Olli Frangen

Teollisuusautomaation sovellusohjelmoinnin nykytilanne ja tulevaisuuden näkymät

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Sähkö- ja automaatiotekniikka

Insinöörityö

18.5.2025

Tiivistelmä

Tekijä:	Olli Frangen
Otsikko:	Teollisuusautomaation sovellusohjelmoinnin nykytilanne ja tulevaisuuden näkymät
Sivumäärä:	25 sivua
Aika:	18.5.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Sähkö- ja automaatiotekniikka
Ammatillinen pääaine:	Automaatiotekniikka
Ohjaajat:	Lehtori Reijo Leinonen

Tässä opinnäytetyössä tarkasteltiin teollisuusautomaation sovellusohjelmoinnin nykytilannetta ja tulevaisuuden kehityssuuntia. Työ toteutettiin kirjallisuuskatsauksena, ja se keskittyy erityisesti perinteisesti toteutettuihin prosessinohjauksiin eli IEC 61131-3 -standardin mukaisesti ohjelmoituihin PLC-toteutuksiin.

Opinnäytetyössä käytiin aluksi lyhyesti läpi teollisuusautomaatio sekä teollisuusautomaation sovellusohjelmoinnin ohjelmointikielet. Perinteisesti käytetyt ohjelmointikielet koostuvat viidestä IEC 61131-3 -standardin määrittelemästä ohjelmointikielestä, jotka ovat tikapuukaavio, toimilohkokaavio, ST-kieli, sekvenssikaavio ja käskylista. Opinnäytetyössä todettiin tikapuukaavion, toimilohkokaavion ja ST-kielen olevan standardin määrittelemistä ohjelmointikielistä tällä hetkellä käytetyimpiä.

Seuraavaksi työssä kartoitettiin teollisuusautomaation sovellusohjelmoinnin nykytila, mikä aloitettiin tarkastelemalla perinteisiä lähestymistapoja. Perinteisesti toteutettu prosessinohjaus on IEC 61131-3 -standardin mukainen, enimmäkseen yhden laitevalmistajan instrumenteista koostuva, erittäin toimintavarma sekä luotettava, mutta hyvin jäykkä. Työssä todettiin, että perinteisissä toteutuksissa muutosten teko on työlästä sekä laitevalmistajariippuvuus aiheuttaa vaikeuksia kunnossapidon suhteen.

Perinteisten lähestymistapojen ja niistä juontavien ongelmakohtien kartoituksen jälkeen tarkasteltiin moderneja trendejä, joiden todettiin perustuvan pitkälti nykyisiin ongelmakohtiin. IEC 61499 -standardi pyrki vastaamaan moderneihin vaatimuksiin, mutta opinnäytetyössä todettiin, että IEC 61499 -standardin tarjoamista potentaalisista parannuksista huolimatta sen implementointiaste on vielä matala, mutta tulee luultavasti ajan myötä nousemaan.

Opinnäytetyössä todettiin, että modernit trendit, kuten laitevalmistajariippumattomuus, ohjauksen hajautus, tapahtumapohjaisuus, tekoäly ja langattomuus, tulevat lähitulevaisuudessa todennäköisesti integroitumaan automaatioalalle. Lopuksi arvioitiin tulevaisuuden kehityssuuntia, minkä todettiin olevan pääosin spekulaatiota.

Avainsanat: teollisuusautomaatio, sovellusohjelmointi

Abstract

Author:	Olli Frangen
Title:	Current State and Future Prospects of Industrial Automation Application Programming
Number of Pages:	25 pages
Date:	18 May 2025
Degree:	Bachelor of Engineering
Degree Programme:	Electrical and Automation Engineering
Professional Major:	Automation Engineering
Supervisor:	Reijo Leinonen, Senior Lecturer

The objective of this Bachelor's thesis was to assess the current state and future prospects of application programming in industrial automation from a broad perspective.

The thesis consists of a brief overview of industrial automation in general, the most used programming languages and the current state and future prospects of application programming.

In the summary it is stated that the thesis succesfully covered the current state and near future of application programming in industrial automation, and that evaluating the possible development directions that the future beyond the next few years might bring was purely speculative.

Keywords: Application Programming, Industrial Automation

Sisällys

Lyhenteet

DCS: *Distributed Control System.* Hajautettu hallintajärjestelmä. Laajoissa tuotannoissa käytetty ohjausjärjestelmä, joka kykenee hallinnoimaan useaa pienempää kokonaisuutta samanaikaisesti.

IEC: *International Electrotechnical Commission*. Kansainvälinen sähköalan standardointiorganisaatio.

PLC: *Programmable Logic Controller.* Ohjelmoitava logiikka. Teollinen tietokone, joita käytetään tuotantoprosessien automatisointiin.

1 Johdanto

Tämä opinnäytetyö on tehty Metropolia Ammattikorkeakoululle keväällä 2025. Opinnäytetyön tavoite on tarkastella ja kartoittaa teollisuusautomaation sovellusohjelmoinnin nykytilannetta ja tulevaisuuden näkymiä.

Opinnäytetyö on tarkoitettu erityisesti uusille automaatioalan opiskelijoille, jotka omaavat jo kohtalaisen tuntemuksen teollisuusautomaation sovellusohjelmoinnista. Tavoitteena on tuottaa laaja-alainen ja kattava kuvaus aiheesta menemättä liian syvälle yksityiskohtiin. Lähteinä käytetään automaatioalan kirjallisuutta nykyhetken selvittämiseen sekä yleisesti teknologia-aiheista kirjallisuutta mahdollisten tulevaisuuden kehityssuuntien hahmottamiseen.

Opinnäytetyössä käydään läpi sovellusohjelmoinnin eri toteutusmuotoja ja käyttökohteita nykypäivänä, tarkastellaan lyhyesti, miten nykytilanteeseen on päädytty, mihin tämänhetkinen tilanne vaikuttaisi kehittyvän sekä mitä tulevaisuudessa mahdollisesti saatettaisiin nähdä. Sovellusohjelmointi on laaja aihe, joten tarkastelu ja kartoitus rajataan käsittämään vain yleisimpien toteutustapojen keskeisimmät piirteet.

Opinnäytetyössä on käytetty kieliasun muotoilussa sekä tiedonhaun ideoinnissa OpenAI:n ChatGPT:n versiota GPT-4. Opinnäytetyön tekijä on vastuussa kaikesta opinnäytetyön sisällöstä ja muotoilusta.

2 Teollisuusautomaatio lyhyesti

Tuotantoprosessien automatisointi on nykypäivänä useimmiten valittu toteutustapa, kun tavoitteena on tuotannon määrällinen optimointi, toimintavarmuus sekä henkilökustannusten minimointi. Pienissä ja keskikokoisissa tuotannoissa valitaan useimmiten toteutukseen PLC eli ohjelmoitava logiikka, joka ohjaa tuotannossa toimivia toimilaitteita ohjelmoinnin pohjalta. Ohjelmoitavien logiikoiden valikoima markkinoilla on valtava: Ne voivat olla pieniä niin kooltaan kuin kyvykkyydeltäänkin mahdollistaen vain muutaman toimilaitteen lukemisen ja

ohjaamisen. Kyvykkyyden noustessa myös laitteiston fyysinen koko kasvaa. Useimmiten jo keskikokoisissakin ohjelmoitavilla logiikoilla toteutetuissa tuotannoissa kuitenkin hajautetaan ohjausta useisiin itsenäisiin yksiköihin, joita jokaista ohjaa oma PLC.

Laajemmissa tuotannoissa, kuten paperituotannossa ja öljynjalostuksessa, käytetään ohjelmoitavien logiikoiden sijaan DCS-järjestelmiä. Toisin kuin PLC-järjestelmissä DCS ei varsinaisesti hallinnoi koko prosessia samanaikaisesti. DCS-toteutuksissa kaikki prosessidata kulkee ohjausjärjestelmän läpi ja täten prosessia voidaan tarkastella kokonaisuutena. DCS-järjestelmiin voidaankin liittää lukuisten toimilaitteiden ohella myös kokonaisia PLC-toteutuksia.

Tämä opinnäytetyö koskee lähes puhtaasti perinteisten järjestelmien ohjelmointia eli PLC ja DCS-toteutuksia. Perinteisillä korkeamman tason ohjelmointikielillä toteutetut automaatioprojektit, kuten esimerkiksi kiinteistöautomaatiossa yleiset pilvessä toteutetut tapahtumapohjaiset automaatiototeutukset, jäävät tämän opinnäytetyön tarkastelun ulkopuolelle.

