilmoitettiin, että uutta .NET Standardia ei tulla enää julkaisemaan Frameworkille. Tämän jälkeen Framework tultaisiin siirtämään ylläpitotilaan, jolloin siihen julkaistaisiin ainoastaan korjaus- ja turvallisuuspäivityksiä. Myöhemmin Microsoft tiedotti uudesta .NET Core 3.1 päivityksestä ja sen tulevaisuuden suunnitelmista. .NET Coren nimeä ja versiointia tullaan muuttamaan tulevaisuudessa, ja seuraavaksi versionimeksi tulisi .NET 5. Microsoft kertoi, että muutoksen takana on seuraavat syyt:

- Tarkoituksena on luoda yksi ja yhtenäinen .NET runtime ja ohjelmointikehys, jota voidaan hyödyntää kaikkialla.
- Mahdollisuus laajentaa .NET ominaisuuksia, hyödyntämällä parhaita .NET Core-, .NET Framework-, Xamarin- ja Mono-ominaisuuksia.
- Mahdollisuus rakentaa yhdestä koodikokonpanosta mahdollisimman monipuolinen, jota voidaan kehittää myös yhteisön voimin.

Tulevaisuudessa tulemme näkemään suuria muutoksia Microsoftin alustoilla. C#-ohjelmointikieltä tullaan integroimaan entistä enemmän Visual Studioon ja .NET 5:een, sekä uusia työkaluja tullaan kehittämään alustariippumattomaan ohjelmoimiseen. Kuvassa 23 nähdään mitä .NET 5 tulee sisältämään Microsoftin ilmoitusten perusteella. (Microsoft, 2019)



KUVA 20. Microsoftin visio .NET 5 kokonaisuudesta. (Microsoft, 2019)

3.1.1.4 Windows Forms

Windows Forms, tunnetaan myös nimellä WinForms, on graafinen luokkakirjasto, jonka on keittänyt Microsoft osana .NET Frameworkia. Ensimmäinen virallinen versio julkaistiin vuoden 2002 alussa. WinFormsin tarkoituksena on mahdollistaa monipuolisen sovelluksen rakentamista PC ja tabletti

käyttöön. WinForms on tapahtumapohjainen sovellus, eli se vie suurimman osan ajasta odottamalla käyttäjän syöttävän jotakin tietoa tai tekemällä jotain sovelluksen kanssa. (Microsoft, 2017)

Windows Formissa visualliset komponentit ovat peräisin yhdestä ”Controls”-luokasta. Se tarjoaa laajan käyttöliittymäkomponentti valikoiman, jolla voidaan esimerkiksi muokata komponentin, sijaintia, tekstiä, värimaailmaa ja taustaa, kokoa sekä monia muita komponenttiominaisuuksia. Myös tapahtumien käyttö on mahdollista, jolloin sovellus laukaisee tiettyjä toimintoja, kun ehdot täyttyvät sen laukaisemiseen. Control-luokalla on telakointituki, eli komponentti saadaan tartutettua isäntäkomponenttiin halutulla tavalla, tämä ominaisuus mahdollistaa esim. komponenttien oikean skaalauksen, kun isäntä komponenttia venytetään. (Microsoft, 2017)

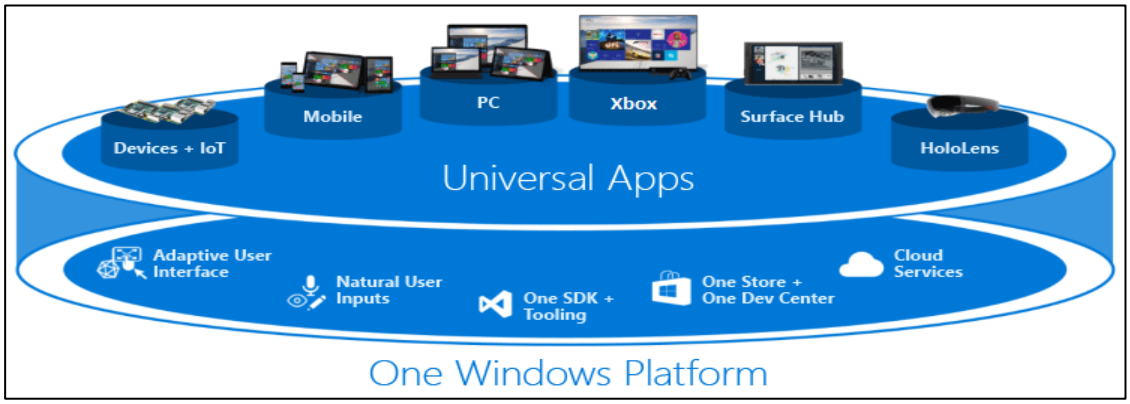
WinFormsilla on monia hyviä etuja, verrattuna muihin luokkakirjastoihin. Esimerkiksi WinFormsista löytyy laaja dokumentaatio ja hyviä koodiesimerkkejä internetistä. Myös kolmansien osapuolien luomia tyylejä, komponentteja ja kirjastoja löytyy internetistä. Huonoina puolina voidaan pitää esteettisen ulkonäön rakentamisen haasteellisuutta ja työllisyyttä sekä alustariippuvuutta. (Martin, 2019)

3.1.1.5 Windows Presentation Foundation

Windows Presentation Foundation, lyhennettynä WPF, on Microsoftin kehittämä luokkakirjasto, jonka avulla voidaan luoda työpöytäsovelluksia Windows-käyttöjärjestelmälle. WPF-kehitysalusta tukee monia sovelluskehitysmahdollisuuksia, kuten esimerkiksi sovellusmalleja, ohjaimia, grafiikoita ja datan sitomista komponentteihin. WPF tarjoaa XAML (Extensible Application Markup Language) kielen, jonka avulla sovelluskehittäjä pystyvät kuvastamaan graafista käyttöliittymää haluamallaan tavallaan. Tämä menetelmä eroaa suuresti esimerkiksi Windows Formsista, missä graafinen käyttöliittymän rakentaminen toteutetaan yhdellä ja samalla ohjelmointikielellä. (Microsoft, 2019)

XAML ei ole ainoa etu WPF:ssa verrattuna muihin kirjastoihin. WPF tarjoaa myös ”Data Binding”-ominaisuuden, jonka tarkoituksena on mahdollistaa datan kiinnittämistä graafiseen komponenttiin. WPF mahdollistaa myös 3D piirtämisen, animaatioiden luomisen ja media komponenttien käytön ilman erillistä työkalua. WPF pystyy myös hyödyntämään laitteistokiihdyttämistä edellä mainittujen ominaisuuksien kanssa, jolloin saavutetaan enemmän tehoa esim. 3D kuvien piirtämisessä. Vaikka WPF:n avulla voidaan luoda graafisesti hienoja käyttöliittymiä, se vie kehittäjältä paljon aikaa. Varsinkin XAML:n opettelu voi viedä suuren määrän aikaa, jotta saadaan haluttu lopputulos. Myös renderöinti vaatii tietokoneelta paljon tehoa, ja pahimmassa tapauksessa erillistä grafiikkakorttia. (Martin, 2019)

3.1.1.6 UWP



KUVA 21. UWP mahdollistaa sovelluksien kehittämistä monelle eri alustalle. Kuvassa näkyy mihin kaikkialle UWP:tä voi käyttää ja mitä työkaluja voidaan hyödyntää sovelluskehityksessä. (Microsoft, 2018)

UWP, muodostuu sanoista "Universal Windows Platform", on Microsoftin tarjoama tuorein luokkakirjasto, joka on suunnattu monelle eri alustalle, mutta pääsääntöisesti Microsoftin omille alustoille. Se tukee mm. Windows 10, Windows 10 Mobile, Xbox One ja HoloLens käyttöjärjestelmiä ja laitteita. Sovelluskehittäjät voivat rakentaa yhden sovelluksen, joka toimii kaikissa edellä mainituissa alustoissa. UWP on rakennettu myös Microsoft Storen ympärille, jolloin sovelluskehittäjä pystyy hyödyntämään tätä portaalia sovelluksen jakamisessa. (Microsoft, 2018)

Vaikka UWP on rakennettu alustariippumattomuuden ympärille, se toimii osittain sitä itseään vastaan. UWP sovelluksien käyttö on hyvin rajallista, koska todellisuudessa kaikkia sovelluksia ei tarvitse julkaista kaikille alustoille, eikä kaikki sovellukset pysty hyödyntämään toistensa ominaisuuksia. Samalla alustariippumattomuus syö sovelluskehityksestä monta hyödyllistä työkalua pois. Microsoftia on myös kritisoitu UWP:n vahvaa sidosta Microsoft Storen kanssa, koska Store on ainoa paikka, mistä käyttäjät voivat ladata sovelluksen julkisesti. (Microsoft, 2018)

3.1.1.7 GTK ja gtkmm

GTK, tai "GIMP toolkit", on monialustainen työkalu graafisien käyttöliittymien luomiseen. Se tarjoaa monipuolisia työkaluja ja laajoja kirjastoja sovelluskehittäjille, joidenka avulla voidaan rakentaa tehokkaita käyttöliittymiä moneen erilaiseen tarkoitukseen. Se on rakennettu neljästä avoimenlähdekoodinkirjastosta Glib, Pango, Cairo ja ATK. GTK-kehitysalusta tukee monia ohjelmointikieliä, mutta parhaimmat ja tehokkaimmat työkalut löytyvät C, C++ (gtkmm), JavaScriptille, Pythonille, ja Valalle. (GTK Project, 2019)

GTK ja gtkmm hyödyntävät erillistä grafiikkasuunnitteluun tarkoitettua työkalua, nimeltä "Glade Interface Designer". Sen tarkoituksena on luoda graafinen käyttöliittymä työkalu, jonka avulla sovelluskehittäjät näkevät suoraan minkä näköinen sovellusikkunasta on tulossa. Tämä mahdollistaa nopean, tehokkaamman ja turvallisen sovelluskehityksen. (Glade, 2018)

gtkmm, on GTK:n C++ versio. Tässä työssä sitä verrataan sen kilpailijaan Qt. gtkmm eroaa rakenteellisesti suuresti kilpailijastaan. Qt:n verrattuna, se on huomattavasti C++ painotteisempi, jossa käytetään valmiita C++-kirjastoja ja alkuperäisiä muistinhallinta ominaisuuksia. Myös sen komponenttien luominen ja ohjelmointirajapinnat ovat huomattavasti täsmällisempiä. (Gnome, 2014)

3.1.1.8 Qt

Qt on ilmainen ja avoimen lähdekoodin työkalu, graafisienkäyttöliittymien luomiseen. Qt:sta löytyy myös maksullinen ja suljetun lähdekoodin versio. Qt:n avulla sovelluskehittäjä voi luoda monelle eri alustalle graafisesti viehättäviä ja tehokkaita sovelluksia, kuten esimerkiksi Windowsille, Linuxille, OS X:lle, Androidille ja muille sulatetuille käyttöjärjestelmille. Se tarjoaa mm. 2D ja 3D piirto mahdollisuudet, XML ja JSON yhteensopivuuden, SQL yhteensopivuuden ja verkko mahdollisuudet. Sille löytyy myös laaja valikoima erilaisia lisätyökaluja ja moduuleja, jotka soveltuvat spesifiseen käyttötarkoitukseen, esimerkiksi sensori ja langattomaan tekniikkaan. (Digia, 2010; Inria, 2013)

3.1.2 Tietokanta hallintajärjestelmien tutkiminen

Tässä tietokanta osiossa tutkitaan tarkemmin sopivia tietokantoja käyttöliittymää varten, ja samalla tutkitaan tarkemmin niiden rakenteita ja mitä ominaisuuksia milläkin tietokannalla on. Tutkinnan alla on yleisimmät tietokannat. Toimeksiantaja on antanut yhden vaatimuksen tietokannan suhteen: yhteys sovelluksen ja tietokannan välillä on rakennetta ODBC-standardin avulla. Tässä osiossa käsitellään myös ODBC menetelmää tarkemmin ja lyhyesti esitelleen myös toista kilpailevaa standardia. Esittelyn tarkoituksena on laajentaa tietämystä muistakin tekniikoista ja standardeista, joita voidaan tarvittaessa hyödyntää projektin jatkokehityksessä.

3.1.2.1 Yleisimmät tietokannat ja tietokantamallit

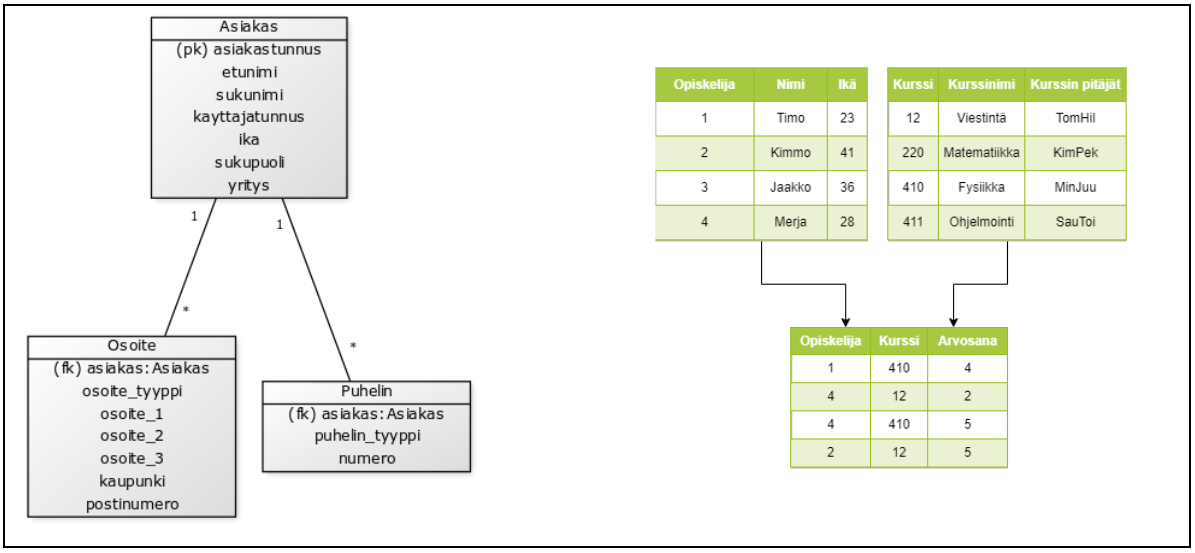
Markkinoilta löytyy monia erilaisia tietokantoja, joidenka suosio on vuosikymmenien aikana vaihdellut. Ensimmäiset nykyaikaiset tietokannat ovat kehitelty jo 80-luvun alussa, mutta uusia ja moderneja tietokantoja tuodaan markkinoille nykypäivä entistä enemmän. Varsinkin pilvipalveluiden tullessa osaksi nykytekniikkaa, on tullut markkinoille tuotu entistä enemmän siihen sovellettuja tietokantoja.

01/2020	Tietokannan ohjausjärjestelmä	Tietokantamalli	Pisteytys 01/2020
1.	Oracle Database	Relaatio ja asiakirjakeskeinen	1346,68
2.	MySQL	Relaatio ja asiakirjakeskeinen	1274,65
3.	Microsoft SQL Server	Relaatio ja asiakirjakeskeinen	1098,55
4.	PostgreSQL	Relaatio ja asiakirjakeskeinen	507,19
5.	MongoDB	Asiakirjakeskeinen ja hakukone	426,97
6.	IBM Db2	Relaatio ja asiakirjakeskeinen	168,70
7.	Elasticsearch	Hakukone ja asiakirjakeskeinen	151,44
8.	Redis	Avainarvo ja asiakirjakeskeinen	148,75
9.	Microsoft Access	Relaatio	128,58
10.	SQLite	Relaatio	122,14
11.	Cassandra	Leveäsarakkeinen	120,66
12.	Splunk	Hakukone	88,67
13.	MariaDB	Relaatio ja asiakirjakeskeinen	87,45
14.	Hive	Relaatio	84,24
15.	Teradata	Relaatio ja asiakirjakeskeinen	78,29

TAULUKKO 1. DB-Enginesin luoma taulukko, jossa on pisteytetty 15 suosituinta tietokantaa vuonna 2020. Tarkempi pisteidenlaskumenetelmä löytyy DB Enginesin verkkosivuilta. (DB Engines, 2020)

Taulukosta 1 nähdään vuoden 2020 suosituimpia tietokantoja. Taulukon perusteella, tietokannat voidaan jakaa kolmeen eri ryhmään, pisteiden perusteella Suosituimmat tietokannat ovat Microsoft SQL Server, MySQL ja Oracle Database, joista jokainen on kerännyt yli 1000 pistettä. Toinen ryhmä

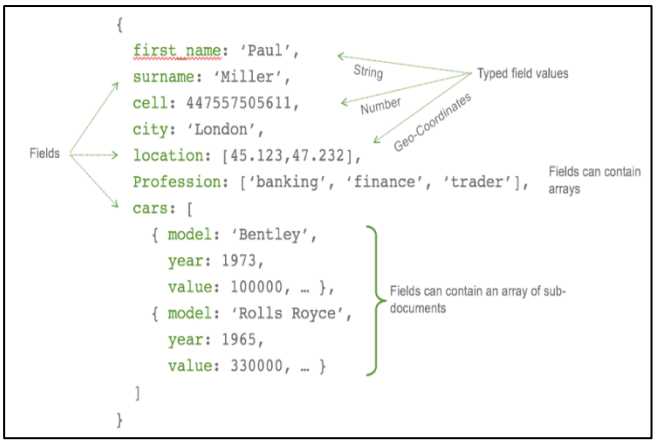
voidaan muodostaa PostgreSQL:stä ja MongoDB:stä, joidenka pisteet pyörivät noin 400–550 pisteen välillä. Loput tietokannat muodostavat viimeisen ryhmän, eli alle 170 pisteen ryhmän. Yksi merkittävä huomio taulukosta ja edellä nimettyjen tietokantojen välillä on relaatiomallin suosio. Viiden suosituimman tietokannan joukossa on ainoastaan yksi ei-relaatiomallinen tietokanta.



KUVA 22. Esimerkki relaatiomallista. (Vihavainen & Luukkainen, 2017)

Kuvassa 25 nähdään, minkälainen suhde ja rakenne relaatiomallia noudattavissa taulukoissa on. Tässä mallissa, taulukkoja voidaan luoda nimeämällä ne, ja määrittämällä niille kiinteä määrä tietotyyppejä. Jokainen rivi taulukossa kuvastaa yhtä tietokokonaisuutta, joka koostuu monesta pienestä tieto attribuutista. Relaatio koostuu kahdesta tai useammasta tietokokonaisuudesta. Relaatiotietokannan hallintajärjestelmää ohjataan SQL-kielellä, jonka avulla voidaan mm. luoda tai poistaa taulukoita, muokata taulukoissa olevaa tietoa, hakea tietoa yhdistämällä tiettyjä attribuutteja eri taulukoista ja monia muita hyödyllisiä ominaisuuksia. (DB-Engines, 2019)

Toinen yleinen tietokantamalli on asiakirjakeskeinenmalli, jossa tietorakennetta voidaan järjestellä hyvin vapaasti. MongoDB on oiva esimerkki tästä mallista. Kuvasta 26 nähdään tyypillinen asiakirjamallin rakenne, joka muistuttaa suuresti JSON-datamuotoa. Tämän mallin etuihin kuuluu rakenteen vapaus, yksinkertaisuus ja tietojen sisentämisen mahdollisuus, mutta tietokannan laajentuessa, sen rakenne muuttuu hyvin monimutkaiseksi ja pahimmassa tapauksessa tietokannan hallitseminen muuttuu äärimmäisen työlääksi ja haastavaksi. (DB-Engines, 2019)



KUVA 23. Esimerkki MongoDB:n asiakirjamallista. (Walters, 2018)

3.1.2.2 Microsoft SQL Server

Microsoft SQL Server on Microsoftin kehittämä relaatiotietokannan hallintajärjestelmä, joka tarjoaa monipuolisia tapoja käsitellä tietokannan tapahtumia, yritystietojen analysointia ja mahdollisuuden käyttää analysointisovelluksia eri ympäristöissä. Microsoft SQL Server on yksi kolmesta markkinoiden johtavimmista tietokantateknologioista Oraclen Database ja MySQL:n rinnalla. Muiden relaatiotietokantojen tapaan, myös Microsoft SQL Server on rakennettu SQL:n päälle. Se on standardoitu ohjelmointikieli, jota sovelluskehittäjät ja järjestelmävalvojat hyödyntävät omassa työssään. Microsoft SQL Server on rakennettu Transact-SQL:n ympärille, ja se osaa hyödyntää Transact-SQL:n tarjoamia käskyjä palvelimen ja asiakkaan välillä. (Margaret;Hughes;& Craig, 2019)

Microsoft tarjoaa ensisijaisesti neljä erilaista SQL Server-pakettia: Developer, Express, Enterprise ja Standard. Kaksi ensimmäistä ovat ilmaisia. Developer-paketti tarjoaa kaikki ominaisuudet käyttöön, mutta tämä paketti on tarkoitettu ainoastaan kehitys ja testi käyttöön. Express-paketti mahdollistaa pienen tietokannan rakentamisen, jonka maksimi koko on enintään 10 gigatavua. Enterprise-paketti on maksullinen ja ns. täysiverinen tietokanta ilman rajoituksia. Standard-paketti on myös maksullinen, mutta se on osittain rajoitettu prosessoivien ytimien ja erilaisten asetuksien osalta. Microsoft tarjoaa myös lisätyökaluja tietokantojen tehostamiseksi, joita voita hyödyntää tietyn tyyppisissä tapauksissa. Kaikki Microsoft SQL Server -paketit löytyvät alla olevasta taulukosta, erilisillä tarkennuksilla. (Margaret;Hughes;& Craig, 2019)

Enterprise	Täysiverinen tietokanta, kaikilla ominaisuuksilla.
Standard	Rajoitettu versio Enterprise paketista.
Workgroup	Sopii suuremman yrityksen etätoimisto käyttöön.
Web	Soveltuu web-sovelluksiin.
Developer	Sama kuin Enterprise, mutta soveltuu ainoastaan kehitys-käyttöön, testaamiseen ja esittämiseen. Helposti päivitettävissä Enterprise-pakettiin.
Express	Ilmainen, mutta rajoitettu tietokantapaketti. Toimii ainoastaan yhdellä prosessorilla ja muistin määrä on rajoitettu 1 gigatavuun. Tietokannan koko rajoitettu 10 gigatavuun.
Compact	Ilmainen sulatettutietokanta, suunnattu mobiili käyttöön. Tietokannan maksimi koko on 4 gigatavua.
Datacenter	Soveltuu suurien yrityksien tietokannaksi, rajauksia ei ole ja löytyy monipuolisia työkaluja tietokannan lisäksi.
Business Intelligence	Muistuttaa Standard-pakettia. Lisäksi löytyy Power View ja PowerPivot työkalut.
Enterprise Evaluation	Testipaketti, joka mahdollistaa Enterprise-paketin käytön 6 kuukaudeksi ilmaiseksi.