3 Sovellusohjelmoinnin ohjelmointikielet

Sovelluksia voidaan ohjelmoida monella eri tavalla, ja käytetty tapa usein riippuukin käyttökohteesta ja käytössä olevasta laitteistosta. Maailmanlaajuisesti käytössä on lukemattomasti erilaisia yhdistelmiä ohjelmointitapoja ja käytettyjä laitteistoja, mutta ajan myötä tiettyjä trendejä on havaittavissa. Erilaisissa teollisissa prosesseissa on paljon yhtäläisyyksiä, vaikka lopputuotteet eroaisivatkin, ja tiettyjen tavoitteiden saavuttaminen ratkaistaankin useasti samanlaisin metodein.

Alan edelleen käytetyin, vuonna 1992 julkaistu IEC 61131-3 -standardi määrittelee viisi erilaista ohjelmointikieltä, jotka muodostuvat kahdesta tekstimuotoisesta sekä kahdesta graafisesta ohjelmointikielestä sekä *SFC*-kielestä, josta on sekä tekstimuotoinen että graafinen versio [1]. Teollisuusautomaatiossa, varsinkin laajemman kokoluokan tuotannoissa, käytetään pääosin näitä standardissa

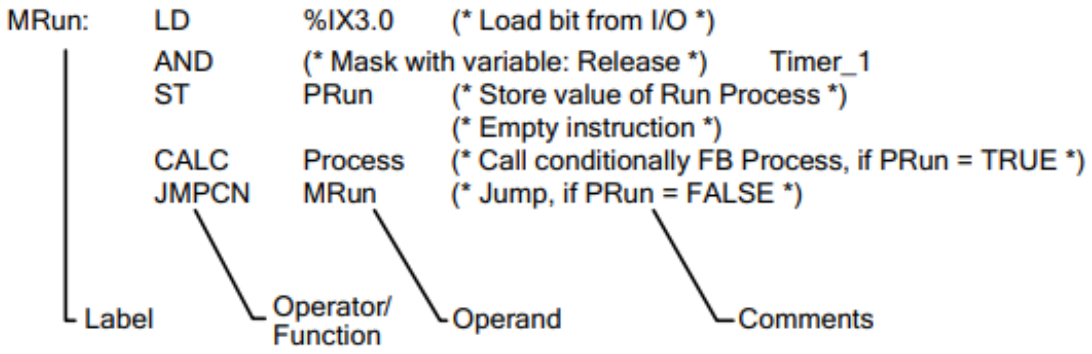
määriteltyjä ohjelmointikieliä. Koska tämän opinnäytetyön tarkastelun kohteena on lähinnä PLC-järjestelmät, tulee tämä osio käsittelemään vain IEC 61131-3 -standardin määrittelemiä ohjelmointikieliä, jotka on esitetty taulukossa 1. Seuraavassa osiossa käydään läpi lyhyesti IEC 61131-3 -standardin määrittelemät ohjelmointikielet ja niiden ominaisuudet sekä rakenteet.

Taulukko 1. IEC 61131-3 -standardin määrittelemät ohjelmointikielet [1].

Tekstimuotoiset ohjelmointikielet	**Graafiset ohjelmointikielet**
Instruction list (IL) – käskylista	*Ladder Diagram* – tikapuukaavio
Structured Text (ST) – rakenteinen teksti, ST-kieli	*Function Block Diagram* – toimilohkokaavio
Sequential Function Chart – sekvenssikaavio, tekstiversio	*Sequential Function Chart* – sekvenssikaavio, graafinen versio

3.1 Käskylista

Käskylistan syntaksi koostuu käskyistä, jotka ovat omilla riveillään. Perusrakenteeltaan se koostuu operaattorista, jota seuraa operaation kohde, operandi. Ajojärjestystä on mahdollista muuttaa asettamalla tunniste käskyrivin alkuun, sekä koodin luettavuutta voidaan parantaa lisäämällä kommentteja riveille. [2, s. 101.] Käskylistan syntaksi on havainnollistettuna kuvassa 1.



Example 4.1. Various IL instructions

Kuva 1. Käskylistan syntaksi [2, s. 101].

Toiminnallisuudeltaan käskylista on kuin yksinkertaistettu korkeamman tason ohjelmointikieli, kuten C. Käskylistaan on sisään rakennettuna perustason matemaattiset operaattorit, kuten summaaminen, kertominen ja jakojäännös. Käyttäjä voi myös määritellä muuttujia, funktioita sekä toimilohkoja ja kutsua niitä myöhemmin koodissa. Toistorakenteita ei käskylistassa kuitenkaan ole, sekä syntaksi on rajoitettu vain yhteen käskyyn riviä kohden. [3.]

Käskylista on *assembly*-tyylinen ohjelmointikieli eli symbolinen konekieli. Tietokoneen suorittimet vastaanottavat kaikki käskyt konekielimuodossa, joka koostuu puhtaasti biteistä eli ykkösistä ja nollista. Tämä muoto on kuitenkin hankalalukuinen ihmiselle, ja se mielellään muutetaankin selkeämpään muotoon. Symboliset konekielet ovat ensimmäinen askel tähän suuntaan. Koska käskylista on rakenteeltaan konekielipohjainen, sitä käytetäänkin usein välitasokielenä, johon ja jonka kautta muita tekstimuotoisia ja graafisia kieliä käännetään [2, s. 100].

3.2 Rakenteinen teksti

Rakenteinen teksti eli ST-kieli muistuttaa niin syntaksiltaan kuin toiminnaltaankin huomattavan paljon perinteisiä ohjelmointikieliä, kuten *C*:tä. Sitä on täten lähtökohtaisesti helpompaa ihmisen kirjoittaa ja lukea. Helppolukuisuus

ihmiselle kuitenkin tarkoittaa, että koodi pitää ajettaessa kääntää koneelle luettavaan muotoon, joka maksaa ylimääräistä laskentakykyä. [2, s. 116.] Kuvassa 2 havainnollistetaan ST-kielen helppolukuisuutta esimerkillä *if*-toistorakenteesta.

```
IF Condition1 THEN
  Statements1;
  (* Execute Statements1, if Condition1 is TRUE, continue after " End_of_IF"; otherwise: *)
  ELSIF Condition2 THEN
    Statements2;
    (* Execute Statements2, if Condition2 is TRUE, continue after " End_of_IF"; otherwise:
    *)
    ELSIF Condition3 THEN
      ...
  ELSE Statements;
  (* Execute Statements if no previous condition evaluates to TRUE *)
END_IF;
(* End_of_IF *)
```

Example 4.20. Example of an IF cascade

Kuva 2. If-toistorakenne ST-kielellä toteutettuna [2, s. 125].

ST-kieli sisältää toiminnallisuudeltaan kaikki samat toiminnot kuin käskylistakin sekä lisäksi toistorakenteet. Sisäänrakennetut laskennalliset operaattorit seuraavat matemaattista laskujärjestystä. ST-kielen helppolukuisuuden ansiosta sillä on käytännöllisempää rakentaa monimutkaisempia ja pidempiä ohjelmakokonaisuuksia kuin käskylistalla. Toisin kuin käskylistassa ST-kielen syntaksissa käskyt voivat jatkua rivien yli.

Monimutkaisemmissa toteutuksissa, varsinkin enemmän matemaattista prosessointia vaativissa tapauksissa, ST-kieli on osoittautunut erityisen tehokkaaksi. ST-kielellä on paljon samankaltaisia ominaisuuksia kuin perinteisillä ohjelmointikielillä, sekä se on ulkomuotonsa ansiosta helpompi omaksua ohjelmistokehitysalalta automaatioalalle siirtyneille. Perinteisiin ohjelmointikieliin verrattavissa olevan rakenteensa ansiosta sillä on helppoa jakaa laajoja toteutuksia pienempiin osiin, mistä on hyötyä niin virheiden välttämisen kuin luettavuudenkin suhteen. [3.]