TAULUKKO 2. Microsoft SQL Serverin pakettivariaatioita ja niiden tarkennukset. (TutorialsPoint, 2017)

Microsoft SQL Server tukee paikallisien palvelimien lisäksi pilvipohjaisia alustoja. Sen etuihin kuluu mm. sen luotettavuus ja nopea suorituskyky, hyvät tulevaisuudennäkymät ylläpidon kannalta, loistavat lisätyökalut esimerkiksi suorituskyvyn seuraamiseen tai analyysien laatimiseen, hyvät visuaaliset ominaisuudet mobiililaitteiden kanssa ja vankka toimivuus muiden Microsoft tuotteiden kanssa. Tietokanta soveltuu erinomaisesti niihin yrityksiin, missä löytyy valmiiksi Microsoftin tarjoamia palveluita ja sovelluksia. Microsoft SQL Serverin heikkoutena pidetään sen hintaa, koska hyödyllisimmät paketit löytyvät maksumuurin takaa, vaikka Express-paketilla voidaan päästä hyvinkin alkuun pienemmissä projekteissa. (Arsenault, 2017)

3.1.2.3 Oracle Database

Oracle Database on 1970-luvulla kehitelty tietokanta, joka on edelleen pitänyt suurta suosiota monien eri sovelluskehittäjien keskuudessa. Sen suuri suosio perustuu monipuolisuuteen, hyvään alustatukeen, muokattavuuteen ja suurien tietomäärien erinomaiseen hallintaan. Kuten edellinenkin tietokanta, tämänkin tietokanta on rakennettu relaatiomallin pohjalta ja sen toimintakielenä toimii Oraclen oma SQL-kieli PL/SQL. (Guru99, 2018)

Oracle Database Standard Edition 2	Sisältää kaikki tarvittavat ominaisuudet tietokannan kehittämistä työpöytä- tai websovelluksen varten.
Oracle Database Enterprise Edition	Tuo tietokantaan lisää tehoa, käytettävyyttä, skaalaavuutta ja turvallisuutta sovelluksien kehitysvaiheessa. Soveltuu parhaiten sovelluksiin, missä haetaan intensiivisesti tietoa tietokannasta.
Oracle Database Enterprise Edition on Engineered Systems	Pohjautuu edelliseen lisenssipakettiin, mutta lisänä tuodaan joustavuutta ja lisätyökaluja. Soveltuu erinomaisesti paikalliseen käyttöön.
Oracle Database Personal Edition	Muistuttaa Enterprise pakettia, mutta soveltuu yhdenkäyttäjän-sovelluksiin. Toimii ainoastaan Windows ja Linux alustoilla.

TAULUKKO 3. Oraclen ”Database 19”-version eri lisenssipaketteja. Taulukossa on pelkästään paikalliseen käyttöön tarkoitettuja lisenssejä. Oracle tarjoaa myös pilvipohjaisia palveluita. (Oracle, 2019)

Oracle Databasen merkittävimpiin etuihin kilpailijoidensa nähden, on sen monipuolinen alustariippumattomuus. Tämä mahdollistaa eri sovelluksien toiminnan, monilla eri alustoilla. Oracle on tuonut Databaseen VMWare tuen, jonka ansioista kehittäjät voivat esimerkiksi kokeilla tuotteitaan SAP ympäristössä. Toisena merkittävänä etu on sen muistinkäsittelyn tehokkuus, joka helpottaa isojen tietokantojen hallintaa ja nopeuttaa hakujen suorittamista. Oracle tarjoaa myös paljon visualisointi työkaluja sovelluskehittäjille, jotka luovat miellyttävämmän ulkonäön sovelluksille. Oracle Databasen huonoina puolina pidetään sen maksumuuria ja vaativuutta. Maksumuurilla tarkoitetaan sitä, että Oracle ei tarjoa mitään ilmaisversiota Databasesta, ja kaikki lisenssit ovat maksullisia. Databasen vaativuudella viitataan PL/SQL-skriptikieleen, joka on tehokas mutta hyvin työläs opetella. (Guru99, 2018)

3.1.2.4 MySQL

Oraclella on myös tarjolla toinen tietokanta hallintajärjestelmä, nimeltä MySQL. Kuten kaksi edellistäkin tietokantaa, tämäkin noudattaa relaatiomallia pääsääntöisesti, mutta isojakin eroja löytyy. MySQL perustuu avoimenlähdekoodiin, ja se on suunnattu pääsääntöisesti ilmaiseen käyttöön. Sen tarkoituksena on tuoda markkinoille nopea, helppokäyttöinen ja kustannustehokas tietokanta, joka soveltuu erinomaisesti PK yritysten käyttöön. (Edureka, 2019)

MySQL on yksi maailman suosituimmista tietokannoista työpöytä- ja web-sovelluksissa. Sen Community-lisenssipaketti tarjoaa ilmaisen ja ajantasaisen tietokannan yritysten käyttöön, jota päivitetään jatkuvasti. Siihen tuodaan jatkuvasti esimerkiksi uusia ominaisuuksia ja turvallisuuspäivityksiä. (Edureka, 2019)

MySQL Community	Ilmainen avoimenlähdekoodin tietokanta. Tukee sekä SQL:llä että NoSQL:llä. Paljon lisätyökaluja saatavilla ja toimii yli 20 eri alustalla.
MySQL Standard Edition	Tehokas ja luotettava paketti, mutta löytyy rajauksia. Soveltuu parhaiten sovelluksiin, missä luodaan hakuja intensiivisesti verkossa. Maksullinen. Ainoastaan SQL toiminnot saatavilla.
MySQL Enterprise Edition	Täysiverinen tietokantapaketti, josta löytyy kaikki tarvittavat työkalut. Mahdollisuus pilvipalvelun ja NoSQL:n käyttämistä.
MySQL Cluster CGE	Sama kuin edellinen, mutta lisänä tuodaan ”Network Database”-ominaisuus ja klusterityökalu.

TAULUKKO 4. MySQL lisenssipaketteja. (Oracle, 2020)

MySQL tarjoaa helpon ja turvallisen vaihtoehdon tietokanta hallintajärjestelmäksi. Sen avoinlähdekoodi takaa läpinäkyvyyttä ja kriittisyyttä sen rakenteelle, ja samalla tuo lisää turvallisuutta tätä kautta. Avoinlähdekoodi toimii myös kaksipuoleisena kolikkona. Tässä tapauksessa sovelluskehittäjän pitää ylläpitää tietokannan tuoreinta versiota, jotta turvallisuus pystyttäisiin maksimoimaan. Tietokannalle löytyy myös paljon lisätyökaluja, joita voidaan hyödyntää käyttäjien omiessa tarpeissa. MySQL tietokantaa on kehuttu sen turvallisuudesta, tehokkuudesta ja muokattavuudesta, ja kaikki tämä on tarjolla ilmaisessa Community paketissa. (Edureka, 2019)

3.1.2.5 MongoDB

MongoDB on vuonna 2009 julkaistu NoSQL tietokanta hallintajärjestelmä. Se edustaa asiakirjamallia, jossa luodaan tiedoista kokoelmia kuvan 26 mukaisesti. MongoDB:n kehittäjillä oli visiona luoda uudentyyppinen ja ajankohtainen tietokanta markkinoille. Sen tarkoituksena on helpottaa ja luoda relevantti tietokanta sovelluskehittäjälle.

Community Server	Ilmainen tietokantapaketti. Toimii Windows, OS X ja Linux käyttöjärjestelmillä.
Enterprise Server	Tuo lisää turvallisuutta ja parempia seurantatyökaluja edelliseen verrattuna. Maksullinen.

TAULUKKO 5. MongoDB:n eri tuotepaketteja. (MongoDB, 2020)

MongoDB on erinomainen vaihtoehto NoSQL tietokannaksi, mutta siitä löytyy myös haittapuolia. Se on nopea, helppokäyttöinen ja siitä löytyy JSON-tuki, mutta NoSQL voi olla hyvinkin vieras joillekin sovelluskehittäjille. MongoDB:ssä ei haeta tietoa SQL-kielellä, vaan siinä hyödynnetään JavaScript ohjelmointikieltä. Internetistä löytyy joitakin hakumuunnintyökalua sovelluskehittäjille, jotka mieluummin käyttävät SQL-kieltä heidän ohjelmissaan. MongoDB:n asentamisen vaikeutta on kritisoitu useaan otteeseen ja ilmaisversion turvallisuutta on myös kyseenalaistettu. (Arsenault, 2017)

3.1.2.6 PostgreSQL

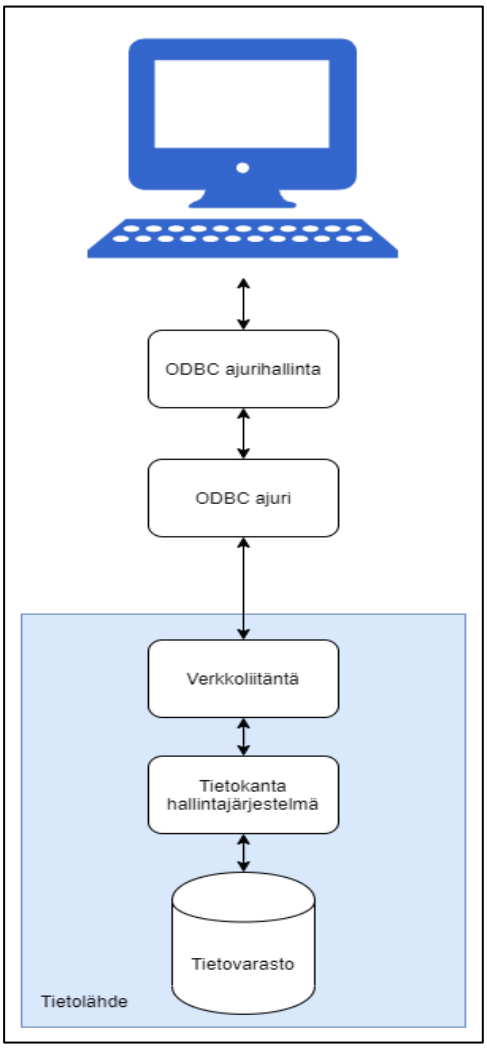
PostgreSQL, tunnetaan myös nimellä Postgres, on viimevuosina nostanut suosiotaan ilmaisena tietokantana web- ja työpöytäsovelluksissa. Postgres on yritystason avoimenlähdekoodin tietokantahallintajärjestelmä. Se tukee sekä SQL:n relaatiomallia että muutamia NoSQL malleja. Postgres tarjoaa ilmaiseksi monia hyödyllisiä ominaisuuksia, mitä Microsoft SQL Server ja Oracle Database pitävät maksumuurin takana. Esimerkiksi Postgres tarjoaa edistyneitä tiedostotyyppejä ja optimoitua suorituskykyä sovelluksille. (Guru99, 2017)

Postgresin takana on suuri sovelluskehittäjäyhteisö, joka luo uusia ominaisuuksia tietokannalle. Se tarjoaa alustariippumattomuutta, laadukasta ja turvallista hallintajärjestelmää ja JSON tukea. Postgresia on myös kritisoitu sen hitaudesta tietyissä tapauksissa ja puutteellista tukipalvelua, joita taas löytyy kilpailijoilta paremmin. (Guru99, 2017)

3.1.3 Tietokannan ja sovelluksen välinen tietoyhteys

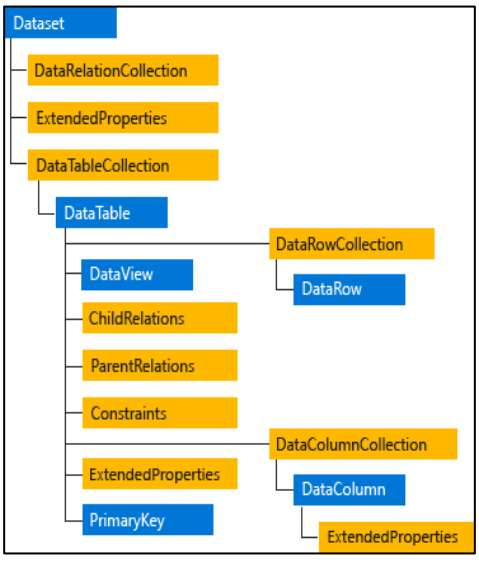
Tietokantojen ja sovelluksien välisiä rajapintoja voidaan rakentaa monen eri standardin pohjalta. Tässä työssä käsitellään kahta eri tapaa yhdistää sovellusta tietokantaan, ODBC-standardin ja ADO.NET-standardin avulla. Toimeksiantaja on määritellyt tietoyhteydeksi ODBC-standardin, mutta tässä tutkimuksessa tutustutaan ADO.NET vaihtoehtoon ja sen toimintaan.

ODBC, lyhenne tulee sanoista *"Open Database Connectivity"*, on standardoitu rajapinta tietokannan ja sovelluksen välillä. Tämä rajapinta on riippumaton käyttäjän käyttämästä ohjelmointikielestä, käyttöjärjestelmästä, tietokannasta ja sen hallintajärjestelmästä. Rajapinta vaatii itselleen erillisen väliohjelmistona toimivan ajurin, jonka kautta pystytään hallitsemaan tietokantaa. ODBC:tä käytetään edelleen maailmanlaajuisesti, ja siihen soveltuvia ajureita voi löytää ympäri verkkoa. Sen käyttö on kuitenkin laskussa, koska markkinoille on tuotu monia muita menetelmiä yhdistää sovelluksia tietokantoihin, esimerkiksi MySQL tarjoaa suorat muuttujat tietokannan käsittelyyn web-sovelluskehittäjille. ODBC:n aktiivinen kehitys on myös lopetettu, eikä sitä tueta enää niin vahvasti. (Microsoft Docs, 2017; Progress, 2020)



KUVA 24. ODBC arkkitehtuuri. Kuvassa näkyy ajurin ja rajapinnan toiminta sovelluksen ja tietokannan välillä.

ADO.NET on Microsoftin kehittämä tiedonsaanti menetelmä, jonka avulla sovellus pystyy hakemaan tietoa esimerkiksi SQL Serveriltä, XML-dokumentista ja muista siihen soveltuvista tietolähteistä. .NET Framework sisältää ADO.NET-standardin määrittelemiä komponentteja, jotka tarjoavat yhteyden rakentamista tietotarjoajaan. Kun tietotarjoajaan on saatu yhteys, sen kautta sovellus pystyy suorittamaan käyttäjän haluttuja komentoja. Haettu tieto sijoitetaan "DataSet"-objektiin, jota voidaan ohjelmassa käsitellä sovelluskehittäjän haluamalla tavalla. ADO.NET sekä SQL että NoSQL (Microsoft, 2017)



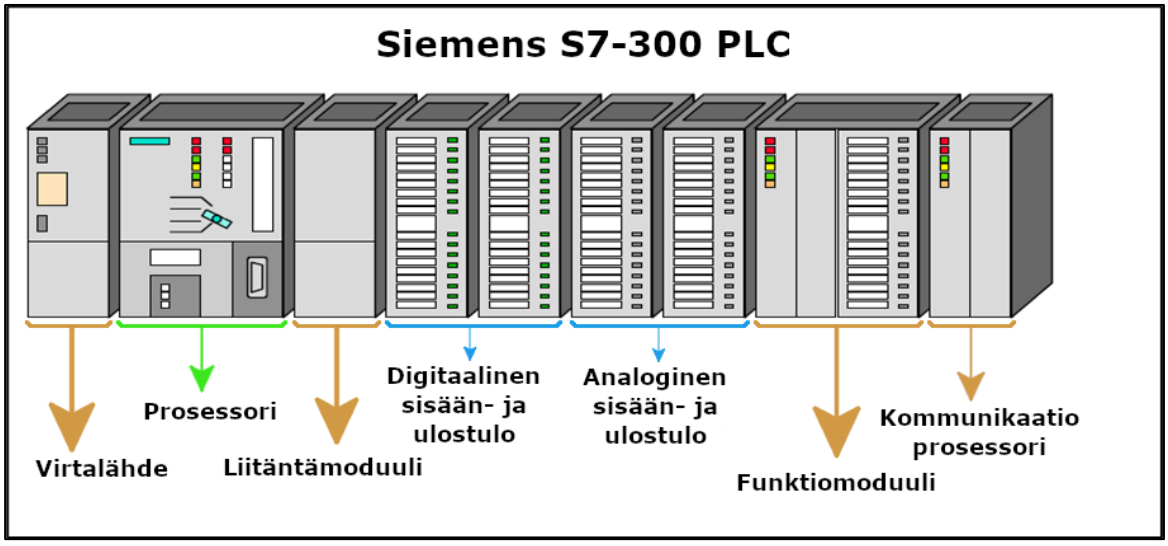
KUVA 25. ADO.NET:n tarjoama "DataSet"-objektin rakenne. (Microsoft, 2017)

3.2 PLC, siihen soveltuvat käyttöliittymät ja rajapinnat

Tässä osiossa tutustutaan tarkemmin opinnäytetyössä käytettyihin Siemens laitteisiin, kuten PLC:n, ja automaatiossa käytettyihin käyttöliittymä malleihin, kuten HMI ja SCADA-järjestelmään. Luvun lopussa käsitellään avointa OPC UA-standardi, jonka tarkoituksena on luoda yhteys kahden laitteen välille alustariippumattomasti ja avoimen tiedonsiirtomekanismin avulla. OPC UA:n avulla pystytään rakentamaan monipuolisia sovellutuksia, ja samalla yhdistämään eri valmistajien laitteitta keskenään.

3.2.1 PLC

PLC, muodostuu sanoista ”Programmable Logic Control”, on ohjelmoitava logiikka tai mikroprosessori, jota käytetään vahvasti automaatioprosesseissa ja monissa eri ympäristöissä. Sen kehitystarpeet syntyivät, kun vanhat relepohjaiset järjestelmät aiheuttivat suurta energian kulutusta ja ylimääräistä lämpöä tehtailla. PLC:n tarkoituksena on ohjata ja lukea eri matalan tason laitteita ja antureita, kuten pumppuja ja kytkimiä. PLC:n rakenne on hyvin modulaarinen. Sen pääosa on rakennettu prosessorista, jonka vierelle voidaan kytkeä muita oheislaitteita, kuten virtalähteitä, erillisiä moduuleja sekä analogisia ja digitaalisia sisään- ja ulostuloja. Oheislaitteita ei välttämättä tarvitse liittää fyysisesti vierekkäin, ja joissakin tapauksissa nämä laitteet voidaan sijoittaa eri tehdasrakennuksiin. (RealPars, 2018)



KUVA 26. Siemensin S7-300 PLC, jonka vierestä löytyy erilaisia moduuleja. (Parrish, 2016)

PLC voidaan ohjelmoida monella eri tavalla. Jokaisella valmistajalla on omat ohjelmistoympäristöt ja variaatiot ohjelmointikielistä. Vaikka yleisin ohjelmointi menetelmä on tikapuumenetelmä, monet valmistajat toteuttavat tämän eri tavalla, ja samaa ohjelmaa ei pystytä suoraan siirtämään toisen valmistajan PLC-laitteeseen. Ongelmaa on yritetty korjata luomalla yhtenäinen standardi IEC 61131-3:2013. (RealPars, 2018)

3.2.2 HMI

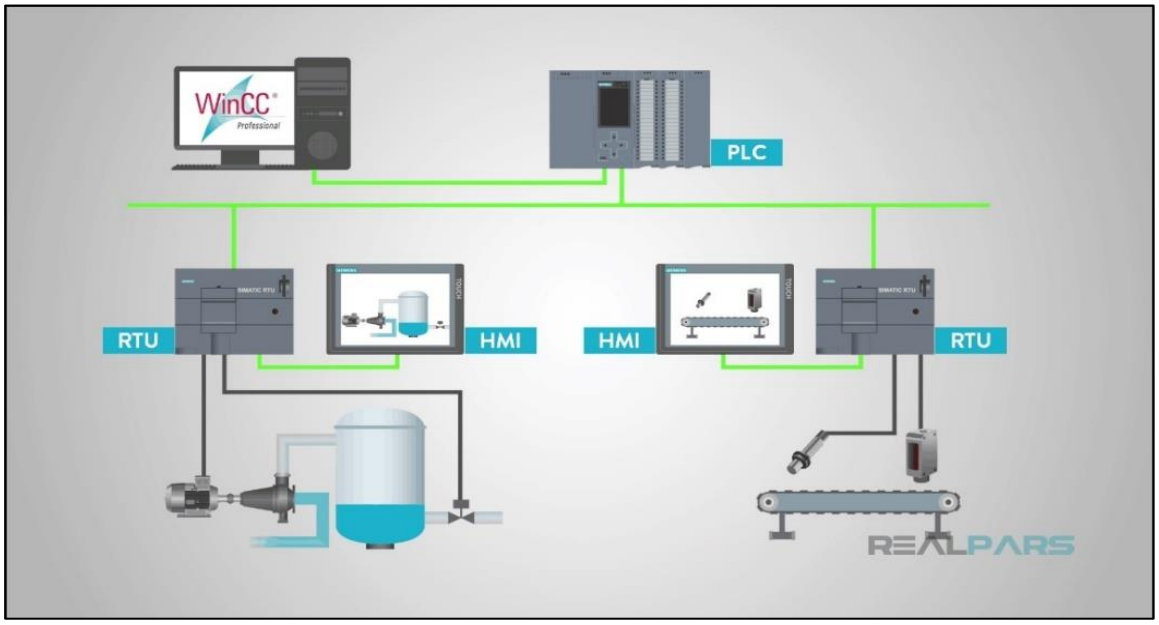
HMI, muodostuu sanoista ”Human-Machine Interface”, on käyttäjän ja koneen välinen käyttöliittymä, jonka avulla ihminen pystyy ohjaamaan suuriakin järjestelmiä isoissa teollisuuslaitoksissa. HMI sovellutuksia löytyy suurimmaksi osaksi automaatiojärjestelmistä, mutta hyvä esimerkki arkielämästä löytyy pankkiautomaateista. HMI rakenne on hyvin yksinkertainen: silloin kun koneen pitää näyttää jotakin, se tuo tiedon käyttäjälle näyttöpaneelin välityksellä, ja kun käyttäjän pitää syöttää tietoa koneelle, se hyödyntää näyttöpaneelin kosketusnäyttöominaisuutta tai fyysisiä kosketusnappeja. HMI paneelit ovat kytkettyinä PLC-laitteeseen, jonka kautta paneelille voidaan siirtää esimerkiksi lämpötila arvoja, nopeus arvoja tai monia muita mitattavia arvoja. Paneeleita voidaan käyttää myös vian tai hädän ilmoittaessa, jolloin käyttäjälle voidaan tarkasti ilmoittaa vian tai hätätilan sijainti, syy ja tapahtuma-aika. (RealPars, 2018)

Markkinoilta löytyy laaja valikoima erilaisia HMI-paneeleita, joista löytyy erilaisia ominaisuuksia ja kokoja. Paneeleita on mahdollista saada esimerkiksi langattomina, jolloin käyttäjät pääsevät fyysisesti vaihtamaan sijaintiaan ja samalla ohjaamaan laitteita paneelin avulla. Monet valmistajat myös tarjoavat eri tapoja ohjelmoida näitä paneeleita, ja kaikilla valmistajilla on omat työkalut ohjelmoimiseen. Ohjelmoijan on tärkeätä tietää sovellutuksen kokonaiskuva, jotta hän pystyisi rakentamaan mahdollisimman turvallisen ja tehokkaan ohjelman paneelia varten. Ohjelmoijan tärkein työkalu HMI-käyttöliittymän rakentamisessa on PLC ulos- ja sisääntulot. Nämä tulot mahdollistavat tiedon saannin ja lähettämisen. Esimerkiksi jos lämpötila-anturi näyttää yli raja-arvon tai jokin kriittinen laite käynnistetään, siitä pitää tulla tieto käyttäjälle, jolloin tilanteeseen voidaan reagoida oikealla tavalla ajoissa. (RealPars, 2018)

KUVA 27. Siemensin SIMATIC HMI, KTP700, 7” kosketusnäyttö ja fyysisillä napeilla varustettu HMI-paneeli. (Siemens AG, 2020)

3.2.3 SCADA ja Siemens WinCC

SCADA, muodostuu sanoista ”Supervisory Control And Data Acquisition”, on laitteistoista ja ohjelmasovelluksista koostuva järjestelmä, jonka tarkoituksena on seurata ja hallita isoja teollisuus kokonaisuuksia. Tämä järjestelmä seuraa, hakee ja prosessoi tietoa reaali-aikaisesti, joka mahdollistaa monipuolisen ja turvallisen järjestelmäseurannan. Edellisessä kappaleessa mainittu HMI voi olla osana SCADA järjestelmää, jossa HMI-paneelit mahdollistavat tiettyjen järjestelmäosien paikallisen tiedottamisen ja ohjaamisen käyttäjälle. Seuraavassa kuvassa näkyy yksi tapa rakentaa SCADA-järjestelmä. Kuvasta pystyy helposti hahmottamaan eri tasot: alimmalta tasolta löytyy mm. anturit ja venttiilit. Keskimmäiseltä tasolta löytyy HMI ja RTU (Remote Terminal Unit, suomeksi etäpäätelaite), jotka hallitsevat näitä antureita ja näyttävät tietoa paikallisella tasolla. Ylimmältä tasolta löytyy PLC ja PC, jotka näyttävät järjestelmän kokonaiskuvan ja kertovat reaali-aikaisesti prosessin tilan. (RealPars, 2019; RealPars, 2019)