3.3 Sekvenssikaavio

Sekvenssikaavio eroaa muista ohjelmointikielistä siten, että se hyödyntää toiminnassaan vähintään yhtä muuta standardissa määriteltyä ohjelmointikieltä. Sekvenssikaavio määrittelee ohjelman sekvenssisen toiminnan, eli ohjelma etenee sekvenssistä toiseen tietyn ehdon täytyttyä. Ohjelman sekvenssit, niin kuin sekvenssien väliset ehdotkin, kirjoitetaan jollakin standardissa määritellyllä ohjelmointikielellä. Tämä määritelmä sumentaa sekvenssikaavion määritelmää varsinaisena ohjelmointikielenä. Tämä kyseinen vaatimus kuitenkin tekee sekvenssikaaviosta helposti lähestyttävän kelle tahansa ohjelmoijalle, jolle sovellusohjelmointi on jo tuttua. Sekvenssikaavion luonne tekee sen ohjelmointikieleksi valitsemisesta myös tavallaan helppoa; sekvenssikaavioon ei tarvitse koskea, ellei ohjattava toteutus ole puhtaasti sekventiaalinen. Ohjelman sekventiaalinen ajaminen on myös yleensä ohjelmoitavan logiikan suorittimelle kevyempää, sillä ajettavana on aina kerrallaan vain yksi koko ohjelman osa. Tämä helpottaa myös vianetsintää. [3.]

Sekvenssikaavion voi esittää graafisesti kaikilla ohjelmointikielillä paitsi käskylistalla toteutettuna, ja aina tekstimuodossa. Kuvassa 3 on esimerkki sekvenssikaaviosta ja sen toteutuksesta eri ohjelmointikielillä esitettynä. Tekstimuoto on olemassa lähinnä ohjelmien siirtämiseen eri järjestelmien välillä sekä varmuuskopiointia varten. Sekvenssikaavio-ohjelmassa voi olla useita rinnakkaisia sekvenssejä samanaikaisesti. Ne voivat toimia itsenäisesti tai edetä toiminnassaan samojen ehtojen pohjalta. Sekvenssikaavio-ohjelmaa voi kommentoida samoin kuin muilla ohjelmointikielillä toteutettuja ohjelmia, sekä sekvenssikaavio-ohjelman sisällä olevia sekvenssejä ja ehtoja voidaan myös kommentoida erikseen. [2, s. 169.]

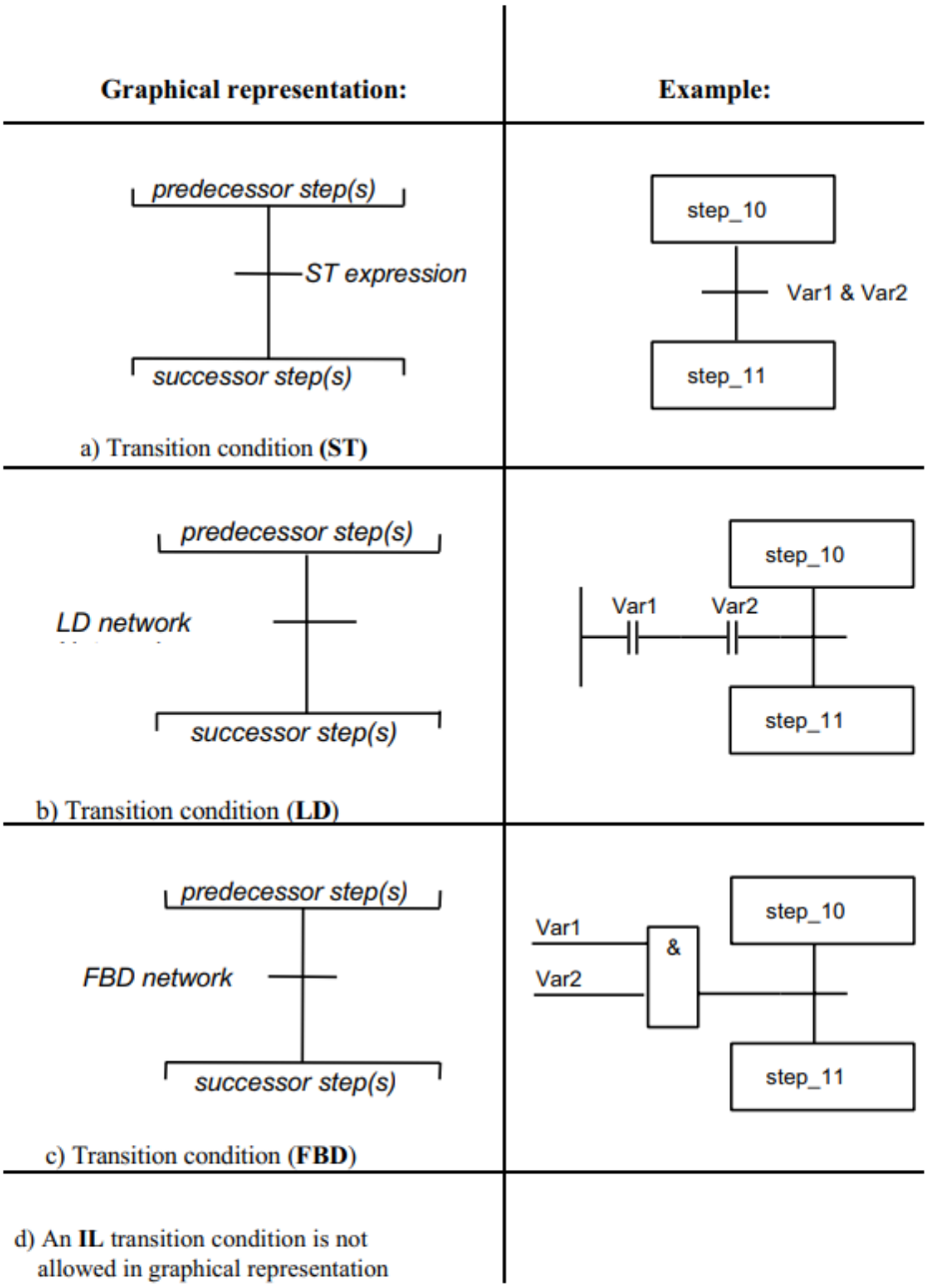


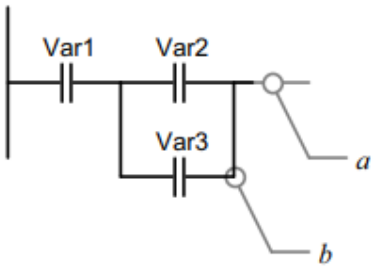
Figure 4.20. Transition (graphical), syntax: ***immediate***. The transition condition is a Boolean expression written in ST, LD or FBD (IL and SFC not allowed).

Kuva 3. Esimerkki sekvenssikaaviosta ja sen graafisesta esityksestä eri ohjelmointikielillä toteutettuna [2, s. 180].

Esimerkkitoteutus sekvenssikaaviosta voisi olla pyykinpesukoneen pesuohjelma, jossa itse pesuohjelman vaiheet ovat ohjelmoitu ST-kielellä. Siirtyminen pesuohjelman vaiheiden välillä voidaan määrittää esimerkiksi ajastimella tai anturimittauksella.

3.4 Tikapuukaavio

Tikapuukaavio-ohjelmointi juontaa juurensa releohjaukseen, joka on ohjelmallisen automaation edeltäjä. Tikapuukaavio-ohjelmointi on tarkoitettu ensisijaisesti hyödyntämään Boolen algebraa. Kuvassa 4 on esitettynä yksinkertainen kolmen kytkimen tikapuukaavio ja sen totuustaulu. Tikapuukaavio on rajoitettu vasemmalta sekä oikealta puolelta virtakiskoilla. Vasen virtakisko on aina aktiivinen, ja positiivinen signaali virtaa kaaviossa vasemmalta oikealle. Kaavioon voidaan asettaa kytkimiä, jotka tyypistään riippuen vaikuttavat signaalin etenemiseen, ja käämejä, jotka tallettavat senhetkisen signaalin arvon muuttujaan. Yleisesti tikapuukaavio jaetaan keskeltä pystysuuntaisesti kahtia, jolloin vasemmalla puolella on signaalin prosessointi ja oikealla puolella signaalin arvon talletus muuttujiin. [2, s. 147; 3.]



Var1	Var2	Var3	value of a	value of b
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	1	0
0	0	1	0	0
1	0	1	1	1
0	1	1	0	0
1	1	1	1	1

Example 4.37. Three open contacts in an AND/OR operation: Connection a has the result of "Var1 AND (Var2 OR Var3)"; b has the value "Var1 AND Var3".

Kuva 4. Yksinkertainen tikapuukaavio ja sen totuustaulu [2, s. 150].

Kytkimiä on neljää eri tyyppiä, normaalisti auki sekä kiinni olevat kytkimet sekä reunakytkimet nousevalle ja laskevalle reunalle. Käämejä on myös eri tyyppejä, ne voivat joko kääntää signaalin arvon, asettaa signaalin reunan mukaan arvon tai asettaa muuttujan arvon kiikkutoimilohkon lailla.