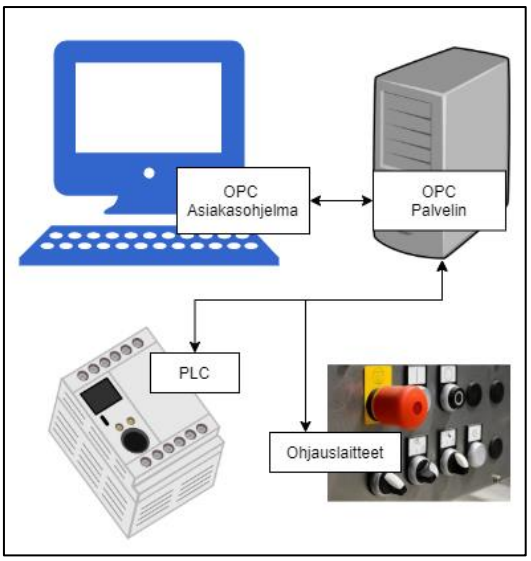


KUVA 28. Yksinkertaistettua SCADA-järjestelmän kokonaiskuva. Kuvassa näkyy mistä osista SCADA-järjestelmä koostuu ja kuinka HMI on osa SCADA-järjestelmää. (RealPars, 2019)

WinCC on Siemensin tarjoama SCADA-järjestelmä. WinCC lyhenne muodostuu sanoista ”Windows Control Center”. Nimensä mukaisesti, WinCC toimii ainoastaan Windows alustoilla, ja hyödyntää Microsoft SQL Serveriä järjestelmän tietokantana. WinCC mahdollistaa monia SCADA-järjestelmälle tyypillisiä piirteitä, kuten: graafinen prosessinseurantajärjestelmä, PLC:n hallintajärjestelmä, hälytys- ja varoitusjärjestelmä. WinCC tarjoaa myös lisäksi: raportointi palveluita, omien skriptien käyttöä sovelluksissa, OPC yhteensopivuutta ja mahdollisuutta liittää muita tietokantoja ODBC:n ja OLE:n avulla. (Siemens, 2016)

3.2.4 OPC ja OPC UA

OPC, muodostuu sanoista ”Open Platform Communications”, on standardi, joka on luotu turvallisen ja luotettavan tiedonsiirtoa varten teollisuusautomaatiossa. Sen tarkoituksena on mahdollistaa eri laitevalmistajien laitteiden välinen kommunikaatio. Standardi julkaistiin vuonna 1996, ja sen tarkoituksena oli abstrahoida PLC:lle spesifiä protokollia yhdeksi tarkemmaksi standardiksi. Tämä mahdollisti HMI/SCADA-järjestelmien välisen rajapinnan. OPC-standardi on sarja tarkennuksia, joita on kehitetty eri automaatioteollisuustekijöiden, loppukäyttäjien ja sovelluskehittäjien yhteisvoimin. Standardissa tarkennetaan ja



KUVA 29. OPC rakenne yksinkertaistettuna. Kuvassa näkyy, kuinka tietoa siirretään PLC:ltä tietokoneelle.

määritellään esimerkiksi rajapintoja, kuinka rajapinnat toimivat eri tapauksissa, tiedon reaaliaikaisuutta tai sen historiaa ja monia muita sovellutuksia. Nykyisin standardia kehittää ja ylläpitää OPC Foundation-konsortio (OPC Foundation, 2017)

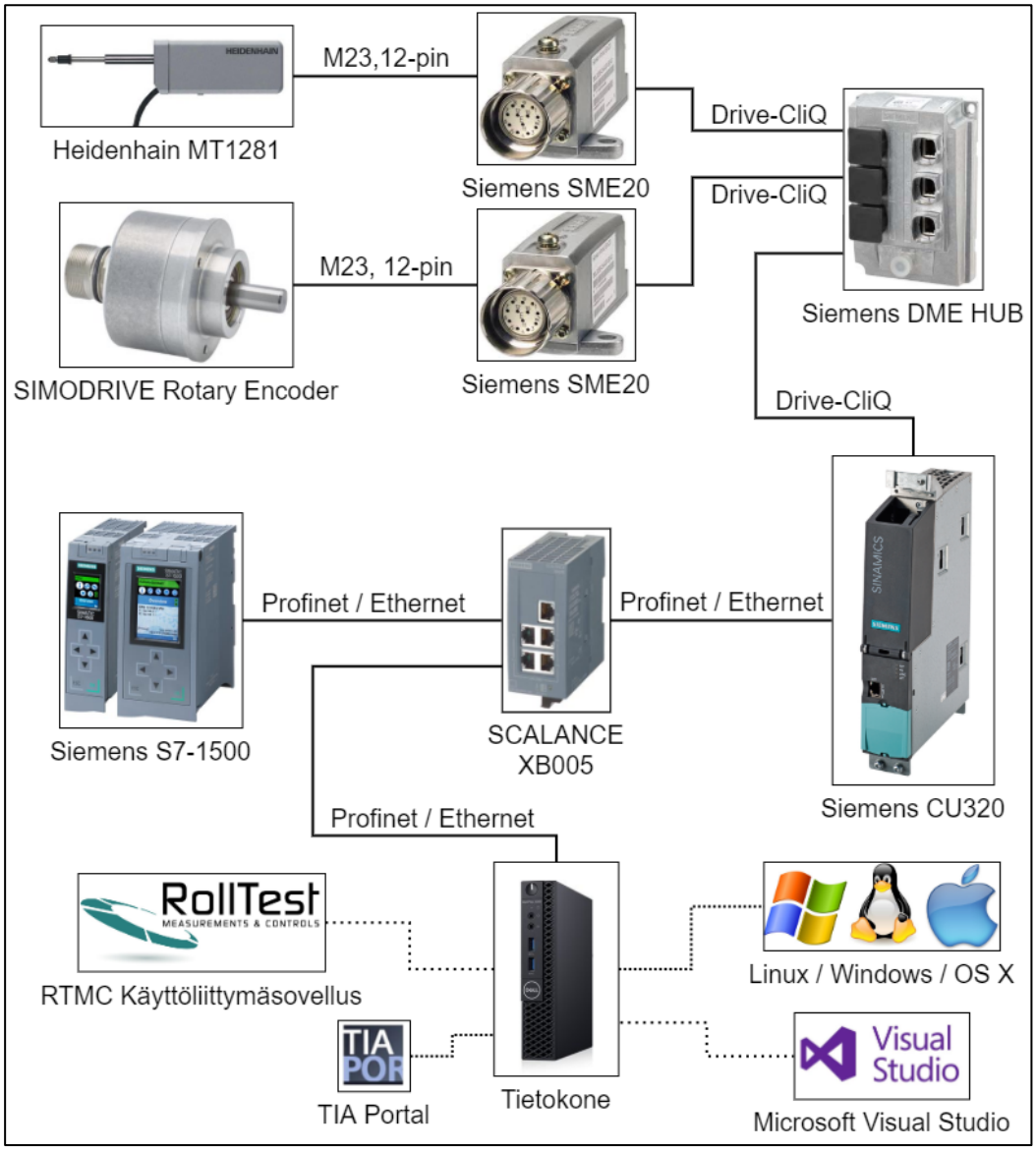
Palvelukeskeisten arkkitehtuurien yleistyessä syntyi haasteita turvallisuuden ja tietomallien suhteen. OPC Foundation-konsortiolle tuli tarve kehittää uusi ja moderni standardi, jonka nimeksi tuli OPC UA. Sen tarkoituksena oli ratkaista vanhat ongelmat ja samalla tuoda uusia yleishyödyllisiä ominaisuuksia, kuten avoimenalustan arkkitehtuuri, skaalautuvuus ja monia muita hyödyllisiä ominaisuuksia. (OPC Foundation, 2017)

OPC UA (Open Platform Communications Unified Architecture), on vuonna 2008 julkaistu avoin standardi, jonka tarkoituksena on määritellä teollisuuslaitteiden välisen tiedonjakoa. Tämä standardi on suunnattu varsinkin niihin laitteisiin, jotka ovat koneissa, koneiden välissä ja suurempien järjestelmä kokonaisuuksien välissä. OPC UA:N etuihin kuuluu seuraavat ominaisuudet:
(OPC Foundation, 2019):

- Toiminnallisuus: Kaikki aikaisemmat OPC määrittelyt on kartoitettu uuteen UA-standardiin.
- Alusta riippumattomuus: Toimii monenlaisissa alustoissa ja käyttöjärjestelmissä. Soveltuu mm. sulatettuihin järjestelmiin, pilvipalveluihin ja Unix-alustoille.
- Turvallinen: Standardiin on tuotu salausmenetelmiä, todennusmenetelmiä ja auditointi.
- Laajennettavissa: Mahdollisuus tuoda uusia ominaisuuksia, vaikuttamatta nykyisiin ja vanhempiin sovellutuksiin.
- Kattava tiedon mallintaminen: Mahdollisuus määritellä monimutkaisia tietomalleja ja tyyppejä.

3.3 Järjestelmän lopullinen suunnittelu, kokonaisuuden muodostaminen ja topologia

Tässä osiossa käsitellään lopputyön lopullista suunnitelmaa, minkä pohjalta opinnäytetyöntekijä tulee rakentamaan prototyypin. Luvussa tutustutaan tarkemmin aikaisempiin tutkimuksiin ja pohditaan mitkä teknologiat ja työkalut soveltuvat parhaiten prototyypin rakentamiseen.



KUVA 30. Topologikuva, missä anturilta saatu tieto siirretään käyttöliittymälle. (Heidenhain, 2019; Siemens, 2020; Siemens, 2020; Siemens, 2020; Dell, 2020; Long, 2012) (Siemens, 2020; Siemens, 2020; Siemens, 2020; VentureBeat, 2019; RTMC, 2015; PLCtraining, 2018)

Kuvassa 34 on alustava topologiakuva järjestelmän rakenteesta. Seuraavassa luettelossa käsitellään jokaista topologissa olevaa komponenttia lyhyesti ja kerrotaan sen merkityksestä lopputyön kannalta:

- Heidenhain MT1281 – Pituusanturi. Tuottaa mittaustietoja järjestelmälle.
- SIMODRIVE Rotary Encoder – Kulma-anturi. Tuottaa mittaus- ja sijaintitietoja järjestelmälle.
- SME20 – Anturimoduuli. Muokkaa antureilta saatua tietoa järjestelmälle sopivaksi.
- DME HUB – Jakaja. Yhdistää anturimoduuleja yhteen.
- CU320 – Ohjausyksikkö. Vastaanottaa ja lähettää ohjaustietoa. Voidaan esimerkiksi ohjata moottoria tämän yksikön avulla.
- SCALANCE XB005 – Verkkokytkin. Mahdollistaa verkkoyhteyden jakamisen eri komponenteille.
- S7 1500 – Ohjelmoitava logiikka. Automaatiojärjestelmä, joka voidaan ohjelmoida halutulla tavalla. Esimerkiksi voidaan seurata tiettyjä sisääntuloja ja tuottaa tiettyjä ulostuloja.
- Tietokone/PC. Toimii sovelluksien rautana.
- Linux / Windows / OS X – Käyttöjärjestelmä. Mahdollistaa sovelluksien toimimisen raudan kanssa.
- Microsoft Visual Studio – Ohjelmointiympäristö. Sovellus, jonka avulla käyttöliittymä ohjelmoidaan.
- TIA Portal – Ohjelmointiympäristö. Sovellus, jonka avulla PLC-laitetta ohjelmoidaan.
- RTMC Käyttöliittymäsovellus – Sovellus. Toimii käyttäjän ja laitteen välisenä ohjausjärjestelmänä tietokoneen puolella.

3.3.1 WinCC ja OPC UA

Luvussa 3.2 tutkittiin tarkemmin PLC-laitetta, ja kuinka siitä saatua tietoa voidaan hyödyntää mm. HMI- ja SCADA-järjestelmissä. Näiden järjestelmien tarkoituksena on visualisoida teollisuusprosessi käyttäjälle ja samalla mahdollistaa sen hallinta. Luvussa tarkennettiin myös SCADA:n ja HMI:n välisiä eroja. SCADA on tarkoitettu valvomokäyttöön ja HMI paikalliseen (PLC tai RTU) seurantaan ja ohjaukseen.

Luvun aikana myös käsiteltiin Siemensin omaa SCADA-järjestelmää nimeltä WinCC. Sen kehitysympäristönä toimii Siemensin TIA Portal, jonka tehtävänä on yhtenäistää Siemensin eri ohjelmointijärjestelmiä yhdeksi isommaksi kokonaisuudeksi. Aikaisempia kehitysalustoja olivat esimerkiksi STEP 7-ohjelmointiympäristö, jonka avulla Siemensin omia PLC-laitteita voitiin ohjelmoida. Tarkemmin TIA Portalista ja sen sisältämistä ohjelmista kerrotaan myöhemmässä luvussa. (RealPars, 2019)

WinCC on osana TIA Portalin kehitysympäristöä, joka tuo yhtenäistä toimivuutta eri Siemens laitteiden kesken. WinCC sovelluksia voidaan kehittää TIA Portalissa, ja niiden yhdistäminen esimerkiksi HMI-paneeliin on suhteellisen yksinkertaista kehitysympäristön ansiosta. Toisaalta TIA Portalia on kritisoitu sen kompleksisuudesta ja laajasta työkalu valikoimasta. Vuosien saatossa TIA Portalista on kasvanut valtavan kokoinen kehitysalusta, jonka myötä projekteista on tullut monoliittisia ja rakenteeltaan suuria. Tämä voi olla kokeneelle sovelluskehittäjälle hyvä asia, mutta aloittelevalle sovelluskehittäjälle tämä tuo korkean oppimiskynnyksen kehitysalustalle.

WinCC:tä on mahdollista käyttää tässä lopputyössä, mutta sen tuoma hyöty lopputyölle on suhteellisen pienehkö. Vaikka itse käyttöliittymän perusrakenteen luominen on helppoa, se vaatii paljon ylimääräistä työtä. Esimerkiksi WinCC:tä ei löydy valmiita grafiikkakomponentteja lopputyötä varten, mutta niitä voidaan luoda .NET-kehitysalustan avulla ja tuoda tätä kautta WinCC-projektiin. Mutta tuoko tämä mitään todellista hyötyä lopputyölle, jos projektissa pitää kumminkin opetella .NET-kehitysympäristön käyttö?

Luvussa 3.2.4 tutustuttiin OPC:n ja OPC UA:n. Tämä standardi tuo toisen vaihtoehdon WinCC:n rinnalle. Käyttöliittymä voitaisiin rakentaa perinteellisellä työpöytäsovelluksella, ja kommunikaatioyhteyden muodostaminen tietokoneen ja PLC:n välille pystyttäisiin toteuttamaan OPC UA-standardin pohjalta. Esimerkiksi Siemens tarjoaa yksinkertaisen esimerkkisovelluksen, joka on luotu Windows Formsilla. Tämä sovellus pystyy kommunikoimaan PLC:n kanssa apuluokan avulla, joka noudattaa OPC UA-standardia. OPC UA tarjoaa turvallisen ja luotettavan kommunikaatio menetelmän kahden laitteen välille.

WinCC:n tarkoituksena on luoda teollisuusautomaatioon tarkoitettuja käyttöliittymiä, joita käytetään yhdessä HMI-paneelien ja SCADA-järjestelmien kanssa. Sen kirjastosta löytyy paljon valmiita grafiikkakomponentteja, joidenka avulla pystytään visuaalisesti näyttämään laitteen eri tiloja käyttäjälle. WinCC mahdollistaa myös omien räätälöityjen komponenttien käyttöä, mutta niiden luominen vaatii erillisiä työkaluja ja ohjelmia. Tämän lopputyön kannalta on tärkeätä luoda mahdollisimman turvallinen ja miellyttävä käyttäjäkokemus hiojalle, mutta WinCC tuoma hyöty on pienehkö OPC UA-standardin tarjoamaan vaihtoehtoon verrattuna. Sovelluksesta saadaan huomattavasti enemmän irti, jos se luodaan .NET-kehitysalustan ja standardin pohjalta.

3.3.2 Ohjelmointiympäristö, -kehys ja -kieli

Edellisessä luvussa käsiteltiin OPC UA-standardia, joka yksinään ei riitä luomaan käyttöliittymäsovellusta tietokoneen ja PLC:n välille. OPC UA:n rinnalle voidaan liittää luvussa 3.1 tutkittuja ohjelmointikehyksiä ja ohjelmointikieliä.

Tutkimuksessa käsiteltiin kahta eri ohjelmointikieltä, joista kummatkin ovat suhteellisen tuttuja opinnäytetyöntekijälle. Kummatkin kielet edustavat olio-ohjelmointia ja kummastakin löytyy paljon tietoa verkosta. Näistä kahdesta ohjelmointikielestä C++ on vanhempi ja tehokkaampi, mutta kaikki on suhteellista. C++ tehokkuus näkyy ainoastaan tietyissä tapauksissa, ja todellisuudessa kuumatkin vaihtoehdot ovat tehokkaita. C++ avulla pystytään ohjelmoimaan monella eri ohjelmistokehyksellä työpöytäkäyttöisiä sovelluksia, sekä .NET-ympäristössä että vanhemmilla Qt ja GTK:lla. C# on nuorempi ohjelmointikieli, mutta sen vahvuudet näkyvät ainoastaan Microsoftin valmistamissa tuotteissa. Microsoft on optimoinut monet työkalut C# -ohjelmointikielelle ja vahvistanut sen asemaa Visual Studio -ohjelmointiympäristön avulla.

Ohjelmointikehyksistä ainoastaan WPF on tuttu opinnäytetyöntekijälle, mutta kaikilla tutkimuksessa mainituilla ohjelmointikehyksillä voidaan luoda lopputyöhön soveltuva käyttöliittymäsovellus. Liitteessä yksi ja kaksi verrataan näitä kehyksiä toisiinsa. Seuraavaksi käydään jokainen ohjelmointikehys läpi erikseen, ja tutkitaan sen soveltuvuutta lopputyöhön:

- Windows Forms, soveltuu hyvin prototyypin rakentamiseen. Sillä pystytään tekemään nopeasti pelkistettyjä käyttöliittymiä ja sen käytössä riittää ainoastaan C#. Jatkokehityksen kannalta WinForms ei välttämättä ole paras vaihtoehto kaikista, koska se on vanha ja sitä ei enää kehitetä samalla tavalla, kuten WPF tai UWP. Microsoft on ilmoittanut, että WinFormsia tullaan vielä ylläpitämään, mutta mitään uusia ominaisuuksia sille ei enää tuoda tulevaisuudessa. Sovelluksen ulkonäön suhteen, WinForms on huono vaihtoehto. Sillä voidaan luoda nopeasti toimivia sovelluksia, mutta miellyttävän näköisiä niistä tulee ainoastaan kovalla työllä. Pahimmillaan miellyttävän ulkonäön saavuttamiseksi tarvitaan maksullisia kolmannen osapuolen kirjastoja.

  Windows Presentation Foundation soveltuisi parhaiten lopputyöhön, kaikista .NET ohjelmointikehyksistä. Sitä voidaan käyttää vanhemmilla Windows-alustoilla, ja sen avulla pystytään rakentamaan visuaalisesti paljon kauniimpia käyttöliittymiä edelliseen ohjelmistokehykseen verrattuna. WPF hyödyntää myös tietokoneen grafiikkakorttia esimerkiksi 3D grafiikan piirtämisessä. Lopputyön kannalta WPF soveltuisi hyvin siihen, mutta XAML tuo ulkonäön rakentamiseen ylimääräistä työtä, eikä ulkonäkö ole vielä tässä vaiheessa tuotekehitysprojektia oleellinen asia.

- Universal Windows Platform, eli UWP, voidaan suoraan tiputtaa kaikista vaihtoehdoista, koska sillä on liian paljon rajoituksia tämän kaltaiseen projektiin. Esimerkiksi WinRT tuottaa ylimääräistä haittaa yksinkertaisen sovelluksen rakentamisessa. UWP on myös sidottu hyvin vahvasti Microsoft Storeen ja Windows 10, mutta sovelluksia pystytään käyttämään, asentamaan ja poistamaan ilman Storeakin. UWP soveltuu erinomaisesti omaan tarkoitukseensa: sovellusten luonti monelle eri alustalle. Lopputyön kannalta, UWP ei tarjoa mitään merkittävää etua muihin kehyksiin nähden, mutta ehkäpä tulevaisuudessa UWP:n merkitys vaihtuu markkinoilla, ja sitä pystytään soveltamaan paremmin silloisiin alustoihin.

- Qt ja GTK eivät kuulu .NET-standardin alle, ja eroavat suuresti kolmeen edelliseen ohjelmistokehyksiin verrattuna. Kummatkin näistä kehyksistä on kehitetty 1990-luvun puolella ja soveltuvat hyvin monelle eri alustalle. Tähän projektiin nämä kehykset soveltuisivat suhteellisen hyvin, mutta prototyypin rakentamiseen ne ovat liian työläitä. Myös visuaalisen käyttöliittymän rakentaminen näillä työkaluilla on raskasta, koska valmiit komponentit, kuten napit ja tekstikentät, muistuttavat paljolti 2000-luvun tekniikkaa ja vaativat uudelleenräätälöintiä. Qt ja GTK ohjelmistokehitykset soveltuvat tässä työssä ainoastaan, jos on tarvetta rakentaa sovelluksesta alustariippumaton ja mahdollisimman suorituskykyinen. .NET tarjoaa paljon helpommat ja modernimmat työkalut työpöytäsovelluksen rakentamiseen ja prototyypin rakentaminen onnistuu helpoiten sen pohjalta.

Kuten aikaisemmin raportissa tuli ilmi, kaikki mainitut ohjelmistokehykset mahdollistavat käyttöliittymäsovelluksen kehittämisen. Jokainen ohjelmistokehys soveltuu omaan tarkoitukseensa, mutta lopputyön kannalta riittää, että voidaan luoda yksinkertainen sovellus, joka toimii OPC UA:n kanssa. Suuren painoarvon tuo myös Siemensin tarjoama esimerkkisovellus, jossa käytetään WinFormsia ja OPC UA:n apuluokkaa. Opinnäytetyöntekijän kannalta WPF on hänelle tutumpi, mutta OPC UA ei niinkään. WinForms ei ole myöskään niin tuttu opinnäytetyöntekijälle, mutta jos tutkimustietoon on luottamista, sen avulla on helpompaa ja nopeampaa kehittää yksinkertainen käyttöliittymä, johonka voidaan liittää mm. tietokanta ja työkaluja, joilla pääsee PLC:n käsiksi.

Alustavasti opinnäytetyöntekijä lähtee rakentamaan sovellusta WinForms ohjelmointikehyksellä. Jos projektin elinkaaren aikana tulee tarvetta vaihtaa esimerkiksi WinFormsista WPF:lle, konvertointi onnistuu suhteellisen helposti. Kummatkin ohjelmistokehyksistä on rakennettu saman alustan ja standardin päälle ja kummasakin voidaan käyttää C#. Täten ohjelmistokehyksen vaihtaminen pitäisi onnistua ilman ongelmia. Ainoana haasteena koodin konvertoimisessa on WPF:n XAML. WinFormsin ”Back-end” on suoraan yhteensopiva WPF:n kanssa, mutta ”Front-end” pitää muokata C#:sta XAML:iin.

Jos tulevaisuudessa tulee tarvetta siirtyä muille alustoille, myös Qt ja GTK ovat mahdollisia vaihtoehtoa, mutta tässä vaiheessa on hyvä odottaa ja katsoa mihin suuntaan .NET Core ja .NET 5 ovat menossa. On turha spekuloida, mutta tulevaisuudessa Microsoft voisi tarjota todellisen alustariippumattoman ohjelmistokehyksen työpöytäsovelluksille. Microsoft on tuonut jo WinFormsin ja WPF:n .NET Core alustalle, jonka perimmäinen ajatus on alustariippumattomuus ja avoinlähdekoodi, mutta Microsoft on perustellut tätä tekoa .NET Coren paremmalla ohjelmarakenteella ja modernisuudella. WPF:n ja WinFormsin historia ja alustariippuvuus estää niiden siirtymistä toisille alustoille, mutta niiden pohjalta voitaisiin rakentaa uusi kehys, joka soveltuisi monelle eri alustalle. Tämän kaltainen projekti on jo aloitettu ohjelmistoyhteisön puolella, ja sitä kutsutaan Avalonia UI:ksi. Se on rakennettu WPF:n pohjalta, ja toimii myös muilla käyttöjärjestelmillä, mutta se on epävirallinen ja Microsoft ei ole itse missään tekemisissä tämän projektin kanssa. Se, että luoko Microsoft alustariippumattoman ohjelmistokehyksen jää nähtäväksi.

Lopuksi käsitellään projektissa käytettyä ohjelmistoympäristöä. Edellisissä kappaleessa avattiin syitä, miksi projekti kirjoitetaan C#-ohjelmointikielellä ja WinForms-ohjelmointikehyksellä. Nämä tekniikat pyörivät .NET-ympäristössä, ja Microsoft on optimoinut Visual Studion tämänkaltaisille projekteille ja ympäristöille. Microsoft tarjoaa ilmaisen Community-version Visual Studiosta, ja tätä hyödynnetään tämän projektin ohjelmistoympäristönä.

3.3.3 Tietokanta

Tietokanta on tärkeässä osassa onnistunutta käyttäjäkokemusta. Sen pitää pystyä toimimaan nopeasti ja turvallisesti, jotta käyttäjän ei tarvitsisi odotella hakujen tuottamia tuloksia tai tiedon joutumista vääriin käsiin. Tämä kaikki voidaan saavuttaa onnistuneella tietokanta hallintajärjestelmän valinnan avulla. Jokainen hallintajärjestelmä sopii omaan tarkoitukseen ja soveltuu parhaiten tiettyihin

tuotteisiin ja yrityksiin. Pienemmissä projekteissa on mahdollista käyttää kaikkia tutkimuksessa mainittuja tietokantoja. Isommissa projekteissa voidaan käsitellä paljon arkoja tietoja ja hakuja voidaan toteuttaa pahimmassa tapauksessa monia satoja sekunnin aikana.

Tietokanta hallintajärjestelmän valinta aloitetaan tietokantamallista. Tämän tarkoittaa käytännössä joko SQL relaatiomallin tai NoSQL ei-relaatiomallin käyttämistä. Kummatkin malleista tarjoavat tehokkaan tietokannan, mutta tekniset ja rakenteelliset erot pitää ottaa huomioon ennen tietokannan implementointia sovellukseen. SQL tarjoaa kypsän ja hyvin dokumentoidun tietoarkiston, laajan yhteisön sekä kolmannen osapuolien kirjastoja. Relaatiomallit ovat myös luotettavia, koska niiden rakenne noudattaa ACID-standardia. Tämä estää mm. tiedon häviämistä tietokannasta tai sen korruptoitumista, ja pitää tietokannan mahdollisimman eheänä. Relaatiomallin heikkoutena voidaan pitää sen joustamattomuutta ja monimutkaisuutta. Joustamattomuus syntyy siitä, että tietokannan kaikkia ”soluja” ei voida täyttää aina tiedolla, jolloin syntyy turhia soluja tietokantaan. Monimutkaisuus syntyy sekä itse relaatiomallin ajatusmaailmasta että tietokannan kasvaessa liian isoksi. (Harkushko, 2020)

NoSQL:llä voidaan nykyisin pitää pätevänä kilpailijana SQL:lle. Se tuo monia muita vaihtoehtoja sovelluskehittäjille, joilla voidaan rakentaa myös laadukkaita tietokanta hallintajärjestelmiä. NoSQL etuna on sen joustavuus, jota SQL:ssä ei niinkään ole tarjolla. Eri tietotyyppejä voidaan tallentaa ja käsitellä monella eri tavalla. Tämä tuo sovelluskehittäjän repertuaariin lisää joustavuutta ja skaalautuvuutta tietokannan suhteen. Toisaalta NoSQL:n heikkoutena pidetään sen nuorta ikää ja dokumentaation nihkeyttä verrattuna SQL:ään. NoSQL ei myöskään noudata ACID-standardia kaikissa tapauksissa, johtuen sen joustavuudesta ja skaalautuvuudesta. Pitää myös muistaa, että NoSQL tarjoaa monia erilaisia malleja, jotkut mallit soveltuvat eri tarkoituksiin paremmin kuin toiset mallit. Eli NoSQL sisältää monta erilaista mallia, kun taas SQL sisältää relaatiomallin pääsääntöisesti. (Harkushko, 2020)

Vuonna 2011 esitettiin uusi termi nimeltä NewSQL, joka on määritelty relaatiomalliseksi, mutta edustaa sekä NoSQL että SQL. NewSQL:n tarkoituksena on yhdistää SQL:n ja NoSQL:n parhaimmat ominaisuuden yhdeksi kokonaisuudeksi. NewSQL hyödyntää SQL:n puolelta sen tehokasta tapahtumien prosessointia ja NoSQL:n joustavuutta, skaalautuvuutta ja palvelimetonta arkkitehtuuria. NewSQL noudattaa myös ACID-standardia, joka on merkittävä osa onnistuneen tietokannan rakentamista. NewSQL on vielä tekniikkana hyvin epäkypsä, eikä se ole saavuttanut suurta suosiota tietokantayhteisön piireissä, vaikka ajatus onkin hyvinkin lupaava. Tekniikan kehittyessä NewSQL:n läpimurto tulee entistä ajankohtaisemmaksi, mutta tämän projektin kannalta NewSQL ei nähdä relevanttina vaihtoehtona. (Harkushko, 2020)

Lopuksi on tärkeätä ottaa muutama muu tekijä huomioon tietokantaa valittaessa. Mitä tietotyyppejä tullaan käsittelemään tietokannassa? Pitääkö tietokannan pystyä skaalautumaan sovelluksen kasvaessa entistä isommaksi, ja mihinkä suuntaan sen pitää pystyä skaalautumaan, vertikaalisesti tai horisontaalisesti? Onko tietokannan turvallisuus kuinka oleellinen asia sovelluksen käytön kannalta, ja onko kolmansien osapuolien työkalujen tarvetta päästä tietokantaan käsiksi?

Opinnäytetyöntekijälle tarjottiin vapaat kädet ensimmäisen tietokannan valinnan suhteen. Kaikkien edellä mainittujen tietokanta hallintajärjestelmistä valinta kohdistui PostreSQL:n. Se soveltuu parhaiten prototyypin rakentamiseen alustavasti, ja tarvittaessa hallintajärjestelmää voidaan vaihtaa sovelluksen kehittyessä entistä pidemmälle. Opinnäytetyöntekijän mielestä relaatiomalli voisi sopia erinomaisesti tämän kaltaiseen sovellukseen, koska tietokannan rakenne pysyy aina samana. Sovellukseen pitää pystyä tallentamaan esimerkiksi teloja ja mittaustuloksia, jotka ovat rakenteellisesti aina lähes samanlaisia. PostreSQL tarjoaa myös laajan valikoiman eri tietotyyppejä ja hyvät turvallisuus ominaisuudet, joita voidaan hyödyntää sovelluksessa. PostgreSQL:n heikkoutena on pidetty sen huonoa suorituskykyä, mutta tässä vaiheessa sovelluksen elinkaarta, tietokannan rakenne tulee pysymään minimaalisena, joten riittää että hakumoottori pystyy hakemaan tarpeeksi nopeasti, että se ei tuottaisi negatiivista käyttäjäkokemusta.

4 PLC-OHJELMOINTI JA SEN OHJELMOINTIYMPÄRISTÖ

Tässä luvussa käsitellään PLC-ohjelmoinnin perusteita ja ohjelmointiympäristöjä Siemensin alustoilla. Opinnäytetyöntekijä koki, että tämä on olennainen osa hänen omaa oppimistaan, koska hänellä ei ole mitään aikaisempaa kokemusta automaatiosta ja sen kehittämisestä. Tämä on olennainen osa projektia ja samalla tutustutaan siihen, kuina opinnäytetyöntekijä on itsenäisesti opetellut PLC-ohjelmoinnin hyödyntäen Udemy-opetusalustalla olevaa verkkokurssia. Verkkokurssiksi valikoitui Paul Lynnin ”Siemens TIA Portal Step 7 WinCC PLC HMI (PLC-S)”-verkkokurssi. Kurssissa käsitellään kakkia IEC:n (International Electrotechnical Commission) määrittelemiä PLC-ohjelmointikieliä ja tutustutaan HMI-käyttöliittymien rakentamiseen WinCC:n avulla.

Udemy on verkkosivusto, joka tarjoaa maksullisia ja ilmaisia verkkokursseja kaikille ihmisille, mutta kohderyhmänä ovat aikuiset ja opiskelijat. Alusta tarjoaa ammattilaisille ja opettajille mahdollisuuden luoda verkkokursseja, joissa voidaan jakaa opiskelijoille tietoa videoiden, dokumenttien ja pienien kokeiden muodossa. Opinnäytetyöntekijällä on aikaisempia kokemuksia tästä alustasta, ja sivustolta löytyy laaja valikoima hyviä verkkokursseja, jotka soveltuvat erinomaisesti itseopiskeluun. Sillä on monia etuja verrattuna perinteisiin opetusmenetelmiin. Esimerkiksi opinnäytetyöntekijä koki, että videoiden tauotus mahdollisuus parantaa huomattavasti hänen keskittymiskykyään. Aina kun keskittyminen meinasi herpaantua, niin video pysäytettiin ja tehtiin jotakin muuta, jonka jälkeen videon katselua pystyi jatkamaan.

4.1 Siemens TIA Portal

TIA Portal, lyhenne muodostuu sanoista ”Totally Integrated Automation Portal”, on Siemensin tarjoama kehitysalusta, jonka tarkoituksena on yhdistää monia erilaisia sovelluksia yhdeksi suureksi sovellukseksi. Sen avulla kehittäjät pystyvät rakentamaan yhdellä alustalla lähes kaikki tarvittavat asiat teollisuusautomaatioon liittyen. TIA Portal on rakennettu seuraavista ohjelmistoista ja eri lisenssipaketteja (PLC Programming, 2018):

- STEP 7: Tarkoitettu PLC-laitteiden ohjelmoimiseen.
    - Basic, Professional, ja valinnainen Safety Advanced.
- WinCC: HMI ja muiden ohjauslaitteiden suunnitteluun ja ohjelmoimiseen.
    - Basic, Comfort, Advanced ja Professional.
- Startdrive: Moottorikäyttöön tarkoitettu ohjelma.
- SCOUT TIA: Liikeohjaukseen tarkoitettu ohjelma.

Vaikka TIA Portal on tehokas ja monipuolinen työkalu teollisuusautomaatiossa, sen oppimiskäyrä on jyrkkä ja se vaatii paljon aikaa ja opettelua. Myös monipuolisuus voi toimia negatiivisena asiana, koska se luo monimutkaisen rakenteen sovellukselle, jonka seurauksena syntyy herkästi huonoja käyttäjäkokemuksia kehittäjille.

4.2 IEC 61131-3-standardi ja ohjelmointikielet

IEC, lyhenne muodostuu sanoista ”International Electrotechnical Commission”, on organisaatio, joka luo maailmanlaajuisesti standardeja, yhdessä sisaryhtiöidensä kanssa ISO (International Organization for Standardization) ja ITU (International Telecommunication Union) kanssa. IEC keskittyy luomaan standardeja elektroniikkaan, automaatioon ja sähkölaitteisiin. IEC 61131-3 on osa laajempaa IEC 61131-standardia, jossa käsitellään ohjelmoitavia logiikoita. Se on ainoa globaali standardi, jossa käsitellään automaatioteollisuuden ohjelmointia. IEC 61131-3-standardissa käsitellään tarkemmin ohjelmien arkkitehtuurisia rakenteita ja mitä ohjelmointikieliä voidaan käyttää PLC:n ohjelmoimisessa. Sen tarkoituksena on yhdenmukaistaa ihmisten luomia suunnitelmia ja käytössä olevia teollisuusohjauksia, luomalla yhtenäinen ohjelmointirajapinta. Tämä mahdollistaa monen eri sovelluskehittäjän ja ajatusmaailman yhdistämistä yhdeksi standardoiduksi kokonaisuudeksi, jota voidaan hyödyntää monessa eri vaiheessa sovelluskehitystä. Standardissa käsitellään seuraavia ohjelmointikieliä: (IEC, 2020; PLCopen, 2018)

- Ladder Diagram (LD, LAD), Tikapuulogiikka.
- Function Block Diagram (FBD), Toimintolohkokaavio.
- Structured Text (ST) / Structured Control Language (SCL), Jäsennelty teksti.
- Instruction List (IL) / Statement List (STL), Ohjelista.
- Sequential Function Chart (SFC, GRAPH), Sekvenssinen toimintokaavio.

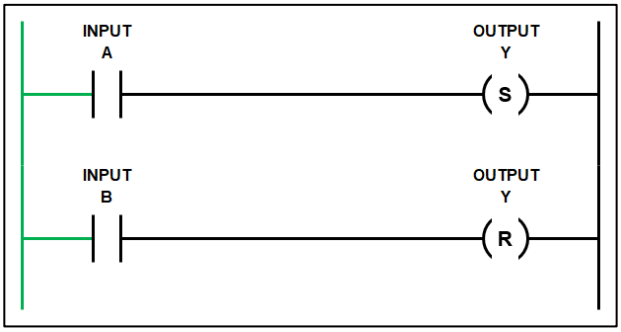
Standardissa otetaan kantaan: ohjelmoinnin vähimmäisvaatimuksiin, sen peruselementteihin, syntaksisiin ja semanttisiin sääntöihin yleisimmille ohjelmointikielille. Noudattamalla tätä standardia ja hyviä ohjelmointitapoja, saadaan ohjelmakoodista mahdollisimman looginen, modulaarinen, helppolukuinen ja virheetön. Standardia on myös kritisoitu, koska siinä ei määritellä loogisten komponenttien standardisoitua käyttöä. Tällä tarkoitettaan sitä, että standardissa kyllä kerrotaan minkälaisia ohjelmointikieliä ja minkälaista syntaksia ohjelmassa pitäisi olla, mutta esimerkiksi ajastimien ja muiden komponenttien käytöstä ei kerrota. Tämä voi luoda erimielisyyksiä kahden tai useamman ohjelmoijan kesken, koska standardissa ei käsitellä, mikä on hyvä ja oikea tapa ohjelmoida. (PLCopen, 2018; Bacidore, 2018)

Vaikka moni valmistaja noudattaa tätä standardia, se on myös aiheuttanut suuria eroavaisuuksia valmistajien kesken. Moni valmistaja tukee edellä mainittujen ohjelmointikielien perusrakennetta, mutta jokainen valmistaja on tuonut myös omia lisäominaisuuksia näihin ohjelmointikieliin. Tämä on johtanut siihen, että kahden valmistajan ohjelmointikielet voivat muistuttaa toisiaan, mutta toisen valmistajan ohjelmasta voi löytyä jokin komponentti, mitä toisen ohjelmasta ei löydy tai on toteutettu toisenlaisella komponentilla. Tämä voi vaikuttaa suuresti, jos asiakkaalla on kahden tai useamman eri valmistajan laitteita käytössä. Tämä on yksi merkittävä syy, miksi uuden kokonaisuuden suunnittelussa halutaan käyttää ainoastaan yhden valmistajan laitteita. (Bacidore, 2018)

4.2.1 Ladder Diagram – Tikapuulogiikka

Tikapuulogiikka on vanhimpia ohjelmointikieliä tässä standardissa. Se kehiteltiin muutama vuosikymmentä sitten korvaamaan vanhoja relejärjestelmiä. Se on yksi suosituimmista ohjelmointikielistä maailmalla, ja on monen teollisuusautomaatio ohjelmoijan ensimmäinen ohjelmointikieli. Sitä käytetään erityisesti Yhdysvalloissa. On jopa väitetty, että noin 95 % kaikista sovellutuksista Yhdysvalloissa, on toteutettu tikapuumenetelmällä. Vaikka tikapuulogiikka on suosittu, sen suosio on laskussa, mutta sitä pidetään edelleen oletuskielenä teollisuusautomaatiossa. Se pohjautuu graafiseen ohjelmointiin, ja sen rakenne muistuttaa tikapuita. (Rexroth Bosch Group, 2009)

Tikapuulogiikassa ohjelmaa aloitetaan suorittamaan ylhäältä alas ja vasemmalta oikealle. Eli mitä aikaisemmin halutaan suorittaa jokin asia, sitä ylemmäs ohjelmassa se pitää sijoittaa. Ohjelman vasemmalle puolelle sijoitetaan ehtoja ja oikealle puolelle suoritukset. Kuvassa 35 on yksikertainen tikapuulogiikka esimerkki. (Bacidore, 2018; Rexroth Bosch Group, 2009)



KUVA 31. Yksinkertainen tikapuulogiikka. Vasemmalla on kaksi ehtoa ja oikealla on kaksi lopputulemaa. ”S” tarkoittaa set, eli asettamista, ja ”R” tarkoittaa restart, eli nollaamista. Kun ”A” ehto täyttyy, ”Y” asetetaan päälle. Kun ”B” ehto täyttyy, ”Y” nollataan. (Ladder Logic World, 2019)

Tikapuulogiikka soveltuu erinomaisesti pienempiin projekteihin, missä ei ole tarvetta kirjoittaa paljon ohjelmakoodia. Sen suuri suosio takaa sen, että moni teollisuusautomaatiossa oleva ohjelmoija pystyy muokkaamaan ja lukemaan tätä ohjelmointikieltä. Sen oppimiskäyrä on loiva muihin ohjelmointikieliin verrattuna. (Rexroth Bosch Group, 2009)

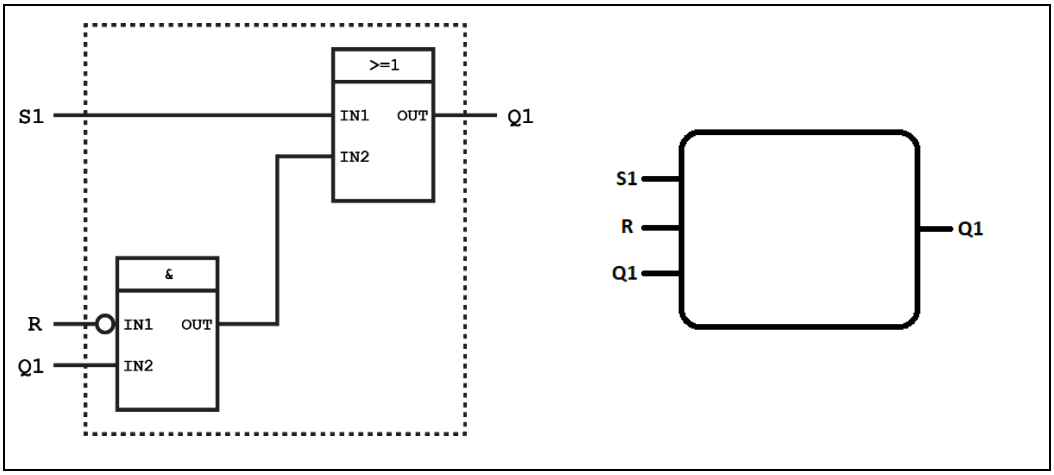
Tämän ohjelmointikielen ei sovellu isoihin projekteihin, koska ohjelmakoodin kasvaessa, se muuttuu ”spagettikoodiksi”, joka vaikeuttaa ohjelman lukemista ja rakentamista. Pahimmillaan ulkopuolinen ohjelmoija voi käyttää rutkasti työaikaa selvittääkseen ohjelman perusrakenteen. Tikapuulogiikka ei

sovellu myöskään algoritmien ja matemaattisien kaavojen rakentamiseen, koska se syö paljon tehoa ja tuo lisää kompleksisuutta ohjelmaan. (Bacidore, 2018; Rexroth Bosch Group, 2009)

4.2.2 Function Block Diagram – Toimintolohkokaavio

Toimintolohkot ja toimintolohkokaaviot soveltuvat erinomaisesti tikapuulogiikan kanssa. Sillä pystytään helpottamaan tikapuulogiikan rakennetta ja luomaan yksinkertaisia ja tehokkaita toimintoja ohjelmaan. Toimintalohkokaavio on myös erillään hyvin tehokas ohjelmointikieli, ja sen suosio on edelleen kasvussa. (Bacidore, 2018)

Toimintalohkokaavio on graafinen ohjelmointikieli, ja se muistuttaa etäisesti tikapuulogiikkaa. Tässä ohjelmointikielessä luodaan ”palikoita”, joiden sisälle rakennetaan logiikkaa. Alla olevassa kuvassa oikealla on yksinkertainen toimintalohkokaavio, ja vasemmalla on toimintalohkokaavion logiikka sisällä. Rakenteellisesti se on hyvin yksinkertainen: tuoduista sisääntuloista saadaan tietty ulostulo.