Tikapuukaaviossa voidaan hyödyntää myös toimilohkoja, mutta näin toimiessa tulee olla tarkkana muuttujien arvojen talletuksen suhteen. Jos jossakin kaavion

osassa saman muuttujan arvoa mitataan sekä talletetaan, voi tämä tapahtua järjestelmästä riippuen eri järjestyksessä ja johtaa ohjelman arvaamattomaan toimintaan. Jos ohjelmassa on tarve toimintalohkoille sekä liukuluvuille, on suositeltavaa käyttää ensisijaisesti jotakin muista ohjelmointikielistä. [2, s. 157.]

Tikapuukaavio muodostui nopeasti automaatioalan ohjelmointikielistä suosituimmaksi joustavuutensa sekä tutun ulkomuotonsa ansiosta. Tikapuukaavion esitysmuodon ansiosta sillä toteutetut ohjelmat ovat myös tuotannon operaattoreille helposti luettavissa sekä tarvittaessa jopa muokattavissa. Automaatioalalla pitkään työskennelleille automaatioinsinööreille se on usein tutuin ohjelmointikieli, minkä ansiosta tikapuukaaviolla toteutetut ohjaukset ovat edelleen yleisiä. Se soveltuu parhaiten toteutuksiin, joissa käsitellään lähinnä binäärisiä sekvenssiohjauksia ja yksinkertaisia logiikkasäätöjä, jotka olivat vielä standardin julkaisun aikaan yleisimpiä automaation toteutusmuotoja. [3; 4.]

3.5 Toimilohkokaavio

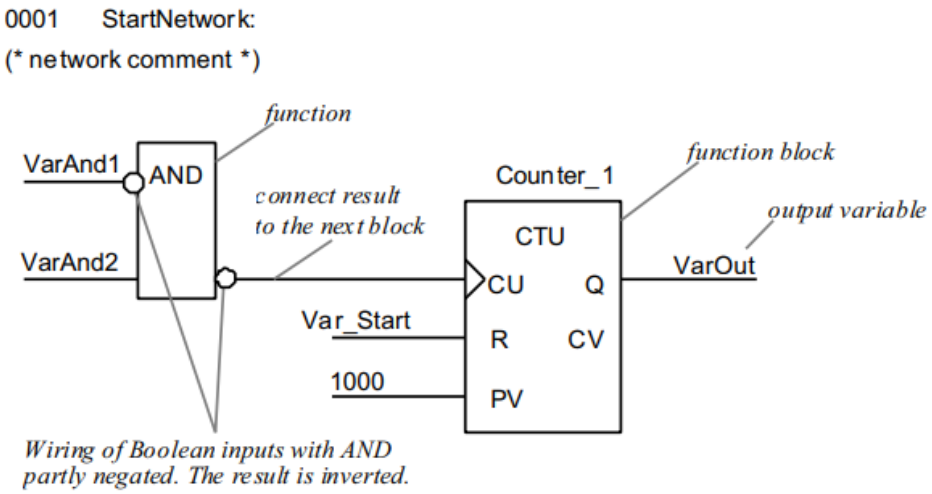
Toimilohkokaavio on standardissa määritellyistä ohjelmointikielistä varsinkin automaatioalan aloittelijoille varmasti tutuin. Graafisen käyttöliittymänsä ansiosta sen lukeminen ja ohjelmoiminen on intuitiivisempaa verrattuna tekstipohjaisiin ohjelmointikieliin, varsinkin tuoreille insinööreille ja automaatioalan opiskelijoille. Kyky käsitellä niin Boolen algebrallisia kuin myös liukulukumuuttujia vaivattomasti samanaikaisesti on monissa toteutuksissa keskeinen kyky. Toimilohkoilla voidaan toteuttaa intuitiivisesti laajojakin ohjelmia, sillä ohjelma voidaan pilkkoa useampiin pienempiin osiin, jolloin ohjelman kulun ymmärtäminen helpottuu. [3.]

Toimilohkokaavio tulee alun perin signaaliprosessoinnin alalta, jossa kokonaislukujen lisäksi liukulukujen käsittely on keskeistä. Toimilohkokaavio mahdollistaa Boolen algebran avulla niin kokonais- kuin liukulukujenkin käsittelyn samanaikaisesti. Tämä kyvykkyys yhdistettynä intuitiiviseen käyttöliittymään on johtanut toimilohkokaavion menestykseen yhtenä keskeisimmistä ohjelmointikielistä teollisen automaation saralla. Signaaliprosessointikykyjensa ansiosta

toimilohkot ovat muodostuneet erityisen keskeisiksi DCS-järjestelmien ohjelmoinnissa. [2, s. 134; 3.]

Niin kuin muissakin ohjelmointikielissä toimilohkokaaviot toimivat yhdistämällä yksinkertaisia operaatioita funktioiden avulla kokonaisuuksiin. Nämä operaatiot toteutetaan toimilohkoilla, joihin syötetään arvoja, ja vastaavasti ulostulona saadaan arvoja. Sisään tulevat arvot voivat olla valmiiksi määriteltyjä vakioita tai muuttujia, ja ulostuloarvot voidaan tallentaa muuttujiin, tai toimilohkoja voidaan myös ketjuttaa, jolloin ketjun seuraavan toimilohkon ulostulo riippuu edeltävän toimilohkon ulostulosta. Kuvassa 5 on esitettynä AND- ja Counter- toimilohkoilla toteutettu ohjelma. Toimilohkoja on yleensä valmistajien tuottamissa ohjelmointiympäristöissä kirjastomallisesti valmiiksi tarjolla, sekä niitä voidaan ohjelmoida vapaasti standardin määrittelemillä kielillä. Käyttäjän määrittelemiä toimilohkoja voidaan käyttää eri kohdissa ohjelmaa, ja ne ajetaan aina samalla lailla. [3.]

Yksi toimilohkokaavio-ohjelmoinnin keskeisistä rajoituksista liittyy toimilohkojen ulostulojen yhdistämiseen. Toimilohkon ulostulon voi kyllä jakaa useamman toimilohkon sisääntuloon, jolloin kaikkiin määränpäihin päätyy sama arvo. Rajoituksena on toimilohkojen ulostulojen yhdistys, eli useamman toimilohkon ulostuloja ei voida ohjata vähäisempään määrään sisääntuloja, jolloin osa ulostuloarvoista yhdistyisivät. Ohjelmallisesti tällöin tulisi priorisoida jonkin tietyn toimilohkon ulostuloarvoa, sekä erityyppiset arvot saattaisivat luoda konflikteja arvojen käsittelyssä. [2, s. 139.]



Example 4.27. Elements of an FBD network

Kuva 5. Yksinkertainen kahden toimilohkon toimilohkokaaviototeutus [2, s. 137].

Kuvan esittämässä ohjelmassa AND-toimilohkossa on sisään- sekä ulostulona Boolen algebralliset muuttujat. Lisäksi toimilohkon ulostulo on negatoitu eli käännetty. Counter-toimilohko laskee syklejä, jolloin AND-toimilohkon ulostulon arvo on 0. Counter-toimilohko niin sanotusti käynnistetään muuttujalla Var_Start, se laskee PV-kohtaan asetettuun arvoon eli 1000:een asti, ja antaa ulostulona muuttujan VarOut.

4 Sovellusohjelmoinnin nykytila

Kuten monilla kaupallisilla aloilla, myös teollisuusautomaation sovellusohjelmoinninkin kehityssuuntaa on määrännyt alusta asti kustannustehokkuus. Tuotannon seisaukset ovat erittäin epätoivottuja, joten varmatoimisuus sekä vikasietoisuus prosessinohjauksessa ovat olleet keskeisimpiä kiintopisteitä teollisuusautomaation ohjelmistokehityksessä alusta asti. Ensimmäiset merkittävät askeleet automaatioalalla otettiin, kun teolliseen sovellusohjelmointiin alettiin tuoda perinteisestä ohjelmistosuunnittelusta tuttuja oppeja. Aluksi nämä käsittivät lähinnä vain parannuksia itse kirjoitetun koodin tehokkuuteen ajettaessa sekä ohjelmiston pienempään muistijalanjälkeen. Ohjelmisto, joka vaatii

vähemmän muistitilaa ja laskentatehoa, mahdollistaa edullisempien ohjelmoitavien logiikoiden käytön. [4.]

Siinä missä perinteisesti keskeiset ominaisuudet, kuten toimintavarmuus, vikasietoisuus ja myöhemmin myös tehokkuus, ovat edelleen keskeisiä, on automaatioalalla kyllä huomattavissa tahtoa siirtyä enemmän ja enemmän modernin ohjelmistokehityksen suuntaan. Konseptit, kuten olio-ohjelmointi, ketterät menetelmät, modulaarisuus, koodin uudelleenkäytettävyys sekä skaalautuvuus, ovat nykyajan ohjelmistokehityksessä saaneet jo tukevan jalansijan, mutteivät niinkään teollisuusautomaation sovellusohjelmoinnissa. Automaatioalaa ohjaa edelleenkin pitkälti IEC 61131-3 -standardi, joka oli julkaisuaikanaankin jo jokseenkin suppea eikä useampien päivitystenkään jälkeen kykene vastaamaan jatkuvasti kehittyviin vaatimuksiin tyydyttävästi. [4; 5.]

Teollisuusautomaatio alana liikkuu hitaasti ja varmasti: menetelmiin, joihin vuosikymmenten saatossa on totuttu, on juututtu, vaikka uudistamalla saavutettaisiin merkittäviäkin hyötyjä. Innovaatiota on kyllä nähtävissä pienempien toimijoiden toimesta, mutta ne ovat usein toimijakohtaisia eivätkä yksinään muodosta sen suurempia trendejä.

4.1 Perinteiset lähestymistavat

Niin kuin on mainittu, automaatioalaa on ohjannut jo pitkään sekä ohjaa edelleen merkittävissä määrin 1990-luvun alussa julkaistu IEC 61131-3 -standardi. Standardin mukaisesti ohjelmoidut PLC-toteutukset ovat toki omalla suunnitellullaan toimialueella monipuolisia sekä kykeneviä, mutteivät rajoituksitta. Standardissa määritelty ohjelmoitavien logiikoiden toimintatapa, syklinen suoritusmalli eli jatkuva sisääntulojen skannaus ja koodin ajaminen, on kyllä ennustettava ja varmatoiminen menetelmä, muttei kevyt taikka ketterä. Syklisessä suoritusmallissa koko ohjelma ajetaan aina kokonaan tietyssä järjestyksessä, vaikkei koko prosessissa tapahtuisi ensimmäistäkään muutosta. Syklin nopeus, eli kuinka usein ohjelma ajetaan, määrätään etukäteen, ja se riippuu niin kyseessä olevan toteutuksen tarpeista kuin saatavilla olevasta laskentatehostakin. Koko

ohjelman ajaminen vaatii merkittävästi laskentatehoa, ja se onkin yksi keskeisimmistä kehityksen kohteista, joita on lähdetty automaatioalan edettyä parantamaan. [5; 6.]

Tuotantojen koon kasvaessa prosessinohjauksen hajautus on ollut keskeinen mielenkiinnon kohde. Perinteisesti se on toteutettu joko DCS-järjestelmällä tai sijoittamalla prosessin eri osa-alueille omat PLC:t. DCS-järjestelmät ovat kuitenkin pieniin ja keskikokoisiin tuotantoihin yliampuvan tehokkaita sekä kalliita. Useammasta itsenäisestä PLC:stä koostuva toteutus on kyllä toimiva muttei kovin elegantti ratkaisu. Tuotannon kokonaisvaltainen samanaikainen tarkastelu ja ohjaaminen on tällöin monimutkaisempaa, sekä eri osa-alueiden PLC:iden välinen kanssakäyminen vaatii lähes poikkeuksetta saman laitevalmistajan laitteiden käytön kautta koko prosessin. [4.]

Perinteisesti toteutetuissa hajautetuissa järjestelmissä myös skaalautuvuus on osoittautunut haasteeksi. Yksittäinen PLC ei kykene käsittelemään määräänsä enempää sisääntuloja, joten instrumenttien määrän kasvaessa on tehtävä kompromisseja. Tuotannon suunnitteluvaiheessa yleensä määritellään odotettu PLC:n käsittelemien instrumenttien määrä, sekä tässä vaiheessa voidaan suunnitella myös kasvuvaraa. Käsiteltävien instrumenttien kasvaessa käsiteltävän datan määrä ja täten myös vaadittu laskentateho kasvaa. Laskentatehon loppuessa kesken voidaan prosessia yrittää optimoida tai syklinopeutta vähentää. Kompromisseja on kuitenkin tehtävä, mikä johtaa vähintään lisäkuluihin. [4; 5.]

4.2 Modernit trendit

Teollisuusautomaation sovellusohjelmointi on murrospisteessä perinteisten PLC-pohjaisten toteutusten ja modernien iäti kehittyvien vaatimusten kohdatessa. Vuonna 2013 IEC 61131-3 -standardiin julkaistu päivitys lisäsi tuen olio-ohjelmoinnille, mikä olikin uudistuksena erittäin tervetullut. IEC 61499 -standardi pyrkiikin uudistamaan teollisuusautomaation vastaamaan paremmin moderneja vaatimuksia, ja siihen syvennytään seuraavassa osiossa. Kuitenkin päivityksineenkin IEC 61131-3 -standardi on rajallinen, mikä on johtanut uusiin yrityksiin

päivittää automaatioala nykypäivän tasalle. Industry 4.0 eli neljäs teollinen vallankumous on keskeisessä roolissa modernien trendien muodostuksessa. Teollisuusautomaatio alana liikkuu hyvin hitaasti ja varovasti, mutta kehityssuunnat ovat selvästi havaittavissa.

Dynaamisuus on yksi viime aikoina esiin noussut trendi, jolle automaatioalalla olisi reilusti kysyntää. Perinteisesti tuotantoon muutoksia tehdessä ohjelmia ei ole pystytty helposti muokkaamaan siten että ne toimisivat suoraan, esimerkiksi uudelleen järjestellyssä tuotantoyksikössä. Kaikille instrumenteille on ennalta määritetyt tunnisteet, ja jos näitä lähdetään muokkaamaan, pitää koko ohjelma käydä läpi alusta asti. Dynaamisille järjestelmille, jotka tunnistaisivat muutokset ja osaisivat reagoida niihin itsenäisesti sekä joissa voitaisiin käyttää uudelleen ohjelman osia vaivattomasti, on ehdottomasti kysyntää. [5; 7.]

Perinteinen prosessinohjauksen syklinen toiminta on myös muodostunut haasteelliseksi tuotantojen kasvaessa ja monimutkaistuessa. Tapahtumapohjaiseen prosessinohjaukseen siirtymiselle olisi selvästi mielenkiintoa, sillä se toisi mukanaan paljon etuja, muun muassa vapauttaen huomattavasti laskentatehoa ja yksinkertaistaen prosessinvalvontaa. Tapahtumapohjaisuuteen siirtyminen mahdollistaisi myös tuotantoverkkojen yhdistämisen tietoverkkoihin paremmin, mihin automaatioalalla on myös selvästi mielenkiintoa. Perinteisesti tuotantojen valvontatoiminnot on toteutettu laitevalmistajan ympäristön sisällä, mikä rajoittaa datan käsittelyä merkittävästi. Perinteisestikin on ollut kyllä mahdollista saada tuotannosta dataa ulos käsiteltäväksi, mutta se on jouduttu tekemään laitteiston sallimien rajojen sisällä. Prosessidatan siirtäminen turvallisesti pilveen ja sieltä eteenpäin käsiteltäväksi ilman erillisiä sovitekerroksia on kyvykkyys, jolle on selvästi kysyntää. [4.]

Myös ohjelmiston siirrettävyys eli modulaarisuus sekä ohjelmien automaattinen muodostaminen ovat nousseet esiin viime vuosina. Perinteisesti ohjelmat on räätälöity aina tiettyyn toteutukseen, eikä ohjelman siirtäminen edes toteutuksen sisäisesti ole ollut mahdollista tai ainakaan suoraviivaista. Perinteisesti ohjelmat on aina kirjoitettu manuaalisesti toki ohjelmointialustasta löytyviä

perusrakennuspalikoita hyödyntäen. Tämä toimintatapa tuo haasteita ohjelmien siirrossa laitteistosta toiseen varsinkin laitteistonvalmistajan vaihtuessa. Eri tapoja luoda koodia automaattisesti, varsinkin eri laitteistonvalmistajien välisesti yhteensopivasti, on esitetty, mutta tähän mennessä esiintyneet menetelmät eivät ole vielä juurtuneet automaatioalalle. [4.]