KUVA 32. Oikealla kuvassa on toimintalohkokaavio ja vasemmalla on sen sisällä oleva logiikka. (PLC Academy, 2018)

Tämän ohjelmointikielen etuna pidetään sen yksinkertaisuutta ja modulaarisuutta. Toimintalohkokaavio kärsii samoista ongelmista kuin tikapuulogiikka, se ei sovellu suuriin ohjelmiin eikä tietyntyyppisiin sisään-/ulostuloihin ja funktioihin. Se vaatii myös laajaa suunnittelua, koska kaikki tiedot pitäisi olla tiedossa, jotta ”palikan” voisi luoda. Kolmannen osapuolen kirjastot suosivat toimintalohkoja, koska niiden taakse pystytään helposti lukitsemaan funktioita, joita voidaan myydä haluaville asiakkaille. (Rexroth Bosch Group, 2009)

4.2.3 Structured Text – Jäsennelty teksti

Structured Text on tekstipohjainen ohjelmointikieli, joka on yksi suosituimmista kielistä nykyaikana. Se soveltuu erinomaisesti kaiken kokoisiin ohjelmiin, raskaiden algoritmien ja matemaattisien kaavojen ohjelmoimiseen, koska sillä pystytään saavuttamaan tarvittava laskentateho näiden luomiseen. Sitä myös hyödynnetään usein SFC:n kanssa, jolloin Structured Textillä kirjoitetaan itse ohjelmoitava logiikka ja SFC:n avulla ohjelmoidaan erilaisia logiikkatiloja. Tarvittaessa, koodista saadaan myös

modulaarinen tekemällä pieniä muutoksia, jolloin sitä voidaan soveltaa useammassa projektissa. (Bacidore, 2018)

Jäsennelty teksti muistuttaa suuresti tavallista sovellusohjelmointia. Siinä käytetään paljon if-lauseita ja silmukoita. Tämä ohjelmointikieli soveltuu erinomaisesti sovelluskehittäjille, ketkä ovat perehtyneet tai tulevat tietotekniikan puolelta, missä ohjelmia kirjoitetaan esimerkiksi C tai PASCAL ohjelmointikielillä. Tämä suurentaa potentiaalisien ohjelmoijien haarukkaa, koska ohjelmoijia voidaan hakea myös IT-alalta. ST ei sovellu ensimmäiseksi ohjelmointikieleksi, koska sen lukeminen on haasteellista aloittaville ohjelmoijille ja graafiset ohjelmointikielet soveltuvat paremmin ensimmäiseksi ohjelmointikieleksi. Uusien ohjelmoijien on myös vaikeata omaksua tekstipohjaista koodia, koska se vaatii paljon keskittymistä ja rakenteen seuraamista. Varsinkin jos koodia ei ole kommentoitu, pahimmillaan uudet ohjelmoijat voivat käyttää useita työpäiviä, jos kohdeprojektin koodimäärä on laaja. (Rexroth Bosch Group, 2009)

```
PROGRAM stexample           // Luodaan ohjelma

    VAR                     // Aloitetaan muuttujien määrittely
        x : BOOL;           // Luodaan bool-muuttuja x
    END_VAR                 // Lopetetaan muuttujien määrittely

    x := TRUE;              // Asetetaan muuttujan arvo

    REPEAT                  // Aloitetaan toistosilmukka
        x := FALSE;         // Asetetaan muuttujan arvo
    UNTIL x := FALSE;       // Toistetaan, kunnes x:n arvo on FALSE
    END_REPEAT              // Lopetetaan toistosilmukka

END_PROGRAM;                // Lopetetaan ohjelma
```

KOODINPÄTKÄ 1. Yksinkertainen ”Structured Text”-koodinpätkä, missä luodaan muuttuja, alustetaan se ja muokataan muuttujaa silmukassa. (PLC Academy, 2015)

4.2.4 Instruction List – Ohjelista

Ohjelista on tekstipohjainen ja matalantason PLC ohjelmointikieli. Sitä ei juurikaan käytetä teollisuudessa enää, mutta käyttökohteina ovat nopeasti ja laskentateho vaativissa laitteissa. IL on yksi ensimmäisistä ohjelmointikielistä yhdessä tikapuulogiikan kanssa. Se ei sovellu juurikaan missään määrin aloittelijoille, koska siinä on korkea oppimiskäyrä, haastavaa lukee ja sen tuottama hyöty tavallisissa sovellutuksissa on pienehkö. Myös IL:n uudelleen ohjelmoiminen on haastavaa, jos ohjelmoija ei ole aikaisemmin ollut tekemisissä ohjelman kanssa. Se vaatii paljon sisäänajoa ja voi vaatia pahimmillaan useita työpäivä huoltotyöntekijältä. Vaikka IL on tekstipohjainen ohjelmointikieli, se eroaa ST ohjelmointikielestä suuresti. ST:n avulla on suhteellisen helppo ohjelmoida matemaattisia kaavoja, mutta IL:n avulla se on haastavaa. (Bacidore, 2018; Rexroth Bosch Group, 2009)

IL:n ohjelmarakenne on mielenkiintoinen ja siinä pitää ymmärtää matalantason ohjelmoinnin ajatusmaailmaa. ST mahdollistaa yhdellä ohjelmointirivillä tekemään monta eri asiaa, esimerkiksi summataan kaksi muuttujaa yhteen ja tallennetaan tulos erilliselle muuttujalle. IL ohjelmoinnissa tämän

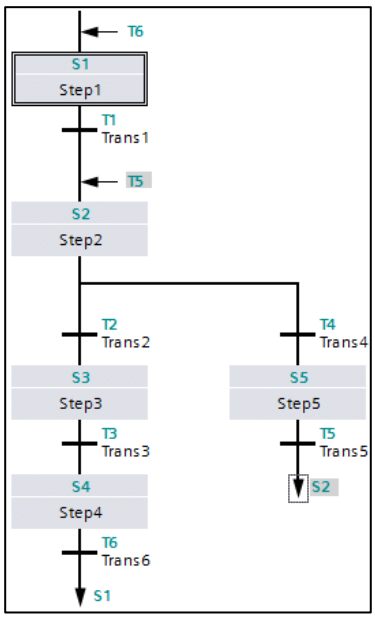
kaltainen ohjelmointi ei toimi. Jokaisella ohjelmarivillä toteutetaan yksi tietty toiminto, eli ohjelmanrakenne muuttuu hyvin pitkäksi sarakkeeksi, josta tulee myös ohjelmointikielen nimi ”ohjelista”. Jos esimerkiksi ST-koodi konvertoitaisiin IL ohjelmointikieleen, niin ST-koodissa yksi ohjelmarivi pitäisi muokata kolmeksi IL:ssä. Aluksi pitäisi ajaa ensimmäinen muuttuja ”sisään”, tämän jälkeen ensimmäiseen muuttujaan summataan toinen muuttuja ja lopuksi siirretään arvo kolmanteen muuttujaan. Koodinpätkässä 2, alemmassa esimerkissä toteutetaan tämänkaltainen muutos. (Lynn, 2019)

```
// ===== IL Esimerkki 1 =====
// ==========================
// ==========================

   A   "Muuttuja1"                        // A = AND(&&). Tarkistetaan
                                          // onko muuttuja tosi

   JC  Muuttuja1OnTosi                    // Jos edellinen muuttuja on
                                          // tosi, hypätään "funktioon"

   Muuttuja1OnTosi : S "Muuttuja1OliTosi" // Asetetaan Muuttuja1OliTosi todeksi
   S   "Muuttuja2"                        // Asetetaan Muuttuja2 todeksi


// ===== IL Esimerkki 2 =====
// ==========================
// ==========================

   L   "Arvo1"                            // Ajetaan Arvo1 sisään
   +I  "Arvo2"                            // Lisätään (integer) Arvo2 Arvo1:een
   T   "Arvo3"                            // Siirretään saatu summa Arvo3:een
```

KOODINPÄTKÄ 2. Kaksi esimerkkiä IL ohjelmointikielestä.

4.2.5 Sequential Function Chart – Sekvenssinen toimintokaavio

Sekvenssinen toimintokaavio muistuttaa suuresti vuokaaviota, jossa tutkitaan, onko ehto tosi vai epätosi ja toimitaan sen mukaisesti eteenpäin kaaviossa. Ensimmäinen toimintalaatikko toimii alustuksena, jonka jälkeen toimintalaatikosta aloitetaan prosessimuuttujien seuranta. SFC tukee muita edellä mainittuja ohjelmointikieliä, ja se soveltuu erinomaisesti kokonaisprosessin rakentamiseen. Esimerkiksi tikapuulogiikalla voidaan rakentaa haluttuja ohjelmia, jotka voidaan lisätä myöhemmin osaksi tätä toimintakaaviota. (Rexroth Bosch Group, 2009)



KUVA 33. GRAPH-ohjelmointikielen esimerkki koodinpätkä.

Siemensin GRAPH-ohjelmointi kieli eroaa hieman standardin omasta kielestä. Runko on sama, mutta Siemens on lisännyt omia ominaisuuksia ohjelmointikieleen. Oikealla olevassa kuvassa on esimerkki Siemensin GRAPH-ohjelmointikielestä. Siinä näkyy toimintalaatikot ”Step” ja ehtoja ”Trans”. Toimintalaatikossa suoritetaan jonkin tietty toimintologiikka, joka voidaan ohjelmoida esim.

tikapuulogiikalla. Ehdoilla seurataan, että kaikki tarvittavat ehdot toimintalaatikon logiikan toteuttamiseen ovat saavutettu. (Lynn, 2019)

Sekvenssinen toimintakaavio soveltuu erinomaisesti kokonaiskuvan rakentamiseen, sen avulla loppukäyttäjät pystyvät hahmottamaan prosessin rakenteen. Sen tuoma modulaarisuus on myös erinomainen ominaisuus, jota voidaan hyödyntää myös muissa projekteissa tai projekteissa missä on paljon toistoa.

4.2.6 Verkkokurssilla käydyt aiheet

Lynn tarjosi laadukkaan ja laajan verkkokurssin, jossa käsiteltiin kaikki Siemensin tukemia ohjelmointikieliä ja HMI-käyttöliittymien rakentamisen perusteita. Kurssin aikana Lynn tutustutti opiskelijat jokaisen ohjelmointikielen perusteisiin ja niiden käyttökohteisiin. Kurssi aloitettiin tutustumalla TIA Portal-kehitysalustaan, jossa Lynn opasti projektien luomisessa ja niiden ajamista PLC:lle. Tämän jälkeen Lynn kävi jokaisen ohjelmointikielen läpi, kertomalla niiden erikoisuuksista, käyttökohteista ja komponenteista. Lopuksi Lynn asetti kurssin opiskelijoille loppuprojektin, jossa suunniteltiin voimalaitoksen automaatiota. Projekti suunniteltiin jokaisella ohjelmointikielellä ja kurssin lopussa jokainen opiskelija oli toteuttanut voimalaitoksen automaation kaikilla ohjelmointikielillä. Lynn käsitteli myös HMI-käyttöliittymän rakentamista kurssilla, mutta kurssin painoarvo jäi enimmäkseen ohjelmointiin. Kurssilla käsiteltiin seuraavia aiheita:

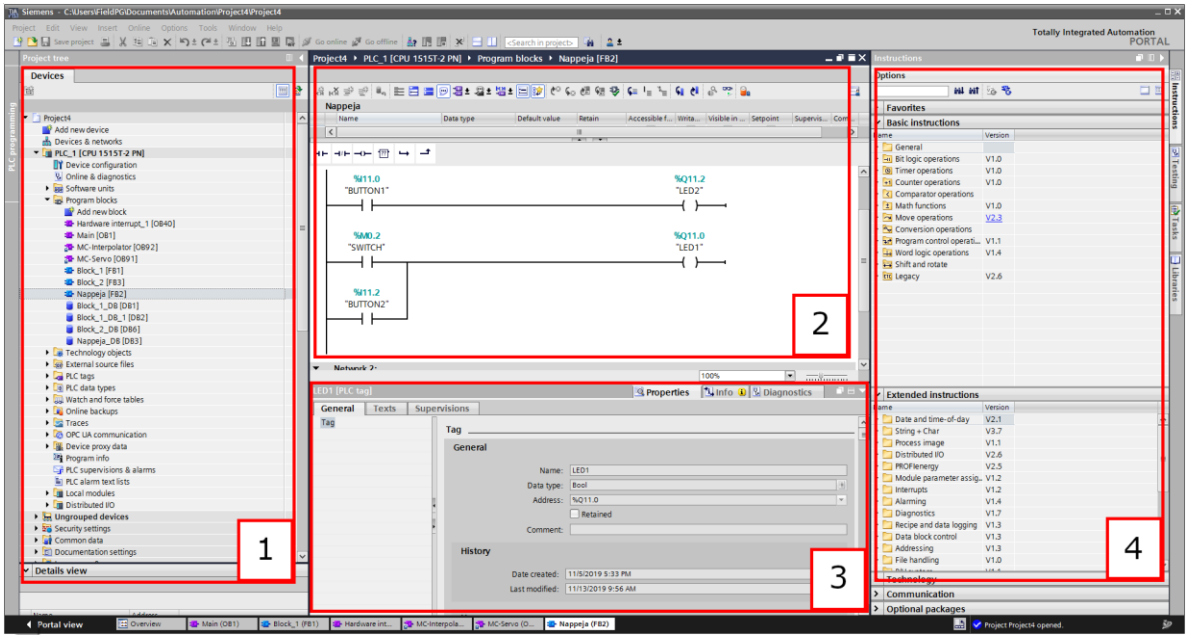
- Ohjelmointi:
    - Muistinkäsittelyä, eri muistityyppien käyttöä ja niiden käyttökohteita.
    - Osoitteiden ja PLC ”tagien” käyttöä.
    - Topologian ja verkon rakentamista TIA Portal ympäristössä.
    - Vianetsintää.
    - Logiikkakomponentteja: Välityslogiikka, bittien muokkausta, ajastimien käyttöä, laskureiden käyttöä, vertailijoiden käyttöä, matemaattisten komponenttien käyttöä ja monia muita olennaisia komponentteja.
    - PID (Proportional Integral Derivative) perusteet.
    - Function Block ja niiden hyödyntämistä muualla.
    - Graafisen ja tekstipohjaisen ohjelmoinnin perusteita.
    - Hälytysten ja ilmoitusten luonti, ja niiden käyttö.
- HMI:
    - UI ja UX
    - Ohjelman ja HMI:n välinen yhteys ja sen rakentaminen
    - Grafiikkakirjastot
    - Käyttöliittymän rakentamisen ja käyttöönoton perusteet

5 TESTILAITE

Testilaitteen rakentamisessa hyödynnetään aikaisempia tietoja ohjausjärjestelmistä ja osittain tutkimuksesta saatua tietoa. Luvussa 3.3 käsitellään tarkemmin testilaitteen suunnittelua ja kuinka siihen on päädytty. Tässä luvussa tutustutaan tarkemmin itse testilaitteen rakentamiseen ja siihen liittyvistä tekniikoista.

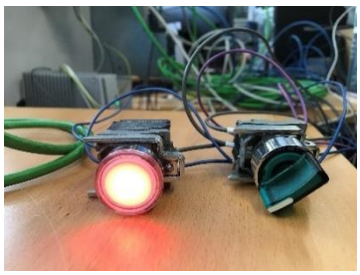
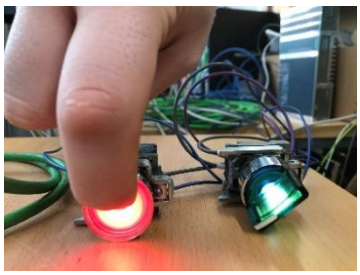
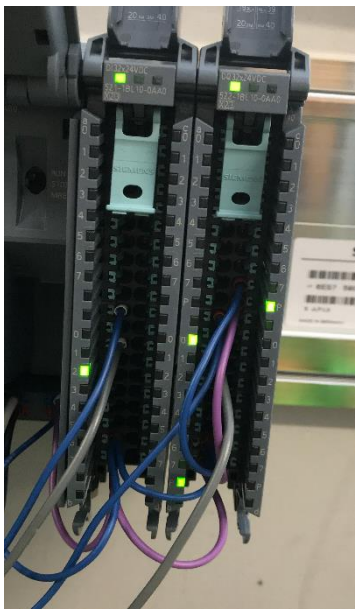
5.1 Ohjelmoitava logiikka ja kehitysympäristö

Aikaisemmissa ohjausjärjestelmissä tietokoneen paino on ollut suuri, koska PLC:n käsittelynopeudet eivät ole vastanneet silloisia vaatimuksia. Tämä on johtanut siihen, että PLC:n rooli itse järjestelmässä on ollut pieni ja tietokone on joutunut tekemään raskaita prosesseja, joka on karsinut myös tietokoneen nopeusresursseja. Nykyiset PLC ovat kehittyneet paljon muutaman vuosikymmenen aikana, ja monia uusia ominaisuuksia on tuotu uusille PLC laitteille. Niiden suoritustehoa on nostettu merkittävästi ja entistä enemmän niistä rakennettuja sovellutuksia löytyy teollisuusautomaation puolelta.



KUVA 34. TIA Portal kehitysympäristö. 1.) Projektivalikko. 2.) Ohjelmointiympäristö. 3.) Ominaisuusvalikko. 4.) Komponenttivalikko.

Ohjelmoitava logiikka on merkittävässä osassa tätä projektia, se on osa tiedon kulku tietokoneen ja anturin välillä, ja sen sisällä voidaan prosessoida erilaisia tietoja ja karsia tätä kautta tietokoneelle tuotua tietoa. Tässä työssä, PLC:ltä haetaan ainoastaan anturitietoa ja rakennetaan yksinkertainen ohjelma, missä voidaan syöttää kokonaislukuja muuttujiin ja simuloida hätäseis tilannetta painikkeiden avulla. Kuvassa 38 näkyy tikapuulogiikka, jossa napin painalluksella saadaan LED-valaisin sytytettyä. Seuraavassa kuvasarjassa nähdään, kuinka kyseinen ohjelma toimii käytännössä ja mitenkä I/O-moduulissa (sisään- ja ulostulomoduulissa) reagoidaan napin ja nokkakytkimen käyttöön.


KUVA 35. Vasemmalla näkyy, kuinka nokkakytkin laukaisee PLC koodin avulla toisen LED-valon päälle, ja kuinka I/O-moduuli reagoi tähän. Oikealla näkyy, kuinka napin painasu laukaisee ohjelman kautta toisen LED-valaisimen päälle.

5.2 Käyttöliittymä

Käyttöliittymä on olennainen osa tätä työtä. Sen avulla käyttäjän pitää pystyä ohjaamaan ja lukemaan PLC:n avulla järjestelmätietoja, kontrolloimaan moottoreita ja ohjaamaan muita oheislaitteita. Käyttöliittymästä pitää löytyä seuraavat ominaisuudet:

- Tietokanta, jonne voidaan tallentaa esim. tela- tai mittaustietoja
- Graafinen näkymä, jossa esitetään antureilta saatua tietoa
- Lukea ja kirjoittaa tietoa PLC:lle

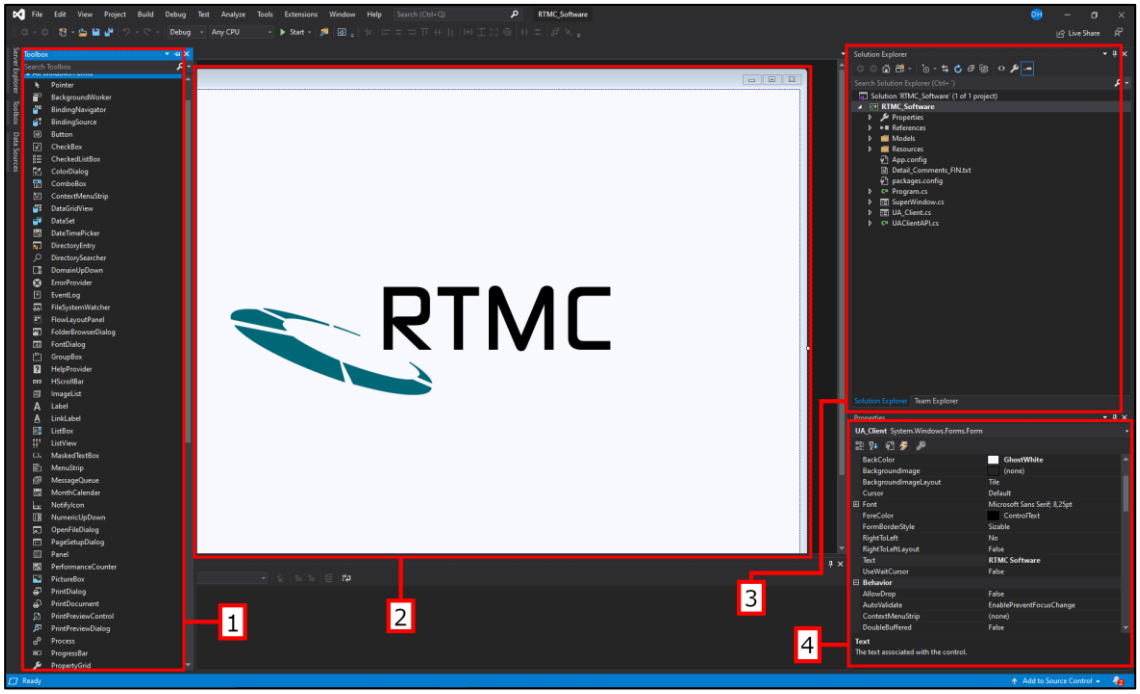
Tässä luvussa tullaan tutustumaan tarkemmin käyttöliittymän ohjelmointiin ja sen ympäristöön, tietokantaan ja sen yhdistämistä ohjelmaan ja edellä mainittujen ominaisuuksien kehittämistä ohjelmaan. Salassapito syistä, ohjelmakoodia ei tulla julkaisemaan tässä työssä missään muodossa raporttiin. Ainoastaan pieniä ja yleisiä pätkiä koodista tullaan näkemään tässä luvussa.

Käyttöliittymän rakentaminen aloitettiin luvun 3.3.2 suunnitelman pohjalta ja liitteestä numero kolme löytyy alustava hahmotelma käyttöliittymästä. Ohjelmointiympäristöksi valikoitui Visual Studio 2019,

ohjelmointikieleksi C# ja ohjelmointikehykseksi Windows Forms. Käyttöliittymän pohjustusta saadaan myös Siemensin avoimesta OPC UA -ohjelmasta, joka on rakennettu edellä mainittujen työkalujen avulla. Siemensin OPC UA -ohjelmassa käsitellään kaikkia OPC UA:n olennaisia ominaisuuksia .NET -ympäristössä, ja näitä ominaisuuksia voidaan hyödyntää sovelluksen rakentamisessa. Ohjelmassa käytetään OPC Foundationin tarjoamaa rajapintaa ja Siemensin UAClientHelperAPI apuluokkaa, josta kerrotaan tarkemmin luvussa 5.2.3.1.

5.2.1 Kehitysympäristö

Yhdessä Windows Formsin ja Visual Studion kanssa, on mahdollista rakentaa nopeasti ja suhteellisen helposti tavallinen käyttöliittymä, jos ulkonäkö ei ole tärkeässä roolissa. Windows Forms toimii yksinkertaisella ”vedä ja pudota”-menetelmällä, missä grafiikka komponentteja valitaan valikosta hiirellä ja tiputetaan ikkunalle. Tätä työkalua kutsutaan ”Windows Form Designer”, ja se on oleellinen osa ohjelmiston kehitystä. Se luo ulkonäkökomponenteista automaattisesti oikeanlaisia ja virheettömiä koodeja erilliseen tiedostoon. Tähän tiedostoon ei saa kirjoittaan erillistä koodia missään tapauksessa. Käsin kirjoitetut koodinpätkät eivät tallennu tähän tiedostoon oikealla tavalla, ja seuraavan ohjelmistorakennuksen aikana kaikki ylimääräinen koodi poistetaan tästä tiedostosta. Tästä syystä on suositeltavaa käyttää tätä graafistatyökalua ohjelmistokehityksen aikana.



KUVA 36. Visual Studio -kehitysympäristö. 1.) Komponenttivalikko. 2.) Sovellusikkuna. 3.) Tiedostovalikko. 4.) Komponenttien ominaisuusvalikko.

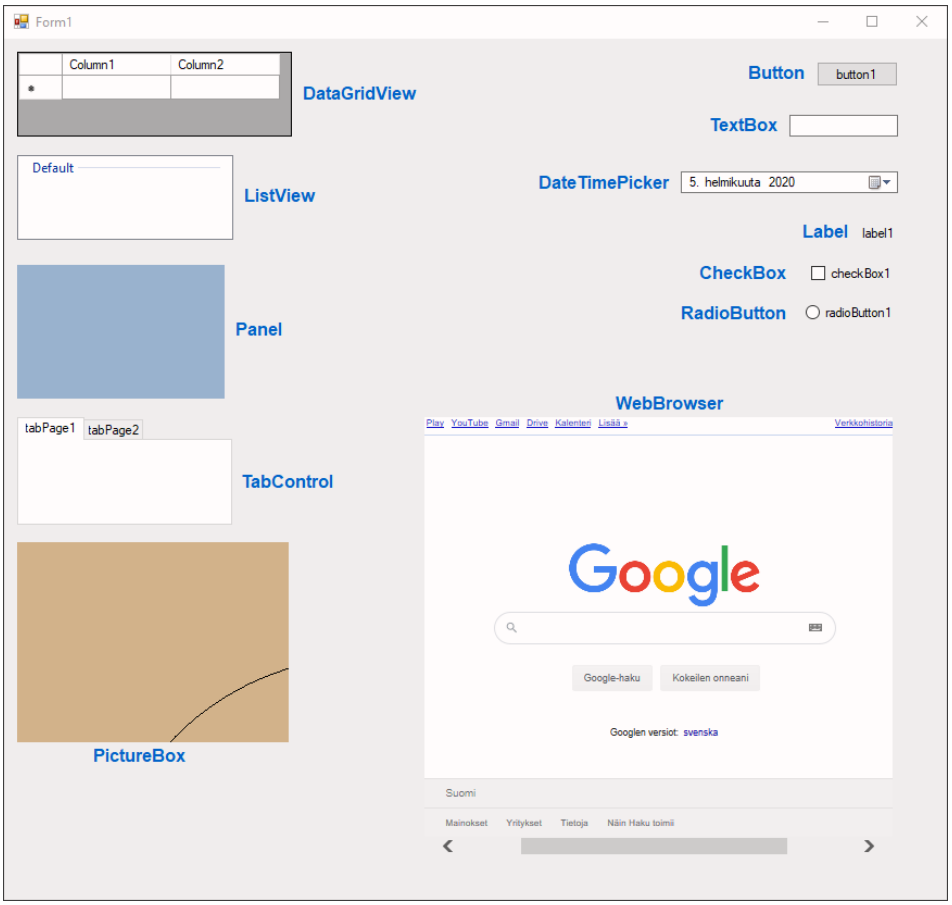
Kuvassa 40 löytyy tiedostovalikko, ikkunasta numero kolme. Tämä valikko näyttää ohjelman kokonaisrakenteen, josta löytyy kaikki oleellisimmat tiedostot ohjelmalle. Merkittävimmät tiedostot tässä projektissa olivat:

- Program.cs – Main-tiedosto, josta sovellus aloittaa toimintansa.
- UA_Client.cs – Käyttöliittymäikkunan kooditiedosto. Sisältää Designer tiedoston.