Langattomuus on lisääntynyt viimeisten vuosikymmenten aikana arkielämässä valtavissa määrin. Täten langattomuuden lisääntyminen myös automaatioalalla on odotettavissa sekä myös jo nykyään havaittavissa. Langattomuuden sovellusta on jo nyt nähtävissä automaatioalalla, mutta vielä pienissä määrin. Langattomuuteen siirtyminen tuo mukanaan lukuisia etuja kuin myös haasteita. Langattomuuden lisääntyminen teollisuusautomaatiossa tulee helpottamaan merkittävästi uusien tuotantojen suunnittelua, käyttöönottoa ja muutostöiden tekoa, sekä se tulee edistämään tuotantojen modulaarisuutta ja dynaamisuutta. [8.]

Haasteista keskeisin on ennen kaikkea langattomien järjestelmien tietoturvallisuus. Tietoturvallisuus ei luonnollisesti ole vain langattomia järjestelmiä koskeva riski, mutta perinteisten toteutusten tietoturvallisuus on jo hyvin tutkittua ja pitkälle kehittynyttä. Perinteisen tehtaan tietoturva alkaa jo etuovelta kulunvalvonnan muodossa, sillä perinteisesti kaikki tehtaan sisäinen tiedonsiirto on kirjaimellisesti vain rakennuksen sisällä. Toki tehtaat kytketään internetiin, mutta avoimen internetin ja tehtaan sisäverkon välissä on lukuisia kerroksia tuoden tietoturvaa. Langattomia tehtaita koskevat samat tietoturvauhat kuin perinteisiäkin tehtaita, sekä langattomuuden myötä tietoturvauhkia on vielä enemmissä määrin. Langattoman tehtaan sisältä väistämättä pääsee langattomia lähetyksiä rakennuksen ulkopuolelle, jolloin kehittyneen salauksen hyödyntäminen on keskeistä. Langattomuus on kuitenkin lisääntymässä jatkuvasti, eli nykyiset uhkat tunnetaan ja niitä vastaan osataan suojautua tehokkaasti. [8.]

4.3 IEC 61499 -standardi

Vuonna 2005 julkaistu IEC 61499 -standardi tuli IEC 61131-3 -standardin rinnalle. IEC 61499 -standardi on rakennettu vanhemman IEC 61131-3

-standardin pohjalle, ja sen keskeisimpinä uusina ominaisuuksina ovat laitteistojen valmistajariippumaton yhteensopivuus, tapahtumapohjainen toiminta, prosessinohjauksen hajautus sekä modulaarisuus. Vanhan standardin pohjalta rakennetuissa tuotannoissa on lähes poikkeuksetta käytettävä saman valmistajan tuotteita prosessin kaikilla osa-alueilla, jotka ovat keskenään vuorovaikutuksessa. Laitteistoa uusiessa aiemmin käytettyjen tuotteiden saatavuus nousee ongelmaksi, jonka ratkaisuna saattaa olla pahimmassa tapauksessa ainoastaan kaikkien vanhojen laitteiden uusiminen. Vaihtoehtoisesti eri laitevalmistajien tuotteita voidaan yrittää saada kommunikoimaan keskenään, mutta tämän toimivuus on täysin tapauskohtaista sekä yleisesti haastavaa ja kallista. [4.]

Tapahtumapohjainen prosessinohjaus mahdollistaa perinteiseen ohjelmoitavien logiikoiden suorittamaan jatkuvaan skannaukseen verrattuna merkittävästi kevyemmän toteutuksen, joka myös reagoi muutoksiin ketterämmin. Tapahtumapohjainen ohjaus kuitenkin toimii vain toteutuksissa, joissa tuotannon instrumenttien jatkuva lukeminen ei ole tarpeellista. Jatkuvaan skannaukseen perustuva prosessinohjaus on täten edelleen luotetuin ja käytetyin menetelmä. [4.; 7.]

IEC 61499 -standardi määrittelee toimilohkot käytännössä kokonaan uudelleen. Samoin kuin IEC 61131-3 -standardissa toimilohkoja voidaan ohjelmoida eri ohjelmointikielillä, mutta IEC 61499 -standardissa toimilohkot ovat modulaarisia ja tapahtumapohjaisia. Lähtökohtaisesti toimilohkot ovat valmiustilassa säästäen laskentatehoa ja ne aktivoituvat vain tarvittaessa. Samasta toimilohkosta voidaan nyt myös saada eri syötteellä eri ulostulo, joten samaa toimilohkoa voidaan käyttää ohjelman eri kohdissa modulaarisesti. Uudet toimilohkot lisäävät täten merkittävästi prosessinohjauksen modulaarisuutta. Esimerkiksi, jos tuotantovolyymiä kasvattaakseen tuotantoon lisätään uusi osa-alue, voidaan tähän uuteen tuotannon osaan vain lisätä jo olemassa olevan tuotannon ohjelma, joka tarjoaa niin kutsutun plug-and-play-kyvykkyyden. [7.]

Kuten jo aiemmin mainittiin, perinteisesti prosessinohjauksen hajautus ohjelmoitavia logiikoita käytettäessä on ollut haastavaa. IEC 61499 -standardi pyrkii mahdollistamaan aidosti hajautetun prosessinohjauksen hajauttamalla

prosessia ohjaavan älyn älykkäille instrumenteille. Nämä älykkäät instrumentit, kuten älykkäät anturit tai robotit, voidaan asettaa prosessin eri osa-alueille, ja ne voivat toimia itsenäisemmin, kun perinteisesti prosessia on ohjannut keskitetysti PLC. Uudet modulaariset ja tapahtumapohjaiset toimilohkot voivat keskustella keskenään ilman PLC:tä välikätenä, jolloin tieto siirtyy vain tarvittaessa ja käytännössä välittömästi, kun seuraavaa skannaussykliä ei tarvitse odottaa. [5.]

Laitevalmistajariippumattomuuteen standardissa on pyritty muun muassa määrittelemällä toimilohkojen rakenteelle sekä niiden väliselle kanssakäymiselle täsmälliset säännöt. Toimilohkojen toiminta on nyt jaettu erikseen tapahtumiin ja dataan, minkä ansiosta eri laitevalmistajien laitteiden on mahdollista kommunikoida keskenään ilman erillistä sovituskerrosta. Toimilohkolle tuleva syöte voidaan käsitellä laitevalmistajasta riippuen toimilohkon sisällä eri tavoin, mutta toimilohkojen välillä kulkeva data on määritelty laitevalmistajasta riippumatta samalla tavalla. Tämän kyvykkyyden ansiosta muun muassa tuotannon vikasietoisuus kasvaa, kun korvattavaa komponenttia etsiessä on tarjolla useamman valmistajan tuotteita. [7.]

Tarjoamistaan merkittävistä potentiaalisista parannuksistaan huolimatta uusi standardi ei kuitenkaan tähän mennessä ole kunnolla juurtunut automaatioalalle ja sen käyttöönottoa on nähtävissä vain satunnaisesti. Standardiin siirtyessä lähes kaikki jo olemassa olevat tuotannon laitteet tulisi uusia. Tästä johtuen uuden standardin seuraaminen tulee huomioida jo suunnitteluvaiheessa, eikä muutoksia voida toteuttaa pikkuhiljaa. Tuotannon tehokkuus sekä vikasietoisuus ovat keskeisiä huomion kohteita, jolloin uuden kokeileminen vanhan, toimivaksi tunnetun käyttämisen sijaan ei vain ole yleistä. Tietotaidon puolesta uuteen standardiin siirtyminen on myös haastavaa. Vaikka IEC 61499 -standardi pohjautuukin vanhaan ja hyvin tunnettuun IEC 61131-3 -standardiin, vaatii sen hyödyntäminen varsin paljon uuden opettelua. [5; 9.]

4.4 Sovellusohjelmoinnin elinkaari

Tässä osiossa käydään lyhyesti läpi teollisuusautomaation sovellusohjelmoinnin elinkaari. Tämä koostuu kaikesta siitä, mitä automaatiototeutuksen ohjelmistopuolella tulee huomioida toteutuksen suunnitteluvaiheesta aina toteutuksen hyllytykseen saakka. Tarkastelussa kiinnitetään myös erityisesti huomiota siihen, millaisia etuja IEC 61499 -standardi mahdollistaa toteutuksen koko elinkaaren aikana.

Kirjassa *Guide to the software engineering body of knowledge* [10] ohjelmistokehityksen keskeisimmät piirteet on määritelty seuraavasti:

- vaatimukset
- ohjelmistosuunnittelu ja -kehitys
- testaus
- ylläpito
- konfiguraationhallinta
- ohjelmistokehityksen johtaminen
- ohjelmistoprosessi
- työkalut ja menetelmät
- ohjelmiston laatu.

Kyseinen teos on laadittu yleisesti ohjelmistokehitystä varten, ja vaikka vaatimukset, menetelmät sekä tavoitteet, ovatkin teollisuusautomaation sovellusohjelmoinnissa perusluonteeltaan erilaisia kuin perinteisessä ohjelmistokehityksessä, on kyseinen suunnitteluraami täysin sovellettavissa. [4.]

Uuden ohjelmiston kehityksessä ensimmäisenä haasteena on vaatimusten määrittely. Toisin kuin perinteisessä ohjelmistokehityksessä vaatimusten määrittely ja itse prosessinohjauksen suunnittelu teollisuusautomaation sovellusohjelmoinnissa usein sulautuvat toisiinsa. Prosessinohjauksessa itse ohjelmiston lisäksi keskeisessä osassa on tuotannosta jo valmiiksi löytyvä laitteisto, joka pitkälti määrää, mitä lisälaitteistoa tullaan tarvitsemaan. Prosessikohtaiset

vaatimukset luovat jokaiselle projektille ainutlaatuisen yhdistelmän laitteisto- ja ohjelmistovaatimuksia. [6.]

Perinteisesti uuden toteutuksen suunnitteluvaiheessa on jo varhain valittu laitevalmistaja, jonka laitteita aiotaan käyttää. Tämä johtuu IEC 61131-3 -standardin jäykkyydestä eri laitevalmistajien välisten laitteiden kanssakäymisessä. Suunnittelu siis aloitetaan valitun laitevalmistajan suunnittelu- ja ohjelmointialustoilla, minkä johdosta valmiit suunnitelmat soveltuvat hyvin pitkälti vain kyseiseen toteutukseen. Tällöin saatetaan myös törmätä tilanteisiin, joissa suunnittelun loppupuolella käykin ilmi, että jonkin toisen laitevalmistajan laitteisto olisi saattanut soveltua johonkin tiettyyn tarkoitukseen paremmin, mutta alkuperäisessä suunnitelmassa on pysyttävä ajallisten sekä taloudellisten tavoitteiden johdosta. IEC 61499 -standardi tuo tähänkin osaan teollisuusautomaatiota modulaarisuutta sekä joustavuutta, sillä uusi standardi mahdollistaa avoimeen lähdekoodiin pohjautuvien suunnittelu- ja ohjelmointialustojen käytön. Tällöin toteutuksen suunnittelu voidaan toteuttaa täysin laitevalmistajasta riippumatta ja laitehankinnat tehdä vasta suunnittelun loppupuolella. [4; 5.]

Kuten jo aiemmin mainittiin, uudella standardilla toteutetuissa tuotannoissa modulaarisuus ja ketteryys on kehittynyt merkittävästi perinteiseen malliin verrattaessa. Perinteisesti kun tuotannosta hajoaa laite ja se tulee korvata, ei joustavuutta ole valtavasti korvaavaa laitetta etsiessä. Toki yhteensopivien laitteiden kirjoa määrää moni muuttuja, kuten millainen laite on kyseessä sekä minkä laitteiden kanssa se kommunikoi. Vaihtoehtojen määrä päätyy kuitenkin useimmiten olemaan melko rajattu, varsinkin uuden standardin mahdollistamaan avoimempaan ympäristöön verratessa, jossa lähestulkoon kaksi mitä tahansa laitetta on mahdollista saada toimimaan yhdessä. [5; 7.]

Perinteisesti lähes kaikki tuotantoon tehdyt muutokset vaativat jossain määrin tuotannon seisauttamista, sekä varsinkaan laitteiden vaihdot eivät perinteisesti ole olleet kovin virtaviivaisia. Tämä pätee niin rikkinäisten laitteiden vaihtamiseen kuin myös tuotannon muunteluun ja päivitykseen. Koko toteutuksen elinkaaren aikana perinteisesti toteutetut tuotannot ovat kerta kaikkiaan jäykkiä;

muutoksia on kyllä mahdollista tehdä, mutta se vaatii usein merkittävästi työtä, vaikka muutos koostuisikin vain yhdestä vaihdetusta laitteesta. Uusi standardi keventää erityisesti muutoksia tehdessä työkuormaa merkittävästi, kun laajoja osia ohjelmasta ei tarvitse enää kirjoittaa uudelleen. [5; 7.]

5 Tulevaisuuden näkymät

Tämän osuuden tarkoituksena on arvioida automaatioalan kehityssuuntaa seuraavien vuosien ja vuosikymmenien aikana. On haastavaa, ellei mahdotonta sanoa varmasti, mihin suuntaan automaatioala kehittyy, mutta selviä merkkejä kehityssuunnista on kyllä havaittavissa. Industry 4.0 eli neljäs teollinen vallankumous määrittää pitkälti, mitä lähitulevaisuudelta voidaan odottaa. Lähitulevaisuuden osalta aihetta on käsitelty jo runsaasti aiemmassa osiossa, joten tämä osio tulee keskittymään kauemmas tulevaisuuteen. Lähitulevaisuutta kauemmas ennustaminen lienee enemmän taidetta kuin tiedettä, mutta ajankohtaiseen tietoon perustuvia arvioita voidaan tehdä. Merkittävät uudet keksinnöt ja kehitykset tulevat kyllä löytämään tiensä teollisuusautomaationkin piiriin, joskin siinä voi automaatioalan jäykkyydestä johtuen kestää hieman pidempään kuin muilla aloilla.

5.1 Tekoäly

Tällä hetkellä teknologia-alan keskeisin puheenaihe on ehdottomasti tekoäly. Tekoäly on levinnyt jo laajasti ihmisten jokapäiväiseen elämään ja on havaittavissa arkielämän lähes jokaisella osa-alueella. Tekoäly on luonnollisesti tällä hetkellä myös automaatioalalla ajankohtainen puheenaihe. Tekoälyä voidaan, ainakin periaatteessa, hyödyntää automaatiototeutuksessa lähes koko elinkaaren aikana aina suunnittelusta kunnossapitoon asti. Tekoälyllä voitaisiin esimerkiksi toteuttaa koko suunnitteluvaihe pelkän syötteen pohjalta; tekoälyyn syötettäisiin, millainen tuotanto halutaan, ja vastaukseksi saataisiin valmis ohjelma ja lista tarvittavista laitteista. Tämä on kuitenkin vielä tässä vaiheessa luonnollisesti vain toivopuhetta ja spekulointia.

Tekoälyn käyttö, niin kuin kaikki automaatioalan uudistukset, tulee alkamaan pienimuotoisesti ja ensisijaisesti pienissä toteutuksissa. Tekoälyn potentiaali automaatioalalla on kuitenkin mittava. Ensimmäinen varsinainen tekoälyn sovellus automaatioalalla on konenäkö. Konenäköä on hyödynnetty automaatioalalla jo pitkään, ja sen käyttö on laaja-alaista. Konenäön lisäksi automaatioalalla on havaittavissa pienissä määrin muitakin tekoälyn sovelluksia, ja ne keskittyvät lähinnä prosessin säädön optimointiin, kunnossapitoon sekä laadunvalvontaan. Maallikoille tutun tekoälyn yleistymiseen automaatioalalla tulee varmasti vielä menemään aikaa. [11.]

Uskon, että muutaman kymmenen vuoden kuluttua tekoäly tulee olemaan erottamaton osa automaatiojärjestelmää. Tekoäly tulee olemaan keskeinen osa prosessinohjausta kattaen joka osa-alueen. Uskon, että tekoäly tulee valvomaan prosessia ja tekemään muutoksia täysin itsenäisesti, jolloin ihmisen tulee puuttua itse prosessinohjaukseen vähemmän ja harvemmin. Tekoäly tulee myös olemaan keskeisessä osassa muutoksia tehdessä: tuotannon osa-alueita voidaan järjestellä uudelleen, muuttaa tai uusia ja tekoäly tulee vastaamaan laitteiden konfiguroinnista ja ohjelman muuttamisesta. Ihmisen rooli tulee olemaan vain tehtyjen muutosten tarkastaminen ja hyväksyminen.