- UA_Client.Designer.cs - Koodi generoidaan automaattisesti hyödyntäen ”Windows Forms Designer”-työkalua.
- UAClientAPI.cs – Sovellusliittymä, jonka avulla sovellus kommunikoi PLC:n kanssa.
- Detail_Comments_FIN.txt – Tekstitiedosto, mihin on kirjoitettu tarkempia kommentteja sovelluksen toiminnasta ja sen rakenteesta.
- Models – Kansio, joka sisältää datamalleja.
  - NodeDatum.cs – Datamalli, jota käytetään PLC:n kommunikoinnissa.
  - RollInfo.cs – Datamalli, jota käytetään tietokannan kanssa.

5.2.2 Ulkonäkö ja sen rakentaminen

Alemmassa kuvassa näkyy erilaisia komponentteja, joilla voidaan rakentaa sovellusta. Kaikkia kuvassa näkyviä komponentteja ei käytetty sovelluksen kehittämisessä.



KUVA 37. Erilaisia Windows Formsin ohjelmistokehyksen tarjoamia komponentteja sovelluksen kehittämiseen.

Komponenttivalikosta voidaan valita monia erilaista grafiikkakomponentteja sovelluskehittämistä varten. Kuvassa 41 on muutama esimerkki komponentti, mitä voidaan käyttää sovelluskehityksessä. Projektissa käytettiin seuraavia grafiikkakomponentteja:

- Form: Ikkuna, minkä sisällä on mm. sovelluksen ikoni, nimi, ikkunan sulkemispainike ja muita ikkunaan liittyviä toimintoja.

- TabControl: Välilehden hallintakomponentti, joka sisältää välilehdet ja niihin liittyviä graafisia ominaisuuksia, kuten fontti ja väri.
- Button: Painike. Voidaan käyttää eri toimintojen laukaisemisessa.
- Label: Teksti. Näytetään eri tiloja tai tarkennetaan käyttäjälle eri toimintoja.
- CartesianChart: Kolmannen osapuolen grafiikkakirjasto, jonka tarkoituksena on visualisoida tietoa erilaisien graafien avulla. Kirjaston nimi on LiveChart ja se on ilmainen.
- TextBox: Tekstilaatikko, tai syöttökenttä, jonka tarkoituksena on vastaanottaa käyttäjältä saatua tietoa. Voidaan myös käyttää tiedon näyttämisenä käyttäjälle.
- ListView: Listanäkymä, jonka tarkoituksena on toimia eräänlaisena taulukkona.
- CheckBox: Valintaruutu, johon käyttäjä voi laittaa ruksin.
- GroupBox: Kehys, jonka tarkoituksena on visuaalisesti eristää tietyt komponentit yhteen pakettiin. Esimerkiksi tietyt syöttökentät voidaan kehystää yhdeksi ja nimetä ne tarvittaessa.
- DataGridView: Taulukko, joka muistuttaa suuresti Excel-taulukkoa. Ei ole suoraan samanlainen kuin aikaisemmin mainittu ListView.
- Panel: Piirtoalusta, johon voidaan piirtää erilaisia kuvioita.
- PictureBox: Kuvalaatikko, johon voidaan asettaa kuva. Toimii myös samalla tavalla, kun Panel-komponentti.

```
// Alustusfunktio
private void InitializeComponent()
{
    ...
    // Luodaan "tabControl" komponentti
    this.tabControl = new System.Windows.Forms.TabControl();
    ...
    // Keskeytetään komponentit toiminta
    this.tabControl.SuspendLayout();
    ...
    // Määritellään komponentin ominaisuuksia, esim. fonttia ja sen kokoa
    this.tabControl.Alignment = System.Windows.Forms.TabAlignment.Bottom;
    this.tabControl.Controls.Add(this.mainPage);
    this.tabControl.Controls.Add(this.settingsPage);
    this.tabControl.Dock = System.Windows.Forms.DockStyle.Fill;
    this.tabControl.Font = new System.Drawing.Font("Arial", 16F);
    this.tabControl.ImeMode = System.Windows.Forms.ImeMode.NoControl;
    this.tabControl.ItemSize = new System.Drawing.Size(0, 50);
    this.tabControl.Location = new System.Drawing.Point(0, 0);
    this.tabControl.Margin = new System.Windows.Forms.Padding(0);
    this.tabControl.Name = "tabControl";
    this.tabControl.RightToLeft = System.Windows.Forms.RightToLeft.No;
    this.tabControl.SelectedIndex = 0;
    this.tabControl.Size = new System.Drawing.Size(1300, 911);
    this.tabControl.TabIndex = 0;
    ...
    // Lisätään komponentti ikkunaan ja jatketaan sitä
    this.Controls.Add(this.tabControl);
    this.tabControl.ResumeLayout(false);
    ...
}
// Referenssi muuttujaan
private TabControl tabControl;
```

KOODINPÄTKÄ 3. Graafisenkomponentin perusrakenne alustusfunktiossa.

Yllä olevassa koodinpätkässä nähdään ”TabControl”-komponentin perusrakenne, joka noudattaa yleistä generointimenetelmää. Alustusfunktion alussa luodaan itse komponentti, jonka jälkeen sen toiminto keskeytetään. Tämän jälkeen komponentille kerrotaan kaikki tarvittavat tiedot sen rakentamiseen, kuten fontti, sijainti ikkunalla, mikä on komponentin isäntäkomponentti ja muita erilaisia ominaisuuksia. Lopuksi komponentti lisätään sen isäntäkomponenttiin ja käynnistetään uudelleen. Kuvan alareunassa näkyy myös referenssi, jonka avulla komponenttimuuttujaan voidaan viitata koodista.

5.2.3 Sovelluksen taustajärjestelmä

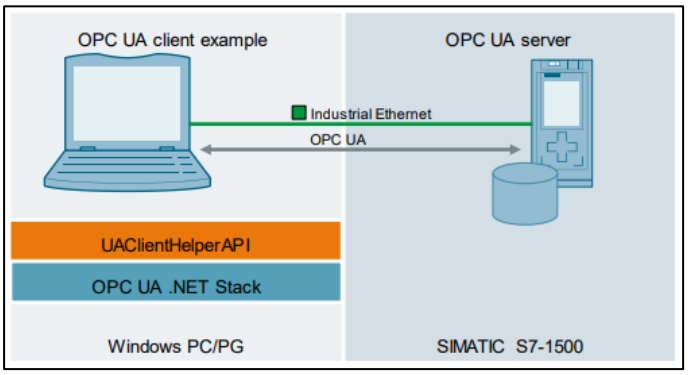
UA_Client.cs -tiedoston sisälle rakennettiin sovelluksen taustajärjestelmä, jonka tarkoituksena on prosessoida käyttäjän tuottamia tekoja ja tietoja. Esimerkiksi jos käyttäjä painaa nappia, niin taustajärjestelmä laukaisee painikkeeseen liitetyn tapahtumakäsittelijän, joka suorittaa sille määrätyn toimintoketjun. UA_Client.cs on rakennettu seuraavalla tavalla:

```
public partial class UA_Client : Form
{
    // Muuttujat ja niiden alustukset
    #region Variables

    // Pääfunktio. Alustetaan sovellus ja
    // UAClientAPI-rajapinta
    #region UA_Client() Init

    // Yhteyden rakentaminen PLC:lle ja
    // tarvittavien tietojen hakeminen sovellukselle
    #region PLC (OPC UA)

    // Kaavio, johon piirretään anturilta tulevia arvoja
    #region Chart

    // Tietokannan alustaminen ja käyttö
    #region Database

    // Alustus, jossa haetaan ja ajastetaan muuttujia
    #region RTMC Init

    // Piirtofunktio, jonka avulla piirretään
    // anturilta tulevia tietoja
    #region Draw

    // Käyttöliittymän grafiikkakomponenttien
    // uudelleenkäynnistys
    #region Reset UI

    // Hallintakäyttäjän luominen
    #region SuperUser Login

    // Ajastimen luonti ja hallinnointi
    #region Timer

    // Tapahtumakäsittelijät ja muita projektissa
    // käytettyjä funktioita
    #region EventHandlerit ja muut funktiot
}
```

KOODINPÄTKÄ 4. Taustajärjestelmän rakenne projektissa.

5.2.3.1 OPC UA ja UAClientHelperAPI

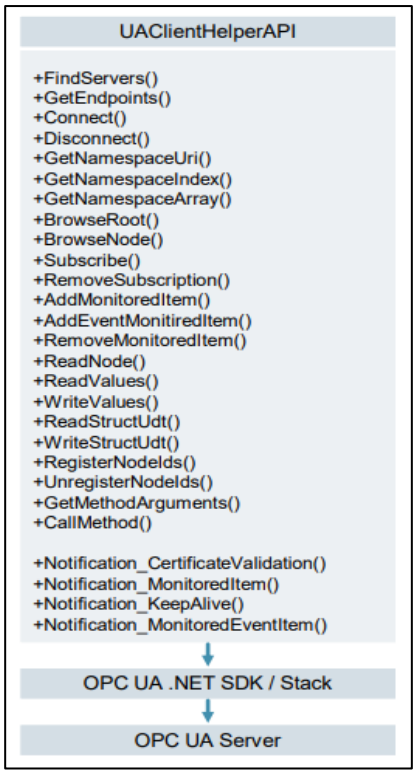
Luvuissa 3.2.4 ja 3.3.1 tarkasteltiin ja käsiteltiin tarkemmin OPC UA -standardia ja tässä luvussa tutustutaan tarkemmin sen käyttöä tässä projektissa. OPC UA -standardin pohjalta on luotu monia erilaisia rajapintoja, monille eri alustoille. Koska tässä projektissa käytetään .NET -ympäristöä, käytetään siihen soveltuvaa rajapintaa. Kyseinen rajapinta on saatavilla OPC Foundationin GitHub-sivustolta ja siihen soveltuva apuluokka Siemensiltä UAClientHelperAPI.



KUVA 38. OPC UA .NET Stack ja apuluokka UAClientHelperAPI. (Siemens, 2018)

Kuvassa 42 visualisoidaan PLC:n ja tietokoneen välinen kommunikaatioyhteys, joka noudattaa OPC UA -standardia. Siemens on implementoinut SIMATIC S7-1500 laitteistoperheelle OPC UA palvelinjärjestelmän, jota voidaan muokata TIA Portalin avulla. Tietokoneelle rakennetaan työpöytäsovellus hyödyntämällä .NET -ympäristöä ja C#-ohjelmointikieltä, jonka avulla pystytään ottamaan yhteys PLC-laitteeseen ja kommunikoimaan sen kanssa. OPC UA .NET Stack ja UAClientHelperAPI mahdollistavat seuraavat ominaisuudet sovelluksen kehittämiseen:

- Palvelimien etsimisen ja niihin liittyminen.
- Istunnon luomisen ja lopettamisen PLC:n ja tietokoneen välillä.
- Navigointi PLC:n osoitealueessa.
- Tunnisteiden lukemista ja kirjoittamista.
- Tunnisteiden tilaamista, eli niiden tilan muuttumista.
- Menetelmien kutsumista.



KUVA 39. UAClientHelperAPI apuluokan eri metodeja (Siemens, 2018)

Metodi	Selitys
FindServers()	Etsii OPC UA palvelimia verkosta.
Connect()	Muodostaa turvallisen yhteyden palvelimeen.
BrowseRoot()	Palauttaa silmukkakokoelmia juuritasolta.
BrowseNode()	Palauttaa silmukkakokoelman silmukkatasolta.
Subscribe()	Tilaus-metodi, jolla voidaan seurata tiettyä muuttujaa, kun sen arvo muuttuu.
AddMonitoredItem()	Lisätään muuttuja tilauksen seurantaan.
ReadNode()	Luetaan tietyn silmukan metadata.
ReadValues()	Luetaan muuttujan arvot silmukasta.
WriteValues()	Kirjoitetaan uusi arvo muuttujaan.
CallMethod()	Kutsuu metodia palvelimella.

TAULUKKO 6. Projektissa käytettyjä metodeja ja niiden selityksiä. (Siemens, 2018)

5.2.3.2 PostreSQL ja ODBC

.NET -ympäristö ja CLR mahdollistavat monia eri menetelmiä tietokannan yhdistämistä sovellukseen. Tässä projektissa tietokanta rakennetaan ODBC-rajapinnalla ja hallintajärjestelmänä toimii PostreSQL. Luvuissa 3.1.3 ja 3.3.3 käsiteltiin tarkemmin ODBC ja PostreSQL. .NET Framework -ohjelmistokehykseen ei ole implementoitu ODBC-tukea, joten rajapinta vaatii erillisen tiedoston asentamista projektiin. Tiedostot ovat kuitenkin saatavilla suoraan Microsoftin omilta nettisivuilta. Tiedoston mukana tulee kaksi tiedostoa:

- Microsoft.Data.Odbc.dll: dll-tiedosto, joka sisältää kaikki tarvittavat työkalut rakentamaan ODBC-rajapinnan. Tiedosto pitää linkittää sovellukseen, jotta se toimisi oikein.
- Odbcref.chm: Dokumentaatiotiedosto, josta löytyy kaikki selvennykset edellisestä tiedostosta.

Lopuksi tarvitaan vielä erillinen PostgreSQL ODBC -ajuri, jonka voi ladata verkosta. Ajurin asennuksen jälkeen, tietokoneelle pitää lisätä uusi käyttäjän tietolähde. Tämä onnistuu etsimällä tietokoneen hakukoneesta ”Data Source Administrator”, jonka kautta avautuu ikkuna, josta luodaan uusi tietolähde. Luomisen aikana on tärkeätä muistaa luoda uniikki DSN nimi, jotta siihen voitaisiin sovelluksessa viitata. Tämän kaiken jälkeen sovelluskehittäjä voi siirtyä takaisin ohjelmoimiseen, ja seuraavan koodinpätkän avulla voidaan rakentaa sovelluksesta yhteys tietokantaan:

```
// Tietokannan alustusfunktio
public void InitDatabase()
{
    // "Connection String" ja yhteyden rakentaminen.
    // Yhteysmuuttujan avulla päästään käsiksi tietokantaan.
    connectionString = GetConnectionString();
    connection = new OdbcConnection(connectionString);

    // Yhteyden avaaminen/katkaiseminen ja tilan kertominen
    connection.Open();
    Console.WriteLine("Status: " + connection.State.ToString());
    connection.Close();
}

// Yhteysmuuttuja/"Connection String" - Yhteyden muodostamista varten
static private string GetConnectionString()
{
    string OdbcDNS = "PostgreSQL";       // Databasen DNS nimi
    string OdbcUid = "postgres";         // Databasen käyttäjätunnus
    string OdbcPassword = "Salasana";    // Databasen salasana
    return String.Format("DSN={0};UID={1};PWD={2}", OdbcDNS, OdbcUid, OdbcPassword);
}
```

KOODINPÄTKÄ 5. Tietokannan alustamista ja yhteysmuuttujan luonti.

Edellisessä kuvassa nähdään, kuinka tietokannan yhteyden rakentaminen muodostetaan. Olennainen osa yhteyden rakentamista on yhteysmuuttuja, johon liitetään kaikki tarvittavat tiedot oikean tietokannan löytämistä varten. Monilla tietokannoilla ja rajapinnoilla on monia erilaisia menetelmiä luoda tämänkaltaisia yhteysmuuttujia, joihin voidaan lisätä muitakin tietoja tarvittaessa. Seuraavassa kuvassa nähdään, kuinka tietokannasta voidaan hakea tietoa SQL-komentoja hyödyntäen:

```
private void readDatabase()
{
    // Avataan yhteys
    connection.Open();

    // SQL-komento
    string query = "SELECT X FROM Y";

    // Ajetaan komento avatussa yhteydessä
    OdbcCommand command = new OdbcCommand(query, connection);

    // Tallennetaan saatu tieto siihen soveltuvaan muuttujaan
    OdbcDataReader reader = command.ExecuteReader(CommandBehavior.CloseConnection);

    var dataTable = new DataTable();  // Luodaan muuttuja, johon tieto sijoitetaan
    dataTable.Load(reader);           // Ajetaan tieto muuttujaan
    dataGridView.DataSource = dataTable;    // Näytetään tieto käyttöliittymässä

    reader.Close();                   // Suljetaan yhteys ja lukija
}
```

KOODINPÄTKÄ 6. Esimerkki kuinka haetaan tietoa tietokannasta ja sen sijoittamista dataGridView -taulukkoon.

Tietokantaan kirjoittaminen toimii suhteellisen helposti. Aluksi avataan aikaisemmin luotu yhteys, jonka jälkeen kirjoitetaan haluttu SQL-komento. Komento ajetaan ”ExecuteReader”-metodilla, jonka jälkeen saadaan tietokannasta haluttu tulos, tämä voidaan lopuksi sijoittaa esimerkiksi sovelluksessa olevaan ”dataGridView”-taulukkoon. Seuraavaksi nähdään, kuinka tietokantaan voidaan kirjoittaa tietoa:

```
private void SendToDatabase_Click(object sender, EventArgs e)
{
    // Kerätään tietoa tietokantaan lähettämistä varten
    string x = "xValue";
    string y = "yValue";
    string yx = "yxValue";

    // Luodaan SQL-komento, johon aikaisemmat tiedot lisätään
    string insertSQL = String.Format(
        "INSERT INTO rolltab(x, y, yx) VALUES ({0},{1},{2})", x, y, xy);

    // Luodaan ODBC-komentomuuttuja
    OdbcCommand command = new OdbcCommand(insertSQL, connection);
    // Kokeillaan lähettää tietoa tietokantaan
    try
    {
        connection.Open();              // Avataan yhteys
        command.ExecuteNonQuery();      // Ajetaan komento
    }
    // Virhe
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message)   // Näytetään virhe käyttäjälle
    }

    connection.Close();                 // Suljetaan yhteys
}
```

KOODINPÄTKÄ 7. Lähetetään tietoa tietokantaan käyttöliittymän kautta.

5.2.4 Käyttöliittymän lopullinen ulkorakenne

Sovelluksen lopullinen ulkonäkörakenne löytyy liitteestä neljä. Pieniä eroja löytyy mallikuvaan verrattuna, mutta mitään suuria eroavaisuuksia ei syntynyt. Seuraavassa listassa kerrotaan mitä löytyy mistäkin välilehdestä ja mitä työkaluja niissä on käytetty:

- Main Page: Etusivu, mistä löytyy OPC UA:n tilatieto ja PLC:n yhdistämispainike.
- Chart: Grafiikkasivu, mistä löytyy LiveChart-kirjaston tarjoama grafiikkakomponentti.
- Rotation: Perustietoa kulma-anturista.
- Status: Eri PLC-muuttujien tiloja.
- Database: Tietokantasivu, mistä voi hakea tietokannasta tietoa tai kirjoittaa tietokantaan tietoa.

- Circle Diameter: Toinen grafiikkasivu, missä visualisoidaan antureilta saatua tietoa.
- Settings: Asetussivu, mistä pääsee pääkäyttäjän tunnuksilla tutkimaan PLC:n muistia ja tarkemmin silmukkarakenteita.

6 JATKOKEHITYS

Tässä luvussa pohditaan tarkemmin mitä asioista ei käsitelty opinnäytetyössä, ja kuinka niitä voisi hyödyntää sovelluksen jatkokehityksessä. Luvussa käsitellään myös testipenkki, joka suunniteltiin alun perin projektissa käytettävän. Seuraavaksi esitetään lista, jossa tarkennetaan jatkokehityskohteita ja perusteluja niille

- Ohjelmoitava Logiikka:
    - Täyden potentiaalin käyttö – PLC on oleellinen osa projektia, eikä tässä projektissa hyödynnetty sitä niinkään paljon, mitä sitä olisi voinut käyttää. Ohjelma jäi hyvin yksinkertaiseksi, missä seurattiin ja käytettiin ainoastaan muutamaa muuttujaa.
    - Testipenkki – Suunnitelmissa oli rakentaa testipenkki sorviin, mutta kiireiden takia sitä ei saatu toteutettua. Sorvilla olisi pystytty simuloimaan telahiomakonetta, hyödyntämällä servomoottoreita ja muita laitteita. PLC-ohjelma olisi voinut olla rakenteellisesti monipuolisempi, jos servomoottorit olisi saatu projektiin mukaan.
- Käyttöliittymäsovellus:
    - WPF – Seuraava askel miellyttävämpään ulkonäköön. Jatkokehityksen kannalta helppo siirtyä, koska taustakoodi toimii yhtenevästi Windows Formsin kanssa. Haasteita voi tuottaa XAML ja ohjelmointikehykseen tutustuminen.
    - Web-sovellus – Tulevaisuudessa web-sovellukset yleistyvät suurella vauhdilla, mutta sen soveltuvuutta tämän projektin kannalta pitää tarkastella. Web-sovelluksien etuna on sen alustariippumattomuus, mutta ne ovat merkittävästi hitaampia kuin työpöytäsovellukset.
    - Tietokanta – Vaikka PostreSQL on suosittu tietokanta hallintajärjestelmä, markkinoilla on tarjolla monia muitakin vaihtoehtoa. Raportissa tutkitaan tarkemmin muita vaihtoehtoa, ja tulevaisuudessa on mahdollista ottaa käyttöön jokin näistä tietokanta hallintajärjestelmistä.
    - Rasitustesti ja virheiden etsiminen – Sovellukselle ei ole toteutettu yhtäkään rasitustestiä, eli sovellus voi kaatua kriittisellä hetkellä. Myös virheiden etsimistä ei ole toteutettu tarkemmin sovelluksen kannalta.

7 YHTEENVETO

Työ onnistui opinnäytetyöntekijän mielestä suhteellisen onnistuneesti. Aihe oli mukaansatempaava ja piristävä kouluelämään verrattuna. Opinnäytetyöntekijän mielestä työ tarjosi laajan kuvan tyypillisestä teollisuuslaitteekokonaisuudesta, missä käsitellään monia eri insinöörialaan tyypillisiä ominaisuuksia. Aihe tarjosi mahdollisuuden luoda monipuolisen insinöörityön, jossa opinnäytetyöntekijä oppi mm. koneteknisiä termejä ja rakenteita, automaatio-ohjelmointia, sovelluskehittämistä ja tiedon hakemista monista eri lähteistä. Työssä tutustuttiin tarkemmin telahiomakoneeseen, jonka kautta päästiin mittalaitteeseen ja sen ydinolemukseen.

Alustava suunnitelma oli hyvin kunnianhimoinen, mutta työn edetessä työnkuvaa rajattiin hieman. Alustavat tavoitteet olivat: rakentaa testipenkki, johon sijoitettaisiin ohjausjärjestelmä, jonka avulla pystyttäisiin simuloimaan telan hiomista ohjausjärjestelmän avulla. Työtä rajattiin lopuksi, ja lopullinen tavoite oli saada antureilta tieto käyttöliittymäsovellukselle asti ja muuntaa tieto graafiseksi. Suunnittelun ohella opinnäytetyöntekijä teki tutkimusta eri ohjelmointikehyksistä, ohjelmointikielistä, tietokanta hallintajärjestelmistä ja opetteli PLC-ohjelmointia verkkokurssin avulla.

Tutkimusvaihe oli erittäin mielenkiintoinen, koska opinnäytetyöntekijä pääsi tutustumaan tarkemmin eri teknologioihin ja rakentamaan isompaa kokonaiskuvaa tämän avulla. Myös telahiomakoneen ja mittalaitteen rakenteita oli mielekästä tutkia. PLC-ohjelmointi vaati eniten aikaa, koska kurssin aikana käsiteltiin montaa eri ohjelmointikieltä ja opinnäytetyötekijällä ei ollut aikaisempaa kokemusta automaatiosta tai sen järjestelmistä. Haastavinta lopputyössä oli kirjoittaa laadukas raportti itse työstä, mutta ajan myötä kirjoittaminen helpottui. Varsinkin ensimmäiset sivut tuottivat haasteita, koska pelkkä ajatuskin työ laajuudesta aiheutti tuskanhikiä. Alusta lähtien opinnäytetyöntekijä otti tavoitteen kirjoittaa muutama sataa sanaa päivässä, jotta työ etenisi joka päivä hieman.