5.2 Kvanttilaskenta

Kvanttilaskenta on ollut pinnalla jo pidempään, mutta hiljattain jäänyt hieman vähemmälle huomiolle tekoälyn noustessa pinnalle. Kvanttilaskenta kehittyy kuitenkin teknologiana jatkuvasti, ja tulevaisuudessa se tulee todennäköisesti leviämään laajasti teollisuuteen. Kvanttilaskennan hyödyt eivät tule esille automaatioalan kontekstissa niinkään yksittäisten laitteiden ohjauksessa, vaan laajemmissa prosessinohjauksissa tuotannoissa, joissa muuttujia on valtava määrä. Kvanttilaskentaa voitaisiin hyödyntää niin säätötekniikassa säätöprosessin optimoimista varten kuin myös tehtaan energiakulutuksen hallinnassa. [12.]

Kvanttilaskennan kyky käsitellä kertaluokkia suurempia määriä dataa kuin nykyinen laskentateho sallii voisi mahdollistaa esimerkiksi adaptiivisen

tuotannonsäätelyn sähkön hinnan, lämpötilan ynnä muiden muuttujien pohjalta. Koko tuotannon yhtäaikainen valvominen ja kunnon seuranta helpottuisi myös. Uskonkin, että tulevaisuudessa kvanttilaskentaa hyödyntävä tekoäly tulee prosessinohjauksessa olemaan lyömätön työkalu.

5.3 Langattomuus

Niin kuin jo aiemmassa osiossa mainittiin, langattomuus on ilmiönä havaittavissa automaatioalalla jo tänä päivänä, ja uskonkin, että tulevaisuudessa langattomuus tulee lisääntymään merkittävästi tuotantoprosesseissa. Langattomuuden hyötyjä ja haittoja käsiteltiin jo myös aiemmassa osiossa, joten tämä osio tulee käsittelemään lähinnä spekulatiivisesti langattomuuden roolia tulevaisuuden teollisuusautomaatiossa. Uskon, että tulevaisuudessa tulee olemaan täysin langattomia tehtaita, joissa kaikki instrumentit ja prosessilaitteet kommunikoivat puhtaasti langattomasti. Tämä varsinkin tekoälyyn sekä kvanttilaskennan mahdollistaman lisälaskentatehoon yhdistettynä voisi johtaa tehtaisiin, joiden tuotanto on täysin muuteltavissa aivan hetkessä. Tilanteessa, jossa tuotettavaa lopputuotetta halutaan vaihtaa, voitaisiin tuotantolaitteita sekä instrumentteja siirrellä vapaasti kenttäväylistä, Ethernet-kaapeleista ja muista johdotuksista huolehtimatta.

6 Yhteenveto

Tässä opinnäytetyössä lähdettiin kartoittamaan modernin teollisuusautomaation sovellusohjelmoinnin nykytilannetta ja pohtimaan tulevaisuuden kehityssuuntia niin seuraavan muutaman vuoden kuin seuraavien vuosikymmentenkin osalta.

Opinnäytetyössä kävi ilmi, että teollisuusautomaation sovellusohjelmoinnin nykytilanne koostuu pitkälti perinteisesti toteutetuista tuotannoista, joissa tahto kehittyä eteenpäin on selkeästi havaittavissa kuin myös tarve pysyä kiinni metodeissa, jotka on jo todettu toimiviksi. Nämä kaksi vastakkaista voimaa ovat kamppailleet automaatioalan alusta asti, ja lopputuloksena on se, että kehitystä kyllä tapahtuu mutta varsin hitaasti. Uudistukset pannaan lähes aina toimeen

pienempien toimijoiden toimesta, ja ajan kuluessa toimiviksi todetut uudistukset etenevät pikkuhiljaa myös valtavirtaan.

Opinnäytetyössä käytiin läpi keskeisimpiä tällä hetkellä havaittavissa olevia kehityssuuntia, joiden todettiin perustuvan pääosin tämänhetkisiin ongelmakohtiin. IEC 61131-3 -standardin mukaiset perinteiset toimintatavat ovat kyllä johtaneet luotettaviin ja toimintavarmoihin tuotantoihin, mutta niiden jäykkyys on noussut ongelmaksi. Teknologinen kehitys on ajan myötä siirtynyt enemmissä määrin langattomuuteen ja ketteriin järjestelmiin, joita voidaan tarpeen mukaan siirrellä ja muokata ilman merkittäviä tuotannon seisauksia.

IEC 61499 -standardi rakennettiin IEC 61131-3 -standardin pohjalle tarkoituksena uudistaa automaatioala vastaamaan moderneja vaatimuksia. IEC 61499 -standardin keskeisimmät kehityskohteet olivat jatkuvasta skannauksesta tapahtumapohjaisuuteen siirtyminen, prosessinohjauksen hajautus sekä laitevalmistajariippumattomuus. Opinnäytetyössä todettiin, että IEC 61499 -standardin tarjoamista potentiaalisista parannuksista huolimatta sen implementointiaste on pääosin vielä melko matala.

Tämän tarkastelun pohjalta osattiin arvioida lähitulevaisuudessa tapahtuvaa kehitystä varsin hyvin. Modernit trendit, kuten tekoäly, langattomuus, ohjauksen hajautus, tapahtumapohjaisuus sekä laitevalmistajariippumattomuuus, saapuvat hitaasti mutta varmasti automaatioalalle. Lähitulevaisuudessa IEC 61499 -standardin implementointiaste luultavasti nousee, sekä standardin ulkopuolella tehtävä kehitystyö pienempien toimijoiden toimesta saattaa johtaa ylemmän tason ohjelmointikieliin perustuvien automaatiototeutusten yleistymiseen.

Lähitulevaisuuden jälkeen tapahtuvaa kehitystä eli vuosikymmenten päässä odottavaa tilannetta arvioitiin viimeisessä osiossa. Tämä osio päätyi olemaan lähes puhdasta spekulaatiota, kuten voitiin odottaakin. Lähitulevaisuuden näkymät ovat melko selkeät nykytilanteen pohjalta, mutta tarkan arvion muodostaminen siitä, mihin maailma kehittyy kymmenessä tai kahdessakymmenessä vuodessa on käytännössä mahdotonta.

Lähteet

1 SFS-EN 61131-3. 2013. Programmable controllers – part 3: Programming languages. SFS Suomen Standardit.

2 John, Karl-Heinz & Tiegelkamp, Michael. 2010. IEC 61131-3: Programming Industrial Automation Systems. Berlin: Springer-Verlag.

3 Ramanathan, Ramakrishnan. 2014. The IEC 61131-3 Programming Languages Features for Industrial Control Systems. 2014 World Automation Congress (WAC). S. 598-603.

4 Vyatkin, Valeriy. 2013. Software Engineering in Industrial Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics Vol. 9, No. 3, S. 1234-1249.

5 Thramboulidis, Kleanthis & Frey, Georg. 2011. Towards a Model-Driven IEC 61131-Based Development Process in Industrial Automation. Verkkoaineisto. Journal of Software Engineering and Applications Vol. 4, No. 4. <http://dx.doi.org/10.4236/jsea.2011.44024>. 4/2011. Luettu 19.4.2025

6 Dubey, Alpana. 2011. Evaluating Software Engineering Methods in the Context of Automation Applications. 2011 9th IEEE International Conference on Industrial Informatics. S. 585-590.

7 Vyatkin, Valeriy. 2011. IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review. IEEE Transactions on Industrial Informatics Vol. 7, No. 4, S. 768-781.

8 Liang, W. & al. 2019. WIA-FA and Its Applications to Digital Factory: A Wireless Network Solution for Factory Automation. Proceedings of the IEE Vol. 107, No. 6, S. 1053-1073.

9 Ratavaara, Johanna. 2024. IEC 61499 -standardi ja sen käyttösovellukset. Insinöörityö. Metropolia-ammattikorkeakoulu. Theseus-tietokanta.

10 Borque, Pierre & Fairley, Richard. 2014. Guide to the software engineering body of knowledge: SWEBOK, Version 3.0. Washington: IEEE Computer Society.

11 Shahade, Aniket & Deshmukh, Priyanka. 2025. A Critical and Scientific Overview of Artificial Intelligence in the Industrial Revolution. 2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT). S. 783-788.

12 Quantum Technology and Application Consortium – QUTAC & al. 2021. Industry quantum computing applications. Verkkoaineisto. EPJ Quantum Technology, Vol. 8, No. 1. <https://doi.org/10.1140/epjqt/s40507-021-00114-x>. 11/2021. Luettu 17.5.2025.