Jälkiviisaana olisi voinut tehdä paremman tutkimustyön projektin alussa ja tutustua tarkemmin muihinkin osa-alueisiin projektissa. Myös PLC-ohjelmointiin olisi voinut panostaa enemmän, koska verkkokurssin aikana käsiteltiin eri ohjelmointikieliä hyvinkin laajasti, mutta projektissa sitten luotiin hyvin yksinkertainen ohjelma, missä seurataan muutamaa muuttujaa ja ohjataan muutamaa LED-valaisinta. Lopuksi jäi harmittamaan, että sorvia ei päästy hyödyntämään projektissa. Olisi ollut hienoa, jos olisi päästy simuloimaan telahiontaa ohjausjärjestelmällä ja ohjaamaan PLC:llä servomoottoreita.

8 LÄHTEET JA TUOTETUT AINEISTOT

Aaltonen, J. (2019). *TOR® Tasolaakerinauhat*. (Machine Partner) Haettu 27. Joulukuu 2019 osoitteesta Machine Partner: https://www.machinepartner.fi/Tasolaakerinauhat/TOR/

Arjas, A. (1983). Paperin valmistus III osa 2. Teoksessa A. Arjas, *Suomen paperi-insinöörien yhdistyksen oppi- ja käsikirja.* Oy Turun Sanomat/Serioffset. Haettu 13. Helmikuu 2020

Arsenault, C. (20. Huhtikuu 2017). *The pros and cons of 8 popular databases*. Noudettu osoitteesta Keycdn: https://www.keycdn.com/blog/popular-databases

Bacidore, M. (15. Toukokuu 2018). *How many IEC 61131-3 languages do I need?* Haettu 3. Helmikuu 2020 osoitteesta Control Design: https://www.controldesign.com/articles/2018/how-many-iec-61131-3-languages-do-i-need/

Chand, M. (28. Elokuu 2019). *What Is C# and What Is C# Used For*. Haettu 11. Tammikuu 2020 osoitteesta C# Corner: https://www.c-sharpcorner.com/article/what-is-c-sharp/

Chandrasekhar, S. (30. Huhtikuu 2017). *The Big List of Windows 10 Mobile and PC (UWP) Apps from Windows Store*. Haettu 27. Tammikuu 2020 osoitteesta 1red Drop: https://1reddrop.com/2017/04/30/great-windows-10-mobile-pc-uwp-apps-windows-store/

DataFlair. (3. Helmikuu 2020). *C++*. Haettu 12. Helmikuu 2020 osoitteesta Advantages and Disadvantages of C++ | Make your Next Move!: https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/

DB Engines. (Tammikuu 2020). *DB-Engines Ranking*. Haettu 15. Tammikuu 2020 osoitteesta DB Engines: https://db-engines.com/en/ranking

DB-Engines. (2019). *Document Stores*. Haettu 15. Tammikuu 2020 osoitteesta DB-Engines: https://db-engines.com/en/article/Document+Stores

DB-Engines. (2019). *Relational DBMS*. Haettu 15. Tammikuu 2020 osoitteesta DB-Engines: https://db-engines.com/en/article/Relational+DBMS?ref=RDBMS

Dell. (2020). *OptiPlex 3060 Micro*. Haettu 30. Tammikuu 2020 osoitteesta Dell: https://www.dell.com/en-us/work/shop/desktops-n-workstations/3060-micro/spd/optiplex-3060-micro/s019o3060mffus

Digia. (2010). *Qt basics*. (T. Aidantausta, Toimittaja) Haettu 27. Tammikuu 2020 osoitteesta Appro JYU: http://appro.mit.jyu.fi/2010/syksy/gko/luennot/luento21/Qt%20basics%20GKO%202010.pdf

Edelphi. (29. Syyskyy 2018). *Ohjelmistokehykset*. Haettu 23. Tammikuu 2020 osoitteesta Edelphi: https://www.edelphi.fi/ohjelmistokehykset/

Edureka. (22. Toukokuu 2019). *What is MySQL? – An Introduction To Database Management Systems*. (S. Kappagantula, Toimittaja) Haettu 16. Tammikuu 2020 osoitteesta Edureka: https://www.edureka.co/blog/what-is-mysql/#MySQL%20&%20its%20features

Elinkeinoelämän keskusliitto. (2018). *Ulkomaankauppa*. (J. Kangasniemi, Toimittaja;& Elinkeinoelämän keskusliitto) Haettu 25. Joulukuu 2019 osoitteesta Elinkeinoelämän keskusliitto EK: https://ek.fi/mita-teemme/talous/perustietoja-suomen-taloudesta/ulkomaankauppa/

Esri. (2008). *Choosing between Qt and GTK*. Haettu 27. Tammikuu 2020 osoitteesta Esri: http://resources.esri.com/help/9.3/arcgisengine/com_cpp/Cpp/programming/qt_gtk.htm

Farrat. (2019). *Isolated Foundations, Low frequency isolation of machinery*. Haettu 27. Joulukuu 2019 osoitteesta Farrat: https://www.farrat.com/vibration-control/industrial-vibration-control/isolated-foundations

Github. (1. Joulukuu 2018). *UWP vs. WPF*. (jbe2277, Toimittaja) Haettu 27. Tammikuu 2020 osoitteesta Github: https://github.com/jbe2277/waf/wiki/UWP-vs.-WPF

Glade. (12. Maaliskuu 2018). *Glade - A user Interface Designer*. Haettu 24. Tammikuu 2020 osoitteesta Glade Gnome: https://glade.gnome.org/

Gnome. (16. Tammikuu 2014). *gtkmm Frequently Asked Questions*. Haettu 24. Tammikuu 2020 osoitteesta Wiki Gnome: https://wiki.gnome.org/Projects/gtkmm/FAQ

GTK Project. (2019). *Language Bindings*. Haettu 27. Tammikuu 2020 osoitteesta GTK Project: https://www.gtk.org/language-bindings.php

GTK Project. (2019). *What is GTK, and how can I use it?* Noudettu osoitteesta GTK: https://www.gtk.org/

Guru99. (2017). *What is PostgreSQL? Introduction, History, Features, Advantages*. Haettu 21. Tammikuu 2020 osoitteesta Guru9: https://www.guru99.com/introduction-postgresql.html#1

Guru99. (2018). *Oracle Vs. SQL Server: Key Differences*. Haettu 16. Tammikuu 2020 osoitteesta Guru99: https://www.guru99.com/oracle-vs-sql-server.html

Harkushko, L. (2020). *Choosing the Right Database: Factors That Will Help You Make a Wise Decision*. Noudettu osoitteesta Yalantis: https://yalantis.com/blog/

Heidenhain. (2019). *Length Gauges*. Haettu 30. Tammikuu 2020 osoitteesta Heidenhain: https://www.heidenhain.com/en_US/products/length-gauges/

Holma, M. (2016). *TELAHUOLTO- JA -HIONTAPROSESSIN KEHITYS.* Oulun ammattikorkeakoulu, Kone- ja tuotantotekniikan koulutusohjelma. Haettu 13 . Helmikuu 2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/109474/holma_miika.pdf?sequence=1&isAllowed=y

IEC. (2020). *About the IEC*. Haettu 31. Tammikuu 2020 osoitteesta IEC: https://www.iec.ch/about/

Ilves, L. E. (Huhtikuu 2009). Superkalanterin polymeeritelojen vaikutus LWCpaperin laatuun. *Tampereen ammattikorkeakoulu*, 10-14. Suomi. Haettu 6. Tammikuu 2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/9410/Ilves.Leena%20Emilia.pdf?sequence=1&isAllowed=y

Inria. (Syyskuu 2013). *Introduction to Qt*. Haettu 27. Tammikuu 2020 osoitteesta Inria: https://team.inria.fr/imagine/files/2013/09/Introduction-to-Qt.pdf

Koivukunnas, P. (14. Toukokuu 2004). *Suomi Patenttinro FI 113556 B.* Haettu 3. Tammikuu 2020 osoitteesta https://patentimages.storage.googleapis.com/5e/d2/5c/49164228957bc2/FI113556B.pdf

Kähkönen, T. (2003). Insinöörityö. *Paperikoneen telahionnan hiomakivien määrittely erilaisille telapinnoitteille*. Kajaani, Suomi: Kajaanin ammattikorkeakoulu. Haettu 23. Joulukuu 2019 osoitteesta https://www.theseus.fi/bitstream/handle/10024/7830/TEL9STeemuK.pdf?sequence=1&isAllowed=y

Ladder Logic World. (2019). *Ladder Logic Programming Examples*. Haettu 3. Helmikuu 2020 osoitteesta Ladder Logic WORLD: https://ladderlogicworld.com/ladder-logic-programming-examples/

Lahtinen, K.-M. (27. Maaliskuu 2009). *Kalanterin telan vyöhykesäätöventtiilien kunnonvalvonta*. Metropolia Ammattikorkeakoulu. Haettu 3. Tammikuu 2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/1939/kalanteri.pdf?sequence=1&isAllowed=y

Long, R. (30. Syyskuu 2012). *Windows vs. Linux vs. Mac Web Hosting*. Haettu 30. Tammikuu 2020 osoitteesta Omegaweb: https://www.omegaweb.com/windows-v-s-linux-v-s-mac-web-hosting/

Lynn, P. (2019). Siemens TIA Portal STEP 7 WinCC PLC HMI (PLC-S). *Paul Lynn luoma verkkokurssi*. Udemy. Haettu 4. Helmikuu 2020 osoitteesta Udemy: https://www.udemy.com/course/siemens-tia-portal-step-7-wincc-plc-hmi/

Margaret, R.;& Sarah, L. (2020). *Object-oriented programming (OOP)*. Haettu 11. Tammikuu 2020 osoitteesta SearchAppArchitecture: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP

Margaret, R.;Hughes, A.;& Craig, S. (Kesäkuu 2019). *Search SQLServer, Microsoft SQL Server*. Haettu 14. Tammikuu 2020 osoitteesta TechTarget: https://searchsqlserver.techtarget.com/definition/SQL-Server

Martin, D. (6. Kesäkuu 2019). *WPF Vs WinForms - What to choose?* Noudettu osoitteesta RD Global Inc: https://www.rdglobalinc.com/wpf-vs-winforms-what-to-choose/

Materials and Processes in Manufacturing. (2003). Teoksessa E. Degarmo;J. Black;& R. Kohser, *Materials and Processes in Manufacturing* (9 p., s. 388). Wiley. Haettu 2. Tammikuu 2020 osoitteesta https://en.wikipedia.org/wiki/Rolling_(metalworking)#/media/File:Rolling_mill_roll_deflection.svg

Menges Roller Co.: Matthew Menges. (2009). *Crowning Benefits*. Haettu 30. Joulukuu 2019 osoitteesta PFFC-Online: https://www.pffc-online.com/web-handling/132-web-handling-main/6986-paper-crowning-benefits

Menges Roller Company. (1. 1 2014). *Manufacturing News*. Haettu 30. Joulukuu 2019 osoitteesta Pressure-Sensitive Paper Tests Nipping Systems: https://www.mfgnewsweb.com/archives/4/39438/Current-News-jan14/Pressure-Sensitive-Paper-Tests-Nipping-Systems.aspx

Microsoft. (20. Heinäkuu 2015). *Introduction to the C# language and the .NET Framework*. Haettu 11. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Microsoft. (30. Maaliskuu 2017). *ADO.NET Architecture*. Haettu 22. Tammikuu 2020 osoitteesta MIcrosoft Docs: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-architecture

Microsoft. (30. Maaliskuu 2017). *ADO.NET Overview*. Haettu 22. Tammikuu 2020 osoitteesta MIcrosoft Docs: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview

Microsoft. (30. Maaliskuu 2017). *Windows Forms*. Noudettu osoitteesta Microsoft Docs : https://docs.microsoft.com/en-us/dotnet/framework/winforms/

Microsoft. (5. Heinäkuu 2018). *What's a Universal Windows Platform (UWP) app?* Haettu 24. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

Microsoft. (4. Joulukuu 2019). *.NET Core guide*. Haettu 23. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/fi-fi/dotnet/core/ ja https://docs.microsoft.com/fi-fi/dotnet/core/about

Microsoft. (11. Huhtikuu 2019). *Choose your app platform*. Haettu 12. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform

Microsoft. (6. Toukokuu 2019). *Introducing .NET 5*. Haettu 23. Tammikuu 2020 osoitteesta Microsoft Blog: https://devblogs.microsoft.com/dotnet/introducing-net-5/

Microsoft. (2019). *What is .NET Framework* . Noudettu osoitteesta Microsoft Docs: https://docs.microsoft.com/fi-fi/dotnet/framework/get-started/index#Introducing ja https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework

Microsoft. (7. Heinäkuu 2019). *What is Windows Presentation Foundation*. Haettu 12. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/desktop-wpf/overview/

Microsoft. (20. Tammikuu 2020). *.NET Framework versions and dependencies*. Haettu 23. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies

Microsoft Docs. (2017). *What Is ODBC?* Haettu 22. Tammikuu 2020 osoitteesta Microsoft Docs: https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc?view=sql-server-ver15

Minal, J. (12. Helmikuu 2018). *The Future of the C# Programming Language*. Noudettu osoitteesta WebDesignerPad: https://www.webdesignerpad.com/2018/02/the-future-of-the-c-programming-language.html

MongoDB. (2020). *Welcome to the MongoDB Docs*. Haettu 17. Tammikuu 2020 osoitteesta MongoDB Docs: https://docs.mongodb.com/

Moulding, P. (28. Toukokuu 2019). *GTK or Qt?* Haettu 27. Tammikuu 2020 osoitteesta Peter Moulding.com: https://petermoulding.com/gtk_or_qt

OPC Foundation. (28. Syyskuu 2017). *UA Overview*. Haettu 30. Tammikuu 2020 osoitteesta Wiki OPC Foundation: http://wiki.opcfoundation.org/index.php?title=UA_Overview&oldid=853

OPC Foundation. (15. Kesäkuu 2017). *What is OPC?* Haettu 29. Tammikuu 2020 osoitteesta OPC Foundation: https://opcfoundation.org/about/what-is-opc/

OPC Foundation. (26. Syyskuu 2019). *Unified Architecture*. Haettu 29. Tammikuu 2020 osoitteesta OPC Foundation: https://opcfoundation.org/about/opc-technologies/opc-ua/

Oracle. (2019). *Database Licensing Information User Manual*. Haettu 16. Tammikuu 2020 osoitteesta Oracle Docs: https://docs.oracle.com/en/database/oracle/oracle-database/19/dblic/Licensing-Information.html#GUID-B6113390-9586-46D7-9008-DCC9EDA45AB4

Oracle. (2020). *MySQL Editions*. Haettu 20. Helmikuu 2020 osoitteesta MySQL Editions: https://www.mysql.com/products/

Parrish, P. (2016). The SIMATIC S7 System Family. Haettu 26. Helmikuu 2020

Piiroinen, J. (2008). Visual overview of the Common CLR Language Infrastructure, and how the components relate to each other. *.NET Framework ohjelman ajamisen rakenne.* Wikipedia. Haettu 26. Helmikuu 2020 osoitteesta https://commons.wikimedia.org/wiki/File:Overview_of_the_Common_Language_Infrastructure.svg

PLC Academy. (21. Heinäkuu 2015). *Structured Text Tutorial to Expand Your PLC Programming Skills*. Haettu 4. Helmikuu 2020 osoitteesta PLC Programming & Automation Online: https://www.plcacademy.com/structured-text-tutorial/

PLC Academy. (13. Maaliskuu 2018). *Function Block Diagram (FBD) Programming Tutorial*. Haettu 3. Helmikuu 2020 osoitteesta PLC Programming & Automation Online: https://www.plcacademy.com/function-block-diagram-programming/

PLC Programming. (26. Marraskuu 2018). *01 - The TIA Portal - Introduction (English)*. Haettu 31. Tammikuu 2020 osoitteesta Youtube: https://www.youtube.com/watch?v=VfByu6nb-DA

PLCopen. (2018). *Logic*. Haettu 3. Helmikuu 2020 osoitteesta PLCopen: https://plcopen.org/technical-activities/logic

PLCtraining. (2018). *Partnet Conf*. Haettu 3. Helmikuu 2020 osoitteesta dl-plctraining.ir: http://dl.plctraining.ir/upload/Tia%20Portal%20V15/Partner%20conf%20-%20TIA%20Portal%20V15.pdf

Progress. (2020). *What Is an ODBC Driver?* Haettu 22. Tammikuu 2020 osoitteesta Progress: https://www.progress.com/faqs/datadirect-odbc-faqs/what-is-an-odbc-driver

Pulp & Paper Canada. (1997). New roll measurement and control technology result in better machine performance. *Systems can be retrofitter to existing roll grinders*, 72-75. Kanada: Pulp & Paper Canada. Haettu 23. Tammikuu 2020

Qt. (2019). *Supported Platforms*. Haettu 27. Tammikuu 2020 osoitteesta Qt: https://doc.qt.io/qt-5/supported-platforms.html

Qt. (2020). *Extensive Cross-Platform and Language Support*. Haettu 27. Tammikuu 2020 osoitteesta Qt: https://www.qt.io/platforms-languages

Rauhala, P. (4 2010). Telahionnan kehittäminen. Mikkeli: Mikkelin Ammattikorkeakoulu. Haettu 3. 1 2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/13659/Rauhala_Pekka_Telahionnan_kehittaminen.pdf?sequence=1&isAllowed=y

RealPars (Ohjaaja). (2018). *PLC Basics* [Elokuva]. Haettu 28. Tammikuu 2020 osoitteesta https://www.youtube.com/watch?v=PbAGl_mv5XI

RealPars (Ohjaaja). (2018). *What is an HMI?* [Elokuva]. Haettu 28. Tammikuu 2020 osoitteesta https://www.youtube.com/watch?v=kujHQgK352o

RealPars (Ohjaaja). (2019). *What is SCADA?* [Elokuva]. Haettu 28. Tammikuu 2020 osoitteesta https://www.youtube.com/watch?v=nlFM1q9QPJw&t=268s

RealPars. (3. Kesäkuu 2019). *What is SCADA?* Haettu 28. Tammikuu 2020 osoitteesta RealPars: https://realpars.com/scada/

Rexroth Bosch Group. (2009). *Drive & Control profile*. Haettu 3. Helmikuu 2020 osoitteesta Automation: https://www.automation.com/pdf_articles/IEC_Programming_Thayer_L.pdf

RollTest. (2001). *RollCal ClassicWin Manuaali.* Varkaus: RollTest Measurements & Controls. Haettu 14. Helmikuu 2020

RTMC. (2012). *Telahiomakoneen kuva arkistosta.* RollTest Measurements and Controls, Varkaus. Haettu 26. Joulukuu 2019

RTMC. (2014). *RollCal Classic Manuaali.* Varkaus: RollTest Measurements & Controls. Haettu 15. Helmikuu 2020

RTMC. (2015). Logo-kuva. *RTMC-logo*. Varkaus: RollTest Measurements & Controls. Haettu 3. Helmikuu 2020

RTMC. (2018). *RollMike™*. Haettu 15. Helmikuu 2020 osoitteesta RollMike: http://rolltestmc.com/products/rollmike/

RTMC. (2018). *RollSaddle - Roll caliper*. Haettu 15. Helmikuu 2020 osoitteesta RollSaddle: RollSaddle - Roll caliper

RTMC. (2019). *Sisäinen materiaali*. Haettu 12. Helmikuu 2020

RTMC, Väänänen. (1. 12 2019). Viestikeskustelu aiheesta bombeeraus.

Siemens. (28. Heinäkuu 2016). *SIMATIC HMI WinCC V7.4 WinCC/IndustrialDataBridge*. Haettu 29. Tammikuu 2020 osoitteesta Siemens Support Industry: https://support.industry.siemens.com/cs/mdm/109739626?c=91423633419&lc=en-WW

Siemens. (28. Kesäkuu 2018). *Creating of OPC UA clients with .NET and helper class*. Haettu 10. Helmikuu 2020 osoitteesta Support Industry Siemens: https://support.industry.siemens.com/cs/document/109737901/creating-of-opc-ua-clients-with-net-and-helper-class?dti=0&lc=en-US

Siemens. (Helmikuu 2018). *OPC UA .NET Client for the SIMATIC S7-1500 OPC UA Server*. Haettu 10. Helmikuu 2020 osoitteesta Cache Industry Siemens: https://cache.industry.siemens.com/dl/files/901/109737901/att_955026/v2/109737901_OPC_UA_Client_S7-1500_DOKU_V13_en.pdf

Siemens. (2020). *6FX2001-5QE13*. Haettu 3. Helmikuu 2020 osoitteesta Support Industy Siemens: https://support.industry.siemens.com/cs/pd/407093?pdti=pi&dl=en&lc=en-WW

Siemens. (2020). *6GK5005-0BA00-1AB2, SCALANCE XB005*. Haettu 3. Helmikuu 2020 osoitteesta Support Industry Siemens: https://support.industry.siemens.com/cs/pd/343384?pdti=pi&dl=en&lc=en-US

Siemens. (2020). *6SL3040-1MA01-0AA0*. Haettu 30. Tammikuu 2020 osoitteesta Mall Industry Siemens: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6SL3040-1MA01-0AA0

Siemens. (2020). *6SL3055-0AA00-6AB0*. Haettu 3. Helmikuu 2020 osoitteesta Mall Industry Siemens: https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6SL3055-0AA00-6AB0

Siemens. (2020). *S7-1500*. Haettu 30. Tammikuu 2020 osoitteesta Mall Industry Siemens: https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10204162

Siemens. (2020). *SME20/SME25 Sendos Modules External*. Haettu 30. Tammikuu 2020 osoitteesta Mall Industry Siemens: https://mall.industry.siemens.com/mall/en/cn/Catalog/Products/10088768

Siemens AG. (1. Tammikuu 2020). *Siemens 6AV21232GB030AX0*. Haettu 28. Tammikuu 2020 osoitteesta Siemens Industry: https://mall.industry.siemens.com/mall/en/us/Catalog/Product/6AV21232GB030AX0

Stack Overflow. (2008). *What real world WPF applications are out there?* Haettu 27. Tammikuu 2020 osoitteesta Stack Overflow: https://stackoverflow.com/questions/7837/what-real-world-wpf-applications-are-out-there

Stack Overflow. (2019). *Can WPF applications be run in Linux with .Net Core 3?* Haettu 23. Tammikuu 2020 osoitteesta Stack Overflow: https://stackoverflow.com/questions/53954047/can-wpf-applications-be-run-in-linux-with-net-core-3

Timperi, R. (2015). Telahiomon tuotannon läpimenoajan ja laadun optimointi. 20. (P. J. Varis, Toim.) Lappeenranta University of Technology, Department of Mechanical Engineering. Haettu 30. Joulukuu 2019 osoitteesta https://docplayer.fi/37643938-Professori-juha-varis.html

TutorialsPoint. (2017). *MS SQL Server - Editions*. Haettu 16. Tammikuu 2020 osoitteesta TutorailsPoint: https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_editions.htm

TutorialsTeacher. (2017). *What is LINQ?* Haettu 22. Tammikuu 2020 osoitteesta TutorialsTeacher: https://www.tutorialsteacher.com/linq/what-is-linq

W3Schools. (2017). *C++ Introduction*. Haettu 11. Tammikuu 2020 osoitteesta W3Schools: https://www.w3schools.com/cpp/cpp_intro.asp

Valmet. (14. Kesäkuu 2013). *Practical Roll Techniques: Nip & Crown*. Haettu 27. Joulukuu 2019 osoitteesta https://www.valmet.com/globalassets/media/downloads/white-papers/roll-services/wpr_nipandcrown.pdf

Walters, R. (30. Huhtikuu 2018). *Getting Started with Python and MongoDB*. Haettu 15. Tammikuu 2020 osoitteesta Code Project: https://www.codeproject.com/Articles/1241998/Getting-Started-with-Python-and-MongoDB-2

VentureBeat. (4. Marraskuu 2019). *Microsoft launches Visual Studio Online public preview and ML.NET 1.4*. Noudettu osoitteesta VentureBeat: https://venturebeat.com/2019/11/04/microsoft-launches-visual-studio-online-public-preview-and-ml-net-1-4/

Widmaier, T. (2012). *Optimisation of the roll geometry for production conditions.* Helsinki: Aalto University. Haettu 14. Helmikuu 2020 osoitteesta https://aaltodoc.aalto.fi/bitstream/handle/123456789/7290/isbn9789526048796.pdf?sequence=1&isAllowed=y

Vihavainen, A.;& Luukkainen, M. (2017). *Tietokantojen perusteet*. Haettu 15. Tammikuu 2020 osoitteesta University of Helsinki, Department of Computer Science: https://tietokantojen-perusteet.github.io/

9 LIITE 1. UWP, WPF JA WINFORMS -ALUSTOJEN VÄLINEN VERTAILU

LÄHDE: (Microsoft, 2019)

Ominaisuus ja suunnittelutyökalut	UWP	WPF	WinForms
Tuetut versiot	Windows 10	Windows 7 ja uudemmat	Windows 7 ja uudemmat
Ohjelmointikielet	C#, C++/WinRT, C++/CX, VB ja JS	C#, C++/CLI (C++ tarkoitettu laajennus), F# ja VB	C#, C++/CLI (C++ tarkoitettu laajennus), F# ja VB
Käyttöliittymä suoritusaika	Natiivi (C++/WinRT ja C++/CX) ja hallinnoitu (.NET Natiivi)	Hallinnoitu (.NET Framework ja .NET Core 3.1)	Hallinnoitu (.NET Framework ja .NET Core 3.1)
Avoin lähdekoodi	Kyllä (Windows UI-kirjasto ainoastaan)	Kyllä (.NET Core ainoastaan)	Kyllä (.NET Core ainoastaan)
XAML tuki	Kyllä	Kyllä	Ei
Vahvuudet	- XAML käyttöliittymäsuunnittelutyökalu - Rikas ja muokattava UX - Korkea DPI-tuki - Tuki useammalle syöttötyypille, esim. näppäimistölle, hiirelle, kosketusnäytölle ja jne. - Tukee HoloLenseja ja IoT/Surface hubeja - Mahdollisuus kompaktiin käyttöliittymään - Optimoitu akunkesto - Nykyaikainen saavutettavuustuki (kuten näytönlukijat) - Mustetuki - Suojattu suorittaminen sovellussäiliöiden avulla - Rikkaan tekstidatan mahdollisuudet (esim. sisäänrakennettu oikeinkirjoituksen seuranta)	- XAML käyttöliittymäsuunnittelutyökalu - Rikas ja muokattava UX - Laaja kokoelma hallintalaitteita Microsoftilta ja muilta yhteistyökumppaneilta - Tiheä käyttöliittymä - Windows 7 tuki - Alustan tuki tulojen tarkistukseen.	- Nopea sovelluskehitys - WYSIWYG (What You See Is What You Get) -editori mahdollisuus käyttöliittymien rakentamista varten. - Laaja valikoima eri komponentteja Microsoftilta ja muilta sen yhteistyökumppaneilta - Tiheä käyttöliittymä - Windows 7 tuki - Näppäimistön ja hiiren sisääntulot

Tilanteet, joissa tuki on rajoitettu ja ominaisuudet ovat heikkoja	- Usean ikkunan tuki * - Alustan tuki tulojen validoinnille * - Windows 7 ei tueta - Jotkut toiminnot vaativat tietyn Windows 10 minimiversion, jotta ne toimisivat - Puuttuu täysi käyttöjärjestelmä tuki - Suora pääsy kaikkiin levyllä olevien tiedostoihin - ADO.NET - Huono yhteensopivuus kirjastojen kanssa, jotka eivät käytä .NET Standardia tai eivät ole yhteensopivia Windows App Certification Kit:n kanssa - Paikallisen verkon silmukkatuki - Intensiivitiedoston sisään- ja ulostulot	- Korkea DPI-tuki ** - Kosketus sisääntulotyyppi **	- Korkea DPI-tuki ** - Kosketus sisääntulotyyppi ** - Muokattava käyttöliittymä - Rikas graafinen näkymä ja laadukkaat käyttökokemukset (kosketus ja animaatioiden puute)

* Microsoft on ilmoittanut ominaisuudesta, joka käsitellään tässä tilanteessa tulevissa Windows 10 versioissa.

** Vaikka alustasta puuttuu tämä ominaisuus, on tapoja kiertää/korjata tämä ominaisuus toimivaksi.

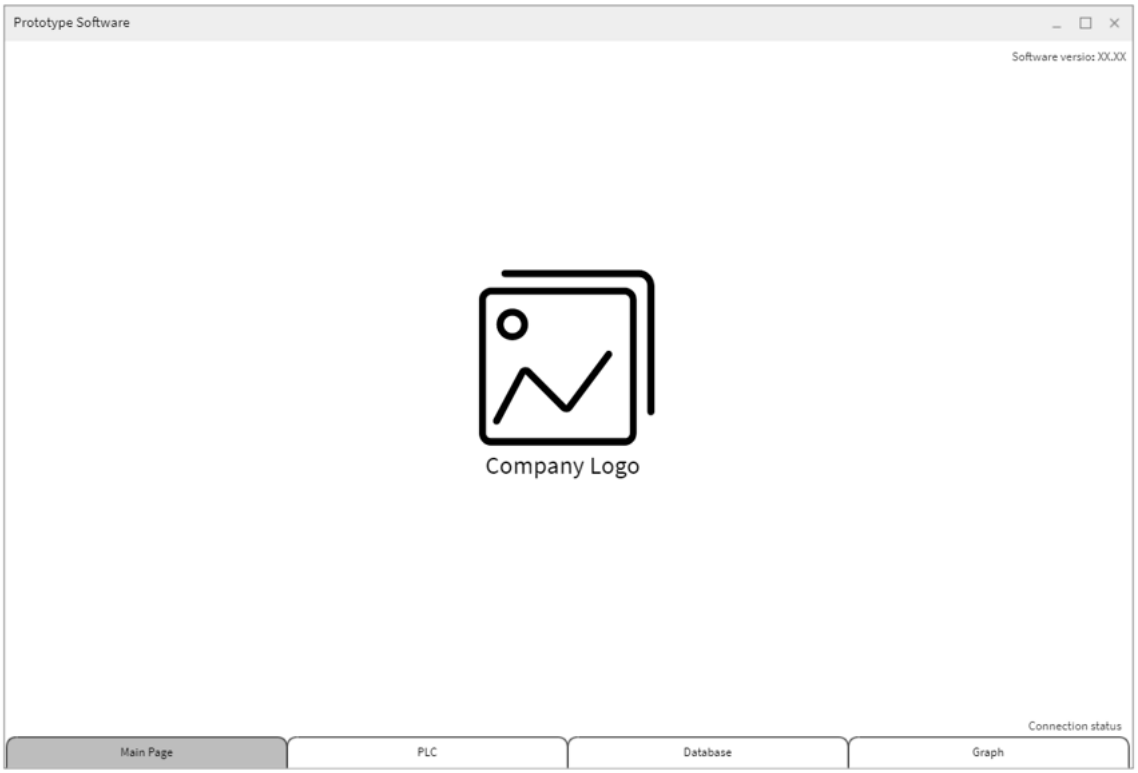
10 LIITE 2. GRAAFISTEN KÄYTTÖLIITTYMÄKIRJASTOJEN VÄLINEN VERTAILU

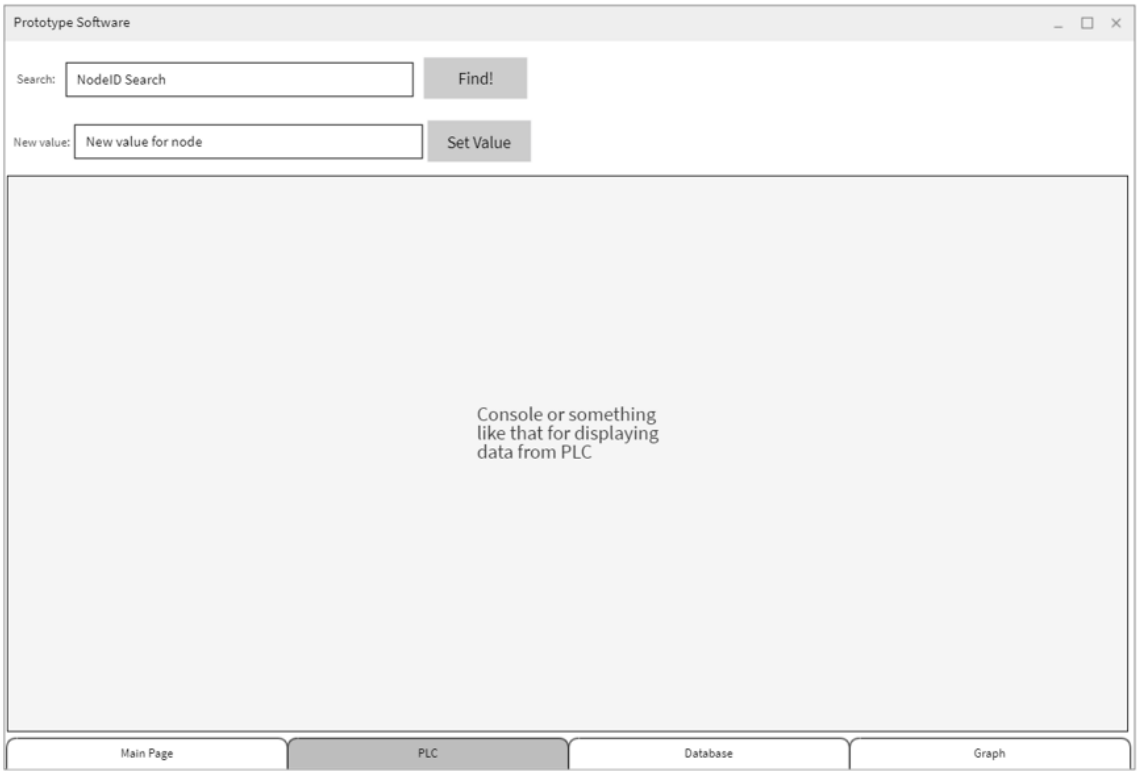
Ominaisuus / GUI	WinForms	WPF	UWP	GTK / gtkmm	Qt
Julkaisu vuosi	2002	2006	2015	1998	1995
Ohjelmointikielet (Pääkielet korostettu)	**C#**, C++/CLI (C++ tarkoitettu laajennus), F# ja **VB**	**C#**, C++/CLI (C++ tarkoitettu laajennus), F# ja **VB**	**C#,** C++/WinRT, C++/CX, **VB** ja JS	C#, **C/C++,** Java, JS, Pascla, Python, Vala ja monia muita.	**C++,** Python, JS ja Qt QML
Käyttöjärjestelmät ja alustat	Microsoft Windows 7 ja uudemmat	Microsoft Windows 7 ja uudemmat	Windows 10	GNU/Linux, Unix, Windows, Mac OS X ja Linux-pohjaisia mobiilikäyttöjärjestelmiä	Linux/Unix, macOS, Windows, Android, iOS, UWP ja WebAssembly
Avoinlähdekoodi	Kyllä (.NET Core ainoastaan)	Kyllä (.NET Core ainoastaan)	Kyllä (Windows UI-kirjasto ainoastaan)	Kyllä	Kyllä, mutta löytyy myös suljettu lähdekoodi
Vahvuudet	- Nopea sovelluskehitys - Laaja valikoima kolmansien osapuolien kirjastoja - Tukee WPF - Kehittynyt alusta ja laaja dokumentaatio - Tehokas ohjelmointiympäristö (Visual Studio) - Mahdollisuus luoda omia komponentteja - .NET API	- Viehättäviä ja monipuolisia käyttöliittymien rakentaminen onnistuu suhteellisen helposti - Tukee XAML ja .NET API - Mahdollisuus luoda omia komponentteja - Tiedon sidonta komponentteihin - Hyödyntää näytönohjainta grafiikan piirtämisessä Tehokkaat 2D ja 3D grafiikat. Hyödynnetään DirectX	- Uusi alusta - Tuki monelle Microsoftin alustalle - Optimoitu kosketuskäyttöön - Tukee XAML - Tehokkaat 2D ja 3D grafiikat. Hyödynnetään DirectX - Korkea DPI-tuki	- Tukee monia alustoja - Moni Linux/Unix käyttöjärjestelmä hyödyntää GTK:ta ja sen työkaluja - Kypsä ja laajasti dokumentoitu - Joustava lisenssi - Suorituskykyisempi kuin .NET alustoilla olevat verrokit.	- Tukee monia alustoja - Kypsä ja laajasti dokumentoitu - Puhtaampi C++-ohjelmointi - Graafinen ohjelmointityökalu löytyy jokaiselta alustalta - Hyvä tuki ja laaja yhteisö - Suorituskykyisempi kuin .NET alustoilla olevat verrokit

Heikkoudet	- Kirjastot voivat olla maksullisia - Toimii ainoastaan Windows alustoilla - Työlästä tehdä hienoja ja laadukkaita käyttöliittymiä - Optimoitu ainoastaan hiirelle ja näppäimistölle, huono tuki kosketusnäytöille - Win32 API:n hyödyntäminen	- Grafiikan suunnittelu voi olla työlästä, koska pitää opetella XAML - Toimii ainoastaan Windows alustoilla - Optimoitu ainoastaan hiirelle ja näppäimistölle, huono tuki kosketusnäytöille - Win32 API:n hyödyntäminen - Hallittu koodi, hitaampi kuin natiivi koodi	- WPF:n tapainen, mutta ei ole yhtä tehokas ja löytyy tiettyjä rajauksia - Sovellus voidaan julkaista ainoastaan Microsoft Storeen ja sen täytyy läpäistä Microsoftin sertifikaatti prosessin. Microsoft ottaa myös pienen osan voitoista. Mutta tämän pystyy kiertämään tarvittaessa. - Ei ole kypsä verrattuna muihin verrattaviin - Ainoastaan Microsoftin omille alustoille soveltuva - Uusi alusta	- Työlästä ohjelmoida - Ei ole keskitettyä tukipalvelua - Työlästä asentaa tietyille alustoille, kuten Windowsille. - Pelkistetty käyttöliittymä	- Työlästä ohjelmoida ja ei ole läheskään niin joustava kuin GTK - C ohjelmointikielen puute, mutta saatavilla erillisillä työkaluilla
Esimerkki sovellukset	LightningChart ja LiveChart	Visual Studio 2010, Nokia Music ja BMW Product Navigator	Audiocloud, Dropbox ja Shazam (Windows Store)	GIMP, Transmission, MonoDevelop	VLC, TeamViewer, VirtualBox

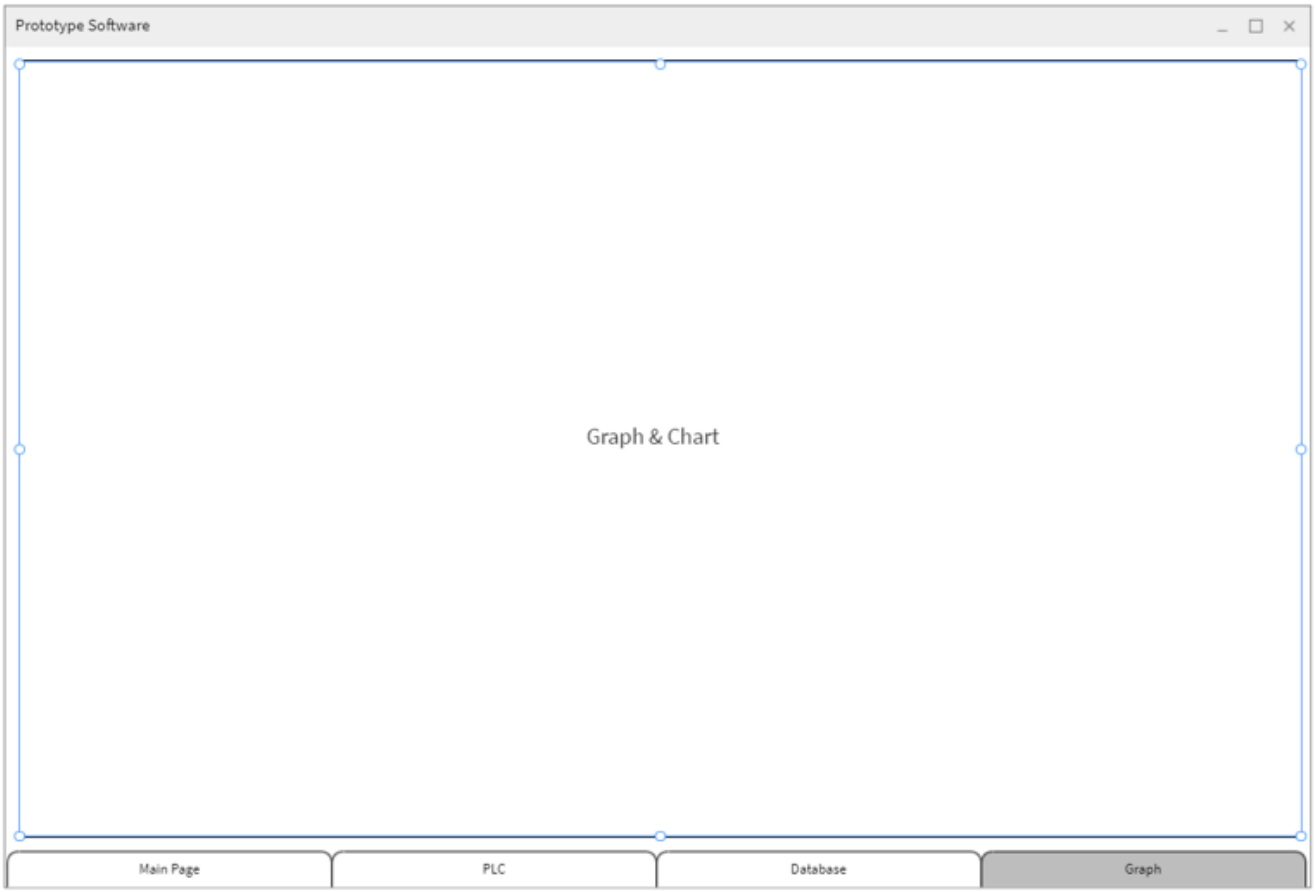
Lähteet: (Qt, 2020; GTK Project, 2019; Qt, 2019; Martin, 2019; Github, 2018; Esri, 2008; Moulding, 2019; Digia, 2010; Stack Overflow, 2008; Chandrasekhar, 2017)

11 LIITE 3. KÄYTTÖLIITTYMÄN ALUSTAVA MALLIKUVA

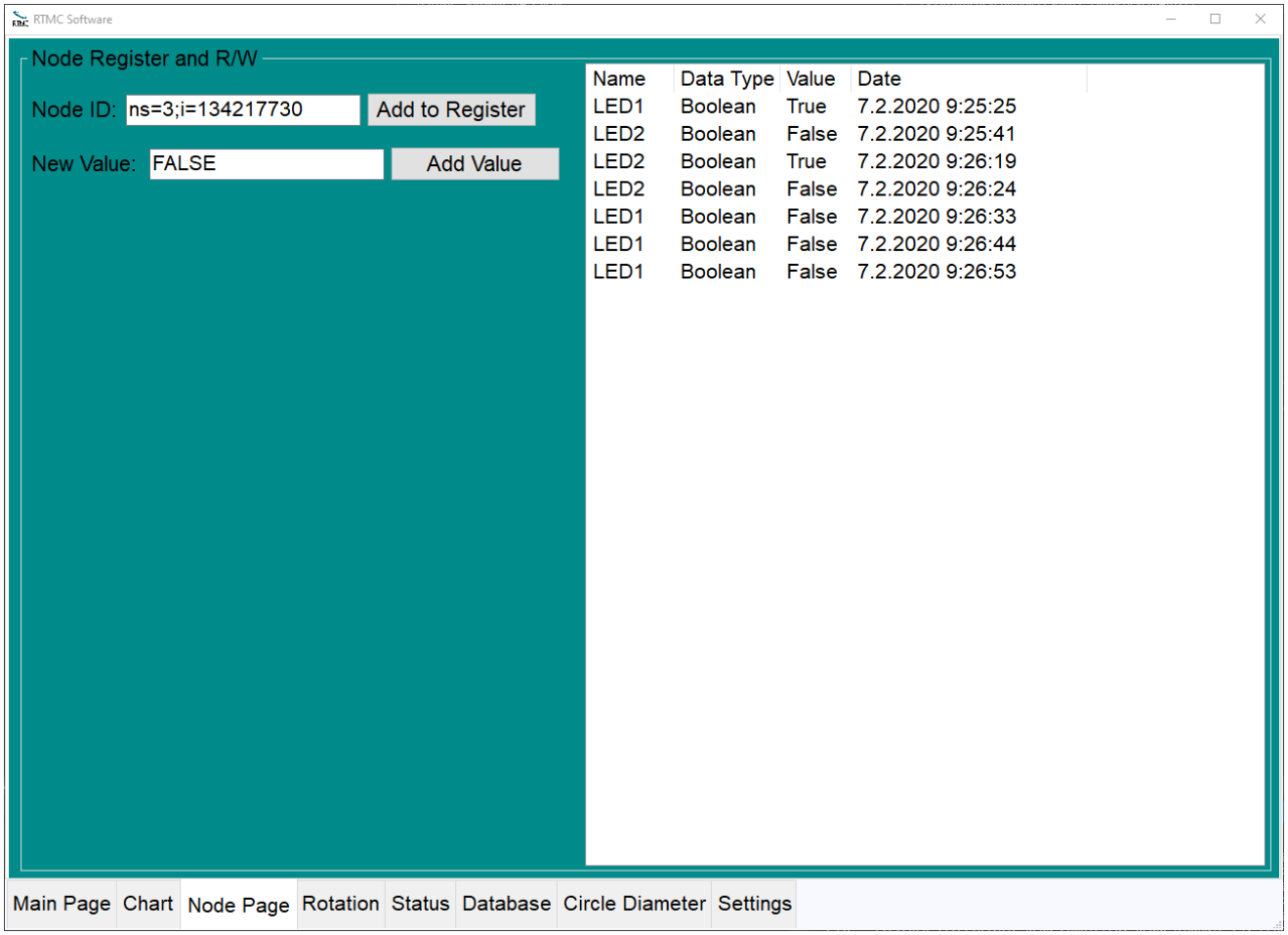





Prototype Software
Value1:
Value 1
Add
Value2:
Value 2
Value3:
Value 3
Show database data here
Main Page
PLC
Database
Graph


Prototype Software
Graph & Chart
Main Page
PLC
Database
Graph

12 LIITE 4. SOVELLUKSEN LOPULLINEN ULKONÄKÖ



RTMC Software

Serial Number: [text]
Type: [text]
Length: [mm]
Diameter: [mm]
Weight: [kg]
Gear Ratio: [int]

Add

Chosen Roll
SN TestiSarjanumero Type TestiTyyppe

serial_number	r_type	r_length	diameter	weight	gearratio	last_measure_time
TestiSarjanumero	TestiTyyppe	123,123	321,321	10000,11	1,5	21.11.2019 9:00
TestiSarjanumero123	TestiTyyppe12	123	321	10000,11	12,5	21.11.2019 9:01
TestiSarjanumero1238	TestiTyyppe12	123	321	10000,4121	12,5	21.11.2019 9:02
{0}	{1}	2	3	4	5	2.4.1902 12:23
sadad	dsdasd	321	32	32	23	19.10.2004 10:23
rrr	rrrr	111	111	111	222	28.11.2019 9:10
asdasd	3	123	32	323	1213	28.11.2019 9:26
1234a	asdsad	1233	232	2	1	28.11.2019 9:27
1234as	asdsad	1233	232	2	1	28.11.2019 9:27
1234asd	asdsad	1233	232	2	1	28.11.2019 9:27
1234asdf	asdsad	1233	232	2	1	28.11.2019 9:27
123ads	asda	11	11	11	1	28.11.2019 9:48
Testitela1	Tyyppi1	12	32	323	3232	28.11.2019 9:54
Testitela12	Tyyppi1	12	32	323	3232	28.11.2019 9:54
Testitela124	Tyyppi1	12	32	323	3232	28.11.2019 9:54

Main Page | Chart | Node Page | Rotation | Status | Database | Circle Diameter | Settings